Classification: Evaluation and Validation (some of the material taken from the AIMA book, chapter 19)

Learning: definition

- An agents learns if it improves its performance in future tasks after observing past or current situations.”

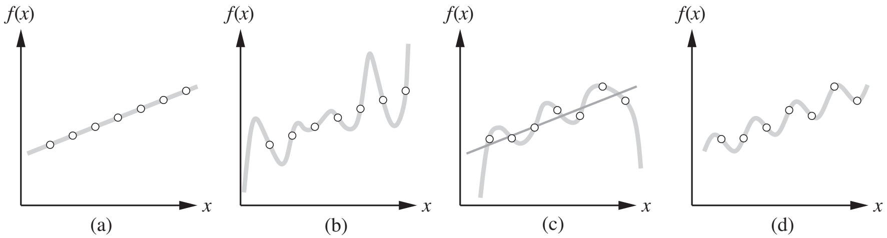
Learning from observations

- Representation of components: various methods
- Feedback:
  - **supervised learning:** given a set of pairs observations-labels, learn a function that can predict the label for each observation
  - **reinforcement learning**: agent receives rewards for each action taken. It can use this info to judge the quality of future actions
  - **unsupervised learning**: agent learns patterns about observations even not having labels
- background/prior knowledge: description/representation of observations

Inductive Learning

- In supervised learning, the agent has access to the correct or approximate value of the function applied to the observation (label).
- **Bias**: preference for one of the possible functions.

Inductive Learning



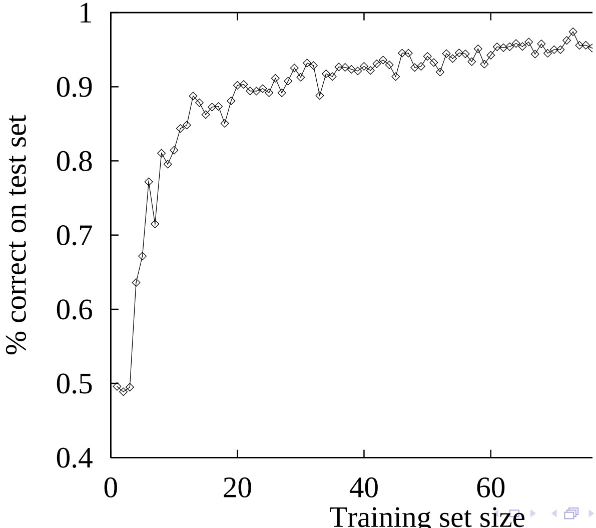
Inductive Learning

- Problem: how can we know if a learning algorithm will produce a theory or hypothesis (model) that will have a good predictive power?

Performance of a learning algorithm

- A supervised learning algorithm is good if it produces hypothesis (models) that can predict well new unseen cases.
- Verify predictions in a **test** set:
  1. Choose a dataset
  2. Divide it in **training** and **test** sets.
  3. Use the learning algorithm with the training set and generate the model (hypothesis H).
  4. Calculate percentage of examples correctly predicted in the test set
  5. Repeat steps 1 to 4 using different sizes of training sets chosen randomly
- Result: learning curve

Performance of a learning algorithm



Evaluation Metrics

Most metrics used for model evaluation in classification tasks are based on the confusion (or contingency) matrix. For binary classification problems, we define one class as positive and the second as negative. We then define:

- **TP**: True Positives (number of instances that are positive and were predicted as positive)
- **TN**: True Negatives (number of instances that are negative and were predicted as negative)
- **FP**: False Positives (number of instances that are negative and were predicted as positive)
- **FN**: False Negatives (number of instances that are positive and were predicted as negative)

Evaluation Metrics

- $TP + FN$ corresponds to the total number of positive instances
- $TN + FP$ total number of negative instances
- $TN + TP$ number of correctly classified instances
- $FP + FN$ number of incorrectly classified instances

Evaluation Metrics

An example of a confusion matrix for a binary classification problem:

Class/Predicted	+	-	Total
+	10	1	11
-	2	35	37
Total	12	36	48

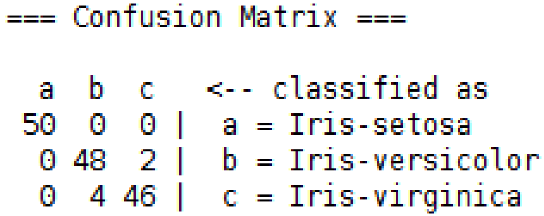
Evaluation Metrics

From this table, we can extract:

- Total number of instances: 48, from which 11 are positive and 37 are negative.
- The classifier misclassified 1 positive and 2 negative instances (secondary diagonal shows the errors).
- The classifier correctly classified 10 out of the 11 positives and 35 out of the 37 negatives (main diagonal shows the correct classified instances).
- The classifier predicted 12 instances as being of class positive and 36 instances as being of class negative.

Evaluation Metrics

NOTE: if we have more than 2 classes, the confusion matrix will have more rows and columns. For example, assume the iris dataset (with classes setosa, virginica and versicolor):



```
=== Confusion Matrix ===

  a  b  c   <-- classified as
 50  0  0 |  a = Iris-setosa
  0 48  2 |  b = Iris-versicolor
  0  4 46 |  c = Iris-virginica
```

This was generated by running the weka toolkit with a Naive Bayes model. In this matrix, all setosa examples were correctly classified by the model. The model missed 2 versicolor cases (these 2 were incorrectly mistaken with virginica). From the 50 cases virginica, 4 were misclassified as versicolor. In these cases, metrics are calculated for each class.

Evaluation Metrics

- Recall = True Positive Rate (TPR) = Sensitivity =

$$\frac{TP}{TP + FN}$$

  Meaning: from all positives, how many were actually predicted as positive?

- Specificity = True Negative Rate (TNR) = 1 - FPR (False Positive Rate) =

$$\frac{TN}{TN + FP}$$

- False Positive Rate (FPR) = 1 - TNR

Evaluation Metrics

- Accuracy = Correctly Classified Instances (CCI) =

  $$\frac{TP + TN}{TP + FN + FP + TN}$$

  Meaning: from all instances, how many were actually correctly predicted?

- Error rate = 1 - CCI =

  $$\frac{FP + FN}{TP + FN + FP + TN}$$

Evaluation Metrics

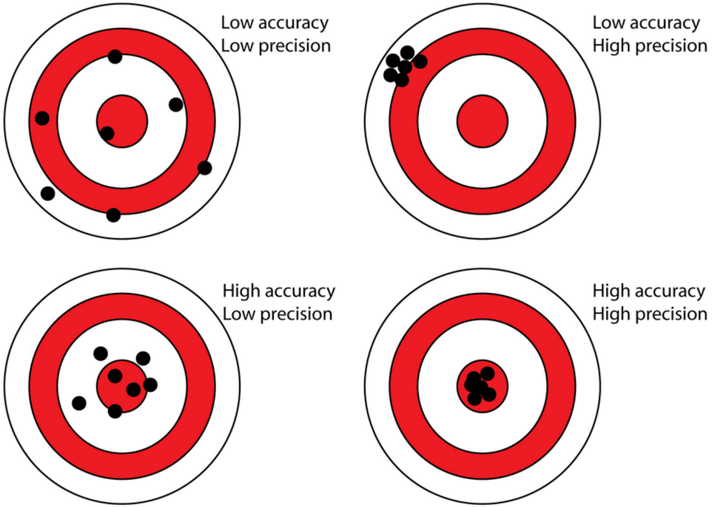
- Precision = Positive Predictive Value (PPV) =

$$\frac{TP}{TP + FP}$$

  Meaning: from all instances predicted as positive, how many are actually positive?
- $F_\beta measure = (1 + \beta^2)\frac{Precision \times Recall}{(\beta^2 \times Precision) + Recall}$
  When $\beta = 1$, the F_1 measure is the harmonic mean between Precision and Recall.

Evaluation Metrics

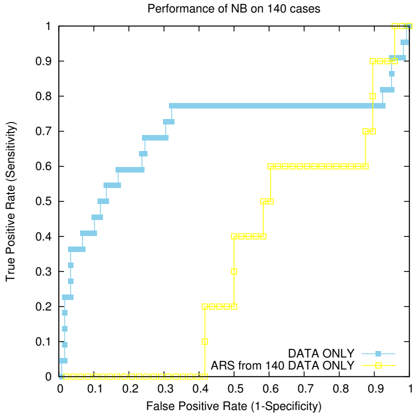
Nice illustration of Precision x Accuracy



Evaluation Metrics

- Receiver Operating Characteristic (ROC) curve: values in the curve between 0 and 1, according to threshold variation.
- X axis represents False Positive Rate (FPR = 1 - Specificity). Best point is zero.
- Y axis represents True Positive Rate (TPR). Best point is 1.
- Curve is plotted with respect to a given class value (positive or negative).
- Used to help specialists to choose cut points related with false positive rate and true positive rate.
- ROC curves are useful when the model can predict numerical values.

ROC example



- This figure shows two ROC curves, one for each trained model.
- Points of the ROC curve are obtained by thresholding.
- The ideal point in an ROC curve is TPR=1 and FPR=0.

ROC example: thresholding

- For example, suppose we have the following expected and predicted values (where P is a positive class and N is negative)
- Also assume that your model gives you a number as a prediction:

Expected	Predicted
P	0.8
P	0.6
P	0.2
N	0.1
N	0.9
N	0.7
N	0.6
N	0.5

ROC example: thresholding

- The algorithm to calculate the curve points is:

```
Initalize arrays TP, FN, TP, FP with zeros
for i = 0 to 1 step 0.1 {
   if Expected == P and Predicted >= i TP[i]++
   if Expected == P and Predicted < i FN[i]++
   if Expected == N and Predicted >= i FP[i]++
   if Expected == N and Predicted < i TN[i]++
}
```

- This cycle may also vary around the predicted values: 0.1, 0.2, 0.5, 0.6, 0.7, 0.8, 0.9, but it needs to contain points 0.0 and 1.0.

ROC example: thresholding

- For this example, our arrays will be:

Threshold	# TP	# FP	TPR	FPR
0.0	3	5	1	1
0.1	3	4	1	4/5
0.2	3	4	1	4/5
0.3	2	3	2/3	3/5
0.4	2	2	2/3	2/5
0.5	2	1	2/3	1/5
0.6	2	1	2/3	1/5
0.7	1	1	1/3	1/5
0.8	1	1	1/3	1/5
0.9	0	0	0	0
1.0	0	0	0	0

ROC: area

- The ROC curve defines an area
- The area under the curve is also a very popular metric (AUC or AUCROC)
- This area varies between 0 and 1
- The closer to 1 the better

ROC: analysis

- When analyising an ROC curve, a specialist may decide which threshold to use in the model
- The specialist use cut points by using verticals that crosses different points in the X-axis
- Depending on the domain, specialists will look for thresholds that minimise either FP or FN

ROC: problem

- If the classes are imbalanced the ROC curve may show optimistic results
- If the positive class is much smaller than the negative, an error in the negative class will be much less significant than an error in the positive class
- In these cases, another curve is used: the Precision-Recall (PR) curve
- In the PR curve, the X-axis has the TPR (Recall) values and the Y-axis has the Precision values
- In the ROC curve we plot $\frac{TP}{TP+FN}$ against $\frac{FP}{FP+TN}$
- In the PR curve we plot $\frac{TP}{TP+FN}$ against $\frac{TP}{FP+TP}$
- For the same TP, the denominator of the PR curve will not dominate as much as the denominator of the ROC curve

Validation

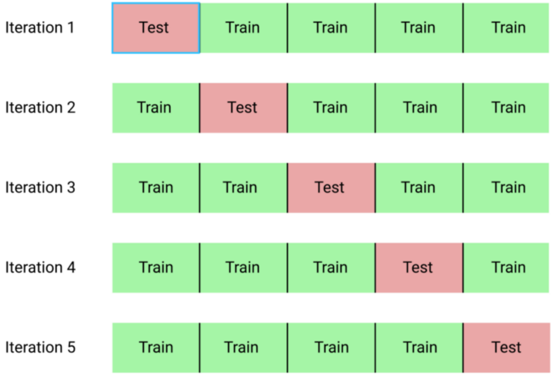
- Models need to be validated and an estimate of the error needs to be calculated
- In order to do that, we usually divide our entire dataset in **training** and **testing** data
- There exists at least two methods for model validation, which require iterative training and testing
    - cross-validation
    - bootstrapping

Cross-Validation

- In cross-validation, the dtaset is divided in k partitions (folds) of approximately the same size
- Training is performed k times, each time using one of the partitions for testing and the reamining for training
- Usually, cross-validation is **stratified**, meaning that, each fold will have approximately the same number of positive and negative examples...
- ...except if it is **leave-one-out**, where the dataset of size n is divided in $n - 1$ partitions and each test set has exactly one example
- leave-one-out is normally used when the dataset is small
- Care needs to be taken when calculating performance metrics in the context of cross-validation (see paper `https://dl.acm.org/doi/10.1145/1882471.1882479`)

Cross-Validation

5-fold cross-validation:



Bootstrapping

- In bootstrapping, the dataset is divided in training set partition and test set partition k times
- Usual values for partitioning may be 70%/30%, 80%/20%, 67%/33%
- In that case, metrics need to be calculated per each one of the k samples and an average is reported

Performance evaluation

- Quantitative performance
- How about **qualitative** performance?
  - Interpretability?
  - Explainability?
  - Usefulness?