Extração de Conhecimento de Dados

Data Mining

JOÃO GAMA
ANDRÉ PONCE DE LEON CARVALHO
KATTI FACELI
ANA CAROLINA LORENA
MÁRCIA OLIVEIRA

3ª EDIÇÃO
Revista e Aumentada

Visite a Sílabo na rede
www.silabo.pt

Este trabalho é financiado por Fundos FEDER através do Programa Operacional Fatores de Competitividade – COMPETE e por Fundos Nacionais através da FCT – Fundação para a Ciência e a Tecnologia no âmbito do projeto «FCOMP--01-0124-FEDER-022701».

This work is funded by the ERDF – European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT – Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within project «FCOMP-01-0124-FEDER-022701»

Editor: Manuel Robalo

FICHA TÉCNICA:

Título: Extração de Conhecimento de Dados – *Data Mining*
Autores: João Gama, André Ponce de Leon Carvalho, Katti Faceli,
Ana Carolina Lorena, Márcia Oliveira

Capa: Pedro Mota

1ª Edição – Lisboa, outubro de 2012.
3ª Edição – Lisboa, setembro de 2017.
Impressão e acabamentos: Cafilesa – Soluções Gráficas, Lda.
Depósito Legal:
ISBN: 978-972-618-914-5

EDIÇÕES SÍLABO, LDA.

R. Cidade de Manchester, 2
1170-100 Lisboa
Tel.: 218130345
Fax: 218166719
e-mail: silabo@silabo.pt
www.silabo.pt

Índice

Capítulo 1

Introdução

Em computação, muitos problemas são resolvidos por meio da escrita de um algoritmo que especifica, passo a passo, como resolver um problema. No entanto, não é fácil escrever um programa de computador que realize com eficiência algumas tarefas que realizamos com facilidade no nosso dia a dia. Por exemplo, como reconhecer pessoas pelo rosto ou pela fala? Que caraterísticas dos rostos ou da fala devem ser consideradas? Como podemos codificar aspetos como diferentes expressões faciais de uma mesma pessoa, alterações na face (e.g. óculos, bigode, cortes de cabelo), mudanças na voz (e.g. devido a uma gripe) ou estados de espírito? No entanto, os seres humanos conseguem realizar estas tarefas com relativa facilidade. Fazem isso por meio de reconhecimento de padrões , quando aprendem o que deve ser observado num rosto ou na fala para conseguir identificar pessoas, após terem tido vários exemplos de rostos ou falas com identificação clara.

Um bom médico consegue diagnosticar se um paciente está com problemas cardíacos, tendo por base um conjunto de sintomas e de resultados de exames clínicos. Para esse efeito, o médico utiliza o conhecimento adquirido durante a sua formação e a experiência proveniente do exercício da profissão. Como escrever um programa de computador que, dados os sintomas e os resultados de exames clínicos, apresente um diagnóstico que seja tão bom quanto o de um médico experiente?

Também pode ser difícil escrever um programa que efetue a análise de dados das vendas de uma loja. Para descobrir quantas pessoas fizeram mais de uma compra numa loja no ano anterior, podem ser facilmente utilizados os Sistemas de Gestão de Bases de Dados (SGBD). No entanto, como podemos escrever um programa que responda a questões mais complicadas, como por exemplo:

- Identificar conjuntos de produtos que são frequentemente vendidos em conjunto.
- Recomendar novos produtos a clientes que costumam comprar produtos semelhantes.
- Agrupar os consumidores da loja em grupos de forma a melhorar as operações de marketing.

Não obstante a dificuldade em escrever um programa de computador que possa lidar

de forma eficiente com estas tarefas, o número de vezes em que tarefas tão complexas como estas necessitam ser realizadas diariamente é frequente. Além disso, o volume de informação que precisa ser considerado na sua implementação torna difícil, ou mesmo impossível, a sua realização.

As técnicas de Inteligência Artificial (IA), em particular de Extração de Conhecimento de Dados (ECD) ou Aprendizagem Automática, têm sido utilizadas com sucesso num grande número de problemas reais, incluindo os problemas citados anteriormente.

Este capítulo está organizado da seguinte forma. A Seção 1.1 apresenta a relação entre ECD e IA, mostrando alguns exemplos da utilização de ECD em problemas reais. Na Secção 1.2 é introduzida a relação entre um conjunto de dados e a qualidade da hipótese induzida por um algoritmo de ECD. O conceito de viés indutivo, essencial para que a aprendizagem ocorra, é discutido na Secção 1.3. A Secção 1.4 descreve as diferentes tarefas de aprendizagem, que são agrupadas em aprendizagem preditiva e aprendizagem descritiva. Por fim, a Secção 1.5 apresenta a estrutura dos capítulos do livro.

1.1 Inteligência Artificial e Aprendizagem Automática

Até há alguns anos atrás, a área de IA era vista como uma área teórica, com aplicações apenas em pequenos problemas curiosos, desafiantes, mas de pouco valor prático. Grande parte dos problemas práticos que necessitavam de computação eram resolvidos pela codificação, nalguma linguagem de programação, dos passos necessários à respetiva resolução. A partir da década de 1970, verificou-se uma maior disseminação do uso de técnicas de computação baseadas em IA para a resolução de problemas reais. Muitas vezes, estes problemas eram computacionalmente tratados por meio da aquisição de conhecimento de especialistas do domínio (e.g. especialistas da área da medicina), que era então codificado por regras lógicas, num programa de computador. Estes programas eram conhecidos como Sistemas Especialistas ou Sistemas Baseados em Conhecimento. O processo de aquisição do conhecimento envolvia entrevistas com os especialistas para descobrir as regras utilizadas na tomada de decisão. Esse processo possuía várias limitações, tais como: subjetividade, decorrente do fato dos especialistas sustentarem, com frequência, a tomada de decisão na sua intuição; e dificuldade em verbalizar e exteriorizar esse conhecimento.

Nas últimas décadas, a crescente complexidade dos problemas a serem tratados computacionalmente e o volume de dados gerados em diferentes setores, reforçou a necessidade de desenvolvimento de ferramentas computacionais mais sofisticadas e autónomas, que reduzissem a necessidade de intervenção humana e a dependência de especialistas. Para alcançar estes objetivos, as técnicas desenvolvidas devem ser capazes de criar, de forma autónoma e a partir da experiência passada, uma hipótese, ou função, capaz de resolver o problema que se deseja tratar. Um exemplo simples é a descoberta de uma hipótese, na forma de uma regra ou conjunto de regras, para definir quais os clientes de um supermercado que devem receber publicidade de um novo produto, utilizando os dados de compras anteriores dos clientes registados na base de dados do supermercado. A este processo de indução de uma hipótese (ou aproximação de função) a partir da experiên-

cia passada, dá-se o nome de *Aprendizagem Automática* ou *Extração de Conhecimento de Dados* (ECD).

A capacidade de aprendizagem é considerada essencial para um comportamento inteligente. Atividades como memorizar, observar e explorar situações para aprender fatos, melhorar habilidades motoras/cognitivas através da prática, organizar conhecimento novo e utilizar representações apropriadas, podem ser consideradas atividades relacionadas com aprendizagem. Existem várias definições de ECD na literatura. Uma delas, apresentada em Mitchell (1997), define ECD como:

> *'A capacidade de melhorar o desempenho na realização de alguma tarefa por meio da experiência.'*

Em ECD, os computadores são programados para aprender com a experiência passada. Para tal, empregam um princípio de inferência denominado indução, no qual se obtêm conclusões genéricas a partir de um conjunto particular de exemplos. Assim, os algoritmos de ECD aprendem a induzir uma função, ou hipótese, capaz de resolver um problema, a partir de dados que representam instâncias do problema a ser resolvido. Estes dados formam um conjunto, simplesmente denominado conjunto de dados (Seção 1.2). Embora ECD esteja naturalmente associada à IA, outras áreas de investigação são importantes e têm contribuições diretas e significativas no avanço da ECD, tais como, Probabilidade e Estatística, Teoria da Computação, Neurociência, Teoria da Informação, entre outras. ECD é uma das áreas de investigação da computação que mais tem crescido nos últimos anos. Diferentes algoritmos de ECD, diferentes formas de utilizar os algoritmos existentes e adaptações de algoritmos são continuamente propostos. Além disso, em cada instante surgem novas variações nas caraterísticas dos problemas reais a serem tratados.

Existem várias aplicações bem-sucedidas de técnicas de ECD na resolução de problemas reais, entre as quais podem ser citadas:

- Reconhecimento de voz - palavras faladas;
- Predição de taxas de cura de pacientes com diferentes doenças;
- Deteção do uso fraudulento de cartões de crédito;
- Condução autónoma de automóveis;
- Ferramentas que jogam gamão e xadrez ao nível de campeões;
- Diagnóstico de cancro por meio da análise de dados de expressão genética.

Além do enorme volume de aplicações que beneficiam das caraterísticas da área de ECD, outros fatores têm favorecido a expansão desta área, nomeadamente, o desenvolvimento de algoritmos cada vez mais eficazes e eficientes, e a elevada capacidade dos recursos computacionais atualmente disponíveis. Outras motivações para a investigação em ECD incluem a possibilidade de aumentar a compreensão de como se processa a aprendizagem nos seres vivos. Além disso, algumas tarefas são naturalmente melhor definidas por meio de exemplos. Os modelos gerados são, ainda, capazes de lidar com situações não apresentadas durante o seu desenvolvimento, sem necessariamente necessitar de uma nova fase de projeto.

1.2 Indução de Hipóteses

Para caracterizar um conjunto de dados, vamos considerar a informação sobre os doentes de um hospital. Neste conjunto, cada observação corresponde a um doente. Cada observação, também denominada objeto, exemplo ou registo, é uma tupla formada pelos valores das caraterísticas que descrevem os principais aspetos desse doente. Essas características são designadas por atributos ou variáveis independentes). A título de exemplo, os atributos de um doente podem ser: a sua identificação, o nome, a idade, o género, o estado civil, os sintomas e resultados de exames clínicos. Exemplos de sintomas podem ser presença e distribuição de manchas na pele, o peso e a temperatura do corpo.

Conforme será visto mais adiante, em algumas tarefas de extração de conhecimento, um dos atributos é considerado o atributo de saída (também denominado atributo alvo ou variável dependente), cujos valores podem ser estimados utilizando os valores dos demais atributos, denominados atributos de entrada, ou atributos previsores. O objetivo de um algoritmo de ECD utilizado nestas tarefas é aprender, a partir de um subconjunto dos dados, denominado conjunto de treino, um modelo ou hipótese capaz de relacionar os valores dos atributos de entrada de um objeto com o valor do seu atributo de saída.

Um requisito importante para os algoritmos de ECD é a capacidade de lidar com dados imperfeitos. Muitos conjuntos de dados apresentam algum tipo de problema, como presença de ruído, dados inconsistentes, dados em falta e dados redundantes. Idealmente, os algoritmos de ECD devem ser robustos a estes problemas, minimizando a sua influência no processo de indução de hipóteses. Porém, dependendo da sua extensão, estes problemas podem prejudicar o processo indutivo. Por esse motivo, técnicas de pré-processamento são frequentemente utilizadas na identificação e minimização da ocorrência desses problemas.

Retomando o exemplo dos pacientes, considere a situação em que um algoritmo de ECD é utilizado para aprender uma hipótese (por exemplo, uma regra) capaz de diagnosticar pacientes de acordo com os valores associados aos seus atributos de entrada. Os atributos referentes à identificação e nome do paciente não são considerados entradas relevantes, uma vez que não possuem qualquer tipo de relação com o diagnóstico de uma doença. Na verdade, o que se pretende, é induzir uma hipótese capaz de realizar diagnósticos corretos para novos pacientes, i.e. pacientes diferentes daqueles que foram utilizados na aprendizagem da regra de decisão. Assim, uma vez induzida uma hipótese, é desejável que esta também seja válida para outros objetos do mesmo domínio, ou problema, que não fazem parte do conjunto de treino. A esta propriedade de uma hipótese continuar a ser válida para novos objetos dá-se o nome de *capacidade de generalização da hipótese*. Para que uma hipótese se revista de utilidade quando aplicada a novos dados, é fundamental que apresente uma boa capacidade de generalização.

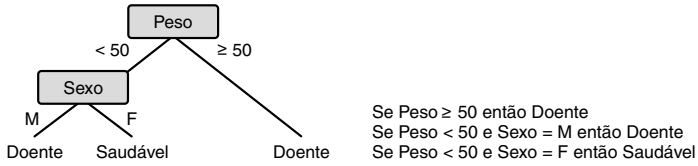
Quando uma hipótese apresenta uma capacidade de generalização reduzida, pode ser porque esta se encontra superajustada aos dados (*overfitting*). Neste caso, diz-se que a hipótese memorizou, ou especializou-se nos dados de treino. No caso inverso, o algoritmo de ECD pode induzir hipóteses que apresentem uma baixa taxa de acerto mesmo no conjunto de treino, gerando uma condição de subajustamento (*underfitting*). Esta situação

pode ocorrer, por exemplo, quando os exemplos de treino disponíveis são pouco representativos, ou o modelo usado é muito simples e, por conseguinte, incapaz de capturar os padrões existentes nos dados (Monard e Baranauskas, 2003). Estes conceitos são ilustrados e novamente discutidos na Secção 7.2.1. São feitas então considerações e motivações sobre a escolha de modelos com boa capacidade de generalização.

1.3 Viés Indutivo

Quando um algoritmo de ECD aprende a partir de um conjunto de dados de treino, procura uma hipótese, no espaço de hipóteses possíveis, que seja capaz de descrever as relações entre os objetos, e que melhor se ajuste aos dados de treino.

Cada algoritmo utiliza uma forma, ou representação, para descrever a hipótese induzida. Por exemplo, as redes neuronais artificiais representam uma hipótese por um conjunto de valores reais, associados aos pesos das conexões da rede. As árvores de decisão utilizam uma estrutura de árvore em que cada nó interno é representado por uma pergunta referente ao valor de um atributo e cada nó externo está associado a uma classe. A representação utilizada define a preferência ou viés (*bias*) de representação do algoritmo e pode restringir o conjunto de hipóteses que podem ser induzidas pelo algoritmo. A Figura 1.1 ilustra o viés de representação utilizado por técnicas de indução de árvores de decisão, redes neuronais artificiais e regras de decisão.



(a) Árvore de decisão (b) Conjunto de regras

$$\begin{pmatrix} 0{,}45 & -0{,}40 & 0{,}54 & 0{,}12 & 0{,}98 & 0{,}37 \\ -0{,}45 & 0{,}11 & 0{,}91 & 0{,}34 & -0{,}20 & 0{,}83 \\ -0{,}29 & 0{,}32 & -0{,}25 & -0{,}51 & 0{,}41 & 0{,}70 \end{pmatrix}$$

(c) Redes neuronais

Figura 1.1 *Diferentes vieses de representação.*

Além do viés de representação, os algoritmos de ECD possuem também um viés de procura. O viés de procura de um algoritmo é a forma como o algoritmo procura a hipótese que melhor se ajusta aos dados de treino. Este viés define como as hipóteses são procuradas no espaço de hipóteses. Por exemplo, o algoritmo ID3, que é utilizado na indução de árvores de decisão, tem como viés de procura a preferência por árvores de decisão com poucos nós, conforme será apresentado no Capítulo 6.

Assim, cada algoritmo de ECD possui dois vieses: um *viés de representação* e um *viés de procura*. O viés é necessário para restringir as hipóteses a serem visitadas no

espaço de procura. Sem viés não haveria aprendizagem/generalização. Os modelos seriam especializados para os exemplos individuais. Embora, à primeira vista, o viés pareça ser uma limitação dos algoritmos de ECD, segundo Mitchell (1997), sem viés um algoritmo de ECD não consegue generalizar o conhecimento adquirido durante o treino para aplicá-lo com sucesso a novos dados.

1.4 Tarefas de aprendizagem

Os algoritmos de ECD têm sido amplamente utilizados em diversas tarefas, que podem ser organizadas de acordo com diferentes critérios. Um deles diz respeito ao paradigma de aprendizagem a ser adotado para lidar com a tarefa. De acordo com esse critério, as tarefas de aprendizagem podem ser divididas em: **Preditivas** e **Descritivas**.

Em tarefas de previsão, o objetivo consiste em encontrar uma função (também denominada de modelo, ou hipótese), a partir dos dados de treino, que possa ser utilizada para prever um rótulo, ou valor, que caraterize um novo exemplo, com base nos valores dos seus atributos de entrada. Para esse efeito, é necessário que cada objeto do conjunto de treino possua atributos de entrada e de saída.

Os algoritmos, ou métodos, de ECD utilizados nesta tarefa induzem modelos preditivos. Estes algoritmos seguem o paradigma da aprendizagem supervisionada. O termo *supervisionada* vem da simulação da presença de um *supervisor externo*, que conhece a saída (rótulo) associada a cada exemplo (conjunto de valores para os atributos de entrada). Com base neste conhecimento, o supervisor externo pode avaliar a capacidade da hipótese induzida em predizer o valor de saída para novos exemplos.

Em tarefas de descrição, o objetivo consiste em explorar, ou descrever, um conjunto de dados. Os algoritmos de ECD utilizados nestas tarefas ignoram o atributo de saída. Por esse motivo, diz-se que estes algoritmos seguem o paradigma de aprendizagem não supervisionada. Por exemplo, uma tarefa descritiva de agrupamento de dados tem por meta encontrar grupos de objetos semelhantes no conjunto de dados. Outra tarefa descritiva consiste em encontrar regras de associação que relacionam um grupo de atributos com outro grupo de atributos.

A Figura 1.2 apresenta uma hierarquia de aprendizagem, de acordo com os tipos de tarefas de aprendizagem. No topo aparece a aprendizagem indutiva, processo pelo qual são realizadas as generalizações a partir dos dados. Em seguida, surgem os tipos de aprendizagem supervisionada (preditivo) e não supervisionada (descritivo). As tarefas supervisionadas distinguem-se pelo tipo dos rótulos dos dados: discreto, no caso de classificação; e contínuo, no caso de regressão. As tarefas descritivas são genericamente divididas em: agrupamento, em que os dados são agrupados de acordo com sua semelhança; sumarização, cujo objetivo é encontrar uma descrição simples e compacta de um conjunto de dados; e associação, que consiste em encontrar padrões frequentes de associações entre os atributos de um conjunto de dados. Com exceção da sumarização, as demais tarefas serão descritas neste livro.

Note-se que, apesar desta divisão básica de modelos em preditivos e descritivos, um

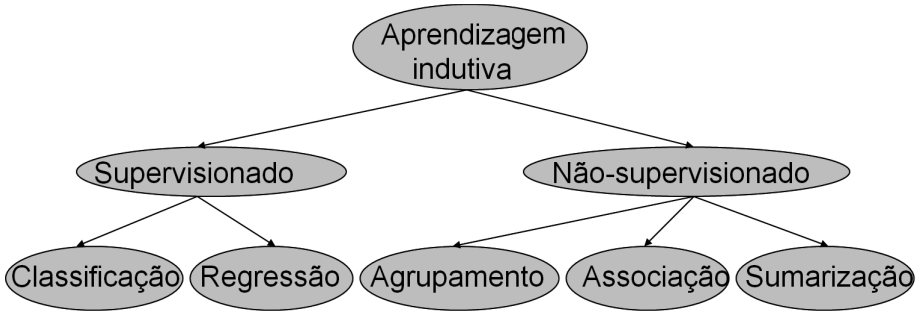


Figura 1.2 *Hierarquia de aprendizagem.*

modelo preditivo também providencia uma descrição compacta de um conjunto de dados, e um modelo descritivo pode efetuar previsões após ser validado.

Uma tarefa de aprendizagem que não se enquadra nas tarefas anteriores é a de *aprendizagem por reforço*. Nesta tarefa, o objetivo é reforçar, ou recompensar, uma ação considerada positiva, e punir uma ação considerada negativa. Um exemplo de tarefa de reforço é ensinar um robô a encontrar a melhor trajetória entre dois pontos. Os algoritmos de aprendizagem utilizados nesta tarefa, em geral, punem a passagem por caminhos pouco promissores, e recompensam a passagem por caminhos promissores. Devido ao foco adotado para este livro, esta tarefa não será abordada.

1.5 Estrutura do Livro

Este livro tem por objetivo apresentar os principais conceitos e algoritmos de ECD e mostrar como ECD pode ser utilizada para na resolução de problemas reais. Para esse efeito, serão cobertos tanto temas tradicionais como resultados de pesquisas recentes na área.

De forma a agrupar os temas cobertos de uma maneira mais uniforme, os capítulos do livro foram organizados em três grandes temas ou módulos:

- **Preparação de dados:** engloba tópicos de descrição dos dados, análise estatística de dados e pré-processamento de dados.
- **Métodos preditivos:** este módulo está relacionado com o paradigma da aprendizagem supervisionada e, após definir os conceitos gerais referentes a este tema, descreve os principais algoritmos de aprendizagem preditiva, explica como as hipóteses podem ser combinadas formando comités, introduz possíveis estratégias para planear experiências com esses métodos, e descreve as principais métricas utilizadas na avaliação do seu desempenho.

- **Modelos descritivos:** este módulo foca a aprendizagem não supervisionada. São abordados os temas de padrões frequentes e análise de agrupamentos. Descrevemos os conceitos básicos, os algoritmos principais, e as formas de combinação. É também discutido como as experiências utilizando estes métodos podem ser planeados e avaliados.
- **Tópicos avançados:** inclui temas de investigação recente na área de ECD. Os temas considerados são: fluxos de dados, meta-aprendizagem, estratégias para classificação multiclasse, classificação hierárquica, classificação multirótulo e análise de redes sociais.

Estes tópicos foram cuidadosamente escolhidos, de modo a que os leitores tenham acesso a uma dose equilibrada entre abrangência e profundidade dos temas básicos e avançados nas áreas de Inteligência Artificial, que utilizam aprendizagem automática na indução de modelos de decisão. Esperamos que este livro, ao mesmo tempo que introduz o leitor aos principais aspetos de ECD e a temas de investigação recentes, sirva de alicerce à realização de investigação que promova o crescimento e o fortalecimento da área. Esperamos ainda que o livro estimule o leitor a utilizar as várias técnicas aqui abordadas na resolução de problemas reais.

Parte I

Preparação de Dados

Introdução

Todos os dias é gerada uma enorme quantidade de dados. Estima-se que, a cada 20 meses, a quantidade de dados armazenada em todas as bases de dados do mundo duplica (Witten et al., 2011). Estes dados são gerados por atividades como transações financeiras, monitorização ambiental, obtenção de dados clínicos e genéticos, captura de imagens, tráfego na internet, entre outras. Os dados podem, ainda, assumir vários formatos diferentes, como séries temporais, conjuntos de produtos em transações, grafos ou redes sociais, textos, páginas web, imagens (vídeos) e áudio (músicas). Com o aumento crescente da quantidade de dados gerada, o fosso entre a quantidade de dados existente e a porção de dados que é analisada e compreendida tem-se acentuado de forma significativa ao longo do tempo.

Conjuntos de dados são formados por objetos que podem representar um objeto físico, como uma cadeira, ou uma noção abstrata, como os sintomas apresentados por um paciente que se dirige a um hospital. Em geral, cada objeto é descrito por um conjunto de atributos de entrada, ou vetor de caraterísticas. Cada objeto corresponde a uma ocorrência dos dados. Cada atributo está associado a uma propriedade do objeto.

Formalmente, um conjunto de dados pode ser representado por uma matriz de objetos $\mathbf{X}_{n\times d}$, em que n é o número de objetos e d é o número de atributos de entrada de cada objeto. O valor de d define a dimensionalidade dos objetos, ou do espaço de objetos (também denominado espaço de entradas, ou espaço de atributos). Cada elemento dessa matriz, x_i^j, contém o valor da j-ésima caraterística para o i-ésimo objeto. Os d atributos também podem ser vistos como um conjunto de eixos ortogonais e os objetos, como pontos no espaço de dimensão d, também designado por espaço de objetos. A Figura 1.3 ilustra um exemplo de um espaço de objetos. Nesse espaço, a posição de cada objeto é definida pelos valores de dois atributos de entrada ($d = 2$) que, neste caso, representam os resultados de dois exames clínicos. O atributo de saída é representado pelo formato do objeto na figura: círculo para pacientes doentes e triângulo para pacientes saudáveis.

Apesar do crescente número de bases de dados disponíveis, na maioria das vezes não é possível utilizar diretamente algoritmos de ECD sobre esses dados. Técnicas de pré-processamento são frequentemente utilizadas para tornar os conjuntos de dados mais adequados para o uso de algoritmos de ECD. Essas técnicas podem ser agrupadas nas seguintes tarefas: **Integração de dados**, **Amostragem de dados**, **Balanceamento de dados**, **Limpeza de dados**, **Redução de dimensionalidade**, e **Transformação de dados**.

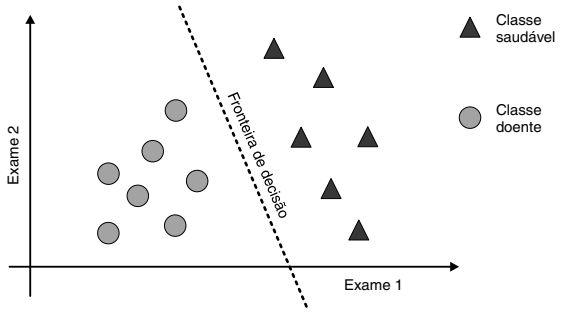


Figura 1.3 *Objetos no espaço definido pelos atributos.*

Grande parte das empresas, órgãos do governo e outras organizações possuem os seus dados armazenados em bases de dados. Assim, os dados podem ser oriundos de mais do que uma fonte, ou tabela atributo-valor. Quando os dados presentes em diferentes conjuntos precisam ser utilizados por um algoritmo de ECD, esses conjuntos devem ser integrados de forma a constituir uma única tabela. Essa integração pode levar a inconsistências e redundâncias. Os algoritmos de ECD também podem apresentar dificuldades quando precisam de lidar com um grande volume de dados. Essa grande quantidade pode estar relacionada com o número de objetos, com o número de atributos, ou ambos. Problemas como a redundância e a inconsistência, estão muitas vezes relacionados com a quantidade de dados. Técnicas de amostragem e de seleção de atributos têm sido empregues para amenizar estes problemas. Em dados reais, a distribuição dos objetos entre as classes pode não ser uniforme. Por conseguinte, algumas classes podem ter um número de objetos muito superior a outras, formando um conjunto de dados desbalanceado. Alguns algoritmos de ECD têm dificuldade em induzir um bom modelo a partir de conjuntos desbalanceados. Muitos dos conjuntos de dados reais apresentam problemas, tais como, a presença de ruído e dados incompletos e/ou inconsistentes. Os dados podem estar incompletos devido à ausência de valores. Os dados podem ser inconsistentes por causa de erros na sua geração, captação ou entrada. O desempenho da maioria dos algoritmos de ECD é afetado pela presença destes problemas. Para lidar com eles, diversas técnicas para limpeza de dados têm sido propostas e investigadas na literatura de ECD. Mesmo após a eliminação de atributos por especialistas no domínio, os atributos que restam podem dificultar a tarefa de algoritmos de ECD, devido a diversos motivos, como sejam a presença de um número muito grande de atributos, de atributos redundantes, irrelevantes e/ou inconsistentes.

Vários algoritmos de ECD têm dificuldade em utilizar os dados no seu formato original. Para tratar este problema, são efetuadas transformações aos dados originais antes destes serem utilizados pelo algoritmo. Um exemplo de transformação é a conversão de valores simbólicos em valores numéricos.

Capítulo 2

Análise Exploratória de Dados

A análise das caraterísticas presentes num conjunto de dados permite a descoberta de padrões e tendências que podem fornecer informações valiosas que ajudem a compreender o processo que gera os dados. Muitas dessas caraterísticas podem ser obtidas por meio da aplicação de fórmulas estatísticas simples. Outras podem ser observadas por meio do uso de técnicas de visualização. Neste capítulo são descritas as principais caraterísticas para a análise e compreensão de um conjunto de dados utilizados em experiências de ECD. Analisa-se a forma como os dados podem estar organizados e os tipos de valores que estes podem assumir. Serão apresentados ainda vários tipos de gráficos que facilitam a análise visual da distribuição dos valores em conjuntos de dados com uma ou mais variáveis. Este capítulo está organizado da seguinte maneira. A Seção 2.1 descreve como os atributos de um conjunto de dados podem ser caraterizados pelo seu tipo e escala. Por fim, na Seção 2.2, são apresentadas várias medidas, assim como gráficos, que permitem descrever conjuntos de dados, tanto univariados como multivariados.

2.1 Caraterização de Dados

Os conjuntos de dados são formados por objetos que podem representar um objeto físico, como uma cadeira, ou uma noção abstrata, como os sintomas apresentados por um paciente que se dirige a um hospital. Estes objetos são também designados por instâncias, objetos, registos ou exemplos. Em geral, são representados por um vetor de caraterísticas que os descreve, também denominadas atributos, campos ou variáveis. Cada objeto corresponde a uma linha na tabela de dados. Cada atributo está associado a uma propriedade do objeto.

Formalmente, os dados podem ser representados por uma matriz de objetos $\mathbf{X}_{n\times d}$, em que n é o número de objetos e d é o número de atributos de entrada de cada objeto. O valor de d define a dimensionalidade dos objetos ou do espaço de objetos. Cada elemento dessa matriz, x_i^j, contém o valor da j-ésima caraterística para o i-ésimo objeto. Os d atributos também podem ser interpretados como um conjunto de eixos ortogonais, e os objetos como pontos no espaço de dimensão d, denominado espaço de objetos.

Para ilustrar os conceitos abordados neste capítulo com um exemplo prático, considere

novamente o conjunto de dados provenientes de pacientes de um hospital, denominado `hospital`, ilustrado pela Tabela 2.1. Este conjunto foi inicialmente apresentado no Capítulo 1. No conjunto `hospital`, cada objeto corresponde a um paciente, sendo por isso formado pelos valores de atributos de entrada (também denominados atributos preditivos) referentes ao paciente. Esses atributos são: identificação, nome, idade, sexo, sintomas e resultados de exames clínicos. Exemplos de sintomas são presença e distribuição de manchas na pele, peso do paciente e temperatura do seu corpo. Além destes atributos, a tabela apresenta um atributo alvo, também denominado atributo meta ou de saída, que representa o fenómeno de interesse sobre o qual se pretende fazer previsões. Em tarefas descritivas, os dados não apresentam este atributo e, muitas vezes, a definição deste tipo de atributo pode ser obtida como um dos seus resultados. Por outro lado, as tarefas preditivas baseiam-se na presença deste atributo, como mencionado no Capítulo 1. Na maioria dos casos, os dados apresentam apenas um atributo alvo [1].

Quando um atributo alvo contém rótulos que identificam categorias ou classes às quais os objetos pertencem, é denominado classe e assume valores discretos $1,\ldots,k$. Nestes casos, estamos perante um problema de classificação. Se o número de objetos por classe for diferente, a classe mais frequente é denominada classe maioritária, e a menos frequente, minoritária. Por outro lado, se o atributo alvo é descrito por valores numéricos contínuos, estamos perante um problema de regressão (Mitchell, 1997). Um caso especial de regressão é a previsão de valores em séries temporais, que se carateriza pelo fato de os seus valores apresentarem uma relação de periodicidade. Quer em problemas de classificação quer em problemas de regressão, os restantes atributos são denominados atributos preditivos, por poderem ser utilizados na previsão do valor do atributo alvo.

Na Tabela 2.1, para cada paciente são apresentados os valores dos atributos *Id.* (identificação do paciente), *Nome*, *Idade*, *Sexo*, *Peso*, *Manchas* (presença e distribuição de manchas no corpo), *Temp.* (temperatura do corpo), *#Int.* (número de internamentos), *Est.* (estado de origem) e *Diagnóstico*, que indica o diagnóstico do paciente e corresponde ao atributo alvo. As tabelas com este formato também são denominadas por tabelas atributo-valor.

O domínio de um atributo, ou seja os possíveis valores que um atributo pode assumir, determina o tipo de análise que podemos efetuar. Neste livro, consideramos dois aspetos: *tipo* e *escala*. O tipo de um atributo diz respeito ao grau de quantificação nos dados, e a escala indica a significância relativa dos seus valores. Conhecer o tipo e a escala dos atributos permite identificar a forma adequada para a preparação dos dados e posterior modelação. As definições que apresentamos são utilizadas para classificar os valores que os atributos podem assumir no que diz respeito aos dois aspetos mencionados (Jain e Dubes, 1988; Barbara, 2000; Yang et al., 2005).

[1] Resultados mais recentes consideram dados com mais de um atributo alvo. Este é o foco, por exemplo, da classificação multirótulo.

Tabela 2.1 *Conjunto de dados* `hospital` *com seus atributos*

Id.	Nome	Idade	Sexo	Peso	Manchas	Temp.	# Int.	Est.	Diagnóstico
4201	João	28	M	79	Concentradas	38,0	2	SP	Doente
3217	Maria	18	F	67	Inexistentes	39,5	4	MG	Doente
4039	Luiz	49	M	92	Espalhadas	38,0	2	RS	Saudável
1920	José	18	M	43	Inexistentes	38,5	8	MG	Doente
4340	Cláudia	21	F	52	Uniformes	37,6	1	PE	Saudável
2301	Ana	22	F	72	Inexistentes	38,0	3	RJ	Doente
1322	Marta	19	F	87	Espalhadas	39,0	6	ECD	Doente
3027	Paulo	34	M	67	Uniformes	38,4	2	GO	Saudável

2.1.1 Tipo

O tipo define se o atributo representa quantidades, sendo então denominado quantitativo ou numérico, ou qualidades, sendo então designado de qualitativo, simbólico ou categórico, pois os seus valores podem ser associados a categorias. Exemplos de conjuntos de valores qualitativos são {*pequeno*, *médio*, *grande*} e {*matemática*, *física*, *química*}. Apesar de alguns atributos qualitativos poderem ter os respetivos valores ordenados, não podem ser aplicadas operações aritméticas. Os atributos quantitativos são numéricos, como no conjunto de valores $\{23, 45, 12\}$. Os valores de um atributo quantitativo são ordenados e podem ser utilizados em operações aritméticas. Os valores quantitativos podem ser ainda contínuos ou discretos.

Os atributos contínuos podem assumir um número infinito de valores. Geralmente esses atributos são resultados de medidas, sendo os seus valores representados por números reais. No entanto, deve-se timar em consideração que em computadores digitais a precisão para valores reais é, geralmente, limitada. Exemplos de atributos contínuos são atributos que representam pesos, tamanhos ou distâncias.

Os atributos discretos contêm um número finito ou infinito contável de valores. Um caso especial de atributos discretos são os atributos binários (ou booleanos), que apresentam apenas dois valores, como 0/1, sim/não, ausência/presença e verdadeiro/falso. Para efeitos ilustrativos, na Tabela 2.2 é apresentada a classificação por tipo dos atributos presentes no conjunto de dados da Tabela 2.1.

É importante observar que uma medida quantitativa possui, além do valor numérico, uma unidade, por exemplo, *metro*. No processo de extração de conhecimento, se o atributo *altura* assume o valor 100, o valor em si não indica se a altura é medida em centímetros, metros ou jardas, e essa informação pode ser importante na avaliação do conhecimento adquirido.

Os atributos quantitativos ou numéricos podem assumir valores binários, inteiros ou reais. Por outro lado, atributos qualitativos são, geralmente, representados por um número finito de símbolos ou nomes. Em alguns casos, os atributos categóricos são representados por números. No entanto, estes números não representam quantidades pelo que não são passíveis de serem submetidos a operações aritméticas. Por exemplo, qual seria o sentido

Tabela 2.2 *Tipo dos atributos do conjunto* `hospital`

Atributo	Classificação
Id.	Qualitativo
Nome	Qualitativo
Idade	Quantitativo discreto
Sexo	Qualitativo
Peso	Quantitativo contínuo
Manchas	Qualitativo
Temp.	Quantitativo contínuo
#Int.	Quantitativo discreto
Est.	Qualitativo
Diagnóstico	Qualitativo

de calcular a média dos valores de um atributo categórico representando o número de identificação de um paciente?

2.1.2 Escala

A escala define as operações que podem ser realizadas sobre os valores do atributo. Em relação à escala, os atributos podem ser classificados como nominais, ordinais, intervalares e racionais. Os dois primeiros são do tipo qualitativo e os dois últimos são do tipo quantitativo. Estas quatro escalas são seguidamente definidas em detalhe.

Na escala nominal, os valores consistem apenas nomes diferentes, carregando a menor quantidade de informação possível, não existindo uma relação de ordem entre os seus valores. Consequentemente, as operações mais utilizadas na manipulação dos seus valores são as de igualdade e desigualdade entre valores. Por exemplo, se o atributo representa continentes do planeta, apenas é possível verificar se dois valores são iguais ou diferentes (a não ser que se pretenda ordenar os continentes por ordem alfabética, mas nesse caso o atributo seria do tipo ordinal). São exemplos de atributos com escala nominal: nome do paciente, RG, CPF, número da conta no banco, CEP, cores (com as categorias verde, amarelo, branco etc.) e sexo (com as categorias feminino e masculino).

Os valores numa escala ordinal refletem também uma ordem das categorias representadas. Dessa forma, além dos operadores anteriores, podem também ser utilizados operadores como $<$, $>$, $\leq$, $\geq$. Por exemplo, quando um atributo categórico possui como domínio o conjunto *pequeno*, *médio* e *grande*, é possível definir uma relação de ordem, ou seja indicar se um valor é igual, maior ou menor que outro. Exemplos de atributos com escala ordinal incluem a hierarquia militar e avaliações qualitativas de temperatura, como frio, morno e quente.

Na escala intervalar, os atributos são representados por números que variam dentro de um intervalo. Assim, é possível definir tanto a ordem como a diferença em magnitude entre dois valores. A diferença em magnitude indica a distância que separa dois valores no

Tabela 2.3 *Escala dos atributos do conjunto* `hospital`

Atributo	Classificação
Id.	Nominal
Nome	Nominal
Idade	Racional
Sexo	Nominal
Peso	Racional
Manchas	Nominal
Temp.	Intervalar
#Int.	Racional
Est.	Nominal
Diagnóstico	Nominal

intervalo de possíveis valores. O valor zero não tem o mesmo significado que o zero utilizado em operações aritméticas. Por exemplo, considere a escala de temperatura medida em graus Celsius. Se o instituto de previsão do tempo em dias de verão, para uma dada cidade, informa que a temperatura vai variar entre 26 e 34 graus e em dias de inverno, para a mesma cidade, informa que vai variar entre 13 e 21 graus, temos que a temperatura dessa cidade apresenta uma variação de 8 graus entre a mínima e a máxima, independente da estação do ano. por outro lado, 90 graus Celsius é diferente de 90 graus Fahrenheit, apesar de ambos os valores se referirem ao atributo temperatura, do mesmo modo que não é possível afirmar que a cidade é duas vezes mais quente no verão do que no inverno. Não faz sentido, para este tipo de atributo, utilizar como informação o quociente entre dois valores. Isto ocorre porque o ponto em que o atributo assume o valor 0, denominado de ponto zero ou origem da escala, é definido de forma arbitrária. Esta é uma caraterística dos atributos intervalares. Este problema seria eliminado caso fosse utilizada a escala de temperatura Kelvin, cujo valor do ponto zero é o ponto zero verdadeiro (Pyle, 1999). Outros exemplos são a duração de um evento em minutos e datas num calendário.

Os atributos com escala racional são os que contêm mais informação. Os números têm um significado absoluto, ou seja, existe um zero absoluto juntamente com uma unidade de medida que confere significado ao quociente. Por exemplo, considerando o número de vezes que uma pessoa foi ao hospital, o ponto zero está associado à ausência de visitas. Se um paciente foi internado duas vezes e outro foi internado oito vezes, é correto afirmar que o segundo paciente esteve internado quatro vezes mais que o primeiro. Ou seja, faz sentido, para estes atributos, utilizar a razão entre dois valores. Outros exemplos de atributos com escala de razão incluem o tamanho, a distância e os valores financeiros, tais como salário ou saldo em conta-corrente.

Tendo por base a descrição anterior, os atributos do conjunto de dados da Tabela 2.1 podem ser classificados como indicado na Tabela 2.3.

Recentemente, têm sido adotados tipos mais complexos de atributos, como os atributos

hierárquicos, cujos valores representam uma hierarquia de valores, ou atributos simbólicos, cujos valores podem assumir intervalos ou histogramas.

2.2 Exploração de Dados

A informação obtida na análise exploratória de dados pode ajudar, por exemplo, na seleção das técnicas mais apropriadas para o pré-processamento dos dados assim como para a construção de modelos preditivos. Uma das formas mais simples de explorar um conjunto de dados consiste em extrair estatísticas descritivas. A estatística descritiva resume de forma quantitativa as principais caraterísticas de um conjunto de dados. Muitas dessas medidas são calculadas rapidamente. Por exemplo, no conjunto de dados de pacientes, duas medidas estatísticas podem ser facilmente calculadas: a idade média dos pacientes e a percentagem de pacientes do sexo masculino.

A estatística descritiva assume que os dados são gerados por um processo estatístico. Como o processo pode ser caraterizado por vários parâmetros, as medidas podem ser interpretadas como estimativas dos parâmetros estatísticos da distribuição que gerou os dados. Por exemplo, os dados podem ter sido gerados por uma distribuição normal com média 0 e variância 1. Estas medidas permitem capturar informações como:

- Frequência;
- Localização ou tendência central (por exemplo, a média);
- Dispersão (por exemplo, o desvio padrão);
- Distribuição ou formato.

A medida mais simples é a frequência, que mede a proporção de vezes que um atributo assume um dado valor num determinado conjunto de dados. Pode ser aplicada quer a valores quer numéricos quer a valores simbólicos. Um exemplo da sua utilização seria: *Num conjunto de dados médicos, 40% dos pacientes têm febre*.

As outras medidas diferem nos casos em que os dados apresentam apenas um atributo (dados univariados) ou mais do que um atributo (dados multivariados), sendo geralmente aplicadas a dados numéricos. Apesar da maioria dos conjuntos de dados utilizados em ECD apresentar mais de um atributo, as análises realizadas para cada atributo podem oferecer informações valiosas sobre os dados.

Seguidamente serão apresentadas as principais medidas de centralidade, dispersão e distribuição para dados uni e multivariados.

2.2.1 Dados Univariados

Para os dados univariados, um objeto x_i é descrito por apenas um atributo. Um conjunto com n objetos pode ser então representado por $\mathbf{x}^j = \{x_1, x_2, ..., x_n\}$, em que cada x_i é representado por um único valor. É importante observar que o termo *conjunto* não tem o

mesmo significado do termo utilizado em teoria dos conjuntos. Num conjunto de dados, o mesmo valor pode surgir mais do que uma vez num mesmo atributo.

Medidas de Centralidade

As medidas de centralidade definem pontos de referência nos dados e variam para dados numéricos e simbólicos. Para dados simbólicos, geralmente utiliza-se a moda, que corresponde ao valor encontrado com maior frequência para um dado atributo. Por exemplo, a moda do atributo manchas na Tabela 2.1 é o valor *Inexistentes*, que aparece em três dos oito pacientes.

Para os atributos numéricos, as medidas mais utilizadas são a média, a mediana e o percentil. Suponha um conjunto de n valores numéricos: $\mathbf{x}^j = \{x_1, x_2, ..., x_n\}$. O valor médio desse conjunto pode ser facilmente calculado pela Equação 2.1.

$$\bar{x}^j = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{2.1}$$

Um dos problemas associados à média é a sua sensibilidade à presença de *outliers*, que são valores muito diferentes dos demais valores observados para o mesmo atributo (ver detalhes no Capítulo 3). A média é um bom indicador do valor central de um conjunto de valores apenas se os valores estão distribuídos de forma simétrica. Um valor muito mais alto ou muito mais baixo que os demais valores do conjunto pode gerar um valor da média distorcido, o que poderia mudar radicalmente se o *outlier* for retirado. Este problema é minimizado com o uso da mediana, que é menos sensível a *outliers*. Para utilizar a mediana, o primeiro passo consiste em ordenar de forma crescente o conjunto de valores. Ordenados os valores, a Equação 2.2 pode ser utilizada para o cálculo da mediana.

$$\text{mediana}(\mathbf{x}^j) = \begin{cases} \frac{1}{2}(x_r + x_{r+1}) & \text{se } n \text{ for par } (n = 2r) \\ x_{r+1} & \text{se } n \text{ for ímpar } (n = 2r+1) \end{cases} \tag{2.2}$$

Assim, se o número de valores, n, é ímpar, a mediana é igual ao valor do meio do conjunto ordenado. Caso contrário, se for par, é dada pela média dos dois valores do meio. Por exemplo, seja o conjunto de valores $\{17,4,8,21,4\}$. A ordenação desse conjunto gera a sequência de valores $(4,4,8,17,21)$. Observe que valores repetidos são mantidos na sequência. Como o número de valores é ímpar, 5, a mediana é dada pelo terceiro valor, isto é, mediana $= 8$. Se o conjunto de valores fosse formado pelos elementos, $\{17,4,8,21,4,15,13,9\}$, como o número de elementos, 8, é par, a mediana é dada pela média entre o quarto e o quinto valor da sequência ordenada $(4,4,8,9,13,15,17,21)$. Nesse caso, mediana $= (9+13)/2 = 11$. O recurso à mediana facilita a observação da assimetria da distribuição e da existência de *outliers*. Existem ainda variações da média e da mediana, como, por exemplo, a média truncada, que minimiza os problemas da média por meio do descarte dos exemplos nos extremos da sequência ordenada dos valores. Para isso, é necessário definir a percentagem de exemplos a serem eliminados em cada extremidade.

Algoritmo 2.1 Algoritmo para cálculo do percentil

Entrada: Um conjunto de n valores e o percentil p (valor real entre $0,0$ (equivale a 0%) e $1,0$ (equivale a 100%)) a ser retornado
Saída: Valor do percentil

```
1  Ordenar os n valores em ordem crescente
2  Calcular o produto np
3  se np não for um número inteiro então
4      Arredondar para o próximo inteiro
5      Retornar o valor dessa posição na sequência
6  fim
7  senão
8      Considerar np = k
9      Retornar a média entre os valores nas posições k e k + 1
10 fim
```

Outras medidas muito utilizadas são os quartis e os percentis. Assim como a mediana, estas medidas são utilizadas após prévia ordenação dos valores. Enquanto a mediana divide os dados ao meio, estas outras medidas utilizam pontos de divisão diferentes. Os quartis dividem os valores ordenados em quartos. Assim, o 1º quartil de uma sequência, $Q1$, é o valor para o qual 25% dos outros valores são inferiores a ele. Este também é o valor do 25º percentil, $P25$. O 2º quartil é a mediana, que é equivalente ao 50º percentil, $P50$.

Seja p um valor entre 0 e 100. O pº percentil, Pp, é um valor x_i do conjunto de valores tal que p% dos valores observados são menores que x_i. Assim, o 40º percentil, $P40$, de um conjunto de valores é o valor para o qual 40% dos demais valores são menores ou iguais a ele. Para calcular o pº percentil, basta seguir os passos do Algoritmo 2.1.

Uma forma simples de visualizar a distribuição dos dados consiste em utilizar técnicas de visualização como, por exemplo, *boxplots*, histogramas e *scatter plots*, que serão introduzidos posteriormente neste capítulo. Para efeitos de ilustrar algumas das medidas e gráficos, será utilizado o conjunto de dados `iris` (Fisher, 1936). O conjunto de dados `iris` é um dos mais populares em ECD, contendo 150 objetos, cada um deles caraterizado por quatro atributos de entrada (tamanho da sépala, largura da sépala, tamanho da pétala e largura da pétala), que assumem valores contínuos, e um atributo alvo com três valores nominais ou classes, que representam as espécies íris setosa, íris versicolor e íris virgínica.

A Figura 2.1 mostra duas variações do gráfico *boxplot* para o atributo largura da sépala, do conjunto de dados `iris`. O *boxplot*, também designado por diagrama de Box e Whisker ou *caixa de bigodes*, apresenta um resumo dos valores para o 1º , 2º (mediana) e 3º quartis, além dos limites inferior e superior.

Do lado esquerdo da figura é apresentado o gráfico *boxplot* original, onde a linha ho-

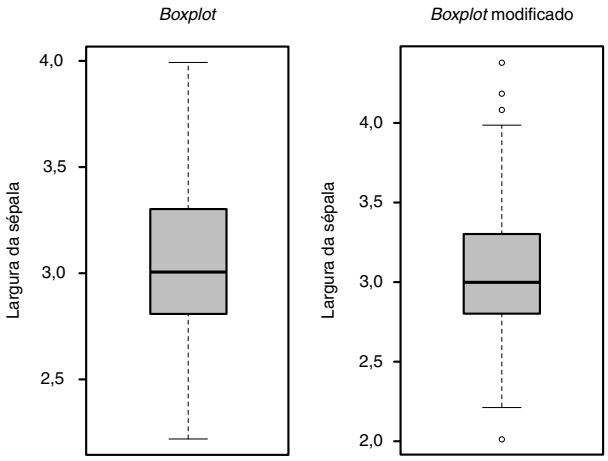


Figura 2.1 *Boxplots para o atributo largura da sépala do conjunto de dados* `iris`.

rizontal mais baixa e a linha horizontal mais alta indicam, respetivamente, os valores mínimo e máximo presentes nos dados. Os lados inferior e superior do retângulo representam o 1º quartil e o 3º quartil, respetivamente. A linha no interior do retângulo é o 2º quartil, ou mediana. O limite superior (inferior) da linha tracejada vai até o maior (menor) valor do conjunto de dados.

O gráfico da direita ilustra uma variação do gráfico *boxplot*, conhecido como *boxplot* modificado. Nesse gráfico, o limite superior (inferior) da linha tracejada vai até o maior (menor) valor apenas se esse valor não for muito distante do 3º (1º) quartil (no máximo $1,5 \times$ intervalo entre quartis). Os valores acima do limite superior e abaixo do limite inferior são considerados *outliers*. Neste gráfico, 4 valores *outliers* são representados por círculos, 3 maiores que 3º quartil + 1,5 $\times$ (3º quartil $-$ 1º quartil) e 1 menor que 1º quartil $-$ 1,5 $\times$ (3º quartil $-$ 1º quartil).

Medidas de Dispersão

As medidas de dispersão medem a variabilidade de um conjunto de valores. Permitem averiguar se os valores se encontram amplamente dispersos ou relativamente concentrados em torno de um valor, por exemplo, a média. As medidas de dispersão mais comuns são: **Intervalo**, **Variância**, **Desvio padrão**.

O intervalo é a medida mais simples indicando a dispersão máxima entre os valores de um conjunto. Assim, sejam $\mathbf{x}^j = \{x_1, x_2, ..., x_n\}$ os valores do atributo para n objetos. O intervalo desse conjunto é medido pela Equação 2.3.

$$\text{intervalo}(\mathbf{x}^j) = \max_{i=1,...,n}(x_i) - \min_{i=1,...,n}(x_i) \tag{2.3}$$

Se a maioria dos valores for próxima de um ponto, com um pequeno número de valores extremos, o intervalo não será uma boa medida da dispersão dos valores. A medida mais utilizada para avaliar a dispersão de valores é a variância, que é dada pela Equação 2.4.

$$\text{variância}(\mathbf{x}^j) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x}^j)^2 \tag{2.4}$$

Nesta equação, $\bar{x}^j$ representa a média dos valores de x. O uso do denominador $n-1$, conhecido por correção de Bessel, oferece uma melhor estimativa da variância verdadeira do que o uso de n. Outra medida de dispersão, o desvio padrão, é dada pela raiz quadrada da variância.

Assim como a média, o valor da variância pode ser distorcido pela presença de *outliers*, uma vez que a variância calcula a diferença entre cada valor e a média. Outras estimativas mais robustas da dispersão frequentemente utilizadas são:

- Desvio médio absoluto (AAD, do inglês *absolute average deviation*), ilustrado pela Equação 2.5.
- Desvio mediano absoluto (MAD, do inglês *median absolute deviation*), ilustrado pela Equação 2.6.
- Intervalo interquartil (IQR, do inglês *interquartil range*), ilustrado pela Equação 2.7.

$$AAD(\mathbf{x}^j) = \frac{1}{n}\sum_{i=1}^{n}\left|x_i - \bar{x}^j\right| \tag{2.5}$$

$$MAD(\mathbf{x}^j) = \text{mediana}\left(\left\{\left|x_1 - \bar{x}^j\right|, ..., \left|x_n - \bar{x}^j\right|\right\}\right) \tag{2.6}$$

$$IQR(\mathbf{x}^j) = P_{75\%} - P_{25\%} \tag{2.7}$$

Medidas de Distribuição

As medidas que são definidas em torno da média de um conjunto de valores, como as medidas média e desvio padrão são, na sua maioria, instanciações de uma medida denominada momento, que é definida pela Equação 2.8.

$$\text{momento}_k(\mathbf{x}^j) = \frac{\sum_{i=1}^{n}(x_i - \bar{x}^j)^k}{(n-1)} \tag{2.8}$$

Para cada valor do parâmetro k, uma medida diferente de momento é definida. Assim:

- quando $k = 1$, tem-se o valor 0, que é o primeiro momento em torno da origem ou primeiro momento central;

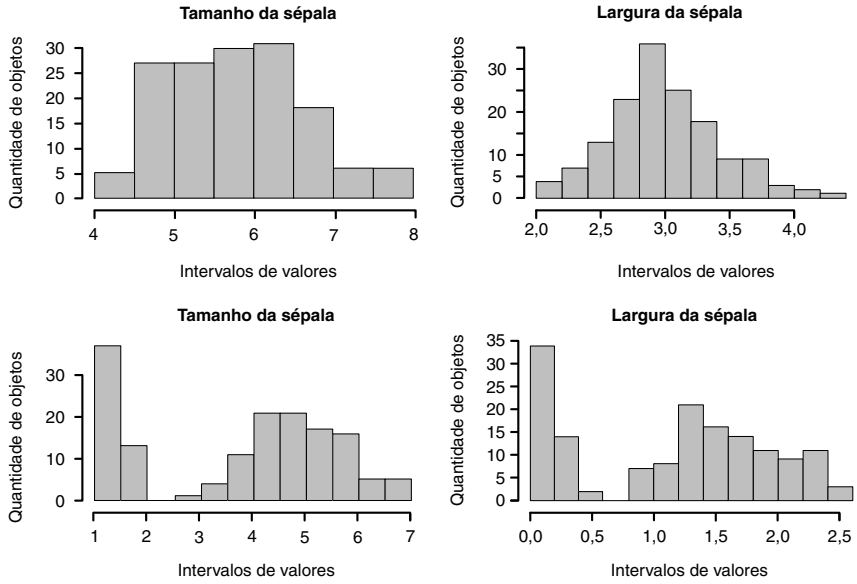


Figura 2.2 *Histograma para a distribuição de valores dos atributos de entrada do conjunto* `iris`.

- quando $k = 2$, tem-se a variância, que é o segundo momento central;
- quando $k = 3$, tem-se a obliquidade, que é o terceiro momento central;
- quando $k = 4$, tem-se a curtose, que é o quarto momento central.

Conforme mencionado, os dois primeiros momentos, o valor 0 e o desvio padrão, são medidas de localização e dispersão, respetivamente. O terceiro e quarto momentos, obliquidade e curtose, são medidas de distribuição, uma vez que mostram como os valores estão distribuídos.

Outra forma simples de visualizar a distribuição dos dados é por via da sua representação num histograma. Um histograma divide os valores de um conjunto de dados em intervalos. Para cada intervalo, é desenhada uma barra cuja altura é proporcional ao número de elementos nesse intervalo. Para valores numéricos, os valores são divididos em intervalos contíguos, geralmente, do mesmo tamanho. O formato do histograma depende do número de intervalos utilizados. Na Figura 2.2, é apresentado um histograma com a distribuição de valores para cada atributo do conjunto de dados `iris`.

O terceiro momento, obliquidade (em inglês *skewness*), mede a simetria da distribuição dos dados em torno da média. Numa distribuição simétrica, se os valores forem distribuídos em intervalos do mesmo tamanho, um histograma com a quantidade de valores em cada intervalo tem a mesma aparência à direita e à esquerda do ponto central. A Equação 2.9 ilustra o cálculo da obliquidade, que utiliza $k = 3$ e divide o valor calculado pelo desvio padrão elevado ao cubo, s^3, para tornar a medida independente da escala.

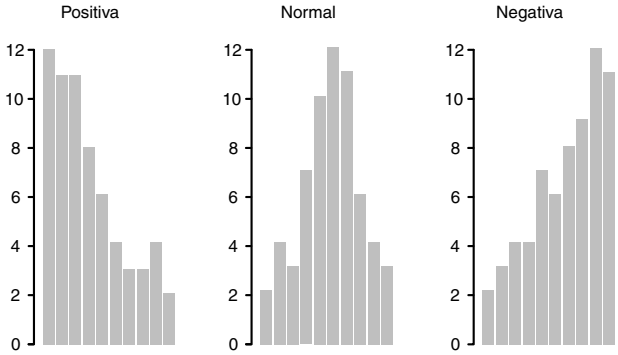


Figura 2.3 *Distribuição dos valores de obliquidade.*

$$\text{obliquidade}(\mathbf{x}^j) = \frac{\text{momento}_3(\mathbf{x}^j)}{s^3} = \frac{\sum_{i=1}^{n}(x_i - \bar{x}^j)^3}{(n-1)s^3} \tag{2.9}$$

A distribuição dos valores num conjunto de dados está associada ao valor da obliquidade da seguinte forma:

- obliquidade $= 0$ (simétrica): a distribuição é aproximadamente simétrica (ocorre para uma distribuição normal);
- obliquidade > 0 (positiva): a distribuição concentra-se mais no lado esquerdo;
- obliquidade < 0 (negativa): a distribuição concentra-se mais no lado direito.

A Figura 2.3 ilustra estes três tipos de distribuição medidos pela obliquidade.

O quarto momento calcula a curtose (em inglês *kurtosis*), que é uma medida de dispersão que captura o achatamento da função de distribuição. A medida de curtose, calculada pela Equação 2.10, verifica se os dados apresentam um pico ou se são achatados em relação a uma distribuição normal. À semelhança da Equação 2.9, para tornar a medida independente de escala, o quarto momento é dividido pelo desvio padrão elevado à quarta potência, s^4.

$$\text{curtose}(\mathbf{x}^j) = \frac{\text{momento}_4(\mathbf{x}^j)}{s^4} = \frac{\sum_{i=1}^{n}(x_i - \bar{x}^j)^4}{(n-1)s^4} \tag{2.10}$$

Visto que, para uma distribuição normal com média 0 e variância 1, o valor da curtose é igual a 3, é feita uma correção na sua fórmula para que a distribuição normal padrão tenha curtose igual a 0. Para isso, geralmente é utilizada a Equação 2.11.

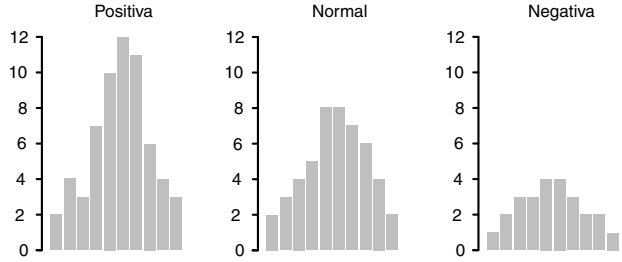


Figura 2.4 *Distribuição dos valores de curtose.*

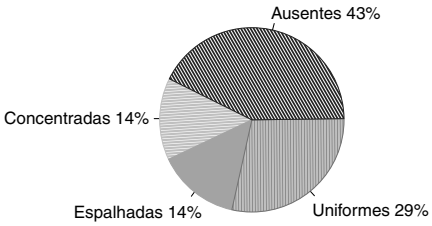


Figura 2.5 *Gráfico circular para a distribuição de valores do atributo Manchas.*

$$\text{curtose}(\mathbf{x}^j) = \frac{\text{momento}_4(\mathbf{x}^j)}{\sigma^4} - 3 = \frac{\sum_{i=1}^{n}(x_i - \bar{x}^j)^4}{(n-1)\sigma^4} - 3 \tag{2.11}$$

Assim como na medida de obliquidade, a seguinte relação, ilustrada pela Figura 2.4, é observada entre o valor da curtose e a distribuição dos valores num conjunto de dados:

- curtose $= 0$ (normal): o histograma de distribuição dos dados apresenta o mesmo achatamento que uma distribuição normal;
- curtose > 0 (positiva): o histograma de distribuição dos dados apresenta uma distribuição mais alta e concentrada que a distribuição normal;
- curtose < 0 (negativa): o histograma de distribuição dos dados apresenta uma distribuição mais achatada que a distribuição normal.

Outro gráfico muito utilizado para ilustrar a distribuição de um conjunto de valores é o gráfico circular. Neste gráfico, cada valor ocupa uma fatia cuja área é proporcional ao número de vezes que o valor surge no conjunto de dados. Este tipo de gráfico é indicado para valores qualitativos. A Figura 2.5 mostra a distribuição dos valores do atributo *Manchas* do conjunto de dados `hospital` utilizando um gráfico circular. Para atributos quantitativos, os valores podem, à semelhança do que é feito em histogramas, ser agrupados em intervalos.

2.2.2 Dados Multivariados

Dados multivariados são aqueles que possuem mais do que um atributo de entrada. O conjunto de dados `iris` é um conjunto multivariado. Nesses casos, as medidas de centralidade podem ser obtidas calculando a medida de centralidade de cada atributo separadamente. Por exemplo, a média para um conjunto de objetos com d atributos pode ser calculada pela Equação 2.12.

$$\bar{\mathbf{x}} = (\bar{x}^1, ..., \bar{x}^d) \tag{2.12}$$

As medidas de dispersão podem ser calculadas para cada atributo independentemente dos demais utilizando qualquer medida de dispersão. Conforme visto na seção anterior para o conjunto de dados `iris`, cada atributo pode ser explorado visualmente de forma independente.

Os dados multivariados permitem ainda análises sobre a relação entre dois ou mais atributos. Por exemplo, para atributos quantitativos, a dispersão de um conjunto de dados é capturada melhor através de uma matriz de covariância, em que cada elemento da matriz corresponde à covariância entre dois atributos. Cada elemento cov_{ij} de uma matriz de covariância **Cov** mede a covariância entre os atributos $\mathbf{x}^i$ e $\mathbf{x}^j$, que é dada pela Equação 2.13.

$$\text{covariância}(\mathbf{x}^i, \mathbf{x}^j) = \frac{1}{n-1} \sum_{k=1}^{n} (x_k^i - \bar{x}^i)(x_k^j - \bar{x}^j) \tag{2.13}$$

Nesta equação, $\bar{x}^i$ representa o valor médio do i-ésimo atributo e x_k^i, o valor do i-ésimo atributo para o k-ésimo objeto. É importante observar que covariância($\mathbf{x}^i, \mathbf{x}^i$) = variância($\mathbf{x}^i$). Deste modo, a diagonal da matriz de covariância contém as variâncias dos atributos.

A covariância entre dois atributos mede o grau com que os atributos variam juntos. Seu valor depende da magnitude dos atributos. Um valor próximo de 0 indica que os atributos não têm um relacionamento linear. Um valor positivo indica que os atributos são diretamente relacionados. Quando o valor de um dos atributos aumenta, o do outro também aumenta. O contrário ocorre se a covariância for negativa. A medida de covariância é afetada pela dimensão dos atributos avaliados. Assim, dois atributos com dimensão elevada podem apresentar um valor de covariância maior do que dois atributos mais semelhantes entre si, mas de menor dimensão. Por isso, não é possível avaliar o relacionamento entre dois atributos observando apenas a covariância entre eles. A medida de correlação elimina esse problema removendo a influência da escala. Como resultado, a correlação apresenta uma indicação mais clara da força da relação linear entre dois atributos, sendo, por esse motivo, mais utilizada para explorar dados multivariados do que a covariância. A matriz de correlação apresenta a correlação entre cada possível par de atributos de um conjunto de dados. Cada elemento da matriz de correlação tem o seu valor definido pela Equação 2.14.

$$\text{correlação}(\mathbf{x}^i, \mathbf{x}^j) = \frac{\text{covariância}(\mathbf{x}^i, \mathbf{x}^j)}{s_i s_j} \tag{2.14}$$

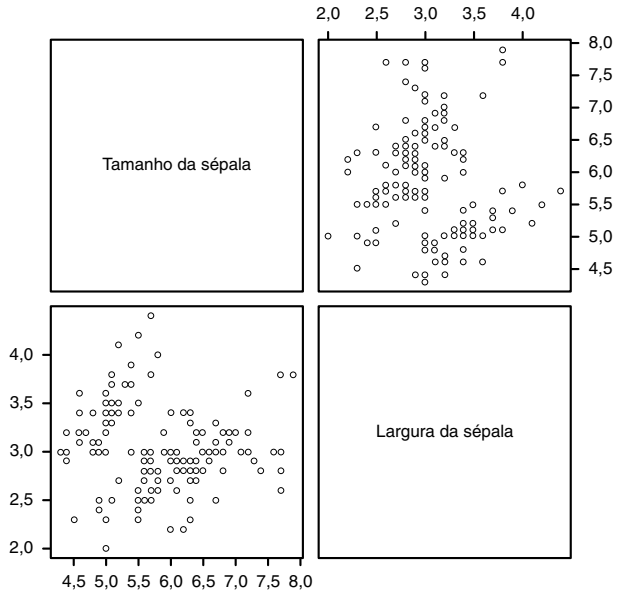


Figura 2.6 *Matriz de scatter plot para dois atributos do conjunto de dados* `iris`.

Nesta equação, $\mathbf{x}^i$ é o i-ésimo atributo e s_i é o desvio padrão dos valores desse atributo. De realçar que correlação$(\mathbf{x}^i, \mathbf{x}^i) = 1$. Desta forma, os elementos da diagonal têm valor 1. Os outros elementos assumem um valor entre -1 (correlação negativa máxima) e $+1$ (correlação positiva máxima).

Assim como na medida de covariância, quando dois atributos apresentam uma correlação positiva, o aumento do valor de um deles é geralmente acompanhado por um aumento no valor do outro. Da mesma forma, quando dois atributos estão negativamente correlacionados, a redução no valor de um deles é geralmente acompanhada da redução do valor do outro.

A análise de dados multivariados pode ser facilitada pelo uso de recursos de visualização. Assim como histogramas, gráficos de pizza e *boxplots* são utilizados para visualizar dados univariados, outros diagramas têm sido adotados para visualizar dados multivariados, particularmente a relação entre os diferentes atributos. Entre esses diagramas, um dos mais utilizados é o *scatter plot*, que ilustra a correlação linear entre dois atributos.

Num *scatter plot*, a cada objeto, considerando apenas dois de seus atributos, é associada uma posição ou ponto num plano bidimensional. Os valores dos atributos, que podem ser números inteiros ou reais, definem as coordenadas desse ponto. Na Figura 2.6, um *scatter plot* ilustra a correlação entre dois atributos do conjunto de dados `iris`: tamanho da sépala e largura da sépala.

No *scatter plot* do canto superior direito desta figura, cada ponto representa a *Largura da sépala* (eixo horizontal) e o *Tamanho da sépala* (eixo vertical) de um dos 150 objetos

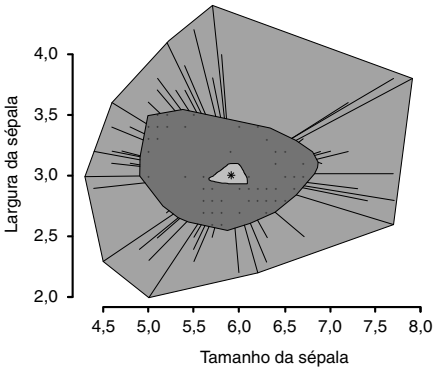


Figura 2.7 *Diagramas de bagplot para dois atributos do conjunto de dados* `iris`.

do conjunto de dados `iris`. No *scatter plot* do canto inferior esquerdo ocorre o contrário, isto é, cada ponto representa o *Tamanho da sépala* (eixo horizontal) e a *Largura da sépala* (eixo vertical) de um objeto. Gráficos de *scatter plot* para diferentes combinações de atributos podem ser exibidos em matrizes de *scatter plot*. Embora os *scatter plots* com duas dimensões sejam os mais comuns, também podem ser utilizados para visualizar a relação entre três atributos, definindo *scatter plots* tridimensionais. Os atributos adicionais podem ser exibidos utilizando tamanho, formato e cor nos marcadores que representam os objetos. Quando as classes dos objetos são disponibilizadas, o *scatter plot* pode ser usado para investigar o grau com que dois atributos separam as classes, se for possível separar a maioria dos objetos de uma das classes com uma reta.

Outras técnicas de visualização para dados multivariados bastante utilizadas são os *bagplots*, as faces de Chernoff, os *star plots* e os *heatmaps* (Wu et al., 1998).

O *bagplot* é uma generalização bivariada do *boxplot* que permite apresentar, numa mesma figura, o *boxplot* de dois atributos ou variáveis (Rousseeuw et al., 1999). Cada eixo do gráfico pode ser considerado um *boxplot* associado a um dos dois atributos. A Figura 2.7 ilustra um gráfico de *bagplot* para dois atributos do conjunto de dados `iris`: tamanho da sépala e largura da sépala. O gráfico apresenta três regiões convexas. A menor delas tem apenas um objeto, representado por um asterisco, cujas coordenadas são definidas pela mediana de seus dois atributos. A segunda região, denominada *bag*, possui 50% dos objetos, cujas coordenadas são definidas pelos valores entre o primeiro e terceiro quartis para cada um dos atributos. A maior região, chamada *loop*, é a *bag* expandida três vezes o intervalo entre quartis nas duas dimensões, $1,5$ vez em cada sentido. Para cada eixo, o valor máximo e o valor mínimo são definidos, respetivamente, pelos limites superior e inferior do atributo correspondente. Objetos que se encontrem fora dos limites da maior região são considerados *outliers*. Cada dimensão vista isoladamente representa o *boxplot* para o atributo associado.

No diagrama de Chernoff, cada atributo é representado por uma ou mais caraterísticas de uma face, como altura e largura da cabeça, da boca, do cabelo, do nariz, das orelhas,

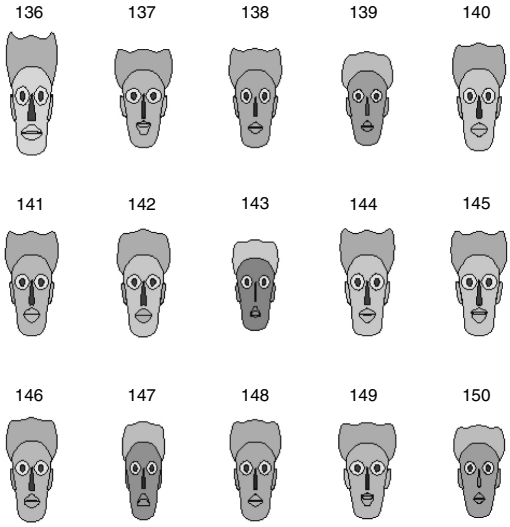


Figura 2.8 *Diagramas de Chernoff para 15 objetos do conjunto de dados* `iris`.

se a face apresenta um sorriso e o estilo do cabelo. O número de atributos representados é limitado pelo número de caraterísticas presentes na implementação do algoritmo que desenha as faces. Se o conjunto de dados tem menos atributos que o número de possíveis caraterísticas, um mesmo atributo pode ser representado por mais de uma caraterística da face. O diagrama de Chernoff para 15 exemplos do conjunto de dados `iris` pode ser visto na Figura 2.8. Nesta figura, o tamanho da sépala, por exemplo, é representado pelas caraterísticas altura da face, largura da boca, altura do cabelo e largura do nariz. Os outros atributos do conjunto de dados `iris` são representados nas figuras por outras caraterísticas da face.

Assim como o diagrama de Chernoff, o *star plot* desenha uma figura geométrica para cada objeto, normalmente um polígono. Cada linha do polígono corresponde a um dos atributos, e o tamanho da linha é proporcional ao valor do atributo. Um segmento de reta liga o centro do polígono a cada um de seus vértices. Um conjunto de *star plots* para os objetos do conjunto `iris` é ilustrado pela Figura 2.9. Quanto mais atributos são considerados, mais o polígono se assemelha a uma estrela. Quando dois ou mais atributos

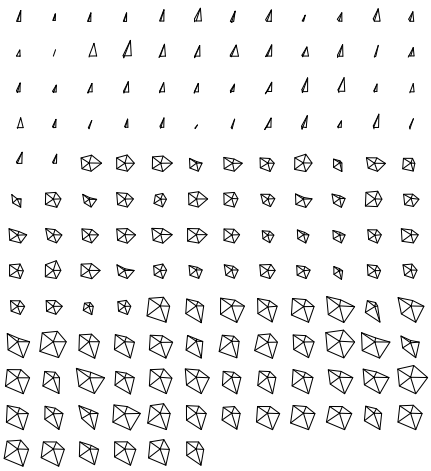


Figura 2.9 *Star plots para os 150 objetos do conjunto de dados* `iris`.

de um objeto têm valores semelhantes, apresentam coordenadas similares no diagrama, deformando o formato de estrela.

Um *heatmap* representa a relação entre os exemplos e as classes, associando um eixo para cada um. Para cada eixo é construída uma imagem gráfica referente ao agrupamento hierárquico (dendrograma) dos elementos do eixo, utilizando como entrada os elementos do outro eixo, como pode ser visto na Figura 2.10. Para facilitar a visualização, em vez do nome completo de cada atributo, foram utilizadas apenas as suas iniciais. Por exemplo, TS significa *Tamanho da Sépala*.

Nesta imagem, cada linha da imagem representa um objeto do conjunto de dados `iris` e cada coluna, um dos atributos desse conjunto. Elementos semelhantes são agrupados juntos, apresentando cores ou tonalidades semelhantes. No topo da imagem é apresentado um dendrograma das três classes do conjunto `iris` e no lado esquerdo um dendrograma para os 150 objetos. Como os elementos em cada eixo são agrupados na imagem, de acordo com a sua semelhança, os *heatmaps* permitem ver tendências nos seus valores. Em bioinformática, os *heatmaps* são muito utilizados para análise de dados de expressão genética.

2.3 Considerações Finais

Antes de aplicar os algoritmos de ECD a um conjunto de dados, é importante que os dados sejam analisados. Essa análise, que pode ser realizada por meio de técnicas estatísticas e de visualização, permite uma melhor compreensão da distribuição dos dados e pode suportar a escolha de formas para modelar o problema.

Neste capítulo, foram apresentados conceitos considerados importantes para analisar

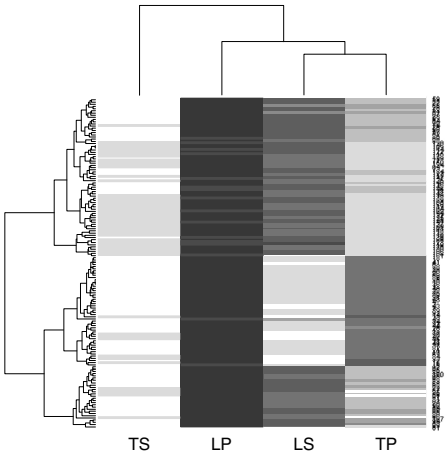


Figura 2.10 *Heatmap para atributos de entrada do conjunto de dados* `iris`.

os principais aspetos de um conjunto de dados. Após uma caraterização dos tipos de dados encontrados na maioria das aplicações de ECD, foram apresentadas medidas estatísticas frequentemente utilizadas para exploração de dados univariados e multivariados. Por fim, diversas técnicas de visualização foram descritas para dados com uma ou mais varáveis: *boxplots*, histogramas, gráficos de pizza, *scatter plots*, *bagplots*, faces de Chernoff, *star plots* e *heatmaps*. Note-se que existem outras técnicas de visualização, sendo que neste capítulo se introduziram as mais conhecidas e utilizadas.

Capítulo 3

Pré-processamento de Dados

Apesar de os algoritmos de ECD serem frequentemente adotados para extrair conhecimento de conjuntos de dados, o seu desempenho é geralmente afetado pelo estado dos dados. Diferentes conjuntos de dados podem apresentar diferentes caraterísticas, dimensões ou formatos. Por exemplo, conforme visto no capítulo anterior, os valores dos atributos de um conjunto de dados podem ser numéricos ou simbólicos. Podem ainda estar limpos ou conter ruído e imperfeições, com valores incorretos, inconsistentes, duplicados ou ausentes. Por outro lado, os atributos podem ser independentes ou estar relacionados. Além disso, os conjuntos de dados podem apresentar poucos ou muitos objetos que, por sua vez, podem ser caraterizados por um pequeno ou elevado número de atributos.

As técnicas de pré-processamento de dados são frequentemente utilizadas para melhorar a qualidade dos dados por via da eliminação ou minimização dos problemas citados. Esta melhoria pode facilitar a utilização de técnicas de ECD, permitir a construção de modelos mais fiéis à distribuição real dos dados reduzindo a sua complexidade computacional, facilitar e acelerar o ajuste dos parâmetros do modelo e a sua posterior utilização. Adicionalmente pode facilitar a interpretação dos padrões extraídos pelo modelo.

As técnicas de pré-processamento de dados são úteis não apenas devido à sua capacidade de minimizar ou eliminar problemas existentes num conjunto de dados, mas também porque podem melhorar a adequabilidade dos dados a um determinado algoritmo de ECD. Por exemplo, alguns algoritmos de ECD trabalham apenas com valores numéricos.

Neste capítulo serão apresentadas algumas operações de pré-processamento que podem ser realizadas nos conjuntos de dados antes da aplicação de algoritmos de ECD. Estas operações englobam o uso de técnicas de amostragem, tratamentos para dados desbalanceados, modificações para adequação dos tipos de atributo, limpeza dos dados, integração de dados, transformações dos dados e redução de dimensionalidade. Note-se que não existe uma ordem fixa para a aplicação das diferentes técnicas de pré-processamento.

Este capítulo está organizado de acordo com as diferentes tarefas de pré-processamento. A Secçao 3.1 apresenta os conceitos básicos da teoria da informação. Na Seção 3.2 é mostrado como atributos claramente irrelevantes podem ser identificados e extraídos do conjunto de dados. Na Seção 3.3 é discutida a agregação de dados de diferentes fontes. A seleção de um subconjunto dos dados originais através de amostragem de dados é abor-

dada na Seção 3.4. A Seção 3.5 trata do tema de conjuntos de dados desbalanceados. Técnicas para a limpeza de conjuntos de dados são introduzidas na Seção 3.6. Na Seção 3.7 são descritas operações para a transformação do tipo dos valores de um atributo. Os problemas causados pelo elevado número de atributos preditivos, bem como formas de reduzir a quantidade desses atributos são analisados na Seção 3.8. A Seção 3.9 remata o Capítulo com alguns comentários finais.

3.1 Teoria da Informação

A teoria da informação é baseada na estatística e na teoria da probabilidade. As medidas de informação mais importantes são a *entropia*, que mede o grau de aleatoriedade de uma variável aleatória, e a *informação mútua*, que mede a quantidade de informação partilhada por duas variáveis aleatórias. A escolha da base logarítmica nas seguintes fórmulas determina a unidade de entropia da informação que é usada. A unidade de informação mais comum é o *bit*, para a função do logaritmo base 2. Por convenção, uma expressão da forma $0 \times log_2(0)$ é considerada como sendo igual a zero.

A entropia H da variável aleatória discreta X mede a incerteza associada ao valor de X.

$$H(X) = -\sum_{x \in X} p(x) \times log(p(x)) \tag{3.1}$$

Considere uma variável aleatória discreta X cujo domínio é $\{x_1, x_2, ..., x_v\}$. Suponha que a probabilidade de observar cada valor é $p_1, p_2, ..., p_v$. A entropia de X é dada por: $H(X) = -\sum_i p_i \times \log_2 p_i$. A função entropia de uma variável aleatória que pode assumir v valores distintos, apresenta as seguintes propriedades:

1. $H(X) \in [0, \log_2(v)]$;
2. $H(X)$ tem um máximo igual a $\log_2(v)$ se $p_i = p_j, \forall i \neq j$;
3. $H(X)$ tem um mínimo igual a 0 se $\exists i\colon\ p_i = 1$.

A entropia condicional de X, dada a variável aleatória Y, é a média da entropia condicional sobre Y:

$$H(X|Y) = -\sum_{y \in Y} p(y) \sum_{x \in X} p(x|y) \log p(x|y) = -\sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(y)} \tag{3.2}$$

A entropia condicional tem as seguintes propriedades básicas:

1. $H(X|X) = 0$
2. $H(X|Y) = H(X,Y) - H(Y)$
3. $H(X|Y) = H(X)$ se X e Y forem variáveis independentes

A informação mútua entre duas variáveis aleatórias é dada pela expressão:

$$I(X,Y) = \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)\,p(y)} \tag{3.3}$$

Pode ser compreendida como a diminuição na incerteza de uma das variáveis, proporcionada pelo conhecimento da outra. A informação mútua é simétrica: $I(X,Y) = I(Y,X)$ e pode ser considerada uma estatística para avaliar a independência entre um par de variáveis.

3.2 Eliminação Manual de Atributos

Observando novamente a Tabela 2.1 no capítulo anterior, onde é apresentado o conjunto de dados `hospital`, pode ser facilmente percebido que nem todos os atributos apresentados são necessários para o diagnóstico clínico de um paciente. Não faz sentido, por exemplo, usar os valores dos atributos *Nome* e *Id.* (identificação do paciente) para o diagnóstico. Quando um atributo claramente não contribui para a estimativa do valor do atributo alvo, ele é considerado irrelevante.

O conjunto de atributos que formarão o conjunto de dados a ser analisado é geralmente definido de acordo com a experiência de especialistas no domínio do problema de decisão. Os especialistas podem decidir, por exemplo, que atributos associados à identificação do paciente, ao nome do paciente e ao estado de origem do paciente não são relevantes para seu diagnóstico clínico. A Tabela 3.1 mostra o conjunto de dados `hospital` após remover os atributos considerados irrelevantes.

Tabela 3.1 *Conjunto de dados sem atributos considerados irrelevantes*

Idade	Sexo	Peso	Manchas	Temp.	# Int.	Diagnóstico
28	M	79	Concentradas	38,0	2	Doente
18	F	67	Inexistentes	39,5	4	Doente
49	M	92	Espalhadas	38,0	2	Saudável
18	M	43	Inexistentes	38,5	8	Doente
21	F	52	Uniformes	37,6	1	Saudável
22	F	72	Inexistentes	38,0	3	Doente
19	F	87	Espalhadas	39,0	6	Doente
34	M	67	Uniformes	38,4	2	Saudável

Existem outras situações em que um atributo irrelevante pode ser facilmente detetado. Por exemplo, quando um atributo possui o mesmo valor para todos os objetos. Tal atributo não contém informação que ajude a distinguir os objetos. Por esse motivo, pode ser considerado irrelevante. Um atributo não precisa ter exatamente o mesmo valor para todos os objetos para ser irrelevante. Neste capítulo iremos ver algumas das técnicas de seleção de atributos, utilizadas para eliminar atributos irrelevantes.

3.3 Integração de Dados

Conforme mencionado anteriormente, quando os dados a serem utilizados numa aplicação de ECD estão distribuídos em diferentes bases de dados, os dados têm de ser integrados antes do início do uso da técnica de ECD. Na integração, é necessário identificar quais são os objetos que estão presentes nos diferentes conjuntos a serem combinados. Este problema é conhecido como o problema de identificação de entidade. Esta identificação é realizada por meio da procura por atributos comuns nos conjuntos a serem combinados. Por exemplo, os conjuntos de dados médicos podem ter um atributo que identifica o paciente. Desta forma, os objetos dos diferentes conjuntos que possuem o mesmo valor para o atributo que identifica o paciente são combinados num único objeto do conjunto integrado. É fácil ver que o(s) atributo(s) utilizado(s) para combinação deve(m) ter um valor único para cada objeto.

Alguns aspetos podem dificultar a integração. Por exemplo, atributos correspondentes podem ter nomes diferentes em diferentes bases de dados. Além disso, os dados a serem integrados podem ter sido atualizados em momentos diferentes. Para minimizar esses problemas, é comum o uso de metadados em bases de dados. Os metadados são dados sobre dados que, ao descrever as principais caraterísticas dos dados, podem ser utilizados para evitar erros no processo de integração. O processo de integração origina um depósito ou repositório de dados (*data warehouse*), que funciona como uma base de dados centralizada.

Com ou sem integração, é cada vez mais frequente o uso de técnicas de ECD em grandes conjuntos de dados, que muitas vezes crescem com o tempo. Um conjunto de dados é considerado 'grande' quer porque contem um número elevado de objetos, quer porque cada objeto é descrito por um número elevado de atributos. Em geral, o desempenho de um algoritmo de aprendizagem melhora com o aumento do número de objetos, e diminui com o crescimento do número de atributos.

A existência de um conjunto de dados grande, tanto em termos de número de objetos como de atributos, não implica que um algoritmo de ECD o deva utilizar todo. Muitas vezes é mais eficiente usar apenas parte do conjunto original. É sabido que um número elevado de atributos pode comprometer o desempenho do algoritmo de aprendizagem. Este é um problema conhecido como a *maldição da dimensionalidade*, que será discutido no Capítulo 4. As técnicas para lidar com um grande número de atributos serão abordadas na Seção 3.8. Com relação à quantidade de objetos, problemas podem ocorrer por causa de saturação de memória e aumento do tempo computacional necessário para ajustar os parâmetros do modelo. Para minimizar esses problemas, podem ser utilizadas técnicas de amostragem, que serão apresentadas a seguir.

3.4 Amostragem de Dados

Os algoritmos de ECD podem ter dificuldades em lidar com um número elevado de objetos. Este problema é observado em algoritmos de ECD baseados em instâncias, como

k-vizinhos mais próximos (*k*-NN, do inglês *k-nearest neighbours* (Fix, 1951)) (descritos no Capítulo 4), que podem apresentar problemas de saturação de memória quando um conjunto de dados contém um elevado número de exemplos.

Associado ao número de objetos num conjunto de dados, existe um equilíbrio entre eficiência computacional e taxa de acerto (taxa de predições corretas). Quanto maior a quantidade de dados utilizada, maior tende a ser a taxa de acerto do modelo e menor a eficiência computacional do processo indutivo, uma vez que um número muito elevado de objetos pode aumentar significativamente o tempo de processamento. Para se obter um bom compromisso entre eficiência e taxa de acerto, geralmente, trabalha-se com uma amostra ou um subconjunto dos dados. Muitas vezes, o recurso a uma amostra permite alcançar o mesmo desempenho que se obteria utilizando o conjunto de dados completo, porém com um custo computacional muito menor.

Note-se, porém, que uma amostra pequena pode não representar bem o problema que se pretende modelar. A amostra deve ser representativa do conjunto de dados original. Se as amostras não forem representativas, diferentes amostras de uma mesma população podem gerar modelos diferentes, uma vez que caraterísticas importantes do problema ou da distribuição que gerou os dados poderem não estar presentes. Por outro lado, se a amostra for muito grande, são reduzidas as vantagens de utilizar amostragem. Assim, deve-se novamente procurar um compromisso entre a eficiência e o desempenho.

O ideal é que a amostra não seja grande, mas que os dados que a constituem obedeçam à mesma distribuição estatística que gerou o conjunto de dados original. Desta forma, seria capaz de fornecer uma estimativa da informação contida na população original, permitindo tirar conclusões de um todo a partir de uma parte. Por exemplo, a média dos valores para cada atributo dos dados originais deve ser semelhante à média observada nos valores desses atributos na amostra gerada. Embora não seja possível garantir que isso aconteça, existem técnicas de amostragem estatística que aumentam a probabilidade de isso ocorrer.

Existem basicamente três abordagens para amostragem:

- Amostragem aleatória simples;
- Amostragem estratificada;
- Amostragem progressiva.

A amostragem aleatória simples possui duas variações: amostragem simples sem reposição de exemplos, em que os exemplos são extraídos do conjunto original para a amostra a ser utilizada, com a condição de que cada exemplo apenas pode ser selecionado uma vez; e amostragem simples com reposição, quando uma cópia dos exemplos selecionados é mantida no conjunto de dados original. A amostragem com reposição é mais fácil de analisar, pois a probabilidade de escolher qualquer objeto se mantém constante. No entanto, as duas formas são semelhantes quando o tamanho da amostra é muito menor que o tamanho do conjunto original.

A amostragem estratificada é usada quando as classes apresentam propriedades diferentes, por exemplo, números de objetos bastante diferentes. Em problemas de classificação, um aspeto que deve tomar em consideração na amostragem diz respeito à distribuição

dos dados para as diferentes classes. A existência de classes com um número de exemplos significativamente maior que as demais pode levar à indução de classificadores tendenciosos para as classes maioritárias. Nesse caso, o conjunto de dados é dito desbalanceado. Este problema será retomado na Seção 3.5. Por outro lado, pode ocorrer que algumas classes sejam mais difíceis de classificar do que outras e isso possa ser minimizado no processo de amostragem. Esta abordagem também possui variações. A mais simples consiste em manter o mesmo número de objetos para cada classe. Outra opção é manter o número de objetos em cada classe proporcional ao número de objetos da classe no conjunto original.

A terceira alternativa, a amostragem progressiva, começa com uma amostra pequena e aumenta progressivamente o tamanho da amostra extraída, enquanto a taxa de acerto preditiva continuar a melhorar. Como resultado, é possível definir a menor quantidade de dados necessária, reduzindo ou eliminando a perda de taxa de acerto. O tamanho pode ser confirmado com outras amostras de tamanho semelhante. Geralmente, esta abordagem fornece uma boa estimativa para o tamanho da amostra.

O especialista do domínio pode também decidir que um subconjunto dos objetos deve ser utilizado nas suas análises. Por exemplo, numa análise de pacientes de um hospital, podem ser utilizados apenas os objetos correspondentes a pacientes do sexo feminino.

3.5 Dados Desbalanceados

O problema dos dados desbalanceados é um tópico da área de classificação de dados. Em vários conjuntos de dados reais, o número de objetos varia para as diferentes classes. Este problema é comum em aplicações em que dados de um subconjunto das classes aparecem com uma frequência maior que os dados das restantes classes. Recorrendo ao exemplo do conjunto de dados de pacientes de um hospital, suponha que 80% dos pacientes que se deslocam a esse hospital apresentam sintomas de uma dada doença. O conjunto de dados apresentará então 20% dos seus objetos relacionados com pacientes saudáveis e 80% associados a pacientes com a doença. A classe com pacientes doentes seria então a classe maioritária, enquanto que a classe de pacientes saudáveis, seria a classe minoritária. Outro exemplo seria um conjunto de dados de clientes de um banco, em que cada cliente é rotulado como tendo ou não ficado com o saldo da sua conta negativo nos últimos 90 dias. Se a percentagem de clientes que ficou com saldo negativo nesse período for de 5%, a classe maioritária terá 95% dos dados.

Para ser aceitável, a taxa de acerto preditiva de um classificador para um conjunto de dados desbalanceados deve ser maior que a taxa de acerto obtida quando se atribui a classe maioritária a todo e qualquer objeto. Vários algoritmos de ECD pioram o seu desempenho na presença de dados desbalanceados. Quando treinados com dados desbalanceados, estes algoritmos tendem a favorecer a classificação de novos dados na classe maioritária. Se for possível gerar novos dados através do mesmo processo que gerou o conjunto atual, o conjunto de dados pode ser naturalmente balanceado. No entanto, na maioria das aplicações práticas isso não é possível. Nestes casos, podem ser utilizadas técnicas que procuram ba-

lancear artificialmente o conjunto de dados. As principais técnicas propostas na literatura seguem uma destas alternativas:

- Redefinir o tamanho do conjunto de dados;
- Utilizar diferentes custos de classificação para as diferentes classes;
- Induzir um modelo para uma classe.

No primeiro caso, tanto pode ocorrer o acréscimo de objetos à classe minoritária como a eliminação de objetos da classe maioritária. No caso de se acrescentarem novos objetos, existe o risco dos objetos acrescentados representarem situações que nunca ocorrerão, induzindo um modelo inadequado para os dados. Além disso, pode ocorrer um problema conhecido como *overfitting*, em que o modelo é superajustado aos dados de treino. Por outro lado, quando se eliminam objetos da classe maioritária, é possível que dados de grande importância para a indução do modelo correto sejam perdidos. Isto pode levar ao problema de *underfitting*, em que o modelo induzido não se ajusta aos dados de treino. Os conceitos de *overfitting* e *underfitting* serão explicados mais adiante no livro.

A utilização de custos de classificação diferentes para as classes maioritária e minoritária tem como dificuldade a definição desses custos. Por exemplo, se o número de exemplos da classe maioritária for o dobro do número de exemplos da classe minoritária, um erro de classificação para um exemplo da classe minoritária pode equivaler à ocorrência de dois erros de classificação para um exemplo da classe maioritária. Entretanto, a definição dos diferentes custos geralmente não é tão direta. Outro problema desta abordagem é a dificuldade de incorporar a consideração de diferentes custos em alguns algoritmos de ECD. Além disso, esta abordagem pode apresentar um desempenho baixo quando uma boa parte dos objetos da classe maioritária apresenta um elevado grau de semelhança. A existência de um grande número de objetos semelhantes na classe maioritária pode aproximar a distribuição de exemplos relevantes para o treino, à distribuição presente na classe minoritária, fazendo com que os custos diferentes privilegiem a classificação na classe minoritária.

A última alternativa inclui as técnicas de classificação com apenas uma classe, em que a classe minoritária ou a classe maioritária (ou ambas as classes) são aprendidas separadamente. Neste caso, podem ser utilizados algoritmos de classificação para uma classe apenas (Manevitz et al., 2001). Estes algoritmos são treinados utilizando apenas exemplos da classe positiva. A classe positiva pode ser, por exemplo, a classe minoritária.

3.6 Limpeza de Dados

Os conjuntos de dados também podem apresentar dificuldades relacionadas com a qualidade dos dados. Exemplos frequentes dessas dificuldades incluem dados ruidosos (que possuem erros ou valores que são diferentes do esperado), inconsistentes (que não combinam ou contradizem valores de outros atributos do mesmo objeto), redundantes (quando dois ou mais objetos têm os mesmos valores para todos os atributos ou dois ou mais atributos têm os mesmos valores para dois ou mais objetos) ou incompletos (ausência de valores,

para alguns dos atributos, em parte dos dados). Dados inconsistentes, redundantes ou com valores ausentes são de fáceis de detetar. A principal dificuldade consiste na deteção de dados ruidosos.

Essas deficiências nos dados podem ser causadas por problemas nos equipamentos que realizam a recolha, a transmissão e o armazenamento dos dados, ou ainda problemas no preenchimento dos dados por seres humanos.

Algumas técnicas de ECD conseguem lidar bem com algumas dessas imperfeições nos dados. Por exemplo, as máquinas de vetores de suporte lidam com dados que apresentam ruído e um número elevado de atributos preditivos. Outras técnicas apresentam dificuldades ou não conseguem lidar com dados que apresentem algumas das deficiências mencionadas anteriormente. Por exemplo, o algoritmo de agrupamento de dados *k*-médias, não é robusto em dados com ruido.

Mesmo que a técnica seja robusta o suficiente para lidar com tais imperfeições, podem reduzir a qualidade da análise. A presença destas deficiências num conjunto de dados pode resultar em estatísticas e análises incorretas. Portanto, todas as técnicas beneficiam da melhoria na qualidade dos dados. Cada um destes problemas será detalhado a seguir.

3.6.1 Dados Incompletos

Como já foi previamente mencionado, um dos problemas que pode ser encontrado em conjuntos de dados é a ausência de valores para alguns atributos de alguns objetos. Na Tabela 3.2 é ilustrado um exemplo de um conjunto de dados em que três dos objetos possuem três, dois e um atributos com valores ausentes, respetivamente. Na tabela, os atributos com valores ausentes são assinalados pelo valor "—".

Tabela 3.2 *Conjunto de dados com atributos com valores ausentes*

Idade	Sexo	Peso	Manchas	Temp.	# Int.	Diagnóstico
—	M	79	—	38,0	—	Doente
18	F	67	Inexistentes	39,5	4	Doente
49	M	92	Espalhadas	38,0	2	Saudável
18	—	43	Inexistentes	38,5	8	Doente
21	F	52	Uniformes	37,6	1	Saudável
22	F	72	Inexistentes	38,0	3	Doente
—	F	87	Espalhadas	39,0	6	Doente
34	M	67	Uniformes	38,4	2	Saudável

A ausência de valores em alguns dos atributos da Tabela 3.2 pode ter diferentes causas, nomeadamente:

- O atributo não foi considerado importante quando os primeiros dados foram recolhidos. Considere, por exemplo, que um dos atributos é o *email* do paciente, que não era comum na década de 1990.

- Desconhecimento do valor do atributo no momento do preenchimento dos valores do objeto. Uma situação possível seria não saber o tipo sanguíneo de um paciente quando foi efetuado o seu cadastro.
- Distração na hora do preenchimento.
- Falta de necessidade ou obrigação de apresentar um valor para o(s) atributo(s), para alguns objetos. Por exemplo, se houver um atributo renda, alguns pacientes podem não querer preenchê-lo.
- Inexistência de um valor para o atributo em alguns objetos. Esta situação pode ocorrer se um dos atributos especificar o número de partos e o paciente em questão for do sexo masculino.
- Problema com o equipamento ou com o processo utilizado para recolha, transmissão e armazenamento de dados.

Algumas técnicas de ECD podem gerar erro de execução quando um ou mais atributos do conjunto de treino não apresentam valor. Várias alternativas têm sido propostas para lidar com esses atributos. As alternativas mais utilizadas são:

- Eliminar os objetos com valores ausentes. Esta alternativa é geralmente empregue quando um dos atributos com valores ausentes num objeto é o que indica a sua classe. Esta alternativa não é indicada quando poucos atributos do objeto possuem valores ausentes, quando o número de atributos com valores ausentes varia muito entre os objetos com esse problema ou quando o número de objetos que restarem for reduzido.
- Definir e preencher manualmente valores para os atributos com valores ausentes. Esta alternativa não é factível quando o número de objetos ou atributos com valores ausentes for muito grande.
- Utilizar algum método ou heurística para definir automaticamente valores para atributos com valores ausentes. Esta é a alternativa mais utilizada. Diferentes abordagens podem ser utilizadas, como será discutido em seguida.
- Aplicar algoritmos de ECD que lidam internamente com valores ausentes. Este é o caso, por exemplo, de alguns algoritmos indutores de árvores de decisão (Seção 6.1).

A definição automática de valores para completar os valores ausentes tem seguido três abordagens diferentes:

- Criar um novo valor para o atributo que indique que o atributo possuía um valor desconhecido. Esse valor pode ser comum a todos os atributos ou um valor diferente para cada atributo. O problema desta alternativa é que o algoritmo indutor pode assumir que o valor desconhecido representa um conceito importante.

- Utilizar a média, ou a moda (no caso de valores simbólicos) ou a mediana dos valores conhecidos para esse atributo. Esta medida pode ser calculada utilizando todos os objetos ou apenas os objetos da mesma classe do objeto com o atributo a ser preenchido. Outra variação desta abordagem utiliza o valor mais frequente nos k objetos mais semelhantes àquele cujo valor ausente necessita de ser estimado. Se os objetos tiverem uma relação temporal, a medida pode ser calculada utilizando os objetos associados ao instante imediatamente anterior e posterior ao objeto modificado.
- Utilizar um indutor para estimar o valor do atributo. Neste contexto, o atributo alvo é o atributo com valores desconhecidos e os demais atributos seriam os atributos de entrada. A vantagem deste método é justamente a utilização de informação presente nos restantes atributos para inferir o valor do atributo ausente. Esta abordagem é a mais popular.

Se os atributos com valores ausentes forem substituídos pelo valor da média (valores numéricos) ou da moda (valores simbólicos) dos valores de cada um desses atributos, a Tabela 3.2 seria substituída pela Tabela 3.3. A imputação de valores ausentes pela média pode gerar inconsistências como, por exemplo, um paciente de 2 anos de idade com peso igual a 60 quilos.

Tabela 3.3 *Conjunto de dados com substituição dos valores ausentes*

Idade	Sexo	Peso	Manchas	Temp.	# Int.	Diagnóstico
27	M	79	**Inexistentes**	38,0	**4**	Doente
18	F	67	Inexistentes	39,5	4	Doente
49	M	92	Espalhadas	38,0	2	Saudável
18	**F**	43	Inexistentes	38,5	8	Doente
21	F	52	Uniformes	37,6	1	Saudável
22	F	72	Inexistentes	38,0	3	Doente
27	F	87	Espalhadas	39,0	6	Doente
34	M	67	Uniformes	38,4	2	Saudável

3.6.2 Dados Inconsistentes

Dados inconsistentes são aqueles que possuem valores contraditórios nos seus atributos. Este tipo de inconsistência pode verificar-se entre valores de atributos de entrada (por exemplo, valor 120 para o atributo *Peso* e o valor 3 para o atributo *Idade*) ou entre todos os valores dos atributos de entrada e o valor do atributo de saída (por exemplo, dois pacientes com os mesmos valores para os atributos de entrada e diagnósticos diferentes, um saudável e o outro doente). Inconsistências podem ser identificadas quando relações conhecidas entre os atributos são violadas. Por exemplo, quando se sabe que os valores de um atributo variam de forma inversamente proporcional em relação aos valores de um outro atributo.

Dados inconsistentes podem resultar do processo de integração de dados de fontes ou tabelas diferentes, ou da presença de ruído nos dados.

Dados inconsistentes são muitas vezes produzidos no processo de integração de dados. Por exemplo, diferentes conjuntos de dados podem usar escalas diferentes para a mesma medida (metros e centímetros) ou codificar de forma diferente um atributo associado ao tamanho.

Na Tabela 3.4 é ilustrado um exemplo de conjunto de dados em que se verificam inconsistências. Essa inconsistência pode ser observada nos dois objetos destacados, que apresentam valores de entrada iguais para valores de saída diferentes.

Tabela 3.4 *Conjunto de dados com objetos inconsistentes*

Idade	Sexo	Peso	Manchas	Temp.	# Int.	Diagnóstico
28	M	79	Concentradas	38,0	2	Doente
18	F	67	Inexistentes	39,5	4	Doente
49	M	92	Espalhadas	38,0	2	Saudável
18	M	43	Inexistentes	38,5	8	Doente
21	F	52	Uniformes	37,6	1	Saudável
22	F	72	Inexistentes	38,0	3	**Doente**
19	F	87	Espalhadas	39,0	6	Doente
22	F	72	Inexistentes	38,0	3	**Saudável**

Existem formas de prevenir a ocorrência de inconsistências. Algoritmos simples podem verificar automaticamente se relacionamentos existentes entre atributos são violados. Quando o conjunto de dados não é muito grande, dados inconsistentes podem ser removidos manualmente.

3.6.3 Dados Redundantes

Um conjunto de dados pode conter quer objetos, quer atributos redundantes. Um objeto é redundante quando é muito semelhante a um outro objeto do mesmo conjunto de dados, ou seja, os seus atributos possuem valores muito semelhantes aos atributos de, pelo menos, outro objeto. No caso extremo, apresentam os mesmos valores para cada um dos atributos. Por outro lado, um atributo é redundante quando o seu valor pode ser deduzido a partir do valor de um ou mais atributos. No caso extremo, possui o mesmo valor que um outro atributo para cada um dos objetos do conjunto de dados.

Para ilustrar a presença de objetos redundantes, observe-se a Tabela 3.5, que apresenta redundância entre três objetos. Nesta tabela, o segundo e o quarto pacientes têm os mesmos valores para todos os atributos, e por isso são considerados objetos redundantes, conforme destacado.

Objetos redundantes num conjunto de dados participam mais de uma vez no processo de ajuste de parâmetros de um modelo contribuindo, assim, mais do que os outros objetos na definição do modelo final. Isto pode dar ao modelo a falsa impressão de que esse

Tabela 3.5 *Conjunto de dados com objetos redundantes*

Idade	Sexo	Peso	Manchas	Temp.	# Int.	Diagnóstico
28	M	79	Concentradas	38,0	2	Doente
18	F	67	Inexistentes	39,5	4	Doente
49	M	92	Espalhadas	38,0	2	Saudável
18	F	67	Inexistentes	39,5	4	Doente
18	M	43	Inexistentes	38,5	8	Doente
21	F	52	Uniformes	37,6	1	Saudável
22	F	72	Inexistentes	38,0	3	Doente
19	F	87	Espalhadas	39,0	6	Doente
34	M	67	Uniformes	38,4	2	Saudável

perfil de objeto é mais importante que os demais. Geralmente, é desejável a eliminação de redundâncias, o que pode ser feita em dois passos:

- Identificação de objetos redundantes;
- Eliminação das redundâncias encontradas.

A eliminação da redundância pode ocorrer pela eliminação dos objetos semelhantes a um dado objeto ou pela combinação dos valores dos atributos dos objetos semelhantes. A eliminação de redundância é, geralmente, efetuada no final do processo de limpeza.

Se a redundância não for eliminada, o algoritmo de ECD utilizado pode atribuir ao objeto repetido uma importância maior que aos demais objetos. Por exemplo, um objeto *A* com duas cópias adicionais num conjunto de dados é considerado pelo algoritmo de indução três vezes mais importante que um objeto *B* que não está replicado. Se o objeto *A* "puxar" a fronteira de decisão para o lado esquerdo e o objeto *B* para o lado direito, o movimento da fronteira para o lado esquerdo será três vezes maior do que para o lado direito. Note-se que algumas técnicas de ECD, como o *boosting*, utilizam esse artifício para duplicar a quantidade de exemplos difíceis de ser classificados.

Conforme mencionado no início da seção, além dos objetos, também os atributos podem apresentar redundância. Um atributo é considerado redundante se o seu valor puder ser estimado a partir de, pelo menos, um dos demais atributos. Isto ocorre quando dois ou mais atributos têm a mesma informação preditiva. No caso extremo, dois atributos podem compartilhar o mesmo valor entre si para cada um dos objetos de um conjunto de dados. Um exemplo simples de redundância de atributos é a presença de um atributo *Idade* e de um atributo *Data de nascimento* num conjunto de dados, pois é fácil definir o valor do atributo *Idade* usando o valor do atributo *Data de nascimento*. Outro exemplo com mais de dois atributos seria ter um atributo *Quantidade de vendas*, um atributo *Valor por venda* e um atributo *Venda total*. Neste caso, o valor do atributo *Venda total* pode ser facilmente definido a partir do valor dos dois outros atributos. Um atributo redundante pode supervalorizar um dado aspeto dos dados, por estar presente mais de uma vez, ou tornar mais lento o processo de indução, devido ao maior número de atributos que têm de ser analisados pelo

algoritmo de ECD. Assim, o desempenho de um algoritmo de ECD geralmente melhora com a eliminação de atributos redundantes. Muitas vezes a redundância entre atributos não é tão clara. Atributos redundantes são geralmente eliminados por técnicas de seleção de atributos, como será visto mais adiante.

A redundância de um atributo está relacionada com a correlação com um ou mais dos atributos do conjunto de dados. Dois ou mais atributos estão correlacionados quando apresentam um perfil de variação semelhante para os diferentes objetos. Quanto mais correlacionados os atributos, maior o grau de redundância. Se a correlação ocorrer entre um atributo de entrada e o atributo rótulo, então, esse atributo de entrada terá uma grande influência na predição do valor do atributo rótulo.

Na Tabela 3.6 é ilustrado um conjunto de dados em que um dos atributos é claramente redundante. Esta tabela adiciona à Tabela 2.1 o atributo número de visitas (*#Vis.*), que indica quantas vezes um dado paciente esteve no hospital. O atributo redundante, *#Vis.*, está destacado a negrito. Nestes casos, a redundância é removida mantendo um dos atributos e eliminando o outro.

Tabela 3.6 *Conjunto de dados com atributos redundantes*

Idade	Sexo	Peso	Manchas	Temp.	# Int.	# Vis.	Diagnóstico
28	M	79	Concentradas	38,0	2	**2**	Doente
18	F	67	Inexistentes	39,5	4	**4**	Doente
49	M	92	Espalhadas	38,0	2	**2**	Saudável
18	M	43	Inexistentes	38,5	8	**8**	Doente
21	F	52	Uniformes	37,6	1	**1**	Saudável
22	F	72	Inexistentes	38,0	3	**3**	Doente
19	F	87	Espalhadas	39,0	6	**6**	Doente
34	M	67	Uniformes	38,4	2	**2**	Saudável

3.6.4 Dados com Ruído

Dados com ruído são dados que contêm objetos que, aparentemente, não pertencem à distribuição que gerou os dados analisados. Ruído pode ser definido como uma variância ou erro aleatório no valor gerado ou medido para um atributo (Han e Kamber, 2000). Dados inconsistentes podem resultar da presença de ruído.

Podem ser várias as causas da presença de ruído, as quais foram discutidas anteriormente na Secção 3.6. Dados com ruído podem levar a um superajuste do modelo utilizado, pois o algoritmo que induz o modelo, pode se ajustar às especificidades relacionadas com o ruído, em vez da distribuição que gerou os dados. Por outro lado, a eliminação de dados ruidosos pode conduzir à perda de informação importante. A eliminação desses dados pode fazer com que algumas regiões do espaço de atributos não sejam consideradas no processo de indução de hipóteses.

É importante observar que não é possível garantir que o valor de um atributo é ou

não afetado pela presença de ruído, mas apenas ter uma indicação ou indício de que um dado valor para um atributo pode ter sido gerado com ruído. Um indicador da possível presença de ruído é a existência de *outliers*, que são valores que estão além dos limites aceitáveis, ou são muito diferentes dos demais valores observados para o mesmo atributo, representando, por exemplo, exceções raramente vistas. Na Tabela 3.7, o valor do atributo *peso* do segundo objeto é um *outlier*.

Tabela 3.7 *Conjunto de dados com com ruído*

Idade	Sexo	Peso	Manchas	Temp.	# Int.	Diagnóstico
28	M	79	Concentradas	38,0	2	Doente
18	F	**300**	Inexistentes	39,5	4	Doente
49	M	92	Espalhadas	38,0	2	Saudável
18	M	43	Inexistentes	38,5	8	Doente
21	F	52	Uniformes	37,6	1	Saudável
22	F	72	Inexistentes	38,0	3	Doente
19	F	87	Espalhadas	39,0	6	Doente
34	M	67	Uniformes	38,4	2	Saudável

Existem diversas técnicas de pré-processamento que podem ser aplicadas na deteção e remoção de ruído. Em Estatística, este problema é normalmente solucionado por meio de técnicas baseadas em distribuições conhecidas, nas quais o ruído é identificado como observações que diferem de uma distribuição utilizada na modelação dos dados (Barnett e Lewis, 1994). O maior problema desta abordagem está em assumir que a distribuição dos dados é conhecida *a priori*, o que não reflete a realidade em grande parte das aplicações práticas.

Uma segunda categoria de técnicas aplicadas em Estatística utiliza o conceito de profundidade na procura de ruído (Barnett e Lewis, 1994; Nuts e Rousseeuw, 1996). Estas técnicas organizam os dados em camadas. O ruído é identificado como objetos pertencentes a níveis superficiais. Este tipo de estratégia tem custos computacionais, principalmente para dados de grande dimensão.

Outras técnicas podem ser utilizadas para reduzir o ruído num atributo. De forma resumida, podem ser reunidas em cinco grupos:

- Técnicas de intervalo: Estas técnicas suavizam o valor de um atributo da seguinte forma. Primeiro, os valores de um atributo são ordenados. Em seguida, esses valores são divididos em intervalos, cada uma com o mesmo número de valores. Os valores num intervalo são substituídos, por exemplo, pela média ou mediana dos valores presentes no intervalo.
- Técnicas baseadas em agrupamento dos dados: Estas técnicas podem ser utilizadas tanto para os objetos como para os atributos. No caso dos atributos, os valores dos atributos são agrupados por uma técnica de agrupamento. Valores de atributos que não formem um grupo com outros valores são considerados ruído ou *outliers*.

O mesmo se aplica a objetos que forem colocados num grupo no qual os demais objetos pertencem a outra classe.

- Técnicas baseadas em distância: A presença de ruído num ou mais atributos de um objeto frequentemente provoca o afastamento desse objeto em relação aos restantes objetos da sua classe. As técnicas baseadas em distância verificam a que classe pertencem os objetos mais próximos de cada objeto **x**. Se esses objetos mais próximos pertencem a outra classe, a probabilidade de o objeto **x** apresentar ruído é elevada.
- Técnicas baseadas em regressão ou classificação: As técnicas baseadas em regressão utilizam uma função de regressão para, dado um valor com ruído, estimar o seu valor verdadeiro. Se o valor a ser estimado for simbólico, são utilizadas técnicas de classificação.

3.6.5 Deteção de *Outliers*

Numa das definições de *outliers* globalmente aceites, Hawkins (1980) afirma que um *outlier* é um objeto que se desvia significativamente dos outros objetos, como se fosse gerado por um mecanismo diferente. *Outliers* são também conhecidos como valores aberrantes, valores anormais ou valores extremos. *Outliers* são pontos que devem ser identificados e eventualmente removidos nas análises posteriores. Deteção de anomalias, de avarias, fraudes, intrusão, etc são tarefas que usualmente recorrem à identificação de *outliers*.

Dependendo da natureza do *outlier*, Aggarwal (2013) classificou-os como:

- *Outliers* pontuais: o tipo mais simples de *outlier* é uma observação que se afasta das outras observações. Pode ser um erro de medição ou um comportamento ou característica anormal do objeto. Por exemplo, uma grande quantidade de água consumida num mês numa casa pode sugerir um tubo quebrado ou outro problema na infra-estrutura.
- *Outliers* contextuais: por vezes, os valores anormais não são óbvios devido ao contexto em que aparecem. Por exemplo, se observarmos um uso muito baixo da eletricidade numa casa familiar em agosto, não pensamos nisso como um *outlier*: além de ser um consumo menor normal no verão, a casa provavelmente estava vazia por algumas semanas, enquanto a família ia de férias. Mas se acontecesse em janeiro, consideraríamos uma observação anormal.
- *Outliers* coletivos: *outliers* podem ser uma sequência de valores. Por exemplo, vamos considerar a distância de viagem diária de uma pessoa e assumimos que esse indivíduo viaja mais tempo nos dias úteis do que nos fins de semana. Se, durante sete dias consecutivos, essa pessoa percorrer distância curtas e depois retornar à distância usual, podemos assumir que, por algum motivo, essa pessoa não foi trabalhar nessa semana. Portanto, os valores de distância não foram considerados *outliers*, pois eram normais nos fins de semana, mas o fato de aparecerem por sete observações consecutivas é considerado anormal.

Para identificar *outliers* univariados, os métodos estatísticos estão entre os mais simples. Assumindo uma distribuição gaussiana e aprendendo os parâmetros a partir dos dados, os métodos paramétricos identificam os pontos com baixa probabilidade como *outliers*. Um dos métodos utilizados para detetar esses valores abertos é o método *boxplot*[1], introduzido por Tukey (1977). Com base no primeiro quartil ($Q1$), no terceiro quartil ($Q3$) e na faixa interquartil ($IQR = Q3 - Q1$) dos dados, determina que o intervalo $[Q1 - 1.5 * IQR, Q3 + 1.5 * IQR]$ contém 99,3% dos dados. Portanto, pontos fora desse intervalo são considerados como valores *outliers* moderados, e pontos fora do intervalo $[Q1 - 3 * IQR, Q3 + 3 * IQR]$ são considerado *outliers* extremos.

Em relação aos valores abertos multi-variados, um dos métodos mais populares para a deteção não supervisionada é a abordagem baseada na densidade conhecida por *Local Outlier Factor - LOF* (Breunig et al., 2000; Aggarwal, 2013). LOF é uma quantificação do grau de outlier dos pontos. O LOF baseia-se no conceito de densidade local, onde a localidade é dada por k vizinhos mais próximos, cuja distância é usada para estimar a densidade. Ao comparar a densidade local de um objeto com as densidades locais de seus vizinhos, pode-se identificar regiões de densidade similar. Portanto, *outliers* são pontos incluídos em regiões com uma densidade local substancialmente inferior à da sua vizinhança. A densidade é calculada como o inverso da distancia média do ponto aos k vizinhos mais próximos:

$$densidade(x,k) = \left(\frac{\sum_{y \in N(x,k)} distancia(x,y)}{|N(x,k)|} \right)^{-1} \tag{3.4}$$

O algoritmo 3.1 apresenta a versão básica do algorimo LOF.

Algoritmo 3.1 *Local Outlier Factor - LOF*

Entrada: Um conjunto de dados: $\mathbf{D} = \{\mathbf{x}_i, i = 1, \ldots, n\}$
Numero de vizinhos a considerar: k

1 **para cada** $\mathbf{x} \in D$ **faça**
2 Determina $N(\mathbf{x},k)$, os k vizinhos de $\mathbf{x}$;
3 Determina $densidade(x,k)$: a densidade de $\mathbf{x}$ (Equation 3.4) usando a vizinhança $N(\mathbf{x},k)$.
4 **fim**
5 **para cada** $\mathbf{x} \in D$ **faça**
6 Calcula o outlier $score(x,k) = \frac{densidade(x,k)}{\sum_{y \in N(x,k)} (densidade(y,k)/|N(x,k)|)}$
7 **fim**

[1] Descrito na página 28.

3.7 Transformação de Dados

Algumas técnicas de ECD estão limitadas à manipulação de valores de determinado tipo, por exemplo, apenas valores numéricos ou apenas valores simbólicos. Adicionalmente, o desempenho de algumas técnicas é influenciado pelo intervalo de variação dos valores numéricos. Esta secção divide as diferentes técnicas para abordar esse problema em três partes. A primeira parte descreve técnicas que podem ser utilizadas para converter valores simbólicos em valores numéricos. Podem ser utilizadas técnicas diferentes, dependente se os valores simbólicos são nominais ou ordinais. A segunda parte apresenta técnicas para converter valores numéricos em valores simbólicos. Finalmente, a terceira parte ilustra os casos em que a conversão não altera o tipo do atributo. Estas técnicas são utilizadas em atributos numéricos, e procedem á transformação do atributo, por exemplo, mudança de escala ou de intervalo de valores.

3.7.1 Conversão Simbólico-Numérico

Técnicas como redes neuronais artificiais e máquinas de suporte vetorial, e vários algoritmos de agrupamento, lidam apenas com dados numéricos. Assim, quando o conjunto de dados utilizado por estas técnicas apresenta atributos simbólicos, os valores destes atributos devem ser convertidos para valores numéricos.

Quando o atributo é do tipo nominal e assume apenas dois valores, isto é, se os valores denotam a presença ou ausência de uma caraterística ou se apresentam uma relação de ordem, um dígito binário é suficiente. No primeiro caso, o valor 0 indica a ausência e o valor 1, a presença da caraterística. No segundo caso, o menor valor ordinal assume o valor zero e o outro assume o valor 1.

Para um atributo simbólico com mais de dois valores, a técnica utilizada na conversão depende do fato de o atributo ser nominal ou ordinal. Se não houver uma relação de ordem entre os valores do atributo, a inexistência de uma relação de ordem deve manter-se nos valores numéricos gerados. Ou seja, a diferença entre quaisquer dois valores numéricos deve ser a mesma. Uma forma de garantir esta consistência consiste em codificar cada valor nominal por uma sequência de c bits, em que c é igual ao número de possíveis valores ou categorias.

Na codificação $1-de-c$, também denominada canónica ou topológica, cada sequência possui apenas um bit com o valor 1 e os demais com o valor zero. A diferença entre as sequências é definida pela posição que o valor 1 ocupa nelas. Para definir a diferença entre dois valores, pode ser utilizada a distância de Hamming. A distância de Hamming entre duas sequências binárias com mesmo número de elementos é igual ao número de posições em que as sequências apresentam valores diferentes. É fácil verificar que a distância de Hamming entre qualquer par de valores é igual a 2 (apenas duas posições do *string* binário têm valores diferentes).

Nesta codificação, cada posição da sequência binária corresponde a um possível valor do atributo nominal. Por exemplo, se a sequência binária possui 4 bits, o primeiro bit

corresponde ao primeiro valor, o segundo bit ao segundo valor e assim por diante. Como apenas um dos bits pode assumir o valor 1, o bit que assumir esse valor sinaliza a presença do valor nominal correspondente àquele bit. A moda do valor de um atributo para o conjunto de objetos é definida pela posição (bit) da sequência que apresenta o maior número de valores iguais a 1. Isto indica que o valor correspondente àquela posição é o que aparece com maior frequência no conjunto de dados.

Na Tabela 3.8 é ilustrada a codificação $1-de-c$ para um conjunto de seis valores nominais, em que cada valor representa uma cor.

Dependendo do número de valores nominais, a sequência binária para representar cada valor pode ficar muito longa. Considere que se pretende codificar os nomes de países com a codificação $1-de-c$. Como existem 193 países (incluindo o Vaticano), seria necessário utilizar vetores com 193 elementos.

Tabela 3.8 *Codificação* $1-de-c$

Atributo nominal	**Código** $1-de-c$
Azul	100000
Amarelo	010000
Verde	001000
Preto	000100
Marrom	000010
Branco	000001

Uma alternativa consiste na representação dos possíveis valores nominais por um conjunto de pseudo atributos. Os valores dos pseudo atributos podem ser do tipo binário, inteiro ou real. Para o exemplo anterior, a Tabela 3.9 mostra como é possível transformar um conjunto de 193 atributos, um para cada país, num conjunto de cinco pseudo atributos, que podem ser utilizados para identificar cada um dos países (os valores entre parêntesis denotam se o valor é do tipo binário (b) ou inteiro (i), respetivamente). O primeiro desses cinco pseudo atributos é o atributo *continente*, um atributo nominal que pode assumir um de entre sete valores possíveis. Os outros quatro pseudo atributos, Produto Interno Bruto (*PIB*), *População*, temperatura média anual (*TMA*) e *Área* podem assumir um valor inteiro cada. Tal como para o atributo original, uma combinação de valores para os cinco pseudo atributos representa um único país.

Tabela 3.9 *Pseudoatributos e os seus valores possíveis*

Pseudoatributo	**#Valores**
Continente	7 (b)
PIB	1 (i)
População	1 (i)
TMA	1 (i)
Área	1 (i)

Quando o atributo é do tipo ordinal, existe uma relação de ordem entre os valores do atributo, e a codificação deve preservar essa ordem. Para isso, deve ser utilizada uma codificação na qual a ordem dos valores seja evidente. Quando o valor numérico é um número inteiro ou real, essa transformação é simples e direta: basta ordenar os valores categóricos ordinais e codificar cada valor de acordo com a sua posição na ordem, como ilustrado no exemplo da Tabela 3.10.

Tabela 3.10 *Conversão de valor ordinal para inteiro*

Valor ordinal	Valor inteiro
Primeiro	0
Segundo	1
Terceiro	2
Quarto	3
Quinto	4
Sexto	5

Nesta tabela, a distância entre os valores varia de acordo com a respetiva proximidade, ao contrário do que ocorreu com a codificação ilustrada na Tabela 3.8.

Se for necessário converter valores ordinais em valores binários, pode ser utilizado o código cinza ou o código termómetro. O código cinza é constantemente utilizado para correções de erro em comunicações digitais. No código termómetro, o aumento dos valores assemelha-se ao aumento de temperatura num termómetro analógico. Estes dois códigos são ilustrados na Tabela 3.11. Nesta tabela é possível verificar que, em ambos os casos, dois valores próximos diferem por apenas um bit (possuem distância de Hamming igual a 1). A tabela também mostra que o código termómetro, em que um valor 1 é acrescentado à medida que se avança na escala de valores, utiliza sequências binárias maiores (mais bits) que o código cinza.

Tabela 3.11 *Conversão de valor ordinal para binário*

Valor ordinal	Código cinza	Código termómetro
Primeiro	000	00000
Segundo	001	00001
Terceiro	011	00011
Quarto	010	00111
Quinto	110	01111
Sexto	100	11111

3.7.2 Conversão Numérico-Simbólico

Algumas técnicas de ECD foram desenvolvidas para trabalhar com valores qualitativos, por exemplo, alguns modelos Bayesianos 5. Alguns desses algoritmos podem lidar com dados quantitativos, mas o seu desempenho pode ficar limitado.

Se o atributo quantitativo for do tipo discreto e binário, com apenas dois valores, a conversão é trivial. Basta associar um nome a cada valor. Se o atributo original for formado por sequências binárias sem uma relação de ordem entre si, cada sequência pode ser substituída por um nome ou categoria. Nos restantes casos, o recurso a métodos de discretização permite transformar atributos quantitativos em qualitativos. Estes métodos transformam valores numéricos em intervalos ou categorias. A escolha do método de discretização depende do problema de aprendizagem. Métodos de discretização podem ser utilizados de forma isolada ou composta, quando mais de um método é utilizado. Existe um grande número de métodos, que podem ser classificados de acordo com diferentes critérios (Yang et al., 2005).

Quando um atributo quantitativo é discretizado, o conjunto de valores possíveis é dividido em intervalos, e cada intervalo de valores quantitativos é convertido num valor qualitativo. Em alguns métodos, o utilizador pode influenciar a definição dos intervalos, definindo valores para parâmetros como, por exemplo, o número máximo de intervalos. Este tipo de métodos são denominados paramétricos. Os métodos não paramétricos definem os intervalos utilizando apenas as informações presentes nos valores do atributo.

Os métodos de discretização podem ser supervisionados ou não supervisionados. No primeiro caso, é utilizada a informação sobre a classe dos exemplos. As técnicas supervisionadas geralmente conduzem a melhores resultados, uma vez que a definição dos intervalos sem conhecimento das classes pode originar mistura de classes. Uma abordagem supervisionada simples seria escolher pontos de corte que maximizam a pureza dos intervalos. Isso pode ser feito utilizando o conceito de entropia. Note-se que discretizar cada atributo separadamente conduz, em geral, a resultados subótimos.

A definição do processo de mapeamento dos valores dos atributos quantitativos em valores qualitativos, a definição do tamanho dos intervalos ou a quantidade de valores nos intervalos fica geralmente a cargo do método de discretização. Algumas das estratégias utilizadas pelos diferentes métodos são seguidamente apresentadas:

- Largura igual: Divide o intervalo de valores original em subintervalos com a mesma largura. O desempenho desta estratégia pode ser afetado pela presença de *outliers*.
- Frequência igual: Atribui o mesmo número de objetos a cada subintervalo. Esta estratégia pode gerar intervalos de tamanhos muito diferentes.
- Utilização de um algoritmo de agrupamento de dados.
- Inspeção visual.

3.7.3 Transformação de Atributos Numéricos

Algumas vezes, o valor numérico de um atributo necessita de ser transformado noutro valor numérico. Isto geralmente ocorre quando os limites inferior e superior dos valores dos atributos são muito diferentes, o que leva a uma grande variação de valores ou, ainda, quando vários atributos se encontram expressos em escalas diferentes. Esta transformação é geralmente realizada para evitar que um atributo predomine sobre outro.

Quando necessário, a operação de transformação é aplicada aos valores de um dado atributo. A título de exemplo, considere que, para um problema de decisão, apenas a magnitude, e não o sinal, é relevante. Uma transformação necessária consistiria em converter os valores desse atributo para seu valor absoluto.

Outra transformação que é muito utilizada é a normalização de dados. A normalização de dados é recomendável quando os limites de valores de atributos distintos são muito diferentes, sendo de evitar que um atributo predomine sobre o outro (a menos que isso seja importante). Quando recomendada, a normalização é aplicada a cada atributo individualmente e pode ocorrer de duas formas:

- Amplitude;
- Distribuição.

A normalização por amplitude pode ser realizada através da alteração da escala ou padronização. A primeira define uma nova escala de valores (limites mínimo e máximo) para cada atributo. A segunda define um valor central e um valor de dispersão comuns para todos os atributos.

Na normalização por alteração da escala, também chamada normalização min-max, são inicialmente definidos os valores mínimo (min) e máximo (max) para os novos valores de cada atributo. Depois, as seguintes operações são realizadas para cada atributo. Primeiro, o menor valor do atributo, (menor), é subtraído a cada valor. Cada valor resultante é, em seguida, dividido pela diferença entre o maior e o menor valores originais do atributo, (maior − menor). Cada novo valor é depois multiplicado pela diferença entre os valores limites da nova escala, max − min. No final, o valor min é somado a cada valor produzido. Estas operações são ilustradas pela Equação 3.5. Para que os limites superior e inferior sejam 1 e 0, respetivamente, basta fazer max = 1 e min = 0.

$$v_{Novo} = \min + \frac{v_{Atual} - \text{menor}}{\text{maior} - \text{menor}} (\max - \min) \tag{3.5}$$

Para a normalização por padronização, a cada valor do atributo a ser normalizado é adicionada ou subtraída uma medida de localização, sendo o valor resultante multiplicado ou dividido por uma medida de escala. Através desta operação, diferentes atributos podem apresentar limites inferiores e superiores diferentes, mas terão os mesmos valores para as medidas de escala e dispersão. Se as medidas de localização e de escala forem a média (μ) e a variância (σ), respetivamente, os valores de um atributo são convertidos para um novo conjunto de valores com média 0 e variância 1. A Equação 3.6 resume esta transformação.

$$v_{Novo} = \frac{v_{Atual} - \mu}{\sigma} \tag{3.6}$$

Geralmente, é preferível padronizar a reescalar, pois a padronização lida melhor com *outliers*. Durante a operação de normalização, é possível fazer com que os atributos mais importantes possuam limites maiores. Para isso, basta fazer com que a padronização gere um intervalo maior (por exemplo, o limite máximo de um atributo importante pode ser o dobro do limite máximo utilizado para os restantes atributos) ou que a medida de reescala gere uma maior variância para um atributo importante (por exemplo, esse atributo pode ter uma variância que é o dobro da variância dos restantes atributos, o que faz com que sua amplitude de valores seja o dobro da faixa utilizada pelos outros atributos).

A normalização por distribuição altera a escala de valores de um atributo. Um exemplo desse tipo de normalização consiste na aplicação de uma função para ordenar os valores do atributo a ser normalizado, e a substituição de cada valor pela posição que ele ocupa no *ranking* (por exemplo, a aplicação desta normalização aos valores 1, 5, 9 e 3 gera, respetivamente, os valores 1, 3, 4 e 2). Se todos os valores originais forem distintos, o resultado é uma distribuição uniforme.

Outro tipo de transformação para atributos de um mesmo tipo é a tradução, em que o valor de um atributo de um dado tipo é traduzido para um valor do mesmo tipo, que seja mais facilmente manipulável. Por exemplo, a conversão de um atributo com data de nascimento para idade, de graus Celsius para Fahrenheit, ou da localização dada por um aparelho de GPS para código postal.

3.8 Redução de Dimensionalidade

Muitos problemas que podem ser tratados através de técnicas de ECD apresentam um elevado número de atributos. Um exemplo de problemas em que o número de atributos é muito grande são as aplicações de reconhecimento de imagens. Se cada pixel [2] da imagem for considerado um atributo, cada imagem ou instância de uma imagem com 1024 por 1024 pixels teria mais de um milhão de atributos. Outro exemplo são os dados de expressão genética, que geralmente apresentam algumas dezenas de objetos (amostras), cada um com milhares de atributos (genes). Poucas técnicas de ECD são capazes de lidar com um número tão elevado de atributos.

O efeito do número muito elevado de atributos em algoritmos de ECD é descrito pelo problema da maldição da dimensionalidade. Se cada atributo for visto como uma coordenada num espaço d-dimensional, em que d é o número de atributos, o hipervolume que representa esse espaço cresce exponencialmente com a adição de novos atributos. Para clarificar o problema, considere um conjunto de dados em que cada objeto possui apenas um atributo e que esse atributo pode assumir um de entre 10 valores. Esse conjunto de dados pode ter então 10^1 ou 10 objetos diferentes, um para cada valor diferente do atributo.

[2] Um pixel é a unidade básica de uma imagem.

Se o número de atributos aumentar para cinco, o número de objetos possíveis passa a ser 10^5, que é um número de objetos possíveis muito superior ao caso em que apenas um atributo foi utilizado. Uma forma de minimizar o impacto do problema da dimensionalidade consiste em combinar, ou eliminar, parte dos atributos irrelevantes.

Em muitos algoritmos de ECD, para que os dados com um número elevado de atributos possam ser utilizados, o número de atributos precisa ser reduzido. A redução do número de atributos pode ainda melhorar o desempenho do modelo induzido, reduzir o seu custo computacional e facilitar a interpretação dos resultados obtidos. Diferentes técnicas originárias de áreas como Reconhecimento de Padrões, Estatística e Teoria da Informação podem ser utilizadas para a redução do número de atributos. Essas técnicas podem ser divididas em duas grandes abordagens:

- Agregação;
- Seleção de Atributos.

Enquanto as técnicas de agregação substituem os atributos originais por novos atributos formados pela combinação de grupos de atributos, as técnicas de seleção mantêm uma parte dos atributos originais e descartam os demais atributos.

3.8.1 Agregação

As principais técnicas utilizadas para reduzir as dimensões por agregação combinam os atributos originais por meio de funções lineares ou não lineares. Uma das técnicas mais conhecidas é a Análise de Componentes Principais (PCA, do inglês *Principal Component Analysis*) (Pearson, 1901). A técnica PCA é um procedimento matemático que utiliza uma transformação ortogonal para converter um conjunto de observações de variáveis possivelmente correlacionadas num conjunto de novos valores de variáveis não-correlacionadas. As novas variáveis são combinações lineares das variáveis originais, e são designadas por componentes principais.

Nas técnicas de agregação, a combinação de atributos, conduz à perda dos valores originais. Em várias aplicações, por exemplo, nas áreas de biologia, finanças, medicina e monitorização ambiental, geralmente é importante preservar os valores dos atributos para que os resultados obtidos possam ser interpretados, associando os resultados produzidos por uma técnica estatística ou de ECD aos valores dos atributos. Por esse motivo, nestas áreas, é mais frequente reduzir o número de atributos pelo recurso a técnicas de seleção.

3.8.2 Seleção de Atributos

Várias aplicações reais apresentam um elevado número de atributos. Além do problema da maldição da dimensionalidade, alguns destes atributos podem ser irrelevantes ou redundantes.

A seleção de atributos permite:

- Identificar atributos importantes;
- Melhorar o desempenho de várias técnicas de ECD;
- Reduzir a necessidade de memória e tempo de processamento;
- Eliminar atributos irrelevantes e reduzir ruído;
- Lidar com a maldição da dimensionalidade;
- Simplificar o modelo gerado e facilitar a sua compreensão;
- Facilitar a visualização dos dados;
- Reduzir o custo de recolha de dados.

Conforme visto anteriormente, alguns atributos são claramente redundantes ou irrelevantes, podendo ser manualmente eliminados. No entanto, na prática, vários atributos passíveis de eliminação não são facilmente identificados, o que torna pouco eficiente o uso exclusivo de técnicas visuais. Possíveis razões subjacentes a esta dificuldade incluem:

- Número muito elevado de exemplos;
- Número muito elevado de atributos;
- Relações complexas entre atributos, que dificultam a descoberta de relações entre eles.

Para lidar com estes casos, diversas técnicas automáticas têm sido propostas na literatura para a seleção de atributos. Estas técnicas procuram um subconjunto ótimo de atributos de acordo com um dado critério. As técnicas propostas podem ser classificadas de diferentes formas. Uma delas diz respeito à avaliação do conjunto de atributos selecionados. Neste caso específico, as técnicas existentes podem estar integradas num algoritmo de indução ou ser independentes do algoritmo. Para avaliar a qualidade ou desempenho de um subconjunto de atributos, três abordagens têm sido utilizadas:

- Embutida;
- Baseada em filtro;
- Baseada em *wrapper*.

Na abordagem embutida, a seleção do subconjunto é embutida ou integrada no próprio algoritmo de aprendizagem. As árvores de decisão (Seção 6.1) realizam este tipo de seleção interna de atributos.

Nas abordagens baseadas em filtros, como o próprio nome indica, é utilizado um filtro, numa etapa de pré-processamento, sobre o conjunto de atributos original, que seleciona um subconjunto de atributos, sem levar em consideração o algoritmo de aprendizagem que utilizará esse subconjunto. As técnicas que seguem esta abordagem analisam, por exemplo, a correlação entre os atributos. A medida de correlação mais utilizada é a correlação de Pearson, definida pela Equação 3.7, em que $\bar{x}_i = \sum_{l=1}^{d} x_i^l / d$. Permite identificar

atributos redundantes: aqueles com elevada correlação entre si. Esta medida é insensível a diferenças na magnitude dos atributos, sendo muito usada para determinar a semelhança entre objetos em áreas como Bioinformática, em que apenas o padrão de variação dos atributos dos objetos é importante.

$$\begin{aligned} \text{pearson}(\mathbf{x}_i, \mathbf{x}_j) &= \frac{\text{covariância}(\mathbf{x}_i, \mathbf{x}_j)}{\sqrt{\text{variância}(\mathbf{x}_i)\text{variância}(\mathbf{x}_j)}} \\ &= \frac{\sum_{l=1}^{d} (x_i^l - \bar{x}_i)(x_j^l - \bar{x}_j)}{\sqrt{(\sum_{k=1}^{d} (x_i^l - \bar{x}_i)^2 \sum_{l=1}^{d} (x_j^l - \bar{x}_j)^2)}} \end{aligned} \tag{3.7}$$

As abordagens baseadas em *wrappers* utilizam o próprio algoritmo de aprendizagem como uma caixa-preta para a seleção de atributos. Geralmente são utilizadas juntamente com uma técnica de amostragem. Para cada possível subconjunto, o algoritmo é consultado e o subconjunto que apresentar a melhor combinação entre redução da taxa de erro e redução do número de atributos é em geral selecionado.

Algumas vantagens da abordagem baseada em filtro podem ser destacadas:

- como o processo de seleção não depende de nenhum indutor, as caraterísticas selecionadas podem ser utilizadas por diferentes algoritmos de ECD;
- as heurísticas utilizadas para avaliar um subconjunto são computacionalmente 'leves', pelo que a aplicação de filtros pode ser bastante eficiente;
- os filtros conseguem lidar eficientemente com uma grande quantidade de dados.

A principal desvantagem dos filtros refere-se à sua independência em relação ao algoritmo de ECD. Como a seleção de atributos e a classificação são processos separados, o viés[3] de um não interage com o viés de outro, o que pode levar à construção de classificadores com um desempenho aquém do desejado.

As técnicas baseadas em *wrapper* representam uma alternativa simples e poderosa para selecionar atributos. Em geral, as técnicas embutidas fazem melhor uso dos dados disponíveis do que as técnicas baseadas em *wrapper*. Além disso, por não precisar retreinar um algoritmo de ECD para cada novo conjunto de atributos, as técnicas embutidas são, em geral, mais rápidas (Guyon e Elisseeff, 2003).

Outra forma de análise, está relacionada com o fato de a seleção dos atributos ser feita de forma individual ou coletiva. No primeiro caso, os atributos são ordenados de acordo com a sua relevância para discriminar os objetos das diferentes classes. Boa parte das técnicas de seriação pode ser apenas utilizada em problemas de classificação binária (pro-

[3]O viés refere-se à tendência dos algoritmos em favorecer, de forma sistemática, determinadas situações, dentre as muitas disponíveis. Em ECD, um exemplo pode ser a preferência de certos algoritmos em construir classificadores mais ou menos complexos. Na seleção de atributos, um algoritmo pode preferir atributos que possuam certas propriedades.

blemas com duas classes). A segunda alternativa seleciona um subconjunto dos atributos originais que melhor separe os exemplos das diferentes classes.

Finalmente, algumas técnicas de seleção de atributos utilizam a informação sobre a classe, sendo denominadas *supervisionadas*, e outras, por não utilizar essa informação, são denominadas de *não supervisionadas*.

A seleção de atributos, tanto por ordenação, como por seleção de subconjuntos, e utilizando ou não informação sobre a classe, pode ser feita quer com as abordagens filtro quer por *wrapper*. A abordagem embutida trabalha com seleção de subconjuntos. Na Seção seguinte, apresentamos detalhes sobre as técnicas de ordenação e seleção por subconjuntos.

3.8.3 Técnicas de Seriação

A seriação de atributos pode ser encarada como uma forma simples de seleção, em que os atributos são ordenados de acordo com a sua relevância para um dado critério, por exemplo, classificação dos objetos nas diferentes classes. Em problemas de classificação, os atributos no topo da seriação são selecionados para utilização pelo classificador. Grande parte das técnicas existentes foi desenvolvida para a redução do número de genes em problemas de análise de expressão genética por meio de microarranjos. Nestes problemas, cada gene pode ser visto como um atributo e cada objeto é descrito por milhares de genes ou atributos.

A escolha da melhor abordagem para a seleção de atributos depende das propriedades a serem medidas (Dopazo et al., 2001). Várias das abordagens descritas na literatura são técnicas para ordenação (*ranking*), que atribuem uma pontuação ou medida a cada subconjunto de atributos. Algumas destas técnicas avaliam a semelhança (medidas de correlação), enquanto que outras avaliam a diferença (medidas de distância) entre vetores. As medidas podem ou não considerar a informação sobre a classe.

As medidas podem ainda ser divididas em paramétricas e não paramétricas. As medidas paramétricas assumem uma distribuição estatística para cada grupo ou classe. As medidas não paramétricas não fazem essa assunção, sendo mais robustas. As medidas não paramétricas geralmente especificam uma hipótese em termos de distribuições populacionais, em vez de parâmetros, como a média e o desvio padrão. Estas medidas conseguem detetar diferenças entre populações quase tão bem quanto as medidas paramétricas, quando assunções como a normalidade necessitam de ser verificadas. Quando estas assunções não são satisfeitas, as medidas não paramétricas são, potencialmente, mais poderosas que as paramétricas na deteção de diferenças entre populações.

Note-se que, no caso da seriação dependente de classe, o primeiro atributo é aquele que melhor discrimina os objetos das diferentes classes, o segundo é o segundo melhor atributo a discriminar as classes, e assim sucessivamente. Na seleção do subconjunto, os atributos que fazem parte do subconjunto selecionado não estariam necessariamente no topo da lista se uma técnica de seriação fosse utilizada. Neste caso, o que importa é como os atributos selecionados atuam de forma coletiva. Isto ocorre, por exemplo, devido a complementaridades e interações entre os atributos.

3.8.4 Técnicas de Seleção de Subconjunto

A seleção de um subconjunto de atributos é um processo computacionalmente mais custoso que a ordenação dos atributos. Esta desvantagem acentua-se com o crescimento do número de atributos. A seleção de subconjuntos de atributos pode ser intratável quando o número de atributos é muito elevado. Uma alternativa, consiste em ordenar, em primeiro lugar, os d atributos originais e, em seguida, selecionar um subconjunto a partir dos $d_{reduzido}$ atributos melhor posicionados.

Por incorporar o viés do classificador, as técnicas baseadas em *wrapper* conseguem obter, em geral, um conjunto de atributos que conduz a um melhor desempenho do modelo.

A seleção de um subconjunto de atributos pode ser interpretada como um problema de procura, em que, cada ponto no espaço de procura é visto como um possível subconjunto de atributos. Nesta perspetiva, um método de seleção deve definir (Blum e Langley, 1997):

- Qual(is) será(ão) o(s) ponto(s) de partida, ou a direção em que a procura será realizada;
- Que estratégia de procura será adotada;
- Qual o critério a ser utilizado na avaliação dos subconjuntos gerados;
- Qual será o critério de paragem.

Os critérios de avaliação já foram discutidos anteriormente, que são as abordagens filtro, *wrapper* ou embutida. A seguir serão discutidos os outros três aspetos. No que respeita ao ponto de partida e á direção da procura, quatro diferentes alternativas podem ser adotadas:

- Eliminação para trás (*backward elimination*), que inicia com todos os atributos e remove um atributo de cada vez.
- Geração para a frente (*forward generation*), que começa sem nenhum atributo e adiciona um atributo de cada vez.
- Geração bidirecional (*bidirectional generation*), em que a procura pode começar em qualquer ponto, e os atributos podem ser adicionados e/ou removidos.
- Geração estocástica (*random generation*), quando o ponto de partida da procura e os atributos a serem removidos ou adicionados são decididos de forma estocástica.

Em relação à estratégia de procura a ser adotada, as principais abordagens são:

- Procura completa (exponencial ou exaustiva), que avalia todos os possíveis subconjuntos.
- Procura heurística (sequencial), que utiliza regras e métodos para conduzir a procura e que não garante que uma solução ótima seja encontrada.
- Procura não determinística, que está relacionada com a geração estocástica. Neste caso, embora possa encontrar uma boa solução, não é possível garantir que será encontrada a melhor solução possível.

Finalmente, no que respeita a terminar a procura, pode ser conduzida uma procura exaustiva, em que a procura termina quando todos os subconjuntos forem testados, ou pode ser adotado um critério de paragem, que define quando terminar a procura pelo melhor subconjunto de atributos. Este critério pode ser, por exemplo, um número máximo de alternativas testadas, um número de atributos a serem selecionados sem degradação do desempenho do classificador ou o tempo de processamento.

3.9 Considerações Finais

A aplicação prévia de técnicas de pré-processamento ao conjunto de dados pode facilitar o processo de aprendizagem e melhorar o desempenho dos algoritmos de ECD, uma vez que estas técnicas podem eliminar ou reduzir problemas presentes nos dados.

Neste capítulo foram apresentadas diversas técnicas utilizadas no pré-processamento de dados. Encetando com a eliminação manual de atributos que se mostrarem claramente desnecessários para o uso dos dados por algoritmos de ECD, discutiu-se a integração de dados provenientes de diferentes fontes, a utilização de técnicas de amostragem para selecionar subconjuntos representativos de dados, como lidar com dados desbalanceados e como operações de limpeza de dados podem tratar dados incompletos, inconsistentes, redundantes e com ruído. De seguida, foram apresentadas alternativas para a transformação de dados, com o intuito de facilitar o seu uso por diferentes algoritmos de ECD, tais como a conversão de dados de um tipo para outro e a normalização de dados numéricos. Por fim, foi observada a importância da redução da dimensionalidade dos dados, principalmente quando o número de atributos é muito elevado.

Os problemas mencionados são muito frequentes em conjuntos de dados reais e o pré-processamento conduz, em geral, a um melhor desempenho do algoritmo de ECD utilizado. Por esse motivo, a etapa de pré-processamento assume uma importância preponderante em aplicações reais de ECD.

Parte II

Modelos Preditivos

Introdução

Um algoritmo de ECD preditivo é uma função que, dado um conjunto de exemplos etiquetados, constrói um estimador. O rótulo, ou etiqueta, toma valores num domínio conhecido. Se o domínio for um conjunto de valores nominais, estamos perante um problema de classificação, e o estimador gerado é um classificador. Se o domínio for um conjunto infinito e ordenado de valores, estamos perante um problema de regressão, que induz um regressor. Um classificador (ou regressor) é, também, uma função que, dado um exemplo não rotulado, atribui a esse exemplo uma das possíveis classes (ou um valor real) (Dietterich, 1998).

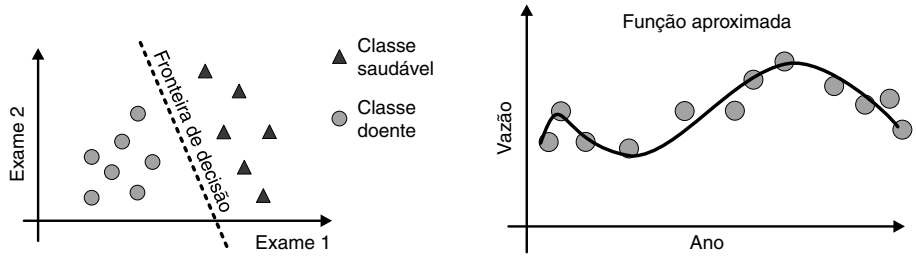
Uma definição formal é: dado um conjunto de observações de pares $\mathbf{D} = \{(\mathbf{x}_i, f(\mathbf{x}_i)), i = 1, \ldots, n\}$, em que f representa uma função desconhecida, um algoritmo de ECD preditivo aprende uma aproximação $\hat{f}$ da função desconhecida f. Essa função aproximada, $\hat{f}$, permite estimar o valor de f para novas observações de $\mathbf{x}$. De acordo com a natureza de f, é comum distinguir duas situações possíveis:

- Classificação: $y_i = f(\mathbf{x}_i) \in \{c_1, \ldots, c_m\}$, ou seja, $f(\mathbf{x}_i)$ assume valores num conjunto discreto, não ordenado;
- Regressão: $y_i = f(\mathbf{x}_i) \in \Re$, ou seja, $f(\mathbf{x}_i)$ assume valores num conjunto infinito e ordenado.

A Figura 3.1(a) ilustra um cenário onde se pretende discriminar pessoas saudáveis de pessoas doentes usando a informação proveniente de dois exames. O objetivo é encontrar uma fronteira de decisão que separe os exemplos de uma classe dos exemplos da outra classe. Se os exemplos de uma classe forem linearmente separáveis dos exemplos da outra classe, uma vez que os exemplos são caraterizados por dois atributos, a fronteira de decisão pode ser uma reta - combinação linear dos atributos.

Diferentes algoritmos de ECD encontram diferentes fronteiras de decisão. Além disso, diferenças no conjunto de treino, alterações na ordem de apresentação dos exemplos durante o treino e processos estocásticos internos podem fazer com que um mesmo algoritmo de ECD encontre fronteiras de decisão diferentes.

A Figura 3.1(b) ilustra um caso de regressão em que o objetivo consiste em aprender uma função que estime o caudal da água de um dado rio ao longo do tempo. Neste caso é utilizado, apenas, um atributo de entrada. Unindo os pontos formados pelo par



(a) Classificação (b) Regressão

Figura 3.1 *Gráficos ilustrativos das tarefas de classificação e regressão.*

Tabela 3.12 *Exemplo de um conjunto de dados para um problema de classificação*

Tamanho (P)	Largura (P)	Tamanho (S)	Largura (S)	Espécie
5,1	3,5	1,4	0,2	Setosa
4,9	3,0	1,4	0,2	Setosa
7,0	3,2	4,7	1,4	Versicolor
6,4	3,2	4,5	1,5	Versicolor
6,3	3,3	6,0	2,5	Virgínica
5,8	2,7	5,1	1,9	Virgínica

$\{dia, caudal\}$, obtemos uma curva. Espera-se que essa curva se aproxime da curva verdadeira da função que, dado um determinado dia, retorna uma estimativa do caudal.

Nas Tabelas 3.12 e 3.13 são apresentados exemplos de dois conjuntos de dados. O primeiro conjunto (Tabela 3.12) apresenta exemplos de dados para um problema de classificação, o problema `iris`, abordado no Capítulo 2. O atributo alvo, que aparece na última coluna, assume valores discretos e não ordenados. O segundo conjunto, apresentado na Tabela 3.13, refere-se a um problema de regressão. O atributo alvo, que aparece na última coluna, assume valores contínuos e ordenados.

Nos dois conjuntos, cada linha corresponde a um objeto ou observação. A última coluna apresenta o valor da função f para essa observação, que corresponde ao atributo

Tabela 3.13 *Exemplo de um conjunto de dados para um problema de regressão*

Fertilidade	Agricultura	Educação	Renda	Mortalidade
80,2	17,0	12	9,9	22,2
83,1	45,1	9	84,8	22,2
92,5	39,7	5	93,4	20,2
85,8	36,5	7	33,7	20,3
76,9	43,5	15	5,2	20,6

alvo. Esta coluna é também designada por variável dependente ou variável objetivo. No caso particular de problemas de classificação, é designada por *classe*. As colunas restantes são designadas por atributos de entrada, atributos preditivos ou variáveis independentes. Os atributos serão utilizados como entrada do algoritmo na realização da predição.

O primeiro conjunto de dados, denominado `iris`, é bastante utilizado para ilustrar o uso de técnicas de ECD na solução de problemas de classificação. Com base em algumas caraterísticas externas de flores de uma planta denominada *íris*, nomeadamente tamanhos e larguras das suas pétalas e sépalas, é possível separá-las numa de três classes, que designam diferentes variedades dessa planta: *setosa*, *versicolor* e *virgínica*. Um dos motivos para esse conjunto de dados ser largamente utilizado está relacionado com o fato de se conhecer de antemão que uma das classes, a variedade *setosa*, ser linearmente separável das outras duas. No caso das variedades *versicolor* e *virgínica*, por sua vez, a separação existe, mas não é linear e de fato há vários objetos dessas classes próximos entre si.

O segundo conjunto de dados, apresentado na Figura 3.13, é denominado `swiss`. Para este conjunto, deseja-se relacionar estatísticas de uma população, tais como nível de educação, taxa de fertilidade, entre outras, à previsão da sua taxa de mortalidade. Em ambos os casos, o objetivo da aprendizagem preditiva é aprender uma função $\hat{f}(\mathbf{x})$ que mapeia as variáveis independentes, os atributos de entrada, na variável objetivo, o atributo alvo. A função $\hat{f}$ pode assumir diferentes formas. Por exemplo, combinações lineares ou não lineares dos atributos de entrada, funções por ramos, expressões lógicas etc., que são estudadas nos capítulos seguintes. É importante notar que $\hat{f}$ é uma aproximação de uma função desconhecida f, ou seja podem existir $\mathbf{x}$ para os quais $\hat{f}(\mathbf{x}) \neq f(\mathbf{x})$. A estimativa da qualidade de um modelo preditivo é dada pelo custo associado às previsões do modelo. Dependendo do tipo de problema, classificação ou regressão, são utilizadas diferentes funções de custo. No caso de problemas de classificação, é usual utilizar a função de custo $0-1$: o custo de uma previsão incorreta ($\hat{f}(\mathbf{x}) \neq f(\mathbf{x})$) é 1, e o custo de uma previsão correta ($\hat{f}(\mathbf{x}) = f(\mathbf{x})$) é 0. Em problemas de regressão, é comum utilizar o erro quadrático médio. As funções de custo e as metodologias para as estimar são estudadas no Capítulo 9.

Considere, para simplificar, um problema de classificação com duas classes. Assuma que é conhecida a função densidade de probabilidade (pdf, do inglês *probability density function*) para cada classe. A Figura 3.2 mostra as duas pdfs (uma para cada classe) num problema definido por um único atributo de entrada. O melhor classificador possível divide o domínio da variável no ponto de interseção das duas pdfs, ou seja, classifica um objeto na classe com maior função densidade de probabilidade. A área sombreada representa o erro cometido por esse classificador. É intuitivo que esse classificador tem erro mínimo: movendo em qualquer direção na superfície de decisão, o erro sempre cresce. O erro desse classificador é conhecido como *erro de Bayes ótimo* e é um mínimo teórico da capacidade de generalização de qualquer classificador. No Capítulo 9 são indicadas metodologias para estimar a qualidade de $\hat{f}$, ou seja, quão aproximado é $\hat{f}$ de f.

Na literatura de aprendizagem automática é usual distinguir *modelos generativos* de *modelos discriminativos*. Os modelos generativos estimam uma função densidade de probabilidade condicionada à classe c_i. Um exemplo de teste é classificado na classe que

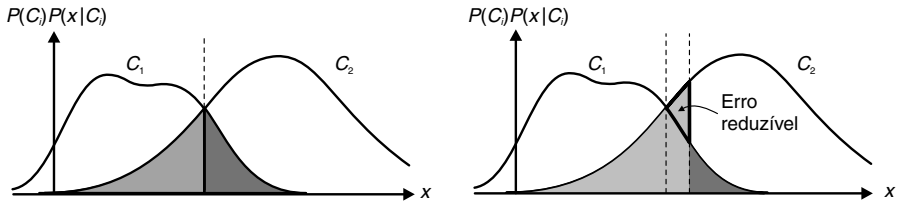


Figura 3.2 *A figura ilustra um problema de decisão com duas classes, mostrando a função distribuição de probabilidade para cada classe. A figura da esquerda representa a superfície de decisão ótima. A área cinzenta corresponde à probabilidade de erro ao decidir por C_1 quando a classe correta é C_2; a área cinza-escura é o erro oposto. Se movemos a superfície de decisão, como se vê na figura da direita, o erro sempre aumenta.*

maximiza a probabilidade *a posteriori*. São designados generativos porque assumem que os exemplos de cada classe são gerados por uma função densidade de probabilidade condicionada. O *naive* Bayes (Duda et al., 2001) é um exemplo clássico deste grupo de modelos. Os modelos discriminativos não modelam a distribuição de probabilidade, mas calculam diretamente as probabilidades *a posteriori*. Os modelos tradicionais incluem árvores de decisão (Quinlan, 1993), redes neuronais (Mitchell, 1997) e vizinhos mais próximos (Aha et al., 1991).

Nos capítulos seguintes serão descritos os principais métodos preditivos de ECD. Estão organizados em métodos baseados em distâncias, métodos probabilísticos, métodos baseados em procura e métodos baseados em otimização. No Capítulo 8, são apresentadas estratégias de combinar métodos de classificação em modelos preditivos múltiplos. Finalmente, são discutidas formas de desenhar experiências de ECD e de analisar os resultados obtidos.

Capítulo 4

Métodos Baseados em Distâncias

4.1 Introdução

Neste capítulo serão apresentadas técnicas de ECD que consideram a proximidade entre os dados na realização de predições. A hipótese de base é que dados similares tendem a estar concentrados na mesma região no espaço de entrada. Alternativamente, dados que não são similares estarão distantes entre si.

A título ilustrativo, na Figura 4.1 é apresentada a projeção em duas dimensões do conjunto de dados `iris`. Os objetos da mesma classe estão representados pela mesma cor. É fácil observar visualmente a existência de áreas densas com objetos pertencentes à mesma classe, evidenciando que a distância entre os objetos está relacionada com a definição de suas classes.

Um método baseado em distância utilizado com frequência é o algoritmo dos vizinhos mais próximos. Este algoritmo é o mais simples de todos os algoritmos de aprendizagem automática. A intuição por trás do algoritmo é:

Objetos do mesmo conceito são semelhantes entre si.

O algoritmo dos vizinhos mais próximos classifica um novo objeto com base nos exemplos do conjunto de treino que são próximos desse objeto. É um algoritmo preguiçoso (*lazy*), porque não aprende um modelo compacto para os dados, apenas memoriza os objetos de treino. Uma vantagem deste algoritmo é que pode ser utilizado tanto em problemas de classificação como em problemas de regressão de maneira direta, sem necessidade de alterações significativas. Para algumas técnicas, como as Máquinas de Vetores de Suporte, essa generalização envolve grandes alterações no algoritmo de aprendizagem, conforme será discutido no Capítulo 7.

Neste capítulo serão apresentados alguns métodos de aprendizagem baseados em distâncias. Na Seção 4.2 apresentamos a descrição do funcionamento do algoritmo dos vizinhos mais próximos utilizando o caso mais simples, isto é apenas um vizinho. A Seção 4.3 apresenta o algoritmo k-NN, que é analisado na Seção 4.4. Variações recentemente desenvolvidas para o algoritmo k-NN são objeto da Seção 4.5. A Seção 4.6 apresenta uma

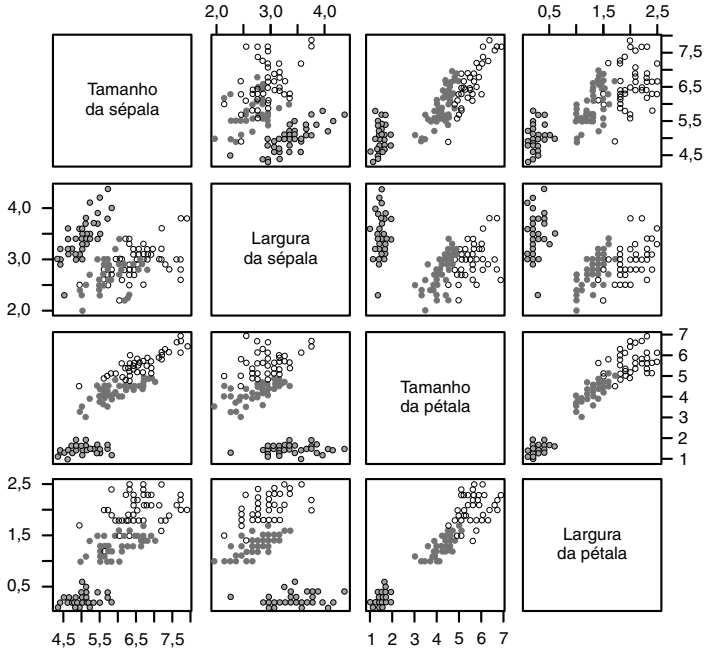


Figura 4.1 *Projeção sobre o plano definido por dois atributos dos objetos do conjunto de dados* `iris`.

metodologia de ECD que utiliza distância para recuperar a solução de problemas passados que sejam semelhantes ao problema que se deseja resolver, conhecida como raciocínio baseado em casos. A última Seção apresenta as considerações finais para os temas estudados neste capítulo.

4.2 O Algoritmo do 1-Vizinho Mais Próximo

O algoritmo dos vizinhos mais próximos tem variações definidas pelo número de vizinhos considerados. Dessas variações, a mais simples é o algoritmo 1-vizinho mais próximo (1-NN, do inglês *1-Nearest Neighbour*).

Nesse algoritmo, cada objeto representa um ponto num espaço definido pelos atributos, denominado espaço de entrada, como ilustrado no Capítulo 1. Definindo uma métrica nesse espaço, é possível calcular as distâncias entre cada dois pontos. A métrica mais usual é a distância euclidiana, dada pela Equação 4.1, em que $\mathbf{x}_i$ e $\mathbf{x}_j$ são dois objetos representados por vetores no espaço $\Re^d$, e x_i^l e x_j^l são elementos desses vetores, que correspondem aos valores da coordenada l (atributos).

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{l=1}^{d} (x_i^l - x_j^l)^2} \tag{4.1}$$

Como dito anteriormente, o algoritmo 1-NN é muito simples, e está ilustrado no Algoritmo 4.1. Na fase de treino, o algoritmo memoriza os exemplos rotulados do conjunto de treino. Para classificar um exemplo não rotulado, ou seja, cuja classe não é conhecida, é calculada a distância entre o vetor de valores de atributos e cada exemplo rotulado em memória. O rótulo da classe associado ao exemplo de treino mais próximo do exemplo de teste é utilizado para classificar o novo exemplo.

Algoritmo 4.1 Versão básica do algoritmo 1-vizinho mais próximo

Entrada: Um conjunto de treino: $\mathbf{D} = \{(\mathbf{x}_i, y_i), i = 1, \ldots, n\}$
Um objeto de teste a ser classificado: $t = \{\mathbf{x}_t, y_t = ?\}$
A função de distância entre objetos: $d(\mathbf{x}_a, \mathbf{x}_b)$
Saída: y_t: Classe atribuída ao exemplo t

```
1 d_min ← +∞
2 para cada i ∈ 1,...,n faça
3     se d(x_i, x_t) < d_min então
4         d_min ← d(x_i, x_t)
5         idx ← i
6     fim
7 fim
8 y_t = y_idx
9 Retorna: y_t
```

A Figura 4.2 apresenta um exemplo ilustrativo de aplicação do algoritmo 1-NN num problema de duas classes. Neste exemplo, considera-se um conjunto de dados cujos objetos são indivíduos que podem ser classificados em saudáveis ou doentes, e o espaço de entrada é definido por dois atributos que representam o resultado de dois exames. O ponto representado por **"?"** é o ponto de teste, ou seja, o indivíduo a ser classificado. Todos os outros pontos são objetos em que a classe é conhecida: saudável, representada por um triângulo ou doente, representada por um círculo. No espaço definido pelos atributos, e usando a distância euclidiana, o objeto de treino mais próximo do objeto de teste pertence à classe doente, que é então atribuída ao objeto de teste.

4.2.1 Superfícies de Decisão

Apesar da sua simplicidade, as superfícies de decisão desenhadas pelo algoritmo do 1-NN são muito complexas: são poliedros convexos com centro em cada objeto do conjunto

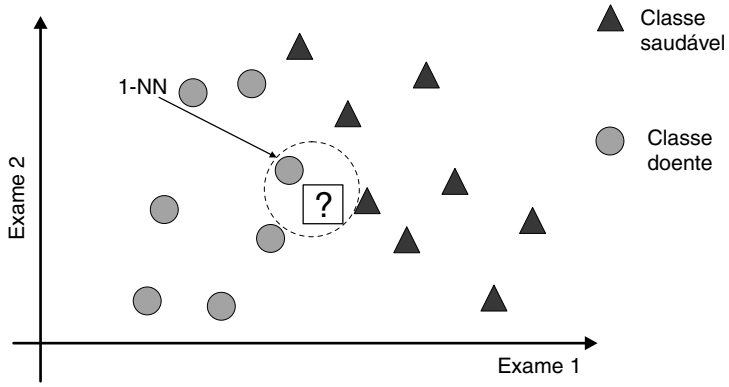


Figura 4.2 *Exemplo ilustrativo do algoritmo* 1*-NN.*

de treino. Todos os pontos no interior de um poliedro pertencem à classe do objeto do conjunto de treino que define o centro desse poliedro. O conjunto desses poliedros é designado *diagrama de Voronoi*. A Figura 4.3 mostra a construção de um exemplo de diagrama de Voronoi.

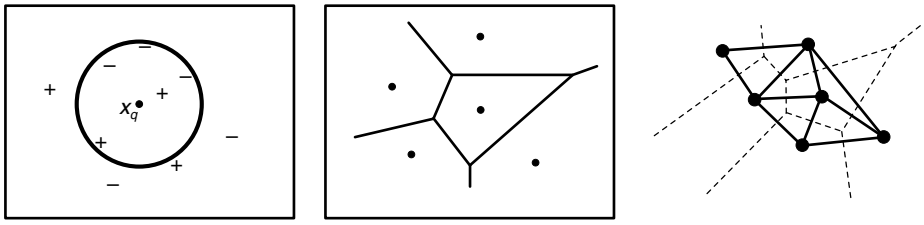


Figura 4.3 *Superfície de decisão: diagrama de Voronoi.*

Para entender a construção do diagrama de Voronoi, seja **D** um conjunto de treino. A célula de Voronoi em torno de um ponto $\mathbf{x} \in \mathbf{D}$ é definida como o conjunto de pontos cuja distância a **x** é menor que a distância a qualquer outro ponto de **D**. A superfície de decisão desenhada pelo algoritmo 1-NN é um conjunto de poliedros convexos contendo cada um dos objetos de treino.

4.2.2 Distâncias

O desempenho dos métodos baseados em distâncias, como o algoritmo dos vizinhos mais próximos, são afetados pela medida ou função de distância utilizada. Um problema da medida de distância apresentada está em pressupor que os dados correspondem a pontos no espaço d-dimensional ($\Re^d$), ou seja, que os seus atributos são numéricos (contínuos). Contudo, em diversos problemas os dados são descritos por atributos qualitativos. Outro

aspeto que deve ser observado no cálculo da distância entre objetos é a escala utilizada para os valores dos atributos. Por exemplo, qual o efeito na função distância da representação de um atributo em *cm* ou *Km*? As medidas de distância são afetadas pela escala dos atributos. Para minimizar esse efeito, os atributos são geralmente normalizados.

4.3 O Algoritmo *k*-NN

Uma extensão imediata ao algoritmo 1-NN é considerar, em vez de 1 vizinho mais próximo, os k objetos do conjunto de treino mais próximos do ponto de teste $\mathbf{x}_t$, em que k é um parâmetro do algoritmo. Quando o valor de k é maior que 1, para cada ponto de teste, são obtidos k vizinhos. Cada vizinho vota numa classe. As previsões dos diferentes vizinhos são agregadas de forma a classificar o ponto de teste. Esta agregação é efetuada de forma diferente em problemas de classificação e de regressão.

Em problemas de classificação, em que a classe toma valores num conjunto discreto, cada vizinho vota numa classe. O objeto de teste é classificado na classe mais votada. Formalmente, esse processo é equivalente a: $\hat{f}(\mathbf{x}_t) \leftarrow \text{moda}(f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots, f(\mathbf{x}_k))$, o que é justificado porque *a constante que minimiza a função de custo 0-1 é a moda*.

Em problemas de regressão, podem ser utilizadas duas estratégias, dependendo da função de custo usada. Se a função de custo for minimizar o erro quadrático, a média dos valores obtidos para cada um dos k vizinhos deve ser utilizada, o que pode ser formalmente definido como: $\hat{f}(\mathbf{x}_t) \leftarrow \text{média}(f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots, f(\mathbf{x}_k))$. Se a função de custo a ser minimizada for o desvio absoluto, em vez da média, deve ser utilizada a mediana. Nesse caso, a função passa a ser: $\hat{f}(\mathbf{x}_t) \leftarrow \text{mediana}(f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots, f(\mathbf{x}_k))$. A justificativa para esses procedimentos é porque a média é a constante que minimiza o erro quadrático, enquanto a constante que minimiza o desvio absoluto é a mediana.

A seguir será apresentado um exemplo ilustrativo simples do uso do algoritmo dos vizinhos mais próximos para diferentes valores de k.

Considere o problema da Figura 4.4 (trata-se do mesmo conjunto de dados apresentado anteriormente na Figura 4.2). Para $k = 3$, o objeto de teste seria classificado como pertencendo à classe "doente", enquanto para $k = 5$ o objeto de teste seria classificado como pertencendo à classe "saudável".

A escolha do valor de k mais apropriado para um problema de decisão específico pode não ser trivial. O valor de k é definido pelo utilizador. Frequentemente, o valor de k é pequeno e ímpar: $k = 3, 5, \ldots$ Em problemas de classificação, não é usual utilizar $k = 2$ ou valores pares, para evitar empates.

Duas estratégias referidas na literatura consistem em:

- Estimar k por validação cruzada (ver Capítulo 9).
- Associar um peso à contribuição de cada vizinho.

Neste último caso, a contribuição de cada um dos k vizinhos é pesada de forma inversamente proporcional à distância ao ponto de teste. Dessa forma, é possível utilizar $k = n$ (todos os objetos de treino).

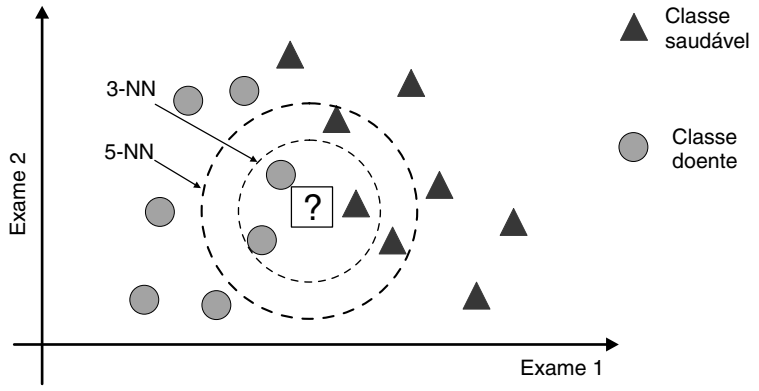


Figura 4.4 *Impacto do valor de k no algoritmo k-NN.*

- Em problemas de classificação:
  - Moda ponderada: $y_t = \arg\max_{c \in Y} \sum_{i=1}^{k} w_i I(c, y_i)$, com $w_i = \frac{1}{d(\mathbf{x}_t, \mathbf{x}_i)}$ e $I(a,b)$ é uma função que retorna 1 se e só se $a = b$;
- Em problemas de regressão:
  - Média ponderada: $y_t = \frac{\sum_{i=1}^{k} w_i y_i}{\sum w_i}$ com $w_i = \frac{1}{d(\mathbf{x}_t, \mathbf{x}_i)}$

Em que y_i é a classe do exemplo x_i, w_i é o peso associado ao exemplo x_i e c é a classe com maior moda ponderada.

4.4 Discussão: Vantagens e Desvantagens

Nesta seção serão analisados os principais aspetos do algoritmo k-NN, salientando seus aspetos positivos e negativos.

4.4.1 Aspetos Positivos

O algoritmo k-NN representa um dos paradigmas mais conhecidos da aprendizagem indutiva: *Objetos com caraterísticas semelhantes pertencem ao mesmo grupo*. O k-NN é um algoritmo baseado em memória, ou um algoritmo preguiçoso, porque toda a computação é adiada até à fase de classificação, já que o processo de aprendizagem consiste apenas em memorizar os objetos.

- O algoritmo de treino é simples, consiste em memorizar os objetos de treino.

- O k-NN constrói aproximações locais da função objetivo, diferentes para cada novo dado a ser classificado. Essa caraterística pode ser vantajosa quando a função objetivo é muito complexa, mas ainda pode ser descrita por uma coleção de aproximações locais de menor complexidade (Mitchell, 1997).
- É aplicável mesmo em problemas complexos.
- O algoritmo é naturalmente incremental: quando novos exemplos de treino estão disponíveis, basta armazená-los na memória.

Um dos aspetos mais relevantes deste algoritmo está relacionado com o seu comportamento no limite. Se forem considerados:

- e: erro do classificador Bayes ótimo;
- $e_{1nn}(\mathbf{D})$: erro do 1-NN;
- $e_{knn}(\mathbf{D})$: erro do k-NN.

Duda et al. (2001) provam os seguintes teoremas:

- $lim_{n\to\infty} e_{1nn}(\mathbf{D}) \leq 2 \times e$;
- $lim_{n\to\infty,k->n} e_{knn}(\mathbf{D}) = e$.

Ou seja, para um número infinito de objetos, o erro do 1-NN é majorado pelo dobro do erro do Bayes ótimo, e o erro do k-NN tende para o erro do Bayes ótimo.

4.4.2 Aspetos Negativos

Também por ser um algoritmo *lazy*, o algoritmo dos vizinhos mais próximos não obtém uma representação compacta dos objetos. De fato, não se tem um modelo explícito e compacto a partir dos dados. A fase de treino requer pouco esforço computacional. No entanto, classificar um objeto de teste requer calcular a distância desse objeto a todos os objetos de treino. Assim, a predição pode ser custosa, e para um conjunto grande de objetos de treino esse processo pode ser demorado. Como todos os algoritmos baseados em distâncias, é afetado pela presença de atributos redundantes e de atributos irrelevantes.

Outro problema do k-NN está relacionado com a dimensionalidade dos exemplos. O espaço definido pelos atributos de um problema cresce exponencialmente com o número de atributos, ou seja, o número de atributos define o número de dimensões do espaço. O ponto que está mais perto de outro pode estar muito distante em problemas de alta dimensionalidade. Para ilustrar as dificuldades levantadas pela dimensionalidade de um problema, considere 100 pontos com distribuição uniforme:

- Num quadrado cujo lado mede 1 unidade
- Num cubo cujo lado mede 1 unidade
- ...

Calculando a distância média entre dois pontos, obtemos:

Núm. dimensões	Distância média
2	0,494
3	0,647
4	0,772
5	0,875
...	
10	1,280

O que se observa é um aumento da distância média entre dois pontos quaisquer. Ou seja, a densidade diminui e o conjunto de dados fica esparso, rarefeito. Para 10 dimensões, a distância média é maior que o tamanho do lado do hipercubo! Quando a dimensão aumenta linearmente, para manter a mesma densidade de pontos, é necessário aumentar de forma exponencial o número de pontos. Beyer et al. (1999) mostram que, sob um amplo conjunto de condições (muito mais amplo do que dimensões independentes e identicamente distribuídas), com o aumento da dimensionalidade, a distância ao vizinho mais próximo aproxima-se da distância ao vizinho mais afastado. A dimensionalidade de um problema pode afetar de forma negativa o desempenho dos algoritmos baseados em distâncias. Uma das formas para reduzir o impacto da dimensionalidade de um problema consiste em selecionar um subconjunto de atributos relevantes para o problema tratado. No Capítulo 3 são apresentados vários algoritmos para seleção de atributos.

4.5 Desenvolvimentos

Uma grande parte dos trabalhos de investigação relacionados com o algoritmo *k*-NN investiga a redução do espaço do problema. Já foi salientado que o *k*-NN é um algoritmo lento no processo de classificação de exemplos de teste. Uma das possibilidades para minimizar esse inconveniente consiste em obter um subconjunto de exemplos representativos. Por exemplo, eliminando objetos redundantes, ou eliminando objetos em que todos os vizinhos são da mesma classe. Outra possibilidade consiste em eliminar objetos com ruído, por exemplo eliminando objetos em que todos os vizinhos são de outra classe.

Aha et al. (1991) apresentaram vários algoritmos para selecionar os objetos mais relevantes, designados por protótipos, para o problema de aprendizagem, de forma a reter em memória apenas esses objetos. As versões do algoritmo Edit *k*-NN para eliminação sequencial, Algoritmo 4.2, e inserção sequencial, Algoritmo 4.3, são dois exemplos de algoritmos que armazenam apenas protótipos. No primeiro caso (Algoritmo 4.2), o algoritmo começa com todos os objetos e vai descartando os objetos que são corretamente classificados pelo conjunto de protótipos atual. No segundo (Algoritmo 4.3), o conjunto de protótipos é inicialmente vazio. Os objetos que são incorretamente classificados pelo conjunto de protótipos são acrescentados a esse conjunto.

A procura linear realizada pelo algoritmo *k*-NN é ineficiente para grandes conjuntos de dados. Estruturas de indexação mais sofisticadas podem permitir uma procura mais efi-

Algoritmo 4.2 Algoritmo para *Edit k*-NN: eliminação sequencial

Entrada: Um conjunto de treino $\mathbf{D} = \{(\mathbf{x}_i, y_i), i = 1, \ldots, n\}$
Saída: Um conjunto de treino $\mathbf{D}' = \{(\mathbf{x}_i, y_i), i = 1, \ldots, m; m < n\}$

```
para cada exemplo (x_i, y_i) ∈ D faça
    se (x_i, y_i) é corretamente classificado por D\{(x_i, y_i)} então
        /* Remove (x_i, y_i) de D */ ;
        D ← D\{(x_i, y_i)} ;
    fim
fim
Retorna: D;
```

Algoritmo 4.3 Algoritmo para *Edit k*-NN: inserção sequencial

Entrada: Um conjunto de treino $\mathbf{D} = \{(\mathbf{x}_i, y_i), i = 1, \ldots, n\}$
Saída: Um conjunto de treino $\mathbf{D}' = \{(\mathbf{x}_i, y_i), i = 1, \ldots, m; m < n\}$

```
D' ← {} ;
para cada exemplo (x_i, y_i) ∈ D faça
    se (x_i, y_i) é incorretamente classificado por D' então
        /* Acrescenta (x_i, y_i) a D' /* ;
        D' ← D' ∪ {(x_i, y_i)} ;
    fim
fim
Retorna: D'
```

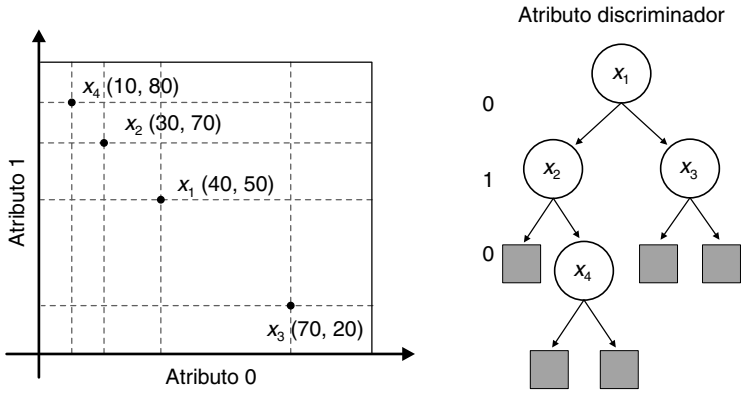
ciente, acelerando a classificação de novos objetos. Uma alternativa consiste na utilização de árvores de procura binária multidimensionais (*kd-trees*, do inglês *k-dimensional trees*) (Bentley, 1975) como estrutura de dados.

Uma *kd-tree* particiona um espaço de busca k-dimensional utilizando os k eixos do sistema de coordenadas. Ela faz isso definindo, recursivamente, o particionamento do espaço dos dados em subespaços disjuntos. No contexto de ECD, cada dimensão representa um atributo de entrada.

Seja um conjunto de dados formado por n objetos. Uma *kd-tree* representando esse conjunto possui n nós, um para cada objeto. A cada nó é associado um atributo discriminador. Todos os nós no mesmo nível da árvore possuem o mesmo discriminador. O atributo discriminador indica qual o atributo utilizado pelo nó para separar os objetos.

A Figura 4.5 ilustra uma 2d-*tree* representando um conjunto formado por quatro objetos, $\mathbf{x_1}$, $\mathbf{x_2}$, $\mathbf{x_3}$ e $\mathbf{x_4}$. O nó raiz da árvore armazena o objeto x_1. Como o atributo discrimi-

nador associado ao nó raiz atual é o atributo 0, o nó raiz da subárvore esquerda do nó raiz atual armazena o objeto com valor para o atributo 0 menor que o valor do atributo 0 do nó raiz, que maximiza a separação entre os objetos que satisfazem essa restrição. Neste caso, foi selecionado o objeto $\mathbf{x_2}$. Um princípio semelhante é utilizado para escolher o objeto que será a raiz da subárvore à direita do nó raiz atual. Este procedimento prossegue de forma recursiva até que cada objeto seja representado por um nó da árvore. A árvore final vai facilitar a procura de objetos no espaço *k*-dimensional.



(a) Particionamento do espaço dos dados

(b) Estrutura da *kd-tree* resultante

Figura 4.5 *Os pontos mostrados em 4.5(a) identificam objetos armazenados como nós internos na kd-tree e são utilizados para particionar o espaço de procura.*

4.6 Raciocínio Baseado em Casos

O raciocínio baseado em casos (RBC) é uma metodologia de ECD para a resolução de problemas fundamentada na utilização de experiências passadas. Assim, um sistema de RBC procura resolver um novo problema através da recuperação de problemas semelhantes, previamente solucionados, de uma memória ou base de casos (BC) e adaptação da solução utilizada no problema recuperado para resolver o novo problema.

RBC difere de outros paradigmas de ECD nos seguintes aspetos (Aamodt e Plaza, 1994):

- Enquanto outros paradigmas utilizam conhecimento geral do domínio ou constroem relações entre problemas e soluções, RBC é capaz de utilizar conhecimento específico de problemas vistos anteriormente.
- RBC possibilita de forma natural a aprendizagem incremental, pela atualização da

BC sempre que um novo problema é resolvido, tornando o novo conhecimento disponível para utilização futura.

4.6.1 Representação de Casos

O desempenho de um sistema de RBC depende da estrutura e conteúdo de sua base de casos. Para a construção de uma base de casos, é necessário decidir o que armazenar num caso, encontrar uma estrutura apropriada para descrever o conteúdo dos casos e definir como os casos devem ser organizados e indexados, para possibilitar a recuperação rápida e a reutilização eficaz de soluções anteriores.

Um caso pode representar diferentes tipos de conhecimento e assumir distintas formas de representação. Uma forma simples de representar casos é através de um conjunto de pares atributo-valor. Este conjunto é dividido em dois subconjuntos. O primeiro possui os atributos que descrevem o problema e o segundo, os atributos relacionados à sua solução.

A Figura 4.6 ilustra um exemplo de BC e de um novo caso. Nessa figura, a base de casos armazena casos relacionados ao problema de escolha de pacotes de viagens. Cada caso tem a descrição de um problema que ocorreu no passado, a descrição dos requisitos de um cliente para um pacote, e a solução utilizada para resolver o problema, com a sugestão de local a ser visitado, meio de transporte para o local, acomodação e opção para refeições. O novo caso descreve um conjunto de requisitos para um novo pacote de viagem. Utilizando essa descrição, um sistema de RBC recupera o caso mais semelhante e adapta-o de forma a sugerir uma solução para esse novo problema. É importante observar que sistemas de RBC podem recuperar e adaptar um ou mais casos.

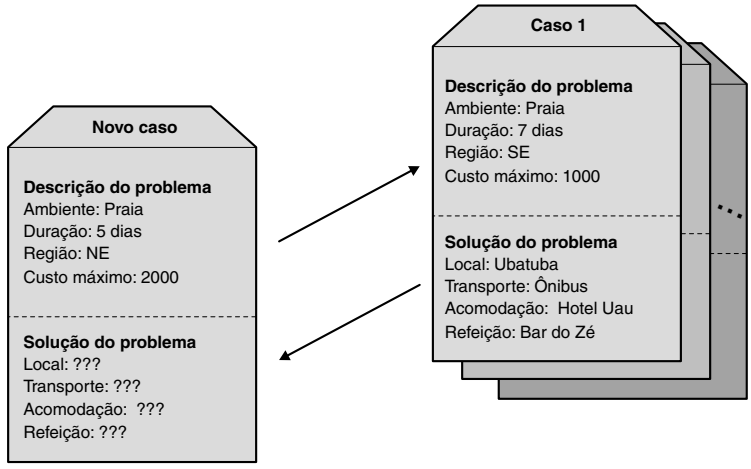


Figura 4.6 *Exemplo de um novo caso e de uma BC.*

Um aspeto importante num sistema de RBC é a forma utilizada para a indexação de casos. No processo de indexação, os casos armazenados recebem índices, que são uti-

lizados para futuras comparações e recuperações de casos (Aamodt e Plaza, 1994). A definição dos índices tem em consideração os aspetos dos casos considerados relevantes para a recuperação de casos similares da BC. Em geral, os índices são um subconjunto dos atributos utilizados para a descrição dos problemas. Os índices precisam ser genéricos o suficiente para facilitar a identificação de casos similares, mas não tão genéricos para evitar a recuperação de casos que tenham pouca relação com o novo caso.

A forma como os casos são armazenados na base de casos influencia a facilidade de sua recuperação. Dois modelos de memória são geralmente utilizados: *memória plana* e *estrutura hierárquica* (Malek e Amy, 1994). Na primeira, todos os casos são armazenados num mesmo nível. Na segunda, os casos são estruturados em forma de árvore.

4.6.2 O Ciclo de Raciocínio Baseado em Casos

Um modelo frequentemente utilizado para descrever as etapas de um sistema RBC é o ciclo de RBC proposto por Aamodt e Plaza (1994), ilustrado na Figura 4.7. Esse ciclo compreende quatro etapas principais:

1. **Recuperação**: Recuperação do caso armazenado na BC mais semelhante ao novo problema apresentado.
2. **Reutilização**: Adaptação da solução do caso recuperado. A solução do problema recuperado é geralmente utilizada como ponto de partida para propor uma solução para o novo problema. Esta etapa também é denominada *adaptação de casos*.
3. **Revisão**: A solução adaptada é revista pelo utilizador para aferir a sua relevância para a resolução do novo problema.
4. **Retenção**: Caso, após a etapa de revisão, a solução adaptada seja considerada relevante, o novo problema, juntamente com essa solução, pode ser armazenado na BC.

A sequência de etapas no ciclo de RBC pode ser resumida da seguinte forma: a descrição de um novo problema define um novo caso. Este novo caso é utilizado para a recuperação de um caso entre aqueles armazenados na coleção de casos previamente vistos (BC). A solução utilizada para resolver o caso recuperado é adaptada de forma a prover uma possível solução para o problema inicial. A solução proposta é avaliada através de um processo de revisão, que pode ocorrer pela sua aplicação no mundo real, por meio de uma simulação, pela avaliação do utilizador ou pelo uso do conhecimento da própria BC. Caso a nova solução seja considerada válida, esta, juntamente com a descrição do problema associado, pode ser armazenada na BC. Desta forma, esta solução pode ser utilizada futuramente para resolver novos casos (Aamodt e Plaza, 1994).

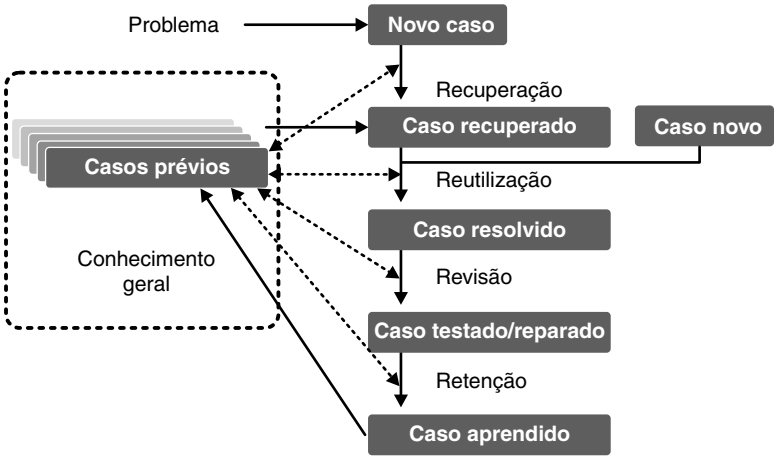


Figura 4.7 *Ciclo de raciocínio baseado em casos (Aamodt e Plaza, 1994).*

4.7 Considerações Finais

Neste capítulo foram descritas técnicas de ECD que se baseiam no cálculo de distância entre objetos. Para isso, foram apresentadas variações do algoritmo dos k vizinhos mais próximos e a metodologia de RBC.

O algoritmo k-NN é um dos principais algoritmos utilizados pela comunidade de ECD, por ser simples e apresentar uma boa taxa de acerto preditiva em vários conjuntos de dados. O desempenho desse algoritmo sofre grande influência do valor de k e da medida de distância utilizada. Existem trabalhos que propõem técnicas para ajuste automático do valor de k. Algumas variações detse algoritmo foram apresentadas neste capítulo.

O RBC pode amenizar alguns problemas encontrados noutros paradigmas de ECD. No entanto, deve ser observado que RBC não oferece uma resposta para todos os problemas encontrados em ECD. Existem situações em que o RBC não pode ser aplicado na resolução de problemas, e situações em que RBC não constitui a solução mais adequada e situações em que RBC pode ser aplicado em conjunto com outras técnicas.

De acordo com Main et al. (2001), Malek (2001), Wiratunga et al. (2002), Jarmulak et al. (2001), Plaza e Arcos (2002), Prentzas e Hatzilygeroudis (2002) e Falkman (2002), as atividades de investigação em RBC focam em problemas de adaptação de casos, utilização de métricas de adaptabilidade durante a recuperação dos casos, sistemas híbridos que combinam RBC com outras técnicas de ECD, manutenção da eficiência do sistema durante seu uso e manutenção do conhecimento armazenado na BC.

Capítulo 5

Métodos Probabilísticos

Outra forma de lidar com tarefas preditivas em ECD, principalmente quando as informações disponíveis são incompletas ou imprecisas, é por meio do uso de algoritmos baseados no teorema de Bayes: os métodos probabilísticos Bayesianos. Os métodos probabilísticos Bayesianos assumem que a probabilidade de um evento A dado um evento B não depende apenas da relação entre A e B, mas também da probabilidade de observar A independentemente de observar B (Duda et al., 2001). A probabilidade de ocorrência do evento A pode ser estimada pela observação da frequência com que este evento ocorre. De forma semelhante, é possível estimar a probabilidade de que um evento B ocorra dado que foi observado o evento A, $P(B|A)$. Nos métodos probabilísticos para problemas de decisão, o objetivo é estimar $P(A|B)$, onde A representa a classe e B o valor observado dos atributos para um exemplo de teste.

Assuma que $P(A)$ representa a probabilidade de ocorrência de uma determinada doença, e $P(B)$ representa a probabilidade de um doente ter um determinado resultado num exame de raios X. Para um determinado paciente, a variável A não é observável. Perante a evidência do resultado do exame de raio X, podemos inferir o valor mais provável de A estimando $P(A|B)$. O teorema de Bayes mostra como calcular $P(A|B)$ utilizando a probabilidade *a priori* da classe, $P(A)$, e a verosimilhança de B dado A, ou seja $P(B|A)$.

Os possíveis valores do conjunto de atributos de entrada definem o espaço amostral (Ω). A probabilidade de ocorrência de um evento E, $P(E)$ (por exemplo, pacientes cujo resultado num exame foi positivo) satisfaz os axiomas de Kolmogorof (Pestana e Velosa, 2002):

- $0 \leq P(E)$;
- Se Ω é o espaço de eventos, então $P(\Omega) = 1$;
- Se A e B são eventos disjuntos, então $P(A \cup B) = P(A) + P(B)$.

A partir destes axiomas e definições, é possível derivar a *lei da probabilidade total*, que afirma que se B_1, B_2, ..., B_n formam uma partição em Ω, então, para qualquer evento

A, tem-se que:

$$P(A) = \sum_{i=1}^{n} P(A|B_i) \times P(B_i)$$

É possível também derivar a *lei da probabilidade condicional*:

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

Esses teoremas permitem deduzir o teorema de Bayes, tendo em conta que:

$$\begin{aligned} P(A \cap B) &= P(B \cap A) \\ P(A|B)P(B) &= P(A \cap B) = P(B|A)P(A) \\ P(A|B) &= \frac{P(B|A)P(A)}{P(B)} \end{aligned}$$

Na próxima seção, é feita uma introdução à aprendizagem Bayesiana, à representação de informação usando modelos probabilísticos gráficos, e à utilização do teorema de Bayes em inferência. A Seção 5.2 descreve um dos classificadores Bayesianos mais populares: o *naive* Bayes. Por último, na Seção 5.3, são apresentadas redes Bayesianas para classificação.

5.1 Aprendizagem Bayesiana

Para exemplificar como os métodos probabilísticos podem ser utilizados em ECD, considere o seguinte cenário: Suponha que a probabilidade de observar alguém com uma dada doença é de 8%. Existe um teste para o diagnóstico dessa doença, cujo resultado possui um grau de incerteza. Sabe-se que em 75% dos casos em que a doença foi confirmada, o resultado do teste foi positivo, e que em 96% dos casos em que o paciente não tinha a doença, o resultado do teste foi negativo.

Como é possível representar essa informação? A doença pode ser considerada uma variável aleatória com dois valores possíveis: presente e ausente. O resultado do teste também tem dois valores mutuamente exclusivos: positivo ou negativo. É fácil observar que o valor da *Doença* influencia o valor do *Teste*, mas o oposto não é verdade. É útil representar essa informação sob a forma de um grafo.

A Figura 5.1 apresenta o grafo que representa a informação do problema descrito. Neste grafo, os nós representam variáveis ou atributos e as arestas representam a influência entre variáveis. A informação gráfica é qualitativa, pois a direção da aresta diz-nos que o valor da variável *doença* influencia o valor da variável *teste*. Também contém informação

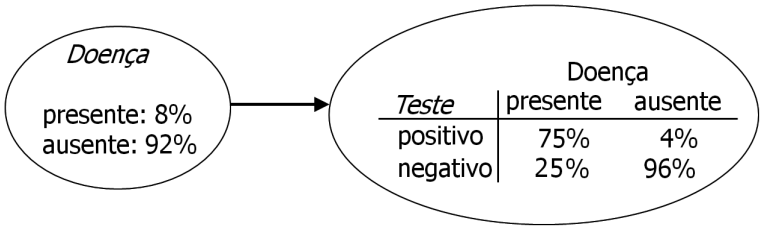


Figura 5.1 *Modelo gráfico probabilístico para representar a informação do problema médico. A figura mostra o modelo qualitativo e o modelo quantitativo.*

quantitativa: a probabilidade *a priori* de observar a doença:

$$P(Doença = \text{presente}) = 0,08$$
$$P(Doença = \text{ausente}) = 0,92$$

A partir de experiências passadas na utilização do teste, é possível inferir as probabilidades condicionais para a variável *Teste*:

$$P(Teste = \text{positivo}|Doença = \text{presente}) = 0,75$$
$$P(Teste = \text{negativo}|Doença = \text{ausente}) = 0,96$$

Esta informação é definida nos nós do modelo gráfico, tal como ilustrado na Figura 5.1. O modelo gráfico probabilístico é constituído pelo modelo qualitativo – um grafo cujos nós representam variáveis – e pelo modelo quantitativo – tabelas com a distribuição das variáveis. A probabilidade de verdadeiros positivos (o *Teste* deu positivo quando a *Doença* está presente) é de 75% (taxa denominada *sensibilidade*) e a probabilidade de verdadeiros negativos (o *Teste* deu negativo quando a *Doença* está ausente) é de 96% (taxa denominada *especificidade*).

Qual é o poder preditivo do *Teste* com respeito à *Doença*? É possível calcular as probabilidades *a priori* para a variável *Teste*. Levando em conta que $P(A) = P(A|B) \times P(B)$, obtemos:

$$\begin{aligned} P(Teste &= \text{positivo}) = \\ &= P(Teste = \text{positivo}|Doença = \text{presente}) \times P(Doença = \text{presente}) + \\ &+ P(Teste = \text{positivo}|Doença = \text{ausente}) \times P(Doença = \text{ausente}) \\ &= 0,75 \times 0,08 + 0,04 \times 0,92 = 0,0968 \end{aligned}$$

$$\begin{aligned} P(\textit{Teste} &= \text{negativo}) = \\ &= P(\textit{Teste} = \text{negativo}|\textit{Doença} = \text{presente}) \times P(\textit{Doença} = \text{presente}) + \\ &+ P(\textit{Teste} = \text{negativo}|\textit{Doença} = \text{ausente}) \times P(\textit{Doença} = \text{ausente}) \\ &= 0,25 \times 0,08 + 0,96 \times 0,92 = 0,9032 \end{aligned}$$

Considere agora que, para um dado paciente, o resultado do *Teste* foi positivo. Podemos concluir que o paciente está doente? Na aprendizagem Bayesiana, o valor de uma variável aleatória tem uma probabilidade associada. A questão que devemos colocar é: Qual é a probabilidade $P(\textit{Doença} = \text{presente}|\textit{Teste} = \text{positivo})$? O teorema de Bayes é usado para calcular a probabilidade *a posteriori* de um evento, dados a sua probabilidade *a priori* e a verossimilhança do novo dado. Neste exemplo, precisamos inverter a probabilidade $P(\textit{Teste} = \text{positivo}|\textit{Doença} = \text{presente})$. Na próxima seção será visto como isso pode ser feito.

5.1.1 O Problema de Inferência e o Teorema de Bayes

A literatura de reconhecimento de padrões (Duda et al., 2001) e aprendizagem automática (Mitchell, 1997) apresenta diversas propostas para lidar com o problema de aprendizagem num cenário probabilístico. Suponha que $P(y_i|\mathbf{x})$ denota a probabilidade de um exemplo $\mathbf{x}$ pertencer à classe y_i. A função de custo zero-um, que representa o custo de associar $\mathbf{x}$ à classe incorreta, é minimizada se, e somente se, $\mathbf{x}$ é associado à classe y_k para a qual $P(y_k|\mathbf{x})$ é máxima (Duda et al., 2001). Este método é designado por estimativa MAP (do inglês, *Maximum A Posteriori*). Formalmente, a classe que deve ser associada ao exemplo $\mathbf{x}$ é dada pela expressão:

$$y_{MAP} = \arg\max_i \; P(y_i|\mathbf{x}) \tag{5.1}$$

na qual $\arg\max_i$ retorna a classe y_i com maior probabilidade de estar associada a $\mathbf{x}$, que é aquela que possui o valor máximo para $P(y_i|\mathbf{x})$.

Qualquer função que calcula as probabilidades condicionadas $P(y_i|\mathbf{x})$ é referida como uma *função discriminante*, por separar exemplos de classes diferentes. Dado um exemplo $\mathbf{x}$, o teorema de Bayes fornece um método para calcular $P(y_i|\mathbf{x})$:

$$P(y_i|\mathbf{x}) = \frac{P(y_i)P(\mathbf{x}|y_i)}{P(\mathbf{x})} \tag{5.2}$$

O denominador, $P(\mathbf{x})$, pode ser ignorado, uma vez que é o mesmo para todas as classes, não afetando os valores relativos de suas probabilidades. Assumindo que as probabilidades *a priori* das hipóteses $\mathbf{y}_i$ são iguais, a Equação 5.2 pode ser simplificada considerando apenas o termo $P(\mathbf{x}|y_i)$ para calcular a hipótese mais provável. $P(\textit{Dados}|\text{hipótese})$ é designado por verosimilhança, e a hipótese que maximiza $P(\textit{Dados}|\text{hipótese})$ é designada por máxima verosimilhança, que pode ser expressa por:

$$h_{MV} = \arg\max_i \ P(\mathbf{x}|y_i) \tag{5.3}$$

Embora esta regra seja ótima, a sua aplicabilidade é reduzida devido ao grande número de exemplos necessários para calcular, de forma viável, $P(\mathbf{x}|y_i)$. Para superar esse problema, várias hipóteses têm sido propostas. Dependendo das hipóteses propostas, são obtidas diferentes funções discriminantes, conduzindo a diferentes classificadores. Neste capítulo, será estudado um tipo de função discriminante que conduz ao classificador *naive* Bayes.

5.2 O Classificador *Naive* Bayes

Assumindo que os valores dos atributos de um exemplo são independentes entre si dada a classe, $P(\mathbf{x}|y_i)$ pode ser decomposto no produto $P(\mathbf{x}_1|y_i) \times \ldots \times P(\mathbf{x}_d|y_i)$, em que $\mathbf{x}_j$ é o j-ésimo atributo do exemplo $\mathbf{x}$. Com isso, a probabilidade de um exemplo pertencer à classe y_i é proporcional à expressão:

$$P(y_i|\mathbf{x}) \propto P(y_i) \prod_{j=1}^{d} P(\mathbf{x}_j|y_i) \tag{5.4}$$

O classificador obtido pelo uso da função discriminante dada pela Equação 5.4 e pela regra de decisão ilustrada na Equação 5.1 é conhecido como classificador *naive* Bayes. O termo *naive* vem da hipótese de que os valores dos atributos de um exemplo são independentes de sua classe.

A fórmula do *naive* Bayes pode ser expressa de uma forma aditiva. Aplicando logaritmos à Equação 5.4, obtém-se:

$$\log(P(y_i|\mathbf{x})) \propto \log(P(y_i)) + \sum_j \log(P(\mathbf{x}_j|y_i)) \tag{5.5}$$

Para o caso particular de duas classes, a Equação 5.4 pode ser reescrita como:

$$\log \frac{P(y_1|\mathbf{x})}{P(y_2|\mathbf{x})} \propto \log \frac{P(y_1)}{P(y_2)} + \sum_j \log \frac{P(\mathbf{x}_j|y_1)}{P(\mathbf{x}_j|y_2)} \tag{5.6}$$

Nesta nova formulação, o sinal de cada termo indica a contribuição de cada atributo para cada classe. Se o quociente $P(\mathbf{x}_j|y_1)/P(\mathbf{x}_j|y_2)$ é maior que 1, o logaritmo é positivo e o atributo contribui para a predição da classe y_1.

5.2.1 Detalhes de Implementação

Todas as probabilidades necessárias para a obtenção do classificador *naive* Bayes são calculadas a partir dos dados de treino. Para calcular a probabilidade *a priori* de observar a

classe y_i, $P(y_i)$, é necessário manter um contador para cada classe. Para calcular a probabilidade condicional de observar um valor de um atributo dado que o exemplo pertence a uma classe, é necessário distinguir entre atributos nominais e atributos contínuos.

No caso de atributos nominais, o conjunto de valores possíveis é um conjunto enumerável. Para calcular a probabilidade condicional, basta manter um contador para cada valor de atributo por classe. No caso de atributos contínuos, quando o número de valores possíveis é infinito, existem duas possibilidades. A primeira é assumir uma distribuição particular para os valores do atributo, sendo geralmente assumida a distribuição normal. A segunda alternativa consiste em discretizar o atributo numa fase de pré-processamento. Já foi mostrado que a segunda possibilidade produz melhores resultados que a primeira (Dougherty et al., 1995; Domingos e Pazzani, 1997).

Vários métodos para a discretização aparecem na literatura. Uma boa discussão sobre discretização é apresentada em Dougherty et al. (1995). Por exemplo, Domingos e Pazzani (1997) propõem que o número de intervalos seja fixado em $k =$ mínimo$(10,$ número de valores diferentes$)$ intervalos do mesmo tamanho. Depois de discretizar um atributo, podemos utiliar um contador para cada classe e para cada intervalo para calcular a probabilidade condicional $P(Atributo_j|Classe_i)$.

5.2.2 Um Exemplo Ilustrativo

Este exemplo utiliza um conjunto de dados para o problema `balance`, que é apresentado na Figura 5.2. Este conjunto de dados foi gerado para modelar resultados de experiências de psicologia. Neste problema, cada exemplo é classificado numa de três posições de uma balança: a balança está inclinada para a direita, para a esquerda ou equilibrada. Os atributos são o peso do lado esquerdo, a dimensão do braço esquerdo, o peso do lado direito e a dimensão do braço direito. A forma correta para encontrar a classe é o maior valor entre: $Distância_{Esq} \times Peso_{Esq}$ e $Distância_{Dir} \times Peso_{Dir}$. Se esses valores forem iguais, o estado da balança, isto é, a sua classe, é equilibrada.

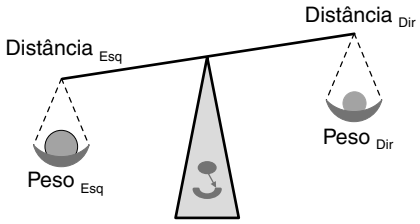


Figura 5.2 *O problema do equilíbrio da balança.*

Na versão do repositório UCI (Frank e Asuncion, 2010) o domínio de todos os atributos para este conjunto de dados, é o conjunto $\{1,2,3,4,5\}$. O conjunto contém 625 exemplos, distribuídos da seguinte forma: em 49 exemplos a balança está equilibrada, em

288 exemplos a balança está inclinada para a esquerda e nos 288 restantes exemplos a balança está inclinada para a direita.

Para calcular as probabilidades *a priori*, $P(Classe_i)$, é necessário contar o número de exemplos para cada classe. Os resultados são apresentados na Tabela 5.1.

Tabela 5.1 *Contagem de valores e probabilidade a priori para as classes*

	Equilibrada	**Esquerda**	**Direita**
Contagem	49	288	288
$P(Classe)$	0,078	0,461	0,461

Para calcular a probabilidade condicional de observar um valor específico de atributo dada a classe, $P(Atributo_j|Classe_i)$, é necessário descobrir o tipo do atributo. Neste problema, todos os atributos são numéricos, pois dizem respeito a distâncias e pesos. Pode-se assumir que o seu domínio é um subconjunto de $\Re$. Sem mais nenhuma informação, a hipótese mais razoável é que estes são normalmente distribuídos. Considerando essa hipótese, para estimar as probabilidades condicionais, é preciso calcular a média e o desvio padrão dos valores dos atributos para cada classe.

Como alternativa, pode-se discretizar os atributos. Neste problema, aplicando a regra $k = \min(10; \text{número de valores diferentes})$, são obtidos cinco intervalos. A Tabela 5.2 apresenta a distribuição de valores para cada atributo, em cada classe.

Tabela 5.2 *Tabelas de distribuição dos valores dos atributos por classe*

	Distribuição normal		**Discretização**				
Peso$_{Esq}$	Média	Desvio padrão	V1	V2	V3	V4	V5
Equilibrado	2,938	1,42	10	11	9	10	9
Esquerda	3,611	1,23	17	43	63	77	88
Direita	2,399	1,33	98	71	53	38	28
Distância$_{Esq}$	Média	Desvio padrão	V1	V2	V3	V4	V5
Equilibrado	2,938	1,42	10	11	9	10	9
Esquerda	3,611	1,22	17	43	63	77	88
Direita	2,399	1,33	98	71	53	38	28
Peso$_{Dir}$	Média	Desvio padrão	V1	V2	V3	V4	V5
Equilibrado	2,938	1,42	10	11	9	10	9
Esquerda	2,399	1,33	71	53	38	28	17
Direita	3,611	1,22	17	43	63	77	88
Distância$_{Dir}$	Média	Desvio padrão	V1	V2	V3	V4	V5
Equilibrado	2,938	1,42	10	11	9	10	9
Esquerda	2,399	1,33	98	71	53	38	28
Direita	3,611	1,22	17	43	63	77	88

A Figura 5.3 ilustra graficamente a Tabela 5.2. Na primeira linha são apresentadas as distribuições discretizadas para os atributos $Peso_{Esq}$ e $Peso_{Dir}$, assim como a *Classe*.

Podemos observar que para a classe **equilibrado** todas as contagens são similares. Para os atributos $Peso_{Esq}$ e $Distância_{Esq}$, a contagem aumenta para a classe esquerda e diminui para a classe direita. Para os atributos $Peso_{Dir}$ e $Distância_{Dir}$, a contagem aumenta para a classe direita e diminui para a classe esquerda. A segunda linha mostra, para os mesmos atributos, a função densidade de probabilidade por classe, assumindo uma distribuição normal.

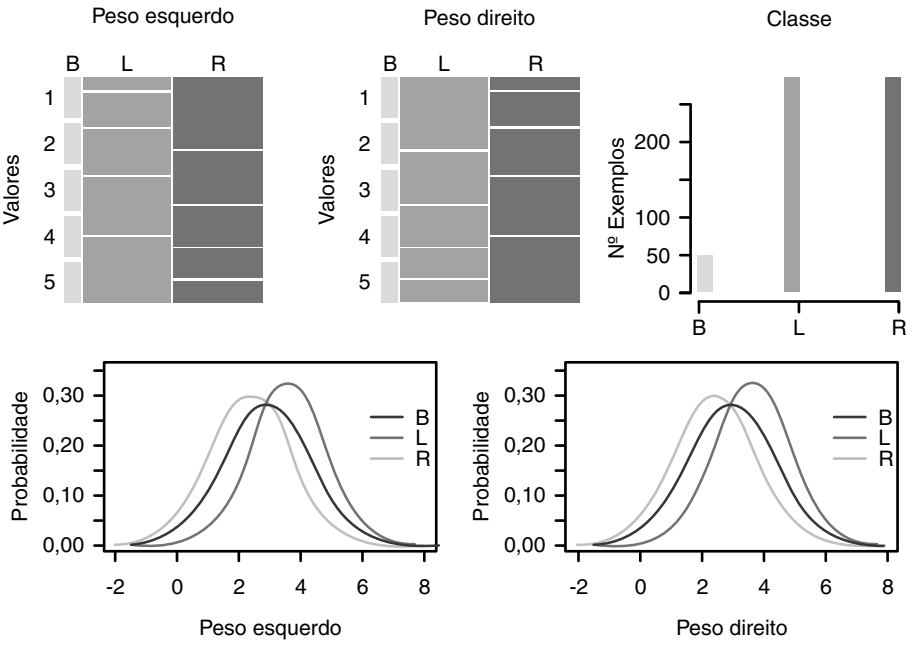


Figura 5.3 *Naive Bayes para o problema da balança.*

5.2.3 Análise do Algoritmo

A superfície de decisão de um classificador *naive* Bayes num problema de duas classes definido por atributos booleanos é um hiperplano, ou seja, a superfície de decisão é linear. Todas as probabilidades exigidas pela Equação 5.4 podem ser calculadas a partir do conjunto de treino numa única passagem. O processo de construir o modelo é bastante eficiente. Outro aspeto interessante do algoritmo é que ele é fácil de implementar de uma forma incremental.

Domingos e Pazzani (1997) mostram que o *naive* Bayes tem um bom desempenho numa grande variedade de domínios, incluindo domínios em que há clara dependência entre os atributos. Os autores argumentam que, em problemas de classificação e para a função de custo $0-1$, um exemplo é corretamente classificado desde que a ordenação das

classes dada pelas estimativas das probabilidades *a posteriori* esteja correta, independentemente de essas estimativas serem (ou não) realistas.

Kononenko (1991) sugere que este classificador é robusto à presença de ruído e atributos irrelevantes. Eles também notaram que as teorias aprendidas são fáceis de compreender pelos especialistas do domínio. Essa observação deve-se ao fato de que o *naive* Bayes resume a variabilidade do conjunto de dados em tabelas de contingência, e assume que estas são suficientes para distinguir entre as classes.

O desempenho do *naive* Bayes não decresce na presença de atributos irrelevantes. Suponha um problema de duas classes, em que o i-ésimo atributo é irrelevante: $P(\mathbf{x}_i|y_1) = P(\mathbf{x}_i|y_2)$. A partir da fórmula do classificador *naive* Bayes:

$$P(y_k|\mathbf{x}_1,\ldots,\mathbf{x}_i,\ldots,\mathbf{x}_d) \propto P(y_k)P(\mathbf{x}_i|y_k)\prod_{l=1}^{i-1}P(\mathbf{x}_l|y_k)\prod_{l=i+1}^{d}P(\mathbf{x}_l|y_k) \tag{5.7}$$

$$P(y_k|\mathbf{x}_1,\ldots,\mathbf{x}_i,\ldots,\mathbf{x}_d) \propto P(y_k|\mathbf{x}_1,\ldots,\mathbf{x}_{i-1},\mathbf{x}_{i+1},\ldots,\mathbf{x}_d) \tag{5.8}$$

O impacto das variáveis redundantes deve ser levado em consideração. Assuma que o $(i-1)$-ésimo e o i-ésimo atributos são redundantes, ou seja, para todas as classes y $P(\mathbf{x}_{i-1}|y) = P(\mathbf{x}_i|y)$.

$$P(y_k|\mathbf{x}_1,...,\mathbf{x}_{i-1},\mathbf{x}_i,\ldots,\mathbf{x}_d) \propto P(y_k)P(\mathbf{x}_{i-1}|y_k)P(\mathbf{x}_i|y_k)\prod_{l=1}^{i-2}P(\mathbf{x}_l|y_k)\prod_{l=i+1}^{d}P(\mathbf{x}_l|y_k) \tag{5.9}$$

$$P(y_k|\mathbf{x}_1,...,\mathbf{x}_{i-1},\mathbf{x}_i,\ldots,\mathbf{x}_d) \propto P(y_k)P(\mathbf{x}_i|y_k)^2\prod_{l=1}^{i-2}P(\mathbf{x}_l|y_k)\prod_{l=i+1}^{d}P(\mathbf{x}_l|y_k) \tag{5.10}$$

5.2.4 Desenvolvimentos

Foram desenvolvidas várias técnicas para melhorar o desempenho do classificador *naive* Bayes. Algumas dessas técnicas aplicam diferentes classificadores *naive* Bayes para diferentes regiões do espaço de entrada. Por exemplo, Langley (1993) apresentou um algoritmo *naive* Bayes que, recursivamente, constrói uma hierarquia das descrições dos conceitos probabilísticos. Kohavi (1996) apresentou uma árvore de *naive* Bayes. É um algoritmo híbrido que gera uma árvore de decisão univariada regular, cujas folhas contêm um classificador *naive* Bayes. O classificador associado a cada nó folha é construído a partir de exemplos que levam a esse nó. A proposta retém a interpretabilidade do *naive* Bayes e das árvores de decisão, resultando num classificador que frequentemente supera ambos os constituintes, especialmente em grandes conjuntos de dados.

Outras técnicas constroem novos atributos que refletem interdependências entre atributos originais. Por exemplo, Kononenko (1991) apresentou um classificador semi-*naive* Bayes. O classificador procura combinar pares de atributos, fazendo um atributo produto-cruzado, baseado em testes estatísticos para independência. A avaliação experimental foi

não conclusiva. Noutro exemplo, Pazzani (1996) apresentou um classificador Bayesiano construtivo. Para isso, emprega um modelo encapsulado (John et al., 1994) para encontrar os melhores atributos do produto cartesiano a partir de atributos nominais existentes. O classificador também considera a eliminação de atributos existentes. Notou-se com isso uma melhoria no classificador *naive* Bayes.

Técnicas que abordam o problema da presença de atributos contínuos estão também presentes na literatura. John (1997) apresentou o *Bayes Flexível* que utiliza, para atributos contínuos, uma estimativa de densidade de funções *kernel* (em vez de uma única hipótese gaussiana), mas retém a hipótese de independência. Para cada atributo contínuo, a densidade estimada é obtida pela expressão $P(\mathbf{x}_t|y_i) = \frac{1}{n}\sum_i N(\mathbf{x}_t, \mathbf{x}_i, \sigma_c)$, em que n representa o número de exemplos de treino da classe y_i, e σ_c representa o tamanho de banda do *kernel*. Desta forma, $P(\mathbf{x}_t|y_i)$ é obtido agregando sobre todos os exemplos de treino da classe y_i.

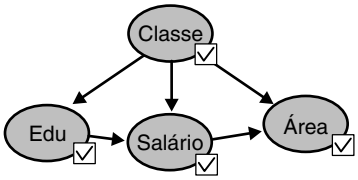
A avaliação experimental em conjuntos de dados do repositório do UCI mostra que o Bayes flexível alcança, em muitos domínios, taxas de acerto preditivas significativamente maiores que o *naive* Bayes. Gama (2000) apresentou um algoritmo *Linear Bayes* que utiliza uma distribuição normal multivariada para cada classe para calcular a verosimilhança $P(\mathbf{x}_i, \ldots, \mathbf{x}_j|y_i)$, em que $\mathbf{x}_i, \ldots, \mathbf{x}_j$ representa o conjunto de atributos contínuos. Esta estratégia mostrou-se melhor do que o *naive* Bayes usando discretização ou uma distribuição gaussiana univariada.

5.3 Redes Bayesianas para Classificação

Foi previamente mencionado a incapacidade do classificador *naive* Bayes em lidar com inter-dependências entre atributos. Recorde-se que duas variáveis aleatórias (atributos) X, Y são *independentes* quando $P(X,Y) = P(X) \times P(Y)$, ou seja, conhecer o valor de uma variável não traz informação sobre o valor da outra variável, o que implica $P(X|Y) = P(X)$. Esta seção tem como tema central a *independência condicional*: casos em que existe uma relação estatística entre duas variáveis quando uma terceira variável é conhecida. Formalmente, X é condicionalmente independente de Y dado Z se $P(X|Y,Z) = P(X|Z)$.

Os modelos gráficos probabilísticos, ou redes Bayesianas (Pearl, 1988), utilizam o conceito de *independência condicional* entre variáveis para obter um equilíbrio entre o número de parâmetros a calcular e a representação de dependências entre as variáveis. Estes modelos representam a distribuição de probabilidade conjunta de um grupo de variáveis aleatórias num domínio específico.

Formalmente, seja $\mathbf{x} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ um conjunto de variáveis aleatórias para um dado domínio. Uma rede Bayesiana (RB) sobre $\mathbf{x}$ é uma tupla (S, Θ_S) em que o primeiro componente, a *estrutura da rede* S, é um *grafo acíclico direcionado* (DAG, do inglês, *Directed Acyclic Graph*) cujos *nós* representam as variáveis aleatórias e as *arestas* representam dependências diretas entre variáveis. Um arco entre dois nós, denota *influência* ou *correlação*. O conjunto de variáveis aleatórias (nós do DAG) que influenciam uma variável x_i é usualmente designado por **Pais** de x_i. A segunda componente Θ_S é o conjunto de *tabelas de probabilidade condicional*.



	Classe	Não		Sim	
	Edu	Não	Sim	Não	Sim
Salário	Baixo	0,6	0,4	0,33	0,66
	Alto	0,4	0,4	0,66	0,33

Figura 5.4 *A Figura mostra o modelo qualitativo – um grafo cujos nós representam variáveis – e o modelo quantitativo para a variável Salário – tabelas com a distribuição de probabilidades dos valores da variável Salário dado o valor das variáveis que a influenciam.*

Assumindo que as variáveis são discretas, cada $P(\mathbf{x}_i|\mathbf{Pai}_i) \in \Theta_S$ representa a *tabela de probabilidade condicional* (TBC) sobre os valores de x_i dados os valores do seu pai $\mathbf{Pai}_i$. Além disso, o DAG S satisfaz a condição de Markov: *cada nó é independente de todos os seus não descendentes, dados os seus pais em S*. Isto permite que a distribuição de probabilidade conjunta sobre $\mathbf{x}$ seja representada na forma fatorizada: $P(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) = \prod_{i=1}^{n} P(\mathbf{x}_i \mid \mathbf{Pai}_i)$. A Figura 5.4 apresenta um exemplo de uma rede Bayesiana. É apresentado o modelo qualitativo – um grafo cujos nós representam variáveis – e o modelo quantitativo – tabelas com a distribuição de probabilidades da variável *Salário* dadas as variáveis que a influenciam. A fatorização conjunto para as variáveis do modelo é:

$$P(Classe)P(Edu|Classe)P(Salário|Edu,Classe)P(Área|Salário,Classe).$$

Uma rede Bayesiana pode ser utilizada para tarefas de classificação de uma forma relativamente simples. Uma das variáveis é selecionada como atributo alvo, e todas as outras variáveis são consideradas como atributos de entrada. O conjunto de variáveis que influenciam o atributo alvo é designado por *Markov Blanquet*: é constituído pelas variáveis pais da variável alvo, pelos filhos da variável alvo e pelos pais dos filhos da variável alvo. Assim, uma rede RB pode ser utilizada como um classificador que, dado um exemplo $\mathbf{x}$, fornece a distribuição de probabilidade *a posteriori* $P(y \mid \mathbf{x})$ do nó classe $y \in Y$. É possível calcular a probabilidade *a posteriori* $P(y \mid \mathbf{x}, S)$ para cada classe $y \in Y$ marginalizando a distribuição de probabilidade conjunta $P(y, \mathbf{x} \mid S)$ e então retornar a classe $\hat{y}$ que a maximiza:

$$\hat{y} = h_{C\mathcal{RB}}(\mathbf{x}) = \arg \max_{j=1\ldots k} P(y_j, \mathbf{x} \mid S) \tag{5.11}$$

Dado um conjunto de treino, o problema de aprendizagem consiste em selecionar o classificador baseado em RB (CRB), isto é, a hipótese $h_C = (S, \Theta_S)$ que produz a classificação com maior taxa de acerto para dados não conhecidos.

Este problema pode ser resolvido inicialmente pela escolha de um modelo de classe adequado, que define o espaço de possíveis estruturas RB. Em seguida, é selecionada uma *estrutura* dentro desse modelo de classe. Finalmente, os parâmetros são estimados a partir dos dados.

O problema de escolher a estrutura mais apropriada para um determinado problema

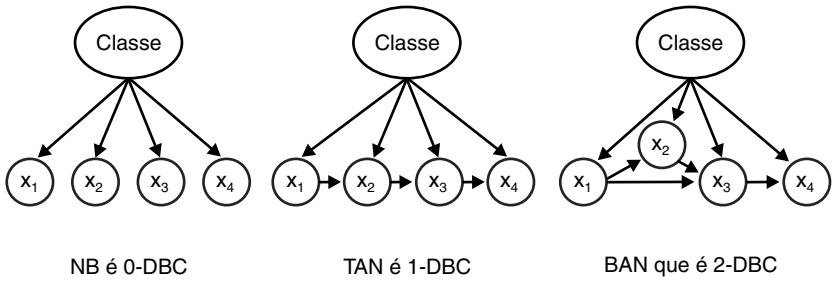


Figura 5.5 *Exemplos de classificadores bayesianos com k-dependências.*

está relacionado com a *seleção de modelos*, uma área da inferência estatística que estuda a seleção, dentre um conjunto de modelos concorrentes, daquele que *"melhor se ajusta"*, em algum sentido, aos dados disponíveis. Uma das propostas consiste em definir uma *função de pontuação* que mede a qualidade de cada hipótese candidata, ou seja, o ajuste do modelo aos dados de treino. As propostas baseadas em pontuação podem ser descritas como um *problema de procura*, em que cada estado no espaço de procura identifica um possível DAG. O método de procura utiliza o valor retornado pela função de pontuação.

O problema da seleção de uma função de pontuação apropriada para aprendizagem de CRBs tem sido alvo de muita atenção (Domingos e Pazzani, 1997; Friedman et al., 1997; Kontkanen et al., 1999). Quando uma RB é induzida para classificação, o objetivo principal é construir um classificador com elevada taxa de acerto preditiva. Para isso, foi sugerido que as estratégias de procura para aprendizagem de CRBs deveriam selecionar entre modelos utilizando pontuações especializadas para classificação (*pontuações supervisionadas*); caso contrário a procura poderia resultar numa escolha subótima (Kontkanen et al., 1999). Deste modo, uma pontuação baseada na distribuição da probabilidade conjunta não será necessariamente ótima em problemas de classificação.

Outro aspeto que também pode influenciar o desempenho dos CRBs induzidos é a seleção de um *modelo de classe* apropriado, que define o espaço de procura e, consequentemente, a complexidade dos CRBs induzidos. A seleção do modelo procura um equilíbrio *viés-variância* a fim de selecionar um modelo com a complexidade apropriada, que é automaticamente regularizado pela função de pontuação (Hastie et al., 2001). Podemos obter o desempenho desejado dos CRBs induzidos se, a cada momento, tentarmos selecionar o modelo de classe apropriado, com a complexidade adequada, para os dados de treino disponíveis.

5.3.1 Classificadores Bayesianos com *k*-Dependências

Os classificadores Bayesianos com *k*-dependências (Sahami, 1996)(*k*-CBDs) representam um enquadramento unificado para todas as classes de CRBs que contêm a estrutura de Bayes simples. Além disso, um *k*-CBD permite que cada atributo tenha no máximo *k* nós atributos como pais. Como ilustrado na Figura 5.5, é possível variar o valor de *k* e obter

classificadores Bayesianos de complexidade crescente, que se movam gradualmente ao longo do espetro de dependências entre atributos. O *naive* Bayes é um 0-CBD e encontra-se no extremo mais restritivo, porque não permite dependências entre atributos. Um classificador TAN (Friedman et al., 1997) é um 1-CBDs (permite, no máximo, um atributo como pai de outro atributo). O BAN (Friedman et al., 1997; Cheng e Greiner, 1999), apresentado na Figura 5.5, é um 2-CBD (tem um máximo duas dependências entre atributos). No extremo mais geral encontra-se o classificador $(n-1)$-CBD, que assume que todos os atributos interagem entre si. A fatoração da probabilidade conjunta $P(Classe, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4)$ das cinco variáveis, codificada pelos modelos apresentados na Figura 5.5, é seguidamente apresentada:

- *Naive* Bayes:
  $P(Classe)P(\mathbf{x}_1|Classe)P(\mathbf{x}_2|Classe)P(\mathbf{x}_3|Classe)P(\mathbf{x}_4|Classe)$
- TAN:
  $P(Classe)P(\mathbf{x}_1|Classe)P(\mathbf{x}_2|\mathbf{x}_1, Classe)P(\mathbf{x}_3|\mathbf{x}_2, Classe)P(\mathbf{x}_4|\mathbf{x}_3, Classe)$
- BAN:
  $P(Classe)P(\mathbf{x}_1|Classe)P(\mathbf{x}_2|\mathbf{x}_1, Classe)P(\mathbf{x}_3|\mathbf{x}_1, \mathbf{x}_2, Classe)P(\mathbf{x}_4|\mathbf{x}_3, Classe)$

Sahami (1996) propôs um algoritmo para indução de k-CBDs que usa o conceito de entropia condicional. Em Castillo e Gama (2005) é proposto um algoritmo de aprendizagem *subida da encosta* (*hill-climbing*) para o mesmo problema. É um algoritmo simples, incremental e de fácil implementação computacional. O algoritmo, cujo pseudocódigo é apresentado no Algoritmo 5.1, é iniciado com a estrutura do *naive* Bayes. Iterativamente, adiciona arestas entre dois atributos que resultam em melhorias máximas na pontuação até que não existam mais melhorias para aquela pontuação, ou até que não seja possível adicionar uma nova aresta. A função de avaliação de modelos é a taxa de erro no conjunto de treino. Alternativamente, pode ser usada a taxa de erro num conjunto de validação. Os resultados obtidos em experiências apresentadas em Sahami (1996), e relacionados com os estudos que utilizam k-CBDs (Blanco et al., 2005; Castillo e Gama, 2005; Webb et al., 2005) mostram que, considerando as dependências entre atributos é possível melhorar os resultados da classificação de Bayes simples. Conhecendo como o desempenho de classificação se altera com o aumento do valor de k, podemos obter uma noção de nível de dependência entre atributos para cada domínio particular. Um resultado interessante foi apresentado em Castillo e Gama (2005), onde é estudada a relação entre o número de exemplos de treino e o valor de k. Neste trabalho, são apresentados resultados experimentais que mostram que, para valores crescentes do número de exemplos de treino, o melhor desempenho é obtido através do aumento do valor de k.

5.4 Considerações Finais

Os modelos gráficos probabilísticos representam a distribuição de probabilidade conjunta de um conjunto de variáveis aleatórias. É possível obter classificadores Bayesianos de

Algoritmo 5.1 O algoritmo subida de encosta para aprendizagem k-CBDs

Entrada: Um conjunto de treino $\mathbf{D} = \{(\mathbf{x}_i, y_i), i = 1, \ldots, n\}$
k: número de dependências admissíveis.
Saída: Um k-DBC com baixo valor de $\texttt{Erro}(S, \mathbf{D})$

/* S é o espaço de possíveis DAGs restritos por k */ ;
Inicialize S com a estrutura Bayes simples;
continuar $\leftarrow$ Verdadeiro ;
enquanto *continuar* **faça**
    Calcular $\texttt{Erro}(S, \mathbf{D})$;
    /* Seja (x', x'') a aresta que minimiza a função de avaliação.*/ ;
    $(x', x'') = \arg\min \texttt{Erro}(S \bigcup (x_i, x_j), \mathbf{D}) \wedge$
    $\wedge |\mathbf{pai}(x_i) \setminus y| < k \wedge\ |\mathbf{pai}(x_j) \setminus y| < k$;
    se *Existe a aresta* $(x', x'') \wedge \texttt{Erro}(S \bigcup (x', x''), \mathbf{D}) < \texttt{Erro}(S, \mathbf{D})$ **então**
        Adicionar a aresta (x', x'') em S ;
    fim
    senão
        continuar $\leftarrow$ Falso ;
    fim
fim
Estimar os parâmetros Θ_S dado S a partir dos dados $\mathbf{D}$;
Retorna: k-DBC=(S, Θ_S);

complexidade crescente, que consideram diferentes graus de dependências entre atributos. O *naive* Bayes é o mais restritivo porque não permite dependências entre atributos. No extremo mais geral, temos classificadores que assumem que todos os atributos interagem entre si. Entre os dois extremos, temos modelos de granularidade crescente. Os modelos gráficos probabilísticos são usados para diferentes tarefas de aprendizagem, desde *previsão*, em que se pretende obter o resultado mais provável para os dados de entrada, até *diagnóstico*, em que se pretende obter as causas mais prováveis para os efeitos observados.

Capítulo 6

Métodos Baseados em Procura

O problema de aprendizagem automática pode ser formulado como um problema de procura num espaço de possíveis soluções. Dada uma linguagem para representar generalizações de exemplos e uma função de avaliação de hipóteses, o problema de aprendizagem pode ser resolvido através de procura no espaço de hipóteses definido pela linguagem de representação. O alvo de estudo deste capítulo são os modelos baseados em árvores (árvores de decisão e regressão) e os modelos baseados em regras.

Os modelos em árvore são designados *árvores de decisão*, no caso de problemas de classificação, e *árvores de regressão* nos problemas de regressão. Quer em árvores de decisão, quer em árvores de regressão, a interpretação dos modelos assim como os algoritmos de indução das árvores são muito semelhantes, pelo que iremos usar o termo *árvores de decisão* de uma forma genérica, explicitando quando necessário.

6.1 Árvores de Decisão e Regressão

Uma árvore de decisão usa a estratégia *dividir para conquistar* para resolver um problema de decisão. Um problema complexo é dividido em problemas mais simples, aos quais recursivamente é aplicada a mesma estratégia. As soluções dos subproblemas podem ser combinadas, na forma de uma árvore, para produzir uma solução do problema original. A vantagem desta proposta prende-se com a sua capacidade em dividir o espaço de instâncias em subespaços, e ajustar cada subespaço recorrendo a diferentes modelos. Esta constitui a ideia base por detrás de algoritmos baseados em árvores de decisão e de regressão tais como: ID3 (Quinlan, 1979), ASSISTANT (Cestnik et al., 1987), CART (Breiman et al., 1984), C4.5 (Quinlan, 1993). Recentemente, diversos pacotes estatísticos, S_{plus}, *Statistica*, *SPSS* (Mattison, 1998), *R* (Ihaka e Gentleman, 1996) e *Microsoft SQL Server* (Seidman, 2001) incorporaram funções que implementam árvores de decisão para problemas de classificação e regressão.

Formalmente, uma árvore de decisão é um grafo acíclico direcionado em que cada nó ou é um *nó de divisão*, com dois ou mais sucessores, ou um *nó folha*:

- Um *nó folha* é rotulado com uma *função*. Usualmente são considerados apenas os valores da variável alvo nos exemplos que chegam a um nó folha. No caso mais simples, a função é a constante que minimiza a função de custo. Em problemas de classificação, e assumindo a função de custo 0-1, essa constante é a *moda*. Em problemas de regressão, a constante que minimiza a função de custo do *erro médio quadrático* é a *média*, enquanto para a função de custo do *desvio absoluto* é a *mediana*.
- Um *nó de divisão* contém um *teste condicional* baseado nos valores do atributo. Na proposta padrão, os testes são univariados: as condições envolvem um único atributo e valores no domínio desse atributo. Exemplos de testes condicionais são:
  - Idade > 18;
  - Profissão $\in \{\text{professor}, \text{estudante}\}$;
  - $0,3 + 0,2 \times \mathbf{x}_1 - 0,5 \times \mathbf{x}_2 \leq 0$.

A Figura 6.1 representa uma árvore de decisão e a divisão correspondente no espaço definido pelos atributos $\mathbf{x}_1$ e $\mathbf{x}_2$. Cada nó da árvore corresponde a uma região nesse espaço. As regiões definidas pelas folhas da árvore são mutuamente exclusivas, e a reunião dessas regiões cobre todo o espaço definido pelos atributos. A interseção das regiões abrangidas por quaisquer duas folhas é vazia. A união de todas as regiões (todas as folhas) é U.

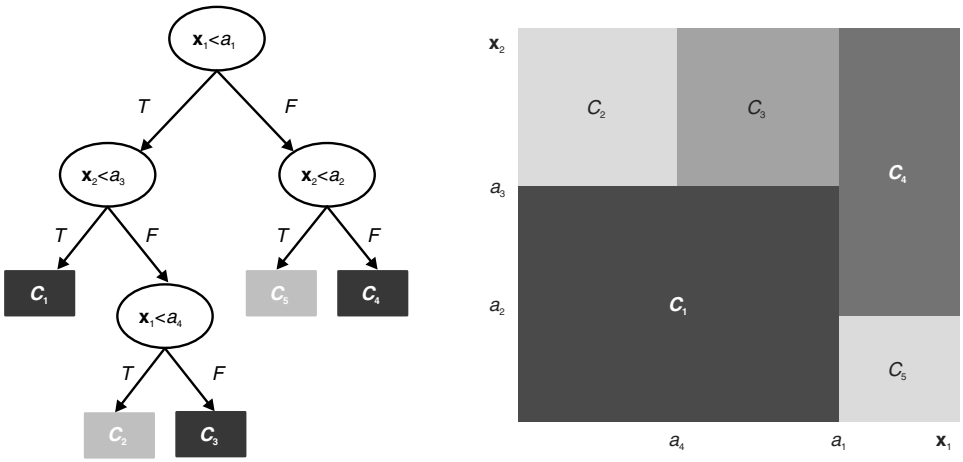


Figura 6.1 *Uma árvore de decisão e as regiões de decisão no espaço definido pelos atributos.*

Uma árvore de decisão abrange todo o espaço de instâncias. Esse fato implica que uma árvore de decisão pode fazer predições para qualquer exemplo de entrada.

O espaço de hipóteses das árvores de decisão enquadra-se dentro do formalismo Forma Normal Disjuntiva (FND). Classificadores gerados por estes sistemas codificam uma FND para cada classe. Para cada FND, as condições ao longo de um ramo (um percurso entre

a raiz e uma folha) são conjunções de condições e os ramos individuais são disjunções. Dessa forma, cada ramo forma uma regra com uma parte condicional e uma conclusão. A parte condicional é uma conjunção de condições. Condições são testes que envolvem um atributo particular, operador (por exemplo =, $\geq$ etc.) e um valor do domínio do atributo. Este tipo de testes corresponde, no espaço de entrada, a um hiperplano que é ortogonal aos eixos do atributo testado e paralelo a todos os outros eixos. As regiões produzidas por estes classificadores são todas hiper-retângulos, conforme pode ser visualizado na Figura 6.1.

6.1.1 Indução de Árvores de Decisão e Regressão

O algoritmo que constrói uma árvore de decisão a partir de dados é muito simples. Os passos principais do algoritmo são descritos no Algoritmo 6.1. A entrada para a função **GeraÁrvore** é um conjunto de dados **D**. No passo 3, o algoritmo avalia o critério de paragem. Se forem necessárias mais divisões do conjunto de dados, é escolhido o atributo que maximiza alguma medida de impureza (passo 5). No passo 7, a função **GeraÁrvore** é recursivamente aplicada a cada partição do conjunto de dados **D**.

Algoritmo 6.1 Algoritmo para construção de uma árvore de decisão

Entrada: Um conjunto de treino $\mathbf{D} = \{(\mathbf{x}_i, y_i), i = 1, \ldots, n\}$
Saída: Árvore de Decisão
1 **/* Função GeraÁrvore(D) */** ;
2 **se** *critério de paragem(***D***) = Verdadeiro* **então**
3 **Retorna:** um nó folha rotulado com a constante que minimiza a função perda ;
4 **fim**
5 Escolha o atributo que maximiza o critério de divisão em **D** ;
6 **para cada** *partição dos exemplos* $\mathbf{D}_i$ *baseado nos valores do atributo escolhido* **faça**
7 Induz uma subárvore $Árvore_i = GeraÁrvore(\mathbf{D}_i)$;
8 **fim**
9 **Retorna:** *Árvore* contendo um nó de decisão baseado no atributo escolhido, e descendentes $Árvore_i$;

O problema de construção de uma árvore de decisão mínima (em termos do número de nós), consistente com um conjunto de exemplos, é um problema *NP completo* (Rivest, 1987). Geralmente, os algoritmos exploram heurísticas que executam localmente uma procura que *olha para a frente um passo*. Uma vez tomada uma decisão, esta nunca é reconsiderada. Este tipo de procura, baseado numa filosofia de *subida de encosta* (*hill-climbing*) sem retrocesso (*backtracking*), é suscetível aos riscos usuais de convergência

para uma solução que é localmente ótima, mas não globalmente ótima. Não obstante, esta estratégia permite construir árvores de decisão em tempo *linear* no número de exemplos.

Nas próximas subseções iremos descrever os aspetos mais importantes deste algoritmo. O foco será direcionado para os dois sistemas mais sucedidos e representativos de árvores de decisão: CART e C4.5.

Regras de Divisão para Classificação

Uma regra de divisão é guiada por uma medida de "*goodness of split*", que indica o quão bem um dado atributo discrimina as classes. Esta regra é usada para selecionar o atributo que maximiza essa medida (cf. passo 5 do Algoritmo 6.1). Uma regra de divisão tipicamente funciona como uma heurística *olha para a frente um passo*. Para cada teste possível, o sistema hipoteticamente considera os subconjuntos dos dados obtidos por aplicação desse teste. O sistema escolhe o teste que maximiza (ou minimiza) uma função heurística sobre os subconjuntos.

Considere um nó t, em que a probabilidade de observar um exemplo da classe c_i é p_i. A probabilidade de observar exemplos de cada classe é dada por $p_1, p_2, ..., p_k$, tal que $\sum p_i = 1$. A impureza do nó t é uma função sobre a proporção da classe daquele nó: $i(t) = \phi(p_1, p_2, ..., p_k)$. Suponha um teste de divisão S que divide os exemplos de treino em dois subconjuntos L e R. A redução na impureza dos testes pode ser medida como:

$\delta(S) = \phi(p_1, p_2, ..., p_k) - P_L \times \phi(p_{1L}, p_{2L}, ..., p_{kL}) - P_R \times \phi(p_{1R}, p_{2R}, ..., p_{kR})$

em que P_L e P_R representam a probabilidade de que um exemplo de t vá para o subconjunto L e R, respetivamente. As caraterísticas gerais de qualquer função de impureza são:

1. Ser simétrica;
2. Ter um máximo quando $p_1 = p_2 = ... = p_k$;
3. Ter um mínimo se $\exists i : p_i = 1$, ou seja, todos os exemplos são de uma classe (i), o que implica que para todas as outras classes j, $p_j = 0$.

Diversos métodos foram descritos na literatura. A maioria deles concorda nos pontos extremos, isto é, que uma divisão que mantém a proporção de classes em todo o subconjunto não tem utilidade, e uma divisão na qual cada subconjunto contém somente exemplos de uma classe tem utilidade máxima. Os casos intermédios podem ser classificados de modo distinto pelas diferentes medidas. Martin (1997) agrupa as medidas nas seguintes categorias das chamadas *funções de mérito*:

1. Medidas de diferença entre a distribuição no nó pai (antes da divisão) e a distribuição nos subconjuntos obtidos por alguma função baseada nas proporções de classe (tal como a entropia). Estas medidas enfatizam a *pureza* dos subconjuntos. CART apelida estas medidas de funções de *impureza*.
2. Medidas da diferença entre os subconjuntos divididos com base em alguma função sobre as proporções de classe (tipicamente a distância ou um ângulo). Estas medidas enfatizam a *disparidade* dos subconjuntos.

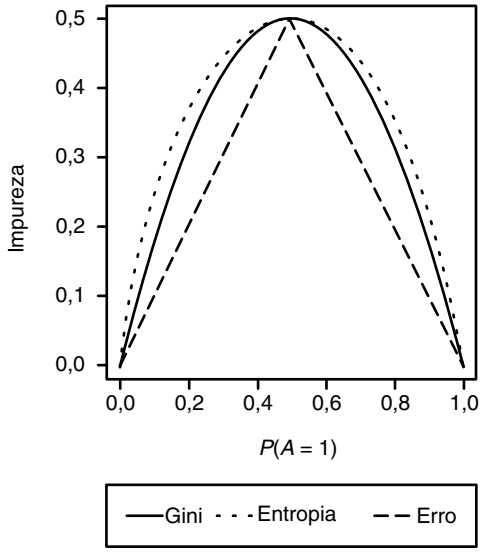


Figura 6.2 *Gráfico da entropia, índice Gini e taxa de erro de uma variável booleana aleatória A.*

3. Medidas estatísticas de independência (tipicamente um teste χ^2) entre as proporções de classe e os subconjuntos divididos. Estas medidas colocam ênfase no peso da evidência, i.e. na *confiança* das predições da classe baseadas no relacionamento do subconjunto.

O remanescente desta seção explica as regras de divisão baseadas no *Ganho de Informação*, usado no C4.5, e no índice Gini, usado no CART.

Ganho de Informação. O conceito fundamental subjacente a esta medida é o conceito de *entropia*. A entropia mede a aleatoriedade de uma variável aleatória (ver Seção 3.1). Suponha uma variável aleatória booleana A. A função entropia, $H(A)$, é $-p \times \log_2(p) - (1-p) \times \log_2(1-p)$, em que p é a probabilidade de observar $A = 0$ e $1-p$ é a probabilidade de observar $A = 1$. A Figura 6.2 mostra o gráfico da entropia para o caso descrito.

No contexto de uma árvore de decisão, a entropia é usada para medir a aleatoriedade (dificuldade para predizer) do atributo alvo. Para cada nó de decisão, o atributo que mais reduz a aleatoriedade da variável alvo será escolhido para dividir os dados. Dado um conjunto de exemplos classificados, qual o atributo que deve ser selecionado como teste de divisão? Os valores de um atributo definem partições no conjunto de exemplos. Para cada atributo, o *ganho de informação* mede a redução na entropia, nas partições obtidas de acordo com os valores do atributo. Informalmente, o ganho de informação é dado pela diferença entre a entropia do conjunto de exemplos original e a soma ponderada da

entropia das partições. A construção da árvore de decisão é guiada pelo objetivo de reduzir a entropia, isto é, a aleatoriedade (dificuldade para predizer) da variável alvo.

Considerando uma árvore de decisão como uma fonte de informação que envia uma mensagem a respeito da classificação de um objeto, e sendo que p e q denotam o número de objetos de duas classes diferentes, o conteúdo esperado da informação da mensagem é:

$$H(p,q) = -\frac{p}{p+q}\log(\frac{p}{p+q}) - \frac{q}{p+q}\log(\frac{q}{p+q}) \tag{6.1}$$

em que a probabilidade de cada possível mensagem é calculada a partir do conjunto de treino. Se o atributo A é selecionado, e assumindo que o domínio de A tem v valores diferentes, a árvore resultante tem um conteúdo de informação esperado de:

$$E(A,p,q) = \sum_{i=1}^{v} \frac{p_i+q_i}{p+q} H(p_i,q_i) \tag{6.2}$$

em que p_i e q_i representam o número de objetos de cada classe na subárvore associada com a partição i, baseada nos valores do atributo A. O ganho de informação (IG) obtido é dado pela seguinte fórmula:

$$IG(A,p,q) = I(p,q) - E(A,p,q) \tag{6.3}$$

A heurística correspondente seleciona o atributo que resulta no máximo ganho de informação para aquele passo.

Muitas vezes um teste num atributo nominal irá dividir os dados em tantos subconjuntos quantos os valores do atributo, embora divisões binárias baseadas na relação de pertença a um subconjunto ($att_i \in \{V_j,...,V_k\}$ e $att_i \notin \{V_j,...,V_k\}$) também sejam possíveis.

Exemplo Ilustrativo. Suponha o conjunto de treino apresentado na Tabela 6.1. O problema de decisão consiste em decidir quando alguém joga, ou não, algum desporto, dadas as condições do tempo. O problema é definido por quatro atributos de entrada: *Tempo*, *Temperatura*, *Humidade* e *Vento*. O conjunto de treino contém 14 exemplos que descrevem observações fatuais do comportamento do indivíduo (coluna *Joga*), dadas as condições do tempo. A primeira variável (atributo), *Tempo*, é qualitativa. O domínio desta variável é o conjunto: {Chuvoso, Ensolarado, Nublado}. As variáveis *Temperatura* e *Humidade* são quantitativas. O seu domínio é um subconjunto de $\Re$. A variável *Vento* é booleana. O domínio da variável alvo, i.e. da variável que queremos prever, é o conjunto: {Não, Sim}.

Qual o atributo que melhor discrimina as classes? Nos exemplos, existem cinco observações em que a variável alvo toma o valor *Não* e nove exemplos em que se observa o valor *Sim*. A entropia da classe para o conjunto de exemplos é, assim, dada por:
$p(Joga = \text{Sim}) = 9/14$
$p(Joga = \text{Não}) = 5/14$
$H(Joga) = -9/14 * \log_2(9/14) - 5/14 * \log_2(5/14) = 0,940$ bit

Tabela 6.1 *Exemplo de conjunto de dados*

Tempo	Temperatura	Humidade	Vento	Joga
Chuvoso	71	91	Sim	Não
Ensolarado	69	70	Não	Sim
Ensolarado	80	90	Sim	Não
Nublado	83	86	Não	Sim
Chuvoso	70	96	Não	Sim
Chuvoso	65	70	Sim	Não
Nublado	64	65	Sim	Sim
Nublado	72	90	Sim	Sim
Ensolarado	75	70	Sim	Sim
Chuvoso	68	80	Não	Sim
Nublado	81	75	Não	Sim
Ensolarado	85	85	Não	Não
Ensolarado	72	95	Não	Não
Chuvoso	75	80	Não	Sim

Calcular o Ganho de Informação para um Atributo Nominal. Considere o atributo *Tempo*. Dividindo o conjunto de treino pelos valores desse atributo, obteremos três partições. Para calcular a informação das partições, devemos estimar (a partir do conjunto de treino) as probabilidades de observar uma classe dado cada valor do atributo. A Tabela 6.2 resume as contagens necessárias. A partir desta tabela, obtemos a entropia de cada partição e o ganho da divisão:

$p(Jogar = \text{Sim}|Tempo = \text{Ensolarado}) = 2/5$
$p(Jogar = \text{Não}|Tempo = \text{Ensolarado}) = 3/5$
$H(Jogar|Tempo = \text{Ensolarado}) = -2/5 * \log_2(2/5) - 3/5 * \log_2(3/5) = 0,971 \text{ bit}$

De modo análogo, para as outras partições obtemos:
$H(Jogar|Tempo = \text{Nublado}) = 0,0 \text{ bit}$
$H(Jogar|Tempo = \text{Chuvoso}) = 0,971 \text{ bit}$
A entropia ponderada para o atributo *Tempo* é:
$H(Tempo) = 5/14 * 0,971 + 4/14 * 0 + 5/14 * 0,971 = 0,693 \text{ bit}$
O ganho de informação obtido pela divisão do conjunto de exemplos usando os valores do atributo *Tempo* é:
$IG(Tempo) = 0,940 - 0,693 = 0,247 \text{ bit}$
Qual o significado deste resultado? Significa que, antes de dividir os exemplos e conhecendo o valor do atributo *Tempo*, necessitamos de menos bits para codificar o valor do atributo alvo.

Tabela 6.2 *Distribuição dos valores da classe pelos valores do atributo Tempo*

	Ensolarado	**Nublado**	**Chuvoso**
Sim	2	4	3
Não	3	0	2

Calcular o Ganho da Informação para um Atributo Contínuo. O domínio de um atributo contínuo é um subconjunto de $\Re$, contendo um número infinito de valores. A estratégia usada no caso de atributos nominais não é aplicável a atributos contínuos. A estratégia usual procura uma partição binária do conjunto de treino:

- Conjunto dos exemplos em que o $atributo \leq valor$
- Conjunto dos exemplos em que o $atributo > valor$

Considere o teste $Temperatura = 70,5$. Este teste dividirá os exemplos em duas partições: exemplos em que a $Temperatura \leq 70,5$ e exemplos em que a $Temperatura > 70,5$. A Figura 6.3 mostra a distribuição das classes em cada partição.

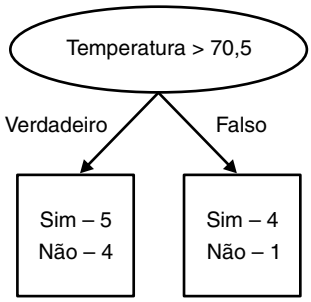


Figura 6.3 *Partições usando o teste* $Temperatura > 70,5$.

Como medir o ganho de informação para esta partição? A partir de cada partição, estimamos as probabilidades condicionais:
$p(Joga = \text{Sim}|Temperatura <= 70,5) = 4/5$
$p(Joga = \text{Não}|Temperatura <= 70,5) = 1/5$
$p(Joga = \text{Sim}|Temperatura > 70,5) = 5/9$
$p(Joga = \text{Não}|Temperatura < 70,5) = 4/9$

A informação das partições é:
$H(Joga|Temperatura <= 70,5) = -4/5 * \log_2(4/5) - 1/5 * \log_2(1/5) = 0,721$ bits
$H(Joga|Temperatura > 70,5) = -5/9 * \log_2(5/9) - 4/9 * \log_2(4/9) = 0,991$ bits
$H(Temperatura) = 5/14 * 0,721 + 9/14 * 0,991 = 0,895$ bits

$IG(Temperatura) = 0,940 - 0,895 = 0,045$ bits

Um teste num atributo contínuo dividirá os dados em dois subconjuntos: $atributo > valor$ e $atributo \leq valor$. Para obter o ponto de corte, os valores do atributo contínuo são, em primeiro lugar, ordenados. O ponto médio entre dois valores consecutivos é um possível ponto de corte e é avaliado pela função mérito. O possível ponto de corte que maximiza a função mérito é escolhido. É importante frisar que não é necessário testar todos os possíveis pontos de corte. Fayyad e Irani (1992) mostraram que, entre todos os possíveis pontos de corte, aqueles que maximizam qualquer função de mérito convexa dividem exemplos de classes diferentes.

O primeiro candidato para o ponto de corte é $64,5$, e o último candidato a ponto de corte é 84. Considerando o ponto de corte $70,5$, a Figura 6.3 mostra as distribuições de classe em cada partição.

Breiman et al. (1984) propuseram a função Gini para medir a impureza, definida como:

$$i(t) = 1 - \sum_i p_i^2 \tag{6.4}$$

em que p_i é a probabilidade para cada classe. Quando um atributo é examinado, a impureza média ponderada dos nós descendentes implícitos é subtraída de $i(t)$ e o atributo que resulta na maior diminuição da impureza é selecionado.

Mingers (1989b) efetuou uma comparação empírica entre diversos critérios de divisão, tendo concluído que:

> "*Os resultados mostram que a escolha da medida afeta o tamanho da árvore, mas não a sua precisão, que permanece inalterada mesmo que os atributos sejam selecionados aleatoriamente.*"

Mais tarde, diversos estudos (Buntine e Niblett, 1992; Esposito et al., 1997) questionaram o desenvolvimento da metodologia usada por Mingers (1989b). Nos estudos citados, a conclusão é que a divisão aleatória conduz a um erro aumentado. No entanto, não há um critério de divisão que seja sistematicamente considerado o melhor de todos.

Regras de Divisão para Regressão

Em problemas de regressão, a função de custo a minimizar é, usualmente, o erro quadrático. Como já referimos, a média é a constante que minimiza o erro quadrático. Por esse motivo, a constante associada às folhas de uma árvore de regressão é a média dos valores do atributo alvo dos exemplos de treino que caem na folha. A construção de uma árvore de regressão é, em tudo, semelhante à construção de uma árvore de regressão, tendo em conta a função de custo referida.

Para estimar o mérito de uma partição obtida por um teste no valor de uma variável, Breiman et al. (1984) propuseram a métrica SDR (do inglês *Standard Deviation Reduction*). Assuma um conjunto de exemplos **D**, composto por n exemplos. A variância da

variável alvo, **y**, é dada pela expressão:

$$sd(\mathbf{D},\mathbf{y}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i-\bar{y})^2} \tag{6.5}$$

Consideremos um teste hipotético h_A sobre o atributo A, por exemplo $A \leq a_1$. Os exemplos do conjunto **D** serão divididos em dois subconjuntos, $\mathbf{D}_L$ e $\mathbf{D}_R$, com tamanhos n_L e n_R, tais que $n = n_L + n_R$. A variância de **y**, a variável alvo, em cada subconjunto $\mathbf{D}_L$ e $\mathbf{D}_R$, é sempre menor ou igual à variância de **y** antes da divisão. Podemos estimar a redução da variância obtida, através da aplicação do teste h_A:

$$SDR(h_A) = sd(\mathbf{D},\mathbf{y}) - \frac{n_L}{n} \times sd(\mathbf{D}_L,\mathbf{y}) - \frac{n_R}{n} \times sd(\mathbf{D}_R,\mathbf{y}) \tag{6.6}$$

Para cada atributo, e para cada possível teste no valor do atributo, é calculada a redução da variância associada a esse teste. O teste que provoca uma maior redução na variância é escolhido como teste para o nó.

6.1.2 Estratégias de Poda

A poda é considerada a fase mais importante do processo de construção da árvore, pelo menos em domínios com ruído. Dados com ruído levantam dois problemas. O primeiro é que as árvores induzidas classificam novos objetos de um modo não confiável. Estatísticas calculadas nos nós mais profundos de uma árvore têm baixos níveis de importância devido ao pequeno número de exemplos que chegam a esses nós. Os nós mais profundos refletem mais o conjunto de treino (superajustamento) e aumentam o erro devido à variância do classificador. O segundo problema é que a árvore induzida tende a ser grande e, portanto, difícil de compreender. Podar uma árvore, i.e. trocar nós profundos por folhas, ajuda a minimizar esses problemas.

Podar uma árvore de decisão quase certamente irá causar a classificação incorreta de alguns exemplos do conjunto de treino. A vantagem da poda torna-se aparente quando se classificam novos exemplos não usados no processo de construção da árvore. Em geral, podar conduz a menores erros de generalização. Os métodos de poda podem ser divididos em dois grupos principais. O primeiro grupo inclui os métodos que param a construção da árvore quando algum critério é satisfeito. Estes métodos são designados como *pré-poda*. O segundo grupo é composto por métodos que constroem uma árvore completa e, posteriormente, realizam a respetiva poda. Estes métodos são designados como *pós-poda*. Todos os métodos mantêm um *ponto de equilíbrio* entre o tamanho da árvore e uma estimativa da taxa de erro (ver Capítulo 9). Tal como as estimativas de erro, podemos diferenciar entre os métodos que estimam o erro a partir do *conjunto de treino*, geralmente conhecido como *erro de substituição*, e métodos que usam um *conjunto de validação* separado, não usado na construção da árvore, para estimar a taxa de erro.

Pré-poda

A pré-poda conta com regras de paragem que previnem a construção daqueles ramos que não parecem melhorar a precisão preditiva da árvore. A pré-poda tem a vantagem de não perder tempo construindo uma estrutura que não é usada na árvore final. No Algoritmo 6.1, a condição de paragem é testada no passo 2. Esposito et al. (1997) apresentam diversas regras de paragem comummente usadas:

1. Todas as observações que alcançam um nó pertencem à mesma classe.
2. Todas as observações que alcançam um nó têm o mesmo vetor de atributos (mas não pertencem necessariamente à mesma classe).
3. O número de observações no nó é menor que um certo limiar.
4. O mérito atribuído a todos os possíveis testes que dividem o conjunto de observações no nó é muito baixo.

As regras 1 e 2 são universalmente aceites. As outras regras foram criticadas porque os testes do limiar muitas vezes terminam o procedimento de divisão prematuramente. Apesar de, em determinadas situações, os atributos individuais parecerem desprovidos de sentido, pode ocorrer que, quando considerados em conjunto com outros atributos, se revelem altamente discriminantes (Esposito et al., 1993). Breiman et al. (1984) também apontam que tais regras de paragem não são fáceis de se obter diretamente: limiares muito altos podem terminar a divisão antes que os seus benefícios se tornem evidentes, enquanto limiares muito baixos resultam em árvores grandes e com um erro de generalização medíocre.

Por essa razão, a pré-poda não será explorada com maior detalhe, embora na literatura se encontrem mais critérios de interrupção, que recorrem a formas de paragem não triviais.

Pós-poda

Este é o método mais comum de poda de árvores de decisão, tendo sido exaustivamente descrito na literatura (Breiman et al., 1984; Quinlan, 1993). Uma árvore completa, superajustada aos dados de treino, é gerada e podada posteriormente. Quinlan (1988) aponta que *"Construir e podar uma árvore é um processo mais lento, mas mais confiável"*. A decisão-chave na pós-poda é podar, ou não, uma subárvore. Um dos métodos mais simples é baseado em duas medidas (Bratko, 1984): o *erro estático* e o *erro de backed-up*. O *erro estático* é o número de classificações incorretas considerando que todos os exemplos que chegam a esse nó são classificados usando a classe maioritária da distribuição de classes desse nó. O *erro de backed-up* é a soma das classificações incorretas de todas as subárvores do nó corrente. Se o *erro de backed-up* é maior ou igual ao *erro estático*, então o nó é trocado por uma folha com a classe maioritária do nó.

A poda *custo de complexidade* foi apresentada por Breiman et al. (1984), e é um dos métodos mais utilizados. Inicialmente é gerada uma árvore completa. Com base nessa árvore é gerada uma sequência de árvores cada vez menores, sendo escolhida uma das

subárvores. Este método é baseado em dois parâmetros: a taxa de erro *R(T)* e o tamanho da árvore, $|T|$, medido em termos das folhas. A medida de *custo-complexidade* para a árvore é:

$$R_\alpha(T) = R(T) + \alpha|T| \tag{6.7}$$

em que α é um parâmetro que pesa a importância relativa do tamanho da árvore em relação à taxa de erro. Para cada valor de α, o objetivo é encontrar a subárvore $|T_\alpha| \leq |T|$ que minimiza $R_\alpha(T)$. Se α é pequeno, a penalidade por ter um grande número de nós terminais é menor e T_α será grande. Quando α aumenta, a subárvore minimizada terá poucos nós terminais. Ainda que α execute através de valores contínuos, o número de subárvores de T é finito. O processo de poda produz, assim, uma sequência finita de subárvores aninhadas $T_1, T_2, T_3, ...$ com nós terminais progressivamente menores. A árvore podada que é selecionada é aquela que minimiza $R_\alpha(T)$ na sequência de subárvores. O livro CART (Breiman et al., 1984) apresenta um algoritmo eficiente para implementar este processo de poda.

A poda pessimista (Quinlan, 1993) estima, para cada subárvore, um erro aparente, baseado na proporção dos exemplos de treino classificados incorretamente nas folhas. Este tipo de poda utiliza o mesmo conjunto de treino para gerar e podar a árvore. A taxa de erro estimada no conjunto de treino é uma estimativa *otimista* e não fornece o melhor critério para escolher a melhor árvore. Por essa razão, Quinlan introduziu a continuidade da correção baseada na distribuição binomial dos erros. Uma subárvore é podada e substituída por uma folha quando a taxa de erro para a subárvore não é significativamente menor que o erro da folha. Uma vez que a poda pessimista utiliza uma estratégia de *cima para baixo* (*top-down*) e não requer um conjunto de poda separado, a sua principal vantagem é a velocidade. A poda pessimista é o método usado no C4.5 (Quinlan, 1988). Esta poda explora a informação do conjunto de treino na construção e simplificação da árvore, usando uma estratégia transversal pós-ordem de *baixo para cima* (*bottom-up*).

Considere um nó gerado a partir de n exemplos de treino. Assumindo a classe maioritária como representando esse nó, os exemplos que não pertencem à classe maioritária seriam classificados como erro. Seja e o número de erros na amostra de n exemplos. Assuma que a verdadeira probabilidade de erro é q e que os n exemplos são gerados por um processo Bernoulli com parâmetro q. Não conseguimos calcular q, mas conseguimos obter um intervalo de confiança $[L_c, U_c]$ que, para um nível de confiança c, contém q.

Dado um nível de confiança c, encontramos os limites z, tais que: $P[X \geq z] = c$. Considerando o conjunto dos exemplos abrangidos por uma folha como uma amostra estatística, é possível estimar um intervalo de confiança $[L_c, U_c]$ relativo à probabilidade de classificação incorreta da folha. O limite superior do intervalo de confiança, U_c, é de particular interesse para a análise do pior caso. Partindo do pressuposto de que os erros no conjunto de treino seguem uma distribuição binomial com probabilidade q em n observações, é possível calcular o valor de U_c:

$$P\left[\frac{e-q}{\sqrt{q(1-q)/n}} > z\right] = c$$

para um nível de confiança c fornecido pelo utilizador, e em que z é o número de desvios padrões correspondentes ao nível de confiança c. Para um nível de confiança de 25%, z é um valor tabelado e igual a 0,69. O limite superior desse intervalo de confiança é dado pela expressão:

$$U_c = \frac{e + \frac{z^2}{2n} + z\sqrt{e/n - e^2/n + z^2/(4n)}}{1 + z^2/n} \tag{6.8}$$

Tendo encontrado o limite superior, a estimativa de erro para as folhas e subárvores são calculadas assumindo que serão usadas para classificar um conjunto de casos não conhecidos do mesmo tamanho que o conjunto de treino. Assim, a taxa de erro estimada para uma folha será $\#exemplos \times U_c$. A soma das taxas de erro estimadas para todas as folhas num nó, é considerada uma estimativa da taxa de erro do próprio nó. Desta forma, comparando a taxa de erro predita para um dado nó, como se ele fosse trocado por uma folha, com a subárvore com raiz nesse nó, podemos decidir se é conveniente podar ou manter o nó.

Existem duas suposições fortes que estão subjacentes a este método de poda. É difícil aceitar que os exemplos do conjunto de treino abrangidos por um nó representem uma amostra estatística, uma vez que a árvore foi construída para se ajustar a esses dados da melhor forma possível. De acordo com Esposito et al. (1997), a suposição de que os erros num exemplo seguem uma distribuição *binomial* é mais questionável.

Mingers (1989a) efetuou uma comparação empírica de cinco métodos de poda para indução de uma árvore de decisão, tendo concluído que: *"Os resultados mostram que dois métodos – complexidade do custo e erro reduzido – têm bom desempenho"*. O autor também mostra que não há interação significativa entre o método usado para gerar uma árvore e os métodos de poda. Esposito et al. (1993) declaram e justificam que a metodologia experimental usada por Mingers é injusta. Tal como as regras de divisão, a poda é um domínio em que nenhuma proposta existente é a melhor para todos os casos. A poda é um enviesamento em direção à simplicidade. Se o domínio do problema admite soluções simples, então a poda é uma opção eficiente (Schaffer, 1993).

6.1.3 Valores desconhecidos

Quando se usa uma árvore como um classificador, o exemplo a ser classificado passa através da árvore. Em cada nó é executado um teste baseado nos valores dos atributos. Se o valor do atributo testado não é conhecido (frequentemente, em dados reais alguns valores de atributos são desconhecidos ou indeterminados), o procedimento pode não determinar o percurso a seguir. Uma vez que uma árvore de decisão constitui uma hierarquia de testes, o *problema do valor desconhecido* assume especial relevância neste tipo de classificadores.

Várias soluções foram propostas na literatura. Quinlan (1986) examinou algumas delas. As mais comuns são:

- Uma estratégia simples consiste em trocar o valor desconhecido pelo valor mais co-

mum para o atributo encontrado no conjunto de treino. Esta estratégia foi discutida no Capítulo 3.

- Outra estratégia consiste em considerar o valor desconhecido como outro valor possível do atributo (Kohavi et al., 1997). Quando se constrói a árvore, cada nó de decisão pode conter um ramo para o caso em que o atributo testado assume um valor desconhecido.
- O C4.5 (Quinlan, 1993) adota uma estratégia mais complexa, que associa uma probabilidade a cada um dos possíveis valores do atributo. As probabilidades são estimadas com base nas frequências observadas dos valores para o atributo nos exemplos do nó corrente. Estes valores fracionários são usados para calcular o *ganho de informação* desse atributo. Quando se classifica um exemplo de teste, o C4.5 passa o exemplo através de todos os ramos em que o valor do atributo desconhecido foi detetado. Cada ramo produz como saída um voto para a classe. A saída final é calculada como a classe maioritária de todas as saídas dos ramos.
- O CART (Breiman et al., 1984) recorre a uma estratégia mais sofisticada, conhecida como *divisão substituta*. Em vez de armazenar, para cada nó, somente o atributo que minimiza a função de impureza, o CART armazena os atributos que produzem uma divisão semelhante, e ordena-os de acordo com o critério de impureza. Quando se procede à classificação de um exemplo, se nesse exemplo o valor do atributo testado no nó da árvore é desconhecido, o CART irá procurar, na lista ordenada de atributos alternativos, o primeiro atributo cujo valor é conhecido no exemplo.

6.1.4 Discussão: Vantagens e Desvantagens

As árvores de decisão têm inúmeras vantagens. São um dos algoritmos mais usados quer em empresas quer no meio académico. Alguns dos pontos mais positivos referenciados na literatura são:

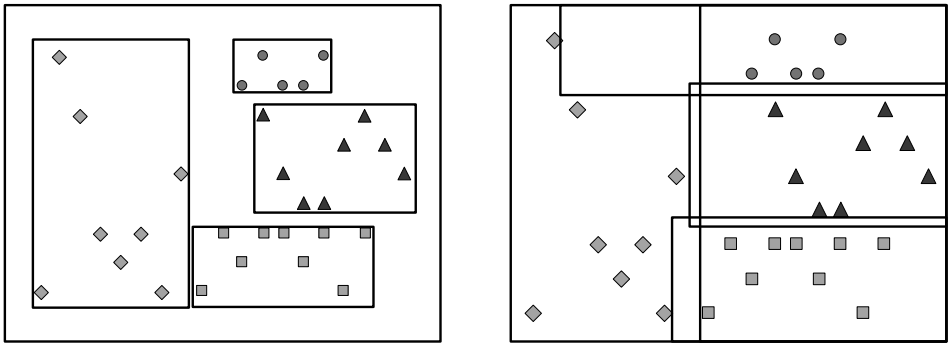
1. *Flexibilidade*
   As árvores de decisão não assumem nenhuma distribuição para os dados. São métodos não paramétricos. O espaço dos objetos é dividido em subespaços, e cada subespaço é aproximado recorrendo a diferentes modelos. Uma árvore de decisão fornece uma cobertura exaustiva do espaço de instâncias. Havendo exemplos suficientes, pode aproximar o *erro de Bayes* de qualquer função.
2. *Robustez*
   As árvores univariadas são invariantes em relação a transformações (estritamente) monótonas de variáveis de entrada. Por exemplo, usar $\mathbf{x}_j$, $\log \mathbf{x}_j$, ou $e^{\mathbf{x}_j}$ como a j-ésima variável de entrada produz árvores com a mesma estrutura. Como consequência dessa invariância, a sensibilidade a distribuições com grande cauda e *outliers* é também reduzida (Friedman, 1999).
3. *Seleção de atributos*
   O processo de construção de uma árvore de decisão seleciona os atributos a usar

no modelo de decisão. Esta seleção de atributos produz modelos que tendem a ser bastante robustos em relação à adição de atributos irrelevantes e redundantes.

4. *Interpretabilidade*
   Decisões complexas e globais podem ser aproximadas por uma série de decisões mais simples e locais. Todas as decisões são baseadas nos valores dos atributos usados para descrever o problema. Ambos os aspetos contribuem para a popularidade das árvores de decisão.
5. *Eficiência*
   O algoritmo para aprendizagem de uma árvore de decisão é um algoritmo guloso que é construído de *cima para baixo* (*top-down*), usando uma estratégia *dividir para conquistar* sem *backtracking*. A sua *complexidade* é $n \times log(n)$, em que n representa o número de exemplos.

Apesar das vantagens previamente mencionadas, as árvores de decisão apresentam alguns problemas. Os mais referenciados na literatura de ECD incluem:

1. *Replicação*
   O termo refere-se à duplicação de uma sequência de testes em diferentes ramos de uma árvore de decisão, levando a uma representação não concisa, que também tende a ter baixa precisão preditiva (ver Capítulo 9). Por exemplo, para representar o conceito $(A \wedge B) \vee (C \wedge D)$, um dos subconceitos $(A \wedge B$ ou $C \wedge D)$ tem de ser duplicado. Suponha que a árvore escolhe para raiz um teste no atributo A, então o conceito $(C \wedge D)$ tem de aparecer nas subárvores descendentes. Pagallo e Haussler (1990) argumentam que a *replicação* é inerente à representação da árvore de decisão.
2. *Valores ausentes*
   Uma árvore de decisão é uma hierarquia de testes. Se o valor de um atributo é desconhecido, isto causa problemas em decidir que ramo seguir. Por esse motivo, os algoritmos devem incorporar mecanismos especiais para lidar com valores omissos. Friedman et al. (1996) observam que *"Cerca de metade do código no CART e 80% do trabalho de programação foram desenvolvidos para tratar valores omissos!"*.
3. *Atributos contínuos*
   O gargalo do algoritmo é a presença de atributos *contínuos*. Nestes casos, é requerido uma operação de *ordenação* para cada atributo *contínuo* em cada nó de decisão. Alguns autores estimam que a operação de ordenação consuma 70% do tempo necessário para induzir uma árvore de decisão em grandes conjuntos de dados com muitos atributos contínuos (Catlett, 1991). Devido a esta observação, alguns investigadores (Catlett, 1991; Fayyad e Irani, 1993) exploraram a possibilidade de discretização dos atributos contínuos.
4. *Instabilidade*
   Muitos investigadores, especialmente Breiman (1996b) e Kohavi e Kunz (1997), apontaram que, pequenas variações no conjunto de treino podem produzir grandes variações na árvore final. A cada nó, o critério de mérito de divisão classifica os



(a) Generalização menos geral (b) Generalização de especificação máxima

Figura 6.4 *Dois exemplos da superfície de decisão desenhadas por um conjunto de regras.*

atributos, e o melhor atributo é escolhido para dividir os dados. Se dois ou mais atributos são classificados de forma semelhante, pequenas variações da classificação dos dados podem alterar a classificação. Consequentemente, todas as subárvores abaixo desse nó mudam. A estratégia da partição recursiva implica que, a cada divisão que é realizada, os dados sejam divididos com base no atributo de teste. Depois de algumas divisões, geralmente existem poucos dados nos quais a decisão se baseia. Há uma forte tendência para as inferências realizadas próximo das folhas serem menos confiáveis do que aquelas que são feitas próximas da raiz.

6.2 Regras de Decisão

Uma regra de decisão é uma implicação da forma: **se** A **então** B. A parte condicional A é uma conjunção de condições. Cada condição é definida por uma relação entre um atributo e os valores do seu domínio. A relação pode ser uma de: $=, <, >, \leq, \geq$ ou $\in$. Por exemplo, pode assumir a forma de: $Atributo_i = \text{valor}_i$, $Atributo_i \leq \text{valor}_i$, $Atributo_i \in \{v_1, v_2\}$ etc, em que valor_i pertence ao domínio do atributo. Um exemplo de uma regra de decisão é: $Tempo = \text{Ensolarado} \wedge Humidade \leq 75 \Rightarrow Jogar = \text{Sim}$. Tal como nas árvores de decisão, o conjunto de regras é disjunto (DNF): $regra_1$ ou $regra_2$ ou ... ou $regra_n$.

As regras de decisão e as árvores de decisão são bastante idênticas na sua forma de representar generalizações dos exemplos. Consequentemente, ambas definem superfícies de decisão semelhantes. As superfícies de decisão definidas pelas regras de decisão correspondem a hiper-retângulos no espaço definido pelos atributos. Dois exemplos de superfícies de decisão esboçados por conjuntos de regras são apresentados na Figura 6.4.

6.2.1 Porquê Regras de Decisão?

Como as árvores de decisão cobrem todo o espaço de instâncias, a vantagem é que qualquer exemplo é classificado por uma árvore de decisão. Porém, cada teste num nó tem um contexto definido por testes anteriores, definidos nos nós do caminho, que podem ser problemáticos se levarmos em conta a interpretabilidade. Por outro lado, as regras são modulares, ou seja, podem ser interpretadas isoladamente.

Cada regra cobre uma região específica do espaço de instâncias. A união de todas as regras pode ser menor que o universo. As caraterísticas das representações das regras são ilustradas na Figura 6.4. Em comparação com as árvores, as regras de decisão:

- Removem condições numa regra sem remover outras regras.
- Não distinguem entre testes perto da raiz e testes perto das folhas.

Um exemplo claro das desvantagens da hierarquia de testes imposta por uma árvore de decisão é conhecido como *fragmentação do conceito*. Propomos como exercício ao leitor obter uma árvore de decisão que represente o conceito: $(A \wedge B) \vee (C \wedge D)$. Como se pode verificar facilmente, qualquer árvore de decisão duplica um dos conceitos, $(A \wedge B)$ ou $(C \wedge D)$.

O Algoritmo OneR

Holte (1993) propôs um algoritmo, OneR (do inglês *One Rule*), que gera regras baseadas num único atributo. O trabalho experimental de Holte (1993) revela que as regras muito simples, como as geradas pelo OneR, são competitivas com métodos muito mais sofisticados.

O *OneR* induz uma árvore de decisão com um nível, ou seja, regras que testam um único atributo. O algoritmo básico consiste em considerar, para cada atributo, uma regra por cada valor desse atributo. Considerando os exemplos de treino, cada uma dessas regras prevê a classe maioritária dos exemplos para os quais o atributo toma esse valor. Ou seja, cada atributo define um conjunto de regras, uma regra para cada valor desse atributo. Para cada atributo, é calculada a taxa de erro, e o atributo com menor taxa de erro é escolhido.

6.2.2 Regras de Decisão a partir de Árvores de Decisão

Árvores de decisão extensas são de difícil compreensão porque o teste de decisão em cada nó aparece num contexto específico, definido pelo resultado de todos os testes nos nós antecedentes. O trabalho desenvolvido por Rivest (1987) apresenta as *listas de decisão*, uma nova representação para a generalização de exemplos que estende as árvores de decisão. A grande vantagem desta representação é a modularidade do modelo de decisão e, consequentemente, a sua interpretabilidade: cada regra é independente das outras regras, e pode ser interpretada isoladamente das outras regras. Como consequência, a representação utilizando regras de decisão permite eliminar um teste numa regra, mas reter o teste noutra

regra. Além disso, como a conjunção de condições é comutativa, a distinção entre testes perto da raiz e testes perto das folhas desaparece.

Indução de Listas de Regras de Decisão. Existem vários algoritmos para indução de regras de decisão (Rivest, 1987; Clark e Niblett, 1989; Cohen, 1995; Domingos, 1996; Weiss e Indurkhya, 1998). Nesta seção, iremos descrever os algoritmos que geram regras de decisão a partir de árvores de decisão, conforme apresentado em Quinlan (1993).

Qualquer árvore de decisão pode ser facilmente reescrita num conjunto de regras de decisão. Cada regra corresponde a um percurso desde a raiz da árvore até uma folha. Existem tantas regras quantas as folhas da árvore de decisão. Este processo gera, assim, um conjunto de regras com a mesma complexidade que uma árvore de decisão. Contudo, alguns antecedentes em regras consideradas individualmente, podem conter condições irrelevantes. `C4.5rules` (Quinlan, 1993, 1995) usa um processo de otimização para simplificar o conjunto de regras, removendo condições irrelevantes.

O processo de otimização consiste em duas fases. Primeiro, cada regra é generalizada pela eliminação de condições que não contribuem para a discriminação das classes. É utilizada uma procura gulosa em que, em cada passo, a regra é avaliada removendo uma das condições. A condição que produz o menor aumento da estimativa pessimista da taxa de erro é eliminada. Para obter a estimativa pessimista da taxa de erro, é utilizado um processo semelhante ao mecanismo de poda usado pelo C4.5. Após a generalização individual das regras, são removidas regras idênticas e as regras sem parte condicional. Numa segunda fase, as regras são agrupadas pela classe que preveem. Para cada classe, o conjunto de regras é simplificado, eliminando as regras que não contribuem para a taxa de acerto do conjunto. O estudo experimental apresentado em Quinlan (1993) mostra que as regras de decisão são mais simples e obtêm taxa de erro menor do que a árvore de decisão a partir da qual foram geradas.

Frank e Witten (1998) apresentam um método para gerar regras por transformação de árvores de decisão, que não necessita recorrer a um processo de otimização global. A ideia base consiste em crescer uma árvore em largura, em vez de crescer a árvore em profundidade. Quando uma folha é encontrada, a regra de decisão correspondente a essa folha é extraída. Os exemplos cobertos pela folha são removidos e, iterativamente, são geradas mais regras com os exemplos não cobertos pelas regras anteriores.

6.2.3 O Algoritmo da Cobertura

Nesta seção estudamos um dos algoritmos mais divulgados para aprender regras de decisão a partir de exemplos. O *algoritmo da cobertura* define o processo de aprendizagem como um processo de procura: sendo dados um conjunto de exemplos classificados e uma linguagem para representar generalizações dos exemplos, o algoritmo efetua, para cada classe, uma procura heurística. Tipicamente, o algoritmo procura regras da forma: **se** $Atributo_i = \text{valor}_j$ **e** $Atributo_l = \text{valor}_k \ldots$ **então** $Classe_z$ A procura pode proceder, quer a partir da regra mais geral (ou seja, de uma regra sem parte condicional) para regras mais

Algoritmo 6.2 Algoritmo de Cobertura: construção de um conjunto de regras

Entrada: Um conjunto de treino $\mathbf{D} = \{(\mathbf{x}_i, y_i), i = 1, \ldots, n\}$
Saída: Um conjunto de regras: **Regras**

```
1  Regras ← {};
2  Seja Y o conjunto das classes em D ;
3  para cada y_i ∈ Y faça
4      repita
5          Regra = Aprende_Uma_Regra(D,y_i) ;
6          Regras ← Regras ∪ {Regra} ;
7          D ← Remove exemplos cobertos pela Regra em D ;
8      até não haver exemplos de y_i;
9  fim
10 Retorna: Regras ;
```

específicas, acrescentando condições; quer a partir de regras muito específicas (por exemplo, um exemplo com restrições em todos os atributos) para regras mais gerais, eliminando restrições. O processo de procura é guiado por uma função de avaliação das hipóteses. Essa função estima a qualidade das regras que são geradas durante o processo. Uma possível função de avaliação é a taxa de erro, com os problemas que já referimos (Breiman et al., 1984). Outra função de avaliação é a informação mútua ou entropia condicional (Clark e Niblett, 1989).

A Ideia Básica

Dado um conjunto de exemplos com várias classes, o *algoritmo de cobertura* consiste em aprender uma regra para cada uma das classes, removendo o conjunto de exemplos cobertos pela regra (ou o conjunto de exemplos positivos), e repetir o processo. O processo termina quando só há exemplos de uma única classe. O algoritmo de cobertura é apresentado no Algoritmo 6.2.

Encontrar uma regra é um problema de procura. Duas estratégias básicas de procura têm sido usadas. A primeira inicia a procura através da regra mais geral, $\{\} \rightarrow Classe$, e aplica operadores de especificação, acrescentando condições à parte condicional da regra. Este tipo de procura é baseado numa estratégia *top-down*, e é orientada pelo modelo. O algoritmo para encontrar uma regra é apresentado no Algoritmo 6.3. É usada em sistemas como CN2 (Clark e Niblett, 1989). A segunda proposta se inicia pela regra mais específica (é escolhido um dos exemplos aleatoriamente, o que implica restrições em todos os atributos) e aplica operadores de generalização, removendo restrições. Essa proposta é *bottom-up* e orientada pelos dados. O algoritmo para encontrar uma regra está apresentado no Algoritmo 6.4. Esta estratégia é usada, por exemplo, no algoritmo AQ (Michalski et al.,

1986; Wnek e Michalski, 1994). A Figura 6.5 apresenta uma ilustração comparativa do funcionamento dos dois algoritmos.

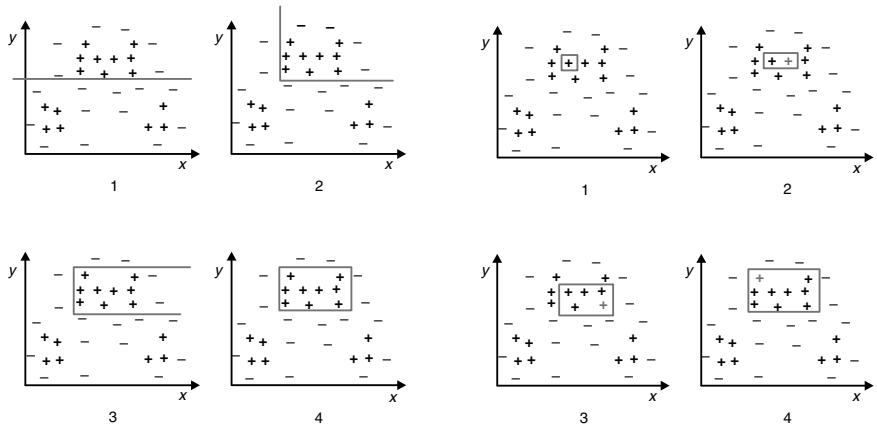
O algoritmo de procura pode ser implementado quer por um algoritmo de *subida em colina* (*hill-climbing*), quer por um algoritmo de *beam-search*. Esta última alternativa requer mais recursos computacionais, mas tem revelado melhor desempenho em termos de capacidade de generalização.

A qualidade de uma regra pode ser medida tendo em conta:

- **ncover**: número de exemplos cobertos pela regra;
- **ncorreto**: número de exemplos cobertos pela regra que foram corretamente classificados pela regra;
- **cobertura**: definida como $ncover/\mathrm{n}$;
- **taxa de acerto**: definida como $ncorreto/ncover$.

Quando é acrescentada uma condição a uma regra, a regra torna-se mais específica e **ncover** diminui. Por outro lado, quando se remove uma condição, a regra torna-se mais geral, e **ncover** aumenta. Qualquer função cujo valor aumente com a cobertura e com a taxa de acerto pode ser utilizada. Por exemplo, o sistema AQ (Wnek e Michalski, 1994) usa a seguinte função de avaliação: $AQ = \frac{ncorreto+1}{ncover+k}$, em que k é o número de classes.

Um dos sistemas de indução de regras de decisão mais populares é o sistema CN2, apresentado em Clark e Niblett (1989). O CN2 usa o algoritmo de cobertura, e a indução de uma regra utiliza o processo *top-down* descrito nesta seção. A função de avaliação de hipóteses usa o conceito de entropia.



(a) *top-down* (b) *bottom-up*

Figura 6.5 *Exemplo ilustrativo do funcionamento do algoritmo para induzir uma regra.*

Algoritmo 6.3 Algoritmo *top-down* para encontrar uma regra

Entrada: Um conjunto de treino $\mathbf{D} = \{(\mathbf{x}_i, y_i), i = 1, \ldots, n\}$
y: classe da regra
Saída: *Regra*: Uma Regra de classificação

```
1  Seja Avs o conjunto de atributo_valores em D ;
2  Regra ← {};
3  v ← Avalia(Regra, D, y);
4  melhor ← v;
5  continua ← Verdadeiro ;
6  enquanto continua faça
7      continua ← Falso ;
8      para cada av_i ∈ Avs faça
9          val ← Avalia(Regra ∪ av_i, D, y) ;
10         se val < melhor então
11             melhor ← val ;
12             Cond ← av_i ;
13             continua ← Verdadeiro ;
14         fim
15     fim
16     se continua então
17         Regra ← Regra ∪ Cond ;
18     fim
19 fim
20 Retorna: Regra ;
```

Exemplo Ilustrativo: Considere novamente o conjunto de exemplos da Tabela 6.1, em que o objetivo é encontrar uma regra para *Joga* = Sim. Neste exemplo ilustrativo, vamos usar como função de avaliação de hipóteses a taxa de erro. Como seria expectável, o objetivo consiste em minimizar esta função.

A procura é encetada com a regra mais geral $\{\} \rightarrow$ Sim. A regra não tem restrições, ou seja, tudo pertence à classe *Sim*. A taxa de erro é de $5/14$. Introduzindo uma restrição, o conjunto de hipóteses é:

- Atributo *Tempo*
  - *Tempo* = Ensolarado $\rightarrow$ Sim; (3/5)
  - *Tempo* = Nublado $\rightarrow$ Sim; (0/4)
  - *Tempo* = Chuvoso $\rightarrow$ Sim; (2/5)
- Atributo *Temperatura*
  - *Temperatura* = Quente $\rightarrow$ Sim; (3/4)

  - *Temperatura* = Fresco → Sim; (1/6)
  - *Temperatura* = Frio → Sim; (1/4)
- Atributo *Humidade*
  - *Humidade* = Alta → Sim; (3/7)
  - *Humidade* = Normal → Sim; (2/7)
- Atributo *Vento*
  - *Vento* = Sim → Sim; (3/6)
  - *Vento* = Não → Sim; (2/8)

Como foi encontrada uma regra com taxa de erro 0, *Tempo* = Nublado → Sim, o processo de encontrar uma regra termina. O algoritmo de cobertura remove os exemplos cobertos pela regra[1] e retoma o processo de encontrar uma nova regra a partir do subconjunto de exemplos.

No sistema AQ (Michalski et al., 1986; Wnek e Michalski, 1994), que segue uma estratégia *bottom-up*, a procura é efetuada do mais específico para o geral (Algoritmo 6.4). Para construir uma regra para uma determinada classe, o ponto de partida é um exemplo dessa classe. O operador de generalização remove condições da regra atual. No caso do sistema AQ, a função de avaliação é: $AQ = (ncorreto + 1)/(ncover + k)$, ou seja, usa a correção de Laplace para a taxa de acerto. Um exemplo ilustrativo da aplicação desse algoritmo é apresentado na Figura 6.6.

Aplicando as Regras de Decisão

A aplicação de um conjunto de regras para classificar um exemplo de teste consiste em verificar qual a regra cuja parte condicional é verificada, e depois decidir de acordo com a conclusão dessa regra. No entanto, no decurso do processo de classificação de um exemplo de teste, podem ocorrer as seguintes situações: *i)* nenhuma regra dispara; *ii)* apenas uma regra dispara; ou *iii)* mais do que uma regra dispara. O primeiro caso pode ser evitado, acrescentando uma regra sem parte condicional e cuja conclusão seja, por exemplo, a classe maioritária. Se mais que uma regra dispara, surgem situações de conflito. Estes conflitos podem ser resolvidos ordenando as regras por prevalência, ou qualquer outro critério de mérito.

Existe uma diferença fundamental entre os dois algoritmos descritos anteriormente, no que se refere à sua aplicação na classificação de exemplos de teste. Enquanto o método *top-down* gera regras ordenadas pela ordem em que são induzidas, o método *bottom-up* gera um conjunto não ordenado de regras. Na aplicação do conjunto de regras a exemplos não classificados, há duas estratégias básicas. No caso de conjuntos ordenados de regras, cada exemplo é classificado pela primeira regra cuja parte condicional é satisfeita. Neste contexto, é frequente adicionar uma regra *default* sem parte condicional, que se aplica quando nenhuma das regras dispara. Conforme visto anteriormente, o algoritmo de cobertura termina quando existem apenas exemplos de uma classe. A regra *default* tem como

[1] Em algumas variantes, são removidos apenas os exemplos corretamente classificados pela regra.

Algoritmo 6.4 Algoritmo *Bottom-Up* para encontrar uma regra

Entrada: Um conjunto de treino $\mathbf{D} = \{(\mathbf{x}_i, y_i), i = 1, \ldots, n\}$
y: classe da regra
Saída: *Regra*: Uma Regra de classificação

Escolhe, aleatoriamente, um exemplo da classe y em $\mathbf{D}$;
Seja *Regra* o conjunto de $atributo\_valor$ desse exemplo ;
$v \leftarrow Avalia(Regra, \mathbf{D}, y)$;
$melhor \leftarrow v$;
$continua \leftarrow$ Verdadeiro ;
enquanto $continua$ **faça**
    $continua \leftarrow$ Falso ;
    para cada $av_i \in Regra$ **faça**
        $val \leftarrow Avalia(Regra \backslash av_i, \mathbf{D}, y)$;
        se $val < melhor$ **então**
            $melhor \leftarrow val$;
            $Cond \leftarrow av_i$;
            $continua \leftarrow$ Verdadeiro ;
        fim
    fim
    se $continua$ **então**
        $Regra \leftarrow Regra \backslash Cond$;
    fim
fim
Retorna: *Regra* ;

conclusão essa classe. No caso de conjuntos de regras não ordenadas, todas as regras cuja parte condicional é verificada são utilizadas para classificar o exemplo, tipicamente por votação pesada pela qualidade da regra.

6.3 Modelos Avançados para Árvores de Decisão

6.3.1 Árvores de Modelos

Na maioria das vezes, as árvores de regressão geram um elevado número de nós, dificultando a interpretação dos dados. Não obstante, em muitos problemas, o seu desempenho costuma ser melhor do que uma simples equação de regressão. No livro de referência de Breiman et al. (1984), os autores escrevem:

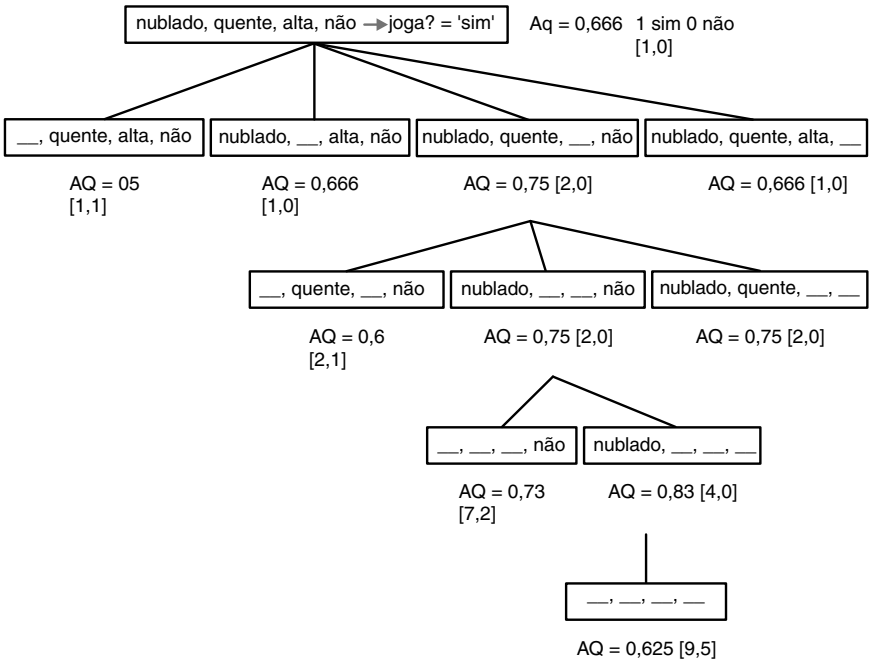


Figura 6.6 *Exemplo do processo bottom-up na construção de uma regra.*

> *'Uma alternativa promissora para melhorar a capacidade de generalização é crescer uma árvore pequena com apenas alguns dos nós mais significativos. Em seguida, obter uma regressão linear múltipla em cada uma das folhas.'*

Uma *árvore de modelos* (do inglês *model tree*) (Quinlan, 1992; Wang e Witten, 1997; Frank et al., 1998) é uma árvore que combina uma árvore de regressão com equações de regressão. Este tipo de árvore funciona da mesma maneira que uma árvore de regressão. A única diferença é que os nós folha contêm expressões lineares em vez de valores agregados (médias ou medianas). A estrutura da árvore divide o espaço dos atributos em subespaços, e os exemplos em cada um dos subespaços são aproximados por uma função linear. A *model tree* é mais compacta e mais compreensível do que uma árvore de regressão e, mesmo assim, apresenta um erro médio menor na predição.

O M5 (Quinlan, 1992) é um indutor para *model tree*. Posteriormente, foi desenvolvido o sistema Cubist (Quinlan, 1998), uma ferramenta para geração de modelos preditivos baseados em regras. O algoritmo do `Cubist` usa o algoritmo do M5 para gerar uma árvore que é posteriormente transformada em regras. O modelo do Cubist consiste numa coleção de regras não ordenadas da forma: **se** condições **então** modelo_linear. A regra indica que, se um caso satisfaz todas as condições, o modelo linear é apropriado para prever o valor do

atributo alvo. Se duas ou mais regras são aplicadas para um caso, é calculada uma média dos valores definidos pelos respetivos modelos lineares para obter uma previsão final. Os modelos gerados pelo Cubist geralmente fornecem melhores resultados do que aqueles que são produzidos por técnicas como a regressão linear multivariada ou redes neuronais, sendo, ao mesmo tempo, mais fáceis de interpretar.

Por exemplo, no problema *Machine-Cpu* disponível no repositório da UCI (Blake et al., 1999), o problema consiste em estimar o desempenho de processadores. O modelo gerado pelo Cubist é: `IF MMAX <= 14000 THEN`
`ERP = 0,55 * CACH + 1,14 * CHMIN + 0,114 CHMAX + 3,15` em que o `ERP` (em português, o equivalente a *Desempenho Relativo Estimado*) é a variável objetivo, e os atributos de entrada são: `MMAX`, a memória RAM, `CACH`, a memória *cache* em kbytes, `CHMIN`, o número mínimo de canais, `CHMAX`, o número máximo de canais.

6.3.2 Árvores de Opção

Os modelos comuns de árvores de decisão incorporam um único teste em cada nó, e cada exemplo segue um único caminho, que é determinado pelo resultado do teste. Testes sucessivos são realizados até que um nó folha seja alcançado e, por conseguinte, seja realizada uma predição para o respetivo exemplo. As *árvores de opção* foram introduzidas por Buntine (1990) como uma generalização das árvores de decisão. Estas árvores podem incluir *nós de opção*, que trocam o teste no valor de um atributo, por um conjunto de testes, cada um dos quais sobre o valor de um atributo. Um nó de opção é como um nó *ou* em árvores *e-ou*. Na construção da árvore, em vez de selecionar o *melhor* atributo, são selecionados todos os atributos promissores, i.e. aqueles com maior valor do ganho de informação. Para cada atributo selecionado, uma árvore de decisão é construída. É importante salientar que uma árvore de opção pode ter três tipos de nós: nós com somente um atributo teste – *nós de decisão*; nós com disjunções dos atributos de teste – *nós de opção*; e folhas.

A classificação de um exemplo **x** usando uma árvore de opção é um processo recursivo:

- Para um nó folha, devolva o rótulo da classe predito pela folha.
- Para um nó de decisão, o exemplo segue o único nó filho.
- Para um nó de opção, o exemplo segue todas as subárvores ligadas ao atributo teste. As previsões do teste disjuntivo são agregadas por um esquema de votação.

Kohavi e Kunz (1997) mostram que a redução de erro verificada nestas árvores é obtida por via da redução da componente de variância (cf. Seção 9.5). Afirmam, ainda, que é possível alcançar uma redução significativa das taxas de erro (em comparação com árvores regulares) usando *árvores de opção* restritas a dois níveis de nós de opção, a partir do topo. Em comparação com o *bagging*, descrito no Capítulo 8, uma *árvore de opção* é determinística e não se baseia em técnicas de amostragem como as expostas no Capítulo 9. As árvores de opção usam todos os dados disponíveis para construir a árvore. A principal

desvantagem das *árvores de opção* é que requerem muito mais memória e espaço em disco do que as árvores normais.

6.4 Considerações Finais

Os modelos de decisão estudados neste capítulo, nomeadamente, árvores e regras de decisão, são dos modelos mais utilizados em aprendizagem automática. Hoje em dia existem algoritmos eficientes para a indução de árvores de decisão ou conjuntos de regras, que obtêm um desempenho equivalente ao de outros modelos (como redes neuronais e SVM), mas com maior grau de interpretabilidade. No entanto, as regras de decisão são, de alguma forma, mais expressivas e flexíveis na representação do conhecimento do que as árvores de decisão.

Capítulo 7

Métodos Baseados em Otimização

Para algumas técnicas de ECD um problema de aprendizagem é formulado como um problema de otimização. O objetivo consiste em minimizar (ou maximizar) uma função de custo que, normalmente, reflete a capacidade de prever a classe.

Neste capítulo são descritas duas técnicas populares de ECD que recorrem à otimização de uma função de custo na fase de treino: as redes neuronais artificiais (RNAs) (Braga et al., 2007; Haykin, 1999) e as máquinas de vetores de suporte (SVMs, do inglês *support vector machines*) (Cristianini e Shawe-Taylor, 2000). As RNAs têm inspiração nas redes neuronais biológicas presentes no cérebro, enquanto as SVMs têm a sua origem na aplicação de conceitos oriundos da Teoria de Aprendizagem Estatística (Vapnik, 1995). O algoritmo de treino mais comum das RNAs envolve uma regra de correção de erros, na qual se recorre à otimização de uma função quadrática do erro entre as respostas da RNA e os rótulos dos exemplos. O treino das SVMs envolve a solução de um problema de otimização quadrática, formulado com o objetivo de maximizar a margem de separação entre os objetos de diferentes classes.

Os principais conceitos de RNAs são introduzidos na Seção 7.1, seguidos por uma introdução às SVMs na Seção 7.2. Em conformidade com a estrutura adotada neste livro, nestas seções as técnicas são apresentadas no contexto da aprendizagem supervisionado. A sua utilização em problemas não supervisionados exige adaptações (Braga et al., 2007; Ben-Hur et al., 2000) que não serão discutidas neste capítulo. A Seção 7.3 apresenta as considerações finais.

7.1 Redes Neuronais Artificiais

O cérebro humano é um modelo que ocorre naturalmente quando pensamos na construção de máquinas inteligentes, ou máquinas com comportamento inteligente. No nosso quotidiano, realizamos diversas tarefas que requerem atenção a diferentes eventos ao mesmo

tempo e o processamento de informações variadas, de forma a tomarmos ações convenientes. Tarefas consideradas simples, como pegar um objeto ou mesmo caminhar, envolvem a ação de diversos componentes, tais como memória, aprendizagem e coordenação física. A complexidade de tais ações simples para a maioria das pessoas é evidenciada pela dificuldade encontrada em ensinar robôs a realizá-las. A realização aparentemente simples de tarefas como essas e muitas outras é possível graças à nossa complexa estrutura biológica, sendo o cérebro humano o grande responsável pelo processamento de informação e pela geração de respostas.

A partir destas motivações, o desenvolvimento das redes neuronais artificiais (RNAs) teve como inspiração a estrutura e o funcionamento do sistema nervoso, com o objetivo de simular a capacidade de aprendizagem do cérebro humano na aquisição de conhecimento.

A procura por modelos computacionais ou matemáticos do sistema nervoso teve início na mesma época em que foram desenvolvidos os primeiros computadores eletrónicos, na década de 1940. Devem-se a McCulloch e Pitts (1943) alguns dos primeiros estudos na área. Em 1943 propuseram um modelo matemático de neurónio artificial em que os neurónios executavam funções lógicas simples e cada um podia executar uma função diferente. Esses neurónios artificiais são conhecidos como unidades lógicas com limiar (LTU, do inglês *Logic Threshold Unit*). McCulloch e Pitts mostraram que a combinação de vários neurónios artificiais em sistemas neuronais produz um elevado poder computacional, uma vez que qualquer função que pudesse ser representada por uma combinação de funções lógicas poderia ser modelada por uma rede formada por esses neurónios. Essas redes iniciais não possuíam capacidade de aprendizagem.

Outros pesquisadores deram importantes contribuições nos anos seguintes, entre eles Hebb (1949), com estudos sobre aprendizagem, Rosenblatt (1958), com sua teoria sobre *perceptrons*, entre outros. As pesquisas na área de RNAs, entretanto, foram interrompidas na década de 1970 por diversos fatores. O principal foi a publicação do livro de Minsky e Papert (1969), no qual os autores apontaram a limitação da rede *perceptron* a problemas linearmente separáveis. Na década de 1980 houve um ressurgimento de interesse na área. Os fatores que contribuíram para a retomada desse interesse foram o aparecimento de computadores mais rápidos, o interesse na construção de computadores paralelos e, principalmente, a proposta de novas arquiteturas de RNAs com maior capacidade de representação e de algoritmos de aprendizagem mais sofisticados.

Os trabalhos iniciais em RNAs tinham por objetivo compreender o cérebro e utilizar o conhecimento obtido para desenvolver sistemas de aprendizagem biologicamente plausíveis. Dessa forma, as RNAs são baseadas em modelos abstratos de como pensamos que o cérebro (e os neurónios) funciona. Como muitos conceitos em RNAs foram inspirados em estudos sobre o sistema nervoso, a próxima seção descreve brevemente os seus principais aspetos.

7.1.1 Sistema Nervoso

O sistema nervoso, do qual faz parte o cérebro, é um conjunto complexo de células que determinam o funcionamento e o comportamento dos seres vivos. Ele está presente em

todos os seres vivos vertebrados e na maioria dos invertebrados. A unidade fundamental do sistema nervoso é a célula nervosa, o neurónio, que se distingue das outras células por apresentar excitabilidade, que lhe permite responder a estímulos externos e internos. Isso possibilita a transmissão de impulsos nervosos a outros neurónios e a células musculares e glandulares.

O principal bloco de construção do cérebro é o neurónio. Os principais componentes de um neurónio são: dendritos, corpo celular e axónio. Um esquema de um neurónio simplificado pode ser visualizado na Figura 7.1. Diferentes tipos de neurónios podem assumir diferentes estruturas. Por estarem fora do âmbito deste livro, estas variações não serão discutidas.

Os dendritos são prolongamentos dos neurónios especializados na receção de estímulos nervosos provenientes de outros neurónios ou do ambiente. Estes estímulos são então transmitidos para o corpo celular ou soma. O soma recolhe as informações recebidas dos dendritos, combina-as e processa-as. De acordo com a intensidade e frequência dos estímulos recebidos, o corpo celular gera um novo impulso, que é enviado para o axónio. O axónio é um prolongamento dos neurónios, responsável pela condução dos impulsos elétricos produzidos no corpo celular até outro local mais distante (usualmente até outros neurónios). Alguns axónios de um adulto humano podem atingir mais de um metro de comprimento. O sinal no neurónio flui então da esquerda para a direita, ou seja, dos dendritos para o corpo celular e em seguida para o axónio. O contato entre a terminação de um axónio e o dendrito de outro neurónio é denominado sinapse. As sinapses são, portanto, as unidades que medeiam as interações entre os neurónios (Haykin, 1999), e podem ser excitatórias ou inibitórias.

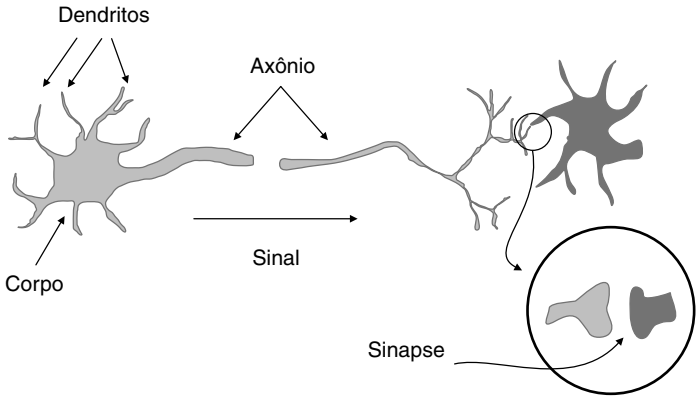


Figura 7.1 *Neurónio biológico simplificado.*

O cérebro humano possui um grande número de neurónios, da ordem de 10 a 500 bilhões. De acordo com estimativas, eles encontram-se organizados em aproximadamente 1000 módulos principais, cada um com 500 redes neuronais. Além disso, cada neurónio pode estar conectado a centenas ou até mesmo milhares de outros neurónios. Essas redes

biológicas trabalham de forma massivamente paralela, revelando uma grande rapidez de processamento. Esta caraterística é evidenciada pelo fato de que, apesar de os neurónios biológicos possuírem um tempo de execução normalmente da ordem de 10^{-3} segundos, o cérebro é capaz de realizar diversas tarefas (como reconhecimento de padrões, perceção e controle motor) várias vezes mais rapidamente que o mais rápido computador digital existente na atualidade.

7.1.2 Componentes Básicos das RNAs

As RNAs são sistemas computacionais distribuídos compostos de unidades de processamento simples, densamente interconectadas. Estas unidades, conhecidas como neurónios artificiais, computam funções matemáticas. As unidades são dispostas numa ou mais camadas e interligadas por um grande número de conexões, geralmente unidirecionais. Na maioria das arquiteturas, estas conexões, que simulam as sinapses biológicas, possuem pesos associados, que ponderam a entrada recebida por cada neurónio da rede. Os pesos podem assumir valores positivos ou negativos, dependendo de o comportamento da conexão ser excitatório ou inibitório, respetivamente. Os pesos têm seus valores ajustados num processo de aprendizagem e codificam o conhecimento adquirido pela rede (Braga et al., 2007).

Uma RNA é portanto caraterizada por dois aspetos básicos: arquitetura e aprendizagem. Enquanto a arquitetura está relacionada com o tipo e número de unidades de processamento e com a forma como os neurónios estão conectados, a aprendizagem diz respeito às regras utilizadas para o ajuste dos pesos da rede e a informação utilizada pelas regras.

Arquitetura

O neurónio é a unidade de processamento fundamental de uma RNA. Na Figura 7.2 é apresentado um modelo simples de neurónio artificial (Haykin, 1999). As unidades de processamento desempenham um papel muito simples. Cada terminal de entrada do neurónio, simulando os dendritos, recebe um valor. Os valores recebidos são ponderados e combinados por uma função matemática f_a, equivalendo ao processamento realizado pelo soma. A saída da função é a resposta do neurónio para a entrada. Várias funções diferentes podem ser utilizadas. Para apresentar as funções, vamos primeiro considerar um objeto $\mathbf{x}$ com d atributos representado na forma de vetor como $\mathbf{x} = [x_1, x_2, ..., x_d]^t$ e um neurónio com d terminais de entrada cujos pesos são $w_1, w_2, ..., w_d$, que podem ser representados na forma vetorial como $\mathbf{w} = [w_1, w_2, ..., w_d]$. A entrada total recebida pelo neurónio, u, pode ser definida pela Equação 7.1:

$$u = \sum_{j=1}^{d} x_j w_j \tag{7.1}$$

Conforme já mencionado, os neurónios podem apresentar conexões de entrada negati-

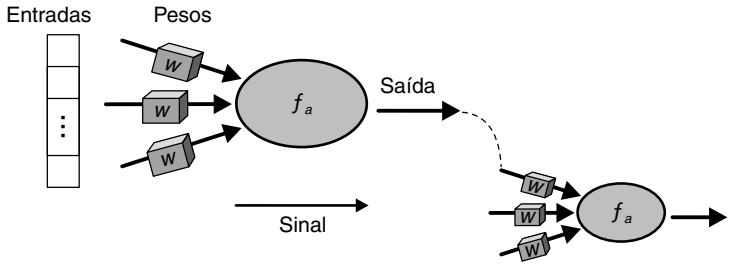


Figura 7.2 *Neurónio artificial.*

vas ($w_j < 0$) ou positivas ($w_j > 0$). Um valor de peso igual a zero equivale a ausência da conexão associada.

A saída de um neurónio é definida por meio da aplicação de uma função de ativação à entrada total, conforme ilustrado na Figura 7.3. Várias funções de ativação têm sido propostas na literatura. A Figura 7.4 mostra a equação e o formato de três dessas funções, as funções linear, limiar e sigmoide. O uso da função linear identidade (Figura 7.4(a)) implica retornar como saída o valor de u. Na função limiar (Figura 7.4(b)), empregue no modelo de neurónio artificial de McCulloch e Pitts (1943), o valor do limiar define quando o resultado da função limiar será igual a 1 ou 0 (alternativamente, pode-se utilizar o valor -1). Quando a soma das entradas recebidas ultrapassa o limiar estabelecido, o neurónio torna-se ativo (saída +1). Quanto maior o valor do limiar, maior tem que ser o valor da entrada total para que o valor de saída do neurónio seja igual a 1. Na função sigmoide (Figura 7.4(c)), diferentes inclinações podem ser utilizadas. A função sigmoidal representa uma aproximação contínua e diferenciável da função limiar.

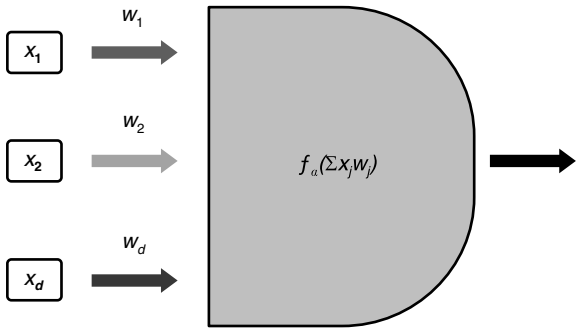
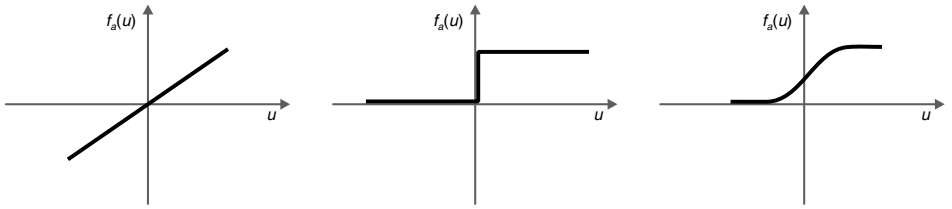


Figura 7.3 *Entrada total num neurónio artificial.*

Numa RNA, os neurónios podem estar dispostos numa ou mais camadas. Quando duas ou mais camadas são utilizadas, um neurónio pode receber nos seus terminais de entrada valores de saída de neurónios da camada anterior e/ou enviar o seu valor de saída para



(a) Linear (b) Limiar (c) Sigmoide

Figura 7.4 *Exemplos de funções de ativação.*

terminais de entrada de neurónios da camada seguinte. A Figura 7.5 ilustra um exemplo de RNA com três camadas. Essa rede recebe como entrada valores de dois atributos de entrada e gera dois valores em sua saída.

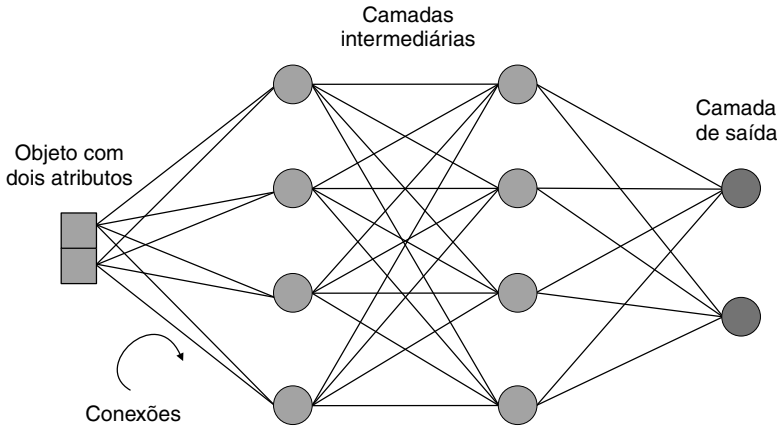
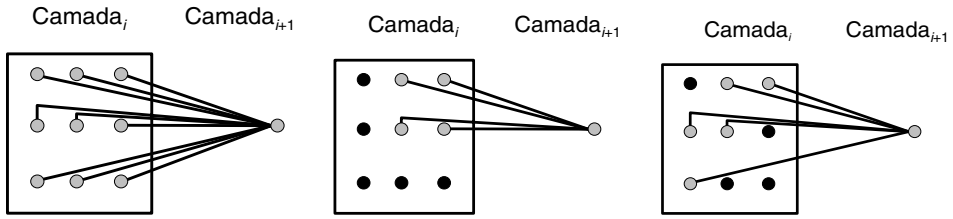


Figura 7.5 *Exemplo de RNA multicamadas típica.*

Uma rede com mais de uma camada de neurónios recebe o nome de rede multicamadas. A camada de neurónios que gera os valores de saída é denominada de camada de saída. As demais camadas são denominadas camadas intermediárias, escondidas ou ocultas. Numa rede multicamadas, as conexões entre os neurónios podem apresentar diferentes padrões de conexão. De acordo com esses padrões, a rede pode ser classificada em:

- Completamente conectada: quando os neurónios da rede estão conectados a todos os neurónios da camada anterior e/ou seguinte.
- Parcialmente conectada: quando os neurónios estão conectados a apenas alguns dos neurónios da camada anterior e/ou seguinte.
- Localmente conectada: são redes parcialmente conectadas, em que os neurónios conectados a um neurónio se encontram numa região bem definida.

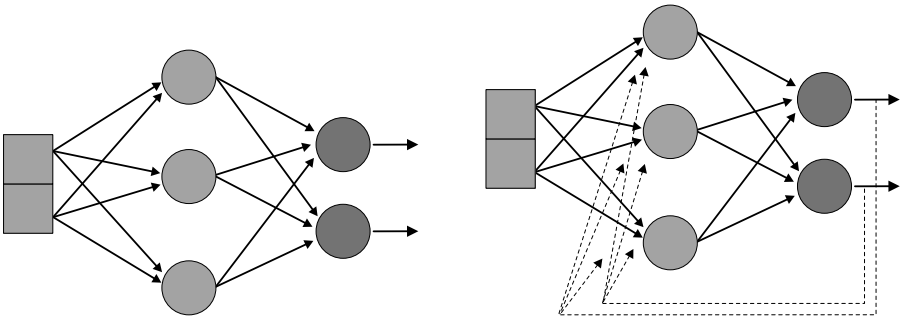
Estes três padrões de conectividade são mostrados na Figura 7.6.



(a) Completamente (b) Localmente (c) Parcialmente

Figura 7.6 *Diferentes padrões de conexão numa RNA multi-camadas.*

Além do grau de conectividade, as RNAs podem apresentar ou não conexões de retroalimentação, ou *feedback*. Numa rede neuronal a informação flui da camada de entrada da rede para os neurónios da camada de saída. Para redes multi-camadas, esse fluxo ocorre camada a camada. As conexões de retroalimentação permitem que um neurónio receba em seus terminais de entrada a saída de um neurónio da mesma camada ou de uma camada posterior. O neurónio pode inclusive receber sua própria saída em um de seus terminais de entrada. As redes com retropropagação, conhecidas como redes recorrentes, são indicadas para aplicações em que é necessário processar informações sequenciais e na simulação de sistemas dinâmicos. Exemplos de aplicações incluem o processamento de linguagem natural e o controle de braços robóticos. Redes sem conexões de retropropagação, que são mais utilizadas na prática, são denominadas RNAs *feedforward*. A Figura 7.7 ilustra exemplos destas duas redes, uma recorrente e uma *feedforward*.



(a) Rede *feedforward* (b) Rede recorrente

Figura 7.7 *Redes neuronais feedforward e recorrente.*

O número de camadas, o número de neurónios em cada camada, o grau de conectividade e a presença ou não de conexões de retropropagação definem a topologia de uma RNA.

Aprendizagem

Vários algoritmos têm sido propostos na literatura para o ajuste dos parâmetros de uma RNA. Por ajuste de parâmetros entende-se principalmente a definição dos valores dos pesos associados às conexões da rede que fazem com que o modelo obtenha melhor desempenho, geralmente medido pela capacidade preditiva. Esses algoritmos, referenciados como algoritmos de treino, são formados por um conjunto de regras bem definidas que especificam quando e como deve ser alterado o valor de cada peso. Diversos autores propuseram algoritmos de treino para RNAs seguindo os paradigmas de aprendizagem supervisionado, não supervisionado e por reforço. Esses algoritmos podem ser divididos em quatro grupos:

1. Correção de erro: geralmente utilizados em aprendizagem supervisionada, procuram ajustar os pesos da RNA de forma a reduzir os erros cometidos pela rede.
2. Hebbiano: frequentemente usados em aprendizagem não supervisionada, são baseados na regra de Hebb, que diz que, se dois neurónios estão simultaneamente ativos, a conexão entre eles deve ser reforçada.
3. Competitivo: utilizados em aprendizagem não supervisionada, promovem uma competição entre neurónios para definir qual ou quais devem ter seus pesos ajustados. Os neurónios que vencem a competição em geral são os que respondem mais fortemente ao objeto apresentado aos seus terminais de entrada.
4. Termodinâmico (Boltzmann): algoritmos estocásticos baseados em princípios observados na metalurgia.

Nas últimas décadas, foi desenvolvido um grande número de arquiteturas de RNAs e algoritmos de treino para treiná-las. A seguir são apresentados algumas das principais arquiteturas existentes e os algoritmos de treino mais utilizados para elas. O foco da apresentação é em modelos e algoritmos do paradigma de aprendizagem supervisionada.

7.1.3 Redes Perceptron e Adaline

A primeira RNA a ser implementada foi a rede *perceptron*, desenvolvida por Rosenblatt (1958). Essa rede, que utiliza o modelo de McCulloch-Pitts como neurónio, introduziu o processo de treino de RNAs. Embora esta rede seja simples, apresentando apenas uma camada de neurónios, obteve boa capacidade preditiva em diversos problemas de classificação. Conforme ilustrado na Figura 7.8, a rede perceptron possuía uma máscara ou retina para receber os objetos de entrada, que eram pré-processados e então apresentados à rede, que possui apenas um neurónio.

A rede perceptron é treinada por um algoritmo supervisionado de correção de erro e usa a função de ativação do tipo limiar. Durante o seu treino, para um objeto $\mathbf{x}_i$, os pesos são ajustados de acordo com a Equação 7.2:

$$w_j(t+1) = w_j(t) + \eta x_i^j (y_i - \hat{f}(\mathbf{x}_i)) \tag{7.2}$$

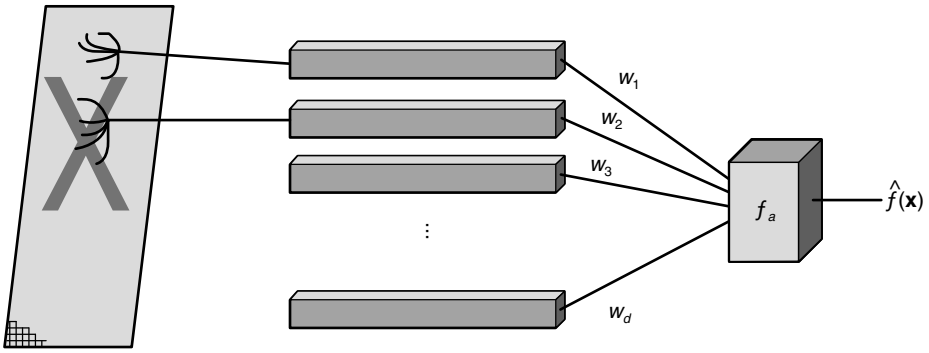


Figura 7.8 *Rede perceptron.*

em que $w_j(t)$ é o peso da j-ésima conexão de entrada no instante de tempo t, η é uma taxa de aprendizagem, x_i^j é o valor do j-ésimo atributo do vetor de entrada $\mathbf{x}_i$, $\hat{f}(\mathbf{x}_i)$ é a saída produzida pela rede no instante de tempo t e y_i é a saída desejada para a rede (o rótulo de $\mathbf{x}_i$). O valor da taxa de aprendizagem define a magnitude do ajuste feito no valor de cada peso. Valores altos fazem com que as variações sejam grandes, enquanto taxas pequenas implicam poucas variações nos pesos. Essa magnitude vai definir a velocidade de convergência da rede. O algoritmo de treino utilizado para a rede perceptron é descrito pelo Algoritmo 7.1.

Algoritmo 7.1 Algoritmo de treino da rede perceptron

Entrada: Um conjunto de n objetos de treino
Saída: Rede perceptron com valores dos pesos ajustados
1 Inicializar pesos da rede com valores baixos
2 **repita**
3 **para cada** *objeto* $\mathbf{x}_i$ *do conjunto de treino* **faça**
4 Calcular valor da saída produzida pelo neurónio, $\hat{f}(\mathbf{x}_i)$
5 Calcular erro = $y_i - \hat{f}(\mathbf{x}_i)$
6 **se** $erro > 0$ **então**
7 Ajustar pesos do neurónio utilizando Equação 7.2
8 **fim**
9 **fim**
10 **até** $erro = 0$;

Alguns anos após propor a rede perceptron, Rosenblatt provou o teorema de convergência da rede perceptron, que diz que, *se é possível classificar um conjunto de entradas linearmente, uma rede perceptron fará a classificação*.

Uma outra rede neuronal com apenas uma camada que surgiu na mesma época da rede perceptron é a rede adaline (formada pelas primeiras sílabas de *Adaptive Linear*) (Widrow e Hoff, 1960). As principais diferenças entre as duas redes é que a rede adaline utiliza uma função de ativação linear e, assim, leva a magnitude do erro em consideração na hora de ajustar os pesos da rede. Para isso, a rede *adaline* utiliza uma regra de ajuste denominada *regra delta*, proposta por Widrow e Hoff (1960), cujo nome vem do uso da diferença entre os valores da saída desejada e da saída produzida. A equação utilizada para o ajuste dos pesos é similar à Equação 7.2 utilizada pela rede perceptron. A diferença está na forma como o valor da saída produzida, $\hat{f}(\mathbf{x}_i)$, é definido, sendo contínuo no caso das redes adaline. Com isso, essas redes são comumente utilizadas em problemas supervisionados de regressão. Em problemas de classificação, as saídas dos neurónios devem ser discretizadas. As redes perceptron, por outro lado, foram propostas para a solução de problemas de classificação.

Uma limitação das redes de uma camada, como as redes perceptron e *adaline*, é que apenas conseguem classificar objetos que são linearmente separáveis. Supondo que os objetos do conjunto de dados possuam apenas dois atributos de entrada, e pertencem a uma de duas classes. Os objetos são linearmente separáveis se após projetar cada objeto num espaço bidimensional, utilizando o valor de cada atributo para definir a posição do objeto, existe uma reta que separa os objetos de uma classe dos objetos da outra classe. Um exemplo de dados com dois atributos que são linearmente separáveis é ilustrado na Figura 7.9. Se em vez de dois os objetos apresentarem d atributos, o espaço de soluções será d-dimensional. Os objetos de duas classes serão linearmente separáveis se houver um hiperplano que separe os dados das duas classes.

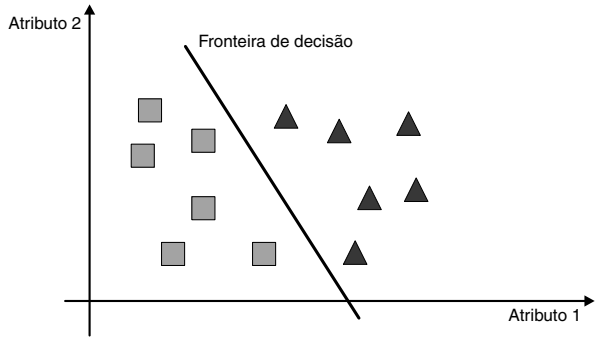


Figura 7.9 *Objetos linearmente separáveis.*

7.1.4 Perceptron Multicamadas

Para resolver problemas não linearmente separáveis utilizando RNAs, a alternativa mais utilizada é adicionar uma ou mais camadas intermediárias. Segundo Cybenko (1989),

uma rede com uma camada intermédia pode implementar qualquer função contínua. A utilização de duas camadas intermédias permite a aproximação de qualquer função.

As redes do tipo perceptron multicamadas (MLP, do inglês *multilayer perceptron*) apresentam uma ou mais camadas intermediárias de neurónios e uma camada de saída. A arquitetura mais comum para uma rede MLP é a completamente conectada, de forma que os neurónios de uma camada l estão conectados a todos os neurónios da camada $l+1$. Caso a camada l seja a primeira camada intermediária, cada um de seus neurónios estará conectado a todos os atributos de entrada $x^j, j = 1 \ldots, d$, para um objeto de entrada $\mathbf{x}$. A Figura 7.5 ilustra uma típica rede MLP.

Redes multicamadas utilizam nas camadas intermediárias funções de ativação não lineares, como a função sigmoide. Pode ser facilmente mostrado utilizando conceitos de operações com matrizes, que uma rede multicamadas com funções de ativação lineares nos neurónios das camadas intermediárias é equivalente a uma rede de uma só camada.

Numa MLP, cada neurónio realiza uma função específica. A função implementada por um neurónio de uma dada camada é uma combinação das funções realizadas pelos neurónios da camada anterior que lhe estão conectados. À medida que o processamento avança de uma camada intermediária para a camada seguinte, o processamento realizado (e a função correspondente) torna-se mais complexo. Na primeira camada, cada neurónio aprende uma função que define um hiperplano. Este hiperplano divide o espaço de entrada em duas regiões. Cada neurónio da camada seguinte combina um grupo de hiperplanos definidos pelos neurónios da camada anterior, formando regiões convexas. Os neurónios da camada seguinte combinam um subconjunto das regiões convexas em regiões de formato arbitrário. A Figura 7.10 exemplifica o papel de cada neurónio na definição das fronteiras de decisão que permitirão à rede classificar novos exemplos. É a combinação das funções desempenhadas por cada neurónio da rede que define a função associada à RNA como um todo.

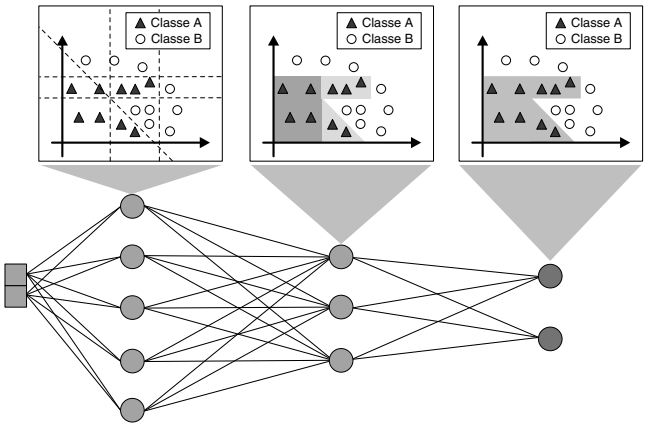


Figura 7.10 *Papel desempenhado pelos neurónios das diferentes camadas da rede MLP.*

Cada neurónio da camada de saída está associado a uma das classes presentes no conjunto de dados. Assim, os valores gerados pelos neurónios de saída para um dado objeto de entrada podem ser representados por um vetor $\mathbf{y} = [y_1, y_2, ..., y_k]^t$, em que k é o número de neurónios da camada de saída (e o número de classes do problema). Para o treino da rede, o vetor de respostas desejadas para cada objeto de entrada tem o valor 1 na posição associada à classe do objeto e 0 nas demais posições. O erro cometido pela rede para a classificação de um dado objeto é então definido pela comparação entre o vetor de saída dos neurónios da camada de saída e o vetor de valores desejados para essas saídas. A rede classifica corretamente um objeto quando o valor de saída mais elevado produzido pela rede é aquele gerado pelo neurónio de saída que corresponde à classe correta do objeto. Um erro de classificação ocorre quando o neurónio de uma outra classe produz o valor de saída mais elevado. Quando nenhum neurónio produz um valor elevado ou o valor elevado é produzido por mais de um neurónio, a rede não tem condições de prever a classe do objeto. Deve-se observar que o uso das MLP também pode ser diretamente estendido a problemas de regressão, mas nesse caso não se tem a discretização imposta pela escolha do neurónio com maior saída na predição.

7.1.5 Algoritmo *Back-propagation*

Um dos obstáculos à aplicação das redes multicamadas era a ausência de um algoritmo para o treino dessas redes. Este obstáculo foi ultrapassado com a proposta de um algoritmo de treino baseado em gradiente descendente denominado *back-propagation* (Rumelhart et al., 1986). Para que esse algoritmo possa ser utilizado, a função de ativação precisa ser contínua, diferenciável e, de preferência, não decrescente. A função de ativação do tipo sigmoide obedece a estes requisitos.

Processo de treino

O algoritmo *back-propagation* é baseado na regra delta utilizada na rede *adaline*, também conhecida como a regra delta generalizada. É um algoritmo iterativo, constituído por duas fases, uma fase para a frente (*forward*) e uma fase para trás (*backward*). Na fase *forward*, cada objeto de entrada é apresentado à rede. O objeto é recebido por cada um dos neurónios da primeira camada intermediária da rede, sendo ponderado pelo peso associado a suas conexões de entrada correspondentes. Cada neurónio nessa camada aplica a função de ativação à soma das suas entradas e produz um valor de saída, que é utilizado como valor de entrada pelos neurónios da camada seguinte. Esse processo continua até que os neurónios da camada de saída produzam o seu valor de saída. Este valor é então comparado ao valor desejado para a saída desse neurónio. A diferença entre os valores de saída produzidos e desejados para cada neurónio da camada de saída indica o erro cometido pela rede para o objeto apresentado.

O valor do erro de cada neurónio da camada de saída é então utilizado na fase *backward* para ajustar os pesos de entrada. O ajuste prossegue da camada de saída até a primeira

camada intermediária. A Equação 7.3 ilustra como é feito o ajuste dos pesos de uma rede MLP pelo algoritmo *back-propagation*.

$$w_{jl}(t+1) = w_{jl}(t) + \eta x^j \delta_l \tag{7.3}$$

Nesta equação, w_{jl} representa o peso entre um neurónio l e o j-ésimo atributo de entrada ou a saída do j-ésimo neurónio da camada anterior, δ_l indica o erro associado ao l-ésimo neurónio e x^j indica a entrada recebida pelo neurónio (o j-ésimo atributo de entrada ou a saída do j-ésimo neurónio da camada anterior).

Como os valores dos erros são conhecidos apenas para os neurónios da camada de saída, o erro para os neurónios das camadas intermédias precisa de ser estimado. O algoritmo *back-propagation* propõe uma maneira de estimar o erro dos neurónios das camadas intermédias utilizando os erros observados nos neurónios da camada posterior. O erro de um neurónio de uma dada camada intermédia é estimado como a soma dos erros dos neurónios da camada seguinte, cujos terminais de entrada lhe conectados, ponderados pelo valor do peso associado a essas conexões. Assim, a forma de calcular o erro depende da camada em que se encontra o neurónio, como mostra a Equação 7.4.

$$\delta_l = \begin{cases} f'_a e_l, & \text{se } n_l \in c_{sai} \\ f'_a \sum w_{lk} \delta_k, & \text{se } n_l \in c_{int} \end{cases} \tag{7.4}$$

Na Equação 7.4, n_l é o l-ésimo neurónio, c_{sai} representa a camada de saída, c_{int} representa uma camada intermediária, f'_a é a derivada parcial da função de ativação do neurónio e e_l é o erro quadrático cometido pelo neurónio de saída quando a sua resposta é comparada à desejada, que é definida pela Equação 7.5, em que k é o número de neurónios na camada de saída.

$$e_l = \frac{1}{2} \sum_{q=1}^{k} (y_q - \hat{f}_q)^2 \tag{7.5}$$

A derivada parcial define o ajuste dos pesos, utilizando o gradiente descendente da função de ativação. Essa derivada mede a contribuição de cada peso no erro da rede para a classificação de um dado objeto $\mathbf{x}$. Se a derivada para um dado peso for positiva, significa que o peso está a contribuir para um aumento da diferença entre a saída da rede e a saída desejada. Assim, a sua magnitude deve ser reduzida para baixar o erro. Se a derivada for negativa, o peso está a contribuir para que a saída produzida pela rede seja mais próxima da desejada. Dessa forma, seu valor deve ser aumentado. O Algoritmo 7.2 ilustra os principais passos do algoritmo *back-propagation*. Nesse algoritmo, denomina-se por ciclo ou época a apresentação de todos os exemplos do conjunto de treino.

Ajuste de Parâmetros

O valor da taxa de aprendizagem η tem uma forte influência no tempo necessário à convergência da rede. Se a taxa de aprendizagem for muito pequena, muitos ciclos podem ser necessários para induzir um bom modelo. Por outro lado, a escolha de uma taxa ele-

Algoritmo 7.2 Algoritmo de treino *back-propagation*

Entrada: Um conjunto de n objetos de treino
Saída: Rede MLP com valores dos pesos ajustados

```
1  Inicializar pesos da rede com valores aleatórios
2  repita
3      Inicializar erro_total = 0
4      para cada objeto x_i do conjunto de treino faça
5          para cada camada da rede, a partir da primeira camada intermediária faça
6              para cada cada neurónio n_jl da camada atual faça
7                  Calcular valor da saída produzida pelo neurónio, f̂
8              fim
9          fim
10         Calcular erro_parcial = y − f̂
11         para cada camada da rede, a partir da camada de saída faça
12             para cada cada neurónio n_jl da camada atual faça
13                 Ajustar pesos do neurónio utilizando Equação 7.3
14             fim
15         fim
16         Calcular erro_total = erro_total + erro_parcial
17     fim
18 até erro_total < ξ;
```

vada pode provocar oscilações que dificultam a convergência. Uma possível medida para amenizar este problema é a introdução do termo *momentum* α, que quantifica o grau de importância da variação de peso do ciclo anterior ao ciclo atual. Esse é o procedimento adotado pelo algoritmo *back-propagation* com termo *momentum* (Rumelhart e McClelland, 1986). Nesse algoritmo, o ajuste dos pesos da rede utiliza a Equação 7.6, tornando a aprendizagem mais estável e acelerando a convergência em regiões planas da função de erro.

$$w_{jl}(t+1) = w_{jl}(t) + \eta x^j \delta_l + \alpha(w_{jl}(t) - w_{jl}(t-1)) \tag{7.6}$$

Nessa equação, x^j indica o valor do j-ésimo atributo do objeto $\mathbf{x}$ ou a saída do j-ésimo neurónio da camada anterior.

Variações

A versão padrão do algoritmo *back-propagation* ajusta os pesos de uma RNA para cada objeto ou exemplo apresentado individualmente. Existe uma variação denominada modo

batch, em que os pesos de cada conexão são ajustados uma única vez para cada ciclo. Todos os exemplos do conjunto de treino são apresentados à rede, o que permite calcular o erro médio. O erro médio é utilizado para o ajuste dos pesos.

Outras variações muito utilizadas do algoritmo *back-propagation* são: Quickprop (Fahlman, 1988), Levenberg-Marquardt (Hagan e Menhaj, 1994), *momentum* de segunda ordem (Pearlmutter, 1992), Newton (Battiti, 1991) e Rprop (Riedmiller e Braun, 1994).

Critério de Paragem

Os ciclos de apresentação dos dados de treino e eventuais ajustes de pesos no *back-propagation* são iterados até que seja atingido um critério de paragem. Diferentes critérios de paragem podem ser utilizados, como, por exemplo, um número máximo de ciclos ou uma taxa máxima de erro. Para reduzir a ocorrência de *overfitting*, parte do conjunto de treino é separado, formando um conjunto de validação. Os dados do conjunto de validação são classificados pela rede a cada n_v ciclos, para estimar a taxa de erro da rede para dados que não fazem parte do conjunto de treino. Se a taxa de erro para os dados de treino e de validação forem mostradas num gráfico, vamos observar que no início do treino as duas taxas tendem a diminuir. A partir de certa altura, a taxa de erro de validação pode começar a subir. Isso é um indício de que a rede parou de aprender e está a superajustar aos dados de treino. Ou seja, estamos perante *overfitting*. Nesse ponto, o treino da rede deve ser terminado. O processo de terminar o treino da rede quando a taxa de erro para o conjunto de validação começa a subir é designado por *early stop*.

Convergência do Algoritmo

A superfície de erro minimizada no *back-propagation*, apresenta regiões de mínimos locais e de mínimo global para problemas complexos. O objetivo do treino é atingir o mínimo global. Entretanto, é possível que o treino com *back-propagation* fique preso a uma região de mínimo local, fazendo com que a rede possua uma baixa capacidade preditiva. Contrariamente às redes perceptron, não existe um teorema de convergência para algoritmos de aprendizagem para o treino de redes multicamadas. Dependendo da distribuição estatística dos objetos, a rede pode convergir para um mínimo local ou demorar muito para encontrar uma solução adequada.

Uma crítica feita ao algoritmo *back-propagation* está relacionado com a sua lentidão na convergência para um bom conjunto de pesos e a sua diminuição de desempenho quando utilizado em grandes conjuntos de dados e problemas complexos. Mesmo para problemas simples, pode ser necessário apresentar o conjunto de treino centenas ou milhares de vezes, o que limita seu uso ao treino de pequenas redes, com no máximo poucos milhares de pesos. Embora alguns problemas práticos possam ser tratados com redes desse tamanho, há problemas reais que requerem redes maiores e mais complexas.

7.1.6 Projeto da Arquitetura de uma RNA

O número adequado de neurónios na camada intermediária de uma RNA depende de vários fatores, como:

- Número de exemplos de treino;
- Quantidade de ruído presente nos exemplos;
- Complexidade da função a ser aprendida;
- Distribuição estatística dos dados de treino.

O primeiro passo para que as RNAs possam induzir um modelo para o conjunto de dados é a definição de sua arquitetura, que engloba a escolha das funções de ativação e da topologia da rede. Conforme explicado anteriormente, a topologia diz respeito ao número de camadas e ao número de neurónios e, para redes multicamadas, qual o padrão de conexões e se existem conexões de retropropagação. A escolha da arquitetura mais promissora para um conjunto de dados não é uma tarefa simples. Geralmente é realizada por um processo de tentativa e erro, avaliando diferentes configurações antes de escolher uma delas. Neste processo de procura, cada arquitetura investigada é treinada e avaliada de acordo com sua capacidade preditiva para o conjunto de dados de treino. Geralmente a arquitetura é definida por um processo de procura exaustiva, que pode ser realizada por diferentes abordagens. As abordagens mais utilizadas são:

- Empírica: consiste na realização de uma procura cega no espaço de possíveis arquiteturas. Assim, diversas arquiteturas são testadas e comparadas até que se encontre uma RNA cuja capacidade preditiva seja adequada. Embora seja a abordagem mais utilizada, a procura cega normalmente apresenta um elevado custo de tempo e esforço. Algumas heurísticas são utilizadas na tentativa de acelerar o tempo de busca por uma RNA apropriada. Por exemplo, explorar apenas RNAs com uma camada intermédia, pois estas já possuem um poder expressivo considerável.
- Meta-heurística: essa abordagem gera um conjunto de variações de RNAs e combina as caraterísticas das que apresentam melhores resultados, gerando assim um novo conjunto de RNAs. Essa técnica utiliza meta-heurísticas, geralmente Algoritmos Genéticos (Holland, 1975), para realizar uma procura global por RNAs eficientes. Como requer treinar um elevado número de RNAs diferentes, esta abordagem tem um elevado custo computacional.
- Poda (ou *Pruning*): nesta abordagem, uma RNA com um grande número de neurônios é treinada até que seja alcançada a precisão desejada. Um algoritmo de poda é utilizado durante ou após o treino para remover conexões ou neurónios redundantes ou irrelevantes das camadas intermediárias da rede. Espera-se como resultado melhorar a capacidade de generalização da rede (Karnin, 1990). Regra geral, no fim do processo, é obtida uma rede mais compacta.

- Construtiva: a abordagem construtiva insere, gradualmente, novos neurónios e conexões numa RNA inicializada sem neurónios intermédios, procurando melhorar seu desempenho face ao problema em questão.

Alternativamente, pode ser utilizada uma rede superdimensionada e evitar o *overfitting* (que ocorreria por causa da superespecialização que está associada a redes muito maiores que o necessário) utilizando validação cruzada ou controlando a norma dos pesos, para que os pesos não assumam valores positivos ou negativos muito elevados.

7.1.7 Aprendizagem profunda

Aprendizagem profunda (do Inglês *Deep Learning*) é uma abordagem para treinar redes neuronais que se tem tornado muito popular nos últimos anos (Goodfellow et al., 2016). Como o nome sugere, aprendizagem profunda é utilizada para o treino de redes neuronais profundas. As redes neuronais profundas são aquelas que possuem pelo menos duas camadas intermediárias. As redes neuronais que possuem menos de duas camadas intermediárias são chamadas de redes neuronais rasas.

Apesar da recente popularidade, redes profundas, treinadas com aprendizagem profunda, já eram utilizadas desde a década de 1970, por exemplo, a rede neuronal *neocognitron* proposta por Fukushima (1980), que já utilizava convolução para preprocessar imagens. Já era conhecido na época que o aumento do número de camadas de uma rede neuronal aumentava sua capacidade de reconhecer padrões. O problema era como treinar eficientemente essas redes.

A pequena quantidade de exemplos disponíveis nos conjuntos de dados, com pouca variação dos objetos de uma mesma classe, e a baixa capacidade de processamento dos computadores da época, que levavam a tempos de treino muito longos, dificultaram a adoção dessas redes e seus algoritmos de treino em aplicações reais. Por exemplo, para até poucos milhares de exemplos de treino, o algoritmo *backpropagation* funciona bem e em tempo útil apenas quando a rede tem poucas camadas intermediárias.

O grande salto na popularidade destas redes ocorreu na década de 1990, quando elas apresentaram desempenho preditivo superior a maioria dos algoritmos de ECD em várias aplicações, nomeadamente classificação de imagens, processamento de língua natural e reconhecimento de voz. O que as destaca em relação a maioria dos outros algoritmos é o preprocessamento realizado pelas primeiras camadas dessas redes. Estas camadas em geral utilizam aprendizado não supervisionado para extrair características dos objetos de entrada.

Na maioria das aplicações de ECD, os dados não são estruturados. As técnicas de extração de características podem ser aplicadas para extrair características consideradas relevantes para construção de modelos, em geral modelos preditivos. Um desafio nesta abordagem é que frequentemente não é clara que características serão mais benéficas para o processo de indução de modelos com alto desempenho preditivo.

Algoritmos de aprendizagem profunda, por outro lado, extraem dos objetos de entrada características que são consideradas relevantes para a extração de modelos com alta ca-

pacidade preditiva. Para isso, as primeiras camadas intermediárias extraem características com níveis de complexidade crescentes. Cada uma dessas camadas transforma as características extraídas pela camada anterior por meio de um processamento simples e não linear. Com isso, as camadas iniciais constroem representações cada vez mais abstratas dos objetos de entrada. Esta transformação amplifica aspetos importantes e elimina variações irrelevantes para a indução de um bom modelo preditivo.

Uma outra característica das redes profundas é o uso de funções de ativação que tornam a aprendizagem mais rápida, como a função ReLU (do inglês *rectified linear unit*), descrita pela Equação 7.7. Conforme ilustrado pela Equação 7.8, o gradiente da função de ativação ReLU é fácil e rápido de calcular, o que acelera o processo de ajuste dos pesos da rede neuronal. Esta propriedade é muito útil em redes com muitas camadas.

$$f(x) = max(0,x) \tag{7.7}$$

$$f'(x) = \begin{cases} 1 & : x > 0 \\ 0 & : x \leq 0 \end{cases} \tag{7.8}$$

Existem várias arquiteturas de redes profundas, assim como vários algoritmos de aprendizagem para treinar modelos a partir de dados de treino. As arquiteturas mais conhecidas são:

- Redes autocodificadoras empilhadas;
- Redes convolucionais;
- Redes de memória de curto prazo.

As redes autocodificadoras empilhadas são redes não suprevisionadas formadas por uma sequência de redes autocodificadoras. As primeiras redes codificadoras são treinadas de forma a que o vetor de saída reproduza o vetor de entrada. Essas redes tinham uma camada intermediária com um número de neurónios menor que o número de entradas e saídas. O objetivo dessas redes consiste em utilizar os neurónios da camada intermedia para codificar o vetor de entradas utilizando um número menor de bits. Uma rede autocodificadora pode ser vista na Figura 7.11 com m unidades de entrada e de saída e k, em que $k < m$ unidades na camada intermediária.

Recentemente foi observado que o desempenho dessas redes era melhorado se o número de neurónios da camada intermediária for superior ao número de elementos de entrada e de saída, mas apenas um número limitado dos neurónios da camada intermediária puder ser utilizado para codificar cada objeto. As redes profundas autocodificadoras em geral utilizam mais de uma camada autocodificadora, as redes autocodificadoras empilhadas. Isso permite a extração de características cada vez mais abstratas da entrada original. A Figura 7.12 apresenta uma rede aucodificadora empilhada.

As redes convolucionais, que são abreviadas para CNN (do inglês *convolutional neural networks*) ou ConvNets, são as redes profundas mais utilizadas em aplicações reais (LeCun

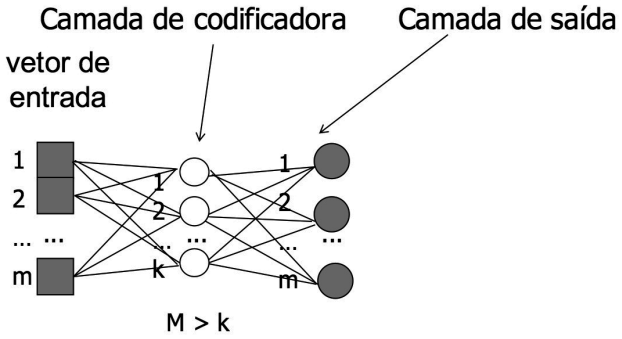


Figura 7.11 *Rede neuronal autocodificadora.*

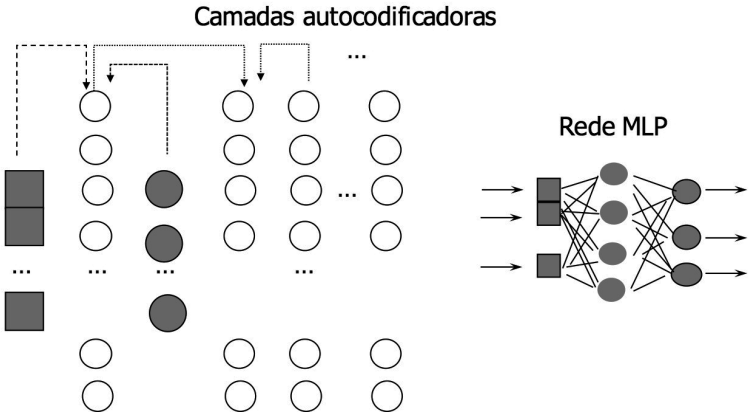


Figura 7.12 *Rede neuronal autocodificadora empilhada.*

et al., 1999). Baseadas na rede *neocognitron*, foram originalmente propostas para visão por computador. Estas redes são estruturadas em dois estágios. O primeiro estágio, por sua vez, é formado por uma sequência de pares de camadas, uma camada de convolução e uma camada de amostragem. Estas camadas aplicam transformações aos pixels da imagem de entrada. O segundo estágio pode ser uma rede MLP convencional. A Figura 7.13 ilustra um exemplo de rede convolucional.

As redes de memória de curto prazo são redes neuronais recorrentes, admitem conexões de feedback que levam a saída de um neurónio a entrada de um ou mais neurónios na mesma camada ou em uma camada anterior. Essas redes são particularmente adequadas para processamento de sequências de objetos, sendo por isso utilizadas com sucesso em problemas onde os objectos apresentam dependências temporais, como identificação de músicas, processamento de lingua natural e sintese de voz.

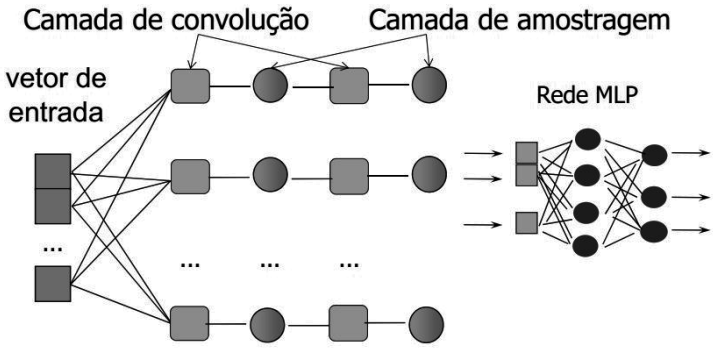


Figura 7.13 *Rede neuronal de convolução*

As redes neuronais profundas, treinadas por algoritmos de aprendizagem profunda, são utilizadas num número crescente de aplicações, que incluem jogos, bioinformática, robótica, veículos autónomos, composição de músicas, geração de texto, tradução automática e aplicações para telemóveis. Os principais assistentes de voz utilizam modelos gerados por redes profundas treinadas com algoritmos de aprendizagem profunda. Novas aplicações surgem a todo instante.

7.1.8 Discussão: Vantagens e Desvantagens

Mesmo com todo o progresso observado no desenvolvimento de RNAs, tanto em diversidade de arquiteturas e algoritmos de treino como em melhorias de desempenho, a capacidade preditiva dessas redes, como das demais técnicas de ECD, ainda está muito aquém da capacidade do cérebro humano. Esta diferença é ainda mais notável se for observado que o cérebro ocupa um volume de 1400 cm^3 e consome apenas 20 W de energia. Embora o tempo de processamento associado a um neurónio seja elevado, da ordem de milissegundos, é cerca de 20 milhões de vezes mais lento que um processador de aproximadamente 2 GHz, essa lentidão é compensada pelo processamento paralelo de um grande número de neurónios densamente conectados a outros neurónios.

Contudo, o constante aumento no número de aplicações que incorporam RNAs confirma a sua importância para a resolução de problemas práticos. As RNAs possuem várias caraterísticas que justificam a sua popularidade, como, por exemplo, generalização e tolerância a falhas e ruídos (Braga et al., 2007; Haykin, 1999). Estas caraterísticas fazem com que as RNAs apresentem um bom desempenho (baixa taxa de erros) quando utilizadas em um grande número de aplicações, destacando-se principalmente tarefas de perceção e controle, tais como visão computacional e robótica.

Contudo, em algumas aplicações, nomeadamente na área médica, o desempenho não é o fator mais importante para a escolha de um modelo de decisão. A facilidade de o uti-

lizador compreender como o modelo classifica um objeto pode ser um critério decisivo. Uma das críticas mais frequentes ao uso de RNAs é a dificuldade de entender como e porquê as RNAs tomam decisões. A principal dificuldade de entender os conceitos representados pelas RNAs consiste no fato de o conhecimento estar armazenado na forma de uma grande quantidade de parâmetros e esses parâmetros serem manipulados através de complicadas fórmulas matemáticas. Devido a essa limitação, RNAs são comumente referenciadas como *caixas-pretas*.

Em contraste com as RNAs, o conhecimento representado por algoritmos simbólicos de Inteligência Artificial, como as regras e árvores de decisão, é geralmente mais amigável e de mais fácil compreensão. Procurando unir o melhor dos dois mundos, vários investigadores dedicam-se ao estudo de técnicas para extração de conhecimento das RNAs que traduzam o conhecimento adquirido pela rede para um formato tão amigável e compreensível quanto aquele gerado por técnicas simbólicas. Entre os vários algoritmos que têm sido propostos para a extração de conhecimento de RNAs, podem ser citados: EN (*Explanation Facility*) (Pau e Gotzche, 1992), KT (*Knowledgetron*) (Fu, 1994), LAP (*Language Processing*) (Hayward et al., 1995), *M-of-N* (Towell e Shavlik, 1993), OLS (Tickle et al., 1995), RULEX (*RULes Extraction*) (Andrews et al., 1996), RuleNeg (Andrews et al., 1995), e TREPAN (*TREes PArroting Networks*) (Craven e Shavlik, 1994). O conhecimento extraído de RNAs assume a forma de um conjunto de regras ou uma árvore de decisão. É importante ressaltar que a extração de conhecimento de uma RNA é uma tarefa que exige recursos e esforços adicionais. Por isso, se não for bem justificada, pode apresentar efeitos negativos. Outra dificuldade das RNAs refere-se à escolha do melhor conjunto de parâmetros para a arquitetura da rede, que faz com que o projeto de redes seja algumas vezes definido como "magia negra" ("black art").

7.2 Máquinas de Vetores de Suporte

As máquinas de vetores de suporte (*Support Vector Machines* – SVMs) (Cristianini e Shawe-Taylor, 2000) tem recebido crescente atenção pela comunidade de ECD. Os resultados da aplicação desta técnica são comparáveis e muitas vezes superiores aos obtidos por outros algoritmos populares de aprendizagem, tal como as RNAs. Exemplos de aplicações de sucesso podem ser encontrados em diversos domínios, com destaque para a categorização de textos (Joachims, 2002) e em Bioinformática (Schölkopf et al., 2003; Noble, 2004).

As SVMs são baseadas na teoria de aprendizagem estatística, desenvolvida por Vapnik (1995) a partir de estudos iniciados em Vapnik e Chervonenkis (1971). Essa teoria estabelece uma série de princípios que devem ser seguidos na obtenção de classificadores com boa capacidade de generalização. Seguidamente é apresentada uma breve introdução aos principais conceitos desta teoria, focando nos aspetos a partir dos quais as SVMs são formuladas. As discussões apresentadas referem-se ao uso de SVMs na solução de problemas de classificação. Uma formulação de SVMs para problemas de regressão é discutida na Seção 7.2.4.

7.2.1 Teoria de Aprendizagem Estatística

Sejam h um classificador e H o conjunto de todos os classificadores que um determinado algoritmo de ECD pode gerar. Durante o processo de aprendizagem, esse algoritmo utiliza um conjunto de treino $\mathbf{X}$, composto de n pares $(\mathbf{x_i}, y_i)$, para gerar um classificador particular $\hat{h} \in H$.

A Teoria de Aprendizagem Estatística (TAE) estabelece condições matemáticas que auxiliam na escolha de um classificador particular $\hat{h}$ a partir de um conjunto de dados de treino. Essas condições levam em conta o desempenho do classificador no conjunto de treino e a sua complexidade, com o objetivo de obter um bom desempenho também para novos dados do mesmo domínio.

Considerações sobre a Escolha do Classificador

Na aplicação da TAE, assume-se inicialmente que os dados do domínio em que a aprendizagem ocorre são gerados de forma independente e identicamente distribuída (i.i.d.) de acordo com uma distribuição de probabilidade $P(\mathbf{x}, y)$ que descreve a relação entre os objetos e os seus rótulos (Burges, 1998). O erro esperado (também denominado risco) de um classificador h para todos os dados desse domínio pode então ser quantificado pela Equação 7.9 (Müller et al., 2001), medindo a capacidade de generalização de h. Na Equação 7.9, $c(h(\mathbf{x}), y)$ é uma função de custo relacionando a previsão $h(\mathbf{x})$ à saída desejada y, em que $y_i \in \{-1, +1\}$. Um tipo de função de custo comumente empregue em problemas de classificação é a $0-1$, definida pela Equação 7.10. Essa função retorna o valor 0 se $\mathbf{x}$ é classificado corretamente e 1 em caso contrário.

$$R(h) = \int c(h(\mathbf{x}), y) dP(\mathbf{x}, y) \tag{7.9}$$

$$c(h(\mathbf{x}), y) = \frac{1}{2} |y - h(\mathbf{x})| \tag{7.10}$$

Infelizmente, não é possível minimizar o risco esperado apresentado na Equação 7.9 diretamente, uma vez que em geral a distribuição de probabilidade $P(\mathbf{x}, y)$ é desconhecida. Tem-se unicamente a informação dos dados de treino, também amostrados de $P(\mathbf{x}, y)$. Normalmente utiliza-se o princípio da indução para inferir uma função $\hat{h}$ que minimize o erro sobre os dados de treino e espera-se que esse procedimento leve também a um menor erro sobre novos dados. Essa é a estratégia adotada pela maioria das técnicas de ECD supervisionadas. O risco empírico de h, fornecido pela Equação 7.11, mede o desempenho do classificador nos dados de treino, por meio da taxa de classificações incorretas obtidas em $\mathbf{X}$.

$$R_{emp}(h) = \frac{1}{n} \sum_{i=1}^{n} c(h(\mathbf{x}_i), y_i) \tag{7.11}$$

Este processo de indução com base nos dados de treino conhecidos constitui o princípio de minimização do risco empírico (Smola e Schölkopf, 2002). Assintoticamente, com

$n \to \infty$, é possível estabelecer condições para o algoritmo de aprendizagem que garantam a obtenção de classificadores cujos valores de risco empírico convergem para o risco esperado (Müller et al., 2001). Para conjuntos de dados menores, porém, geralmente não é possível determinar esse tipo de garantia. Embora a minimização do risco empírico possa conduzir a um menor risco esperado, nem sempre isso se verifica. Considere, por exemplo, um classificador binário (para duas classes) que memoriza todos os objetos de treino e gera classificações aleatórias para outros exemplos (Smola et al., 1999). Embora o risco empírico seja nulo, o risco esperado é 0,5.

A noção expressa nestas considerações é a de que, ao permitir que $\hat{h}$ seja escolhida a partir de um conjunto amplo de funções H, é sempre possível encontrar uma h com pequeno risco empírico. Porém, nesse caso os exemplos de treino podem se tornar pouco informativos para a tarefa de aprendizagem, pois o classificador induzido pode se superajustar a eles. Deve-se então restringir a classe de funções da qual $\hat{h}$ é extraída. Existem diversas abordagens para esse efeito. A TAE lida com esta questão considerando a complexidade (também referenciada por capacidade) da classe de funções que o algoritmo de aprendizagem é capaz de induzir (Smola e Schölkopf, 2002). Nessa direção, a TAE fornece diversos limites no risco esperado de uma função de classificação, os quais podem ser empregues na escolha do classificador. De seguida são relacionados alguns dos principais limites sobre os quais as SVMs se baseiam.

Limites no Risco Esperado

Um limite importante fornecido pela TAE relaciona o risco esperado de uma função ao seu risco empírico e a um termo de capacidade. Esse limite, apresentado na Inequação 7.12, é garantido com probabilidade $1-\theta$, em que $\theta \in [0,1]$ (Burges, 1998).

$$R(h) \leqslant R_{emp}(h) + \sqrt{\frac{VC\left(\ln\left({}^{2n}/_{VC}\right)+1\right) - \ln\left({}^{\theta}/_{4}\right)}{n}} \tag{7.12}$$

Nesta inequação, VC denota a dimensão Vapnik-Chervonenkis (Vapnik, 1995) da classe de funções H à qual h pertence, n representa a quantidade de exemplos no conjunto de treino $\mathbf{X}$ e a parcela de raiz na soma é referenciada como termo de capacidade. A dimensão VC mede a capacidade do conjunto de funções H (Burges, 1998). Quanto maior o seu valor, mais complexas são as funções de classificação que podem ser induzidas a partir de H.

A contribuição principal da Inequação 7.12 está em afirmar a importância de se controlar a capacidade do conjunto de funções H do qual o classificador é extraído. Interpretando-a em termos práticos, tem-se que o risco esperado pode ser minimizado pela escolha adequada, por parte do algoritmo de aprendizagem, de um classificador $\hat{h}$ que minimize o risco empírico e que pertença a uma classe de funções H com baixa dimensão VC. Tendo por base estes objetivos, definiu-se um princípio de indução denominado minimização do risco estrutural (Vapnik, 1998), que procura a função de menor complexidade possível que tenha um baixo erro para os dados de treino.

Embora o limite representado na Inequação 7.12 seja útil na definição do procedimento de minimização do risco estrutural, na prática surgem alguns problemas. Em primeiro lugar, computar a dimensão VC de uma classe de funções geralmente não é uma tarefa trivial. Além disso, o valor da dimensão VC pode ser desconhecido ou infinito (Müller et al., 2001).

Para funções de decisão lineares do tipo $h(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x}$ (em que $\mathbf{w}$ é o vetor normal a h), existem resultados alternativos que relacionam o risco esperado com o conceito de margem (Smola et al., 1999). A margem de um exemplo está relacionada com sua distância à fronteira de decisão induzida, e é uma medida da confiança da previsão do classificador. Para um problema binário, em que $y_i \in \{-1, +1\}$, dados uma função h e um exemplo $\mathbf{x}_i$, a margem $\rho(h(\mathbf{x}_i), y_i)$ com que esse objeto é classificado por h pode ser calculada pela Equação 7.13. Nesta equação, um valor negativo de $\rho(\mathbf{x}_i, y_i)$ denota uma classificação incorreta.

$$\rho\left(h(\mathbf{x}_i), y_i\right) = y_i h\left(\mathbf{x}_i\right) \tag{7.13}$$

Para obter a margem geométrica de um objeto $\mathbf{x}_i$, a qual mede efetivamente a distância de $\mathbf{x}_i$ à fronteira de decisão, divide-se o termo à direita da Equação 7.13 pela norma de $\mathbf{w}$, ou seja, por $\|\mathbf{w}\|$ (Smola e Schölkopf, 2002). Para exemplos classificados incorretamente, o valor obtido equivale à distância com sinal negativo. Para realizar uma diferenciação, a margem da Equação 7.13 será referenciada como margem de confiança.

A partir do conceito introduzido, é possível definir o risco ou erro marginal de uma função h ($R_\rho(h)$) sobre um conjunto de treino. Esse erro fornece a proporção de exemplos de treino cuja margem de confiança é inferior a uma determinada constante $\rho > 0$ (Equação 7.14) (Smola et al., 1999).

$$R_\rho\left(h\right) = \frac{1}{n} \sum_{i=1}^{n} I\left(y_i h\left(\mathbf{x}_i\right) < \rho\right) \tag{7.14}$$

Na Equação 7.14, $I(q) = 1$ se q é verdadeiro e $I(q) = 0$ se q é falso.

Existe uma constante c tal que, com probabilidade $1 - \theta \in [0, 1]$, para todo $\rho > 0$ e H correspondendo à classe de funções lineares $h(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x}$ com $\|\mathbf{x}\| \leq r$ e $\|\mathbf{w}\| \leq 1$, o seguinte limite se aplica (Smola et al., 1999):

$$R\left(h\right) \leqslant R_\rho\left(h\right) + \sqrt{\frac{c}{n}\left(\frac{r^2}{\rho^2}\log^2\left(\frac{n}{\rho}\right) + \log\left(\frac{1}{\theta}\right)\right)} \tag{7.15}$$

Tal como se verifica na Inequação 7.12, na expressão 7.15 tem-se novamente o erro esperado limitado pela soma de uma medida de erro no conjunto de treino, nesse caso o erro marginal, a um termo de capacidade. A interpretação do presente limite é de que uma maior margem ρ implica um menor termo de capacidade. No entanto, a maximização da margem pode levar a um aumento na taxa de erro marginal, pois torna-se mais difícil obedecer à restrição de todos os dados de treino estarem distantes de uma margem maior em relação ao hiperplano separador. Um baixo valor de ρ, em contrapartida, conduz a um

erro marginal menor, porém aumenta o termo de capacidade. Deve-se então encontrar um compromisso entre a maximização da margem e a obtenção de um erro marginal baixo.

Como conclusão tem-se que, na geração de um classificador linear, deve-se procurar um hiperplano que tenha margem ρ elevada e cometa poucos erros marginais, minimizando assim o erro nos dados de treino e também nos novos dados. Esse hiperplano é usualmente denominado ótimo.

Existem outros limites reportados na literatura, assim como outros tipos de medida de complexidade de uma classe de funções. Um exemplo é a dimensão *fat-shattering*, que carateriza o poder de um conjunto de funções em separar os objetos com uma margem ρ (Shawe-Taylor et al., 1998). Contudo, os limites apresentados anteriormente fornecem uma base teórica suficiente à compreensão das SVMs.

7.2.2 SVMs Lineares

As SVMs surgiram pela aplicação direta dos resultados fornecidos pela TAE. Esta seção apresenta o uso de SVMs na obtenção de fronteiras lineares para a separação de objetos pertencentes a duas classes. A primeira formulação, mais simples, lida com problemas linearmente separáveis (Boser et al., 1992). Essa formulação foi posteriormente estendida para definir fronteiras lineares sobre conjuntos de dados mais gerais (Cortes e Vapnik, 1995). A partir desses conceitos iniciais, na Seção 7.2.3 descreve-se a obtenção de fronteiras não lineares com SVMs.

SVMs com Margens Rígidas

As SVMs lineares com margens rígidas definem fronteiras lineares a partir de dados linearmente separáveis. Seja $\mathbf{X}$ um conjunto de treino com n objetos $\mathbf{x}_i \in X$ e seus respetivos rótulos $y_i \in Y$, em que X constitui o espaço de entrada e $Y = \{-1, +1\}$ são as possíveis classes. $\mathbf{X}$ é linearmente separável se é possível separar os objetos das classes $+1$ e -1 por um hiperplano.

Classificadores que separam os dados por meio de um hiperplano são denominados lineares. A equação de um hiperplano é apresentada na Equação 7.16, em que $\mathbf{w} \cdot \mathbf{x}$ é o produto escalar entre os vetores $\mathbf{w}$ e $\mathbf{x}$, $\mathbf{w} \in X$ é o vetor normal ao hiperplano descrito e $\frac{b}{\|\mathbf{w}\|}$ corresponde à distância do hiperplano em relação à origem, com $b \in \Re$.

$$h(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b \tag{7.16}$$

Esta equação pode ser usada para dividir o espaço de entrada X em duas regiões: $\mathbf{w} \cdot \mathbf{x} + b > 0$ e $\mathbf{w} \cdot \mathbf{x} + b < 0$. Uma função sinal $g(\mathbf{x}) = \text{sgn}(h(\mathbf{x}))$ pode então ser empregue na obtenção das classificações, conforme ilustrado na Equação 7.17.

$$g(\mathbf{x}) = \text{sgn}(h(\mathbf{x})) = \begin{cases} +1 \text{ se } \mathbf{w} \cdot \mathbf{x} + b > 0 \\ -1 \text{ se } \mathbf{w} \cdot \mathbf{x} + b < 0 \end{cases} \tag{7.17}$$

A partir de $h(\mathbf{x})$, é possível obter um número infinito de hiperplanos equivalentes, pela multiplicação de $\mathbf{w}$ e b por uma mesma constante. Define-se o hiperplano canónico em relação ao conjunto $\mathbf{X}$ como aquele em que $\mathbf{w}$ e b são escalados de forma a que os exemplos mais próximos ao hiperplano $\mathbf{w} \cdot \mathbf{x} + b = 0$ satisfaçam a Equação 7.18 (Müller et al., 2001).

$$|\mathbf{w} \cdot \mathbf{x}_i + b| = 1 \tag{7.18}$$

Esta forma implica as Inequações 7.19, resumidas na Equação 7.20 e ilustradas na Figura 7.14.

$$\begin{cases} \mathbf{w} \cdot \mathbf{x}_i + b \geqslant +1 & \text{se } y_i = +1 \\ \mathbf{w} \cdot \mathbf{x}_i + b \leqslant -1 & \text{se } y_i = -1 \end{cases} \tag{7.19}$$

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \geqslant 0, \ \ \forall (\mathbf{x}_i, y_i) \in \mathbf{X} \tag{7.20}$$

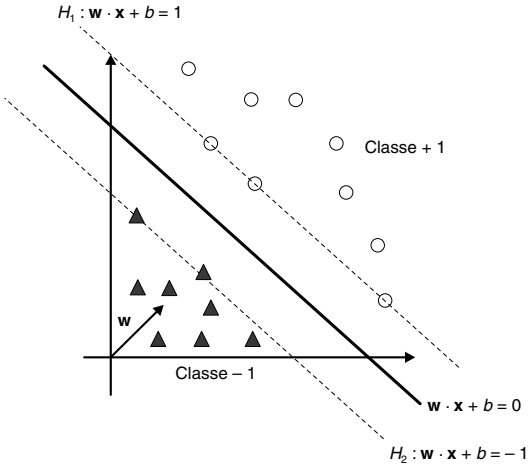


Figura 7.14 *Ilustração de hiperplanos canónicos e separador.*

Manipulações algébricas simples usando pontos sobre os hiperplanos canónicos H_1 e H_2 (Figura 7.14) permitem deduzir que a maximização da margem de separação dos objetos em relação a $\mathbf{w} \cdot \mathbf{x} + b = 0$ pode ser obtida pela minimização de $\|\mathbf{w}\|$ (Campbell, 2000). Dessa forma, recorre-se ao seguinte problema de otimização:

$$\underset{\mathbf{w},b}{\text{Minimizar}} \ \ \frac{1}{2}\|\mathbf{w}\|^2 \tag{7.21}$$

$$\text{Com as restrições: } \ y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \geqslant 0, \ \ \forall i = 1, \ldots, n \tag{7.22}$$

As restrições são impostas de maneira a assegurar que não existam dados de treino entre as margens de separação das classes. Por esse motivo, a SVM obtida possui também a nomenclatura de SVM com margens rígidas.

O problema de otimização obtido é quadrático, cuja solução possui uma ampla e estabelecida teoria matemática. Como a função objetivo sendo minimizada é convexa e os pontos que satisfazem as restrições formam um conjunto convexo, este problema possui um único mínimo global. Problemas deste tipo podem ser solucionados com a introdução de uma função lagrangiana, que engloba as restrições à função objetivo, associadas a parâmetros denominados multiplicadores de Lagrange α_i (Equação 7.23).

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i \left(\mathbf{w} \cdot \mathbf{x}_i + b\right) - 1\right) \tag{7.23}$$

A função lagrangiana deve ser minimizada, o que implica maximizar as variáveis α_i, enquanto $\mathbf{w}$ e b devem ser minimizados. Derivando L em relação a b e $\mathbf{w}$ e igualando o resultado a 0, obtêm-se as Equações 7.24 e 7.25.

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{7.24}$$

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \tag{7.25}$$

Substituindo as Equações 7.24 e 7.25 na Equação 7.23, obtém-se o seguinte problema de otimização:

$$\underset{\alpha}{\text{Maximizar}} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \left(\mathbf{x}_i \cdot \mathbf{x}_j\right) \tag{7.26}$$

$$\text{Com as restrições:} \quad \begin{cases} \alpha_i \geqslant 0, \quad \forall i = 1, \ldots, n \\ \sum\limits_{i=1}^{n} \alpha_i y_i = 0 \end{cases} \tag{7.27}$$

Esta formulação é denominada forma dual, enquanto o problema original é referenciado como forma primal. A forma dual possui a vantagem de apresentar restrições mais simples e permitir a representação do problema de otimização em termos de produtos internos entre objetos, o que será útil na posterior não linearização das SVMs (Seção 7.2.3). É interessante observar também que o problema dual é formulado utilizando apenas os dados de treino e os seus rótulos.

Sejam α^* a solução do problema dual e $\mathbf{w}^*$ e b^* as soluções da forma primal. Obtido o valor de α^*, $\mathbf{w}^*$ pode ser determinado pela Equação 7.25. O parâmetro b^* é definido por α^* e por condições de Kühn-Tucker, provenientes da teoria de otimização com restrições e que devem ser satisfeitas no ponto ótimo. Estas condições afirmam que no ponto ótimo o produto entre as variáveis duais (de Lagrange) e as restrições deve ser nulo (Smola e Schölkopf, 1998). Para o problema dual formulado, tem-se:

$$\alpha_i^* \left(y_i \left(\mathbf{w}^* \cdot \mathbf{x}_i + b^* \right) - 1 \right) = 0, \ \forall i = 1, \ldots, n \tag{7.28}$$

Observa-se nessa equação que α_i^* pode ser diferente de 0 somente para os objetos que se encontram sobre os hiperplanos H_1 e H_2. Estes são os exemplos que se situam mais próximos ao hiperplano separador, exatamente sobre as margens. Para os outros casos, a condição apresentada na Equação 7.28 é obedecida apenas com $\alpha_i^* = 0$. Esses pontos não participam então do cálculo de $\mathbf{w}^*$ (Equação 7.25). Os exemplos que possuem $\alpha_i^* > 0$ são denominados vetores de suporte (*support vectors* - SVs) e podem ser considerados os objetos mais informativos do conjunto de treino (Burges, 1998). O valor de b^* é calculado a partir dos SVs e das condições representadas na Equação 7.28. Calcula-se a média apresentada na Equação 7.29 sobre todos $\mathbf{x}_j$ tal que $\alpha_j^* > 0$, ou seja, todos os SVs. Nessa equação, n_{SV} denota o número de SVs e SV representa o conjunto dos SVs.

$$b^* = \frac{1}{n_{\mathrm{SV}}} \sum_{\mathbf{x}_j \in \mathrm{SV}} \frac{1}{y_j} - \mathbf{w}^* \cdot \mathbf{x}_j = \frac{1}{n_{\mathrm{SV}}} \sum_{\mathbf{x}_j \in \mathrm{SV}} \left(\frac{1}{y_j} - \sum_{\mathbf{x}_i \in \mathrm{SV}} \alpha_i^* y_i \mathbf{x}_i \cdot \mathbf{x}_j \right) \tag{7.29}$$

Como resultado final, tem-se o classificador $g(\mathbf{x})$ apresentado na Equação 7.30.

$$g(\mathbf{x}) = \mathrm{sgn}\left(h(\mathbf{x}) \right) = \mathrm{sgn}\left(\sum_{\mathbf{x}_i \in \mathrm{SV}} y_i \alpha_i^* \mathbf{x}_i \cdot \mathbf{x} + b^* \right) \tag{7.30}$$

SVMs com Margens Suaves

Em situações reais, é difícil encontrar aplicações cujos dados sejam linearmente separáveis. Isto deve-se a diversos fatores, entre eles a presença de ruídos e *outliers* nos objetos ou à própria natureza do problema, que pode ser não linear. As SVMs lineares apresentadas anteriormente podem ser estendidas para lidar com conjuntos de treino mais gerais. Para realizar essa tarefa, permite-se que alguns objetos possam violar a restrição da Equação 7.22. Isso é feito com a introdução de variáveis de folga ξ_i, para todo $i = 1, \ldots, n$. Estas variáveis relaxam as restrições impostas ao problema de otimização primal, que se tornam (Smola e Schölkopf, 2002):

$$y_i \left(\mathbf{w} \cdot \mathbf{x}_i + b \right) \geqslant 1 - \xi_i, \quad \xi_i \geqslant 0, \ \ \forall i = 1, \ldots, n \tag{7.31}$$

A aplicação deste procedimento suaviza as margens do classificador linear, permitindo que alguns objetos permaneçam entre os hiperplanos H_1 e H_2 e também a ocorrência de alguns erros de classificação. Por esse motivo, as SVMs obtidas nesse caso também podem ser referenciadas como SVMs com margens suaves.

Um erro no conjunto de treino é indicado por um valor de ξ_i maior que 1. Logo, a soma dos ξ_i representa um limite no número de erros de treino. Para levar em consideração esse termo, minimizando assim o erro sobre os dados de treino, a função objetivo da Equação 7.21 é reformulada como (Burges, 1998):

$$\underset{\mathbf{w},b,\boldsymbol{\xi}}{\text{Minimizar}} \ \frac{1}{2}\|\mathbf{w}\|^2 + C\left(\sum_{i=1}^{n}\xi_i\right) \tag{7.32}$$

A constante C é um termo de regularização que impõe um peso à minimização dos erros no conjunto de treino em relação à minimização da complexidade do modelo. A presença do termo $\sum_{i=1}^{n}\xi_i$ no problema de otimização também pode ser vista como uma minimização de erros marginais, pois um valor de $\xi_i \in (0,1]$ indica um objeto entre as margens. Tem-se então uma formulação de acordo com os princípios da TAE discutidos na Seção 7.2.1.

Novamente o problema de otimização gerado é quadrático, com as restrições lineares apresentadas na Equação 7.31. A solução envolve passos matemáticos semelhantes aos apresentados anteriormente, com a introdução de uma função lagrangiana e tornando suas derivadas parciais nulas. Tem-se como resultado o seguinte problema dual:

$$\underset{\boldsymbol{\alpha}}{\text{Maximizar}} \ \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) \tag{7.33}$$

$$\text{Com as restrições: } \begin{cases} 0 \leqslant \alpha_i \leqslant C, \ \ \forall i = 1,\ldots,n \\ \sum_{i=1}^{n}\alpha_i y_i = 0 \end{cases} \tag{7.34}$$

Pode-se observar que esta formulação é igual à apresentada para as SVMs de margens rígidas, a não ser pela restrição nos α_i, que agora são limitados pelo valor de C.

Seja α^* a solução do problema dual, enquanto $\mathbf{w}^*$, b^* e ξ^* denotam as soluções da forma primal. O vetor $\mathbf{w}^*$ continua sendo determinado pela Equação 7.25. As variáveis ξ_i^* podem ser calculadas pela Equação 7.35 (Cristianini e Shawe-Taylor, 2000).

$$\xi_i^* = \max\left\{0, 1 - y_i\sum_{j=1}^{n} y_j\alpha_j^*\mathbf{x}_j \cdot \mathbf{x}_i + b^*\right\} \tag{7.35}$$

A variável b^* provém novamente de α^* e de condições de Kühn-Tucker, que nesse caso são:

$$\alpha_i^*\left(y_i(\mathbf{w}^* \cdot \mathbf{x}_i + b^*) - 1 + \xi_i^*\right) = 0 \tag{7.36}$$

$$(C - \alpha_i^*)\,\xi_i^* = 0 \tag{7.37}$$

Como nas SVMs de margens rígidas, os pontos $\mathbf{x}_i$ para os quais $\alpha_i^* > 0$ são denominados vetores de suporte (SVs), e são os objetos que participam da formação do hiperplano separador. Porém, nesse caso, pode-se distinguir tipos distintos de SVs. Se $\alpha_i^* < C$, pela Equação 7.37, $\xi_i^* = 0$ e então, da Equação 7.36, esses SVs encontram-se sobre as margens e também são denominados livres. Os SVs para os quais $\alpha_i^* = C$ podem representar três

casos: erros, se $\xi_i^* > 1$; pontos corretamente classificados, porém entre as margens, se $0 < \xi_i^* \leqslant 1$; ou pontos sobre as margens, se $\xi_i^* = 0$. O último caso ocorre raramente, e os SVs anteriores são denominados limitados. Na Figura 7.15 são ilustrados os possíveis tipos de SVs. Pontos na cor cinza representam SVs livres. SVs limitados são ilustrados em preto. Pontos pretos com bordas extras correspondem a SVs limitados que são erros de treino. Todos os outros objetos, em branco, são corretamente classificados e encontram-se fora das margens, possuindo $\xi_i^* = 0$ e $\alpha_i^* = 0$.

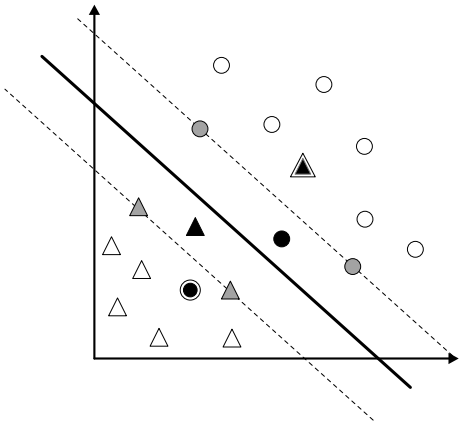


Figura 7.15 *Tipos de SVs: livres (cor cinza) e limitados (cor preta).*

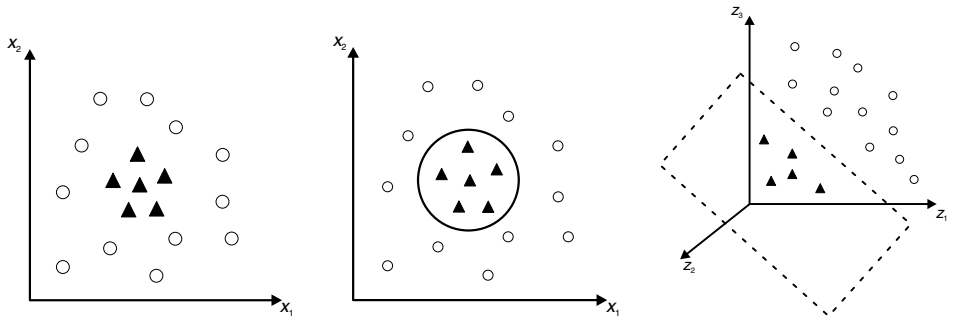
Para calcular b^*, computa-se a média da Equação 7.29 sobre todos SVs $\mathbf{x}_j$ entre as margens, ou seja, com $\alpha_i^* < C$.

Tem-se como resultado final a mesma função de classificação representada na Equação 7.30, porém nesse caso as variáveis α_i^* são determinadas pela solução da Equação 7.33.

7.2.3 SVMs Não Lineares

As SVMs lineares são eficazes na classificação de conjuntos de dados linearmente separáveis ou que possuam uma distribuição aproximadamente linear, e a versão de margens suaves tolera a presença de ruído e *outliers*. Porém, há muitos casos em que não é possível dividir satisfatoriamente os dados de treino por um hiperplano. Um exemplo é apresentado na Figura 7.16, em que o uso de uma fronteira curva seria mais adequada na separação das classes.

As SVMs lidam com problemas não lineares mapeando o conjunto de treino de seu espaço original, referenciado como de entradas, para um novo espaço de maior dimensão, denominado espaço de caraterísticas (*feature space*) (Hearst et al., 1998). Seja $\Phi : X \rightarrow \Im$ um mapeamento, em que X é o espaço de entradas e $\Im$ denota o espaço de caraterísticas. A escolha apropriada de Φ faz com que o conjunto de treino mapeado em $\Im$ possa ser separado por uma SVM linear.



(a) Conjunto de dados (b) Fronteira para duas dimensões (c) Fronteira para três dimensões

Figura 7.16 *Exemplo de transformação realizada em conjunto de dados não linear para espaço de caraterísticas.*

O uso deste procedimento é motivado pelo teorema de Cover (Haykin, 1999). Dado um conjunto de dados não linear no espaço de entradas X, esse teorema afirma que X pode ser transformado num espaço de caraterísticas $\Im$ no qual com alta probabilidade os objetos são linearmente separáveis. Para isso, duas condições devem ser satisfeitas. A primeira é que a transformação seja não linear, enquanto a segunda é que a dimensão do espaço de caraterísticas seja suficientemente alta.

Para ilustrar estes conceitos, considere o conjunto de dados apresentado na Figura 7.16(a) (Müller et al., 2001). Transformando os objetos de $\Re^2$ para $\Re^3$ com o mapeamento representado na Equação 7.38, o conjunto de dados não linear em $\Re^2$ torna-se linearmente separável em $\Re^3$ (Figura 7.16(c)). Então, é possível encontrar um hiperplano capaz de separar esses objetos, descrito na Equação 7.39. Pode-se verificar que a função apresentada, embora linear em $\Re^3$ (Figura 7.16(c)), corresponde a uma fronteira não linear em $\Re^2$ (Figura 7.16(b)).

$$\Phi(\mathbf{x}) = \Phi(x_1, x_2) = \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right) \tag{7.38}$$

$$h(\mathbf{x}) = \mathbf{w} \cdot \Phi(\mathbf{x}) + b = w_1x_1^2 + w_2\sqrt{2}x_1x_2 + w_3x_2^2 + b = 0 \tag{7.39}$$

Logo, mapeiam-se inicialmente os objetos para um espaço de maior dimensão utilizando Φ e aplica-se a SVM linear sobre esse espaço. Esta encontra o hiperplano com maior margem de separação, garantindo assim uma boa generalização. Utiliza-se a versão de SVM linear com margens suaves, que permite lidar com ruídos e *outliers* presentes nos dados. Para realizar o mapeamento, basta aplicar Φ aos exemplos presentes no problema de otimização representado na Equação 7.33, conforme ilustrado a seguir:

$$\underset{\alpha}{\text{Maximizar}} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \left(\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)\right) \tag{7.40}$$

De forma semelhante, o classificador extraído é:

$$g(\mathbf{x}) = \text{sgn}(h(\mathbf{x})) = \text{sgn}\left(\sum_{\mathbf{x}_i \in \text{SV}} \alpha_i^* y_i \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}) + b^*\right) \tag{7.41}$$

em que b^* é adaptado da Equação 7.29 também aplicando o mapeamento aos objetos:

$$b^* = \frac{1}{n_{\text{SV}:\alpha^*<C}} \sum_{\mathbf{x}_j \in \text{SV} \,:\, \alpha_j^*<C} \left(\frac{1}{y_j} - \sum_{\mathbf{x}_i \in \text{SV}} \alpha_i^* y_i \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)\right) \tag{7.42}$$

Como $\Im$ pode ter dimensão muito alta (até mesmo infinita), a computação de Φ pode ser extremamente custosa ou inviável. Porém, percebe-se, pelas Equações 7.40, 7.41 e 7.42, que a única informação necessária sobre o mapeamento é de como realizar o cálculo de produtos escalares entre os objetos no espaço de caraterísticas. Isso é obtido com o uso de funções denominadas kernels.

Um kernel K é uma função que recebe dois pontos $\mathbf{x}_i$ e $\mathbf{x}_j$ no espaço de entradas e calcula o produto escalar desses objetos no espaço de caraterísticas (Herbrich, 2001). Tem-se então:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) \tag{7.43}$$

Para o mapeamento apresentado na Equação 7.38 e dois objetos $\mathbf{x}_i$ e $\mathbf{x}_j$ em $\Re^2$, por exemplo, o kernel é dado por:

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j)^2 \tag{7.44}$$

É comum empregar a função kernel sem conhecer o mapeamento Φ, que é gerado implicitamente. A utilidade dos kernels está, portanto, na simplicidade do seu cálculo e na sua capacidade de representar espaços abstratos.

Para garantir a convexidade do problema de otimização formulado na Equação 7.40 e também que o kernel represente mapeamentos nos quais seja possível o cálculo de produtos escalares conforme a Equação 7.43, utilizam-se funções kernel que seguem as condições estabelecidas pelo teorema de Mercer (Mercer, 1909). De forma simplificada, um kernel que satisfaz as condições de Mercer é caraterizado por dar origem a matrizes positivas semidefinidas $\mathbf{K}$, em que cada elemento K_{ij} é definido por $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$, para todo $i, j = 1, \ldots, n$ (Herbrich, 2001).

Alguns dos kernels mais utilizados na prática são os polinomiais, os de função base radial (*radial basis function* - RBF) e os sigmoidais, listados na Tabela 7.1. Cada um deles apresenta parâmetros que devem ser determinados pelo utilizador, indicados também na tabela. O kernel sigmoide, em particular, satisfaz as condições de Mercer apenas para alguns valores de δ e κ. O kernel polinomial com $d = 1$, $\delta = 1$ e $\kappa = 0$ também é denominado linear, e seu uso implica não realizar um mapeamento dos dados, ou seja, a obtenção de uma SVM linear.

A obtenção de um classificador por meio do uso de SVMs envolve então a escolha

Tabela 7.1 *Funções kernel mais comuns*

Tipo de kernel	Função $K(\mathbf{x}_i, \mathbf{x}_j)$	Parâmetros
Polinomial	$\left(\delta\left(\mathbf{x}_i \cdot \mathbf{x}_j\right) + \kappa\right)^d$	δ, κ e d
RBF	$\exp\left(-\sigma\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2\right)$	σ
Sigmoidal	$\tanh\left(\delta\left(\mathbf{x}_i \cdot \mathbf{x}_j\right) + \kappa\right)$	δ e κ

de uma função kernel, além de parâmetros dessa função e do valor da constante de regularização C. A escolha do kernel e dos parâmetros considerados afeta o desempenho do classificador obtido, pois eles definem a fronteira de decisão induzida. Segundo observado em Hsu et al. (2003), uma boa escolha inicial é empregar o kernel RBF, uma vez que o kernel linear é apontado como um caso especial de função RBF (Keerthi e Lin, 2003) e o kernel sigmoidal pode ter comportamento semelhante ao RBF para certos parâmetros (Lin e Lin, 2003).

O Algoritmo 7.3 resume então a formulação final seguida pelas SVMs em seu treino.

Algoritmo 7.3 Algoritmo de treino de SVM

Entrada: Um conjunto de n objetos de treino
Saída: SVM treinada

1 Seja $\alpha^* = (\alpha_1^*, \ldots, \alpha_n^*)$ a solução de:

2 Maximizar: $\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j K\left(\mathbf{x_i}, \mathbf{x_j}\right)$

3 Sob as restrições: $\begin{cases} \sum_{i=1}^{n} y_i \alpha_i = 0 \\ 0 \leq \alpha_i \leq C, \ i = 1, \ \ldots, \ n \end{cases}$

4 O classificador é dado por: $g(\mathbf{x}) = \operatorname{sgn}(h(\mathbf{x})) = \operatorname{sgn}\left(\sum_{\mathbf{x}_i \in \text{SV}} \alpha_i^* y_i K(\mathbf{x}_i, \mathbf{x}) + b^*\right)$

5 Em que: $b^* = \frac{1}{n_{\text{SV}:\alpha^* < C}} \sum_{\mathbf{x}_j \in \text{SV} \,:\, \alpha_j^* < C} \left(\frac{1}{y_j} - \sum_{\mathbf{x}_i \in \text{SV}} \alpha_i^* y_i K(\mathbf{x}_i, \mathbf{x}_j)\right)$

7.2.4 SVMs em Problemas de Regressão

Embora as descrições anteriores tenham sido focadas em problemas de classificação, as SVMs também podem ser aplicadas na solução de problemas de regressão e no agrupamento de dados (aprendizagem não supervisionada). Contudo, o problema de otimização para o seu treino deve ser reformulado para lidar com as caraterísticas e objetivos desses

problemas. Nesta seção, o uso de SVMs em problemas de regressão é brevemente descrito, tomando como base o tutorial (Smola e Schölkopf, 1998).

O algoritmo ε-SVR (*support vector regression*) (Vapnik, 1995) tem como objetivo encontrar uma função $h(\mathbf{x})$ que produza saídas contínuas para os dados de treino que desviem no máximo de ε de seu rótulo desejado. Essa função deve também ser o mais uniforme e regular possível.

Seguindo a estrutura apresentada para o caso de classificação, consideremos primeiro o uso de funções lineares h (Equação 7.16). Nesse caso, a regularidade reflete-se na procura de uma função com $\mathbf{w}$ pequeno, o que pode ser conseguido pela minimização da norma $\|\mathbf{w}\|$. Tem-se então o problema de otimização:

$$\underset{\mathbf{w},b}{\text{Minimizar}} \quad \frac{1}{2}\|\mathbf{w}\|^2 \tag{7.45}$$

$$\text{Com as restrições:} \quad \begin{cases} y_i - \mathbf{w}\cdot\mathbf{x}_i - b \leqslant \varepsilon_i \\ \mathbf{w}\cdot\mathbf{x}_i + b - y_i \leqslant \varepsilon_i \end{cases} \tag{7.46}$$

Procura-se então a função linear que aproxime os pares $(\mathbf{x}_i, y_i)$ de treino com uma precisão de ε. Na Figura 7.17 é apresentada uma ilustração do procedimento realizado. Procura-se a função linear tal que os dados de treino fiquem dentro de uma região ao redor de h, representada em sombreado na figura.

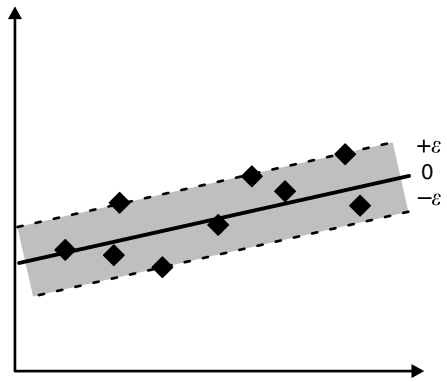


Figura 7.17 *Ilustração simplificada de procedimento realizado por SVR.*

Analogamente ao caso das SVMs de margens suaves, este problema pode ser relaxado com a introdução de variáveis de folga, permitindo assim lidar com ruído e *outliers* nos objetos. As variáveis de folga permitem que alguns exemplos fiquem fora da região entre $-\varepsilon$ e $+\varepsilon$. Tem-se então:

$$\underset{\mathbf{w},b,\xi,\bar{\xi}}{\text{Minimizar}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\left(\sum_{i=1}^{n}\xi_i + \bar{\xi}_i\right) \tag{7.47}$$

$$\text{Com as restrições:} \quad \begin{cases} y_i - \mathbf{w}\cdot\mathbf{x}_i - b \leqslant \varepsilon + \xi \\ \mathbf{w}\cdot\mathbf{x}_i + b - y_i \leqslant \varepsilon + \bar{\xi} \\ \xi_i, \bar{\xi}_i \geqslant 0 \end{cases} \tag{7.48}$$

Nas equações apresentadas, ξ_i e $\bar{\xi}_i$ representam as variáveis de folga e C é uma constante que impõe um *trade-off* entre a regularidade de h e o quanto de desvios são tolerados. Como no caso das SVMs para classificação, monta-se o problema dual equivalente ao anterior pela utilização de uma lagrangiana, tornando nulo o resultado das derivações parciais e substituindo as expressões resultantes na equação lagrangiana inicial.

O problema dual obtido é descrito em termos de produtos internos entre os objetos. Pode-se então recorrer ao uso de kernels para realizar regressões não lineares. O uso do kernel implica o mapeamento dos objetos para um espaço de caraterísticas, onde então a função linear mais regular e com baixo erro de treino é encontrada. O problema de otimização final solucionado é dado por:

$$\underset{\alpha,\bar{\alpha}}{\text{Maximizar}} \quad -\frac{1}{2}\sum_{i,j=1}^{n}(\alpha_i - \bar{\alpha}_i)(\alpha_j - \bar{\alpha}_j)K(\mathbf{x}_i,\mathbf{x}_j) - \varepsilon\sum_{i=1}^{n}(\alpha_i + \bar{\alpha}_i) + \sum_{i=1}^{n} y_i(\alpha_i - \bar{\alpha}_i) \tag{7.49}$$

$$\text{Com as restrições:} \quad \begin{cases} \sum_{i=1}^{n}(\alpha_i - \bar{\alpha}_i) = 0 \\ \alpha_i,\ \bar{\alpha}_i \in [0, C] \end{cases} \tag{7.50}$$

Nas equações apresentadas, α_i e $\bar{\alpha}_i$ representam as variáveis de Lagrange e K é a função kernel, que deve satisfazer as condições de Mercer (os mesmos tipos de kernel da Tabela 7.1 podem ser utilizados para SVR). As variáveis de Lagrange associadas a todos os exemplos que se encontram dentro da margem entre $-\varepsilon$ e $+\varepsilon$ são nulas. Os outros casos correspondem aos SVs.

7.2.5 Discussão: Vantagens e Desvantagens

Com base em princípios da teoria de aprendizagem estatística, as SVMs caraterizam-se por apresentar uma boa capacidade de generalização. As SVMs também são robustas diante de objetos de grande dimensão, sobre os quais outras técnicas de aprendizagem comumente obtêm classificadores super ou subajustados. Outra caraterística atrativa é a convexidade do problema de otimização formulado no seu treino, que implica a existência de um único mínimo global. Esta é uma vantagem das SVMs sobre, por exemplo, as RNAs MLP, em que há mínimos locais na função objetivo minimizada. Além disso, o uso

de funções kernel na não linearização das SVMs torna o algoritmo eficiente, pois permite a construção de hiperplanos simples num espaço de alta dimensão de forma tratável do ponto de vista computacional (Burges, 1998).

Entre as principais limitações das SVMs encontram-se a sua sensibilidade a escolhas de valores de parâmetros e a dificuldade de interpretação do modelo gerado por essa técnica. Esses problemas foram tratados em alguns trabalhos, tais como Chapelle et al. (2002); Duan et al. (2003) e Fu et al. (2004).

Observou-se, no decorrer desta seção, que o raciocínio empregue pelas SVMs na obtenção do classificador final leva a um problema de otimização dual em termos dos dados de treino. Porém, a forma de solução desse problema não foi apresentada. Existem diversos pacotes matemáticos capazes de solucionar problemas quadráticos com restrições. Contudo, geralmente não são adequados a aplicações de ECD, que em geral se caraterizam pela necessidade de lidar com um grande volume de dados. Diversas técnicas e estratégias foram então propostas para adaptar a solução do problema de otimização das SVMs a aplicações de larga escala. Em geral, recorre-se a alguma estratégia decomposicional, em que subproblemas menores são otimizados a cada passo do algoritmo. Uma discussão mais detalhada a respeito dos métodos e algoritmos comumente empregues nesse processo pode ser encontrada em Cristianini e Shawe-Taylor (2000).

Nesta seção limitamo-nos, também, a apresentar a formulação original das SVMs, a qual é capaz de lidar apenas com problemas de classificação binários. Existe uma série de técnicas que podem ser empregues na generalização das SVMs para a solução de problemas multiclasse. Pode-se recorrer à decomposição do problema multiclasse em vários subproblemas binários ou reformular o algoritmo de treino das SVMs em versões multiclasse. Em geral, este último procedimento produz algoritmos computacionalmente custosos (Hsu e Lin, 2002). Por esse motivo, a estratégia de decomposição é mais frequentemente. No Capítulo 18 é apresentada uma revisão de técnicas para a decomposição de problemas multiclasse em múltiplos subproblemas binários.

7.3 Considerações Finais

Este capítulo apresentou duas técnicas de ECD que abordam o problema de aprendizagem como um processo de otimização para a obtenção de um modelo preditivo a partir de um conjunto de dados: as RNAs e as SVMs.

As RNAs mais usualmente empregues na prática são as redes MLP, cujo algoritmo de treino mais popular é o *back-propagation*. O *back-propagation* é baseado no método de otimização gradiente descendente, minimizando o erro quadrático entre os rótulos desejados dos objetos e os produzidos pela rede. No caso das SVMs, recorre-se a um problema de otimização quadrático em função dos dados de treino, o qual maximiza a margem de separação entre as classes. Esse problema apresenta uma única solução global, enquanto no caso das RNAs MLP a função de erro minimizada pode possuir mínimos locais e o método gradiente pode convergir para um deles, dependendo de seus parâmetros iniciais.

Ambas as técnicas são consideradas “caixas-pretas”, no sentido de que o conhecimento

extraído é codificado em equações de difícil interpretação, em contraste com os modelos gerados por técnicas simbólicas como as árvores de decisão. Ambas também possuem parâmetros a serem estimados pelo utilizador que afetam diretamente o seu desempenho, embora o número de parâmetros seja maior no caso das RNAs.

Outra desvantagem das RNAs perante as SVMs é que os seus resultados são estocásticos e dependem fortemente da ordem de apresentação dos objetos e dos pesos iniciais atribuídos às suas conexões. Dessa forma, é recomendável executá-la várias vezes para diferentes ordenações do conjunto de treino e valores iniciais de pesos, obtendo uma média de desempenho. As SVMs, por outro lado, são determinísticas, e os seus resultados não dependem da ordem em que os objetos se encontram no conjunto de treino usado na aprendizagem. Por outro lado, foi observado que as SVMs podem ter tempos de predição maiores que as RNAs (LeCun et al., 1995).

É possível observar ainda que as RNAs e as SVMs lidam apenas com atributos com valores numéricos. No caso de dados categóricos, uma codificação numérica deve ser realizada. Em geral, a codificação mais usada é a um-para-n, em que se tem um bit para cada possível valor que o atributo categórico pode assumir. Além disso, é recomendável normalizar os atributos contínuos para evitar o domínio de atributos em intervalos numéricos maiores sobre aqueles em intervalos menores.

De uma forma geral, as SVMs e RNAs apresentam um bom desempenho preditivo em diversas tarefas de classificação, figurando entre as técnicas mais utilizadas em problemas que requerem alta precisão. Ambas são ainda consideravelmente tolerantes a ruídos. No caso das SVMs, tem-se também uma robustez a dados de alta dimensionalidade. As duas técnicas podem também ser adaptadas para gerar probabilidades de classificação dos exemplos (Haykin, 1999; Platt, 2000).

Capítulo 8

Modelos Múltiplos Preditivos

O termo *modelos múltiplos* é utilizado para identificar um conjunto de preditores cujas decisões individuais são combinadas ou agregadas de alguma forma para efetuar previsões em novos exemplos (Dietterich, 1997). Neste capítulo, iremos focar-nos problemas de classificação. No entanto, todas as técnicas apresentadas são válidas para problemas de regressão com alterações triviais relacionadas com a função de custo utilizada.

Segundo Hoeting et al. (1999), a primeira referência a uma combinação de modelos na literatura estatística surgiu em Barnard (1963), um trabalho que estuda os dados de passageiros de companhias aéreas. Contudo, a maior parte dos trabalhos anteriores na área de combinação de modelos não está presente em revistas científicas de Estatística. O artigo original sobre previsões feito por Granger e Newbold (1976) estimulou, nos anos 1970, uma onda de artigos na literatura de Economia sobre combinação de previsões de diferentes modelos de previsão.

A ideia principal subjacente a qualquer modelo múltiplo é baseada na observação de que diferentes algoritmos de aprendizagem exploram:

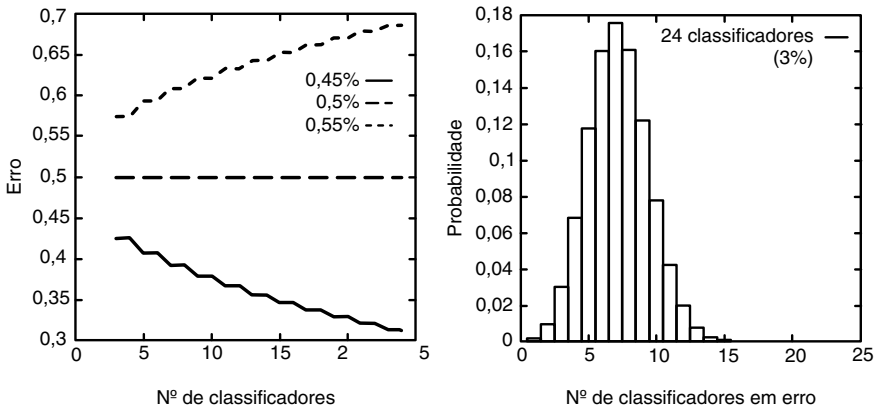
- Diferentes linguagens de representação;
- Diferentes espaços de procura;
- Diferentes funções de avaliação de hipóteses.

Será possível explorar estas diferenças? Será possível desenvolver um conjunto de classificadores que, trabalhando em conjunto, obtêm melhor desempenho do que cada um dos classificadores individuais? Estas são algumas questões que iremos abordar neste capítulo.

Sabe-se que não existe um algoritmo que seja melhor para todos os problemas de decisão. Esta observação é baseada quer em resultados teóricos (o teorema *no-free lunch* Wolpert e Macready (1997)) quer em resultados experimentais do projeto Statlog (Michie et al., 1994). A primeira observação que pode ser feita quando se trabalha com modelos múltiplos é que combinar avaliadores idênticos é inútil. Assim, uma condição necessária para que a combinação de classificadores seja útil é que os avaliadores individuais apresentem um nível substancial de desacordo.

Hansen e Salamon (1990) introduziram a hipótese de que os modelos múltiplos revestem-se de maior utilidade quando os modelos constituintes cometem erros independentes. Os autores provaram que, quando: *i)* todos os modelos têm a mesma taxa de erro, *ii)* a taxa de erro é menor que 0,5, *iii)* todos cometem erros completamente independentes, então o erro esperado do conjunto decresce linearmente com o número de modelos.

Considerando a tarefa de classificação, a Figura 8.1(a) mostra a evolução da taxa de erro obtida variando o número de classificadores no conjunto. Este é um estudo de simulação, num problema de duas classes equiprováveis (ou seja, a probabilidade de observar cada classe é 50%). Neste estudo, consideram-se conjuntos de classificadores com tamanhos que variam entre 3 e 24.



(a) Evolução da taxa de erro (b) Probabilidade de erro

Figura 8.1 *(a) Evolução da taxa de erro variando o número de classificadores num modelo múltiplo. (b) Probabilidade de que exatamente i dos 24 classificadores cometerão um erro.*

Todos os classificadores classificam os exemplos de teste. A classificação final do conjunto é obtida agregando os votos dos classificadores por votação uniforme, isto é, o voto de cada classificador tem peso 1. Ou seja, para cada exemplo, a classificação do conjunto é a classe mais votada pelos classificadores individuais. Todos os classificadores têm a mesma probabilidade de cometer um erro, mas os erros são independentes uns dos outros. Quando essa probabilidade é 45%, a taxa de erro do conjunto decresce linearmente. Quando a probabilidade é 55%, a taxa de erro cresce linearmente, e quando a probabilidade é igual a 50%, a taxa de erro do conjunto permanece constante. Se as taxas de erro de *nc* classificadores são todas iguais a $p < 0,5$ e se os erros são independentes, então a probabilidade da votação por maioria estar errada pode ser calculada como a área sob a curva da distribuição binomial para o caso em que mais de $nc/2$ classificadores estão errados (Dietterich, 1997). A Figura 8.1(b) mostra esta área para a simulação de 24 classificadores, cada um tendo um erro de 30%. A área abaixo da curva, para mais de 12

classificadores é $0,02$, que é bem menor que a taxa de erro dos classificadores individuais $(0,3)$.

Tumer e Ghosh (1995, 1996a,b) mostram a relação existente entre a taxa de erro obtida por um combinador de *nc* classificadores e a taxa de erro de um único classificador. Esta relação pode ser observada na Equação 8.1, em que ρ denota a correlação entre erros de classificadores e $Error_{Bayes}$ é a taxa de erro obtida usando a regra de Bayes, assumindo que todas as probabilidades condicionais são conhecidas. Se $\rho = 0$, o erro do conjunto decresce proporcionalmente com o número de classificadores. Quando $\rho = 1$, o erro do conjunto é igual ao erro de um único classificador.

$$Erro_{conjunto} = \frac{1+\rho(nc-1)}{nc} Erro + Erro_{Bayes} \tag{8.1}$$

Ainda que os pressupostos da análise teórica difiram das aplicações do mundo real, duas ideias principais que surgiram a partir deste estudo são:

- Combinar modelos que cometam erros não correlacionados ou preferencialmente negativamente correlacionados;
- Cada modelo deve obter um melhor desempenho do que uma escolha aleatória.

Duas questões surgem quando se trabalha com *modelos múltiplos*. A primeira é – *como combinar as previsões de diferentes modelos?* A segunda questão é – *como gerar diferentes modelos?* Estas questões serão abordadas nas próximas seções.

8.1 Combinando Previsões de Classificadores

Suponha um determinado problema de classificação onde temos disponível um conjunto de classificadores. Nesta seção, não importa como estes classificadores foram treinados. O conjunto de classificadores podem ser pessoas, especialistas do domínio ou um modelo de decisão treinado por algoritmos de aprendizagem. Denotamos o conjunto de classificadores como *classificadores de base*. Suponha agora um exemplo teste. Cada classificador de base efetua uma previsão para esse exemplo. Como podemos combinar as previsões? Quais são as vantagens de fazer essa combinação? Porquê, como e quando devemos usar um esquema de combinação? A primeira observação relevante é que os classificadores que irão ser combinados têm de efetuar previsões de forma independente, ou seja, efetuar previsões não correlacionadas.

Diz-se que dois modelos cometem um *erro correlacionado* quando *ambos* classificam um exemplo da classe y_a como pertencente à classe y_b, $y_a \neq y_b$. Ali e Pazzani (1996) apresentam uma definição precisa de erros correlacionados: Dado um conjunto de classificadores $F = \{\hat{f}_1(\mathbf{x}), ..., \hat{f}_{nc}(\mathbf{x})\}$, $\hat{f}_i(\mathbf{x}) = y$ denota que o modelo i classificou o exemplo $\mathbf{x}$ na classe y. $f(\mathbf{x})$ denota a verdadeira classe de $\mathbf{x}$. A *correlação de erro* entre dois classificadores i e j é definida como a probabilidade que os modelos $\hat{f}_i$ e $\hat{f}_j$ têm de cometer o mesmo erro:

$$\phi_{ij} = p(\hat{f}_i(\mathbf{x}) = \hat{f}_j(\mathbf{x}), \hat{f}_i(\mathbf{x}) \neq f(\mathbf{x})). \tag{8.2}$$

O grau de correlação dos erros em F, denotado por $\phi_e(F)$, pode ser definido da seguinte maneira:

$$\phi_e(F) = \frac{1}{nc(nc-1)} \sum_{i=1}^{nc} \sum_{j \neq i}^{nc} \phi_{ij} \tag{8.3}$$

em que *nc* representa o número de modelos no conjunto. Esta definição de *correlação de erro* não satisfaz a propriedade de que a correlação entre um objeto e ele mesmo deve ser 1. Para ultrapassar esta dificuldade, definimos correlação de erro entre pares de classificadores como a probabilidade condicional de dois classificadores cometerem o mesmo erro dado que um deles comete um erro. Esta definição de *correlação de erro* reside no intervalo $[0,1]$, e a correlação entre um classificador e ele mesmo é 1. A definição formal é dada por:

$$\phi_{ij} = p(\hat{f}_i(\mathbf{x}) = \hat{f}_j(\mathbf{x}) | \hat{f}_i(\mathbf{x}) \neq f(\mathbf{x}) \vee \hat{f}_j(\mathbf{x}) \neq f(\mathbf{x})). \tag{8.4}$$

Tomando em consideração que ϕ_{ij} é simétrico, a Equação 8.3 pode ser reescrita como:

$$\phi_e(F) = \frac{2}{nc(nc-1)} \sum_{i=1}^{nc} \sum_{j>i}^{nc} \phi_{ij} \tag{8.5}$$

Quando combinamos as predições dos classificadores, podemos diferenciar: **métodos de votação** *versus* **métodos de seriação**. A diferença está no tipo de saída do classificador individual. No primeiro método, o classificador de base produz um rótulo de classe. No segundo, o resultado do classificador de base é probabilístico, isto é, o classificador associa, para cada exemplo teste, uma probabilidade (ou um fator confiante) para cada possível classe. Noutra dimensão distinguimos **métodos dinâmicos** *versus* **métodos estáticos**. O esquema de combinação toma em consideração o exemplo de teste. Enquanto métodos estáticos combinam a predição de todos os classificadores no conjunto, métodos dinâmicos selecionam os classificadores mais apropriados para o exemplo de teste. Para diferentes exemplos de teste, métodos dinâmicos podem usar diferentes conjuntos de classificadores para efetuar a previsão final.

8.1.1 Métodos de Votação versus Métodos de Seriação

Na literatura de reconhecimento de padrões, os métodos discutidos nesta seção surgem com o nome de *fusão de classificadores* (Kittler, 1998). Distinguimos entre classificadores cuja previsão é uma etiqueta para a classe e classificadores cuja previsão assume a forma de distribuição de probabilidades para todos os possíveis valores da classe.

Métodos de Votação

Votação é o método mais usual para combinar classificadores. Como realçado por Ali

e Pazzani (1996), esta estratégia é motivada pela teoria da aprendizagem Bayesiana, que determina que, a fim de maximizar a precisão da predição, em vez de usar apenas um único modelo de decisão, idealmente devem ser usados todos os modelos aceitáveis no espaço da hipótese. O voto de cada hipótese deve ser ponderado pela probabilidade posterior daquela hipótese, dado o conjunto de treino. Muitas variações dos métodos de votação podem ser encontradas na literatura de aprendizagem automática. Uma delas é a *votação uniforme*, em que a opinião de todos os classificadores de base contribui igualmente para a classificação final. Outra variação comum é a *votação com peso*, em que cada classificador de base tem um peso associado, que pode mudar ao longo do tempo, e reforçando assim a classificação atribuída a um *bom* classificador.[1]

Métodos de Seriação

Obtém-se uma melhoria na votação uniforme quando cada classificador é capaz de produzir uma estimativa da *probabilidade* do exemplo pertencer a uma classe, em vez de produzir uma única etiqueta. Dado um exemplo de teste **x**, cada classificador probabilístico reporta a probabilidade de o exemplo pertencer a cada uma das classes: $p_1, \ldots, p_m$. Dado um conjunto de m classificadores probabilísticos, as probabilidades de classe de todos os modelos podem ser combinadas, por exemplo, usando a fórmula: $\frac{1}{m}\sum_{i=1}^{m} p_i$. Este método é conhecido na literatura como *soma da distribuição*. Kittler (1998) apresentou uma perspetiva teórica para este problema, através da análise da sensibilidade de várias combinações de regras na estimação dos erros, e apresentou um enquadramento unificando algumas das regras mais comuns, nomeadamente a *regra do produto*, a *regra da soma*, a *regra do mínimo*, a *regra do máximo* e a *regra da mediana*.

Kittler (1998) estudou várias estratégias para a fusão de m classificadores probabilísticos em problemas com k classes. Assumindo que representamos por P_{ik} a probabilidade dada pelo classificador i do exemplo ser da classe k, então:

- Regra da soma: $S_k = \sum_{i=1}^{m} P_{ik}$
- Regra da média: $S_k = \sum_{i=1}^{m} P_{ik}/m$
- Regra da média geométrica: $S_j = \sqrt[m]{\prod_{i=1}^{m} P_{ik}}$
- Regra do produto: $S_k = \prod_{i=1}^{m} P_{ik}$
- Regra do máximo: $S_k = \max_i P_{ik}$
- Regra do mínimo: $S_k = \min_i P_{ik}$

Após a aplicação de uma das estratégias de fusão enumeradas, cada exemplo é classificado na classe que maximize S_k.

A análise apresentada em Kittler (1998) conclui que a regra da soma é conservadora, mas amplamente utilizada, enquanto que a regra do produto é mais arriscada, mas pode produzir melhores resultados. Uma alternativa interessante consiste em tentar reduzir o conjunto de possíveis rótulos (classes) para cada exemplo de teste.

[1] Domingos (1997a) argumenta que o sucesso da proposta de modelos múltiplos deve-se, sobretudo, à redução da variação e não ao fato de constituir uma melhor aproximação da teoria da aprendizagem Bayesiana.

8.1.2 Métodos Dinâmicos versus Métodos Estáticos

A distribuição da taxa de erros sobre o espaço de atributos geralmente não é homogénea. Dependendo do classificador, a taxa de erro será mais concentrada em certas regiões do espaço de objetos do que noutras. Os métodos estáticos têm em conta as previsões de todos os elementos do conjunto, enquanto os métodos dinâmicos tomam em consideração o exemplo de teste e realizam uma *seleção do modelo* para classificar o dado exemplo de teste. A seguir serão detalhados dois métodos dinâmicos: MAI e SCANN.

MAI: *Model Applicability Induction*

Ortega (1995) propõe o *Model Applicability Induction* para combinar previsões de múltiplos modelos. A proposta consiste em caraterizar as situações em que cada modelo é capaz de fazer previsões corretas. Esta caraterização é conseguida através da aprendizagem de um *metaclassificador* para cada modelo disponível da base. O objetivo deste metaclassificador é predizer onde o modelo de base classificará corretamente o exemplo de teste.

O algoritmo genérico para construir um *metaclassificador* é apresentado no Algoritmo 8.1. Neste caso, aplicamos uma estratégia de avaliação deixar-um-de-fora. Como alternativa, por exemplo, no caso de grandes base de dados, pode ser usada validação cruzada (ver Capítulo 9). Nos dados de $Nível_1$, os exemplos positivos são os exemplos corretamente classificados pelo classificador de base ϕ, e os exemplos negativos são os incorretamente classificados. Os dados de $Nível_1$ representam sempre um problema de duas classes. As regras de classificação geradas usando os dados $Nível_1$ caraterizam os exemplos que o classificador gerado com ϕ classificou corretamente. A Tabela 8.1 ilustra os conjuntos de dados de $Nível_0$ e $Nível_1$. A tabela mostra o conjunto de dados original e o conjunto de dados de $Nível_1$. Neste último conjunto de dados, a coluna **Erro** indica se o classificador de base classificou corretamente (ou não) o exemplo de treino. A Figura 8.2 apresenta as árvores de decisão aprendidas a partir dos dados de $Nível_0$ (8.2(a)) e $Nível_1$ (8.2(b)). A árvore da Figura 8.2(b) carateriza as regiões do espaço onde a árvore da Figura 8.2(a) classifica corretamente os exemplos.

Para objetos novos, estes metaclassificadores são primeiramente consultados para selecionar o modelo de predição mais apropriado, e a predição do modelo selecionado é então devolvida. Repare que os atributos dos dois problemas são os mesmos. Um aspeto interessante desta arquitetura é que o modelo de metaclassificador é definido em termos dos atributos originais. Este modelo define as regiões do espaço de objetos onde os classificadores de base são mais (ou menos) propensos a erro.

Um estudo experimental do *Model Applicability Induction* foi apresentado em Seewald e Fürnkranz (2001). Nesse estudo, os autores usam o termo classificador *grading* para MAI. Os autores concluem:

> *"Esta proposta pode ser vista como uma generalização direta da seleção por validação cruzada, que seleciona sempre o classificador de base que corresponde ao metaconjunto de dados com a maior precisão padrão. Grading*

Algoritmo 8.1 *Model Applicability Induction*: o algoritmo para gerar metaclassificadores

Entrada: Algoritmo de aprendizagem de base ϕ
Algoritmo de aprendizagem de nível meta $\Im$
Um conjunto de treino $\mathbf{D} = \{(\mathbf{x}_i, y_i), i = 1, \ldots, n\}$
Saída: Metamodelo para ϕ em **D**

```
/* Gerar dados de Nível_1 data */
Dados Nível_1 ← {}
para cada exemplo x_i ∈ D faça
    f̂ ← φ(D − {(x_i, y_i)})
    se (y_i = f̂(x_i)) então
        Inserir {(x_i, +)} nos dados de Nível_1
    fim
    senão
        Inserir {(x_i, −)} nos dados de Nível_1
    fim
fim
Metamodelo ← ℑ(Nível_1)
Retorna: (Metamodelo)
```

supera o desempenho de estratégias de votação e a seleção por validação cruzada."

SCANN – Análise da Correspondência Seguida por um Vizinho Mais Próximo

Merz (1998) apresenta um método para combinar modelos para problemas de classificação e regressão. O método consiste em transformar a matriz de predições ($M \times N$), em que cada linha corresponde a um exemplo de treino e cada coluna corresponde a um modelo aprendido, num novo espaço de objetos de menor dimensão. Esta transformação é efetuada usando técnicas de *Análise de Correspondência*. É usada a *decomposição singular de valor* (SVD) para decompor a matriz de predições num conjunto de *modelos básicos*, não correlacionados. A decomposição pode detetar modelos redundantes, e carateriza as áreas do espaço de exemplos onde cada modelo é superior. A representação baseada em SVD ajuda a evitar os problemas associados a predições correlacionadas sem descartar nenhum modelo aprendido. A estratégia do vizinho mais próximo é então usada para classificar exemplos de teste neste novo espaço de representação.

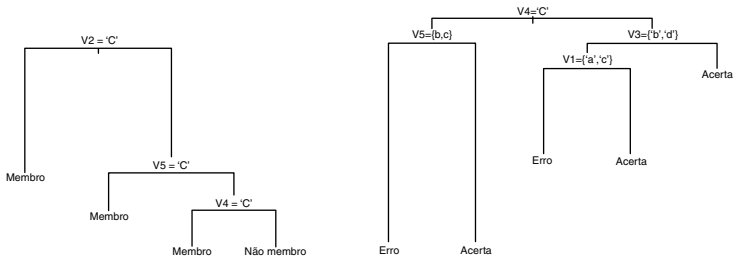
Tabela 8.1 *Exemplo ilustrativo do MAI: tabela mostra o conjunto de dados original e o conjunto de dados de* $Nível_1$

V1	V2	V3	V4	V5	Classe
t	a	c	t	a	membro
t	g	c	t	a	membro
g	t	a	c	t	não membro
a	a	t	t	g	membro
t	c	g	a	t	não membro
a	g	g	g	g	membro

Conjunto de dados original

V1	V2	V3	V4	V5	Erro
t	a	c	t	a	+
t	g	c	t	a	−
g	t	a	c	t	+
a	a	t	t	g	+
t	c	g	a	t	−
a	g	g	g	g	+

Conjunto de dados $Nível_1$



(a) Modelo-base (b) Metamodelo

Figura 8.2 *Exemplo de uma árvore de decisão originada a partir dos dados originais 8.2(a) e um metaclassificador 8.2(b), também na forma de uma árvore de decisão, gerado a partir dos dados de* $Nível_1$.

8.2 Combinando Classificadores Homogéneos

Nesta seção são analisados métodos que combinam modelos gerados por um único algoritmo. *Diversidade* é um dos requisitos quando são usados modelos múltiplos. Várias estratégias foram propostas para a geração de classificadores diferentes usando o mesmo algoritmo de aprendizagem. A maioria manipula o conjunto de treino para gerar múltiplas hipóteses. O algoritmo de aprendizagem é executado várias vezes, utilizando uma distribuição diferente de exemplos de treino em cada uma das vezes. Esta técnica funciona especialmente bem para algoritmos de aprendizagem *instáveis* – algoritmos cuja saída do classificador sofre grandes mudanças em resposta a pequenas mudanças nos dados de treino. Os métodos para combinar classificadores homogéneos podem ser agrupados pela forma como geram diversidade nos classificadores de base: por amostragem dos objetos, amostragem dos atributos, injeção de aleatoriedade e perturbação dos exemplos de teste. Métodos baseados em cada uma dessas estratégias são apresentados de seguida.

8.2.1 Métodos Baseados em Amostragem dos Exemplos de treino

***Bootstrap Aggregating - Bagging*.**

Breiman (1996a) descreve a técnica denominada *Bootstrap Aggregating - bagging*, que produz replicações do conjunto de treino por amostragem com reposição. Cada réplica do conjunto de treino tem o mesmo tamanho que os dados originais. Alguns exemplos não aparecem na amostra, enquanto outros podem aparecer mais do que uma vez. Tal conjunto de treino é denominado *bootstrap replicado* do conjunto de dados original, e a técnica é apelidada de *agregação bootstrap* (da qual o termo *bagging* é derivado). Para um conjunto de treino com n exemplos, a probabilidade de um exemplo ser selecionado é $1-(1-1/n)^n$. Para um valor de n grande, a probabilidade é cerca de $1-1/e$, em que e é a base de logaritmos naturais. Cada amostra contém, em média, 36,8% $(1/e)$ de exemplos duplicados. Para cada réplica do conjunto de treino é gerado um classificador. Todos os classificadores são usados para classificar cada exemplo no conjunto de teste, e a classificação final do exemplo é feita geralmente usando um esquema de voto uniforme. O Algoritmo 8.2 detalha o funcionamento da técnica *bagging*.

Algoritmo 8.2 O algoritmo de *bagging*

Entrada: Um algoritmo de aprendizagem ϕ
Um conjunto de treino $\mathbf{D} = \{(\mathbf{x}_i, y_i), i = 1, \ldots, n\}$
Número de Iterações Nr
Um conjunto de teste com nt exemplos $\mathbf{T} = \{(\mathbf{x}_j, ?), j = 1, \ldots, nt\}$
Saída: Previsões para o conjunto de teste

/* Fase de Aprendizagem */
para cada $l = 1$ **to** Nr **faça**
    D' $\leftarrow$ amostra com reposição de **D**
    $\hat{f}_l \leftarrow \phi(\mathbf{D}')$
fim
/* Fase de Classificação */
para cada $j = 1$ **to** nt **faça**
    $\hat{y}_j = \arg\max_{y \in Y} \sum_{l=1}^{Nr} \hat{f}_l(\mathbf{x}_j \in \mathbf{T})$
fim
Retorna: Vetor de previsões $\hat{\mathbf{y}}$

Por que o *bagging* funciona ? Intuitivamente, considerando o voto maioritário de diferentes hipóteses, a variabilidade aleatória dos classificadores individuais diminui. As árvores de decisão geram modelos instáveis sendo, por isso, o tipo de algoritmo para o qual podemos obter uma grande melhoria com *bagging*. Quando se induz uma árvore de

decisão, há pelo menos duas situações afetadas pelo *bagging*. A primeira delas é a escolha do atributo usado como teste para dividir os exemplos em cada nó. Se dois ou mais atributos são avaliados de forma semelhante por uma dada função de mérito, uma pequena mudança nos dados de treino pode mudar o atributo escolhido. Todas as subárvores descendentes também serão alteradas. O *bagging* afeta também a escolha do *ponto de corte*, para atributos *contínuos*. C4.5 seleciona um valor no conjunto de valores que aparecem no conjunto de treino. Novamente, uma pequena alteração no conjunto de treino pode conduzir a um ponto de corte diferente. Agregar os classificadores por votação uniforme gera superfícies de decisão mais complexas. A Figura 8.3 ilustra esta última situação.

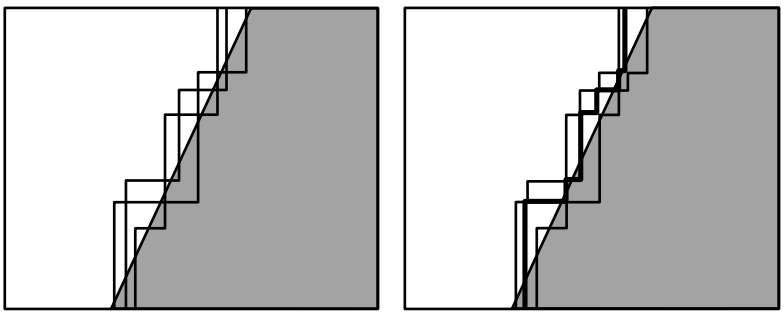


Figura 8.3 *A figura da esquerda representa as superfícies de decisão de três classificadores num problema de duas classes. A figura da direita apresenta, em negrito, a superfície de decisão obtida por votação uniforme dos três classificadores.*

É importante considerar algumas questões aquando da aplicação do *bagging*. A primeira questão relevante é: Quantas amostras são suficientes? A intuição de Breiman (1996a) sugere que: *"mais replicações são necessárias com um aumento no número de classes"*. Breiman também nota que o *"bagging é quase um procedimento de sonho para a computação paralela"*, assumindo que os tempos de execução para um grande número de amostras não é tão relevante. Numa experiência de simulação, variando o número de amostras, verificou-se que um número de exemplos moderado (e.g. 10) é, geralmente, suficiente.

Outra questão a ser considerada é: Todos os modelos gerados por um *bagging* serão úteis para predição? Poderemos eliminar alguns desses modelos? Uma proposta comum para estes problemas consiste em avaliar cada modelo em exemplos *out-of-bag*. Em amostragem com reposição, cerca de 1/3 dos exemplos não será utilizado no treino. Esses exemplos são denominados de exemplos *out-of-bag*. Estes exemplos podem ser usados para estimar o desempenho de um modelo gerado. Se o desempenho está abaixo de algum limiar, o modelo é deixado fora do conjunto e não é usado para fazer predições no conjunto de teste. Um trabalho interessante nesta linha foi apresentado por Martínez-Muñoz e Suárez (2006). Os autores apresentam um método para ordenar os classificadores gerados

no *bagging* e mostram que um subconjunto destes classificadores conduz a um aumento de desempenho.

Concluindo, o *bagging* é um modo simples e fácil para melhorar qualquer método existente. Tudo o que é necessário é, em primeiro lugar, adicionar um ciclo para selecionar a amostra de treino e enviá-la para o algoritmo de aprendizagem, e um procedimento posterior faz a soma dos votos. Em comparação com as árvores, o que se perde é a estrutura simples e interpretável; o que se ganha é um aumento de precisão.

Outro método de amostragem que pode ser usado para construir conjuntos de treino consiste em considerar conjuntos disjuntos dos dados de treino (Dietterich, 1997; Heath et al., 1996). Por exemplo, o conjunto de treino pode ser dividido aleatoriamente em 10 subconjuntos disjuntos. Então, conjuntos de treino sobrepostos podem ser construídos descartando um conjunto diferente desses 10 subconjuntos. Este procedimento é utilizado para construir conjuntos de treino para 10-validação cruzada. Os conjuntos construídos deste modo são usualmente denominados de *comités de validação cruzada.*

Classificadores Fracos e Fortes: *Boosting*

Nos anos 1980, Michael Kearns colocou uma pergunta à comunidade científica:

> *Poderá um conjunto de classificadores fracos gerar um classificador forte?*

Um classificador *fraco* é definido como um classificador cuja capacidade de generalização é pouco melhor que a escolha aleatória. Por outro lado, um classificador *forte* pode aproximar qualquer distribuição com um erro arbitrariamente pequeno. Formalmente, a pergunta é enunciada da seguinte forma:

> *Sendo dados uma margem de erro* ε*, e um nível de confiança* $1-\delta$*, é possível construir um classificador que, com probabilidade* $1-\delta$*, gera uma hipótese com erro* ε *para qualquer distribuição de exemplos gerados para um problema?*

Schapire (1990) responde à pergunta propondo um método geral (*boosting*) para converter um *classificador fraco* num classificador que alcança uma precisão arbitrariamente alta. Um algoritmo de aprendizagem *fraco* tem um desempenho ligeiramente melhor que a escolha aleatória. O trabalho de Schapire mostra como amplificar esses classificadores fracos de forma a obter uma precisão arbitrariamente alta. O algoritmo originalmente desenvolvido foi baseado no modelo teórico de aprendizagem PAC (*Probably Approximately Correct*) (Mitchell, 1997).

A ideia principal subjacente ao algoritmo de *boosting* consiste em associar um peso a cada exemplo no conjunto de treino, que reflita a sua importância. O ajuste de pesos de maneira diferente faz com que o classificador se foque em exemplos diferentes levando a diferentes classificadores. *Boosting* é um algoritmo iterativo. Em cada iteração é gerado um novo classificador. Esse classificador é treinado com a distribuição dos exemplos dado pelos pesos associados. Os pesos são ajustados de acordo com o desempenho do conjunto

de classificadores aprendidos até essa iteração. O peso dos exemplos classificados incorretamente aumenta, enquanto o peso dos exemplos corretamente classificados diminui. O classificador final agrega os classificadores aprendidos em cada iteração pela votação pesada. O peso de cada classificador é uma função da sua precisão.

A investigação em técnicas de *boosting* é bastante ativa. Freund e Schapire (1996) apresentaram o algoritmo *AdaBoost* (de *Adaptive Boosting*), mostrado no Algoritmo 8.3. À semelhança do *bagging*, este algoritmo gera um conjunto de classificadores que participam na classificação de exemplos de teste por votação ponderada. *Boosting* gera classificadores sequencialmente. Em cada iteração, o algoritmo muda o peso dos exemplos do treino levando em consideração o erro do conjunto de classificadores construídos previamente.

Exemplo Ilustrativo. Os exemplos seguintes ilustram o processo de *boosting*. O *classificador fraco* desenha superfícies de decisão que consistem num único hiperplano perpendicular a um dos eixos no espaço de entrada. A Figura 8.4 ilustra a primeira iteração. A distribuição dos pesos é uniforme, isto é, todos os exemplos têm o mesmo peso. Na figura, o peso dos exemplos é ilustrado pelo tamanho do círculo ao redor dos objetos. O classificador fraco encontra o hiperplano que minimiza o erro nessa distribuição.

Na segunda iteração (Figura 8.5), a distribuição dos pesos muda de acordo com o erro dos modelos gerados na primeira iteração. O peso dos exemplos classificados incorretamente é aumentado, enquanto o peso dos exemplos classificados corretamente diminui. O novo conjunto de pesos define outra distribuição dos exemplos. O classificador fraco gera um modelo minimizando o erro da distribuição atual.

Discussão. O bom desempenho verificado com o algoritmo de *boosting* pode ser compreendido através de duas observações. A primeira observação, que se ajusta a muitos problemas do mundo real, é que os exemplos observados tendem a apresentar níveis de dificuldade de classificação diferentes. Por exemplo, os exemplos que se situam perto da superfície de decisão são mais difíceis de classificar do que os exemplos mais afastados. Para tais problemas, o algoritmo *boosting* tende a gerar distribuições que se concentram em exemplos difíceis de classificar. Por outro lado, é requerido que o algoritmo de aprendizagem seja sensível à distribuição dos exemplos de treino, de forma a gerar hipóteses significativamente diferentes quando a distribuição do conjunto de treino se altera. Esta propriedade está relacionada com a segunda razão apontada para a melhoria verificada com o *boosting* – a redução da variância. Intuitivamente, considerando a maioria ponderada sobre muitas hipóteses, o *boosting* tem o efeito de reduzir a variabilidade aleatória das hipóteses individuais.

8.2.2 Métodos Baseados na Amostragem do Conjunto de Atributos

A utilização de *bagging* ou *boosting* com algoritmos do tipo *naive* Bayes ou *k*-vizinhos mais próximos não é unânime. A utilização de *bagging* não é efetiva devido à estabilidade

Algoritmo 8.3 O algoritmo *ADABOOST*

Entrada: Um algoritmo de aprendizagem ϕ
Um conjunto de treino $\mathbf{D} = \{(\mathbf{x}_i, y_i), i = 1, \ldots, n\}$
Número de Iterações Nr
Um conjunto de teste com nt exemplos $\mathbf{T} = \{(\mathbf{x}_j, ?), j = 1, \ldots, nt\}$
Saída: Previsões para o conjunto de teste

/* Fase de treino */ ;
para cada *exemplo* $i \in \mathbf{D}$ **faça**
    $w(i) \leftarrow 1/n$;
fim
para cada $l = 1$ **to** Nr **faça**
    para cada *exemplo* $i \in \mathbf{D}$ **faça**
        $p_l(i) \leftarrow w_l(i) / \sum_i w_l(i)$;
    fim
    /* Chamada ao Algoritmo de Aprendizagem */ ;
    $\hat{f}_l \leftarrow \phi(p_l)$;
    /* Calcular o Erro */ ;
    $e_l = \sum_i p_l(i)[\hat{f}_l(\mathbf{x}_i) \neq y_i]$;
    $\beta_l \leftarrow e_l/(1 - e_l)$;
    para cada *exemplo* $i \in D$ **faça**
        $w_{l+1}(i) := w_l(i)\beta_l^{1-[\hat{f}_l(\mathbf{x}_i) \neq y_i]}$;
    fim
fim
/* Fase de Teste */ ;
para cada $j = 1$ **to** nt **faça**
    $\hat{y}_j = \arg\max_{y \in Y} \sum_{l=1}^{Nr} (log \frac{1}{\beta_l})[\hat{f}_l(\mathbf{x}_j \in \mathbf{T}) = y]$;
fim
Retorna: Vetor de previsões $\hat{\mathbf{y}}$;

dos modelos de decisão quando se perturba o conjunto de exemplos de treino. Zheng (1998) aplicou *boosting* com um classificador *naive* Bayes e declara:

> *"Implementamos um algoritmo de boosting com um classificador naive Bayes usando um método semelhante àquele que é utilizado para uma árvore de decisão. Apesar do algoritmo alcançar maior precisão que o classificador naive Bayes em alguns domínios, a melhoria na precisão sobre o classificador naive Bayes num grande conjunto de domínios naturais é muito marginal. A razão deve estar relacionada com o fato de, implicitamente, o boosting requerer a instabilidade do sistema de aprendizagem que usou o boosting."*

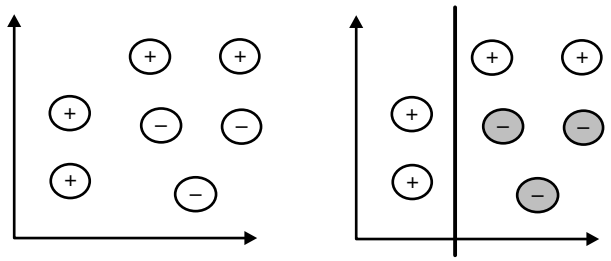


Figura 8.4 *Boosting: primeira iteração. A figura da esquerda mostra a distribuição uniforme original dos exemplos. A figura da direita mostra a superfície de decisão para essa distribuição.*

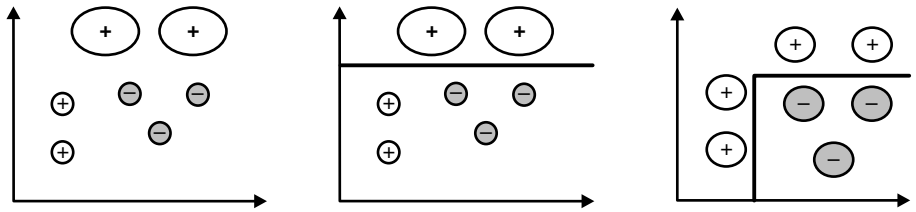


Figura 8.5 *Boosting: segunda iteração. O peso dos exemplos classificados incorretamente na primeira iteração aumentou. A figura do centro mostra a superfície para a nova distribuição dos exemplos. O modelo final, obtido conjugando os dois modelos anteriores, é apresentado na figura da direita.*

Uma melhoria efetiva de um comité de classificadores *naive* Bayes foi obtida usando diferentes subconjuntos de atributos. Um classificador *naive* Bayes não é estável no sentido de que uma mudança no conjunto de atributos pode conduzir a muitos classificadores diferentes. Além disso, devido à suposição de independência do atributo, um classificador *naive* Bayes construído a partir de um subconjunto de atributos pode apresentar melhor desempenho do que um classificador *naive* Bayes gerado usando todos os atributos. Esta técnica foi também usada por Skalak (1997) e Bay (1998) para comités dos classificadores k-vizinhos mais próximos.

8.2.3 Métodos Baseados na Injeção de Aleatoriedade

Alguns algoritmos de aprendizagem usam parâmetros inicializados aleatoriamente. Esta caraterística pode ser explorada no sentido de gerar diferentes modelos pela injeção de aleatoriedade nas entradas ou parâmetros do algoritmo de aprendizagem.

Em redes neuronais, por exemplo, um método comum para gerar diferentes redes usando o mesmo algoritmo de treino e o mesmo conjunto de dados consiste em inicializar os pesos da rede com diferentes valores.

Breiman (2001) introduziu o algoritmo das *florestas aleatórias* (*random forests*). O

modelo gera várias árvores de decisão cujas previsões são combinadas por votação uniforme. O algoritmo usa a amostragem de exemplos com reposição do *bagging* combinada à seleção aleatória de atributos. Considere um conjunto de treino **D** com n exemplos e d atributos. O algoritmo das florestas aleatórias requer dois parâmetros: L é o número de árvores da floresta, e i ($i << d$) é o número de atributos a considerar para testes de decisão em cada nó da árvore de decisão. As florestas aleatórias geram L árvores de decisão. Cada árvore de decisão é induzida a partir de uma amostra com reposição do conjunto de treino. A amostra tem exatamente n exemplos, alguns dos quais repetidos. Os exemplos do conjunto de treino original **D** que não estão na amostra vão ser utilizados para estimar o desempenho da árvore. O algoritmo para construir a árvore é em tudo semelhante ao algoritmo apresentado na Seção 6.1, com uma única diferença. Em cada nó da árvore, para escolher o atributo de teste, apenas são considerados i atributos escolhidos de forma aleatória. Nenhuma das árvores é podada.

Os processos de amostragem, quer dos exemplos, quer dos atributos, vão provocar um aumento da variabilidade das árvores induzidas. Resultados experimentais revelaram que este algoritmo é dos mais competitivos.

8.2.4 Métodos Baseados na Perturbação dos Exemplos de Teste

Nesta seção abordamos técnicas que utilizam um único modelo e adiam, para o estágio de predição, a geração de múltiplas predições pela perturbação do vetor de atributos correspondente ao caso de teste.

Um dos métodos mais ilustrativos desta técnica foi apresentado por Geurts (2000, 2001). O autor apresenta o DPC (do inglês, *Dual Perturba e Combina*), que pode ser aplicado no topo de qualquer modelo produzido por qualquer algoritmo de aprendizagem. Um único modelo é gerado a partir de um conjunto de treino. Na fase de predição, cada exemplo de teste sofre, várias vezes, perturbações. Para inserir perturbações no exemplo de teste, é adicionado um ruído branco ao valor do atributo. O modelo preditivo faz uma predição para cada versão do exemplo de teste com perturbações. A predição final é obtida pela agregação de predições diferentes. Geurts (2000) apresenta evidências experimentais de que este método é eficiente na redução da variância. O algoritmo principal é mostrado no Algoritmo 8.4.

8.3 Combinando Classificadores Heterogéneos

Uma maneira de garantir a diversidade dos classificadores de base é através da utilização de diferentes algoritmos na produção dos classificadores. Neste caso, obtém-se um conjunto heterogéneo de classificadores para combinar.

Algoritmo 8.4 Algoritmo Dual Perturba e Combina (DPC)

Entrada:
Um conjunto de exemplos de teste $\mathbf{T} = \{(\mathbf{x}_i, ?), i = 1, \ldots, n\}$
Um número de Iterações Nr
Um modelo de classificação (regressão) $\hat{f}$
Saída: Vetor de previsões para o conjunto de teste;

1 **para cada** $\mathbf{x}_i \in \mathbf{T}$ **faça**
2 **para cada** $l = 1$ **to** Nr **faça**
3 seja $\mathbf{x}_{i\varepsilon^l}$ uma variante perturbada de $\mathbf{x}_i$, em que ε^l é obtido por uma distribuição gaussiana $N(0, \lambda_l.\sigma_l)$;
4 $p_l \leftarrow \hat{f}(\mathbf{x}_{i\varepsilon^l})$;
5 **fim**
6 Calcula a previsão agregada, dada por: $\hat{y}_i = aggr_{l=1}^{Nr} p_l$, em que $aggr$ é um operador de agregação (média, mediana etc.);
7 **fim**
8 **Retorna:** Vetor de previsões $\hat{\mathbf{y}}$;

8.3.1 Generalização em Pilha

Wolpert (1992) propôs o método *Generalização em Pilha*,[2] que possui uma arquitetura de aprendizagem em camadas. Os classificadores no $Nível_0$ recebem como entrada os dados originais, e cada classificador produz uma predição. Camadas sucessivas recebem como entrada as predições das camadas imediatamente precedentes, e a saída é passada para a próxima camada. Um único classificador no nível mais alto produz a predição final.

Generalização em pilha é um processo para minimizar o erro de generalização usando classificadores nas camadas mais altas para aprender o tipo de erro cometido pelo classificador imediatamente abaixo. Nessa perspetiva, ela pode ser vista como uma extensão para métodos de seleção de modelo, denominados *validação cruzada*, que usa uma estratégia do tipo "o vencedor leva tudo". Um único classificador com erro de validação cruzada mais baixo é selecionado. A ideia subjacente à generalização em pilha é que pode haver um modo mais inteligente para usar o conjunto de classificadores. A regra dos classificadores dos níveis mais altos é aprender como os classificadores anteriores cometem erros, em qual classe eles concordam ou discordam, e usar o seu conhecimento para fazer predições.

A maioria dos trabalhos usando a arquitetura de pilha, por exemplo Wolpert (1992), Ting e Witten (1997), Skalak (1997) e Breiman (1996c), concentra-se na arquitetura de duas camadas, que é ilustrada na Tabela 8.2 e na Figura 8.6. A Tabela 8.2 mostra o conjunto de dados original e o conjunto de dados de $Nível_1$. Neste último caso, cada coluna,

[2] *Stacked Generalization* ou *Stacking*.

designada por P_{ik}, representa a probabilidade, dada pelo classificador i, do exemplo pertencer à classe k. Neste caso, há duas fases diferentes: a fase de treino, ou aprendizagem, e a fase de aplicação. A fase de aprendizagem consiste nos seguintes passos:

1. Treinar cada um dos classificadores Nível_0 usando validação cruzada com o método deixar-um-de-fora da seguinte forma: para cada exemplo no conjunto de treino deixe um de fora e treine com os demais exemplos. Depois do treino, classifique o exemplo excluído. Crie um vetor a partir das predições de todos os classificadores $N\acute{i}vel_0$ e a classe atual daquele exemplo.
2. Treinar o classificador $N\acute{i}vel_1$, usando como conjunto de treino a coleção de vetores gerados nos passos anteriores. O número de exemplos nos dados $N\acute{i}vel_1$ é igual ao número de exemplos no conjunto de treino original.
3. No passo 1, os classificadores são gerados usando o método deixar-um-de-fora. Para explorar completamente o conjunto de treino, todos os classificadores $N\acute{i}vel_0$ são treinados usando o conjunto de treino inteiro. Os modelos gerados são usados para classificar os exemplos no conjunto de teste.

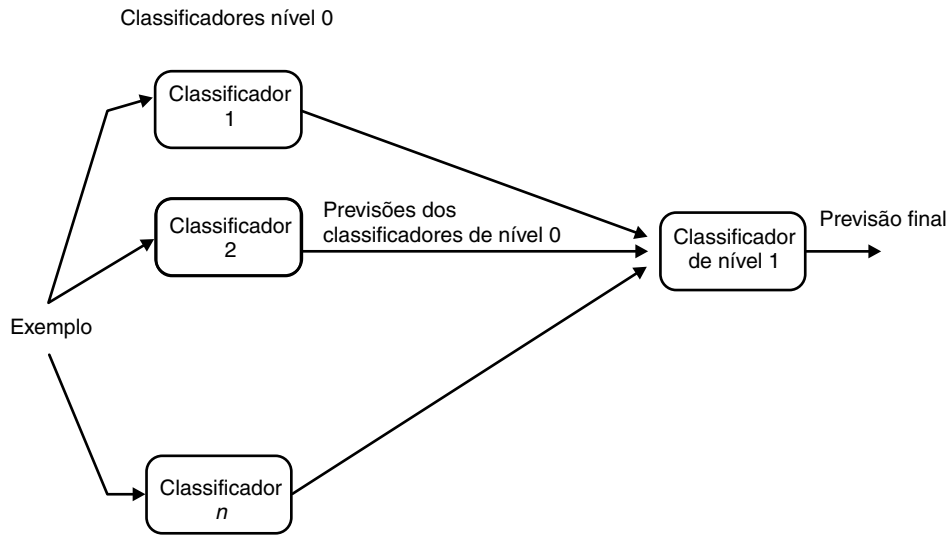


Figura 8.6 *Arquitetura do método de generalização em pilha.*

Na fase de aplicação, quando um novo exemplo é apresentado, este é classificado por todos os classificadores $N\acute{i}vel_0$. O vetor de predições é então classificado pelo classificador $N\acute{i}vel_1$, que produz como saída a predição final para o exemplo (ver Figura 8.6).

O enquadramento geral não é restrito ao modelo básico descrito. Por exemplo, Breiman (1996c) verificou que em alguns problemas de regressão foram obtidos melhores

Tabela 8.2 *Exemplo ilustrativo do método de generalização em pilha: A tabela mostra o conjunto de dados original e o conjunto de dados de* $Nível_1$.

V1	V2	V3	V4	V5	Classe	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	$P_{3,2}$	Classe
t	a	c	t	a	Membro	0,51	0,49	0,13	0,87	0,12	0,88	Membro
t	g	c	t	a	Membro	0,19	0,81	0,07	0,93	0,81	0,19	Membro
g	t	a	c	t	Não Membro	0,68	0,32	0,55	0,45	0,69	0,31	Não Membro
a	a	t	t	g	Membro	0,74	0,26	0,66	0,34	0,94	0,06	Membro
t	c	g	a	t	Não Membro	0,62	0,38	0,01	0,99	0,78	0,22	Não Membro
a	g	g	g	g	Membro	0,65	0,35	0,90	0,10	0,55	0,45	Membro
Conjunto de dados original						Conjunto de dados de $Nível_1$						

resultados usando *10-fold cross-validation* em vez de *leave-one-out*. Ting e Witten (1997) observaram empiricamente que, usando distribuições de probabilidade de classe como atributos no $Nível_1$ e um discriminante linear como classificador $Nível_1$, são obtidos melhores resultados. Em todos os casos, a generalização em pilha é uma técnica sofisticada para reduzir o erro devido a redução do viés (Breiman, 1996c; Skalak, 1997).

8.3.2 Generalização em Cascata

Generalização em cascata (Gama e Brazdil, 2000) é uma composição sequencial de classificadores que em cada nível de generalização aplica um operador construtivo. O operador construtivo constrói novos atributos. Dados um conjunto de treino **D**, um conjunto de teste **T** e dois algoritmos $\Im_1$ e $\Im_2$, a generalização em cascata procede como se segue: o algoritmo $\Im_1$ gera um classificador, $\hat{f}_1$, usando o conjunto de treino **D**. O modelo gerado, $\hat{f}_1$, classifica todos os exemplos de treino e teste. Assume-se que o resultado de aplicar o modelo $\hat{f}_1$ a um exemplo é uma distribuição de probabilidade da classe. Ou seja, um vetor, **c**, com a dimensão igual ao número de classes, em que cada elemento do vetor é a probabilidade de esse exemplo pertencer a uma das classes. O operador construtivo concatena cada exemplo **x** com o vetor **c**. O resultado da aplicação do operador construtivo é um novo conjunto de dados, com o mesmo número de exemplos do conjunto original, mas em que cada exemplo é acrescido de novos atributos, um novo atributo para cada classe. Cada novo atributo é a probabilidade de o exemplo pertencer a uma das classes dado pelo modelo $\hat{f}_1$. A aplicação do operador construtivo gera os conjuntos de dados ditos de $Nível_1$. O classificador $\Im_2$ aprende com os dados de treino $Nível_1$ e classifica os dados de teste $Nível_1$.

Estes passos representam a sequência básica da generalização em cascata. A composição dos dois modelos é representada formalmente pela expressão: $\Im_2 \nabla \Im_1$. Essa é a fórmula mais simples da generalização em cascata. Algumas possíveis extensões incluem a composição de *nc* classificadores e a composição paralela de classificadores. Uma composição de *nc* classificadores é representada por:

$$\Im_{nc} \nabla \Im_{nc-1} \nabla \Im_{nc-2} ... \nabla \Im_1$$

Neste caso, a generalização em cascata gera $nc-1$ níveis de dados. O modelo final é o modelo dado pelo classificador $\Im_{nc}$. Este modelo pode conter termos na forma de condições baseados nos atributos construídos pelos classificadores anteriormente construídos.

Exemplo Ilustrativo Neste exemplo, considera-se o conjunto de dados UCI `Monks-2` (Thrun et al., 1991). Os conjuntos de dados `Monks` descrevem o domínio de um robô artificial e são muito conhecidos na comunidade de Aprendizagem Automática. Os robôs são descritos por seis atributos e classificados numa das duas classes. Escolheu-se o problema `Monks-2` porque se sabe que esta é uma tarefa difícil para sistemas que aprendem usando árvores de decisão. A regra de decisão para este problema é: *O robô é amigo, se exatamente 2 dos seis atributos assumem o 1º valor do domínio*. A regra de decisão combina os diferentes atributos de um modo que se torna complicado descrever na Forma Normal Disjuntiva usando somente os atributos originais. Alguns exemplos dos dados de treino são apresentados na Tabela 8.3.

Tabela 8.3 *Dois exemplos do conjunto de dados de $Nível_0$ no problema* `Monks-2`

Cabeça	Corpo	Sorriso	Objeto	Cor	Gravata	Classe
redonda	redondo	sim	espada	vermelho	sim	inimigo
redonda	redondo	não	balão	azul	não	amigo

Usando *10-fold cross-validation*, a taxa de erro do C4.5 é 32,9%, e para o *naive* Bayes é 34,2%. O modelo composto C4.5 depois de *naive* Bayes, C4.5 ∇ *naive* Bayes opera como se segue. O *naive* Bayes aprende um classificador usando o conjunto de treino de $Nível_0$. Este classificador classifica os exemplos de treino e de teste de $Nível_0$, ou seja, para cada exemplo retorna a probabilidade de o exemplo pertencer a cada classe. Como existem duas classes, obtêm-se duas probabilidades. Cada exemplo é estendido com dois novos atributos dando origem aos conjuntos de dados de $Nível_1$. O exemplo mostrado anteriormente tem a forma ilustrada na Tabela 8.4. Os novos atributos *P(amigo)* e *P(inimigo)* representam a probabilidade de que o exemplo pertença às classes *amigo* e *inimigo*, respetivamente.

Tabela 8.4 *Dois exemplos do conjunto de dados de $Nível_1$*

Cabeça	Corpo	Sorriso	Objeto	Cor	Gravata	P(amigo)	P(inimigo)	Classe
redonda	redondo	sim	espada	vermelho	sim	0,135	0,864	inimigo
redonda	redondo	não	balão	azul	não	0,303	0,696	amigo

C4.5 é treinado nos dados de treino do $Nível_1$ e classifica os dados de teste do $Nível_1$. A composição C4.5 ∇ *naive* Bayes obtém uma taxa de erro de 8,9%, que é substancialmente menor que a taxa de erro tanto de C4.5 quanto de *naive* Bayes. Nenhum dos algoritmos de

forma isolada pode capturar o conceito subjacente ao problema. Nesse caso, a generalização em cascata alcançou um desempenho notável. A Figura 8.7 apresenta uma das árvores geradas pelo C4.5 ∇ *naive* Bayes.

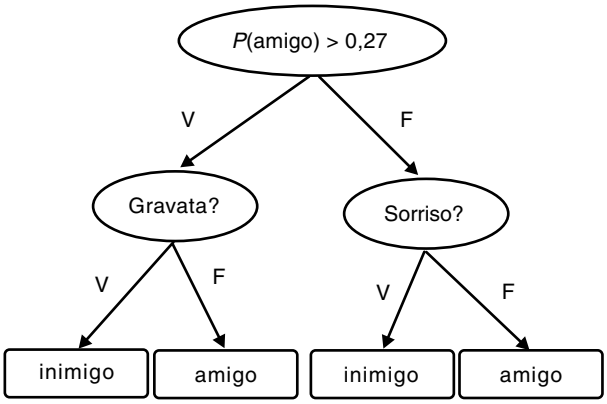


Figura 8.7 *A árvore gerada pelo C4.5 ∇ naive Bayes.*

A árvore contém uma mistura de alguns dos atributos originais (*Sorriso*, *Gravata*) com alguns dos novos atributos construídos pelo *naive* Bayes (*P(amigo)*, *P(inimigo)*). Na raiz da árvore aparece o atributo *P(amigo)*. Este atributo representa a probabilidade de um exemplo pertencer à classe *amigo*, calculada pelo *naive* Bayes. A árvore de decisão gerada pelo C4.5 usa os atributos construídos pelo *naive* Bayes, mas redefinindo diferentes limiares. Sendo um problema de duas classes, os modelos Bayesianos usam *P(amigo)* com limiar 0,5, enquanto a árvore de decisão ajusta o valor do limiar para 0,27. Os nós de decisão com testes nos atributos construídos são um tipo de função dada pela estratégia Bayesiana. Por exemplo, o atributo *P(amigo)* pode ser visto como uma função que calcula $p(Classe = amigo|\mathbf{x})$ usando o teorema de Bayes. Em alguns ramos, a árvore de decisão realiza mais do que um teste de probabilidade de classes. De certa forma, esta árvore de decisão combina duas linguagens de representação: a representação do *naive* Bayes com a linguagem da árvore de decisão. O passo construtivo executado pela generalização em cascata insere novos atributos que incorporam novo conhecimento fornecido pelo *naive* Bayes. Este aspeto permite o aumento significativo do desempenho verificado com a árvore de decisão.

Discussão. A generalização em cascata pertence à família de algoritmos de generalização em pilha. Wolpert (1992) define generalização em pilha como um contexto geral para combinar classificadores. Este contexto envolve a obtenção de predições de vários classificadores e a utilização dessas predições como a base para o próximo estágio de classificação.

A generalização em cascata pode ser considerada um caso especial de generalização

em pilha, principalmente devido à estrutura de aprendizagem em camadas. Alguns aspetos que fazem a generalização em cascata particular são:

- Os novos atributos são contínuos, uma vez que este método obtém a forma da distribuição de probabilidade da classe. Combinar classificadores através de classes categóricas perde a força do classificador na sua predição. O uso da distribuição das classes de probabilidades permite-nos explorar essa informação.
- Todos os classificadores têm acesso aos atributos originais. Qualquer novo atributo construído por um classificador mais baixo é considerado exatamente do mesmo modo que qualquer um dos atributos originais.
- Generalização em cascata não usa validação cruzada interna. Este aspecto afeta a eficiência computacional deste método.

Muitas destas ideias foram discutidas na literatura. Ting e Witten (1997) usaram a distribuição de probabilidade da classe como atributos $N\acute{i}vel_1$, mas não usaram os atributos originais. Chan e Stolfo (1995a) usaram os atributos originais e as predições das classes num esquema denotado como *combinador-atributo-classe*. Explorar todos estes aspetos é o que torna a generalização em cascata bem-sucedida. Porém, esta combinação particular implica algumas diferenças *conceituais*.

- Enquanto a generalização em pilha tem uma natureza paralela, a generalização em cascata é sequencial. O efeito é que os classificadores intermediários têm acesso aos atributos originais mais as predições dos classificadores dos níveis anteriores.
- O objetivo final da generalização em pilha é combinar predições. O objetivo da generalização em cascata é obter um modelo capaz de usar termos na linguagem de representação dos classificadores dos níveis anteriores. Na generalização em cascata, os classificadores de mais baixo nível adiam a decisão final para os classificadores de alto nível.
- A utilização da generalização em cascata usa sequências de poucos classificadores. O perfil para os classificadores no início da sequência é baixa variância. Os classificadores no fim da sequência deverão ter baixo enviesamento.

8.3.3 Meta-Aprendizagem

Chan e Stolfo (1995a) apresentam dois esquemas para combinação de classificadores: *árbitro* e *combinador*. Ambos os esquemas são baseados em meta-aprendizagem, em que um metaclassificador é gerado a partir de metadados, com base nas predições dos classificadores de base. Um árbitro é também um classificador, e é usado para arbitrar entre predições geradas por diferentes classificadores de base. O conjunto de treino para o árbitro é obtido a partir de todos os dados disponíveis usando regras de seleção. Um exemplo de uma regra de seleção é *"Selecione os exemplos cuja classificação não pode ser predita consistentemente usando classificadores de base."* Este árbitro, juntamente com uma regra

de arbitragem, determina a classificação final com base nas predições de base. Um exemplo de uma regra de arbitragem é: *"Use a predição do árbitro quando o classificador de base não obtém a maioria"*. Mais tarde (Chan e Stolfo, 1995b, 1997), este *framework* foi estendido usando árbitro/combinadores de forma hierárquica, gerando árvores binárias de árbitros/combinadores. Uma árvore de árbitros/combinadores é uma estrutura hierárquica composta de árbitros/combinadores que são calculados numa árvore binária de um modo de baixo-para-cima (*bottom-up*). Um árbitro/combinador é inicialmente aprendido a partir da saída de um par de classificadores base, e, recursivamente, um árbitro/combinador é aprendido a partir da saída de dois árbitros. Para *nc* classificadores, há $\log_2(nc)$ níveis gerados.

8.3.4 Sistemas Híbridos

Resultados obtidos em comparações empíricas de algoritmos de aprendizagem existentes (Michie et al., 1994) ilustram que cada algoritmo apresenta uma certa *superioridade seletiva*. Ou seja, o algoritmo pode ser melhor nalgumas tarefas, mas não em todas. Esta é a motivação por detrás do desenvolvimento de algoritmos que podem ser adequados aos dados com representações *heterogéneas*. Isto é, diferentes regiões do espaço de entrada são aproximadas usando diferentes tipos de modelos.

Domingos (1998) propõe CMM, (do inglês *Combined Multiple Models*), um meta-classificador que procura reter os maiores ganhos de precisão de propostas de modelos múltiplos enquanto produz um único modelo compreensível. CMM gera um novo conjunto de treino, composto de um grande número de exemplos gerados e classificados de acordo com um conjunto, mais os exemplos originais. CMM foi usado com regras C4.5 como um classificador de base, e com *bagging*, como a metodologia de modelos múltiplos. Em 26 conjuntos de dados de referência, CMM retém, em média, 60% dos ganhos de precisão obtidos pelo *bagging* relativo a uma única execução das regras C4.5, enquanto produz um conjunto de regras cuja complexidade é tipicamente um pequeno múltiplo da complexidade das regras C4.5.

Brodley (1995, 1993) apresenta uma *Seleção do Modelo de Classe - sistema MCS*, um algoritmo híbrido que combina, numa única árvore, nós que são testes invariantes, testes multivariados gerados por *máquinas lineares* e classificadores baseados em instâncias. Cada nó MCS usa um conjunto de regras *Se-Então* para executar uma pesquisa heurística *best-first* para a melhor hipótese para uma dada partição do conjunto de dados. O conjunto de regras incorpora conhecimento do especialista. Uma dessas regras é:

> **se** o número de exemplos é inferior à capacidade do hiperplano
> **então** encontra o melhor teste univariado
> **caso contrário** gera uma combinação linear LT_n usando todos os atributos.

MCS usa uma estratégia de controlo para executar uma seleção automática do modelo. *MCS* constrói árvores que aplicam diferentes modelos em diferentes regiões do espaço de objetos.

Kohavi (1996) apresenta um sistema híbrido que gera uma árvore de decisão invariante usando classificadores *naive* Bayes nas folhas da árvore. Esta proposta tenta explorar as vantagens, quer das árvores de decisão (i.e., segmentação), quer do *naive* Bayes (acumulação de evidência de múltiplos atributos). O autor alega que o modelo retém a interpretabilidade do *naive* Bayes e das árvores de decisão, enquanto resulta em classificadores que frequentemente superam ambos os constituintes, especialmente em grandes base de dados.

Domingos (1997b, 1996) apresenta um sistema denominado *RISE* (*Rule Induction from a Set of Exemplars*) que unifica a aprendizagem baseado em regras e a aprendizagem baseado em exemplos. Os exemplos são tratados como regras de especificidade máxima. As regras são aprendidas gradualmente pela generalização de exemplos até que não se consiga alcançar mais melhorias na precisão. Os exemplos de teste são classificados usando a regra, que pode ser um exemplo, mais próxima da base de conhecimentos.

8.4 Considerações Finais

Um conjunto de preditores é, em si, um algoritmo de aprendizagem supervisionado. O modelo múltiplo representa uma hipótese; no entanto, esta hipótese não está necessariamente contida no espaço de hipóteses dos modelos base a partir dos quais a hipótese é construída. Assim, os conjuntos podem ter maior flexibilidade nas funções que representam. Muitos métodos procuram promover a diversidade entre os modelos que combinam, por exemplo, perturbando a distribuição dos exemplos no conjunto de treino. *Bagging* e *boosting* Bauer e Kohavi (1999) constituem duas das formas mais eficientes para melhorar a precisão de classificadores, tais como árvores de decisão e redes neuronais.

Quinlan (1996) e Bauer e Kohavi (1999) concordam que o *boosting* apresenta maiores benefícios, mas produz degradações severas em alguns conjuntos de dados. Ali e Pazzani (1996) mostram que o número de exemplos treinados necessários para o *boosting* aumenta em função da precisão do modelo aprendido. Bauer e Kohavi (1999) referem que o principal problema com o *boosting* é a falta de robustez em dados com ruído. Isto é expectável uma vez que exemplos ruidosos tendem a ser classificados incorretamente, sendo o seu peso consequentemente aumentado. A maioria dos estudos em *boosting* e *bagging* requer um número considerável de classificadores. Por exemplo, Bauer e Kohavi (1999) usam 25 classificadores, Freund e Schapire (1996) e Breiman (1998) usam 100 classificadores.

Wolpert (1992) refere que uma implementação bem-sucedida da generalização em pilha para tarefas de classificação é uma espécie de *magia negra*, e as condições sobre as quais o *stacking* trabalha são ainda desconhecidas. Posteriormente, Ting e Witten (1997) mostraram que o sucesso da generalização em pilha requer o uso da distribuição das classes de saída. Nas experiências reportadas, somente o algoritmo MLR (com um discriminante linear) foi apropriado para o generalizador de $N\acute{i}vel_1$.

Capítulo 9

Avaliação de Modelos Preditivos

Na utilização de algoritmos de ECD em problemas reais, o conhecimento que se tem do domínio da aplicação é proveniente unicamente do conjunto de exemplos, a partir do qual a indução de um modelo preditivo é realizada. Nos capítulos anteriores foram apresentadas várias técnicas de ECD que podem ser utilizadas na indução de modelos de classificação e/ou de regressão a partir de um conjunto de exemplos classificados. De maneira geral, pode-se afirmar que não existe uma técnica universal, ou seja, não é possível estabelecer *a priori* uma técnica de ECD que obtenha o melhor desempenho na resolução de qualquer tipo de problema.

Em certos casos, as próprias caraterísticas das técnicas existentes e do problema que está a ser resolvido podem ser consideradas para auxiliar na escolha da técnica a ser utilizada sobre um novo conjunto de dados. Por exemplo, em domínios em que os exemplos possuem alta dimensionalidade, as SVMs são boas candidatas, enquanto o algoritmo k-NN usando a distância euclidiana pode, a princípio, não parecer uma escolha adequada. Caso seja necessário que o modelo obtido seja interpretável, técnicas simbólicas, como as árvores de decisão, podem ser preferíveis a modelos "caixa-preta" como os gerados pelas RNAs e pelas SVMs. Mesmo com o uso destas *heurísticas*, diversos algoritmos podem ser considerados candidatos à solução de um dado problema. Ainda que um único algoritmo seja escolhido, pode ser necessário realizar ajustes nos seus parâmetros livres, o que conduz à obtenção de múltiplos modelos para os mesmos dados.

Os parágrafos anteriores evidenciam uma caraterística particular do domínio de ECD: a necessidade de avaliação experimental. De fato, a validação de qualquer nova técnica de ECD proposta envolve a realização de experiências controladas, em que se demonstre a sua eficiência na solução de diferentes problemas, representados pelos conjuntos de dados associados. Desta forma, é recomendável seguir os procedimentos corretos, de forma a garantir a validade e a replicabilidade das experiências realizadas e, mais importante, das conclusões obtidas a partir de seus resultados.

A avaliação experimental de um algoritmo de ECD pode ser realizada segundo diferentes aspetos, tais como taxa de acerto do modelo gerado, compreensibilidade do conhecimento extraído, tempo de aprendizagem, requisitos de armazenamento do modelo, entre outros. Considerando modelos preditivos, iremos concentrar a nossa discussão em

torno de medidas relacionadas com o desempenho obtido nas predições realizadas. Muitos dos conceitos e procedimentos discutidos são facilmente generalizáveis a outros tipos de medidas de desempenho.

Inicialmente, na Seção 9.1, é realizada uma discussão acerca das principais medidas de erro utilizadas na avaliação de preditores em ECD. Na Seção 9.2 são apresentadas técnicas para a amostragem de dados. Na Seção 9.3 são apresentadas outras medidas de desempenho no contexto de classificação binária, bem como as curvas ROC. Na Seção 9.4 são apresentadas discussões a respeito da comparação estatística do desempenho de diferentes modelos. Este capítulo é concluído com a apresentação da decomposição viés-variância do erro de um preditor (Seção 9.5), seguida por considerações finais (Seção 9.6).

9.1 Métricas de Erro

A avaliação de um algoritmo de ECD supervisionado é normalmente realizada por meio da análise do desempenho do preditor na classificação de novos exemplos; ou seja, exemplos não utilizados no conjunto de treino (Monard e Baranauskas, 2003).

9.1.1 Métricas para Classificação

Uma medida de desempenho usualmente empregue na avaliação de um classificador $\hat{f}$ é a sua taxa de erro ou taxa de classificações incorretas, ilustrada na Equação 9.1, em que $I(a) = 1$ se a é verdadeiro e 0 caso contrário. Dado um conjunto de dados contendo n objetos, sobre o qual a avaliação será realizada, essa taxa equivale à proporção de exemplos desse conjunto classificados incorretamente por $\hat{f}$, sendo obtida pela comparação da classe conhecida de $\mathbf{x}_i$, y_i, com a classe predita, $\hat{f}(\mathbf{x}_i)$. Este tipo de medida equivale ao uso da função de custo 0-1, que relaciona os rótulos dos objetos às predições obtidas (Equação 7.10).

$$err\left(\hat{f}\right) = \frac{1}{n}\sum_{i=1}^{n} I\left(y_i \neq \hat{f}\left(\mathbf{x}_i\right)\right) \tag{9.1}$$

A taxa de erro varia entre 0 e 1, e valores próximos ao extremo 0 são melhores. O complemento dessa taxa corresponde à taxa de acerto do classificador:

$$ac(\hat{f}) = 1 - err(\hat{f}) \tag{9.2}$$

Neste caso, valores próximos de 1 são considerados melhores. Outra alternativa para visualizar o desempenho de um classificador baseia-se no uso de uma matriz de confusão. Essa matriz ilustra o número de predições corretas e incorretas em cada classe. Para um determinado conjunto de dados, as linhas dessa matriz representam as classes verdadeiras, e as colunas, as classes preditas pelo classificador. Logo, cada elemento m_{ij} de uma matriz de confusão $\mathbf{M}_c$ apresenta o número de exemplos da classe i classificados como pertencentes à classe j. Para k classes, $\mathbf{M}_c$ tem então dimensão $k \times k$. A diagonal apre-

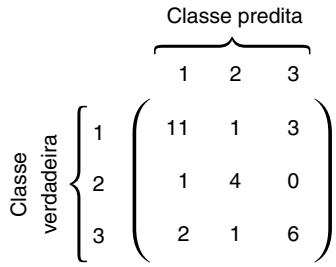


Figura 9.1 *Exemplo de uma matriz de confusão para um problema com três classes.*

senta os acertos do classificador, enquanto os outros elementos correspondem aos erros cometidos nas suas predições. Através da análise desta matriz é possível calcular medidas que indicam quais as classes onde o algoritmo tem maior dificuldade de discriminar. Na Figura 9.1 é apresentado um exemplo de matriz de confusão para um problema com três classes. Segundo essa matriz, onze dos quinze exemplos da classe 1 foram corretamente classificados, um foi incorretamente classificado como pertencente à classe 2 e três foram preditos como sendo da classe 3.

9.1.2 Métricas para Regressão

As considerações anteriores diziam respeito a problemas de classificação. No caso de problemas de regressão, o erro da hipótese $\hat{f}$ pode ser calculado através da distância entre o valor y_i conhecido e o valor predito pelo modelo, ou seja, $\hat{f}(\mathbf{x}_i)$ (Monard e Baranauskas, 2003). As medidas de erro mais conhecidas e usadas neste caso são o erro quadrático médio (MSE - *mean squared error*) e a distância absoluta média (MAD - *mean absolute distance*):

$$\mathrm{MSE}\left(\hat{f}\right) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{f}\left(\mathbf{x}_i\right)\right)^2 \tag{9.3}$$

$$\mathrm{MAD}\left(\hat{f}\right) = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \hat{f}\left(\mathbf{x}_i\right)\right| \tag{9.4}$$

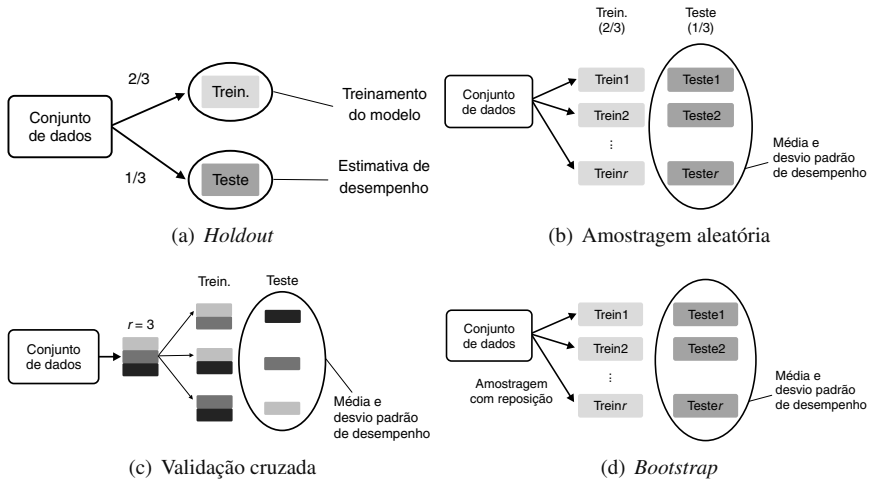
O MSE e MAD são sempre não negativos. Para ambas as medidas, valores mais baixos correspondem a melhores modelos, ou seja, maior capacidade preditiva.

9.2 Amostragem

Em diversos casos, tem-se apenas um conjunto com n objetos, o qual deve ser empregue quer na indução do preditor quer também na sua avaliação. Calcular o desempenho preditivo – em termos de taxa de acerto ou de erro, por exemplo – do modelo nos mesmos

objetos usados no treino produz estimativas otimistas, uma vez que todos os algoritmos de ECD tentam melhorar de alguma forma o seu desempenho preditivo nesses objetos durante a fase indutiva. O uso do mesmo conjunto de exemplos no treino e avaliação do preditor é conhecido como resubstituição (Toussaint, 1974). Em geral, o erro/acerto obtido nesse tipo de avaliação é denominado aparente.

Devem-se então utilizar métodos de amostragem alternativos para obter estimativas de desempenho preditivo mais confiáveis, definindo subconjuntos de treino e de teste. Os dados de treino são usados na indução e no ajuste do modelo, enquanto os exemplos de teste simulam a apresentação de objetos novos ao preditor, os quais não foram vistos em sua indução. Estes subconjuntos são disjuntos para assegurar que as medidas de desempenho sejam obtidas a partir de um conjunto de exemplos diferente daquele que foi usado na aprendizagem. Alguns dos principais métodos de amostragem existentes são ilustrados na Figura 9.2: *holdout* (9.2(a)), amostragem aleatória (9.2(b)), validação cruzada (9.2(c)) e *bootstrap* (9.2(d)).



(a) *Holdout* (b) Amostragem aleatória

(c) Validação cruzada (d) *Bootstrap*

Figura 9.2 *Métodos de amostragem.*

No caso dos procedimentos que envolvem médias de desempenho, deve-se reportar também os valores d0 desvio padrão associados. Um desvio padrão elevado indica uma alta variabilidade nos resultados, ou seja, uma instabilidade do modelo perante mudanças nos objetos que são todos provenientes de uma mesma distribuição. Isso pode ser um indicativo de sensibilidade aos objetos usados no treino. Na validação cruzada com 10 partições, por exemplo, tem-se pequenas mudanças nos objetos em cada partição, conforme será discutido na Seção 9.2.2. Dessa forma, um algoritmo que apresente desvio padrão elevado em validação cruzada apresenta sensibilidade a alterações nos objetos usados no treino.

Para reportar estimativas mais precisas do desempenho esperado do algoritmo nesses

casos, deve-se então apresentar algum intervalo de confiança para a média calculada, que irá envolver também o desvio padrão obtido. De maneira geral, isto é realizado pela comunidade reportando pelo menos os valores de média e desvio padrão das experiências realizadas. Contudo, pode-se também optar por calcular intervalos com maior nível de confiança, recorrendo para tal a métodos estatísticos. Detalhes a respeito do cálculo de intervalos de confiança podem ser consultados em diversos livros da área de Estatística, entre os quais Devore (2006).

De um modo geral, o desvio padrão pode até mesmo auxiliar na escolha entre dois algoritmos com desempenho médio semelhante: o de menor desvio padrão apresenta um desempenho mais estável e deve ser preferido. Contudo, em cenários como o descrito, o recomendável é conduzir uma análise mais rigorosa, envolvendo testes de hipóteses para comparação dos desempenhos de modelos. Este tópico é abordado na Seção 9.4.

9.2.1 *Holdout* e Amostragem Aleatória

No caso do *holdout*, divide-se o conjunto de dados em uma proporção de p para treino e $(1-p)$ para teste, como ilustrado na Figura 9.2(a). Normalmente, considera-se $p = \frac{2}{3}$. Este tipo de separação pode subestimar a taxa de acerto, uma vez que um preditor produzido sobre todos os objetos em geral apresentará uma taxa de acerto maior que a gerada a partir de uma parte deles (Baranauskas e Monard, 2000). Porém, segundo Michie et al. (1994), para conjuntos de dados grandes isso não representa um problema – embora a definição de quando um conjunto de dados é grande o suficiente não seja clara e dependa de vários fatores, entre eles a própria complexidade da estrutura presente nos dados, que não é conhecida de antemão.

Outra crítica usual ao *holdout* é que este não permite avaliar o quanto o desempenho de uma técnica varia quando diferentes combinações de objetos são apresentadas em seu treino. De fato, é possível que, numa divisão realizada, objetos considerados "mais fáceis" tenham sido agrupados no subconjunto de teste.

Para tornar os resultados menos dependentes da partição feita, é possível construir diversas partições aleatórias e obter uma média de desempenho em *holdout*, um método às vezes referenciado como *random subsampling* (amostragem aleatória). Na Figura 9.2(b), é apresentado um exemplo em que são gerados r diferentes subconjuntos aleatórios.

9.2.2 Validação Cruzada

No método de validação cruzada *r-fold cross-validation*, o conjunto de exemplos é dividido em r subconjuntos de tamanho aproximadamente igual. Os objetos de $r-1$ partições são utilizados no treino de um preditor, o qual é então testado na partição restante. Esse processo é repetido r vezes, utilizando em cada ciclo uma partição diferente para teste. O desempenho final do preditor é dado pela média dos desempenhos observados sobre cada subconjunto de teste. Esse processo é ilustrado na Figura 9.2(c), utilizando $r = 3$.

Uma variação desse método para problemas de classificação é o *r-fold cross-validation* estratificado, que mantém em cada partição a proporção de exemplos de cada classe se-

melhante à proporção contida no conjunto de dados total. Se, por exemplo, o conjunto de dados original tem 20% dos objetos na classe c_1 e 80% na classe c_2, cada partição também procura manter essa proporção, apresentando 20% de seus exemplos na classe c_1 e 80% na classe c_2.

No caso extremo em que $r = n$, em que n representa o número de casos disponíveis, tem-se o método *leave-one-out*. No *leave-one-out*, a cada ciclo exatamente um exemplo é separado para teste, enquanto os $n-1$ exemplos restantes são utilizados no treino do preditor. O desempenho é dado pela soma dos desempenhos verificados para cada exemplo de teste individual. Esse método produz uma estimativa mais fiel do desempenho preditivo do modelo. Porém, requer maior esforço computacional, e geralmente só é aplicado em amostras de dados pequenas.

A principal crítica aos procedimentos de validação cruzada é que uma parte dos dados é compartilhada entre os subconjuntos de treino. Para $r \geqslant 2$ partições, uma proporção de $\left(1-\frac{2}{r}\right)$ dos objetos é compartilhada (Monard e Baranauskas, 2003). Por exemplo, utilizando 10 partições (*folds*), 80% dos objetos contidos nos subconjuntos de treino são compartilhados. Apesar de as partições de teste serem distintas entre si, com $r > 2$ não se tem completa independência entre os subconjuntos de treino.

9.2.3 Amostragem com reposição - *Bootstrap*

No método *booststrap*, são gerados r subconjuntos de treino a partir do conjunto de exemplos original (Figura 9.2(d)). Os exemplos são amostrados aleatoriamente desse conjunto, com reposição. Logo, um exemplo pode estar presente num determinado subconjunto de treino mais de uma vez. Os exemplos não selecionados compõem os subconjuntos de teste. O resultado final é dado pela média do desempenho observado em cada subconjunto de teste.

Normalmente adota-se $r \geqslant 100$. A ideia básica consiste em repetir a experiência um número elevado de vezes e estimar o desempenho nesse conjunto de experiências. Por esse motivo, o *bootstrap* é um procedimento custoso sendo geralmente aplicado em amostras de dados pequenas.

Existem vários estimadores *bootstrap*, sendo o mais comum o e_0 (Jain et al., 1987). Neste, cada conjunto de treino tem n exemplos, amostrados com reposição do conjunto original, sendo n o número total de exemplos nesse conjunto. Cada exemplo tem probabilidade $1-(1-1/n)^n$ de ser selecionado pelo menos uma vez. Para valores de n elevados, essa probabilidade tende para $1-1/e = 0,632$, ou seja, a fração média de exemplos não repetidos nos conjuntos de treino é de 63,2%. Exemplos remanescentes formam o subconjunto de teste. O desempenho é dado pela média das iterações. A estimativa de desempenho obtida é estatisticamente equivalente à do *leave-one-out*, mas com menor variância.

9.3 Problemas de Duas Classes e o Espaço ROC

Por simplicidade, considere um problema com duas classes. Usualmente, uma classe é denotada positiva (+) e a outra é denominada como negativa ($-$). Temos então a matriz de confusão ilustrada na Tabela 9.1, em que:

- VP corresponde ao número de verdadeiros positivos, ou seja, o número de exemplos da classe positiva classificados corretamente.
- VN corresponde ao número de verdadeiros negativos, ou seja, o número de exemplos da classe negativa classificados corretamente.
- FP corresponde ao número de falsos positivos, ou seja, o número de exemplos cuja classe verdadeira é negativa mas que foram classificados incorretamente como pertencendo à classe positiva.
- FN corresponde ao número de falsos negativos, ou seja, o número de exemplos pertencentes originalmente à classe positiva que foram incorretamente preditos como da classe negativa.

Além disso, $n = \text{VP} + \text{VN} + \text{FP} + \text{FN}$.

Tabela 9.1 *Matriz de confusão para um problema com duas classes*

		Classe predita	
		+	-
Classe verdadeira	+	VP	FN
	-	FP	VN

9.3.1 Medidas de Desempenho

Com base Na matriz de confusão, é possível obter uma série de medidas de desempenho. Entre elas, temos (Monard e Baranauskas, 2003):

- *Taxa de erro na classe positiva*: proporção de exemplos da classe positiva incorretamente classificados pelo preditor $\hat{f}$, também conhecida como taxa de falsos negativos (TFN).

$$err_{+}\left(\hat{f}\right) = \frac{\text{FN}}{\text{VP} + \text{FN}} \tag{9.5}$$

- *Taxa de erro na classe negativa*: proporção de exemplos da classe negativa incorretamente classificados por $\hat{f}$, também conhecida como taxa de falsos positivos (TFP).

$$err_{-}\left(\hat{f}\right) = \frac{\text{FP}}{\text{FP} + \text{VN}} \tag{9.6}$$

- *Taxa de erro total*:

$$err\left(\hat{f}\right) = \frac{\mathrm{FP} + \mathrm{FN}}{n} \tag{9.7}$$

- *Taxa de acerto*: calculada pela soma dos valores da diagonal principal da matriz, dividida pela soma dos valores de todos os elementos da matriz.

$$ac\left(\hat{f}\right) = \frac{\mathrm{VP} + \mathrm{VN}}{n} \tag{9.8}$$

- *Precisão*: proporção de exemplos positivos classificados corretamente entre todos aqueles preditos como positivos por $\hat{f}$.

$$prec(\hat{f}) = \frac{\mathrm{VP}}{\mathrm{VP} + \mathrm{FP}} \tag{9.9}$$

- *Sensibilidade*: corresponde à taxa de acerto na classe positiva. Também é denominada de taxa de verdadeiros positivos (TVP).

$$sens\left(\hat{f}\right) = rev\left(\hat{f}\right) = \mathrm{TVP}(\hat{f}) = \frac{\mathrm{VP}}{\mathrm{VP} + \mathrm{FN}} \tag{9.10}$$

- *Especificidade*: corresponde à taxa de acerto na classe negativa. Seu complemento corresponde à taxa TFP.

$$esp\left(\hat{f}\right) = \frac{\mathrm{VN}}{\mathrm{VN} + \mathrm{FP}} = 1 - \mathrm{TFP}(\hat{f}) \tag{9.11}$$

As medidas anteriores podem ser facilmente generalizadas a problemas com mais de duas classes pela consideração de cada classe como positiva em relação ao conjunto das demais classes (exceto a taxa de erro/acerto total, em que todas as classes são consideradas globalmente). Obtém-se um valor de desempenho para cada classe. Por exemplo, se a taxa que está sendo observada é a precisão, é obtido um valor de precisão para cada classe, no qual a classe em consideração é vista como positiva, enquanto as demais são consideradas negativas.

A precisão pode ser vista como uma medida de exatidão do modelo, e a sensibilidade, como uma medida de sua completude. Uma precisão de 1,0 para uma classe C significa que cada item rotulado como pertencente à classe C realmente pertence a essa classe, mas não fornece informação a respeito do número de exemplos da classe C que não foram classificados corretamente. Por outro lado, uma revocação de 1,0 significa que cada exemplo da classe C foi rotulado como pertencendo à classe C, mas não diz nada a respeito de quantos outros exemplos foram classificados incorretamente como pertencendo à classe C. Desse modo, geralmente a precisão e a revocação não são discutidas isoladamente, mas são combinadas em uma única medida, como a medida-F, que é a média harmónica

ponderada da precisão e da sensibilidade:

$$F_m(\hat{f}) = \frac{(w+1) \times rev(\hat{f}) \times prec(\hat{f})}{rev(\hat{f}) + w \times prec(\hat{f})} \tag{9.12}$$

Usando um peso igual a 1, que equivale a atribuir o mesmo grau de importância à sensibilidade e à precisão, obtemos a medida F_1:

$$F_1(\hat{f}) = \frac{2 \times prec(\hat{f}) \times rev(\hat{f})}{prec(\hat{f}) + rev(\hat{f})} \tag{9.13}$$

9.3.2 Análise ROC

Uma forma alternativa de avaliar classificadores em problemas binários, ou seja, problemas de duas classes, é baseada na análise das curvas ROC (*Receiving Operating Characteristics*) (Fawcett, 2005). O seu uso inicial em ECD foi reportado em (Spackman, 1989) para avaliação e comparação de algoritmos. Desde então a sua utilização tem sido estendida até mesmo na proposta de novos algoritmos e técnicas de ECD com base na análise ROC (Prati e Flach, 2005).

O gráfico ROC é um gráfico bidimensional num espaço denominado espaço ROC, com eixos X e Y representando as medidas de taxa de falsos positivos (TFP) e taxa de verdadeiros positivos (TVP), respetivamente. O desempenho de um dado classificador pode ser representado por um ponto nesse espaço bidimensional.

Os principais aspetos de um espaço ROC são ilustrados na Figura 9.3. A linha diagonal representa classificadores que realizam predições aleatórias. Qualquer classificador abaixo dessa linha pode então ser considerado pior que o aleatório. A figura inclui, como exemplo, TFP e TVP de modelos gerados por três algoritmos. O ponto (0,1) representa classificações perfeitas, em que todos os exemplos positivos e negativos são classificados corretamente, sendo por isso denominado *céu ROC*. O ponto (1,0), por outro lado, representa o *inferno ROC*. O ponto (1,1) representa classificações sempre positivas, e o ponto (0,0), classificações sempre negativas. Assim, classificadores na região próxima do ponto (1,1) quase sempre rotulam os exemplos como positivos, e classificadores na região próxima do ponto (0,0) rotulam a maioria dos exemplos como negativa. Um classificador é considerado melhor que um outro se seu ponto no espaço ROC se posiciona acima e à esquerda do ponto correspondente ao segundo classificador (Prati, 2006).

Embora existam mecanismos para comparar diferentes classificadores com base nos seus pontos no espaço ROC, o procedimento mais usual consiste em gerar uma curva ROC. Nesse caso, é necessário dispor as previsões de um classificador por uma ordem determinada. Muitos classificadores produzem como saída, um valor contínuo associado à sua classificação (ou podem ser adaptados para tal), que pode então ser usado neste ordenamento. Como exemplos temos: o NB, que produz a probabilidade do exemplo pertencer a uma dada classe; as RNAs MLP, em que o neurónio da camada de saída

Normalmente, para realizar a classificação final, os valores contínuos obtidos são dis-

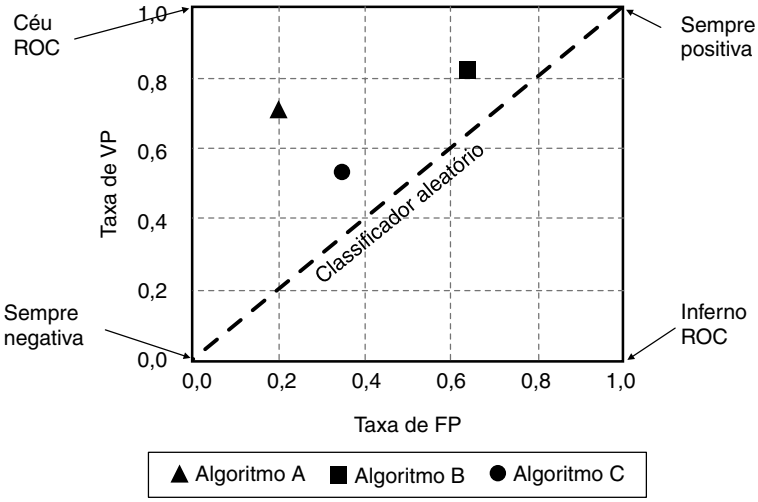


Figura 9.3 *Espaço ROC com três classificadores.*

cretizados de alguma maneira. Geralmente, utiliza-se algum limiar que permite dizer se a predição corresponde à classe positiva ou negativa. No caso do NB, por exemplo, o limiar adotado por defeito é normalmente um valor de probabilidade de 0,5. Um exemplo $\vec{x}$ é da classe +, se a $P(*|\vec{x}) > 0.5$. Usando limiares diferentes, são obtidos novos valores de TFP e TVP. Representando graficamente os valores de TFP e TVP para diferentes limiares e unindo esses pontos, tem-se como resultado uma curva ROC para cada técnica de classificação. Desta maneira a análise é independente do limiar escolhido. Inclusive, é possível escolher valores de limiar mais adequados para cada problema em particular (Matsubara, 2008).

Na Figura 9.4 são ilustradas duas curvas ROC, que correspondem a curvas geradas por dois algoritmos de classificação distintos utilizando cinco limiares diferentes. Neste caso, não há interseção entre as curvas, porém isso pode ocorrer. Quando se comparam duas ou mais curvas, se estas não se intercetarem, aquela que mais se aproxima do ponto (0,1) corresponde ao melhor desempenho. No caso de ocorrerem interseções, cada algoritmo tem uma região com melhor desempenho. É comum, porém, comparar o desempenho dos algoritmos em termos de uma medida única extraída de sua curva ROC: a área abaixo da curva ROC (AUC, do Inglês *Area Under ROC Curve*).

A medida AUC produz valores entre 0 e 1. Valores mais próximos de 1 são considerados melhores. Portanto, ao comparar dois ou mais algoritmos segundo essa medida, aquele que possuir AUC mais próximo de 1 é considerado superior. Contudo, é recomendável comparar as medidas obtidas estatisticamente. Além disso, aconselha-se calcular o AUC usando validação cruzada, tal como é sugerido para as outras medidas de desempe-

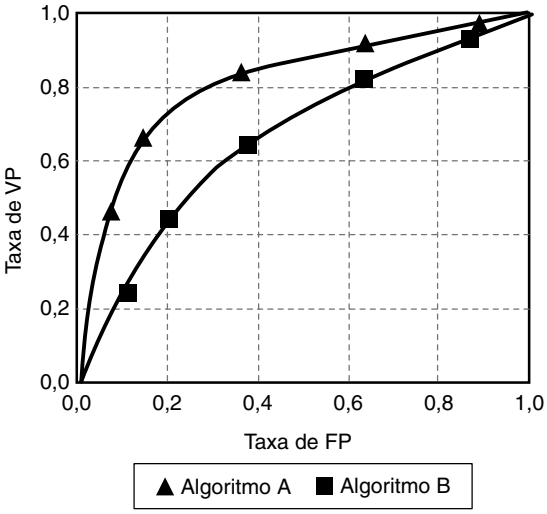


Figura 9.4 *Exemplos de curvas ROC.*

nho, obtendo a média e desvio padrão dos valores obtidos para diferentes partições dos dados.

Entre as principais vantagens da análise ROC estão a possibilidade de realizar medidas de desempenho independentes de condições como o limiar de classificação e também de custos associados às classificações incorretas e à distribuição das classes (Prati, 2006). De fato, o uso de diferentes limiares de classificação representa uma maior ou menor ênfase à classe positiva, permitindo lidar com questões de desbalanceamento das classes e de diferentes custos de classificação. Por outro lado, a taxa de acerto/erro é bastante influenciada por desbalanceamentos. Seja, por exemplo, um conjunto de dados com 90 exemplos positivos e 10 negativos. A obtenção de uma estimativa de taxa de acerto de 0,90 com esses objetos não é necessariamente indicativa de bom desempenho preditivo, uma vez que o modelo pode estar simplesmente classificando todos os exemplos na classe positiva.

Uma desvantagem associada à análise por curvas ROC está relacionada com o fato de ser originalmente limitada a problemas de classificação binários. Existem trabalhos relacionados à generalização dessas curvas a problemas multiclasse, tais como a geração de múltiplas curvas, uma para cada classe, quando essa classe é considerada positiva e as demais são tomadas como negativas (Provost e Domingos, 2001). Contudo, o uso mais difundido da análise ROC tem sido em problemas de classificação binários.

9.4 Testes de Hipóteses

Diversos estudos na área de ECD requerem a comparação de dois ou mais algoritmos na solução de um ou mais problemas práticos. Neste processo, é comum dividir o(s) conjunto(s) de dados, tendo por base uma estratégia de amostragem, e usar para todos os algoritmos comparados exatamente as mesmas partições dos dados. Sem perda de generalidade, iremos assumir nesta seção que o método usado é o *r-fold cross-validation* estratificado. Ou seja, a cada iteração da validação cruzada, todos os algoritmos usam a mesma partição de treino e de teste para obter os respetivos resultados, e, dessa forma, a média de desempenho obtida por todos é calculada sobre os mesmos objetos. Isto significa impor que os algoritmos sejam comparados em igualdade de condições.

Após o processo descrito anteriormente, considerando um conjunto de dados em particular, para cada algoritmo tem-se a média de alguma medida de seu desempenho, como o erro ou o AUC médio e o seu desvio padrão nas partições. Determinar se um algoritmo é melhor do que o outro pelo simples exame de superioridade/inferioridade de médias não é aconselhável. Muitas vezes as diferenças verificadas não são significativas, e pode-se considerar os desempenhos obtidos equivalentes. É necessário conduzir um teste de hipóteses para a comparação dos desempenhos dos modelos que estão a ser analisados.

Uma hipótese estatística é uma alegação sobre o valor de um ou mais parâmetros ou sobre a forma de uma distribuição de probabilidade (Devore, 2006). Se μ_1 e μ_2 são os erros médios de dois modelos em validação cruzada, por exemplo, uma hipótese possível é dada pela expressão $\mu_1 - \mu_2 = 0$. Ou seja, essas médias podem ser consideradas equivalentes. Outra é que $\mu_1 - \mu_2 > 0$, ou seja, que a média 1 é superior à média 2.

Nos testes de hipóteses, normalmente existem duas suposições contraditórias em consideração, tais como $H_0 : \mu_1 - \mu_2 = 0$ *versus* $H_1 : \mu_1 - \mu_2 \neq 0$. Deve-se então decidir qual das duas hipóteses é a correta. A hipótese nula, representada por H_0, é inicialmente assumida como verdadeira. H_1 é denominada hipótese alternativa. A hipótese nula é rejeitada em favor da hipótese alternativa se alguma evidência na amostra representada pelos desempenhos nas experiências sugerir que H_0 é falsa. O teste de hipóteses é então realizado com o objetivo de rejeitar ou não (nesse caso, aceitar) a hipótese H_0. A regra para decidir se H_0 é rejeitada é denominada procedimento de teste.

Num procedimento de teste, tem-se (Devore, 2006):

- Uma estatística de teste, obtida em função dos dados da amostra em que a decisão se baseia;
- Uma região de rejeição, que representa o conjunto de valores da estatística de teste para os quais H_0 é rejeitada. Ou seja, a hipótese nula é rejeitada se e somente se o valor da estatística calculada cair na região de rejeição.

Os procedimentos de teste podem cometer erros, devido à própria variabilidade das amostras sobre as quais são calculados. Um erro do tipo I ocorre quando a hipótese nula é rejeitada apesar de ser verdadeira. Já um erro do tipo II configura-se quando H_0 é falsa e não é rejeitada. Deve-se procurar adotar procedimentos que apresentem um compromisso

em relação à ocorrência de ambos os tipos de erro. Normalmente, cometer erros do tipo I é considerado mais grave. A maioria dos testes realizados envolve o controlo da probabilidade de ocorrência de erros do tipo I, que é denotada usualmente pelo termo α. A região de rejeição é calculada de maneira a manter a probabilidade de ocorrência de erro do tipo I sob controle. O valor α também é denominado nível de significância do teste. É comum adotar-se um valor de 0,05 para α. Isto equivale a dizer que o resultado do teste possui um nível de confiança de 95% de não ter rejeitado a hipótese nula quando esta é verdadeira.

Nas seções seguintes serão discutidos os procedimentos de teste frequentemente utilizados pela comunidade de ECD. É importante destacar que, embora a realização desses testes seja recomendável para a comparação de dois ou mais modelos, não há ainda um consenso quanto à adequabilidade de vários procedimentos de teste existentes no cenário típico das experiências de ECD, em que as amostras utilizadas para a avaliação de desempenho apresentam dependências. Descreveremos dois dos testes mais utilizados atualmente, porém recomendamos ao leitor a literatura relacionada com o tema, que também discute outros procedimentos de teste, tais como Dietterich (1998), Salzberg (1997), Nadeau e Bengio (2003), Demsár (2006), e García e Herrera (2008).

Os testes que serão apresentados, *Wilcoxon signed-rank* e Friedman, são emparelhados e não paramétricos, não impondo a restrição das amostras sobre os quais são aplicados terem que seguir alguma distribuição (como a Normal), o que não é garantido de ocorrer na prática quando modelos de ECD são comparados. Ambos os testes são baseados em ordenações, sendo também interessantes por permitirem comparar outras medidas de desempenho. Pode-se, por exemplo, recorrer à sua utilização para efeitos de comparar os tempos de treino de diferentes algoritmos.

Podemos distinguir dois casos na aplicação dos testes. O primeiro caso ocorre quando se deseja comparar o desempenho dos modelos num único conjunto de dados. Esse cenário ocorre, por exemplo, quando se quer determinar que modelo possui destaque num domínio em particular, representado por um único conjunto de dados. Dietterich (1998) e Salzberg (1997) discutem diferentes testes para esse caso. No segundo caso, os modelos são comparados considerando vários conjuntos de dados. A avaliação em múltiplos conjuntos de dados é recomendada, uma vez que os resultados obtidos num único conjunto de dados podem ser muito específicos para essa amostra em particular. Na proposta de um novo algoritmo de ECD, por exemplo, usualmente deseja-se que este seja aplicável a uma gama de problemas relacionados. O mais indicado é então testar o seu desempenho em vários conjuntos de dados. O trabalho de Demsár (2006) sugere, nesse tipo de trabalho, comparar as médias de desempenho (em validação cruzada, por exemplo) dos modelos nos vários conjuntos de dados. Estes testes serão discutidos a seguir.

9.4.1 Comparando Dois Modelos

Consideremos inicialmente a comparação de dois modelos. Normalmente, um modelo gerado por um algoritmo padrão e outro modelo obtido pelo uso de um novo algoritmo. Pretende-se determinar se o novo algoritmo se destaca em relação ao algoritmo padrão. A hipótese nula H_0 afirma que os desempenhos dos modelos são equivalentes.

Um teste recomendado para a comparação de dois modelos é o *Wilcoxon signed-ranks* (Wilcoxon, 1943). Neste teste, inicialmente calculam-se as diferenças nas medidas de desempenho dos algoritmos. Em seguida, os valores absolutos dessas diferenças são ordenados de forma crescente (menores diferenças aparecem primeiro). Pelo teste, comparam-se as posições das diferenças positivas e negativas entre os algoritmos. Dados dois algoritmos A e B, caso as diferenças sejam calculadas considerando o desempenho de B menos o de A e usando uma medida de desempenho em que valores mais elevados são considerados melhores (como taxa de acerto, AUC, precisão, entre outras), diferenças positivas indicam um melhor desempenho de B, enquanto as negativas refletem um melhor desempenho de A. No caso de medidas de desempenho como o erro, em que valores menores são melhores, basta aplicar o raciocínio inverso.

Seja d_i a diferença entre os desempenhos de A e B num conjunto de dados i. Calculadas todas as diferenças, estas são ordenadas de acordo com os seus valores absolutos. No caso de empates, atribuem-se valores médios das posições na ordenação. Como exemplo, consideremos a Tabela 9.2, em que se comparam duas versões do algoritmo C4.5, a qual foi adaptada de uma tabela apresentada como exemplo em Demsár (2006).

Tabela 9.2 *Exemplo de tabela de diferenças de resultados em teste de Wilcoxon*

Conj. dados	C4.5	C4.5+m	Diferença	Dif_absoluta	Posição
Pulmão	0,583	0,583	0,000	0,000	1,5
Fungo	0,583	0,583	0,000	0,000	1,5
Atmosfera	0,882	0,888	+0,006	0,006	3,0
Mama	0,599	0,591	−0,008	0,008	4,0

Seja $R+$ a soma das posições da seriação (*ranks*) de conjuntos de dados em que o algoritmo B é melhor que o algoritmo A e $R-$ a soma de posições oposta. As posições das diferenças nulas são repartidas igualmente entre as duas somas. Se há um número ímpar de diferenças nulas, uma é ignorada. Temos então:

$$R+ = \sum_{d_i>0} rank\,(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i) \tag{9.14}$$

$$R- = \sum_{d_i<0} rank\,(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i) \tag{9.15}$$

Seja S a menor dessas somas. Alguns livros de Estatística apresentam tabelas com os valores críticos exatos para S, com N variando até 25. Para mais conjuntos de dados, a estatística do teste é:

$$z = \frac{S - \frac{1}{4}N\,(N-1)}{\sqrt{\frac{1}{24}N\,(N+1)\,(2N+1)}} \tag{9.16}$$

Com $\alpha = 0,05$, a hipótese nula de que os algoritmos se comportam de maneira similar pode ser rejeitada se z é menor que $-1,96$.

9.4.2 Comparando Mais Modelos

Quando vários modelos são comparados, o número de comparações é maior e o teste deve ser alterado para levar isso em consideração. Se múltiplos testes são realizados, a probabilidade de pelo menos um deles detetar uma diferença estatística quando esta não existe aumenta. Para J testes, a probabilidade de cometer pelo menos um erro é de $1-(1-\alpha)^J$ (Feelders e Verkooijen, 1996). Para $J = 20$ e $\alpha = 0,05$, por exemplo, a probabilidade de detetar-se uma ou mais diferenças estatísticas quando estas não existem é de 64%. Este problema é conhecido como efeito da multiplicidade (Salzberg, 1997).

Nesta situação, Demsár (2006) recomenda a utilização do teste de Friedman (Friedman, 1937). Este teste também é baseado na comparação de seriações (*rankings*) de desempenho. Contudo, neste caso, para realizar a seriação são considerados o valor absoluto da medida de desempenho de cada algoritmo individualmente e em cada conjunto de dados. Logo, para cada conjunto de dados, realiza-se a ordenação dos algoritmos de acordo com o seu desempenho (dos melhores para os piores). Em caso de empates, são atribuídos valores médios das posições na seriação.

Seja r_j^i a posição do desempenho do algoritmo j (dentre A algoritmos) no conjunto de dados i (dentre N conjuntos de dados). O teste de Friedman irá comparar as seriações médias R_j dos diferentes algoritmos. A hipótese nula H_0 afirma que todos os algoritmos são equivalentes e que as suas posições na seriação são idênticas. A estatística calculada é:

$$F_F = \frac{(N-1)\chi_F^2}{N(A-1)-\chi_F^2} \tag{9.17}$$

em que:

$$\chi_F^2 = \frac{12N}{A(A+1)}\left[\sum_j R_j^2 - \frac{A(A+1)^2}{4}\right] \tag{9.18}$$

A hipótese nula é rejeitada caso a estatística calculada seja maior que $F_{A-1,(A-1)(N-1)}$, em que $F_{A-1,(A-1)(N-1)}$ denota a distribuição de probabilidade F com $A-1$ e $(A-1)(N-1)$ graus de liberdade, que pode ser consultada em livros de Estatística. Caso a hipótese nula seja rejeitada, existe diferença de desempenhos. Porém, o teste não indica diretamente quais os algoritmos que são diferentes. Para conseguir essa informação, deve-se prosseguir com um pós-teste.

No pós-teste, o desempenho de dois algoritmos em particular é estatisticamente diferente caso a diferença entre os seus valores médios de posição na seriação seja maior ou igual ao valor de diferença crítica CD (do inglês *Critical Difference*):

$$CD = q_\alpha \sqrt{\frac{A(A+1)}{6N}} \tag{9.19}$$

Se todos os algoritmos são comparados entre si, os valores de q_α podem ser fornecidos pela estatística de Nemenyi (Nemenyi, 1963). O trabalho de García e Herrera (2008) menciona pós-testes alternativos para comparações entre todos os pares de algoritmos.

Quando a comparação é de vários algoritmos em relação a um único algoritmo (por exemplo, o desempenho de várias modificações de um algoritmo é comparado ao do algoritmo base), q_α pode ser menos restritivo, pois menos comparações são realizadas. Neste caso, pode ser utilizada a estatística de Bonferroni-Dunn (Dunn, 1961). Na Tabela 9.3, adaptada a partir de Demsár (2006), são apresentados os valores de $q_{0,05}$ para diferente número de algoritmos *A* sendo comparados, segundo os pós-testes de Nemenyi e Bonferroni-Dunn.

Tabela 9.3 *Valores de $q_{0,05}$ para diferentes pós-testes*

A	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Nemenyi	1,960	2,343	2,569	2,728	2,850	2,949	3,031	3,102	3,164
Bonferroni-Dunn	1,960	2,241	2,394	2,498	2,576	2,648	2,690	2,724	2,773

Em Demsár (2006) é apresentada ainda uma interessante maneira gráfica de dispor os resultados dos pós-testes. A Figura 9.5(a) apresenta o caso em que quatro algoritmos diferentes, denotados por A, B, C e D, são comparados entre si usando o pós-teste de Nemenyi. Inicialmente, uma escala contendo as seriações de 1 a 4 (quatro algoritmos sendo comparados) é desenhada de maneira a que a posição 1 é colocada mais à direita. Em seguida, os modelos são posicionados nessa escala de acordo com os valores médios de suas posições na seriação calculada no teste de Friedman. Por exemplo, o modelo B apresentou-se em média na segunda posição com relação ao seu desempenho nos conjuntos de dados. Linhas horizontais conectam modelos cujos resultados não tiveram diferença estatística de acordo com o pós-teste. Além disso, o valor crítico usado para verificar se dois algoritmos são diferentes é apresentado acima da linha de *rank* desenhada, a partir da esquerda. Pelo exame da Figura 9.5(a), é possível identificar que D possui resultado significativamente inferior aos de A e B. Nada pode ser dito a respeito de C, somente que nenhuma conclusão pode ser tirada com base nas experiências realizadas, pois um indivíduo não pode pertencer a duas populações diferentes em termos estatísticos.

Na Figura 9.5(b) é apresentado o caso em que os algoritmos A, B e C são todos comparados com o algoritmo base D, segundo o pós-teste de Bonferroni-Dunn. Nesta situação, o CD pode ser desenhado à esquerda e à direita da posição média do algoritmo de controlo D. Qualquer algoritmo que possua posição fora dessa área tem desempenho significativamente diferente do controle.

9.5 Decomposição Viés-Variância da Taxa de Erro

Uma análise informativa acerca do comportamento dos algoritmos de aprendizagem é fornecida pela designada decomposição *Viés-Variância* do erro. A ideia básica por trás desse

(a) Nemenyi (b) Bonferroni-Dunn

Figura 9.5 *Representação gráfica de pós-testes.*

enquadramento teórico é que um algoritmo de aprendizagem comete dois tipos de erros: *erros sistemáticos*, devido à linguagem de representação usada pelo algoritmo de aprendizagem, e erros devidos à dependência do modelo gerado ao conjunto de treino. Nesta seção, apresentamos as ideias base desta análise.

As origens da análise viés-variância estão relacionadas com a *regressão quadrática* (Geman et al., 1992). Esta é uma ferramenta poderosa da teoria da amostragem da estatística para analisar cenários de aprendizagem supervisionada que têm uma *função de custo quadrática*. Sendo dados uma variável objetivo e o tamanho do conjunto de treino, a formulação convencional da decomposição separa o erro esperado na soma de três quantidades não negativas:

- *Ruído intrínseco*
  Esta quantidade é o limite inferior do erro esperado de qualquer algoritmo de aprendizagem. É o erro esperado do classificador de Bayes ótimo.
- *Quadrado do enviesamento*
  Esta quantidade mede quão bem a predição média do algoritmo de aprendizagem (sobre todos os possíveis conjuntos de treino cujo tamanho é igual ao tamanho do conjunto de treino dado) corresponde à variável objetivo.
- *Variância*
  Esta quantidade mede o quanto a predição do algoritmo de aprendizagem se *aproxima* para diferentes conjuntos de objetos de um dado tamanho.

A decomposição viés-variância fornece diversas informações úteis. A mais relevante é conhecida como *equilíbrio viés-variância*. Quando algum aspeto de um algoritmo de aprendizagem particular é modificado, pode haver efeito oposto no viés e na variância. Geralmente, quando se aumenta o número de graus de liberdade de um algoritmo, o viés diminui, mas a variância aumenta. O número ótimo de graus de liberdade otimiza esse equilíbrio entre viés e variância.

Para problemas de classificação, a função de custo quadrática é inapropriada porque os rótulos da classe não são numéricos. Diversas outras propostas para decompor o erro de classificação em viés e variância foram sugeridos, incluindo Breiman (1996b, 1998), Kong e Dietterich (1995), e Kohavi e Wolpert (1996). Não há consenso em relação à decomposição. Aqui seguiremos a decomposição sugerida em Kohavi e Wolpert (1996).

9.5.1 Definição de Viés e Variância

Na aprendizagem preditiva, é assumida a existência de uma função desconhecida f que mapeia o espaço de entrada X no espaço de saída Y. f representa uma distribuição de probabilidade condicional $P(y_f|\mathbf{x})$. Uma *hipótese h* gerada por um algoritmo de aprendizagem representa uma distribuição similar $P(y_h|\mathbf{x})$. Vamos assumir que $P(y_f|\mathbf{x}) = 1$ para um valor de y_f e 0 para todos os outros. De igual forma, $P(y_h|\mathbf{x}) = 1$ para um valor de y_h e 0 para todos os demais.

Na *função de custo 0-1*, $\ell(y_f, y_h) = 0$ se e somente se $y_f = y_h$. O custo esperado para um único exemplo é:

$$E(c) = 1 - \sum_{y \in Y} P(Y_h = Y_f = y) \tag{9.20}$$

O custo esperado para um conjunto de exemplos $\mathbf{X}$ de tamanho n é estimado como a média do custo calculada sobre os n exemplos. Kohavi e Wolpert (1996) mostram que:

$$E(c) = \sum_{\mathbf{x} \in \mathbf{X}} P(\mathbf{x})(\sigma^2_{\mathbf{x}} + \text{viés}^2_{\mathbf{x}} + \text{variância}_{\mathbf{x}}) \tag{9.21}$$

em que

$$\text{viés}^2_{\mathbf{x}} = \frac{1}{2} \sum_{y \in Y} (P(y_f = y) - P(y_h = y))^2 \tag{9.22}$$

$$\text{variância}_{\mathbf{x}} = \frac{1}{2}(1 - \sum_{y \in Y} P(y_h = y)^2) \tag{9.23}$$

$$\sigma^2_{\mathbf{x}} = \frac{1}{2}(1 - \sum_{y \in Y} P(y_f = y)^2) \tag{9.24}$$

Estas definições de viés^2, variância e ruído (σ) têm as seguintes propriedades:

1. O componente viés^2 mede o quadrado da diferença entre a saída média do objetivo e a saída média do algoritmo. É uma quantidade real não negativa e igual a zero somente se $P(y_f|\mathbf{x}) = P(y_h|\mathbf{x})$ para todo $\mathbf{x}$ e y. Essa propriedade é compartilhada pelo viés^2 para o erro quadrático.

2. O termo *variância* mede a *variabilidade* de $P(y_h|\mathbf{x})$. É uma quantidade real não negativa e igual a zero para um algoritmo que faz sempre a mesma previsão independentemente do conjunto de treino (por exemplo, o classificador Bayesiano ótimo). Assim que o algoritmo se torna mais sensível a mudanças no conjunto de treino, a variância aumenta. Dada uma distribuição sobre o conjunto de treino, a variância mede a sensibilidade do algoritmo de aprendizagem a mudanças no conjunto de treino e é independente de y_f.

3. O ruído é independente do algoritmo de aprendizagem.

9.5.2 Medindo os Componentes Viés-Variância

Para um dado algoritmo e um dado conjunto de dados, estimamos o viés[2] e a *variância* como se segue.

1. Dividimos aleatoriamente o conjunto de dados em duas partes, T e E. T é usado como se ele fosse o "universo" a partir da qual amostramos o conjunto de treino, enquanto E é usado para avaliar os termos na decomposição.
2. Geramos N conjuntos de treino a partir de T, cada geração usando amostragem aleatória uniforme sem substituição.
3. Executamos o algoritmo de aprendizagem para cada conjunto de treino e estimamos os termos *variância* da Equação 9.23 e *viés* da Equação 9.22 usando o classificador gerado para cada exemplo $\mathbf{x}$ no conjunto de avaliação E. Todos os termos são estimados usando contadores de frequência.
4. A componente ruído σ^2, que é o erro do classificador de Bayes ótimo, é muito difícil de estimar na prática. Usando um estimador de contagem de frequência, a estimativa de σ^2 poderá ser zero se todas as instâncias forem únicas (independentemente do valor real de σ^2). É usualmente agregado ao termo viés.

Informalmente, o termo viés mede o *erro persistente* do algoritmo de aprendizagem. Esse erro não pode ser eliminado pelo aumento do número de classificadores treinados independentemente. O termo variância mede a *flutuação* do erro que é devido à utilização de uma amostra particular de exemplos de treino. Para um número crescente de exemplos, essa componente se reduz, e tende para zero quando o número de exemplos tende para infinito.

Deve-se notar que a soma do componente viés mais o componente variância pode diferir ligeiramente da taxa de erro medida usando validação cruzada. Pode ocorrer porque são usados conjuntos de treino de diferentes tamanhos para estimar essas quantidades.

9.6 Considerações Finais

Neste capítulo foram discutidas questões relativas ao desenho de experiências e à avaliação de modelos preditivos. Estes aspetos são de grande relevância, uma vez que, em geral, os trabalhos envolvendo o uso de técnicas de ECD devem incluir a realização de experiências controlados.

A medida de desempenho mais utilizada para avaliar classificadores é a taxa de erro (ou de acerto), enquanto em regressão é comum a análise do MSE do modelo. Em trabalhos na área médica, é usual observar a sensibilidade do modelo. No domínio de extração de conhecimento a partir de texto e recuperação de informação, é frequente a análise da precisão do modelo, ou seja, o quanto as suas predições positivas são confiáveis. Todas as medidas de desempenho apresentadas neste capítulo consideram o poder preditivo dos modelos obtidos, embora outros aspetos possam ser levados em consideração, tais como a sua interpretabilidade, o seu tamanho, entre outros. No caso de problemas de classificação

binários, outra medida adotada na avaliação preditiva dos modelos é o AUC, proveniente da análise de curvas ROC.

Foram discutidos ainda métodos de amostragem de dados, com o objetivo de obter subconjuntos para uma estimativa apropriada do desempenho do preditor. Na prática, o *r-fold cross-validation* (estratificado para problemas de classificação) é o método de amostragem mais utilizado, com um número de partições (*folds*) normalmente igual a 10. Em conjuntos de exemplos pequenos, o uso do 10-*fold cross-validation* pode levar à obtenção de subconjuntos de teste com poucos dados. Nestes casos, é recomendável usar o *leave-one-out*, que produz estimativas mais fiáveis do erro esperado, ou o *booststrap*.

O *holdout* é muitas vezes usado para definir subconjuntos para o ajuste de parâmetros das técnicas a partir das partições de treino, também denominados subconjuntos de validação, tais como os usados pelas RNAs na estratégia de treino *early-stop* (Seção 7.1.4). É importante observar que os dados de teste nunca devem ser considerados em nenhuma etapa indutiva ou de ajuste do modelo. Só devem ser utilizados aquando da avaliação final. Dessa forma garante-se uma fiel simulação da chegada de novos dados nunca apresentados ao modelo quando este estiver sendo utilizado na prática na predição dos rótulos de novos dados. Normalmente, nos estudos experimentais realizados em ECD é feita a avaliação de dois ou mais algoritmos, cujos resultados devem ser comparados. Nesse caso, é recomendável proceder a um teste de hipóteses, que poderá indicar a um nível de confiança, normalmente de 95%, se o desempenho dos algoritmos difere de fato ou não. A literatura a respeito da forma apropriada de comparar preditores com os testes estatísticos ainda é muito incipiente, e não há um consenso sobre os testes mais corretos a serem usados na prática. Optou-se por apresentar neste capítulo alguns dos testes mais utilizados recentemente, embora outros trabalhos possam fornecer outras metodologias. Por fim, apresentou-se a teoria de decomposição viés-variância do erro de preditores, uma ferramenta útil na análise do comportamento de algoritmos.

Parte III

Modelos Descritivos

Capítulo 10

Introdução aos Modelos Descritivos

As tarefas da aprendizagem descritiva, ou não supervisionada, referem a identificação de estruturas num conjunto de dados. Essencialmente, a aprendizagem reside na identificação de propriedades intrínsecas aos dados, de maneira a construir representações desses dados que possam servir como auxílio a tomada de decisões ou à descoberta de conhecimento. Estas técnicas são utilizadas principalmente quando o objetivo do aprendizado é encontrar padrões ou tendências que auxiliem na compreensão dos dados (Souto et al., 2003). Mais precisamente, na aprendizagem não supervisionado não existem atributos alvo. A partir do conjunto de dados $\mathbf{X}$, um algoritmo de ECD não supervisionado aprende uma representação compacta dos dados segundo algum critério de qualidade.

Como já definido, as tarefas descritivas podem ser divididas em: sumarização, associação e agrupamento. A sumarização tem como objetivo encontrar uma descrição simples e compacta dos dados. Para isso, podem ser utilizadas desde medidas estatísticas simples como mínimo, média, desvio padrão, até técnicas sofisticadas de visualização e de determinação de relações funcionais entre atributos (Han e Kamber, 2000; Mirkin, 2011). Algumas medidas estatísticas e técnicas de visualização mais simples foram descritas no Capítulo 2. Já a associação refere a busca de padrões frequentes de associações entre os atributos de um conjunto de dados. A análise de agrupamento, por sua vez, lida com a identificação de grupos nos dados de acordo com a semelhança entre os objetos. Nesta parte do livro serão apresentados a associação (Capítulo 11), sendo dada mais ênfase às tarefas de análise de agrupamento (Capítulos 12 a 15). Nos capítulos seguintes, serão apresentados os principais aspectos da aprendizagem não supervisionada, cem particular o tema de análise de agrupamentos.

A extração de um *conjunto de itens* frequentes é um dos temas de grande importância na descoberta de conhecimento em bases de dados. Os primeiros trabalhos nessa área visavam identificar grupos de produtos que frequentemente eram comprados em conjunto para auxiliar, por exemplo, campanhas de marketing. No Capítulo 11 será detalhado o

tema da extração de padrões frequentes, sendo apresentados os principais algoritmos sobre esse tema.

Já as técnicas de agrupamento são instrumentos valiosos na análise exploratória de dados e encontram aplicações em várias áreas, tais como: Biologia, Medicina, Engenharia, Marketing, Visão Computacional, Deteção Remota e Bioinformática. A análise de agrupamento é apropriada para explorar e verificar estruturas presentes em um conjunto de dados.

Como já mencionado, a análise de agrupamentos pertence ao paradigma de aprendizagem não supervisionada, não requerendo conhecimento prévio sobre as suas classes ou categorias (Mitchell, 1997). Considerando a descrição dos dados feita na Seção 2.1, a análise de agrupamentos é voltado para dados que não possuem um atributo de saída ou atributo alvo. Esta característica, aliada à falta de uma definição precisa e única do conceito de *cluster* (ou grupo), gera uma série de dificuldades tanto na escolha dos algoritmos mais apropriados, mas principalmente na avaliação dos resultados obtidos.

Assim, em muitos aspectos, a análise de agrupamentos tem um caráter subjetivo, sendo seus resultados altamente influenciados pelo perfil do profissional que realiza a análise. Por exemplo, é bastante comum os biológos preferirem algoritmos hierárquicos, devido à sua familiaridade com as estruturas obtidas pelos algoritmos, semelhantes às taxonomias bastante comuns na área. Essa escolha, nem sempre, leva à utilização do algoritmo mais apropriado.

Em resumo, os pontos mais críticos quando se aplica a análise de agrupamentos estão relacionados com a grande diversidade de critérios de agrupamento, heterogeneidade de critérios que podem definir uma estrutura dos dados, possibilidade de haver várias estruturas que descrevam um mesmo conjunto de dados e falta de conhecimento das estruturas verdadeiras presentes nos dados para guiar os resultados obtidos e também as comparações entre algoritmos. Todos esses problemas, bem como as principais abordagens para solucioná-los, serão detalhados ao longo dos Capítulos 12 a 15. Mais especificamente, os conceitos básicos de agrupamento, bem como as etapas essenciais à análise de agrupamento, serão apresentados no Capítulo 12. Alguns dos principais algoritmos de agrupamento serão apresentados no Capítulo 13. No Capítulo 14, serão apresentadas formas de combinar múltiplos modelos descritivos. Finalmente, no Capítulo 15 serão discutidas formas de desenhar experiências e de analisar os resultados obtidos na análise de agrupamento.

Capítulo 11

Extração de Padrões Frequentes

A extração de *conjuntos de itens* frequentes[1] é um dos tópicos mais populares na descoberta de conhecimento em bases de dados, e dos mais utilizados na industria e comércio. O trabalho pioneiro nesta área foi a análise de carrinhos de compras, mais especificamente, a análise de dados transacionais que descrevem o comportamento de consumo dos clientes (Agrawal et al., 1993). O objetivo desta análise consiste em descobrir grupos de produtos que são frequentemente comprados em conjunto e, com base nesses grupos, inferir os produtos que serão comprados, dado que foram comprados outros produtos. Para resolver este problema, foi desenvolvido um grande número de algoritmos eficientes, que têm sido aplicados aos mais diversos problemas. Neste capítulo, serão analisados alguns dos principais algoritmos para extração de conjunto de itens frequentes, regras de associação, assim como as suas extensões para sequências de itens, onde é importante considerar a ordem pela qual os itens de um conjunto aparecem.

11.1 Extração de Conjuntos de Itens Frequentes

Seja $A = \{a_1, \ldots, a_m\}$ o universo de m itens. Os itens podem ser produtos num supermercado, componentes de equipamentos, opções de serviço, páginas web, etc. Qualquer subconjunto $I \subseteq A$ é denominado um conjunto de itens (*itemset*). Um conjunto de itens diz respeito a qualquer grupo de produtos que podem ser comprados em conjunto.

Seja $T = (t_1, \ldots, t_n)$ um conjunto de n transações, denotado por *base de dados de transações*. Cada transação é um par $\langle tid_i, k-items_i \rangle$, em que tid_i é a identificação da transação e $k-item_i \subseteq A$ é um conjunto de k itens. Uma base de dados de transações pode listar, por exemplo, o conjunto de produtos adquiridos por um cliente num supermercado, numa dada compra, o conjunto de páginas visitadas por um utilizador de um *site* numa sessão, etc. Cada transação é um conjunto de itens, mas alguns conjuntos de itens podem não aparecer no conjunto T. Diz-se que uma transação $t \in T$ suporta o conjunto de itens I, ou o conjunto de itens I está contido no $k-item_i$ da transação t, se e somente se, $I \subseteq t$. Ou

[1] Também denominado *itemset*.

TID	Itens
1	{a,d,e}
2	{b,c,d}
3	{a,c,e}
4	{a,c,d,e}
5	{a,e}
6	{a,c,d}
7	{b,c}
8	{a,c,d,e}
9	{b,c,e}
10	{a,d,e}

0 itens	1 item	2 itens	3 itens
Ø: 10	{a}: 7	{a,c}: 4	{a,c,d}: 3
	{b}: 3	{a,d}: 5	{a,c,e}: 3
	{c}: 7	{a,e}: 6	{a,d,e}: 4
	{d}: 6	{b,c}: 3	
	{e}: 7	{c,d}: 4	
		{c,e}: 4	
		{d,e}: 4	

Figura 11.1 *Uma base de dados de transações, com 10 transações, e a enumeração de todos os conjuntos de itens frequentes usando o suporte mínimo de* $s_{\min} = 3$.

seja, a transação t contém todos os elementos do conjunto de itens I. Por exemplo, uma transação em que foram comprados os itens queijo, pão e manteiga, suporta, ou dá suporte, ao conjunto de itens formado pelos itens pão e queijo. Intuitivamente dar suporte significa fortalecer ou testemunhar a favor. O conjunto $K(I)$ de transações que suporta o *itemset* I é designado por suporte do *itemset*. A partir de conjuntos frequentes, é possível derivar regras de associação. As regras de associação têm a forma de regras **se** antecedente **então** consequente, em que o antecedente e o consequente são *itemsets*. Por exemplo, se o cliente compra pão então também compra manteiga. Estas regras são calculadas a partir dos dados e têm natureza probabilística. O grau de incerteza de uma regra é dado pela *confiança* da regra, que pode ser definida como a relação entre o número de transações que incluem todos os itens no conjunto $\{consequente \cup antecedente\}$, e o número de transações que incluem todos os itens do antecedente.

O suporte pode ser, ainda, **absoluto** ou **relativo**. O suporte absoluto de um *itemset* é o número total de elementos do conjunto $K_T(I)$. O suporte relativo de um *itemset* é a fração de transações que o contêm. O suporte relativo é calculado através da divisão do suporte absoluto pelo número de transações. A Figura 11.1 apresenta uma base de dados de transações, com 10 transações, e a enumeração de todos os conjuntos de itens frequentes usando o suporte mínimo de $s_{\min} = 3$, ou $\sigma_{\min} = 0,3 = 30\%$.

Formalmente, o suporte absoluto de I, em relação a T, $s_T(I)$, é definido pela expressão $s_T(I) = |K_T(I)|$. A expressão $\sigma_T(I) = \frac{1}{n}|K_T(I)|$ calcula o suporte relativo de I com respeito a T. Algumas vezes, $\sigma_T(I)$ é também denominado de frequência *(relativa)* de I em T. O problema da extração dos conjuntos de itens frequentes pode ser definido como:

- **Dados:**
  - um conjunto $A = \{a_1, \ldots, a_m\}$ de itens,
  - uma tabela $T = (t_1, \ldots, t_n)$ de transações sobre A,
  - um número $\sigma_{\min}$ tal que $0 < \sigma_{\min} \leq 1$, i.e. o **suporte mínimo**.
- **Objetivos:**

- encontrar o conjunto de **itens frequentes**, tais que o suporte relativo de cada conjunto de itens é maior ou igual ao $\sigma_{\min}$ definido pelo utilizador.
- encontrar o conjunto de regras de associação com confiança maior que um mínimo definido pelo utilizador.

Desde a sua introdução por Agrawal et al. (1993), os problemas de aprendizagem de *itemsets* frequentes e regras de associação receberam uma grande atenção. Na década de 1990, foram publicados centenas de artigos propondo novos algoritmos, ou melhorias nos algoritmos existentes, para resolver estes problemas de aprendizagem de uma forma mais eficiente. Um dos objetivos desta área de AM é o de conhecer quais são os artigos comprados em conjunto. Algumas das conclusões que se podem tirar são do tipo: *20% das pessoas que compram café também compram bolachas*. Esta informação, quando conhecida, poderá resultar num conjunto de ações que vão desde a promoção conjunta de artigos até alterações da sua localização no supermercado.

11.1.1 O Espaço de Procura

O espaço de procura de todos os possíveis conjuntos de itens para um conjunto de itens A contém exatamente $2^{|A|}$ *itemsets* diferentes. O espaço de procura pode ser representado por um reticulado, com o *itemset* vazio no topo e o conjunto de todos os itens na base. A Figura 11.2 ilustra o reticulado para 5 itens. Se $|A|$ é suficientemente grande, então uma proposta simples de gerar, e contar, os suportes de todos os *itemsets* sobre a base de dados, não pode ser alcançada dentro de um período de tempo razoável. A principal propriedade explorada pela maioria dos algoritmos de extração de conjuntos de itens frequentes é que o suporte é monotonamente decrescente, em relação ao número de itens de um *itemset*. Ou seja, o conjunto de suporte de um conjunto de itens diminui sempre que se acrescenta um novo item. Formalmente, considere X e Y dois conjuntos de itens numa base de dados de transações T sobre I, tal que $X, Y \subseteq I$. É fácil mostrar que $X \subseteq Y \Rightarrow suporte(Y) \leq suporte(X)$. Esta regra é uma consequência imediata do fato do conjunto de suporte de X estar incluído no conjunto de suporte de Y. Por esse motivo, se um *itemset* é pouco frequente, todos os seus superconjuntos são pouco frequentes. Adicionalmente, a propriedade *todos os subconjuntos de um conjunto de itens frequente são frequentes* é válida. Esta é a *propriedade da monotonia* do suporte.[2]

11.2 O Algoritmo Apriori

O algoritmo *Apriori*, introduzido em Agrawal et al. (1993); Agrawal e Srikant (1994), foi o primeiro algoritmo proposto para a extração de *itemsets* e regras de associação. Baseia-se no princípio de que qualquer subconjunto de *itemsets* frequentes deve ser um *itemset* frequente. O algoritmo Apriori (Algoritmo 11.1) utiliza uma estratégia de procura em largura (do inglês, *breath-first*), com um algoritmo de geração e teste. Em cada nível são

[2] Uma função diz-se monótona em x, se $x_1 < x_2$ implica que $f(x_1) < f(x_2)$.

TID	Item set
1	{a,d,e}
2	{b,c,d}
3	{a,c,e}
4	{a,c,d,e}
5	{a,e}
6	{a,c,d}
7	{b,c}
8	{a,c,d,e}
9	{b,c,e}
10	{a,d,e}

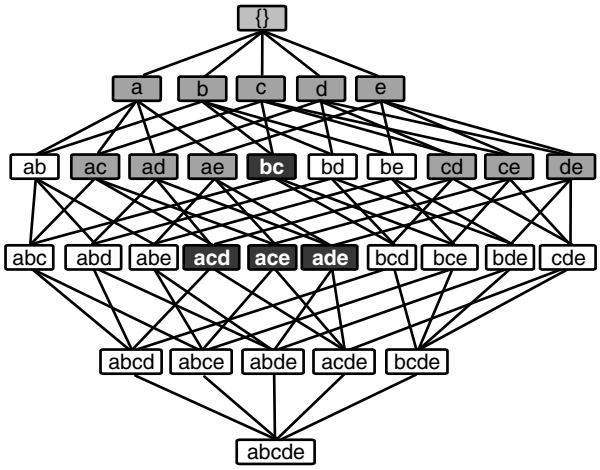


Figura 11.2 *Uma base de dados de transações, com 10 transações, e o espaço de procura para encontrar todos os possíveis itemsets frequentes usando o suporte mínimo de* $s_{\min} = 3$.

gerados os *itemsets* possíveis, tendo em conta os *itemsets* frequentes gerados no nível anterior. Após serem gerados, a frequência desses *itemsets* é testada, percorrendo novamente a base de dados de transações.

O algoritmo Apriori enceta com a geração do conjunto F_1 de *itemsets* de tamanho 1, de modo que, cada item, é um membro do conjunto de *itemsets* candidatos. Os *itemsets* de tamanho $k+1$ são obtidos a partir dos *itemsets* de tamanho k, em dois passos. No primeiro passo, é realizada uma autocombinação sobre o conjunto F_k, quando um conjunto de candidatos com $k+1$ *itemsets* é gerado pela combinação de F_k com ele mesmo. A união $X \cup Y$ dos *itemsets* $X, Y \in F_k$ é gerada se eles têm o mesmo $k-1$-prefixo. Este passo pode ser realizado eficientemente se os *itemsets* estiverem em ordem lexicográfica. No segundo passo, denominado *passo de poda*, $X \cup Y$ é inserido em F_{k+1} somente se, todos os seus k-subconjuntos, ocorrem em F_k. Depois disso, é necessário contar os suportes de todos os $k+1$ *itemsets* candidatos. Para esse efeito, a base de dados é varrida, uma transação por vez, e os suportes de todos os *itemsets* candidatos que estão incluídos naquela transação são incrementados. Todos os *itemsets* que se tornam frequentes são inseridos em F_{k+1}. Deste modo, o algoritmo executa uma procura em largura no espaço de pesquisa.

A Figura 11.1 apresenta uma base de dados de transações e a enumeração de todos os possíveis *itemsets* frequentes, usando o suporte mínimo de $s_{\min} = 3$ ou $\sigma_{\min} = 0,3 = 30\%$. Existem $2^5 = 32$ conjuntos de itens possíveis sobre $A = \{a,b,c,d,e\}$. Nesta base de dados de transações, há 15 conjuntos de itens frequentes (para o suporte mínimo de $0,3$). Há 5 conjuntos frequentes com 1 elemento, 7 conjuntos frequentes com 2 elementos e 3 conjuntos frequentes com 3 elementos (Figura 11.1). Para esse nível de suporte, não há nenhum conjunto frequente com 4 elementos.

Algoritmo 11.1 O algoritmo Apriori para gerar *itemsets* frequentes

Entrada: Uma base de dados de transações *DB*;
Um limiar do suporte mínimo σ;
Saída: Todos os *itemsets* com suporte maior que σ

```
1  Percorrer a base de dados DB uma vez;
2  Calcular C_1, o conjunto de itens frequentes e o suporte de cada item;
3  k ← 1;
4  enquanto C_k ≠ {} faça
5      para cada transação I ∈ DB faça
6          para cada itemset candidato X ∈ C_k faça
7              se X ⊂ I então
8                  Incrementa o suporte de X ;
9              fim
10         fim
11     fim
12     /* Extrai todos os itemsets frequentes */ ;
13     C_k ← {X|X : Suporte ≥ σ} ;
14     /* Gera os novos itemsets candidatos */;
15     para cada X,Y ∈ C_k faça
16         se X[i] = Y[i] para 1 ≥ i ≥ k−1 eX[k] < Y[k] então
17             I = X ∪ {Y[k]} ;
18             se ∀J ∈ I, |J| = k : J ∈ C_k então
19                 C_{k+1} ← C_{k+1} ∪ I
20             fim
21         fim
22     fim
23     k ← k+1;
24 fim
```

11.2.1 Regras de Associação

Além de encontrar quais os termos que são adquiridos em conjunto, também é útil saber quais são as combinações de termos que poderão ser inferidos, tendo por base o conhecimento de alguns dos elementos. As combinações de termos são dadas pela construção de regras sob a forma $A \rightarrow B$, em que $A \cup B$ é um *itemset* frequente. O grau de interesse é representado pela confiança das regras, dada pela Equação 11.1, e que consiste na probabilidade de ocorrer um conjunto de termos dado que ocorreu um outro conjunto.

$$\text{confiança}(A \rightarrow B) = \frac{P(A \cup B)}{P(A)} = \frac{suporte(A \cup B)}{suporte(A)} \tag{11.1}$$

Tabela 11.1 *Regras extraídas*

Regra	Confiança	Suporte do *itemset*
$\{b\} \rightarrow \{c\}$	100,0%	30%
$\{d,e\} \rightarrow \{a\}$	100,0%	40%
$\{e\} \rightarrow \{a\}$	85,7%	60%
$\{a\} \rightarrow \{e\}$	85,7%	60%
$\{d\} \rightarrow \{a\}$	83,3%	50%
$\{a,d\} \rightarrow \{e\}$	80,0%	40%

Algoritmo 11.2 O algoritmo Apriori para gerar regras de associação

Entrada: *Itemset* frequente I;
Limiar de confiança $conf_{min}$;
Saída: R: Todas as regras de associação em I com confiança maior que $conf_{min}$

```
1 Gera todos os subconjuntos não vazios de I ;
2 R = ∅ ;
3 para cada subconjunto não vazio s de I faça
4     Avalia a confiança da regra s → {I\s} ;
5     se suporte(I)/suporte(s) ≥ conf_min então
6         R ← R ∪ {s → {I\s}} ;
7     fim
8 fim
```

A segunda fase do algoritmo Apriori gera regras a partir dos conjuntos de termos frequentes encontrados, desde que estes obedeçam ao critério de confiança mínima. O algoritmo básico, descrito no Algoritmo 11.2, gera para cada *itemset* frequente, todos os subconjuntos não vazios. Sendo s um subconjunto do *itemset* i, é calculada a confiança da regra $s \rightarrow \{I \backslash s\}$. O algoritmo seleciona as regras cuja confiança é superior à confiança mínima definida pelo utilizador.

Retomando o exemplo da Figura 11.1, e para o conjunto de conjuntos frequentes obtidos com suporte mínimo de $0,3$, serão extraídas todas as regras, considerando o valor de 80% para o parâmetro de confiança, conforme ilustrado na Tabela 11.1. O processo consiste em testar todas as combinações de termos que possam formar regras. Deverão existir, pelo menos, um item como antecedente da regra e outro como consequente. A Figura 11.3 ilustra o processo de procura de regras de associação a partir do *itemset* frequente $\{a,d,e\}$. A geração de regras de associação não requer mais nenhuma passagem

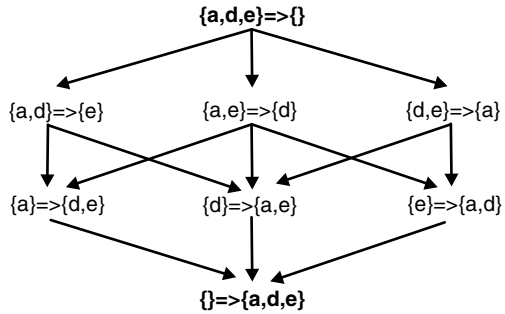


Figura 11.3 *Espaço de procura de regras de associação para um itemset frequente.*

pela base de transações. O Algoritmo 11.2 pode ser otimizado tendo em conta que, quando um item é movido do antecedente para o consequente, a confiança não pode aumentar.

11.2.2 Discussão

A estratégia de procura do algoritmo Apriori implica percorrer diversas vezes a base de dados para calcular o suporte dos *itemsets* frequentes candidatos. Em alternativa, foram propostos diversos algoritmos com o objetivo de minimizar a necessidade de percorrer repetidamente a base de transações. Estes algoritmos geram conjuntos de *itemsets* candidatos adotando uma estratégia de procura em profundidade. O primeiro algoritmo proposto foi o algoritmo Eclat (do inglês *Equivalence CLass Transformation*), desenvolvido por Zaki (2000), e o algoritmo FP-growth (do inglês *Frequent Pattern Growth*) criado por Han et al. (2004). Esse último algoritmo, descrito na Seção 11.3, utiliza uma árvore de prefixos (*trie*) para armazenar *itemsets* (Figura 11.4). Isto evita as autocombinações requeridas no Apriori para geração dos candidatos. Para gerar todas as possíveis extensões de um *itemset* por meio de um único item, simplesmente adiciona-se o item à árvore de sufixo. Esta estratégia de procura, gera cada conjunto de itens candidatos, no máximo, uma vez. O algoritmo FP-growth foi utilizado como um módulo para a construção de *itemsets* frequentes (Chi et al., 2004) e extração de sequências em fluxos contínuos de dados (Pei et al., 2001).

11.3 O Algoritmo FP-growth

A estratégia de procura em profundidade e as árvores de sufixo, utilizadas pelo algoritmo FP-growth, são empregues na maioria dos algoritmos de extração de padrões frequentes aplicados a dados de fluxo contínuo (Chi et al., 2004). Para a construção de regras de associação, o algoritmo FP-growth procede em duas fases. Na primeira fase constrói uma estrutura de dados, a FP-tree, percorrendo a base de dados duas vezes. A FP-tree é então

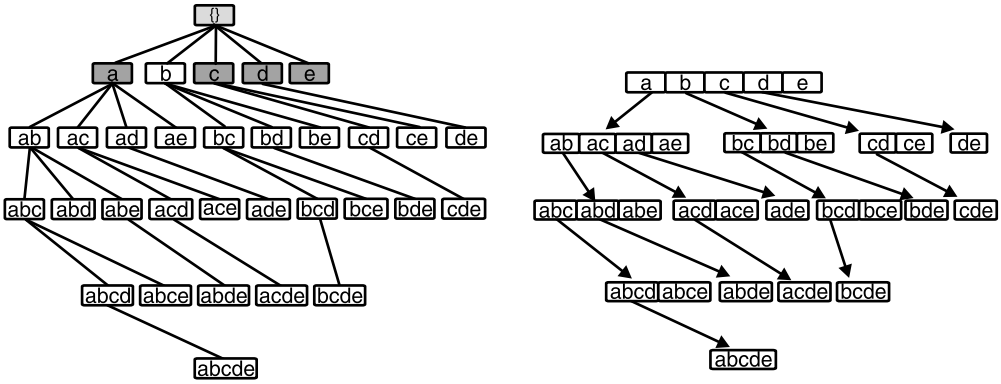


Figura 11.4 *O espaço de procura em profundidade e a árvore de prefixos correspondente para 5 itens.*

usada para encontrar as regras de associação. O pseudocódigo[3] do algoritmo para construir a FP-tree é apresentado no Algoritmo 11.3. A ideia base consiste em percorrer o espaço de procura em profundidade, conforme ilustrado na Figura 11.4.

O processo de construção de uma FP-tree é apresentado na Figura 11.5. O algoritmo atravessa duas vezes a base de dados. Na primeira vez, encontra o conjunto de itens frequentes (1-*itemsets*) e os respetivos suportes. O conjunto de itens frequentes é ordenado por ordem decrescente do seu suporte e armazenado numa matriz **L**. No caso da base de transações da Figura 11.5, o item mais frequente é o a, seguido de b, c, d e e. O algoritmo percorre uma segunda vez a base de transações, construindo a árvore de padrões frequentes, FP-tree, como se segue. Primeiro, cria-se o nó raiz da árvore, rotulado com *null*. Para cada transação presente na base de dados, os itens são processados em ordem decrescente de suporte. Por exemplo, a primeira transação $\{a,b\}$, gera o percurso: $null \rightarrow a \rightarrow b$. Cada nó tem associado um contador de frequência com o valor 1. A segunda transação gera um novo conjunto de nós, correspondente ao percurso: $null \rightarrow b \rightarrow c \rightarrow d$. Esse percurso é disjunto do primeiro porque as transações não compartilham um prefixo comum. Como b aparece nos dois percursos, é estabelecida uma ligação que possibilita o cálculo da frequência de b. A terceira transação partilha um prefixo comum (o item a, com a primeira transação). Por esse motivo, o percurso $null \rightarrow a \rightarrow c \rightarrow d \rightarrow e$ sobrepõe-se ao primeiro percurso, e a frequência de a é incrementada. Esse processo continua até todas as transações terem sido processadas.

A razão para a primeira leitura da base de dados e para o processamento de transações, por ordem decrescente do suporte, está relacionado com a disposição dos itens na *FP-tree*. Quanto mais frequentes, mais próximo da raiz aparecem, o que favorece a possibilidade de serem compartilhados. Dessa forma, a representação da FP-tree da base de dados é

[3] Seguimos o pseudocódigo apresentado em Han et al. (2004).

Algoritmo 11.3 O algoritmo FP-tree

Entrada: Uma Base de Dados de Transações *DB*;
Um limiar do suporte mínimo σ;
Saída: Todos os *itemsets* com suporte maior que σ

```
1  Percorrer a base de dados DB uma vez ;
2  Calcular F, o conjunto de itens frequentes e o suporte de cada item ;
3  Ordenar F em ordem decrescente de suporte como Flist ;
4  Criar a raiz da FP-tree, T, e rotulá-la como null ;
5  para cada transação t ∈ DB faça
6      Selecionar os itens frequentes em t ;
7      Ordená-los de acordo com Flist ;
8      Seja [i|It] os itens frequentes ordenados em t ;
9      Chamar insere_arvore([i|Its], T);
10 fim
11 Função insere_arvore([i|Its], T) ;
12 se T tem um filho N rotulado i então
13     Incremente os contadores de N com 1 ;
14 fim
15 senão
16     Crie um novo nó, N, com o seu contador inicializado com 1, o seu pai ligado a
       T, e o seu nó ligado aos nós com o mesmo rótulo i ;
17     se Its é não vazio então
18         Chamar insere_arvore(Its, N);
19     fim
20 fim
```

mantida tão pequena quanto possível. Além disso, é possível eliminar da análise os itens com suporte inferior ao suporte mínimo.

11.4 Sumarização de *Itemsets*

A abordagem utilizada por um algoritmo de descoberta de regras de associação pode fazer com que o número de regras geradas seja muito grande, e que nem todas as regras encontradas sejam interessantes. Por exemplo, o conjunto de todos os *itemsets* frequentes, pode ser completamente representado por um conjunto com menos elementos, por via da eliminação de todos os *itemsets* que são subconjuntos de outros *itemsets* frequentes. A *propriedade da monotonia* do suporte sugere uma representação compacta do conjunto de *itemsets* frequentes:

Transações

TID	Itens
1	{a,b}
2	{b,c,d}
3	{a,c,d,e}
4	{a,d,e}
5	{a,b,c}
6	{a,b,c,d}
7	{a}
8	{a,b,c}
9	{a,b,d}
10	{b,c,e}

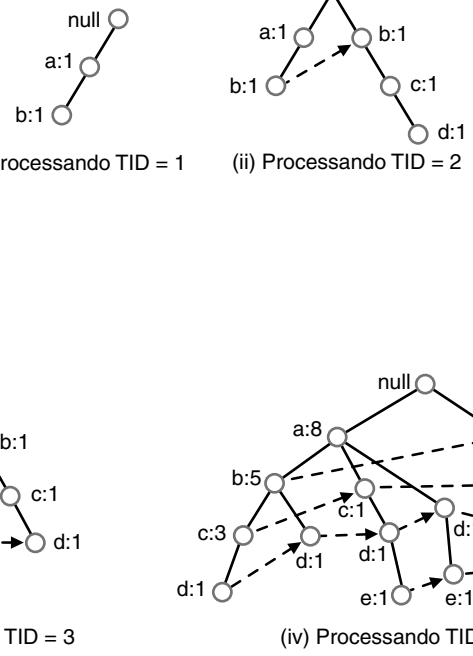


Figura 11.5 *Construção de uma FP-tree.*

- *Itemsets frequentes maximais*
  Um conjunto de itens é maximal se é frequente mas nenhum dos seus superconjuntos próprios é frequente.[4]

- *Itemsets frequentes fechados*
  Um conjunto frequente é chamado *fechado* se, e somente se, não contém tem superconjuntos frequentes com a *mesma frequência*.

No exemplo da Figura 11.1, há 13 conjuntos frequentes fechados (suporte mínimo 0,3): $\{b,c\},\{d,c,a\},\{e,c,a\},\{c,a\},\{d,c\},\{e,c\},\{d,e,a\},\{d,a\},\{d\},\{e,a\},\{a\},\{c\},\{e\}$, e 4 conjuntos de itens maximais, que são: $\{b,c\}\{a,c,d\}\{a,c,e\}\{a,d,e\}$. Todos os *itemsets* frequentes podem ser vistos como um subconjunto de, pelo menos, um dos conjuntos maximais.

O seguinte relacionamento é válido entre esses conjuntos: $Maximal \subseteq Fechado \subseteq Frequente$. Os *itemsets* maximais são um subconjunto dos *itemsets* fechados. A partir dos *itemsets* maximais, é possível derivar todos os *itemsets* frequentes (mas não o seu suporte)

[4] A é um superconjunto próprio de B se, e somente se, a cardinalidade de A é maior que a de B, e todos os elementos de B pertencem a A.

Tabela 11.2 *Preferências sobre o consumo de Chá e Café de 1000 consumidores*

	Café	Não Café	Total
Chá	150	50	200
Não Chá	650	150	800
Total	800	200	1000

calculando todas as interseções não vazias, justificado pelo fato de que, se um *itemset* é frequente, todos os seus subconjuntos são também frequentes.

O conjunto de todos os conjuntos de itens fechados preserva o conhecimento sobre o valor do suporte de todos os *itemsets* frequentes, devido ao fato de que todos os subconjuntos de um *itemset* fechado têm o mesmo suporte.

11.4.1 Heurísticas para Seleção de Regras de Associação

Os algoritmos de regras de associação tendem a gerar um número excessivo de regras. Nos últimos anos, têm sido propostas várias medidas para extrair padrões *interessantes* a partir de grandes bases de dados. A ideia consiste em selecionar um subconjunto de padrões ou regras que de alguma forma sejam mais relevantes. Nesta seção, serão apresentadas as caraterísticas gerais de uma medida de interesse e definidas algumas medidas concretas para extração de padrões interessantes. Uma medida objetiva deverá ser baseada nos dados e ser independente do domínio de dados considerado. Foram definidos por Piatetsky-Shapiro (1991) três princípios a que qualquer medida (M) deve obedecer:

- Se A e B são duas variáveis estatisticamente independentes, então $P(A,B) = P(A).P(B)$ e $M = 0$. Isto significa que, quando duas variáveis são estatisticamente independentes, a medida de interesse assume o valor zero;
- A medida M cresce se $P(A,B)$ crescer, e se todos os restantes parâmetros permanecerem constantes;
- A medida M decresce se $P(A)$, ou $P(B)$, decrescerem, mantendo-se constantes todos os demais parâmetros.

O primeiro princípio estabelece que os padrões que ocorrerem aleatoriamente não são de interesse. O segundo princípio estabelece que o interesse de uma medida deverá ser tanto maior, quanto maior for a sua significância estatística. E, finalmente, o terceiro princípio é utilizado para comparar os valores de interesse de padrões com a mesma significância estatística. Assuma que estamos interessados em estudar a relação entre as pessoas que bebem chá e café. Após obter informação sobre as preferências de 1000 pessoas, sumarizamos esta informação na Tabela 11.2. Esta informação pode ser usada para avaliar a associação da regra $\{\text{Chá}\} \rightarrow \{\text{Café}\}$. Assim, o suporte da regra é $150/1000 = 0,15$, e a confiança $0,15/0,2 = 0,75$. A confiança da regra é elevada, no entanto, a probabilidade de uma pessoa beber café, independentemente de beber chá, é de 80%. Sabendo que

uma pessoa bebe chá, diminui a probabilidade desta pessoa também beber café! A regra $\{\text{Chá}\} \rightarrow \{\text{Café}\}$ de fato é enganadora.

Coeficiente de Interesse ou *Lift*

O coeficiente de interesse, ou *Lift*, reflete a noção estatística de independência entre duas variáveis aleatórias. Foi abordado por muitos autores, como uma medida para avaliar os níveis de associação. Esta métrica é definida pelo quociente entre a probabilidade conjunta de duas variáveis em relação à sua probabilidade, pressupondo a hipótese de independência. É calculada pela Equação 11.2. Nesta equação, o valor 1 corresponde à independência estatística entre as variáveis A e B.

$$I(A,B) = \frac{P(A,B)}{P(A)P(B)} \tag{11.2}$$

O coeficiente de interesse aplicado a regras de associação costuma ser designado por $lift$. O $lift$ não é mais do que o quociente entre a confiança e o valor esperado para a confiança. A confiança esperada é o número de transações que incluem o consequente dividido pelo número total de transações, conforme ilustrado pela Equação 11.3.

$$lift(A \rightarrow B) = \text{confiança}(A \rightarrow B)/\text{suporte}(B) = \frac{\text{suporte}(A \cup B)}{\text{suporte}(A) \times \text{suporte}(B)} \tag{11.3}$$

Um valor de $lift$ igual a 1 indica que A e B são independentes. Valores de $lift$ inferiores a 1 indicam que A e B são negativamente correlacionados, enquanto valores superiores a 1 indicam uma correlação positiva. O $lift$ da regra $\{\text{Chá}\} \rightarrow \{\text{Café}\}$, apresentada na Tabela 11.2, é $0,15/(0,2 \times 0,8) = 0,9375$. Esse resultado evidencia a correlação negativa entre Chá e Café. O $lift$ é uma medida para o desvio da regra em relação à independência estatística entre o antecedente e consequente de uma regra de associação. Toma valores entre 0 e infinito. Um valor de $lift$ superior a 1 indica que A e B aparecem mais frequentemente juntos do que o esperado, o que significa que a ocorrência de A tem um efeito positivo sobre a ocorrência de B. Um $lift$ menor do que 1 indica que A e B aparecem em conjunto com menos frequência do que o esperado, indicando que a ocorrência de A tem um efeito negativo sobre a ocorrência de B. Um valor próximo de 1 indica que A e B aparecem quase sempre juntos. Isto significa que a ocorrência de B não tem efeito sobre a ocorrência de A.

Convicção

A última medida apresentada, a *convicção* de uma regra, mede o quão convincente é a regra. A convicção pode ser interpretada como o quociente da frequência esperada de A ocorrer sem B (ou seja, a frequência de erro da regra), como se A e B fossem inde-

pendentes, dividido pela frequência de previsões incorretas. Esta medida é definida pela Equação 11.4.

$$\text{convicção}(A \rightarrow B) = \frac{1 - \text{suporte}(B)}{1 - \text{confiança}(A \rightarrow B)} \tag{11.4}$$

A convicção da regra $\{\text{Chá}\} \rightarrow \{\text{Café}\}$, apresentada no exemplo anterior, é: $(1 - 0,8)/(1 - 0,75) = 0,8$. Tal como a medida $lift$, na medida de convicção valores inferiores a 1 indicam que a associação entre A e B é aleatória.

11.5 Considerações Finais

A aprendizagem de regras de associação é um tema muito investigado e um método popular para a descoberta de relações interessantes entre as variáveis, em grandes bases de dados. As regras de associação podem ser usadas como base para decisões sobre atividades de marketing como, por exemplo, promoção de preços ou colocação de produtos. Recentemente, a análise de regras de associação tem sido usada em muitas outras áreas de aplicação, incluindo sistemas de recomendação, extração de conhecimento na Web, deteção de intrusos e bioinformática.

Capítulo 12

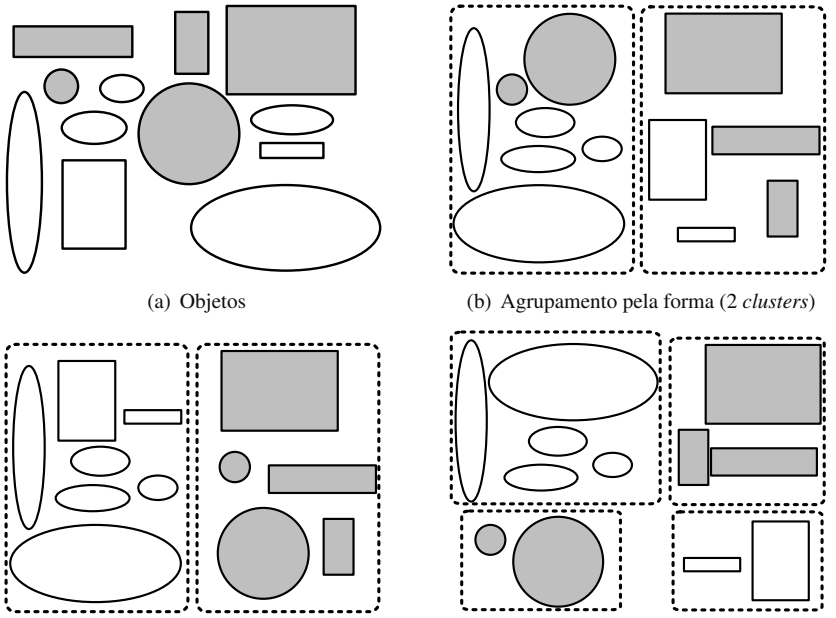
Análise de Agrupamentos

O objetivo da análise de agrupamentos é encontrar uma estrutura de grupos *clusters* nos dados de forma a que os objetos pertencentes a cada *cluster* compartilham alguma característica ou propriedade relevante para o domínio do problema em estudo, ou seja, são de alguma maneira similares (Jain e Dubes, 1988). Por exemplo, a Figura 12.1 ilustra um conjunto de objetos (12.1(a)) agrupados de três maneiras diferentes. Visualmente identificamos que uma das divisões dos objetos em dois grupos agrupa os objetos pela forma 12.1(b) e a outra divide os objetos pelo preenchimento 12.1(c). A divisão em quatro grupos considera uma combinação dessas características 12.1(d). Cada uma dessas maneiras de agrupar os objetos é uma estrutura ou um modelo que descreve os dados e poderia ter sido obtida por meio de um algoritmo de agrupamento.

Neste capítulo serão descritos os principais aspetos relacionados à análise de agrupamentos. Na Seção 12.1 serão apresentados as definições básicas sobre *clusters* e os critérios de agrupamento e relação desses elementos com propriedades dos dados que devem ser consideradas ao se fazer análise de agrupamento. Na Seção 12.2, são descritas as etapas envolvidas no processo de análise de agrupamento: preparação dos dados, escolha da medida de proximidade, aplicação de algoritmos de agrupamento, validação e interpretação dos resultados.

12.1 Definições Básicas

Considerando os objetos pontos em um espaço de dimensão d, um *cluster* pode ser visto como uma coleção de objetos próximos ou que satisfazem alguma relação espacial. Embora a ideia do que constitui um *cluster*, grupo de objetos semelhantes ou similares, seja intuitiva, não existe uma definição formal única e precisa para esse conceito. Pelo contrário, existe uma grande variedade de definições na literatura. Isso é resultado da grande diversidade de visões/objetivos dos investigadores de diferentes áreas que utilizam/desenvolvem algoritmos de análise de agrupamentos. Algumas definições comuns para *cluster* são (Barbara, 2000):



(a) Objetos

(b) Agrupamento pela forma (2 *clusters*)

(c) Agrupamento pelo preenchimento (2 *clusters*)

(d) Agrupamento pelo preenchimento e pela forma (4 *clusters*)

Figura 12.1 *Objetos agrupados de diferentes maneiras.*

- *Cluster bem separado*: um *cluster* é um conjunto de pontos tal que qualquer ponto em um determinado *cluster* está mais próximo a cada outro ponto nesse *cluster* do que a qualquer ponto não pertencente a ele.
- *Cluster baseado em centro*: um *cluster* é um conjunto de pontos tal que qualquer ponto em um dado *cluster* está mais próximo (ou é mais similar) ao centro desse *cluster* do que ao centro de qualquer outro *cluster*. O centro de um *cluster* pode ser um centroide, como a média aritmética dos pontos do *cluster*, ou um medoide (isto é, o ponto mais representativo do *cluster*).
- *Cluster contínuo ou encadeado* (vizinho mais próximo ou agrupamento transitivo): um *cluster* é um conjunto de pontos tal que qualquer ponto em um dado *cluster* está mais próximo (ou é mais similar) a um ou mais pontos nesse *cluster* do que a qualquer ponto que não pertence a ele.
- *Cluster baseado em densidade*: um *cluster* é uma região densa de pontos, separada de outras regiões de alta densidade por regiões de baixa densidade.
- *Cluster baseado em similaridade*: um *cluster* é um conjunto de pontos que são similares, enquanto pontos em *clusters* diferentes não são similares.

Cada possível definição de *cluster* resulta em um critério de agrupamento que essencialmente é uma forma de selecionar uma estrutura (ou modelo) para representar os *clusters* que melhor se ajuste a um determinado conjunto de dados (Estivill-Castro, 2002). Cada algoritmo de agrupamento é baseado em um critério de agrupamento e usa uma medida de proximidade e um método de busca para encontrar uma estrutura ótima ou subótima que descreva os dados, de acordo com o critério de agrupamento adotado (Jiang et al., 2004). Os critérios de agrupamento podem ser agrupados em três grandes categorias, de acordo com o tipo de característica que eles apresentam (Handl et al., 2005):

- *Compactação*: a compactação ou homogeneidade de um *cluster* é geralmente associada a uma variação intracluster pequena. Algoritmos que otimizam esse tipo de critérios tendem a ser muito efetivos na descoberta de *clusters* esféricos e/ou bem separados, mas podem falhar para estruturas mais complexas.

- *Encadeamento ou ligação*: encadeamento é um conceito mais local, baseado na ideia de que objetos vizinhos devem compartilhar o mesmo *cluster*. Esse tipo de critério é bastante apropriado para a deteção de *clusters* de formas arbitrárias, mas não é robusto para os casos em que há pouca separação espacial entre os *clusters*.

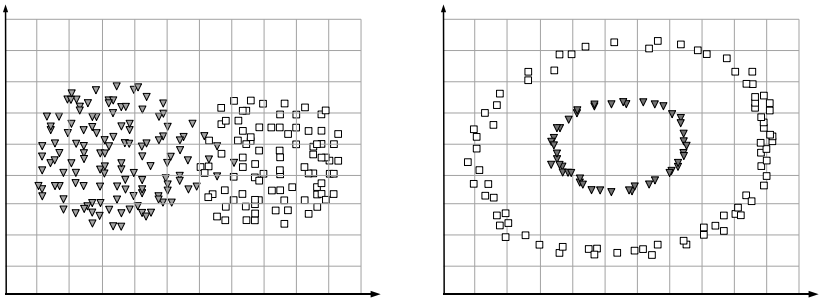
- *Separação espacial*: a separação considera as distâncias entre os *clusters* e, por si só, fornece pouca orientação durante o processo de agrupamento, podendo facilmente levar a soluções triviais. Esse conceito é comumente empregado em associação a outros.

O critério de agrupamento, além de representar o principal aspecto de um algoritmo de agrupamento, também está ligado à maioria das alternativas de avaliação dos resultados. Existe um grande número de algoritmos de agrupamento, cada um buscando *clusters* de acordo com um critério diferente (Law et al., 2004). Algoritmos como o *k*-médias procuram *clusters* compactos, identificando assim mais facilmente *clusters* de forma esférica. Por outro lado, os algoritmos que otimizam um critério baseado no conceito de encadeamento, como o hierárquico de ligação média, captam a densidade local e podem detetar *clusters* de forma arbitrária, mas não são robustos para encontrar *clusters* com uma certa sobreposição. Esses dois algoritmos serão descritos em detalhe no Capítulo 13 e são usados neste capítulo para ilustrar alguns dos aspectos importantes que influenciam na escolha dos algoritmos a serem utilizados.

Mas, independentemente do critério de agrupamento, a análise de agrupamentos compreende diversas etapas que vão além da aplicação de um algoritmo de agrupamento em um conjunto de dados. Essas etapas são discutidas na Seção 12.2.

A Figura 12.2 apresenta dois conjuntos de dados. O conjunto de dados `globular` possui dois *clusters* esféricos que não estão bem separados. Já o conjunto `anel` apresenta dois *clusters* na forma de anel, claramente distintos. Algoritmos cujos critérios se baseiam na compactação, como o *k*-médias (Seção 13.2), conseguem identificar claramente a estrutura do conjunto de dados `globular`, mas falham para o conjunto `anel`. Por outro lado, algoritmos baseados no encadeamento, como o algoritmo hierárquico com ligação simples

(Seção 13.1), identificam facilmente a estrutura do conjunto `anel`, mas têm problemas com a estrutura de `globular`.



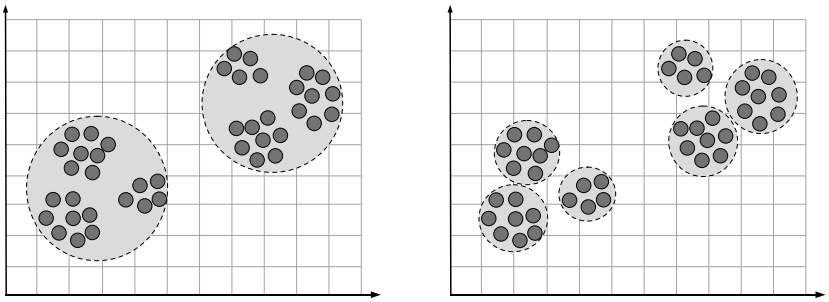
(a) `Globular` (b) `Anel`

Figura 12.2 *Dados com clusters em conformidade com diferentes critérios.*

Um outro aspecto importante sobre algoritmos de agrupamento é que eles podem encontrar estruturas em diferentes níveis de refinamento (números de *clusters* diferentes ou *clusters* de densidades diferentes), dependendo de suas configurações de parâmetros (Jain e Dubes, 1988). A Figura 12.3 ilustra um conjunto de dados que apresenta duas estruturas claramente distintas em dois níveis de refinamento, ambas em conformidade com o conceito de compactação. Na Figura 12.3(a) observa-se uma estrutura com dois *clusters*, enquanto na Figura 12.3(b) observa-se uma estrutura com seis *clusters*.

Essa questão ressalta a importância do ajuste de parâmetros. Neste exemplo em particular, o ajuste seria em relação ao número de *clusters*. Mas como decidir qual deles é o mais adequado? A etapa de validação auxilia tanto na escolha do algoritmo mais apropriado como no ajuste de parâmetros (Handl et al., 2005). No entanto, é importante ter em mente que a maioria das técnicas usadas para validação também é tendenciosa para um critério de agrupamento, assim como os algoritmos. Assim, várias medidas de validação diferentes devem ser aplicadas para selecionar os resultados mais consistentes entre os obtidos com uma variedade de algoritmos de agrupamento usando configurações diferentes de parâmetros (Handl et al., 2005). É importante observar que esse processo exige um conhecimento relativamente profundo de análise de agrupamento, o que os especialistas no domínio de dados, aplicando análise de agrupamento, normalmente não têm.

Um outro aspecto importante a ser considerado é que cada algoritmo procura por uma estrutura homogênea, ou seja, em que todos os *clusters* estejam em conformidade com o mesmo critério. Entretanto, os dados podem apresentar uma estrutura heterogénea, em que cada *cluster* esteja em conformidade com um critério de agrupamento diferente (Law et al., 2004). Por exemplo, o conjunto de dados ilustrado na Figura 12.4 contém três *clusters*, um dos quais em forma de anel, que é facilmente identificado com um critério baseado no encadeamento, como previamente discutido, e dois com formato globular, facilmente identificados com critérios relacionados à compactação. Nesse caso, um algoritmo como



(a) 2 *clusters* (b) 6 *clusters*

Figura 12.3 *Dados com clusters em diferentes níveis de refinamento.*

o *k*-médias identificaria os *clusters* globulares, mas não o que possui forma de anel. Por outro lado, o algoritmo hierárquico com ligação simples provavelmente encontraria apenas o *cluster* em anel, misturando os objetos dos dois *clusters* globulares em um único *cluster*. Isso é evidenciado na Figura 12.5, que ilustra o resultado da aplicação dos algoritmos *k*-médias e hierárquico com ligação simples ao conjunto de dados da Figura 12.4. Na verdade, não existe um único algoritmo de agrupamento capaz de encontrar todos os tipos de agrupamentos que podem estar presentes em um conjunto de dados (Estivill-Castro, 2002; Kleinberg, 2002).

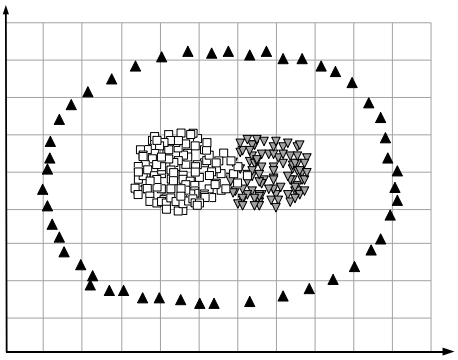
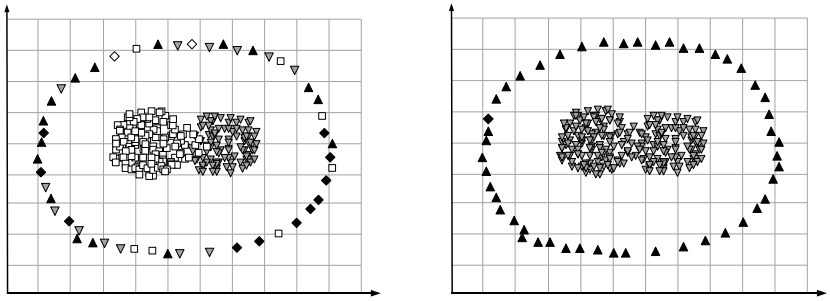


Figura 12.4 *Conjunto de dados com uma estrutura de clusters heterogênea.*

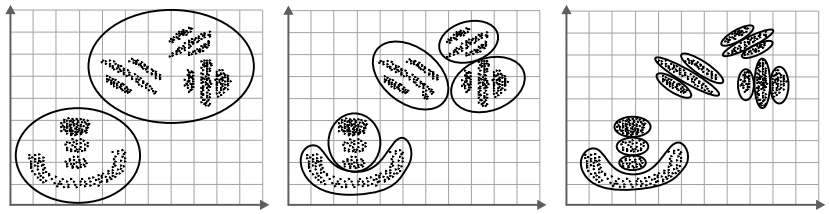
Todos esses aspectos podem estar simultaneamente presentes num conjunto de dados. Por outras palavras, um mesmo conjunto de dados pode ter mais de uma estrutura, cada uma representando uma diferente interpretação dos dados (Handl e Knowles, 2007). Cada uma dessas estruturas pode ser compatível com um critério de agrupamento diferente, estar em um nível de refinamento diferente, ou ainda ser heterogénea. Por exemplo, considere o conjunto de dados mostrado na Figura 12.6. Esse conjunto de dados contém três estruturas



(a) *k*-médias (b) Hierárquico com ligação simples

Figura 12.5 *Resultados dos algoritmos em dados com estrutura heterogênea.*

distintas: E1 (Figura 12.6(a)), E2 (Figura 12.6(b)) e E3 (Figura 12.6(c)). A estrutura mais simples e evidente é E1. Essa estrutura tem dois *clusters* bem separados e de formato esférico. Em princípio, qualquer algoritmo de agrupamento seria capaz de identificá-la. A estrutura E2 é um refinamento de E1, contendo cinco *clusters*. E3, por sua vez, é um refinamento de E2, com 13 *clusters*. E2 e E3 são altamente heterogéneas, em relação à forma de *clusters*. A aplicação de algoritmos tradicionais para explorar um conjunto de dados como esse concentra-se na descoberta de apenas uma única estrutura que melhor se ajusta aos dados. Nesse caso, a estrutura que seria facilmente encontrada por qualquer algoritmo seria a estrutura E1, que pode ser considerada a mais evidente. As demais estruturas dificilmente seriam encontradas com a aplicação de um único algoritmo. Como a quantidade de conhecimento que pode ser obtido com a aplicação usual da análise de agrupamento é limitada, esse aspecto deve ser considerado. Abordagens mais recentes, como as descritas no Capítulo 14, lidam melhor com várias das questões aqui apresentadas.



(a) Estrutura E1 (b) Estrutura E2 (c) Estrutura E3

Figura 12.6 *Conjunto de dados com várias estruturas heterogêneas.*

12.2 Etapas da Análise de Agrupamentos

Podemos dividir o processo de agrupamento nas etapas ilustradas na Figura 12.7 que vão desde a preparação dos dados, até a interpretação dos *clusters* obtidos (Jain et al., 1999; Barbara, 2000). Nessa figura, também são ressaltadas as informações utilizadas e geradas em cada etapa. Cada uma dessas etapas será descrita mais detalhadamente nas próximas seções.

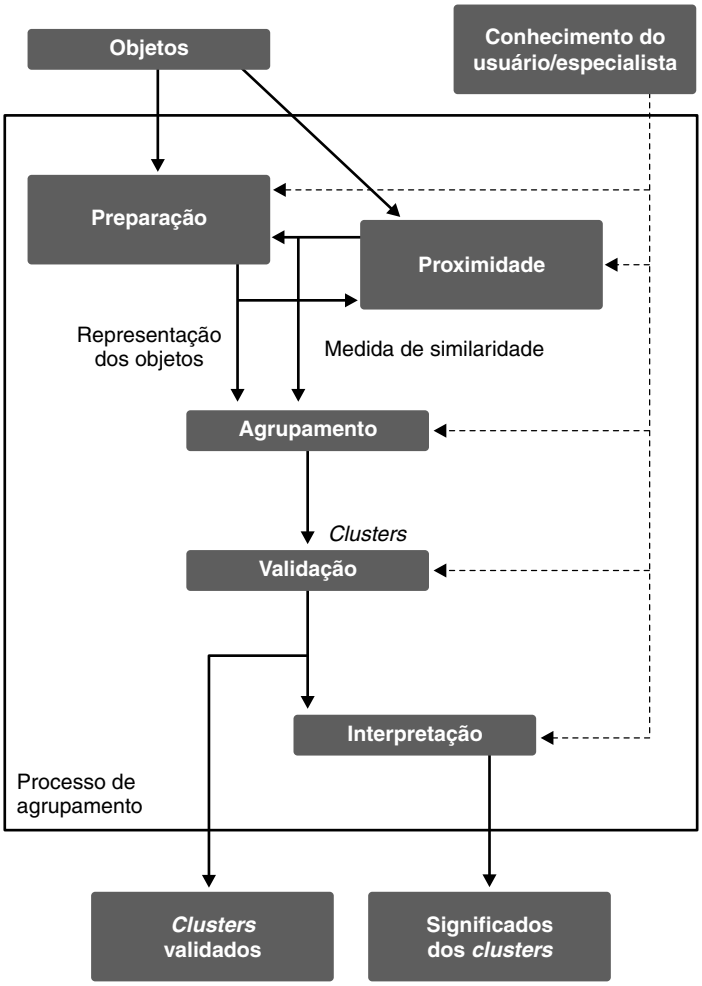


Figura 12.7 *Etapas do processo de agrupamento.*

Essas etapas são importantes para que se possa garantir que os resultados sejam real-

mente significativos e úteis, uma vez que qualquer algoritmo de agrupamento obtém um resultado, quer realmente haja uma estrutura subjacente nos dados, em conformidade com o critério do algoritmo, quer não. Assim, para que se possa aplicar com sucesso cada uma das etapas e obter um resultado significativo da análise de agrupamento, é importante ter em mente as possíveis características dos dados, algumas das quais previamente descritas na Seção 12.1, pois tais características impõem certos desafios na aplicação da análise de agrupamento, que, se não considerados, podem levar a conclusões erróneas sobre a estrutura encontrada nos dados.

Considerando, por exemplo, a preparação dos dados, se eles possuem valores dos atributos em escalas diferentes, dependendo da medida usada, um pode dominar o outro no cálculo da proximidade. Quando isso acontece, pode-se tratar a questão com duas alternativas: normalizar os dados (Seção 3.7) ou escolher uma proximidade que não dependa da magnitude dos atributos. Isso ressalta a interdependência entre as etapas de preparação dos dados e escolha da medida de proximidade a ser aplicada, também evidenciada no fluxo dos dados entre essas duas etapas mostrado na Figura 12.7.

Ainda como pode ser observado na Figura 12.7, apesar de ser uma tarefa não supervisionada, o conhecimento do especialista no domínio dos dados pode ser considerado nas várias etapas. Entretanto, é importante ressaltar que ele é usado de maneira diferente do aprendizado supervisionado. Em agrupamento, o conhecimento do especialista ajuda a guiar as escolhas das técnicas aplicadas em cada etapa, e principalmente na interpretação dos *clusters* obtidos na última etapa do processo.

12.2.1 Preparação dos Dados

A preparação dos dados para o agrupamento engloba vários aspectos relacionados ao seu pré-processamento e à forma de representação apropriada para sua utilização por um algoritmo de agrupamento.

O pré-processamento em análise de agrupamento pode incluir normalizações, conversão de tipos e redução do número de atributos por meio de seleção ou extração de características (Jain et al., 1999). Entretanto, é importante ressaltar que na análise de agrupamento não se tem informações a respeito de possíveis classificações dos dados. Assim, várias das técnicas de seleção ou extração de características descritas no Capítulo 3 não se aplicam, ou precisam ser adaptadas para serem utilizadas em agrupamento.

Quanto à representação, na maioria dos casos os objetos a serem agrupados são representados pela matriz de objetos $\mathbf{X}$, descrita no Capítulo 1. Entretanto, alguns algoritmos de agrupamento exigem uma forma de representação específica, ou algumas vezes apenas a relação de proximidade entre os objetos é conhecida. Assim, além da matriz de objetos, outras duas formas de representação bastante comuns são as matrizes e os grafos de similaridade/dissimilaridade.

A matriz de similaridade/dissimilaridade representa a similaridade ou a dissimilaridade entre cada par de objetos, isto é, cada elemento da matriz $\mathbf{S}_{n\times n}$, s_{ij}, é dado pela distância, $d(\mathbf{x}_i, \mathbf{x}_j)$, ou pela similaridade, $s(\mathbf{x}_i, \mathbf{x}_j)$, entre os objetos $\mathbf{x}_i$ e $\mathbf{x}_j$ (Jain e Dubes, 1988). Já para os grafos, existem diversas alternativas, como o diagrama de Delaunay e árvores

geradoras mínimas, todas elas representando os objetos de acordo com algum aspecto de proximidade e/ou a topologia dos dados. Considerando a representação em grafos, realizar um agrupamento é equivalente a quebrar o grafo em componentes conectados, cada um representando um *cluster*. Muitos dos algoritmos de agrupamento são naturalmente descritos usando uma representação de grafo, conforme será visto no Capítulo 13.

12.2.2 Proximidade

Esta etapa consiste na definição de medidas de proximidade apropriadas ao domínio da aplicação e ao tipo de informação que se deseja extrair dos dados. Existem diferentes níveis de proximidade que podem ser considerados em agrupamento: a proximidade entre objetos, a proximidade entre um objeto e um grupo de objetos e a proximidade entre dois grupos de objetos (He, 1999). Todos os algoritmos de agrupamento consideram a similaridade/dissimilaridade entre objetos, e um mesmo algoritmo pode ser implementado considerando medidas diferentes. Por outro lado, as similaridades entre objetos e grupos e entre dois grupos fazem parte da caraterização de cada algoritmo específico, sendo usadas, geralmente, para decidir a atribuição de um objeto a um *cluster* ou para unir/dividir *clusters*, e dependem do tipo de *cluster* que o algoritmo visa identificar. Esse é o caso das métricas de integração usadas nos algoritmos de agrupamento hierárquicos, que serão descritos na Seção 13.1.

A medida de proximidade entre pares de objetos pode ser uma medida de similaridade ou de dissimilaridade entre dois objetos. A escolha da medida de proximidade deve considerar os tipos e escalas dos atributos que definem os objetos, além das propriedades dos dados que se deseja focar. Por exemplo, deve-se ter em mente se a magnitude relativa dos atributos descrevendo dois objetos é suficiente ou se seu valor absoluto deve ser considerado (Gordon, 1999). As medidas de similaridade/dissimilaridade, em geral, consideram que todos os atributos são igualmente importantes, ou seja, todos contribuem da mesma maneira para o cálculo da medida. Considerando a questão da escala dos valores dos atributos, qual seria o efeito na função distância da representação de um atributo em cm e outro em km, por exemplo? As medidas de distância são diretamente afetadas pela escala dos atributos. Para minimizar esse efeito, os atributos são usualmente normalizados, conforme mencionado anteriormente. Esse problema não ocorre, por exemplo, quando todos os atributos dos objetos assumem apenas valores binários.

Uma das medidas de dissimilaridade mais comum para objetos cujos atributos são todos contínuos é a distância euclidiana, enquanto uma das medidas de similaridade mais usadas é a correlação.

Normalmente, as medidas de similaridade/dissimilaridade satisfazem algumas propriedades. Todas as medidas de distância (que quantificam dissimilaridade entre objetos) satisfazem as propriedades 1, 2 e 3 listadas a seguir. Algumas dessas medidas, chamadas métricas, satisfazem também as propriedades 4 e 5. Para as medidas de distância, quanto menor o valor da medida, mais similares são os objetos.

1. $d(\mathbf{x}_i, \mathbf{x}_i) = 0$ para todo $\mathbf{x}_i$ (Os objetos não são diferentes de si próprios)

2. $d(\mathbf{x}_i, \mathbf{x}_j) = d(\mathbf{x}_j, \mathbf{x}_i)$ (Simetria)
3. $d(\mathbf{x}_i, \mathbf{x}_j) \geq 0$ para todo $\mathbf{x}_i$ e $\mathbf{x}_j$ (Positividade)
4. $d(\mathbf{x}_i, \mathbf{x}_j) = 0$ somente se $\mathbf{x}_i = \mathbf{x}_j$
5. $d(\mathbf{x}_i, \mathbf{x}_l) \leq d(\mathbf{x}_i, \mathbf{x}_j) + d(\mathbf{x}_j, \mathbf{x}_l)$ para todo $\mathbf{x}_i$, $\mathbf{x}_j$ e $\mathbf{x}_l$ (Desigualdade triangular)

Por outro lado, as medidas de similaridade têm uma definição menos rigorosa em relação às propriedades que devem satisfazer. Gordon (1999) apresenta diversas medidas que são mais apropriadas para objetos cujos atributos são todos de um mesmo tipo. Ele classifica as medidas de acordo com o tipo dos atributos para o qual a medida é apropriada.

Medidas para Atributos Quantitativos

Mesmo quando todos os atributos dos objetos são quantitativos, algumas medidas são mais utilizadas quando os valores são contínuos e racionais, e outras, quando os valores são binários.

As medidas mais utilizadas para atributos contínuos e racionais são as medidas de distância baseadas na métrica de Minkowski, como a distância euclidiana, a distância de Manhattan e a distância *supremum*. Quando todos os atributos são binários, é comum a utilização da distância de Manhattan, que neste contexto é chamada de distância de Hamming.

Para dados binários e nominais existem também diversos coeficientes de casamento (*matching*), como o coeficiente de casamento simples e o coeficiente de Jaccard. A seguir são apresentadas as principais medidas de distância e similaridade para valores quantitativos.

A métrica de Minkowski é definida pela Equação 12.1.

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt[p]{\sum_{l=1}^{d} |x_i^l - x_j^l|^p} \tag{12.1}$$

A escolha de diferentes valores para p, com $1 \leq p < \infty$, define variações da métrica. Por isso, essa métrica também é chamada de distância L_p. Os menores valores de p correspondem a estimativas mais robustas (menos sensíveis a *outliers*). As métricas de Minkowski são sensíveis a variações de escala dos atributos, isto é, atributos representados em uma escala maior tendem a dominar os outros. Isso pode ser solucionado pela normalização dos atributos para um intervalo ou variância comum, ou pela aplicação de outros esquemas de ponderação, como os descritos no Capítulo 3 (Jain et al., 1999).

As principais variações da métrica de Minkowski para diferentes valores de p são dadas pelas equações 12.2, 12.3 e 12.4. A Figura 12.8 ilustra graficamente o significado dessas métricas quando os objetos possuem duas dimensões.

- $p = 1$: **Distância de Manhattan** (ou distância bloco-cidade), dada pela Equação 12.2.

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sum_{l=1}^{d} |x_i^l - x_j^l| \tag{12.2}$$

- $p = 2$: **Distância euclidiana**, dada pela Equação 12.3. Essa métrica é a medida de distância mais popular, e uma das mais utilizadas em análise de agrupamentos.

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(\sum_{l=1}^{d} (x_i^l - x_j^l)^2} \tag{12.3}$$

- $p = \infty$: **Distância de Chebyschev ou *supremum***, dada pela Equação 12.4, calcula o máximo da diferença absoluta em coordenadas. Em outras palavras, é a diferença máxima entre quaisquer atributos dos objetos.

$$d(\mathbf{x}_i, \mathbf{x}_j) = \max_{1 \leq l \leq d} |x_i^l - x_j^l| \tag{12.4}$$

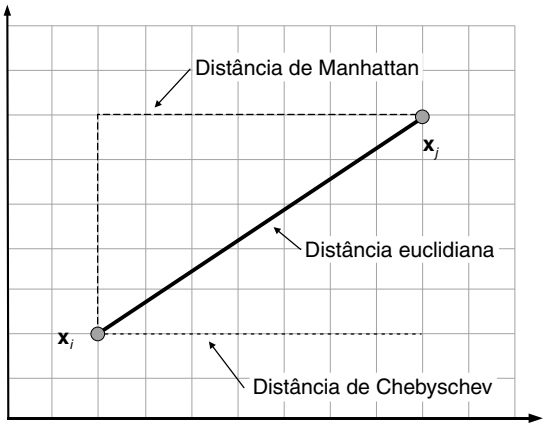


Figura 12.8 *Interpretação das métricas de Minkowski para $d = 2$.*

Duas formas usadas em agrupamento para avaliar a similaridade entre pares de objetos são dadas pelo valor absoluto das medidas de separação angular e correlação de Pearson, que quantificam a correlação entre os objetos $\mathbf{x}_i$ e $\mathbf{x}_j$.

A separação angular, ou simplesmente cosseno, é dada pela Equação 12.5.

$$\text{cosseno}(\mathbf{x}_i, \mathbf{x}_j) = \frac{\sum_{l=1}^{d} x_i^l x_j^l}{\sqrt{(\sum_{l=1}^{d} x_i^{l\,2} \sum_{l=1}^{d} x_j^{l\,2})}} \tag{12.5}$$

Já a correlação de Pearson é dada pela Equação 12.6, em que $\bar{x}_i = \sum_{l=1}^{d} x_i^l / d$. Essa medida é frequentemente descrita como uma medida da forma, no sentido de que é insensível a diferenças na magnitude dos atributos, sendo muito usada para determinar a similaridade entre objetos em áreas como Bioinformática, em que apenas o padrão de variação dos atributos dos objetos é importante.

$$\begin{aligned} \text{pearson}(\mathbf{x}_i, \mathbf{x}_j) &= \frac{\text{covariância}(\mathbf{x}_i, \mathbf{x}_j)}{\text{variância}(\mathbf{x}_i)\text{variância}(\mathbf{x}_j)} \\ &= \frac{\sum_{l=1}^{d} (x_i^l - \bar{x}_i)(x_j^l - \bar{x}_j)}{\sqrt{(\sum_{k=1}^{d} (x_i^l - \bar{x}_i)^2 \sum_{l=1}^{d} (x_j^l - \bar{x}_j)^2)}} \end{aligned} \tag{12.6}$$

Considerando $\mathbf{x}_i$ e $\mathbf{x}_j$ como dois vetores no espaço d-dimensional, a separação angular e a correlação de Pearson podem ser interpretadas geometricamente como o cosseno dos ângulos entre os vetores originais e transformados, respetivamente. A Figura 12.9 ilustra a interpretação quando os objetos são bidimensionais. A separação angular se refere ao cosseno do ângulo α entre os vetores originais, $\mathbf{x}_i$ e $\mathbf{x}_j$. Já a correlação de Pearson corresponde ao cosseno do ângulo β entre os vetores que correspondem aos objetos transformados de maneira a ter média zero e variância 1, ou seja, $\mathbf{x}_i{}' = (\mathbf{x}_i - \bar{x}_i)/\text{variância}(\mathbf{x}_i)$ e $\mathbf{x}_j{}' = (\mathbf{x}_j - \bar{x}_j)/\text{variância}(\mathbf{x}_j)$.

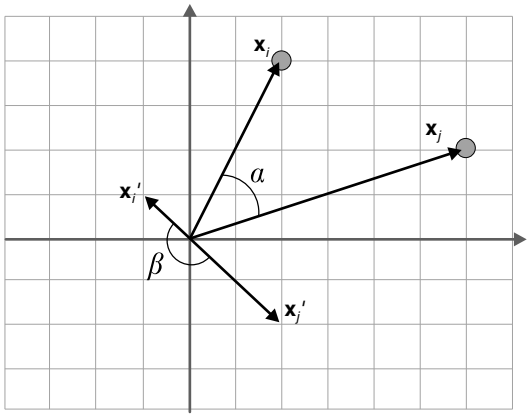


Figura 12.9 *Interpretação geométrica da separação angular e da correlação de Pearson para $d = 2$.*

A correlação de Pearson é sensível a *outliers* e é menos intuitiva do que métricas como a distância euclidiana. Os valores dessas duas medidas variam no intervalo $[-1, 1]$, e sua magnitude indica a força da correlação, enquanto o sinal indica a direção. Assim, valores tanto próximos de 1 quanto próximos de -1 indicam similaridade entre os objetos (ou seja,

os objetos são correlacionados). Enquanto o valor 1 indica que os objetos são diretamente correlacionados, -1 indica que eles são inversamente correlacionados.

Considerando a perspectiva vetorial, uma correlação igual a 1 indica que os vetores representando os objetos são paralelos e apontam no mesmo sentido (ângulo de separação de 0°), uma correlação igual -1 indica vetores paralelos, mas de sentido oposto (ângulo de 180°), e uma correlação igual a 0 indica vetores ortogonais (ângulo de 90°).

Assim, a similaridade entre dois objetos $\mathbf{x}_i$ e $\mathbf{x}_j$, considerando a separação angular, é dada por $s(\mathbf{x}_i, \mathbf{x}_j) = |\text{cosseno}(\mathbf{x}_i, \mathbf{x}_j)|$, e, considerando a correlação de Pearson, é dada por $s(\mathbf{x}_i, \mathbf{x}_j) = |\text{pearson}(\mathbf{x}_i, \mathbf{x}_j)|$. O valor 1 indica a maior similaridade entre os objetos. O valor absoluto deve ser usado para considerar tanto os objetos diretamente quanto os inversamente correlacionados como similares (Jain e Dubes, 1988). Dependendo dos dados e da aplicação, pode ser mais apropriado considerar objetos inversamente relacionados como diferentes, caso em que não é necessário considerar o valor absoluto (dois objetos são similares se possuem valor próximo de 1).

Medidas para Atributos Qualitativos

As medidas para esses tipos de atributo são obtidas pela soma das contribuições individuais de todos os atributos.

Para atributos nominais, uma medida de distância muito utilizada é a distância de Hamming, que é ilustrada pela Equação 12.7, em que $a(\mathbf{x}_i, \mathbf{x}_j)$ é dada pela Equação 12.8. A distância de Hamming conta o número de atributos categóricos com valores diferentes nos dois objetos (Nadler e Smith, 1993). Seu intervalo de variação é $[0, d]$, em que 0 indica a maior similaridade entre os objetos.

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sum_{q=1}^{d} a(\mathbf{x}_i, \mathbf{x}_j) \tag{12.7}$$

$$a(\mathbf{x}_i, \mathbf{x}_j) = \begin{cases} 1 & \text{se } x_i^q \neq x_j^q \\ 0 & \text{caso contrário} \end{cases} \tag{12.8}$$

Medidas para Atributos Heterogéneos

Muitos dos conjuntos de dados utilizados em AM apresentam atributos de tipos diferentes, tanto quantitativos quanto qualitativos. Algumas medidas foram propostas para medir a similaridade entre objetos descritos por atributos de diferentes tipos, por se adequarem a qualquer um dos tipos individualmente. Um exemplo de medida que pode ser aplicada a atributos heterogéneos é o coeficiente geral de similaridade, ilustrado pela Equação 12.9, em que s_{ijk} é a contribuição do k-ésimo atributo para a similaridade e w_{ijk} é 0 ou 1, dependendo de se a comparação para o atributo k é válida ou não. A equação utilizada para s_{ijk} pode variar para cada tipo de atributo.

$$s(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^{d} w_{ijk} s_{ijk} / \sum_{k=1}^{d} w_{ijk} \tag{12.9}$$

Essa medida é adequada para obter a similaridade entre objetos descritos por atributos de diferentes tipos, por se adequar a qualquer um dos tipos individualmente.

Para o caso em que os dados possuem atributos categóricos e contínuos, pode-se utilizar, por exemplo, uma composição das medidas de distância euclidiana e de Hamming. Uma discussão mais abrangente a respeito de medidas de distância heterogéneas pode ser encontrada em Wilson e Martinez (1997).

12.2.3 Análise de Agrupamentos

A etapa central de todo o processo envolvido na análise de agrupamentos é a etapa de agrupamento, em que um ou mais algoritmos de agrupamento são aplicados aos dados para a identificação das possíveis estruturas de *clusters* existentes nos dados. Os diferentes tipos de estruturas que podem ser encontrados por um algoritmo de agrupamento são, por exemplo, partições, hierarquias de partições e partições *fuzzy*. Tradicionalmente, cada algoritmo de agrupamento busca por uma única estrutura de um desses tipos que melhor se ajuste aos dados. Entretanto, várias abordagens recentes, algumas das quais descritas no Capítulo 14, identificam um conjunto de estruturas presentes nos dados. Nesses casos, cada estrutura representa uma visão diferente desses dados. Por exemplo, cada uma dessas estruturas pode estar de acordo com uma definição diferente de *cluster*.

Duas das principais categorizações dos algoritmos de agrupamento estão relacionadas ao tipo de estrutura que pode ser encontrada: agrupamento exclusivo $\times$ não exclusivo e agrupamento hierárquico $\times$ particionamento (Jain e Dubes, 1988).

Um agrupamento exclusivo resulta em uma partição do conjunto de objetos. O conceito de partição (da teoria dos conjuntos) é uma divisão do conjunto de objetos em subconjuntos menores (em análise de agrupamento, são denominados *clusters*), com algumas propriedades. Formalmente, dado o conjunto de dados $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$, uma partição de $\mathbf{X}$ em k *clusters* pode ser definida como: $\pi = \{\mathbf{C}_1, \mathbf{C}_2, ..., \mathbf{C}_k\}$ com $k < n$, tal que (Xu e Wunsch, 2005):

1. $\mathbf{C}_j \neq \emptyset$, $j = 1, ..., k$ (todos os *clusters* contêm pelo menos um objeto)
2. $\bigcup_{j=1}^{k} \mathbf{C}_j = \mathbf{X}$ (todos os objetos pertencem a algum *cluster*)
3. $\mathbf{C}_j \cap \mathbf{C}_l = \emptyset$, $j, l = 1, ..., k$ e $j \neq l$ (cada objeto pertence exclusivamente a um único *cluster*)

O agrupamento que resulta nesse tipo de estrutura também costuma ser chamado de *hard* (um objeto pertence ou não pertence a um dado *cluster*). Um agrupamento não exclusivo, por sua vez, pode associar um mesmo objeto a vários *clusters* ou pode associar a cada objeto uma medida que quantifica a sua proximidade com cada um dos *clusters*.

Nesse último caso se enquadram os algoritmos *fuzzy* e probabilísticos. Assim, os algoritmos de agrupamento *fuzzy*, por exemplo, são uma forma de agrupamento não exclusivo em que cada objeto tem um grau de pertinência a cada um dos *clusters*. A estrutura encontrada por esse tipo de algoritmo é uma partição *fuzzy*. O primeiro caso, em que um objeto pertence a mais de um *cluster*, é apropriado para lidar com dados que possuam mais de uma estrutura. Um exemplo em que objetos podem ser associados a mais de um grupo é o caso de agrupamento de textos por assunto: um mesmo texto pode pertencer a um *cluster* de textos políticos e a um *cluster* de textos desportivos.

Os algoritmos exclusivos podem ainda ser subdivididos em hierárquicos e particionais. O resultado de um algoritmo de particionamento é uma única partição dos dados, enquanto um agrupamento hierárquico resulta em uma sequência aninhada de partições.

Os algoritmos podem ser categorizados ainda de outras maneiras, considerando outros aspectos. Algumas divisões comuns são aglomerativos $\times$ divisivos, seriais $\times$ simultâneos e baseados em teoria dos grafos $\times$ álgebra matricial.

Alguns dos principais algoritmos de agrupamento tradicionais serão apresentados no Capítulo 13, enquanto modelos múltiplos no contexto de análise de agrupamento serão vistos no Capítulo 14.

12.2.4 Validação

Essa etapa avalia o resultado de um agrupamento e deve, de forma objetiva, determinar se os *clusters* são significativos, ou seja, se a solução é representativa para o conjunto de dados analisado. Além de verificar a validade da solução, pode ajudar, por exemplo, na determinação do número apropriado de *clusters* para um conjunto de dados, que em geral não é conhecido previamente.

Como já mencionado, o problema da maioria das abordagens de agrupamento é que elas podem produzir diferentes agrupamentos a partir de um único conjunto de dados (Zeng et al., 2002). Disso surgem algumas questões. Qual resultado é melhor? Quanto se pode confiar nesse resultado? Existe um resultado que seja melhor do que os outros? Se existe, como obtê-lo? Se não, é possível combinar todos os resultados disponíveis para ter um entendimento melhor dos dados?

A maioria dos problemas de agrupamento é NP (não determinístico polinomial), o que significa que eles são intratáveis ou não computáveis em um tempo razoável (Zeng et al., 2002). Como já foi dito, todas as abordagens disponíveis são heurísticas e podem fornecer apenas uma aproximação do resultado ótimo (Zeng et al., 2002). Além disso, apesar do grande número de algoritmos de agrupamento existentes, não existe um algoritmo de agrupamento universal, capaz de revelar toda a variedade de estruturas que podem estar presentes em um conjunto de dados. Como lembra Hartigan (1985), "*diferentes agrupamentos são corretos para diferentes propósitos, assim, não podemos dizer que um agrupamento é melhor*".

A definição da medida de proximidade e do critério de agrupamento utilizados pelos algoritmos geralmente depende implicitamente da imposição de certas hipóteses a respeito

da forma dos *clusters* ou da configuração dos múltiplos *clusters*. Outro aspecto importante é que os dados dificilmente estão estruturados 'idealmente', ou seja, geralmente não formam configurações hiperesféricas, hiperelipsoidais, lineares etc., de modo que cada algoritmo de agrupamento pode apresentar um comportamento superior ao dos demais para uma dada conformação específica dos dados no espaço de atributos.

A análise e a comparação de algoritmos de agrupamento são tarefas complexas e que dependem muito do conhecimento, tanto do domínio da aplicação como das técnicas de agrupamento empregadas.

Uma característica importante, inerente à análise de agrupamento e que torna difíceis a análise do desempenho e a comparação dos algoritmos, é a ausência de uma estrutura ideal, que seja a resposta esperada para o agrupamento. Ou seja, como o agrupamento é uma tarefa não supervisionada, não há uma classificação conhecida dos objetos. É importante ter em mente que, quando se faz análise de agrupamento para de fato explorar um determinado conjunto de dados e extrair conhecimento desse conjunto, nada se sabe sobre sua(s) estrutura(s) subjacente(s). Outras questões que tornam difíceis a análise, a escolha e a comparação de algoritmos são o grande número de algoritmos disponíveis e, segundo Estivill-Castro (2002), a falta de descrição explícita dos princípios indutivos e modelos descritos na literatura.

A análise do desempenho de algoritmos de agrupamento ainda é uma área em aberto. Atualmente, tal análise tem sido feita com base em conjuntos de dados que já têm uma estrutura conhecida, com o objetivo de avaliar e comparar os algoritmos existentes e/ou que estão sendo propostos. Apesar de esse tipo de análise ser útil nesses casos, ela não faz sentido na análise exploratória dos dados.

De acordo com a literatura na área, alguns fatores têm grande influência no desempenho das técnicas de agrupamento: a estrutura dos *clusters* (forma, tamanho, número de *clusters*), a presença de *outliers*, o grau de sobreposição dos *clusters* e a escolha da medida de similaridade (He, 1999).

Segundo Jain e Dubes (1988): "*Uma comparação teórica dos algoritmos de agrupamento não é factível porque os algoritmos de agrupamento são quase impossíveis de modelar de tal forma que os modelos possam ser comparáveis*". Mas alguns critérios são úteis quando se deseja comparar diversos algoritmos de agrupamento. Em primeiro lugar, deve-se ter uma ideia clara do critério de agrupamento no qual se baseia o algoritmo. Também é importante entender como o algoritmo representa os *clusters*, ou seja, como é o modelo gerado. Dado um mesmo contexto (critério de agrupamento e modelo), é possível observar características específicas dos algoritmos relacionadas às suas habilidades, aos resultados que eles podem produzir, aos dados que eles suportam e à necessidade de interação com o utilizador. Entretanto, caso o objetivo seja comparar algoritmos baseados em critérios de tipos diferentes, é preciso saber exatamente o que se deseja comparar e pensar se de fato tal comparação faz sentido, já que o objetivo dos algoritmos pode ser diferente.

As características para se comparar algoritmos em um mesmo contexto são (Halkidi et al., 2001; Jiang et al., 2004):

Relativas ao algoritmo:

- Complexidade do algoritmo;
- Escalabilidade e eficiência para conjuntos de dados grandes;
- Medidas de similaridade que podem ser empregadas pelo algoritmo;
- Robustez relativa a ruídos e *outliers*;
- Se o algoritmo é capaz de lidar com dados de alta dimensionalidade ou encontra *clusters* em subespaços do espaço original;
- Se para cada execução diferente do algoritmo os dados são alocados aos mesmos *clusters* (estabilidade);
- Se o algoritmo é capaz de manipular incrementalmente a adição de novos objetos ou a remoção de objetos antigos.

Relativas ao resultado:

- Forma dos *clusters* que o algoritmo é capaz de encontrar;
- Interpretabilidade dos resultados.

Relativas aos dados:

- Tipos de dados que o algoritmo suporta (contínuos, categóricos, binários);
- Dependência da ordem dos dados.

Relativas à interação com o utilizador:

- Se o algoritmo encontra o número de *clusters* ou se o utilizador deve fornecê-lo;
- Parâmetros requeridos pelo algoritmo e o conhecimento do domínio requerido do utilizador.

Estivill-Castro (2002) discute algumas questões referentes à falta de descrição explícita dos critérios de agrupamento e modelos em muitos dos algoritmos encontrados na literatura, o que pode gerar confusões sobre as propriedades dos algoritmos e tornar difícil a comparação entre eles. Algumas das observações e recomendações de Estivill-Castro são:

- Os algoritmos de agrupamento são categorizados mais de acordo com os modelos do que com os critérios de agrupamento. Então, é preciso atenção na escolha dos algoritmos e no momento da comparação.
- Os pesquisadores devem tentar explicitar matematicamente os modelos e critérios dos algoritmos de agrupamento que estão propondo, facilitando com isso futuras investigações e comparações.

- Os índices de validação de agrupamento são formulações matemáticas diretas de princípios de indução por trás dos critérios de agrupamento. Comparar algoritmos com base nesses índices pode fornecer algumas dicas sobre os contextos nos quais um algoritmo funciona melhor do que outro, mas isso não implica que um algoritmo produza resultados mais válidos que outro. Dois algoritmos aplicados a um conjunto de dados que não possui estrutura irão ambos produzir resultados inválidos.
- Um algoritmo projetado para um universo de modelos não é adequado para conjuntos de dados que têm uma estrutura representável por uma família de modelos radicalmente diferente. Por exemplo, o algoritmo *k*-médias não pode encontrar *clusters* não convexos.

Em Dubes e Jain (1976), um conjunto de critérios de admissibilidade é utilizado para comparar algoritmos de agrupamento. Esses critérios são baseados na maneira como os *clusters* são formados, na estrutura dos dados e na sensibilidade da técnica de agrupamento a mudanças que não afetem a estrutura dos dados. Além desses critérios de admissibilidade, existem algumas questões importantes a serem consideradas, tais como: como os dados deveriam ser normalizados? qual medida de similaridade é apropriada para uma dada situação? como o conhecimento do domínio deve ser utilizado? e como um grande conjunto de dados pode ser agrupado eficientemente?

Assim, é essencial aos utilizadores dos algoritmos de agrupamento ter um bom entendimento da técnica que estão usando, conhecer detalhes do processo de obtenção dos dados, ter algum conhecimento do domínio e ter claramente definido o propósito do agrupamento que deseja obter, para que o agrupamento mais adequado para o problema em questão possa ser obtido.

O conhecimento a respeito dos dados é importante, por exemplo, para determinar as transformações necessárias aos dados antes do agrupamento e para escolher as medidas de similaridade que fazem sentido para esses dados. O conhecimento do domínio e o do propósito do agrupamento permitem determinar as características mais relevantes, os algoritmos de agrupamento mais apropriados e a forma de validação mais adequada.

Toda avaliação dos resultados dos algoritmos de agrupamento, bem como a comparação entre vários algoritmos, deve considerar as questões mencionadas. De maneira prática, a validação de um agrupamento, em geral, é feita com base em índices estatísticos, que julgam, de uma maneira quantitativa e objetiva, o mérito das estruturas encontradas (Jain e Dubes, 1988). Um índice quantifica alguma informação a respeito da qualidade de uma estrutura encontrada por um algoritmo de agrupamento.

Existem três tipos de critérios que empregam os índices estatísticos para investigar a validade de um agrupamento (Jain e Dubes, 1988): critérios relativos, internos e externos. Os índices baseados em critérios relativos comparam diversos agrupamentos para decidir qual deles é o melhor em algum aspecto. Eles podem ser utilizados para comparar algoritmos de agrupamento ou para determinar o valor mais apropriado para um parâmetro de um algoritmo. Índices empregados em tal critério se baseiam apenas nos dados originais. Apesar de sua grande utilidade, é importante mencionar que esses índices são tendenciosos em relação a algum critério de agrupamento.

Os critérios externos e internos são baseados em testes estatísticos e têm um alto custo computacional (Halkidi et al., 2001). O seu objetivo é medir o quanto o resultado obtido confirma uma hipótese pré-especificada. Nesse caso, são utilizados testes de hipóteses para determinar se uma estrutura obtida é apropriada para os dados. Isso é feito testando se o valor do índice utilizado é extraordinariamente grande ou pequeno. Isso requer o estabelecimento de uma população base ou de referência. Um mesmo índice pode ser utilizado em um critério externo e interno, embora as distribuições de referência do índice sejam diferentes (Jain e Dubes, 1988).

O Capítulo 15 contém uma descrição mais aprofundada do processo de validação de agrupamento, bem como de alguns dos índices mais utilizados.

12.2.5 Interpretação

Refere-se ao processo de examinar cada *cluster* com relação a seus objetos para rotulá-los, descrevendo a natureza do *cluster*. A interpretação de *clusters* é mais que apenas uma descrição. Além de ser uma forma de validação dos *clusters* encontrados e da hipótese inicial, de um modo confirmatório, os *clusters* podem permitir avaliações subjetivas que tenham um significado prático. Ou seja, o especialista pode ter interesse em encontrar diferenças semânticas de acordo com os objetos e valores de seus atributos em cada *cluster*.

Nessa etapa, é fundamental o apoio do especialista do domínio, pois é com o conhecimento a respeito dos dados que é possível identificar significados para os *clusters* e possíveis relações entre eles. Além disso, formas de visualizar os *clusters* obtidos são de grande ajuda por fornecer ao especialista do domínio uma maneira fácil e intuitiva de observar os resultados do agrupamento.

12.3 Considerações Finais

Os algoritmos de agrupamento existentes apresentam diferentes formas de explorar e verificar estruturas presentes em um conjunto de dados. Neste capítulo, foram apresentados as principais definições e os principais aspectos relacionados à análise de agrupamentos. Foram destacadas as etapas necessárias para a realização do agrupamento em um conjunto de dados e detalhados alguns aspectos importantes dessas etapas.

Neste Capítulo, foram incluídas a preparação dos dados, a descrição de várias medidas de demelhança que podem ser empregues na análise de agrupamentos, a discussão de aspectos importantes sobre os algoritmos de agrupamentos, a validação de agrupamentos e a análise e comparação de algoritmos de agrupamentos. Por fim, é apresentado como pode ser feita a interpretação dos resultados.

Capítulo 13

Algoritmos de Agrupamentos

Existe uma grande variedade de algoritmos de agrupamento. Cada algoritmo emprega um critério de agrupamento, que impõe uma estrutura aos dados. Se os dados estão em conformidade com as exigências do critério empregado, a estrutura verdadeira de *clusters* pode ser encontrada. Porém, apenas um número pequeno de critérios de agrupamento independentes pode ser entendido sob os pontos de vista matemático e intuitivo. Por isso, muitos dos critérios propostos na literatura são relacionados. Muitas vezes, os mesmos critérios aparecem representados sob diferentes denominações (Jain e Dubes, 1988). Alguns critérios citados por Jain e Dubes (1988) são erro quadrático, ajuste de um modelo de densidade misto (*mixture density model*) aos objetos, estimativa de densidade, conectividade de grafos e vizinhos mais próximos.

Os algoritmos de agrupamento podem ser classificados por meio de diferentes aspectos. Uma das classificações mais frequentes é proposta por Jain et al. (1999) e utilizada por Halkidi et al. (2001), em que os algoritmos são classificados de acordo com o método adotado para definir os *clusters*. Nesse caso, os algoritmos são divididos em algoritmos hierárquicos, particionais, baseados em *grid* e baseados em densidade. Muitos algoritmos se enquadram em mais de uma dessas categorias. Neste capítulo, algumas outras categorias serão também consideradas. Tais categorias não têm necessariamente relação com o método adotado para definir os *clusters*.

Vários dos algoritmos descritos ou mencionados se baseiam na ideia de um objeto representativo que resume as informações contidas no *cluster*. Um elemento representativo bastante usado é o centroide. Seja um *cluster* $\mathbf{C}_k = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{n_k}\}$, com n_k objetos, $\bar{\mathbf{x}}^{(k)}$, o centroide do *cluster* $\mathbf{C}_k$, é dado pela Equação 13.1.

$$\bar{\mathbf{x}}^{(k)} = \frac{1}{n_k} \sum_{\mathbf{x}_i \in \mathbf{C}_k} \mathbf{x}_i \tag{13.1}$$

Neste capítulo serão descritos apenas alguns dos algoritmos mais tradicionais, representantes de cada categoria considerada. Existe uma variedade muito maior de algoritmos, e esse número não para de crescer, com a proposta de novas abordagens. Por razões de espaço, apenas dois dos algoritmos mais conhecidos serão apresentados em detalhes,

representantes das categorias dos algoritmos hierárquicos e dos algoritmos particionais baseados em erro quadrático.

As categorias em que foram divididos os algoritmos são: hierárquicos (Seção 13.1), particionais baseados em erro quadrático (Seção 13.2), baseados em densidade (Seção 13.3), baseados em grafo (Seção 13.4), baseados em redes neurais (Seção 13.5) e baseados em *grid* (Seção 13.6), lembrando que os algoritmos podem se enquadrar em mais de uma categoria. A maioria dos algoritmos nas categorias *grid*, densidade, grafo e redes neuronais (auto-organizáveis) são algoritmos particionais, embora também existam alguns que são hierárquicos.

Alguns outros algoritmos, que não se enquadram nessas categorias, são SVC (do inglês, *Support Vector Clustering*) (Ben-Hur et al., 2001), MSVC (do inglês, *Multiple sphere Support Vector Clustering*) (Chiang e Hao, 2003), SNNC (do inglês, *Shared Nearest Neighbor Clustering*) (Ertöz et al., 2002), *Biclustering* (Cheng e Church, 2000), *Plaid Model* (Lazzeroni e Owen, 2002) e CTWC (do inglês, *Coupled Two-Way Clustering*) (Getz et al., 2003). Esses três últimos algoritmos realizam agrupamento simultâneo dos objetos e dos atributos.

13.1 Algoritmos Hierárquicos

Um algoritmo de agrupamento hierárquico gera, a partir de uma matriz de proximidade, uma sequência de partições aninhadas. O agrupamento hierárquico pode ser dividido em duas abordagens: a aglomerativa, que começa com n *clusters* com um único objeto e forma a sequência de partições agrupando os *clusters* sucessivamente, e a divisiva, que começa com um *cluster* com todos os objetos e forma a sequência dividindo os *clusters* sucessivamente. Um algoritmo hierárquico aglomerativo gera uma sequência de partições de n objetos em k *clusters* em que o nível 1 apresenta n *clusters* de um objeto e o nível n apresenta um *cluster* com todos os objetos. Assim, os dados são agrupados de forma que, se dois objetos são agrupados em algum nível, nos níveis mais altos eles continuam fazendo parte do mesmo grupo, construindo uma hierarquia de *clusters* (Duda et al., 2001).

Os aspectos positivos do agrupamento hierárquico são a sua flexibilidade com respeito ao nível de granularidade, a fácil utilização de qualquer forma de similaridade ou distância e a possibilidade de utilizar qualquer tipo de atributo. Como aspectos negativos, tem-se o critério de terminação vago e o fato de que a maioria dos algoritmos não melhora os *clusters*, uma vez construídos, ou seja, uma vez que um *cluster* foi criado no processo do agrupamento, ele permanece até o final, sem que haja mudanças de seus objetos.

As abordagens mais clássicas de agrupamento hierárquico utilizam métricas de integração (*linkage metrics*). Essas métricas são medidas de distância entre *clusters*. Esse tipo de agrupamento resulta em *clusters* de formas convexas próprias e, em geral, possuem complexidade $O(n^2)$. Existem ainda várias implementações de algoritmos hierárquicos que privilegiam funcionalidades mais específicas, como, por exemplo, uma melhor manipulação de *outliers*, a obtenção de *clusters* de diferentes formas e tamanhos e a escalabilidade para números elevados de exemplos ou atributos.

Uma descrição geral de algoritmos hierárquicos é apresentada a seguir. Outros algoritmos hierárquicos, não descritos neste livro são os algoritmos BIRCH (do inglês, *Balanced Iterative Reducing and Clustering using Hierarchies*) (Zhang et al., 1996), CURE (do inglês, *Clustering Using REpresentatives*) (Guha et al., 1998), CHAMELEON (Karypis et al., 1999), OPTICS (do inglês, *Ordering Points To Identify the Clustering Structure*) (Ankerst et al., 1999) e ROCK (do inglês, *RObust Clustering using linKs*) (Guha et al., 2000).

Os algoritmos baseados nas métricas de integração podem ser divisivos ou aglomerativos e funcionam da seguinte maneira: inicializam um agrupamento como um conjunto de *clusters* de um elemento (aglomerativo) ou um único *cluster* com todos os elementos (divisivo) e iterativamente unem ou dividem os *clusters* mais apropriados, até que seja atingido um critério de parada.

A Figura 13.1 ilustra o processo de união e divisão dos *clusters* nos algoritmos aglomerativos e divisivos, respectivamente. Para agrupar/dividir os *clusters*, cada algoritmo considera uma das alternativas de distância/similaridade entre *clusters* dadas pelas métricas de integração. Cada métrica tem influência direta no funcionamento do algoritmo. Esses algoritmos não têm uma função objetivo global. São baseados em decisões locais. Para as técnicas aglomerativas, o critério de agrupamento é tipicamente agrupar os pares de *clusters* mais próximos, de acordo com a métrica de integração utilizada (Barbara, 2000). Para as técnicas divisivas, o critério é, geralmente, dividir os grupos que possam gerar partições mais diferentes.

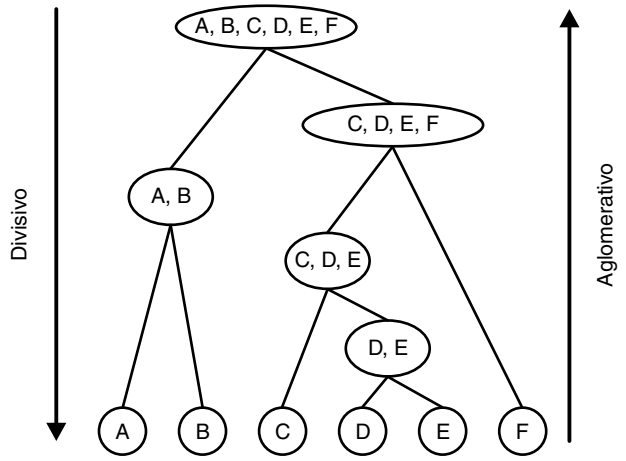


Figura 13.1 *Funcionamento dos algoritmos hierárquicos aglomerativos e divisivos.*

Dados dois *clusters* $\mathbf{C}_1 = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{n_1}\}$ e $\mathbf{C}_2 = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{n_2}\}$, com os respectivos centroides $\bar{\mathbf{x}}^{(1)}$ e $\bar{\mathbf{x}}^{(2)}$, para quantificar distâncias entre os *clusters*, podem ser utilizadas distâncias como euclidiana ou de Manhattan entre os centroides dos *clusters*, $d(\bar{\mathbf{x}}^{(1)}, \bar{\mathbf{x}}^{(2)})$, ou ainda quantificar essas distâncias de acordo com as distâncias entre pares de objetos

dos dois *clusters*. A ideia, nesse caso, é calcular a distância entre todos os possíveis pares de objetos $(\mathbf{x}_i, \mathbf{x}_j)$, sendo um deles do primeiro *cluster*, $\mathbf{C}_1$, e outro do segundo, $\mathbf{C}_2$, e, em seguida, aplicar uma operação específica para traduzir essas distâncias na distância entre os *clusters*. Essa operação pode ser, por exemplo, o mínimo, a média ou o máximo dos valores das distâncias entre os objetos. Cada uma das distâncias entre *clusters* estabelecidas dessa maneira é uma métrica de integração.

A distância entre *clusters*, ilustrada pela Equação 13.2, é dada pela distância entre os objetos dos dois *clusters* que estão mais próximos, ou seja, é a distância mínima entre quaisquer dois objetos, um de cada *cluster*. Essa distância é empregada pelo algoritmo hierárquico com ligação mínima (*single-link*). A Equação 13.3 apresenta a distância média entre os objetos dos dois *clusters*. Essa distância é empregada pelo algoritmo hierárquico com ligação média (*average-link*). Por sua vez, a Equação 13.4 descreve a distância entre os objetos mais distantes dos dois *clusters*. Essa distância é empregada pelo algoritmo hierárquico com ligação máxima (*complete-link*).

$$d(\mathbf{C}_1, \mathbf{C}_2) = \min_{\substack{\mathbf{x}_i \in \mathbf{C}_1, \\ \mathbf{x}_j \in \mathbf{C}_2}} d(\mathbf{x}_i, \mathbf{x}_j) \tag{13.2}$$

$$d(\mathbf{C}_1, \mathbf{C}_2) = \frac{1}{n_1 n_2} \sum_{\substack{\mathbf{x}_i \in \mathbf{C}_1, \\ \mathbf{x}_j \in \mathbf{C}_2}} d(\mathbf{x}_i, \mathbf{x}_j) \tag{13.3}$$

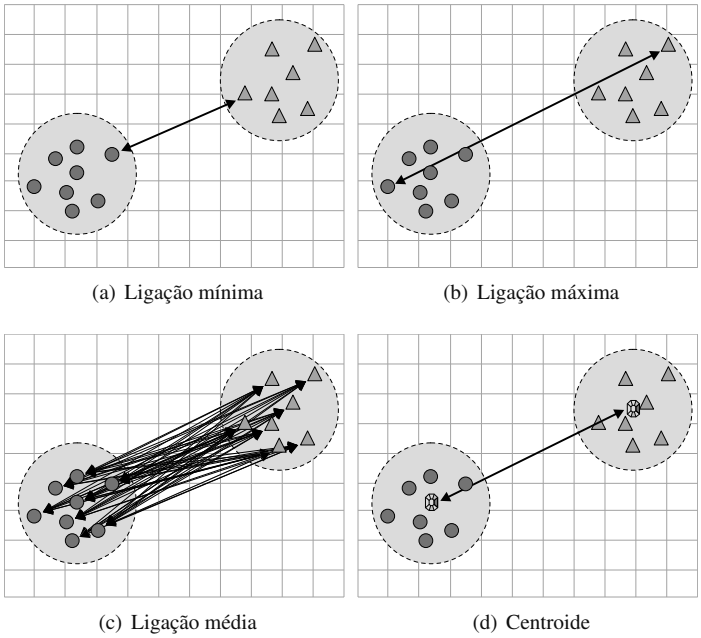
$$d(\mathbf{C}_1, \mathbf{C}_2) = \max_{\substack{\mathbf{x}_i \in \mathbf{C}_1, \\ \mathbf{x}_j \in \mathbf{C}_2}} d(\mathbf{x}_i, \mathbf{x}_j) \tag{13.4}$$

A Figura 13.2 ilustra graficamente o significado das métricas de ligação mínima, máxima e média, bem como a distância euclidiana entre os centroides, entre dois *clusters*.

A métrica utilizada pelo algoritmo com ligação simples é indicada para manipular formas não elípticas, mas é bastante sensível a ruídos e *outliers*. Em geral, essa métrica favorece *clusters* finos e alongados. Já a métrica do algoritmo hierárquico com ligação máxima é menos suscetível a ruídos e *outliers*, mas tende a quebrar *clusters* grandes e tem problemas com formas convexas (Barbara, 2000). Em geral favorece a obtenção de *clusters* esféricos.

Esses algoritmos de agrupamento hierárquico não lidam bem com ruídos e *outliers* e dependem da ordem de entrada dos dados. Por outro lado, não requerem a especificação prévia do número de *clusters*, e seus resultados correspondem a taxonomias, muito comuns nas áreas biológicas.

Quando algoritmos de agrupamento hierárquico são utilizados, as soluções são tipicamente representadas por um dendrograma, que consiste em uma árvore binária que representa uma hierarquia de partições (Barbara, 2000). Essencialmente, um dendrograma é formado por camadas de nós, cada uma representando um *cluster*. Linhas conectam nós representando *clusters* aninhados. O corte de um dendrograma na horizontal representa uma partição.



(a) Ligação mínima (b) Ligação máxima

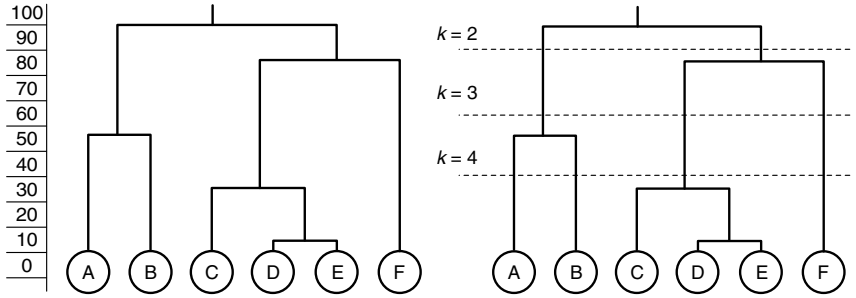
(c) Ligação média (d) Centroide

Figura 13.2 *Distâncias entre clusters.*

A Figura 13.3(a) ilustra um dendrograma com a hierarquia de agrupamento para o conjunto de objetos $\{A,B,C,D,E,F\}$. A altura das ramificações, em geral, é proporcional à distância dos *clusters* que foram agrupados/divididos. Cortes em cada nível do dendrograma representam diferentes partições dos dados, com diferentes números de *clusters*. A Figura 13.3(b) ilustra cortes no dendrograma representando as partições:

- $\{\{A,B\},\{C,D,E,F\}\}$, com $k = 2$,
- $\{\{A,B\},\{C,D,E\},\{F\}\}$, com $k = 3$ e
- $\{\{A\},\{B\},\{C,D,E\},\{F\}\}$, com $k = 4$.

O Algoritmo 13.1 ilustra o funcionamento dos algoritmos de agrupamento hierárquico aglomerativos baseados em métricas de integração que são mais comumente utilizados. Com algumas modificações, obtém-se o algoritmo divisivo.



(a) Hierarquia de agrupamento (b) Cortes

Figura 13.3 *Exemplo de dendrograma.*

Algoritmo 13.1 Algoritmo hierárquico aglomerativo

Entrada: Uma matriz de dissimilaridade entre pares de objetos $\mathbf{S}_{n\times n}$
Saída: Uma hierarquia de partições
1 Alocar cada objeto em um *cluster*
2 **enquanto** *há clusters para agrupar* **faça**
3 Calcular a matriz de distância entre os pares de *clusters* disponíveis, utilizando uma métrica de integração
4 Combinar o par de *clusters* $\mathbf{C}_i$ e $\mathbf{C}_j$ mais próximo, gerando um único *cluster* $\mathbf{C}_{ij}$
5 **fim**

13.2 Algoritmos Particionais Baseados em Erro Quadrático

Esses algoritmos otimizam o critério de agrupamento utilizando uma técnica iterativa. O primeiro passo consiste na criação de uma partição inicial. Em seguida, os objetos são movidos de um *cluster* para outro com o objetivo de melhorar o valor do critério de agrupamento. Esses algoritmos são computacionalmente eficientes, porém podem convergir para um ótimo local.

O critério de agrupamento utilizado por esses algoritmos particionais é o erro quadrático, que garante a propriedade de compactação dos *clusters*. O objetivo desses algoritmos é obter uma partição que minimiza o erro quadrático para um número fixo de *clusters*. Minimizar o erro quadrático, ou a variação dentro de um *cluster*, é equivalente a maximizar a variação entre *clusters* (Jain e Dubes, 1988). O erro quadrático para um agrupamento contendo *k clusters* é a soma da variação dentro dos *clusters*, ilustrada pela Equação 13.5, em que $\bar{\mathbf{x}}^{(j)}$ é o centroide do *cluster* $\mathbf{C}_j$, como definido na Equação 13.1 e $d(\mathbf{x}_i, \bar{\mathbf{x}}^{(j)})$ é a

distância euclidiana entre um objeto $\mathbf{x}_i$ e o centroide $\bar{\mathbf{x}}^{(j)}$. Em alguns casos, a distância de Mahalanobis também pode ser utilizada para definir o erro quadrático. O centroide pode ser a média, como definido na Equação 13.1, ou também a mediana de um grupo de objetos.

$$E = \sum_{j=1}^{k} \sum_{\mathbf{x}_i \in \mathbf{C}_j} d(\mathbf{x}_i, \bar{\mathbf{x}}^{(j)})^2 \tag{13.5}$$

O objetivo desse tipo de agrupamento é encontrar uma partição contendo *k clusters* que minimiza E, para um valor de k fixo. A partição resultante também é chamada de partição de variância mínima. Minimizar essa função é um problema NP-hard (Jain, 2010). Assim, os algoritmos dessa categoria são gulosos e podem convergir para ótimos locais como já mencionado.

O principal representante dessa categoria é o algoritmo *k*-médias, que é um dos algoritmos de agrupamento mais simples e conhecidos. Outros algoritmos nessa categoria são: PAM (do inglês, *Partitioning Around Medoids*) (Kaufman e Rousseeuw, 1990), CLARA (do inglês, *Clustering Large Applications*) (Kaufman e Rousseeuw, 1990) e CLARANS (do inglês, *Clustering Large Applications based on the Randomized Search*) (Ng e Han, 1994).

O algoritmo *k*-médias particiona o conjunto de dados em *k clusters*, em que o valor de *k* é fornecido pelo utilizador (Duda et al., 2001). Esses *clusters* são formados de acordo com alguma medida de similaridade. O algoritmo *k*-médias utiliza uma técnica de realocação iterativa, que pode convergir para um ótimo local. Existem várias versões do algoritmo, cada uma solucionando uma deficiência do algoritmo original. A versão tradicional do algoritmo encontra *clusters* compactos e de formato esférico. Mas existem versões, por exemplo, em que a distância de Mahalanobis pode ser utilizada para encontrar *clusters* hiperelipsoidais. Berkhin (2002) e Jain et al. (1999) discutem brevemente algumas dessas versões.

O Algoritmo 13.2 ilustra o funcionamento do algoritmo *k*-médias básico. Esse algoritmo começa inicializando um conjunto de *k* centroides para os *clusters*. Essa inicialização pode ser feita de diferentes maneiras. Uma bastante comum é a escolha aleatória de *k* objetos do conjunto de dados para representar os centroides iniciais. Em seguida, cada ponto do conjunto de dados é associado ao *cluster* com o centroide mais próximo. Depois disso, os centroides são recalculados. O processo é repetido até que os centroides não sejam mais alterados.

Como já mencionado, o critério de agrupamento do *k*-médias é o erro quadrático definido pela Equação 13.5. Dito de outra maneira, o *k*-médias minimiza a distância entre cada objeto e o centroide do *cluster* ao qual ele pertence (Halkidi et al., 2001). Essa função objetivo é minimizada por *clusters* de formato globular (hiperesférico) do mesmo tamanho (raio) ou *clusters* bem separados.

A complexidade do algoritmo *k*-médias é $O(n)$, uma vez que o número de iterações é tipicamente pequeno e $k << n$ (Barbara, 2000). Além disso, também é considerado que $d << n$.

Algoritmo 13.2 Algoritmo *k*-médias

Entrada: Um conjunto de dados $\mathbf{X}_{n\times d}$
Número de *clusters* k
Saída: Uma partição de $\mathbf{X}$ em k *clusters*
1 Escolher aleatoriamente k valores para centroides dos *clusters*
2 **repita**
3 **para cada** *objeto* $\mathbf{x}_i \in \mathbf{X}$ *e cluster* $\mathbf{C}_j, j = 1,\ldots k$ **faça**
4 Calcular a distância entre $\mathbf{x}_i$ e o centroide do *cluster* $\bar{\mathbf{x}}^{(j)}$: $d(\mathbf{x}_i, \bar{\mathbf{x}}^{(j)})$, utilizando uma medida de distância
5 **fim**
6 **para cada** *objeto* $\mathbf{x}_i$ **faça**
7 Associar $\mathbf{x}_i$ ao *cluster* com centroide mais próximo
8 **fim**
9 **para cada** *cluster* $\mathbf{C}_j, j = 1,\ldots k$ **faça**
10 Recalcular o centroide
11 **fim**
12 **até** *não haver mais alteração na associação dos objetos aos clusters*;

O algoritmo é sensível à escolha inicial dos centroides e da sua forma de atualização. Dependendo da escolha dos centroides, o algoritmo pode convergir para um ótimo local. Além disso, dependendo da distância empregada, é restrito a objetos em espaços euclidianos. Os *clusters* encontrados por esse algoritmo são em geral desbalanceados.

13.3 Algoritmos Baseados em Densidade

Esses algoritmos assumem que os *clusters* são regiões de alta densidade de objetos, separadas por regiões com baixa densidade, no espaço de objetos. Um *cluster* definido como um componente denso conectado cresce em qualquer direção dada pela densidade (Berkhin, 2002). Portanto, os algoritmos baseados em densidade são capazes de obter *clusters* de formas arbitrárias.

Além do algoritmo DENCLUE (do inglês, *DENsity-based CLUstEring*), descrito a seguir, também podem ser mencionados os algoritmos DBSCAN (do inglês, *Density-Based Spatial Clustering of Applications with Noise*) (Ester et al., 1996) e *Wave-cluster* (também baseado em *grid*) (Sheikholeslami et al., 1998) como baseados em densidade.

O algoritmo DENCLUE modela a densidade global de um conjunto de pontos como a soma de funções "influência" associadas a cada *cluster* (Hinneburg e Keim, 1998). A função de densidade global resultante tem picos locais que podem ser utilizados para definir *clusters*. Para cada ponto, encontra-se o pico mais próximo associado a ele. O conjunto de todos os pontos associados a um pico particular (atrator de densidade local) se torna um

cluster. Entretanto, se a densidade em um pico local é muito baixa, os pontos associados a esse *cluster* são considerados ruídos e descartados. Se dois picos locais são conectados por um caminho de pontos e a densidade de cada um desses pontos no caminho está acima de um *threshold* de densidade mínimo ξ, então os *clusters* associados a esses picos são unidos.

O cálculo da função de densidade global requer a soma das funções influência de todos os pontos. Porém, a maioria dos pontos não contribui para a função de densidade global. Por isso, o algoritmo DENCLUE utiliza uma função de densidade local, que considera apenas os pontos que de fato contribuem para a função de densidade global.

O DENCLUE é baseado em estimação de densidade por *kernel* (*kernel density estimation*), que tem como objetivo descrever a distribuição dos dados por uma função global. A contribuição de cada ponto para a função de densidade global é expressa por uma função influência ou *kernel*. A função global é a soma das funções associadas a cada ponto.

Para a definição da função influência, é utilizada uma função de distância qualquer, que seja reflexiva e simétrica. Hinneburg e Keim (1998) utilizam a distância euclidiana. Tipicamente, a função influência decresce com o aumento da distância ao ponto. Uma função *kernel* utilizada frequentemente é a função gaussiana (Equação 13.6), em que σ é um parâmetro que governa o quão rapidamente a influência do ponto diminui.

$$G(x) = \exp^{\frac{-d(\mathbf{x}_i,\mathbf{x}_j)^2}{2\sigma^2}} \tag{13.6}$$

Esse algoritmo possui dois passos: pré-agrupamento e agrupamento. Na etapa de pré-agrupamento, é construído um mapa da porção relevante do espaço de dados para acelerar o cálculo da função de densidade. Na etapa de agrupamento, o algoritmo identifica os atratores de densidade e os pontos atraídos correspondentes.

O mapa é criado dividindo o (hiper-)retângulo de limite mínimo (*minimum bounding hyper-rectangle*) dos dados em hipercubos de dimensão d com aresta de tamanho 2σ. Os hipercubos que contêm pontos de dados são determinados. Em seguida, eles são numerados de acordo com sua posição em relação a uma origem particular. Dessa forma, os hipercubos são mapeados em chaves unidimensionais. As chaves dos cubos povoados são armazenadas em uma árvore de busca para permitir, posteriormente, acesso eficiente.

Para o agrupamento são considerados apenas os cubos mais povoados e os cubos conectados a eles. Para cada ponto $\mathbf{x}_i$ é calculada a função de densidade local considerando apenas os pontos de *clusters* conectados ao *cluster* que contém $\mathbf{x}_i$ e que têm centroides a uma distância de $k\sigma$ de $\mathbf{x}_i$. Cada ponto $\mathbf{x}_j$ no caminho de $\mathbf{x}_i$ ao seu atrator de densidade é associado ao mesmo *cluster* de $\mathbf{x}_i$ se a distância entre $\mathbf{x}_i$ e $\mathbf{x}_j$ for menor ou igual a $\sigma/2$. Os *clusters* associados a um atrator de densidade cuja densidade seja menor que ξ são descartados. Os atratores de densidade ligados por um caminho de pontos de densidade maior que ξ são unidos.

Esse algoritmo tem uma fundamentação sólida e apresenta uma descrição matemática compacta dos *clusters*. Ele pode também ser classificado como baseado em *grid*. Uma deficiência do DENCLUE é que ele é muito sensível à escolha dos valores dos parâme-

tros, que são difíceis de determinar. Dependendo da escolha desses valores, ele pode se comportar como os algoritmos DBSCAN, *k*-médias ou hierárquico.

13.4 Algoritmos Baseados em Grafo

Para a obtenção de um agrupamento por meio de técnicas baseadas na teoria dos grafos, os dados são representados em um grafo de proximidade. No caso mais simples, cada nó representa um objeto e é conectado com os $n-1$ nós restantes, resultando em um grafo completo. Os pesos das arestas representam a similaridade ou a distância entre os objetos. Os métodos de agrupamento decompõem os grafos em componentes conectados pela remoção de arestas inconsistentes, ou, ainda, inserem/removem aresta de acordo com algum critério. Cada um desses componentes resultantes do processo de agrupamento vai representar um *cluster*.

Os algoritmos HSC (do inglês, *Highly Connected Subgraph*) (Hartuv e Shamir, 2000) e CLICK (do inglês, *Cluster Identification via Connectivity Kernels*) (Sharan e Shamir, 2000) são exemplos de algoritmos que utilizam uma abordagem baseada na teoria dos grafos para agrupar os dados. Os dados de entrada são representados como um grafo de similaridade.

Os algoritmos HSC e CLICK separam recursivamente o conjunto atual de elementos em dois subconjuntos. Antes de uma divisão, eles consideram o subgrafo induzido pelo subconjunto atual de elementos. Se o subgrafo satisfaz um critério de parada, então ele é declarado um *kernel*. De outra forma, um corte de peso mínimo é computado naquele subgrafo, e o conjunto é dividido nos dois subconjuntos separados por aquele corte. A saída gerada é uma lista de *kernels* que serve como base para definir os eventuais *clusters*.

A diferença entre os dois algoritmos, HSC e CLICK, está no grafo de similaridade que eles constroem, no critério de parada e no pós-processamento dos *kernels* (Shamir e Sharan, 2002). O algoritmo CLICK é mais recente e, atualmente, mais utilizado.

13.5 Algoritmos Baseados em Redes Neuronais

Como apresentado no Capítulo 7, as redes neuronais são sistemas paralelos distribuídos compostos de unidades de processamento simples que computam determinadas funções matemáticas, sendo dispostas em uma ou mais camadas e interligadas por um grande número de conexões. Existem diversos modelos de RNs voltados ao aprendizado não supervisionado. Dentre os algoritmos de agrupamento baseados em redes neuronais encontram-se os algoritmos SOM (do inglês, *Self Organizing Map*) (Kohonen, 2001), GCS (do inglês, *Growing Cell Structures*) (Fritzke, 1994), SOTA (do inglês, *Self-Organizing Tree Algorithm*) (Herrero et al., 2001), HGSOT (do inglês, *Hierarchically Growing Self-Organizing Tree*) (Luo et al., 2003) e DGSOT (do inglês, *Dynamically Growing Self-Organizing Tree*)(Luo et al., 2004).

O algoritmo SOM (*Self-Organizing Map*) (Kohonen, 2001) é uma rede neural artificial não supervisionada, frequentemente utilizada em tarefas de agrupamento e visualização de dados. É o algoritmo mais tradicional dessa categoria, e será brevemente descrito a seguir.

Nas redes SOM, os neurónios são organizados em um reticulado uni ou bidimensional. Cada neurónio no reticulado está conectado a todas as entradas da rede. Essa rede geralmente utiliza uma única camada computacional. A cada objeto de entrada apresentado à rede, os neurónios computam seus valores de ativação, ativando uma região diferente do reticulado. Para cada objeto de entrada, os neurónios de saída da rede competem entre si para terem seus pesos ajustados. O neurónio com maior valor de ativação é o vencedor da competição. Em seguida, é determinada a localização espacial de uma vizinhança topológica de neurónios ativados, centrada no neurónio vencedor. O próximo passo consiste em uma adaptação dos pesos. Os ajustes dos pesos são tais que o neurónio vencedor e seus vizinhos que tiverem os pesos ajustados aumentam seu valor de ativação para futuros objetos de entrada que sejam similares ao objeto atual. Assim, durante a execução do algoritmo, os vetores de entrada direcionam o movimento dos vetores de peso, promovendo uma organização topológica dos neurónios da rede. Ainda durante o treinamento, a região de vizinhança dos neurónios vencedores é gradativamente reduzida.

O objetivo da rede SOM é encontrar um conjunto de vetores de referência e associar cada objeto do conjunto de dados ao vetor referência mais próximo. O algoritmo depende da inicialização dos vetores de referência. O resultado consiste em um conjunto de vetores de referência que definem implicitamente os *clusters*.

13.6 Algoritmos Baseados em reticulados

Este grupo de algoritmos define um reticulado (*grid*) para o espaço de dados e realiza todas as operações nesse reticulado. Em termos gerais, essa abordagem é muito eficiente para grandes conjuntos de dados, é capaz de encontrar *clusters* de formas arbitrárias e lida bem com *outliers*.

Alguns dos algoritmos que se enquadram nessa categoria são: CLIQUE (do inglês, *Clustering In QUEst*)(Agrawal et al., 1998), descrito a seguir, MAFIA (do inglês, *Merging of Adaptative Finite Intervals*) (Nagesh et al., 2001a,b), OptiGrid (do inglês, *Optimal Grid clustering*) (Hinneburg e Keim, 1999) e STING (do inglês, *Statistical Information Grid-based method*) (Wang et al., 1997).

Alguns desses algoritmos foram projetados com uma funcionalidade em mente. Os algoritmos CLIQUE e MAFIA, por exemplo, são técnicas desenhadas especificamente para trabalhar com dados de alta dimensão.

O algoritmo CLIQUE (Agrawal et al., 1998) encontra *clusters* em subespaços dos dados. O algoritmo é baseado em *grid* e densidade. Identifica *clusters* densos em subespaços de dimensionalidade máxima. Os *clusters* gerados são descritos na forma de expressões FND (Forma Normal Disjuntiva), que são minimizadas para facilitar a compreensão. Os resultados do agrupamento não dependem da ordem de apresentação dos objetos. O algo-

ritmo CLIQUE também não supõe nenhuma forma matemática específica de distribuição dos dados. Além disso, o algoritmo é tolerante a valores ausentes nos dados de entrada.

Como algoritmo baseado em densidade, CLIQUE utiliza a definição de um *cluster* como uma região com maior densidade de pontos do que as regiões à sua volta. O problema tratado pelo algoritmo é identificar automaticamente projeções dos dados de entrada usando subconjunto dos atributos. Identifica *clusters* como as projeções que geram regiões de alta densidade.

Este algoritmo encontra regiões de alta densidade por meio do particionamento do espaço de dados em células (hiper-retângulos) e da localização das células densas. Um *cluster* corresponde à união de todas as células de alta densidade adjacentes. CLIQUE é baseado na seguinte propriedade dos *clusters*: uma vez que um *cluster* representa uma região densa em algum subespaço do espaço de atributos, haverá áreas densas correspondentes ao *cluster* em todos os subespaços de menor dimensão. Com base nisso, o algoritmo inicia encontrando todas as áreas densas em espaços unidimensionais correspondentes a cada atributo. Em seguida, o algoritmo gera um conjunto de células bidimensionais que podem ser densas. Isso é realizado a partir das células unidimensionais densas. Cada célula bidimensional deve ser associada a um par de células unidimensionais densas. Da mesma forma são construídas células densas para as demais dimensões. Os *clusters* são obtidos encontrando um conjunto maximal de unidades densas em um determinado número de dimensões.

Efetivamente, o algoritmo CLIQUE resulta na seleção de atributos (seleciona vários subespaços) e produz uma visão dos dados em diferentes perspectivas.

13.7 Considerações Finais

Neste capítulo foram descritos de forma sucinta algoritmos representantes das principais categorias de algoritmos de agrupamento encontrados na literatura de ECD. As categorias consideradas foram a dos algoritmos hierárquicos (Seção 13.1), particionais baseados em erro quadrático (Seção 13.2), baseados em densidade (Seção 13.3), baseados em grafo (Seção 13.4), baseados em redes neurais (Seção 13.5) e baseados em *grid* (Seção 13.6). Mais especificamente, foram brevemente introduzidos os algoritmos DENCLUE, baseado em densidade, HSC e CLICK, baseados em grafo, SOM, baseado em redes neurais e CLIQUE, baseado em reticulados.

Além disso, foram apresentados maiores detalhes de dois dos algoritmos mais tradicionais e amplamente empregados em análise de agrupamento: os algoritmos hierárquicos aglomerativos baseados em métricas de integração e o *k*-médias. Nessas descrições, foram apresentados os procedimentos utilizados por esses algoritmos para encontrar *clusters* em conjuntos de dados.

A ideia deste capítulo foi apenas introduzir alguns dos algoritmos mais tradicionais. Vários outros algoritmos foram mencionados, mas este texto não tem a intenção de fornecer uma lista exaustiva dos algoritmos existentes. Como já mencionado, a quantidade de algoritmos existentes é bastante extensa, e novos algoritmos estão constantemente sendo

propostos, tanto como variações e melhorias dos algoritmos tradicionais, tanto envolvendo novas abordagens, como as abordagens multiobjetivo e estratégias de combinação, detalhadas nos modelos múltiplos apresentados no Capítulo 14, e também estratégias de agrupamento não exclusivo, de agrupamento simultâneo de objetos e atributos e de agrupamento e subespaços dos atributos (Jain, 2010).

Capítulo 14

Modelos Múltiplos Descritivos

Como já mencionado no Capítulo 12, a análise de agrupamento engloba vários aspectos e levanta uma série de dificuldades. Para ultrapassar essas dificuldades, têm sido propostas diversas abordagens que combinam diferentes agrupamentos, ou consideram distintos critérios de forma combinada. Essas abordagens mostram-se robustas perante diferentes configurações dos dados.

Atualmente, são seguidos três focos principais para o desenvolvimento de abordagens que superem as limitações e dificuldades encontradas na análise de agrupamentos tradicional:

- *Ensembles* de agrupamentos (Fred, 2001; Fred e Jain, 2002; Strehl e Ghosh, 2002; Ghosh et al., 2002; Fred e Jain, 2003; Topchy et al., 2003, 2004; Fern e Brodley, 2004; Law et al., 2004; Kuncheva et al., 2006) e
- Agrupamento multiobjetivo (Handl e Knowles, 2004, 2005a,b, 2007)
- *Ensemble* multiobjetivo (Coelho et al., 2010; Faceli et al., 2009)

Da mesma maneira que a abordagem tradicional de análise de agrupamento, os *ensembles* são direcionados à obtenção de uma única estrutura que melhor se ajuste aos dados e necessitam de ajustes de parâmetros. Além disso, a presença de um grande número de partições iniciais de baixa qualidade influencia negativamente o resultado da combinação, uma vez que todas elas são consideradas simultaneamente para gerar a partição consenso. Por sua vez, a abordagem multiobjetivo fornece como resultado final um conjunto de estruturas alternativas e não requer muitos ajustes de parâmetros (Handl e Knowles, 2004). Porém, quanto maior o número de alternativas, mais difícil é sua análise por parte dos especialistas no domínio (Handl e Knowles, 2004). A abordagem *ensemble* multiobjetivo tenta superar as dificuldades e aproveitar os benefícios dos *ensembles* e abordagem multiobjetivo.

Os *ensembles* de agrupamentos atuam por meio da combinação posterior de um conjunto de agrupamentos obtidos previamente, seja com um único algoritmo de agrupamento ou com vários algoritmos complementares (Handl e Knowles, 2004). A Seção 14.1 contém uma descrição dos principais conceitos relacionados aos *ensembles* de agrupamentos, uma

comparação entre várias técnicas existentes e uma descrição mais detalhada de algumas delas. Os *ensembles* são mais robustos e fornecem soluções de melhor qualidade que os algoritmos tradicionais de agrupamento, porém não exploram todo o potencial do uso de múltiplos critérios, pois *clusters* que não podem ser detetados por algum dos membros do *ensemble* provavelmente não aparecerão na solução final (Handl e Knowles, 2004). Alternativas que tratam dessa limitação incluem as abordagens de agrupamento multiobjetivo, que realizam a otimização direta de vários objetivos (critérios de agrupamento) simultaneamente, detalhada na Seção 14.2, e as técnicas que integram aspectos dos *ensembles* e da otimização de múltiplos objetivos, que serão apresentadas na Seção 14.3.

Existem outros trabalhos que também se nomeiam agrupamento multiobjetivo, como o de Law et al. (2004), mas, apesar de utilizarem vários objetivos (critérios de agrupamento), esses objetivos não são otimizados diretamente. Como nos *ensembles* de agrupamentos, os autores partem de um conjunto de partições iniciais geradas com diferentes algoritmos de agrupamento. Os diferentes algoritmos é que contemplam os diferentes critérios de agrupamento. A partir dessas partições iniciais, Law et al. (2004) selecionam os *clusters* mais adequados para compor a partição final. Apesar de Law et al. (2004) designarem sua técnica como agrupamento multiobjetivo e explicitamente diferenciá-la dos *ensembles*, ela se encaixa mais nos *ensembles* heterogêneos (partições geradas por diferentes algoritmos) do que na otimização direta de diferentes critérios. Nesse trabalho, os autores utilizam o termo *ensembles* apenas para os *ensembles* homogêneos (partições geradas com um mesmo algoritmo).

14.1 *Ensembles* de Agrupamentos

A combinação de estimadores independentes em comitês, ou *ensembles*, é uma técnica usualmente utilizada em problemas de classificação e regressão para melhorar a precisão de estimadores individuais, aproveitando as características intrínsecas de cada um (LeBlanc e Tibshirani, 1993; Krogh e Vedelsby, 1995; Merz, 1998; Sharkey, 1999; Gama, 1999; Kuncheva, 2004). O Capítulo 8 contém uma descrição detalhada de *ensembles* aplicados a esses problemas. Contudo, a aplicação direta das técnicas de combinação de estimadores aos algoritmos de agrupamento não é possível (Topchy et al., 2003). Os algoritmos de classificação e regressão pertencem ao paradigma de aprendizado supervisionado, em que a resposta correta acompanha cada objeto utilizado na construção do modelo.

Já os algoritmos de agrupamento pertencem ao paradigma de aprendizado não supervisionado. Nesse caso, os dados não possuem uma resposta correta associada, ou seja, não existe informação sobre a "resposta verdadeira" que se espera obter com a aplicação do algoritmo. Essa ausência da "resposta verdadeira" é um dos aspectos que impedem a aplicação direta das técnicas de combinação de estimadores (Topchy et al., 2003). Assim, os *ensembles* de agrupamentos precisam de técnicas novas ou da adaptações das técnicas utilizadas em combinação de estimadores. Por isso, as técnicas utilizadas para combina-

ção em aprendizado supervisionado servem apenas de inspiração para o desenvolvimento de técnicas que permitam a combinação de algoritmos de agrupamento.

Uma definição simplificada do problema de combinação de algoritmos de agrupamento é apresentada por Topchy et al. (2003): dado um conjunto de n^I partições, $\Pi = \{\pi^1, \pi^2, ..., \pi^{n^I}\}$, de um conjunto de dados $\mathbf{X}$ resultantes de várias aplicações de um ou mais algoritmos de agrupamento, encontrar uma partição final π^F (**partição consenso**) de melhor qualidade do que as partições iniciais, chamadas também de **partições base**.

Esta *melhor qualidade* depende do objetivo que se deseja atingir com a combinação, que pode estar relacionado a robustez (Strehl e Ghosh, 2002; Topchy et al., 2004; Fern e Brodley, 2004), novidade (Topchy et al., 2003; Law et al., 2004), estabilidade e estimação da confidência (Monti et al., 2003; Topchy et al., 2004), computação distribuída, paralelização e escalabilidade (Strehl e Ghosh, 2002; Topchy et al., 2004), reuso de conhecimento (Strehl e Ghosh, 2002; Ghosh et al., 2002; Topchy et al., 2004), consistência (Fred, 2001; Fred e Jain, 2002, 2003) e desempenho e custo (Topchy et al., 2003). Os objetivos mais comuns para a utilização dos *ensembles* são:

- **Robustez**: desenvolver uma forma de agrupamento mais robusta, que tenha um desempenho médio melhor para diversos domínios e conjuntos de dados, de preferência sem a necessidade de muitos ajustes manuais.
- **Novidade**: encontrar uma partição final que não possa ser obtida com nenhum algoritmo, individualmente.
- **Estabilidade**: obter soluções de agrupamento com menor sensibilidade a ruídos, *outliers*, variações de amostragem ou à variabilidade dos algoritmos.

A partição obtida pela combinação das várias partições iniciais deve ser consistente ou concordar de alguma forma com essas partições, não ser sensível a pequenas variações nas partições individuais e estar de acordo com informações externas sobre a estrutura dos dados, se essas informações estiverem disponíveis Fred e Jain (2003).

As principais tarefas, ou desafios, inerentes ao problema de combinação de múltiplos agrupamentos são: geração dos agrupamentos a serem combinados e determinação de uma função consenso para combinar os agrupamentos (Topchy et al., 2004; Kuncheva et al., 2006; Hadjitodorov et al., 2006). Assim como existem diferentes objetivos para combinar várias partições, também existem várias maneiras para lidar com essas tarefas. Nas Seções 14.1.1 e 14.1.2 são resumidas as abordagens mais comuns na literatura para gerar a diversidade necessária nas partições iniciais e para encontrar a partição consenso. Na Seção 14.1.3 são detalhadas duas técnicas de *ensemble*.

14.1.1 Geração dos Agrupamentos Iniciais

As partições a serem combinadas devem ser diferentes, de forma a acrescentar informações úteis para a partição final, ou seja, deve haver diversidade no conjunto das partições a serem combinadas. Por isso, ao se buscar uma partição consenso, é preciso ter muito

claro o objetivo do agrupamento final e conhecer bem os algoritmos a serem combinados, a fim de garantir que os algoritmos escolhidos possam contribuir para atingir esse objetivo. Assim, de acordo com o objetivo da combinação, deve-se escolher algoritmos de agrupamento, ou formas de aplicação de um algoritmo, que forneçam a diversidade necessária. As formas atualmente empregadas para obter essa diversidade são (Kuncheva, 2004; Kuncheva et al., 2006; Hadjitodorov et al., 2006):

- Utilizar diferentes algoritmos convencionais de agrupamento para gerar as partições (Qian e Suen, 2000; Kellam et al., 2001; Strehl e Ghosh, 2002; Zeng et al., 2002; Weingessel et al., 2003).
- Executar várias vezes um mesmo algoritmo convencional, com diferentes inicializações (Fred, 2001; Fred e Jain, 2002, 2003; Frossyniotis et al., 2002; Weingessel et al., 2003; Topchy et al., 2004).
- Empregar algoritmos mais simples do que os convencionais, chamados de algoritmos de agrupamento fracos (Topchy et al., 2003).
- Avaliar diferentes conjuntos de dados, que podem ser referentes aos mesmos objetos, cada um com um subconjunto dos atributos originais (Strehl e Ghosh, 2002), ou com projeções do conjunto original em um espaço de dimensão menor (Fern e Brodley, 2004), ou ainda referentes a subconjuntos de objetos (reamostragem), com todas as características originais (Strehl e Ghosh, 2002; Monti et al., 2003; Fern e Brodley, 2004).

Nos casos em que as partições base são geradas por um mesmo algoritmo, o *ensemble* é dito homogêneo. Nesses casos, dependendo das características de cada algoritmo, deve ser estabelecido um conjunto de condições iniciais com as quais o algoritmo deve ser executado para gerar as partições. Por exemplo, o algoritmo k-médias depende da inicialização dos centroides e tem como parâmetro o valor de k. Assim, para esse algoritmo, pode-se gerar partições com vários valores de k, ou partições com um mesmo valor de k, mas geradas com diferentes inicializações dos centroides.

Já um *ensemble* heterogêneo combina partições geradas por algoritmos diferentes. Neste caso, devem ser considerados algoritmos com características bastante distintas, para que as partições geradas possam apresentar uma grande diversidade.

14.1.2 Determinação da Função Consenso

A utilização de uma função consenso é a forma empregada para encontrar uma partição consenso (partição final gerada pela combinação) a partir de partições iniciais (base). Ela constitui a essência da combinação, pois diz como as partições são combinadas.

Vários aspectos dificultam a definição de uma função consenso (Topchy et al., 2003). A ausência de rótulos nos objetos a serem agrupados faz com que não haja uma correspondência explícita entre os *clusters* das diversas partições. Isso é agravado quando as

partições possuem diferentes números de *clusters*, resultando em um problema computacional intratável (problema de correspondência de rótulos[1]). De fato, o problema de combinação de agrupamentos é equivalente ao problema de encontrar uma partição mediana em relação às partições dadas, que é um problema comprovadamente NP-completo (Topchy et al., 2003). Assim, como o critério de agrupamento dos algoritmos convencionais, as funções consenso são heurísticas propostas para a resolução do problema formal de obtenção de uma partição consenso. Uma divisão possível das funções consenso é a de Topchy et al. (2004):

Funções Baseadas em Coassociação

A similaridade entre dois objetos pode ser estimada pelo número de *clusters* compartilhados por eles em todas as partições base. As partições são representadas por uma matriz em que essa similaridade é utilizada para representar a força de coassociação entre os objetos. A partição consenso é obtida pela aplicação de um algoritmo de agrupamento qualquer, que seja baseado em similaridade, a essa matriz de coassociação (Kellam et al., 2001; Strehl e Ghosh, 2002; Fred e Jain, 2002, 2003; Monti et al., 2003).

Alguns dos problemas dessa abordagem são: falta de uma metodologia para a definição do algoritmo de agrupamento a ser utilizado para a combinação e a baixa confiança da estimativa dos valores de coassociação quando um pequeno número de partições é utilizado.

Funções Baseadas em Grafo/Hipergrafo

Nesse caso, as partições base são representadas por um grafo ou por um hipergrafo, e a partição consenso é encontrada empregando uma técnica de particionamento de grafos ou hipergrafos (Strehl e Ghosh, 2002; Fern e Brodley, 2004).

Funções Baseadas em Informação Mútua

A função consenso é formulada em termos da informação mútua entre os rótulos na partição consenso e os rótulos nas partições iniciais. Strehl e Ghosh (2002) definem a informação mútua normalizada média entre uma partição qualquer e um conjunto de partições iniciais. A partição consenso é dada pelo máximo dessa função, considerando o número de *clusters* desejado. Porém, a dificuldade de otimização dessa função fez com que os autores utilizassem heurísticas baseadas em coassociação e hipergrafo.

Fred e Jain (2003) também definem formalmente uma função consenso baseada em informação mútua, mas resolvem o problema com uma heurística baseada em coassociação.

Topchy et al. (2003) definem uma função consenso baseada na informação mútua generalizada, que é equivalente à variância intracluster em um espaço de rótulos dos *clusters* especialmente transformado. A função é então otimizada com o algoritmo *k*-médias.

Funções Baseadas em Votação

Um mecanismo de votação é utilizado para atribuir os objetos aos *clusters* da partição

[1] Rótulo dado pelo algoritmo de agrupamento para identificar um *cluster*.

consenso, dado que o problema de correspondência dos rótulos seja solucionado para todas as partições base. Entretanto, esse problema de correspondência dos rótulos é de difícil solução, sendo às vezes intratável. Porém, pode-se obter uma aproximação heurística de uma rotulação consistente. Todas as partições podem ser rerrotuladas com base em sua melhor concordância com uma partição referência, que pode ser uma das partições base ou um novo agrupamento do conjunto de dados. Esse procedimento é utilizado por Fred (2001) e Weingessel et al. (2003).

Frossyniotis et al. (2002) propõem a construção das partições juntamente com um processo de renumeração dos *clusters* seguido de votação. A partir disso, são estabelecidas relações de vizinhança entre os *clusters*. Essas informações são utilizadas para fundir os *clusters* mais próximos, resultando na partição final.

Além desses tipos de função consenso, Topchy et al. (2004) definem uma função consenso baseada em uma solução para o problema de probabilidade máxima para um modelo misto finito do conjunto de partições iniciais. Esse conjunto de partições é modelado como uma mistura de distribuições multinomiais no espaço dos rótulos dos *clusters*. O problema de probabilidade máxima pode ser resolvido com o algoritmo EM.

A Tabela 14.1 contém um resumo das principais características das abordagens citadas, utilizadas para *ensemble* de agrupamentos. Nessa tabela estão resumidos a forma de representação das partições base, a função consenso, o objetivo da combinação, os algoritmos empregados e a maneira como foram utilizados para gerar diversidade para as partições base.

Tabela 14.1 *Comparação das formas de combinação de agrupamentos*

Artigo	Representação	Função consenso	Objetivo da combinação	Algoritmos Combinados	Diversidade
Kellam et al. (2001)	Matriz de concordância	Os *clusters* finais são aqueles que possuem os mesmos objetos em todas as partições	*Clusters* robustos (*clusters* em que os objetos aparecem junto em todas as partições)	Hierárquico, *k*-médias, SOM e algoritmos genéticos, com o coeficiente de correlação de Pearson	Vários algoritmos
Fred (2001)	Matriz de coassociação	Votação	Consistência	*k*-médias	Mesmo algoritmo com diferentes inicializações
Fred e Jain (2002)	Matriz de coassociação	Ligação simples com um novo critério para determinar a partição final	Consistência	*k*-médias	Mesmo algoritmo com diferentes inicializações
Strehl e Ghosh (2002)	Hipergrafo	Particionamento de grafo de similaridade, particionamento de corte mínimo e metaclusters	Reuso de Conhecimento, Computação distribuída e Robustez	Particionamento de grafo e *k*-médias, com várias similaridades SOM e particionamento de hipergrafo	Vários algoritmos e mesmo algoritmo com dados diferentes
Frossyniotis et al. (2002)	Tabela de votação e tabela de vizinhança	Votação	Robustez e estabilidade	*k*-médias e *greedy*-EM	Mesmo algoritmo com diferentes inicializações
Monti et al. (2003)	Matriz consenso	LM determinando *k* com base na estabilidade	Estabilidade	LM e SOM	Mesmo algoritmo com dados diferentes
Fred e Jain (2003)	Matriz de coassociação usando votação	Ligação simples (qualquer função baseada em similaridade)	Consistência, estabilidade e robustez	*k*-médias	Mesmo algoritmo com diferentes inicializações
Weingessel et al. (2003)	Conjunto de matrizes de pertinência das partições iniciais	Votação/fusão	Robustez	*k*-médias, *hard competitive learning* e aprendizagem competitivo *fuzzy*	Vários algoritmos e mesmo algoritmo com diferentes inicializações
Topchy et al. (2003)	Novo conjunto de características dos padrões	Baseada no *k*-médias aplicado no novo espaço de características	Desempenho e custo	Algoritmos fracos que usam projeções ou divisões aleatórias dos dados	Mesmo algoritmo com diferentes inicializações
Topchy et al. (2004)	Novo conjunto de características dos objetos	Probabilidade máxima encontrada com o método EM	Robustez, estabilidade, escalabilidade e reuso do conhecimento	*k*-médias	Mesmo algoritmo com diferentes inicializações
Fern e Brodley (2004)	Grafo	Particionamento de grafo	Robustez	*k*-médias	Mesmo algoritmo com dados diferentes
Law et al. (2004)	Conjunto com todos os *clusters*	*Clusters* mais estáveis	Novidade e robustez	*k*-médias, EM, hierárquico e *spectral clustering*	Vários algoritmos

14.1.3 Algumas Técnicas Ilustrativas

As abordagens para construção de *ensembles* de agrupamentos baseadas em grafos podem ser mais robustas que as técnicas baseadas em matriz de coassociação (Topchy et al., 2005). Uma das técnicas de *ensembles* de agrupamentos mais populares é o algoritmo MCLA (do inglês *Meta-CLustering Algorithm*), proposta por Strehl e Ghosh (2002). Outra abordagem para *ensemble* baseada em particionamento de grafos é o algoritmo HBGF (do inglês *Hybrid Bipartite Graph Formulation*), proposto por Fern e Brodley (2004). Nas duas abordagens, não são necessários os atributos originais dos objetos, apenas os rótulos dos *clusters* de cada objeto nas partições a serem combinadas. Essas duas abordagens são detalhadas a seguir.

Ensemble de Strehl e Ghosh

Strehl e Ghosh (2002) definem formalmente a combinação de algoritmos de agrupamento como um problema de otimização de uma função consenso baseada na informação mútua, compartilhada entre as soluções individuais. Entretanto, como a otimização dessa função é um problema combinatorial difícil, eles propõem três algoritmos de combinação baseados em heurísticas: CSPA (do inglês *Cluster-based Similarity Partitioning Algorithm*), HGPA (do inglês *HiperGraph-Partitioning Algorithm*) e MCLA (do inglês *Meta-CLustering Algorithm*). Como esses algoritmos têm um custo computacional baixo, Strehl e Ghosh (2002) aplicam os três algoritmos, cada um gerando uma partição consenso, e utilizam uma função supraconsenso baseada na informação mútua para escolher qual delas será a partição consenso final. Assim, a partição consenso final será aquela, dentre as três, que tem a melhor informação mútua compartilhada.

A definição formal do problema como de otimização de uma função consenso baseada na informação mútua compartilhada entre as soluções individuais é feita da seguinte maneira. Seja a informação mútua normalizada (*NMI*, do inglês *Normalized Mutual Information*) estimada entre duas partições π^a e π^b, dada pela Equação 14.1, em que $|.|$ indica o número de objetos de um conjunto, k^a e k^b são os números de *clusters* das partições π^a e π^b, respetivamente, $\mathbf{C}_h{}^a$ é o h-ésimo *cluster* de π^a, $\mathbf{C}_l{}^b$ é o l-ésimo *cluster* de π^b e n é o número de objetos do conjunto de dados (Strehl e Ghosh, 2002).

$$\phi^{(NMI)}(\pi^a,\pi^b) = \frac{\sum_{h=1}^{k^a}\sum_{l=1}^{k^b}\left|\mathbf{C}_h{}^a \cap \mathbf{C}_l{}^b\right| \log\left(\frac{n\left|\mathbf{C}_h{}^a \cap \mathbf{C}_l{}^b\right|}{\left|\mathbf{C}_h{}^a\right|\left|\mathbf{C}_l{}^b\right|}\right)}{\sqrt{\left(\sum_{h=1}^{k^a}\left|\mathbf{C}_h{}^a\right| \log\left(\frac{\left|\mathbf{C}_h{}^a\right|}{n}\right)\right)\left(\sum_{l=1}^{k^b}\left|\mathbf{C}_l{}^b\right| \log\left(\frac{\left|\mathbf{C}_l{}^b\right|}{n}\right)\right)}} \tag{14.1}$$

Com base nessa medida de informação mútua entre duas partições, Strehl e Ghosh (2002) definem uma medida de informação mútua entre uma partição π^i e um conjunto

Π de r partições, como a informação mútua normalizada média (*ANMI*), dada pela Equação 14.2.

$$\phi^{(ANMI)}(\pi^i, \Pi) = \frac{1}{r} \sum_{q=1}^{r} \phi^{(NMI)}(\pi^i, \pi^q) \tag{14.2}$$

em que $\phi^{(ANMI)}$ é a função objetivo, e a partição consenso $\pi^{F(k-opt)}$ é aquela com informação mútua normalizada média ($\phi^{(ANMI)}$) máxima, em relação às partições individuais em Π, dado que o número de *clusters* desejado para a partição consenso é k. $\pi^{F(k-opt)}$ é dada pela Equação 14.3, em que π^i corresponde a todas as possíveis partições com k *clusters*.

$$\pi^F = \arg\max_{\pi^i} \sum_{q=1}^{r} \phi^{(NMI)}(\pi^i, \pi^q) \tag{14.3}$$

Como já mencionado, a $\phi^{(ANMI)}$ não é otimizada. Os três algoritmos baseados em heurísticas são executados, e a $\phi^{(ANMI)}$ é utilizada para selecionar a melhor partição dentre as geradas com eles.

Os três algoritmos, baseados em heurísticas e propostos para encontrar uma partição consenso, partem de uma representação inicial das partições na forma de um hipergrafo. No CSPA, o problema não depende dessa representação, mas ela é utilizada para calcular facilmente uma matriz de similaridade que servirá como entrada para um algoritmo de agrupamento. No HGPA o hipergrafo é empregado diretamente. Já no MCLA, ele é utilizado para a construção de um metagrafo a ser particionado e posterior determinação da partição consenso, como descrito mais adiante.

O hipergrafo utilizado é representado por uma matriz de adjacências. Cada coluna da matriz é uma hiperaresta, que representa um *cluster*. Essa matriz é construída pela concatenação das matrizes binárias de pertinência de cada partição. As linhas da matriz de pertinência de uma partição correspondem aos objetos, e as colunas correspondem aos seus *clusters*. Cada célula da matriz contém o valor 1 se o objeto pertence ao *cluster* e 0 em caso contrário.

O algoritmo CSPA utiliza a heurística mais simples, porém possui complexidade quadrática no número de objetos. A função consenso resultante desse algoritmo está classificada entre as funções baseadas em coassociação. Como já mencionado, esse algoritmo se baseia na construção de uma nova matriz de similaridade a partir das partições originais. As entradas dessa matriz denotam a fração das partições nas quais dois objetos pertencem ao mesmo *cluster*. A matriz de similaridade gerada é então utilizada para reagrupar os objetos por meio de um algoritmo de agrupamento qualquer, que seja baseado em similaridade. Strehl e Ghosh (2002) utilizam o algoritmo METIS[2] (Karypis e Kumar, 1999) para particionar o grafo de similaridade induzido.

No algoritmo HGPA, a combinação é tratada como um problema de particionamento de um hipergrafo definido apropriadamente, no qual as hiperarestas representam *clusters*.

[2]http://glaros.dtc.umn.edu/gkhome/metis/metis/overview

Esse particionamento é feito cortando um número mínimo de hiperarestas. Para isso é utilizado o pacote de particionamento de hipergrafos HMETIS.[3]

O algoritmo MCLA trata a combinação como um problema de correspondência dos *clusters* das partições iniciais. Ele se baseia no agrupamento desses *clusters*. Um metagrafo em que cada vértice corresponde a um *cluster* é construído. Em seguida, o metagrafo é particionado de maneira que os *clusters* que permaneceram em um mesmo grupo (metacluster) sejam correspondentes. Os objetos são então atribuídos aos metaclusters com os quais eles estão mais fortemente associados.

Em experimentos controlados realizados por Strehl e Ghosh (2002) para comparar os três algoritmos, foi observado que o MCLA apresentou melhor resultado na presença de uma quantidade média para alta de ruído, o que ocorre com frequência em problemas reais. Em relação à complexidade, o algoritmo MCLA também é o mais vantajoso. Em outros experimentos, Strehl e Ghosh (2002) observaram que cada algoritmo tem um desempenho melhor para uma situação diferente. O algoritmo MCLA é melhor quando há menos diversidade nas partições iniciais, o que está de acordo com a suposição inicial do MCLA de que existe uma correspondência entre os *clusters* das partições a serem combinadas. O MCLA, apresentado no Algoritmo 14.1, será descrito em mais detalhes a seguir. Como nesse algoritmo não é garantido que todo metacluster tenha pelo menos um objeto, a partição π^F tem no máximo (e não exatamente) k *clusters*.

Para o particionamento do metagrafo mencionado na linha 6 do Algoritmo 14.1, o pacote de particionamento de grafos METIS[4] (Karypis e Kumar, 1999) pode ser utilizado. Essa fase de particionamento permite encontrar os *clusters* das partições iniciais que são correspondentes.

Quanto à diversidade das partições iniciais, dependendo do cenário de aplicação considerado, foi definida uma alternativa diferente para gerar as partições iniciais. No primeiro cenário, as partições originais foram formadas com a aplicação de um único algoritmo de agrupamento a diferentes subconjuntos de atributos dos dados. No segundo cenário, as partições foram obtidas também com um único algoritmo, porém aplicado a diferentes subconjuntos de objetos, considerando sempre todos os atributos. Finalmente, no terceiro cenário, as partições iniciais foram geradas a partir da execução de diferentes algoritmos, empregando diferentes medidas de proximidade ao mesmo conjunto de dados.

Para ilustrar o funcionamento do algoritmo MCLA, considere os agrupamentos mostrados na Tabela 14.2 (Strehl e Ghosh, 2002). O hipergrafo que representa esses agrupamentos pode ser observado na matriz de adjacências da Tabela 14.3. Nessa tabela, cada objeto $\mathbf{x}_i$ corresponde a um vértice e cada hiperaresta h_j representa um dos *clusters* de um dos agrupamentos. Com essas informações é construído o metagrafo em que cada hiperaresta é um vértice e cujos pesos das arestas entre os vértices podem ser observados na Tabela 14.4. O particionamento desse grafo em três partes resulta nos metaclusters $\mathbf{C}_1{}^M = \{h_3, h_4, h_9\}$, $\mathbf{C}_2{}^M = \{h_2, h_6, h_8, h_{10}\}$ e $\mathbf{C}_3{}^M = \{h_1, h_5, h_7, h_{11}\}$. Assim, os *clusters* representados por h_3, h_4 e h_9, por exemplo, são correspondentes.

[3] http://glaros.dtc.umn.edu/gkhome/metis/hmetis/overview
[4] http://glaros.dtc.umn.edu/gkhome/metis/metis/overview

Algoritmo 14.1 Algoritmo MCLA

Entrada: Um conjunto de partições base $\Pi = \{\pi^1, \pi^2, ..., \pi^{n^I}\}$
Um conjunto de dados $\mathbf{X}_{n \times d}$
Número máximo de *clusters* de π^F, k
Saída: Uma partição consenso π^F

Construir um metagrafo $G = (V, W)$:
  $V \leftarrow$ vértices representando os *clusters* de todas as partições do conjunto de partições base Π `// as hiperarestas do hipergrafo descrito`
  para cada *aresta ligando os vértices i e* $j \in V$ **faça**
    $w(i, j) \leftarrow \left|\mathbf{C}_i \cap \mathbf{C}_j\right| / \left|\mathbf{C}_i \cup \mathbf{C}_j\right|$ `// razão do número de objetos na interseção e na união dos` *clusters* $\mathbf{C}_i$ `e` $\mathbf{C}_j$`, pertencentes às partições em` Π
  fim
Agrupar as hiperarestas (*clusters*), particionando o metagrafo em k metaclusters balanceados `// Cada metacluster resultante do particionamento representa um grupo de` *clusters* `correspondentes`
Unir os *clusters* de cada metacluster:
  para cada metacluster $\mathbf{C}_i{}^M$ **faça**
    Transformar as hiperarestas em uma única meta-hiperaresta
    Calcular um vetor de associação descrevendo o nível de associação de cada objeto com o metacluster `// O vetor de associação é obtido pelo cálculo da média dos vetores que representam as hiperarestas de um metacluster. Assim, o nível de associação de um objeto a um metacluster é dado pela média do número de` *clusters* `desse metacluster, que contém o objeto.`
  fim
Determinar o metacluster final de cada objeto:
  para cada *objeto* **x faça**
    Associar **x** ao metacluster para o qual ele possui o maior valor de associação `// Desempates são decididos aleatoriamente`
  fim
$\pi^F \leftarrow$ partição dos objetos indicada pelos metaclusters

Em seguida, os *clusters* de cada metacluster são unidos, formando as meta-hiperarestas h_i^M representadas na Tabela 14.5. Nessa tabela também estão representados os respectivos vetores de associação $a(h_i^M)$. Com base nessas informações, o metacluster final de cada objeto é determinado. Assim, $\mathbf{x}_1$, por exemplo, vai pertencer ao metacluster $\mathbf{C}_3{}^M$, pois é

Tabela 14.2 *Exemplo do MCLA - partições*

Partição	*Clusters*
π^1	${\mathbf{C}_1}^1 = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\}$, ${\mathbf{C}_2}^1 = \{\mathbf{x}_4, \mathbf{x}_5\}$, ${\mathbf{C}_3}^1 = \{\mathbf{x}_6, \mathbf{x}_7\}$
π^2	${\mathbf{C}_1}^2 = \{\mathbf{x}_6, \mathbf{x}_7\}$, ${\mathbf{C}_2}^2 = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\}$, ${\mathbf{C}_3}^2 = \{\mathbf{x}_4, \mathbf{x}_5\}$
π^3	${\mathbf{C}_1}^3 = \{\mathbf{x}_1, \mathbf{x}_2\}$, ${\mathbf{C}_2}^3 = \{\mathbf{x}_3, \mathbf{x}_4\}$, ${\mathbf{C}_3}^3 = \{\mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_7\}$
π^4	${\mathbf{C}_1}^4 = \{\mathbf{x}_1, \mathbf{x}_4\}$, ${\mathbf{C}_2}^4 = \{\mathbf{x}_2, \mathbf{x}_5\}$, objetos $\mathbf{x}_3$, $\mathbf{x}_6$ e $\mathbf{x}_7$ não agrupados

Tabela 14.3 *Exemplo do MCLA - hipergrafo*

Vértices	h_1 ${\mathbf{C}_1}^1$	h_2 ${\mathbf{C}_2}^1$	h_3 ${\mathbf{C}_3}^1$	h_4 ${\mathbf{C}_1}^2$	h_5 ${\mathbf{C}_2}^2$	h_6 ${\mathbf{C}_3}^2$	h_7 ${\mathbf{C}_1}^3$	h_8 ${\mathbf{C}_2}^3$	h_9 ${\mathbf{C}_3}^3$	h_{10} ${\mathbf{C}_1}^4$	h_{11} ${\mathbf{C}_2}^4$
$\mathbf{x}_1$	1	0	0	0	1	0	1	0	0	1	0
$\mathbf{x}_2$	1	0	0	0	1	0	1	0	0	0	1
$\mathbf{x}_3$	1	0	0	0	1	0	0	1	0	0	0
$\mathbf{x}_4$	0	1	0	0	0	1	0	1	0	1	0
$\mathbf{x}_5$	0	1	0	0	0	1	0	0	1	0	1
$\mathbf{x}_6$	0	0	1	1	0	0	0	0	1	0	0
$\mathbf{x}_7$	0	0	1	1	0	0	0	0	1	0	0

Tabela 14.4 *Exemplo do MCLA - pesos*

Vértices	h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8	h_9	h_{10}	h_{11}
h_1	-	-	-	0,00	0,50	0,00	0,40	0,20	0,00	0,20	0,20
h_2	-	-	-	0,00	0,00	0,50	0,00	0,25	0,20	0,25	0,25
h_3	-	-	-	0,50	0,00	0,00	0,00	0,00	0,40	0,00	0,00
h_4	0,00	0,00	0,50	-	-	-	0,00	0,00	0,40	0,00	0,00
h_5	0,50	0,00	0,00	-	-	-	0,40	0,20	0,00	0,20	0,20
h_6	0,00	0,50	0,00	-	-	-	0,00	0,25	0,20	0,25	0,25
h_7	0,40	0,00	0,00	0,00	0,40	0,00	-	-	-	0,25	0,25
h_8	0,20	0,25	0,00	0	0,20	0,25	-	-	-	0,25	0,00
h_9	0,00	0,20	0,40	0,40	0,00	0,20	-	-	-	0,00	0,20
h_{10}	0,20	0,25	0,00	0,00	0,20	0,25	0,25	0,25	0,00	-	-
h_{11}	0,20	0,25	0,00	0,00	0,20	0,25	0,25	0,00	0,20	-	-

o metacluster com o qual $\mathbf{x}_1$ tem o maior valor de associação (0,75). A partição consenso π^F fica composta pelos *clusters* ${\mathbf{C}_1}^F = \{\mathbf{x}_6, \mathbf{x}_7\}$, ${\mathbf{C}_2}^F = \{\mathbf{x}_4, \mathbf{x}_5\}$ e ${\mathbf{C}_3}^F = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\}$.

Tabela 14.5 *Exemplo do MCLA - meta-hiperarestas e vetores de associação*

Vértices	$\mathbf{C}_1{}^M = \{h_3, h_4, h_9\}$		$\mathbf{C}_2{}^M = \{h_2, h_6, h_8, h_{10}\}$		$\mathbf{C}_3{}^M = \{h_1, h_5, h_7, h_{11}\}$	
	h_1^M	$a(h_1^M)$	h_2^M	$a(h_2^M)$	h_3^M	$a(h_3^M)$
$\mathbf{x}_1$	0,00	0,00	1,00	0,25	1,00	0,75
$\mathbf{x}_2$	0,00	0,00	0,00	0,00	1,00	1,00
$\mathbf{x}_3$	0,00	0,00	1,00	0,25	1,00	0,50
$\mathbf{x}_4$	0,00	0,00	1,00	1,00	0,00	0,00
$\mathbf{x}_5$	1,00	0,33	1,00	0,50	1,00	0,25
$\mathbf{x}_6$	1,00	1,00	0,00	0,00	0,00	0,00
$\mathbf{x}_7$	1,00	1,00	0,00	0,00	0,00	0,00

Ensemble de Fern e Brodley

A abordagem de Fern e Brodley (2004) utiliza particionamento de um grafo bipartido para, dado um conjunto de partições base, encontrar uma partição consenso. Nessa abordagem, os autores constroem um grafo bipartido a partir do conjunto de partições a serem combinadas. Para isso, modelam simultaneamente tanto objetos quanto *clusters* como vértices do grafo e, posteriormente, particionam o grafo com uma técnica tradicional de particionamento de grafos. O algoritmo de Fern e Brodley (2004), chamado HBGF (do inglês *Hybrid Bipartite Graph Formulation*), é descrito no Algoritmo 14.2.

Para o particionamento do grafo mencionado na linha 14 do algoritmo, Fern e Brodley (2004) utilizam duas técnicas distintas: *Spectral Graph Partitioning* (Ng et al., 2002) e METIS (Karypis e Kumar, 1999).

Para gerar as partições base, Fern e Brodley (2004) consideraram duas abordagens. A primeira aplica um algoritmo a diferentes subconjuntos dos dados (reamostragem). A segunda aplica um algoritmo ao conjunto de dados completo (todos os objetos), porém composto de projeções dos objetos em um espaço de dimensão menor do que o espaço de atributos original.

Nos experimentos apresentados em Fern e Brodley (2004), os autores comentam que sua abordagem, HBGF, apresentou um desempenho equivalente ou superior àqueles obtidos pelas abordagens de Strehl e Ghosh (2002).

Para ilustrar o funcionamento do algoritmo HBGF, considere os agrupamentos ilustrados na Tabela 14.6 (Fern e Brodley, 2004). Inicialmente é construído o grafo bipartido mostrado na Figura 14.1. Os vértices de V^C, representados por um losango, correspondem aos *clusters* de π^1 e π^2, e os vértices de V^O, representados por um círculo, correspondem aos objetos. Todas as arestas representadas têm peso 1 e ligam um objeto a um *cluster*, indicando que o objeto pertence àquele *cluster*. A linha tracejada destaca uma partição desse grafo em duas partes, obtida com algum algoritmo de particionamento de grafos tradicional. A divisão dos objetos resultante desse particionamento é a partição consenso π^F, composta pelos *clusters* $\mathbf{C}_1{}^F = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5\}$ e $\mathbf{C}_2{}^F = \{\mathbf{x}_6, \mathbf{x}_7, \mathbf{x}_8, \mathbf{x}_9\}$.

Algoritmo 14.2 Algoritmo HBGF

Entrada: Um conjunto de partições base Π
Um conjunto de dados $\mathbf{X}_{n\times d}$
Número de *clusters* k
Saída: Uma partição consenso π^F

Construir um grafo $G = (V,W)$ a partir do conjunto de partições base, da seguinte maneira:
  $V^C \leftarrow$ vértices representando os *clusters* do conjunto de partições base Π
  $V^O \leftarrow$ vértices representando os objetos do conjunto de dados $\mathbf{X}$
  $V \leftarrow V^C \cup V^O$
  para cada *aresta ligando os vértices* i *e* $j \in V$ **faça**
    se $i,j \in V^C$ *ou* $i,j \in V^O$ `// ambas representam clusters ou ambas representam objetos`
    então
      $w(i,j) \leftarrow 0$
    senão se *o objeto* $\mathbf{x}_i$ *pertence ao cluster* $\mathbf{C}_j$ **então**
      $w(i,j) \leftarrow 1$
    senão
      $w(i,j) \leftarrow 0$
  fim
Particionar o grafo $G = (V,W)$ em k *clusters* utilizando qualquer técnica de particionamento de grafos tradicional
Compor a partição consenso π^F de acordo com a divisão dos objetos, resultante do particionamento do grafo

14.2 Agrupamento Multiobjetivo

A ideia básica do agrupamento multiobjetivo é otimizar simultaneamente dois ou mais critérios de agrupamento que sejam complementares (Handl e Knowles, 2004, 2005a,b, 2007). Um dos primeiros algoritmos de agrupamento multiobjetivo foi o MOCK (*Multi-Objective Clustering with automatic K-determination*), proposto por Handl e Knowles (2004, 2005a,b, 2007).

O algoritmo MOCK é um algoritmo evolutivo multiobjetivo, capaz de otimizar simultaneamente dois objetivos complementares e identificar automaticamente as melhores partições do fronte de Pareto, determinando de forma automática o número de *clusters*. Com base na forma da fronteira de Pareto, MOCK retorna não apenas um conjunto de partições com diferentes compromissos em um intervalo de números de *clusters*, mas também uma indicação de quais dessas partições são as melhores. MOCK tenta encontrar a fronteira de Pareto mais completa possível para, posteriormente, reduzir esse conjunto a uma

Tabela 14.6 *Exemplo do HBGF - partições*

Partição	*Clusters*
π^1	$\mathbf{C}_1{}^1 = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\}$, $\mathbf{C}_2{}^1 = \{\mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_7, \mathbf{x}_8, \mathbf{x}_9\}$
π^2	$\mathbf{C}_1{}^2 = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_6\}$, $\mathbf{C}_2{}^2 = \{\mathbf{x}_7, \mathbf{x}_8, \mathbf{x}_9\}$

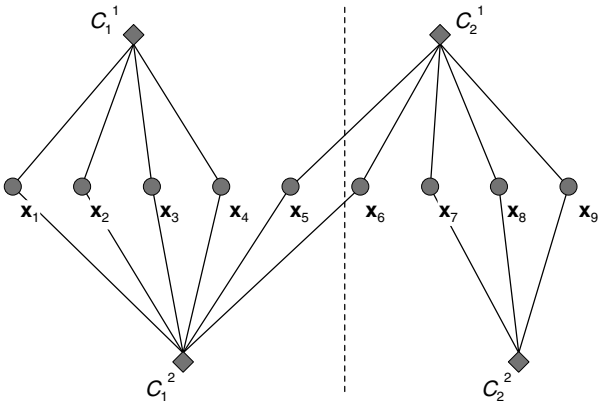


Figura 14.1 *Exemplo do HBGF - grafo bipartido.*

única solução, por meio de uma pontuação calculada com base em frontes de referência (*attainment score*).

O MOCK é baseado no algoritmo evolutivo multiobjetivo PESA-II (Corne et al., 2001). Esse algoritmo mantém duas populações de soluções: uma interna, de tamanho fixo, e uma externa, de tamanho variável e limitado. O propósito da população externa é tirar proveito das boas soluções. Para isso, PESA-II utiliza elitismo, mantendo um conjunto grande e diverso de soluções não dominadas. A população interna é usada para investigar novas soluções por meio dos processos padrão de recombinação e mutação. As soluções presentes na população externa são mantidas em nichos. É mantido um registro do número de soluções que ocupam cada nicho, e esse registro é utilizado para fazer com que as soluções cubram todo o espaço de objetivos, ao invés de se agruparem todas em uma única região. Para isso, as soluções não dominadas que entrariam em uma população externa cheia apenas o farão se elas ocuparem um nicho menos cheio do que algumas outras soluções. Além disso, quando a população interna é construída a partir da população externa, os indivíduos são selecionados uniformemente dentre os nichos povoados (todos os nichos contribuem igualmente). A política de seleção baseada em nichos do PESA-II utiliza uma faixa adaptável de equalização e normalização dos valores das funções objetivos. Isso torna desnecessário o ajuste de parâmetros, que muitas vezes é complicado, e faz com que funções objetivo com variações diferentes possam ser prontamente utilizadas. Além disso, qualquer número de objetivos pode ser utilizado.

A representação dos indivíduos utilizada no MOCK é a representação de adjacência baseada em lócus (*locus-based adjacency representation*) (Park e Song, 1998). Nessa representação, cada indivíduo g consiste em n genes, $g_1, g_2, ..., g_n$. Cada gene g_i pode assumir um valor j no intervalo $[1, n]$, significando que existe uma ligação entre os objetos $\mathbf{x}_i$ e $\mathbf{x}_j$, ou seja, os objetos $\mathbf{x}_i$ e $\mathbf{x}_j$ estão no mesmo *cluster*. A decodificação dessa representação requer a identificação de todos os subgrafos. Todos os objetos pertencentes ao mesmo subgrafo são associados ao mesmo *cluster*.

Como operador de recombinação, é utilizado cruzamento uniforme. Para mutação, os autores originalmente empregaram um operador especializado que reduz significativamente o tamanho do espaço de busca (Handl e Knowles, 2004), a mutação dos vizinhos mais próximos. Nessa mutação, cada objeto pode ter sua ligação alterada apenas para um dos seus v vizinhos mais próximos. Consequentemente, $g_i \in \{nn_{i1}, ..., nn_{iv}\}$, em que nn_{il} é o l-ésimo vizinho mais próximo do objeto $\mathbf{x}_i$. Nesse caso, todos os genes têm uma probabilidade de mutação igual ($\frac{1}{n}$). Posteriormente, os autores propuseram uma modificação na mutação que altera a probabilidade de mutação de ligações individuais, $i \rightarrow j$, para $p_m = \frac{1}{n} + (\frac{l}{n})^2$, em que $j = nn_{il}$ (Handl e Knowles, 2005b).

O procedimento para inicialização da população originalmente utiliza árvore geradora mínima (MST, do inglês *Minimum Spanning Tree*) para gerar os indivíduos (Handl e Knowles, 2004). No início, é gerada uma MST completa utilizando o algoritmo de Prim (Wilson e Watkins, 1990). O i-ésimo indivíduo da população inicial é inicializado pela MST com as $(i-1)$-ésimas ligações mais longas removidas. Porém, esse procedimento tende a gerar soluções boas na região do fronte de Pareto em que a conectividade é baixa e, quando os *clusters* não são bem separados, gerar soluções muito parecidas (Handl e Knowles, 2005b). Para melhorar o espalhamento das soluções iniciais, Handl e Knowles (2005b) propõem uma nova inicialização baseada em uma mistura de soluções geradas com o algoritmo k-médias e com MST.

Para as soluções baseadas na MST, inicialmente é construída uma MST e são identificadas todas as suas b ligações interessantes. Uma ligação $i \rightarrow j$ é considerada interessante se e somente se $i = nn_{jl}$ e $j = nn_{ik}$, com $l > v$ e $k > v$, em que v, o número de vizinhos mais próximos, é dado pelo utilizador. O grau de interesse é $gi = \min(l, k)$. As b ligações interessantes são ordenadas pelo seu grau de interesse. O conjunto de partições baseadas na MST é construído da seguinte maneira: para cada $g \in [0, \min(b, 0,5n^I)]$, em que n^I é o tamanho da população inicial, é gerado um agrupamento π^g removendo as g primeiras ligações interessantes. As ligações perdidas são substituídas por uma ligação com um vizinho j escolhido aleatoriamente, com $j = nn_{il}$ e $l \leq v$. Para as soluções baseadas no k-médias, o algoritmo é executado, gerando partições com números de *clusters* $k \in [2, n^I - (min(b, 0,5n^I) + 1)]$. As partições obtidas dessa maneira são convertidas para a representação apropriada.

Como funções objetivo, MOCK emprega dois objetivos complementares, a variância intracluster (*var*) (Equação 15.1) e a conectividade (*con*) (Equação 15.2), ambos definidos na Seção 15.2.1. Essas medidas foram escolhidas por representar dois aspectos fundamentalmente diferentes de qualidade de um agrupamento. Esses objetivos contrabalançam

suas tendências de aumentar ou diminuir com o número de *clusters*. Isso é importante para explorar bem o espaço de soluções, evitando a convergência para soluções triviais (*n clusters* com um único objeto, no caso da variância intracluster, e apenas um *cluster* com *n* elementos, no caso da conectividade) (Handl e Knowles, 2004).

A aplicação desse algoritmo gera um conjunto de soluções não dominadas com diferentes compromissos dos dois objetivos e com diferentes números de *clusters*. Para encontrar a melhor solução, Handl e Knowles geram o fronte de Pareto mais completo possível e, posteriormente, fazendo uso de várias considerações específicas do domínio, reduzem o conjunto de soluções a uma única solução (Handl e Knowles, 2004, 2005a).

O procedimento para selecionar a melhor solução é baseado na intuição de que a estrutura dos dados está refletida na forma do fronte de Pareto (Handl e Knowles, 2004, 2005a). Das tendências observadas nos objetivos empregados, é possível afirmar que, incrementando o número de *clusters*, k, obtém-se uma melhora na variância, δV, ao custo de uma degradação na conectividade, δC. Para um número de *clusters* k menor do que o verdadeiro, espera-se que a razão $R = \delta V / \delta C$ seja grande, pois a separação de dois *clusters* causa uma grande diminuição na variância, com pouco ou nenhum aumento na conectividade. Para os números de *clusters* maiores do que o verdadeiro, essa razão se torna menor, pois a diminuição na variância é menor, mas ao preço de um aumento maior da conectividade (um *cluster* verdadeiro está sendo dividido). Pela tendência das medidas, as soluções na fronteira de Pareto estão aproximadamente ordenadas por k.

Assim, a mudança evidente em R que ocorre no número correto de *clusters* pode ser observada como um ponto de inflexão no gráfico do fronte de Pareto. Para determinar corretamente esse ponto, os autores utilizam distribuições aleatórias de dados como referência (Handl e Knowles, 2004, 2005a,b). Esses dados são agrupados com o MOCK, gerando um conjunto de frontes de referência. O fronte solução é normalizado, e, para cada ponto nesse fronte, é calculado um *attainment score*, dado pela sua distância até as *attainment surfaces* dos frontes de referência. Em seguida, é traçado um gráfico dos *attainment scores* em função de k. A solução correspondente ao máximo dessa curva é selecionada como a melhor solução.

Além de uma melhor solução, também é possível identificar outros possíveis máximos locais, que podem revelar estruturas em outros níveis. Além disso, o *attainment score* também serve como uma estimativa da qualidade de cada uma das soluções individuais.

Em Handl e Knowles (2004, 2005a), MOCK é comparado a três algoritmos de agrupamento convencionais (k-médias e algoritmos hierárquicos com ligação simples e média) e também ao *ensemble* de agrupamentos proposto por Strehl e Ghosh (2002), utilizando os três algoritmos e a função supraconsenso citados anteriormente. Os autores mostraram que MOCK é mais robusto do que as outras abordagens em relação à variedade de estruturas encontradas em conjuntos de dados diferentes e é capaz de encontrar certas estruturas que outros métodos não conseguem.

14.3 *Ensemble* Multiobjetivo

O *ensemble* multiobjetivo combina as duas abordagens previamente apresentadas. Ele cria um conjunto diverso de partições base que são depois combinadas para a determinação do consenso, utilizando para isso estratégias multiobjetivo. Com isso, os *ensembles* multiobjetivo encontram um conjunto de partições consenso, que representam diferentes compromissos entre os critérios considerados na combinação. Dentre as abordagens de *ensembles* multiobjetivo estão o MOCLE (do inglês *Multi-Objective Clustering Ensemble*) (Faceli et al., 2009) e o MCHPF (*Multi-objective Clustering with Hierarchical Partitions Fusions*) (Coelho et al., 2010).

O MOCLE integra a saída (*output*) de diversos algoritmos de agrupamento, técnicas de validação e *ensemble* de agrupamentos em uma abordagem multiobjetivo, para encontrar um conjunto de estruturas que podem conter informações relevantes para os especialistas no domínio dos dados.

O algoritmo MOCLE, como qualquer *ensemble*, pode ser dividido em dois blocos:

- Geração de um conjunto diverso de partições iniciais a serem combinadas;
- Determinação do consenso.

O MOCLE difere dos *ensembles* tradicionais em dois aspectos, relacionados à obtenção do consenso. Em primeiro lugar, o MOCLE busca por um conjunto de partições consenso, em lugar de uma única partição. Na verdade, o conjunto de soluções que o MOCLE retorna pode conter tanto partições que resultam da combinação de outras partições quanto partições de alta qualidade que já apareciam entre as partições iniciais. A segunda diferença do MOCLE em relação aos demais *ensembles* é que ele combina pares de partições, iterativamente, em um processo de otimização que garante diferentes compromissos de qualidade das soluções. Com isso, o MOCLE consegue evitar a influência negativa das partições iniciais de baixa qualidade que afeta as abordagens tradicionais de *ensemble*.

Mais precisamente, o MOCLE deve ser iniciado com a geração de um conjunto de partições iniciais por meio da aplicação de vários algoritmos de agrupamento conceitualmente diferentes aos dados, também considerando várias configurações de parâmetros. Isso garante a diversidade das partições iniciais do *ensemble*. Em seguida, essas partições iniciais são utilizadas como população inicial para um algoritmo genético multiobjetivo baseado em Pareto. Esse algoritmo vai selecionar e combinar as partições iniciais por meio de duas características particulares: (1) um operador de recombinação especial, que encontra o consenso entre duas partições pais, e (2) a otimização de funções objetivo que representam diferentes medidas de qualidade de uma partição.

O operador de recombinação fornece a característica de *ensemble* ao MOCLE, o que o diferencia da abordagem de agrupamento multiobjetivo pura.

Com essas características, o MOCLE faz uma seleção automática das partições mais significativas, dentre as iniciais e as combinações, sem que sejam necessários muitos ajustes de parâmetros e nem conhecimento profundo em análise de agrupamento. Com isso, ele supera algumas das dificuldades da análise de agrupamento tradicional. Além disso,

a integração das abordagens de *ensemble* e agrupamento multiobjetivo permite superar algumas das dificuldades individuais de ambas as abordagens.

Em resumo, o MOCLE constitui uma abordagem robusta para lidar com diferentes tipos de estrutura que podem estar presentes nos dados, fornecendo como resultado um conjunto conciso e estável de estruturas alternativas de elevada qualidade, sem a necessidade de conhecimento prévio dos dados e nem conhecimento profundo em análise de agrupamento.

O MCHPF, por sua vez, utiliza programação genética para evoluir uma população de *ensembles* e identificar as melhores maneiras de combinar subconjuntos de partições a fim de obter partições relevantes. Nessa abordagem, um conjunto de partições base é inicialmente gerado, da mesma maneira que no MOCLE. Com essas partições, são criados os *ensembles* iniciais, que são os indivíduos da população inicial. Cada *ensemble* é uma hierarquia de fusão gerada aleatoriamente, em que uma ou mais funções consenso tradicionais (como as descritas na Seção 14.1) são aplicadas a um subconjunto das partições base. A população de *ensembles* passa então pelo processo de evolução considerando uma estratégia multiobjetivo, em que são otimizadas as mesmas funções objetivo do MOCLE. Os *ensembles* resultantes do processo de evolução podem, posteriormente, ser executados para a geração das partições consenso. As principais vantagens do MCHPF em relação ao MOCLE se devem ao uso das hierarquias de fusão como soluções, em vez das partições propriamente ditas, aliado ao uso de diferentes funções consenso para a construção de um *ensemble*. Isso permite que as combinações sejam feitas de diferentes maneiras e que se tenha conhecimento das combinações feitas em cada *ensemble*, garantindo a identificação das partições que contribuíram no resultado final, bem como da maneira como as partições foram combinadas.

14.4 Considerações Finais

Neste capítulo, apresentamos diversas abordagens de modelos múltiplos descritivos, mais especificamente relacionados com o tema de análise de agrupamentos. As abordagens apresentadas permitem superar algumas dos problemas tradicionais da análise de agrupamentos. Mais especificamente, foram apresentados os *ensembles* de agrupamentos, os algoritmos de agrupamento multiobjetivo e abordagens que combinam ambas as ideias. Os *ensembles* de agrupamentos atuam por meio da combinação de um conjunto de agrupamentos obtidos previamente, enquanto os algoritmos multiobjetivo realizam a otimização direta de mais de um critério de agrupamento, resultando em um conjunto de soluções. Os *ensembles* são mais robustos e fornecem soluções de melhor qualidade que os algoritmos tradicionais de agrupamento, porém não exploram todo o potencial de se usar múltiplos critérios. Já os algoritmos de agrupamento e os *ensembles* multiobjetivo têm uma maior facilidade para encontrar diferentes tipos de *clusters*, bem como para lidar com estruturas heterogêneas.

Capítulo 15

Avaliação de Modelos Descritivos

A análise e a comparação de resultados em análise de agrupamento podem ser consideradas sob o ponto de vista de dois objetivos diferentes: avaliação e comparação de algoritmos de agrupamento e validação das estruturas encontradas por algoritmos de agrupamento.

A avaliação e comparação de estratégias de agrupamento, podem envolver tanto algoritmos individuais quanto modelos múltiplos. É preciso garantir a validade e a reprodutibilidade das experiências realizadas e a validade das conclusões obtidas a partir de seus resultados.

Na análise exploratória, uma análise prévia dos dados pode ser de grande utilidade para guiar as escolhas feitas em todas as etapas do processo de agrupamento, previamente descritas no Capítulo 12. Tal análise pode ser feita com técnicas da Estatística Descritiva, como as apresentadas no Capítulo 2, ou com técnicas de visualização dos dados. Entretanto, mesmo feitas as escolhas adequadas, é preciso verificar a validade das estruturas encontradas.

Para uma melhor compreensão das discussões apresentadas nas seções seguintes, é importante entender a diferença entre a validação feita para avaliar/comparar algoritmos de agrupamento e aquela feita para validar as estruturas encontradas na análise exploratória dos dados. Levando em conta que a análise de agrupamento é uma tarefa não supervisionada, deve-se ter sempre em mente que não existe um resultado correto, que seja o objetivo final a ser atingido com qualquer algoritmo de agrupamento. Isso é especialmente importante na análise exploratória. Entretanto, na avaliação/comparação de algoritmos, principalmente na avaliação da efetividade de algoritmos novos, é importante ter mecanismos para identificar se o algoritmo está realmente encontrando uma estrutura apropriada, ou se é comparável a outros existentes. Nesses casos, muitas vezes são utilizados dados para os quais se conhecem uma ou mais estruturas, e os algoritmos são avaliados com respeito à sua habilidade em encontrar essas estruturas conhecidas.

Um outro aspecto importante que é preciso considerar quando se avaliam agrupamentos é que nem sempre existe uma solução única, considerando as possíveis definições do que seja um *cluster*, as diferentes maneiras como dois objetos podem ser considerados similares e a possibilidade de existência de várias estruturas, quer homogéneas quer heterogéneas, em um conjunto de dados. Estas características tornam a avaliação dos resultados

em agrupamento uma tarefa complexa, pois não existe uma resposta esperada com a qual comparar os resultados obtidos pelos algoritmos, e, muitas vezes, não existe uma resposta única.

De qualquer maneira, é preciso seguir procedimentos rigorosos, considerando todos os aspectos descritos previamente no Capítulo 12, para garantir resultados efetivos e úteis, e também a reprodutibilidade das análises. Como já mencionado, a avaliação do resultado de um agrupamento deve ser objetiva, visando determinar se a estrutura encontrada é válida, ou seja, se não ocorreu por acaso, ou se é *rara* em algum sentido, já que qualquer algoritmo de agrupamento encontrará *clusters*, independentemente de se existe ou não estrutura nos dados. Entretanto, mesmo que essa estrutura exista, alguns algoritmos podem encontrar *clusters* mais adequados que outros. Diversas técnicas para a validação em análise de agrupamento têm sido discutidas na literatura. Alguns desses estudos podem ser encontrados em Jain e Dubes (1988); Gordon (1999); Halkidi et al. (2001); Handl et al. (2005).

A validação do resultado de um agrupamento, em geral, é baseada em índices estatísticos, que julgam, de uma maneira qualitativa, o mérito das estruturas encontradas. Um índice quantifica alguma informação a respeito da qualidade de um agrupamento. A maneira pela qual um índice é aplicado para validar um agrupamento é dada pelo critério de validação. Assim, um critério de validação expressa a estratégia utilizada para validar uma estrutura de agrupamento, enquanto um índice é uma estatística pela qual a validade é testada. Existem três tipos de critérios para investigar a validade de um agrupamento: critérios internos, externos e relativos. A Seção 15.1 descreve detalhadamente cada um desses critérios. Neste livro, os índices que são mais empregues em critérios externos, internos e relativos serão denominados simplificadamente índices externos, internos e relativos, respectivamente. Posteriormente, nas Seções 15.2, 15.3 e 15.4, serão detalhados alguns dos índices empregados, respectivamente, com critérios relativos, internos e externos, bem como as metodologias mais frequentemente empregadas para a aplicação desses índices.

15.1 Critérios de Validação

Conforme já dito, um critério de validação expressa a estratégia utilizada para validar um agrupamento, enquanto um índice é uma estatística pela qual a validade é testada. Ou, de outra forma, o critério de validação indica a maneira pela qual um índice é aplicado para validar um agrupamento. Existem três tipos de critérios para investigar a validade de um agrupamento:

- **Critérios relativos**: comparam diversos agrupamentos com respeito a algum aspecto (qual é mais o estável ou qual é o mais adequado aos dados, por exemplo). Podem ser utilizados para comparar diversos algoritmos de agrupamento ou para determinar o valor mais apropriado para um ou mais parâmetros de um algoritmo, como o número de *clusters*. Com isso, esses critérios permitem medir quantitati-

vamente qual dentre dois algoritmos melhor se ajusta aos dados ou determinar o número de *clusters* mais apropriado para um agrupamento produzido por um determinado algoritmo.

- **Critérios internos**: medem a qualidade de um agrupamento com base apenas nos dados originais (matriz de objetos ou matriz de similaridade). Por exemplo, um critério interno pode medir o grau com que uma partição obtida por um algoritmo de agrupamento é justificada pela matriz de similaridade.
- **Critérios externos**: avaliam um agrupamento de acordo com uma estrutura estabelecida previamente, que pode refletir, por exemplo, a intuição do pesquisador sobre a estrutura presente nos dados. Essa estrutura pré-especificada pode ser uma partição previamente conhecida para os dados ou um agrupamento sugerido por um especialista da área baseado em conhecimento prévio. Por exemplo, um critério externo pode medir o grau de correspondência entre o número *clusters* obtido com o agrupamento e rótulos já conhecidos para os dados.

Estes critérios de validação podem ser utilizados para avaliar vários tipos de estrutura como hierarquias, partições (*hard* ou *fuzzy*) e *clusters* individuais. A seguir, serão discutidas as principais abordagens para utilização dos três critérios no contexto de avaliação de partições.

Os critérios relativos são utilizados para encontrar o melhor algoritmo de agrupamento (e/ou conjunto de valores de parâmetros para o mesmo algoritmo) em relação a um grupo de outros algoritmos (e/ou conjuntos de valores de parâmetros diferentes para o mesmo algoritmo).

Existem vários índices que podem ser empregados com critérios relativos. Alguns dos índices mais comuns, assim como alguns propostos recentemente, são apresentados na Seção 15.2.1. Esses índices, em geral, podem ainda ser empregados em critérios internos (Jain e Dubes, 1988). O que distingue a utilização de um índice em um ou outro critério é a maneira como o índice é aplicado. A forma mais comum de aplicação de um índice com um critério relativo é o cálculo do seu valor para vários agrupamentos que estão sendo comparados, obtendo-se uma sequência de valores. O melhor agrupamento é determinado pelo valor que se destaca nessa sequência, como um valor máximo, mínimo ou uma inflexão na curva do gráfico construído com a sequência.

A Figura 15.1 resume a metodologia mais utilizada para aplicação do critério relativo de validação. Como indica a Figura 15.1, nesse critério, vários algoritmos, ou um mesmo algoritmo com diferentes valores para seus parâmetros, são aplicados ao mesmo conjunto de dados. O índice é calculado para cada uma das partições obtidas. Esses valores do índice são comparados, em geral com o auxílio de um gráfico, para determinar o melhor algoritmo ou o melhor valor para um ou mais parâmetros de um algoritmo.

Além da forma previamente descrita de validação relativa, outras abordagens têm sido exploradas na literatura. Algumas dessas abordagens serão descritas na Seção 15.2.2. Serão detalhadas a análise de replicação, proposta por McIntyre e Blashfield (1980) e Morey et al. (1983), similar à validação cruzada previamente descrita no Capítulo 9, a abordagem

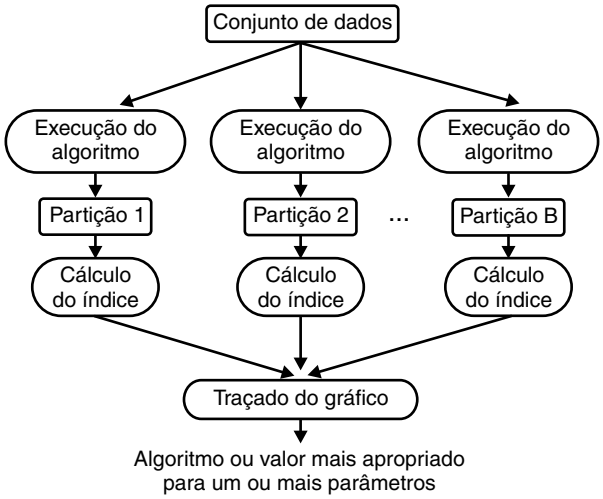


Figura 15.1 *Critério relativo de validação.*

de Law e Jain (2003), que calcula a variabilidade do algoritmo utilizando *bootstrapping*, e as abordagens de Yeung et al. (2001) e Tibshirani et al. (2001a), que se baseiam no poder preditivo do algoritmo.

Os critérios externos e internos de validação são baseados em testes estatísticos e têm um elevado custo computacional (Halkidi et al., 2001). O seu objetivo é medir o quanto o resultado obtido confirma uma hipótese pré-especificada. Nesse caso, são utilizados testes de hipótese para determinar se uma estrutura obtida é apropriada para os dados, o que é feito verificando se o valor do índice utilizado é extraordinariamente grande ou pequeno. Isso requer o estabelecimento de uma população base ou de referência. O mesmo índice pode ser utilizado em um critério externo e interno, embora as distribuições de referência do índice sejam diferentes (Jain e Dubes, 1988).

A proposição de um índice para validação é fácil, porém é muito difícil estabelecer limiares que permitam afirmar que o valor do índice é grande ou pequeno o suficiente para se considerar o agrupamento "raro" e potencialmente útil ou válido.

Os índices de validação são funções dos dados que contêm informações úteis, como o erro quadrático de um agrupamento ou a compactação de seus *clusters*. É importante observar que um índice é uma variável aleatória. Sua distribuição descreve a frequência relativa com a qual seus valores são gerados sob alguma hipótese. Uma hipótese é uma afirmação sobre a frequência relativa de eventos no espaço amostral que expressa um conceito.

No caso de validação de agrupamentos, a hipótese nula, H_0, é uma afirmação de aleatoriedade ou falta de estrutura nos dados. Jain e Dubes (1988) e Gordon (1999) descrevem algumas hipóteses nulas utilizadas para a validação de agrupamentos. Jain e Dubes (1988)

discutem ainda aplicações de cada uma dessas hipóteses. A seleção da hipótese nula depende do tipo dos dados e do aspecto que está sendo analisado sobre os dados.

Jain e Dubes (1988) resumem os principais aspectos relacionados à utilização de um índice de validação:

- **Definição do índice**: o índice deve fazer sentido intuitivamente, deve ter uma base teórica e deve ser prontamente computável.
- **Distribuição de probabilidade-base**: uma distribuição-base é uma distribuição derivada de uma população que não possui estrutura. Uma população referência é definida ou implicada pela distribuição-base.
- **Teste para verificar estrutura não aleatória**: o valor de um índice de validação é comparado a um limiar que estabelece um dado nível de significância. O limiar é definido a partir da distribuição-base, que, em teoria, raramente é conhecida.
- **Teste para verificar um tipo de estrutura**: a habilidade do índice de validação em recuperar uma estrutura conhecida indica seu poder estatístico. A escolha da estrutura depende da aplicação.

As Figuras 15.2 e 15.3 resumem, respectivamente, os critérios externos e internos de validação. Conforme dito anteriormente, nesses critérios é utilizado um teste de hipótese que depende da distribuição do índice sob uma hipótese nula, H_0. A diferença entre esses critérios está nas informações utilizadas. Nos critérios externos, pode ser utilizada uma partição dos dados conhecida previamente, chamada de "partição real" ou "estrutura conhecida", no cálculo do índice. Já nos critérios internos, o cálculo do índice depende somente dos próprios dados, seja na forma de matriz de objetos (conjunto de dados original) ou de matriz de similaridade.

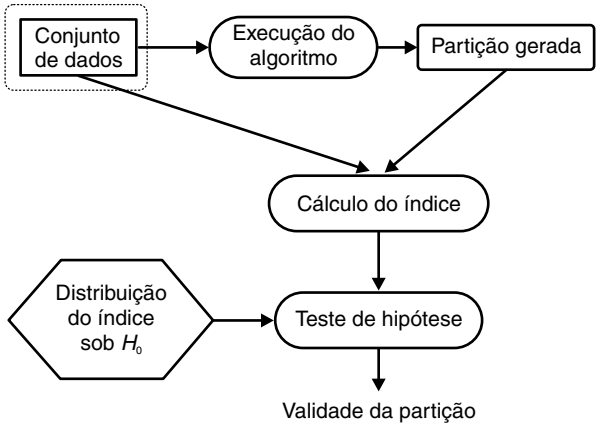


Figura 15.2 *Critério interno de validação.*

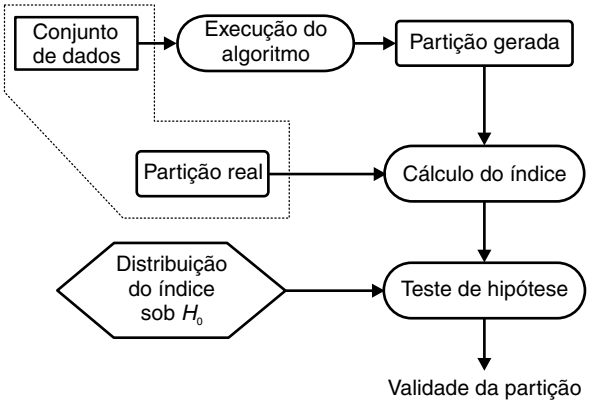


Figura 15.3 *Critério externo de validação.*

Um grande problema relacionado à validação externa e interna de agrupamentos é o estabelecimento da distribuição dos índices (estatísticas) sob a hipótese nula e, consequentemente, a determinação dos limiares que informam se uma partição é adequada de acordo com o índice. Assim, na prática, os testes de validação são geralmente definidos utilizando ferramentas estatísticas, como análise de Monte Carlo e *bootstrapping*.

Análise de Monte Carlo é um método para estimar parâmetros e taxas de probabilidade por meio de amostragem por computador. É utilizada quando tais valores são difíceis ou impossíveis de ser calculados diretamente.

Umas das formas mais comuns de utilização de análise de Monte Carlo para validação de agrupamentos é no estabelecimento da distribuição referência de um índice sob a hipótese nula. Inicialmente, é gerada uma grande quantidade de conjuntos de dados sintéticos de acordo com a distribuição considerada na hipótese nula H_0. Em seguida, cada um desses conjuntos é agrupado e o valor do índice é calculado em cada caso. Com esses valores do índice, é traçado um gráfico de dispersão, que é uma aproximação da função de densidade de probabilidade do índice. Dado o valor do índice para o agrupamento que está sendo validado e a distribuição estimada, determina-se se a hipótese H_0 deve ser aceita ou rejeitada. Com a utilização da análise de Monte Carlo, o limiar para determinar se o valor do índice é grande ou pequeno o suficiente (ou seja, rejeitar H_0) pode ser visto como um intervalo de valores, ao invés do valor único obtido quando a distribuição sob H_0 é conhecida (Jain e Dubes, 1988). A Figura 15.4 resume essa forma de aplicação de análise de Monte Carlo para validação de agrupamentos.

Não é fácil definir o modelo nulo. Existe uma grande quantidade de tipos de modelos nulos, cada um com suas vantagens e desvantagens (Gordon, 1996). Segundo Filho (2003), Gordon (1996) sugere a utilização de mais de um modelo nulo, o que torna o processo de validação ainda mais complexo e demorado. Segundo Jain e Dubes (1988), a parte mais difícil na análise de Monte Carlo é a obtenção de uma amostra de distribuição arbitrária. Geralmente existem geradores de números aleatórios que geram amostras pseudoaleatórias

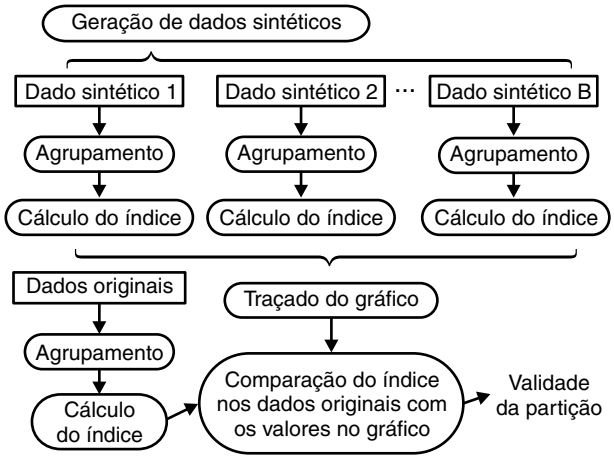


Figura 15.4 *Análise de Monte Carlo para validação de um agrupamento.*

de distribuição $U(1,0)$ e existem técnicas para transformar essas amostras uniformes em amostras de outras distribuições.

Outro problema da análise de Monte Carlo é a necessidade de uma grande quantidade de recursos computacionais, uma vez que é necessário um grande número de replicações para construir a distribuição de referência. Entretanto, com os avanços tecnológicos recentes, esse problema é cada vez menor.

As técnicas de *bootstrapping* utilizam reamostragem dos dados, com substituição, para criar um conjunto 'falso' de dados, que simula uma replicação de um experimento de Monte Carlo (Jain e Dubes, 1988). Nesse caso, as amostras *bootstrap* são utilizadas para construir o modelo nulo. As amostras podem ser obtidas pela reamostragem dos objetos ou dos atributos do conjunto de dados. Com *bootstrapping*, evitam-se os problemas da análise de Monte Carlo relacionados à escolha do modelo nulo. Existem outras formas de aplicação de *bootstrapping* em validação, algumas das quais são empregadas em índices descritos mais adiante.

15.2 Critérios Relativos

Conforme já foi dito, o objetivo da validação relativa é encontrar um agrupamento que melhor se ajuste aos dados, a partir de vários agrupamentos obtidos com um algoritmo sob certas suposições e valores para seus parâmetros, ou encontrar o algoritmo mais apropriado para agrupar os dados analisados de forma a encontrar as estruturas desejadas.

Uma das formas mais comuns de utilização dos critérios relativos é na determinação do número mais adequado de *clusters*. Nesse caso, o algoritmo de agrupamento é executado para todos os possíveis números de *clusters*, k, entre k_{min} e k_{max}, fornecidos. Em seguida,

os valores do índice obtidos a partir dessas execuções são apresentados, na forma de um gráfico, como função de k. O melhor número de *clusters* é dado pelo mínimo, máximo ou inflexão na curva observada. Halkidi et al. (2002b) descrevem resumidamente a utilização de índices em critérios relativos para determinar outros parâmetros de algoritmos de agrupamento.

Um grande número de índices que podem ser empregados com critérios relativos pode ser encontrado na literatura da área. A título de ilustração, a Seção 15.2.1 apresenta alguns desses índices.

Além da aplicação tradicional da validação relativa (cálculo de um índice para várias partições e observação do gráfico do índice em função de k), existem algumas outras abordagens descritas na literatura, algumas das quais serão apresentadas na Seção 15.2.2.

15.2.1 Índices Empregados em Critérios Relativos

Variância Intracluster, $var(\pi)$

A variância intracluster de uma partição π é dada pela Equação 15.1, em que $\bar{\mathbf{x}}^{(k)}$ é o centroide do *cluster* (Handl et al., 2005). Ela mede a qualidade de um agrupamento em termos da compactação de seus *clusters*. Essa medida apresenta valores no intervalo $[0,\infty]$, e quanto menor o valor de *var*, melhor a partição.

$$var(\pi) = \sqrt{\frac{1}{n}\sum_{\mathbf{C}_k\in\pi}\sum_{\mathbf{x}_i\in\mathbf{C}_k} d(\mathbf{x}_i,\bar{\mathbf{x}}^{(k)})} \tag{15.1}$$

Conectividade, $con(\pi)$

A conectividade está ligada ao conceito de encadeamento e reflete o grau com que os objetos vizinhos são colocados no mesmo *cluster*. Os vizinhos mais próximos são aqueles com menor distância (ou maior similaridade) (Handl et al., 2005). A conectividade de uma partição é dada pela Equação 15.2, em que v é o número de vizinhos mais próximos que contribuem para a conectividade e nn_{ij} é o j-ésimo vizinho mais próximo ao objeto $\mathbf{x}_i$. Quanto menor o valor do índice *con*, melhor a partição. Os valores desse índice variam no intervalo $[0,v]$.

$$con(\pi) = \sum_{\mathbf{x}_i\in\mathbf{X}}\sum_{j=1}^{v} f(\mathbf{x}_i,nn_{ij}) \tag{15.2}$$

$$f(\mathbf{x}_i,nn_{ij}) = \begin{cases} 1/j & \text{se } \mathbf{x}_i\in\mathbf{C}_k, nn_{ij}\notin\mathbf{C}_k \\ 0 & \text{caso contrário} \end{cases} \tag{15.3}$$

Família de Índices Dunn, $D(\pi)$

A família de índices Dunn é representada pela Equação 15.4, em que $d(\mathbf{C}_a,\mathbf{C}_b)$ é uma função de distância entre os *clusters* $\mathbf{C}_a$ e $\mathbf{C}_b$ e $d(\mathbf{C}_a)$ é a distância intracluster do

cluster $\mathbf{C}_a$, que mede a dispersão do *cluster* (Halkidi et al., 2002b). No índice Dunn original, $d(\mathbf{C}_a, \mathbf{C}_b)$ é dada pela Equação 15.5, que é a mesma medida usada pelo algoritmo hierárquico com ligação simples, e $d(\mathbf{C}_a)$ é dada pela Equação 15.6. Nesse caso, o índice mede a razão da separação dentro dos *clusters* e entre os *clusters* (Pakhira et al., 2004). Outras variações do índice consideram diferente funções para $d(\mathbf{C}_a, \mathbf{C}_b)$ e $d(\mathbf{C}_a)$.

$$D(\pi) = \min_{a=1,\ldots,k} \left\{ \min_{b=a+1,\ldots,k} \left\{ \frac{d(\mathbf{C}_a, \mathbf{C}_b)}{\max_{l=1,\ldots,k} d(\mathbf{C}_l)} \right\} \right\} \tag{15.4}$$

$$d(\mathbf{C}_a, \mathbf{C}_b) = \min_{\substack{\mathbf{x}_i \in \mathbf{C}_a, \\ \mathbf{x}_j \in \mathbf{C}_b}} d(\mathbf{x}_i, \mathbf{x}_j) \tag{15.5}$$

$$d(\mathbf{C}_a) = \max_{\mathbf{x}_i, \mathbf{x}_j \in \mathbf{C}_a} d(\mathbf{x}_i, \mathbf{x}_j) \tag{15.6}$$

O índice Dunn é apropriado para a identificação de *clusters* compactos e bem separados (Halkidi et al., 2002b). Valores altos do índice indicam a presença desse tipo de *cluster*. O índice $D(\pi)$ não apresenta nenhuma tendência em relação ao número de *clusters*. O ponto de máximo no gráfico de $D(\pi)$ contra k, em que diversas partições π foram obtidas com esses valores de k, pode ser uma indicação do número de *clusters* que mais se ajusta aos dados.

Os problemas do índice Dunn original são a sua complexidade e sensibilidade a ruído. Outros índices da família Dunn são mais robustos à presença de ruídos. Bezdek e Pal (1998) propõem várias generalizações desse índice e as comparam com outros índices. Azuaje (2002) apresenta uma ferramenta que emprega 18 índices baseados no índice Dunn para determinar o melhor número de *clusters*. Esses índices são baseados em diferentes combinações de diferentes distâncias entre *clusters* e intracluster. Esse índice não é apropriado para *clusters* de formas arbitrárias.

Silhueta, $sil(\pi)$

A medida silhueta se baseia na proximidade entre os objetos de um *cluster* e na distância dos objetos de um *cluster* ao *cluster* mais próximo (Rousseeuw, 1987). Ela pode ser utilizada para avaliar uma partição; para isso, avalia também a adequação de cada objeto ao seu *cluster* e a qualidade de cada *cluster* individualmente. O valor de uma medida silhueta é limitado pelo intervalo $[-1, 1]$, em que a melhor partição de acordo com a silhueta é aquela com valor 1.

As silhuetas de um *cluster* e de uma partição para medidas de distância entre objetos são dadas pelas Equações 15.10 e 15.11, respectivamente, em que n é o número de objetos agrupados e $sil(\mathbf{x}_i)$ é a silhueta do objeto $\mathbf{x}_i$, dada pela Equação 15.7. Nessa equação, $a(\mathbf{x}_i, \mathbf{C}_i)$, definida pela Equação 15.8, representa a distância média do objeto $\mathbf{x}_i$ em relação a todos os outros objetos do *cluster* $\mathbf{C}_i$ (*cluster* ao qual o objeto $\mathbf{x}_i$ pertence) e $b(\mathbf{x}_i)$, dada pela Equação 15.9, a menor distância média de $\mathbf{x}_i$ em relação a todos os demais

clusters. Rousseeuw (1987) apresenta uma versão do índice para quando uma medida de similaridade é utilizada, em vez de uma medida de distância.

$$sil(\mathbf{x}_i) = \begin{cases} 1 - a(\mathbf{x}_i, \mathbf{C}_i)/b(\mathbf{x}_i), & a(\mathbf{x}_i, \mathbf{C}_i) < \mathbf{b}(\mathbf{x}_i) \\ 0, & a(\mathbf{x}_i, \mathbf{C}_i) = \mathbf{b}(\mathbf{x}_i) \\ b(\mathbf{x}_i)/a(\mathbf{x}_i, \mathbf{C}_i) - 1, & a(\mathbf{x}_i, \mathbf{C}_i) > \mathbf{b}(\mathbf{x}_i) \end{cases} \tag{15.7}$$

$$a(\mathbf{x}_i, \mathbf{C}_k) = \frac{1}{|\mathbf{C}_k|} \sum_{\substack{\mathbf{x}_i, \mathbf{x}_j \in \mathbf{C}_k \\ \mathbf{x}_i \neq \mathbf{x}_j}} d(\mathbf{x}_i, \mathbf{x}_j) \tag{15.8}$$

$$b(\mathbf{x}_i) = \min_{\substack{\mathbf{x}_i \in \mathbf{C}_i, \\ \mathbf{C}_i \neq \mathbf{C}_j}} a(\mathbf{x}_i, \mathbf{C}_j) \tag{15.9}$$

Quando a silhueta é calculada para cada objeto, seu valor será próximo de 1 se o objeto está bem situado dentro do seu *cluster*. Um valor próximo de -1 indica que o objeto deveria ser associado a outro *cluster* (Yeung, 2001).

A medida silhuetas depende apenas da partição dos dados, independente portanto do algoritmo de agrupamento empregado. Ela pode ser utilizada para melhorar os resultados de uma análise de *cluster* ou para comparar os resultados de diferentes algoritmos aplicados ao mesmo conjunto de dados.

Além da silhueta de cada objeto, pode ser calculada a silhueta de cada *cluster*, conforme a Equação 15.10 e a largura média da silhueta, $sil(\pi)$, utilizando a Equação 15.11. Um modo de escolher o melhor valor de k é selecionar aquele que resulta no maior valor de $sil(\pi)$.

$$sil(\mathbf{C}_k) = \frac{1}{|\mathbf{C}_k|} \sum_{\mathbf{x}_i \in \mathbf{C}_k} sil(\mathbf{x}_i) \tag{15.10}$$

$$sil(\pi) = \frac{1}{n} \sum_{i=1}^{n} sil(\mathbf{x}_i) \tag{15.11}$$

O coeficiente silhueta, *SC*, é o máximo $sil(\pi)$ para π gerada com $k = 2, 3, ...(n-1)$. *SC* é uma medida que quantifica a estrutura descoberta por um algoritmo de agrupamento. Uma interpretação para *SC* é: $SC \leq 0,25$ significa que não foi encontrada uma estrutura substancial, $0,26 \leq SC \leq 0,5$ indica que a estrutura encontrada é fraca e pode ser artificial, $0,5 \leq SC \leq 0,7$ significa que uma estrutura razoável foi encontrada, e $0,71 \leq SC \leq 1$ indica que foi encontrada uma estrutura forte.

As medidas de silhueta são apropriadas para a identificação de *clusters* compactos e bem separados. Assim, funcionam melhor com *clusters* aproximadamente esféricos (Rousseeuw, 1987). Uma vez que a definição da silhueta favorece objetos que estão associados a *clusters* com a similaridade média mais alta, a largura da silhueta é tendenciosa contra *clusters* verdadeiros potencialmente sobrepostos, favorecendo agrupamentos disjuntos.

Existem outros índices para avaliação de partições que, para tornar o texto mais coeso, não serão discutidos em detalhes, como o desvio total (*dev*) (Handl e Knowles, 2007), o índice Davies-Bouldin (*DB*) (Jain e Dubes, 1988), o índice de Calinski-Harabasz (*CH*) (Calinski e Harabasz, 1974), o índice de Krzanowski e Lai (*KL*) (Krzanowski e Lai, 1985), o índice de Hartigan (*H*) (Hartigan, 1975), o índice *S_Dbw* (do inglês *compose density between and within clusters*) (Halkidi e Vazirgiannis, 2001) e o índice de Pakhira et al. (2004), *PBM*. Descrições e comparações entre uma variedade grande de índices de validação podem ser encontradas em Milligan e Cooper (1985) e Vendramin et al. (2010).

Além desses índices, existem vários outros particularmente indicados para a avaliação de partições *fuzzy*. Nesse tipo de agrupamento, cada objeto apresenta um grau de pertinência em relação a cada *cluster*, em vez de ser associado unicamente a um *cluster*. Alguns dos índices mais comuns empregados para validação de agrupamentos *fuzzy* são: coeficiente de partição (*PC*) (Pal e Bezdek, 1995), entropia da partição (*PE*) (Pal e Bezdek, 1995), índice de Xie-Beni (*XB*) (Xie e Beni, 1991; Pal e Bezdek, 1995), índice de Xie-Beni estendido (Pal e Bezdek, 1995), índice de Fukuyama-Sugeno (*FS*) (Pal e Bezdek, 1995), separação da partição (*S*) (Yang e Wu, 2001) e índice de Pakhira et al. (2004) fuzzificado, *PBMF*.

Dentre os índices citados para validação de partições *fuzzy*, os índices *PC* e *PE* utilizam apenas o valor de pertinência dos objetos aos *clusters* para calcular seu valor. Já os demais índices utilizam, além do valor de pertinência, o próprio conjunto de dados.

15.2.2 Outras Abordagens de Validação Relativa

Nesta seção são descritas outras abordagens para validação relativa, em particular a análise de replicação, proposta por McIntyre e Blashfield (1980) e Morey et al. (1983), a abordagem de Law e Jain (2003), que calcula a variabilidade do algoritmo utilizando *bootstrapping*, e a abordagem de Yeung et al. (2001), que se baseia no poder preditivo do algoritmo. Tibshirani et al. (2001a) também apresentam uma abordagem baseada no poder preditivo, que não será detalhada.

A análise de replicação (McIntyre e Blashfield, 1980; Morey et al., 1983) se baseia em um argumento semelhante ao procedimento de *cross-validation* comumente usado em aprendizado supervisionado, descrito no Capítulo 9. A ideia é medir a estabilidade (ou replicabilidade) de um algoritmo de agrupamento. Essa estabilidade é medida comparando a partição obtida pelo algoritmo a uma partição obtida em um subconjunto independente dos dados.

A forma de validação de um agrupamento proposta por Law e Jain (2003) é fundamentada na interpretação de um algoritmo de agrupamento como um estimador estatístico e na avaliação da variabilidade desse estimador por meio de *bootstrapping*. Se uma partição é válida, sua variabilidade deve ser baixa.

As abordagens de Yeung et al. (2001) e Tibshirani et al. (2001a) seguem o princípio de que, se um agrupamento reflete uma estrutura verdadeira, um preditor construído com base nos *clusters* desse agrupamento deve estimar precisamente os rótulos dos *clusters* de novas amostras de teste (Jiang et al., 2004).

Análise de Replicação

A análise de replicação (McIntyre e Blashfield, 1980; Morey et al., 1983; Milligan, 1996) é ilustrada na Figura 15.5 e pode ser resumidamente descrita pelo Algoritmo 15.1. A medida de concordância mencionada na linha 10 pode ser o índice Rand corrigido (Hubert e Arabie, 1985), que será descrito na Seção 15.4. O nível de concordância entre as duas partições reflete a estabilidade do algoritmo de agrupamento.

Algoritmo 15.1 Análise de replicação

Entrada: Um conjunto de dados $\mathbf{X}$
Um algoritmo de agrupamento ϕ
Saída: Estabilidade do algoritmo ϕ

1 Obter duas amostras dos dados, $\mathbf{X_1}$ e $\mathbf{X}_2$, dividindo o conjunto de dados original em dois subconjuntos
2 Aplicar o algoritmo de agrupamento ϕ na amostra $\mathbf{X_1}$, obtendo uma partição π^1
3 Aplicar o algoritmo de agrupamento ϕ na amostra $\mathbf{X_2}$, obtendo uma partição π^2
4 Determinar os centroides dos *clusters* da partição π^1
5 Construir π'_2, agrupando $\mathbf{X}_2$ com base nas características de $\mathbf{X_1}$:
6 Assumir os mesmos centroides de π^1 para π'_2
7 **para cada** $\mathbf{x} \in \mathbf{X_2}$ **faça**
8 Determinar as distâncias entre o objeto $\mathbf{x}$ e os centroides dos *clusters* de π'_2
9 Associar $\mathbf{x}$ ao *cluster* cujo centroide é mais próximo
`// Isso resulta em um agrupamento da segunda amostra com base nas características da primeira`
10 **fim**
11 Calcular uma medida de concordância entre os dois agrupamentos da segunda amostra, π^2 e π'_2

Figura de Mérito

Yeung et al. (2001) propõem uma metodologia para avaliação da qualidade de um agrupamento com base no poder preditivo do algoritmo que o gerou. O poder preditivo é dado pela medida figura de mérito, *FOM* (do inglês *Figure of Merit*), definida pelos autores. A *FOM* foi proposta para o agrupamento de genes utilizando seu nível de expressão medido em diversos experimentos. A intuição por trás dessa medida é a tendência de os genes pertencentes a um mesmo *cluster* terem um nível de expressão similar em experimentos adicionais, não empregados na geração dos *clusters*, se o agrupamento for significativo do ponto de vista biológico.

Um algoritmo de agrupamento é aplicado a todos os atributos, exceto um atributo,

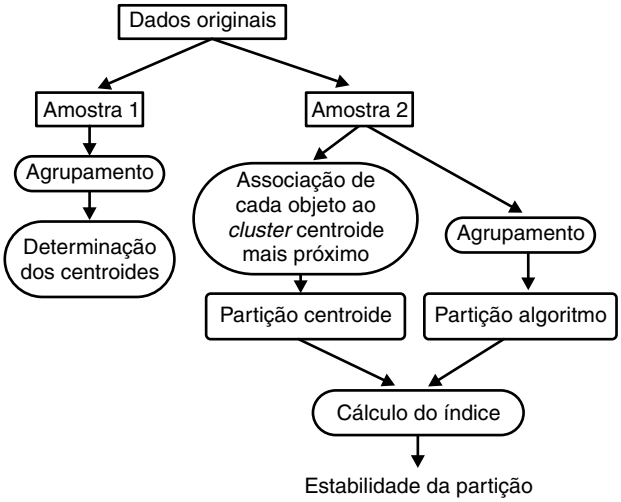


Figura 15.5 *Análise de replicação para validação de um agrupamento.*

denominado a. Esse atributo a é utilizado para estimar o poder preditivo do algoritmo, por meio da variância dentro dos *clusters* dada pela FOM. *Clusters* significativos devem ter menos variação nos outros experimentos do que *clusters* gerados ao acaso. Assim, quanto maior a similaridade intracluster calculada a partir da exclusão do atributo a, mais forte é o poder preditivo e melhor o esquema de agrupamento. Sejam x_i^a o valor do atributo a para o objeto $\mathbf{x}_i$ na matriz de dados bruta e $\bar{x}^a(\mathbf{x}^a, \mathbf{C}_l)$ a média dos valores do atributo $\mathbf{x}^a$ nos objetos pertencentes ao *cluster* $\mathbf{C}_l$. $FOM(\mathbf{x}^a, \pi)$ é dada pela Equação 15.12, para k *clusters* usando o atributo $\mathbf{x}^a$.

$$FOM(\mathbf{x}^a, \pi) = \sqrt{\frac{1}{n} \sum_{l=1}^{k} \sum_{\mathbf{x}_i \in \mathbf{C}_l} (x_i^a - \bar{x}^a(\mathbf{x}^a, \mathbf{C}_l))^2} \tag{15.12}$$

Com a aplicação dessa equação a cada um dos d atributos, obtém-se a FOM agregada, dada pela Equação 15.13, que é uma estimativa do poder preditivo total de um algoritmo sobre todas as amostras para k *clusters*.

$$FOM(\pi) = \sum_{a=1}^{d} FOM(\mathbf{x}^a, \pi) \tag{15.13}$$

O valor da FOM tem a tendência de diminuir com o aumento do número de *clusters*. Para corrigir esse efeito, Yeung et al. (2001) utilizam a FOM ajustada, dada pela Equação 15.14.

$$FOM_{ADJ}(\pi) = \frac{\sum_{a=1}^{d} FOM(\mathbf{x}^a, \pi)}{\sqrt{(n-k)/n}} \tag{15.14}$$

Um valor baixo para as medidas FOM e FOM_{ADJ} indica um algoritmo com um poder preditivo elevado.

A metodologia proposta por Yeung et al. (2001) segue uma abordagem preditiva, ou seja, assume que o atributo excluído contém informação dos experimentos utilizados para produzir os *clusters*. Essa abordagem não é aplicável a todas as situações. Se todos os atributos contêm informações independentes, ela não se aplica. Além disso, não é uma abordagem segura para a comparação de agrupamentos com diferentes números de *clusters* ou obtidos com medidas de similaridade diferentes.

Variabilidade

Law e Jain (2003) propõem uma outra medida de validação para comparar os resultados de diferentes agrupamentos. Essa medida interpreta um algoritmo de agrupamento como um estimador estatístico e utiliza *bootstrapping* para estimar sua variabilidade. Segundo os autores, se uma partição é válida, sua variabilidade deve ser baixa.

Para isso, o algoritmo de agrupamento é considerado um estimador de ponto baseado em uma amostra $\mathbf{X}$ de tamanho n, independente e identicamente distribuída. Assim, o algoritmo é interpretado como um procedimento para estimar a melhor partição do conjunto de dados com base em uma amostra dos dados. O procedimento para o cálculo da variabilidade é apresentado no Algoritmo 15.2. A medida de variabilidade empregada nesse procedimento (linhas 7 e 9) é dada pela Equação 15.15, em $\Pi = \{\pi^1, \pi^2, ..., \pi^B\}$ é um conjunto de B partições e $d(\pi^i, \pi^j)$ é uma medida de distância entre duas partições π^i e π^j. Law e Jain (2003) investigaram cinco medidas de distância diferentes para o cálculo da variabilidade: quatro delas baseadas nos índices Rand, Jaccard, Fowlkes e Malows e Hubert, descritos na Seção 15.4, e a medida de distância proposta em Lange et al. (2003). As quatro primeiras medidas são subtraídas de 1 para obter distâncias, uma vez que são medidas de similaridade entre duas partições.

$$V(\Pi) = \frac{1}{B(B-1)} \sum_{i=1}^{B} \sum_{j=i+1}^{B} d(\pi^i, \pi^j) \tag{15.15}$$

Assim como os algoritmos de AM em geral, todo algoritmo de agrupamento apresenta um viés. Em geral, existe uma explicação para esse viés, e uma partição com baixa variabilidade, mesmo se diferente dos *clusters* reais, provavelmente revela informações úteis sobre os dados (Law e Jain, 2003).

Esse método não identifica explicitamente se um conjunto de dados não apresenta estrutura, mas uma alta variabilidade para diferentes valores de k pode sugerir ausência de estrutura.

Algoritmo 15.2 Variabilidade

Entrada: Um conjunto de dados $\mathbf{X}$
Um algoritmo de agrupamento ϕ
Saída: Variabilidade do algoritmo ϕ

1 Gerar B amostras *bootstrap*, $\mathbf{X}_b$, $b = 1,...,B$, de tamanho n, reamostrando $\mathbf{X}$ com reposição
2 $\Pi_{alg} \leftarrow \emptyset$
3 **para cada** *amostra* $\mathbf{X}_b$ **faça**
4 Aplicar o algoritmo de agrupamento ϕ na amostra, gerando a partição π^b
5 $\Pi_{alg} \leftarrow \Pi_{alg} \cup \pi^b$
6 **fim**
7 Calcular a variabilidade $V_{alg} \leftarrow V(\Pi_{alg})$ de acordo com a Equação 15.15
8 Gerar um conjunto de B partições aleatórias de tamanho n, Π_{ran}
9 Calcular a variabilidade $V_{ran} \leftarrow V(\Pi_{ran})$ de acordo com a Equação 15.15
10 Calcular a variabilidade ajustada $V \leftarrow V_{alg}/V_{ran}$

15.3 Critérios Internos

Os índices internos medem o grau em que uma partição obtida por um algoritmo de agrupamento representa uma estrutura presente nos dados, baseado apenas na matriz de objetos ou na matriz de similaridade. Eles medem o ajuste entre a partição gerada e os dados empregados. Esse tipo de validação, em geral, está relacionado à determinação do número "verdadeiro" de *clusters*.

A aplicação de índices internos para a validação de uma partição apresenta diversas dificuldades (Jain e Dubes, 1988; Halkidi et al., 2001). Uma delas decorre do fato de que os índices mais utilizados, como erro quadrático, provavelmente apresentam valores menores para dados que realmente possuem *clusters* do que para dados aleatórios, sem estrutura. Isso pode levar à conclusão errônea de que uma solução encontrada é válida, mesmo que ela não contenha o número correto de *clusters*. Uma possível solução é a aplicação dos índices internos como índices relativos, conforme já discutido. Uma outra dificuldade é a dependência desses índices dos valores utilizados para as características dos dados, tais como número de objetos, número de dimensões, número de *clusters* e espalhamento (Jain e Dubes, 1988).

Além dos índices relativos descritos anteriormente, que também podem ser empregados como internos, será apresentado, a seguir, o índice *Gap* (Tibshirani et al., 2001b). Um outro índice utilizado é o procedimento *Clest* (Dudoit e Fridlyand, 2002), que, para não tornar o texto muito extenso, não será detalhado neste livro.

15.3.1 Estatística *Gap*

O índice *Gap* (Tibshirani et al., 2001b), dado pela Equação 15.16, avalia a dispersão intracluster em relação à sua esperança sob uma distribuição de referência nula. Nessa equação, o valor de W_k é obtido pela Equação 15.17 e E_n^* é a esperança sob uma amostra de tamanho n para uma distribuição de referência. A variável D_r da Equação 15.17 é, por sua vez, definida pela Equação 15.18, que representa a dispersão intracluster.

Esse índice procura padronizar o gráfico de $log(W_k)$ comparando-o com sua esperança sob uma distribuição de referência dos dados apropriada. A estimativa do número ótimo de *clusters* é o valor de k para o qual $log(W_k)$ se situa o mais abaixo possível dessa curva de referência.

$$Gap_n(k) = E_n^*\{\log(W_k)\} - \log(W_k) \tag{15.16}$$

$$W_k = \sum_{r=1}^{k} \frac{1}{2n_r} D_r \tag{15.17}$$

$$D_r = \sum_{\mathbf{x}_i, \mathbf{x}_j \in \mathbf{C_r}} d(\mathbf{x}_i, \mathbf{x}_j) \tag{15.18}$$

O índice *Gap* foi proposto para estimar o número de *clusters* presentes em um conjunto de dados. O valor estimado de k é o valor que maximiza $Gap_n(k)$ após levar em consideração a distribuição da amostra.

É uma estimativa muito geral, aplicável a qualquer algoritmo de agrupamento e medida de distância. Para tornar a estatística *Gap* um procedimento operacional, é preciso encontrar uma distribuição referência apropriada e avaliar a distribuição amostral da estatística *Gap*. A população de referência pode ser determinada de duas formas:

a. Gerar cada atributo de referência uniformemente no intervalo dos valores observados para aquele atributo ou

b. Gerar os atributos de referência a partir de uma distribuição uniforme sobre um retângulo alinhado com os componentes principais dos dados.

Embora o método a seja mais simples, o método b leva em consideração a forma da distribuição dos dados e torna o procedimento invariante à rotação, se o método de agrupamento for invariante.

Para calcular a estatística *Gap*, estima-se $E_n^*\{\log(W_k)\}$ pela média de B cópias $log(W_k^*)$, cada uma computada de uma amostra de Monte Carlo extraída da distribuição referência. Em seguida, é preciso avaliar a distribuição amostral da estatística *Gap*. Mais detalhadamente, o cálculo da estatística *Gap* pode ser feito conforme o Algoritmo 15.3.

O cálculo da estatística *Gap* assume que há *clusters* uniformes bem separados. Quando existem *subclusters* menores dentro de *clusters* maiores bem separados, a estatística *Gap* pode exibir um comportamento não monotónico. Assim, é importante examinar a curva *Gap* inteira, em vez de simplesmente buscar o seu máximo.

Algoritmo 15.3 Cálculo da estatística *Gap*

Entrada: Um conjunto de dados $\mathbf{X}$
Um algoritmo de agrupamento ϕ
Saída: Estatística *Gap*
Melhor número de *clusters*

Geram-se B conjuntos de referência $\mathbf{X_b}$, utilizando o método a ou b, descritos anteriormente
para cada *número de clusters* $k = 1, 2, ..., k_{max}$ **faça**
    Aplica-se o algoritmo ϕ ao conjunto de dados $\mathbf{X}$, gerando a partição π^k
    Calcula-se a medida de dispersão dentro dos *clusters* (W_k) para a partição π^k
    para cada *conjunto de referência* $\mathbf{X_b}$ **faça**
        Aplica-se o algoritmo ϕ ao conjunto referência $\mathbf{X_b}$, gerando a partição π^{kb}
        Calcula-se a medida de dispersão W^*_{kb} para a partição π^{kb}
    fim
    Computa-se a estatística *Gap* estimada, dada pela Equação 10

$$Gap_n(k) = \frac{1}{B}\sum_{b=1}^{B}\log(W^*_{kb}) - \log(W_k) \tag{15.19}$$

    Computa o desvio-padrão sd_k, dado pela Equação 12, em que $\bar{l} = (1/B)\sum_b \log(W^*_{kb})$

$$sd_k = \sqrt{\frac{1}{B}\sum_{b=1}^{B}\{\log(W^*_{kb}) - \bar{l}\}^2} \tag{15.20}$$

    Define-se $s_k = sd_k\sqrt{1+1/B}$
fim
Escolhe-se o melhor número de *clusters* como sendo o menor k tal que $Gap(k) \geq Gap(k+1) - s_{k+1}$

15.4 Critérios Externos

O objetivo da validação externa é medir o quanto o agrupamento obtido confirma uma hipótese pré-especificada. Para isso são utilizados testes de hipótese, que testam se o valor do índice utilizado é incomumente grande ou pequeno, de acordo com uma distribuição de referência, conforme mencionado na Seção 15.1.

Conforme dito anteriormente, para utilizar os índices em critérios externos, aplicando testes estatísticos, é preciso conhecer suas funções de densidade de probabilidade sob a

hipótese nula H_0, que pressupõe estrutura aleatória do conjunto de dados. O cálculo da função de densidade de probabilidade desses índices é difícil. Uma alternativa usualmente utilizada é a aplicação de análise de Monte Carlo. O procedimento para a utilização de análise de Monte Carlo para determinar a distribuição referência de um índice externo sob a hipótese nula H_0 e responder se um agrupamento π^e é válido de acordo com ele é apresentado no Algoritmo 15.4 (Halkidi et al., 2002a; Jain e Dubes, 1988).

Algoritmo 15.4 Procedimento para análise de Monte Carlo

Entrada: Um conjunto de dados $\mathbf{X}$
Um algoritmo de agrupamento ϕ
Uma partição real π^r
Uma partição obtida com o algoritmo ϕ, π^e

1 Gerar B conjuntos de dados $\mathbf{X}_b$, $b = 1,...,B$, de tamanho n, com a mesma dimensão do conjunto de dados original $\mathbf{X}$, seguindo uma distribuição uniforme (Para um nível de significância α de 0,05, Jain e Dubes (1988) indicam a utilização de $B = 100$).
2 **para cada** *conjuntos de dados* $\mathbf{X}_b$ **faça**
3 Gerar uma partição aleatória π^{rb}, associando cada objeto do conjunto a um dos *clusters* existentes em π^r, aleatoriamente e de forma a que cada *cluster* tenha o mesmo tamanho dos *clusters* em π^r
4 Aplicar ϕ ao conjunto $\mathbf{X}_b$, gerando a partição π^{eb}
5 Calcular o índice escolhido, $ind(\pi^{eb}, \pi^{rb})$, entre as partições π^{eb} e π^{rb}
6 **fim**
7 Calcular o valor do índice entre as partições π^e e π^r, $ind(\pi^e, \pi^r)$
8 Criar um gráfico de dispersão para os B valores do índice, $ind(\pi^{eb}, \pi^{rb})$ (esse gráfico é uma aproximação da função de densidade de probabilidade do índice)
9 Comparar o índice calculado para a partição π^e, $ind(\pi^e, \pi^r)$, aos valores do gráfico, considerando um dado nível de significância. Se s valores de $ind(\pi^{eb}, \pi^{rb})$ são maiores que o valor de $ind(\pi^e, \pi^r)$, com $s = (1 - alpha)B$, a hipótese nula é aceita, indicando que os dados são distribuídos aleatoriamente (Halkidi et al., 2002a)

Os índices externos mais tradicionais, frequentemente utilizados na validação de partições, comparam uma partição resultante da aplicação de um algoritmo, π^e, a uma partição independente dos dados, construída com base na intuição ou conhecimento *a priori* sobre a estrutura real dos dados, π^r. São os casos da estatística Rand (R), descrita na Seção 15.4.1, do coeficiente de Jaccard (J), descrito na Seção 15.4.2, do índice de Fowlkes e Mallows (FM), descrito na Seção 15.4.3, e da estatística Hubert's Γ normalizada, descrita na Seção 15.4.4.

Além das formas tradicionais dos índices, equações corrigidas são frequentemente empregadas. Uma correção bastante comum é a normalização do índice para que ele apre-

sente o valor 0 quando uma partição é selecionada ao acaso e 1 quando uma partição casa perfeitamente com a partição real (Jain e Dubes, 1988; Gordon, 1999). Um dos índices mais populares empregados em validação externa é o índice Rand corrigido, apresentado na Seção 15.4.5.

Além desses índices bastante tradicionais, outros foram propostos com o mesmo objetivo, como é o caso do variação de informação (Meila, 2007), que é um índice baseado na Teoria da Informação, e o índice proposto por Dom (2002). Como representante dos índices mais recentes, a variação de informação será descrita na Seção 15.4.6.

Os índices R, J, FM e Γ se baseiam nas seguintes informações a respeito da relação entre os objetos de π^e e π^r. Um par de objetos, $(\mathbf{x}_i, \mathbf{x}_j)$, é dito:

- SS: se pertencem ao mesmo *cluster* de π^e e ao mesmo *cluster* de π^r.
- SD: se pertencem ao mesmo *cluster* de π^e e a *clusters* diferentes de π^r.
- DS: se pertencem a *clusters* diferentes de π^e e ao mesmo *cluster* de π^r.
- DD: se pertencem a *clusters* diferentes de π^e e a *clusters* diferentes de π^r.

Sejam $a1$, $a2$, $a3$ e $a4$ os números de pares SS, SD, DS e DD, respectivamente. Define-se $M = a1 + a2 + a3 + a4$ como o número máximo de todos os pares no conjunto de dados ($M = n(n-1)/2$). Definem-se também $m_1 = a1 + a2$ e $m_2 = a1 + a3$.

A seguir, são descritos alguns dos índices mais tradicionalmente empregados em validação externa. Além das formas tradicionais desses índices, equações corrigidas também são frequentemente empregadas. Os índices mais tradicionais são também comparados na Seção 15.4.7.

15.4.1 Índice Rand

O índice Rand, definido pela Equação 15.21, computa a probabilidade de que dois objetos pertençam ao mesmo *cluster* ou pertençam a *clusters* diferentes nas duas partições π^e e π^r (Boutin e Hascoët, 2004).

$$R(\pi^e, \pi^r) = \frac{(a1 + a4)}{M} \tag{15.21}$$

15.4.2 Índice Jaccard

Esse índice computa a probabilidade de que dois objetos pertencentes ao mesmo *cluster* em uma das partições também pertençam ao mesmo *cluster* na outra partição (Boutin e Hascoët, 2004). O coeficiente de Jaccard (J) é dado pela Equação 15.22.

$$J(\pi^e, \pi^r) = \frac{a1}{(a1 + a2 + a3)} \tag{15.22}$$

15.4.3 Índice de Fowlkes e Mallows

O índice de Fowlkes e Mallows (FM) é dado pela Equação 15.23. Os maiores valores desse índice indicam semelhança entre as duas partições, e variam no intervalo $[0,1]$.

$$FM(\pi^e,\pi^r) = \frac{a1}{\sqrt{(m_1)(m_2)}} \tag{15.23}$$

15.4.4 Índice Hubert normalizado

A estatística Hubert's Γ normalizada, dada pela Equação 15.24, mede o grau de correspondência linear entre duas partições. Os valores desse índice variam no intervalo $[-1,1]$, e valores absolutos excepcionalmente grandes de Γ indicam que as duas partições concordam, o que deve ser estabelecido com algum teste de hipótese, conforme mencionado anteriormente (Seção 15.1).

$$\Gamma(\pi^e,\pi^r) = \frac{Ma1 - m_1 m_2}{\sqrt{m_1 m_2 (M-m_1)(M-m_2)}} \tag{15.24}$$

15.4.5 Índice Rand Corrigido

A correção da estatística Rand, proposta por Hubert e Arabie (1985), resulta na estatística Rand corrigida (CR), dada pela Equação 15.25, em que n_{ij} é o número de objetos comuns aos *clusters* $\mathbf{C}_i$ de π^e e $\mathbf{C}_j$ de π^r, $n_{i.}$ é o número de objetos no *cluster* $\mathbf{C}_i$ de π^e, $n_{.j}$ é o número de objetos no *cluster* $\mathbf{C}_j$ de π^r e k^e e k^r são os números de *clusters* nas partições π^e e π^r, respectivamente. Esse índice é um dos mais utilizados para validação externa em agrupamentos.

$$CR(\pi^e,\pi^r) = \frac{\sum_{i=1}^{k^e}\sum_{j=1}^{k^r}\binom{n_{ij}}{2} - \left[\sum_{i=1}^{k^e}\binom{n_{i.}}{2}\sum_{j=1}^{k^r}\binom{n_{.j}}{2}\right]/\binom{n}{2}}{\left[\sum_{i=1}^{k^e}\binom{n_{i.}}{2} + \sum_{j=1}^{k^r}\binom{n_{.j}}{2}\right]/2 - \left[\sum_{i=1}^{k^e}\binom{n_{i.}}{2}\sum_{j=1}^{k^r}\binom{n_{.j}}{2}\right]/\binom{n}{2}} \tag{15.25}$$

O índice Rand corrigido varia no intervalo $[-1,1]$, e valores menores ou próximos a 0 indicam que a semelhança entre as partições se deve ao acaso; o valor 1 indica que as partições são idênticas (Hubert e Arabie, 1985). Apesar de o menor valor possível do índice ser -1, esse valor não é atingido. Em geral, partições bastante diferentes resultam em um valor próximo de 0. Poderia ser apropriado ter o limite inferior 0 bem definido, mas a normalização necessária não oferece vantagens práticas, uma vez que valores negativos do índice não têm aplicação prática (Hubert e Arabie, 1985).

Uma outra correção muito utilizada é a correção sugerida por Morey e Agresti (1984) e utilizada por Milligan e Cooper (1986). Entretanto, segundo Hubert e Arabie (1985), essa

correção contém um erro, pois assume inadequadamente que a esperança de uma variável aleatória ao quadrado é o quadrado da esperança.

Uma extensão *fuzzy* para o índice Rand corrigido e outros pode ser encontrada em Campello (2007).

15.4.6 Índice variação de informação

A variação de informação (VI), dada pela Equação 15.26, mede a quantidade de informação perdida ou ganha na mudança da partição π^e para π^r. Nessa equação, $H(\pi^a)$ é a entropia de uma partição π^a, dada pela Equação 15.27, e $I(\pi^a,\pi^b)$ é a informação mútua compartilhada entre as partições π^a e π^b, dada pela Equação 15.28, com $p(i) = \frac{|\mathbf{C}_i{}^a|}{n}$ e $p(i,j) = \frac{|\mathbf{C}_i{}^a \cap \mathbf{C}_j{}^b|}{n}$. Esse índice tem o valor 0 como menor valor, não sendo limitado superiormente. O valor 0 indica que as partições são idênticas.

$$VI(\pi^e,\pi^r) = H(\pi^e) + H(\pi^r) - 2I(\pi^e,\pi^r) \tag{15.26}$$

$$H(\pi^a) = -\sum_{i=1}^{k^a} p(i)\log(p(i)) \tag{15.27}$$

$$I(\pi^a,\pi^b) = \sum_{i=1}^{k^a}\sum_{j=1}^{k^b} p(i,j)\log\left(\frac{p(i,j)}{p(i)p(j)}\right) \tag{15.28}$$

15.4.7 Comparação dos índices tradicionais para validação externa

Os índices R, J, FM, Γ e CR são os mais utilizados para validação externa. A maioria deles pode ser sensível ao número de *clusters* de uma partição ou à distribuição dos objetos nos *clusters* (Filho, 2003). Os índices Γ e R têm a tendência de apresentar valores mais elevados para partições com maior número de *clusters* (Milligan et al., 1983). O índice J apresenta valores mais elevados para partições com menor número de *clusters*. Já o índice CR não tem essa influência do número de *clusters*.

Valores altos de R, FM e Γ indicam forte concordância entre as duas partições π^e e π^r. A estatística Γ é uma correlação, e seu valor está compreendido no intervalo $[-1,1]$. CR também apresenta valores no intervalo $[-1,1]$; entretanto, raramente valores próximos de -1 são produzidos. R e J possuem valores no intervalo $[0,1]$, e o valor máximo 1 não é atingível quando as duas partições têm números de *clusters* diferentes.

Os índices Rand, Jaccard e Fowlkes e Mallows têm como limite inferior o valor 0, mas na prática nunca vão produzir esse valor em um agrupamento de dados real. Milligan e Cooper (1986) confirmaram que o índice CR apresenta valores próximos de 0 quando os *clusters* são gerados a partir de dados aleatórios.

Nas investigações de Milligan et al. (1983) sobre alguns desses índices, foi encontrado um alto grau de consistência. Esses autores encontraram alta similaridade entre os índices

J e FM, talvez devido ao fato de ambos utilizarem o mesmo numerador (número de pares de objetos que foram apropriadamente agrupados pelo algoritmo). Da mesma forma, os índices R e CR se comportaram de forma semelhante, o que é razoável, uma vez que a estatística Rand corrigida (CR) é apenas um ajuste da estatística R original. Milligan et al. (1983) recomendam a utilização de apenas um índice de cada par de índices similares, apontando para o CR e o J como as melhores escolhas de cada grupo.

15.5 Considerações Finais

Neste capítulo foram discutidas questões relativas ao planeamento de experiências e avaliação de modelos descritivos, mais especificamente de abordagens para análise de agrupamento. Assim, tal como para os modelos preditivos, esses aspectos são de grande relevância para a realização de experimentos com algoritmos de agrupamento de dados, de forma a garantir a validade dos resultados alcançados, ou para se escolher o melhor modelo para uso em um determinado problema.

Resumidamente, foram detalhados os três tipos de critérios existentes para avaliação de agrupamentos, que são os critérios relativos, apropriados para comparar diversos agrupamentos com respeito a algum aspecto, os critérios internos, que avaliam a qualidade de um agrupamento de acordo com alguma propriedade existente nos dados originais e os critérios externos, que permitem o confronto do agrupamento obtido por um algoritmo com uma estrutura dos dados previamente conhecida.

Além disso, foram discutidos diversos índices que podem ser aplicados com cada critério, diferentes maneiras como esses índices podem ser utilizados, bem como várias abordagens recentes que consideram aspectos como estabilidade das partições geradas.

Parte IV

Tópicos Avançados

Introdução

O futuro da área de ECD aponta para sistemas autónomos que podem incorporar o conhecimento e aprender a partir de dados distribuídos, que são gerados em ambientes dinâmicos e não estacionários. Os agentes terão capacidade de comunicar entre si e capacidade de transferir o conhecimento entre os problemas de aprendizagem. Neste capítulo apresentamos algumas linhas de investigação em ECD que lidam com diversos tópicos avançados.

No Capítulo 16 é abordado o problema da aprendizagem em fluxos de dados. A maioria dos algoritmos estudados neste livro requer que todos os dados de treino estejam disponíveis em memória. No entanto, o aumento exponencial de dados armazenados em grandes bases de dados, geralmente recolhidos ao longo do tempo, durante meses ou anos, levanta novos problemas. Além disso, em algumas situações, tais como redes de sensores, tráfego TCP / IP, *e-commerce* e *web sites*, os fluxos de dados são gerados a alta velocidade. Neste contexto não é possível armazenar todos os dados na memória e executar várias passagens sobre os dados de treino. Para aumentar a dificuldade do problema de aprendizagem, o conceito a ser aprendido pode evoluir ao longo do tempo (Gama, 2010). Tudo isto requer a investigação sobre o escalonamento de algoritmos de ECD, que permita a aprendizagem sequencial a partir de fluxos de dados em ambientes dinâmicos (Domingos e Hulten, 2000; Gama et al., 2003; Aggarwal et al., 2003).

Um dos problemas atuais na investigação em ECD é o problema da seleção de algoritmos e de modelos. Esta questão pode ser resumida como: *Para um dado conjunto de dados, qual é o algoritmo/modelo mais adequado?* Qualquer algoritmo de ECD tem sua própria área, ou nicho, de aplicabilidade: existem problemas para os quais um algoritmo apresenta uma boa generalização, enquanto que, para outros problemas, não revela uma boa capacidade de generalização. Uma área de investigação promissora é a de meta-aprendizagem (Brazdil et al., 2009). A meta-aprendizagem caracteriza a área de aplicabilidade de algoritmos de aprendizagem, sendo útil na seleção de algoritmos e de modelos em problemas semelhantes. Este tópico é coberto no Capítulo 17.

A maioria das aplicações de ECD em problemas de classificação reportadas na literatura trata de problemas com apenas duas classes. Vários algoritmos de ECD foram desenvolvidos para lidar apenas com este tipo de problemas. No entanto, existem vários problemas reais onde é necessário discriminar mais do que duas classes, conhecidos como problemas de classificação multiclasse. Como veremos no Capítulo 18, as estratégias mais

eficientes para lidar com estes problemas decompõem o problema multiclasse original em vários problemas de classificação binária.

Em geral, os classificadores induzidos por algoritmos de ECD, em problemas multiclasse, atribuem uma única classe a cada objeto. Porém, existem problemas reais de classificação em que um objeto pode pertencer simultaneamente a mais do que uma classe. Estes problemas, conhecidos como problemas de classificação multirótulo, ocorrem com frequência nas áreas de bioinformática e processamento de texto. Um exemplo é a classificação da função de proteínas, em que uma proteína pode ter mais do que uma função. O Capítulo 19 aborda esta problemática.

Outro aspeto que deve ser destacado prende-se com o fato de diversos problemas multiclasse poderem ser melhor representados utilizando uma estrutura hierárquica, em que o processo de classificação pode classificar novos objetos em diferentes níveis de uma hierarquia. Muitos problemas de classificação hierárquica são também problemas de classificação multirótulo. As técnicas de classificação hierárquica, que serão descritas no Capítulo 20, apresentam alternativas para lidar com este tipo de problemas.

Existem várias técnicas de ECD, como as RNAs, que são inspiradas em processos que ocorrem na natureza. Essas técnicas, estudadas no âmbito de uma área denominada Computação Natural, têm sido aplicadas a vários problemas de ECD preditivo. No Capítulo 21 são apresentadas algumas técnicas de Computação Natural, sendo atribuído maior ênfase às técnicas que possuem inspiração biológica, tais como Computação Evolutiva, Inteligência de Enxames e Sistemas Imunológicos Artificiais.

Ao longo do livro (com raras exceções, como os sistemas de CBR), temos assumido uma representação atributo-valor para as observações. Cada exemplo é descrito por um conjunto de variáveis (um registo na terminologia de base de dados) e um conjunto de exemplos é armazenado numa matriz (a relação, ou tabela, na terminologia de Base de Dados). Porém, nalgumas das aplicações de aprendizagem automática mais desafiantes, os dados são descritos por sequências (por exemplo, dados de DNA), árvores (documentos XML) e grafos (componentes químicos, análise de rede). O Capitulo 22 apresenta uma visão das técnicas utilizadas na análise de redes sociais.

Capítulo 16

Aprendizagem em Fluxos Contínuos de Dados

Hoje em dia, a investigação e a prática de ECD são confrontadas com novos problemas e desafios. A forma de recolher dados já não é manual mas sim automática: temos dispositivos, sensores e computadores que recolhem, processam e enviam informações para outros computadores. Por vezes, não é mesmo viável armazenar toda a informação em sistemas de gestão de bases de dados tradicionais, principalmente porque estes sistemas não foram projetados para suportar diretamente consultas que precisam ser executadas de forma contínua (Babcock et al., 2002). Em algumas aplicações, como é o caso de redes de sensores e tráfego TCP/IP, a melhor forma de analisar a informação recolhida não é através de tabelas persistentes, mas sim com base em fluxos transitórios de dados.

Nas últimas décadas, a investigação e a prática de ECD têm-se focado na aprendizagem por lotes, geralmente através do recurso a pequenos conjuntos de dados. Na aprendizagem *offline*, todos os dados do conjunto treino estão disponíveis em memória e um algoritmo aprende um modelo de decisão iterativamente, processando todos os dados em cada iteração. A lógica subjacente a esta prática é que os exemplos são gerados aleatoriamente de acordo com uma distribuição estacionária.

Os desafios na aprendizagem em fluxos contínuos de dados relacionam-se com a capacidade de processar grandes volumes de informação que evoluem ao longo do tempo e que são gerados por distribuições não estacionárias. Neste contexto, o objetivo consiste em manter, de forma permanente, um modelo de decisão preciso e consistente com o estado atual do processo que gera dados. Este problema requer algoritmos de aprendizagem incrementais, que podem modificar o modelo atual sempre que é disponibilizada nova informação. Desta forma, o pressuposto de que os exemplos são gerados aleatoriamente de acordo com uma distribuição estacionária não se verifica. Na presença de uma distribuição não estacionária, o algoritmo de aprendizagem deve incorporar mecanismos de esquecimento, de forma a permitir descartar informações e conceitos que já não refletem o estado atual do sistema que está a ser modelado. A aprendizagem a partir de dados fluxos requer algoritmos de aprendizagem incrementais, que funcionam com recursos computa-

cionais limitados, e que são capazes de tomar em consideração mudanças e a evolução dos conceitos a aprender.

Neste Capítulo serão enumerados os principais desafios na aprendizagem em fluxos contínuos de dados. Na Seção 16.2 são apresentados algoritmos para indução de árvores de decisão e análise de agrupamento para fluxos contínuos de dados. A Seção 16.3 trata do problema de dados não estacionários (*concept drift*), um dos tópicos que mais diferencia a aprendizagem *offline* da aprendizagem em tempo real. O capítulo termina com a identificação de problemas em aberto e de linhas de investigação futura.

16.1 Desafios na Aprendizagem em Fluxos Contínuos de Dados

A aprendizagem a partir de fluxos contínuos de dados, produzidos a alta velocidade, em ambientes dinâmicos e não estacionários, requer novas técnicas de amostragem, algoritmos incrementais, capacidade de processar dados à velocidade a que estes são disponibilizados, bem como a capacidade de esquecer dados desatualizados. Domingos e Hulten (2001) identificam as propriedades de um modelo computacional nestes sistemas de aprendizagem:

1. Incrementalidade;
2. Aprendizagem em tempo real;
3. Capacidade para processar exemplos em tempo constante e usando memória limitada;
4. Acesso limitado a exemplos já processados;
5. Capacidade de detetar e adaptar o modelo de decisão a mudanças do conceito.

Alguns exemplos de algoritmos de aprendizagem projetados para processar fluxos de dados incluem:

- Aprendizagem preditiva
  - Árvores de decisão: Domingos e Hulten (2000); Gama et al. (2003); Jin e Agrawal (2003);
  - Regras de decisão: Ferrer-Troyano et al. (2005); Gama e Kosina (2011);
- Aprendizagem descritiva
  - Variantes do algoritmo *k*-médias: Zhang et al. (1996); Sheikholeslami et al. (1998);
  - *Micro Clustering*: Callaghan et al. (2002); Aggarwal et al. (2003);
  - Agrupamento hierárquico de séries temporais: Rodrigues et al. (2006);

- Regras de associação
  - *Itemsets* frequentes: Han et al. (2000);
  - Padrões frequentes: Giannella et al. (2003);
- Deteção de novidade: Markou e Singh (2003); Spinosa et al. (2008);
- Redução de dimensionalidade: Sousa et al. (2007).

Todos estes algoritmos compartilham as propriedades previamente enumeradas, com restrições de tempo e memória. Estes algoritmos mantêm modelos de decisão dinâmicos, que evoluem ao longo do tempo à medida que novos dados estão disponíveis, podem ser utilizados em qualquer momento do processo de aprendizagem, e são capazes de adaptar o modelo de decisão aos dados mais recentes.

16.2 Algoritmos de Aprendizagem em Fluxos de Dados

Para elucidar o funcionamento do processo de aprendizagem em fluxos de dados, apresentamos três exemplos ilustrativos de algoritmos de aprendizagem que mantêm, continuamente, um modelo de decisão que evolui ao longo do tempo. No primeiro caso, apresentamos um algoritmo de indução de árvores de decisão. Como segundo exemplo, é apresentado um algoritmo de agregação hierárquica de séries temporais. Este algoritmo foi projetado para lidar com milhares de séries temporais que fluem em alta velocidade. O terceiro exemplo é uma adaptação simples de redes neuronais à aprendizagem em fluxos de dados a alta velocidade.

16.2.1 O Algoritmo *Árvore de Decisão Muito Rápida*

Aprender a partir de grandes conjuntos de dados pode ser mais eficiente quando se utilizam algoritmos que colocam maior ênfase na gestão do viés. Um desses algoritmos é o sistema *árvore de decisão muito rápida* (VFDT, do inglês *Very Fast Decision Tree*) (Domingos e Hulten, 2000), apresentado no Algoritmo 16.1.

No VFDT, uma árvore de decisão é construída recursivamente substituindo as folhas por nós de decisão. Cada folha armazena as estatísticas suficientes sobre os valores dos atributos. Estas estatísticas são as requeridas para estimar a função de avaliação que calcula o mérito dos testes de divisão, de acordo com os valores dos atributos. Quando um exemplo está disponível, atravessa a árvore desde a raiz até uma folha, avaliando o atributo apropriado para cada nó e seguindo o ramo correspondente ao valor do atributo no exemplo. Quando o exemplo alcança uma folha, as estatísticas suficientes são atualizadas. Quando essa folha acumula informação sobre um número mínimo de exemplos, é avaliada cada possível condição baseada no valor dos atributos. Existindo suporte estatístico suficiente em favor de um teste sobre os outros, o nó é transformado num nó de decisão. Este nó de decisão terá tantos descendentes quantos os valores possíveis para o atributo escolhido. Os nós de decisão apenas mantêm a informação sobre o teste de divisão instalado.

Algoritmo 16.1 O algoritmo da árvore de Hoeffding

Entrada: Uma sequência de exemplos de treino $\mathbf{D} = \{(\mathbf{x}_i, y_i), i = 1, \ldots, \infty\}$
Uma função de avaliação de divisão H
Número mínimo de exemplos N_{min}
δ: Probabilidade desejada de escolher o atributo correto em qualquer nó.
τ: Constante usada para resolver desempates.
Saída: HT: Árvore de Decisão

```
1  Seja HT ← Folha Vazia (Raiz) ;
2  para cada exemplo (x_i, y_i) ∈ D faça
3      Atravessar a árvore HT a partir da raiz até à folha l;
4      Atualizar as estatísticas suficientes em l;
5      se O número de exemplos em l é maior que N_min então
6          Calcular H(at_i) para todos os atributos ;
7          Seja at_a o atributo com maior H; e at_b o atributo com o segundo maior H ;
8          Calcular o limite de Hoeffding ε ;
9          se (H(at_a) − H(at_b) > ε ∨ ε < τ) então
10             Substituir l por um teste de divisão baseado no atributo at_a ;
11             Adicionar uma nova folha vazia para cada possível valor de at_a ;
12         fim
13     fim
14 fim
```

A principal inovação do sistema VFDT é o uso do limite de Hoeffding para decidir quantos exemplos devem ser observados antes de instalar um teste de divisão num nó. Suponha que foram observados n valores independentes de uma variável aleatória X que toma valores num intervalo de amplitude R. Assuma que a média de X na amostra é $\bar{x}$. O limite de Hoeffding afirma que, com probabilidade $1-\delta$, a média de x, μ, é pelo menos $\bar{x} - \varepsilon$, onde $\varepsilon = \sqrt{R^2 \frac{\ln(\frac{1}{\delta})}{2n}}$.

Seja H a função de avaliação de um atributo. Para o ganho de informação, R é $log_2(k)$, em que k denota o número de classes. Sejam at_a o atributo com o maior H, at_b o atributo com o segundo maior H e $\overline{\Delta H} = \overline{H}(at_a) - \overline{H}(at_b)$ a diferença entre os dois melhores atributos. Então, se $\overline{\Delta H} > \varepsilon$, o limite de Hoeffding diz que, com probabilidade $1-\delta$, at_a é realmente o atributo que melhor discrimina as classes. Neste caso, a folha deve ser transformada num nó de decisão que divide os exemplos pelo valor de at_a. Se $\overline{\Delta H} < \varepsilon$ a folha não é expandida, e o algoritmo prossegue processando mais exemplos. Como ε diminui com o aumento de n, a expansão de uma folha requer sucessivamente menores valores para $\overline{\Delta H}$.

Porém, a avaliação da função de mérito para cada exemplo do fluxo de dados não é computacionalmente eficiente. O VFDT só calcula a função de avaliação dos atributos, H,

depois de observar um número mínimo de exemplos. O número mínimo de exemplos é um parâmetro definido pelo utilizador, tipicamente na ordem das centenas.

Um problema desta estratégia ocorre quando existem dois, ou mais, atributos igualmente relevantes para discriminar entre as classes. Nesta situação, as sucessivas avaliações de H irão retornar valores semelhantes, mesmo após o processamento de um grande número de exemplos, e o limite de Hoeffding não decidirá qual o atributo mais discriminativo. Para resolver este problema, o VFDT usa uma constante τ introduzida pelo utilizador. Por exemplo, se $\overline{\Delta H} < \varepsilon < \tau$, então a folha é transformada num nó de decisão. O teste de divisão é baseado no melhor atributo.

Mais tarde, os mesmos autores apresentaram o algoritmo CVFDT (do inglês, *Concept-adapting Very Fast Decision Tree*) (Hulten et al., 2001) , uma extensão do VFDT desenvolvido para fluxos de dados que mudam ao longo do tempo. O CVFDT gera árvores de decisão alternativas nos nós em que há evidência de que o teste de divisão previamente instalado já não é apropriado, tendo por base os dados mais recentes. O sistema substitui a árvore antiga pela árvore alternativa, quando a última se torna mais precisa.

O VFDT é um algoritmo de aprendizagem de árvores de decisão que ajusta o seu viés de forma dinâmica e à medida que novos exemplos vão sendo disponibilizados. A questão basilar na indução de árvores de decisão, prende-se com a decisão do momento em que a árvore deve ser expandida, por via da instalação de um teste de divisão e da geração de novas folhas. A ideia básica do VFDT consiste na utilização de um pequeno conjunto de exemplos para selecionar o teste de divisão a ser incorporado num determinado nó da árvore de decisão. O VFDT apenas toma uma decisão (isto é, adiciona um teste de divisão nesse nó) quando existe evidência suficiente, com suporte estatístico, em favor de um teste. Esta estratégia garante estabilidade ao modelo (baixa variância) e controla o sobre ajustamentos, podendo aumentar o número de graus de liberdade (baixo viés) com o processamento de mais exemplos.

16.2.2 Análise de Agrupamentos em Séries Temporais Contínuas

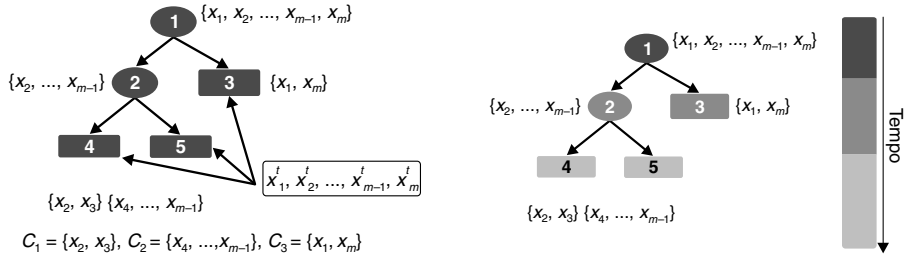
A maior parte do trabalho elaborado no âmbito da análise de agrupamentos em fluxos contínuos de dados, concentra-se no agrupamento de exemplos (Callaghan et al., 2002; Aggarwal et al., 2003). No entanto, em muitas aplicações é igualmente relevante a análise de grupos de variáveis (atributos). Por exemplo, em redes de sensores, estamos interessados em agrupar sensores com comportamento correlacionado. Dada uma matriz estática de dados, a diferença entre agrupamento de exemplos ou agrupamento de variáveis não é relevante. Podemos usar os mesmos algoritmos de análise de agrupamentos diretamente sobre a matriz de dados ou sobre a sua transposta. Contudo, quando lidamos com o processamento de fluxos de dados, em tempo real, a matriz de dados deixa de ser estática e torna-se dinâmica, evoluindo com o tempo à medida que os novos exemplos ficam disponíveis. Neste contexto, a transposta da matriz de dados é um operador de bloqueio.[1] Por esse motivo, e no contexto de fluxos de dados, as técnicas de análise de agrupamentos de

[1] Um operador de bloqueio só retorna resultados depois de processar toda a informação na entrada.

variáveis (séries temporais) requerem algoritmos diferentes dos algoritmos de análise de agrupamento de exemplos.

Um dos primeiros algoritmos para análise de agrupamento de variáveis em fluxos contínuos de dados é o ODAC (do inglês, *Online Divisive-Agglomerative Clustering*) (Rodrigues et al., 2006). O ODAC é um sistema de agrupamento de séries temporais que constrói uma hierarquia de grupos de forma incremental e divisiva. As folhas são os grupos resultantes. Cada folha contém um conjunto de variáveis que são mais correlacionadas entre si do que com as variáveis associadas a outras folhas. A união das folhas representa o conjunto total de variáveis. Por sua vez, a interseção das folhas é o conjunto vazio, uma vez que cada variável pertence a um, e apenas um, dos grupos. O sistema calcula a distância entre variáveis de forma incremental, e executa procedimentos de expansão e agregação da estrutura baseados num intervalo de confiança dado pelo limite de Hoeffding. O ODAC usa dois operadores para manter a estrutura de grupos consistente com a informação mais recente. O operador de *divisão* transforma uma folha f em não-folha, com duas folhas descendentes. As variáveis em f são distribuídas pelas novas folhas descendentes, formando uma estrutura de grupos mais detalhada. Esse operador é aplicado sempre que a informação acumulada numa folha é suficiente para formar grupos mais detalhados. O segundo operador é um operador de *agregação*. Este operador agrupa duas folhas numa única folha. O operador é aplicado quando o sistema deteta que a estrutura de correlação entre as variáveis numa das folhas não é consistente com a estrutura de correlação observada no nó que gerou essa folha. Este operador permite, assim, adaptar a estrutura de grupos quando são observadas mudanças na estrutura de correlação entre variáveis do processo que gera os dados. A Figura 16.1(a) ilustra a estrutura de grupos no instante t. Inicialmente, todas as variáveis pertencem ao mesmo grupo: o grupo 1. O grupo 1 foi expandido dando origem aos grupos 2 e 3. Posteriormente, o grupo 2 foi expandido, dando origem aos grupos 3 e 4. A atual estrutura de grupos é dada pelas folhas 4, 5, e 3. O exemplo recebido no instante t, que contém o valor de todas as variáveis nesse momento, atualiza as estatísticas suficientes nas folhas da estrutura. A Figura 16.1(b) ilustra as diferentes janelas temporais definidas sobre o fluxo de dados . O nó 1 recebeu informação da parte inicial do fluxo de dados. Quando o nó 1 é expandido, os novos exemplos vão atualizar os nós descendentes. Em cada instante, as folhas da estrutura recebem a informação mais recente.

A medida de dissemelhança é o *Rooted Normalized One-Minus-Correlation*. De acordo com esta medida, a dissemelhança entre as variáveis at_a e at_b é definida por: $rnomc(at_a, at_b) = \sqrt{(1 - corr(at_a, at_b))/2}$, em que $corr(at_a, at_b)$ é a correlação de Pearson (Equação 3.7). Esta medida é calculada de forma incremental, e cada exemplo é processado apenas uma vez. A atualização, em tempo real, das estatísticas suficientes necessárias ao cálculo da matriz de dissemelhança requer o cálculo: das somas das variáveis, das somas dos quadrados das variáveis e das somas das multiplicações par a par entre as variáveis. A matriz de distâncias para cada folha é calculada quando a folha é testada para divisão ou agregação, após receber um número mínimo de exemplos, dado pelo limite de Hoeffding. Em cada folha, se a diferença entre a máxima distância entre duas variáveis e a segunda maior distância for estatisticamente significativa, conforme verificado pelo limite



(a) Crescimento da estrutura de grupos.

(b) Múltiplas janelas temporais.

Figura 16.1 *ODAC: Análise de Agrupamentos de Séries Temporais.*

de Hoeffding, então a maior distância é o diâmetro da folha. Neste caso, considera-se que a folha se encontra pronta para ser submetida aos testes de divisão e agregação.

O critério de divisão do ODAC possui dois conceitos inerentes: quanto mais afastadas estiverem as distâncias máxima e mínima de um grupo, maior a probabilidade de ocorrer uma divisão; e, quanto mais afastada estiver a distância média, da média entre as distâncias máxima e mínima, maior a probabilidade de divisão. De forma a distinguir os casos em que o grupo tem todas as variáveis equidistantes dos casos em que possui duas ou mais variáveis muito distantes, foi introduzido um parâmetro, τ, que determina o tempo dado ao sistema para encontrar o diâmetro de uma folha, até que seja forçado o teste de divisão. Quando um ponto de divisão é reportado, as duas variáveis que definem o diâmetro, designadas *pivô*, vão dar origem a dois novos grupos. Cada uma das variáveis restantes integra o novo grupo que contém o *pivô* mais próximo. As novas folhas reinicializam as estatísticas suficientes. Este critério de divisão garante que, sempre que a estrutura de grupos é expandida, o diâmetro dos grupos diminui.

A monotonia do critério de divisão constitui a base para a deteção de alterações na estrutura de correlação entre variáveis. Estas alterações podem ser provocadas, por exemplo, por mudanças estruturais no processo que gera os dados. Em fluxos de dados não estacionários, o operador de agregação permite agregar folhas como resposta a alterações no processo que gera os dados. O critério de agregação é baseado na comparação entre os diâmetros dos nós descendentes e o diâmetro do nó ascendente. Conforme mencionado anteriormente, o critério de divisão garante que, sempre que a estrutura de grupos é expandida, o diâmetro dos grupos deve diminuir. A observação de que o diâmetro de uma folha é significativamente maior do que o diâmetro do nó ascendente, indica uma mudança na estrutura de correlação entre as variáveis nessa folha. Por esse motivo, procede-se à agregação das folhas descendentes desse nó, bem como à reinicialização das estatísticas suficientes nesse nó.

Apesar dos requisitos de espaço do sistema serem quadráticos em relação ao número de variáveis, estes requisitos são constantes no número de exemplos. De modo análogo, os requisitos de tempo do sistema são quadráticos no número de variáveis, mas também constantes no número de exemplos. Uma importante caraterística deste algoritmo é que,

de cada vez que ocorre uma divisão numa folha com d variáveis, o número de distâncias que é necessário calcular diminui em, no mínimo, $d-1$ diminuindo, assim, o tempo gasto, por exemplo. Os resultados experimentais com o ODAC, utilizando conjuntos de dados artificiais e reais, revelaram a sua capacidade de evolução do modelo ao longo do tempo, bem como a respetiva adaptação na presença de alterações de conceito.

16.2.3 Redes Neuronais

RNAs são modelos computacionais que podem aproximar qualquer função contínua (Craven e Shavlik, 1997) com erro arbitrariamente pequeno. Um dos problemas das RNAs é a lentidão do processo de aprendizagem. Outros problemas conhecidos são a tendência para sobre ajustamento, ou *overfitting*, a variância do erro esperado e a convergência para mínimos locais. À primeira vista, estes aspetos parecem limitar a aplicação de redes neuronais ao cenário de fluxo de dados. O processo habitual para treinar uma rede neuronal consiste no processamento dos exemplos de treino por épocas. Mas qual a necessidade de processar o mesmo conjunto de exemplos várias vezes? A única motivação prende-se com a dimensão reduzida do conjunto de treino. Uma das principais caraterísticas de fluxos contínuos de dados é a abundância de dados. Neste contexto, uma rede neuronal pode ser treinada processando cada exemplo uma única vez (Chen e Pung, 2008). Sempre que um exemplo de treino é disponibilizado, o vetor dos atributos é propagado pela rede, e o erro é retropropagado através da rede apenas uma vez. Esta metodologia de treino apresenta grandes vantagens, nomeadamente o fato de permitir a aplicação de redes neuronais em processos que geram um grande volume de dados a alta velocidade. Outra vantagem é a adaptação gradual em fluxos de dados dinâmicos, em que o conceito a aprender evolui ao longo do tempo. Craven e Shavlik (1997) referem que o viés indutivo de redes neuronais é o mais adequado para tarefas de previsão sequenciais e temporais.

16.3 Deteção de Mudança

Mudança de conceitos (do inglês *concept drift*) (Klinkenberg, 2004) significa que o processo que gera os dados evolui ao longo do tempo. Exemplos de conceitos que evoluem ao longo do tempo incluem: o interesse dos utilizadores, o tipo de fraude e *spam*, a qualidade de um produto num processo de produção, entre outros. A evidência de mudanças no conceito a aprender reflete-se, de alguma forma, nas observações sobre esses processos. Observações mais antigas, que refletem o comportamento passado, tornam-se irrelevantes para o estado atual do processo em observação. Os fluxos contínuos de dados evoluem ao longo do tempo com dinâmicas desconhecidas. Assumir que os exemplos são gerados aleatoriamente de acordo com uma distribuição estacionária é irrealista (Basseville e Nikiforov, 1987). Em sistemas complexos, temos de considerar, cumulativamente, as alterações no processo que gera os dados, as mudanças (suaves ou abruptas) na distribuição dos exemplos, e a evolução do conceito a aprender. Nestes contextos, a aprendizagem

automática requer algoritmos adaptativos, ou seja, algoritmos incrementais que detetam e reagem a mudanças do conceito a aprender.

A natureza e a origem das mudanças são diversas. As alterações podem ocorrer, quer devido a mudanças em variáveis que deixaram de ser observadas, quer devido a modificações nas propriedades caraterísticas das variáveis observadas. Muitos algoritmos de aprendizagem utilizam métodos *cegos*, que adaptam o modelo de decisão em intervalos regulares, sem considerar se houve, de fato, alguma mudança. Muito mais interessantes são os mecanismos que detetam essas mudanças explicitamente. As vantagens associadas a este tipo de mecanismos estão relacionadas com a sua habilidade em proporcionar informações significativas sobre o processo em análise, indicando pontos de mudança, ou janelas temporais onde a mudança ocorre, assim como a quantificação do grau de mudança. Podemos agrupar estas abordagens em dois grandes grupos:

- Monitorizar a evolução de indicadores de desempenho, por exemplo, usando técnicas utilizadas em Controle Estatístico de Processos (Gama et al., 2004).
- Monitorizar a distância entre as distribuições em duas janelas temporais (Kifer et al., 2004). Neste caso, o método acompanha a evolução de uma função de distância entre duas distribuições: uma janela de referência e numa janela contendo as observações mais recentes.

As métricas e os critérios para a seleção de algoritmos de deteção de mudança normalmente tomam em consideração a taxa de erro, a rapidez de deteção, assim como a robustez a falsos alarmes e a ruído.

Na aprendizagem preditiva, uma das formas mais eficazes para detetar mudanças, ou desvios, no conceito a aprender consiste em monitorizar o processo de aprendizagem dos modelos preditivos. Por exemplo, por via da monitorização da evolução de uma medida de desempenho. Na literatura da área foram propostos vários algoritmos de deteção de mudança, que monitorizam a evolução da taxa de erro, entre eles: o *Page-Hinkley* teste (Hinkley, 1970), o SPC (do inglês *Statistical Process Control*) (Gama et al., 2004) ou o Adwin *Adaptive Window* (Bifet e Gavaldà, 2007). Estes algoritmos monitorizam a evolução do erro de um classificador. Assumem que, em processos estacionários, o erro do classificador deve manter-se constante ou diminuir. Se o processo não é estacionário, o erro aumenta. No entanto, um aumento do erro pode ser motivado por ruído nos dados. Este constitui o principal problema dos algoritmos de deteção de mudança: distinguir mudanças estruturais no processo que gera os dados, de perturbações temporárias nesse processo.

Nesta seção apresentamos o algoritmo SPC, apresentado em Gama et al. (2004). O algoritmo SPC monitoriza a evolução da taxa de erro de um classificador. Sinaliza alertas de degradação do processo de aprendizagem, e indica mudança quando essa degradação corresponde (para um determinado nível de confiança) a uma alteração estrutural no processo que gera os dados. No SPC, um *alerta de mudança* é sinalizado em situações em que se observa um aumento do erro, mas em que ainda não há evidências suficientes para sinalizar uma *mudança*. O SPC mantém uma memória de curto prazo de exemplos etiquetados, que foram recebidos entre o sinal de alerta e o sinal de mudança. Desta forma, é

possível explorar os exemplos armazenados na memória de curto prazo para inicializar o modelo de decisão sempre que uma mudança é sinalizada.

O algoritmo SPC, cujo pseudo código é apresentado no Algoritmo 16.2, mantém dois registros: p_{min} e s_{min}, em que p representa uma probabilidade de erro e s, a variância associada à estimativa de p. p é calculado como a frequência de erro e s como a variância do erro dada por $s = \sqrt{p \times (1-p)/n}$, em que n representa o número de exemplos processados. Sempre que um novo exemplo i é processado, esses valores são atualizados se $p_i + s_i < p_{min} + s_{min}$. É utilizado um nível de aviso para definir o tamanho ideal da *janela de contexto*. A janela de contexto contém os exemplos que estão no novo contexto, bem como um número mínimo de exemplos associados ao contexto antigo. O nível de alerta é atingido se $p_i + s_i \geq p_{min} + 2 \times s_{min}$, e o nível de mudança é alcançado se $p_i + s_i \geq p_{min} + 3 \times s_{min}$. Suponha uma sequência de exemplos em que o erro do modelo aumenta, atingindo o nível de alerta no exemplo k_w, e o nível de mudança no exemplo k_d. Um novo modelo é induzido usando os exemplos na memória de curto prazo de k_w a k_d.

16.4 Considerações Finais

A aprendizagem em fluxos contínuos de dados é uma área de investigação em crescente expansão e com desafios ao nível das aplicações e das oportunidades de investigação em áreas científicas como: Bases de Dados, Teoria de Algoritmos, Aprendizagem de Máquina e Extração de Conhecimento de Dados. O ambiente onde decorre a aprendizagem gera dados de forma continuada e com dinâmicas desconhecidas. Neste contexto, a aprendizagem é um processo contínuo que evolui ao longo do tempo. Os algoritmos de ECD devem ser capazes de incorporar nova informação no modelo de decisão, detetar e reagir a mudanças, usando recursos computacionais limitados (Gama, 2010).

Exemplos de aplicações relevantes onde a análise de fluxo de dados tem sido bem sucedida incluem: redes de sensores, dados científicos, monitorização de processos industriais, análise de dados da Web e análise de tráfego em redes de computadores. A aprendizagem em tempo real, a capacidade de incorporar nova informação de forma contínua, a capacidade de esquecer, a capacidade de auto-adaptação e auto-reação constituem as principais caraterísticas de qualquer sistema inteligente. Estas são, também, as propriedades caraterísticas dos algoritmos de aprendizagem sobre fluxos contínuos de dados.

Algoritmo 16.2 O Algoritmo SPC para deteção de mudança

Entrada: Um modelo de decisão atual $\hat{f}$
Uma sequência de exemplos $\{(\mathbf{x}_k, y_k), k = 1, \ldots, n\}$

Seja $(\mathbf{x}_j, y_j)$ o exemplo atual
Calcula a previsão do modelo: $\hat{y}_j \leftarrow \hat{f}(\mathbf{x}_j)$
Calcula o erro: $erro_j$
Calcula a média dos erros: p_j e a variância s_j
se $p_j + s_j < p_{min} + s_{min}$ **então**
    $p_{min} \leftarrow p_j$; $s_{min} \leftarrow s_j$
fim
se $p_j + s_j < p_{min} + \beta \times s_{min}$ **então**
    $Alerta \leftarrow Falso$
    Atualiza o modelo de decisão usando o exemplo atual: $(\mathbf{x}_j, y_j)$
fim
senão
    se $p_j + s_j < p_{min} + \alpha \times s_{min}$ **então**
        se *Alerta == 'Falso'* **então**
            Inicializa a memória de curto prazo com o exemplo $(\mathbf{x}_j, y_j)$
            $Alerta \leftarrow Verdadeiro$
        fim
        senão
            Acrescenta o exemplo $(\mathbf{x}_j, y_j)$ à memória de curto prazo
        fim
    fim
    senão
        Reaprende um novo modelo de decisão com os exemplos armazenados na memória de curto prazo
        $Alerta \leftarrow Falso$
        Reinicializa p_{min} e s_{min} e a memória de curto prazo
    fim
fim

Capítulo 17

Meta aprendizagem

Um dos principais desafios enfrentados quando estamos perante um problema de decisão é a escolha do algoritmo (ou conjunto de algoritmos) mais apropriado para esse problema. Como vimos nos capítulos anteriores, diferentes algoritmos de ECD podem ser utilizados em problemas reais. Embora exista um grande número de algoritmos de ECD, faltam regras ou indicações que auxiliem na escolha do algoritmo mais apropriado para um dado problema. Essa escolha geralmente recorre a processos de tentativa e erro ou é influenciada pela disponibilidade do algoritmo na ferramenta computacional utilizada, pela experiência anterior do utilizador ou por sugestões de especialistas em ECD. Estas abordagens, além de serem subjetivas, podem ter um custo computacional elevado.

De acordo com Wolpert (1996), não existe um único algoritmo de ECD que seja sempre melhor que os demais, pois o desempenho de um algoritmo depende de como o seu viés indutivo se adapta a propriedades presentes no conjunto de dados a ser utilizado. A investigação realizada em ECD, numa área denominada *meta-aprendizagem*, indica que é possível, durante o processo de aprendizagem, aprender quais são os algoritmos mais apropriados para um novo conjunto de dados. A análise de resultados experimentais obtidos na investigação realizada nessa área sugere que a meta-aprendizagem pode constituir uma ferramenta poderosa de apoio à seleção de algoritmos, não apenas de ECD, mas de algoritmos baseados em qualquer princípio que possam ser utilizados na aproximação de funções.

Sem perda de generalidade, neste texto será assumido que a meta-aprendizagem será utilizada no âmbito da seleção de algoritmos de ECD para problemas de classificação. É fácil adaptar os conceitos apresentados para outras classes de problemas, como regressão, previsão de séries temporais, agrupamento de dados ou otimização combinatória.

De acordo com Brazdil et al. (2009), a meta-aprendizagem é o estudo dos principais métodos que exploram metaconhecimento para obter modelos e soluções eficientes por meio da adaptação de processos de aprendizagem de máquina e mineração de dados. Os mesmos autores apresentam uma definição mais específica na qual definem meta-aprendizagem como o uso da abordagem de ECD para gerar metaconhecimento mapeando as caraterísticas de problemas (meta atributos) ao desempenho relativo de algoritmos. Para esse efeito, a meta aprendizagem investiga como selecionar os algoritmos mais promisso-

res para um dado conjunto de dados (Vilalta e Drissi, 2002; Brazdil et al., 2009; Smith-Miles, 2008; Prudêncio e Ludermir, 2004; de Souza, 2010).

A meta-aprendizagem difere da aprendizagem convencional no nível em que ocorre a adaptação. Na aprendizagem convencional, que ocorre num nível inicial, denominado nível de base, o processo de aprendizagem ocorre num conjunto de dados de cada vez. Na meta-aprendizagem, que ocorre num nível mais avançado, denominado meta-nível, a aprendizagem dá-se por meio do acumular de experiências obtidas observando o desempenho de diversos algoritmos de ECD quando aplicados a vários conjuntos de dados (Brazdil et al., 2009).

Os dois grandes objetivos da meta-aprendizagem consistem em: *i* fornecer suporte a utilizadores que não possuam experiência na seleção de algoritmos de ECD, *ii* estudar como utilizar o conhecimento obtido da experiência acumulada pela aplicação de diversos algoritmos de ECD, a vários conjuntos de dados, para a geração de modelos com maior capacidade preditiva.

Um processo de meta-aprendizagem é encetado com a criação de um grupo de conjuntos de dados e a seleção de um grupo de algoritmos de ECD. Para cada conjunto, são extraídas caraterísticas que descrevem as suas principais propriedades e anotado o desempenho dos algoritmos de ECD selecionados quando aplicados ao conjunto, formando um meta-dado. O conjunto de meta-dados gerados para todos os conjuntos de dados é, posteriormente, utilizado para a construção de um sistema de recomendação, capaz de selecionar os algoritmos mais adequados para um novo problema.

Uma abordagem simples para o sistema de recomendação é utilizar um algoritmo de aprendizagem baseado em instâncias, como o algoritmo k-NN, que, dadas as caraterísticas do conjunto de dados associados a um novo problema, seleciona os algoritmos com melhor desempenho nos k conjuntos de dados mais semelhantes. A semelhança entre conjuntos de dados é definida pelas caraterísticas extraídas e pode ser calculada, por exemplo, através da distância Euclidiana. Numa abordagem mais sofisticada, o processo de meta-aprendizagem utiliza um algoritmo de ECD que, após treinado com os meta-dados, induz um modelo capaz de associar as caraterísticas de um conjunto de dados ao desempenho obtido pelos diferentes algoritmos de ECD. Este modelo é, então, utilizado num sistema de recomendação que, dado um novo conjunto de dados, recomenda os algoritmos de ECD mais promissores.

O resultado produzido por um sistema de recomendação pode ser um algoritmo ou um conjunto de algoritmos. Quando um conjunto de algoritmos é selecionado, podemos associar um *score* a cada um dos algoritmos. Tendo por base este *score*, os algoritmos são passíveis de ser ordenados de acordo com a sua adequação (Soares, 2004). Segundo Kalousis (2002), um sistema de recomendação pode ser analisado de acordo com os seguintes aspetos:

- Propriedades utilizadas na caraterização do conjunto de dados;
- Medidas de avaliação dos algoritmos;
- Forma de apresentação das sugestões;

- Métodos para construção das sugestões.

Nas seções seguintes será apresentada uma breve explicação de cada um destes aspetos.

17.1 Caraterização de Conjuntos de Dados

A caraterização de conjuntos de dados procura extrair caraterísticas presentes nos dados que possam influenciar o desempenho de algoritmos de ECD. Essas caraterísticas podem ser utilizadas durante o processo de meta-aprendizagem para, por exemplo, recomendar os algoritmos de ECD mais promissores para conjuntos de dados com tais caraterísticas. Para isso, é necessário conhecer de antemão como diferentes algoritmos se comportam em conjuntos de dados com diferentes caraterísticas. Por exemplo, sabe-se que, em geral: o NB se comporta bem quando os atributos de entrada são independentes; que SVMs lidam bem quando o número de atributos de entrada é elevado; e que k-NN não se comporta bem na presença de atributos irrelevantes. Estas relações entre caraterísticas de um conjunto de dados e o desempenho de algoritmos de ECD são exploradas por esse critério (de Souza, 2010).

Michie et al. (1994) sugere que as caraterísticas, ou medidas, extraídas de conjuntos de dados contenham informações relevantes que permitam estimar o desempenho relativo entre algoritmos de ECD. Além disso, a extração dessas medidas deve ter um custo computacional reduzido. De acordo com Vilalta et al. (2005), os estudos realizados em caraterização de dados são geralmente divididos em três abordagens:

- Caraterização direta;
- Caraterização por propriedades de modelos;
- Caraterização baseada em *landmarking*.

Os principais aspetos de cada uma destas abordagens serão seguidamente explicados.

17.1.1 Caraterização Direta

A caraterização direta é a forma mais simples de extrair propriedades de um conjunto de dados. Num dos primeiros trabalhos a utilizar caraterização direta de dados, o projeto STATLOG (Michie et al., 1994), as medidas extraídas de cada conjunto de dados, denominadas meta-atributos, foram divididas em três classes:

- **Medidas simples:** incluem descrições gerais do conjunto de dados, como, por exemplo, número de classes, número de atributos, número de atributos binários e número de objetos.
- **Medidas baseadas em estatística:** são medidas que descrevem estatisticamente os dados, que podem ser valores de obliquidade, valores de curtose, correlação entre atributos por classe, desvio-padrão dos atributos.

- **Medidas baseadas em teoria da informação:** procura quantificar a informação presente nos dados, utilizando medidas como a entropia e a informação mútua.

Cada conjunto de dados pode ser representado por um vetor constituído pelos valores dos seus meta-atributos. Este procedimento foi adotado em diversos estudos. O projeto STATLOG extraiu valores para esses meta-atributos, de 21 conjuntos de dados. Para cada conjunto de dados, foram conduzidas experiências com 23 algoritmos de classificação diferentes, avaliando a capacidade preditiva de cada um deles. Um dos objetivos do projeto era identificar a classe de problemas mais apropriada para cada algoritmo e as medidas de desempenho capazes de caraterizar o sucesso de um algoritmo num dado problema. A principal medida de desempenho utilizada foi a taxa de erro de classificação. Em Sovat (2002), foi proposto um sistema que utilizava raciocínio baseado em casos para, dados os valores para um conjunto de meta atributos, que incluíam as medidas propostas no projeto STATLOG, selecionar uma rede neuronal e os valores dos seus parâmetros.

No âmbito do projeto METAL (Brazdil et al., 2009) foram propostas novas medidas para a caraterização de dados. Este projeto disponibilizou um sistema de recomendação na Internet que retornava sugestões de algoritmos para conjuntos de dados submetidos por interessados via Internet. O principal objetivo do projeto METAL foi o de melhorar o uso de ferramentas de mineração de dados e gerar ganhos significativos no tempo dedicado à condução de experiências (Vilalta et al., 2005). Em de Souza (2010), é investigado um novo conjunto de medidas, baseadas em índices de validação de agrupamentos de dados. Outras sugestões para medidas de caraterização direta podem ser encontradas em Kalousis (2002) e em Soares (2004).

17.1.2 Caraterização por Propriedades de Modelos

Na caraterização via modelos, os meta atributos gerados para um conjunto de dados são propriedades de um ou mais modelos induzidos utilizando esse conjunto (Bensusan, 1998; Bensusan et al., 2000; Peng et al., 2002). Vilalta et al. (2005) enunciaram as vantagens associadas a este tipo de caraterização, conforme se apresenta:

- O conjunto de dados é sumarizado por uma estrutura de dados que incorpora a complexidade e o desempenho da hipótese induzida, e não apenas a distribuição dos dados.
- A representação obtida pode ser utilizada para explicar o desempenho do algoritmo de ECD.

Quando são utilizados modelos na caraterização de conjuntos de dados, o espaço de procura do processo de meta-aprendizagem muda, passando do espaço de objetos para o espaço de modelos (Bensusan et al., 2000). Como o novo espaço permite uma procura eficiente no espaço de hipóteses, espera-se uma sumarização maior e mais eficiente do conjunto de dados originais, gerando melhores meta atributos.

Vilalta et al. (2005) citam como exemplo a indução de uma árvore de decisão para representar um conjunto de dados. Exemplos de propriedades do modelo induzido que

podem ser utilizadas para representar o conjunto de dados incluem: o número de nós folha, o formato da árvore, a profundidade máxima da árvore e o grau de balanceamento da árvore. De acordo com Bensusan et al. (2000), existem evidências empíricas de uma possível relação entre as propriedades dos conjuntos de dados e as estruturas de árvores de decisão não podadas.

17.1.3 *Landmarking*

Na caraterização baseada em *landmarking*, é explorada a informação relativa ao desempenho de um conjunto de algoritmos de classificação rápidos e simples, denominados *landmarkers*, para os conjuntos de dados do repositório (Pfahringer et al., 2000; Bensusan e Giraud-Carrier, 2000a).

Nesta abordagem, os conjuntos de dados são caraterizados pelo desempenho alcançado por diferentes algoritmos, quando a eles aplicados. É expectável que conjuntos de dados gerem meta atributos com valores semelhantes, quando o desempenho dos classificadores nesses conjuntos for idêntico. Podem ser utilizadas diferentes medidas de desempenho como meta atributos como, por exemplo, precisão, revocação e área sob a curva ROC. Outras abordagens mais sofisticadas também foram investigadas (Brazdil et al., 2009).

Para este tipo de abordagem, é sugerido o recurso a diferentes algoritmos, que sejam adequados a diferentes configurações de dados. É também aconselhável que os conjuntos de dados utilizados cubram os domínios dos problemas em que os algoritmos se destacam.

Soares et al. (2001) investigaram uma variante denominada *Landmarking* baseado em amostras, que combina os conceitos de amostragem de dados e *Landmarking*. Para isso, são utilizadas amostras de conjuntos de dados na estimação do desempenho de algoritmos em conjuntos completos. As estimativas de desempenho são utilizadas para caraterizar os conjuntos de dados.

17.2 Medidas de Avaliação dos Algoritmos

A seleção do algoritmo de ECD mais apropriado para um conjunto de dados pressupõe a utilização de medidas que avaliem o desempenho de algoritmos de ECD. Em princípio, qualquer medida de desempenho usada na avaliação dos algoritmos candidatos pode ser utilizada. Para problemas de classificação, podem ser utilizadas, por exemplo, medidas de desempenho preditivo, como taxa de classificações incorretas, revocação, precisão, medida-F e a área sob a curva ROC (cf. Secção 9). Outras medidas além da precisão podem ser utilizadas, como, por exemplo, o custo computacional para as etapas de treino e de teste, quantidade de memória necessária, complexidade do modelo induzido e facilidade de interpretação do modelo.

Nada impede que duas ou mais medidas que avaliem diferentes aspetos possam ser utilizadas em conjunto, atribuindo um peso a cada uma delas, definindo uma ordem de importância entre elas ou empregando técnicas de otimização multiobjetivo. Esta combinação

de medidas distintas pode ocorrer, por exemplo, no desenvolvimento de uma ferramenta computacional baseada em ECD para dispositivos com pouca capacidade de memória.

17.3 Formas de Apresentação de Sugestões

A forma de seleção define quantos algoritmos são selecionados e como esses algoritmos são apresentados ao utilizador. De acordo com Kalousis (2002), as sugestões podem ser apresentadas de três formas:

- Sugestão do melhor algoritmo;
- Sugestão de um grupo com os melhores algoritmos;
- Sugestão de um *ranking* dos melhores algoritmos.

Na primeira forma, é apresentado o algoritmo que induziu o melhor modelo para o novo conjunto de dados, de acordo com alguma medida de avaliação. Esta abordagem é utilizada em Koepf et al. (2000) e Bensusan e Giraud-Carrier (2000b). A ausência de alternativas para o algoritmo recomendado pode ser um problema quando este não puder ser utilizado. Isto ocorre, por exemplo, quando o algoritmo não estiver implementado na ferramenta a ser utilizada. Este problema é resolvido pela segunda forma, que recomenda o conjunto dos melhores algoritmos. Fazem parte desse conjunto o melhor algoritmo e os algoritmos com desempenhos estatisticamente semelhantes ao melhor. Esta forma de apresentação de sugestões foi utilizada no projeto STATLOG. A terceira forma vai um passo além, ordenando os melhores algoritmos de acordo com o desempenho obtido, gerando assim um *ranking* de algoritmos. A formação do *ranking* tanto pode ser baseada numa única medida, como numa combinação de medidas. Soares (2004) investigou o uso de *ranking* para a sugestão de algoritmos de ECD.

17.4 Recomendação com base na Caraterização

Uma das principais tarefas da meta-aprendizagem consiste em relacionar as caraterísticas das bases de dados com o desempenho dos algoritmos de ECD, de forma a permitir a recomendação dos algoritmos mais promissores para um novo conjunto de dados. A construção desta relação pode ser interpretada como um problema de ECD, situado num meta-nível. As caraterísticas das bases de dados são utilizadas para gerar meta-exemplos. Os meta-exemplos são caraterizados por meta atributos e possuem como classe associada alguma informação acerca do desempenho dos algoritmos de ECD disponíveis. O conjunto de meta-exemplos constitui o meta-conjunto de dados, que pode ser utilizado por um algoritmo de ECD na indução de um modelo a ser utilizado por um sistema de recomendação.

O modelo induzido para sugerir algoritmos de ECD tanto pode ser um regressor, denominado *meta regressor*, que associa valores reais a um dos algoritmos de ECD avaliado,

como um classificador, denominado *meta classificador*, que possui como classes os algoritmos avaliados. O resultado gerado por um meta regressor, para um novo conjunto de dados, pode ser a estimativa do desempenho do algoritmo de ECD associado ao meta regressor para esse novo conjunto. Os resultados gerados por um meta regressor, para cada um dos algoritmos avaliados, pode ser utilizado para sugerir um *ranking* dos melhores algoritmos. Quando um meta classificador é utilizado, o resultado pode ser a previsão de uma ou mais classes. Quando mais de uma classe puder ser prevista, tem-se um problema de classificação multirótulo. O tema da classificação multirótulo é abordada no Capítulo 19.

17.5 Estudo de Casos

No projeto STATLOG, foi observado o desempenho de algoritmos de classificação para uma grande variedade de problemas. Mais recentemente, foram realizados trabalhos em que a meta-aprendizagem foi utilizada na seleção de algoritmos em domínios bem definidos.

Em Prudêncio e Ludermir (2006), a meta aprendizagem é utilizado na seleção de modelos para efeitos de análise de séries temporais. Foram realizadas experiências com séries apresentando diferentes caraterísticas, obtidas tanto de um repositório internacional como de uma competição internacional. Foram propostas e utilizadas medidas de caraterização direta para séries temporais. Diferentes estratégias foram investigadas para a seleção de modelos. A mais simples selecionava entre dois modelos. Uma segunda abordagem criava um *ranking* de três possíveis modelos. Para esse efeito, estes modelos foram divididos em três grupos de dois modelos cada. Foi treinada uma rede MLP para classificar uma série num dos modelos de cada grupo. As classificações das redes foram utilizadas para criar um *ranking* de modelos (Prudêncio e Ludermir, 2004). Uma terceira abordagem também utiliza uma rede MLP, mas agora para regressão, definindo um meta regressor (Prudêncio e Ludermir, 2006). O objetivo consiste em encontrar os pesos para ponderar a participação de dois modelos de previsão numa combinação de modelos. Resultados experimentais mostraram uma redução significativa dos erros de previsão de séries temporais.

O domínio de classificação de dados de expressão genética é abordado em de Souza et al. (2010) e de Souza (2010). Nestes trabalhos efetuou-se a avaliação do desempenho de 7 algoritmos de classificação em 49 conjuntos de dados de análise de expressão genética. Foram escolhidos algoritmos que fossem facilmente encontrados em ferramentas computacionais públicas, que possuíssem um custo computacional reduzido e que fossem reconhecidos por apresentarem um bom desempenho preditivo. Foram também propostas novas medidas para a caraterização direta de dados, baseadas em índices de validação de agrupamentos de dados. Uma vez que os dados de expressão genética apresentam milhares de atributos, foram investigadas alternativas para reduzir esse número. A seleção de algoritmos foi realizada de diferentes formas. Foi proposto o uso de um algoritmo *k*-NN com pesos e diferentes funções *kernel*, que privilegiava meta exemplos mais próximos, o uso de meta regressores baseados em SVMs e novos métodos de *ranking* de algoritmos

que empregam árvores de decisão e *bagging*. Outro estudo que utiliza meta-aprendizagem em dados de expressão genética, mas para fins de seleção de algoritmos de agrupamento de dados, pode ser encontrado em de Souto et al. (2008).

A meta-aprendizagem também tem sido utilizada na seleção de técnicas de otimização. Em Kanda et al. (2010), a meta-aprendizagem é utilizada com o objetivo de selecionar os algoritmos de otimização mais promissores para novas instâncias do problema do caixeiro-viajante (Gutin et al., 2002). Quando aplicada a este problema, o objetivo de uma técnica de otimização consiste em encontrar a melhor sequência de viagens entre as cidades, sujeita à restrição de que cada cidade apenas pode ser visitada uma vez. O problema do caixeiro-viajante pode ser representado através de um grafo, em que cada cidade é representada por um vértice do grafo, e a distância entre duas cidades é associada ao tamanho da aresta que conecta as duas cidades. Para isso, são definidos novos meta atributos baseados em propriedades de grafos. Os algoritmos de classificação são utilizados para induzir modelos capazes de, dada uma nova instância do problema, prever qual a técnica de otimização mais promissora. Neste trabalho, é permitido que diferentes técnicas de otimização fiquem empatadas aquando da seleção da técnica mais promissora. Para lidar com esta situação, diferentes estratégias para classificação multirótulo cf. Capítulo 19. são investigadas.

17.6 Considerações Finais

Neste capítulo foram apresentados os principais conceitos que estão na base de uma área recente de investigação em ECD: a área de meta-aprendizagem. Os trabalhos nesta área ajudam a conhecer o potencial, bem como as limitações dos algoritmos de ECD, na resolução de problemas reais. A meta-aprendizagem permite o desenvolvimento de sistemas de recomendações que, de acordo com as caraterísticas presentes num conjunto de dados, sugerem os algoritmos mais promissores para serem aplicados a esses dados.

A investigação na área da meta-aprendizagem sofreu um impulso significativo nos últimos anos, com propostas em vários temas, tais como: definição de meta atributos para domínios de aplicação específicos; novos métodos para a ordenação de algoritmos; e recomendação de valores de parâmetros para algoritmos de ECD. Outro benefício associado à utilização de meta-aprendizagem é a possibilidade de construir mecanismos para a transferência de conhecimento entre diferentes domínios ou tarefas (Brazdil et al., 2009). Assim, o conhecimento adquirido no decurso da aplicação de algoritmos de ECD em problemas oriundos de um determinado domínio, pode ser indiretamente utilizado na aprendizagem em outros domínios, permitindo a transferência de conhecimento entre os respetivos domínios. Alguns conjuntos de dados estão em constante expansão, com novos objetos sendo continuamente gerados. Isto ocorre, em particular, quando os dados são gerados em fluxos contínuos, conforme visto no Capítulo 16. Nesses casos, os algoritmos mais adequados num dado momento podem não ser os mais adequados num momento posterior. Como resultado, a sugestão dos algoritmos indicados precisa ser atualizada quando houver mudanças no perfil dos dados.

Capítulo 18

Decomposição de Problemas Multiclasse

Diversas técnicas de ECD foram originalmente formuladas para a resolução de problemas de classificação contendo apenas duas classes. Entre elas podem-se citar as SVMs e as RNAs Perceptron. Contudo, muitos problemas de classificação são multiclasse, isto é, apresentam mais de duas classes. A generalização de técnicas de classificação binária para problemas multiclasse pode ser realizada basicamente por meio de duas estratégias. A primeira consiste na combinação de preditores gerados em subproblemas binários, enquanto na segunda realizam-se adaptações nos algoritmos originais das técnicas consideradas. A extensão direta de um algoritmo binário a uma versão multiclasse nem sempre é possível ou fácil de realizar. Para as SVMs, em particular, Hsu e Lin (2002) e Rifkin e Klautau (2004) observaram que a reformulação dessa técnica em versões multiclasse conduz a algoritmos computacionalmente pesados. Dessa forma, é comum recorrer-se à alternativa de decompor o problema multiclasse em múltiplos subproblemas binários, uma estratégia denominada de decomposição. As saídas dos preditores binários gerados na solução de cada um desses subproblemas são então combinadas na obtenção da classificação multiclasse final.

Segundo Moreira e Mayoraz (1997), existe uma série de motivações para utilizar estratégias de decomposição na solução de problemas multiclasse. Além de haver técnicas cuja formulação é originalmente binária, alguns algoritmos não são adequados a problemas com um número elevado de classes ou apresentam dificuldades em lidar com grandes volumes de dados de treino. Há ainda algoritmos capazes de processar problemas com múltiplas classes que contêm procedimentos internos restritos a problemas de duas classes (por exemplo: a regra *towing* e a divisão do subconjunto de atributos nominais no CART (Breiman et al., 1984), a divisão do critério de seleção no QUEST (Loh e Shih, 1997)).

Mesmo que o algoritmo seja capaz de trabalhar com problemas de múltiplas classes de grande escala, o uso de um procedimento de decomposição pode reduzir a complexidade computacional envolvida na solução do problema total, por meio da divisão deste em

subtarefas mais simples (Furnkranz, 2002). Knerr et al. (1992), por exemplo, observaram que as classes num problema de reconhecimento de dígitos eram linearmente separáveis quando consideradas em pares.

O recurso a técnicas de decomposição envolve dois passos: uma fase de decomposição, que ocorre antes da aprendizagem, e uma fase de reconstrução, que ocorre depois da predição. Dado um problema de decisão $\mathbf{D} = \{(\mathbf{x}_j, y_j), j = 1, \ldots, n\}$, em que $y \in \{1, ..., k\}$ e $k > 2$, a fase de decomposição consiste em obter múltiplos subproblemas binários na forma de: $\mathbf{B}_i = \{(\mathbf{x}_j, y'_j), j = 1, \ldots, n\}$, em que $y'_j \in \{-1, +1\}$. Um algoritmo de aprendizagem constrói um modelo de decisão para cada problema B_i. Depois, os modelos de decisão são usados para classificar os exemplos de teste. A reconstrução refere-se à forma como as saídas dos classificadores binários são combinadas na determinação da classe de um exemplo.

Na Seção 18.1 são apresentadas algumas das principais estratégias para a decomposição de problemas multiclasse descritos na literatura. Na Seção 18.2, é abordada a obtenção das previsões multiclasse numa etapa de reconstrução. Na Seção 18.3 são apresentadas as considerações finais do capítulo.

18.1 Fase de Decomposição

Várias alternativas podem ser empregues na decomposição de problemas multiclasse num conjunto de problemas de classificação binária. Genericamente, essas decomposições podem ser descritas por uma abordagem proposta por Allwein et al. (2000), em que elas são representadas por uma matriz de códigos $\mathbf{M}$. As linhas dessa matriz contêm códigos que são atribuídos a cada classe. As colunas de $\mathbf{M}$ definem partições binárias das k classes e correspondem aos rótulos que essas classes assumem na geração dos classificadores binários. Tem-se então uma matriz de dimensão $k \times l$, em que k é o número de classes do problema e l representa o número de classificadores binários utilizados na solução multiclasse. Na Figura 18.1 é apresentado um exemplo de matriz de códigos na qual o número de classes k é igual a quatro e o número de classificadores binários l também é quatro. A figura apresenta, além da matriz de códigos, as partições binárias das classes realizadas por cada classificador binário representado em cada coluna.

Cada elemento da matriz $\mathbf{M}$ assume valores em $\{-1, 0, +1\}$. Um elemento m_{ij} com valor $+1$ indica que a classe correspondente à linha i assume rótulo positivo na indução do classificador $\hat{f}_j$. O valor -1 designa um rótulo negativo, e um valor 0 indica que os dados da classe i não participam do processo de indução do classificador $\hat{f}_j$. Para cada coluna, tem-se um conjunto de treino diferente, e classificadores binários são então treinados de forma a aprender os rótulos representados nas colunas de $\mathbf{M}$.

A decomposição mais compacta de um problema com k classes pode ser realizada com o uso de $l = \lceil \log_2(k) \rceil$ classificadores binários (Mayoraz e Moreira, 1996). Um exemplo de matriz compacta para um problema com quatro classes é apresentado na Figura 18.2(a).

O número total de diferentes preditores binários para um problema com k classes é $0,5 \times \left(3^k + 1\right) - 2^k$, considerando que $\hat{f} = -\hat{f}$, ou seja, que a inversão das classes positivas

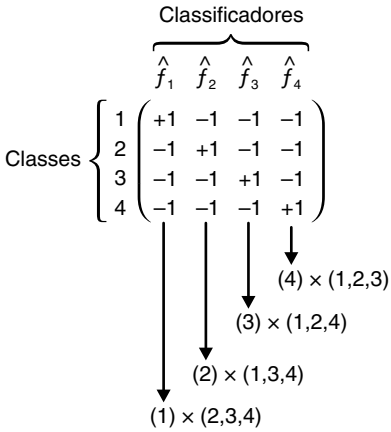


Figura 18.1 *Exemplo de uma matriz de códigos para um problema com quatro classes.*

e negativas produz o mesmo classificador. Desses, $2^{k-1}-1$ classificadores incluem todas as classes simultaneamente, ou seja, possuem somente rótulos $+1$ e -1, sem o elemento 0. Para quatro classes, tem-se nesse caso a matriz representada na Figura 18.2(b).

Entre as estratégias de decomposição mais comuns encontradas na literatura estão a um-contra-todos (Rifkin e Klautau, 2004), a todos-contra-todos (Hastie e Tibshirani, 1998; Furnkranz, 2002, 2003; Moreira, 2000), a baseada em códigos de correção de erros (*Error Correcting Output Codes* - ECOC) (Dietterich e Bariki, 1995) e as decomposições hierárquicas, que são descritas a seguir.

18.1.1 Um-contra-todos

Na estratégia um-contra-todos (OAA, do inglês *one-against-all*), dado um problema com k classes, k classificadores binários $\hat{f}_i(\mathbf{x})$ são gerados. Cada um desses preditores é treinado de forma a distinguir uma classe i das demais. A representação dessa técnica é dada por uma matriz de dimensão $k \times k$, na qual os elementos da diagonal possuem o valor $+1$ e os demais, o valor -1. Uma matriz do tipo OAA para um problema de quatro classes é apresentada na Figura 18.2(c).

A decomposição OAA pode apresentar desvantagens quando a proporção de exemplos de uma classe é muito pequena em relação à do conjunto formado pelos dados das outras classes. Esse tipo de desbalanceamento pode dificultar a indução de um preditor que apresente bom desempenho no reconhecimento da classe considerada.

18.1.2 Todos-contra-todos

Na decomposição todos-contra-todos, também denominada um-contra-um (OAO, do inglês *one-against-one*) e em pares (*pairwise*), dadas k classes, $\frac{k(k-1)}{2}$ classificadores biná-

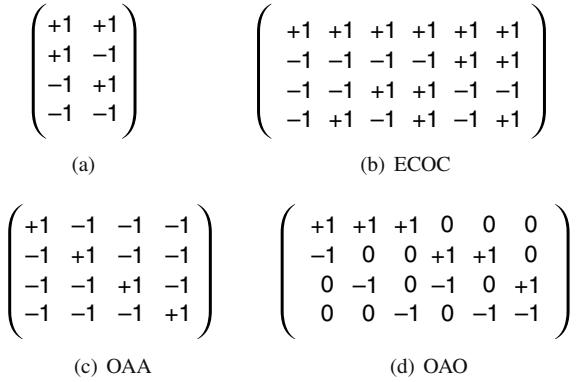


Figura 18.2 *Matrizes de diferentes decomposições para um problema com quatro classes.*

rios são gerados. Cada um deles é responsável por diferenciar um par de classes (i, j), em que $i \neq j$. A matriz de códigos nesse caso possui então dimensão $k \times \frac{k(k-1)}{2}$, e cada coluna corresponde a um classificador binário para um par de classes. Numa coluna representando o par (i, j), o valor do elemento correspondente à linha i é $+1$, e o valor do membro correspondente a j é igual a -1. Todos os outros elementos da coluna possuem o valor 0, indicando que os dados de outras classes não participam do processo de indução do classificador. A Figura 18.2(d) apresenta uma matriz OAO para um problema com quatro classes.

Embora o número de classificadores gerados na decomposição OAO seja da ordem de k^2, o treino de cada um deles envolve dados de apenas duas classes. Com isso, mesmo com um número elevado de classes, o tempo total despendido na geração dos preditores geralmente não é grande.

Um problema apontado na decomposição OAO é que a resposta de um preditor para um par de classes (i, j) na realidade não fornece informação nenhuma quando o exemplo não pertence às classes i ou j. Suponha um problema com 10 classes. Dos 45 problemas de decisão binária, somente 9 podem classificar corretamente o exemplo teste. Todos os outros 36 irão classificar o exemplo de forma errada. Portanto, nessa proposta, $(k-1)(k-2)/2$ irão classificar incorretamente qualquer exemplo, e somente $k-1$ podem prover a classificação correta.

18.1.3 Códigos de Correção de Erros de Saída

A transmissão da informação num canal com ruído pode envolver perda de informação. Com o desenvolvimento da tecnologia digital, é possível ao recetor da mensagem detetar e corrigir erros. A ideia base consiste em codificar a mensagem previamente à transmissão, em vez de a transmitir no seu formato original. A codificação envolve a introdução

de alguma redundância. A mensagem codificada é enviada através do canal ruidoso. O recetor descodifica a mensagem, e, devido à redundância de eventuais erros, estes podem ser detetados e corrigidos. Este é o contexto em que os códigos de correção de erros aparecem. Shannon (1948) foi o primeiro a mostrar a possibilidade de usar esses códigos em comunicação, obtendo um bom equilíbrio entre redundância e capacidade de recuperar erros. Posteriormente, Hamming (1950) apresentou a matriz Hamming, usada para codificar 4 bits de informação usando 7 bits:

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Os 7 bits usados para codificar uma mensagem de 4 bits são: $C(d_1,d_2,d_3,d_4) = (d_1+d_2+d_4, d_1+d_3+d_4, d_1, d_2+d_3+d_4, d_2, d_3, d_4)$. Por exemplo, a mensagem 1001 é codificada em: 0011001. Suponha que a mensagem seja recebida como: $\mathbf{m} = 0010001$. A matriz Hamming indica se houve um erro e onde o erro está. Se não há erro, $\mathbf{Hm} = 0$; senão, podemos determinar onde está o erro, calculando:

$$\mathbf{Hm} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Rodando **Hm** no sentido horário, obtemos 100, indicando o quarto bit como erro.

Dietterich e Bariki (1995) propuseram o uso de códigos de correção de erros para representar as k classes de um problema multiclasse em ECD. Esta técnica é designada *decomposição por códigos de correção de erros* (*Error Correcting Output Codes* – ECOC). Neste caso, as matrizes de códigos assumem valores em $\{-1,+1\}$. Mesmo com a substituição dos valores 0 por -1, podemos notar que a matriz de Hamming **H** não pode ser diretamente usada no contexto de ECOC para classificação: a última coluna não definirá um problema de decisão, e várias colunas são complementares (como a primeira e a sexta colunas, por exemplo).

A possibilidade de correção de erros requer que os códigos das classes contidos em **M** sejam bem separados segundo a distância de Hamming. Sendo d_l a distância mínima entre qualquer par de linhas de **M**, então o classificador multiclasse final é capaz de corrigir pelo menos $\left\lfloor \frac{d_l-1}{2} \right\rfloor$ bits incorretos numa predição. Além disso, na construção de bons códigos de correção de erros, deve-se estimular também que os erros dos classificadores binários gerados sejam não correlacionados. Exige-se então a separação entre as colunas de **M**, ou seja, a distância de Hamming entre cada par de colunas deve ser grande. Se no algoritmo de aprendizagem a inversão das classes positivas e negativas produz o mesmo classificador (ou seja, $\hat{f} = -\hat{f}$), então deve-se também fazer com que a distância de Hamming entre cada coluna e o complemento das outras seja grande. Por fim, nenhuma coluna deve ser

constante (composta somente por $+1$ ou -1), pois, deste modo, não se gera um problema de decisão.

Com base nestas observações, Dietterich e Bariki (1995) propuseram quatro técnicas para o desenvolvimento de matrizes de códigos com boa capacidade de correção de erros. A escolha de cada uma delas é determinada pelo número de classes do problema. Para $k \leqslant 7$, estes autores recomendam o uso de um código exaustivo, que consiste na combinação de todos os $2^{k-1}-1$ classificadores binários unicamente com rótulos $+1$ e -1, conforme ilustrado na Figura 18.2(b) para um problema com quatro classes. A distância d_l numa matriz gerada pelo método exaustivo é de 2^{k-2}. Se $8 \leqslant k \leqslant 11$, é aplicado um método que seleciona colunas do código exaustivo. Para $k > 11$, tem-se duas opções: um método baseado no algoritmo *hill-climbing* e a geração de códigos BCH (Boser e Ray-Chaudhuri, 1960), os quais são oriundos da teoria de códigos de correção de erros em comunicação. Há outros trabalhos mais recentes com estratégias alternativas para construir ECOCs, entre os quais Pimenta et al. (2007) e Tapia et al. (2010).

Allwein et al. (2000) apontam que, apesar de os códigos gerados pelo ECOC possuírem boa propriedade de correção de erros, vários dos subproblemas binários criados podem ser difíceis de aprender. Por esse motivo, as técnicas mais simples OAA e OAO têm apresentado resultados comparáveis ou superiores ao ECOC em várias aplicações (Allwein et al., 2000; Rifkin e Klautau, 2004).

18.1.4 Decomposições Hierárquicas

Uma maneira alternativa de solucionar um problema multiclasse com preditores binários pode ser conduzida com a sua decomposição hierárquica. De forma geral, a introdução de uma hierarquia numa aplicação multiclasse pode reduzir a complexidade de sua solução. A ideia é realizar inicialmente discriminações mais gerais, as quais são refinadas sucessivamente até a obtenção da classificação final.

Na Figura 18.3 são apresentados dois exemplos de decomposições hierárquicas utilizadas na solução de um problema com quatro classes. O classificador da Figura 18.3(a) possui uma estrutura de árvore direcionada binária, em que exatamente uma aresta chega a cada nó e no máximo duas arestas partem deles. Na Figura 18.3(b) tem-se uma estrutura mais geral, de um grafo direcionado acíclico, em que mais de uma aresta pode apontar para um mesmo nó. Por definição, as árvores direcionadas binárias são um tipo de grafo direcionado acíclico. Porém, para simplificar a exposição realizada nesta seção, estas estruturas serão tratadas como sendo de tipos distintos. Em ambas as estruturas, cada nó interno corresponde a um classificador binário que distingue dois subconjuntos de classes, enquanto os nós terminais, denominados folhas, representam as classes individuais.

Os preditores contidos nas decomposições hierárquicas também podem ser representados por uma matriz de códigos. O grafo da Figura 18.3(b), por exemplo, corresponde à mesma matriz da decomposição OAO (Figura 18.2(d)).

Para a obtenção das hierarquias, que equivale à etapa de decomposição do problema, várias estratégias podem ser empregues. Algumas delas são discutidas a seguir. Inicial-

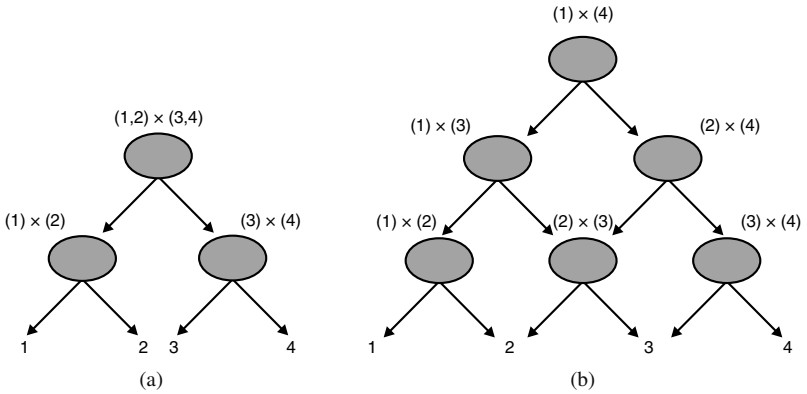


Figura 18.3 *Decomposições hierárquicas para um problema com quatro classes.*

mente são apresentadas abordagens com estrutura de grafos direcionados acíclicos, seguidas de estruturas de árvores direcionadas binárias.

Grafos Direcionados Acíclicos

Platt et al. (2000) sugerem que os classificadores produzidos pela decomposição OAO sejam dispostos num grafo de decisão direcionado acíclico (DDAG, do inglês *Decision Directed Acyclic Graph*). Logo, cada nó do grafo corresponde a um preditor binário para um par de classes. Essa estratégia será referenciada como DDAG neste documento.

A Figura 18.3(b) ilustra um exemplo de DDAG para um problema com quatro classes. Uma desvantagem do DDAG, apontada por Kijsirikul e Ussivakul (2002), é a sua dependência em relação à sequência de classificadores binários contidos nos nós do grafo. Essa característica afeta a sua confiabilidade, uma vez que diferentes permutações de nós no grafo podem produzir resultados distintos. Outra deficiência do DDAG é que, dependendo da posição da classe correta no grafo, o número de avaliações com essa classe é desnecessariamente grande, resultando num erro cumulativo elevado.

Esses fatores motivaram Kijsirikul e Ussivakul (2002) a desenvolver uma nova estratégia hierárquica para combinar as saídas produzidas por classificadores obtidos pela decomposição OAO. A nova estrutura, denominada DAG Adaptável (ADAG, do inglês *Adaptive Directed Acyclic Graph*), corresponde a um DDAG com estrutura reversa. Na Figura 18.4 é apresentado um exemplo de ADAG para um problema com oito classes. O ADAG tem $k-1$ nós, cada qual correspondendo a um classificador binário para um par de classes. A primeira camada tem $\lceil k/2 \rceil$ nós, seguidos por $\lceil k/2^2 \rceil$ na segunda camada e assim por diante, até que uma camada com um único nó é atingida, a qual produz a saída final. Os nós da primeira camada diferenciam pares distintos de todas as classes. No caso de o número de classes ser ímpar, um dos nós contém apenas uma classe, a qual é diretamente

passada para o segundo nível. Os níveis seguintes são adaptativos, sendo determinados de acordo com as previsões realizadas pelos classificadores dos níveis anteriores.

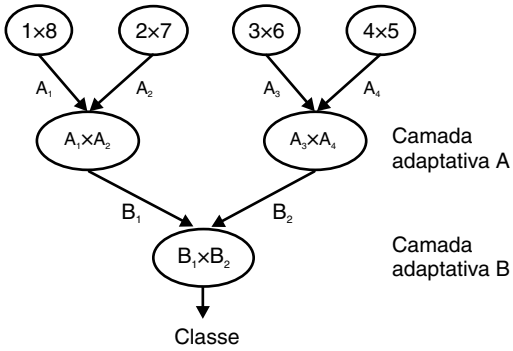


Figura 18.4 *Exemplo de ADAG para problema com oito classes (Kijsirikul e Ussivakul, 2002).*

O ADAG também foi proposto num trabalho diferente como uma analogia a um torneio de tênis (Pontil e Verri, 1998). Primeiramente, todos os jogadores (classes) fazem partidas em pares. Com base no resultado dessas partidas, uma nova fase do torneio é iniciada, na qual os vencedores das partidas anteriores são emparelhados de acordo com a sua chave. Este processo é repetido até que reste apenas uma única classe.

O ADAG minimiza a ocorrência de erros cumulativos, especialmente em problemas com um número de classes elevado. Embora o ADAG também tenha demonstrado menor dependência que o DDAG em relação à ordem dos classificadores binários no grafo, ainda ocorrem diferenças de desempenho entre estruturas distintas. Logo, a definição dos classificadores que compõem o primeiro nível do ADAG afeta o seu resultado.

Alguns trabalhos investigaram heurísticas para determinar as estruturas de ADAGs e/ou DDAGs, por exemplo: Phetkaew et al. (2003), Takahashi e Abe (2003) e Lorena e Carvalho (2007). O número possível de diferentes DDAGs para um problema com k classes é $k!/2$ e de ADAGs é $k!/2^{\lfloor k/2 \rfloor}$. Na obtenção desses valores, classificadores para os pares de classes (i, j) são considerados equivalentes àqueles para os pares (j, i), ou seja, considera-se $\hat{f} = -\hat{f}$.

Árvores Direcionadas Binárias

Diversos trabalhos utilizam uma estrutura em forma de árvore direcionada binária na obtenção do classificador multiclasse hierárquico. Tais estruturas também são denominadas dicotomias embutidas. As árvores possuem $k-1$ classificadores binários e, portanto, envolvem o treino de $k-1$ preditores. Além disso, os nós de níveis inferiores envolvem cada vez menos classes e, portanto, menos dados de treino para os classificadores binários correspondentes.

Para um problema com $k \geqslant 3$ classes, existem $\prod_{i=3}^{k} 2i-3$ estruturas de árvores distintas

(Frank e Kramer, 2004). Duas possíveis árvores para um problema com quatro classes são ilustradas na Figura 18.5. Tal como no DDAG e no ADAG, a estrutura da árvore, ou seja, que classificadores são dispostos na árvore e a sua posição, influencia o seu resultado. Os trabalhos nesta área diferenciam-se então no processo de obtenção das partições binárias das classes em cada nó da árvore e, consequentemente, na determinação da sua estrutura. Todos aplicam algum critério recursivamente sobre subconjuntos de classes, particionando-os em dois até que eles possuam apenas uma classe.

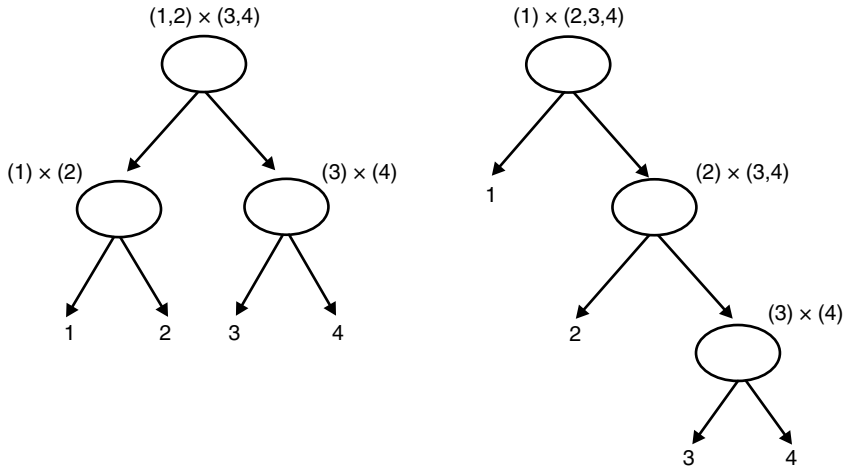


Figura 18.5 *Duas árvores para um problema com quatro classes (Frank e Kramer, 2004).*

Kumar et al. (2002), por exemplo, propõem que as partições das classes sejam feitas de maneira a maximizar a discriminação entre os grupos de classes formados. Para esse efeito, estes autores usam uma generalização do algoritmo de agrupamento *k*-médias que combina ideias de otimização por *cristalização simulada* (*simulated annealing*). Lorena e Carvalho (2008) propuseram um algoritmo para definir as partições binárias de classes numa árvore que realiza um agrupamento hierárquico das classes de acordo com a sua semelhança. Diferentes critérios podem ser utilizados para medir a semelhança entre as classes, tal como a distância entre seus centroides, a sua separabilidade (Lorena e Carvalho, 2010), entre outros.

18.2 Fase de Reconstrução

A fase de reconstrução define como as saídas dos classificadores binários serão combinadas na obtenção das previsões multiclasse. Nesse processo, um novo exemplo $\mathbf{x}$ pode ser classificado através da avaliação das predições dos l classificadores, que geram um vetor $\hat{\mathbf{f}}(\mathbf{x})$ de tamanho l na forma $\hat{\mathbf{f}}(\mathbf{x}) = (\hat{f}_1(\mathbf{x}), \ldots, \hat{f}_l(\mathbf{x}))$. Este vetor é então comparado com as linhas de $\mathbf{M}$. O exemplo é classificado na classe cuja linha de $\mathbf{M}$ é mais próxima de

$\hat{\mathbf{f}}(\mathbf{x})$ de acordo com alguma medida. Seja $\mathbf{m}_q$ a q-ésima linha de $\mathbf{M}$, a qual apresenta o código referente à classe q. O processo de descodificação equivale a calcular a Equação 18.1, em que $\hat{f}(\mathbf{x})$ representa o classificador multiclasse final e d denota uma função de descodificação.

$$\hat{f}(\mathbf{x}) = \underset{1 \leqslant q \leqslant k}{\arg\min} \left(d\left(\mathbf{m}_q, \hat{\mathbf{f}}(\mathbf{x})\right)\right) \tag{18.1}$$

Caso mais de um código minimize a Equação 18.1, temos uma situação de empate. Escolhe-se então aleatoriamente uma das classes envolvidas no empate ou com o uso de alguma informação *a priori*.

Existem diversas funções de descodificação que podem ser utilizadas na integração dos classificadores binários na Equação 18.1 (Passerini et al., 2004; Allwein et al., 2000; Windeatt e Ghaderi, 2003). A mais simples é a de Hamming, que conta o número de ocorrências diferentes entre o vetor $\hat{\mathbf{f}}(\mathbf{x})$ e cada um dos códigos. Esta função pode ser visualizada na Equação 18.2, em que sgn corresponde à função sinal. Um rótulo 0 contribui com $\frac{1}{2}$ no calculo da soma apresentada. A função descrita equivale à distância de Hamming caso $\mathbf{M}$ possua unicamente os rótulos $+1$ e -1.

$$d_H\left(\mathbf{m}_q, \hat{\mathbf{f}}(\mathbf{x})\right) = \sum_{i=1}^{l} \frac{1 - \operatorname{sgn}\left(m_{qi} * \hat{f}_i(\mathbf{x})\right)}{2} \tag{18.2}$$

Usualmente a classificação de um novo exemplo na estratégia OAA corresponde a escolher o classificador com maior saída. Isto equivale a usar alguma função de descodificação que considere a confiança dos classificadores nas predições obtidas. No ECOC de Dietterich e Bariki (1995) propõe-se o uso da descodificação pela função de Hamming. Finalmente, para o OAO, a agregação mais usual é por uma votação por maioria. Dado um novo exemplo $\mathbf{x}$, cada classificador vota numa classe preferida. O resultado final é dado pela classe que recebeu mais votos. Esta estratégia pode ser implementada pelo uso da distância de Hamming sobre a matriz de códigos da estratégia OAO.

No caso das estratégias hierárquicas, nem todos os classificadores binários necessitam de ser avaliados numa predição, ao contrário do apresentado anteriormente. Os classificadores consultados são determinados a partir das predições realizadas para cada exemplo de teste, nível a nível nas hierarquias. Em geral, pode-se afirmar que as estratégias hierárquicas apresentam tempos menores de predição do que as estratégias OAA, OAO com votação por maioria e ECOC. Isso ocorre porque, na classificação de um dado, apenas uma parte dos preditores binários é consultada.

Pode-se verificar que, na classificação de um novo exemplo com o DDAG e o ADAG, $k-1$ classificadores binários são avaliados. Estas estruturas aceleram a fase de predição da abordagem OAO tradicional usando votação por maioria. No ADAG, a classe correta é testada contra as outras classes no máximo $\lceil \log_2 k \rceil$ vezes, enquanto no DDAG pode requerer até $k-1$ avaliações com essa classe. No caso das árvores direcionadas binárias, no melhor caso, dependendo da estrutura da árvore, é possível classificar o exemplo logo

no primeiro nó. No pior caso, os $k-1$ classificadores devem ser consultados. Logo, a fase de teste pode ser acelerada nesta estrutura perante as anteriormente discutidas.

A descodificação utilizada na matriz de códigos de estruturas hierárquicas é conhecida como eliminação (Klautau et al., 2003) e procede iterativamente. Inicialmente todos os classificadores binários (colunas da matriz) são considerados ativos. Numa determinada iteração, um deles é consultado. As classes que "perdem" são eliminadas, assim como os classificadores binários relacionados. Este processo é repetido até que reste um único classificador binário, que fornece a classificação final.

18.3 Considerações Finais

Embora algumas técnicas de aprendizagem, como as SVMs, sejam originalmente formuladas para a solução de problemas de classificação binários, o seu uso pode ser estendido a aplicações multiclasse. Em geral, dois tipos de estratégias podem ser empregues neste processo: decomposição e diretas. As decomposicionais dividem o problema total em múltiplos subproblemas binários, cujas saídas são combinadas na obtenção da previsão multiclasse. Nas estratégias diretas, reformula-se o algoritmo original da técnica de aprendizagem numa versão multiclasse. Porem, esse procedimento nem sempre é simples de ser realizado. Neste capítulo foram revistas as principais estratégias decomposicionais propostas na literatura. É importante ressaltar que o estudo das estratégias decomposicionais se aplica a qualquer técnica de aprendizagem, bastando para isso utilizá-la na geração de classificadores binários.

Além das abordagens mencionadas neste capítulo, existem também trabalhos que combinam as saídas dos preditores binários por meio do uso de outro classificador. Esse classificador, que pode ser produzido por outra técnica de aprendizagem distinta da utilizada na obtenção dos preditores binários, é treinado de forma a ponderar as saídas dos classificadores binários na previsão multiclasse. Entre os trabalhos que utilizaram este tipo de estratégia, podem-se mencionar: Mayoraz e Alpaydim (1998) e Savicky e Fürnkranz (2003).

Também é possível obter probabilidades de classificação a partir das previsões de vários classificadores binários. Entre os trabalhos nessa direção, podem-se mencionar Hastie e Tibshirani (1998), Zadrozny (2001), Passerini et al. (2004) e Wu et al. (2004).

Capítulo 19

Classificação Multirótulo

Em geral, os classificadores induzidos por algoritmos de ECD associam uma única classe ou rótulo a cada exemplo. Estes classificadores serão denominados neste capítulo classificadores de um único rótulo (ou simples-rótulo). Nestes problemas, um classificador é treinado num conjunto de exemplos, em que cada exemplo $\mathbf{x}_i$ é associado a uma única classe y_i de um conjunto R de k classes disjuntas.

Entretanto, existe um grupo de problemas reais de classificação, conhecidos como problemas de classificação multirótulo, em que cada exemplo pode pertencer simultaneamente a mais do que uma classe. Um classificador multirótulo pode ser formalmente definido como uma função $H : X \rightarrow 2^k$ que mapeia cada objeto $\mathbf{x}_i$ num conjunto de classes ou rótulos $\mathbf{y}_i$, em que y_i é um vetor binário com k elementos. Cada elemento y_i^j tem valor 1 se o elemento pertence à j-ésima classe e 0 caso contrário. Assim, $1 < sum(\mathbf{y}_i) < k$, em que $sum(\mathbf{y}_i|)$ é o número de rótulos com valor 1 em $\mathbf{y}_i$.

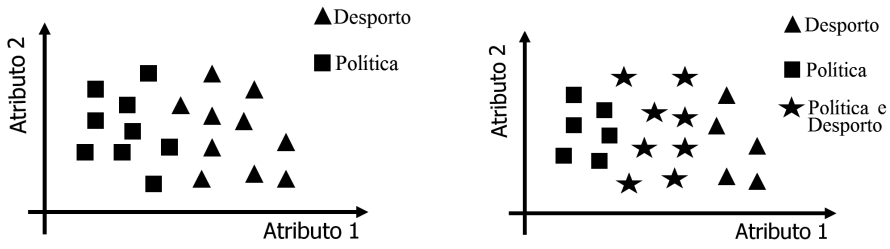
Problemas de classificação multirótulo ocorrem com frequência nas áreas de Bioinformática e processamento de textos. O estudo de métodos de classificação multirótulo tem sido motivado, principalmente, pelas tarefas de classificação de textos (Gonçalves e Quaresma, 2003; Lauser e Hotho, 2003; Luo e Zincir-Heywood, 2005). Num problema de classificação de textos, cada documento pode pertencer simultaneamente a mais do que uma classe (ou tópico). Um documento, por exemplo, pode ser classificado como pertencente à área de Ciência da Computação e Física. Um artigo de jornal que aborda as reações de políticos a pacotes económicos pode ser classificado ao mesmo tempo nas categorias Política e Economia.

Um exemplo de problema de classificação multirótulo na área de Bioinformática é a classificação da função de proteínas, pois uma proteína pode ter mais do que uma função.

Problemas de classificação multirótulo podem também ser encontrados em várias outras áreas, como diagnóstico médico (Karalic e Pirnat, 1991; Tsoumakas e Katakis, 2007), classificação de imagens (Boutell et al., 2004; Shen et al., 2004) e bioinformática (Clare e King, 2001; Zhang e Zhou, 2005; Elisseeff e Weston, 2001b). Na área de diagnóstico médico, um paciente pode sofrer de diabetes e gripe ao mesmo tempo. Um problema de classificação multirótulo de imagens é a classificação de fotografias de paisagens, quando

uma mesma imagem pode ser classificada, por exemplo, como de bosque e de pôr-do-sol. Um exemplo de aplicação na área de Bioinformática é a definição das funções de uma proteína, uma vez que uma proteína pode desempenhar simultaneamente várias funções. Portanto, o estudo de técnicas de classificação capazes de fornecer uma solução eficiente para estes problemas pode acarretar grandes benefícios em diferentes áreas.

A Figura 19.1 apresenta uma comparação entre um problema de classificação convencional, onde os exemplos podem ser associados a apenas uma classe, e um problema de classificação multirótulo. A Figura 19.1(a) ilustra um problema de classificação onde os exemplos pertencem ou à classe ■ ou à classe ▲, mas nunca às duas classes ao mesmo tempo. A Figura 19.1(b) mostra um exemplo de classificação multirótulo onde os exemplos pertencentes simultaneamente às classes ■ e ▲ são representados por ★.



(a) Típico (b) multirótulo

Figura 19.1 *Problemas de classificação*

Três abordagens foram propostas na literatura para tratar problemas de classificação multirótulo. Numa delas, classificadores simples-rótulo são combinados para tratar problemas de classificação multirótulo. Na segunda, os classificadores simples-rótulo são modificados através de adaptações nos seus mecanismos internos, de forma a poderem ser utilizados em problemas multirótulo. Na última abordagem, novos algoritmos são desenvolvidos especificamente para tratar problemas de classificação multirótulo (de Carvalho e Freitas, 2009).

Este capítulo está estruturado da seguinte forma. Na Seção 19.1 são discutidas as principais abordagens para lidar com problemas de classificação multirótulo. Dois conceitos importantes para classificação multirótulo, de cardinalidade de rótulo e densidade de rótulo, são explicados na Seção 19.2. A Seção 19.3 apresenta algumas das principais medidas de avaliação utilizadas nesta área. As considerações finais deste capítulo são objeto da Seção 19.4.

19.1 Principais Abordagens

Geralmente, quando se treina um classificador, é possível associar uma probabilidade a cada uma das classes existentes no problema e utilizá-la na classificação de um novo exem-

plo. Se o problema de classificação tem k classes, uma probabilidade p_i, com $1 \leq i \leq k$, na qual $0 \leq p_i \leq 1$, pode ser atribuída a cada classe. Quando o classificador é treinado num problema de classificação simples-rótulo, há uma restrição que diz que $\sum p_i = 1$. Em problemas de classificação multirótulo, não se introduz esta restrição não é adotada (de Carvalho e Freitas, 2009). Problemas de classificação binária e multiclasse podem ser considerados casos especiais de problemas de classificação multirótulo, nos quais o número de classes atribuídas a cada exemplo é igual a 1 (Elisseeff e Weston, 2001a).

A Figura 19.2 ilustra os diferentes métodos propostos na literatura para tratar problemas de classificação multirótulo (de Carvalho e Freitas, 2009). De acordo com a figura, estes métodos podem ser divididos em duas grandes abordagens: abordagem independente do algoritmo e abordagem dependente do algoritmo. Como pode ser observado na figura, a abordagem independente do algoritmo utiliza algoritmos tradicionais de classificação para tratar problemas multirótulo, transformando o problema multirótulo original num conjunto de problemas simples-rótulo. A abordagem dependente do algoritmo cria algoritmos específicos para tratar o problema multirótulo. Estes algoritmos podem ser baseados em técnicas de classificação convencionais, como SVMs e árvores de decisão, ou podem ser especificamente desenvolvidos para classificação multirótulo.

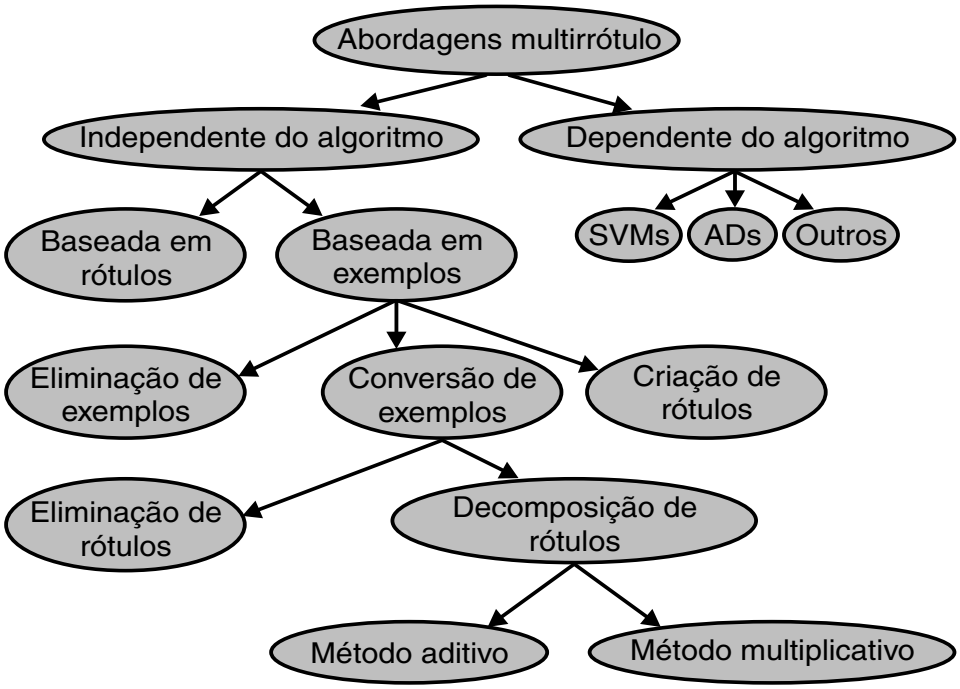


Figura 19.2 *Métodos para classificação multirótulo.*

19.1.1 Abordagem Independente de Algoritmo

Nesta abordagem, qualquer algoritmo tradicional de classificação pode ser utilizado para tratar o problema. Para isso, basta transformar o problema multirótulo original num conjunto de problemas de classificação simples-rótulo. Essa transformação pode ser baseada nos rótulos das classes dos exemplos ou nos próprios exemplos de treino.

Transformação Baseada nos Rótulos das Classes

Neste tipo de transformação, são utilizados k classificadores, sendo k o número de classes do problema. Cada classificador é associado a uma classe e treinado para resolver um problema de classificação binária, que discrimina essa classe de todas as outras classes envolvidas. Este método é também chamado de método binário ou um-contra-todos (Tsoumakas e Vlahavas, 2007).

Um ponto fraco deste método é que assume que as classes atribuídas a um exemplo são independentes entre si. Isso nem sempre se verifica, e ignorar as possíveis correlações entre as classes pode fazer com que o método tenha pouca capacidade de generalização.

O processo de transformação desse método é reversível, ou seja, é possível recuperar as classes do problema original a partir do novo problema criado.

Transformação Baseada nos Exemplos

Nesta abordagem, o conjunto de classes associado a cada exemplo é redefinido, de maneira a converter o problema multirótulo original num ou mais problemas simples-rótulo. Contrariamente ao método anterior, este método não produz apenas problemas de classificação binária, podendo produzir problemas tanto binários quanto multiclasse.

Três estratégias diferentes têm sido propostas para este tipo de tranformação (de Carvalho e Freitas, 2009):

- Eliminação de exemplos multirótulo;
- Criação de novos rótulos para os exemplos multirótulo existentes;
- Conversão de exemplos multirótulo em exemplos simples-rótulo.

Eliminação de Exemplos multirótulo

A estratégia mais simples que existe para a transformação baseada em exemplos, mas também a mais ineficaz, consiste em eliminar do conjunto de dados os exemplos que são multirótulo. A eliminação dos exemplos com mais de uma classe não resolve o problema multirótulo original, uma vez que apenas o transforma num problema mais simples, de menor relevância que o original.

Um exemplo de perda de informação causada por esta estratégia pode ser dado por um problema de classificação de proteínas. As proteínas podem desempenhar mais do que uma função, e eliminar essas proteínas do conjunto de dados reduz a significância do classificador (de Carvalho e Freitas, 2009), que não teria como predizer as outras funções

de uma proteína. Esta transformação é irreversível, pois não é possível descobrir, no novo problema criado, quais os exemplos que foram eliminados do problema original. O número de classificadores necessários também não é alterado.

Criação de Novos Rótulos para os Exemplos multirótulo Existentes

Nesta estratégia, para cada exemplo, todas as classes atribuídas ao exemplo são combinadas numa nova e única classe. Com esta combinação, o número de classes envolvidas no problema pode aumentar consideravelmente, e algumas classes podem terminar com poucos exemplos que as representem.

Com a criação de novas classes, as classes do problema original não se perdem. Se forem utilizados classificadores multiclasse no novo problema gerado, a quantidade desses classificadores mantém-se inalterada. Se forem utilizados classificadores binários, o número de classificadores necessários aumenta.

Conversão de Exemplos multirótulo em Exemplos Simples-rótulo

Existem duas variações para esta estratégia. Na primeira, todos os exemplos multirótulo são convertidos em exemplos simples-rótulo, num processo denominado de simplificação ou eliminação de rótulos. Uma segunda variação, denominada de decomposição de rótulos, decompõe todos os exemplos multirótulo num conjunto de exemplos simples-rótulo.

Na simplificação de rótulos, quando um exemplo possui mais do que uma classe, uma dessas classes é escolhida e as restantes são eliminadas. Esta escolha pode ser feita de forma determinística, selecionando, dentre as classes às quais o exemplo pertence, aquela que possui maiores hipóteses de ser a classe verdadeira; ou pode ser feita de forma aleatória, selecionando uma classe aleatoriamente.

Se na escolha das classes for adotado um critério determinístico, é possível retornar ao problema multirótulo original a partir do problema simples-rótulo criado. Se as classes forem escolhidas de forma aleatória, esse retorno não é possível. O número de classificadores utilizados no problema multirótulo e no simples-rótulo geralmente é o mesmo.

No processo de decomposição de rótulos, um problema multirótulo com k classes e n exemplos é dividido em ps conjuntos de problemas simples-rótulo. O valor de ps varia de 1, quando nenhum exemplo possui mais do que uma classe, a $(k-1)^n$, se todos os exemplos possuem $k-1$ classes. O processo de decomposição pode ser dividido em dois métodos: aditivo e multiplicativo (de Carvalho e Freitas, 2009).

No método aditivo, o número de classificadores é igual ao número de classes que rotulam pelo menos um exemplo multirótulo. Este método permite que o problema multirótulo original seja recuperado a partir do problema simples-rótulo criado.

No método multiplicativo, por outro lado, é utilizada uma combinação de todos os possíveis problemas simples-rótulo. Este método é semelhante ao método todos-contra-todos (explicado no Capítulo 18), utilizado para dividir um problema multiclasse num conjunto

de problemas binários. O número de classificadores utilizados no método multiplicativo é igual a $\prod k_i$, que é o produto do número de classes presentes em cada exemplo do conjunto de dados.

O número de classificadores deste método cresce exponencialmente com o número de classes de cada exemplo do conjunto de dados. É possível notar também que o método aditivo produz um subconjunto de problemas simples-rótulo produzidos pelo método multiplicativo. O método multiplicativo também é reversível, permitindo a recuperação do problema multirótulo original. O método multiplicativo também minimiza a deficiência dos métodos que eliminam ou combinam rótulos e perdem informação. Porém, não toma em consideração possíveis relações entre os rótulos de classes de um mesmo exemplo.

Muitos dos métodos propostos para tratar problemas multirótulo de forma independente fazem uso de SVMs (Cristianini e Shawe-Taylor, 2000). No trabalho de Pavlidis e Grundy (1999) foram utilizadas SVMs para a classificação multirótulo de funções de genes, utilizando decomposição binária. Para as experiências, os autores utilizaram uma base de dados heterogénea, consistindo em dados de expressão genética e perfis filogenéticos. De acordo com os autores, estes dois tipos de dados proporcionam uma visão mais exata de subconjuntos sobrepostos de categorias de funções genéticas presentes numa célula, resultando num melhor desempenho na classificação. Foi observado também que essa melhoria não é uniformemente distribuída entre as classes presentes no problema, portanto essa combinação só deve ser utilizada se houver evidência do seu benefício.

Em Su et al. (2005), um problema multirótulo foi decomposto num conjunto de problemas binários utilizando a estratégia um-contra-todos. As experiências foram realizadas utilizando um conjunto de dados de predição da localização subcelular de proteínas.

19.1.2 Abordagem Dependente do Algoritmo

Na abordagem dependente do algoritmo, como o próprio nome sugere, novos algoritmos são propostos para tratar os problemas de classificação multirótulo como um todo, numa única etapa. Um algoritmo específico, desenvolvido para um determinado problema de classificação real e difícil, pode apresentar resultados melhores do que métodos que seguem a abordagem independente do algoritmo. Embora tenham sido propostas diversas soluções baseadas nesta abordagem, as principais dizem respeito a algoritmos de indução de árvores de decisão.

Um novo método de classificação baseado em árvores de decisão, denominado de "Árvore de Decisão Alternada" (ADT), foi proposto por Freund e Mason (1999). Esse método é uma generalização das árvores de decisão, e o seu princípio indutivo é baseado no método *boosting* (Freund e Schapire, 1999). Uma extensão do método ADT foi proposta por de Comite et al. (2003), e é baseada nos métodos *AdaBoost* (Freund e Schapire, 1995) e *ADTBoost* (Freund e Mason, 1999). Esse algoritmo estende o ADT pela decomposição de problemas multiclasse usando a abordagem um-contra-todos (de Carvalho e Freitas, 2009).

Outro trabalho que utiliza árvores de decisão foi proposto por Clare e King (2001). Neste trabalho, os autores modificaram o algoritmo C4.5 (Quinlan, 1993) para a classi-

ficação de proteínas de acordo com as suas funções. O algoritmo C4.5 define os nós da árvore de decisão através de uma medida chamada entropia. Os autores modificaram a fórmula dessa medida, originalmente elaborada para problemas simples-rótulo, de forma a permitir a sua utilização em problemas multirótulo. Outra modificação foi a utilização dos nós folha da árvore para representar conjuntos de rótulos de classes. Quando um nó folha, alcançado na classificação de um exemplo, contém um conjunto de classes, uma regra separada é produzida para cada classe (de Carvalho e Freitas, 2009).

Em Zhang e Zhou (2005) é proposto um novo método para classificação multirótulo baseado no algoritmo *k*-NN, denominado ML-*k*-NN. Nesse método, para cada exemplo, as classes associadas aos *k* exemplos vizinhos mais próximos são recuperadas, e é feita uma contagem dos vizinhos associados a cada classe. Então, o princípio *maximum a posteriori* (Saridis, 1983) é utilizado para definir o conjunto de classes de um novo exemplo.

Em Schapire e Singer (1999, 2000) são propostas duas extensões para o algoritmo *Adaboost* (Freund e Schapire, 1995), de maneira a permitir a sua utilização em problemas multirótulo. Na primeira, é feita uma modificação na forma de avaliar o desempenho preditivo do modelo induzido, verificando sua capacidade de predizer um conjunto correto de classes para um dado exemplo. Na segunda, uma mudança no algoritmo faz com que ele passe a predizer um *ranking* de classes para cada exemplo de entrada.

O número de exemplos multirótulo e classes de uma base de dados pode influenciar o desempenho dos métodos de classificação. A próxima seção faz algumas considerações a esse respeito.

19.2 Densidade e Cardinalidade do Rótulo

Os conjuntos de dados não são todos igualmente multirótulo. Em alguns casos, o número de classes de cada exemplo é pequeno se comparado ao número total de exemplos n, enquanto noutros, esse número é grande. Esse número pode ser um parâmetro que influencia o desempenho dos diferentes métodos de classificação multirótulo (Tsoumakas e Katakis, 2007). Nesta seção são apresentados os conceitos de cardinalidade de rótulo e densidade de rótulo num conjunto de dados. Sendo $\mathbf{X}$ um conjunto de dados multirótulo consistindo em n exemplos multirótulo $(\mathbf{x}_i, \mathbf{y}_i)$, com $i = 1, 2, ..., n$, pode-se definir cardinalidade e densidade como:

- A cardinalidade de rótulo de $\mathbf{X}$ é dada pelo número médio de rótulos dos exemplos de $\mathbf{X}$:

$$CR(\mathbf{X}) = \frac{1}{n} \sum_{i=1}^{n} sum(\mathbf{y}_i) \tag{19.1}$$

- A densidade de rótulo de $|D|$ é dada pelo número médio de rótulos dos exemplos de $\mathbf{X}$ dividido por $|k|$, o número total de classes:

$$DR(\mathbf{X}) = \frac{1}{n} \sum_{i=1}^{n} \frac{sum(\mathbf{y}_i)}{k} \tag{19.2}$$

Nestas equações, $sum(\mathbf{y}_i)$ é o número de rótulos do objeto $\mathbf{x}_i$. A cardinalidade de rótulo é independente do número de possíveis classes, k, e é utilizada para quantificar o número de rótulos alternativos. A densidade de rótulo toma em conta o número de possíveis rótulos. Dois conjuntos de dados com a mesma cardinalidade de rótulo, mas com uma grande diferença no número de rótulos (diferentes densidades de rótulo), podem apresentar propriedades diferentes, que podem afetar o desempenho dos algoritmos de classificação multirótulo. Considere, por exemplo, dois conjuntos de dados com a mesma cardinalidade de dois rótulos por objeto e com diferentes densidades, dois rótulos por objeto dentre quatro possíveis classes e dois rótulos por objeto dentre 40 possíveis classes. O número de possíveis combinações no segundo caso é bem maior que no primeiro caso. Estas duas métricas estão relacionadas: $CR(\mathbf{X}) = k \times DR(\mathbf{X})$ (Tsoumakas e Katakis, 2007).

Para avaliar o desempenho dos classificadores multirótulo, algumas medidas foram propostas na literatura. A próxima seção apresenta algumas delas.

19.3 Medidas de Avaliação

A avaliação de classificadores multirótulo requer métricas diferentes das utilizadas em problemas de classificação simples-rótulo. Diferentemente da classificação simples-rótulo, em que um exemplo é classificado de maneira errada ou correta, na classificação multirótulo, um exemplo pode ser classificado de maneira parcialmente errada ou parcialmente correta. Esses casos acontecem quando um classificador atribui corretamente a um exemplo pelo menos uma das classes a que ele pertence, mas também não atribui ao exemplo uma ou mais classes às quais ele pertence. Pode igualmente verificar-se o caso do classificador atribuir a um exemplo uma ou mais classes às quais ele não pertence. Nesta seção serão apresentadas algumas das métricas propostas na literatura para avaliar classificadores multirótulo.

Basicamente, o critério de avaliação utilizado pode ser baseado na classificação multirótulo feita pelo classificador, que utiliza os rótulos atribuídos por um classificador a um dado exemplo, ou baseado numa função de *ranking*, em que, para cada exemplo, o classificador produz um *ranking* de rótulos.

Seja $\mathbf{X}$ um conjunto de dados multirótulo contendo n exemplos multirótulo $(\mathbf{x}_i, \mathbf{y}_i)$, com $i = 1, 2..., n$ e $sum(\mathbf{y}_i) < k$, em que k é o conjunto de possíveis classes. Sejam ainda $\hat{f}$ um classificador multirótulo e $\mathbf{z}_i = \hat{f}(\mathbf{x}_i)$ um vetor binário com k elementos representando o conjunto de classes preditas por $\hat{f}$ para um dado exemplo $\mathbf{x}_i$.

Para a avaliação baseada na classificação, uma medida muito comum é o *Hamming Loss*, definida na Equação 19.3. Foi utilizada, por exemplo, no trabalho de Schapire e Singer (2000).

$$HammingLoss(\hat{f}, \mathbf{X}) = \frac{1}{n} \sum_{i=1}^{n} \frac{a(\mathbf{y}_i, \mathbf{z}_i)}{k} \tag{19.3}$$

Nesta medida, o $a(\mathbf{y}_i, \mathbf{z}_i)$ representa a distância de Hamming entre dois vetores, e cor-

responde à operação *XOR* da lógica booleana (Tsoumakas e Katakis, 2007). Quanto menor for o valor do *Hamming Loss*, melhor é a classificação. A situação perfeita ocorre quando o seu valor é igual a zero.

No mesmo trabalho, os autores utilizaram outras medidas de avaliação baseadas em *ranking*. Essas medidas são denominadas na literatura de *one-error*, *coverage* e *average precision*. A medida *one-error* calcula o número de vezes que um rótulo com a melhor posição no *ranking* de rótulos não está presente no conjunto de rótulos corretos de um dado exemplo de entrada (Shen et al., 2004). Na medida *coverage*, é calculado o quão longe, em média, se deve percorrer o *ranking* de rótulos ordenados de maneira a atribuir, a um exemplo de entrada, todos os rótulos que a este deveriam ser atribuídos (de Carvalho e Freitas, 2009). O terceiro método, originalmente proposto para a tarefa de recuperação de informação, avalia a proporção média de rótulos que ocupam, no *ranking*, uma posição superior a um dado rótulo particular e que pertencem ao conjunto de rótulos desejados (de Carvalho e Freitas, 2009).

Outras métricas (taxa de erro, precisão e revocação) foram utilizadas no trabalho de Godbole e Sarawagi (2004), estão apresentadas nas Equações 19.4, 19.5 e 19.6.

$$\text{taxa de erro}(\hat{f},\mathbf{X}) = \frac{1}{n}\sum_{i=1}^{n}\frac{\mathbf{y}_i AND \mathbf{z}_i}{\mathbf{y}_i OR \mathbf{z}_i} \tag{19.4}$$

$$\text{precisão}(\hat{f},\mathbf{X}) = \frac{1}{n}\sum_{i=1}^{n}\frac{\mathbf{y}_i AND \mathbf{z}_i}{sum(\mathbf{z}_i)} \tag{19.5}$$

$$\text{revocação}(\hat{f},\mathbf{X}) = \frac{1}{n}\sum_{i=1}^{n}\frac{\mathbf{y}_i AND \mathbf{z}_i}{sum(\mathbf{y}_i)} \tag{19.6}$$

em que AND e OR representam as operações booleanas *bitwise* AND e OR aplicadas a dois vetores binários.

No trabalho de Boutell et al. (2004), é utilizada uma versão mais generalizada da taxa de erro, apresentada na Equação 19.4. Nessa versão, é utilizado um parâmetro $\alpha \geq 0$, chamado de "taxa de perdão" (*forgiveness rate*). Esta taxa reflete o grau de penalização dos erros cometidos na predição dos rótulos de classes. Quanto menor o valor de α, maior tolerância é atribuída aos erros, e quanto maior o valor de α, maior a penalização dos erros. O cálculo desta métrica é apresentado na Equação 19.7.

$$\text{taxa de erro}(\hat{f},\mathbf{X}) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{\mathbf{y}_i AND \mathbf{z}_i}{\mathbf{y}_i OR \mathbf{z}_i}\right)^{\alpha} \tag{19.7}$$

19.4 Considerações Finais

Este capítulo apresentou os conceitos de classificação multirótulo em que um exemplo pode pertencer simultaneamente a mais do que uma classe. Vários problemas reais apresentam este perfil, principalmente problemas de classificação de texto e de Bioinformática.

Após contextualizar este grupo de problemas de classificação, foram descritos os conceitos básicos e apresentadas as principais abordagens utilizadas para lidar com este tipo de problemas de classificação. Estas abordagens combinam técnicas tradicionais de classificação ou desenvolvem novas técnicas, que podem ser obtidas pela alteração de procedimentos utilizados por técnicas convencionais. Em seguida, foram discutidos os conceitos de cardinalidade e densidade e apresentadas algumas medidas de desempenho que podem ser utilizadas na avaliação de classificadores multirótulo. Alguns problemas de classificação multirótulo são também problemas de classificação hierárquica. O próximo capítulo apresenta os conceitos de classificação hierárquica.

Capítulo 20

Classificação Hierárquica

Nos problemas de classificação hierárquica, as classes podem apresentar uma relação de taxonomia ou dependência, formando subclasses e superclasses. A classificação plana, em que não se tem uma relação hierárquica entre as classes, é o tipo mais comum de classificação. Contudo, se a hierarquia das classes é conhecida ou pode ser construída, a sua consideração pode levar à indução de classificadores com melhor desempenho preditivo.

Como exemplo de aplicação, as funções exercidas por uma proteína no meio celular podem ser organizadas hierarquicamente. A indução de um classificador para a previsão da função de uma nova proteína, dadas as suas caraterísticas, configura assim um problema de classificação hierárquica (Clare e King, 2003). Outro exemplo são os problemas de categorização de textos. Classes de textos relacionados são tipicamente agrupadas em tópicos, os quais, por sua vez, também podem ser agrupados em temas principais (Sun e Lim, 2001). Assim, um texto que trata de uma partida de futebol pode ser classificado na seção de desporto que, por sua vez, pode também incluir tópicos como basquete, futebol, ténis, entre outros ligados ao tema. Desse modo, o modelo induzido deve ser capaz de classificar o texto mencionado, tanto na categoria *futebol*, como na respetiva supercategoria *desporto*. Silla e Freitas (2010) também reportam o uso de hierarquias na classificação de géneros musicais, fonemas, objetos 3D, animais e imagens.

É também importante destacar que, embora uma hierarquia de classes também tenha sido introduzida como uma possibilidade de solução de problemas multiclasse (Seção 18.1.4), este último não é considerado um problema de classificação hierárquica, uma vez que se configura apenas quando as classes possuem uma relação taxonómica inerente.

Ao induzir-se modelos de classificação para problemas hierárquicos, a relação hierárquica entre as classes deve ser tomada em consideração. Este capítulo apresenta os principais conceitos relacionados com problemas de classificação hierárquica (Seção 20.1), as possíveis abordagens para a sua solução (Seção 20.2) e algumas técnicas específicas para avaliação de classificadores hierárquicos (Seção 20.3), tendo como base os trabalhos de Freitas e de Carvalho (2007) e Silla e Freitas (2010).

20.1 Tipos de Problemas

É possível distinguir diferentes tipos de problemas de classificação hierárquica, que variam de acordo com três caraterísticas (Silla e Freitas, 2010):

1. Tipo de hierarquia em que as classes se organizam, que pode ser em forma de árvore ou de um DAG.
2. Se os dados podem ou não seguir mais do que um caminho na hierarquia. Caso possam, tem-se um problema de classificação hierárquica com múltiplos rótulos.
3. A profundidade das rotulações dos dados. Tem-se dois casos: todos os objetos possuem rotulação até aos nós folha, que representam os níveis mais profundos da hierarquia; ou, pelo menos um dos objetos possui uma rotulação parcial, que não atinge um nó folha.

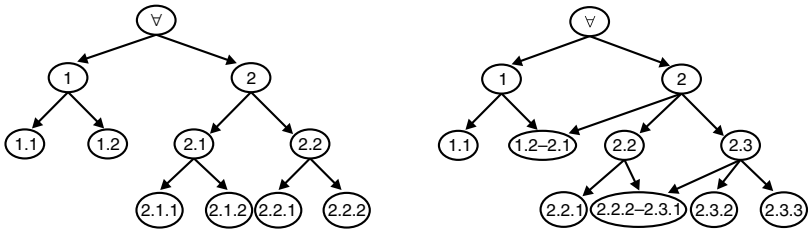
No que se refere ao tipo de hierarquia em problemas hierárquicos, as classes podem organizar-se de duas formas: em árvore ou em DAG, conforme ilustrado nas Figuras 20.1(a) e 20.1(b), respetivamente. Em ambas as estruturas, cada nó corresponde a uma classe. Iniciando pelo nó raiz, as classes são recursivamente particionadas em subclasses.

Um nó que aponta para outro nó é designado de *pai* do segundo, que é então *filho* do primeiro. Como exemplo, considerando a hierarquia da Figura 20.1(a), o nó 2 é pai dos nós 2.2 e 2.3, que são seus filhos. Os nós que têm o mesmo pai são designados de *irmãos*. Logo, 2.2 e 2.3 são irmãos na Figura 20.1(a). Quando um nó possui outros nós a si conectados num nível mais baixo, é designado de *nó interno* ou *não-folha*. Quando um nó não possui conexões a um nível inferior da hierarquia é designado por *nó-folha*. Na figura mencionada, todos os nós do terceiro nível da hierarquia são folhas.

O conteúdo dos nós indica a classe que lhes está associada. Para isso, é utilizada uma lista numérica para os diferentes níveis, em que, a cada passagem de nível, as superclasses correspondentes são adicionadas à notação. Logo, para um dado nó, o número na listagem mais à direita indica a sua classe no nível em que este se encontra, e o(s) número(s) à esquerda indica(m) os seu(s) pai(s) nos níveis mais elevados. Por exemplo, é possível verificar no DAG da Figura 20.1(b), que o nó $2.2.2 - 2.3.1$, que se encontra segundo nível, tem dois nós do primeiro nível como pais, nomeadamente 2.2 e 2.3. O nó raiz indica qualquer classificação, uma vez que um item nesse nível pode pertencer a qualquer uma das classes.

A distinção entre o DAG e a árvore consiste no fato de que, na primeira estrutura, um nó pode possuir mais do que um pai (como o nó $1.2 - 2.1$). Na realidade, as árvores são um caso particular de DAG em que todos os nós têm um único pai. Os ancestrais de um determinado nó na hierarquia são os nós que se encontram no(s) caminho(s) da raiz a esse nó, incluindo ele próprio. Por exemplo, os ancestrais do nó 2.1.1, na Figura 20.1(a), são os nós 2.1.1, 2.1, 2 e a raiz.

Embora nos problemas de classificação hierárquica os dados apresentem naturalmente múltiplos rótulos, uma vez que um objeto pertencente a uma subclasse também pertence



(a) Árvore (b) DAG

Figura 20.1 *Exemplos de hierarquias.*

às respetivas superclasses, é comum designar-se como *problemas de classificação hierárquica multirótulo* os problemas em que as previsões podem seguir múltiplos caminhos na hierarquia. Por exemplo, um objeto pode pertencer simultaneamente às classes 2.1 e 2.2 (ou 1.1 e 2.1) da hierarquia apresentada na Figura 20.1(a). Assim, se o único caminho percorrido for: 2, 2.1 e 2.1.1, o objeto é atribuído a uma dessas classes, de preferência à mais específica que, neste caso é 2.1.1.

Na maioria dos problemas de classificação hierárquica, é comum que todos os exemplos possuam rótulos que correspondem a nós folha. Este caso corresponde a uma rotulação completa em profundidade. Ou seja, todo o objeto possui rótulos de classes em todos os níveis de um dado caminho da hierarquia, que vai desde a raiz até a uma dada folha. Porém, pode haver exemplos para os quais a classificação pára num nó interno da hierarquia, tipicamente porque a classificação em níveis mais profundos ainda é desconhecida para esses exemplos. Tem-se, então, uma rotulação parcial em profundidade.

As caraterísticas que distinguem os diferentes tipos de problemas de classificação hierárquica, são seguidamente resumidas.

- *Tipo de hierarquia*:
  - Árvore;
  - DAG.
- *Caminho das rotulações*:
  - Múltiplo (multirrótulo);
  - Único;
- *Profundidade das rotulações*:
  - Completa (nós folha);
  - Parcial.

Cada combinação possível de valores de caraterísticas conduz a um problema distinto. Por exemplo, um determinado problema pode ser caraterizado por possuir uma hierarquia

estruturada em árvore, em que os dados possuem rótulos que podem seguir múltiplos caminhos na hierarquia (multirótulo), e existem dados para os quais os rótulos correspondem a nós internos (não folha). A partir da determinação dessas caraterísticas, é então possível escolher uma determinada estratégia algorítmica, apropriada à solução do problema.

20.2 Algoritmos para Classificação Hierárquica

Na seção anterior, discorreu-se sobre os tipos de problemas de classificação hierárquica, que podem ser distinguidos de acordo com caraterísticas associadas à hierarquia que seguem, e às rotulações dos objetos nos respetivos conjuntos de dados. No que concerne aos algoritmos existentes na literatura para a resolução de problemas de classificação hierárquica, também é possível fazer uma distinção, de acordo com quatro caraterísticas (Silla e Freitas, 2010):

1. O tipo de hierarquia com que o algoritmo pode lidar: árvore ou DAG.
2. Se o algoritmo é capaz de rotular os dados de acordo com múltiplos caminhos na hierarquia (multirótulo), ou não.
3. A profundidade das predições do algoritmo, i.e. se estas ocorrem sempre em nós folha, ou não.
4. Os tipos de classificadores que são usados na resolução do problema hierárquico: locais ou globais. Para classificadores locais, tem-se ainda a possibilidade de utilizar um classificador por nó, um classificador por nó pai ou um classificador por nível.

Os itens 1 a 3 estão claramente relacionados com os itens 1 a 3 que distinguem os problemas de classificação hierárquica descritos na seção anterior, evidenciando a importância da determinação das caraterísticas do problema na escolha de uma abordagem de solução apropriada.

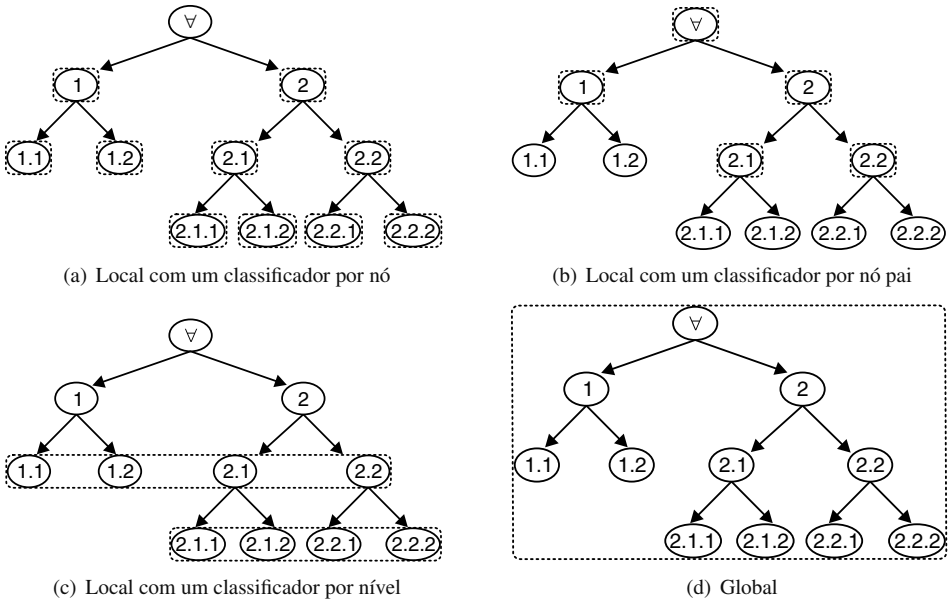
Algoritmos capazes de realizar predições para hierarquias do tipo DAG, também são capazes de lidar com hierarquias de árvores, uma vez que estas são DAGs simplificadas. Contudo, os algoritmos capazes de lidar, apenas, com hierarquias de árvores devem ser significativamente estendidos para lidar com DAGs. Seguindo este raciocínio, é possível considerar que os problemas com hierarquia em DAG são mais complexos do que aqueles com uma hierarquia em árvore.

A rotulação em múltiplos caminhos na hierarquia, implica a possibilidade de resolução de problemas de classificação hierárquica multirótulo. Existem algoritmos que realizam predições seguindo apenas um caminho na hierarquia. Contudo, também é possível realizar transformações no problema de múltiplos caminhos, transformando-o num, ou mais, problemas de caminho único, usando abordagens como as descritas no Capítulo 19, mas adaptadas para o caso hierárquico.

A profundidade da predição do algoritmo está relacionada com a sua capacidade em realizar sempre predições nos nós folha (predição obrigatória em nós folha) ou, alternativamente, com a sua capacidade em realizar predições em qualquer nível da hierarquia,

incluindo as folhas (predição não obrigatória em nós folha). Embora a confiabilidade da predição possa ser maximizada na predição não obrigatória em nós folha, a classificação torna-se menos específica e, por esse motivo, pode revelar-se de menor utilidade.

Na Figura 20.2 são ilustradas as diferentes abordagens utilizadas na resolução de problemas de classificação hierárquica de acordo com o tipo de classificadores usados. Os nós destacados com bordas pontilhadas adicionais representam os classificadores usados em cada caso. Por uma questão de simplicidade, nestes exemplos considera-se uma hierarquia estruturada em árvore. Existe ainda uma quinta abordagem bastante simplificada para resolver problemas de classificação hierárquica, que consiste em usar um único classificador plano,[1] geralmente distinguindo somente as classes em nós folha. Por exemplo, a predição numa classe 1.2 também indica que esse exemplo em particular pertence à classe 1. Esta abordagem corresponde, assim, à simples transformação do problema hierárquico num único problema de classificação plana, que pode ser usado, tanto no caso de árvores, como no caso de DAGs. Esta abordagem possui a desvantagem de poder ter que distinguir um grande número de classes (todas as folhas) sem explorar a informação dos relacionamentos hierárquicos presentes. Além disso, neste caso as predições são necessariamente obrigatórias em nós folha.



(a) Local com um classificador por nó

(b) Local com um classificador por nó pai

(c) Local com um classificador por nível

(d) Global

Figura 20.2 *Abordagens para a classificação hierárquica.*

[1] Classificadores planos são classificadores induzidos para a resolução de problemas de classificação planos, em que não existe uma relação hierárquica entre as classes.

Seguidamente são descritas as abordagens mais comuns para a classificação hierárquica, que se encontram ilustradas na Figura 20.2.

20.2.1 Classificadores Locais

Inicialmente, introduzem-se as abordagens que exploram informação local da hierarquia no processo de solução (casos a, b e c na Figura 20.2). Existem três maneiras padrão de usar a informação local, empregando um classificador plano por: (a) nó da hierarquia, exceto a raiz; (b) nó pai da hierarquia; (c) nível da hierarquia, com exceção da raiz. O recurso a informação local permite a aplicação direta de classificadores planos na resolução do problema hierárquico.

A utilização de um classificador plano por nó, que representa uma classe na hierarquia, é a estratégia mais usada. Esta estratégia requer a combinação de classificadores binários, um por nó da hierarquia, com exceção da raiz (Figura 20.2(a)). Diferentes estratégias podem ser empregues para determinar quais os exemplos que devem ser considerados positivos e negativos pelo algoritmo de classificação utilizado em cada nó (Eisner et al., 2005; Fagni e Sebastiani, 2007). Uma das políticas mais usadas nesta definição é a de *irmãos*, em que, para um dado nó representando a classe c_j, os exemplos positivos são aqueles com rótulo c_j e de suas subclasses, enquanto que os exemplos negativos são os de nós irmãos de c_j, segundo a hierarquia, e de suas respetivas subclasses. Por exemplo, considerando o classificador para a classe 2.1, na Figura 20.2(a), os objetos positivos na política de irmãos são os de rótulo 2.1, 2.1.1 e 2.1.2, enquanto os negativos são os de rótulo 2.2, 2.2.1 e 2.2.2. Outras políticas podem ser consultadas em Silla e Freitas (2010).

O uso de um classificador plano por nó pai é ilustrado na Figura 20.2(b). Para cada um desses nós, devem-se distinguir as subclasses contidas nos respetivos nós filhos, usando classificadores multiclasse, ou combinações de classificadores binários, como as apresentadas no Capítulo 18. Esta estratégia não costuma ser utilizada em hierarquias do tipo DAG, uma vez que pode existir muita redundância nos conjuntos de treino de diferentes classificadores.

Na Figura 20.2(c) é ilustrada a bordagem baseada na utilização de um classificador plano multiclasse, por nível da hierarquia (i.e., três classificadores no exemplo considerado). Esta é a abordagem menos utilizada na literatura da área, pois podem surgir conflitos nas predições realizadas para os diferentes níveis.

Na etapa de teste, uma estratégia adotada para prever a classe de um novo exemplo, utilizando classificadores locais, é denominada *top-down* (de cima para baixo). Outras estratégias são analisadas em (Silla e Freitas, 2010). Nesta estratégia, partindo do(s) classificador(es) em níveis iniciais da hierarquia, as predições obtidas são usadas para escolher um novo classificador até que se atinja um nó folha ou, alternativamente, até se decidir parar a predição num nível intermédio da hierarquia. Uma desvantagem associada a esta estratégia é que, quando um exemplo é erroneamente classificado num determinado nível, existe o risco desse erro ser propagado nos níveis seguintes da hierarquia. Este fenómeno de propagação ocorre sempre para estruturas em árvore, mas pode não acontecer em hierarquias DAG.

Se o problema não exige predições obrigatórias em nós folha, é possível empregar algum mecanismo que permita a paragem da classificação nalgum nível. Por exemplo, o classificador no próximo nível é consultado somente se a confiança na predição atual é maior do que um determinado limiar. Problemas de classificação multirótulo também podem ser tratados através da consideração de múltiplos caminhos na hierarquia, de acordo com os níveis de confiança das predições obtidas.

20.2.2 Classificadores Globais

Os classificadores globais consideram a hierarquia como um todo na etapa de treino e não possuem a modularidade caraterística das abordagens locais, como ilustrado na Figura 20.2(d). O modelo final obtido nesta abordagem, também denominada de *big-bang*, pode ser menor do que o modelo formado por múltiplos classificadores locais. Não obstante, este classificador tende a ser mais complexo do que os classificadores individuais induzidos na abordagem local. Daqui se deduz que, na abordagem *big-bang*, geralmente é induzido um único modelo. Na etapa de teste, o modelo induzido pode ser potencialmente utilizado na predição de qualquer classe da hierarquia.

Por exemplo, em Kiritchenko et al. (2006), o problema hierárquico é transformado num plano multirótulo. Para esse efeito, a cada exemplo de treino são adicionados os rótulos dos nós antecessores.

Existem, ainda, diferentes algoritmos de ECD que foram adaptados para considerarem a hierarquia, como um todo, durante a etapa de treino, gerando um único classificador capaz de realizar predições hierárquicas. Em Clare e King (2003) é apresentada uma adaptação do algoritmo C4.5 para a indução de árvores de decisão. Neste trabalho, o algoritmo C4.5 (Quinlan, 1993) foi modificado para a tarefa de classificação hierárquica. A ideia básica desta modificação consistiu na substituição da fórmula da entropia, comummente utilizada para decidir qual o atributo que será selecionado para um dado nó da árvore, por uma entropia ponderada. Esta entropia ponderada toma em consideração as classes, em níveis mais elevados da hierarquia, que tendem a ter menores valores de entropia do que as classes em níveis mais profundos. Porém, as classes que se encontram nos níveis mais profundos são normalmente preferíveis, uma vez que fornecem conhecimento mais específico. Silla e Freitas (2009) apresentam uma adaptação do algoritmo *naive* Bayes.

Também existem algoritmos especialmente desenvolvidos para a classificação hierárquica, como é o caso do Clus-HMC (Blockeel et al., 2002), que combina árvores tendo por base a noção de *predictive clustering trees*.

20.3 Avaliação de Classificadores Hierárquicos

Dada a particularidade das predições terem que seguir uma hierarquia predefinida, existe a necessidade de adaptar as medidas de avaliação de classificadores hierárquicos. Por exemplo, estas medidas devem permitir a atribuição de maior utilidade às predições obtidas em níveis mais profundos da hierarquia, dando pesos diferentes a classificações obtidas em

níveis distintos da hierarquia. Outra possibilidade é permitir classificações parcialmente corretas, em que as predições são corretas para apenas alguns níveis da hierarquia. Ainda não existe consenso em relação às medidas que devem ser adotadas para avaliar os resultados desta tarefa, e muitos trabalhos limitam-se a usar as medidas de desempenho para classificadores planos, o que não é a melhor opção.

As medidas de avaliação hierárquicas podem ser genericamente divididas como baseadas (Costa et al., 2007):

- Em distância;
- Em profundidade;
- Em semelhança (semântica) entre as classes e a hierarquia;
- Nas relações de descendência e/ou ancestralidade das classes na hierarquia.

Num dos primeiros trabalhos que propõe a adaptação de medidas de desempenho para o cenário da classificação hierárquica, Sun e Lim (2001) modificaram as medidas de precisão, sensibilidade, e taxa de erro de maneira a considerar a distância entre a classe predita e a classe real, na hierarquia. Esta distância é calculada como o número de arestas que separam esses nós na hierarquia. Por exemplo, na Figura 20.1(a) a distância entre 1.2 e 2.2 é 4, e a distância entre 2.1 e 2.2 é 2. Maiores distâncias implicam maior erro. De fato, se um novo objeto pertence à classe 2.2, é pior classificá-lo na classe 1.2 do que na classe 2.1, uma vez que, no último caso, pelo menos as classes 2.1 e 2.2 pertencem à mesma superclasse na hierarquia (i.e. a classe 2).

Contudo, esta distância não toma em consideração o fato das classificações em níveis mais profundos serem mais difíceis de realizar e, geralmente, serem mais úteis. Isto pode ser corrigido por via da associação de pesos a cada aresta, de acordo com a sua posição na hierarquia, diminuindo o peso com a profundidade (Blockeel et al., 2002). Contudo, a definição dos valores dos pesos a serem atribuídos não é trivial, sendo dependente da hierarquia. Para DAGs, devido aos múltiplos caminhos existentes para uma mesma predição, o uso desta métrica pode ser considerado pouco viável.

Sun e Lim (2001) também propuseram considerar a semelhança entre as classes na medição dos erros de classificação, numa abordagem semântica. Cada classe é inicialmente descrita por um vetor de caraterísticas, e a semelhança entre as classes predita e real é calculada a partir destes vetores. Essa semelhança é, então, utilizada para definir novas medidas de precisão, revocação, e taxa de erro. Estas medidas foram aplicadas ao domínio de categorização hierárquica de textos, podendo não se revelar apropriadas quando aplicadas a outros domínios.

As medidas que foram consideradas como mais interessantes, no contexto de classificação hierárquica, por Silla e Freitas (2010), são as que se baseiam nas relações de ancestralidade na hierarquia, tais como as propostas por Kiritchenko et al. (2004). Nestas medidas, é considerado o número de ancestrais comuns existentes entre a(s) classe(s) verdadeira(s) do exemplo e aquela(s) predita(s) pelo classificador. No cálculo da precisão, este valor é dividido pelo número de ancestrais da(s) classe(s) predita(s), como ilustrado na Equação 20.1 para um objeto em particular, em que c_p representa a(s) classe(s) predita(s),

c_v corresponde à(s) classe(s) verdadeira(s) do objeto, $Ancestral(z)$ fornece o conjunto de antecessores de um determinado nó z na hierarquia e $|Z|$ é o operador que fornece a cardinalidade do conjunto Z. É importante sublinhar que o conjunto $Ancestral(Z)$ inclui a classe Z. A raiz da hierarquia não é considerada um ancestral.

$$P_H = \frac{|Ancestral(c_p) \cap Ancestral(c_v)|}{|Ancestral(c_p)|} \tag{20.1}$$

Na medida de revocação, o número de ancestrais em comum é dividido pelo número de ancestrais da(s) classe(s) verdadeira(s) do objeto, conforme apresentado na Equação 20.2.

$$R_H = \frac{|Ancestral(c_p) \cap Ancestral(c_v)|}{|Ancestral(c_v)|} \tag{20.2}$$

Para obter a precisão e a revocação, num conjunto contendo n dados, basta somar, para cada objeto individual, as medidas calculadas anteriormente.

Estas medidas são consideradas mais genéricas e podem ser aplicadas em diferentes tipos de problemas de classificação hierárquica, a saber: árvore ou DAG, multirótulo ou não, com profundidade de rotulação completa. Apenas podem ocorrer deficiências no caso em que a profundidade de rotulação é parcial (predição não obrigatória em nós folha), pois o desconhecimento da classe real mais específica é penalizado. Este desconhecimento pode, porém, ser o resultado do fato do domínio ainda ser desconhecido, e as predições obtidas em níveis mais profundos podem estar trazendo informações valiosas.

20.4 Considerações Finais

Neste capítulo foram apresentados os conceitos fundamentais de classificação hierárquica, comum em domínios como a categorização de textos e a Bioinformática. Neste tipo de problema, as classes seguem uma relação hierárquica, que deve ser tomada em consideração na etapa indutiva e preditiva. Foram apresentados os vários tipos de problemas hierárquicos e as abordagens algorítmicas propostas para o tratamento destes problemas. Estes problemas foram distinguidos de acordo com a hierarquia tratada, os caminhos e a profundidade das classificações e, no caso dos algoritmos, consoante a abordagem adotada na resolução do problema.

Por fim, foi discutida a questão da necessidade de considerar as caraterísticas dos problemas hierárquicos na avaliação do classificador obtido neste domínio. Embora muitos trabalhos se limitem a utilizar medidas de desempenho para classificadores planos, é fundamental considerar as particularidades introduzidas por uma hierarquia de classes no desenvolvimento de novas medidas.

Capítulo 21

Computação Natural

A Computação natural é uma área de investigação associada à ECD, à Estatística e à Otimização, que tem registado um forte crescimento nos últimos anos. A Computação Natural inspira-se em processos que ocorrem na natureza para o desenvolvimento de novos algoritmos que possam ser utilizados em problemas reais. Esta inspiração tem origem nos mais diversos processos, o que motivou o desenvolvimento de várias técnicas como, por exemplo, RNAs, computação evolutiva, computação quântica, computação molecular, autómatos celulares e geometria fractal.

Um dos principais argumentos a favor da computação natural é que, quando fenómenos naturais complexos são analisados e modelados computacionalmente, é possível compreender melhor a natureza e utilizar esse conhecimento para criar novos algoritmos computacionais. A pesquisa realizada nesta área também tem investigado a utilização de materiais naturais, como moléculas de DNA, na execução de tarefas de computação. Estes novos materiais podem complementar os materiais atualmente utilizados na construção de computadores digitais (Castro, 2007). Uma subárea da computação natural, conhecida como computação bio-inspirada, tal como o próprio nome sugere, inspira-se sobretudo em processos biológicos. Este capítulo descreve técnicas de computação bio-inspirada utilizadas em ECD. Estas técnicas constituem uma classe de meta-heurísticas (Blum e Roli, 2003; Dorigo et al., 2006). Uma meta-heurística é um processo iterativo de geração que guia uma heurística, subordinada à combinação inteligente de diferentes conceitos, na exploração do espaço de busca. Neste processo, são utilizadas estratégias para encontrar, de forma eficiente, soluções próximas do ótimo (Osman e Laporte, 1996). De acordo com Maniezzo et al. (2004), as meta-heurísticas utilizam algumas heurísticas básicas para contornar o problema da obtenção de mínimos locais: iniciam a procura a partir de uma solução inicial (ou conjunto de soluções iniciais) e adicionam elementos até obterem uma boa solução ou, alternativamente, iniciam a procura com uma solução completa e, iterativamente, procedem a modificações de alguns dos seus elementos, até que seja atingido um critério de paragem.

As seções seguintes apresentam algumas das principais técnicas de computação bio-inspirada propostas na literatura. Na Seção 21.1 são introduzidos os conceitos básicos de inteligência de enxames, e são apresentadas duas das suas variações: otimização por

colónia de formigas e otimização por enxame de partículas. A computação evolutiva, uma das técnicas mais populares da computação bio-inspirada, é explicada na Seção 21.2. Na última seção, Seção 21.3, são apresentadas as considerações finais para este capítulo.

21.1 Inteligência de Enxames

Na natureza, a inteligência de enxames, ou inteligência coletiva, diz respeito a agentes (indivíduos) que apresentam um nível superior de inteligência dentro do comportamento social. Os indivíduos devem ser capazes de interagir entre si e com o ambiente. A incorporação de vida social numa meta-heurística permite considerar, no processo de procura, aspetos como: maior facilidade em encontrar alimento, melhor divisão de trabalho, melhor aproveitamento das capacidades de cada indivíduo e como evitar predadores e facilitar a operação de caça (Castro, 2006).

Os algoritmos baseados em inteligência de enxames manipulam indivíduos simples, que atuam de forma auto-organizada, isto é, sem qualquer tipo de controlo central sobre os membros do enxame. Segundo Millonas (1994), os sistemas baseados em inteligência coletiva seguem cinco princípios:

1. **Proximidade:** os indivíduos de uma população devem interagir entre si.
2. **Qualidade:** os indivíduos devem ser capazes de avaliar a interação entre si e com o ambiente.
3. **Diversidade:** a capacidade de um sistema reagir a ações inesperadas.
4. **Estabilidade:** os indivíduos não podem modificar o seu comportamento em resposta a qualquer modificação no ambiente.
5. **Adaptabilidade:** os indivíduos devem ser capazes de se adaptar às mudanças do ambiente e da população.

As técnicas *otimização por colónia de formigas* e *otimização por enxame de partículas* são variações da inteligência de enxames. A primeira é baseada no comportamento de formigas na busca por alimentos, e a segunda é inspirada na organização existente entre bandos de animais, como pássaros e peixes, e no comportamento social do ser humano.

21.1.1 Otimização por Colónia de Formigas

Colónias de formigas inspiraram o desenvolvimento de várias meta-heurísticas. Entre essas, a mais estudada e que tem alcançado maior sucesso é uma técnica de otimização de propósito geral, conhecida como ACO (do inglês, *Ant Colony Optimization*) (Dorigo et al., 2006). ACO é inspirada no comportamento das formigas no processo de procura de alimentos. O principal aspeto deste comportamento é a comunicação que se estabelece entre formigas de uma colónia, por via do depósito de feromona nas trilhas percorridas.

Inicialmente, durante a busca por alimentos, as formigas exploram de maneira aleatória uma dada região. Durante a movimentação, essas formigas depositam feromonas

no solo, ao longo do caminho percorrido (Blum, 2005). A feromona é uma substância química cujo odor é sentido pelas formigas. Ao escolher um caminho entre vários caminhos possíveis, estudos sugerem que as formigas escolhem o caminho marcado com uma maior concentração de feromona. Como a probabilidade das formigas que alcançaram o alimento pelo menor caminho retornarem antes das que escolheram o caminho mais longo é maior, o caminho mais curto ficará com uma maior concentração de feromonas e, provavelmente, será o caminho seguido pelas próximas formigas. A Figura 21.1 ilustra esta procura do caminho mais curto, mostrando o aumento do número de formigas (círculos cinza) que utilizam o caminho mais curto entre o ninho (N) e a fonte de alimento (A) ao longo de três instantes de tempo. Com o passar do tempo, o menor caminho possuirá a maior quantidade de feromonas depositadas, atraindo futuramente um maior número de formigas.

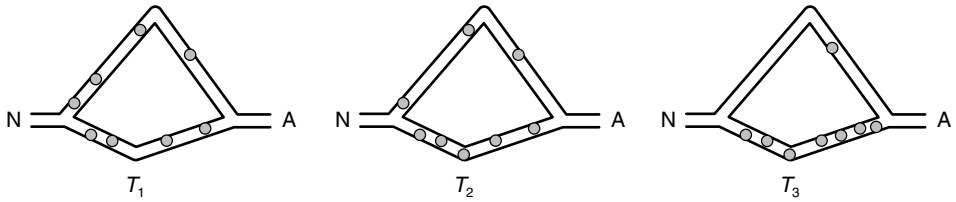


Figura 21.1 *Formigas em busca de alimento.*

ACO explora um mecanismo semelhante para resolver problemas de otimização, e foi formalizado como meta-heurística para problemas de otimização combinatória por Dorigo e Di-Caro (1999). Um problema de otimização por maximização gera como solução o indivíduo que maximiza o valor de uma dada função objetivo. Uma função objetivo simples seria encontrar o caminho mais curto entre dois pontos. Neste contexto, ACO pode ser facilmente utilizada no problema do caixeiro-viajante [1]. Porém, a função objetivo pode representar qualquer problema de otimização.

O funcionamento da ACO pode ser resumido como um conjunto de agentes computacionais concorrentes e assíncronos (como, por exemplo, uma colónia de formigas) que se movimentam através de estados do problema, que correspondem às soluções parciais no espaço de procura. A movimentação dos agentes é baseada em dois parâmetros: trilha e atratividade. No decurso do seu movimento, cada formiga constrói incrementalmente uma solução para o problema. Durante a fase de construção, ou finalização de uma solução, a formiga avalia a solução encontrada e modifica o valor de feromona associado às trilhas percorridas. O total de feromona nas trilhas será utilizado pelas restantes formigas na procura pela melhor solução (Maniezzo et al., 2004). Um algoritmo ACO inclui dois mecanismos adicionais: evaporação da trilha e, opcionalmente, ações. A *evaporação da trilha* diminui a quantidade de feromona em todas as trilhas à medida que o tempo passa,

[1] No problema do caixeiro-viajante, dado um grupo de cidades e a distância entre elas, procura-se o caminho que visite todas as cidades uma única vez, e cuja distância seja mínima.

com o intuito de evitar o acumular ilimitado de feromona. Por outro lado, as *ações* podem ser utilizadas para permitir ações centralizadas, o que não acontece com as colónias naturais de formigas (Maniezzo et al., 2004).

Em ACO para problemas combinatórios, as formigas constroem as soluções de modo incremental. Inicialmente, cada formiga começa com um conjunto vazio de soluções. A cada passo de construção, uma componente da solução é adicionada ao conjunto de soluções. A definição de componente da solução é dependente da aplicação. Por exemplo, no problema do caixeiro-viajante, uma componente da solução é uma cidade que é adicionada ao percurso. Para escolher qual a componente da solução que deve ser adicionada ao conjunto, é realizada uma escolha probabilística, que normalmente considera o total de feromona associada a uma determinada componente e uma possível informação heurística sobre o problema (Socha, 2004).

A ACO foi inicialmente desenvolvida para resolver problemas de otimização combinatória, pelo que a versão original não cobre problemas de otimização cujas variáveis sejam contínuas. Uma extensão para a ACO, desenvolvida por Socha (2004), permite a otimização de problemas com variáveis contínuas e mistas (discretas e contínuas). Posteriormente, Socha e Dorigo (2008) desenvolveram o $\text{ACO}_{\mathbb{R}}$ para problemas de domínios contínuos. A ideia principal da $\text{ACO}_{\mathbb{R}}$ é substituir a distribuição de probabilidade discreta, utilizada no ACO convencional, por uma distribuição contínua, i.e. por uma função densidade de probabilidade. No Algoritmo 21.1, é apresentado um pseudo código com os principais passos do algoritmo ACO.

Algoritmo 21.1 Algoritmo ACO

Entrada: Uma função objetivo capaz de avaliar cada solução e um grafo com os possíveis caminhos
Saída: Um conjunto de soluções otimizadas

```
1  Gerar aleatoriamente uma população inicial de formigas
2  Inicializar valores de feromona nas arestas
3  repita
4      para cada formiga faça
5          Mover a formiga por um caminho completo até a construção de uma
           solução para o problema
6          Avaliar a sua aptidão para resolver o problema
7          Aumentar a quantidade de feromonas nas arestas percorridas
8          Evaporar parte das feromonas de todas as arestas
9      fim
10 até Critério de paragem for satisfeito;
```

21.1.2 Otimização por Enxame de Partículas

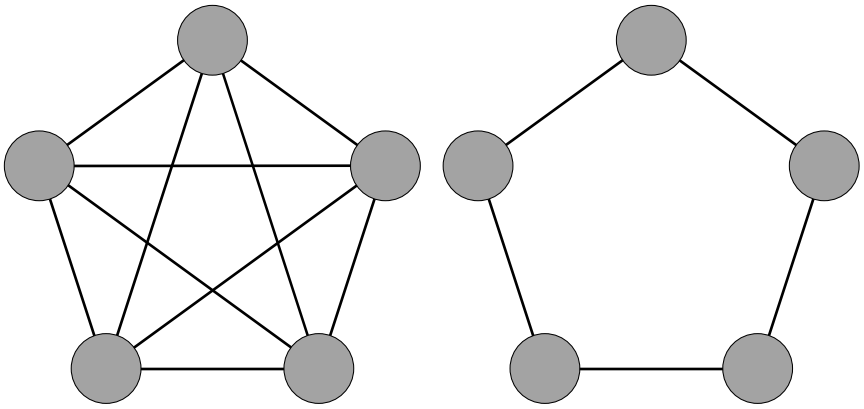
PSO (do inglês, *particle swarm optimization*) é uma técnica de otimização global desenvolvida por Kennedy e Eberhart (1995) e inicialmente introduzida para otimização de funções contínuas não lineares. Esta técnica é fortemente baseada no conceito de que, a partilha de informações entre indivíduos confere uma vantagem evolutiva. O seu desenvolvimento foi inspirado no comportamento social de pássaros e peixes, e no comportamento social dos seres humanos.

Na técnica PSO, as soluções são denominadas de partículas. As partículas movimentam-se por um espaço de procura, e são capazes de armazenar informações passadas e compartilhar informações com outras partículas. Estes dois tipos de informações correspondem à *aprendizagem individual* (cognitiva) e à *transmissão cultural* (social). Adotando este procedimento, as partículas utilizam as melhores soluções no seu processo de "evolução". Kennedy e Eberhart (2001) recorreram a três princípios para explicar, de forma sucinta, o processo de adaptação cultural:

1. **Avaliar:** cada partícula deve avaliar a solução que encontrou no espaço de procura.
2. **Comparar:** cada partícula deve comparar a solução que obteve com as soluções obtidas pelas demais partículas.
3. **Imitar:** as partículas devem imitar o funcionamento da partícula que mais se aproximou da solução desejada.

O compartilhamento de informações é realizado por partículas topologicamente vizinhas. A vizinhança considerada pelo PSO não é definida pelos valores dos atributos de cada partícula. A vizinhança pode ser global (ou estrela), quando cada partícula é vizinha de todas as outras, e local (ou anel), quando os vizinhos de cada partícula são os seus k vizinhos mais próximos (Castro, 2006). Na vizinhança global, cada partícula compartilha informação com todas as outras. As Figuras 21.2(a) e 21.2(b) apresentam exemplos de topologias com cinco partículas para vizinhança global e local, respetivamente.

O algoritmo PSO pode ser aplicado, quer a problemas binários, quer a problemas contínuos. Cada partícula é representada pela sua posição atual, pela sua velocidade e pela melhor posição que encontrou. Cada partícula é tratada como um ponto num espaço d-dimensional. A posição da partícula l é dada por $P_l = (p_{l1}, p_{l2}, \ldots, p_{ld})$; a sua velocidade é dada por $V_l = (v_{l1}, v_{l2}, \ldots, v_{ld})$, e a melhor posição encontrada por essa partícula por $M_l = (m_{l1}, m_{l2}, \ldots, m_{ld})$ (Castro, 2006). Para o caso da vizinhança global, a melhor posição encontrada, entre todas as partículas, é representada pelo símbolo M_g. Uma partícula irá mover-se numa determinada direção em função da sua posição atual, da sua velocidade, da melhor posição encontrada por si, e da melhor posição encontrada pelos seus vizinhos. As Equações 21.1 e 21.2 determinam a forma com que a velocidade e a posição das partículas são atualizadas, respetivamente. Nestas equações, t representa a iteração, w o peso da inércia, cujo papel é balancear a procura global e a procura local (introduzido por Shi e Eberhart (1998)), r_1 e r_2 são dois valores aleatórios independentes e uniformemente distribuídos no intervalo $[0,1]$ (utilizados para gerar um comportamento estocástico), e φ_1 e φ_2



(a) Global (b) Local

Figura 21.2 *Gráfico ilustrativo de vizinhança global e local.*

são taxas de aprendizagem. Para evitar a explosão de velocidade das partículas, a velocidade máxima é limitada. A posição da partícula também pode ser restringida ao intervalo do espaço de procura utilizado. No Algoritmo 21.2 são descritos os principais passos do algoritmo PSO. No final da execução do algoritmo, a posição da partícula com melhor avaliação representa a melhor solução encontrada para resolver o problema de otimização. Cada componente do vetor que representa a posição, constitui uma parte dessa solução.

$$v_{ld}(t+1) = w \cdot v_{ld}(t) + \varphi_1 \cdot r_1 \cdot (m_{ld} - p_{ld}(t)) + \varphi_2 \cdot r_2 \cdot (m_{gd} - p_{ld}(t)) \tag{21.1}$$

$$p_{ld}(t+1) = p_{ld}(t) + v_{ld}(t) \tag{21.2}$$

21.2 Computação Evolutiva

A computação evolutiva (CE) engloba um conjunto de técnicas computacionais para a resolução de problemas baseados na Genética e na Teoria da Evolução. As técnicas de CE podem ser divididas em três grandes áreas:

- Algoritmos genéticos (AGs);
- Estratégias evolutivas (EEs);
- Programação evolutiva (PE).

Como os aspetos inerentes a cada uma destas áreas são continuamente assimilados pelas outras áreas, é difícil definir as fronteiras entre estas técnicas. Por exemplo, tanto

Algoritmo 21.2 Algoritmo PSO

Entrada: Uma função objetivo capaz de avaliar cada solução
Saída: Um conjunto de soluções otimizadas
Inicializar as partículas
repita
    para cada *partícula* **faça**
        Avaliar a sua aptidão para resolver o problema
    fim
    Escolher partícula com maior aptidão
    para cada *partícula* **faça**
        Atualizar a sua velocidade
        Atualizar a sua posição
    fim
até *Critério de paragem for satisfeito*;

as EEs como a PE foram originalmente propostas para a otimização de funções contínuas. Por outro lado, os AGs são, em geral, utilizados em problemas de otimização combinatória. Para exemplificar o uso destas técnicas, nesta seção serão detalhados os principais aspetos dos AGs, que concentram o maior número de aplicações utilizando CE.

Segundo a Teoria da Evolução, os organismos que melhor se adaptam ao seu ambiente apresentam maiores hipóteses de terem as suas caraterísticas reproduzidas numa nova geração. Por outro lado, a área da Genética explica como os filhos herdam caraterísticas dos pais. A incorporação das ideias e conceitos oriundos da teoria da evolução e da genética, num algoritmo computacional projetado para solucionar problemas reais ocorre, de forma simplificada, da seguinte maneira: é gerada uma população inicial de possíveis soluções para o problema. Através de um processo iterativo, procura-se gerar uma boa solução por via da evolução das melhores soluções da população atual, de acordo com uma função de aptidão, que mede a qualidade de cada solução. Para a definição de cada nova população a partir das soluções escolhidas, três operadores genéticos são geralmente aplicados: elitismo (cópias simples das melhores soluções), cruzamento (combinação de partes de pares de soluções) e mutação (alteração da composição de algumas soluções, permitindo a criação de soluções que ainda não foram observadas).

Nos AGs, uma solução é também denominada de *indivíduo* ou *cromossoma*. Um aspeto importante relativo aos AGs, é a forma de representação de uma solução para o problema em análise. Frequentemente, os indivíduos são representados por vetores binários. Nesses vetores, cada elemento, denominado *gene*, representa a presença (valor 1) ou ausência (valor 0) de alguma caraterística (Carvalho et al., 2003). Adotando esta representação, os indivíduos podem ser representados por sequências de valores binários. Valores

Algoritmo 21.3 Algoritmo genético básico

Entrada: Uma função objetivo capaz de avaliar cada solução
Saída: Um conjunto de soluções otimizadas
1 Gerar população inicial de indivíduos
2 **repita**
3 **para cada** *indivíduo da população* **faça**
4 Avaliar a sua aptidão para resolver o problema
5 **fim**
6 Selecionar indivíduos que irão integrar a próxima geração
7 Aplicar operadores genéticos aos indivíduos selecionados
8 **até** *Critério de paragem for satisfeito*;

inteiros e valores reais também têm sido utilizados. O tipo dos valores depende da natureza do problema tratado.

Quando são utilizados AGs, o processo de procura por uma boa solução ocorre em várias gerações. Em cada geração, os mecanismos de seleção tendem a selecionar os indivíduos mais aptos, enquanto os operadores de reprodução geram uma nova população de indivíduos. Para a avaliação de cada indivíduo de uma população é utilizada uma função de aptidão, que mede a qualidade da solução, representada por um indivíduo para o problema tratado. Embora diferentes implementações de AGs variem em alguns aspetos, o seu funcionamento básico pode ser sumarizado pelo Algoritmo 21.3.

No Algoritmo 21.3, a população inicial geralmente ocorre pela atribuição de valores aleatórios aos genes dos indivíduos, que podem, dessa forma, ser representados por sequências de valores binários. No processo de seleção, são selecionados, de forma probabilística, os indivíduos mais aptos que integram a população atual. Existem vários métodos para a seleção dos indivíduos. Os mais conhecidos são a *seleção por roleta* e a *seleção por torneio*. O recurso a probabilidades faz com que o indivíduo com maior valor de aptidão tenha maiores hipóteses de participar na fase de reprodução. Como resultado, são gerados indivíduos cada vez mais aptos, enquanto os indivíduos menos aptos tendem a desaparecer. Na *seleção por roleta*, cada indivíduo da população é representado em roleta por uma fatia proporcional ao seu índice de aptidão. Para selecionar *ni* indivíduos, a roleta é girada *ni* vezes, selecionando, em cada rodada, o indivíduo cuja fatia é apontada pela agulha da roleta. Por sua vez, na *seleção por torneio*, a seleção de *ns* indivíduos na fase de reprodução é efetuada através da realização de *ns* torneio. A cada torneio, *nt* indivíduos da população atual são aleatoriamente escolhidos, e o indivíduo que vencer o torneio é selecionado para a fase de reprodução.

Na fase de reprodução, são aplicados operadores genéticos sobre os indivíduos selecionados, criando novos indivíduos. Os principais operadores genéticos são os operadores de *cruzamento* (ou recombinação) e de *mutação*. O operador de cruzamento é aplicado,

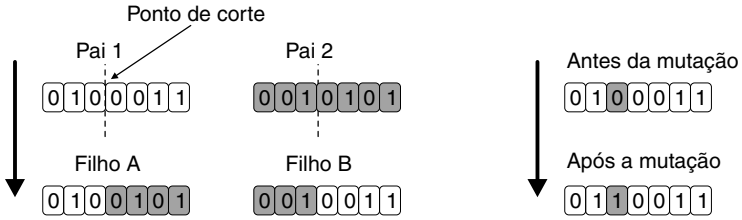


Figura 21.3 *Operadores genéticos.*

aleatoriamente, a pares de indivíduos selecionados na fase de reprodução, de acordo com uma taxa de cruzamento. O operador de cruzamento tradicional produz dois novos indivíduos (filhos) a partir de dois indivíduos selecionados (pais). No cruzamento de um ponto, o mais simples, um ponto de corte divide cada pai em duas partes. Cada filho recebe uma parte de um dos pais. Dessa forma, ele permite que caraterísticas dos dois pais passem para as próximas gerações. A Figura 21.3(a) ilustra o funcionamento do operador de cruzamento de um ponto. O operador de mutação permite a introdução e a manutenção da diversidade genética na população. Para esse efeito, este operador altera, de forma aleatória, uma ou mais componentes de uma estrutura escolhida. A sua aplicação é definida de forma probabilística utilizando uma taxa de mutação. O uso de um operador de mutação faz com que qualquer ponto do espaço de procura possa ser atingido. Em geral, o operador de mutação é aplicado a cada indivíduo após a aplicação do operador de cruzamento. A Figura 21.3(b) ilustra o funcionamento do operador de mutação.

Outro operador muito utilizado é o operador de elitismo. Quando este operador é aplicado, o indivíduo que apresenta maior desempenho é automaticamente selecionado para integrar a próxima geração, evitando modificações desse indivíduo pelos operadores genéticos. Ou seja, este operador é utilizado para garantir que os melhores indivíduos não desaparecem da população. O operador de elitismo também pode selecionar mais do que um indivíduo por geração.

A sequência de operações seleção-reprodução é repetida até que seja atingido um critério de paragem. Diferentes critérios de paragem têm sido utilizados. Exemplos incluem: número máximo de repetições do ciclo (gerações), obtenção de pelo menos uma solução satisfatória, ou a falta de melhoria, por um dado número de gerações, do valor de aptidão do melhor indivíduo (Carvalho et al., 2003).

Algumas das vantagens dos AGs residem no fato de estes algoritmos realizarem procuras simultâneas, em várias regiões do espaço de soluções (Michaelewicz, 1996). Isto permite que se encontrem soluções diferentes, elevando-os à categoria de métodos de procura global. Por esse motivo, os AGs também são definidos como técnicas de otimização global. Além disso, os AGs são capazes de realizar procuras em espaços de soluções (hi-

póteses) com regiões complexas, em que o impacto de cada parte, na solução do problema, pode ser difícil de modelar através de técnicas convencionais (Mitchell, 1997).

Por outro lado, uma das principais desvantagens dos AGs é o seu custo computacional. Dependendo da complexidade da aplicação investigada, a sua utilização pode revelar-se computacionalmente cara. Contudo, os AGs são facilmente paralelizáveis, possibilitando a exploração do crescente uso de tecnologias distribuídas.

Existem duas variantes de AGs que obtiveram grande destaque nas últimas décadas: a Programação Genética (PG) (Koza, 1992), utilizada para evoluir programas, e os Sistemas de Classificadores (LCSs, do inglês *Learning Classifier Systems*) (Sigaud e Wilson, 2007), utilizada para evoluir regras de classificação. Classificados, por alguns, como uma subárea da CE, a PG permite evoluir programas. Para isso, as soluções são em geral representadas por grafos em forma de árvores, em que os nós internos representam comandos, chamadas de funções e operações, e os nós externos, ou nós folha, representam variáveis, constantes ou comandos de entrada ou saída de dados. Na PG, a população inicial é formada por um conjunto de programas de computador, usualmente gerados de forma aleatória. A avaliação de cada indivíduo é realizada através da execução do programa representado por este indivíduo. Para que os programas possam ser executados, os indivíduos devem ser programas sintaticamente válidos. Os LCSs utilizam conceitos de AGs para evoluir uma população de regras, ou componentes de regras, para lidar com um problema de classificação. Nestes sistemas, a aptidão, ou qualidade, de uma regra pode ser estimada com recurso a diferentes critérios, tais como: a sua capacidade preditiva e a compreensibilidade.

21.3 Considerações Finais

Neste capítulo foram apresentadas duas das principais áreas da computação bio-inspirada, que constitui um subgrupo da computação natural, nomeadamente: a *inteligência de enxames* e a *computação evolutiva*. Foram apresentados os conceitos basilares envolvidos na inteligência de enxames, e foram abordadas as meta-heurísticas bio-inspiradas para problemas de *otimização por colónia de formigas* e *otimização por enxame de partículas*. Foram também introduzidos os principais conceitos de um dos tópicos mais populares na computação bio-inspirada: a *Computação Evolutiva*. Mostrámos as diferentes variações de CE, e apresentámos os principais passos seguidos pelos AGs na resolução de problemas. Foram, ainda, explicados os operadores de seleção e de reprodução mais utilizados nos trabalhos com AGs. Existem outras técnicas de computação bio-inspiradas, como os *sistemas imunes artificiais* e a *computação baseada em moléculas*, que assentam em modelos de raciocínio semelhantes ao material apresentado neste capítulo.

Capítulo 22

Análise de Redes Sociais

22.1 Introdução

O mundo é um sistema complexo de partes interligadas. As relações entre estas componentes podem ser representadas por uma rede, cuja estrutura pode ser analisada através da perspetiva de Análise de Redes Sociais (ARS) (*Social Network Analysis - SNA*). ARS é uma área de investigação interdisciplinar com contribuições da Sociologia, Psicologia Social, Antropologia, Física, Matemática, Ciências da Computação, entre outras, sendo um campo de conhecimento que tem beneficiado de forma significativa dos esforços de colaboração de investigadores de diferentes áreas. Uma vez que as redes foram estudadas de forma independente por disciplinas distintas, cada uma desenvolveu o seu próprio jargão. Para evitar ambiguidade e clarificar a linguagem adotada apresentamos, na Tabela 22.1, a terminologia de rede usada em diferentes áreas.

As origens da ARS, como base para o desenvolvimento de conceitos sociológicos, remontam à década de 1930, quando Moreno (1953) desenvolveu a abordagem sociométrica como uma forma de conceptualizar a estrutura das relações sociais estabelecida entre pequenos grupos de indivíduos. Estes vínculos interpessoais entre os membros de um grupo foram descritos usando os denominados *sociogramas*, que podem ser definidos como gráficos onde os indivíduos são representados como nós e as relações entre eles são representadas por linhas. Tais diagramas revelaram-se úteis na descoberta das estruturas ocultas dos grupos, por meio da identificação de, por exemplo, subgrupos, alianças entre atores sociais e ligações fortes e fracas.

Num sentido mais amplo, uma rede social é construída a partir de dados relacionais e pode ser definida como um conjunto de entidades sociais, tais como pessoas, grupos e organizações, com algum padrão de relações ou interações entre elas. Estas redes são normalmente modeladas através de grafos matemáticos, onde os vértices representam as entidades sociais e as arestas representam os vínculos estabelecidos entre elas. A estrutura subjacente a estas redes é o objeto de estudo da ARS. Nesse sentido, métodos e técnicas de ARS foram concebidos para descobrir padrões de interação entre atores sociais, em redes sociais.

Tabela 22.1 *Terminologia utilizada em diferentes áreas de conhecimento*

Matemática	Ciências de Computação	Sociologia	Física	Outros
Vértice	Nó	Ator/Agente	Sítio	Ponto
Aresta	Ligação/Conexão	Vínculo relacional	Vínculo	Arco

Assim, o foco da ARS incide sobre as relações estabelecidas entre as entidades sociais, e não nas entidades sociais em si. Não obstante, é possível complementar a análise das redes com informação sobre as caraterísticas individuais dos atores sociais, como forma de auxiliar a compreensão dos fenómenos sociais. Porém, o principal objetivo da ARS consiste em examinar tanto o conteúdo como o padrão dos relacionamentos sociais (e não as entidades sociais isoladas), com o intuito de compreender as relações entre os atores sociais e as implicações dessas relações.

Tarefas comuns da ARS envolvem a identificação dos atores mais centrais, mais influentes e com maior prestígio, utilizando medidas estatísticas; a identificação de *hubs* e *authorities*, usando algoritmos de análise de ligações; e a descoberta de comunidades, utilizando técnicas de deteção de comunidades. Estas tarefas são extremamente úteis no processo de extração de conhecimento de redes e, consequentemente, no processo de resolução de problemas. Devido à natureza apelativa de tais tarefas e ao elevado potencial deste tipo de análises, a ARS tornou-se uma abordagem popular numa miríade de áreas, desde a Biologia à Gestão. Por exemplo, algumas empresas utilizam ARS com o intuito de maximizar o passa-palavra positivo dos seus produtos, visando os clientes com maior *valor de rede*, i.e. aqueles com maior influência e suporte (Domingos e Richardson, 2001; Richardson e Domingos, 2002; Leskovec et al., 2007). Outras empresas, nomeadamente as que operam no sector das telecomunicações móveis, aplicam técnicas de ARS às redes de chamadas telefónicas de modo a identificar perfis de cliente e, com base neles, recomendar tarifários móveis personalizados. Estas empresas também recorrem à ARS para previsão de *churn*, i.e. para detetar clientes que potencialmente podem mudar para uma operadora móvel concorrente, por meio da deteção de mudanças nos padrões dos contactos telefónicos (Wei e Chiu, 2002; Dasgupta et al., 2008).

Outra aplicação interessante emerge no domínio da Deteção de Fraude. Por exemplo, as técnicas de ARS podem ser aplicadas a redes de comunicação organizacional (e.g. o conjunto de dados da empresa Enron), a fim de realizar uma análise da frequência e da direção da comunicação formal/informal, realizada via correio eletrónico, que pode revelar padrões de comunicação entre os colaboradores e os gestores. Estes padrões podem ajudar a identificar pessoas envolvidas em atividades fraudulentas, promovendo assim a adoção de formas mais eficientes de atuar com vista à erradicação do crime (Xu e Che, 2005; Shetty e Adibi, 2004).

Além das redes sociais, existem outro tipo de estruturas do mundo real passíveis de

serem representadas por redes. De acordo com Newman (2003b), as redes reais podem ser classificadas em quatro tipos principais: *redes sociais*, *redes de informação* (ou redes de conhecimento), *redes tecnológicas* e *redes biológicas*. Como mencionado anteriormente, as *redes sociais* são as que surgem como resultado das interações humanas e sociais, e englobam estudos de redes de amizade (Van De Bunt et al., 1999), redes de comunicação informal dentro das empresas (Ritter, 1999), redes de colaboração (Newman, 2001) (e.g. redes de coparticipação de atores em filmes, em que dois atores estão ligados se apareceram juntos em, pelo menos, um filme; e as redes de coautoria entre académicos, na qual dois indivíduos estão ligados se tiverem publicado pelo menos um artigo científico em conjunto), entre outros. Por sua vez, *redes de informação* são baseadas na troca de informações entre entidades, geralmente com o objetivo de promover a difusão do conhecimento, de negócios ou de objetivos sociais. Exemplos de redes de informação incluem as redes de citações de artigos científicos, geralmente representadas por um grafo acíclico direcionado, onde os vértices representam os artigos científicos e existe uma aresta direcionada entre dois vértices A e B, se o artigo A cita o artigo B; e redes de preferência, que geralmente são modeladas através de grafos bipartidos, e representam as preferências de consumo dos indivíduos por um determinado produto comercial (e.g. livros) (Truyen et al., 2007). Outro exemplo importante de uma rede de informação é a *World Wide Web*, que pode ser representada como um grafo direcionado, no qual os vértices representam as páginas estáticas da Web e as arestas correspondem aos *hiperlinks* que se estabelecem entre elas (Broder et al., 2000). As *redes tecnológicas* são redes construídas pelo Homem para efetuar a distribuição de alguns bens ou recursos (e.g. eletricidade, informação). As redes de estradas e caminhos-de-ferro, as redes de rotas aéreas e as redes de ligação física entre computadores, constituem alguns exemplos deste tipo de redes. O último tipo de redes são as designadas *redes biológicas* (Alon, 2003) e, tal como o nome indica, são aquelas que emergem de processos biológicos, tais como redes de reações químicas entre metabolitos, redes de interação proteica, redes de regulação genética, redes neuronais reais e redes alimentares ou redes predador-presa.

Não obstante o facto da origem dos estudos de redes remontarem a algumas décadas atrás, nos últimos anos tem-se assistido a um avanço impressionante em campos de conhecimento relacionados com as redes. Este fenómeno deve-se mormente ao crescente interesse nas redes sociais, que se tornou um tema importante e alvo de um foco de atenção considerável. Por este motivo, a ARS tem despertado a curiosidade e o interesse de muitos estudantes, profissionais e investigadores, motivando-os a explorar, mesmo que superficialmente, o potencial das técnicas de ARS no estudo dos seus problemas. Tendo isto em mente, o objetivo deste capítulo consiste em fornecer uma visão geral e concisa dos fundamentos da ARS para os interessados em saber um pouco mais sobre a área e fortemente orientados para a utilização de ARS em problemas práticos.

Este capítulo encontra-se organizado da seguinte forma. Começamos por indicar alguns tipos de representações que podem ser utilizados para modelar redes sociais. Na Secção 22.3 são introduzidas as medidas estatísticas mais conhecidas para analisar redes, a dois níveis de análise: ao nível do ator e ao nível da rede. A secção seguinte é dedicada

à análise de ligações. Nesta secção, é descrito o funcionamento do algoritmo PageRank e explicado como este tipo de algoritmos permite identificar nós influentes. Na Secção 22.6 distinguem-se dois importantes modelos de rede e são introduzidas as principais propriedades das redes reais. Posteriormente, dedicamos a Secção 22.5 ao problema da descoberta de comunidades em redes. Este capítulo encerra com a identificação das tendências atuais no campo da redes sociais.

22.2 Representação de Redes Sociais

Uma rede social consiste num conjunto finito de atores e nas relações, ou vínculos, definidas sobre os mesmos (Wasserman e Faust, 1994). As relações estabelecidas podem ser de natureza pessoal ou profissional, e podem variar entre relações casuais a fortes vínculos familiares. Além de relações sociais, as ligações também podem representar fluxo de informação/bens/ dinheiro, interações, semelhança, entre outros. A estrutura dessas redes é geralmente representada com recurso a grafos matemáticos. Por esse motivo, as redes são muitas vezes encaradas como equivalentes a grafos.

Um grafo é composto por duas unidades fundamentais: vértices e arestas. Cada aresta é definida por um par de vértices, denominados *extremidades* da aresta. Vértices são passíveis de representar uma grande variedade de entidades individuais (e.g. pessoas, organizações, países, artigos científicos, produtos, plantas e animais), consoante a área de aplicação. Por sua vez, uma aresta é a linha que liga dois vértices e, analogamente, pode representar vários tipos de relacionamentos entre entidades individuais (e.g. comunicação, cooperação, amizade, parentesco, relações casuais e trocas). Arestas podem ser *direcionadas* ou *não direcionadas*, dependendo se a natureza da relação é assimétrica ou simétrica. Formalmente, um grafo G consiste num conjunto não vazio de vértices $V(G)$ e um conjunto de arestas $E(G)$, e pode ser definido como $G = (V(G), E(G))$. Segundo Diestel (2005), a *ordem* de um grafo G é dada pelo número total de vértices n ou, matematicamente, $|V(G)| = n$. Analogamente, e segundo o mesmo autor, o *tamanho* de um grafo G é o número total de arestas $|E(G)| = m$. O número máximo de arestas num grafo com n vértices é dado pela expressão $m_{max} = \frac{n(n-1)}{2}$, para grafos não direcionados, e $m_{max} = n(n-1)$, para grafos direcionados.

Na literatura de Teoria dos Grafos, são referidos dois tipos de estruturas de dados para representar grafos: o primeiro é *estruturas de lista* e o segundo é *estruturas matriciais*. Estes dois tipos de estruturas são adequados para armazenar grafos em computadores, a fim de analisá-los usando ferramentas automáticas. Estruturas de lista, tais como listas de incidência e listas de adjacência, são adequadas para o armazenamento de grafos esparsos, uma vez que reduzem o espaço de armazenamento necessário. Por outro lado, estruturas matriciais, tais como matrizes de incidência ($A_{n\times m}$), matrizes de adjacência ou *sociomatrices* ($A_{n\times n}$), matrizes Laplacianas[1] e matrizes de distância (distinguem-se das matrizes de adjacência pelo facto das entradas da matriz representarem o comprimento do

[1] As matrizes Laplacianas incluem informação sobre a adjacência e sobre o grau de cada vértice.

caminho mais curto entre pares de vértices), são apropriadas na representação de matrizes completas.

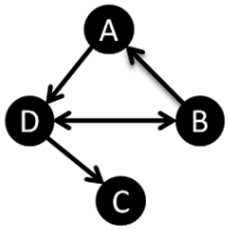
Diversos tipos de grafos podem ser utilizados para modelar diferentes tipos de redes sociais. Por exemplo, os grafos podem ser classificados de acordo com a direção das ligações. Isto conduz à diferenciação entre *grafos não direcionados* e *grafos direcionados*. Grafos não direcionados (ou redes não direcionadas) são grafos cujas arestas conectam pares de vértices não ordenados ou, analogamente, cada aresta do grafo liga concomitantemente dois vértices. Um tipo de grafo mais restrito é o denominado grafo direcionado (ou rede direcionada). Grafos direcionados ou, na forma abreviada, *dígrafos*, podem ser definidos como grafos cujas arestas (também denominadas de *arcos*) têm uma orientação atribuída. Por esse motivo, é possível atribuir um significado à ordem dos vértices. Formalmente, um grafo direcionado D é um par ordenado $(V(D),A(D))$ constituído por um conjunto não vazio de vértices $V(D)$ e um conjunto $A(D)$, disjunto de $V(D)$, de arcos. Se e_{12} é um arco e v_1 e v_2 são vértices de tal forma que $e_{12} = (v_1, v_2)$, então podemos afirmar que e_{12} liga v_1 a v_2, sendo o primeiro vértice v_1 denominado *vértice inicial*, ou *cauda*, e o segundo vértice v_2 denominado *vértice terminal*, ou simplesmente *cabeça*. Graficamente, arestas direcionadas são representadas através de setas, que indicam a direção da ligação.

Este tipo de grafos podem ser cíclicos, i.e. grafos contendo circuitos fechados de arestas ou estruturas em anel, ou *acíclicos* (e.g. árvores). Um exemplo típico de um grafo não direcionado é o Facebook™ uma vez que, nesta rede social, o vínculo de amizade estabelecido é mútuo, ou recíproco. Isto porque, se um indivíduo aceitar o pedido de amizade de outro indivíduo, assume-se implicitamente que ambos os indivíduos são amigos. Por outro lado, o Twitter™ é um exemplo de um grafo direcionado, uma vez que um indivíduo pode ser seguido por outros indivíduos, sem necessariamente ter de segui-los também. Neste caso, o vínculo estabelecido entre um par de indivíduos é direcionado, sendo a cauda da aresta o indivíduo "seguidor"e a cabeça da aresta o o indivíduo "seguido", gerando uma relação do tipo unidirecional.

No que respeita aos valores atribuídos às arestas, é possível fazer uma distinção entre *grafos pesados* e *grafos não pesados*. A não ser que seja expressamente dito o contrário, assume-se sempre que os grafos são não pesados. Grafos não pesados são binários uma vez que as arestas existem, ou não existem. Por outro lado, os grafos pesados são grafos que contêm informação mais rica uma vez que cada aresta tem associado um peso $w \in \mathbb{R}_0^+$, que fornece informação adicional ao utilizador sobre, por exemplo, a força da ligação do par de vértices. De acordo com Mark Granovetter (1973, 1974), em redes sociais o peso de uma ligação geralmente é função da duração, intensidade emocional, frequência de interação, intimidade e troca de serviços. Assim, ligações fortes geralmente representam amigos próximos, e ligações fracas representam conhecidos. Noutro tipo de redes, o peso de uma ligação pode representar uma panóplia de coisas, dependendo do contexto; por exemplo, uma ligação pode representar o número de assentos nos aeroportos, o número de produtos transacionados, etc.

Para grafos não direcionados e não pesados, as matrizes de adjacência são *binárias*, devido ao facto dos grafos serem não pesados, e *simétricas*, como consequência dos grafos

Vértice	A	B	C	D
A	-	0	0	1
B	1	-	0	1
C	0	0	-	0
D	0	1	1	-



Vértice	Vértice
A	D
B	A
B	D
D	B
D	C

Figura 22.1 *Grafo direcionado D representado através de uma matriz de adjacência (lado esquerdo da Figura) e de uma lista de adjacência (lado direito da Figura).*

serem não direcionados (i.e. $a_{ij} = a_{ji}$). Assim, se existir uma aresta a conectar os vértices i e j, a entrada da matriz assume a forma de $a_{ij} = 1$; alternativamente, se não existir ligação entre o par de vértices (i, j), a entrada da matriz é nula, ou seja, $a_{ij} = 0$. Por outro lado, se os grafos forem direcionados e pesados, as entradas da matriz de adjacência assumem valores pertencentes ao intervalo $[0, max(w)]$, e são assimétricas. Em ambos os casos, as matrizes são não negativas. Na Figura 22.1 é apresentado um exemplo de como um grafo direcionado D pode ser representado através de uma lista de adjacência e de uma matriz de adjacência.

22.3 Medidas Estatísticas Elementares

Como se pôde constatar na Secção anterior, para representar redes é comum recorrer à Matemática, mais especificamente ao ramo da Teoria dos Grafos. Porém, quando se pretende analisar estas redes torna-se fundamental recorrer a outro ramo da Matemática: a Estatística. Nesta subsecção introduzimos algumas medidas utilizadas na análise de redes sociais e que são oriundas do campo da Estatística. A utilidade destas medidas está diretamente relacionada com o facto de permitirem compreender e avaliar a estrutura da rede sem a necessidade de conhecer a sua representação gráfica. Assim, o objetivo destas medidas é quantificar a estrutura das redes, com vista a possibilitar ao analista o alcance de uma compreensão de alto nível do comportamento dos fenómenos sociais que geraram essas redes.

As medidas que irão ser introduzidas nas subsecções seguintes podem ser classificadas de acordo com o nível de análise que se pretende realizar: ao nível dos atores, ou ao nível da própria rede. No primeiro caso são exploradas medidas gerais de centralidade que permitam perceber como a posição de um dado ator está embutida na estrutura global da rede. Por conseguinte, estas medidas ajudam a identificar os principais agentes da rede. Por sua vez, o segundo nível de análise produz informação mais compacta que permite

a avaliação da estrutura global da rede, providenciando conhecimento sobre propriedades importantes dos fenómenos sociais subjacentes.

22.3.1 Medidas Estatísticas ao Nível do Ator

A *centralidade*, ou *prestígio*, é um indicador geral da posição de um ator na estrutura global da rede social, podendo ser calculado recorrendo a várias medidas. As mais populares são o *grau*, a *intermediação*, a *proximidade* e a *centralidade do vetor próprio*. As três primeiras medidas foram propostas por Freeman (1979), tendo inicialmente sido desenvolvidas para redes não pesadas. Recentemente, Brin e Page (2010) estenderam estas medidas para redes pesadas. A quarta medida - centralidade do vetor próprio - foi posteriormente proposta por Bonacich (1987) e os seus fundamentos assentam na Teoria Espetral de Grafos. Esta medida tornou-se especialmente popular depois de ter sido utilizada como base do célebre algoritmo do Google, o PageRank, o qual iremos introduzir na próxima secção.

Embora tenham sido propostas na literatura outras medidas estatísticas ao nível do ator, nesta subsecção iremos concentrar-nos na explicação das medidas mencionadas. Estas medidas determinam a importância relativa de um ator dentro da rede e indicam a sua posição social, uma vez que permitem demonstrar como as relações estabelecidas entre os vários atores de uma rede estão concentradas num pequeno grupo de indivíduos. Atores que obtêm valores elevados nas medidas de centralidade são, por norma, atores poderosos na rede, uma vez que a sua posição central lhes confere inúmeras vantagens, nomeadamente: acesso mais fácil e mais rápido a outros atores na rede, o que se torna útil no acesso a recursos (e.g. informação); e capacidade de exercer controlo sobre o fluxo de comunicação estabelecido entre os restantes atores (Freeman, 1979). Estes atores centrais são também denominados de *pontos focais*. No final da secção, introduzimos também o conceito de *transitividade* e explicamos como este conceito pode ser quantificado usando o *coeficiente de agrupamento*. É importante sublinhar que, para efeitos de comparação de redes com diferentes ordens e tamanhos, convém proceder à normalização de algumas destas medidas (e.g. grau, intermediação e proximidade).

Centralidade do Grau ou Valência

O *grau* ou *valência*, de um nó v, geralmente denotado por k_v, é uma medida da adjacência imediata e do envolvimento do nó na rede, sendo calculado como o número de arestas incidentes num dado nó ou, de forma análoga, como o número de vizinhos do nó v. Neste contexto, a vizinhança é definida pelo conjunto de nós que estão diretamente ligados a v. Tendo por base estas definições, é possível deduzir que o grau pode ser calculado de duas formas alternativas: com base na matriz de adjacência, ou com base na vizinhança do nó. Nas Equações 22.1 e 22.2 apresentamos cada uma das alternativas, para o caso de redes não direcionadas. Não obstante a sua simplicidade, o grau revela-se uma medida eficaz na avaliação da importância e da influência de um ator numa rede social. No entanto, apresenta algumas limitações. A principal limitação prende-se com o facto desta medida

não tomar em consideração a estrutura global da rede, uma vez que é calculada de forma local (i.e. para cada nó).

$$k_i = \sum_{j=1}^{n} a_{ij}, 0 < k_i < n \quad (22.1) \qquad k_v = |N_v|, 0 < k_v < n \quad (22.2)$$

onde a_{ij} é a entrada da i-ésima linha e da j-ésima coluna da matriz de adjacência A e N_v é a vizinhança do nó v.

Para redes direcionadas, existem duas variantes da centralidade do grau: *grau de entrada*, denotado por k_v^+, e *grau de saída*, denotado por k_v^-. O *grau de entrada* é dado pelo número de nós de entrada (ou seja, pelo número de arestas que começam no vértice v) e o *grau de saída* pelo número de nós de saída (i.e. pelo número de arestas que terminam no vértice v), conforme definido nas Equações 22.3 e 22.4. A medida do grau em redes direcionadas também é comummente referida como *prestígio*. Esta expressão é especialmente utilizada na literatura de redes sociais, uma vez que foi desenvolvida para medir a proeminência, ou importância, dos atores numa rede. Existem dois tipos de prestígio: o *suporte* e a *influência*. O *suporte* está relacionado com o grau de entrada, que pode ser visto como uma medida de apoio, e a segunda variante está relacionada com o grau de saída, que é visto como uma medida de influência. Em redes pesadas, a *força* é equivalente ao grau, sendo calculada como a soma dos pesos das arestas adjacentes a um dado nó, conforme expresso na Equação 22.5.

$$k_i^+ = \sum_{j=1}^{n} a_{ji} \quad (22.3) \qquad k_i^- = \sum_{j=1}^{n} a_{ij} \quad (22.4)$$

$$k_i^w = \sum_{j=1}^{n} a_{ij}^w \quad (22.5)$$

Um esforço significativo de investigação foi empreendido no âmbito do estudo da *distribuição do grau* de vários tipos de redes, que tornou possível classificar uma rede com base na sua distribuição do grau. Por exemplo, Barabási et al. (Barabasi e Albert, 1999; Barabasi e Bonabeau, 2003) descobriram que a distribuição do grau de algumas redes segue uma *lei de potência*, pelo menos assintoticamente. Isto significa que, nessas redes, a distribuição do grau dos vértices é muito heterogénea e apresenta uma cauda pesada, uma vez que a grande maioria dos vértices apresenta um grau baixo e um pequeno número de vértices apresenta graus elevados. Estas redes são, por isso, conhecidas por *redes livres de escala* (*scale-free* em inglês), que é uma expressão cunhada pelos mesmos investigadores que descobriram esta propriedade (Barabasi e Albert, 1999; Barabasi e Bonabeau, 2003). Outras formas funcionais comuns são distribuições *exponenciais* (e.g. vias férreas e redes de energia elétrica) e leis de potência (e.g. redes de atores de cinema e algumas redes de colaboração).

Centralidade de Intermediação

A *intermediação de um nó* b_v indica em que medida um dado nó se situa entre os outros nós da rede e pode ser calculado usando a fórmula apresentada na Equação 22.6. Os nós que obtêm valores altos de intermediação tendem a ocupar papéis críticos na estrutura da rede, uma vez que geralmente detêm uma posição na rede que lhes permite funcionar como interface entre grupos fortemente coesos, sendo elementos vitais na conexão de diferentes regiões da rede. Na perspetiva das redes sociais, as *'interações entre dois atores não adjacentes pode depender de outros atores, especialmente dos atores que se localizam nos caminhos entre esses dois atores'* (Wasserman e Faust, 1994), o que salienta a importância de obter um bom valor de intermediação. Esses atores são também denominados de *gatekeepers*, uma vez que tendem a controlar o fluxo de informação que se estabelece entre as comunidades.

$$b_v = \sum_{s,t \in V(G) \setminus v} \frac{\sigma_{st}(v)}{\sigma_{st}} \tag{22.6}$$

onde σ_{st} representa o número de caminhos mais curtos entre os vértices s e t (geralmente, $\sigma_{st} = 1$) e $\sigma_{st}(v)$ indica o número de caminhos mais curtos que passam pelo vértice v.

Esta medida também por ser calculada para as arestas (ver Equação 22.7). A *intermediação de uma aresta* b_e é comummente definida como o número de caminhos mais curtos entre pares de nós que percorrem uma aresta específica da rede. Esta medida é bastante útil em ARS uma vez que permite identificar *pontes* e *pontes locais* que, por definição, são arestas com elevada intermediação. No contexto de ARS, *pontes* são ligações fora do círculo de conhecidos de um indivíduo. Estas ligações têm um enorme interesse para os indivíduos que procuram aceder a novas informações e recursos, uma vez que facilitam a difusão de novas informações a várias comunidades da rede (Kossinets e Watts, 2006). Porém, situações como estas são bastante raras em cenários do mundo real e, quando ocorrem, tendem a conferir vantagens temporárias devido à instabilidade temporal deste tipo de arestas (i.e. pontes). A situação mais comum e realista são as denominadas *pontes locais*, i.e. arestas que permitem encurtar significativamente a distância entre vários subgrupos na rede mas que, ao contrário das *pontes*, não representam o único caminho possível de ligação entre estes subgrupos.

$$b_e = \sum_{u,v \in V(G)} \frac{\sigma_{uv}(e)}{\sigma_{uv}} \tag{22.7}$$

onde $\sigma_{uv}(e)$ denota o número de caminhos mais curtos que passam pela aresta e. O somatório indica que esta fração tem de ser calculada para todos os pares de nós (u,v) na rede.

Centralidade de Proximidade

A *proximidade* é uma medida aproximada da posição global de um dado ator na rede, fornecendo uma ideia de quanto tempo um dado nó inicial demora a alcançar todos os outros nós da rede. Formalmente, a *proximidade* é dada pelo comprimento médio de todos os caminhos mais curtos entre um dado nó e todos os outros nós da rede. Devido à forma como é definida, geralmente a *proximidade* só é calculada para os nós contidos na maior componente conectada da rede (do inglês *giant component*), usando a fórmula apresentada na Equação 22.8. No contexto das redes sociais, a proximidade é um indicador de acessibilidade que mede a rapidez com que um determinado ator consegue alcançar todos os outros atores na rede.

$$Cl_v = \frac{n-1}{\sum_{u \in V(G) \setminus v} d(u,v)} \tag{22.8}$$

onde $d(u,v)$ representa o comprimento do caminho mais curto entre os nós u e v.

Centralidade do Vetor Próprio

Esta medida, proposta por Bonacich (1987), tem por base a ideia de que a centralidade de um dado ator social é definida pela centralidade dos atores com quem estabelece relações. Ou seja, o poder e o status de um dado ator é recursivamente definido pelo poder e status dos seus *alters* [2]. Assim, a centralidade de um determinado ator pode ser definida como a combinação linear das centralidades dos seus vizinhos de primeira ordem. Esta é a suposição por detrás da fórmula da *Centralidade do Vetor Próprio*, definida da seguinte maneira:

$$x_i = \frac{1}{\lambda} \sum_{j=1}^{n} a_{ij} x_j \tag{22.9}$$

onde x_i/x_j denota a centralidade do nó i/j, a_{ij} representa a entrada da matriz de adjacência $\mathbf{A}$ ($a_{ij} = 1$ se os nós i e j estão ligados por uma aresta, e $a_{ij} = 0$, caso contrário) e λ indica o maior valor próprio da matriz $\mathbf{A}$. A *centralidade do vetor próprio* é uma versão mais elaborada da *centralidade do grau*, visto que assume que nem todas as ligações de um dado ator têm a mesma importância social, tomando em consideração não apenas a quantidade, mas sobretudo a qualidade das suas ligações.

Coeficiente de Agrupamento Local

As redes sociais são naturalmente transitivas, o que significa que existe uma elevada probabilidade dos amigos de um determinado ator também serem amigos. A transitividade

[2] *Alters* é um termo frequentemente utilizado na abordagem egocêntrica da Análise de Redes Sociais, referindo-se aos atores que estão diretamente ligados a um ator específico, ou nó focal, comummente designado por *ego*.

é uma propriedade que pode ser quantificada com base num *coeficiente de agrupamento*, que pode ser *global*, ou seja, calculado para toda a rede, ou *local*, i.e. calculado para cada nó. Watts e Strogatz (1998) propuseram uma versão local do coeficiente de agrupamento, denotado por c_i ($i = 1, ..., n$). Neste contexto, a transitividade é uma propriedade local da vizinhança de um nó que indica o nível de coesão entre os respetivos vizinhos. Tendo por base esta definição, o coeficiente de agrupamento local é dado pela fração de pares de nós, que são vizinhos de um dado nó v e que estão ligados uns aos outros através de arestas, conforme apresentado na Equação 22.10.

$$c_i = \frac{2|e_{jk}|}{k_i(k_i-1)} : v_j, v_k \in N_i, e_{jk} \in E \tag{22.10}$$

onde N_i é a vizinhança do nó v_i, e_{jk} representa a aresta que liga o nó v_j ao nó v_k, k_i é o grau do nó v_i, e $|e_{jk}|$ indica a proporção de ligações estabelecidas entre os nós da vizinhança de v_i.

22.3.2 Medidas Estatísticas ao Nível da Rede

Antes de explicar cada uma das medidas estatísticas ao nível da rede, é fundamental introduzir três conceitos basilares: *caminho*, *distância geodésica* (ou *caminho mais curto*) entre dois vértices e *excentricidade* de um vértice.

Um *caminho* é uma sequência de vértices em que pares consecutivos de vértices não repetidos estão ligados por uma aresta. O primeiro vértice de um caminho é denominado de *vértice inicial*, enquanto o *último vértice* de um caminho é denominado de *vértice final*. Outro conceito de especial interesse em ARS é o de *distância geodésica*, ou *caminho mais curto*, entre os vértices i e j, denotado por $d(i, j)$. A *distância geodésica* define-se como o comprimento do caminho mais curto, ou do caminho mínimo, entre os vértices i e j. Por sua vez, a *excentricidade* é dada pela maior distância geodésica entre um dado vértice v e qualquer outro vértice no grafo, conforme definido na Equação 22.11.

$$\varepsilon_v = \max_{i \in V(G) \setminus v} d(v, i) \tag{22.11}$$

Estes três conceitos encontram-se na base da maioria das medidas ao nível da rede que irão ser introduzidas, nomeadamente, do *diâmetro/raio*, da *distância geodésica média*, do *grau médio*, da *reciprocidade*, da *densidade* e do *coeficiente de agrupamento global*.

Diâmetro e Raio

O *diâmetro D* é dado pela excentricidade máxima do conjunto de vértices que define a rede e, analogamente, o *raio R* pode ser definido como a excentricidade mínima do conjunto de vértices, conforme definido nas Equações 22.12 e 22.13. Geralmente, redes esparsas apresentam maior diâmetro do que redes completas, devido à existência de caminhos menores entre pares de vértices. Leskovec et al. (2005) descobriu que, para certo tipo de redes reais,

o diâmetro efetivo diminui ao longo do tempo, o que contraria a sabedoria convencional de diâmetros cada vez maiores. No contexto da ARS, esta medida fornece uma ideia da proximidade de pares de atores na rede, indicando quão distantes dois atores estão um do outro, no pior dos casos.

$$D = \max\{\varepsilon_v : v \in V\} \quad (22.12) \qquad R = \min\{\varepsilon_v : v \in V\} \quad (22.13)$$

Distância Geodésica Média

A *distância geodésica média* para todas as combinações de pares de vértices numa rede é geralmente denotada por l, e calculada com base na Equação 22.14. Esta medida fornece uma ideia de quão longe dois vértices estão um do outro, em média. Por exemplo, no contexto da ARS, a *distância geodésica média* pode ser utilizada para medir a eficiência do fluxo de informação no interior da rede.

$$l = \frac{1}{\frac{1}{2}n(n-1)} \sum_{i \geq j} d(i,j) \quad (22.14)$$

onde $d(i,j)$ representa a distância geodésica, ou comprimento do caminho mais curto, entre os vértices i e j, e $\frac{1}{2}n(n-1)$ indica o número máximo de arestas numa rede composta por n vértices. A fórmula anterior não se aplica quando uma rede contém mais do que uma componente conectada, visto que, por convenção, a distância geodésica é infinita quando não existe um caminho na rede que ligue um dado par de vértices. Em tais situações, é mais apropriado recorrer à *média harmónica da distância geodésica*, definida na Equação 22.15, uma vez que permite transformar distâncias infinitas em distâncias nulas.

$$l^{-1} = \frac{1}{\frac{1}{2}n(n+1)} \sum_{i \geq j} \frac{1}{d(i,j)} \quad (22.15)$$

Grau Médio

O *grau médio* é simplesmente a média dos graus de todos os vértices na rede, conforme indicado na Equação 22.16. De acordo com Costa et al. (2011), o grau médio pode ser encarado como uma medida da conectividade global da rede.

$$\bar{k} = \frac{1}{n} \sum_{i=1}^{n} k_i \quad (22.16)$$

Reciprocidade

A *reciprocidade r* é uma grandeza específica para redes direcionadas, que mede a tendência de pares de vértices para criarem ligações mútuas entre si. Existem várias formas para calcular esta medida. A forma mais popular e intuitiva consiste em calcular a razão entre o número de ligações mútuas na rede e o número de todas as ligações existentes, conforme

expresso pela Equação 22.17. O valor da reciprocidade representa, assim, a probabilidade de existência de simetria nas ligações estabelecidas entre pares de vértices ($a_{ij} = 1$ e $a_{ji} = 1$). Por definição, em redes não direcionadas, a reciprocidade é sempre máxima, uma vez que todos os pares de vértices têm ligações simétricas.

$$r = \frac{\#mut}{\#mut + \#asym}, 0 < r < 1 \tag{22.17}$$

onde *#mut* indica o número de *díades* (i.e. pares de vértices) mútuas e *#asym* representa o número de *díades* assimétricas.

Adotando as definições propostas por Wasserman e Faust (1994), considera-se que uma díade assimétrica é um par de vértices ligados por um arco, cuja direção assume o sentido de um dos vértices, mas não de ambos. Por sua vez, uma díade mútua define-se como um par de vértices conectados por dois arcos, cada um deles indo numa direção diferente (e.g. $a \rightarrow b$ e $b \rightarrow a$, sendo a e b dois vértices de uma rede).

Densidade

A *densidade* ρ é uma importante medida ao nível da rede, uma vez que é capaz de quantificar o nível de conectividade presente numa rede. Esta medida pode ser definida como a proporção de arestas presentes na rede em relação ao número máximo possível de arestas (Equação 22.18). A densidade é uma quantidade que assume um valor mínimo de 0, quando a rede não tem arestas, e um valor máximo de 1, quando a rede está perfeitamente conectada [3]. Deste modo, valores elevados de densidade estão associados a *redes densas*, enquanto valores baixos de densidade estão associados a *redes esparsas*.

$$\rho(G) = \frac{m}{m_{max}}, 0 < \rho < 1 \tag{22.18}$$

onde m representa o número de arestas presentes numa rede e m_{max} indica o número máximo de arestas nessa rede. Quando as redes são não direcionadas, o número máximo de arestas é dado por $\frac{n(n-1)}{2}$, onde n denota o número de vértices da rede. Por outro lado, se a rede for direcionada, o número máximo de arestas é dado por $n(n-1)$.

Coeficiente de Agrupamento Global

Existem diversas formas de calcular a versão global do coeficiente de agrupamento. Neste manual, adotamos o método proposto por Watts e Strogatz (1998), que obtém o *coeficiente de agrupamento global*, para toda a rede, através do cálculo da média de todos os valores locais c_i ($i = 1, ..., n$), conforme apresentado na Equação 22.19. As denominadas redes de "pequenos mundos" (Watts e Strogatz, 1998), que emergem tipicamente em contextos sociais reais, são caraterizadas por coeficientes de agrupamento global elevados, o que

[3] Redes perfeitamente conectadas são também denominadas de *redes completas*, ou *cliques*

significa que a propriedade de transitividade entre pares de nós na rede se verifica com maior frequência, aumentando, assim, a probabilidade de formação de *cliques*.

$$c = \frac{1}{n}\sum_i c_i \tag{22.19}$$

22.4 Análise de Ligações

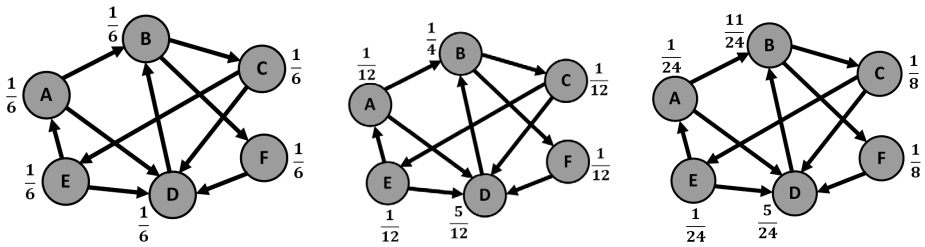
Em determinados cenários, tais como a Web, o analista pode estar interessado em descobrir o nó dominante, mais valioso ou com maior influência (e.g. página Web mais influente) ou uma lista ordenada de nós com essas caraterísticas. Neste âmbito, foram desenvolvidos vários algoritmos de análise de ligações, sendo os algoritmos Pagerank (Brin e Page, 1998) e HITS (Kleinberg, 1999) os mais populares. Estes algoritmos exploram a relação existente entre as ligações e o conteúdo das páginas Web, com o intuito de melhorar a tarefa de *recuperação de informação* na Web, sendo extremamente importantes no desenho de motores de busca eficientes. Visto que o desenvolvimento destes métodos foi motivado pelo problema de pesquisas na Web, por uma questão de simplicidade, iremos explicá-los neste contexto.

Antes de avançar, é essencial introduzir alguns conceitos elementares, nomeadamente, os conceitos de *hub* e *authority*. No contexto da Web, um *hub* pode ser entendido como uma página Web que aponta para muitas outras páginas Web ou, dito de outra forma, como uma compilação de páginas Web que abordam um tema específico. A qualidade de um *hub* é geralmente determinada pela qualidade das *authorities* para as quais aponta. Por outro lado, as *authorities* são páginas Web citadas por vários *hubs* diferentes, o que significa que a sua importância é medida pelo número de ligações que recebem de outras páginas. Normalmente, boas *authorities* são fontes confiáveis de informação sobre um determinado tema.

22.4.1 Algoritmo PageRank

O PageRank é um algoritmo de análise de ligações que se baseia no conceito de centralidade do vetor próprio. Este algoritmo é utilizado pelo motor de busca da Google™ na medição da importância, ou relevância, das páginas da Internet. A relevância de uma página é medida com base no valor da informação transmitida por essa mesma página. Desta forma, as páginas cuja informação é considerada mais valiosa tendem a aparecer no topo dos resultados das pesquisas efetuadas no motor de busca da Google™.

A ideia do algoritmo é que a informação da Web pode ser classificada de acordo com a popularidade da ligação (i.e. quanto maior o número de páginas Web ligadas a uma dada página Web, maior a sua popularidade). No entanto, neste processo de atribuição de pesos às páginas Web, não só é importante o número de ligações de uma dada página Web (i.e. o grau do nó) mas também a relevância das respetivas ligações. Deste modo, o PageRank mede a importância relativa de um conjunto de páginas Web tendo por base



(a) Inicialização (b) Iteração 1 (c) Iteração 2

Figura 22.2 *Ilustração do processo subjacente ao algoritmo PageRank, numa rede composta por* 6 *nós. A primeira rede* **(a)** *corresponde ao passo de inicialização. Na rede* **(b)** *apresentam-se os valores PageRank atualizados no final da primeira iteração do algoritmo. Note-se que o nó D é, até ao momento, o nó com maior autoridade, uma vez que possui o maior valor PageRank (*$PR(D) = \frac{5}{12}$*). A rede do lado direito da figura, i.e. a rede* **(c)**, *corresponde à segunda (ou última) iteração do PageRank. Nesta última rede, é possível verificar que o nó B ultrapassa o nó D, em termos de valor PageRank.*

não apenas a quantidade, mas sobretudo a qualidade das respetivas ligações. De acordo com a definição fornecida por Easley e Kleinberg (2010), o algoritmo PageRank básico funciona da seguinte maneira:

Inicialização: Numa rede com n nós (ou páginas Web), atribuir a cada nó um valor PageRank dado por $\frac{1}{n}$, e escolher o número de iterações k do algoritmo.

1. Aplicar sequencialmente a seguinte regra, para efeitos de atualização dos valores PageRank de cada nó:

   Regra Básica de Atualização do PageRank: dividir o atual valor PageRank do nó p pelo número de ligações de saída e transmitir estas *shares* (i.e. quotas, partes) aos nós para os quais aponta. Note-se que se um nó p não tem ligações de saída, a *share* do PageRank é transmitida ao próprio nó. A atualização do valor PageRank de um nó consiste na soma das *shares* que o nó recebe em cada iteração

2. Aplicar esta regra até à k-ésima iteração ou, alternativamente, até os resultados convergirem.

Para efeitos de ilustração, considere o seguinte exemplo: numa rede composta por 6 nós denominados A, B, C, D, E e F, como encontrar o nó com maior influência usando o algoritmo PageRank? Primeiro, o algoritmo é inicializado através da atribuição de um valor igual de PageRank a cada nó da rede, dado por $PR = \frac{1}{n} = \frac{1}{6}$, conforme representado na Figura 22.2-(a). O passo seguinte consiste na atualização dos valores PageRank de cada nó, por via da aplicação da *regra básica de atualização do PageRank*. Por uma questão de simplicidade, neste exemplo consideramos apenas duas iterações, i.e. $k = 2$.

Tabela 22.2 *Valores PageRank atualizados após a primeira iteração $k = 1$.*

Nó	**A**	**B**	**C**	**D**	**E**	**F**
Shares	$\frac{1}{12}$	$\frac{1}{12}$	$\frac{1}{12}$	$\frac{1}{6}$	$\frac{1}{12}$	$\frac{1}{6}$
Valor PageRank atualizado	$\frac{1}{12}$	$\frac{1}{4}$	$\frac{1}{12}$	$\frac{5}{12}$	$\frac{1}{12}$	$\frac{1}{12}$

Tabela 22.3 *Valores PageRank atualizados após a segunda (e última) iteração $k = 2$.*

Nó	**A**	**B**	**C**	**D**	**E**	**F**
Shares	$\frac{1}{24}$	$\frac{1}{8}$	$\frac{1}{24}$	$\frac{5}{12}$	$\frac{1}{24}$	$\frac{1}{12}$
Valor PageRank atualizado	$\frac{1}{24}$	$\frac{11}{24}$	$\frac{1}{8}$	$\frac{5}{24}$	$\frac{1}{24}$	$\frac{1}{8}$

Para aplicar esta regra, primeiro é necessário calcular as *shares* de todos os nós. Depois, procede-se à soma de todas as *shares* que um dado nó recebe, de modo a obter o novo valor PageRank, conforme apresentado na Tabela 22.2 e representado na Figura 22.2-(b). Por exemplo, a *share* do nó D, que possui apenas uma ligação de saída, é calculada da seguinte maneira: $share(D) = \frac{1/6}{1} = \frac{1}{6}$. O novo valor PageRank deste nó é dado pela soma das *shares* das suas ligações de entrada, nomeadamente, das ligações provenientes dos nós A, C, E e F:

$$PR(D) = \frac{1}{12} + \frac{1}{12} + \frac{1}{12} + \frac{1}{6} = \frac{5}{12} \tag{22.20}$$

Após calcular estes valores para todos os nós da rede, o processo é repetido na segunda iteração do algoritmo, obtendo-se os resultados apresentados na Tabela 22.3 e ilustrados na Figura 22.2-(c). Esta regra é aplicada de forma iterativa até à k-ésima iteração ou até os resultados do algoritmo convergirem. Dado que apenas se consideraram duas iterações, procuraremos interpretar os resultados e extrair algumas conclusões tendo por base apenas a informação disponível nas Tabelas 22.2 e 22.3. No final da primeira iteração, o nó D parecia ser o nó mais promissor, com um valor PageRank de $\frac{5}{12}$; porém, na segunda iteração o nó B supera o nó D, assumindo o primeiro lugar da lista de nós ordenados com base na relevância. Esta mudança repentina confirma a ideia subjacente ao algoritmo PageRank, que se baseia na medição da qualidade, em detrimento da quantidade, das ligações de um nó. Assim, e apesar do nó D ser o que recebe um maior número de ligações de entrada, a importância destes nós não é significativa. Por outro lado, o nó B possui apenas duas ligações de entrada, mas uma delas é de extrema importância, nomeadamente, a ligação proveniente do nó D. Esta constitui a principal razão porque o nó B recebe o valor PageRank mais elevado no final da segunda iteração, tornando-se o nó mais influente

da rede (i.e. torna-se uma *authority*). Se B representasse um ator social, ele(ela) seria considerado(a) o(a) ator mais importante da rede, uma vez que o valor PageRank obtido indica que grande parte da informação que circula na rede passa por ele(ela).

Apesar do desenvolvimento do PageRank, e de outros algoritmos de análise de ligações, ter sido originalmente motivado pela necessidade de extrair e compreender a informação gerada pela Web, estes algoritmos também são utilizados noutros domínios do conhecimento, nomeadamente nas Ciências Sociais. No campo das Ciências Sociais, as ligações podem ser analisadas através de duas perspetivas diferentes, mas inter-relacionadas: a perspetiva centrada na informação e a perspetiva centrada nos atores. Tipicamente, estas perspetivas ajudam a compreender os fenómenos sociais subjacentes, por meio da identificação das fontes de informação mais valiosas ou, alternativamente, dos atores sociais mais relevantes. No entanto, ainda se verifica alguma ausência de princípios orientadores consensuais sobre como interpretar os resultados da análise de ligações no contexto das Ciências Sociais. Thelwall (2006) enfatiza a importância de desenvolver diretrizes no sentido de melhorar o processo de interpretação destes resultados, e propõe uma metodologia para solucionar este problema.

22.5 Deteção de Comunidades

Uma das caraterísticas únicas das redes sociais é que estas possuem *estrutura de comunidade*. Normalmente, esta propriedade emerge como consequência da heterogeneidade global e local da distribuição das arestas num grafo. Por conseguinte, neste tipo de redes é comum encontrar concentrações elevadas de arestas em determinadas regiões do grafo, denominadas *comunidades*, e baixa concentração de arestas entre essas regiões.

Comunidades, também denominadas por *clusters*, podem ser definidas como grupos de vértices densamente conectados, com ligações esparsas entre eles. De acordo com Newman e Girvan (2004), existem duas linhas de investigação principais na descoberta de comunidades em redes. A primeira teve origem no campo da Ciência de Computadores e é conhecida como *partição de grafos*, enquanto a segunda foi essencialmente desenvolvida por sociólogos, sendo usualmente referida como *blockmodeling*, *agrupamento hierárquico* ou *deteção de estrutura de comunidades*. A primeira linha de investigação surgiu originalmente no campo da Ciência de Computadores, tendo o seu surgimento sido motivado pela necessidade de encontrar a melhor forma de alocar tarefas a processadores de modo a minimizar as comunicações entre eles. Esta tarefa de otimização da rede tinha como objetivo melhorar a computação, num ambiente de computação paralela. Por sua vez, a segunda linha de investigação foi motivada pela descoberta de grupos de comunidades na sociedade, com o intuito de simplificar a análise dos fenómenos sociais através do agrupamento dos atores sociais de acordo com o seu grau de semelhança. O processo básico subjacente aos algoritmos de deteção de comunidades baseia-se na divisão do grafo original num conjunto de subgrafos disjuntos, por via da otimização de uma dada função objetivo (e.g. modularidade). O propósito de ambas as abordagens é descobrir grupos de vértices relacionados e, se possível, definir a respetiva organização hierárquica, tendo apenas por base informação

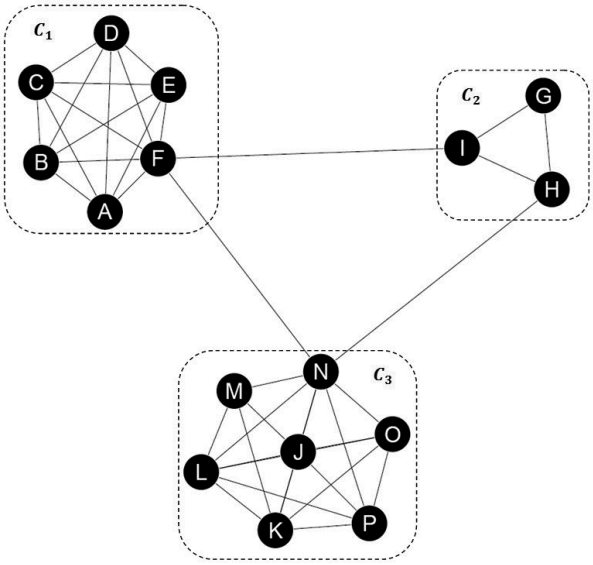


Figura 22.3 *Ilustração de uma rede com três comunidades distintas:* $C_1 = \{A,B,C,D,E,F\}$, $C_2 = \{G,H,I\}$ e $C_3 = \{J,K,L,M,N,O,P\}$.

fornecida pela topologia da rede. Isto é geralmente realizado removendo iterativamente as arestas *ponte* que ligam grupos de vértices, conforme sugerido por Girvan e Newman (2002).

Para compreender melhor os conceitos introduzidos, na Figura 22.3 está representada uma rede simples com três comunidades, denominadas C_1, C_2 e C_3. Esta figura representa uma situação ideal visto que cada comunidade é um subgrafo completo ou, analogamente, um *clique* de tamanho variável ($C_1 = K_6$, $C_2 = K_3$ e $C_3 = K_7$). Além disso, a densidade das ligações entre diferentes comunidades é bastante reduzida. As poucas ligações que existem são *pontes*, i.e. são as únicas ligações disponíveis entre diferentes partes da rede.

Na vida real é possível encontrar uma variedade considerável de exemplos de grupos coesos, ou comunidades. Como a lista de exemplos é muito longa, iremos citar apenas alguns. A sociedade é um ambiente rico para encontrar comunidades, uma vez que as pessoas têm uma tendência natural para formar grupos. Estes grupos podem ser famílias, círculos de amigos, grupos religiosos ou de trabalho, cidades, nações, etc. Se também considerarmos grupos formados por empresas, ou consumidores de um dado produto, é possível identificar comunidades com relevância para a área da Economia e da Gestão. A Biologia é outra atividade onde métodos de deteção de comunidades se podem revelar bastante úteis, sobretudo no âmbito das denominadas *redes metabólicas*. Por exemplo, em redes de interação proteína-proteína é possível encontrar grupos de proteínas que desempenham funções similares na célula. Por outro lado, na rede da Internet, é comum

encontrar comunidades virtuais online, ou grupos de páginas Web que abordam tópicos relacionados, o que pode revelar-se útil no desenvolvimento de sistemas de recomendação automáticos e eficientes.

A importância de estudar estas comunidades é intuitiva em domínios como a ARS. Para sublinhar esta importância, Fortunato (2010) afirmou que a análise da posição estrutural dos nós, em cada módulo (ou comunidade) da rede, pode ajudar a identificar *atores centrais* (i.e. atores que assumem posições centrais), comumente associados a funções de estabilidade e controle do grupo, bem como *atores intermediários*, que são aqueles que se localizam nas fronteiras das comunidades e desempenham um papel fundamental na disseminação e troca de informação e novas ideias, criando pontes entre comunidades. A tarefa de descoberta de comunidades também oferece a possibilidade de analisar outro tipo de descrições do grafo original. Um exemplo consiste no estudo de grafos em que os vértices representam as comunidades e as arestas são um indicador de sobreposição entre comunidades. Esta estratégia é utilizada por Oliveira e Gama (2010) na deteção de transições de comunidades.

As subsecções seguintes são dedicadas à introdução dos métodos mais populares para resolver o problema da descoberta de comunidades em redes. A grande maioria destes algoritmos tradicionais assumem partições de vértices, em vez de *covers*[4], i.e. não permitem sobreposição de comunidades, pelo que cada vértice pertence a uma única comunidade.

No entanto, se se suspeitar que a natureza da rede promove a existência de comunidades sobrepostas, uma escolha possível é o *Método de Percolação de Cliques* (do inglês *Clique Percolation Method*), proposto pelos físicos Palla et al. (2005). A principal caraterística desta abordagem prende-se com a capacidade de descobrir comunidades sobrepostas numa rede, ao permitir que cada vértice pertença a mais do que um grupo. Esta caraterística torna-se particularmente apelativa no domínio das Ciências Sociais, uma vez que os indivíduos tendem a pertencer a mais do que uma comunidade (e.g. família, grupo de amigos, grupo de trabalho, etc.) ao mesmo tempo. Para os leitores interessados em utilizar o *Método de Percolação de Cliques* na deteção de comunidades sobrepostas, encontra-se livremente disponível em `www.cfinder.org`, um pacote de software, denominado `CFinder`, que foi desenvolvido pelos próprios criadores do método (Palla et al., 2005).

22.5.1 Agrupamento Hierárquico

O agrupamento hierárquico é uma classe popular de métodos para encontrar *clusters*, ou grupos, uma vez que não exige nenhuma assunção no que respeita ao número de grupos, dimensão desses grupos e pertença dos objetos aos grupos. Os algoritmos de agrupamento hierárquico geram estruturas aninhadas flexíveis (pequenos grupos inseridos dentro de grupos maiores que, por sua vez, se encontram inseridos em grupos ainda maiores), tipicamente representadas através de dendrogramas que revelam a estrutura multinível da

[4] *Cover* é um sinónimo de *partição* no contexto do agrupamento difuso (do inglês *fuzzy clustering* ou *soft clustering*) de vértices em comunidades.

rede. Tais caraterísticas são desejáveis em domínios onde existe pouca informação disponível sobre a estrutura de comunidade da rede. Além disso, estes métodos têm-se revelado bastante eficazes na resolução de problemas de análise de grupos, tendo por esse motivo despertado um enorme interesse na resolução de problemas análogos, como sejam, o *particionamento de grafos* e a *deteção de comunidades*.

O procedimento tradicional associado ao agrupamento hierárquico é bastante intuitivo, sendo baseado na definição de semelhança. Normalmente, o primeiro passo consiste na escolha de uma medida de semelhança (ou dissemelhança), que irá ser utilizada para avaliar quão semelhantes são dois nós, de acordo com uma dada propriedade global ou local. Exemplos de tais medidas incluem a *semelhança do cosseno*, o *índice de Jaccard*, a *distância Euclidiana*, a *distância de Manhattan*, a *distância de Hamming* entre pares linhas numa matriz de adjacência, entre outras. O passo seguinte consiste no cálculo da matriz de semelhança entre todos os pares de nós, independentemente desses nós estarem, ou não, conectados um ao outro. De seguida, é selecionada a abordagem que irá ser utilizada apara agrupar os nós: *abordagem aglomerativa* ou *abordagem divisiva*. Dependendo da escolha, uma dada medida de distância é escolhida para calcular a semelhança entre grupos. Exemplos destas medidas incluem o *single linkage*, ou vizinho mais próximo, o *complete linkage*, ou vizinho mais afastado, e o *método de Ward*. O resultado final deste processo é um dendrograma que ilustra a organização dos nós retornada pelo algoritmo hierárquico. Para selecionar a melhor partição, i.e. o melhor número de comunidades k, uma estratégia comum consiste em calcular o valor de *modularidade* (Newman e Girvan, 2004) para cada número possível de comunidades e selecionar o número que maximiza essa função.

Como anteriormente mencionado, existem duas grandes abordagens para o agrupamento hierárquico, nomeadamente:

- *Métodos Divisivos*: esta classe de métodos foca-se na identificação e remoção das ligações que conectam regiões densamente conectadas na rede (Easley e Kleinberg, 2010), nomeadamente, *pontes* e *pontes locais*. O célebre algoritmo proposto por Girvan e Newman (2002) explora este método.
- *Métodos Aglomerativos*: esta classe de métodos foca-se nas regiões mais densas da rede, em vez de se focar nas ligações existentes nas fronteiras destas regiões. O algoritmo *Walktrap* (Pons e Latapy, 2005) é um bom exemplo de um algoritmo baseado neste método.

Na subsecção seguinte, é introduzido um dos algoritmos hierárquicos mais conhecido e utilizado na descoberta de comunidades em redes sociais: o *algoritmo de Girvan-Newman*.

Algoritmo de Girvan-Newman

Um dos algoritmos mais populares e difundidos na literatura para resolver problemas de deteção de comunidades em redes é o denominado *algoritmo de Girvan-Newman*, desenvolvido por Girvan e Newman (2002). Este algoritmo usa uma técnica hierárquica divisiva que desconstrói progressivamente a rede inicial em pequenas partes conectadas,

Algoritmo 22.1 Algoritmo de Girvan-Newman.

Entrada: Um grafo
Saída: Comunidades

1. Cálculo do valor de intermediação de todas as arestas na rede;
2. Remoção da aresta que obtenha maior valor de intermediação. Este passo pode causar a divisão da rede em regiões desconectadas. Estas regiões constituem o primeiro nível de regiões na partição do grafo.
3. Repetição dos passos anteriores até que todas as arestas do grafo tenham sido removidas.

até ao ponto em que não é possível remover mais arestas e cada nó representa uma única comunidade. Tendo em conta que comunidades são grupos coesos de nós, com ligações esparsas entre eles, o critério para remoção das ligações, proposta por Girvan e Newman (2002), é a medida de centralidade *intermediação da aresta*. O motivo subjacente a esta escolha está relacionada com o facto desta medida de centralidade ser capaz de identificar arestas que se localizam numa zona da rede onde passa uma grande quantidade de caminhos mais curtos entre nós e, por essa razão, tendem a conectar diferentes comunidades não sobrepostas. Em suma, a ideia principal deste algoritmo é que as comunidades de uma rede podem ser isoladas através da identificação e remoção de arestas *ponte*.

Este algoritmo, por ser baseado no conceito de intermediação, é apenas apropriado para redes de dimensão moderada, devido ao elevado custo de computação exigido no cálculo da medida de intermediação. O *input* deste algoritmo é um grafo e o *output* é uma estrutura, ou árvore hierárquica, mais especificamente, um dendrograma. Diferentes partições da rede podem ser obtidas por via do corte do dendrograma a diferentes níveis da hierarquia. O algoritmo 22.1 sumariza os passos principais.

22.5.2 Otimização da Modularidade

Outra classe de métodos bastante popular e amplamente utilizada na deteção de comunidades em redes é a denominada *otimização da modularidade*. A *modularidade Q* é uma função de qualidade que procura avaliar e medir o mérito de uma dada partição da rede em comunidades. Esta função tem sido utilizada não só para efeitos de comparação da qualidade das partições obtidas por diferentes métodos de deteção de comunidades, mas também como uma função objetivo em problemas de otimização. Segundo Newman (2006), a modularidade é representada pela diferença normalizada entre o número de arestas observadas no interior de cada grupo de nós da rede e o número de arestas que seria expectável observar no interior desse mesmo grupo numa rede equivalente onde as arestas são geradas aleatoriamente. A modularidade é calculada da seguinte forma:

$$Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j) \tag{22.21}$$

onde m indica o número de arestas; k_i e k_j representam, respetivamente, o grau dos vértices i e j; A_{ij} é a entrada da matriz de adjacência que indica o número de ligações estabelecidas entre os vértices i e j; $\frac{k_i k_j}{2m}$ representa o número esperado de arestas que deveria existir entre o par de vértices (i, j); c_i e c_j denotam os grupos a que os vértices i e j pertencem; e $\delta(c_i, c_j)$ representa o delta de Kronecker.

Com base nesta fórmula, é possível deduzir que $Q \in [-1, 1]$, podendo assumir valores positivos ou negativos. Se Q for positivo, então existe a possibilidade de encontrar estrutura de comunidade na rede. Se Q for um número positivo e elevado, então a respetiva partição tem uma maior probabilidade de refletir a estrutura de comunidade verdadeira. De acordo com Clauset et al. (2004), uma modularidade que assuma valores superiores ou iguais a 0.3 é um bom indicador da existência de comunidades com significado na rede.

Com base neste raciocínio, e sabendo que quanto maior a modularidade, melhor é a divisão da rede obtida, uma abordagem natural consistiria em maximizar esta medida, por via do cálculo desta medida para todas as possíveis partições da rede, e selecionar a partição associada a um maior valor desta medida. Esta ideia simples originou uma nova classe de métodos cujos fundamentos são baseados na maximização da modularidade. Porém, e apesar desta abordagem ser bastante atrativa, a procura exaustiva sobre todas as possíveis divisões da rede é geralmente intratável em termos computacionais. Este efeito indesejado da ineficiência computacional tem sido contornado pela adoção de métodos heurísticos a este problema de otimização específico. Seguindo esta estratégia, é possível obter uma aproximação bastante boa do ótimo global (neste caso, o ótimo global corresponderia ao valor máximo de modularidade), num tempo aceitável. Vários algoritmos adotam esta estratégia, nomeadamente, o algoritmo proposto por Blondel et al. (2008), que efetua uma otimização hierárquica da modularidade explorando técnicas gulosas, e o algoritmo proposto por Guimera e Amaral (2005), que aplica o procedimento de *Simulated Annealing* ao problema de otimização de modularidade. Para os leitores interessados em explorar mais aprofundadamente este tópico, é aconselhada a leitura do artigo de Fortunato (2010).

A ideia base do algoritmo proposto por Blondel et al. (2008) consiste em encontrar a partição que maximiza a modularidade. É algoritmo de passos múltiplos com base na optimização local da modularidade na vizinhança de cada nó. Após a identificação de uma partição, as comunidades são substituídos por supernós, dando origem a uma rede ponderada menor. O procedimento é então iterativo, até modularidade, calculada em relação ao grafo original, não aumentar mais. Este algoritmo obtêm um compromisso entre a precisão da estimativa da máxima modularidade, e complexidade computacional, que é linear no número de ligações do grafo. O algoritmo está descritos nos Algoritmos 22.2 e 22.3. A grande vantagem deste algoritmo é o baixo esforço computacional, devido à sucessiva redução do número de vértices. O algoritmo pode ser aplicado redes com milhões de nós. A principal desvantagem, é que não é um algoritmo determinista. O resultado vai depender da escolha inicial da ordem pela qual se percorrem os vértices.

Algoritmo 22.2 Algoritmo Blondel - 1 Fase: maximizar a modularidade a nível local

Entrada: Um grafo
Saída: Comunidades

1. Cada nó é um comunidade;
2. Numa ordem pré-definida, determinar para cada nó o ganho de modularidade por mudar para uma comunidade vizinha.
3. Se a modularidade aumentar, move o nó para essa comunidade;
4. Repetir este processo até não se conseguir ganho na modularidade.

Algoritmo 22.3 Algoritmo Blondel - 2 Fase: Construção nova rede (supergraph)

Entrada: Um grafo
Saída: Comunidades

1. Cada nó (supervértice) agrega os nós de cada comunidade identificada na 1ª fase;
2. Supervértices estão ligados se existir pelo menos uma ligação entre nós dessas comunidades.

22.6 Propriedades de Redes Reais

Os problemas do mundo real são uma fonte inesgotável de inspiração para as teorias de redes. A grande maioria dos eventos do mundo real e das atividades que desenvolvemos, observamos e estudamos podem ser facilmente modeladas utilizando grafos e, posteriormente, analisadas à luz da metodologia das redes sociais.

O foco deste capítulo cinge-se às redes sociais, que são redes que emergem como consequência de interações sociais e humanas. No entanto, existem outro tipo de redes. Dada a enorme diversidade de redes passíveis de encontrar no mundo real, os investigadores resolveram classificá-las em tipos principais, apesar desta classificação não ser totalmente consensual. Uma classificação possível é a proposta por Newman (2003b), que foi previamente explicada na Introdução a este capítulo.

Apesar destas redes derivarem de campos de conhecimento distintos e serem oriundas de diferentes problemas do mundo real, partilham um conjunto de propriedades que as tornam peculiares, opondo-se, desta forma, a dois modelos de redes bem conhecidos: as *redes aleatórias* e as *redes regulares*. Por esse motivo, estas redes tendem a denominar-se *redes complexas*. O mais célebre e mais simples modelo de redes é o denominado *grafo*

aleatório (Rapoport, 1953; Erdos e Renyi, 1960). Este tipo de grafo é caraterizado pela geração aleatória de arestas entre um número fixo de vértices n, para efeitos de criação de uma rede na qual cada uma das $\frac{1}{2}n(n-1)$ arestas possíveis é gerada de forma independente com probabilidade p. Quando $p = 0$, obtém-se um grafo de ordem perfeita - *grafo regular* -, e quando $p = 1$ obtém-se um grafo aleatório que, devido à ausência de regularidade, representa o caos total. Visto que ambos os tipos de grafos - *grafos regulares* e *grafos aleatórios* - representam extremos, não refletem a realidade. Para modelar redes complexas e atípicas como aquelas que encontramos no mundo real, são necessárias propriedades adicionais (Newman, 2003b). Em suma, pode-se afirmar que redes reais são grafos não-regulares e não-aleatórios com caraterísticas únicas, onde a 'ordem coexiste com a desordem' (Fortunato, 2010). Nesta secção introduzimos e explicamos algumas destas propriedades das redes complexas, nomeadamente:

1. Fenómeno do 'Mundo Pequeno';
2. Transitividade ou Agrupamento;
3. Distribuição do Grau dos Nós e Lei de Potência;
4. Resiliência da Rede;
5. Padrões de Mistura;
6. Estrutura de Comunidade.

Propriedade 1: Fenómeno do 'Mundo Pequeno'

Stanley Milgram (1967), um psicólogo social americano, foi o primeiro a sugerir a existência de efeitos dos "pequenos mundos" em redes sociais reais, por via de uma série de experiências famosas que são atualmente conhecidas por a *Experiência de Milgram*. Esta experiência foi conduzida com o intuito de testar a ideia especulativa do efeito de "pequenos mundos" e constitui uma das primeiras demonstrações diretas deste efeito. A hipótese principal do estudo era de que pares de indivíduos aparentemente distantes estão ligados por um caminho curto, i.e. por um pequeno número de conhecidos, na rede. Para investigar a distribuição do comprimento dos caminhos que ligam pares de indivíduos, Milgram pediu a cerca de 300 participantes aleatórios, residentes em Omaha, Wichita e Nebraska, para enviar um *dossier* a alguém que conhecessem pessoalmente com o objetivo final de fazer chegar esse *dossier* a um determinado indivíduo-alvo (neste caso, um corretor de Boston). Com base nesta experiência foi possível demonstrar que o comprimento médio dos caminhos que alcançaram o indivíduo-alvo era aproximadamente 6, o que explica claramente a origem do conceito dos *seis graus de separação*.

A conclusão da experiência de Milgram foi aceite de forma generalizada. De facto, o efeito dos "pequenos mundos" tem sido amplamente observado em redes do mundo real, manifestando-se pela existência de *atalhos* (do inglês *shortcuts*) entre a maioria dos pares de nós numa rede. Em cenários sociais, isto significa que duas pessoas aparentemente separadas uma da outra podem rapidamente entrar em contacto uma com a outro através

de um número incrivelmente baixo de conhecidos ou amigos. Esta descoberta tem importantes implicações em processos dinâmicos, uma vez que implica que, por exemplo, a propagação de doenças contagiosas a toda a população ocorre a um ritmo mais rápido do que aquele que seria expectável. Em termos matemáticos, o efeito dos "pequenos mundos" implica que a distância geodésica média (i.e. o caminho mais curto) entre pares de nós cresça logaritmicamente com a dimensão de uma rede de grau médio fixo (Newman, 2003b). Esta propriedade também se observa em grafos aleatórios, onde o diâmetro é muito reduzido (apenas cresce logaritmicamente com a dimensão n da rede) e todos os vértices têm aproximadamente o mesmo grau.

Propriedade 2: Transitividade ou Agrupamento

De acordo com Wasserman e Faust (1994), a *transitividade* é uma propriedade que considera *triplas de nós* (i.e. conjuntos de três vértices, no qual pelo menos um dos vértices está conectado aos outros dois vértices) num grafo. Dito de outra forma, a transitividade é uma propriedade que mede a densidade de *triângulos* (i.e. três nós interligados por três arestas na rede, o que significa que cada nó está totalmente conectado aos restantes nós) na rede. Na linguagem de redes sociais, isto significa que a probabilidade de um amigo do meu amigo também ser meu amigo é elevada.

Esta propriedade é quantificada usando um *coeficiente de agrupamento*, que pode ser *local* ou *global*, conforme mencionado na Secção 22.3.

Propriedade 3: Distribuição do Grau dos Nós e Lei de Potência

A *distribuição do grau* $P(k)$ é a distribuição de probabilidade dos graus dos nós numa rede. Deste modo, $P(k)$ representa a probabilidade de um vértice escolhido aleatoriamente e de forma uniforme possuir grau k, e é definida como a fração de nós na rede que têm grau k. Isto significa que, se o número total de nós na rede é n, e n_k destes nós possuem grau k então, para este valor do grau, a probabilidade é dada pela expressão $P(k) = \frac{n_k}{n}$. Se esta probabilidade for calculada para cada valor de grau k encontrado na rede, obtém-se a distribuição de probabilidade do grau desta rede.

Grafos aleatórios, tais como os estudados por Erdos e Renyi (1960), seguem uma distribuição binomial do grau, uma vez que a presença, ou ausência, de uma aresta é equiprovável (i.e. igual para todos os pares de vértices existentes). No limite, i.e. para grafos de elevada dimensão, a distribuição binomial do grau é bem aproximada por uma distribuição de Poisson. Deste modo, esta classe específica de grafos tende a apresentar distribuições do grau altamente homogéneas, dado que a maioria dos vértices da rede possui grau igual ou semelhante.

O mesmo não se verifica em redes reais, onde a distribuição do grau é bastante distinta da distribuição do grau observada em redes aleatórias. Barabasi e Albert (1999) descobriram que, em redes reais, a distribuição dos graus dos vértices é assimétrica positiva e muito heterogénea, uma vez que a grande maioria dos vértices assume valores pequenos de grau

e uma pequena minoria obtém valores elevados nesta medida. Esta descoberta vem reforçar o anterior trabalho de Price (1965) sobre redes de citações de artigos científicos. Em ambos os casos, os investigadores argumentam que a distribuição do grau em redes reais, tais como redes de citações, seguem uma lei de potência (pelo menos assintoticamente) e, por esse motivo, estas redes são comummente referidas como *redes livres de escala* (do inglês *scale-free*) (Barabasi e Bonabeau, 2003). Normalmente, distribuições que seguem uma lei de potência emergem quando a quantidade que um indivíduo obtém de alguma coisa depende da quantidade que esse indivíduo já possui dessa mesma coisa. Uma analogia comum é a de que *dinheiro faz dinheiro* ou *os ricos ficam mais ricos*. Price (1976) utilizou o termo *vantagem cumulativa* para se referir a este mecanismo, que se acredita ser a explicação mais provável para distribuições de grau que seguem leis de potência em redes reais, e que incluem, mas não estão restritas a, redes de colaboração e a World Wide Web. Nos dias de hoje, este processo é melhor conhecido como *ligação preferencial* (do inglês *preferential attachment*), expressão cunhada por Barabasi e Albert (1999). No seu artigo seminal, Barabasi e Albert (1999) descrevem um modelo de crescimento de redes que se tornou conhecido como o modelo *Barabási-Albert*. Este trabalho mostra que o crescimento de uma rede com ligações preferenciais irá torná-la numa rede livre de escala, devido à estratégia *os ricos ficam mais ricos* empregue no modelo.

Propriedade 4: Resiliência da Rede

A resiliência da rede mede o impacto na conectividade da rede quando um, ou mais, vértices são removidos, sendo um bom indicador da coesão da rede. Diferentes tipos de redes exibem diferentes níveis de resiliência. A maioria das redes são robustas à remoção aleatória de vértices mas consideravelmente menos robustas à remoção propositada de vértices com grau elevado. Além disso, quando os vértices que definem uma aresta *ponte* são eliminados, verificam-se fortes mudanças na rede no que respeita à capacidade de comunicação entre pares de vértices, dado que alguns destes pares ficam desconectados. A centralidade de intermediação dos vértices também pode ser vista como uma medida de resiliência, uma vez que indica quantos caminhos geodésicos irão tornar-se mais longos se um dado vértice for removido da rede. Porém, em cenários reais, a remoção de um único nó não é, geralmente, motivo para alarme, uma vez que as redes são compostas por milhões, ou mesmo biliões, de nós. Nestes casos, considera-se mais apropriado testar a resiliência da rede tendo por base a remoção de uma dada percentagem de nós.

Propriedade 5: Padrões de Mistura

Algumas redes são compostas por diferentes tipos de vértices. Neste tipo de redes, a ligação entre vértices ou, analogamente, a probabilidade de ligação entre pares de vértices, tende a ser seletiva e bastante dependente do tipo de vértice (e.g. nas redes alimentares, os vértices podem representar plantas, animais herbívoros ou animais carnívoros). Em redes sociais, esta propriedade também é evidente, uma vez que os indivíduos tendem a interagir com indivíduos semelhantes. Esta ligação seletiva é usualmente denominada

por *assortatividade* ou *homofilia*, sendo a *mistura por raça* um exemplo clássico deste fenómeno. Por norma, as redes sociais apresentam maior tendência para a assortatividade.

Newman (2003a) propôs um *coeficiente de assortatividade* para, tal como o próprio nome indica, quantificar a tendência para a assortatividade numa dada rede. Este coeficiente substitui o coeficiente previamente proposto por Gupta et al. (1989), e permite distinguir redes de mistura aleatória de redes perfeitamente assortativas.

Propriedade 6: Estrutura de Comunidade

A grande maioria das redes sociais reais apresentam estrutura de comunidade, o que significa que é possível encontrar grupos de vértices densamente conectados que estão esparsamente conectados a outros grupos de vértices na rede. Este tópico vai ser aprofundado na secção seguinte.

22.7 Conclusões e Tendências Atuais

Nos primórdios da ARS, as análises baseavam-se em redes pequenas. Os primeiros estudos assentavam em dados recolhidos usando questionários diretos, onde se pedia aos inquiridos que detalhassem as suas interações sociais. Os dados recolhidos eram depois representados através de grafos matemáticos, onde os vértices representavam os indivíduos (i.e. os participantes inquiridos) e as arestas denotavam as relações estabelecidas entre eles. Estes estudos tradicionais acarretavam alguns problemas, tais como imprecisão, subjetividade e impossibilidade de generalização dos resultados, devido à reduzida dimensão das amostras.

O avanço substancial da tecnologia, a facilidade de acesso a computadores e o desenvolvimento de redes de comunicação sofisticadas contribuíram para o surgimento de novos movimentos de investigação na área das redes. Estas novas abordagens começaram a focar-se na análise das propriedades estatísticas de redes complexas de grande dimensão, que eram facilmente recolhidas usando computadores e outros dispositivos eletrónicos. Esta mudança de escala exigiu uma correspondente mudança na abordagem analítica tradicional. Neste contexto, os métodos de *Data Mining* assumiram uma grande importância, devido à sua capacidade para extrair padrões e conhecimento de quantidades massivas de dados (Newman, 2003b). Atualmente, o fácil acesso a software, ferramentas e librarias de ARS (mais de 50, de acordo com a Wikipedia (2012)) reflete a evolução desta área de conhecimento. Devido a estes avanços tecnológicos e ao consequente impacto na acessibilidade e disponibilização de dados sob a forma de redes, novos desafios estão a ser colocados ao campo de conhecimento da ARS, e um novo paradigma está a emergir. Este paradigma tem em conta novos fatores na análise de redes sociais, tais como a dimensão das redes, que está a tornar-se incrivelmente grande, e as mudanças ocorridas no tempo e no espaço.

O primeiro fator tem implicações diretas nos métodos de ARS existentes, que agora necessitam de melhorar a sua eficiência e escalabilidade, de modo a adaptarem-se a redes

sociais de elevada dimensão. Por exemplo, a descoberta de comunidades em redes sociais de grande escala continuará a ser um problema desafiante de investigação. Também na área de deteção de comunidades, a necessidade de desenvolver novos métodos que, além de escaláveis e eficientes, são também totalmente automáticos, no sentido em que não exigem ao utilizador a definição de parâmetros (e.g. número e dimensão das comunidades), também continuará a ser um importante problema de investigação nesta área. O segundo fator mencionado é de extrema relevância, visto que a rapidez a que atualmente os dados são recolhidos está a tornar as análises estáticas obsoletas. Por esse motivo, o estudo da dinâmica e da evolução de redes sociais, que incluem, mas não estão restritas, à descoberta das propriedades globais que regem a evolução temporal das redes sociais e à deteção e compreensão das mudanças espaciais e temporais nessas redes, irá também continuar a ser uma vertente relevante de investigação no campo da ARS.

É igualmente expectável que a popularidade da ARS continue a aumentar, atraindo cada vez mais investigadores para a área e impulsionando um número crescente de empresas a incorporar os métodos de ARS nos seus processos de negócio, generalizando o seu uso enquanto ferramenta estratégica.

Bibliografia

Aamodt, A. e Plaza, E. (1994). Case based reasoning: foundational issues, methodological variations, and systems approaches. *AI Communications*, 7:39–59. Citado em: pages 84, 86, 87

Aggarwal, C., Han, J., Wang, J. e Yu, P. (2003). A framework for clustering evolving data streams. In: *Proceedings of the 29th International Conference on Very Large Databases*, p. 81–92. Morgan Kaufmann. Citado em: pages 307, 310, 313

Aggarwal, C. C. (2013). *Outlier Analysis*. Springer. Citado em: pages 55, 56

Agrawal, R., Gehrke, J., Gunopulos, D. e Raghavan, P. (1998). Automatic subspace clustering of high dimensional data for data mining applications. In: *Proceedings of 1998 ACM-SIGMOD International Conference on Management of Data*, p. 94–105. Citado em: pages 259

Agrawal, R., Imielinski, T. e Swami, A. (1993). Mining association rules between sets of items in large databases. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*, p. 207–216, Washington D.C.. Citado em: pages 215, 217

Agrawal, R. e Srikant, R. (1994). Fast algorithms for mining association rules in large databases. In: Bocca, J. B., Jarke, M. e Zaniolo, C. (Ed.) *Proceedings of the 20th International Conference on Very Large Data Bases, VLDB*, p. 487–499, Santiago, Chile. Citado em: pages 217

Aha, D. W., Kibler, D. e Albert, M. K. (1991). Instance-based learning algorithms. *Machine Learning*, 6:37–66. Citado em: pages 74, 82

Ali, K. e Pazzani, M. (1996). Error reduction through learning multiple descriptions. *Machine Learning, Vol. 24, No. 1*. Citado em: pages 169, 170, 189

Allwein, E. L., Shapire, R. E. e Singer, Y. (2000). Reducing multiclass to binary: a unifying approach for margin classifiers. In: *Proceedings of the 17th International Conference on Machine Learning*, p. 9–16. Citado em: pages 330, 334, 338

Alon, U. (2003). Biological networks: The tinkerer as an engineer. *Science*, 301(5641):1866–1867. Citado em: pages 373

Andrews, R., Cable, R., Diederich, J., Geva, S., Golea, M., Hayward, R., Ho-Stuart, C. e Tickle, A. B. (1996). An evaluation and comparison of techniques for extracting and refining rules from artificial neural networks. Relatório técnico, Queensland University of Technology. Citado em: pages 149

Andrews, R., Diederich, J. e Tickle, A. B. (1995). A Survey and Critique of Techniques for Extracting Rules from Trained Artificial Neural Networks. *Knowledge-Based Systems*, 8(6):373–389. Citado em: pages 149

Ankerst, M., Breunig, M. M., Kriegel, H.-P. e Sander, P. (1999). Optics: Ordering points to identify the clustering structure. In: *Proceedings of ACM SIGMOD International Conference on Management of Data (SIGMOD'1999)*, p. 49–60. Citado em: pages 251

Azuaje, F. (2002). A cluster validity framework for genome expression data. *Bioinformatics*, 18(2):319–320. Citado em: pages 291

Babcock, B., Babu, S., Datar, M., Motwani, R. e Widom, J. (2002). Models and issues in data stream systems. In: Kolaitis, P. G. (Ed.) *Proceedings of the 21nd Symposium on Principles of Database Systems*, p. 1–16. ACM Press. Citado em: pages 309

Barabasi, A.-L. e Albert, R. (1999). Emergence of scaling in random networks. *Science*, 286(5439):509–512. Citado em: pages 378, 395, 396

Barabasi, A. L. e Bonabeau, E. (2003). Scale-free networks. *Scientific American*, 288:60–69. Citado em: pages 378, 396

Baranauskas, J. A. e Monard, M. C. (2000). Reviewing some machine learning concepts and methods. Relatório Técnico 102, Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos. Disponível em: `ftp://ftp.icmc.usp.br/pub/BIBLIOTECA/rel_tec/RT_102.ps.zip`. Citado em: pages 195

Barbara, D. (2000). An introduction to cluster analysis for data mining. `http://www-users.cs.umn.edu/~han/dmclass/cluster\_survey\_10\_02\_00.pdf` [Acessado em 12/Nov./2003]. Citado em: pages 22, 229, 235, 251, 252, 255

Barnard, G. A. (1963). New methods of quality control. *Journal Royal Statistical Society - Series A*, p. 126–255. Citado em: pages 167

Barnett, V. e Lewis, T. (1994). *Outliers in statistical data*. John Wiley and Sons. Citado em: pages 54

Basseville, M. e Nikiforov, I. (1987). *Detection of abrupt changes: theory and applications*. Prentice-Hall Inc. Citado em: pages 316

Battiti, R. (1991). First and second-order methods for learning: between steepest descent and Newton's method. Relatório técnico, University of Trento. Citado em: pages 143

Bauer, E. e Kohavi, R. (1999). An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36:105–139. Citado em: pages 189

Bay, S. D. (1998). Combining nearest neighbor classifiers through multiple feature subsets. In: Shavlik, J. (Ed.) *Proceedings of the 15th International Conference -ICML'98*. Morgan Kaufmann. Citado em: pages 180

Ben-Hur, A., Horn, D., Siegelmann, H. e Vapnik, V. (2001). Support vector clustering. *Journal of Machine Learning Research*, 2:125–137. Citado em: pages 250

Ben-Hur, A., Horn, D., Siegelmann, H. T. e Vapnik, V. N. (2000). A support vector clustering method. In: *Proceedings of the International Conference on Pattern Recognition (ICPR'00)*, vol. 2, p. 724–727. Citado em: pages 129

Bensusan, H. (1998). God doesn't always shave with occam's razor - learning when and how to prune. In: *Proceedigs of the 10th European Conference on Machine Learning*, p. 119–124. Springer. Citado em: pages 324

Bensusan, H. e Giraud-Carrier, C. (2000a). Casa batlo is in passeig de gracia or landmarking the expertise space. In: *Proceedings of the ECML'2000 Workshop on Meta-Learning: building automatic advice strategies for model selection and method combination*, p. 29–47. ECML'2000. Citado em: pages 325

Bensusan, H. e Giraud-Carrier, C. (2000b). Discovering task neighbourhoods through landmark learning performances. In: Zighed, D., Komorowski, J. e Zytkow, J. (Ed.) *Proceedings of the 4th European Conference on Principles and Practice of Knowledge Discovery in Databases*, p. 325–331. Springer. Citado em: pages 326

Bensusan, H., Giraud-Carrier, C. e Kennedy, C. (2000). A higher-order approach to meta-learning. In: *Proceedings of the ECML'2000 Workshop on Meta-Learning: building automatic advice strategies for model selection and method combination*, p. 109–117. ECML'2000. Citado em: pages 324, 325

Bentley, J. (1975). Multidimensional binary search tree used for associative searching. *Machine Learning*, 18(9):509–517. Citado em: pages 83

Berkhin, P. (2002). Survey of clustering data mining techniques. Relatório técnico, Accrue Software, San Jose, CA. `http://www.accrue.com/products/rp_cluster_review.pdf` [Acessado em 05/Fev./2004]. Citado em: pages 255, 256

Beyer, K., Goldstein, J., Ramakrishnan, R. e Shaft, U. (1999). When is "nearest neighbor" meaningful? In: *International Conference on Database Theory*, p. 217–235. ACM. Citado em: pages 82

Bezdek, J. C. e Pal, N. R. (1998). Some new indexes of cluster validity. *IEEE Trans. Syst., Man, Cybernetics - Part B: Cybernetics*, 28(3):301–315. Citado em: pages 291

Bifet, A. e Gavaldà, R. (2007). Learning from time-changing data with adaptive windowing. In: *Proceedings SIAM International Conference on Data Mining*, p. 443–448, Minneapolis, USA. SIAM. Citado em: pages 317

Blake, C., Keogh, E. e Merz, C. (1999). UCI repository of Machine Learning Databases. Citado em: pages 127

Blanco, R., Inza, I., Merino, I., Quiroga, M. e Larrañaga, P. (2005). Feature selection in Bayesian classifiers for the prognosis of survival of cirrhotic patients treated with tips. *Internat.Journal of Biomed. Informatics*, 38(5). Citado em: pages 101

Blockeel, H., Bruynooghe, M., Dzeroski, S., Ramon, J. e Struyf, J. (2002). Hierarchical multi-classification. In: *Proceedings of the ACM SIGKDD 2002 Workshop on Multi-Relational Data Mining (MRDM 2002)*, p. 21–35. Citado em: pages 357, 358

Blondel, V., Guillaume, J.-L., Lambiotte, R. e Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008. Citado em: pages 392

Blum, A. L. e Langley, P. (1997). Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97:245–271. Citado em: pages 67

Blum, C. (2005). Ant colony optimization: introduction and recent trends. *Physics of Life Reviews*, 2:353–373. Citado em: pages 363

Blum, C. e Roli, A. (2003). Metaheuristics in combinatorial optimization: overview and conceptual comparison. *ACM Comput. Surv.*, 35(3):268–308. Citado em: pages 361

Bonacich, P. (1987). Power and centrality: A family of measures. *The American Journal of Sociology*, 92(5):1170–1182. Citado em: pages 377, 380

Boser, B. E., Guyon, I. L. e Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In: *Proceedings of the 5th Annual Workshop on Computational Learning Theory*, p. 144–152, Pittsburg, Pennsylvania, US. Citado em: pages 153

Boser, R. C. e Ray-Chaudhuri, D. K. (1960). On a class of error-correcting binary group codes. *Information and Control*, 3:68–79. Citado em: pages 334

Boutell, M. R., Luo, J., Shen, X. e Brown, C. M. (2004). Learning multi-label scene classification. *Pattern Recognition*, 37(9):1757–1771. Citado em: pages 341, 349

Boutin, F. e Hascoët, M. (2004). Cluster validity indices for graph partitioning. In: *Eighth International Conference on Information Visualisation (IV'2004)*, p. 376–381, London, England. Citado em: pages 301

Braga, A. P., Carvalho, A. C. P. L. F. e Ludermir, T. B. (2007). *Redes neurais artificiais: teoria e aplicações*. LTC. Citado em: pages 129, 132, 148

Bratko, I. (1984). *Prolog, Programming for Artificial Intelligence*. Addison-Wesley Publishing Company. Citado em: pages 113

Brazdil, P., Giraud-Carrier, C., Soares, C. e Vilalta, R. (2009). *Metalearning: applications to data mining*. Cognitive Technologies. Springer. Citado em: pages 307, 321, 322, 324, 325, 328

Breiman, L. (1996a). Bagging predictors. *Machine Learning*, 24:123–140. Citado em: pages 175, 176

Breiman, L. (1996b). Bias, variance, and arcing classifiers. Relatório Técnico 460, Statistics Department, University of California. Citado em: pages 117, 207

Breiman, L. (1996c). Stacked regressions. *Machine Learning*, 24:49–64. Citado em: pages 182, 183, 184

Breiman, L. (1998). Arcing classifiers. *The Annals of Statistics*, 26(3):801–849. Citado em: pages 189, 207

Breiman, L. (2001). Random forests. *Machine Learning*, 45:5–32. Citado em: pages 180

Breiman, L., Friedman, J., Olshen, R. e Stone, C. (1984). *Classification and Regression Trees*. Wadsworth International Group., USA. Citado em: pages 103, 111, 113, 114, 116, 121, 125, 329

Breunig, M. M., Kriegel, H.-P., Ng, R. T. e Sander, J. (2000). Lof: identifying density-based local outliers. In: *ACM Sigmod Record*, vol. 29, p. 93–104. ACM. Citado em: pages 56

Brin, S. e Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117. Citado em: pages 384

Brin, S. e Page, L. (2010). Node centrality in weighted networks: Generalizing degree and shortest paths. *Social Networks*, 32(3):245–251. Citado em: pages 377

Broder, A., Kumar, R., Maghoul, F., Raghavan, P., Rajagopalan, S., Stata, R., Tomkins, A. e Wiener, J. (2000). Graph structure in the web. *Computer Networks*, 33(1–6):309–320. Citado em: pages 373

Brodley, C. (1993). Addressing the selective superiority problem: Automatic algorithm / model class selection problem. In: Utgoff, P. (Ed.) *Machine Learning, Proceedings of the 10th International Conference*. Morgan Kaufmann. Citado em: pages 188

Brodley, C. (1995). Recursive automatic bias selection for classifier construction. Machine Learning, 20. Citado em: pages 188

Buntine, W. (1990). *A Theory of Learning Classification Rules*. Tese de Doutoramento, University of Sydney. Citado em: pages 127

Buntine, W. e Niblett, T. (1992). A further comparison of splitting rules for decision-tree induction. *Machine Learning*, 8:75–85. Citado em: pages 111

Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Knowledge Discovery and Data Mining*, 2(2):1–43. Citado em: pages 150, 151, 156, 164

Calinski, R. e Harabasz, J. (1974). A dendrite method for cluster analysis. *Communications in Statistics*, 3:1–27. Citado em: pages 293

Callaghan, L., Mishra, N., Meyerson, A., Guha, S. e Motwani, R. (2002). Streaming-data algorithms for high-quality clustering. In: *Proceedings of IEEE International Conference on Data Engineering*, p. 685. Citado em: pages 310, 313

Campbell, C. (2000). An introduction to kernel methods. In: Howlett, R. J. e Jain, L. C. (Ed.) *Radial Basis Function Networks: Design and Applications*, p. 155–192, Berlin. Springer Verlag. Citado em: pages 154

Campello, R. J. G. B. (2007). A fuzzy extension of the rand index and other related indexes for clustering and classification assessment. *Pattern Recogn. Lett.*, 28:833–841. Citado em: pages 303

Carvalho, A. C. P. L. F., Braga, A. P. e Ludermir, T. B. (2003). Computação evolutiva. In: Rezende, S. O. (Ed.) *Sistemas Inteligentes: Fundamentos e Aplicações*, Capítulo 9, p. 225–248. Editora Manole Ltda. Citado em: pages 367, 369

Castillo, G. e Gama, J. (2005). Bias management of Bayesian networks classifiers. In: *Proceedings of the 8th International Conference of Discovery Science*, vol. 3735 of *LNAI*, p. 70–83. Springer Verlag. Citado em: pages 101

Castro, L. N. (2006). *Fundamentals of Natural Computing: Basic Concepts, Algorithms, and Aplications*. Chapman & Hall/CRC. Citado em: pages 362, 365

Castro, L. N. (2007). Fundamentals of natural computing: an overview. *Physics of Life Reviews*, 4(1):1–36. Citado em: pages 361

Catlett, J. (1991). On changing continuous attributes into ordered discrete attributes. In: Kodratoff, Y. (Ed.) *European Working Session on Learning -EWSL91*. LNAI 482 Springer Verlag. Citado em: pages 117

Cestnik, B., Kononenko, I. e Bratko, I. (1987). Assistant 86: a knowledge-elicitation tool for sophisticated users. In: Bratko, I. e Lavrac, N. (Ed.) *European Working Session on Learning -EWSL87*. Sigma Press, Wilmslow, England. Citado em: pages 103

Chan, P. e Stolfo, S. (1995a). A comparative evaluation of voting and meta-learning on partitioned data. In: Prieditis, A. e Russel, S. (Ed.) *Machine Learning Proc. of 12th International Conference*. Morgan Kaufmann. Citado em: pages 187

Chan, P. e Stolfo, S. (1995b). Learning arbiter and combiner trees from partitioned data for scaling machine learning. In: Fayyad, U. M. e Uthurusamy, R. (Ed.) *Proc. of the First Intern. Conference on Knowledge Discovery and Data Mining*. AAAI Press. Citado em: pages 188

Chan, P. e Stolfo, S. (1997). On the accuracy of meta-learning for scalable data mining. *Journal of Intelligent Information systems*, 8:5–28. Citado em: pages 188

Chapelle, O., Vapnik, V., Bousquet, O. e Mukherjee, S. (2002). Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1-3):131–159. Citado em: pages 164

Chen, L. e Pung, H. K. (2008). Convergence analysis of convex incremental neural networks. *Annals of Mathematics and Artificial Intelligence*, 52:67–80. Citado em: pages 316

Cheng, J. e Greiner, R. (1999). Comparing Bayesian network classifiers. In: *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence*, p. 101–108. Morgan Kaufmann Publishers Inc. Citado em: pages 101

Cheng, Y. e Church, G. (2000). Biclustering of expression data. In: *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology (ISMB'2000)*, p. 93–103. Citado em: pages 250

Chi, Y., Wang, H., Yu, P. S. e Muntz, R. R. (2004). Moment: Maintaining closed frequent itemsets over a stream sliding window. In: *Proceedings of the IEEE International Conference on Data Mining*, p. 59–66, Brighton, UK. Citado em: pages 221

Chiang, J.-H. e Hao, P.-Y. (2003). A new kernel-based fuzzy clustering approach: support vector clustering with cell growing. *IEEE Transactions on Fuzzy Systems*, 11(4):518– 527. Citado em: pages 250

Clare, A. e King, R. D. (2001). Knowledge discovery in multi-label phenotype data. In: *5th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD2001), volume 2168 of Lecture Notes in Arti Intelligence*, p. 42–53. Springer. Citado em: pages 341, 346

Clare, A. e King, R. D. (2003). Predicting gene function in saccharomyces cerevisiae. *Bioinformatics*, 19(2):42–53. Citado em: pages 351, 357

Clark, P. e Niblett, T. (1989). The CN2 induction algorithm. *Machine Learning*, 3:261–283. Citado em: pages 120, 121, 122

Clauset, A., Newman, M. E. J. e Moore, C. (2004). Finding community structure in very large networks. *Physical Review E*, 70(6):066111. Citado em: pages 392

Coelho, A. L., Fernandes, E. e Faceli, K. (2010). Inducing multi-objective clustering ensembles with genetic programming. *Neurocomputing*, In Press, Corrected Proof. Citado em: pages 263, 280

Cohen, W. (1995). Fast effective rule induction. In: Prieditis, A. e Russel, S. (Ed.) *Machine Learning, Proceedings of the 12th International Conference*. Morgan Kaufmann. Citado em: pages 120

Corne, D. W., Jerram, N. R., Knowles, J. D. e Oates, M. J. (2001). PESA-II: Region-based selection in evolutionary multiobjective optimization. In: *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, p. 283–290, San Francisco, California, USA. Morgan Kaufmann. Citado em: pages 277

Cortes, C. e Vapnik, V. N. (1995). Support vector networks. *Machine Learning*, 20(3):273–296. Citado em: pages 153

Costa, E. P., Lorena, A. C., Carvalho, A. C. P. L. F. e Freitas, A. A. (2007). A review of performance evaluation measures for hierarchical classifiers. *AAAI07 - II Workshop on Evalution for Machine Learning - 22nd Conference on Artificial Intelligence*, p. 1–6. Citado em: pages 358

Costa, L., Jr., O. N. O., Travieso, G., Rodrigues, F. A., Boas, P. R. V., Antiqueira, L., Viana, M. P. e Rocha, L. E. C. d. (2011). Analyzing and modeling real-world phenomena with complex networks: A survey of applications. *Advances in Physics*, 60(3):329–412. Citado em: pages 382

Craven, M. e Shavlik, J. (1997). Using neural networks for data mining. *Future Generation Computer Systems*, 13:211–229. Citado em: pages 316

Craven, M. e Shavlik, J. W. (1994). Using sampling and queries to extract rules from trained neural networks. In: *ICML*, p. 37–45. Citado em: pages 149

Cristianini, N. e Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press. Citado em: pages 129, 149, 157, 164, 346

Cybenko, G. (1989). Approximation by superpositions of a sigmoid function. *Mathematics of Control, Signals and Systems*, 2:303–314. Citado em: pages 138

Dasgupta, K., Singh, R., Viswanathan, B., Chakraborty, D., Mukherjea, S., Nanavati, A. A. e Joshi, A. (2008). Social ties and their relevance to churn in mobile telecom networks. In: *Proceedings of the 11th International Conference on Extending Database Technology: Advances in Database Technology*, p. 668–677, New York, NY, USA. ACM. Citado em: pages 372

de Carvalho, A. e Freitas, A. (2009). A tutorial on multi-label classification techniques. vol. Foundations of Computational Intelligence Vol. 5 of *Studies in Computational Intelligence 205*, p. 177–195. Springer. Citado em: pages 342, 343, 344, 345, 346, 347, 349

de Comite, F., Gilleron, R. e Tommasi, M. (2003). Learning multi-label alternating decision trees from texts and data. In: *International Conference on Machine Learning and Data Mining*, number 2734 In: LNAI, p. 35–49. SV. Citado em: pages 346

de Souto, M. C. P., Prudêncio, R. B. C., Soares, R. G. F., de Araujo, D. S. A., Costa, I. G., Ludermir, T. B. e Schliep, A. (2008). Ranking and selecting clustering algorithms using a meta-learning approach. In: *IJCNN*, p. 3729–3735. Citado em: pages 328

de Souza, B. F. (2010). *Meta-aprendizagem aplicada à classificação de dados de expressão gênica*. Tese de Doutoramento, Universidade de São Paulo. Citado em: pages 322, 323, 324, 327

de Souza, B. F., Carvalho, A. C. e Soares, C. (2010). A comprehensive comparison of ml algorithms for gene expression data classification. In: *2010 International Joint Conference on Neural Networks*, p. 98–105. IEEE. Citado em: pages 327

Demsár, J. (2006). Statistical comparisons of classifiers over multiple datasets. *Journal of Machine Learning Research*, 7:1–30. Citado em: pages 203, 204, 205, 206

Devore, J. L. (2006). *Probabilidade e Estatística: para Engenharia e Ciências*. Thompson. Citado em: pages 195, 202

Diestel, R. (2005). *Graph Theory*. Springer, 3rd ed. Citado em: pages 374

Dietterich, T. (1997). Machine learning research: four current directions. *AI Magazine*, 18(4):97–136. Citado em: pages 167, 168, 177

Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1924. Citado em: pages 71, 203

Dietterich, T. G. e Bariki, G. (1995). Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286. Citado em: pages 331, 333, 334, 338

Dom, B. E. (2002). An information-theoretic external cluster-validity measure. In: *Proceedings of the 18th Annual Conference on Uncertainty in Artificial Intelligence (UAI-2002)*, p. 137–145, San Francisco, CA. Morgan Kaufmann Publishers. Citado em: pages 301

Domingos, P. (1996). Unifying instance-based and rule-based induction. *Machine Learning*, 24:141–168. Citado em: pages 120, 189

Domingos, P. (1997a). Bayesian model averaging in rule induction. In: Smyth, P. e Madigan, D. (Ed.) *Preliminary papers of the Sixth International Workshop on Artificial Intelligence and Statistics*. Citado em: pages 171

Domingos, P. (1997b). *A Unified Approach to Concept Learning*. Tese de Doutoramento, University of California, Irvine. Citado em: pages 189

Domingos, P. (1998). Knowledge discovery via multiple models. *Intelligent Data Analysis*, 24(2). Citado em: pages 188

Domingos, P. e Hulten, G. (2000). Mining High-Speed Data Streams. In: Parsa, I., Ramakrishnan, R. e Stolfo, S. (Ed.) *Proceedings of the ACM Sixth International Conference on Knowledge Discovery and Data Mining*, p. 71–80. ACM Press. Citado em: pages 307, 310, 311

Domingos, P. e Hulten, G. (2001). A general method for scaling up machine learning algorithms and its application to clustering. In: *Proceedings of the Eighteenth International Conference on Machine Learning*, p. 106–113. Citado em: pages 310

Domingos, P. e Pazzani, M. (1997). On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 29:103–129. Citado em: pages 94, 96, 100

Domingos, P. e Richardson, M. (2001). Mining the network value of customers. In: *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, p. 57–66, New York, NY, USA. ACM. Citado em: pages 372

Dopazo, J., Zanders, E., Dragoni, I., Amphlett, G. e Falciani, F. (2001). Methods and approaches in the analysis of gene expression data. *Journal of Immunological Methods*, 250(1-2):93 – 112. Citado em: pages 66

Dorigo, M., Birattari, M. e Stützle, T. (2006). Ant colony optimization - artificial ants as a computational intelligence technique. *IEEE Comput. Intell. Mag*, 1:28–39. Citado em: pages 361, 362

Dorigo, M. e Di-Caro, G. (1999). The ant colony optimization metaheuristic. In: Corne, D., Dorigo, M. e Glover, F. (Ed.) *New Ideas in Optimization*, p. 11–32. McGraw-Hill, London, UK. Citado em: pages 363

Dougherty, J., Kohavi, R. e Sahami, M. (1995). Supervised and unsupervised discretization of continuous features. In: Prieditis, A. e Russel, S. (Ed.) *Machine Learning Proc. of 12th International Conference*. Morgan Kaufmann. Citado em: pages 94

Duan, K., Keerthi, S. S. e Poo, A. N. (2003). Evaluation of simple performance measures for tuning SVM hyperparameters. *Neurocomputing*, 51:41–59. Citado em: pages 164

Dubes, R. e Jain, A. (1976). Clustering techniques: The user's dilemma. *Pattern Recognition*, 8:247–260. Citado em: pages 246

Duda, R. O., Hart, P. E. e Stork, D. G. (2001). *Pattern Classification*. Wiley-Interscience, 2. ed. Citado em: pages 74, 81, 89, 92, 250, 255

Dudoit, S. e Fridlyand, J. (2002). A prediction-based resampling method for estimating the number of clusters in a dataset. *Genome Biology*, 3(7):research0036.1–0036.21. Citado em: pages 297

Dunn, O. J. (1961). Multiple comparisons among means. *Journal of the American Statistical Association*, 56(293):52–64. Citado em: pages 206

Easley, D. e Kleinberg, J. (2010). *Networks, Crowds and Markets: Reasoning about a Highly Connected World*. Cambridge University Press, Cambridge. Citado em: pages 385, 390

Eisner, R., Poulin, B., Szafron, D., Lu, P. e Greiner, R. (2005). Improving protein function prediction using the hierarchical structure of the gene ontology. In: *Proceedings of the IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, p. 1–10. Citado em: pages 356

Elisseeff, A. e Weston, J. (2001a). Kernel methods for multi-labelled classification and categorical regression problems. Relatório técnico, Biowulf Technologies. Citado em: pages 343

Elisseeff, A. E. e Weston, J. (2001b). A kernel method for multi-labelled classification. In: *In Advances in Neural Information Processing Systems 14*, p. 681–687. MIT Press. Citado em: pages 341

Erdos, P. e Renyi, A. (1960). On the evolution of random graphs. *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, 5:17–61. Citado em: pages 394, 395

Ertöz, L., Steinbach, M. e Kumar, V. (2002). A new shared nearest neighbor clustering algorithm and its applications. In: *Proceedings of the Workshop on Clustering High Dimensional Data and its Applications, 2nd SIAM International Conference on Data Mining (SDM'2002)*, p. 105–115. Citado em: pages 250

Esposito, F., Malerba, D. e Semeraro, G. (1993). Decision tree prunning as a search in the state space. In: Brazdil, P. (Ed.) *Machine Learning: ECML-93*. LNAI 667, Springer Verlag. Citado em: pages 113, 115

Esposito, F., Malerba, D. e Semeraro, G. (1997). A comparative analysis of methods for pruning decision trees. *Transactions on Pattern Analysis and Machine Intelligence*, 19(5). Citado em: pages 111, 113, 115

Ester, M., Kriegel, H.-P., Sander, J. e Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In: *Proceedings of 2nd Int. Conf. on Knowledge Discovery and Data Mining (KDD'1996)*, p. 226–231. Citado em: pages 256

Estivill-Castro, V. (2002). Why so many clustering algorithms - a position paper. *SIGKDD Explorations*, 4(1):65–75. Citado em: pages 231, 233, 244, 245

Faceli, K., Souto, M. C. P., de Araújo, D. S. A. e Carvalho, A. C. F. L. F. (2009). Multi-objective clustering ensemble for gene expression data analysis. *Neurocomputing*, 72(13-15):2763–2774. Citado em: pages 263, 280

Fagni, T. e Sebastiani, F. (2007). On the selection of negative examples for hierarchical text categorization. In: *Proceedings of the 3rd Language Technology Conference*, p. 24–28. Citado em: pages 356

Fahlman, S. E. (1988). An empirical study of learning speed in backpropagation networks. Relatório técnico, Carnegie Mellow University. Citado em: pages 143

Falkman, G. (2002). Adaptation using interactive estimations. In: Craw, S. e Preece, A. (Ed.) *6th European Conference on Case-Based Reasoning, Aberdeen, Scotland, Uk*, p. 88–102. Springer Verlag. Citado em: pages 87

Fawcett, T. (2005). An introduction to ROC analysis. *Pattern Recognition Letters*, p. 861–874. Citado em: pages 199

Fayyad, U. e Irani, K. (1993). Multi-interval discretization of continuous-valued attributes for classification learning. In: *13th International Joint Conference of Artificial Intelligence*, p. 1022–1029. Morgan Kaufman. Citado em: pages 117

Fayyad, U. M. e Irani, K. B. (1992). On the handling of continuous-valued attributes in decision tree generation. *Machine Learning*, 8:87–102. Citado em: pages 111

Feelders, A. e Verkooijen, W. (1996). On the statistical comparison of inductive learning methods. In: Fisher, D. e Lenz, H.-J. (Ed.) *Learning from Data: Artificial Intelligence and Statistics V*, p. 272–279. Springer Verlag, New York, NY. Citado em: pages 205

Fern, X. Z. e Brodley, C. E. (2004). Solving cluster ensemble problems by bipartite graph partitioning. In: *Proceedings of the Twenty First International Conference on Machine Learning (ICML'2004)*, p. 36, New York, NY, USA. ACM Press. Citado em: pages 263, 265, 266, 267, 269, 270, 275

Ferrer-Troyano, F., Aguilar-Ruiz, J. e Riquelme, J. (2005). Incremental rule learning and border examples selection from numerical data streams. *Journal of Universal Computer Science*, 11(8):1426–1439. Citado em: pages 310

Filho, I. G. C. (2003). Comparative analysis of clustering methods for gene expression data. Dissertação de Mestrado, Centro de Informática, Universidade Federal de Pernambuco, Recife. Citado em: pages 288, 303

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals Eugen.*, 7:179–188. Citado em: pages 28

Fix, E. (1951). Discriminatory analysis: Nonparametric discrimination: Consistency properties. Relatório Técnico Project 21-49-004, Report Number 4, USAF School of Aviation Medicine, Randolf Field, Texas. Citado em: pages 45

Fortunato, S. (2010). Community detection in graphs. *Physics Report*, 486(3-5):75–174. Citado em: pages 389, 392, 394

Frank, A. e Asuncion, A. (2010). UCI Machine Learning Repository. http://archive.ics.uci.edu/ml. Citado em: pages 94

Frank, E. e Kramer, S. (2004). Ensembles of nested dichotomies for multi-class problems. In: *Proceedings of the 21st International Conference on Machine Learning*, p. 305–312. Citado em: pages 337

Frank, E., Wang, Y., Inglis, S., Holmes, G. e Witten, I. H. (1998). Using model trees for classification. *Machine Learning*, 32(1):63–76. Citado em: pages 126

Frank, E. e Witten, I. H. (1998). Generating accurate rule sets without global optimization. In: Shavlik, J. (Ed.) *Proceedings of the 15th International Conference -ICML'98*, p. 144–151. Morgan Kaufmann. Citado em: pages 120

Fred, A. e Jain, A. (2002). Evidence accumulation clustering based on the k-means algorithm. In: *Proceedings of Structural and Syntactic Pattern Recognition (SSPR'2002)*, p. 442–451, Windsor, Canada. Citado em: pages 263, 265, 266, 267, 269

Fred, A. e Jain, A. K. (2003). Robust data clustering. In: *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, (CVPR'2003)*, vol. II, Madison - Wisconsin, USA. Citado em: pages 263, 265, 266, 267, 269

Fred, A. L. N. (2001). Finding consistent clusters in data partitions. In: Kittler, J. e Roli, F. (Ed.) *Second International Workshop on Multiple Classifier Systems (MCS'2001)*, vol. 2096 of *Lecture Notes in Computer Science*, p. 309–318, Cambridge, UK. Citado em: pages 263, 265, 266, 268, 269

Freeman, L. C. (1979). Centrality in social networks: Conceptual clarification. *Social Networks*, 1(3):215–239. Citado em: pages 377

Freitas, A. A. e de Carvalho, A. C. P. L. F. (2007). A tutorial on hierarchical classification with applications in bioinformatics. *Intelligent Information Technologies: Concepts, Methodologies, Tools and Applications*, 1:114–145. Citado em: pages 351

Freund, Y. e Mason, L. (1999). The alternating decision tree learning algorithm. In: *In Machine Learning: Proceedings of the Sixteenth International Conference*, p. 124–133. Morgan Kaufmann. Citado em: pages 346

Freund, Y. e Schapire, R. (1999). A short introduction to boosting. *Japanese Society for Artificial Intelligence*, 14(5):771–780. Citado em: pages 346

Freund, Y. e Schapire, R. E. (1995). A decision-theoretic generalization of on-line learning and an application to boosting. In: *European Conference on Computational Learning Theory*, p. 23–37. Citado em: pages 346, 347

Freund, Y. e Schapire, R. E. (1996). Experiments with a new boosting algorithm. In: Saitta, L. (Ed.) *Machine Learning, Proc. of the 13th International Conference*. Morgan Kaufmann. Citado em: pages 178, 189

Friedman, J. (1999). Greedy Function Approximation: a gradient boosting machine. Relatório técnico, Statistics Department, Stanford University. Citado em: pages 116

Friedman, J. H., Kohavi, R. e Yun, Y. (1996). Lazy decision trees. In: AAAI (Ed.) *Thirteenth National Conference on Artificial Intelligence*. MIT Press. Citado em: pages 117

Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32:675–701. Citado em: pages 205

Friedman, N., Geiger, D. e Goldszmidt, M. (1997). Bayesian network classifiers. *Machine Learning*, 29(2-3):131–163. Citado em: pages 100, 101

Fritzke, B. (1994). Growing cell structures - a self-organizing network for unsupervised and supervised learning. *Neural Networks*, 7(9):1441–1460. Citado em: pages 258

Frossyniotis, D., Pertselakis, M. e Stafylopatis, A. (2002). A multi-clustering fusion algorithm. In: Vlahavas, I. P. e Spyropoulos, C. D. (Ed.) *Methods and Applications of Artificial Intelligence, Proceedings of the 2nd Hellenic Conference on AI (SETN'2002)*, vol. 2308 of *Lecture Notes in Computer Science*, p. 225–236, Thessaloniki, Greece. Springer Verlag. Citado em: pages 266, 268, 269

Fu, L. (1994). Rule generation from neural networks. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(8):1114–1124. Citado em: pages 149

Fu, X., Ong, C., Keerthi, S., Hung, G. G. e Goh, L. (2004). Extracting the knowledge embedded in support vector machines. In: *Proceedings IEEE International Joint Conference on Neural Networks*, vol. 1, p. 296. Citado em: pages 164

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202. Citado em: pages 145

Furnkranz, J. (2002). Round Robin classification. *Journal of Machine Learning Research*, 2:721–747. Citado em: pages 330, 331

Furnkranz, J. (2003). Round Robin ensembles. *Intelligent Data Analysis*, 7(5):385–404. Citado em: pages 331

Gama, J. (2000). A linear-Bayes classifier. In: Monard, C. e Sichman, J. (Ed.) *Advances on Artificial Intelligence -SBIA2000*, p. 269–279. LNAI 1952 Springer Verlag. Citado em: pages 98

Gama, J. (2010). *Knowledge Discovery from Data Streams*. Chapman and Hall / CRC Data Mining and Knowledge Discovery Series. CRC Press. Citado em: pages 307, 318

Gama, J. e Brazdil, P. (2000). Cascade generalization. *Machine Learning*, 41:315–343. Citado em: pages 184

Gama, J. e Kosina, P. (2011). Learning decision rules from data streams. In: *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, p. 1255–1260. IJCAI/AAAI. Citado em: pages 310

Gama, J., Medas, P., Castillo, G. e Rodrigues, P. (2004). Learning with drift detection. In: Bazzan, A. L. C. e Labidi, S. (Ed.) *Advances in Artificial Intelligence - SBIA 2004*, vol. 3171 of *Lecture Notes in Computer Science*, p. 286–295. Springer Verlag. Citado em: pages 317

Gama, J., Rocha, R. e Medas, P. (2003). Accurate decision trees for mining high-speed data streams. In: *Proceedings of the ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, p. 523–528. ACM Press. Citado em: pages 307, 310

Gama, J. M. P. (1999). *Combining Classification Algorithms*. Tese de Doutoramento, Faculdade de Ciências da Universidade do Porto. Citado em: pages 264

García, S. e Herrera, F. (2008). An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of Machine Learning Research*, 9:2677–2694. Citado em: pages 203, 206

Geman, S., Bienenstock, E. e Doursat, R. (1992). Neural networks and the bias/variance dilema. In: *Neural Computation*, vol. 4, p. 1–58. Citado em: pages 207

Getz, G., Gal, H., Kela, I., Notterman, D. A. e Domany, E. (2003). Coupled two-way clustering analysis of breast cancer and colon cancer gene expression data. *Bioinformatics*, 19:1079–1089. Citado em: pages 250

Geurts, P. (2000). *Contributions to Decision Tree Induction:bias/variance tradeoff and time series classification*. Tese de Doutoramento, University of Liege. Citado em: pages 181

Geurts, P. (2001). Dual perturb and combine algorithm. In: *Proc. of the Eighth International Workshop on Artificial Intelligence and Statistics*, p. 196–201. Springer Verlag. Citado em: pages 181

Ghosh, J., Strehl, A. e Merugu, S. (2002). A consensus framework for integrating distributed clusterings under limited knowledge sharing. In: *Proceedings of NSF Workshop on Next Generation Data Mining*, p. 99–108. Citado em: pages 263, 265

Giannella, C., Han, J., Pei, J., Yan, X. e Yu, P. (2003). Mining frequent patterns in data streams at multiple time granularities. In: Kargupta, H., Joshi, A., Sivakumar, K. e Yesha, Y. (Ed.) *Next Generation Data Mining*. AAAI/MIT. Citado em: pages 311

Girvan, M. e Newman, M. E. J. (2002). Community structure in social and biological networks. *Proceedings of the National Academy of Sciences of the United States of America*, 99(12):7821–7826. Citado em: pages 388, 390, 391

Godbole, S. e Sarawagi, S. (2004). Discriminative methods for multi-labeled classification. In: *Advances in Knowledge Discovery and Data Mining*, p. 22–30. Citado em: pages 349

Gonçalves, T. e Quaresma, P. (2003). A preliminary approach to the multilabel classification problem of portuguese juridical documents. In: *EPIA*, p. 435–444. Citado em: pages 341

Goodfellow, I., Bengio, Y. e Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`. Citado em: pages 145

Gordon, A. (1999). *Classification*. Chapman & Hall/CRC. Citado em: pages 237, 238, 284, 286, 301

Gordon, A. D. (1996). *From Data to Knowledge: Theoretical and Practical Aspects of Classification, Data Analysis and Knowledge Organization*, Capítulo Null models in cluster validation, p. 32–44. Springer-Verlag. Citado em: pages 288

Granger, C. W. J. e Newbold, P. (1976). The use of r2 to determine the appropriate transformation of regression variables. *J. Econometrics*, 4:205–210. Citado em: pages 167

Granovetter, M. (1973). The strength of weak ties. *American Journal of Sociology*, 78(6):1360–1380. Citado em: pages 375

Granovetter, M. (1974). *Getting a Job: A Study of Contacts and Careers*. Harvard University Press, Cambridge, MA. Citado em: pages 375

Guha, S., Rastogi, R. e Shim, K. (1998). CURE: an efficient clustering algorithm for large databases. In: *Proceedings of ACM SIGMOD International Conference on Management of Data*, p. 73–84. Citado em: pages 251

Guha, S., Rastogi, R. e Shim, K. (2000). ROCK: A robust clustering algorithm for categorical attributes. *Information Systems*, 25(5):345–366. Citado em: pages 251

Guimera, R. e Amaral, L. A. N. (2005). Functional cartography of complex metabolic networks. *Nature*, 433(7028):895–900. Citado em: pages 392

Gupta, S., Anderson, R. M. e May, R. M. (1989). Networks of sexual contacts: Implications for the pattern of spread of hiv. *AIDS*, 3(12):807–817. Citado em: pages 397

Gutin, G., Punnen, A., Barvinok, A. e Edward Kh. Gimadi, A. I. S. (2002). *The Traveling Salesman Problem and Its Variations (Combinatorial Optimization)*. Springer. Citado em: pages 328

Guyon, I. e Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182. Citado em: pages 65

Hadjitodorov, S. T., Kuncheva, L. I. e Todorova, L. P. (2006). Moderate diversity for better cluster ensembles. *Information Fusion*, 7(3):264–275. Citado em: pages 265, 266

Hagan, M. e Menhaj, M. (1994). Training feedforward networks with the Marquardt algorithm. *IEEE Transactions on Neural Networks*, 5(6):989–993. Citado em: pages 143

Halkidi, M., Batistakis, Y. e Vazirgiannis, M. (2001). On clustering validation techniques. *Intelligent Information Systems Journal*, 17(2-3):107–145. Citado em: pages 244, 247, 249, 255, 284, 286, 297

Halkidi, M., Batistakis, Y. e Vazirgiannis, M. (2002a). Cluster validity methods: Part I. *SIGMOD Record*, 31(2):40–45. Citado em: pages 300

Halkidi, M., Batistakis, Y. e Vazirgiannis, M. (2002b). Cluster validity methods: Part II. *SIGMOD Record*, 31(3):19–27. Citado em: pages 290, 291

Halkidi, M. e Vazirgiannis, M. (2001). A data set oriented approach for clustering algorithm selection. In: *Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery*, p. 165–179. Citado em: pages 293

Hamming, R. (1950). Error-detecting and error-correcting codes. *Bell System Technical Journal*, 29:147–160. Citado em: pages 333

Han, J. e Kamber, M. (2000). *Data Mining: Concepts and Techniques*. Morgan Kaufmann. Citado em: pages 53, 213

Han, J., Pei, J. e Yin, Y. (2000). Mining frequent patterns without candidate generation. In: *SIGMOD '00: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, p. 1–12, New York, NY, USA. ACM Press. Citado em: pages 311

Han, J., Pei, J., Yin, Y. e Mao, R. (2004). Mining frequent patterns without candidate generation. *Data Mining and Knowledge Discovery*, 8:53–87. Citado em: pages 221, 222

Handl, J. e Knowles, J. (2004). Multiobjective clustering with automatic determination of the number of clusters. Technical Report TR-COMPSYSBIO-2004-02, UMIST, Manchester. Citado em: pages 263, 264, 276, 278, 279

Handl, J. e Knowles, J. (2005a). Exploiting the trade-off - the benefits of multiple objectives in data clustering. In: Coello, C. A. *et al.* (Ed.) *Proceedings of the Third International Conference on Evolutionary Multi-Criterion Optimization (EMO'2005)*, vol. 3410 of *Lecture Notes in Computer Science*, p. 547–560, Guanajuato, Mexico. Springer-Verlag. Citado em: pages 263, 276, 279

Handl, J. e Knowles, J. (2005b). Improvements to the scalability of multiobjective clustering. In: *IEEE Congress on Evolutionary Computation*, p. 438–445. IEEE Computer Society Press. Citado em: pages 263, 276, 278, 279

Handl, J. e Knowles, J. (2007). An evolutionary approach to multiobjective clustering. *IEEE Transactions on Evolutionary Computation*, 11(1):56–76. Citado em: pages 233, 263, 276, 293

Handl, J., Knowles, J. e Kell, D. (2005). Computational cluster validation in post-genomic data analysis. *Bioinformatics*, 21(15):3201–3212. Citado em: pages 231, 232, 284, 290

Hansen, L. e Salamon, P. (1990). Neural networks ensembles. *Transactions on Pattern Analysis and Machine Intelligence*, 12(10). Citado em: pages 167

Hartigan, J. (1975). *Clustering Algorithms*. Wiley. Citado em: pages 293

Hartigan, J. A. (1985). Statistical theory in clustering. *Journal of Classification*, 2(1):63–76. Citado em: pages 243

Hartuv, E. e Shamir, R. (2000). A clustering algorithm based on graph connectivity. *Information Processing Letters*, 76(200):175–181. Citado em: pages 258

Hastie, T. e Tibshirani, R. (1998). Classification by pairwise coupling. *The Annals of Statistics*, 26(2):451–471. Citado em: pages 331, 339

Hastie, T., Tibshirani, R. e Friedman, J. (2001). *The Elements of Statistical Learning*. Springer New York. Citado em: pages 100

Hawkins, D. M. (1980). *Identification of outliers*. Chapman and Hall.. Citado em: pages 55

Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation*. Prentice-Hall, Upper Saddle River, NJ, USA, 2. ed. Citado em: pages 129, 131, 132, 148, 159, 165

Hayward, R., Tickle, A. B. e Diederich, J. (1995). Extracting rules for grammar recognition from cascade-2 networks. In: *Learning for Natural Language Processing*, p. 48–60. Citado em: pages 149

He, Q. (1999). A review of clustering algorithms as applied in IR. Relatório Técnico UIUCLIS–1999/6+IRG, Information Retrieval Group, University of Illinois. Citado em: pages 237, 244

Hearst, M. A., Schölkopf, B., Dumais, S., Osuna, E. e Platt, J. (1998). Trends and controversies - support vector machines. *IEEE Intelligent Systems*, 13(4):18–28. Citado em: pages 158

Heath, D., Kasif, S. e Salzberg, S. (1996). Committees of decision trees. In: *Cognitive Technology: in Search of a Humane Interface*, p. 305–317. Elsevier Science. Citado em: pages 177

Hebb, D. O. (1949). *The Organization of Behavior*. Wiley. Citado em: pages 130

Herbrich, R. (2001). *Learning Kernel Classifiers: Theory and Algorithms*. MIT Press. Citado em: pages 160

Herrero, J., Valencia, A. e Dopazo, J. (2001). A hierarchical unsupervised growing neural network for clustering gene expression patterns. *Bioinformatics*, 17(2):126–136. Citado em: pages 258

Hinkley, D. (1970). Inference about the change point from cumulative sum-tests. *Biometrika*, 58:509–523. Citado em: pages 317

Hinneburg, A. e Keim, D. A. (1998). An efficient approach to clustering in large multimedia databases with noise. In: *Proceedings of 4rd Int. Conf. on Knowledge Discovery and Data Mining*, p. 58–65. AAAI Press. Citado em: pages 256, 257

Hinneburg, A. e Keim, D. A. (1999). Optimal grid-clustering: Towards breaking the curse of dimensionality in high-dimensional clustering. In: *Proceedings of the 25th International Conference on Very Large Databases*, p. 506–517. Citado em: pages 259

Hoeting, J. A., Madigan, D., Raftery, A. E. e Volinsky, C. T. (1999). Bayesian model averaging: A tutorial. *Statistical Science*, 14(1):382–401. Citado em: pages 167

Holland, J. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor. Citado em: pages 144

Holte, R. C. (1993). Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11:63–91. Citado em: pages 119

Hsu, C.-W., Chang, C.-C. e Lin, C.-J. (2003). A Practical Guide to Support Vector Classification. Relatório técnico, Department of Computer Science, National Taiwan University. Citado em: pages 161

Hsu, C.-W. e Lin, C.-J. (2002). A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425. Citado em: pages 164, 329

Hubert, L. J. e Arabie, P. (1985). Comparing partitions. *Journal of Classification*, 2:193–218. Citado em: pages 294, 302

Hulten, G., Spencer, L. e Domingos, P. (2001). Mining time-changing data streams. In: *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, p. 97–106, San Francisco, California. ACM Press. Citado em: pages 313

Ihaka, R. e Gentleman, R. (1996). R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5(3):299–314. Citado em: pages 103

Jain, A. e Dubes, R. (1988). *Algorithms for Clustering Data*. Prentice Hall. Citado em: pages 22, 229, 232, 236, 241, 242, 244, 246, 247, 249, 254, 284, 285, 286, 287, 288, 289, 293, 297, 300, 301

Jain, A., Murty, M. e Flynn, P. (1999). Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323. Citado em: pages 235, 236, 238, 249, 255

Jain, A. K. (2010). Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651 – 666. Award winning papers from the 19th International Conference on Pattern Recognition (ICPR), 19th International Conference in Pattern Recognition (ICPR). Citado em: pages 255, 261

Jain, A. K., Dubes, R. C. e Chen, C.-C. (1987). Bootstrap techniques for error estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(5):628–633. Citado em: pages 196

Jarmulak, J., Craw, S. e Rowe, R. (2001). Using case-base data to learn adaptation knowledge for design. In: Nebel, B. (Ed.) *17th International Joint Conference on Artificial Intelligence, Seattle, EUA*, p. 1011–1020. Morgan Kaufmann. Citado em: pages 87

Jiang, D., Tang, C. e Zhang, A. (2004). Cluster analysis for gene expression data: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1370–1386. Citado em: pages 231, 244, 293

Jin, R. e Agrawal, G. (2003). Efficient decision tree construction on streaming data. In: P.Domingos e Faloutsos, C. (Ed.) *Proceedings of the Ninth International Conference on Knowledge Discovery and Data Mining*. ACM Press. Citado em: pages 310

Joachims, T. (2002). *Learning to Classify Text using Support Vector Machines*. Kluwer/Springer. Citado em: pages 149

John, G. (1997). *Enhancements to the Data Mining Process*. Tese de Doutoramento, Stanford University. Citado em: pages 98

John, G., Kohavi, R. e Pfleger, K. (1994). Irrelevant features and the subset selection problem. In: Cohen, W. e Hirsh, H. (Ed.) *Machine Learning, Proceedings of the 11th International Conference*. Morgan Kaufmann. Citado em: pages 98

Kalousis, A. (2002). *Algorithm Selection via Meta-Learning*. Tese de doutorado, Centre Universitaire d'Informatique, Université de Genève, Genebra, Suíça. Citado em: pages 322, 324, 326

Kanda, J., de Carvalho, A. C. P. L. F., Hruschka, E. e Soares, C. (2010). Using meta-learning to classify traveling salesman problems. In: *XI Simpósio Brasileiro de Redes Neurais*, p. 73–78. Citado em: pages 328

Karalic, A. e Pirnat, V. (1991). Significance level based multiple tree classification. In: *Informatica*, vol. 5. Citado em: pages 341

Karnin, E. D. (1990). A simple procedure for pruning back-propagation trained neural networks. *IEEE Transactions on Neural Networks*, 1(2):239–242. Citado em: pages 144

Karypis, G., Han, E.-H. S. e Kumar, V. (1999). Chameleon: Hierarchical clustering using dynamic modeling. *Computer*, 32(8):68–75. Citado em: pages 251

Karypis, G. e Kumar, V. (1999). A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392. Citado em: pages 271, 272, 275

Kaufman, L. e Rousseeuw, P. J. (1990). *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons. Citado em: pages 255

Keerthi, S. S. e Lin, C.-J. (2003). Asymptotic behaviors of support vector machines with Gaussian kernel. *Neural Computation*, 15(7):1667–1689. Citado em: pages 161

Kellam, P., Liu, X., Martin, N. J., Orengo, C., Swift, S. e Tucker, A. (2001). Comparing, contrasting and combining clusters in viral gene expression data. In: *Proceedings of 6th Workshop on Intelligent Data Analysis in Medicine and Pharmacology*, p. 56–62. Citado em: pages 266, 267, 269

Kennedy, J. e Eberhart, R. (1995). Particle swarm optimization. In: *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4, p. 1942–1948, Perth, Australia. Citado em: pages 365

Kennedy, J. e Eberhart, R. (2001). *Swarm Intelligence*. Morgan Kaufmann Publishers. Citado em: pages 365

Kifer, D., Ben-David, S. e Gehrke, J. (2004). Detecting change in data streams. In: *VLDB 04: Proceedings of the 30th International Conference on Very Large Data Bases*, p. 180–191. Morgan Kaufmann Publishers Inc. Citado em: pages 317

Kijsirikul, B. e Ussivakul, N. (2002). Multiclass support vector machines using adaptive directed acyclic graph. In: *Proceedings of International Joint Conference on Neural Networks (IJCNN 2002)*, p. 980–985. Citado em: pages 335, 336

Kiritchenko, S., Matwin, S. e Famili, A. F. (2004). Hierarchical text categorization as a tool of associating genes with gene ontology codes. In: *Proceedings of the 2nd European Workshop on Data Mining and Text Mining for Bioinformatics*, p. 26–30. Citado em: pages 358

Kiritchenko, S., Matwin, S., Nock, R. e Famili, A. F. (2006). Learning and evaluation in the presence of class hierarchies: Application to text categorization. In: *Proceedings of the 19th Canadian Conference on Artificial Intelligence*, vol. 4013 of *Lecture Notes in Artificial Intelligence*, p. 395–406. Citado em: pages 357

Kittler, J. (1998). Combining classifiers: A theoretical framework. *Pattern Analysis and Applications*, 1(1). Citado em: pages 170, 171

Klautau, A., Jevtić, N. e Orlistky, A. (2003). On nearest-neighbor error-correcting output codes with application to all-pairs multiclass support vector machines. *Journal of Machine Learning Research*, 4:1–15. Citado em: pages 339

Kleinberg, J. (1999). Authorative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632. Citado em: pages 384

Kleinberg, J. (2002). An impossibility theorem for clustering. *Advances in Neural Information Processing Systems*, 15:446–453. Citado em: pages 233

Klinkenberg, R. (2004). Learning drifting concepts: Example selection vs. example weighting. *Intelligent Data Analysis*, 8(3):281–300. Citado em: pages 316

Knerr, S., Personnaz, L. e Dreyfus, G. (1992). Handwritten digit recognition by neural networks with single-layer training. *IEEE Transactions on Neural Networks*, 3(6):962–968. Citado em: pages 330

Koepf, C., Taylor, C. C. e Keller, J. (2000). Meta-analysis: From data characterisation for meta-learning to meta-regression. In: Brazdil, P. e Jorge, A. (Ed.) *Proceedings of the PKDD-00 Workshop on Data Mining, Decision Support,Meta-Learning and ILP: Forum for Practical Problem Presentation and Prospective Solutions*, Lyon, France. Citado em: pages 326

Kohavi, R. (1996). Scaling up the accuracy of naive-Bayes classifiers: a decision tree hybrid. In: Simoudis, E., Han, J. W. e Fayyad, U. (Ed.) *Proc. of the 2nd International Conference on Knowledge Discovery and Data Mining*, p. 202–207. AAAI Press, USA. Citado em: pages 97, 188

Kohavi, R. e Kunz, C. (1997). Option decision trees with majority votes. In: Fisher, D. (Ed.) *Machine Learning Proc. of 14th International Conference*. Morgan Kaufmann. Citado em: pages 117, 127

Kohavi, R., Sommerfield, D. e Dougherty, J. (1997). Data mining using MLC++, a machine learning library in C++. *International Journal of Artificial Intelligence Tools*, 6 Nr. 4. Citado em: pages 116

Kohavi, R. e Wolpert, D. (1996). Bias plus variance decomposition for zero-one loss functions. In: Saitta, L. (Ed.) *Proceedings of the 13th International Conference on Machine Learning*, p. 275–283. Morgan Kaufmann. Citado em: pages 207, 208

Kohonen, T. (2001). *Self-Organizing Maps*. Springer, Berlin. Citado em: pages 258, 259

Kong, E. B. e Dietterich, T. (1995). Error-correcting output coding correct bias and variance. In: Prieditis, A. e Russel, S. (Ed.) *Proceedings of the 12th International Conference on Machine Learning*. Morgan Kaufmann. Citado em: pages 207

Kononenko, I. (1991). Semi-naive Bayesian classifier. In: Kodratoff, Y. (Ed.) *European Working Session on Learning -EWSL91*. LNAI 482 Springer Verlag. Citado em: pages 97

Kontkanen, P., Myllymäki, P., Silander, T. e Tirri, H. (1999). On supervised selection of Bayesian networks. In: *Proceedings of the 15th International Conference on Uncertainty in Artificial Intelligence*, p. 334–342. Morgan Kaufmann Publishers, Inc. Citado em: pages 100

Kossinets, G. e Watts, D. J. (2006). Empirical analysis of an evolving social network. *Science*, 311(5757):88–90. Citado em: pages 379

Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection (Complex Adaptive Systems)*. The MIT Press, 1 ed. Citado em: pages 370

Krogh, A. e Vedelsby, J. (1995). Neural network ensembles, cross validation, and active learning. In: *Advances in Neural Information Processing Systems*, vol. 7, p. 231–238. Citado em: pages 264

Krzanowski, W. e Lai, Y. (1985). A criterion for determining the number of groups in a dataset using sum of squares clustering. *Biometrics*, 44:23–34. Citado em: pages 293

Kumar, S., Gosh, J. e Crawford, M. M. (2002). Hierarchical fusion of multiple classifiers for hyperspectral data analysis. *Pattern Analysis and Appplications*, 5:210–220. Citado em: pages 337

Kuncheva, L. I. (2004). *Combining Pattern Classifiers*. John Wiley & Sons. Citado em: pages 264, 266

Kuncheva, L. I., Hadjitodorov, S. T. e Todorova, L. P. (2006). Experimental comparison of cluster ensemble methods. In: *Proceedings of FUSION 2006*, p. 105–115. Citado em: pages 263, 265, 266

Lange, T., Braun, M., Roth, V. e Buhmann, J. (2003). Stability-based model selection. In: *Advances in Neural Information Processing Systems*, vol. 15, p. 617–624. Citado em: pages 296

Langley, P. (1993). Induction of recursive bayesian classifiers. In: Brazdil, P. (Ed.) *Machine Learning: ECML-93*. LNAI 667, Springer Verlag. Citado em: pages 97

Lauser, B. e Hotho, A. (2003). Automatic multi-label subject indexing in a multilingual environment. In: *Proc. of the 7th European Conference in Research and Advanced Technology for Digital Libraries, ECDL 2003*, vol. 2769, p. 140–151. Springer. Citado em: pages 341

Law, M., Topchy, A. e Jain, A. K. (2004). Multiobjective data clustering. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, p. 424–430. Citado em: pages 231, 232, 263, 264, 265, 269

Law, M. H. e Jain, A. K. (2003). Cluster Validity by Bootstrapping Partitions. Relatório Técnico MSU-CSE-03-5, Department of Computer Science and Engineering, Michigan State University. Citado em: pages 286, 293, 296

Lazzeroni, L. e Owen, A. (2002). Plaid models for gene expression data. *Statistica Sinica*, 12(1):61–86. Citado em: pages 250

LeBlanc, M. e Tibshirani, R. (1993). Combining estimates in regression and classification. Relatório Técnico 9318, Department of Statistics, University of Toronto. Citado em: pages 264

LeCun, Y., Haffner, P., Bottou, L. e Bengio, Y. (1999). *Object Recognition with Gradient-Based Learning*, p. 319–345. Springer Berlin Heidelberg, Berlin, Heidelberg. Citado em: pages 146

LeCun, Y. A., Jackel, L. D., Bottou, L., Brunot, A., Cortes, C., Denker, J. S., Drucker, H., Guyon, I., Müller, E., Säckinger, E., Simard, P. Y. e Vapnik, V. N. (1995). Comparison of learning algorithms for handwritten digit recognition. In: Foulgeman-Soulié, F. e Gallinari, P. (Ed.) *Proceedings of the International Conference on Artificial Neural Networks (ICANN´95)*, vol. 2, p. 53–60, Nanterre, France. Citado em: pages 165

Leskovec, J., Adamic, L. A. e Huberman, B. A. (2007). The dynamics of viral marketing. *ACM Transactions on the Web*, 1(1):1–5. Citado em: pages 372

Leskovec, J., Kleinberg, J. e Faloutsos, C. (2005). Graphs over time: Densification laws, shrinking diameters and possible explanations. In: *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, p. 177–187, New York, NY, USA. ACM. Citado em: pages 381

Lin, H.-T. e Lin, C.-J. (2003). A study on sigmoid kernels for SVM and the training of non-psd kernels by smo-type methods. Relatório técnico, Department of Computer Science, National Taiwan University. Citado em: pages 161

Loh, W. e Shih, Y. (1997). Split selection methods for classification trees. *Statistica Sinica*, 7:815–840. Citado em: pages 329

Lorena, A. C. e Carvalho, A. C. P. L. F. (2007). Design of directed acyclic graph multiclass structures. *Neural Network World*, 17:657–674. Citado em: pages 336

Lorena, A. C. e Carvalho, A. C. P. L. F. (2008). Hierarchical decomposition of multiclass problems. *Neural Network World*, 5:407–425. Citado em: pages 337

Lorena, A. C. e Carvalho, A. C. P. L. F. (2010). Building binary-tree-based multiclass classifiers using separability measures. *Neurocomputing*, 73:2837–2845. Citado em: pages 337

Luo, F., Khan, L., Bastani, F., Yen, I.-L. e Zhou, J. (2004). A dynamical growing self-organizing tree (DGSOT) for hierarchical clustering gene expression profiles. *Bioinformatics*, 20(16):2605–2617. Citado em: pages 258

Luo, F., Tang, K. e Khan, L. (2003). Hierarchical clustering of gene expression data. In: *3rd IEEE Symposium on BioInformatics and BioEngineering (BIBE'2003)*, p. 328–335. Citado em: pages 258

Luo, X. e Zincir-Heywood, N. A. (2005). Evaluation of two systems on multi-class multi-label document classification. In: *International Syposium on Methodologies for Intelligent Systems*, p. 161–169. Citado em: pages 341

Main, J., Dillom, T. e Shiu, S. (2001). A tutorial on case based reasoning. In: Pal, S., Dillon, T. e Yeung, D. (Ed.) *Soft Computing in Case Based Reasoning*. Springer Verlag. Citado em: pages 87

Malek, M. (2001). Hybrid approaches for integrating neural networks and case-based reasoning: From loosely coupled to tightly coupled models. In: Pal, S., Dillon, T. e Yeung, D. (Ed.) *Soft Computing in Case Based Reasoning*. Springer Verlag. Citado em: pages 87

Malek, M. e Amy, B. (1994). Integration of Case-based Reasoning and Neural Networks Approaches for Classification. Relatório Técnico 131 IMAG - 28 LIFIA, laboratoire Leibntz - IMAG. Disponível em http://www-leibniz.imag.fr/RESEAUX/pub/malek.cbrnn.e.ps.gz. Citado em: pages 86

Manevitz, L. M., Yousef, M., Cristianini, N., Shawe-taylor, J. e Williamson, B. (2001). One-class SVMs for document classification. *Journal of Machine Learning Research*, 2:139–154. Citado em: pages 47

Maniezzo, V., Gambardella, L. M. e Luigi, F. (2004). Ant colony optimization. In: Onwubolu, G. C. e Babu, B. V. (Ed.) *New Optimization Techniques in Engineering*, p. 101–117. Springer-Verlag, Berlin, Heidelberg. Citado em: pages 361, 363, 364

Markou, M. e Singh, S. (2003). Novelty detection: A review - part 1: Statistical approaches. *Signal Processing*, 83(12):2481–2497. Citado em: pages 311

Martin, J. (1997). An exact probability metric for decision tree spliting and stopping. *Machine Learning*, 28:257–291. Citado em: pages 106

Martínez-Muñoz, G. e Suárez, A. (2006). Pruning in ordered bagging ensembles. In: Cohen, W. e Moore, A. (Ed.) *Machine Learning, Proceedings of the 23th International Conference*. OmniPress. Citado em: pages 176

Matsubara, E. T. (2008). *Relações entre Ranking, Análise ROC e Calibração em Aprendizado de Máquina*. Tese de Doutoramento, Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos-SP. Citado em: pages 200

Mattison, R. (1998). *AnswerTree Algorithm User's Guide*. SPSS Inc. USA. Citado em: pages 103

Mayoraz, E. e Alpaydim, E. (1998). Support vector machines for multi-class classification. Research Report IDIAP-RR-98-06, Dalle Molle Institute for Perceptual Artificial Intelligence. Citado em: pages 339

Mayoraz, E. e Moreira, M. (1996). On the decomposition of polychotomies into dichotomies. Research Report 96-08, IDIAP, Dalle Molle Institute for Perceptive Artificial Intelligence, Martigny, Valais, Switzerland. Citado em: pages 330

McCulloch, W. S. e Pitts, W. (1943). A logical calculus of the ideas in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133. Citado em: pages 130, 133

McIntyre, R. M. e Blashfield, R. K. (1980). A nearest-centroid technique for evaluating the minimum-variance clustering procedure. *Multivariate Behavioral Research*, 15:225–238. Citado em: pages 285, 293, 294

Meila, M. (2007). Comparing clusterings - an information based distance. *Journal of Multivariate Analyis*, 98:873–895. Citado em: pages 301

Mercer, J. (1909). Functions of positive and negative type and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society*, A 209:415–446. Citado em: pages 160

Merz, C. J. (1998). *Classification and Regression by Combining Models*. Tese de Doutoramento, University of California, Irvine. `http://www.ics.uci.edu/~cmerz/thesis.ps` [Acessado em 19/Jul./2002]. Citado em: pages 173, 264

Michaelewicz, Z. (1996). *Genetic Algorithms + Data Structures = Evolution Programs*. Springer Verlag, 3rd ed. Citado em: pages 369

Michalski, R. S., Mozetic, I., Hong, J. e Lavrac, N. (1986). The multi-purpose incremental learning system AQl5 and its testing application to three medical domains. In: *Proceedings of the Fifth National Conference on Artificial Intelligence*, p. 1041–1045. Morgan Kaufmann. Citado em: pages 121, 124

Michie, D., Spiegelhalter, D. J. e Taylor, C. C. (1994). *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, Upper Saddle River, NJ, USA. Citado em: pages 167, 188, 195, 323

Milgram, S. (1967). The small world problem. *Psychology Today*, 1:61–67. Citado em: pages 394

Milligan, G. e Cooper, M. (1985). An examination of procedures for determining the number of clusters in a data set. *Pshychometrika*, 50:159–179. Citado em: pages 293

Milligan, G. e Cooper, M. (1986). A study of the comparability of external criteria for hierarchical cluster analysis. *Multivariate Behavioral Research*, 21:441–458. Citado em: pages 302, 303

Milligan, G., Soon, T. e Sokol, L. (1983). The effect of cluster size, dimensionality, and the number of clusters on recovery of true cluster structure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(1):40–47. Citado em: pages 303, 304

Milligan, G. W. (1996). *Clustering and Classification*, Capítulo Clustering validation: results and implications for applied analyses. World Scientific Publ. Citado em: pages 294

Millonas, M. M. (1994). Swarms, phase transitions and collective intelligence. In: Langton, C. (Ed.) *Artificial Life III*. Addison-Wesley. Citado em: pages 362

Mingers, J. (1989a). An empirical comparison of pruning methods for decision tree induction. *Machine Learning*, 4:227–243. Citado em: pages 115

Mingers, J. (1989b). An empirical comparison of selection measures for decision-tree induction. *Machine Learning*, 3:319–342. Citado em: pages 111

Minsky, M. e Papert, S. (1969). *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Massachusetts. Citado em: pages 130

Mirkin, B. (2011). *Core Concepts in Data Analysis: Summarization, Correlation and Visualization*. Springer. Citado em: pages 213

Mitchell, T. (1997). *Machine Learning*. McGraw-Hill. Citado em: pages 11, 14, 22, 74, 81, 92, 177, 214, 370

Monard, M. C. e Baranauskas, J. A. (2003). Conceitos de aprendizado de máquina. In: Rezende, S. O. (Ed.) *Sistemas inteligentes - Fundamentos e aplicações*, p. 89–114. Editora Manole. Citado em: pages 13, 192, 193, 196, 197

Monti, S., Tamayo, P., Mesirov, J. e Golub, T. (2003). Consensus clustering: A resampling-based method for class discovery and visualization of gene expression microarray data. *Machine Learning*, 52(1-2):91–118. Citado em: pages 265, 266, 267, 269

Moreira, M. (2000). *The Use of Boolean Concepts in General Classification Contexts*. Tese de Doutoramento, Ecole Polytechnique Federale de Lausanne. Citado em: pages 331

Moreira, M. e Mayoraz, E. (1997). Improved pairwise coupling classification with correcting classifiers. Research report IDIAP-RR 97-09, IDIAP, Dalle Molle Institute of Perceptual Artificial Intelligence, Martigny, Switzerland. Citado em: pages 329

Moreno, J. L. (1953). *Who Shall Survive?* Beacon House, New York. Citado em: pages 371

Morey, L. e Agresti, A. (1984). The measurement of classification agreement: An adjustment to the Rand statistic for chance agreement. *Educational and Psychological Measurement*, 44:33–37. Citado em: pages 302

Morey, L. C., Blashfield, R. K. e Skinner, H. A. (1983). A comparison of cluster analysis techniques within a sequential validation framework. *Multivariate Behavioral Research*, 18:309–329. Citado em: pages 285, 293, 294

Müller, K. R., Mika, S., Rätsch, G., Tsuda, K. e Schölkopf, B. (2001). An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12(2):181–201. Citado em: pages 150, 151, 152, 154, 159

Nadeau, C. e Bengio, Y. (2003). Inference for the generalization error. *Machine Learning*, 52(3):239–281. Citado em: pages 203

Nadler, M. e Smith, E. P. (1993). *Pattern Recognition Engineering*. Wiley. Citado em: pages 241

Nagesh, H., Goil, S. e Choudhary, A. (2001a). Adaptive grids for clustering massive data sets. In: *SIAM Conference on Data Mining*. Citado em: pages 259

Nagesh, H., Goil, S. e Choudhary, A. (2001b). *Data Mining for Scientific and Engineering Applications*, Capítulo Parallel algorithms for clustering high-dimensional large-scale datasets, p. 335–356. Kluwer Academic Publishers. Citado em: pages 259

Nemenyi, P. B. (1963). *Distribution-free Multiple Comparisons*. Tese de Doutoramento, Princeton University. Citado em: pages 206

Newman, M. E. J. (2001). The structure of scientific collaboration networks. *Proceedings of the National Academy of Sciences*, 98(2):404–409. Citado em: pages 373

Newman, M. E. J. (2003a). Mixing patterns in networks. *Physical Review E*, 67(2):026126. Citado em: pages 397

Newman, M. E. J. (2003b). The structure and function of complex networks. *SIAM Review*, 45(23):167–228. Citado em: pages 373, 393, 394, 395, 397

Newman, M. E. J. (2006). Modularity and community structure in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 103(23):8577–8582. Citado em: pages 391

Newman, M. E. J. e Girvan, M. (2004). Finding and evaluating community structure in networks. *Physical Review E*, 69(2):026113. Citado em: pages 387, 390

Ng, A. Y., Jordan, M. I. e Weiss, Y. (2002). On spectral clustering: Analysis and an algorithm. In: *Advances in Neural Information Processing Systems*, vol. 14, p. 849–856. MIT Press. Citado em: pages 275

Ng, R. e Han, J. (1994). Efficient and effective clustering methods for spatial data mining. In: *Proceedings of 20th International Conference on Very Large Databases*, p. 144–155, Santiago, Chile. Citado em: pages 255

Noble, W. S. (2004). Support vector machine applications in computational biology. In: Schölkopf, B., Tsuda, K. e Vert, J.-P. (Ed.) *Kernel Methods in Computational Biology*, p. 71–92. MIT Press. Citado em: pages 149

Nuts, R. e Rousseeuw, P. (1996). Computing depth countours of bivariate point clouds. *Journal of Computational Statistics and Data Analysis*, 23:153–168. Citado em: pages 54

Oliveira, M. e Gama, J. (2010). Mec - monitoring clusters' transitions. In: Agotnes, T. (Ed.) *Proceedings of the 5th Starting AI Researchers' Symposium*, p. 212–224. IOS Press, Amsterdam, The Netherlands. Citado em: pages 389

Ortega, J. (1995). Exploiting multiple existing models and learning algorithms. In: *AAAI 96 - Workshop in Induction of Multiple Learning Models*. Citado em: pages 172

Osman, I. e Laporte, G. (1996). Metaheuristics: A bibliography. *Annals of Operations Research*, 63:511–623. 10.1007/BF02125421. Citado em: pages 361

Pagallo, G. e Haussler, D. (1990). Boolean feature discovery in empirical learning. *Machine Learning*, 5:71–99. Citado em: pages 117

Pakhira, M. K., Bandyopadhyay, S. e Maulik, U. (2004). Validity index for crisp and fuzzy clusters. *Pattern Recognition*, 37(3):487–501. Citado em: pages 291, 293

Pal, N. R. e Bezdek, J. C. (1995). On cluster validity for fuzzy c-means model. *IEEE Transactions on Fuzzy Systems*, 3(3):370–379. Citado em: pages 293

Palla, G., DerÃ©nyi, I., Farkas, I. e Vicsek, T. (2005). Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818. Citado em: pages 389

Park, Y.-J. e Song, M.-S. (1998). A genetic algorithm for clustering problems. In: *Proceedings of the Third Annual Conference on Genetic Programming*, p. 568–575, San Francisco, CA, USA. Morgan Kaufmann Publisher. Citado em: pages 278

Passerini, A., Pontil, M. e Frasconi, P. (2004). New results on error correcting output codes of kernel machines. *IEEE Transactions on Neural Networks*, 15:45–54. Citado em: pages 338, 339

Pau, L. e Gotzche, T. (1992). Explanation facility for neural networks. *Journal of Intelligent and Robotic Systems*, 5:193–206. Citado em: pages 149

Pavlidis, P. e Grundy, W. N. (1999). Combining microarray expression data and phylogenetic profiles to learn functional categories using support vector machines. p. 44–59. Routledge. Citado em: pages 346

Pazzani, M. (1996). Constructive induction of Cartesian product attributes. In: *Proc. of the Conference ISIS96: Information, Statistics and Induction in Science*, p. 66–77. World Scientific. Citado em: pages 98

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, Inc. Citado em: pages 98

Pearlmutter, B. (1992). Gradient descent: second order momentum and saturation error. In: Moody, J. E., Hanson, S. e Lippmann, R. (Ed.) *Advances in Neural Information Processing Systems 2*, p. 887–894. Morgan Kaufmann. Citado em: pages 143

Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2:559–572. Citado em: pages 63

Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U. e Hsu, M. (2001). Prefixspan: Mining sequential patterns by prefix-projected growth. In: *Proceedings of the 17th International Conference on Data Engineering*, p. 215–224, Heidelberg, Germany. Citado em: pages 221

Peng, Y., Flach, P. A., Soares, C. e Brazdil, P. (2002). Improved dataset characterisation for meta-learning. In: *DS '02: Proceedings of the 5th International Conference on Discovery Science*, p. 141–152, London, UK. Springer-Verlag. Citado em: pages 324

Pestana, D. e Velosa, S. (2002). *Introdução à probabilidade e à estatística*. Fundação Calouste Gulbenkian. Citado em: pages 89

Pfahringer, B., Bensusan, H. e Giraud-Carrier, C. (2000). Meta-learning by landmarking various learning algorithms. In: *Proceedings of the Seventeenth International Conference on Machine Learning, ICML'2000*, p. 743–750. Morgan Kaufmann. Citado em: pages 325

Phetkaew, T., Kijsirikul, B. e Rivepiboon, W. (2003). Reordering adaptive directed acyclic graphs: an improved algorithm for multiclass support vector machines. In: *Proceedings of the International Conference on Neural Networks*, p. 1605–1610. Citado em: pages 336

Pimenta, E., Gama, J. e Carvalho, A. C. P. L. F. (2007). Pursuing the best ecoc dimension for multiclass problems. In: *Proceedings of the 20th International Florida Artificial Intelligence Research Society Conference (FLAIRS 2007)*, p. 622–627. AAAI Press. Citado em: pages 334

Platt, J. (2000). Probabilities for sv machines. In: *Advances in Large Margin Classifiers*, p. 61–74. Citado em: pages 165

Platt, J. C., Cristiani, N. e Shawe-Taylor, J. (2000). Large margin DAGs for multiclass classification. In: Solla, S. A., Leen, T. K. e Müller, K.-R. (Ed.) *Advances in Neural Information Processing Systems*, vol. 12, p. 547–553. The MIT Press. Citado em: pages 335

Plaza, E. e Arcos, J. (2002). Constructive adaptation. In: Craw, S. e Preece, A. (Ed.) *6th European Conference on Case-Based Reasoning, Aberdeen, Scotland, Uk*, p. 306–320. Citado em: pages 87

Pons, P. e Latapy, M. (2005). Computing communities in large networks using random walks. In: Yolum, P., Gungor, T., Gurgen, F. S. e Ozturan, C. C. (Ed.) *ISCIS*, p. 284–293. Springer, Berlin. Citado em: pages 390

Pontil, M. e Verri, A. (1998). Support vector machines for 3-D object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(6):637–646. Citado em: pages 336

Prati, R. (2006). *Novas Abordagens em Aprendizado de Máquina para a Geração de Regras, Classes Desbalanceadas e Ordenação de Casos*. Tese de Doutoramento, Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos-SP. Citado em: pages 199, 201

Prati, R. C. e Flach, P. A. (2005). ROCCER: an algorithm for rule learning based on ROC analysis. In: *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, p. 823–828. Citado em: pages 199

Prentzas, J. e Hatzilygeroudis, I. (2002). Integrating hybrid rule-based with case-based reasoning. In: Craw, S. e Preece, A. (Ed.) *6th European Conference on Case-Based Reasoning*, p. 336–349. Springer Verlag. Citado em: pages 87

Price, D. D. S. (1965). Networks of scientific papers. *Science*, 149(3683):510–515. Citado em: pages 396

Price, D. D. S. (1976). A general theory of bibliometric and other cumulative advantage processes. *Journal of the American Society for Information Science*, 27:292–306. Citado em: pages 396

Provost, F. e Domingos, P. (2001). *Well-trained PETs: Improving Probability Estimation Trees*. Relatório técnico, CeDER Working paper IS-00-04, Stern School of Business, New York University. Citado em: pages 201

Prudêncio, R. B. C. e Ludermir, T. B. (2006). A machine learning approach to define weights for linear combination of forecasts. In: *ICANN (1)*, p. 274–283. Citado em: pages 327

Prudêncio, R. B. C. e Ludermir, T. B. (2004). Meta-learning approaches to selecting time series models. *Neurocomputing*, 61:121–137. Citado em: pages 322, 327

Pyle, D. (1999). *Data Preparation for Data Mining*. Morgan Kaufmann. Citado em: pages 25

Qian, Y. e Suen, C. (2000). Clustering combination method. In: *Proceedings of the International Conference on Pattern Recognition (ICPR'2000)*, vol. II, p. 736–739. Citado em: pages 266

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Mateo, CA, USA. Citado em: pages 74, 103, 113, 114, 116, 120, 346, 357

Quinlan, R. (1979). Discovering rules by induction from large collections of examples. In: Michie, D. (Ed.) *Expert Systems in the Microelectronic Age*, p. 168–201. Edinburgh University Press. Citado em: pages 103

Quinlan, R. (1986). Induction of decision trees. *Machine Learning*, 1:81–106. Citado em: pages 115

Quinlan, R. (1988). Simplifying decision trees. In: Gaines, B. e Boose, J. (Ed.) *Knowledge Acquisition for Knowledge Based Systems*. Academic Press. Citado em: pages 113, 114

Quinlan, R. (1995). Mdl and categorical theories (continued). In: Prieditis, A. e Russel, S. (Ed.) *Machine Learning Proc. of 12th International Conference*. Morgan Kaufmann. Citado em: pages 120

Quinlan, R. (1996). Bagging, boosting and C4.5. In: *Proc. 13th American Association for Artificial Intelligence*. AAAI Press. Citado em: pages 189

Quinlan, R. (1998). Data mining tools See5 and C5.0. Relatório técnico, RuleQuest Research. Citado em: pages 126

Quinlan, R. J. (1992). Learning with continuous classes. In: *5th Australian Joint Conference on Artificial Intelligence*, p. 343–348. Citado em: pages 126

Rapoport, A. (1953). Spread of information through a population with socio-structural bias i: Assumption of transitivity. *Bulletin of Mathematical Biophysics*, 15(4):523–533. Citado em: pages 394

Richardson, M. e Domingos, P. (2002). Mining knowledge-sharing sites for viral marketing. In: *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, p. 61–70, New York, NY, USA. ACM. Citado em: pages 372

Riedmiller, M. e Braun, H. (1994). *RPROP – Description and Implementation Details*. Relatório técnico, University of Karlsruhe. Citado em: pages 143

Rifkin, R. e Klautau, A. (2004). In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:1533–7928. Citado em: pages 329, 331, 334

Ritter, T. (1999). The networking company: Antecedents for coping with relationships and networks effectively. *Industrial Marketing Management*, 28(5):467–479. Citado em: pages 373

Rivest, R. L. (1987). Learning Decision Lists. *Machine Learning*, 2:229–246. Citado em: pages 105, 119, 120

Rodrigues, P., Gama, J. e Pedroso, J. (2006). Odac: Hierarchical clustering of time series data streams. In: Ghosh, J., Lambert, D., Skillicorn, D. e Srivastava, J. (Ed.) *Proceedings of the Sixth SIAM International Conference on Data Mining*, p. 499–503, Bethesda, Maryland, USA. Society for Industrial and Applied Mathematics. Citado em: pages 310, 314

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Reviews*, 65:386–408. Citado em: pages 130, 136

Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65. Citado em: pages 291, 292

Rousseeuw, P. J., Ruts, I. e Tukey, J. W. (1999). The bagplot: a bivariate boxplot. *The American Statistician*, 53(4):382–387. Citado em: pages 36

Rumelhart, D. E., Hinton, G. E. e Willians, R. J. (1986). Learning internal representations by error propagation. In: Rumelhart, D. E. e McClelland, J. L. (Ed.) *Parallel Distributed Processing: Foundations*, vol. 1, p. 318–362. MIT Press, Cambridge, MA. Citado em: pages 140

Rumelhart, D. E. e McClelland, J. L. (1986). *Parallel Distributed Processing*, vol. 1: Foundations. The MIT Press. Citado em: pages 142

Sahami, M. (1996). Learning limited dependence Bayesian classifiers. In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, p. 335–338. AAAI Press. Citado em: pages 100, 101

Salzberg, S. L. (1997). A method for identifying splice sites and translational start sites in eukaryotic mRNA. *Computer Applications in Biosciences*, 13(4):365–376. Citado em: pages 203, 205

Saridis, G. (1983). Parameter estimation: Principles and problems. *Automatic Control, IEEE Transactions on*, 28(5):634–635. Citado em: pages 347

Savicky, P. e Fürnkranz, J. (2003). Combining pairwise classifiers with stacking. In: Berthold, M. R., Lenz, H.-J., Bradley, E., Kruse, R. e Borgelt, C. (Ed.) *Advances in Intelligent Data Analysis V, 5th International Symposium on Intelligent Data Analysis, IDA 2003*, p. 219–229. Citado em: pages 339

Schaffer, C. (1993). Overfitting avoidance as bias. *Machine Learning*, 10:153–178. Citado em: pages 115

Schapire, R. (1990). The strength of weak learnability. *Machine Learning*, 5:197–227. Citado em: pages 177

Schapire, R. E. e Singer, Y. (1999). Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336. Citado em: pages 347

Schapire, R. E. e Singer, Y. (2000). Boostexter: a boosting-based system for text categorization. In: *Machine Learning*, p. 135–168. Citado em: pages 347, 348

Schölkopf, B., Guyon, I. e Weston, J. (2003). Statistical learning and kernel methods in bioinformatics. In: Frasconi, P. e Shamir, R. (Ed.) *Artificial Intelligence and Heuristic Methods in Bioinformatics*, p. 1–21. IOS Press. Citado em: pages 149

Seewald, A. e Fürnkranz, J. (2001). An evaluation of grading classifiers. In: Hoffmann, F., Hand, D., Adams, N. e Guimaraes, G. (Ed.) *Advances in Intelligent Data Analysis - IDA01*, p. 115–124. LNCS 2189 Springer Verlag, Berlin. Citado em: pages 172

Seidman, C. (2001). *Data Mining with Microsoft SQL Server 2000 Technical Reference*. Microsoft Press. Citado em: pages 103

Shamir, R. e Sharan, R. (2002). *Current Topics in Computational Biology*, Capítulo Algorithmic approaches to clustering gene expression data, p. 269–299. MIT Press. Citado em: pages 258

Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423. Citado em: pages 333

Sharan, R. e Shamir, R. (2000). CLICK: A clustering algorithm with applications to gene expression analysis. In: *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology (ISMB'2000)*, p. 307–316. Citado em: pages 258

Sharkey, A. J. C. (1999). *Combining Artificial Neural Nets: Ensemble and Modular Multi-Net Systems (Perspectives in Neural Computing)*. Springer Verlag. Citado em: pages 264

Shawe-Taylor, J., Barlett, P. L., Williamson, R. C. e Anthony, M. (1998). Structural risk minimization over data-dependent hierarquies. *IEEE Transactions on Information Theory*, 44(5):1926–1940. Citado em: pages 153

Sheikholeslami, G., Chatterjee, S. e Zhang, A. (1998). WaveCluster: A multi-resolution clustering approach for very large spatial databases. In: *Proceedings of the 24th International Conference on Very Large Data Bases*, p. 428–439, New York City. ACM Press. Citado em: pages 256, 310

Shen, X., Boutell, M., Luo, J. e Brown, C. (2004). Multi-label machine learning and its application to semantic scene classification. In: *International Symposium on Electronic Imaging*, San Jose, CA. Citado em: pages 341, 349

Shetty, J. e Adibi, J. (2004). Email dataset database schema and brief statistical report. Relatório técnico. Citado em: pages 372

Shi, Y. e Eberhart, R. C. (1998). A modified particle swarm optimizer. In: *Evolutionary Computation Proceedings, IEEE World Congress on Computational Intelligence*, p. 69–73. Citado em: pages 365

Sigaud, O. e Wilson, S. W. (2007). Learning classifier systems: a survey. *Soft Computing*, 11:1065–1078. Citado em: pages 370

Silla, C. N. e Freitas, A. A. (2009). A global-model naive Bayes approach to the hierarchical prediction of protein functions. In: *Proceedings of the 9th IEEE International Conference on Data Mining*, p. 992–997. Citado em: pages 357

Silla, C. N. e Freitas, A. A. (2010). A survey of hierarchical classification across different application domains. *Journal Data Mining and Knowledge Discovery*. Citado em: pages 351, 352, 354, 356, 358

Skalak, D. (1997). *Prototype Selection For Composite Nearest Neighbor Classifiers*. Tese de Doutoramento, University of Massachusetts Amherst. Citado em: pages 180, 182, 184

Smith-Miles, K. (2008). Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Comput. Surv.*, 41(1). Citado em: pages 322

Smola, A. e Schölkopf, B. (1998). *A Tutorial on Support Vector Regression*. Relatório Técnico NC2-TR-1998-030, NeuroCOLT2. Citado em: pages 155, 162

Smola, A. J., Barlett, P., Schölkopf, B. e Schuurmans, D. (1999). Introduction to large margin classifiers. In: Smola, A. J., Barlett, P., Schölkopf, B. e Schuurmans, D. (Ed.) *Advances in Large Margin Classifiers*, p. 1–28. MIT Press. Citado em: pages 151, 152

Smola, A. J. e Schölkopf, B. (2002). *Learning with Kernels*. The MIT Press, Cambridge, MA. Citado em: pages 150, 151, 152, 156

Soares, C. (2004). *Learning Rankings of Learning Algorithms: recommendation of algorithms with meta-learning*. Tese de Doutoramento, Universidade do Porto, Porto, Portugal. Citado em: pages 322, 324, 326

Soares, C., Petrak, J. e Brazdil, P. (2001). Sampling-based relative landmarks: Systematically test-driving algorithms before choosing. In: P, B. e A, J. (Ed.) *Progress in Artificial Intelligence, Knowledge Extraction, Multi-agent Systems, Logic Programming and Constraint Solving, Proceedings of the 10th Portuguese Conference on Artificial Intelligence (EPIA 2001)*, vol. 2258 of *Lecture Notes in Computer Science*, p. 88–95, Porto, Portugal. Springer. Citado em: pages 325

Socha, K. (2004). Aco for continuous and mixed-variable optimization. In: *ANTS Workshop*, p. 25–36. Citado em: pages 364

Socha, K. e Dorigo, M. (2008). Ant colony optimization for continuous domains. *European Journal of Operational Research*, 185(3):1155–1173. Citado em: pages 364

Sousa, E., Traina, A., Traina, J. C. e Faloutsos, C. (2007). Evaluating the intrinsic dimension of evolving data streams. *New Generation Computing*, 25(1):33–60. Citado em: pages 311

Souto, M. C. P., Lorena, A. C., Delbem, A. C. B. e Carvalho, A. C. P. L. F. (2003). *Técnicas de Aprendizado de Máquina para Problemas de Biologia Molecular*, p. 103–152. Minicursos de Inteligência Artificial, Jornada de Atualização Científica em Inteligência Artificial, XXIII Congresso da Sociedade Brasileira de Computação. Citado em: pages 213

Sovat, R. (2002). *Uma Abordagem Híbrida Baseada em Casos e Redes Neurais. Uma aplicação: escolha e configuração de modelos de redes neurais*. Tese de Doutoramento, Universidade de São Paulo. Citado em: pages 324

Spackman, K. A. (1989). Signal detection theory: valuable tools for evaluating inductive learning. In: *Proceedings of the 6th International Workshop on Machine Learning*, p. 160–163. Citado em: pages 199

Spinosa, E., Gama, J. e Carvalho, A. (2008). Cluster-based novel concept detection in data streams applied to intrusion detection in computer networks. In: *Proceedings of the 2008 ACM Symposium on Applied Computing*, p. 976–980. ACM Press. Citado em: pages 311

Strehl, A. e Ghosh, J. (2002). Cluster ensembles - a knowledge reuse framework for combining multiple partitions. *Journal on Machine Learning Research (JMLR)*, 3:583–617. Citado em: pages 263, 265, 266, 267, 269, 270, 271, 272, 275, 279

Su, C.-Y., Lo, A., Lin, C.-C., Chang, F. e Hsu, W.-L. (2005). A novel approach for prediction of multi-labeled protein subcellular localization for prokaryotic bacteria. In: *CSBW '05: Proceedings of the 2005 IEEE Computational Systems Bioinformatics Conference - Workshops*, p. 79–82, Washington, DC, USA. IEEE Computer Society. Citado em: pages 346

Sun, A. e Lim, E.-P. (2001). Hierarchical text classification and evaluation. In: *Proceedings of the 1st IEEE International Conference on Data Mining*, p. 521–528. Citado em: pages 351, 358

Takahashi, F. e Abe, S. (2003). Optimizing directed acyclic graph support vector machines. In: *Proceedings of Artificial Neural Networks in Pattern Recognition*, p. 166–170. Citado em: pages 336

Tapia, E., Bulacio, P. e Angelone, L. (2010). Recursive ecoc classification. *Pattern Recognition Letters*, 31(3):210–215. Citado em: pages 334

Thelwall, M. (2006). Interpreting social science link analysis research: A theoretical framework. *Journal of the American Society for Information Science and Technology*, 57(1):60–68. Citado em: pages 387

Thrun, S., Bala, J., Bloedorn, E., Bratko, I., Cestnik, B., Cheng, J., Jong, K. D., Dzeroski, S., Hamann, R., Kaufman, K., Keller, S., Kononenko, I., Kreuziger, J., Michalski, R., Mitchell, T., Pachowicz, P., Roger, B., Vafaie, H., de Velde, W. V., Wenzel, W., Wnek, J. e Zhan, J. (1991). The MONK's problems: A Performance Comparison of Different Learning Algorithms. Relatório Técnico CMU-CS-91-197, Carnegie Mellon University. Citado em: pages 185

Tibshirani, R., Walther, G., Botstein, D. e Brown, P. (2001a). Cluster Validation by Prediction Strength. Relatório técnico, Department of Statistics, Stanford University. `citeseer.nj.nec.com/tibshirani01cluster.html`. Citado em: pages 286, 293

Tibshirani, R., Walther, G. e Hastie, T. (2001b). Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423. Citado em: pages 297, 298

Tickle, A., Orlowski, M. e Diederich, J. (1995). DEDEC: decision detection by rule extraction from neural networks. Relatório técnico, Queensland University of Technology. Citado em: pages 149

Ting, K. e Witten, I. (1997). Stacked generalization: when does it work? In: *Proc. International Joint Conference on Artificial Intelligence*. Morgan Kaufmann. Citado em: pages 182, 184, 187, 189

Topchy, A., Jain, A. e Punch, W. (2003). Combining multiple weak clusterings. In: *Proceedings of the IEEE International Conference on Data Mining (ICDM'2003)*, p. 331–338, Melbourne, Florida, USA. Citado em: pages 263, 264, 265, 266, 267, 269

Topchy, A., Jain, A. e Punch, W. (2004). A mixture model for clustering ensembles. In: *Proceedings of the SIAM International Conference on Data Mining (SDM'2004)*, p. 331–338, Lake Buena Vista, Florida, USA. Citado em: pages 263, 265, 266, 267, 268, 269

Topchy, A., Jain, A. K. e Punch, W. (2005). Clustering ensembles: models of consensus and weak partitions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(12):1866–1881. Citado em: pages 270

Toussaint, G. T. (1974). Bibliography on estimation of misclassification. *IEEE Transactions on Information Theory*, 20(4):472–479. Citado em: pages 194

Towell, G. e Shavlik, J. W. (1993). The extraction of refined rules from knowledge-based neural networks. *Machine Learning*, 131:71–101. Citado em: pages 149

Truyen, T. T., Phung, D. Q. e Venkatesh, S. (2007). Preference networks: Probabilistic models for recommendation systems. In: *Proceedings of the 6th Australasian Conference on Data Mining and Analytics*, p. 195–202, Darlinghurst, Australia. Australian Computer Society, Inc.. Citado em: pages 373

Tsoumakas, G. e Katakis, I. (2007). Multi label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3):1–13. Citado em: pages 341, 347, 348, 349

Tsoumakas, G. e Vlahavas, I. (2007). Random k-labelsets: An ensemble method for multilabel classification. In: *Proceedings of the 18th European Conference on Machine Learning (ECML 2007)*. Citado em: pages 344

Tukey, J. W. (1977). *Exploratory Data Analysis*. Addison-Wesley. Citado em: pages 56

Tumer, K. e Ghosh, J. (1995). Classifier combining: analytical results and implications. In: *AAAI 96 - Workshop in Induction of Multiple Learning Models*. Citado em: pages 169

Tumer, K. e Ghosh, J. (1996a). Error correlation and error reduction in ensemble classifiers. *Connection Science, Special issue on combining artificial neural networks: ensemble approaches*, 8, N.3-4:385–404. Citado em: pages 169

Tumer, K. e Ghosh, J. (1996b). Theoretical foundations of linear and order statistics combiners for neural pattern classifiers. Relatório Técnico TR-95-02-98, University of Texas at Austin, The Computer and Vision Research Center. Citado em: pages 169

Van De Bunt, G. G., Van Duijn, M. A. J. e Snijders, T. A. B. (1999). Friendship networks through time: An actor-oriented dynamic statistical network model. *Computational and Mathematical Organization Theory*, 5(2):167–192. Citado em: pages 373

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag, New York. Citado em: pages 129, 149, 151, 162

Vapnik, V. N. (1998). *Statistical Learning Theory*. John Wiley and Sons. Citado em: pages 151

Vapnik, V. N. e Chervonenkis, A. Y. (1971). On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):283–305. Citado em: pages 149

Vendramin, L., Campello, R. J. G. B. e Hruschka, E. R. (2010). Relative clustering validity criteria: A comparative overview. *Statistical Analysis and Data Mining*, 3:209–235. Citado em: pages 293

Vilalta, R. e Drissi, Y. (2002). A perspective view and survey of meta-learning. *Artif. Intell. Rev.*, 18(2):77–95. Citado em: pages 322

Vilalta, R., Giraud-Carrier, C., e Brazdil, P. (2005). *Data Mining and Knowledge. Discovery Handbook: A Complete Guide for Practitioners and Researchers*, Capítulo Meta-Learning: Concepts and techniques, p. 1–17. Kluwer Academic Publishers. Citado em: pages 323, 324

Wang, W., Yang, J. e Muntz, R. (1997). STING: A statistical information grid approach to spatial data mining. In: *Proceedings of the 23rd International Conference on Very Large Databases*, p. 186–195, Athens, Greece. Citado em: pages 259

Wang, Y. e Witten, I. H. (1997). Induction of model trees for predicting continuous classes. In: *9th European Conference on Machine Learning*. Citado em: pages 126

Wasserman, S. e Faust, K. (1994). *Social Network Analysis: Methods and Applications*. Cambridge University Press, Cambridge. Citado em: pages 374, 379, 383, 395

Watts, D. J. e Strogatz, S. H. (1998). Collective dynamics of small-world networks. *Nature*, 393(6684):440–442. Citado em: pages 381, 383

Webb, G., Boughton, J. e Wang, Z. (2005). Not so naïve Bayes: Aggregating one-dependence estimators. *Machine Learning*, 58(1):5–24. Citado em: pages 101

Wei, C.-P. e Chiu, I.-T. (2002). Turning telecommunications call details to churn prediction: a data mining approach. *Expert Systems with Applications*, 23(2):103–112. Citado em: pages 372

Weingessel, A., Dimitriadou, E. e Hornik, K. (2003). An ensemble method for clustering. In: *Distributed Statistical Computing (DSC'2003)*, Wien, Austria. `http://www.ci.tuwien.ac.at/Conferences/DSC-2003/` [Acessado em: 19/Jan/2004]. Citado em: pages 266, 268, 269

Weiss, S. e Indurkhya, N. (1998). *Predictive Data Mining, a Practical Guide*. Morgan Kaufmann Publishers. Citado em: pages 120

Widrow, B. e Hoff, M. (1960). Adaptative switching circuits. *Institute of Radio Engineers, Western Electronic Show and Convention*. Citado em: pages 138

Wikipedia (2012). Social network analysis software. [Online; accessed 02-Fevereiro-2012]. Citado em: pages 397

Wilcoxon, F. (1943). Individual comparisons by ranking methods. *Biometrics*, 1:80–83. Citado em: pages 204

Wilson, D. R. e Martinez, T. R. (1997). Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research*, 6:1–34. Citado em: pages 242

Wilson, R. J. e Watkins, J. J. (1990). *Graphs: An Introductory Approach - A First Course in Discrete Mathematics*. John Wiley & Sons. Citado em: pages 278

Windeatt, T. e Ghaderi, R. (2003). Coding and decoding strategies for multi-class learning problems. *Information Fusion*, 4(1):11–21. Citado em: pages 338

Wiratunga, N., Craw, S. e Rowe, R. (2002). Learning to adapt for case-based design. In: Craw, S. e Preece, A. (Ed.) *6th European Conference on Case-Based Reasoning*, p. 421–435. Springer Verlag. Citado em: pages 87

Witten, I., Frank, E. e Hall, M. (2011). *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufman Publishers Inc., San Francisco, CA, USA, 3 ed. Citado em: pages 19

Wnek, J. e Michalski, R. S. (1994). Hypothesis-driven constructive induction in AQ17-HCI: A method and experiments. *Machine Learning*, 14:139–168. Citado em: pages 122, 124

Wolpert, D. (1992). Stacked generalization. *Neural Networks*, 5:241–260. Citado em: pages 182, 186, 189

Wolpert, D. e Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Trans. Evolutionary Computation*, 1(1):67–82. Citado em: pages 167

Wolpert, D. H. (1996). The lack of a priori distinctions between learning algorithms. *Neural Comput.*, 8(7):1341–1390. Citado em: pages 321

Wu, T.-F., Lin, C.-J. e Weng, R. C. (2004). Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research*, 5:975–1005. Citado em: pages 339

Wu, W., Guo, Q., de Aguiar, P. F. e Massart, D. L. (1998). The star plot: an alternative display method for multivariate data in the analysis of food and drugs. *J Pharm. Biomed. Anal.*, 17(6-7):1001–13. Citado em: pages 36

Xie, X. L. e Beni, G. (1991). A validity measure for fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(8):841–847. Citado em: pages 293

Xu, J. e Che, H. (2005). Criminal network analysis and visualization. *Communications of the ACM*, 48(6):101–107. Citado em: pages 372

Xu, R. e Wunsch, D. (2005). Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678. Citado em: pages 242

Yang, M.-S. e Wu, K.-L. (2001). A new validity index for fuzzy clustering. In: *IEEE International Fuzzy Systems Conference*, p. 89–92. Citado em: pages 293

Yang, Y., Webb, G. I. e Wu, X. (2005). Discretization methods. In: Maimon, O. e Rokach, L. (Ed.) *The Data Mining and Knowledge Discovery Handbook*, p. 113–130. Springer US. Citado em: pages 22, 60

Yeung, K. (2001). *Cluster Analysis of Gene Expression Data*. Tese de Doutoramento, University of Washington. Citado em: pages 292

Yeung, K. Y., Haynor, D. R. e Ruzzo, W. L. (2001). Validating clustering for gene expression data. *Bioinformatics*, 17(4):309–318. Citado em: pages 286, 293, 294, 295, 296

Zadrozny, B. (2001). Reducing multiclass to binary by coupling probability estimates. In: *Proceedings of the 14th Annual Conference Neural Information Processing Systems, Advances in Neural Information Processing Systems*, vol. 14, p. 1041–1048. Citado em: pages 339

Zaki, M. J. (2000). Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering*, 12(3):372–390. Citado em: pages 221

Zeng, Y., Tang, J., Garcia-Frias, J. e Gao, G. (2002). An adaptive meta-clustering approach: Combining the information from different clustering results. In: *IEEE Computer Society Bioinformatics Conference (CSB'2002)*, p. 276–287, Stanford, California. Citado em: pages 243, 266

Zhang, M.-L. e Zhou, Z.-H. (2005). A k-Nearest Neighbor Based Algorithm for Multi-label Classification. In: *IEEE International Conference on Granular Computing*, vol. 2, p. 718–721. Citado em: pages 341, 347

Zhang, T., Ramakrishnan, R. e Livny, M. (1996). BIRCH: an efficient data clustering method for very large databases. In: *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, p. 103–114, Montreal, Canada. Citado em: pages 251, 310

Zheng, Z. (1998). Naive Bayesian classifier committees. In: Nedellec, C. e Rouveirol, C. (Ed.) *Machine Learning ECML-98*. Springer Verlag. Citado em: pages 179

Índice Remissivo