# Regression and KNN

Inês Dutra (based on slides of Alípio Jorge)

December 2024

# Machine Learning tasks

- **Classification**
  - Given a sample of pairs $< Obj, Class >$, where
    - $Obj \in Objects$
    - $Class \in Classes$
  - obtain a function $f : Objects \rightarrow Classes$
- **Regression**
  - Given a sample of pairs $< Obj, Val >$, where
    - $Obj \in Objects$
    - $Val \in Values \subseteq \mathbb{R}$
  - obtain a function $f : Objects \rightarrow Values$

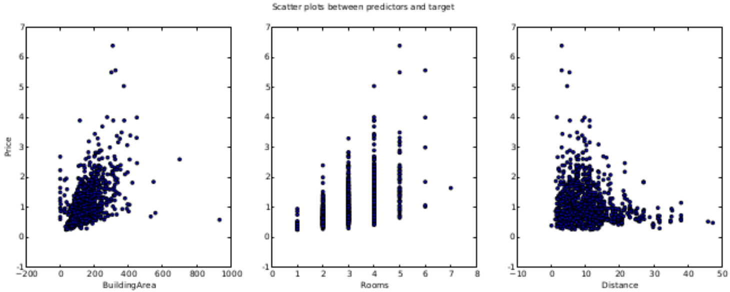
# A regression example: data understanding

- Let's look at the data (using part of **Melbourne Housing** data set)
  - Predictors: **BuildingArea**, **Rooms** and **Distance**
  - Target: **Price** (Median value of homes)

| | BuildingArea | Rooms | Distance |
|---|---|---|---|
| 11039 | 144.0 | 3 | 4.5 |
| 9984 | 131.0 | 3 | 13.5 |
| 8259 | 67.0 | 1 | 8.8 |
| 9156 | 150.0 | 2 | 2.1 |
| 6567 | 87.0 | 2 | 8.7 |

# A regression example: data understanding

- Can we predict *Price* from the predictors?
  - What do plots tell us?



# A regression example: modeling: Linear Regression

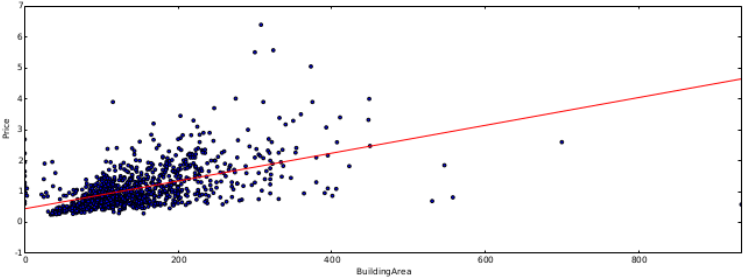
- We can obtain a **linear function** $f$ from the data such that
  - $\widehat{Price} = f(BuildingArea, Rooms, Distance)$
  - that can be done using **Linear Regression**
- If we have $m$ attributes

$$\hat{y} = f(x_1, x_2, \ldots, x_m) = \beta_0 + \sum_{i=1}^{m} \beta_i . x_i$$

- There is an **algorithm** that, given the data, finds the **parameters** $\beta_i$ - it is based on a centuries old mathematical procedure

# A regression example: modeling: Linear Regression

- Linear Regression
  - Let's visualize the effect of LR with one predictor: *BuildingArea*
  - This is called **simple regression**
  - The red line was **algorithmically** obtained from the data



# A regression example: modeling: Linear Regression

- Linear Regression
  - Let's see the function obtained

```
Model slope:     4487.65167729

Model intercept: 441450.158562
```

$$\widehat{Price} = 441450.16 + 4487.65 \times BuildingArea$$

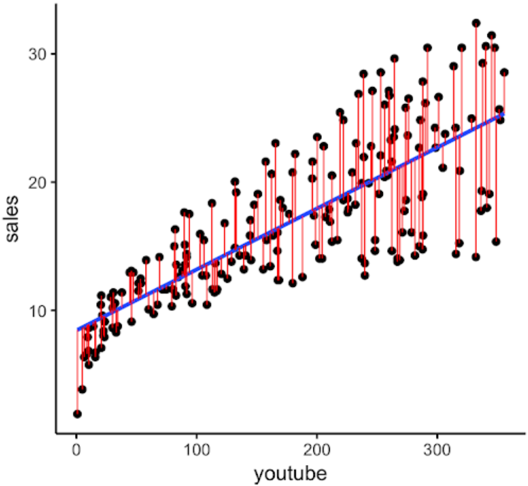
# A regression example: modeling: Linear Regression

- Linear Regression
  - **how good** is the model?
  - we can measure $R^2$ (r-squared), a measure of **fit**
    - 0 is the worst fit (predicting average)
    - 1 is the best fit (got them all)
    - a low value indicates **underfit**
  - a fit well above 0 may be useful
    - depending on the problem
    - in this case it is above zero but not high

```
Model R2:     0.29210253038
```

# A regression example: modeling: R squared

- What is $R^2$ measuring?
  - How much the predicted $\hat{y}$ are close to the actual $y$
  - The difference $e_i = \hat{y}_i - y_i$ is a **residual** or error
    - Best fit has $e_i = 0$ for all $i$



# A regression example: modeling: R squared

- The **sum of the squares** of the residuals is a measure of total **error**

$$SS_{res} = \sum_{i=1}^{n} e_i^2$$

- We **normalize** this error with the error predicting the mean $SS_{tot} = \sum_{i=1}^{n} (y_i - \overline{y})^2$
- And subtract from 1

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

# A regression example: modeling: multivariate regression

- Use more predictors
  - the prediction is now a **hyperplane** on a 3 dimensional space
  - the value of $R^2$ increases considerably

`LinearRegression(fit_intercept=True)`

```
Predictors: ['BuildingArea', 'Rooms', 'Distance']
Coefficients (alphas):     [  2727.02768147 303464.96545695 -39576.58981672]
Model intercept: 211030.8952584623
Model R2:     0.48841571723126587
```

# Regression: finding the model

- The regression model is found **analytically**
  - $\overrightarrow{\beta} = [\ \beta_0, \beta_1, \ldots, \beta_m\ ]$
  - the $i^{th}$ case is $x_i = [1, x_1^i, \ldots, x_m^i]$
  - then, we can use the dot product for estimating $y_i$

$$\hat{y}_i = \overrightarrow{\beta}.x_i$$

  - $X$ is the $n \times (m+1)$ matrix of independent variables with a right column of 1s
  - $Y$ is the $n \times 1$ matrix of target/dependent values

$$\overrightarrow{\beta} = (X^T X)^{-1} X^T Y$$

# Regression: finding the model

- Where does this equation **come from**?
- Aim is to find $\beta_i$ that **minimize** the squares of the residuals
  - **least squares** approach

$$\min_{\overrightarrow{\beta}} \sum_{i=1}^{n} (\overrightarrow{\beta}.x_i - y_i)^2$$

- by deriving and equaling to zero we get to the $\overrightarrow{\beta}$ equation

# Regression: finding the model: complexity

- **Computational complexity** analysis
- How **hard** is it to compute the $\beta_i$?
  - matrix multiplications can be $O(n.m^2)$
  - matrix **inversion** can be $O(m^3)$
- **not so bad**
  - linear with the number of cases (great)
  - problematic with many predictors (**usually** not a problem)

# More on regression

- Lasso regression (L1 Regularization):
  Adds a penalty for the absolute size of coefficients:

$$\text{Loss Function} = RSS + \lambda \sum_{j=1}^{n} |\beta_j|$$

- Ridge regression (L2 Regularization):
  Adds a penalty for large coefficients:

$$\text{Loss Function} = RSS + \lambda \sum_{j=1}^{n} \beta_j^2$$

- Polynomial regression
- Logistic regression

# Logistic Regression

- Statistical method to model binary outcomes (yes/no, 0/1, -1/1 etc)
- Instead of directly modeling the otucome, it gives the probability of the outcome belonging to a particular class using the logistic (sigmoid) function:

$$P(y = 1 \mid X) = \frac{1}{1 + \epsilon^{-(w_0 + w_1 x_1 + \ldots + w_n x_n)}}$$

# Logistic Regression: example

- Problem: a company wants to predict whether customers will purchase a product (yes=1, no=0) based on the amount of advertisements ($x_1$).

Dataset:

| Advertising budget ($x_1$) | Pruchase ($y$) |
|---|---|
| 2 | 0 |
| 4 | 0 |
| 6 | 1 |
| 8 | 1 |

$$P(y = 1 \mid x_1) = \frac{1}{1 + \epsilon^{-(w_0 + w_1 x_1)}}$$

  - train the model

# Logistic Regression: example

- Assume $w_0 = -4$ and $w_1 = 0.9$

$$P(y = 1 \mid x_1) = \frac{1}{1 + \epsilon^{-(4+0.9x_1)}}$$

- Interpreting results: for $x_1 = 5$ (advertising budget of \$5):

$$P(y = 1 \mid x_1 = 5) = \frac{1}{1 + \epsilon^{-(4+0.9\times5)}} \approx 0.62$$

- meaning: there is a 62% chance that the customer will purchase the product
- interpreting weights:
  - $w_0 = -4$: baseline log-odds when $x_1 = 0$
  - $w_1 = 0.9$: for every 1-unit increase in $x_1$, the log-odds of purchase increase by 0.9

# The nearest neighbor approach

- The aim of modeling is to **discover the hidden function** $f$
  - $f$ is able to **estimate** the target $y$ for new cases $x$
- Linear regression approach
  - $f$ is **assumed** to have a linear form
  - all we have to find are the **parameters** $\beta_i$
  - they are found **analytically** from the data

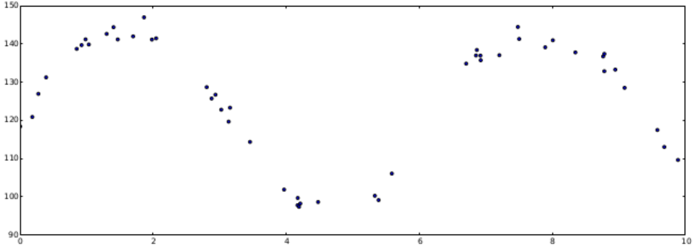
# The nearest neighbor approach

- **Nearest neighbor** approach
  - $f$ is assumed to be **locally smooth**
  - *nearby* cases tend to have **similar values** for $f$
    - if $sim(x_1, x_2)$ is small then $f(x_1) \approx f(x_2)$
  - we can estimate $f(x)$ from the neighbors of $x$

# The nearest neighbor approach

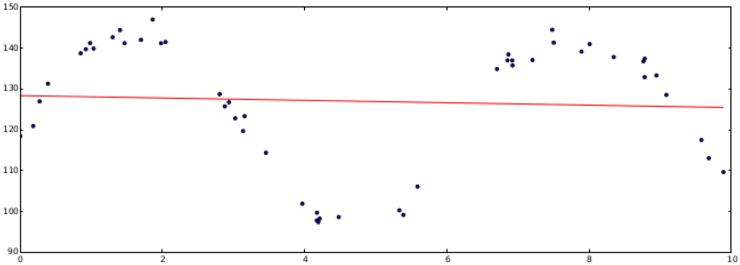
- Suppose we want to model the number of customers in a shop given the time of the day
  - These are the observations (the data)

```
(0, 10)
```



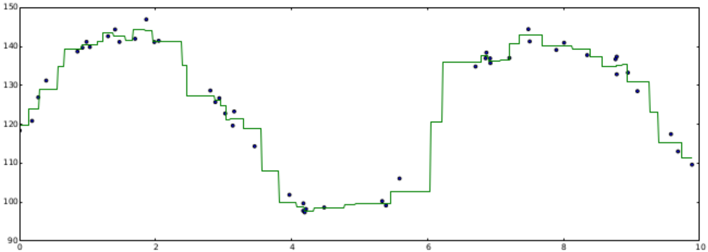
# The nearest neighbor approach

- Linear regression **does not find** a good solution
  - the linear assumption is **too strong**



# The nearest neighbor approach

- A nearest neighbour approach finds a **better** solution
  - using 2 nearest neighbors
  - the corresponding $f$ adapts to the data
  - **be careful!** - it may **overfit**



# The $k$ nearest neighbor approach: kNN

- **Input**:
  - data $X$, $y$
  - parameter $k$, number of neighbors
  - distance measure $d$
  - new case $x_{new}$
- **Output**:
  - estimated value $\hat{y}(x_{new})$
- **Algorithm**:
  - calculate $d(x_i, x_{new})$ for each $x_i \in X$
  - obtain the $k$ $x_{(1)}, \ldots, x_{(k)}$ points that minimize $d$
  - output $\hat{y}(x_{new}) = avg_i \;\; x_{(i)}$

# The nearest neighbor approach

- no model is produced
  - **lazy** learning
    - only use the data when you have to predict
  - opposed to **eager** learning
    - build the model as soon as you have the data

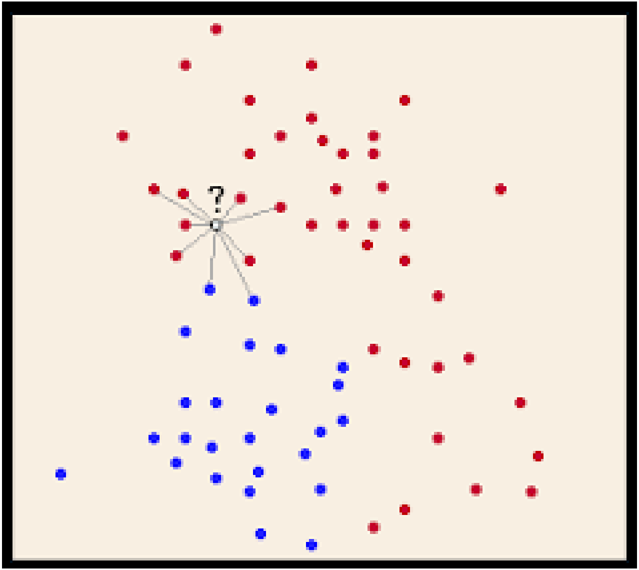
# A classification example

- The kNN approach can also be used for **classification**
  - *The credit office of the bank also has records of previous loan applications and the outcome of the credit (payed with no difficulty, not an easy payment process). The aim is to find a model that automatically supports the decision of the bank credit office for loans*
- This is a **two class** problem
  - class1=‘easy’, class2=‘difficult’

# A classification example: kNN

- The kNN approach for **classification**
  - given a new application $x_{new}$
  - find the $k$ applications closer to $x_{new}$
  - output the **majority** class in those cases

# A classification example: kNN



# Relevant issues

- **Non-numerical** variables in regression
  - categorical can be binarized (dummy variables)
- The importance of **distance functions** in kNN
  - hybrid distances
- The importance of **normalization** in kNN
  - the $<$age,salary$>$ example
- How do these methods cope with **missing data**?
  - matrix operations
  - distance functions

# References


- Books
  - Han, Kamber & Pei, Data Mining Concepts and Techniques, Morgan Kaufman.
- Data
  - https://www.kaggle.com/schirmerchad/bostonhoustingmlnd?select=ho
- Blog articles
  - Computational complexity of machine learning algorithms, https://www.thekerneltrip.com/machine/learning/computational-complexity-learning-algorithms/
- Manuals
  - Nearest neighbors, Scikit learn, https://scikit-learn.org/stable/modules/neighbors.html