

Data Mining

Data Understanding

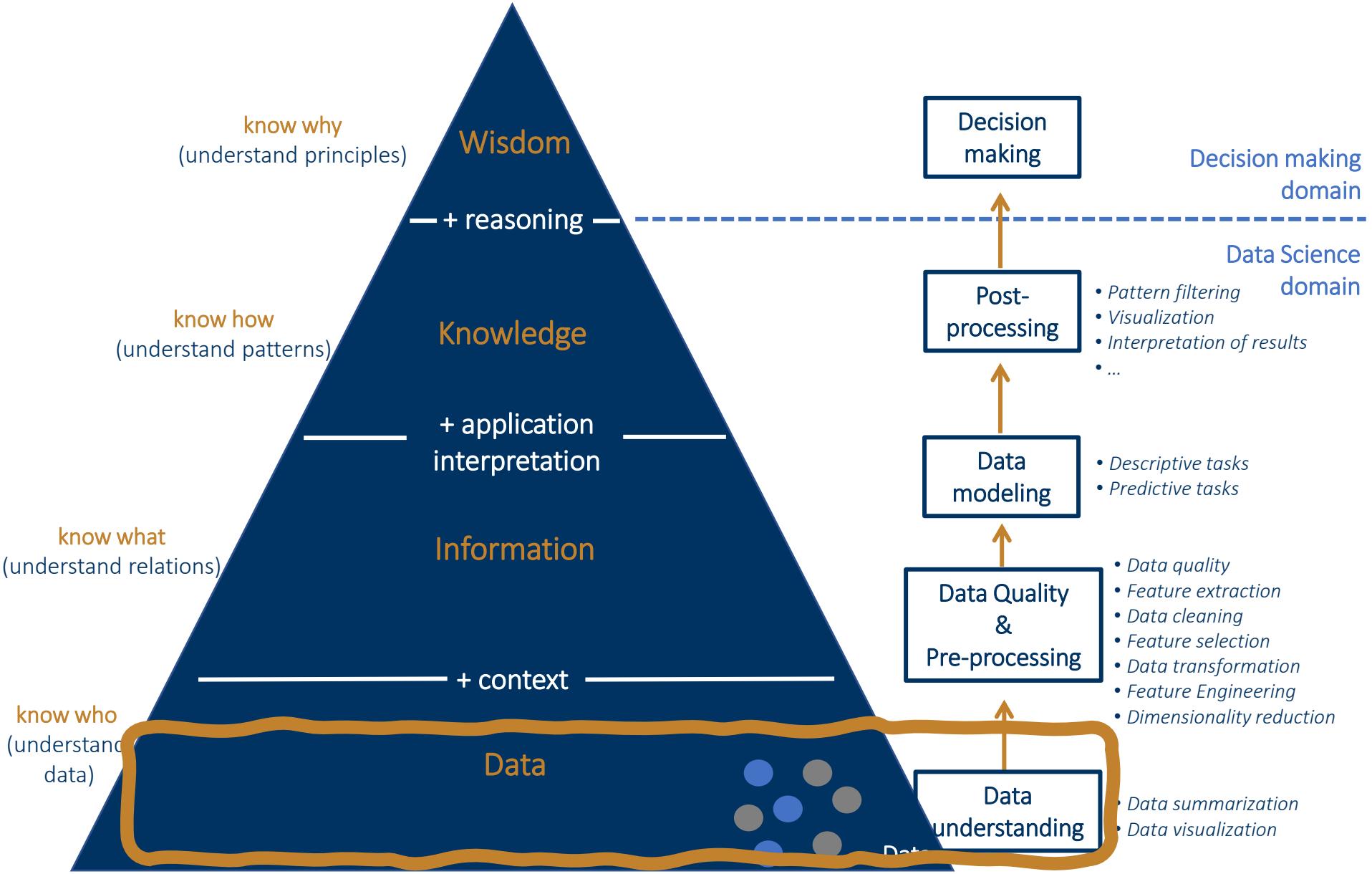
Raquel Sebastião

Departamento de Eletrónica, Telecomunicações e Informática

Universidade de Aveiro

raquel.sebastiao@ua.pt

2022/2023



Contents

- Attributes and Datasets
- Data Summarization
- Data Visualization
- Proximity measures
- Summary

Why get insights from our data?

similarity
visualize
noise
outliers

types analysis
description
distribution

attributes

volume values

What is data?

Collection of objects/examples described by attributes (taken from a domain)

- **Object** – entity
 - described by attributes
 - samples, examples, instances, data points, observations
- **Attribute** – data field
 - represent a characteristic/feature of a data object
 - dimension, feature, variable, characteristics

Attribute values are **numbers** or **symbols** assigned to an attribute for a particular object

Attribute values

Attribute values are numbers or symbols assigned to an attribute for a particular object

- Distinction between attributes and attribute values
 - Same attribute can be mapped to different attribute values
 - Example: height can be measured in feet or meters
 - Different attributes can be mapped to the same set of values
 - Example: Attribute values for ID and age are integers
 - But properties of attribute can be different than the properties of the values used to represent the attribute

Attributes

Type and scales of attributes

- Categorical
 - Nominal
 - Ordinal
- Numeric
 - Interval
 - Ratio

Scale of attributes

Categorical attributes

- finite number of symbols or names
- Sometimes, represented as integers (but they don't represent quantities)

Nominal

- there is no relationship between the values
- only equality is meaningful

Ordinal

- there is a meaningful order (ranking), but magnitude between successive symbols/values are unknown
- both equality and inequality is meaningful

Scale of attributes

Numeric attributes

- real-valued or integer-valued domain

Interval

- values vary within an interval
- equality, inequality and differences are meaningful
- the value 0 or scale origin, is defined arbitrarily
- no true **zero-point**

Ratio

- numbers have an absolute meaning
- equality, inequality, differences and ratios are meaningful
- inherent **zero-point**

Types and scales of attributes

- **Categorical:** set-valued domain composed of a set of symbols (qualitative)
 - **Nominal:** only equality (are two values the same?) is meaningful
 - $\text{domain}(\text{HairColor}) = \{\text{brown, blond, black}\}$, $\text{domain}(\text{Sex}) = \{\text{M, F}\}$
 - **Ordinal:** both equality and inequality (is one value less than another?) are meaningful
 - $\text{domain}(\text{Education}) = \{\text{High School, BS, MS, PhD}\}$, $\text{domain}(\text{size}) = \{\text{S, L}\}$
- **Numeric:** real-valued or integer-valued domain (quantitative)
 - **Interval-scale:** only differences are meaningful
 - temperature (Celsius or Fahrenheit), calendar years
 - **Ratio-scale:** differences and ratios are meaningful
 - Age, temperature (Kelvin), distance, income, counts, elapsed time

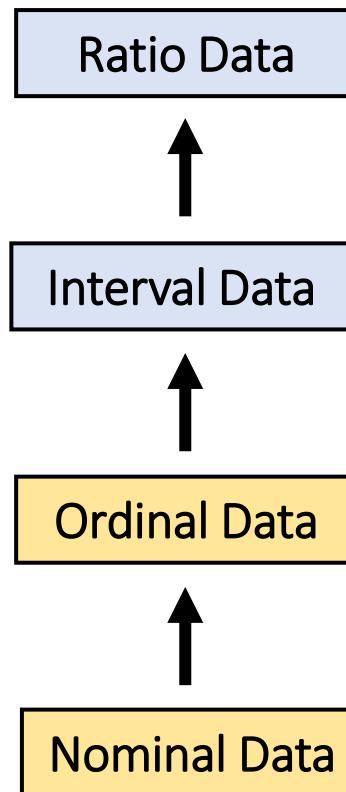
Types and scales of attributes

Differences between measurements,
true zero exists

Differences between measurements
but no true zero

Ordered Categories
(rankings, order, or scaling)

Categories
(no ordering or direction)



Quantitative
Data

Qualitative
Data

Types and scales of attributes

Amount of Information
↓

Attributes		distinctness	order	meaningful differences	operations
Type	Scale	(= , ≠)	(< , >)	(+, -)	(* , /)
Categorical	Nominal	✓			
	Ordinal	✓	✓		
Numeric	Interval	✓	✓	✓	
	Ratio	✓	✓	✓	✓

Types and scales of attributes

Attribute Type	Description	Examples	Operations
Categorical Qualitative	Nominal Nominal attribute values only distinguish. ($=, \neq$)	zip codes, employee ID numbers, eye color, sex: { <i>male, female</i> }	mode, entropy, contingency correlation, χ^2 test
	Ordinal Ordinal attribute values also order objects. ($<, >$)	hardness of minerals, { <i>good, better, best</i> }, grades, street numbers	median, percentiles, rank correlation, run tests, sign tests
Numeric Quantitative	Interval For interval attributes, differences between values are meaningful. ($+, -$)	calendar dates, temperature in Celsius or Fahrenheit	mean, standard deviation, Pearson's correlation, <i>t</i> and <i>F</i> tests
	Ratio For ratio variables, both differences and ratios are meaningful. (* , /)	temperature in Kelvin, monetary quantities, counts, age, mass, length, current	geometric mean, harmonic mean, percent variation

This categorization of attributes is due to S. S. Stevens

Types and scales of attributes

	Attribute Type	Transformation	Comments
Categorical Qualitative	Nominal	Any permutation of values	If all employee ID numbers were reassigned, would it make any difference?
	Ordinal	An order preserving change of values, i.e., $new_value = f(old_value)$ where f is a monotonic function	An attribute encompassing the notion of good, better best can be represented equally well by the values {1, 2, 3} or by { 0.5, 1, 10}.
Numeric Quantitative	Interval	$new_value = a * old_value + b$ where a and b are constants	Thus, the Fahrenheit and Celsius temperature scales differ in terms of where their zero value is and the size of a unit (degree).
	Ratio	$new_value = a * old_value$	Length can be measured in meters or feet.

This categorization of attributes is due to S. S. Stevens

Discrete vs. continuous attributes

- Discrete Attribute

- finite or countably infinite set of values
- it can take only distinct or separate values
 - zip codes, counts, or the set of words
(binary attributes - special case of discrete attributes)

- Continuous Attribute

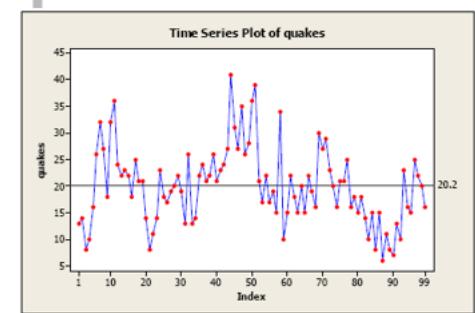
- infinite set of values
- real numbers
 - temperature, height, weight, distance

Special case: Binary attributes

- **Binary Attribute**
- Nominal attribute with only 2 states (usually: 0 and 1)
- Symmetric binary: both outcomes equally important
 - gender, sizes
- Asymmetric binary: outcomes not equally important
 - Only presence (a non-zero attribute value) is regarded as important
 - **Convention: assign 1 to most important outcome (e.g., HIV positive)**
 - medical test (positive vs. negative), failure/not failure, words present in documents, items present in customer transactions

Types of datasets

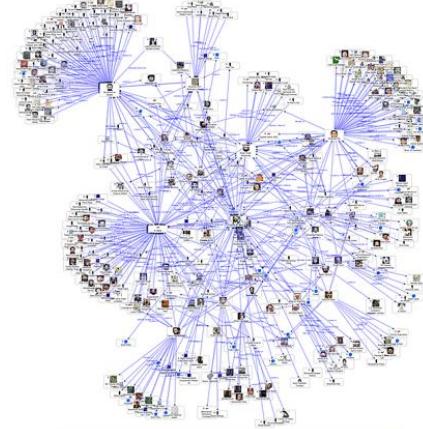
- Data tables
 - Data matrix, document data, transactional data
- Graphs and networks
 - Molecular structures, transportation networks, social networks
- Ordered data
 - temporal data (time-series), data streams genetic sequences
- Multimedia data
 - Maps, images, videos, audios
- ...



ID	Items
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

Types of datasets

- Data tables
 - Data matrix, document data, transactional data
- Graphs and networks
 - Molecular structures, transportation networks, social networks
- Ordered data
 - temporal data (time-series), data streams
- genetic sequences
 - genetic sequences
- Multimedia data
 - Maps, images, videos, audios
- ...



A social network

Start

	Human	Chimpanzee	Macaque
GTTTGGAGG	..-ATGTTCAACAAATGCTCCCTTCATTCCTCTATTACAGACCTGGCGCA		
GTTTGGAGG	-..ATGTTCAATAATGCTGCCTTCACTCTCTATTACAGACCTGGCGCA		
GTTTGGAGG	-..ATGCTCAATAATGCTCCCTTCATTCCTCCATTACAACCTGGCGCA		
GACAAATTCTGCTAGCAGCC	TTTGCTATTATCTGTGCTAAACCTTAGTAATTGAGTGT		
GACAAATTCTGCTAGCAGCC	TTTGCTATTATCTGTGCTAAACCTTAGTAATTGAGTGT		
GACAAATTCTGCTAGCAGCC	TTTGCTATTATCTGTGCTAAACCTTAGTAATTGAGTGT		
GATCTGGAGACTAA	CCTGAAATAAAATAAAGCTGATTATTATTATTCTCAAACAA		
GATCTGGAGACTAA	CCTGAAATAAAATAAAGCTGATTATTATTATTCTCAAACAA		
TATCTGGAGACTAA	TCTGAAATAAAATAAAGCTGATTATTATTATTCTCAAACAA		
GAGAAATCGATTAGCAAATTAAGCTTAAAGATATTATTTACATTCTATATCTCCTA			
GAGAAATCGATTAGCAAATTAAGCTTAAAGATATTATTTACATTCTATATCTCCTA			
GAGAAATCGATTAGCAAATTAAGCTTAAAGATATTATTTACATTCTATATCTCCTA			
CCCTGAGTTGATGTGAGCAAATAGCTACCTTCAAAAGCCAGGTATACG	..-TTATG		
CCCTGAGTTGATGTGAGCAAATAGCTACCTTCAAAAGCCAGGTATACG	..-TTATG		
CCCTGAGTTGATGTGAGCAAATAGCTACCTTCAAAAGCCAGGTATACG	..-TTATG		
GACAGGTAAGTAAAAAACATATATTATCTACGTTTGTGCAAAATTAAATTTC	H I Y S T F L S K		
GACAGGTAAGTAAAAAACATATATTATCTACGTTTGTGCAAAATTAAATTTC	H I Y S T F L S K		
GACAGGTAAGTAAAAACATATATTATCTACGTTTGTGCAAAATTAAATTTC	H I Y S T F L S K		
AACTGTTGCGCGTGTGTTGGAA	-..TGTAAAAACAAACTCAGTACA		
AACTGTTGCGCGTGTGTTGGAA	-..TGTAAAAACAAACTCAGTACA		
AACTGTTGCGCGTGTGTTGGAA	-..TGTAAAAACAAACTCAGTACA		
AACTGTTGCGCGTGTGTTGGAA	-..CGTAAAAACAAATTCACTAOG		

Types of datasets

- Data tables
 - Data matrix, document data, transactional data
- Graphs and networks
 - Molecular structures, transportation networks, social networks
- Ordered data
 - temporal data (time-series), data streams genetic sequences
- Multimedia data
 - Maps, images, videos, audios
- ...

Data matrix:

Attributes

Id	Age	Height	Sex	Marital Status	Ec. Status
1	56	182	masc	single	low inc.
2	43	176	masc	married	high inc.
3	53	164	fem	married	high inc.
4	47	171	fem	widowed	high inc.
5	52	170	fem	widowed	low inc.
6	49	169	fem	married	middle inc.
7	51	173	fem	married	middle inc.
8	48	173	fem	single	middle inc.
9	47	179	masc	married	low inc.
10	39	183	masc	single	low inc.
11	58	165	fem	married	low inc.
12	57	184	masc	married	middle inc.

Observations

Document data:

	course	department	university	class	student
Doc 1	3	6	4	0	8
Doc 2	0	5	3	2	3
Doc 3	0	1	0	1	0

Important characteristics of datasets

- Dimensionality
 - high dimensional data brings several challenges
 - Curse of dimensionality
- Size
 - type of analysis may depend on size of data
- Sparsity
 - only presence counts
- Resolution
 - patterns depend on the scale
 - *motivation for data preparation: next classes...*

Data matrix

Data represented as a *matrix* with N rows and D columns

$$\left(\begin{array}{c|cccc} & X_1 & X_2 & \dots & X_D \\ \hline x_1^T & x_{11} & x_{12} & \dots & x_{1D} \\ x_2^T & x_{21} & x_{22} & \dots & x_{2D} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_N^T & x_{N1} & x_{iN2} & \dots & x_{ND} \end{array} \right)$$

- **Rows:** Also called *instances, examples, records, transactions, objects, points, feature-vectors*, etc. Given as a D -tuple¹

$$x_i^T = [x_{i1} \quad x_{i2} \quad \dots \quad x_{iD}]^T = (x_{i1} \quad x_{i2} \quad \dots \quad x_{iD})$$

- **Columns:** Also called *attributes, properties, features, dimensions, variables, fields*, etc. Given as an N -tuple

$$X_j = \begin{bmatrix} x_{1j} \\ x_{2j} \\ \dots \\ x_{Nj} \end{bmatrix} = [x_{1j} \quad x_{2j} \quad \dots \quad x_{Nj}]$$

¹the class label usually is excluded of this description (x_i, label)

Data matrix

Data represented as a *matrix* with N rows and D columns

$$\left(\begin{array}{c|cccc} & X_1 & X_2 & \dots & X_D \\ \hline \mathbf{x}_1^T & x_{11} & x_{12} & \dots & x_{1D} \\ \mathbf{x}_2^T & x_{21} & x_{22} & \dots & x_{2D} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}_N^T & x_{N1} & x_{iN2} & \dots & x_{ND} \end{array} \right)$$

Feature Vector \mathbf{X}

- the $i - th$ feature is X_i

Feature Space: dimension is equal to the number of features.

- with $i - th$ coordinate related to $i - th$ feature

Data: algebraic and geometric view

For numeric data matrix \mathbf{X} :

- each row or point is a D -dimensional ² row vector

$$\mathbf{x}_i^T = [x_{i1} \quad x_{i2} \quad \dots \quad x_{iD}]^T = (x_{i1} \ x_{i2} \ \dots \ x_{iD}) \in \mathbb{R}^D$$

- each column or attribute is a N -dimensional column vector

$$X_j = \begin{bmatrix} x_{1j} \\ x_{2j} \\ \dots \\ x_{Nj} \end{bmatrix} = [x_{1j} \ x_{2j} \ \dots \ x_{Nj}] \in \mathbb{R}^N$$

² Classification problems: the class label is not included

Feature vector and feature space

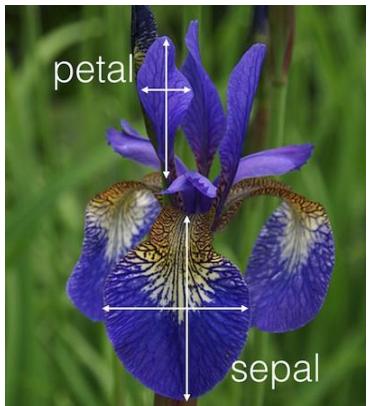
Feature vector

$$\mathbf{x} = [x_1 \quad x_2 \quad \dots \quad x_D]^T = (x_1 \ x_2 \ \dots \ x_D)$$

- Each feature represents one dimension, and its values represent positions along one of the orthogonal coordinate axes in **feature space**

Feature space: set of all possible values for a chosen set of features from that data.

Example: Iris Dataset



	Sepal length	Sepal width	Petal length	Petal width	Class
	x_1	x_2	x_3	x_4	x_5
x_1	5.9	3.0	4.2	1.5	Iris-versicolor
x_2	6.9	3.1	4.9	1.5	Iris-versicolor
x_3	6.6	2.9	4.6	1.3	Iris-versicolor
x_4	4.6	3.2	1.4	0.2	Iris-setosa
x_5	6.0	2.2	4.0	1.0	Iris-versicolor
x_6	4.7	3.2	1.3	0.2	Iris-setosa
x_7	6.5	3.0	5.8	2.2	Iris-virginica
x_8	5.8	2.7	5.1	1.9	Iris-virginica
:	:	:	:	:	:
x_{149}	7.7	3.8	6.7	2.2	Iris-virginica
x_{150}	5.1	3.4	1.5	0.2	Iris-setosa



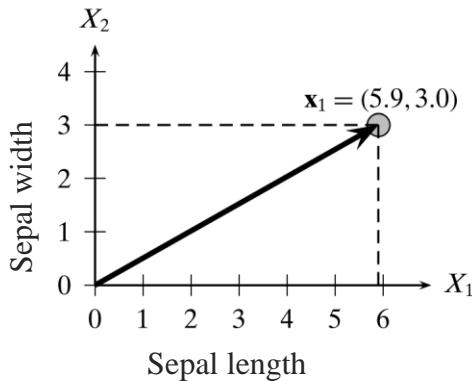
Example: Iris Dataset

Feature vector

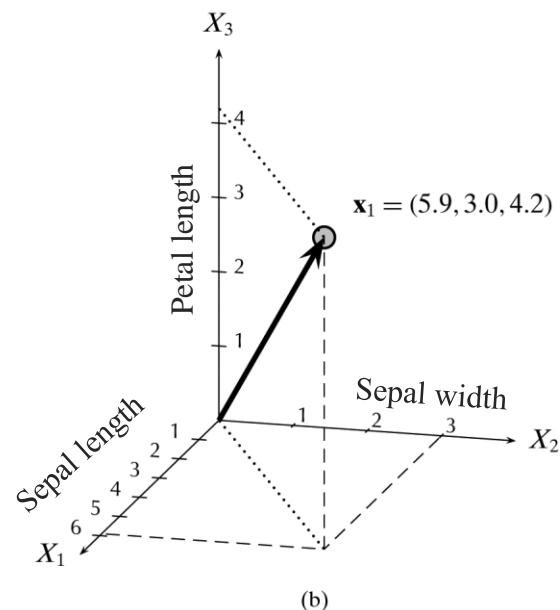
$$\mathbf{x} = [x_1 \quad x_2 \quad \dots \quad x_D]^T = (x_1 \ x_2 \ \dots \ x_D)$$

Feature space: set of all possible values for a chosen set of features from that data

- Visualization : $x_1 = (5.9; 3.0; 4.2; 1.5)$ in 2D and 3D



(a)



(b)

	Sepal length x_1	Sepal width x_2	Petal length x_3	Petal width x_4	Class x_5
x_1	5.9	3.0	4.2	1.5	Iris-versicolor
x_2	6.9	3.1	4.9	1.5	Iris-versicolor
x_3	6.6	2.9	4.6	1.3	Iris-versicolor
x_4	4.6	3.2	1.4	0.2	Iris-setosa
x_5	6.0	2.2	4.0	1.0	Iris-versicolor
x_6	4.7	3.2	1.3	0.2	Iris-setosa
x_7	6.5	3.0	5.8	2.2	Iris-virginica
x_8	5.8	2.7	5.1	1.9	Iris-virginica
⋮	⋮	⋮	⋮	⋮	⋮
x_{149}	7.7	3.8	6.7	2.2	Iris-virginica
x_{150}	5.1	3.4	1.5	0.2	Iris-setosa

Homework...

- Assignment I (see eLearning)
 - Data Understanding:
 - Hands on: Datasets and attributes

Contents

- Attributes and Datasets
- Data Summarization
- Data Visualization
- Proximity measures
- Summary

Data summarization

Common questions in data analysis

- What is the most common value?
- What is the variance in the values?
- Are there strange values?

Choosing the appropriate data analysis depends on

- number of variables/attributes: univariate or multivariate
- type of variables/attributes: categorical or numeric

Data summarization

Motivation

- Big datasets: it's hard to understand data
- Data summaries: useful synopsis of key properties of data
- To **better understand** the data:
 - Frequency
 - Central tendency
 - Dispersion / Spread

Describe important properties of data distribution

Data summarization

Summary statistics for data exploration

better understand the data

- Frequency
- Central tendency
- Dispersion / Spread

Data summarization: univariate data

Frequency*

- Absolute (or relative) occurrence of each value
 - e.g. nr. of days by outlook (year)

sunny	overcast	rainy
198	65	102
54.2%	17.8%	27.9%

- e.g. exam grades

8	10	13	14	17	19
1	2	9	8	3	1
4.2%	8.3%	37.5%	33.3%	12.5%	4.2%

*For both categorical and numeric variables/attributes (typically used for categorical data)

Data summarization

Central tendency

- Mean: the average value (sensitive to extremes)
- Quartiles
 - Median: the "middle" value (from a sorted list of values)
- Mode*: the most frequent value

*For both categorical and numeric variables/attributes

Data summarization: univariate data

Central tendency

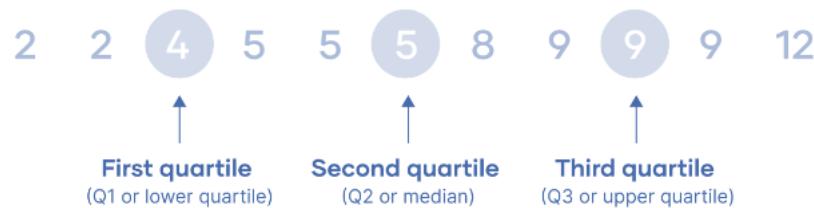
- Mean: the average value (sensitive to extremes)

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

Data summarization: univariate data

Central tendency: Quartiles

- Q_1 (1st quartile or the lower quartile)
 - value halfway between the lowest value and the middle value
- Q_2 (2nd quartile or the median)
 - value halfway between the lowest value and the highest value
- Q_3 (3rd quartile or the upper quartile)
 - value halfway between the middle value and the highest value



Data summarization: univariate data

Central tendency

- Median: the "middle" value (from a sorted list of values)
 - Middle value if odd number of values, or average of the middle two values otherwise
 - Estimated by interpolation (for *grouped data*):

Approximate
median

$$\text{median} = L_1 + \left(\frac{n/2 - (\sum freq)_l}{freq_{median}} \right) width$$

Sum before the median interval

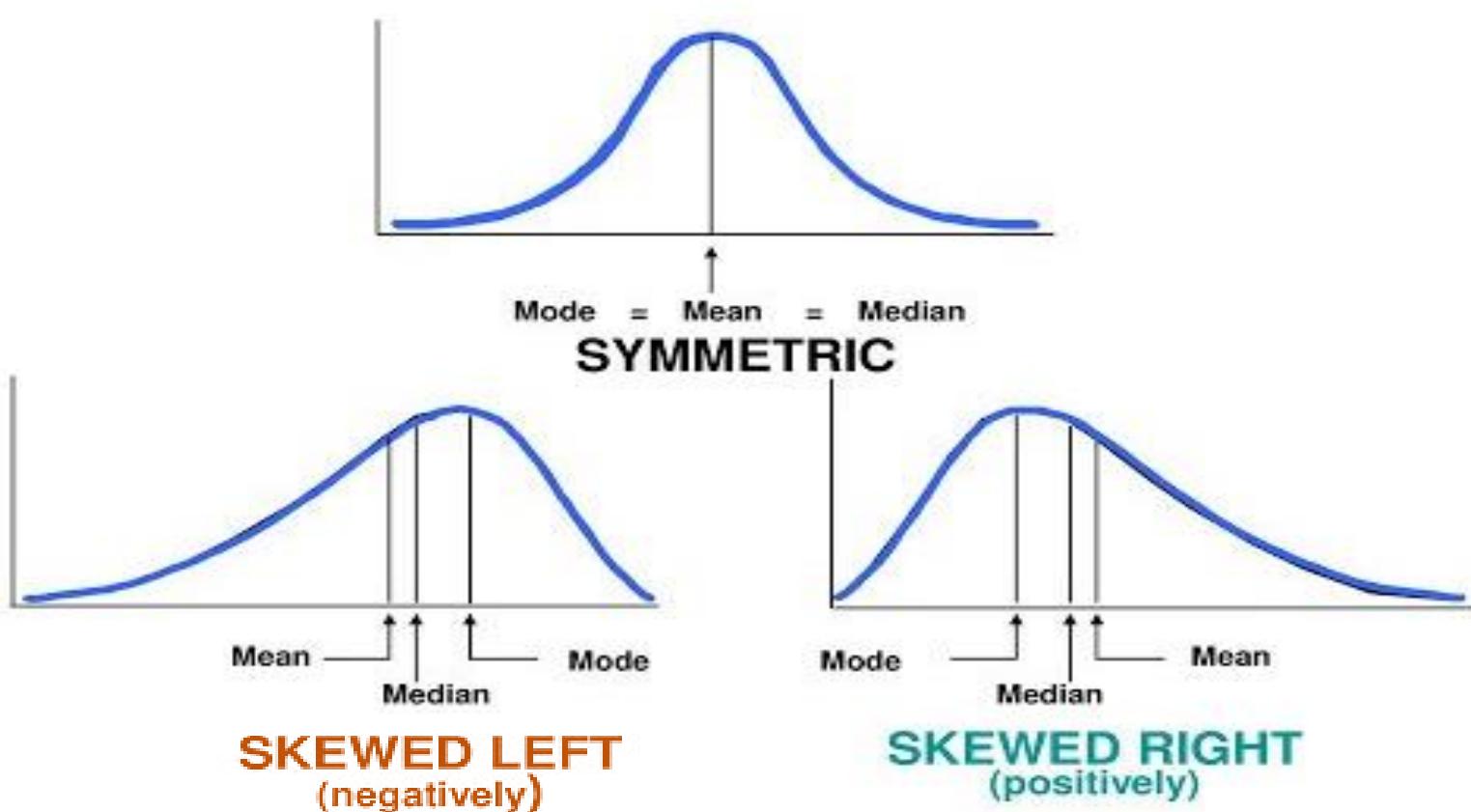
Low interval limit

Interval width ($L_2 - L_1$)

age	frequency
1–5	200
6–15	450
16–20	300
21–50	1500
51–80	700
81–110	44

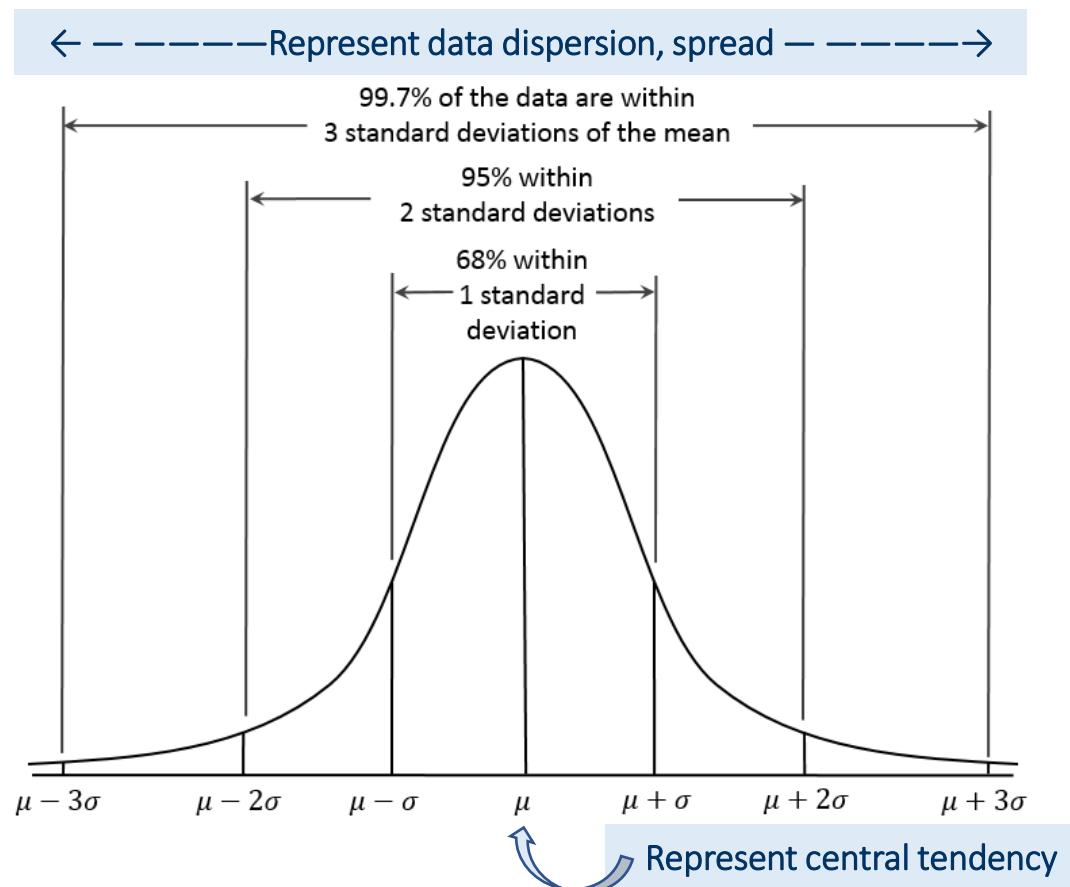
Data summarization: univariate data

Symmetric vs. Skewed data



Data summarization: univariate data

Properties of normal distribution curve



Data summarization: univariate data

Dispersion (spread)

- Inter-quartile range: $IQR = Q_3 - Q_1$
- Range: $\max_x - \min_x$
- Median/Average absolute deviation
- Standard deviation: sensitive to extreme values
- Variance: sensitive to extreme values

Data summarization: univariate data

Dispersion (spread)

- Median absolute deviation

$$MAD = median(x_i - \bar{x})^2$$

- Average absolute deviation

$$AAD = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2$$

Data summarization: univariate data

Dispersion (spread)

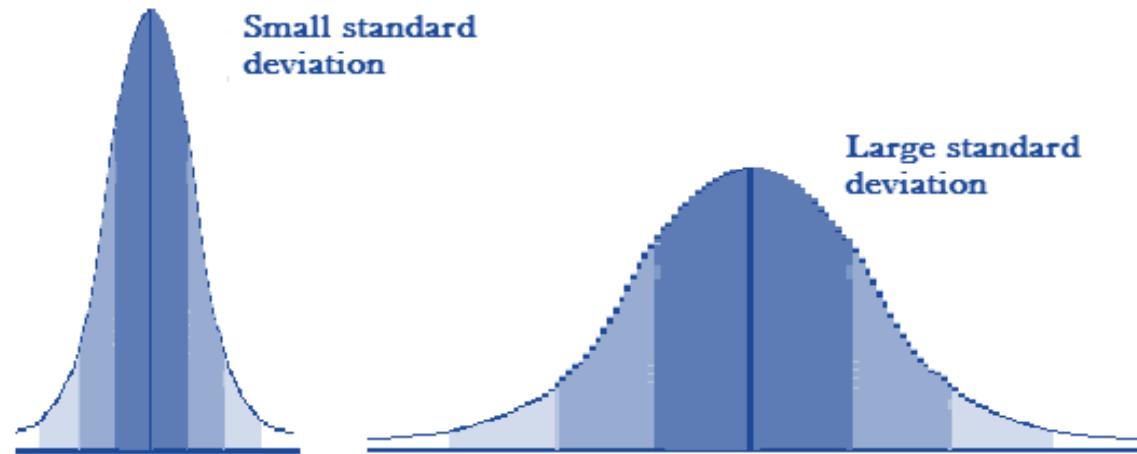
- Standard deviation

$$s_X = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2}$$

- Variance: s_X^2

Data summarization: univariate data

Dispersion (spread)



Data summarization: multivariate data

Frequency*

- Contingency tables: cross-frequency of values for two variables

Season and outlook (year)

	winter	spring	summer	autumn
sunny	15	62	106	37
rainy	57	23	22	34

- Summarize the relationship between pairs of variables in a data set
- Gives the number of occurrences of $X = b_i$ and $Y = a_j$ in the data set.

*For both categorical and numeric variables/attributes (typically used for categorical data)

Data summarization: multivariate data

Dispersion (spread)

- **Covariance matrix:** variance between every pair of numeric variables
 - the value depends on the magnitude of the variable

$$cov(x, y) = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})$$

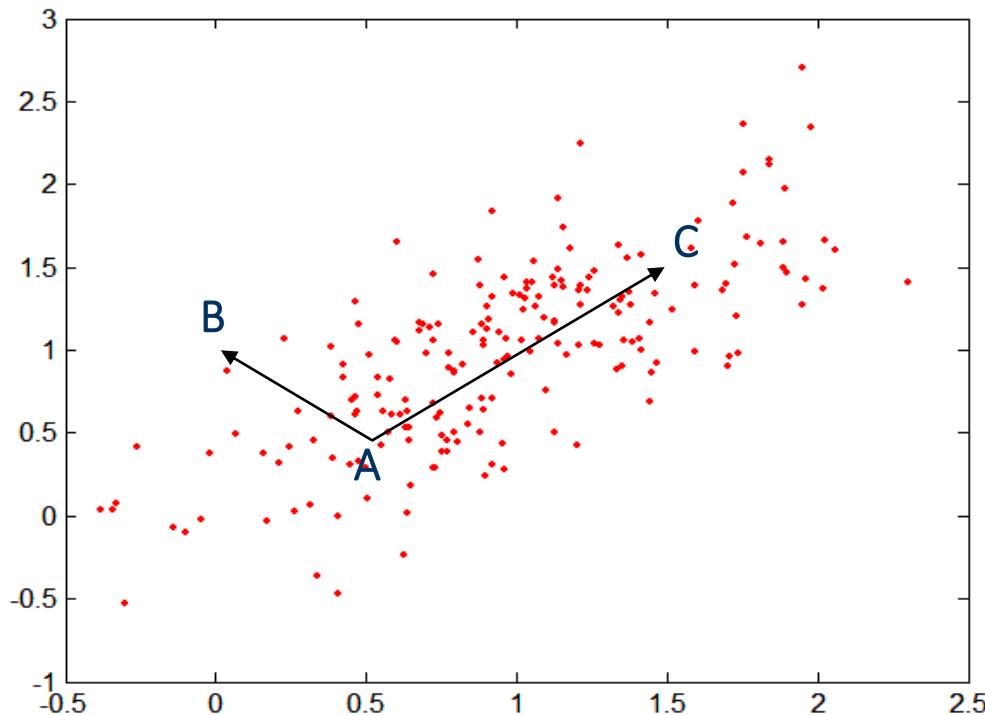
$$\Sigma = \begin{pmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1d} \\ \sigma_{21} & \sigma_2^2 & \cdots & \sigma_{2d} \\ \cdots & \cdots & \cdots & \cdots \\ \sigma_{d1} & \sigma_{d2} & \cdots & \sigma_d^2 \end{pmatrix}$$

- non-diagonal entries represent the covariance between pairs of variables
- diagonal is the variance of the variables
- matrix is symmetric

Data summarization: multivariate data

Dispersion (spread)

- Covariance matrix



$$\Sigma = \begin{bmatrix} 0.58 & 0.25 \\ 0.25 & 0.25 \end{bmatrix}$$

A: (0.5, 0.5)

B: (0, 1)

C: (1.5, 1.5)

Data summarization: multivariate data

Association (relationship)

- Correlation matrix: correlation between every pair of numeric variables (interval or scale)
 - the influence of the magnitude is removed

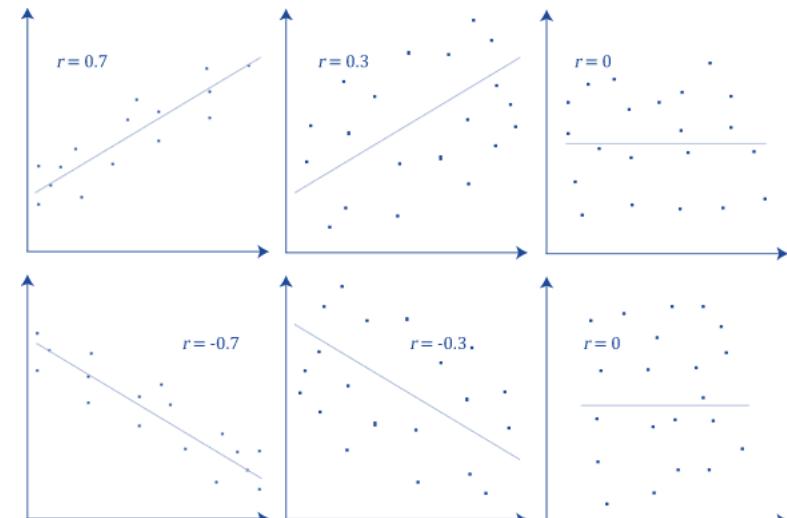
$$\text{corr}(x, y) = \frac{\text{cov}(x, y)}{\text{std}(x)\text{std}(y)}$$

$$-1 \leq \text{corr}(x, y) \leq 1$$

Data summarization: multivariate data

- Pearson's correlation coefficient (ρ_{xy})
 - Measures the linear correlation between a pair of **numeric** variables (interval or scale)
$$-1 \leq \rho_{xy} \leq 1$$
 - measures the strength and direction of the linear relationship between a pair of variables

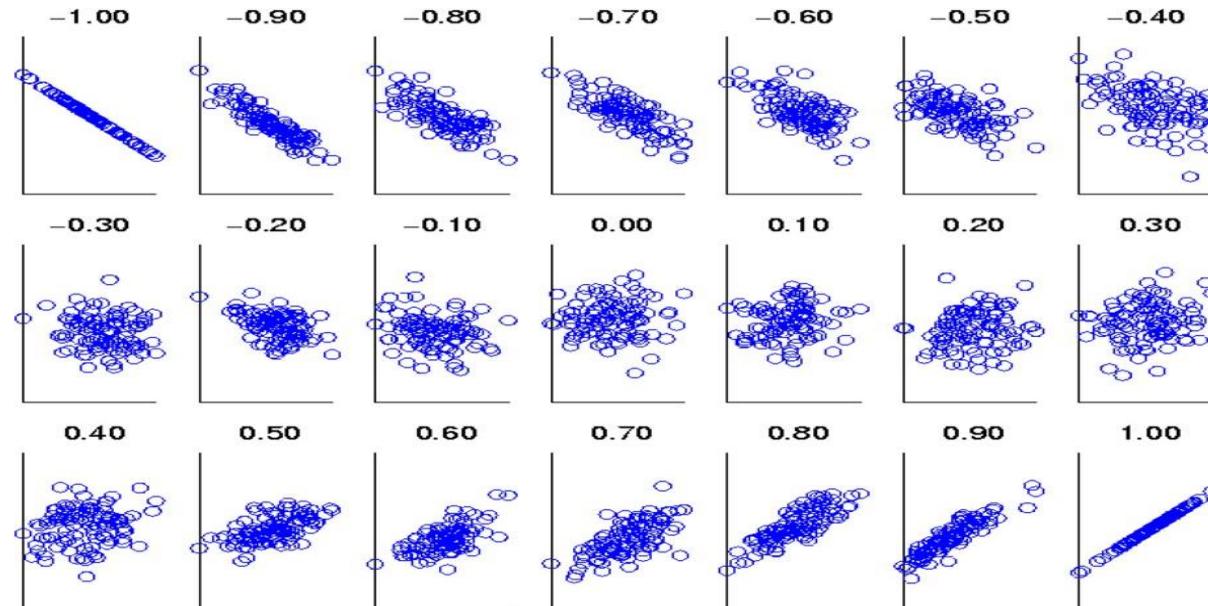
$$\rho_{xy} = \text{corr}(x, y) = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}}$$



Data summarization: multivariate data

- Pearson's correlation coefficient (ρ_{xy})
 - Measures the linear correlation between a pair of **numeric** variables (interval or scale)
$$-1 \leq \rho_{xy} \leq 1$$

Scatter plots of pair of variables / features / attributes



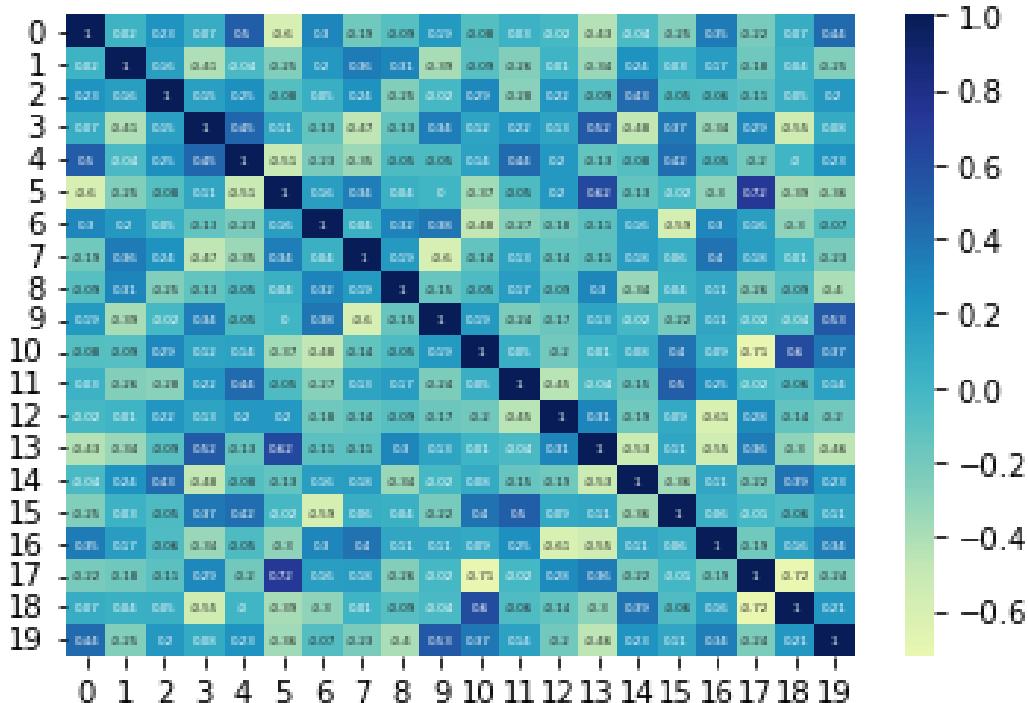
Data summarization: multivariate data

- Pearson's correlation coefficient (ρ_{xy})

- Measures the linear correlation between a pair of **numeric** variables (interval or scale)

$$-1 \leq \rho_{xy} \leq 1$$

Heatmaps



Data summarization: multivariate data

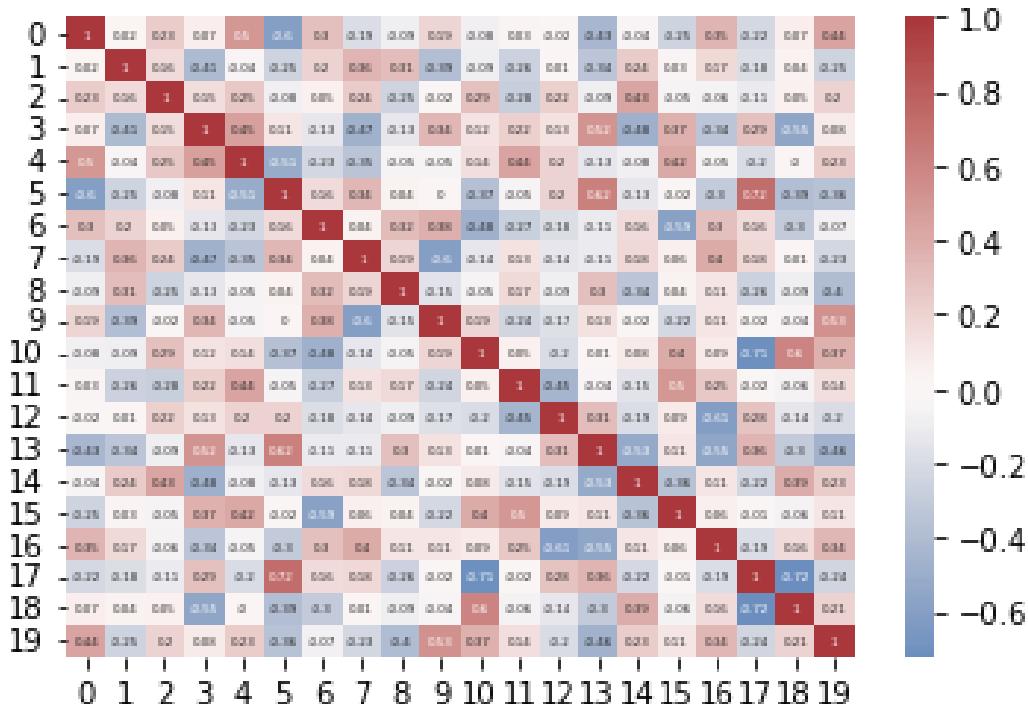
- Pearson's correlation coefficient (ρ_{xy})

- Measures the linear correlation between a pair of numeric variables (interval or scale)

$$-1 \leq \rho_{xy} \leq 1$$

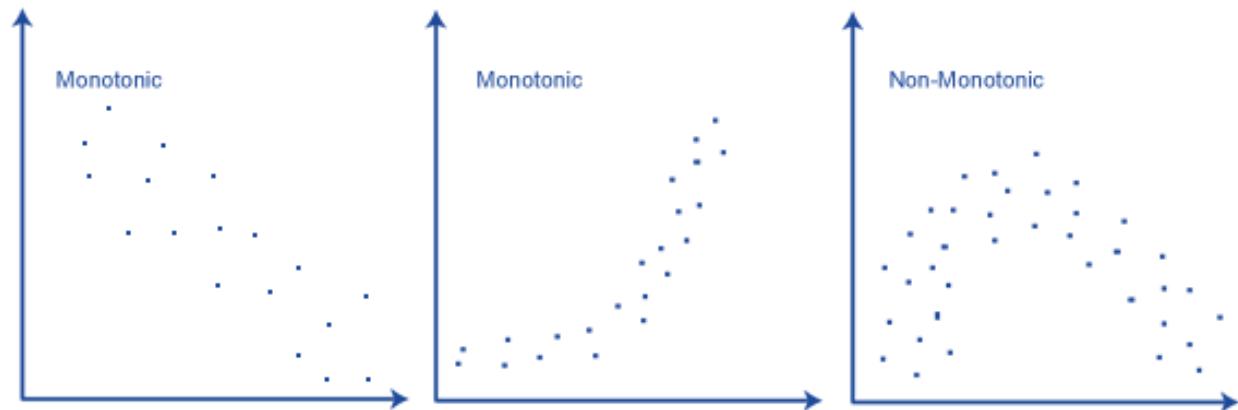
Heatmaps

Easier interpretation



Data summarization: multivariate data

- Spearman's rank-order correlation coefficient (rs_{xy})
 - Measures the monotonic association between a pair of variables
$$-1 \leq rs_{xy} \leq 1$$
 - two variables can be related according to a type of non-linear but still monotonic relationship
 - measures the strength and direction of the monotonic relationship between a pair of variables



Data summarization: multivariate data

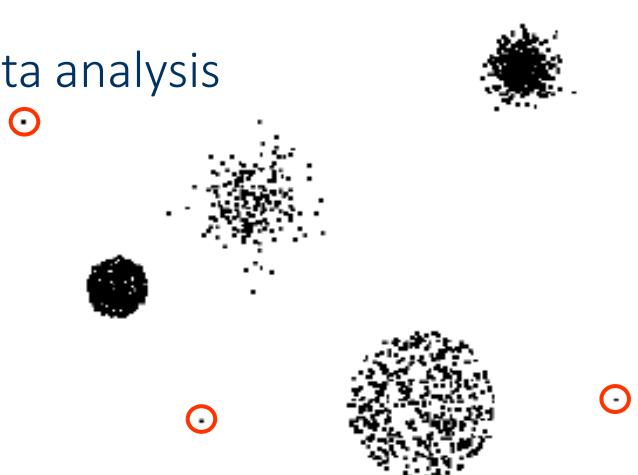
- Spearman's rank-order correlation coefficient (rs_{xy})
 - Measures the monotonic association between a pair of variables
$$-1 \leq rs_{xy} \leq 1$$
 - rank-based and non-parametric version of Pearson correlation coefficient
 - Ordinal variables
 - Numerical variables (when the assumptions for Pearson coefficient are violated)

$$rs_{xy} = \rho_{rank_x rank_y}$$

Data summarization: outliers

"An outlier is a point that deviates so much from the other data points as to arouse suspicions that it was generated by a different mechanism" (Hawkins, 1980)"
Hawkins, 1980

- **Outliers** can be univariate or multivariate
- **Outliers** are data objects with characteristics that are considerably different than most of the other data objects in the data set
- **Case 1:** Outliers are noise that interferes with data analysis
- **Case 2:** Outliers are the goal of our analysis
 - Credit card fraud
 - Intrusion detection



Data summarization: outliers

"An outlier is a point that deviates so much from the other data points as to arouse suspicions that it was generated by a different mechanism" (Hawkins, 1980)"
Hawkins, 1980

- Statistical Parametric Techniques:
 - univariate case: boxplot definition (Tukey, 1977) is the most used:
any value outside the interval $[Q_1 - 1.5 \text{ IQR}, Q_3 + 1.5 \text{ IQR}]$
 - multivariate case: Mahalanobis distance (Mahalanobis, 1936).
- Statistical Non-parametric Techniques
 - Kernel functions
 - (...)

Homework...

- Assignment I (see eLearning)
 - Data Understanding:
 - Hands on: Summarization
 - Notes with exercises: Ex. 1.1 and 1.2

Contents

- Attributes and Datasets
- Data Summarization
- Data Visualization: Amounts, Distributions, Associations, Trends
- Proximity measures
- Summary

Data visualization: what?

Visualization is the conversion of data into a visual format so that the characteristics of the data and the relationships among data items or variables can be analyzed or reported.

Main types of visualization

- Amounts
- Associations
- Geospatial data
- Distributions
- Trends
- Uncertainty

Main types of visualization techniques

- Pie charts
- Bar plots
- Histograms
- Density plots
- Scatter plots
- QQ plots
- Heatmaps
- Boxplots
- Parallel coordinates
- Violin plots
- Correlograms
- (...)

Data visualization: why?

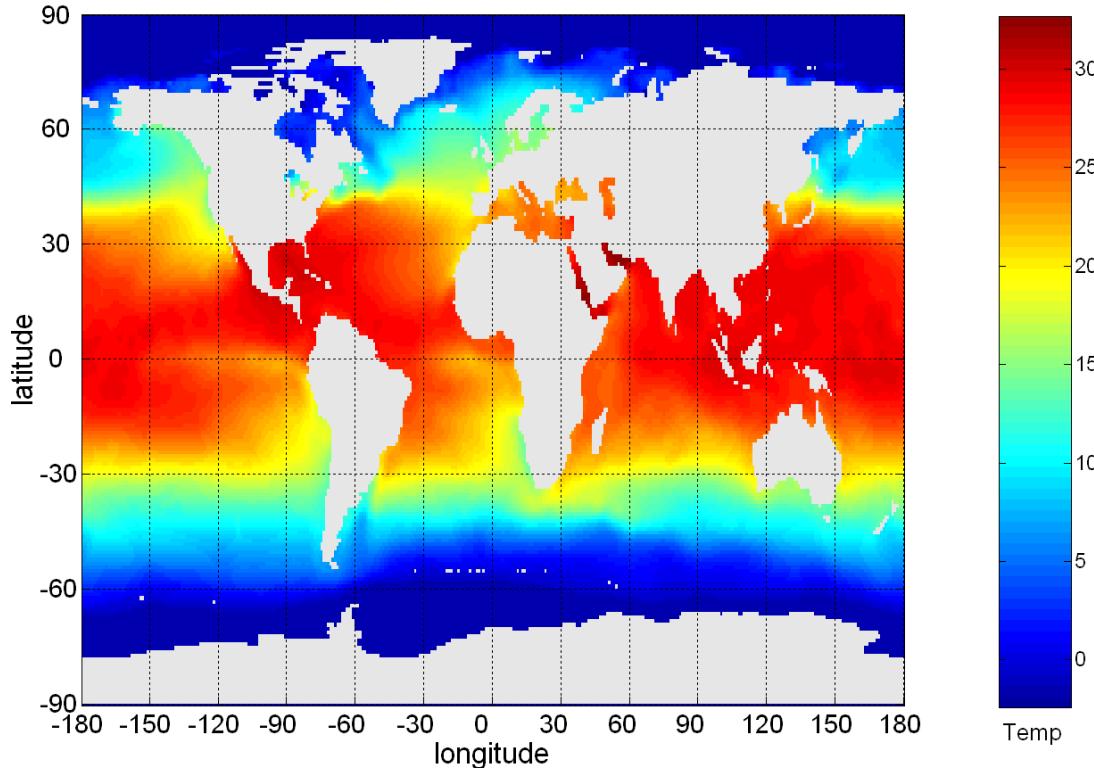
Visualization of data is one of the most powerful and appealing techniques for **data exploration**:

- Provide **graphical display** of basic description and **qualitative overview** of large data sets
- Humans have a **well developed ability** to analyze large amounts of information that is presented visually
- **Gain insight** into an information space
- Help **detecting patterns**, trends, structure, irregularities, relationships among data
- Help **detecting** unusual **patterns**, outliers, irregularities
- Help find **interesting regions and suitable parameters** for further quantitative analysis

Data visualization: example

The following map shows the **Sea Surface Temperature (SST)**

- Thousands of data points are summarized in a single figure

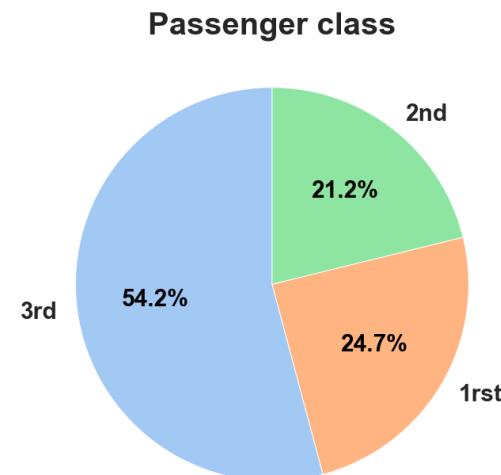
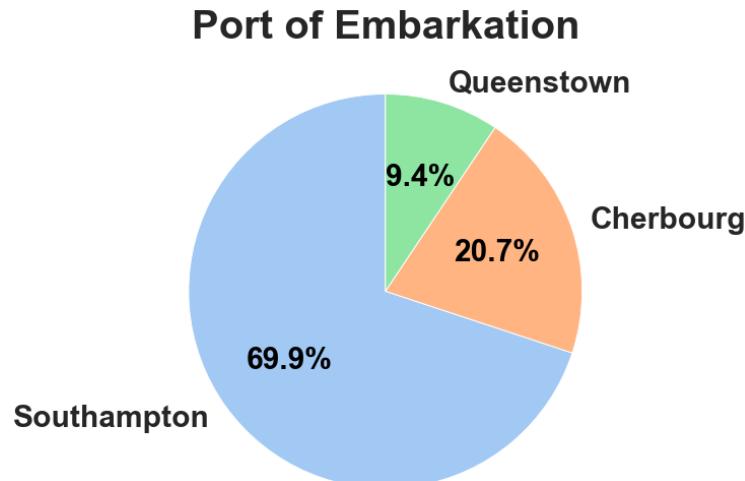


Data visualization: amounts

Pie charts

Typically used with categorical variables / attributes

- The slices correspond to categories:
 - each slice of the pie chart represents a **value** of the categorical variable
 - each slice of the pie chart display the **relative frequency** of that value



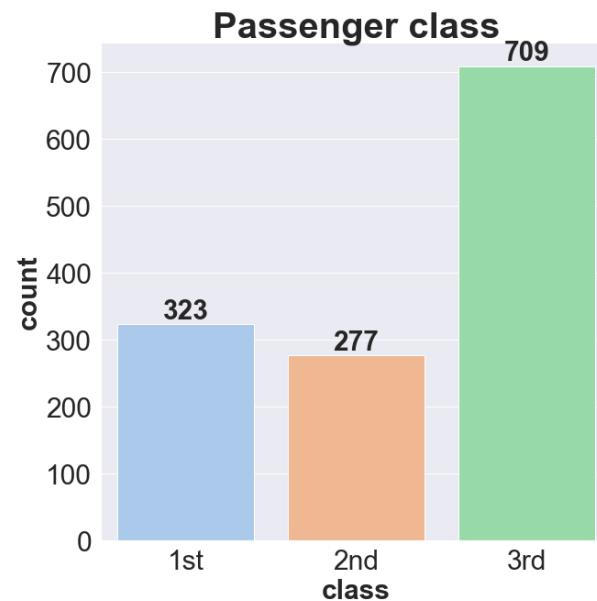
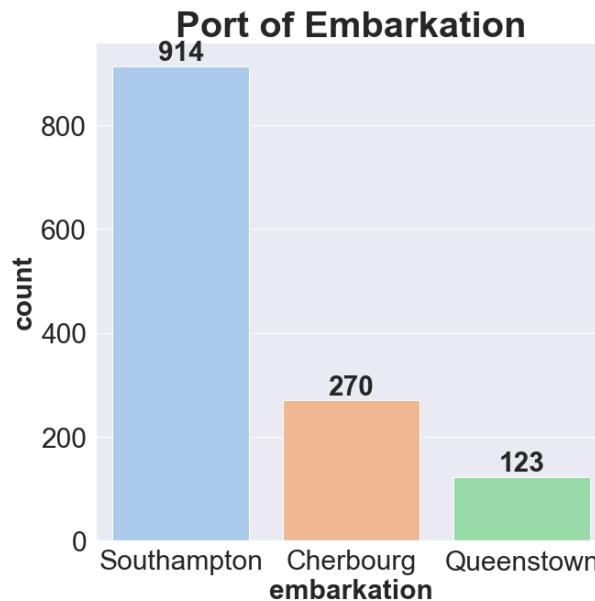
- Pie charts effectively illustrate the **different sizes for parts of the whole**
- **Not a good** option for **comparative** purposes.

Data visualization: amounts

Bar plots

Typically used with categorical variables / attributes

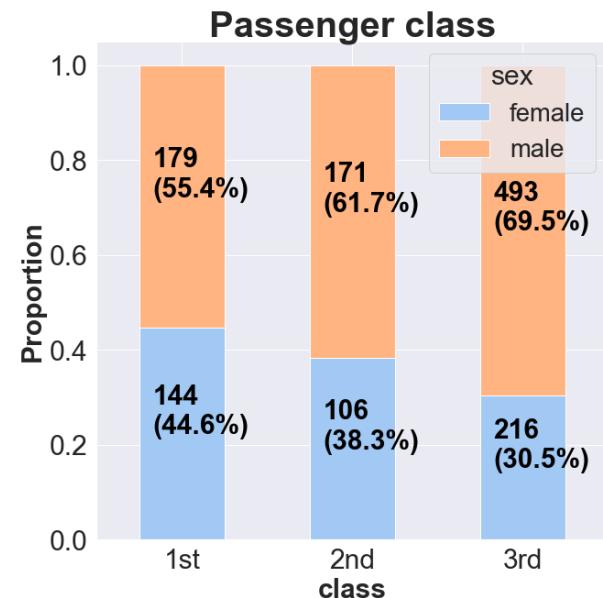
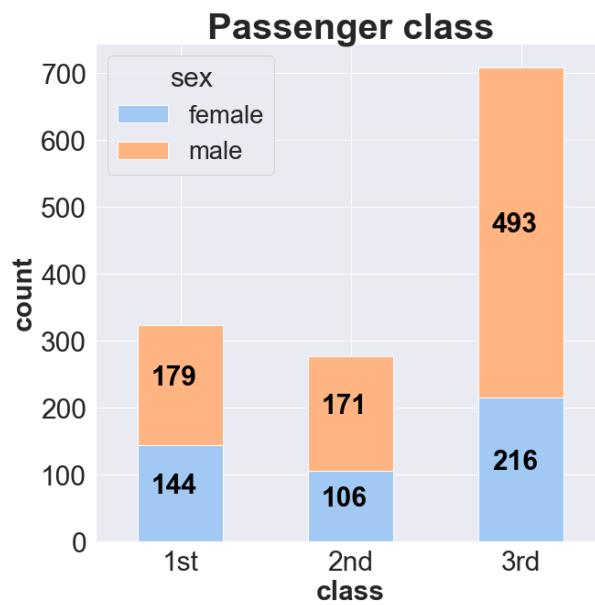
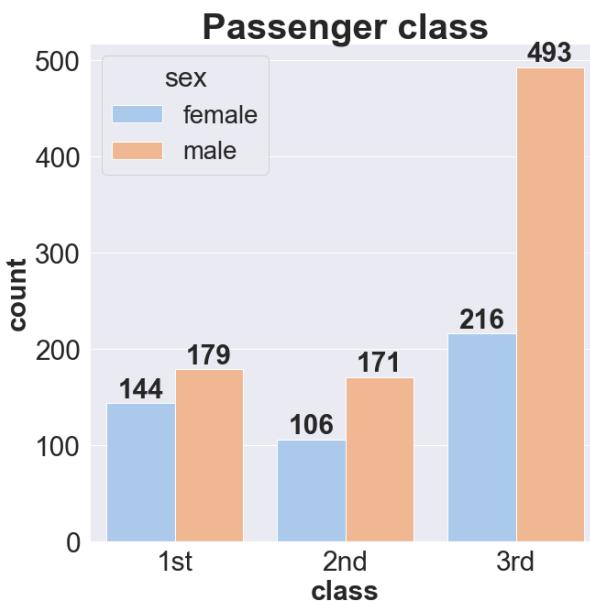
- The bars correspond to categories:
 - each bar represents a **value** of the categorical variable
 - The height of each bar display the **relative frequency** of that value



Data visualization: amounts

Bar plots with two variables

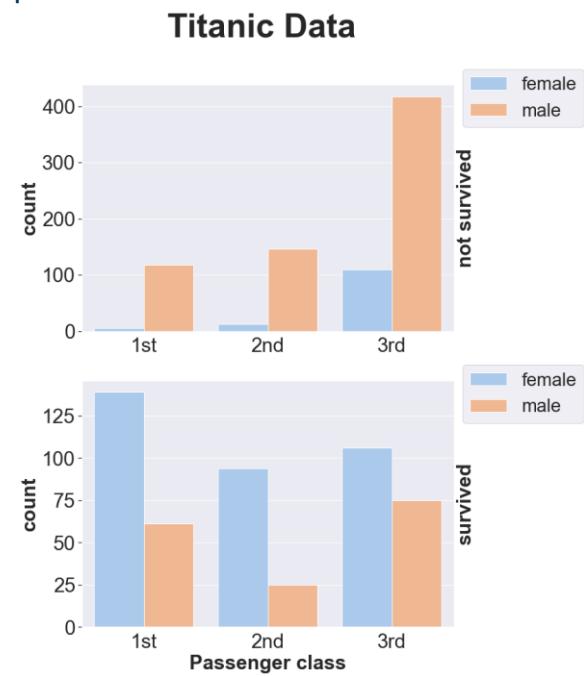
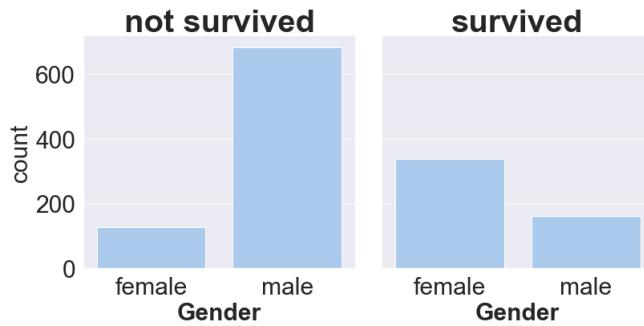
- Grouped
- Stacked
- Percent stacked



Data visualization: amounts

Bar plots w.r.t. other variables

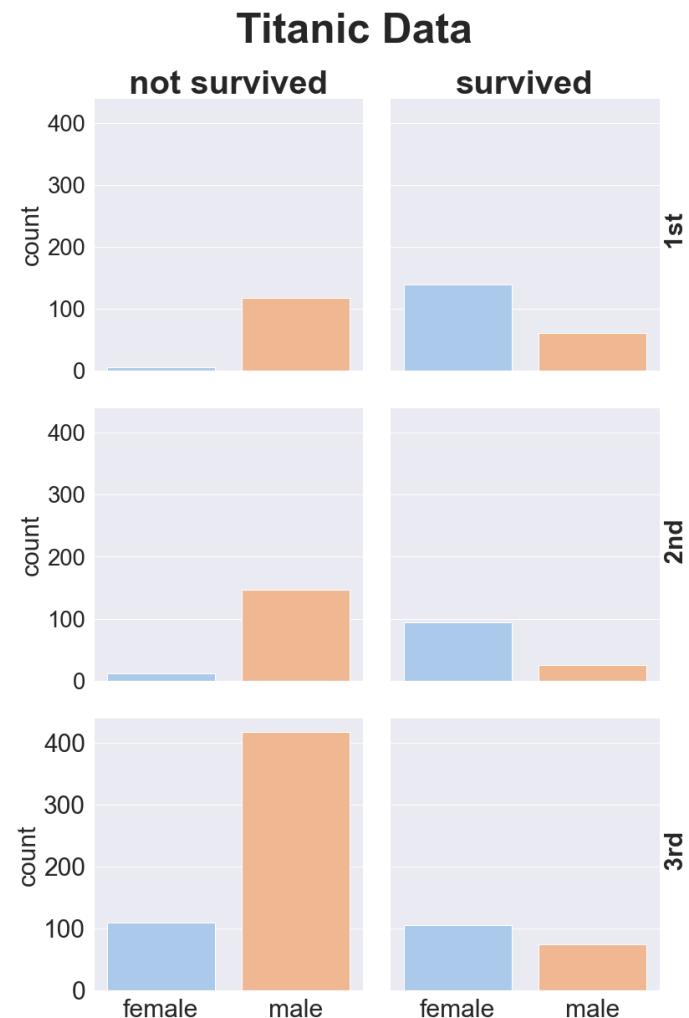
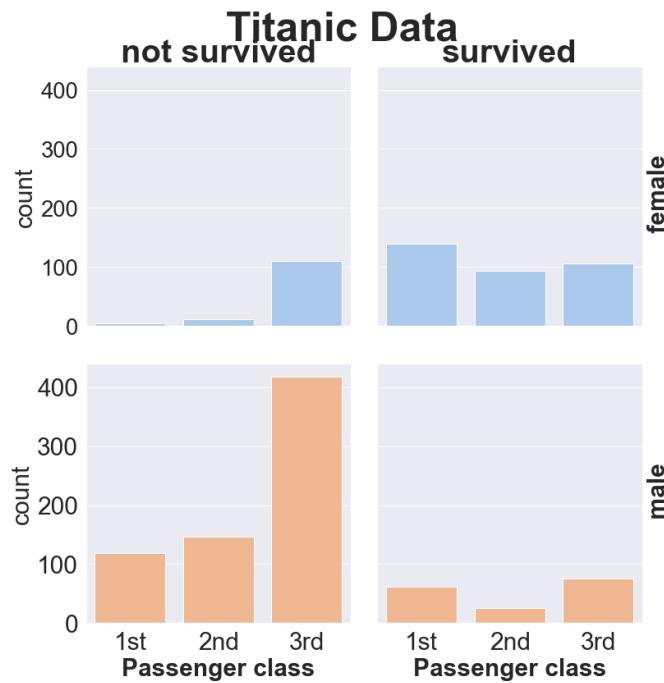
- conditional plots
 - display the distribution of a variable (or the relationship between multiple variables) separately within **sub-groups** of the data set
 - allow finding **eventual differences** between the sub-groups
- can be drawn with up to three dimensions:
 - **row, col**
 - **hue (third dimension)**
 - different levels are plotted with different colors)



Data visualization: amounts

Bar plots w.r.t. other variables

- conditional plots



Data visualization: distributions

Histograms

Show how the values of **continuous** variables are distributed

- Usually shows the distribution of values of a single variable
- Give an idea of the shape of the data

Constructing a histograms

- The range of the variable is divided into a set of *bins* (intervals of values)
- The number of occurrences of values in each bin is counted
- A bar with this number is plotted
 - The height of each bar indicates the number of occurrences
- The shape of histogram depends on the number of bins

Data visualization: distributions

Histograms

- Relative (frequency) histograms: counts are replaced by the relative frequency
- Equal-width: all bins have the same width
- Equal-frequency: all bins have the same frequency

Disadvantages of histograms

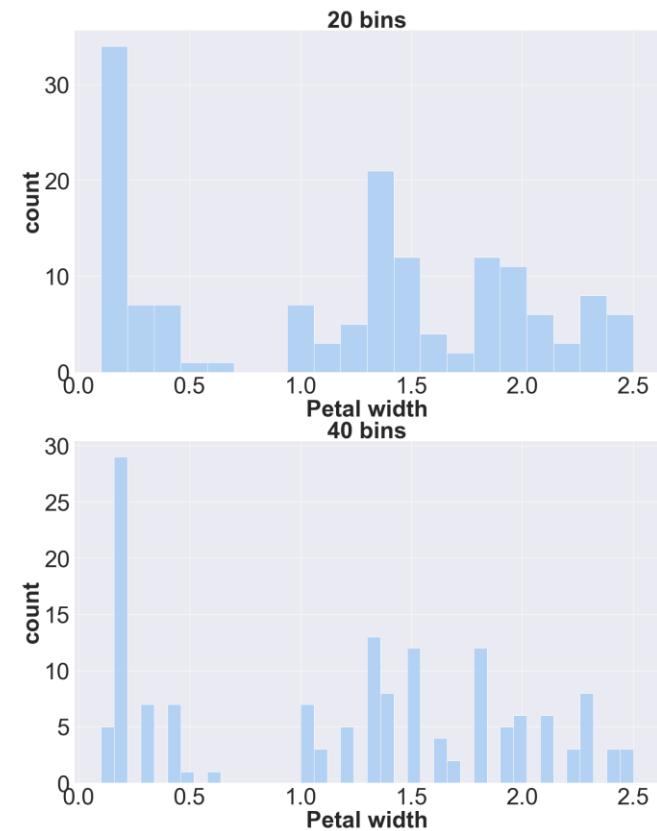
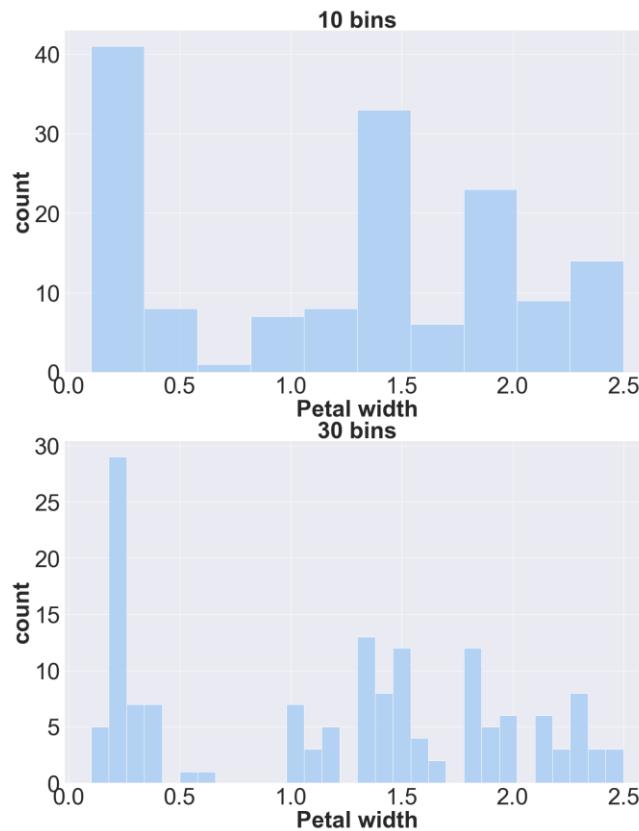
- Requires to choose the number of bins (several algorithms for that)
- Shape depends on the number of bins
- Can be misleading
 - Small data sets
 - Intervals can hide individual values (difficult to detect relevant values)
- Hard to compare several distributions

Data visualization: distributions

Histograms

- Shape depends on the number of bins

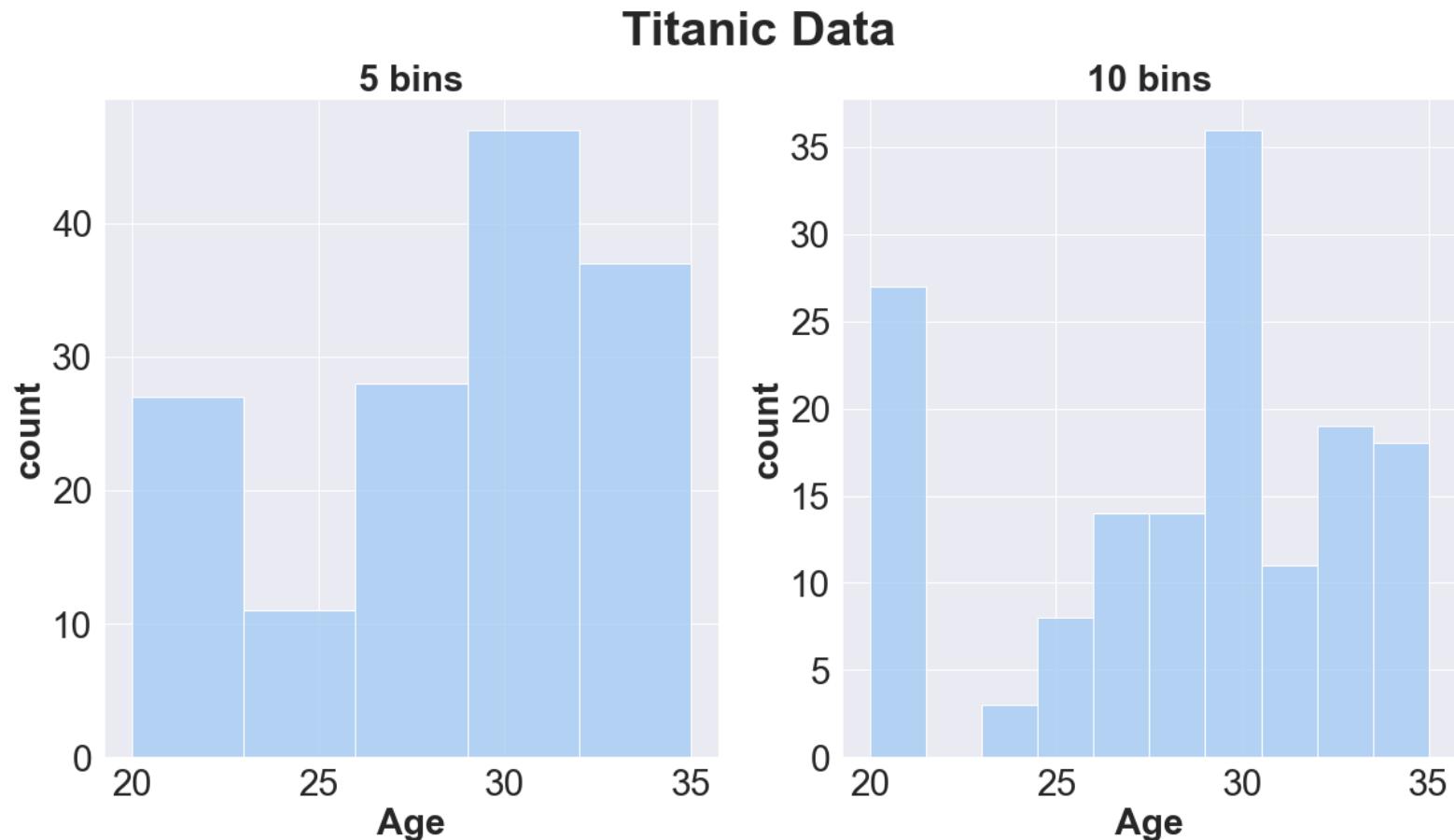
Iris Dataset



Data visualization: distributions

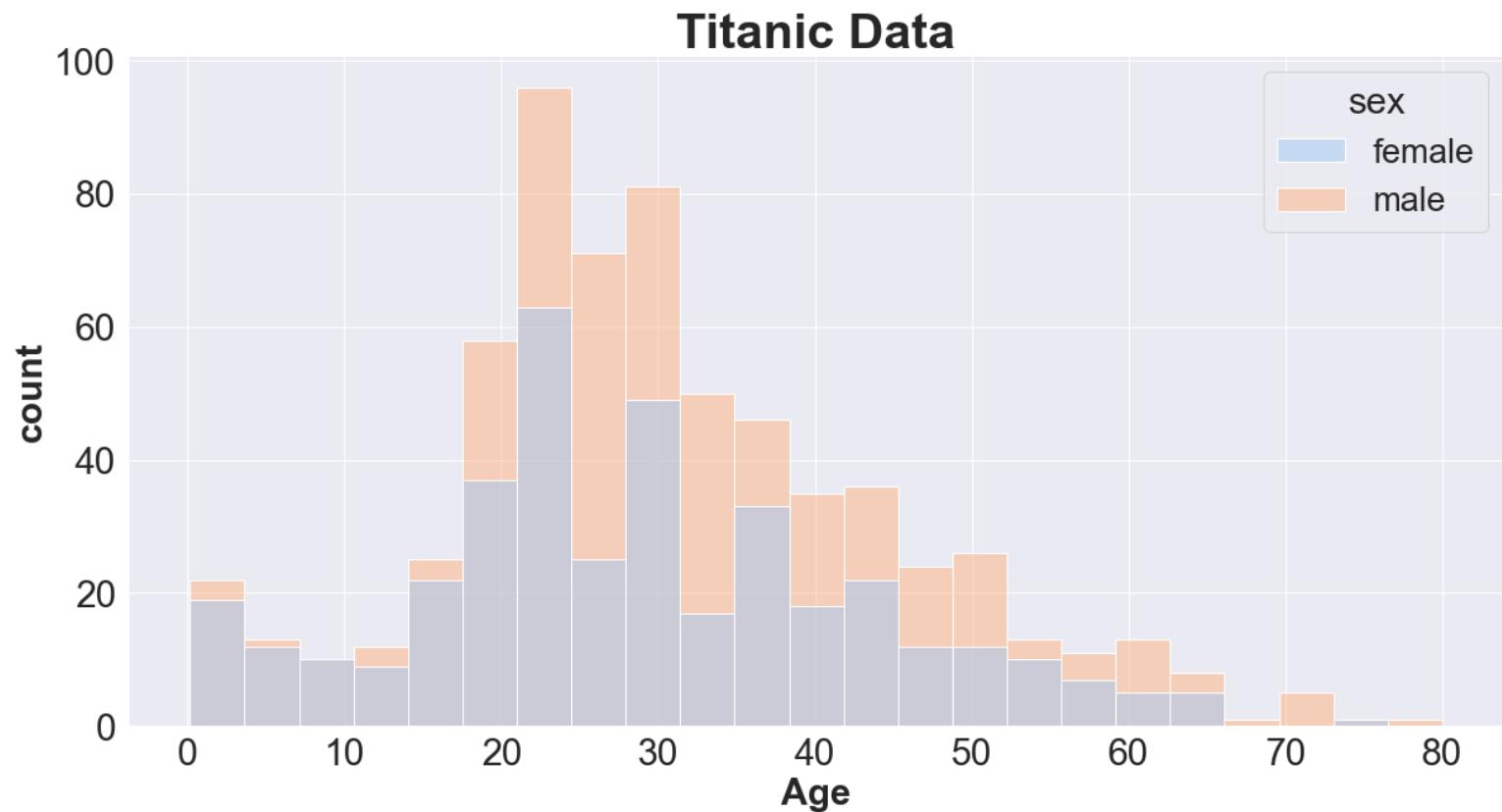
Histograms

- Intervals can hide individual values (difficult to detect relevant values)



Data visualization: distributions

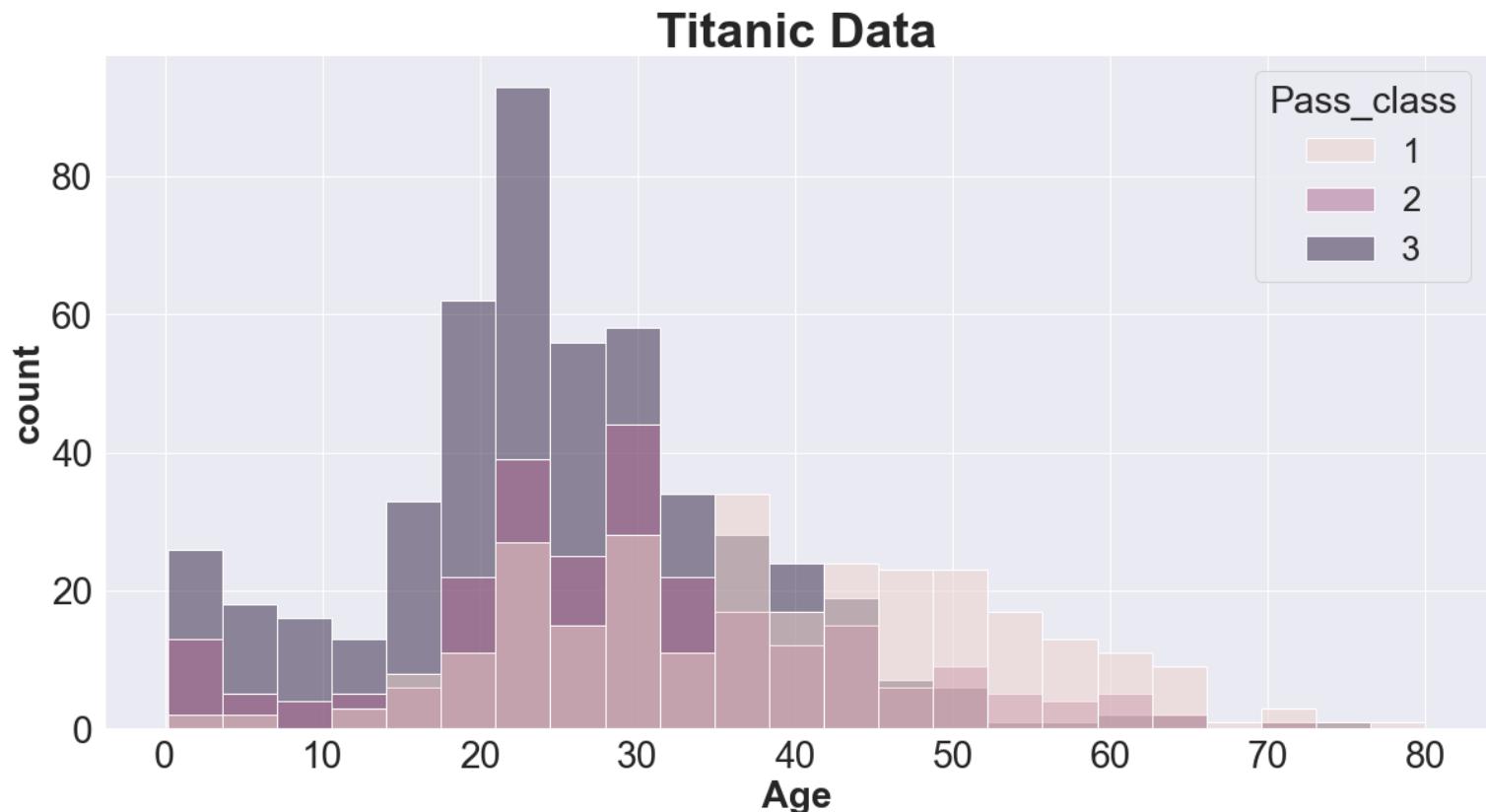
Histograms w.r.t. a categorical variable



Data visualization: distributions

Histograms w.r.t. a categorical variable

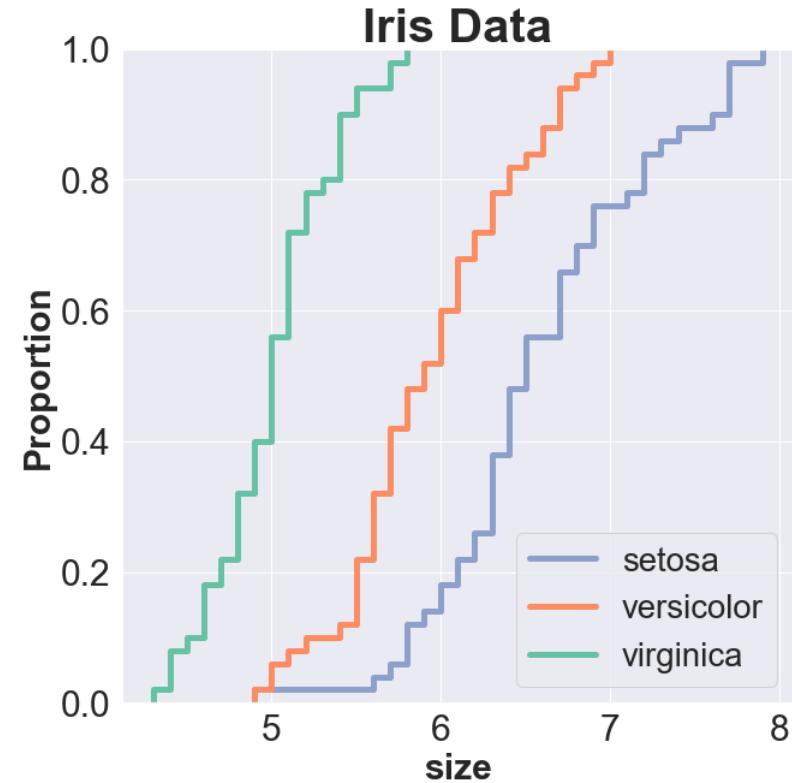
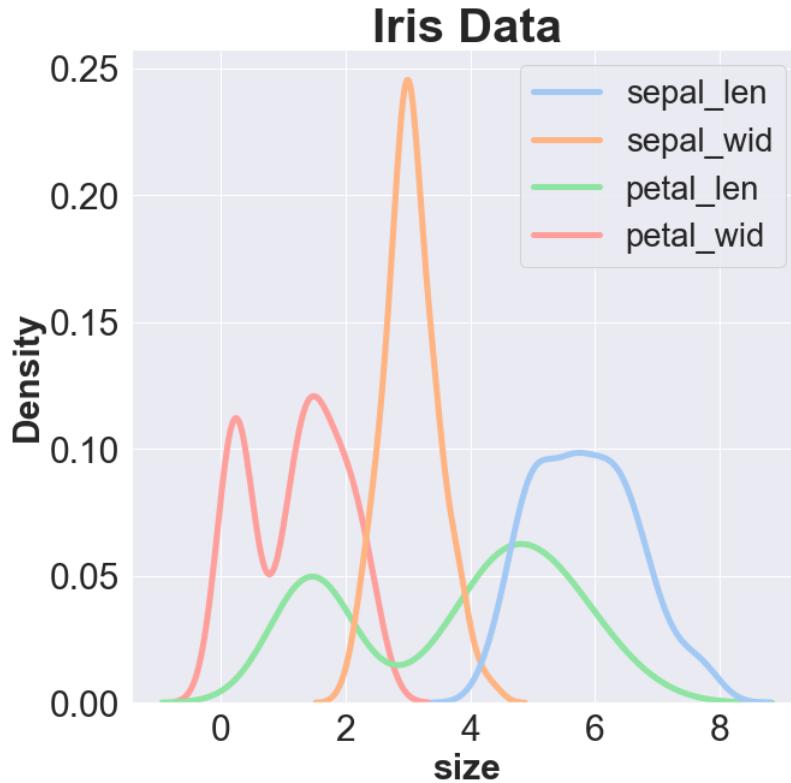
- Can be hard to compare several distributions



Data visualization: distributions

Density plots: overcoming some drawbacks of histograms

- Kernel Density Estimate (KDE)
- Cumulative Distribution Function (CDF)



Data visualization: distributions

Kernel Density Estimate (KDE)

- Smooth the estimates of the distribution of the values
- Kernel estimates compute the estimate of the distribution at a certain point by smoothly averaging over the neighboring points
- The density is estimated by

$$\hat{f}_h(x) = \frac{1}{n} \sum_i^n K\left(\frac{x - x_i}{h}\right)$$

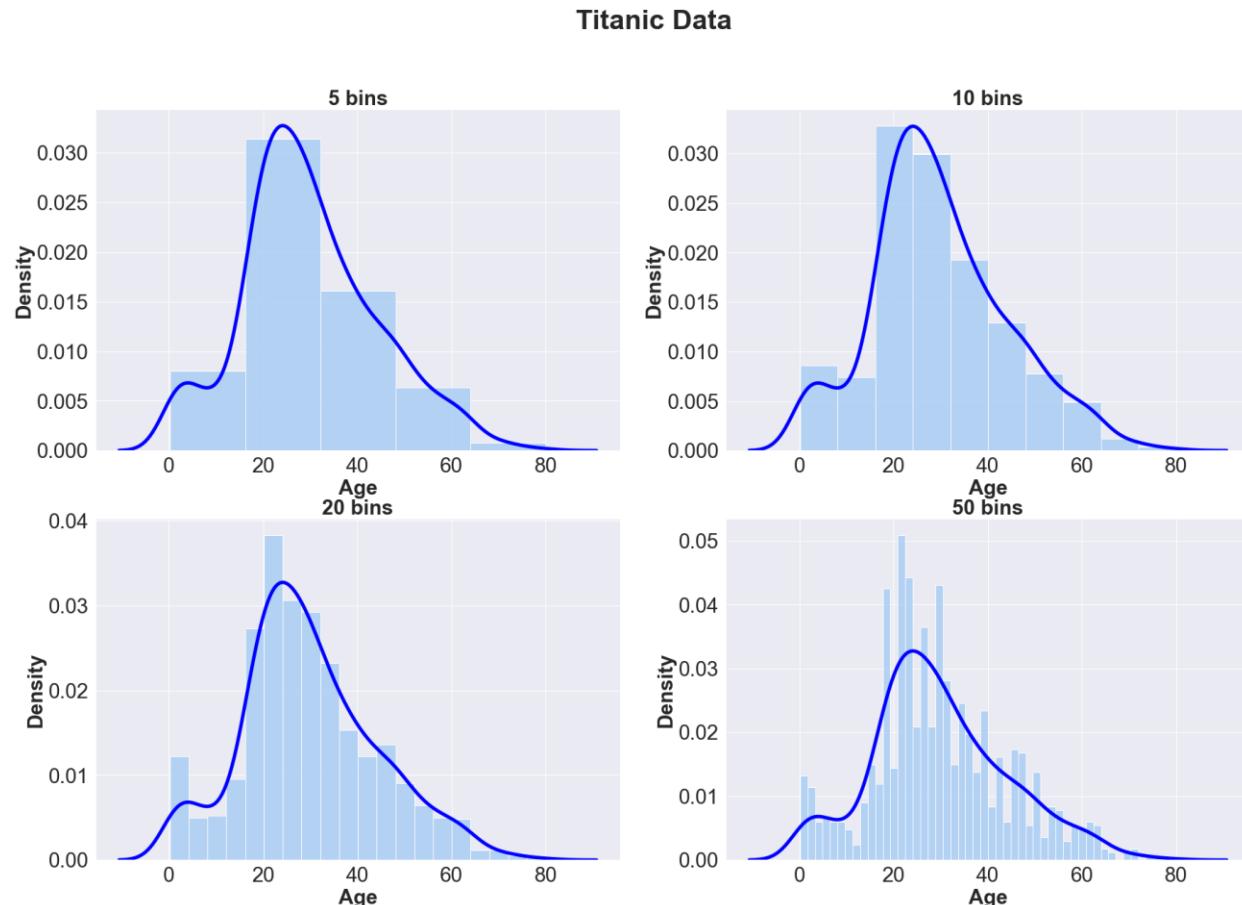
where

- $K(\cdot)$ is the kernel — a non-negative function
- $h > 0$ is a smoothing parameter called the bandwidth.
- $x \in [x_i - h, x_i + h]$

Data visualization: distributions

Kernel Density Estimate (KDE)

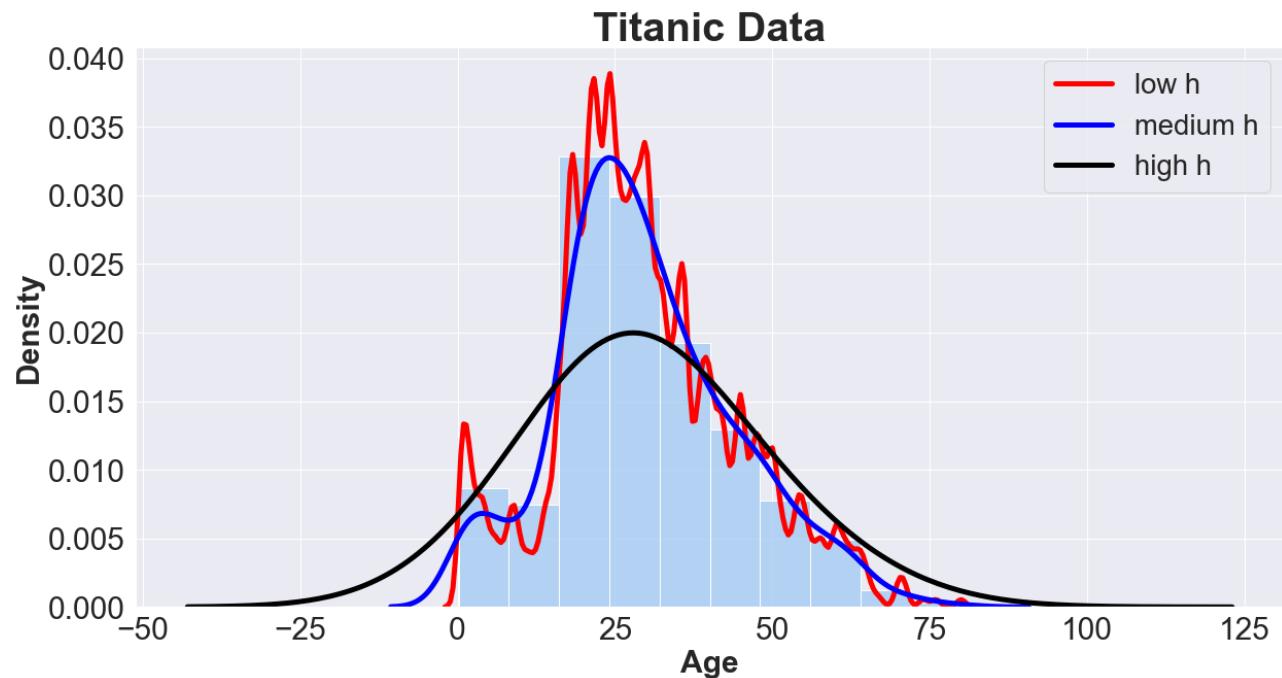
- Bandwidth (h) can be estimated with several methods



Data visualization: distributions

Kernel Density Estimate (KDE)

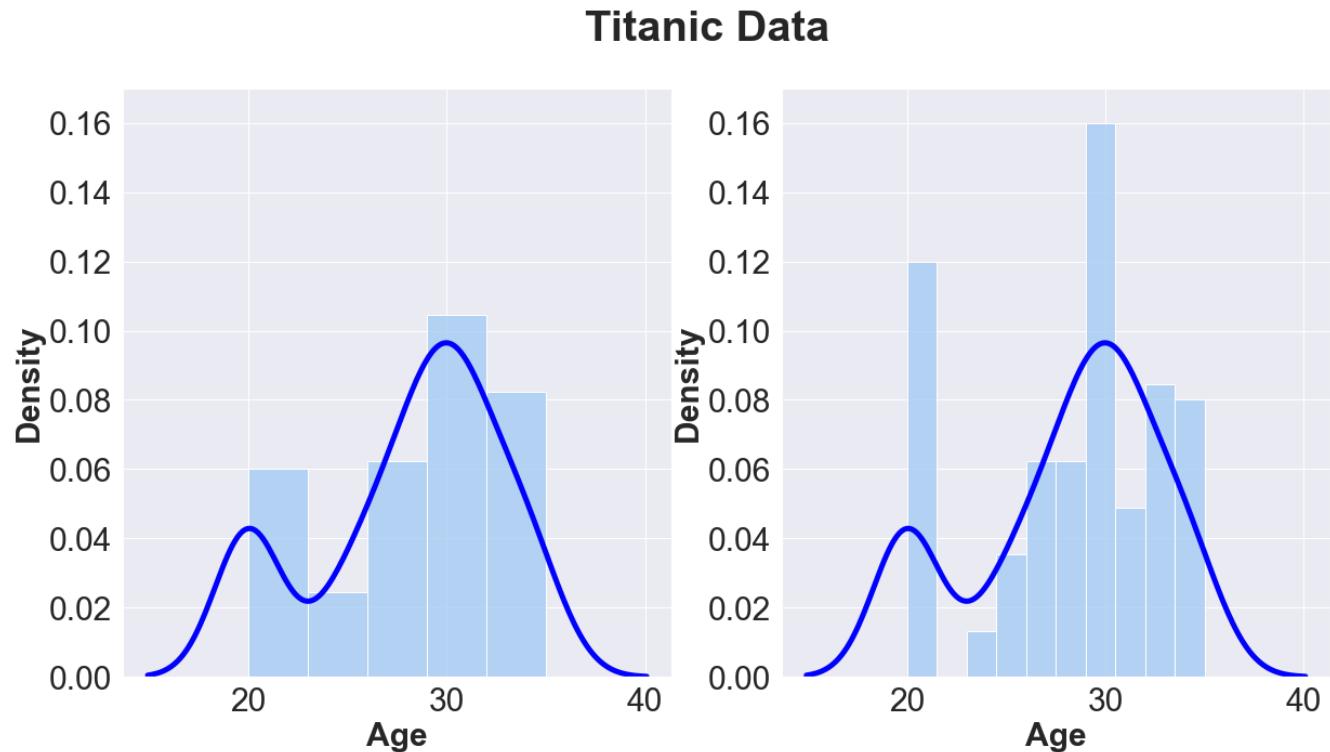
- Bandwidth (h) can be estimated with several methods
- Different values of the bandwidth (h) affect the shape of the KDE



Data visualization: distributions

Kernel Density Estimate (KDE)

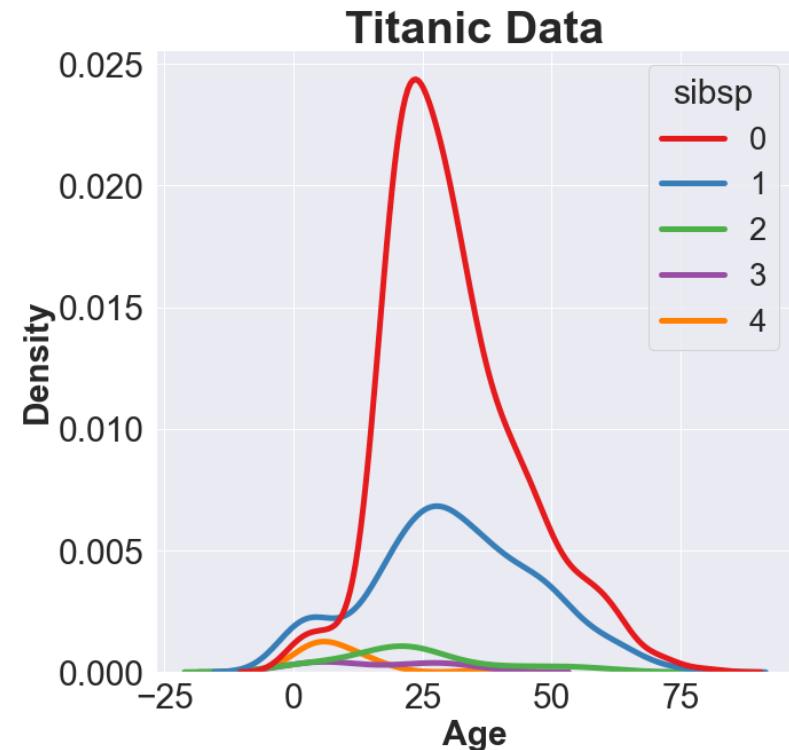
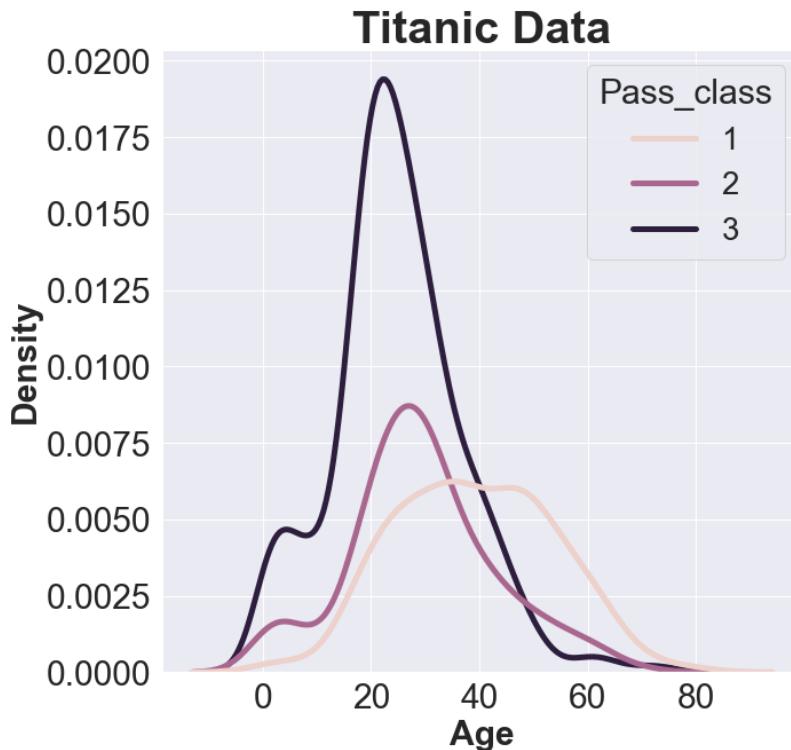
- Identify relevant values is easier (than with histograms)



Data visualization: distributions

Kernel Density Estimate (KDE) w.r.t. a categorical variable

- Compare distributions is easier (than with histograms)



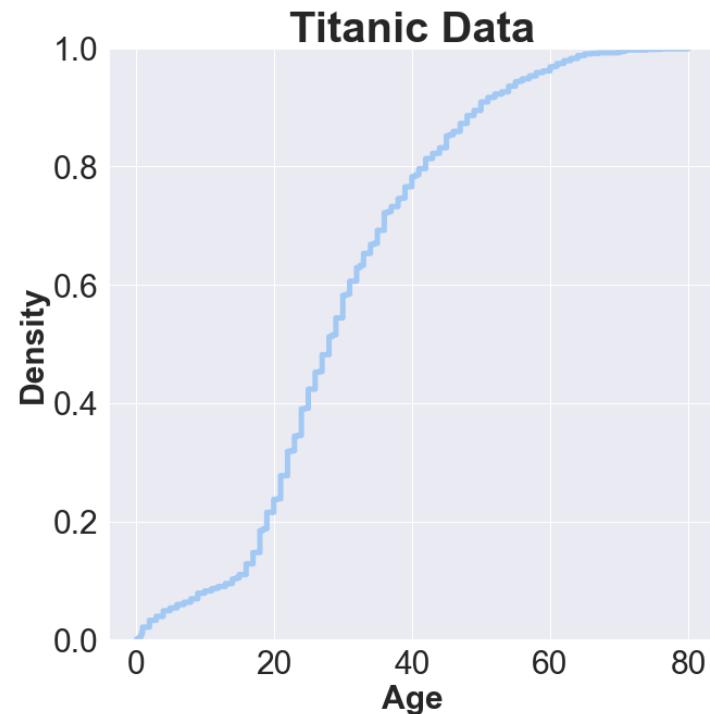
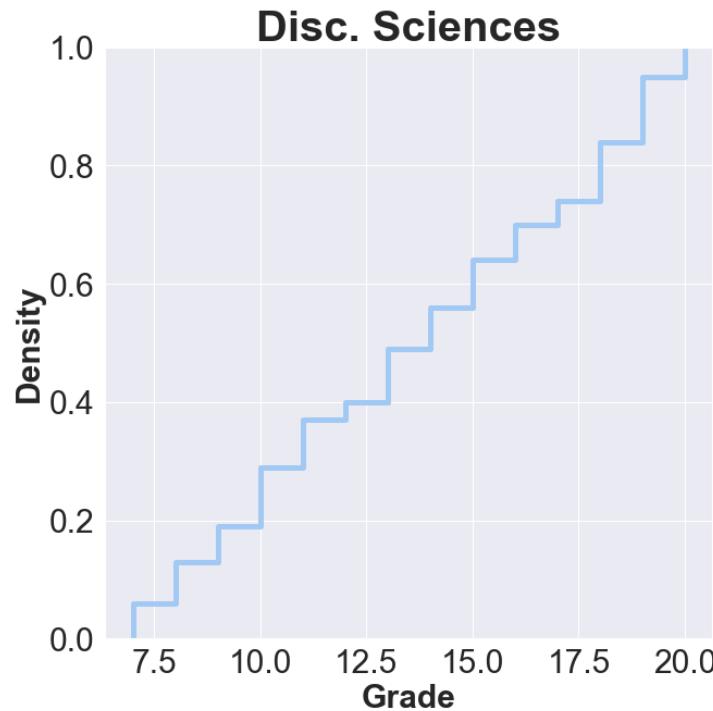
Data visualization: distributions

Cumulative Distribution Function (CDF)

- Gives the probability that the variable takes a value less than or equal to x:

$$F_X(x) = P(X \leq x)$$

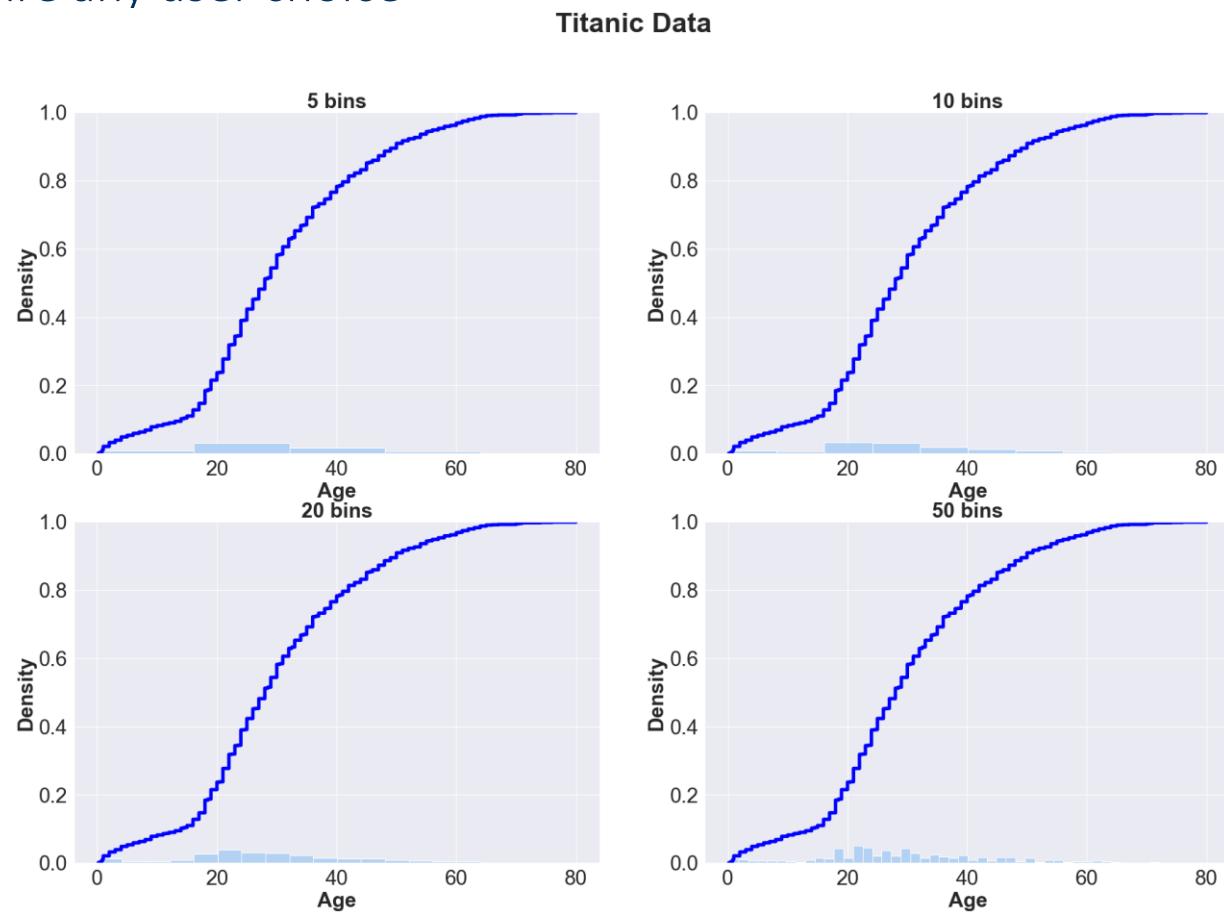
- It allows to recognize a discrete variable at first glance



Data visualization: distributions

Cumulative Distribution Function (CDF)

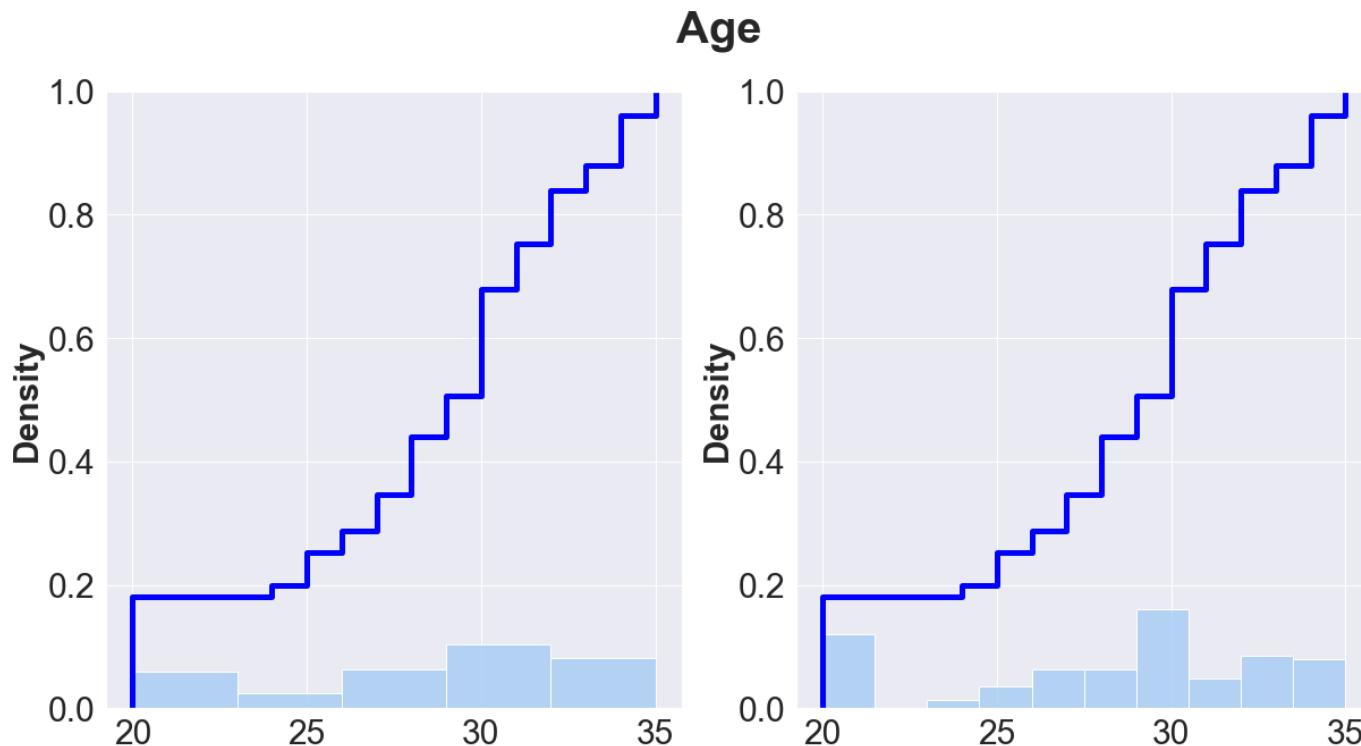
- It doesn't require any user choice



Data visualization: distributions

Cumulative Distribution Function (CDF)

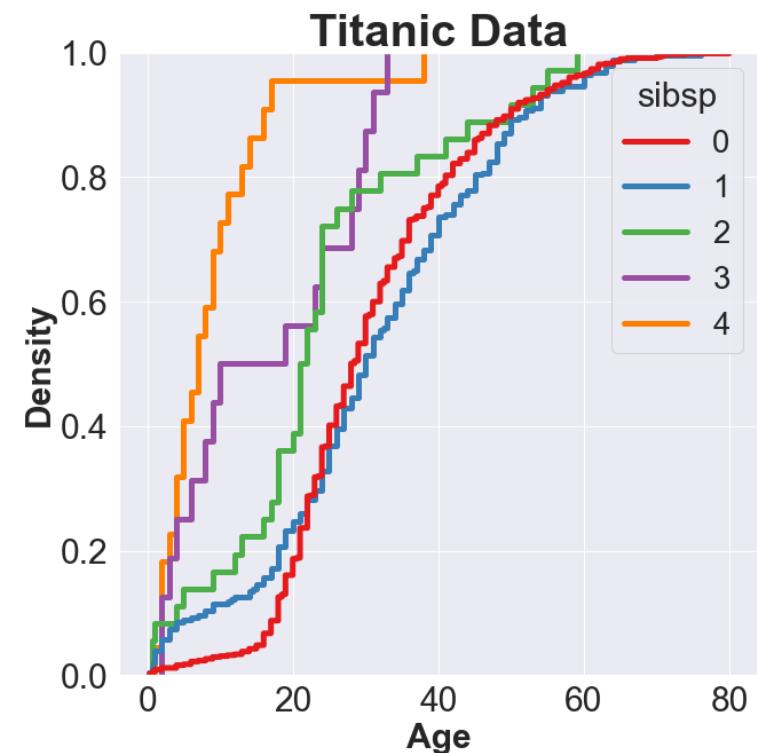
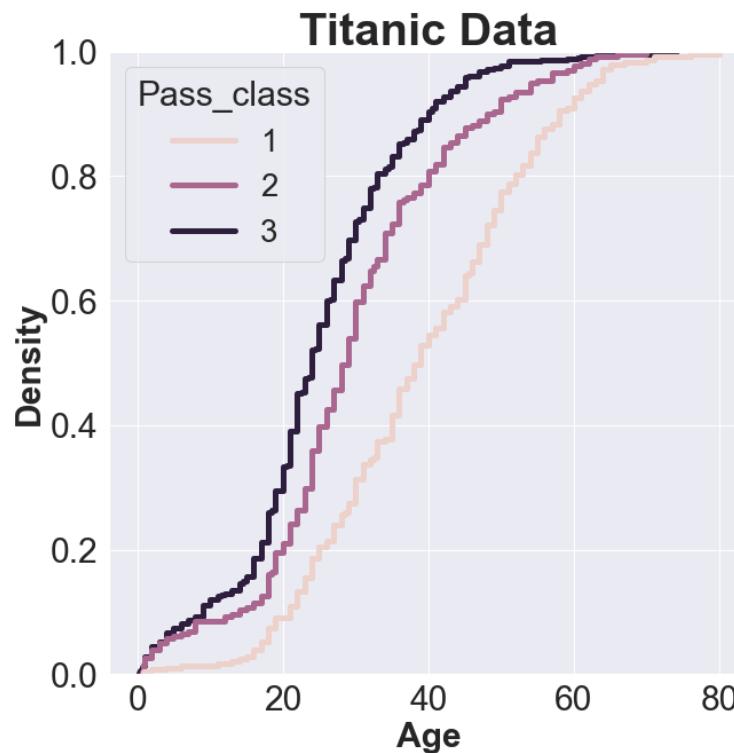
- Identify relevant values is easier (than with histograms)



Data visualization: distributions

Cumulative Distribution Function (CDF) w.r.t. a categorical variable

- Compare distributions is easier (than with histograms)



Data visualization: distributions

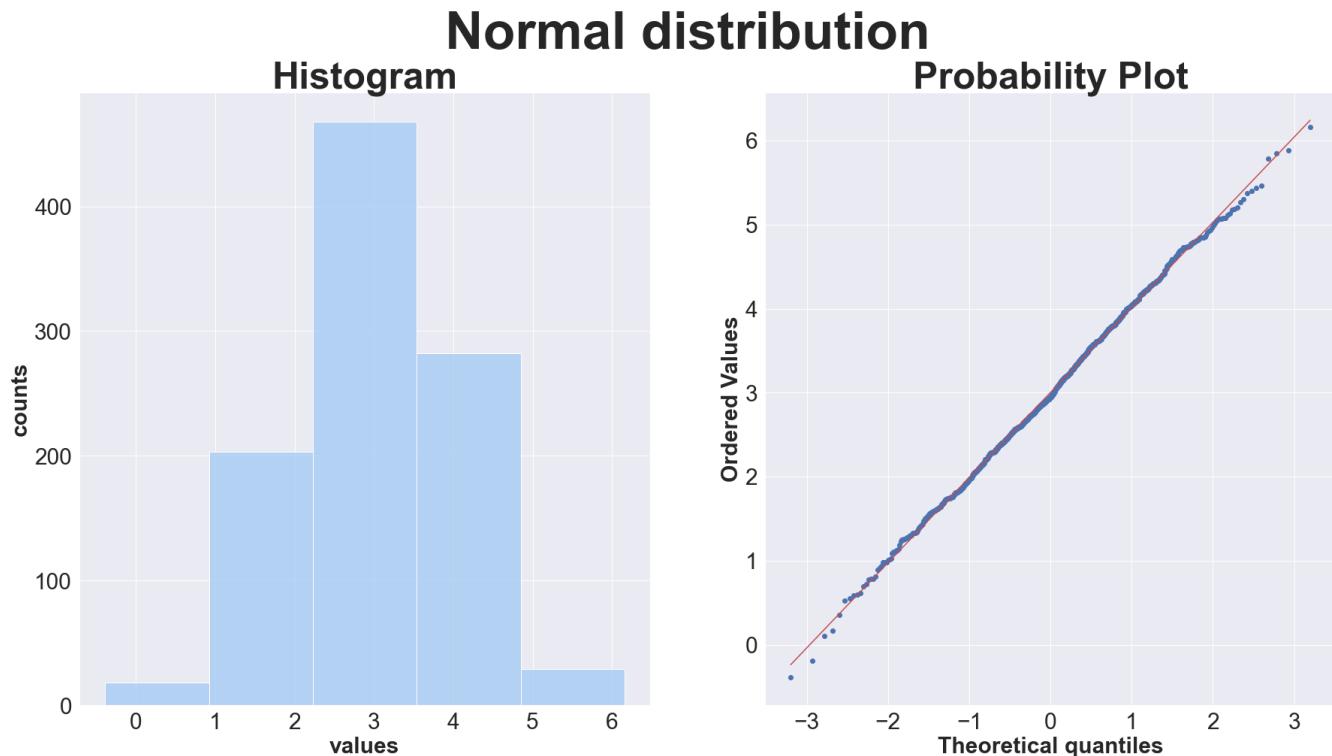
Quantile-Quantile plots (QQ plots)

- Graphs that can be used to compare the observed distribution against the Normal distribution
- Can be used to visually check the hypothesis that the variable under study follows a normal distribution
- Obviously, more formal tests also exist

Data visualization: distributions

Quantile-Quantile plots (QQ plots)

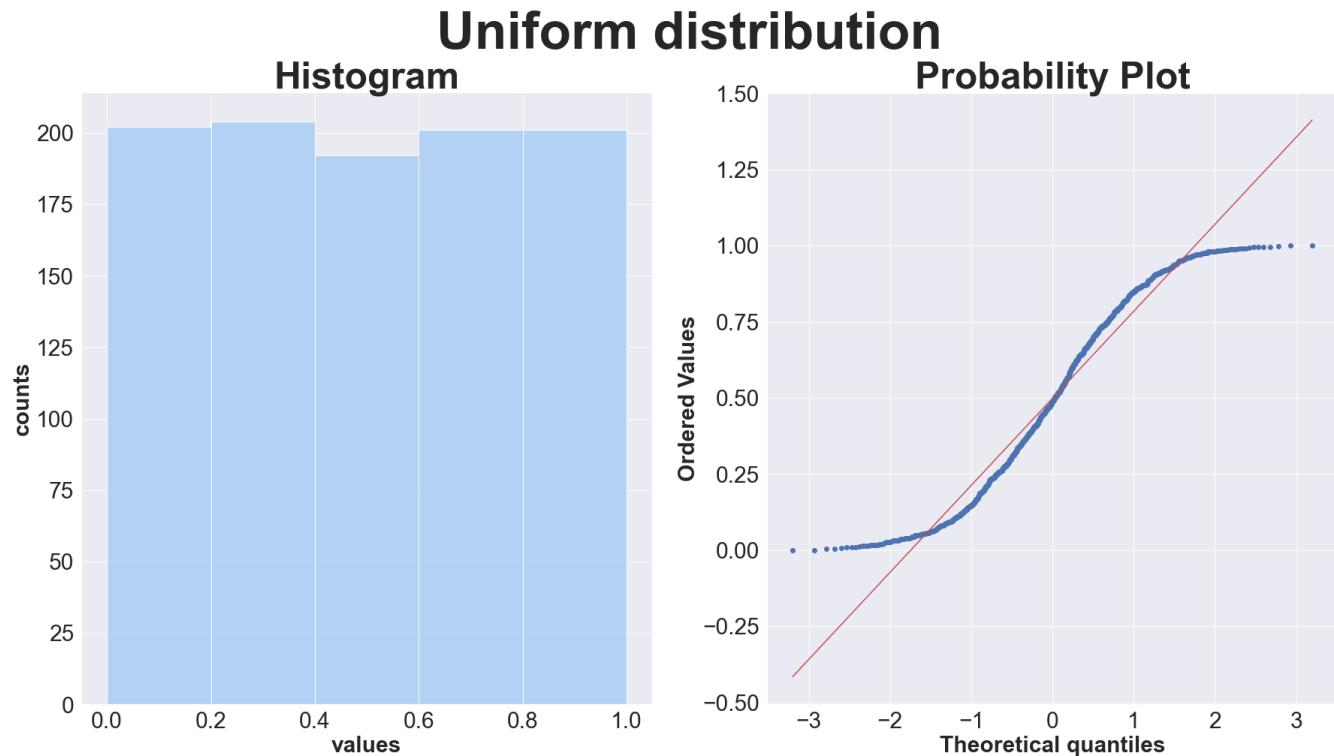
- It doesn't require any user choice
- Lesser intuitive than histograms or density plots



Data visualization: distributions

Quantile-Quantile plots (QQ plots)

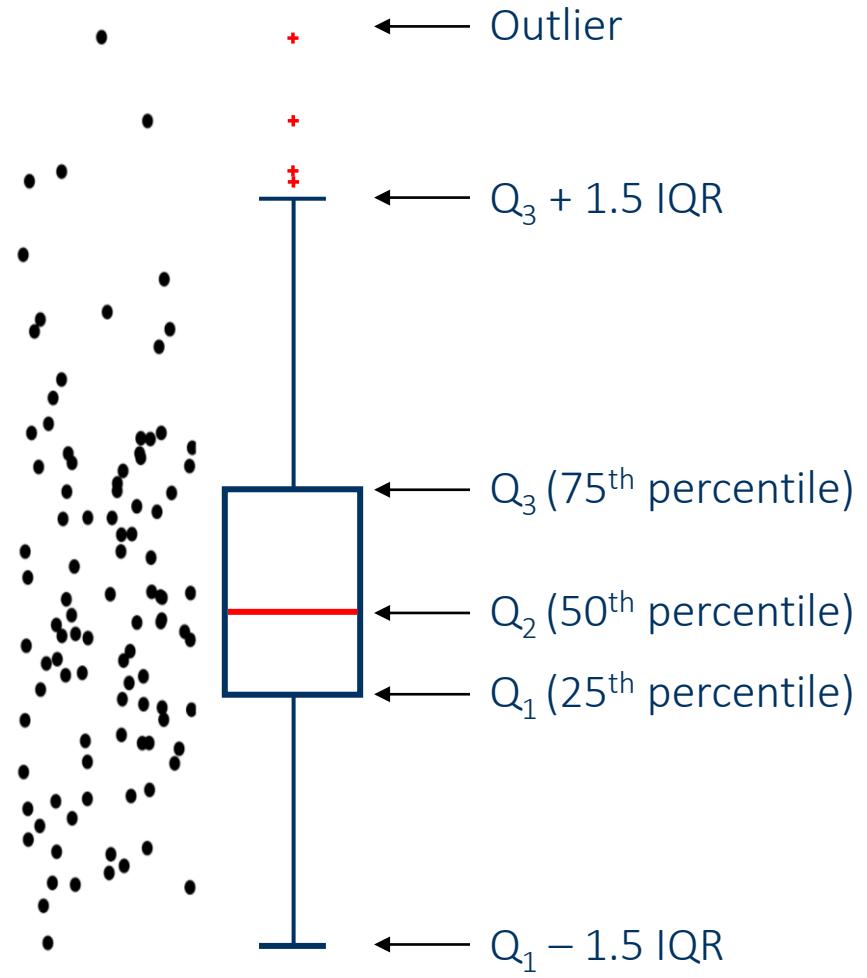
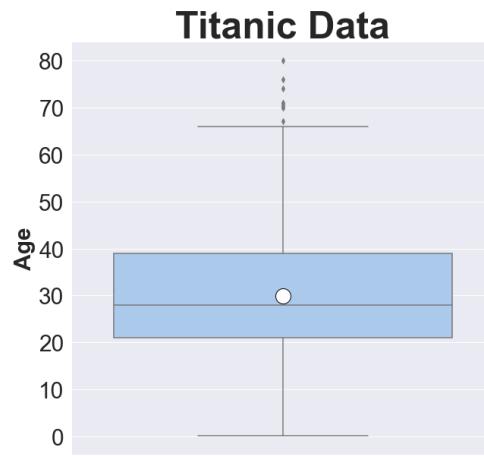
- It doesn't require any user choice
- Lesser intuitive than histograms or density plots



Data visualization: distributions

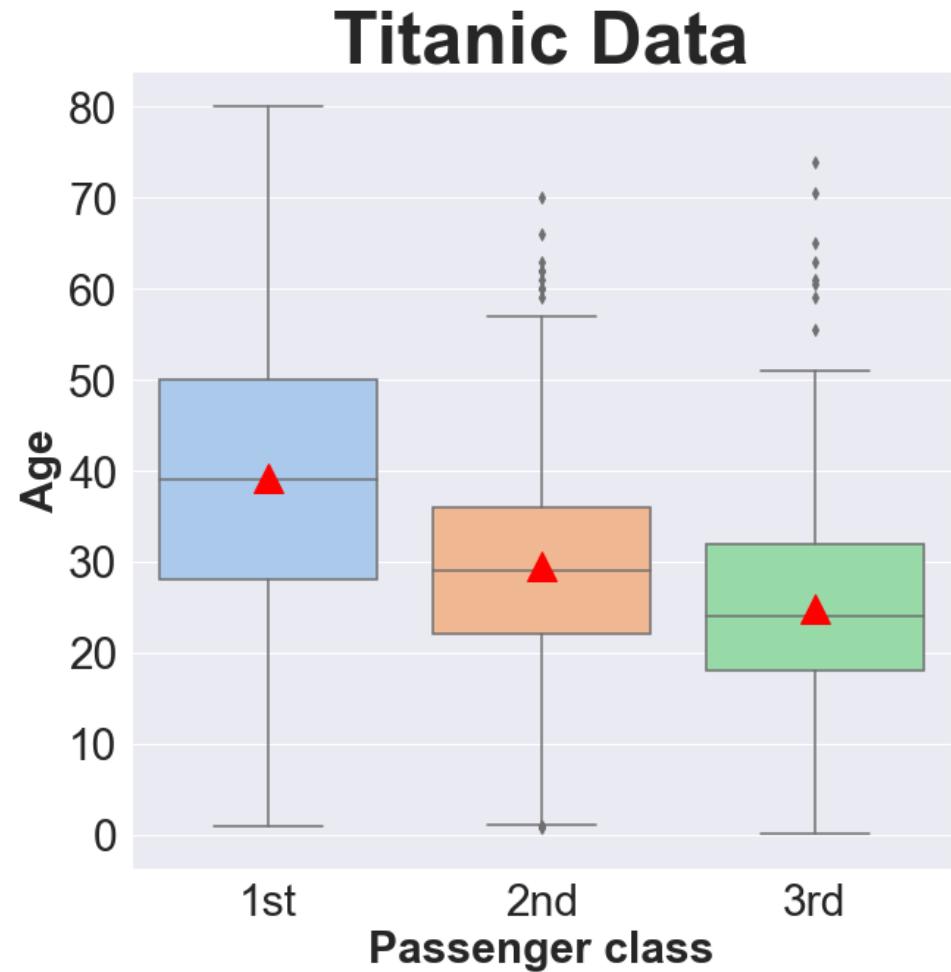
Boxplots (Tukey, 1977)

- Summary of a variable distribution
- Understand the spread of data
- Information on the interquartile range
- Determine if data is skewed
- Identify outliers (if any)



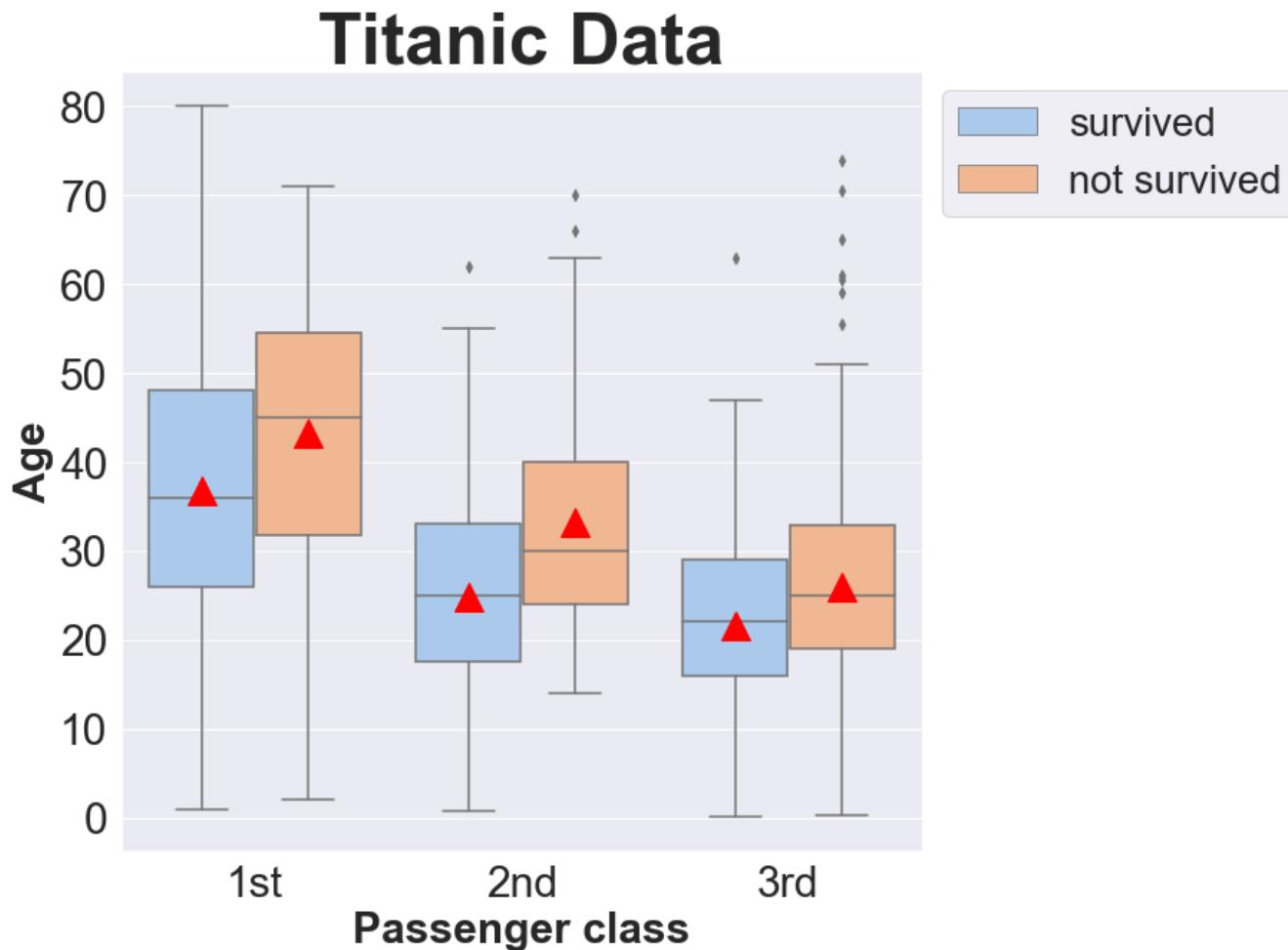
Data visualization: distributions

Boxplots w.r.t. a categorical variable



Data visualization: distributions

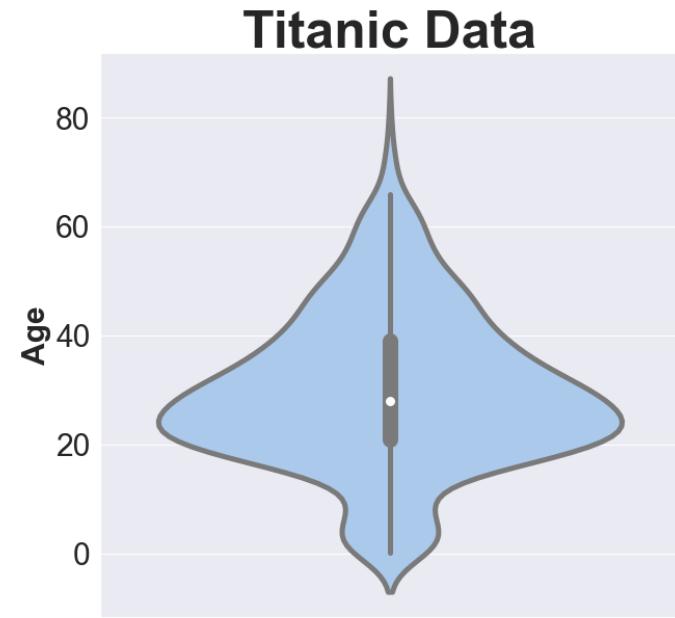
Boxplots w.r.t. two categorical variables



Data visualization: distributions

Violinplots*

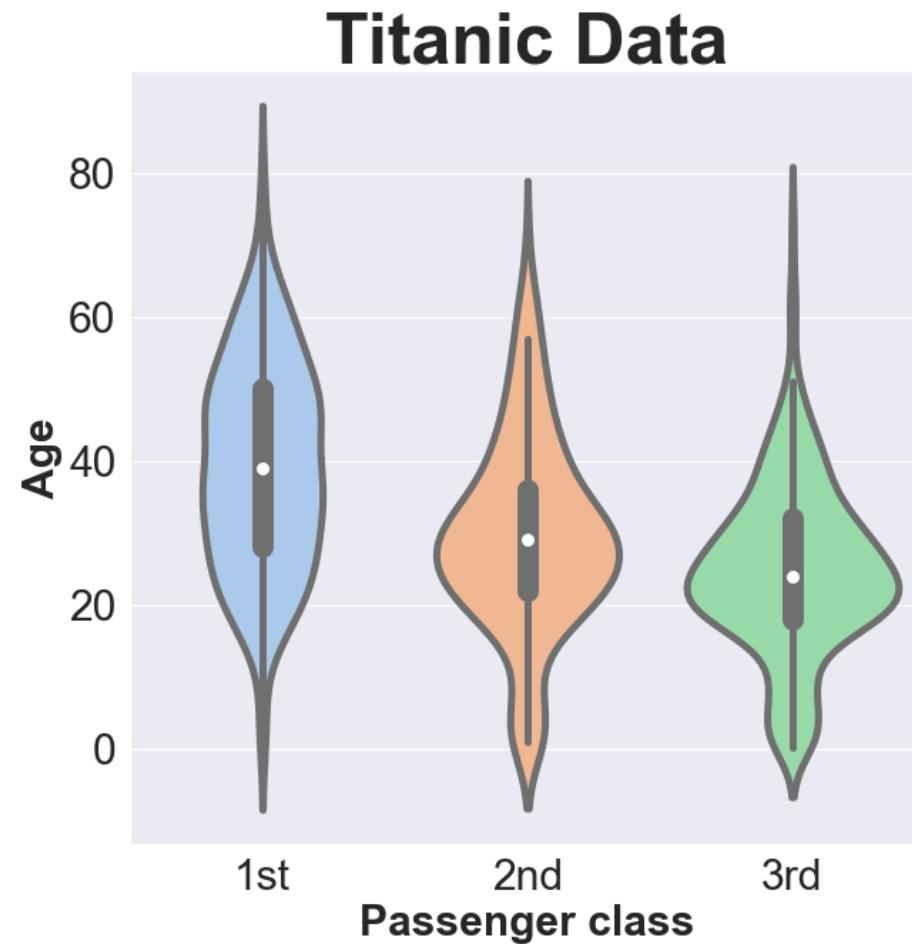
- Similar to boxplots
- Shows the probability density of the data at different value
 - can capture the distribution better compared to boxplot
 - can capture the structure in the data



*Need enough data to estimate the density

Data visualization: distributions

Violinplots w.r.t. a categorical variable



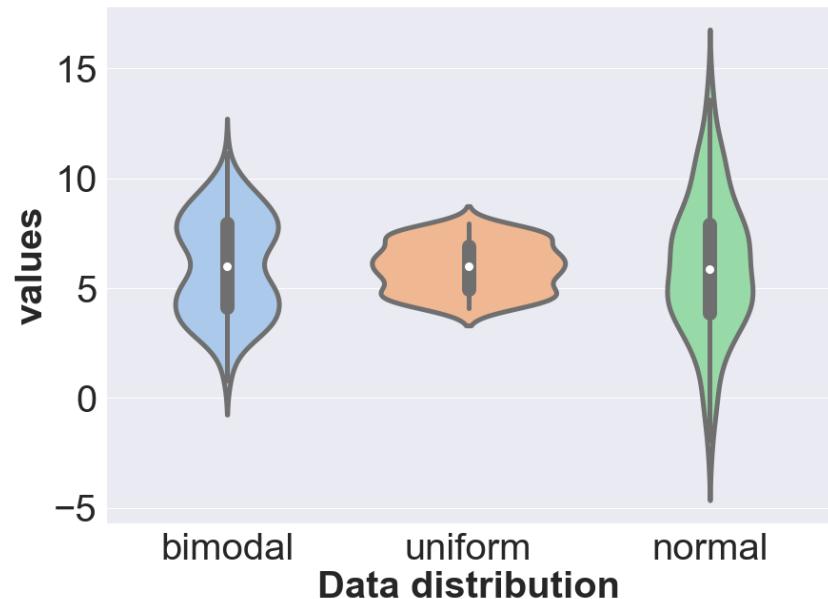
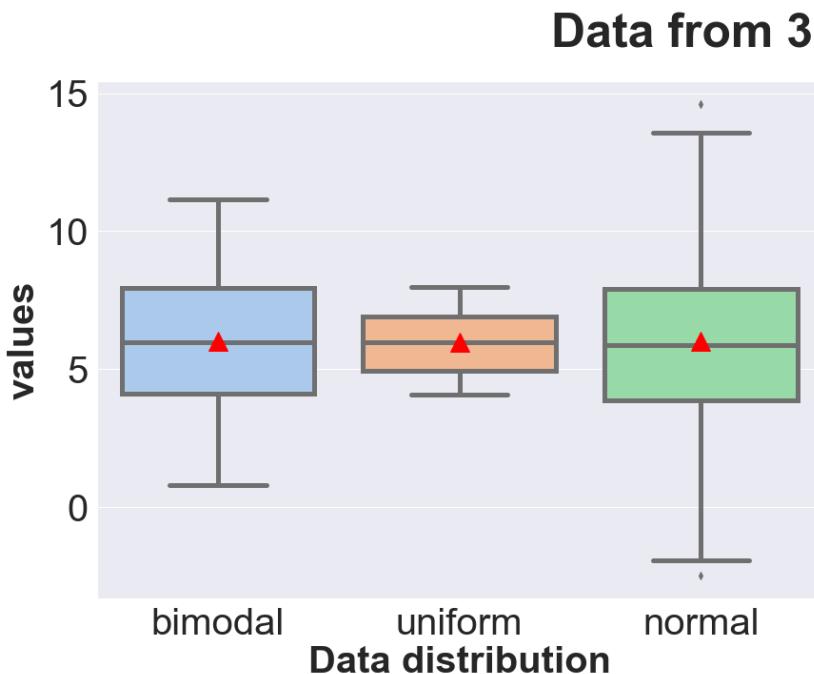
Data visualization: distributions

Violinplots w.r.t. a categorical variable

- Similar median values
- distributed differently

Violinplot: adds **density** information

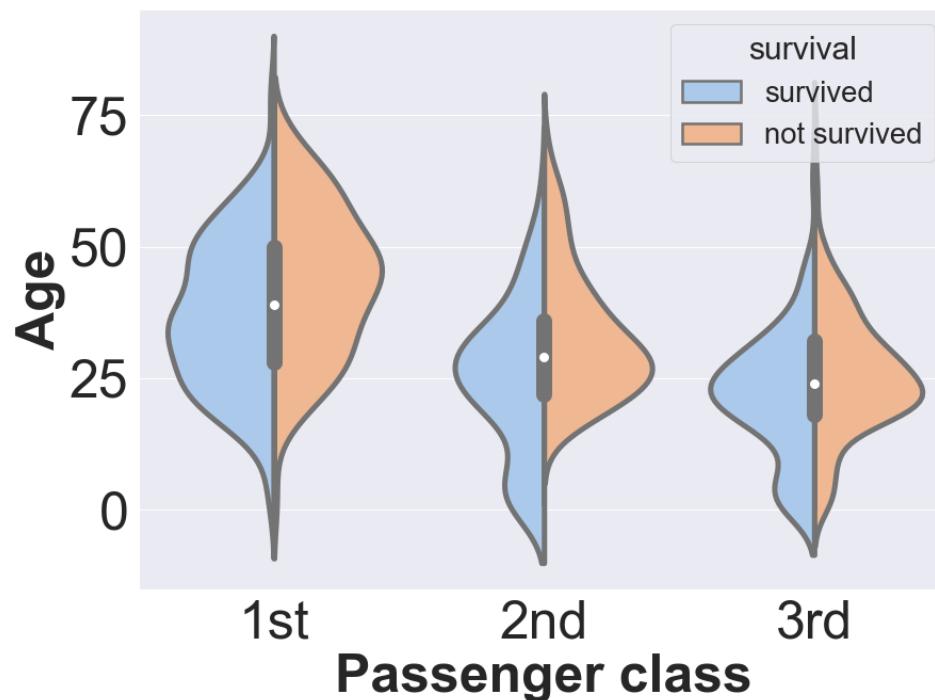
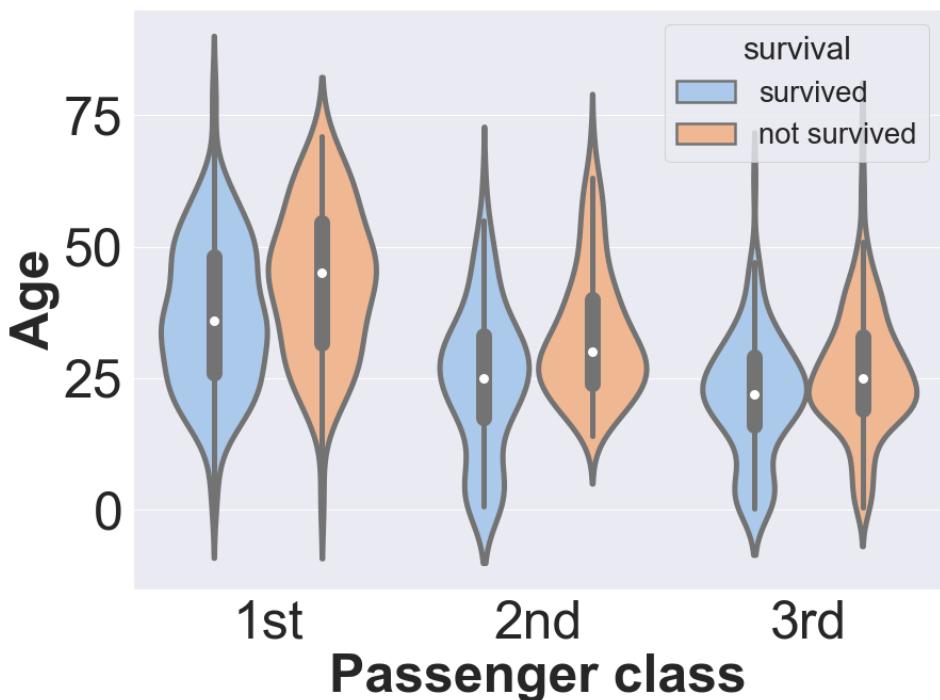
- revealing the structure in the data



Data visualization: distributions

Violinplots w.r.t. two categorical variables

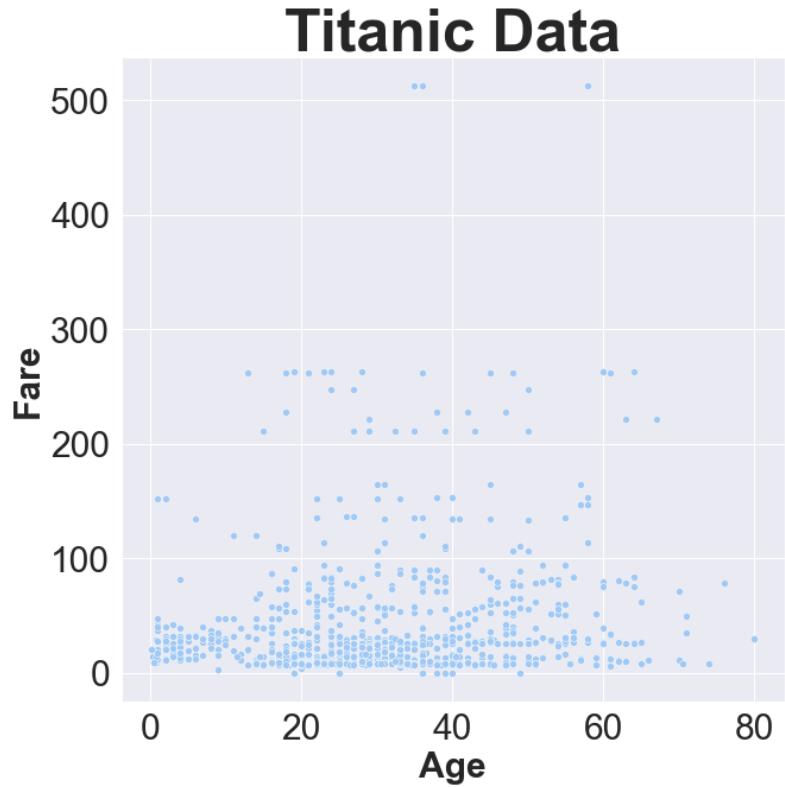
Titanic Data



Data visualization: associations

Scatter plots

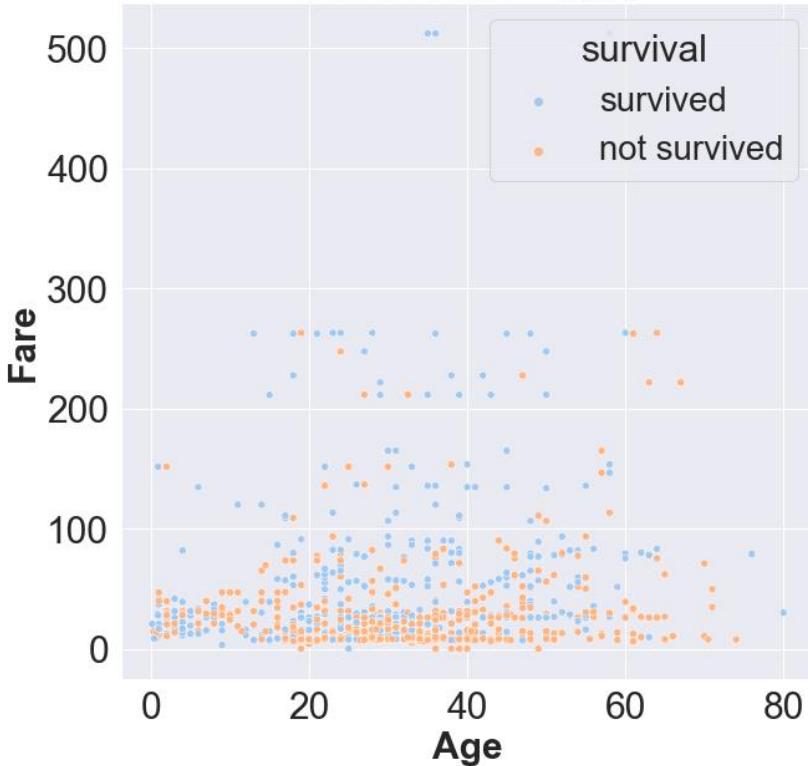
- Show one quantitative variable relative to another
- Data points are shown as dispersed cloud



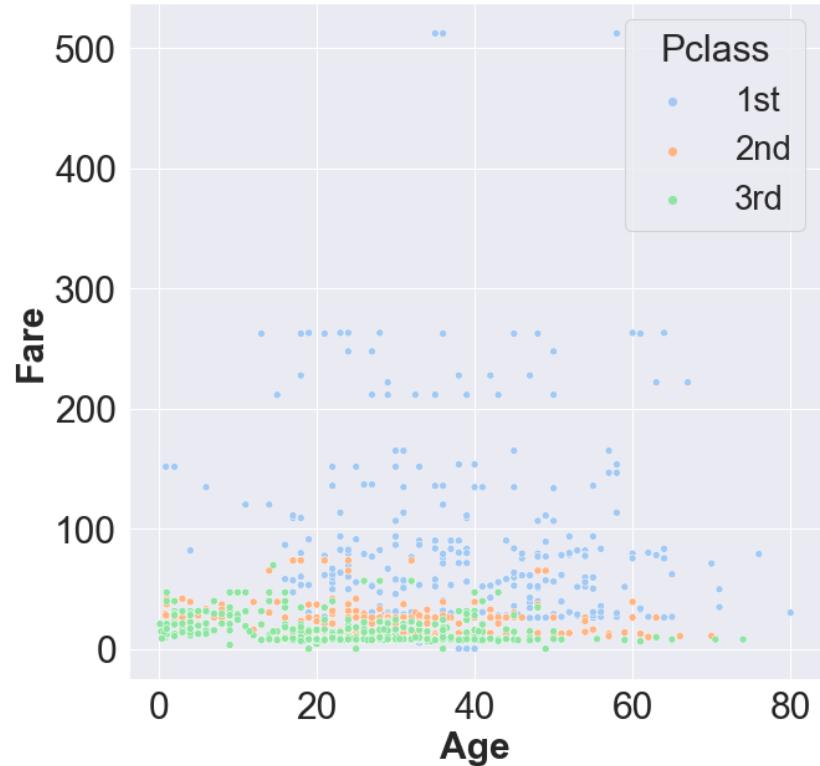
Data visualization: associations

Scatter plots w.r.t categorical variable

Titanic Data



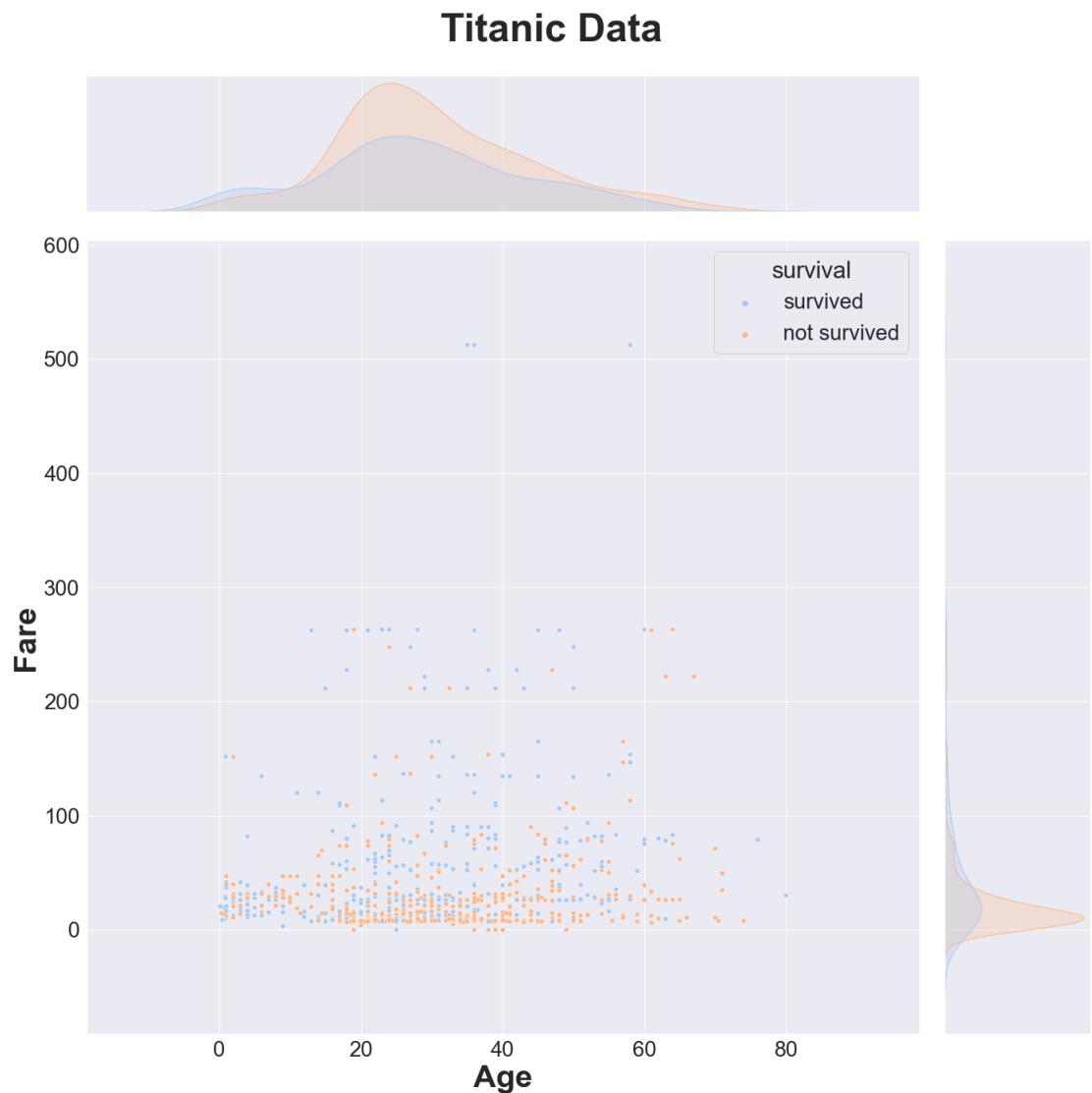
Titanic Data



Data visualization: associations

Scatter plots

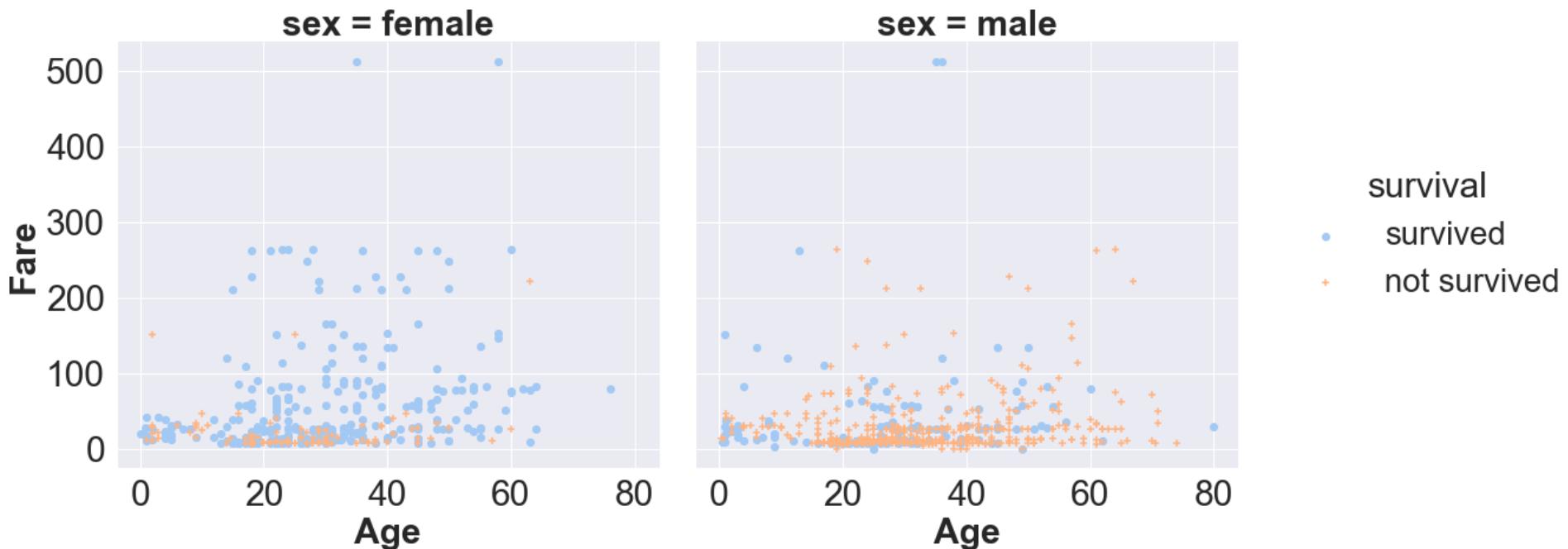
Multiple views on the data



Data visualization: associations

Scatter plots w.r.t. two categorical variables

Survival by Gender , Age and Fare

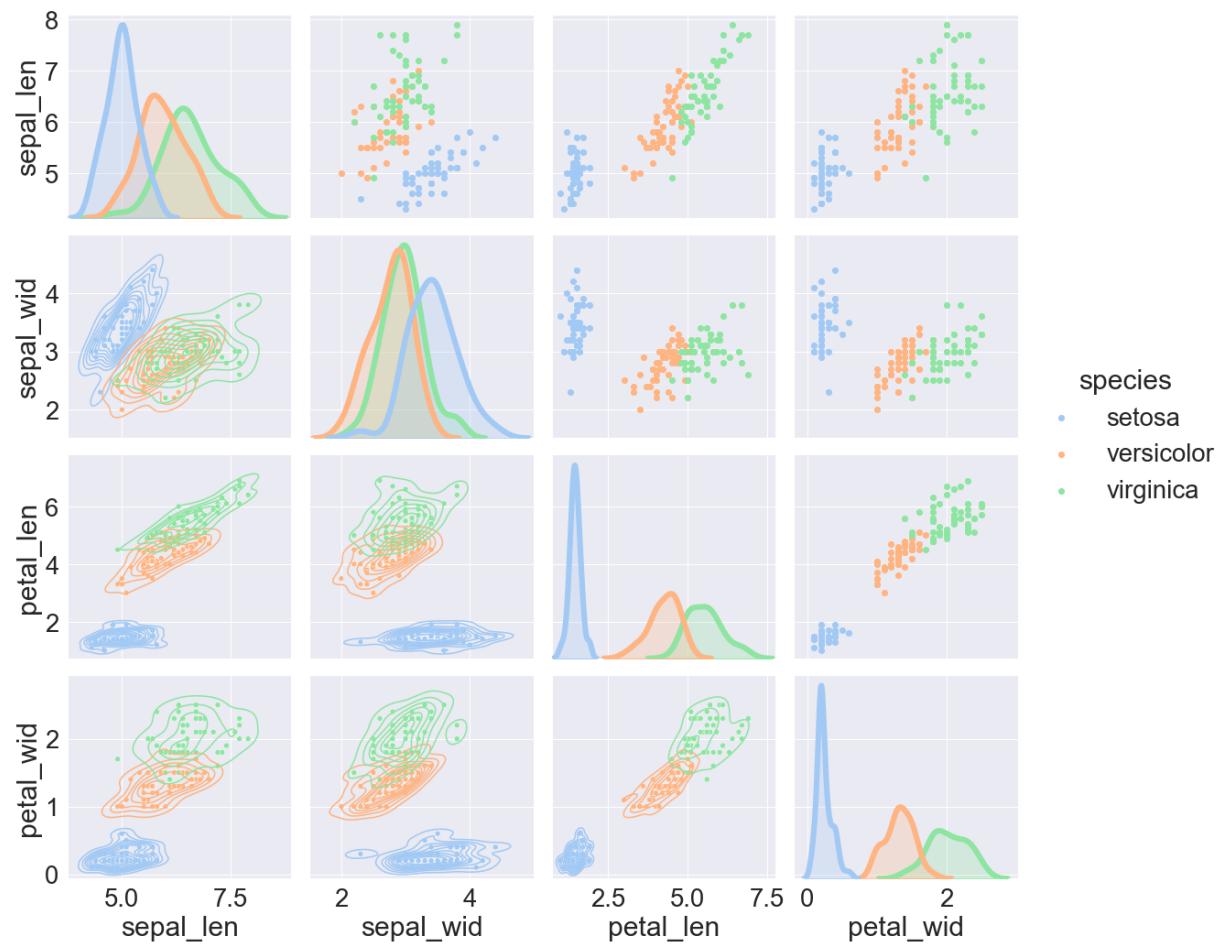


Data visualization: associations

Scatter matrix: pairwise scatter plots

- all-against-all

Iris dataset

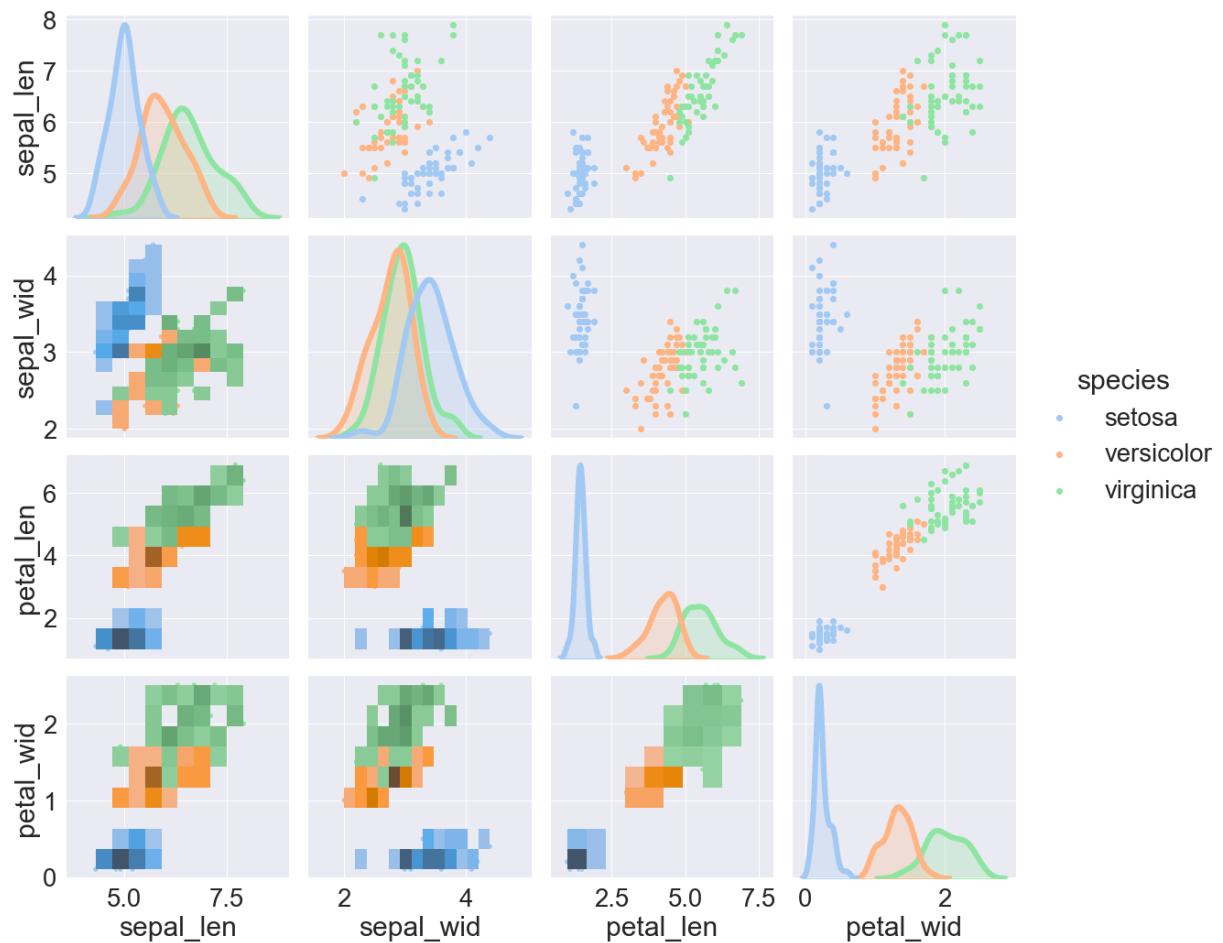


Data visualization: associations

Scatter matrix: pairwise scatter plots

- all-against-all

Iris dataset



Data visualization: associations

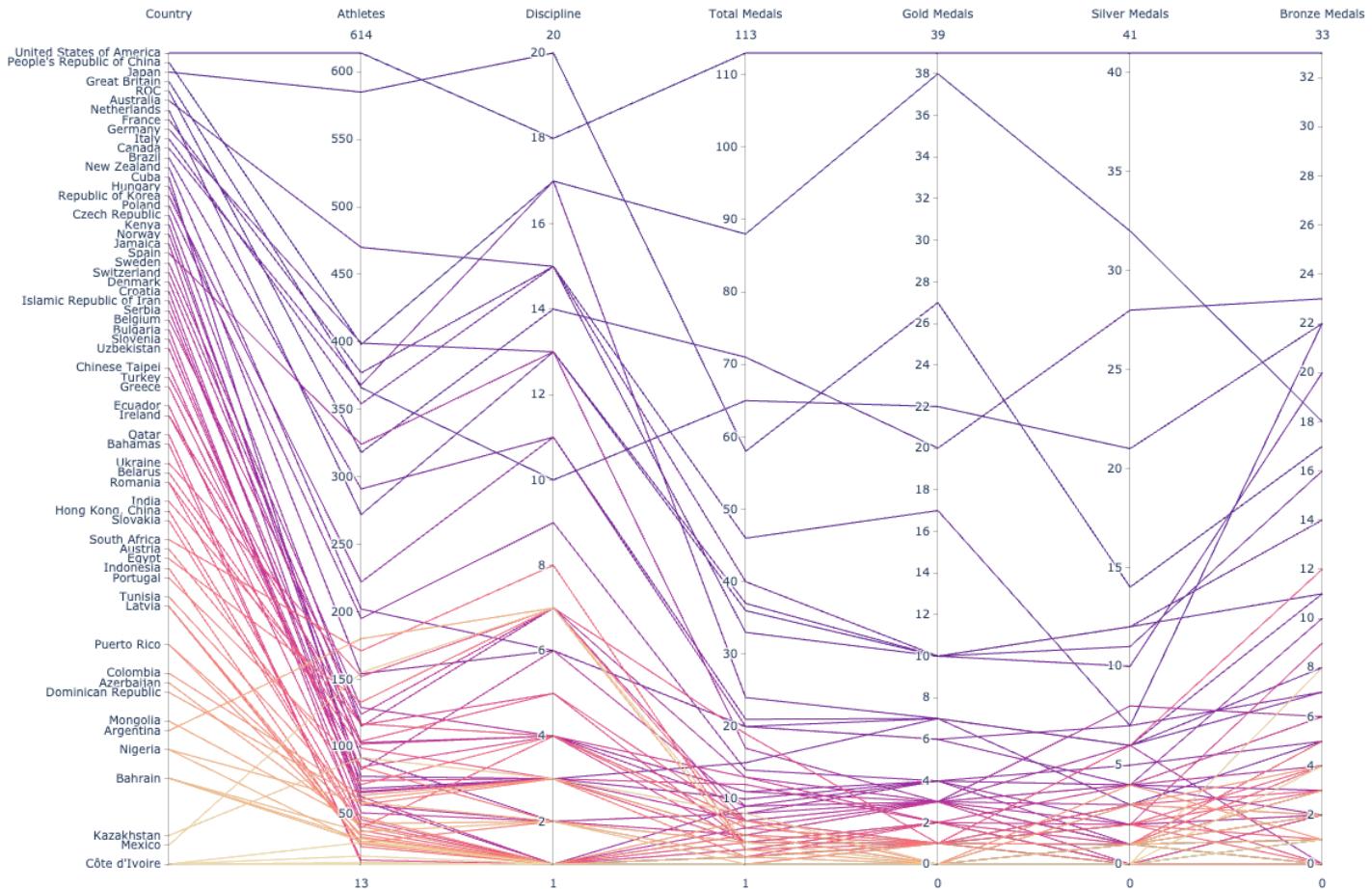
Parallel coordinates plots

- Map n-dimensional relations into 2D patterns
- Allows a comparison of the samples across multiple numerical variables
- Used for high dimensional numerical
- Each variable is represented by a separate axis (equidistants)
 - All the axes are equally spaced and parallel to each other
 - Each axis can have a different scale and unit of measurement: scaled to the [minimum, maximum] of the corresponding variable
- Each sample/observation corresponds to a polygonal line, intersecting the axis of each numerical variable at the point which corresponds to the value for that variable
- The order of variables is important to show groups of observations

Data visualization: associations

Parallel coordinates plots

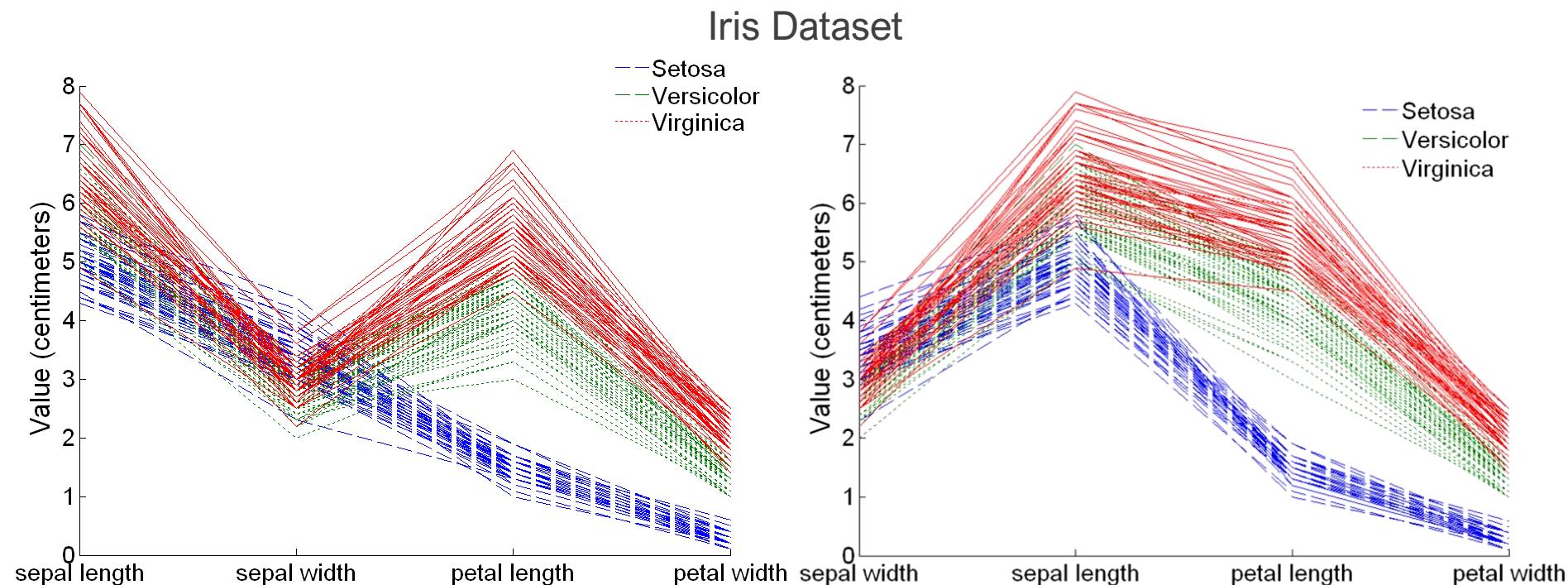
Olympics 2021 dataset



Data visualization: associations

Parallel coordinates plots

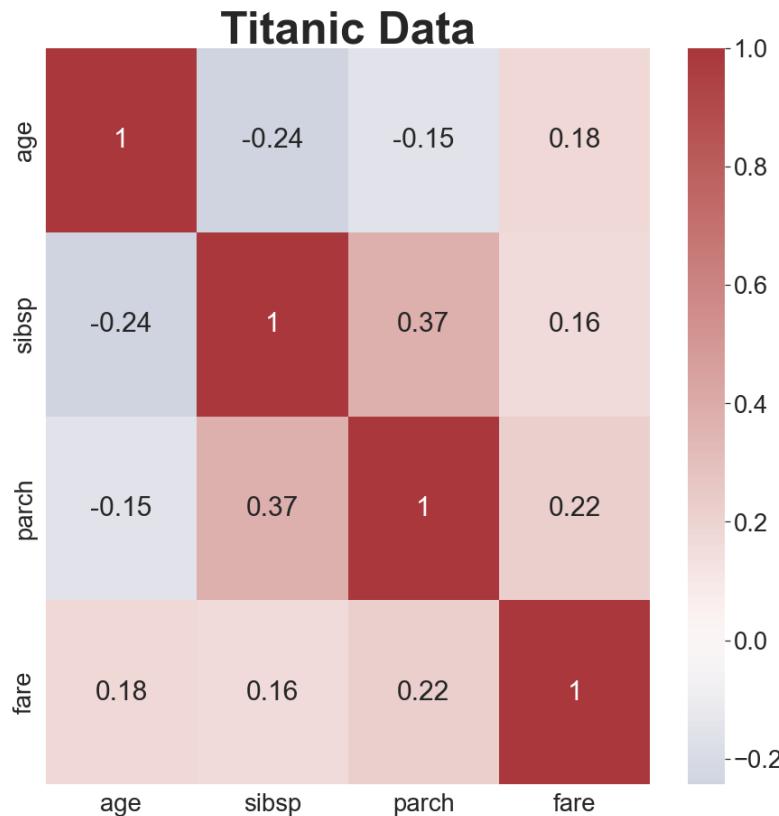
- The **order** of variables is important to show **groups** of observations



Data visualization: associations

Correlograms

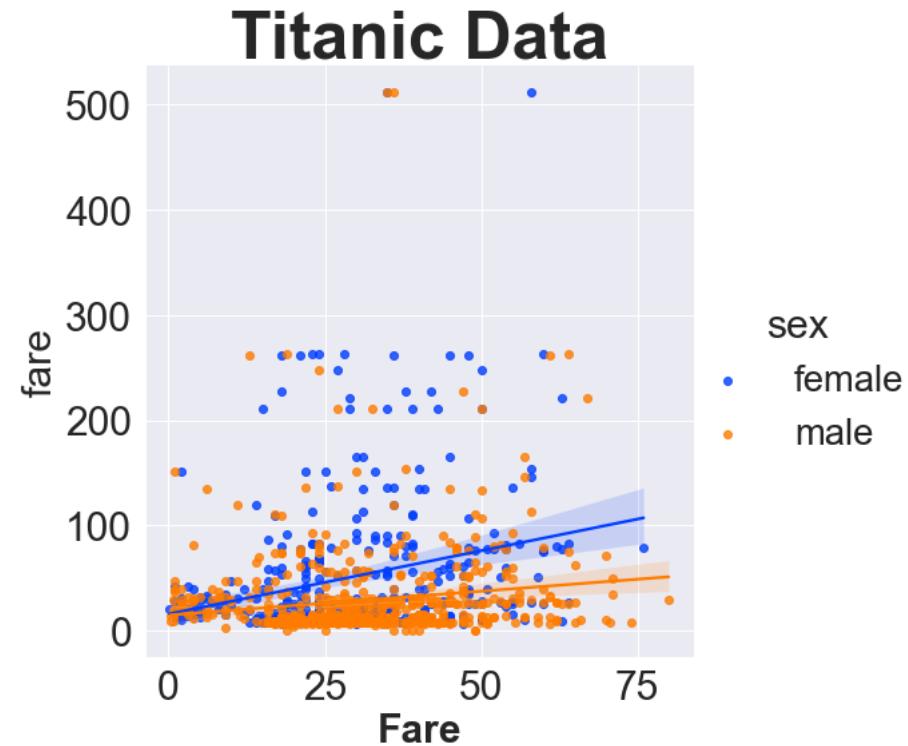
- Display the correlation between every pair of **numeric** variables



Data visualization: trends

Scatterplots

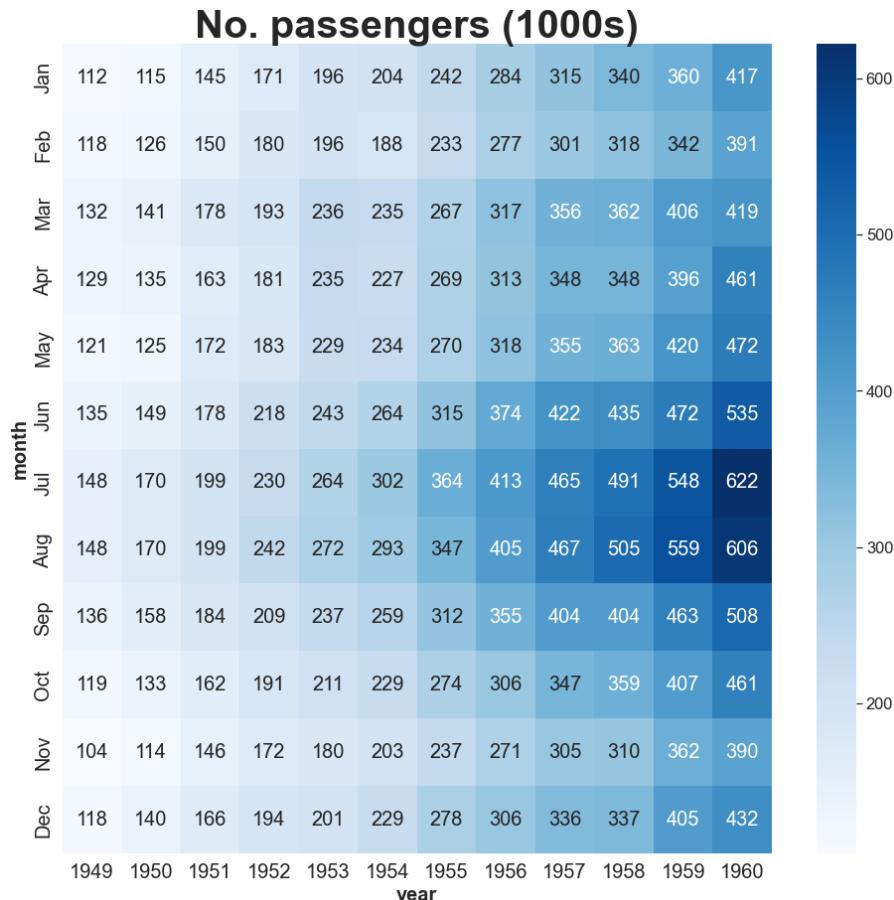
- Several functions allows to approximate the relationship between two numeric variables
- Scatter plot helps to perceive the trends



Data visualization: trends

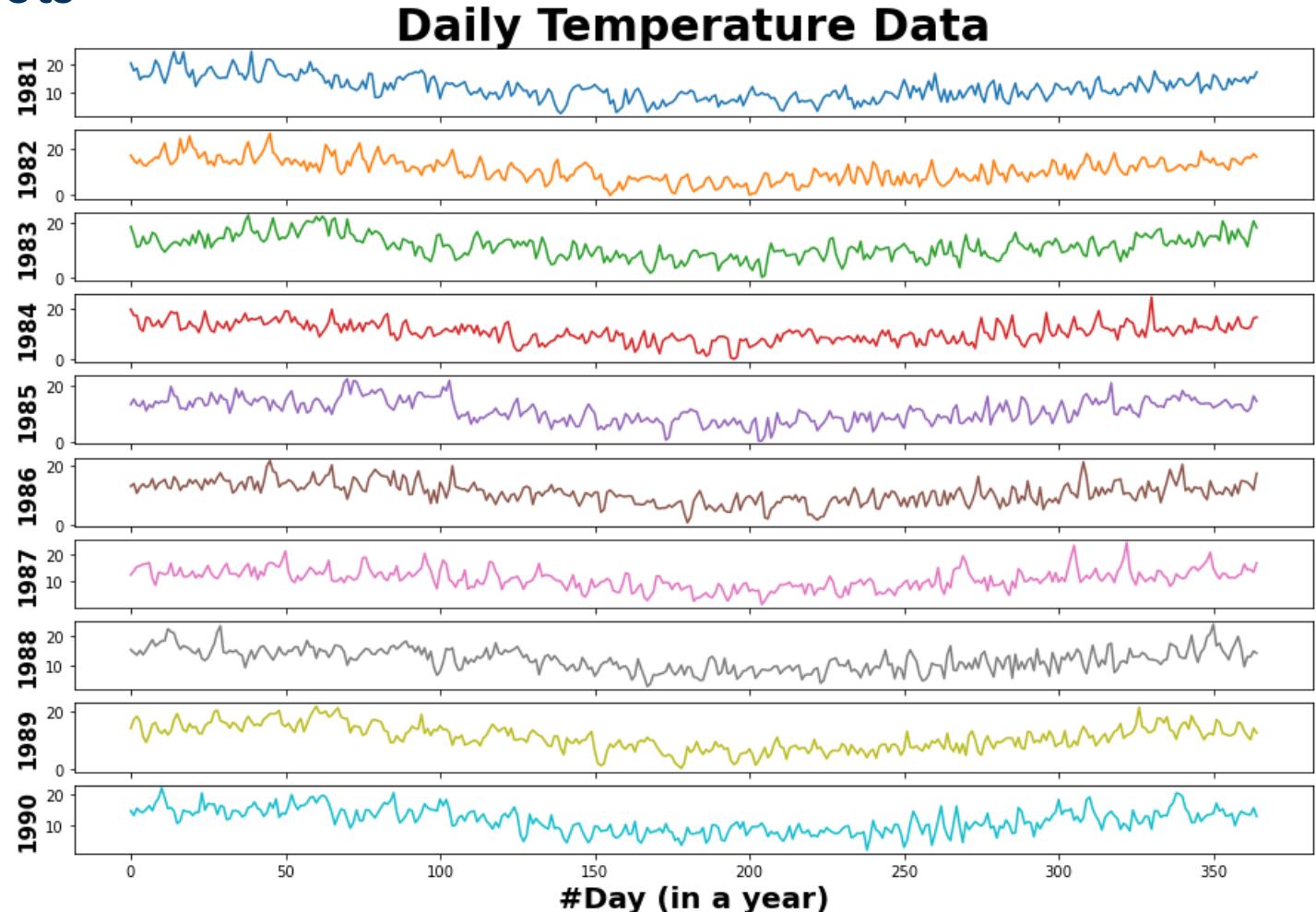
Heatmaps

- Display the cross tabulation of two categorical variables
- Allow to see gradual trends in data



Data visualization: trends

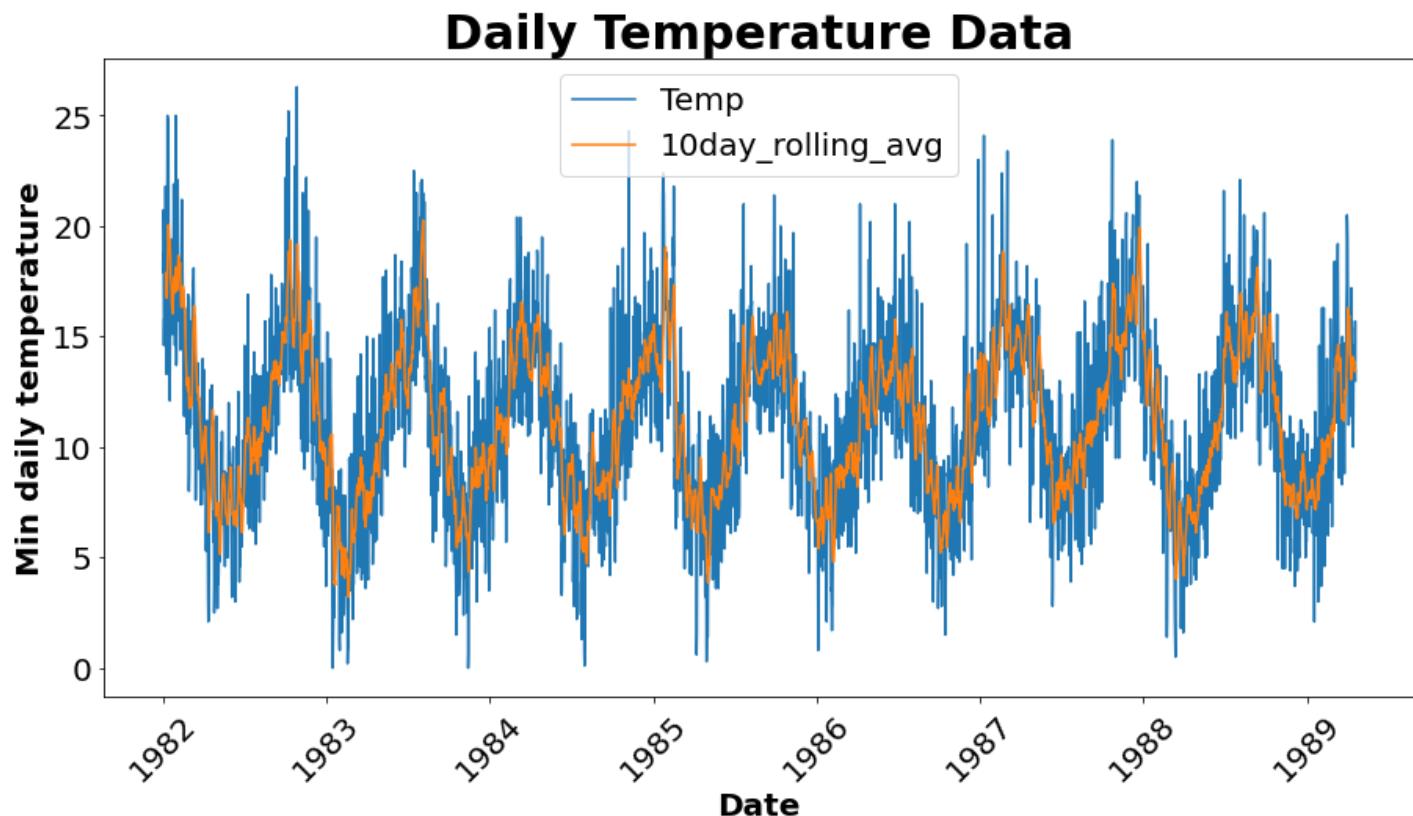
Time Series Plots



Data visualization: trends

Time Series Plots

- moving average and other smoothing functions can be drawn on top of the original time series to perceive trends



Data visualization: remarks

Data visualization must **accurately** convey the data

- **Color scales**
 - Distinguish groups of data from each other
 - Represent data values
 - Highlight data/information
- **Right context** with appropriate
 - title
 - axis labels
 - legends
 - other annotations

Data visualization in Python

Matplotlib

<https://matplotlib.org/>

Seaborn

<https://seaborn.pydata.org/>

Pandas.DataFrame.plot

<https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.plot.html>

Homework...

- Assignment I (see eLearning)
 - Data Understanding:
 - Hands on: Visualization

Contents

- Attributes and Datasets
- Data Summarization
- Data Visualization
- Proximity Measures
- Summary

Proximity Measures

Similarity: a numerical measure indicating how attribute/object/set are alike

Dissimilarity: a numerical measure indicating how attribute/object/set are different

- Proximity usually refers to similarity or dissimilarity

Applied to:

- Attributes/variables: Given the values for i^{th} attribute compare to values j^{th} attribute
- Objects: Given two data points x_i and x_j , the measure deals with the two points (i, j)
- Sets: Given two data groups $X = \{x_1, x_2, \dots\}$ and $Y = \{y_1, y_2, \dots\}$ the measure addresses X and Y or characteristics of the sets

Proximity Measures

- Similarity measure or similarity function
 - Numerical measure that quantifies the similarity between attribute/object/set
 - Indicates how two attribute/object/set are alike
 - The higher value, the more alike
 - Often falls in the range [0,1]: 0 --> no similarity; 1: completely similar

Proximity Measures

- Dissimilarity (or distance) measure
 - Numerical measure that quantifies the difference between attribute/object/set
 - In some sense, the inverse of similarity:
 - The lower value, the more alike
 - Minimum dissimilarity is often 0: completely similar
 - Range $[0, 1]$ or $[0, \infty)$, depending on the definition

Proximity Measures

Similarity and dissimilarity between two objects, x and y

Attribute Type	Dissimilarity	Similarity
Nominal	$d = \begin{cases} 0 & \text{if } x = y \\ 1 & \text{if } x \neq y \end{cases}$	$s = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{if } x \neq y \end{cases}$
Ordinal	$d = x - y /(n - 1)$ (values mapped to integers 0 to $n-1$, where n is the number of values)	$s = 1 - d$
Interval or Ratio	$d = x - y $	$s = -d, s = \frac{1}{1+d}, s = e^{-d},$ $s = 1 - \frac{d - \min_d}{\max_d - \min_d}$

- Nominal: Female and Male
 - Dissimilarity = $d = 1$, Similarity = $s = 0$
- Ordinal: Good and Excellent (Categories: Good, Very Good and Excellent)
 - Dissimilarity = $d = |2 - 0|/2 = 1$, Similarity = $s = 1 - d = 0$

Distances (Dissimilarity): properties

A distance (dissimilarity) d has the following properties:

1. Non-negativity

$$d(x, y) \geq 0 \text{ for all } x \text{ and } y$$

2. Identity

$$d(x, y) = 0 \text{ iff } x = y$$

3. Symmetry

$$d(x, y) = d(y, x) \text{ for all } x \text{ and } y$$

4. Triangle Inequality

$$d(x, z) \leq d(x, y) + d(y, z) \text{ for all } x, y, \text{ and } z$$

A distance that satisfies these properties is a **metric**

Similarities: properties

A similarity s has the following properties:

1. Maximum similarity

$$s(x, y) = 1 \text{ iff } x = y$$

2. Symmetry

$$d(x, y) = d(y, x) \text{ for all } x \text{ and } y$$

Data matrix and proximity matrix

- Data matrix
 - A data matrix of n data points with D dimensions
- Proximity matrix
 - $n \times n$ matrix
 - n data points, but registers only the proximity $p(i, j)$ (typically metric)
 - Usually **symmetric**, thus a **triangular matrix**

$$Data = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1D} \\ x_{21} & x_{22} & \dots & x_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nD} \end{pmatrix} \quad Prox = \begin{pmatrix} 0/1 & & & \\ p(2,1) & 0/1 & & \\ \vdots & \vdots & \ddots & \\ p(n,1) & p(n,2) & \dots & 0/1 \end{pmatrix}$$

0 for a dissimilarity measure
1 for a similarity measure

Proximity measures: examples

- Given the numerical vectors \mathbf{x} and \mathbf{y}

	Measure	Remarks
Euclidean	$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^D (x_i - y_i)^2}$	
Manhattan	$d(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^D x_i - y_i $	Dissimilarity $\mathbf{x} = \mathbf{y} \rightarrow d = 0$
Chebyshev	$d(\mathbf{x}, \mathbf{y}) = \max_i x_i - y_i $	
Cosine	$s(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^D x_i y_i}{\sqrt{(\sum_{i=1}^D x_i^2)(\sum_{i=1}^D y_i^2)}}$	Similarity $\mathbf{x} = \mathbf{y} \rightarrow s = 1$

Proximity measures: examples

Euclidean distance (L_2 norm)

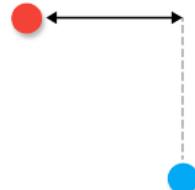
$$d(x,y) = \sqrt{\sum_{i=1}^D (x_i - y_i)^2}$$

Manhattan distance (L_1 norm)

$$d(x,y) = \sum_{i=1}^D |x_i - y_i|$$

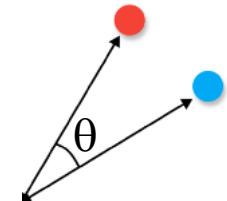
Chebyshev distance

$$d(x,y) = \max_i |x_i - y_i|$$



Cosine similarity

$$s(x,y) = \frac{x^T y}{\|x\| \|y\|} = \cos(\theta)$$



$\|\cdot\|$ is the length of the vector

Euclidean distance

Euclidean distance

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^D (x_i - y_i)^2}$$

where

- D is the number of dimensions (attributes)
- x_i and y_i are, respectively, the entries of the i^{th} attribute of observations \mathbf{x} and \mathbf{y}

Remark:

- Standardization is necessary, if scales differ

Minkowski distance

Minkowski distance is a generalization of Euclidean distance

$$d(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^D |x_i - y_i|^r \right)^{1/r}$$

where

- r is a parameter
- D is the number of dimensions (attributes)
- x_i and y_i are, respectively, the entries of the i^{th} attribute of observations \mathbf{x} and \mathbf{y}

Minkowski Distance: special cases

- $r = 1$: Manhattan distance (City block , taxicab, L_1 norm)

$$d(i, j) = |x_{i1} - y_{i1}| + |x_{i2} - y_{i2}| + \cdots + |x_{iD} - y_{iD}|$$

- A common example of this for binary vectors is the Hamming distance, which is just the number of bits that are different between two binary vectors
- $r = 2$: Euclidean distance

$$d(i, j) = \sqrt{|x_{i1} - y_{i1}|^2 + |x_{i2} - y_{i2}|^2 + \cdots + |x_{iD} - y_{iD}|^2}$$

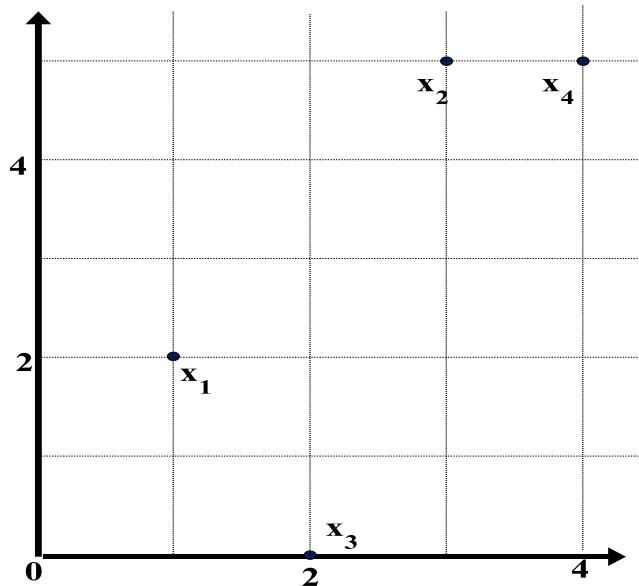
- $r \rightarrow \infty$: “supremum” distance (L_{\max} norm, L_∞ norm)

$$d(i, j) = \lim_{r \rightarrow \infty} \sqrt[r]{|x_{i1} - y_{i1}|^r + |x_{i2} - y_{i2}|^r + \cdots + |x_{iD} - y_{iD}|^r}$$

- This is the maximum difference between any component of the vectors

Data matrix and distance matrix: examples

point	attribute 1	attribute 2
x1	1	2
x2	3	5
x3	2	0
x4	4	5



Manhattan (L_1)

L_1	x1	x2	x3	x4
x1	0			
x2	5	0		
x3	3	6	0	
x4	6	1	7	0

Euclidean (L_2)

L_2	x1	x2	x3	x4
x1	0			
x2	3.61	0		
x3	2.24	5.1	0	
x4	4.24	1	5.39	0

Supremum (L_∞)

L_∞	x1	x2	x3	x4
x1	0			
x2	3	0		
x3	2	5	0	
x4	3	1	5	0

Proximity measures: binary data

Given **two objects** (vectors) with **Binary** entries

- Compute the contingency table
 - f_{11} = number of attributes where **x** was **1** and **y** was **1** (matching)
 - f_{00} = number of attributes where **x** was **0** and **y** was **0** (matching)
 - f_{10} = number of attributes where **x** was **1** and **y** was **0** (not matching)
 - f_{01} = number of attributes where **x** was **0** and **y** was **1** (not matching)

		Object y		
		1	0	sum
Object x	1	f_{11}	f_{10}	$f_{11}+f_{10}$
	0	f_{01}	f_{00}	$f_{01}+f_{00}$
sum		$f_{11}+f_{01}$	$f_{10}+f_{00}$	

Proximity measures: binary data

		Object y	
		1	0
Object x	1	f_{11}	f_{10}
	0	f_{01}	f_{00}
	sum	$f_{11}+f_{01}$	$f_{10}+f_{00}$

Simple matching coefficient

$$\bullet \ SMC = \frac{\text{number of matches}}{\text{number of matches} + \text{number of no-matches}} = \frac{f_{11}+f_{00}}{f_{11}+f_{00}+f_{10}+f_{01}}$$

Jaccard coefficient

(useful for asymmetric binary attributes)

$$\bullet \ J = \frac{\text{number of 1-1 matches}}{\text{number of non 0-0 matches}} = \frac{f_{11}}{f_{11}+f_{10}+f_{01}}$$

Hamming distance

(useful for asymmetric binary attributes)

$$\bullet \ SMC = \frac{\text{number of no-matches}}{\text{number of matches} + \text{number of no-matches}} = \frac{f_{10}+f_{01}}{f_{11}+f_{00}+f_{10}+f_{01}}$$

Proximity measures: binary data

Example:

$x = [1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$

$y = [0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1]$

- $f_{11} = 0$

- $f_{00} = 7$

- $f_{10} = 1$

- $f_{01} = 2$

		Object y		
		1	0	sum
Object x	1	0	1	1
	0	2	7	9
sum		2	8	

- Simple matching coefficient: $SMC = \frac{f_{11}+f_{00}}{f_{11}+f_{00}+f_{10}+f_{01}} = \frac{7}{10} = 0.7$

- Jaccard coefficient: $J = \frac{f_{11}}{f_{11}+f_{10}+f_{01}} = \frac{0}{3} = 0$

- Hamming distance: $SMC = \frac{f_{10}+f_{01}}{f_{11}+f_{00}+f_{10}+f_{01}} = \frac{1+2}{10} = 0.3$

Proximity measures: example

Dissimilarity between asymmetric binary variables

Name	Gender	Fever	Cough	Test-1	Test-2	Test-3	Test-4
Jack	M	Y	N	P	N	N	N
Mary	F	Y	N	P	N	P	N
Jim	M	Y	P	N	N	N	N

- Gender is a symmetric attribute (not counted in)
- The remaining attributes are asymmetric binary
- Let the values Y and P be 1, and the value N be 0
- Distance: $DABV = \frac{f_{10}+f_{01}}{f_{11}+f_{01}+f_{10}}$
- $d(jack, mary) = \frac{0+1}{2+0+1} = 0.33$
- $d(jack, jim) = \frac{1+1}{1+1+1} = 0.67$
- $d(jim, mary) = \frac{1+2}{1+1+2} = 0.75$

		Mary		Σ_{row}
		1	0	
Jack	1	2	0	2
	0	1	3	4
Σ_{col}		3	3	6

		Jim		Σ_{row}
		1	0	
Jack	1	1	1	2
	0	1	3	4
Σ_{col}		2	4	6

		Mary		Σ_{row}
		1	0	
Jim	1	1	1	2
	0	2	2	4
Σ_{col}		3	3	6

Proximity measures and attributes types

Nominal

- Hamming distance (normalized)
 - Example: color, profession

Proximity measures and attributes types

Ordinal: Can be treated as interval-scaled

- Replace an *ordinal variable value* by its **rank**: $r_{if} \in \{1, \dots, M_f\}$
- **Map the range** of each variable into **[0, 1]** by replacing i^{th} object in the f^{th} variable by $z_{if} = \frac{r_{if}-1}{M_f-1}$
- Compute the dissimilarity using methods for interval-scaled variables
 - Euclidian distance, Manhattan (city block), cosine similarity, etc.

Example

- Acceptable -> 0; Good -> 1/3; Very good -> 2/3; Excellent -> 1

Then distance: $d(\text{Acceptable}, \text{Excellent}) = 1$, $d(\text{Very good}, \text{Excellent}) = 1/3$

Proximity measures and attributes types

Mixed type

- A data set may contain all attribute types
 - Nominal, symmetric binary, asymmetric binary, numeric, and ordinal
- Global proximity measure $d(i, j)$
 - a combination of measures applied individually to each attribute

$$d(i, j) = \frac{\sum_{f=1}^D w_f d_f(i, j)}{\sum_{f=1}^D w_f}$$

where

- w_f is a weight for the f^{th} attribute
- to control missing values on the data ($w_f = 0$, if the f^{th} attribute is missing either in i or j objects)

Proximity measures and attributes types

Mixed type

- A data set may contain all attribute types
 - Nominal, symmetric binary, asymmetric binary, numeric, and ordinal
- Global proximity measure $d(i, j)$
 - a combination of measures applied individually to each attribute

$$d(i, j) = \frac{\sum_{f=1}^D w_f d_f(i, j)}{\sum_{f=1}^D w_f}$$

- w_f is a weight for the f^{th} attribute
- to control missing values on the data ($w_f = 0$, if the f^{th} attribute is missing either in i or j objects)

Cosine similarity of two vectors

Cosine measure

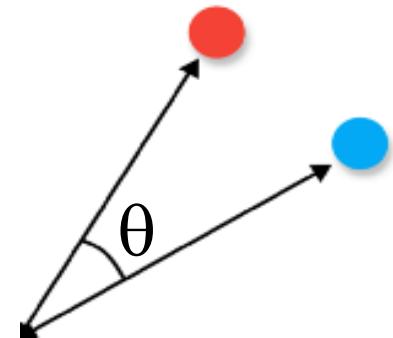
- If d_1 and d_2 are two vectors, then

$$\cos(d_1, d_2) = \frac{d_1 \cdot d_2}{\|d_1\| \times \|d_2\|} = \cos(\theta)$$

where

- indicates vector dot product

$\|d\|$ the length of vector d



Cosine similarity of two vectors

- A **document** can be represented by a bag of terms or a long vector, with each attribute recording the *frequency* of a particular term (such as word, keyword, or phrase) in the document

Document	team	coach	hockey	baseball	soccer	penalty	score	win	loss	season
Document1	5	0	3	0	2	0	0	2	0	0
Document2	3	0	2	0	1	1	0	1	0	1
Document3	0	7	0	2	1	0	0	3	0	0
Document4	0	1	0	0	1	2	2	0	3	0

- Other example: Gene features in micro-arrays
- Applications: Information retrieval, biologic taxonomy, gene feature mapping, etc.

Cosine similarity of two vectors: example

Find the **similarity** between documents 1 and 2.

$$d_1 = (5, 0, 3, 0, 2, 0, 0, 2, 0, 0)$$

$$d_2 = (3, 0, 2, 0, 1, 1, 0, 1, 0, 1)$$

1. calculate vector **dot product**

$$d_1 \bullet d_2 = 5 \times 3 + 0 \times 0 + 3 \times 2 + 0 \times 0 + 2 \times 1 + 0 \times 1 + 0 \times 1 + 2 \times 1 + 0 \times 0 + 0 \times 1 = 25$$

2. calculate $\|d_1\|$ and $\|d_2\|$

$$\|d_1\| = \sqrt{5 \times 5 + 0 \times 0 + 3 \times 3 + 0 \times 0 + 2 \times 2 + 0 \times 0 + 0 \times 0 + 2 \times 2 + 0 \times 0 + 0 \times 0} = 6.481$$

$$\|d_2\| = \sqrt{3 \times 3 + 0 \times 0 + 2 \times 2 + 0 \times 0 + 1 \times 1 + 1 \times 1 + 0 \times 0 + 1 \times 1 + 0 \times 0 + 1 \times 1} = 4.12$$

3. Calculate **cosine similarity**

$$\cos(d_1, d_2) = \frac{d_1 \bullet d_2}{\|d_1\| \times \|d_2\|} = 25 / (6.481 \times 4.12) = 0.94$$

Proximity measures: issues in calculation

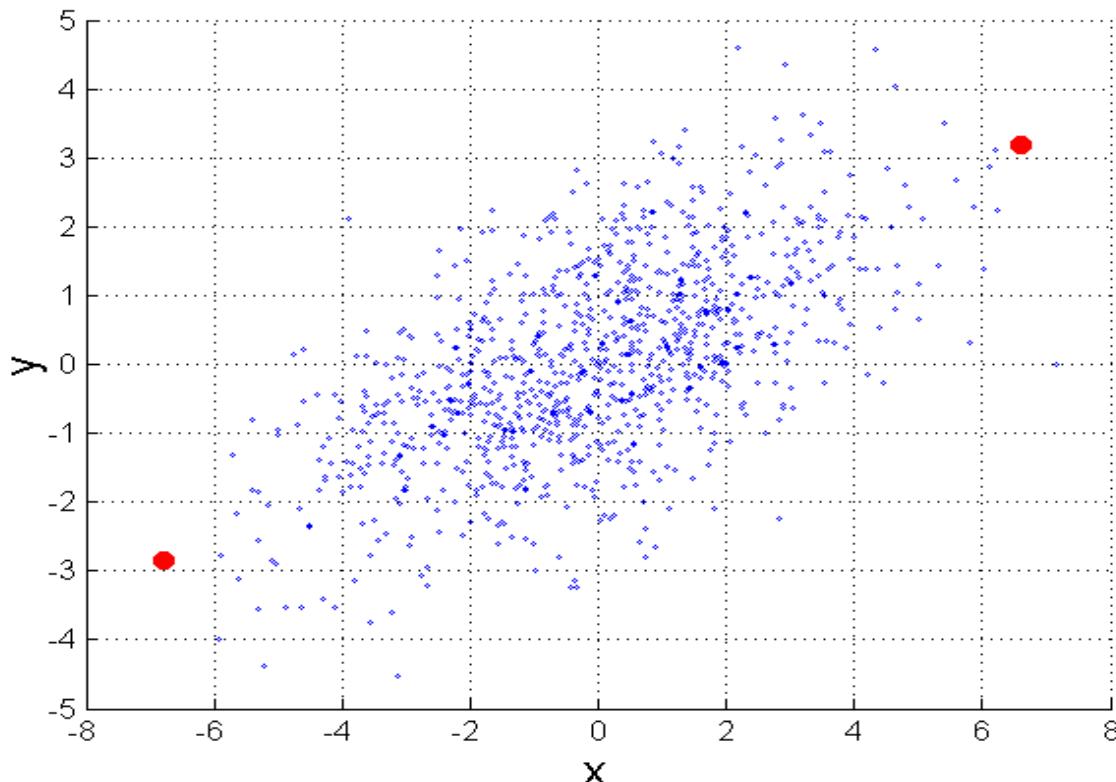
How to deal with

- features having **different ranges**
 - normalization before calculation (advisable Euclidean distance)
- **correlated features**
 - Mahalanobis distance is better than Euclidean
- features of **different types** (quantitative and qualitative)

Proximity measures: correlated features

Mahalanobis distance

$$d(x,y) = ((x-y)^T \Sigma^{-1} (x-y))^{-1/2}, \text{ where } \Sigma \text{ is the covariance matrix}$$



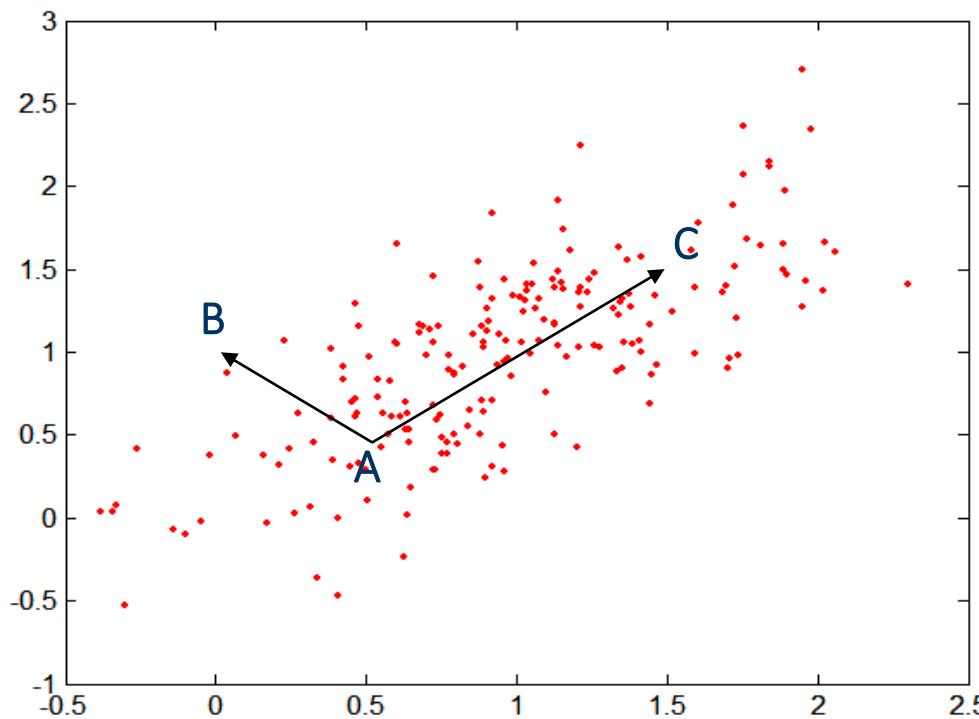
- For red points:
 - Euclidean distance is 14.7,
 - Mahalanobis distance is 6.

Proximity measures: correlated features

Dispersion (spread)

- Covariance matrix

$$\Sigma = \begin{bmatrix} 0.58 & 0.25 \\ 0.25 & 0.25 \end{bmatrix}$$



A: (0.5, 0.5)

B: (0, 1)

C: (1.5, 1.5)

$$Mahal(A, B) = 5$$

$$Mahal(A, C) = 4$$

Correlation vs Cosine vs Euclidean Distance

Compare the three proximity measures according to their behavior under variable transformation

- scaling: multiplication by a value
- translation: adding a constant

Property	Cosine	Correlation	Euclidean Distance
Invariant to scaling (multiplication)	Yes	Yes	No
Invariant to translation (addition)	No	Yes	No

- Consider the example
 - $x = (1, 2, 4, 3, 0, 0, 0)$, $y = (1, 2, 3, 4, 0, 0, 0)$
 - $y_s = y * 2$ (scaled version of y), $y_t = y + 5$ (translated version)

Measure	(x, y)	(x, y_s)	(x, y_t)
Cosine	0.9667	0.9667	0.7940
Correlation	0.9429	0.9429	0.9429
Euclidean Distance	1.4142	5.8310	14.2127

Homework...

- Assignment I (see eLearning)
 - Notes with exercises: Ex. 1.3 and 1.4

Contents

- Attributes and Datasets
- Data Summarization
- Data Visualization: Amounts, Distributions, Associations, Trends
- Proximity Measures
- **Summary**

Summary

- **Types** and **scales** of attributes
 - nominal, ordinal, interval-scaled, ratio-scaled
- Many types of data sets
- **Gain insight** into the data by:
 - Basic statistical data **description**: frequency, central tendency, dispersion / spread
 - Data **visualization**: map data onto graphical primitives
 - Measure data **similarity**

Data Understanding - the beginning of **Data Preparation/Preprocessing**

Bibliography

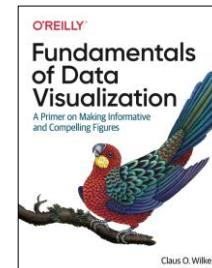
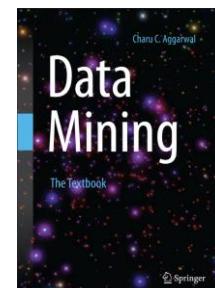
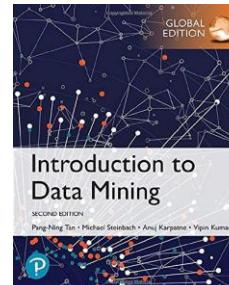
Introduction to Data Mining, Pang-Ning Tan, Michael Steinbach, Anuj Karpatne, Vipin Kumar, *Pearson*, 2019 (chap 2.1 & 2.4)

Data Mining, the Textbook, Charu C. Aggarwal, *Springer*, 2015 (chap 1.3 & chap 2)

Fundamentals of Data Visualization, Claus O. Wilke, *O'Reilly*, 2022

<https://www.pythongraph-gallery.com/parallel-coordinate-plot-plotly>

<https://towardsdatascience.com/9-distance-measures-in-data-science-918109d069fa>



Data Mining

Overview

Raquel Sebastião

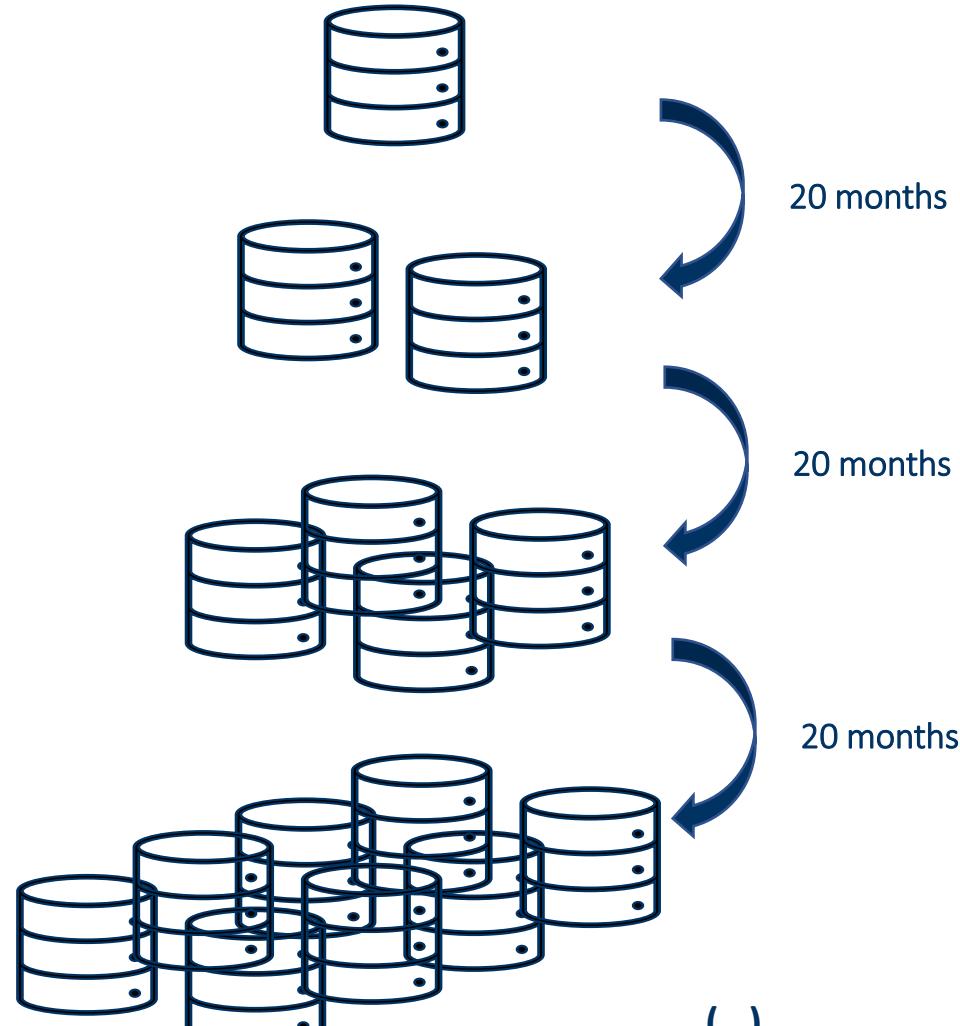
Departamento de Eletrónica, Telecomunicações e Informática

Universidade de Aveiro

raquel.sebastiao@ua.pt

2022/2023

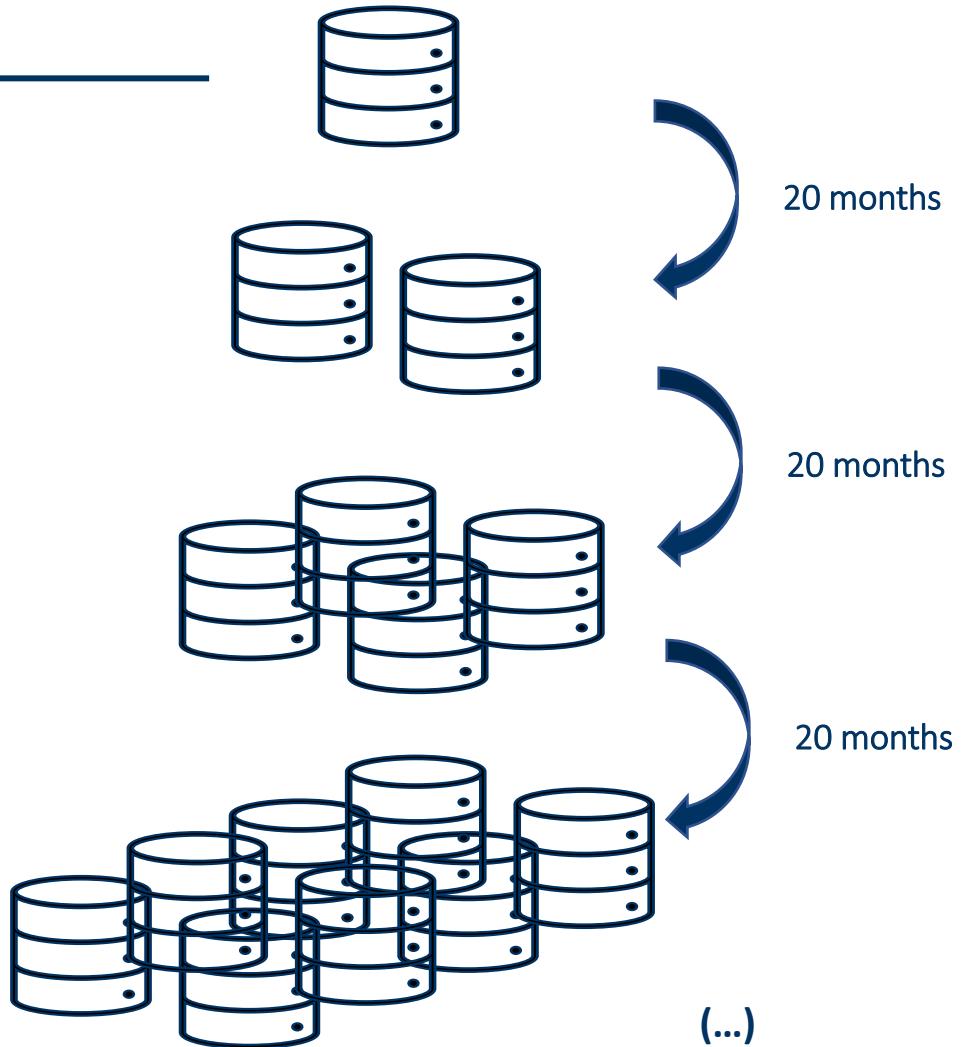
Why data mining?



Overwhelmed with data

Why data mining?

“We are drowning in data,
but starving for knowledge.”



Motivation

“Necessity is the mother of invention”
proverb (Plato)

The amount and type of data are ever-increasing

- Several data collection methods have been advanced
- Increase in the storage capacity and computational power

Data contain potentially useful (and interesting) information

Overwhelmed with data

- Manual inspection is almost impossible
- Automatic data analysis methods are required

From Data to Knowledge

Data

- Facts, numbers, or text that can be processed by a computer

Metadata

- Data about the data itself such as logical database design or data dictionary definitions

Information

- The patterns, associations, or relationships among all this data can provide information

Knowledge

- Information can be converted into knowledge about historical patterns and future trends

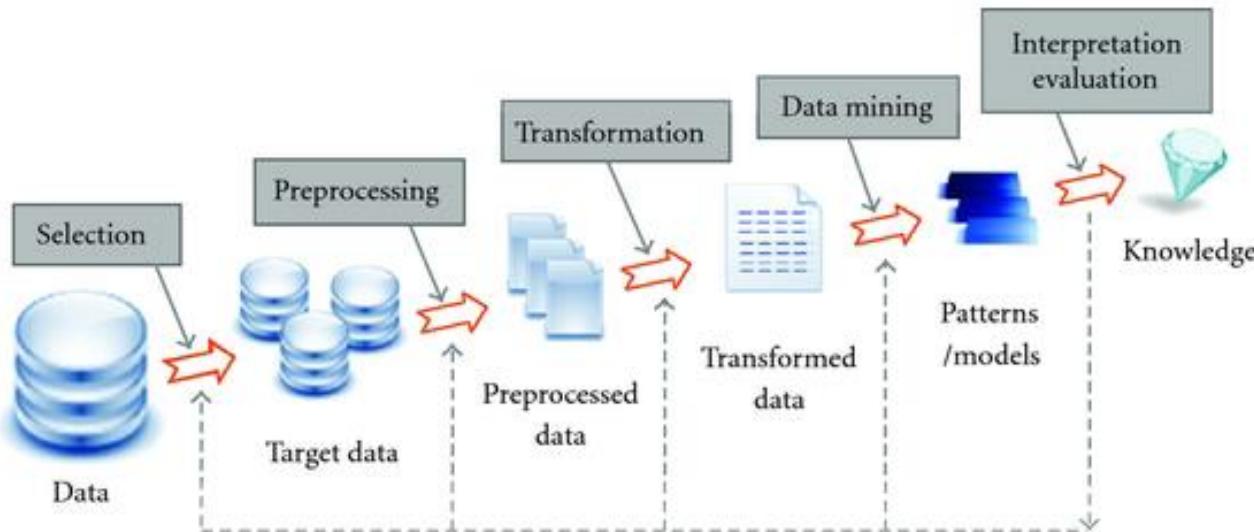
From Data to Knowledge

Criteria to assess Knowledge:

- correctness (probability, success in tests);
- generality (domain and conditions of validity);
- usefulness (relevance, predictive power);
- comprehensibility (simplicity, clarity, parsimony);
- novelty (previously unknown, unexpected)

Data Mining is the process of knowledge discovery from data!

Knowledge Discovery from Data (KDD)



Data Mining is the process of knowledge discovery from data!

Data mining: possible definitions

“is the process of automatically discovering useful information in large data repositories”

Introduction to Data Mining, Tan et al.

“is the process of discovering interesting patterns from massive amounts of data”

Data Mining: Concepts and Techniques, Han et al.

“is defined as the process of discovering patterns in data”

Data Mining: Practical Machine Learning Tools and Techniques, Witten et al.

“is the study of collecting, cleaning, processing, analyzing, and gaining useful insights from data”

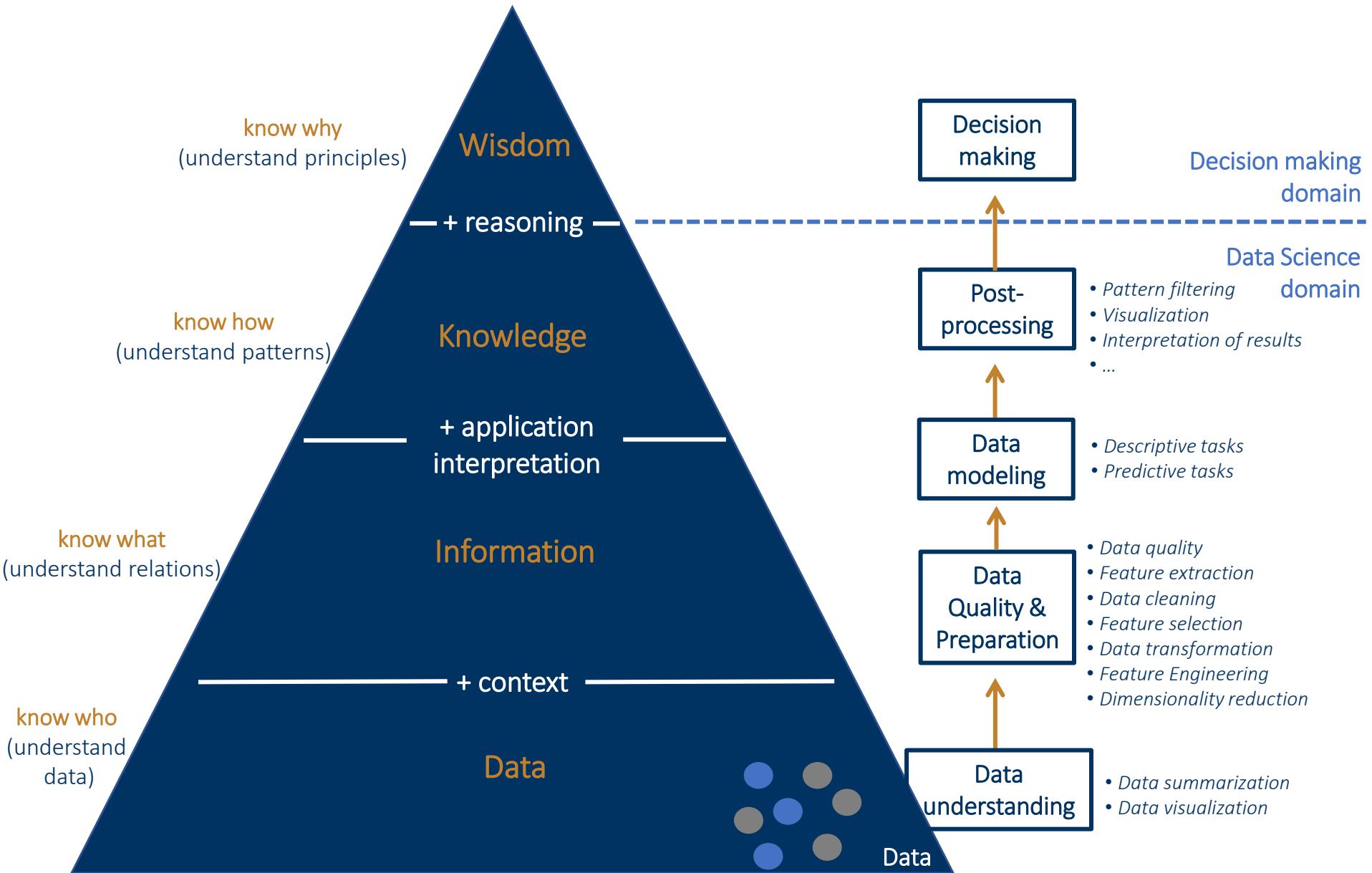
Data Mining: The Textbook, Aggarwal

Humorous definition:

Data Mining, noun 1. Torturing the data until it confesses ... and if you torture it enough, you can get it to confess to anything. ☺
ACM SIGKDD

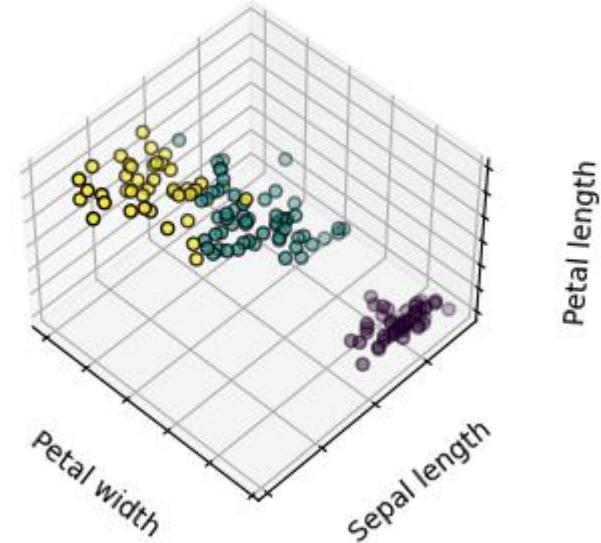
Data mining

Involves the manipulation of large amounts of **data**, usually stored in databases, in order to **discover** implicit, previously unknown, and potentially useful **information** that can be conveyed into **knowledge**



Clustering :example

- Finding groups of items that are similar
- Clustering is *unsupervised*
 - The class of an example is not known
- Success often measured **subjectively**

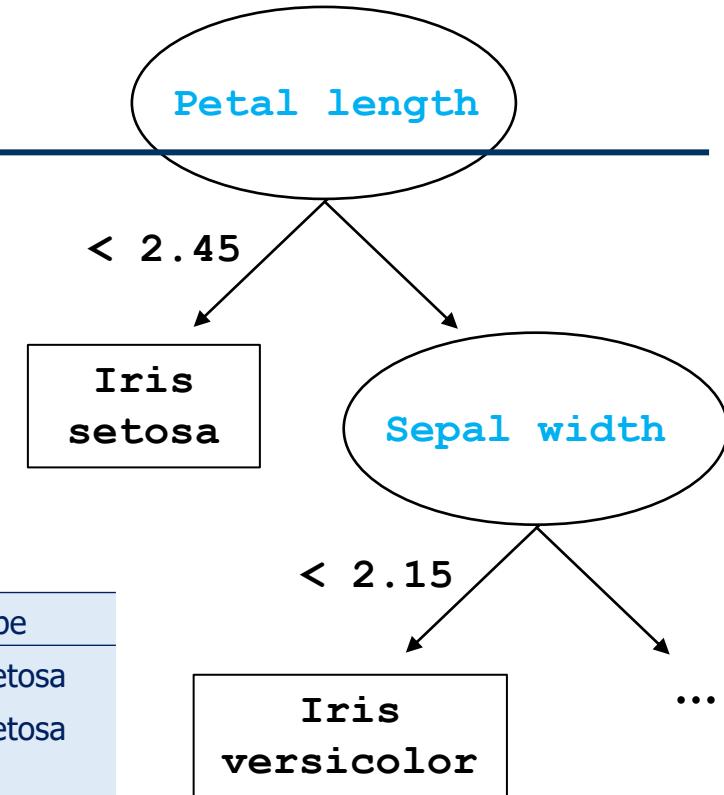


	Sepal length	Sepal width	Petal length	Petal width	Type
1	5.1	3.5	1.4	0.2	Iris setosa
2	4.9	3.0	1.4	0.2	Iris setosa
...					
51	7.0	3.2	4.7	1.4	Iris versicolor
52	6.4	3.2	4.5	1.5	Iris versicolor
...					
101	6.3	3.3	6.0	2.5	Iris virginica
102	5.8	2.7	5.1	1.9	Iris virginica
...					

Classification: example

- Predicting the target/class
- Classification is *supervised*
 - The class of an example is known
- Success measured *objectively*

	Sepal length	Sepal width	Petal length	Petal width	Type
1	5.1	3.5	1.4	0.2	Iris setosa
2	4.9	3.0	1.4	0.2	Iris setosa
...					
51	7.0	3.2	4.7	1.4	Iris versicolor
52	6.4	3.2	4.5	1.5	Iris versicolor
...					
101	6.3	3.3	6.0	2.5	Iris virginica
102	5.8	2.7	5.1	1.9	Iris virginica
...					



If petal length < 2.45 then Iris setosa
If sepal width < 2.10 then Iris versicolor
...

Practical assignment: key issues

- Presentation:
 - Introduction to the **problem** and **application domain**, description of the **dataset**, and summarization of **results**, **conclusions**, and further research of the **reference literature**.
- Project (and analysis of the report)
 - **Data Structure**
 - what to measure? pre-processing steps? ...
 - **Model Structure**
 - what type of model(s) should we build? ...
 - **Score Function**
 - how to evaluate the obtained models? ...
 - **Optimization and Search Method**
 - how to search and optimize the models in the context of the selected structure? ...
 - **Data Management Strategy**
 - how to handle the data efficiently during model construction/evaluation? ...

Data mining: origins

Originally proposed to solve perceptual tasks like:

- Optical character recognition
- Face recognition
- Voice recognition
- ...

which the humans can perform well, however there is
no mathematical model to address the problems

Data mining: the role of machine learning

Learning computational models from examples

- WHY?
 - the representation of the problem is hard (or even impossible) to define with equations
- WHAT?
 - The free parameters of the model
- WHAT FOR?
 - New examples can be processed by the models to help in new decisions

Example: in medical image analysis is the classification of objects such as lesions into certain categories (e.g., abnormal or normal, tumor or non-tumor)

Data mining: main disciplines

Machine Learning:

- learning from examples to construct models
 - Computer Science Community

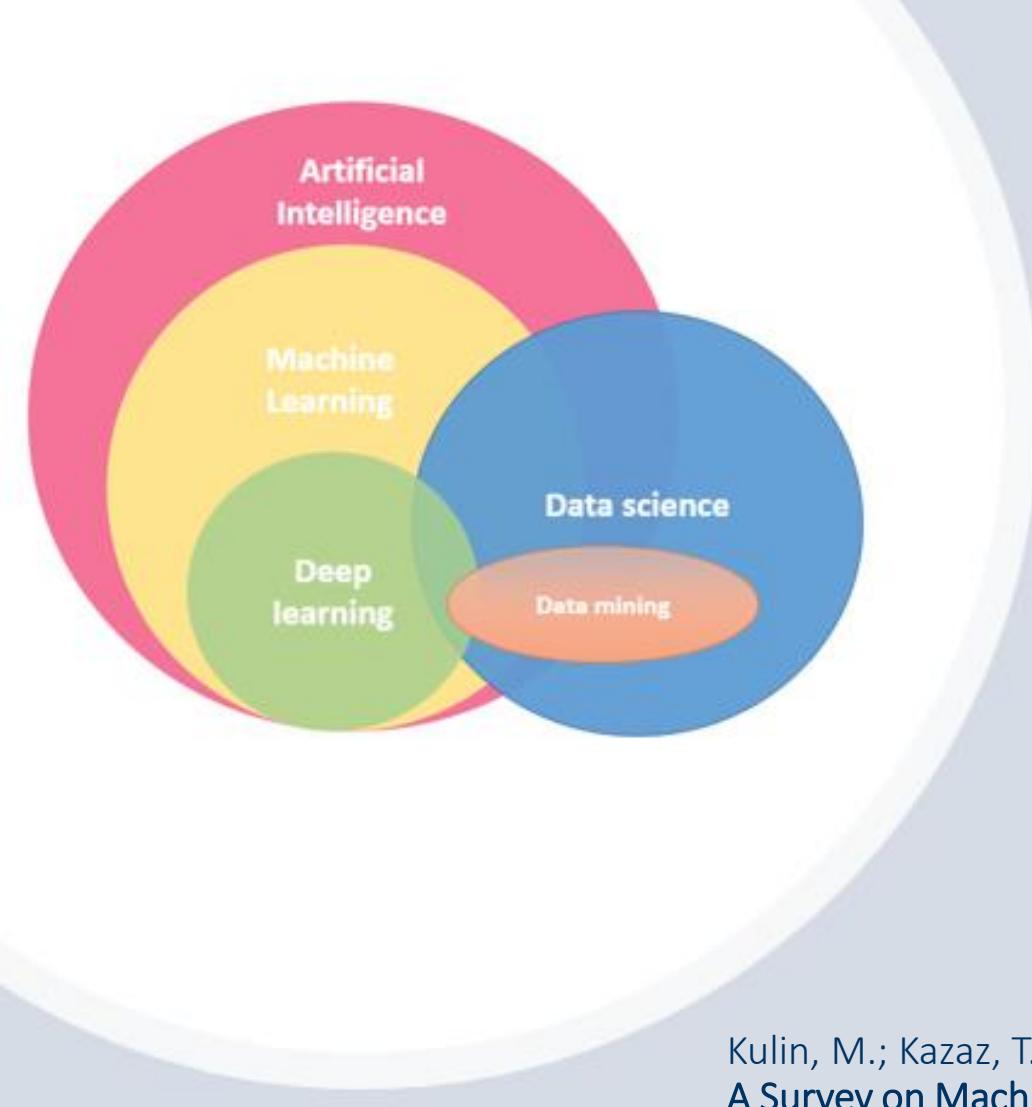
Pattern Recognition

- identifying patterns not necessarily with a learning phase
 - Electrical Engineering Community

Data Mining

- involves the manipulation of large amounts of data, usually stored in databases, in order to discover patterns

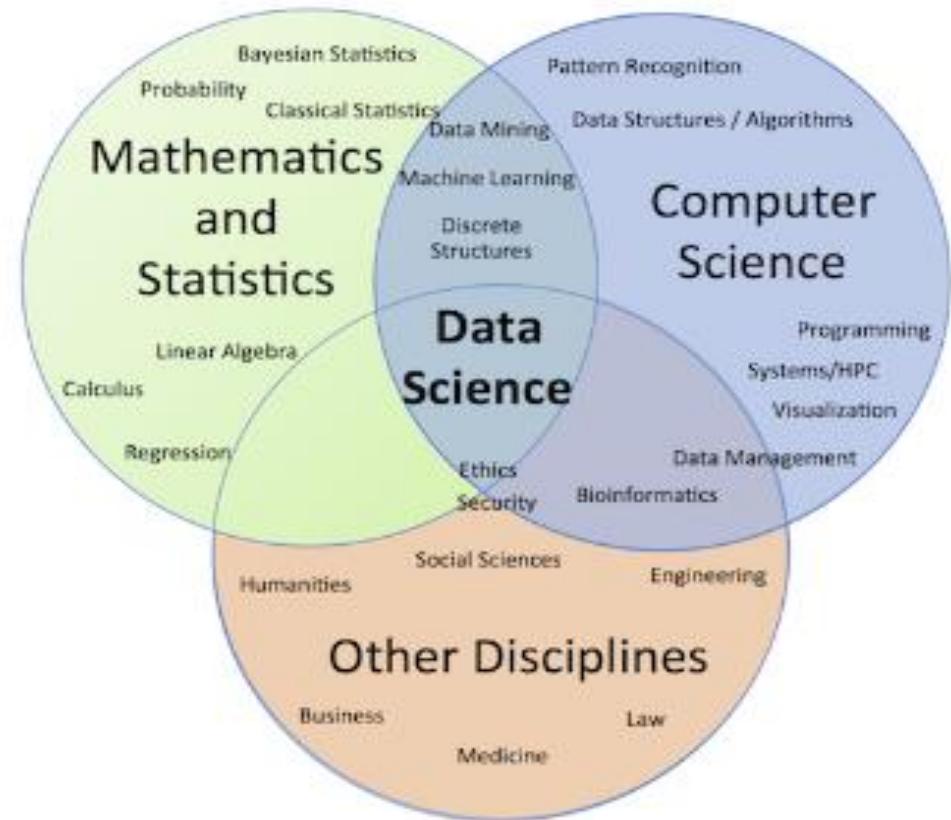
Generally, all use similar algorithms or methods



Data Science

Kulin, M.; Kazaz, T.; De Poorter, E.; Moerman, I.
A Survey on Machine Learning-Based Performance Improvement
of Wireless Networks: PHY, MAC and Network Layer.
Electronics 2021, 10, 318.
<https://doi.org/10.3390/electronics10030318>

Data Science



ACM Data Science Task Force
<https://dstf.acm.org/>

Data mining: tasks

Exploratory Data Analysis

- Summarization
- Visualization tools

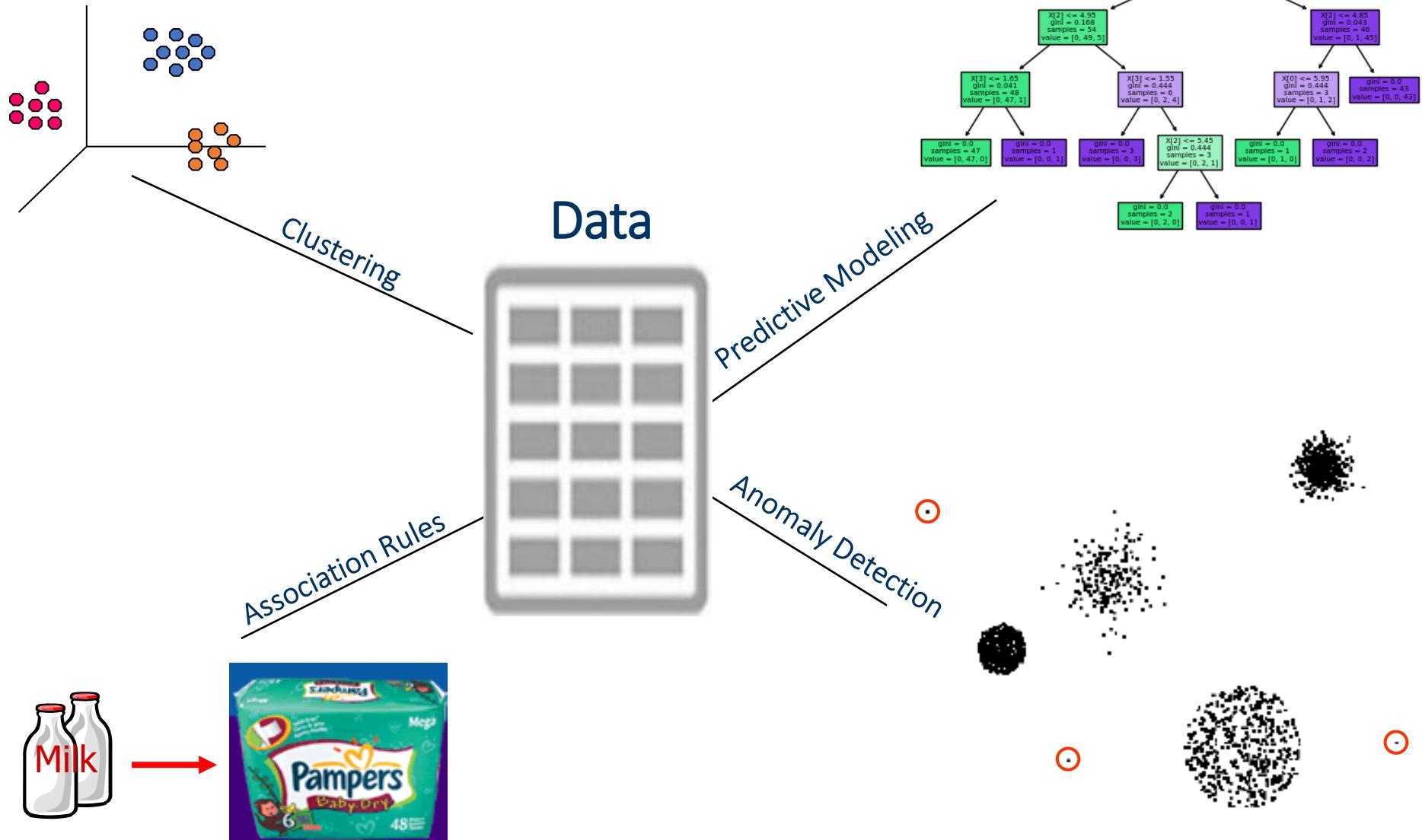
Descriptive tasks

- Find human-interpretable patterns that describe the data.
 - Clustering
 - Association analysis
 - Anomaly detection

Predictive tasks

- Use some variables (features) to predict unknown or future values of other variables.
 - Classification
 - Regression

Data mining: tasks



Classification vs. Association rules: examples

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	Normal	False	Yes
...

Classification rule:

- predicts value of a given attribute (the classification of an example)

```
If outlook = sunny and humidity = high  
then play = no
```

Association rule:

- predicts value of arbitrary attribute (or combination)

```
If temperature = cool then humidity = normal  
If humidity = normal and windy = false  
then play = yes  
If outlook = sunny and play = no  
then humidity = high  
If windy = false and play = no  
then outlook = sunny and humidity = high
```

The role of domain knowledge: example

```
If leaf condition is normal  
and stem condition is abnormal  
and stem cankers is below soil line  
and canker lesion color is brown  
then  
diagnosis is rhizoctonia root rot
```

```
If leaf malformation is absent  
and stem condition is abnormal  
and stem cankers is below soil line  
and canker lesion color is brown  
then  
diagnosis is rhizoctonia root rot
```

- But in this domain, “leaf condition is normal” implies “leaf malformation is absent”!

Key issues in a Data Mining Project

Data Structure

- what to measure? pre-processing steps?

Model Structure

- what type of model(s) should we build?

Score Function

- how to evaluate the obtained models?

Optimization and Search Method

- how to search and optimize the models in the context of the selected structure?

Data Management Strategy

- how to handle the data efficiently during model construction/evaluation?

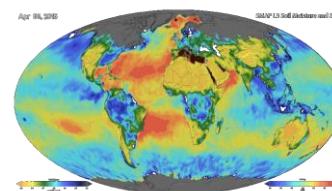
BIG data



E-Commerce

The amount and type of data are ever-increasing

- Several **data collection** methods have been advanced
- Increase in the **storage capacity** and **computational power**



Surface Temperature of Earth



Bio-informatics



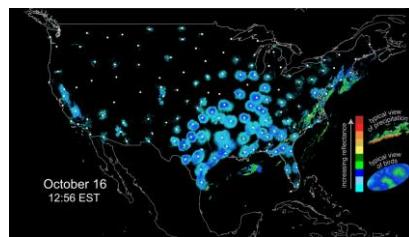
Sky Survey Data

Data is useless!

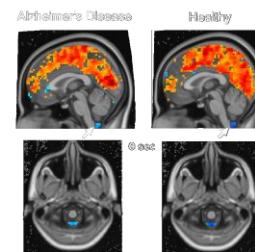
convert it to useful information and into knowledge



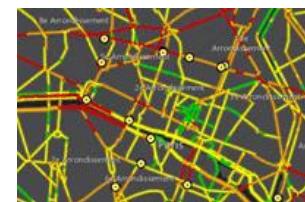
Cyber Security



Bird migration



fMRI Data from Brain



Traffic Patterns

Data mining and Big data

The amount and type of data are ever-increasing

Big Data has three dimensions described by the **Three V's** Gandomi and Haider, 2015:

- Volume: massive, high dimensional, distributed data sets
- Velocity: generated at high-speed
- Variety: heterogeneous, complex

Data mining and Big data

Traditional techniques may be **unsuitable** as new applications provide data that is:

- Large-scale
- High dimensional
- Complex
- Heterogeneous

A key **challenge** for data mining is to develop techniques that can cope with **Big Data**.

Data mining tasks: challenges

- Scalability: capacity of dealing with massive data sets (large number of objects);
- Dimensionality: capacity of dealing with lots of attributes/features for each object;
- Complex and Heterogeneous Data: Different type of attributes;
- Data Quality: Usually data does not result of a designed data collection as in traditional statistical experiment;
- Data Ownership and Distribution: data stored and owned by various organizations;
- Privacy Preservation: development of data mining has the capacity to compromise privacy in ways not previously possible;
- Streaming Data: extracting information of an ordered sequence of data records.

Data mining: applications

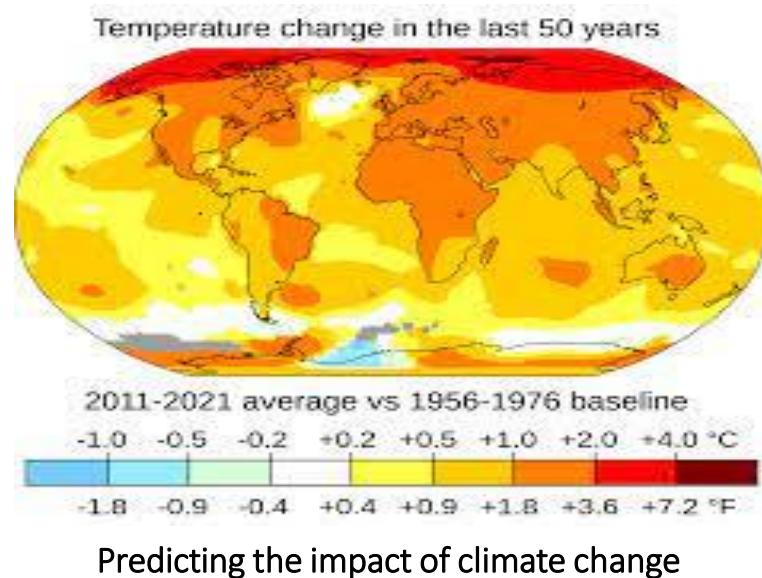
- Collaborative analysis & recommender systems
- Basket data analysis to targeted marketing
- Biological and medical data analysis: classification, cluster analysis (microarray data analysis), biological sequence analysis, biological network analysis
- Banking, fraud analysis, stock market analysis
- Telecommunications
- Diagnosis of machine faults

Data mining: solving society's major problems?



Improving health care and reducing costs

Data mining: solving society's major problems?



Data mining: solving society's major problems?



Finding alternative/ green energy sources

Data mining: solving society's major problems?



Reducing hunger and poverty by increasing agriculture production

Summing up...

- Data to be mined
 - Database data, data warehouse, heterogeneous data, transactional data, stream, spatiotemporal, time-series, sequence, text and web, multi-media, social data, information networks, ...
- Data mining tasks
 - Exploratory Data Analysis
 - Descriptive vs. predictive
 - Clustering
 - Association analysis
 - Anomaly detection
 - Prediction (classification/regression)
- Techniques
 - Data-intensive, data warehouse (OLAP), pattern recognition, statistics, machine learning, visualization, ...
- Applications
 - Retail, telecommunications, banking, fraud analysis, bio-data mining, stock market analysis, text mining, web mining, fault diagnosis, ...

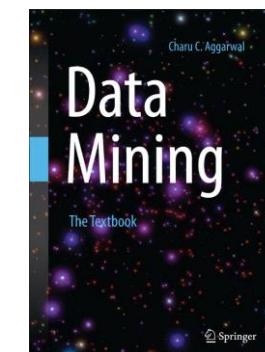
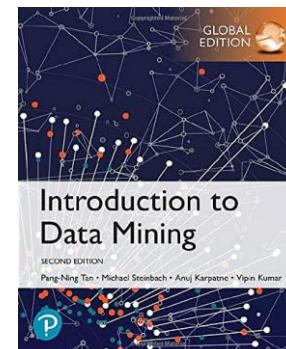
Bibliography

Introduction to Data Mining, Pang-Ning Tan, Michael Steinbach, Anuj Karpatne, Vipin Kumar, *Pearson*, 2019

Data Mining, the Textbook, Charu C. Aggarwal, *Springer*, 2015

Sebastian Raschka, **Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning**.

<https://arxiv.org/abs/1811.12808>



Some online resources

- [SIGKDD](#)
- [Data Science, Machine Learning, AI & Analytics – Kdnuggets](#)
- [Category: Machine Learning - VideoLectures.NET](#)
- [UCI Machine Learning Repository](#)
- [Kaggle: Your Machine Learning and Data Science Community](#)
- [Dataset Search \(google.com\)](#)

Lectures and Laboratories

Weekly session (3h)

- Exposition (lecture): Slides will be available
- Paper and pencil: Solving a couple of exercises from the booklet exercises
- Programming: Jupyter Notebooks to practice the use of different packages

Mandatory tools

- Jupyter Environment
- Python Programming (NumPy, Pandas, Matplotlib, Seaborn, Scikit-learn, Mlxtend)
- ANACONDA could be used to manage all the facilities

Assessment

- Practical assignment (groups with 3 students)
 - Presentation & project
 - Presentation of the application and the available data set: **10%**
 - Project Notebook. Using the data set developing an exploratory data mining project: **25%**
- Participation (lab exercises, lectures and discussion): **10%**
- Written exam: **55%**

Practical assignment: groups

Please, send an email (up to 5th October) with the names of the group members.

Data Mining

Data Preparation

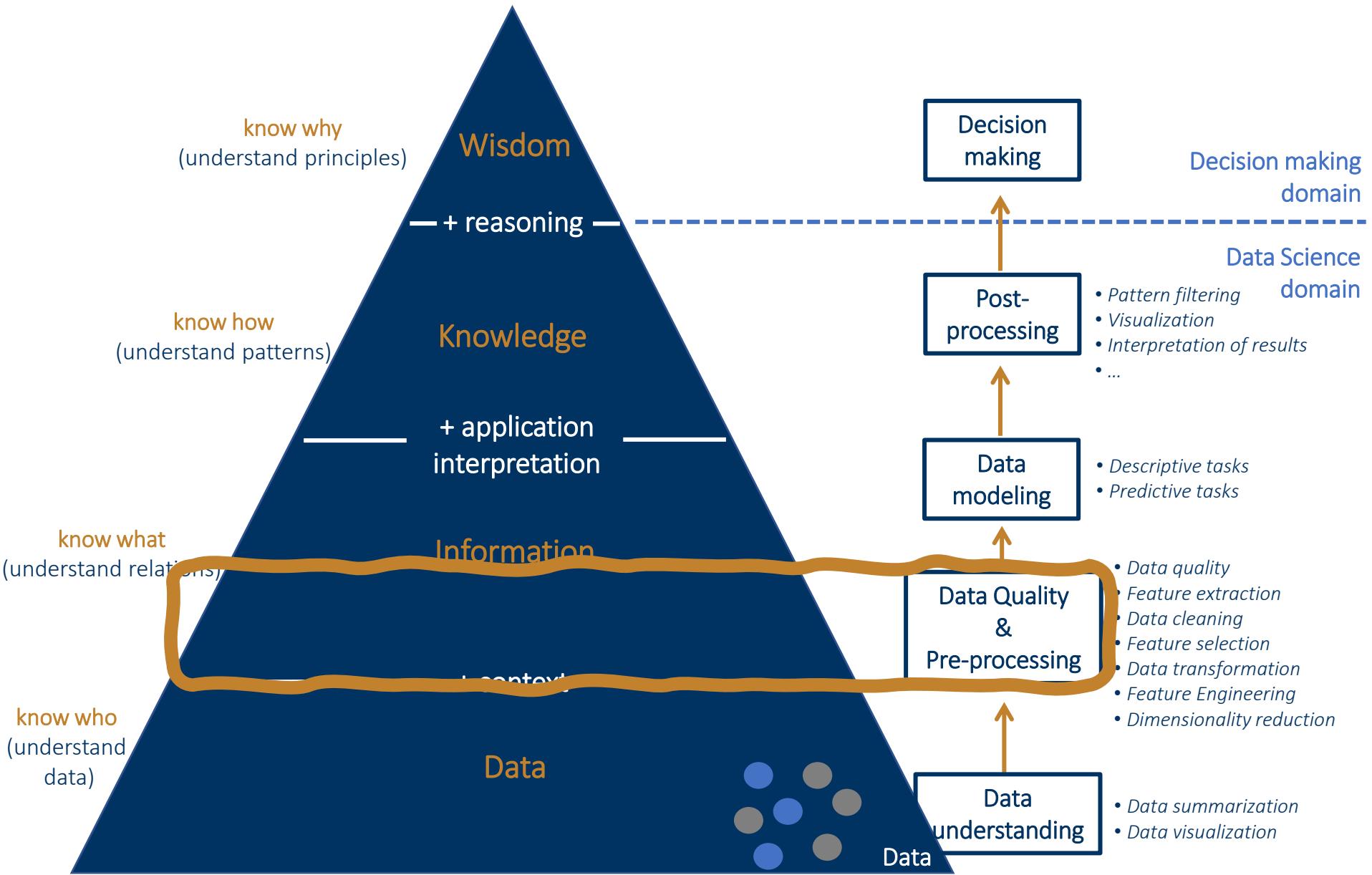
Raquel Sebastião

Departamento de Eletrónica, Telecomunicações e Informática

Universidade de Aveiro

raquel.sebastiao@ua.pt

2022/2023



Contents

- Data Quality
- Data Pre-processing
 - Feature extraction
 - Data integration
 - Data cleaning
 - Feature transformation
 - Feature Engineering
 - Data reduction
- Summary

Data quality

Poor data quality **negatively affects effective** data analysis

Example: a classification model for detecting client's loan risks is built using poor data

- Some credit-worthy candidates are denied loans
- More loans are given to individuals that default

Data quality

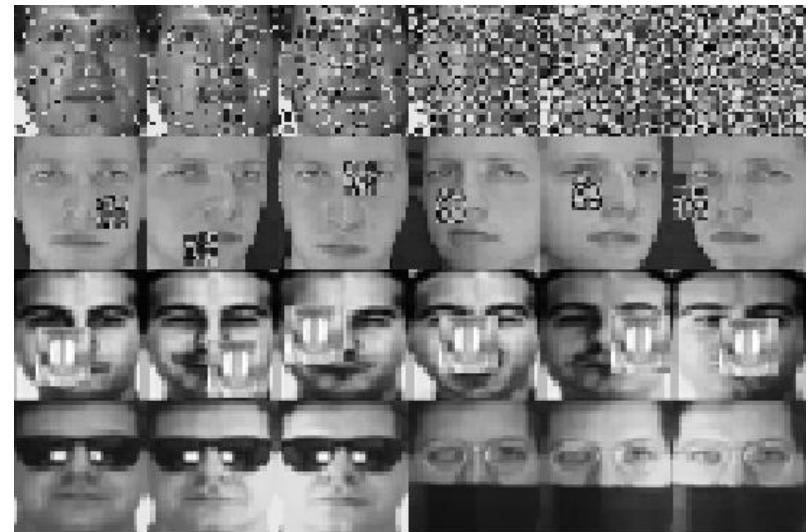
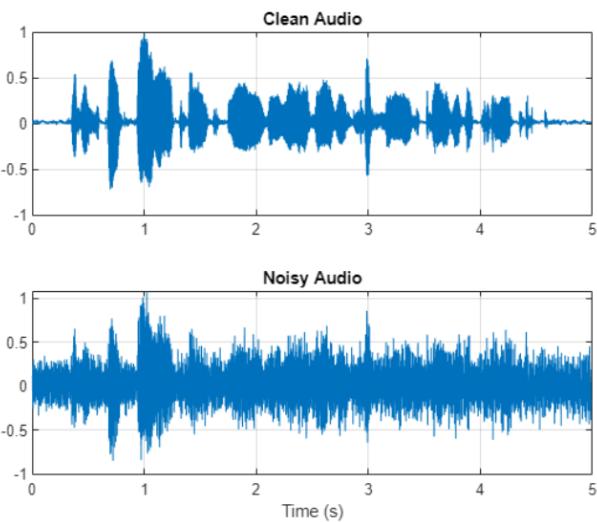
- What kinds of data quality problems?
- How can we detect problems with the data?
- What can we do about these problems?

Examples of data quality problems

- Missing values
- Duplicate data
- Noise and outliers
- Wrong data
- Fake data
- Inconsistent across different data sources

Data quality: noise

- For objects, noise is an extraneous object
- For attributes, noise refers to modification of original values
 - distortion of a person's voice when talking on a poor phone
 - "snow" on television screen

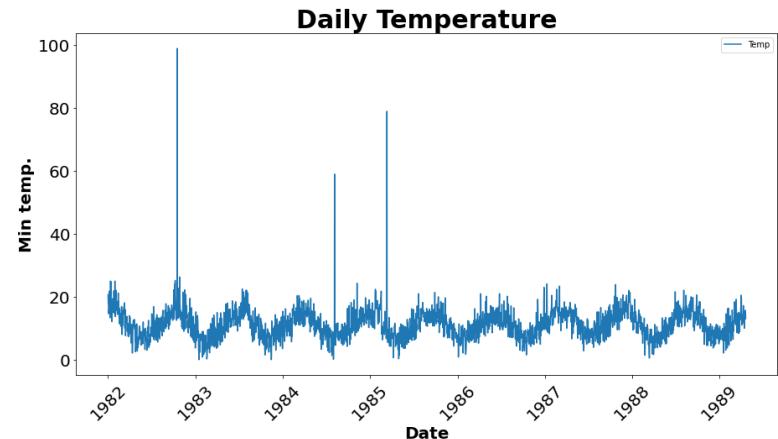


Some corrupted face images from the Yale dataset

Data quality: outliers

"An outlier is a point that deviates so much from the other data points as to arouse suspicions that it was generated by a different mechanism" (Hawkins, 1980)"
Hawkins, 1980

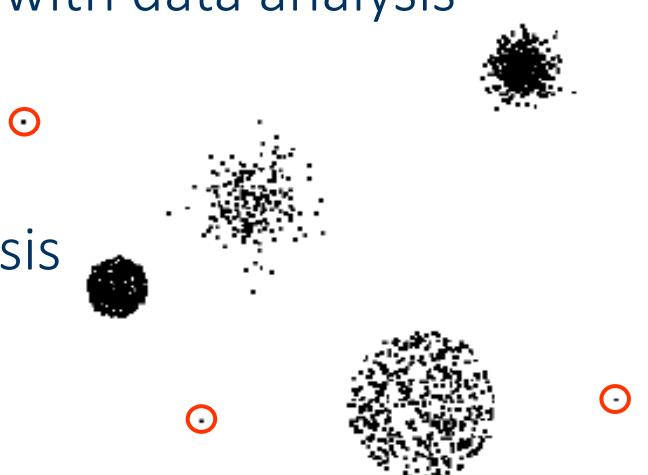
- **Outliers** are data objects with characteristics that are considerably different than most of the other data objects in the data set
- **Case 1:** Outliers are noise that interferes with data analysis
 - Min. temperature values above 50°C



Data quality: outliers

"An outlier is a point that deviates so much from the other data points as to arouse suspicions that it was generated by a different mechanism" (Hawkins, 1980)"
Hawkins, 1980

- **Outliers** are data objects with characteristics that are considerably different than most of the other data objects in the data set
- **Case 1:** Outliers are noise that interferes with data analysis
 - Min. temperature values above 50°C
- **Case 2:** Outliers are the goal of our analysis
 - Credit card fraud
 - Intrusion detection



Data quality: missing values

- Information was **not collected**
 - Missing value is related to unobserved data of the variable
 - people decline to give their age and weight
- Attributes may **not be applicable** to all cases
 - Missing value is related to observed data, not to unobserved data
 - annual income is not applicable to children

Data quality: missing values

Missing data may be due to

- Equipment malfunction
- Incongruent with other recorded data and thus deleted
- Data were not entered due to misunderstanding
- Certain data may not be considered important at the time of entry
- Did not register history or changes of the data

Data quality: duplicates

- Data set may include data objects that are duplicates, or almost duplicates of one another
 - Major issue when merging data from heterogeneous sources
- Examples:
 - Same person with multiple email addresses
- Necessary a process of dealing with duplicate data issues
 - When should duplicate data not be removed?

Data quality: inconsistent data

Typical when the data is available from different sources in different formats

- Examples
 - Person's name may be spelled out differently in different sources
 - John Smith, J. Smith, Smith J.
 - Person's height should not be negative
 - Same object: Attribute *country* = 'United States' & attribute *city* = 'Shanghai'

Contents

- Data Quality
- Data Pre-processing
 - Feature extraction
 - Data integration
 - Data cleaning
 - Feature transformation
 - Feature Engineering
 - Data reduction
- Summary

Data pre-processing: what and why?

- Extremely important
- Time-consuming

Steps carried out before any further analysis of the available data

- Data can come from several sources (in different formats)
- Data sets may have unknown attributes values
- Many data mining methods are sensitive to the scale and/or the type of attributes
- The need to create new attributes to achieve data analysis goals
- The need to select representative subsets of data, as the data set may be too large for some methods to be applicable

Data pre-processing: major tasks

- Feature extraction
- Data integration
- Data cleaning
- Feature transformation
 - Aggregation
 - Scaling
 - Discretization
 - Binarization
- Feature Engineering
- Data reduction
 - Numerosity reduction
 - Dimensionality reduction
 - Feature selection
 - Singular Value Decomposition (SVD)
 - Principal Component Analysis (PCA)
 - Linear Discriminant Functions

Data pre-processing: feature extraction

Extract features from raw data

... features need to be extracted for processing

- Text to categorical and numeric data
- Time Series to discrete sequence data
- Time Series to numeric data
- Discrete sequence to numeric data
- Spatial to numeric data
- Graphs to numeric data
 - Sensor data
 - Image data
 - Web logs
 - Network traffic
 - Document data

Data pre-processing: feature extraction

Extract features from raw data

... features need to be extracted for processing

- **sensor data**
 - large volume of low-level signals associated with date/time attributes
- **image data**
 - very high-dimensional data that can be represented by pixels, color histograms, etc.
- **web logs**
 - text in a prespecified format with both categorical and numerical attributes
- **network traffic**
 - network packets information
- **document data**
 - raw and unstructured data

Data pre-processing: data integration

Redundant and inconsistent data occur often when integration of multiple datasets

- The same attribute or object may have different names in different datasets
- An attribute may be a “derived” from another attribute or set of attributes in another table
 - *Age* = “42”, *Birthday* = “03/07/1980”

Careful integration of the data may help reduce/avoid redundancies and inconsistencies

Data pre-processing: data cleaning

Data in the Real World Is **Dirty**

- Lots of potentially incorrect data
 - e.g., instrument faulty, human or computer error, and transmission error

Poor data quality **negatively affects** data processing tasks and impacts performance of the models

Data pre-processing: data cleaning

Poor data quality negatively affects data processing tasks and impacts performance of the models

- Handle missing values
- Deal with duplicate data
- Smooth noisy data
 - In columns (e.g., features): due to sensing errors
 - In rows: extraneous object
- Identify or remove outliers
 - In columns (e.g., features): univariate statistics (can be noise)
 - In rows (might be the goal of analysis)
- Resolve inconsistencies
 - Some easy to detect. For instance: person's height should not be negative
 - Correction of inconsistencies requires redundant or additional information

Data pre-processing: data cleaning

Data in the Real World Is Dirty

Ultimate goal

- Make the data set **tidy**
 - each value belongs to an attribute and an object
 - each attribute contains all values of a certain property measured across all objects
 - each object contains all values of the attribute measured for the respective case
- These properties lead to **data tables**
 - each row represents an object
 - each column represents an attribute measured for each object

Data cleaning: handling missing values

- Information was **not** collected
- Features/attributes may **not** be applicable to all cases

Main Strategies to Handle missing values

- **Elimination**
 - Eliminating Rows (e.g., objects)
 - Eliminating Columns (e.g., features)
- **Imputation:** substituting missing by
 - mean, median (numerical feature)
 - mode (categorical feature)
 - linear interpolation of nearby values in time and/or space
- **Ignore** the missing value during analysis
 - methods inherently designed to work robustly with missing values

Data cleaning: handling missing values

Consider the following "data tables" with missing values (marked- ?)

A1	A2	A3	A4	A5
	?			
	?			
	?			
	?			

A1	A2	A3	A4	A5
?		?		
		?		?

A1	A2	A3	A4	A5
		?		
	?			
				?

- Select the best strategy to handle the missing data
- Advantages and Disadvantages

Data cleaning: handling incorrect values

- Inconsistent detection
 - Data integration techniques
- Domain knowledge
 - Data auditing: by analyzing data to identify features' ranges or discover constraints/rules that specify the relationships across different features
- Data-centric methods
 - Statistical-based methods to detect outliers

Data pre-processing

"At the end of the day, some machine learning projects succeed and some fail.

What makes the difference?

Easily the most **important** factor is the **features used**."

Pedro Domingos, in "A Few Useful Things to Know about Machine Learning".
DOI:10.1145/2347736.2347755

- Feature transformation
 - Aggregation
 - Scaling
 - Discretization
 - Binarization
- Feature Engineering
- Data reduction

Data pre-processing: feature transformation

Map the entire set of values of a given feature to a new set of replacement values such that each old value can be identified with one of the new values

Why it may be useful?

- two features (e.g., age, salary) with very different scales
- Any aggregation function (e.g., Euclidean distance) computed on the set of objects, will be dominated by the feature of larger magnitude

Some common strategies:

- Aggregation
- Scaling
- Discretization
- Binarization

Feature transformation: aggregation

Combining two or more **features** (or objects) into a single **features** (or object)

Goals

- More “stable” data - aggregated data tends to have less variability
- Data reduction - reduce the number of attributes (or objects)
 - Products aggregated into **categories** (grocery, electronics, clothing, toys, ...)
 - Days aggregated into **weeks**, **months**, or **years**



Feature transformation: scaling

What?

Techniques to adjust to differences among attributes in terms of frequency of occurrence, mean, variance, range

- Representing all numerical (integer or real) features in the same scale

Why?

Due to the sensitivity of aggregation functions (e.g., Euclidean distance) to the scale/magnitude of the input values

Feature scaling is important to:

- PCA, LDA, kNN, LR, SVM, Neural Networks, k-Means, Regression, ...

Feature scaling is not important (not distance-based)

- Naïve-Bayes and Decision trees

Feature transformation: scaling

Min-Max scaling (range-based scaling) *

$$\tilde{x}_i = \frac{x_i - \min}{\max - \min}$$

- \min and \max minimum and maximum, respectively, of feature \mathbf{x}_i
 - Scaled values lie int the range [0,1]
 - Outliers are not correctly handled
 - if an erroneous age value of 800 is registered instead of 80, most of the values will be in the range [0;0.1]
- * Just scales the data

Feature transformation: scaling

Standardization (z-score normalization)*

$$\tilde{x}_i = \frac{x_i - \mu_i}{\sigma_i}$$

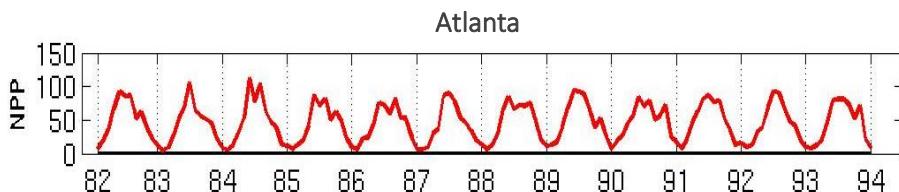
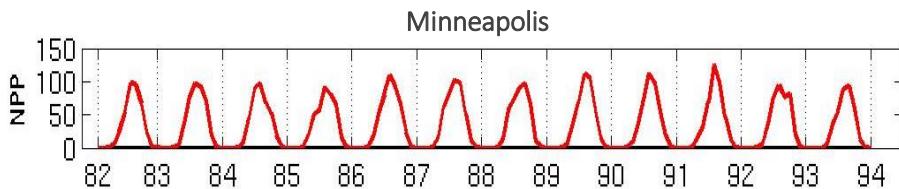
- μ_i and σ_i are the mean and standard deviation, respectively, of feature x_i
- values are scaled s.t. $\mu_i=0$ and $\sigma_i=1$
- Scaled values, typically, lie in the range [-3 , 3] under a normal distribution assumption

* More than scaling, it changes the distribution of the data

Feature transformation: scaling

Time-series

- Adjust differences in terms of mean, variance and range
- Take out unwanted, common signal, e.g., seasonality



Net Primary Production (NPP) is a measure of plant growth used by ecosystem scientists.

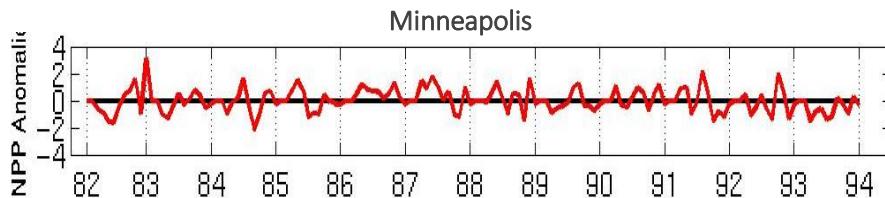
Correlations between time series

	Minneapolis	Atlanta	Sao Paolo
Minneapolis	1.0000	0.7591	-0.7581
Atlanta	0.7591	1.0000	-0.5739
Sao Paolo	-0.7581	-0.5739	1.0000

Feature transformation: scaling

Time-series

- Adjust differences in terms of mean, variance and range
- Take out unwanted, common signal, e.g., seasonality



Normalized using **monthly z-Score**

- Subtract off monthly mean and divide by monthly standard deviation



Correlations between time series



	Minneapolis	Atlanta	Sao Paolo
Minneapolis	1.0000	0.0492	0.0906
Atlanta	0.0492	1.0000	-0.0154
Sao Paolo	0.0906	-0.0154	1.0000

Feature transformation: discretization

Discretization: converting a continuous feature into an ordinal feature

- A potentially infinite number of values are mapped into a small number of categories

Unsupervised discretization: find breaks in the data values

- **Equal-width:** divides the range into equal-width intervals
 - it may be affected by the presence of outliers
 - Skew data is not correctly handled
- **Equal-frequency:** divides the range into intervals with the same number of values
 - it can generate ranges with very different amplitudes
 - Good data scaling
- **Clustering** approaches

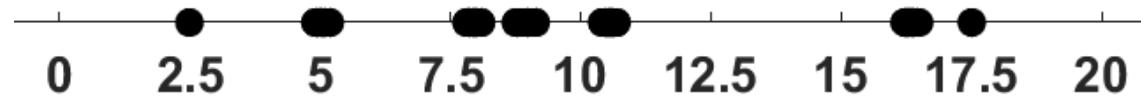
Supervised discretization: use class labels to divide data values

- Decision-tree analysis, target correlation analysis

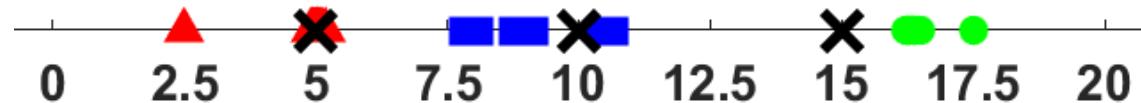
Feature transformation: discretization

Examples (unsupervised discretization)

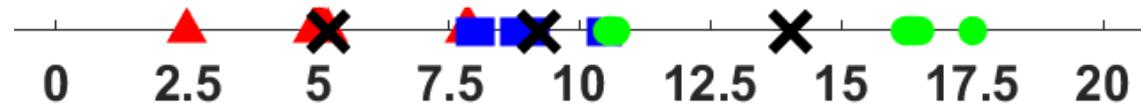
Data consists of four groups of points and two outliers



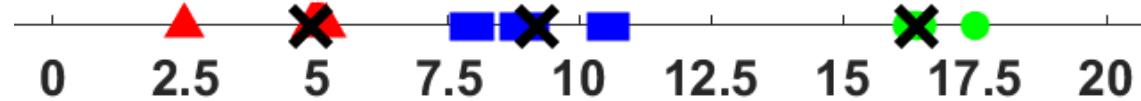
Equal interval width approach to obtain 3 values



Equal frequency approach to obtain 3 values



K-means approach to obtain 3 values



Feature transformation: binarization

Binarization maps a categorical nominal feature into one or more binary features (numeric)

- **Binarization:** if the feature has only 2 possible nominal values, it can be transformed into 1 binary feature
 - survived: yes/no -> survived: 1/0
- **One-Hot Encoding:** creates a binary variable for each category

Hair color	<i>Hair_brown</i>	<i>Hair_blonde</i>	<i>Hair_black</i>
Brown	1	0	0
Blond	0	1	0
Black	0	0	1

Disadvantages: the number of features and **sparsity** on data increase

Feature transformation: ordinal features

Transformation of ordinal features

- Map the features into an interval [0,1]
- Map the features into integers
- Examples:
 - $\{ \text{Good}, \text{Very Good} \text{ and } \text{Excellent} \} \rightarrow \{0, 1, 2\}$
 - Likert scale applied by social sciences

$\{\text{Extremely Unlikely (1)} \text{ to } \text{Extremely Likely (7)}\} \rightarrow \{1, 2 \dots, 7\}$

Categorical ordinal

Sizes	A
Very Small	1
Small	2
Medium	3
Large	4
Very Large	5

- **Ordinal Encoder:** Encode categorical features as an integer array, allows to specify the order of the categories

Data pre-processing

"At the end of the day, some machine learning projects succeed and some fail.

What makes the difference?

Easily the most **important** factor is the **features used**."

Pedro Domingos, in "A Few Useful Things to Know about Machine Learning".
DOI:10.1145/2347736.2347755

- Feature transformation
- **Feature Engineering**
 - Creation of features
- Data reduction

Data pre-processing: feature engineering

Creation of features maps the entire set of values of given features to a new set of replacement values such that each old value can be identified with one of the new values

- The process of using **domain knowledge** of the data to create features that might help when solving the problem.
- New features that can capture **the important information** in a data set much more efficiently than the original features.

Feature engineering: creation of features

Express known relationships between existing variables

- create ratios and proportions like credit card sales per person
- the average web session duration per user, frequency of access, ...

Example: features X_1 : distance and X_2 : duration

$$\text{create speed } X_3 = X_1 / X_2$$

Express known case dependencies

- Create features using the information about case dependencies relationships (time, space, space-time)

Feature engineering: creation of features

Express known case dependencies

Time series

- create feature that represent **relative values** instead of absolute values, so to avoid trend effects.

$$y_t = \frac{x_t - x_{t-1}}{x_{t-1}}$$

- **Time Delay Embedding** - create features whose values are the value of the same variable in previous time steps

- Standard tools will be able to model the time relationships

X_{t-3}	X_{t-2}	X_{t-1}	X_t
X_{t_1}	X_{t_2}	X_{t_3}	X_{t_4}
X_{t_2}	X_{t_3}	X_{t_4}	X_{t_5}
...			
$X_{t_{n-3}}$	$X_{t_{n-2}}$	$X_{t_{n-1}}$	X_{t_n}

- Similar “tricks” can be done with space and space-time dependencies

Data pre-processing

"At the end of the day, some machine learning projects succeed and some fail.

What makes the difference?

Easily the most **important** factor is the **features used**."

Pedro Domingos, in "A Few Useful Things to Know about Machine Learning".
DOI:10.1145/2347736.2347755

- Feature transformation
- Feature Engineering
- **Data reduction**
 - Numerosity reduction
 - Dimensionality reduction
 - Feature selection
 - Singular Value Decomposition (SVD)
 - Principal Component Analysis (PCA), Kernel PCA
 - Linear Discriminant Functions

Data pre-processing: data reduction

- Numerosity reduction
- Dimensionality reduction
 - Feature selection
 - Singular Value Decomposition (SVD)
 - Principal Component Analysis (PCA)
 - Kernel PCA
 - Linear Discriminant Functions

Data reduction: numerosity reduction

What?

- Obtain a reduced representation of the data set
 - much smaller in volume but yet produces *almost* the same analytical results

Why?

- A data set may store terabytes of data
 - Complex analysis may take a very long time to run on the complete data set

Methods

- Regression and Log-Linear Models
- Histograms, clustering, sampling
- Data cube aggregation
- Data compression

Numerosity reduction: sampling

What?

Obtaining a **smaller data set** to represent the whole **data set of size N**

Why?

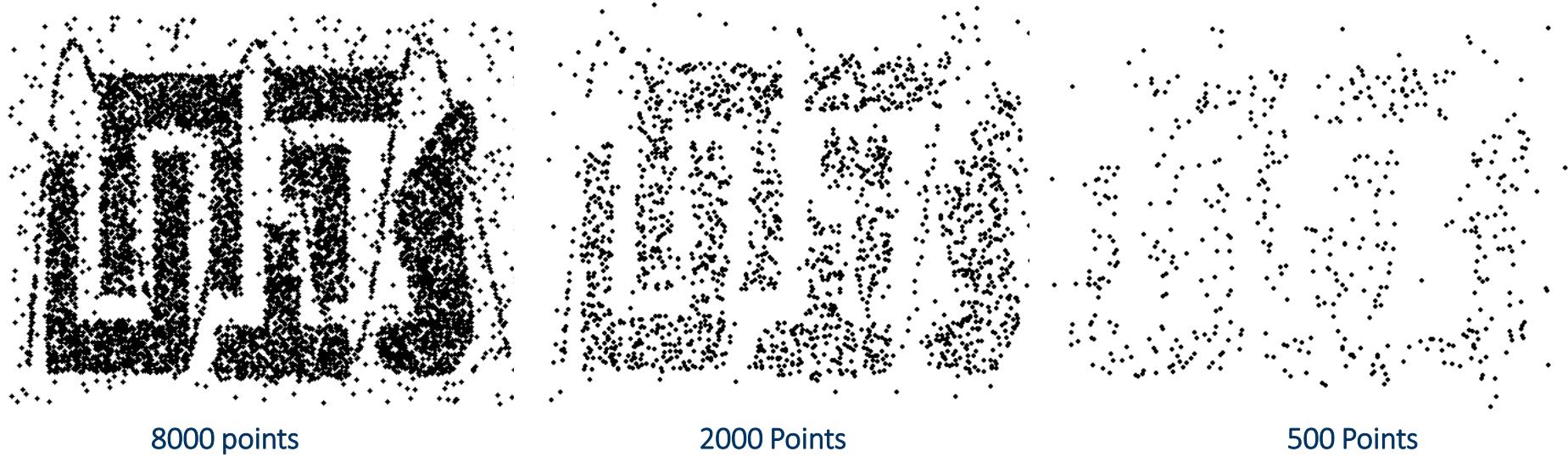
- Processing the entire set of data of interest is too expensive or time consuming
- To allow a mining algorithm to run in complexity that is potentially sub-linear to the size of the data

It is often used for both the preliminary investigation of the data and the final data analysis

Numerosity reduction: sampling

Key principle: Choose a **representative** subset of the data

- using a sample will work almost as well as using the entire data set, if the sample is representative
- a sample is representative if it has approximately the same properties (of interest) as the original set of data



Numerosity reduction: types of sampling

Simple random sampling

- Equal probability of selecting any particular object
- **Sampling without replacement**
 - Once an object is selected, it is removed from the population
- **Sampling with replacement**
 - A selected object is not removed from the population
 - The same object can be picked up more than once

Stratified sampling

- Split the data into several partitions
- Draw random samples from each partition
(proportionally, i.e., approximately the same percentage of the data)

Incremental sampling

Data reduction: dimensionality reduction

Key questions

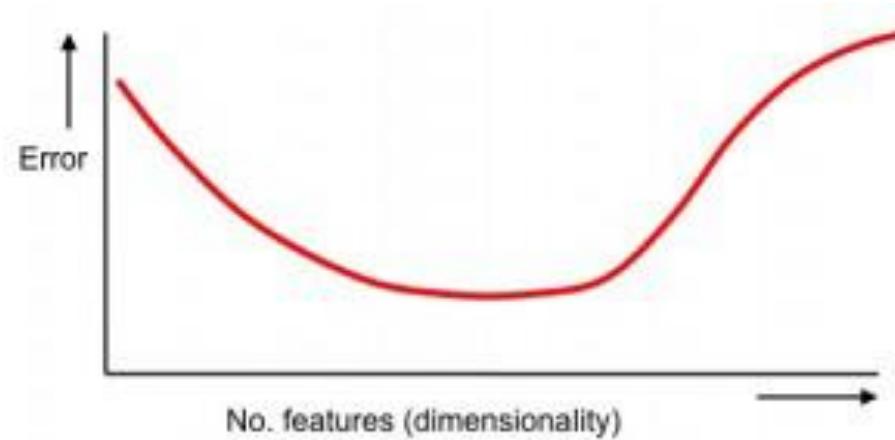
- How many features are required?
- Is there a point where we have too many features?
- How do we know beforehand which features will work best?
- What happens when there is feature redundancy/correlation?

Dimensionality reduction: The curse of dimensionality

- When dimensionality of feature space increases, the number of possible combinations of feature values increases exponentially
- The data becomes increasingly sparse in the space that it occupies
- We may assume that the more details (features) of the object we collect, the better description of the situation we have at hand
- Counter-intuitively, it is not valid

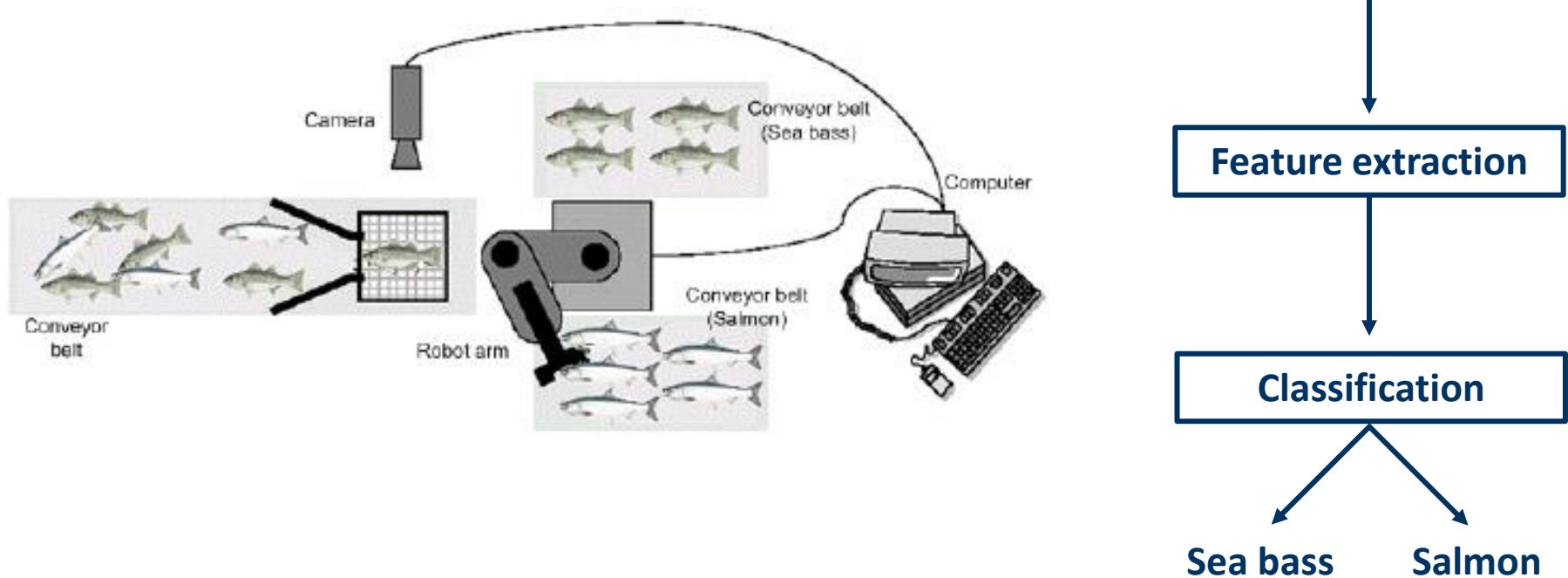
Dimensionality reduction: The curse of dimensionality

- There is a certain point after which adding new details becomes useless, and moreover, they may work against your model.
- In very high dimensional data many data mining algorithms do not work effectively.



- For example, distance between points, which is critical to some algorithms, becomes less meaningful.

Dimensionality reduction: Toy example sorting fish



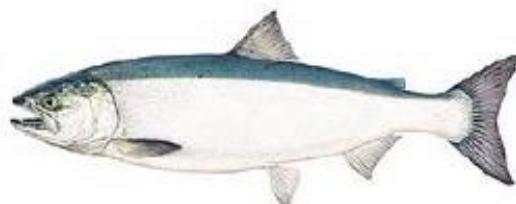
GOAL: A decision is made by processing the image of a single fish taken by the camera

Dimensionality reduction: Toy example sorting fish

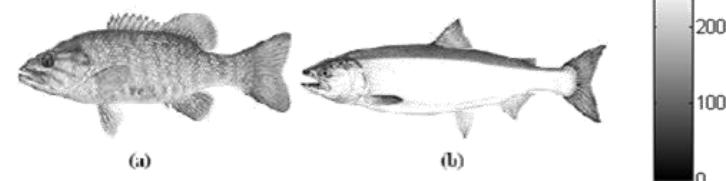
one feature



(a)



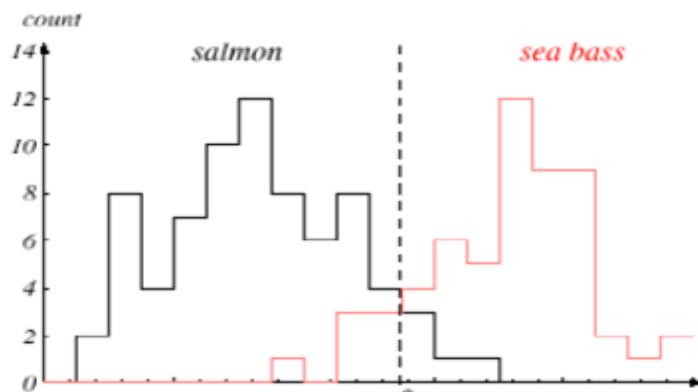
(b)



Intensity of the two fish images are in different ranges:

salmon is typically darker

Histogram of intensity values of a set of fishes

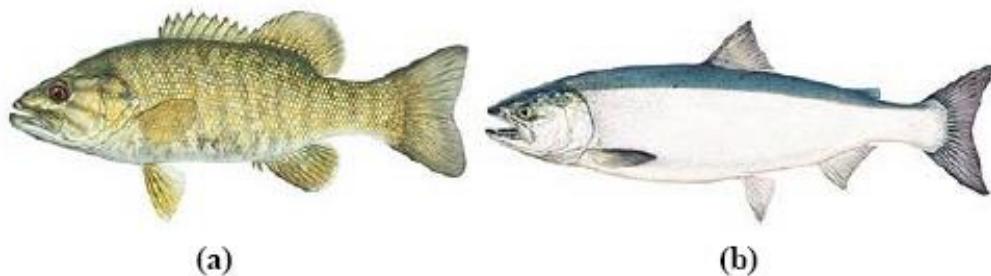


With histogram:

- GOAL of learning: estimate a threshold (model)
- Future decisions are made based on the learned threshold value

Dimensionality reduction: Toy example sorting fish

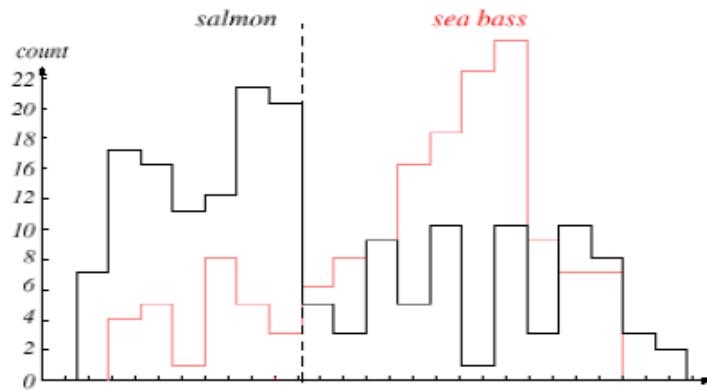
two features



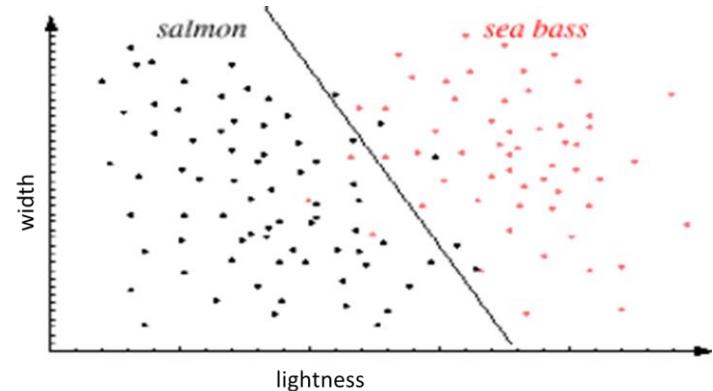
Length of the two fish images is different:

sea bass is typically wider

Histogram of the length of a set of fishes



Scatter plot:
values of the two features of the data set



In the 2D feature space: the two categories occupy distinct regions

With scatter plot:

- GOAL of Learning: find out the separation surface
- Future decisions are made based on the learned surface

Dimensionality reduction: Toy example sorting fish

The two features obviously separate the classes much better than one alone. This suggests adding a third feature. And a fourth feature. And so on.

Key questions

- How many features are required?
- Is there a point where we have too many features?
- How do we know beforehand which features will work best?
- What happens when there is feature redundancy/correlation?

Dimensionality reduction

Purpose:

- Avoid **curse of dimensionality**
 - Help **eliminate irrelevant** features or reduce noise
 - **Reduce** amount of time and **memory** required by data mining algorithms
 - Allow data to be more **easily visualized**
 - May increase **interpretability** (e.g., avoiding huge DT)
-
- **How**
 - Feature selection
 - Singular Value Decomposition (SVD)
 - Principal Components Analysis (PCA), Kernel PCA
 - Linear Discriminant Analysis (LDA)

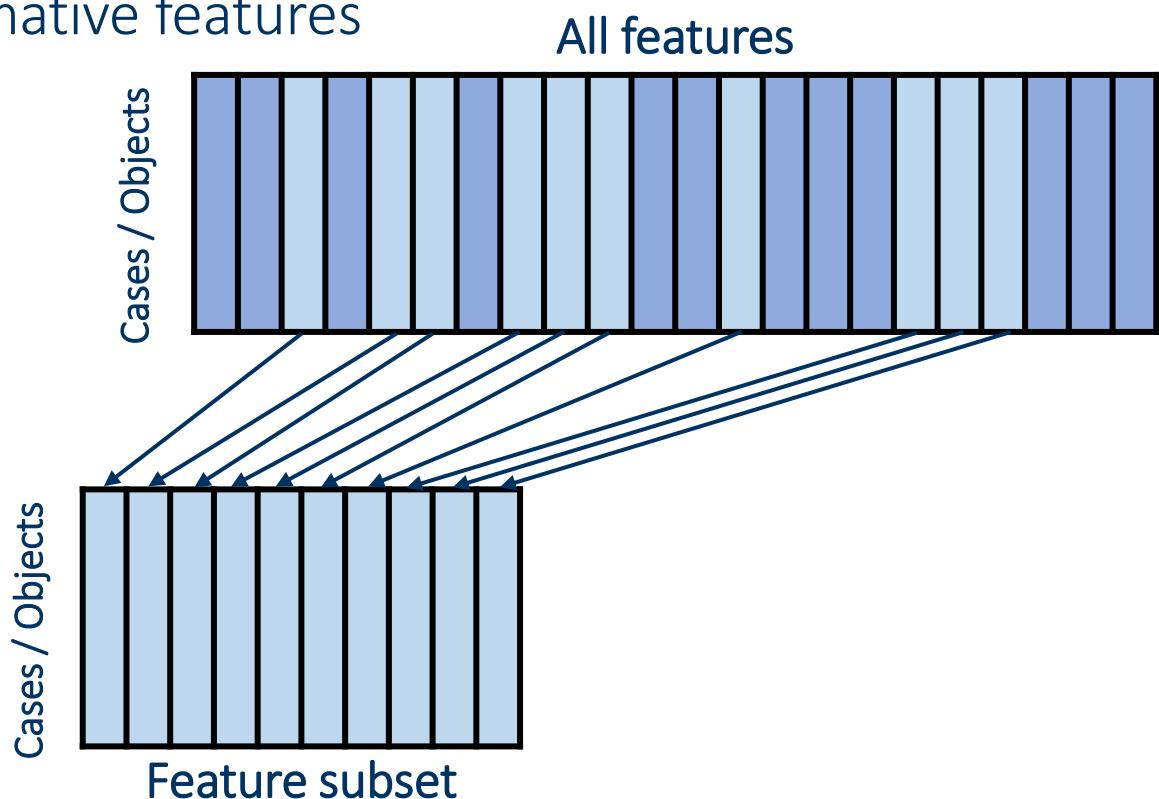
Dimensionality reduction: feature selection

- Discard **Redundant** Features
 - Duplicate much or all the information in one or more other attributes
 - Example: purchase price of a product and the amount of sales tax paid
- Discard **Irrelevant** features
 - Do not contain useful information for the data mining task at hand
 - Example: students' ID is often irrelevant to the task of predicting grades

Dimensionality reduction: feature selection

WHY?

- to achieve dimension reduction
- to construct more accurate classification models
- to find the more informative features



Dimensionality reduction: feature selection

Feature selection methods

Unsupervised

Filters

Drop incomplete features

Drop features with near-zero variance

Drop feature based on pairwise-correlation

Supervised

Filters

Feature score

Wrappers

Forward selection

Backward selection

Recursive selection

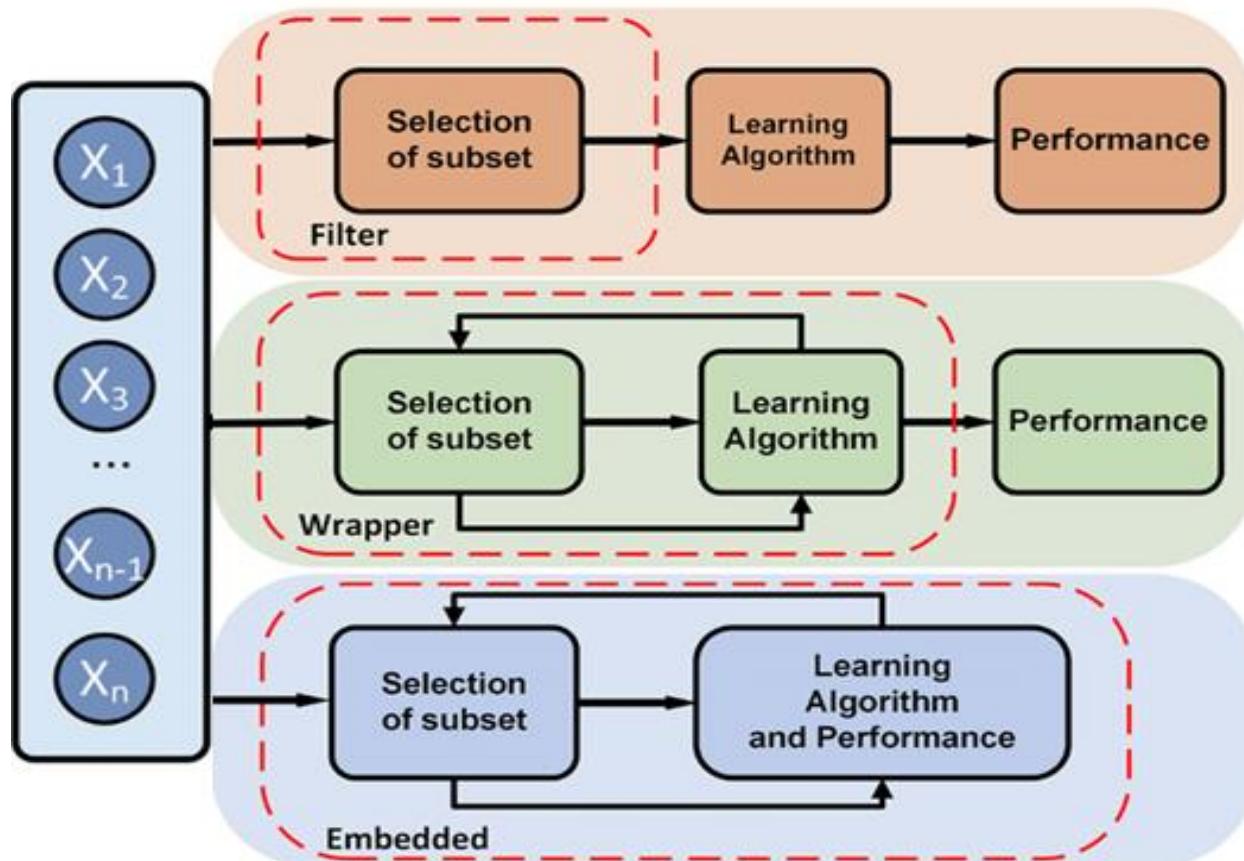
Embedded

Decision tree

Lasso

Dimensionality reduction: feature selection

Supervised feature selection methods



Computational Diagnostic Techniques for Electrocardiogram Signal Analysis. Liping Xie Zilong, Li Yihan Zhou, Jiaxin Zhu. *Sensors*. 2020 (adapted)

Dimensionality reduction: feature selection

Filter methods

- Selects a subset of variables independently of the classification model
 - Removing features with low variance (rank by cut-off)
 - Pairwise-correlation-based (rank by cut-off)
 - Ranking features by relevancy measure, depending on relationship with the target

Wrapper methods

- Selects a subset of variables taking into account the classification
 - Search for optimal subset of features

Embedded methods

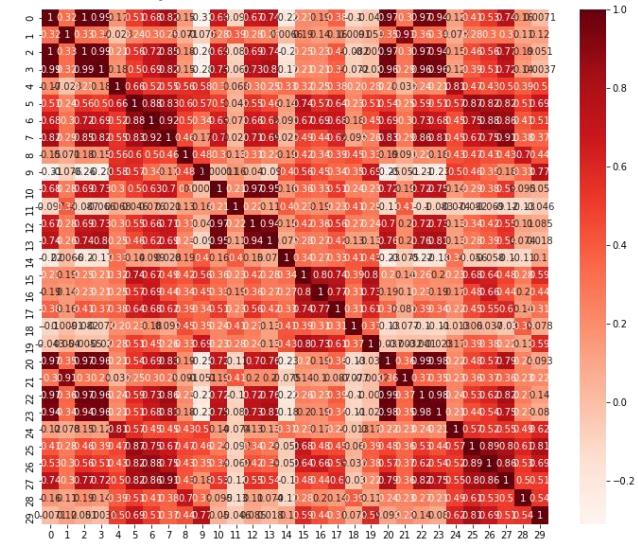
- The feature selection method is built in the classification model (or rather its training algorithm) itself (e.g. decision trees)

Feature selection: filter methods

Unsupervised feature selection methods

Filter methods: Non-iterative process

- Selects a subset of variables independently of the classification model
 - Drop **incomplete** features (e.g., with great number of missing values)
 - Drop features with **near-zero variance** (rank by cut-off)
 - Drop feature based on **pairwise-correlation** (rank by cut-off)
 - Eliminate one feature of a pair if correlation coefficient is larger than a threshold
 - The user chooses threshold (usually larger than 0.8)



Feature selection: filter methods

Supervised feature selection methods

Filter methods: Non-iterative process

- Selects a subset of features independently of the ML algorithm
 - Ranking features by **relevancy** measure, **depending on relationship** with the target
 - Select the features which are **highly dependent on the target**
 - Applied in parallel to all features providing scores

Select **k best features** based on a **relevance** measure to rank the features

- F-test, chi-square, mutual-information



Feature selection: filter methods

Ranking features by **relevancy** measure, depending on relationship with the target

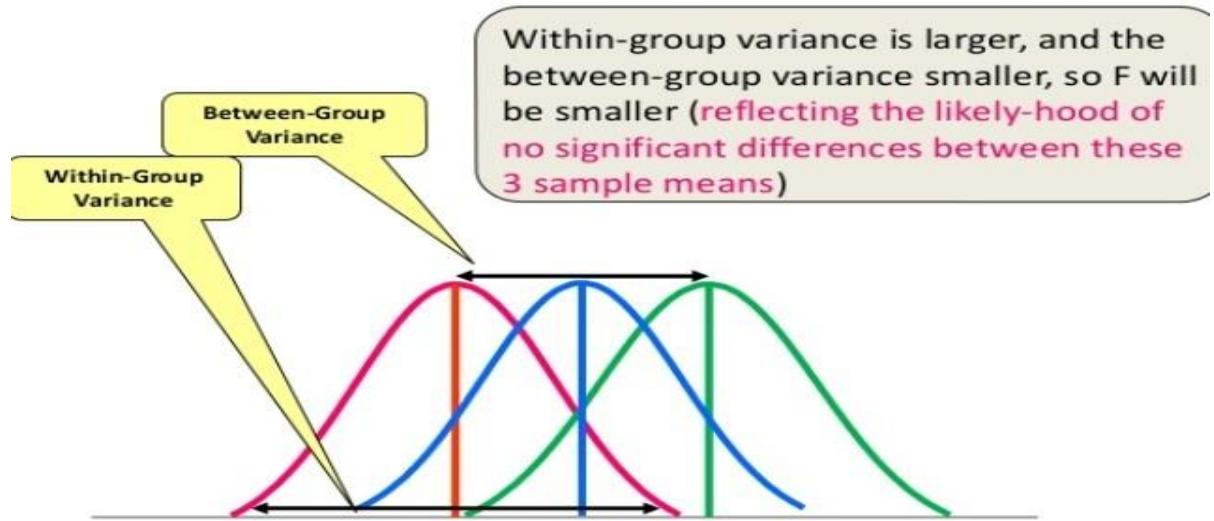
- Select the features which are **highly dependent on the target**
- Applied in parallel to all features providing scores

Numeric feature: Hypothesis test (measures if the mean of one feature is equal for all groups/classes of the target)

- F – value or t – value are used to rank the features
 - select the K features corresponding to K largest F-value

Feature selection: filter methods

Example: F-value qualitative interpretation



Selection Procedure:

- The groups: the categorical variable to be predicted on a classification task
- Calculate F value for all features
- Choose the K features corresponding to K largest F-value

Feature selection: filter methods

Ranking features by **relevancy** measure, depending on relationship with the target

- Select the features which are **highly dependent on the target**
- Applied in parallel to all features providing scores

Categorical feature: Hypothesis test (measures if target and feature are independent)

- χ^2 – value is used to rank the features
 - select the K features corresponding to K largest chi-square values

Feature selection: filter methods

Ranking features by **relevancy** measure, depending on relationship with the target

- Select the features which are **highly dependent on the target**
- Applied in parallel to all features providing scores

Disadvantages

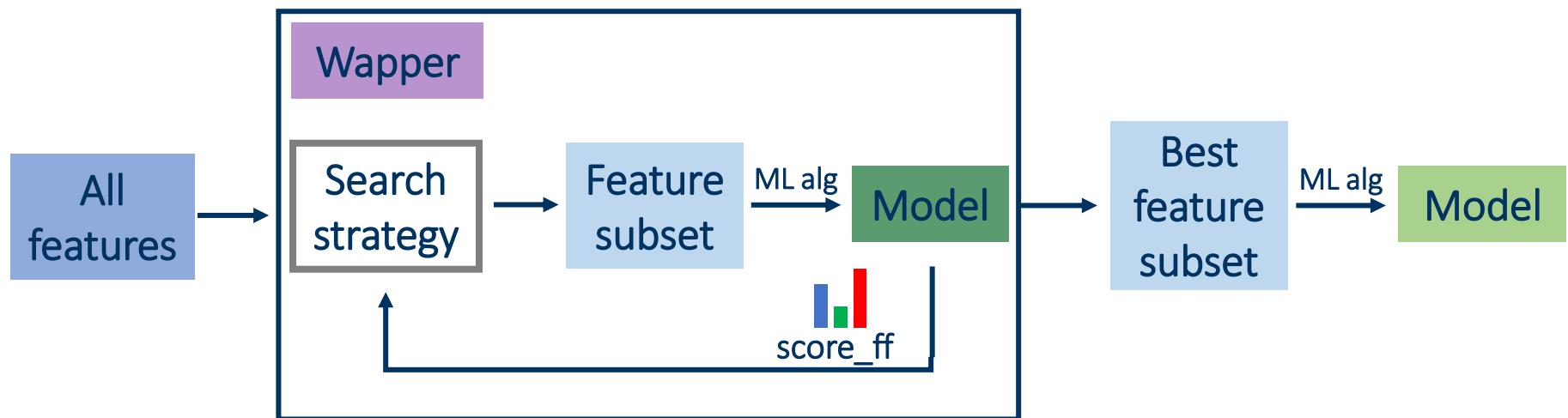
- Fail to recognize that a feature is important in combination with another variable
- Select a group of variables that are dependent and carry similar (or the same) information about the class label
- Often causes overfitting

Feature selection: wrapper methods

Supervised feature selection methods

Wrapper methods: Wrapper around learner

- Select features
- Evaluate learner (e.g., cross-validation)



Feature selection: wrapper methods

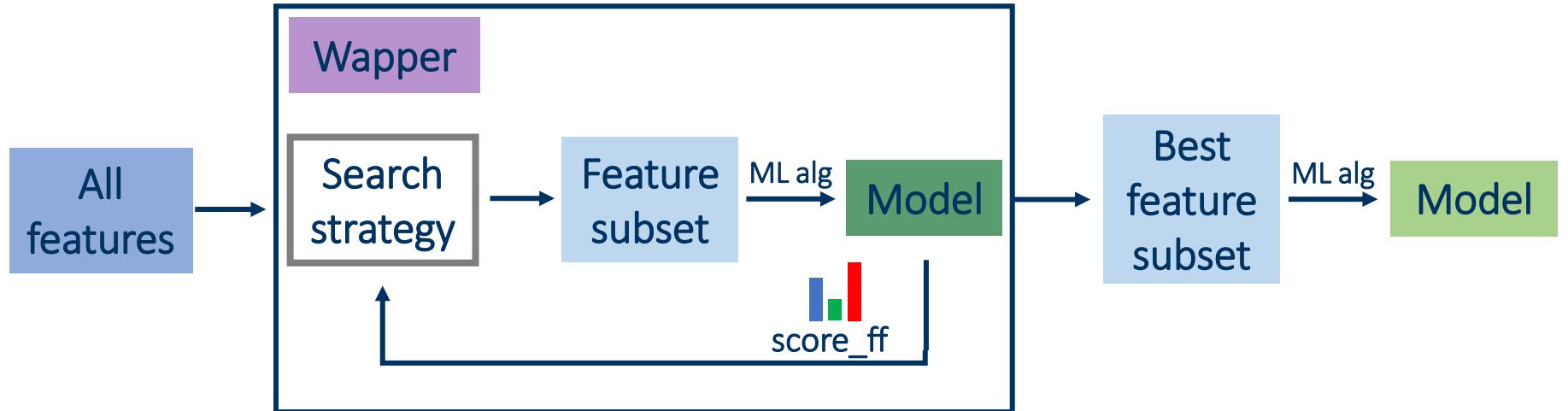
Supervised feature selection methods

Wrapper methods: Wrapper around learner

- Select features
 - Evaluate learner (e.g., cross-validation)
 - Iterative procedure: select 1 attribute, remove, repeat / select 1, add, repeat
- or
- Recursive procedure: attributes are recursively removed from current set
 - Several subsets of features are generated and tested on the particular model

Linear SVM and tree-based learners are often used

Feature selection: wrapper methods



Disadvantages

- Learner-dependent (selection for specific learner)
- Expensive
 - Greedy search: $O(k^2)$ for k attributes
 - When using a prior ranking (only find cut-off): $O(k)$

Feature selection: wrapper methods

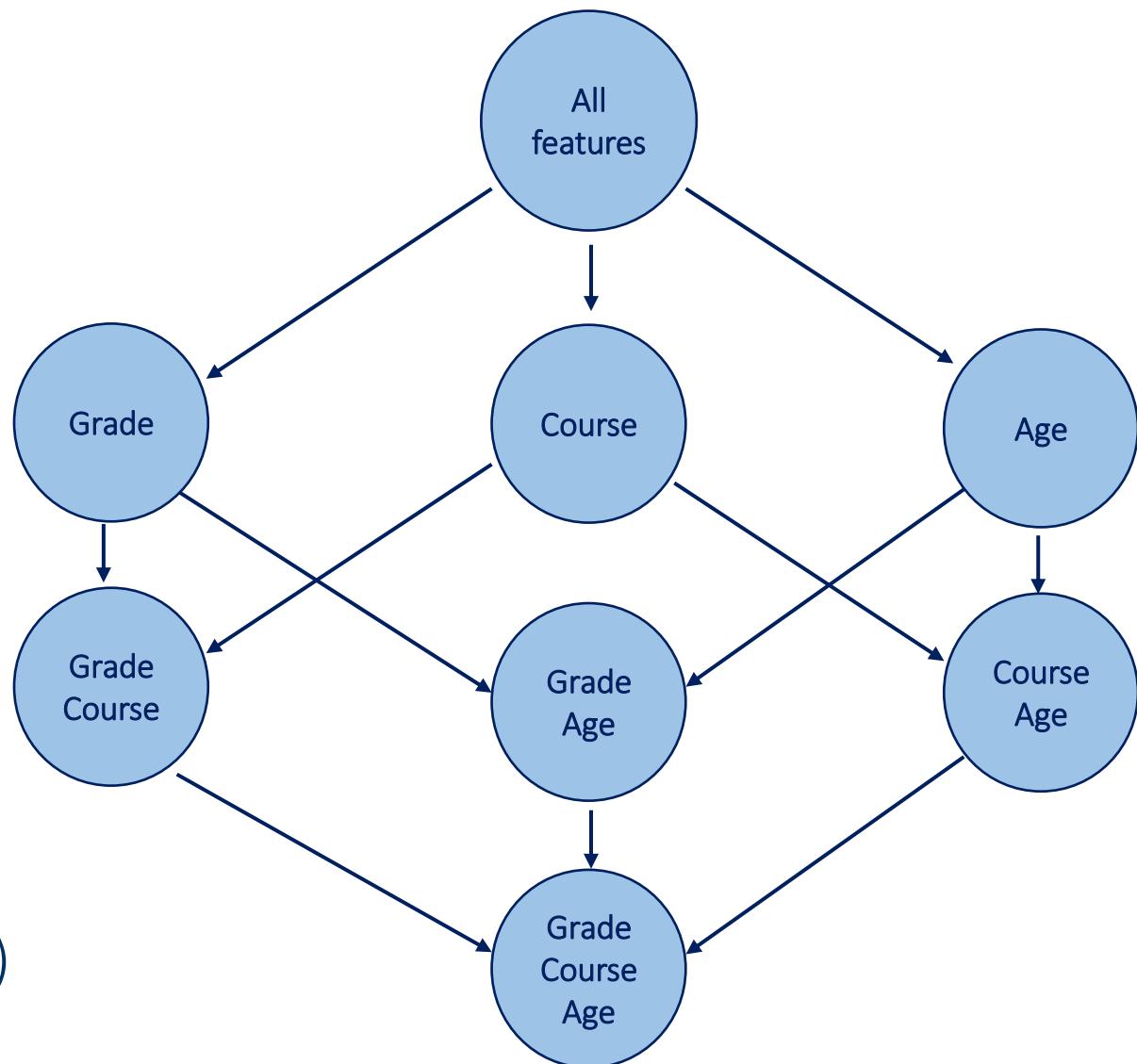
Example

Greedy search

Forward selection
(add one, select best)



Backward elimination
(remove one, select best)



Feature selection: embedded methods

Supervised feature selection methods

Embedded methods

- The classification method includes feature selection

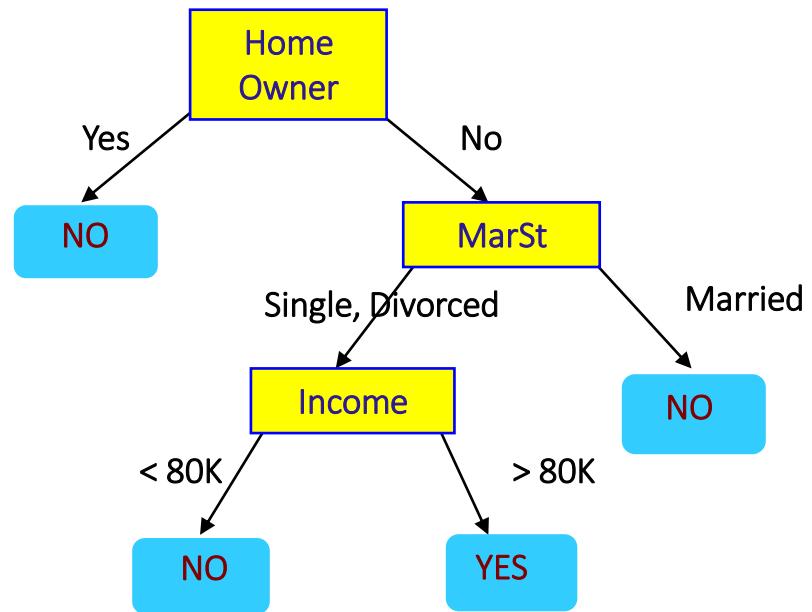
- models that show importance of features

- Tree-based learners

Training Data

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Model: Decision Tree



(Introduction to Data Mining, Tan et al.)

Dimensionality reduction: feature selection

Filter vs Wrapper Methods

- **Filter Methods**

- faster, as they do not involve training the models
- use statistical methods for evaluation of a subset of features
- fail to recognize importance of combined features
- select features that carry similar information about the target

- **Wrapper Methods**

- computationally more expensive
- use model performance estimation strategies
- provide the best subset of features
- ML algorithm dependent

Dimensionality reduction

Instead of selecting features, we can replace by “new” features

- A new (smaller) set of features where most of the "information" on the problem is still expressed.
- Sometimes, the correlation among the features is not perfect (redundant) but there may exist significant dependencies.



Dimensionality reduction

Main methods:

- Singular Value Decomposition (SVD)
- Principal Component Analysis (PCA)
- Kernel PCA
- Linear Discriminant Functions
- Others: supervised and non-linear techniques



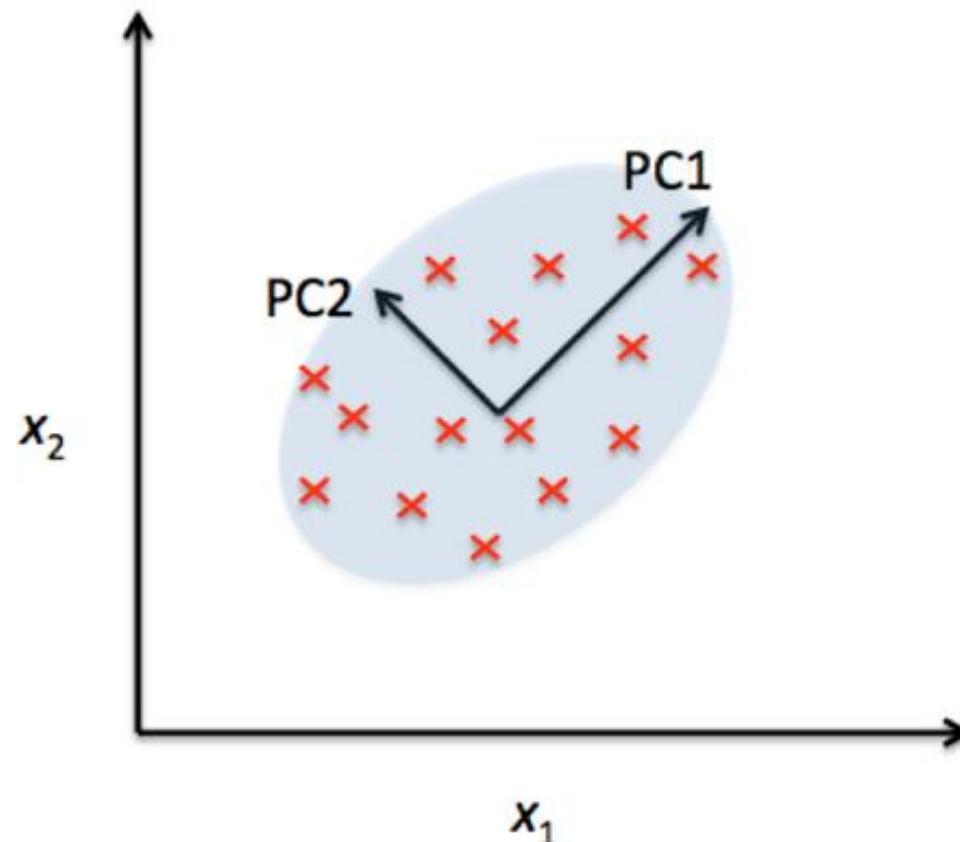
Dimensionality reduction: PCA

Principal Component Analysis (PCA)

- **Unsupervised** data reduction technique
- Reduces high dimensions into low dimension subspace
- Projects the data points onto new axes such that these new components **carry most of the essential information** of all the features
- These new components are a **linear combination** of all the features and the components thus formed are nothing but the **eigenvectors** which are now called the **Principal components**
- The eigenvalue corresponding to each of these eigenvectors will tell us about **how much variation** in the data has been captured by that **particular Principal Component**
- Principal components are **orthogonal to each other** and are **uncorrelated** **that increases maximum variance**. As they are uncorrelated it solves the problem of multicollinearity

Dimensionality reduction: PCA

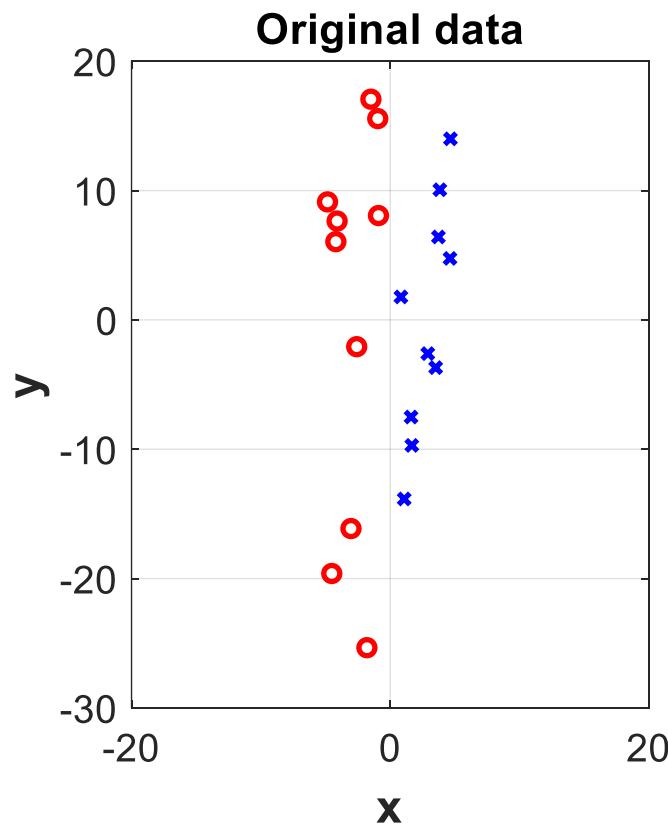
Principal Component Analysis (PCA)



Dimensionality reduction: PCA

Principal Component Analysis (PCA)

Target categories/classes are not taken into consideration



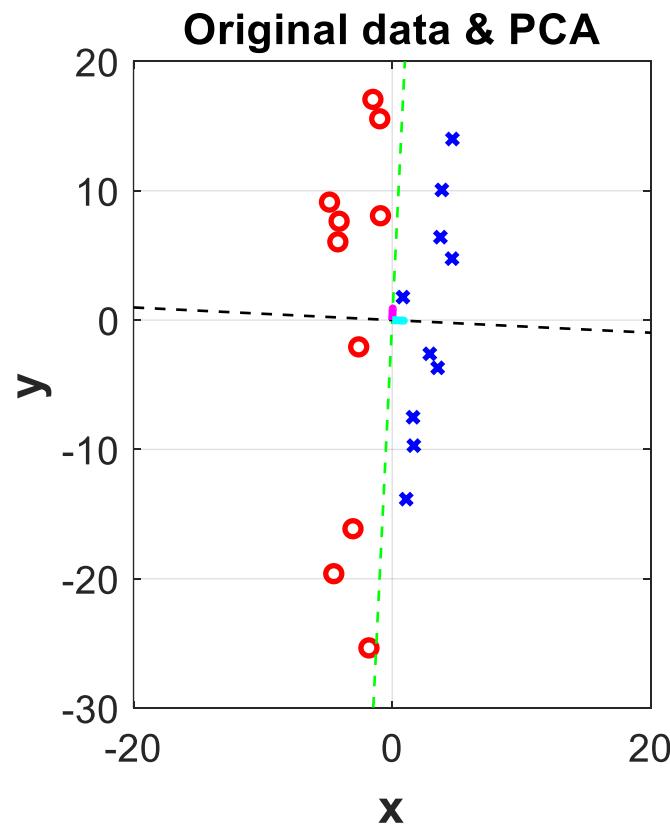
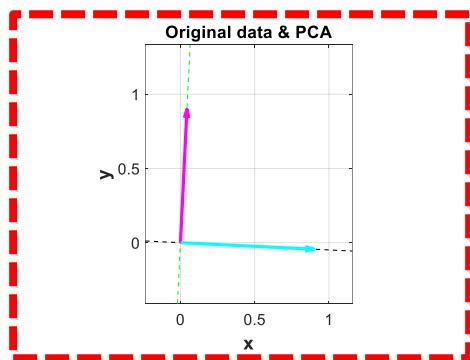
Dimensionality reduction: PCA

Principal Component Analysis (PCA)

Eigenvectors

$$\text{ev1} = [0.0488 \quad 0.9988]$$

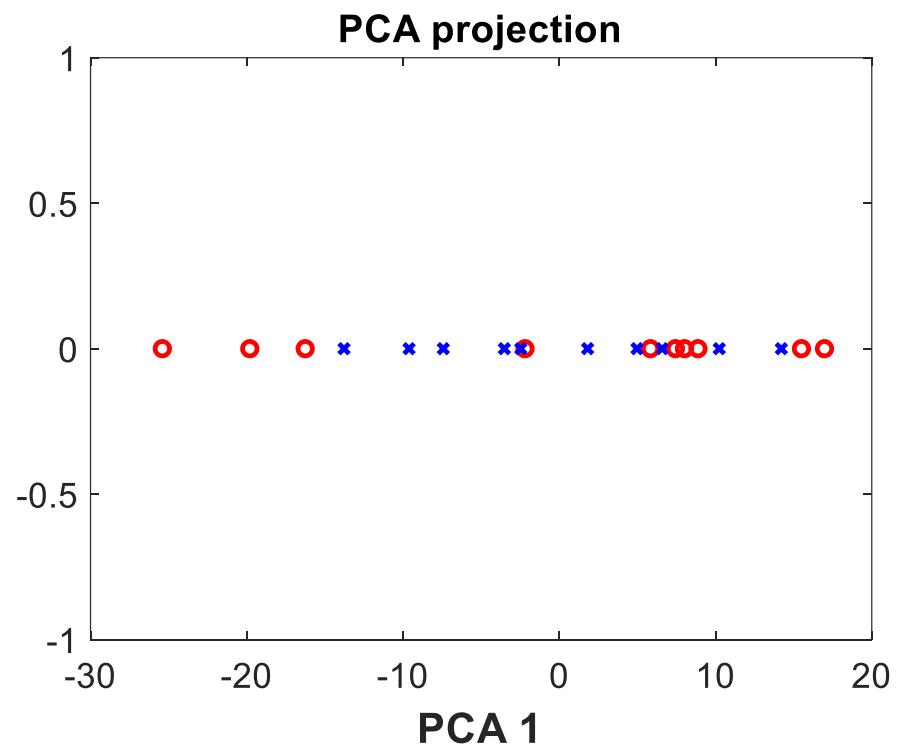
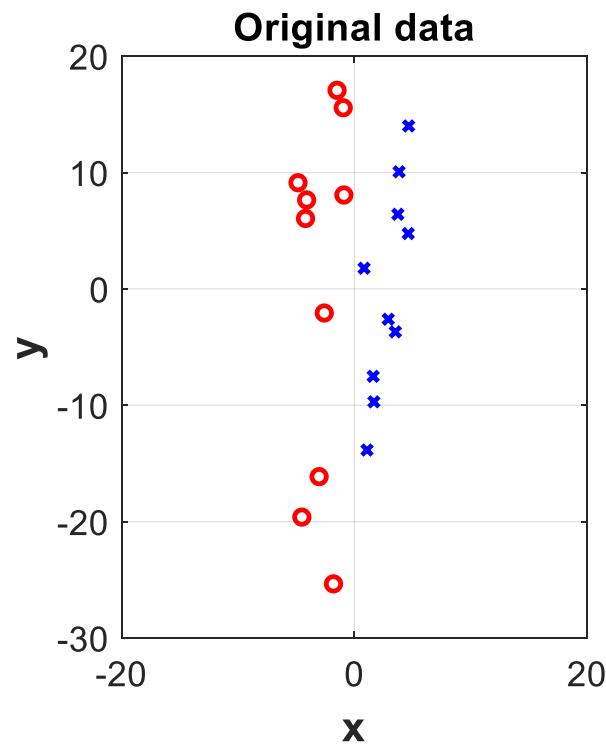
$$\text{ev2} = [0.9988 \quad -0.0488]$$



Dimensionality reduction: PCA

Principal Component Analysis (PCA)

$$\text{PC1} = [0.0488 \quad 0.9988]$$

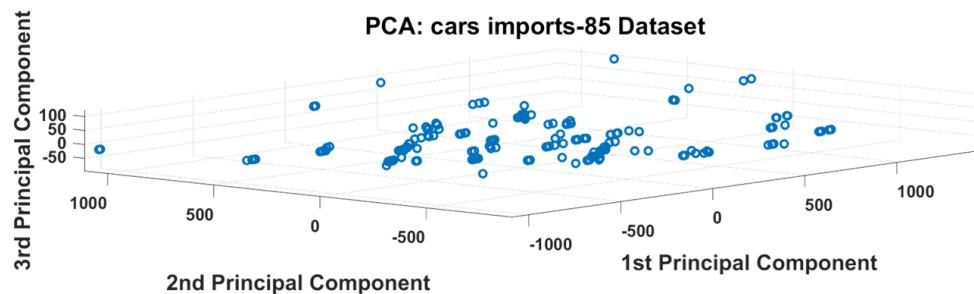


Dimensionality reduction: PCA example

Principal Component Analysis (PCA)

Data matrix

- 13 features
- 205 objects



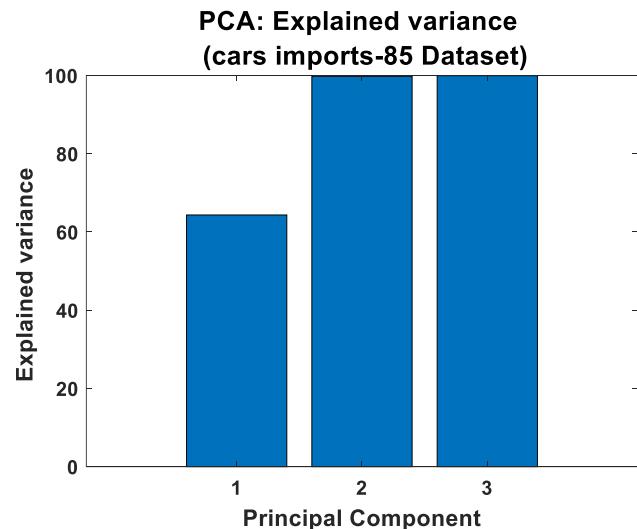
Explained variance

PC1: ~64.3%

PC2: ~35.4%

PC3: ~0.15%

remain: (...)



The first three components explain 99.95% of all variability

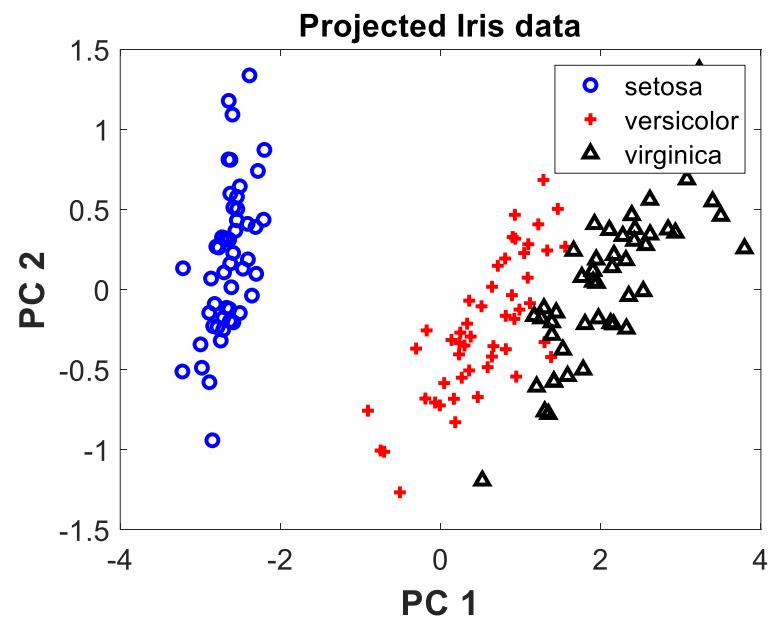
Dimensionality reduction: PCA

Principal Component Analysis (PCA)

- Find a **first linear combination** that **better** captures the **variability** in the data
- Move to the **second linear combination** to try to capture the variability not explained by the first one
- Continue until the set of new variables explains **most of the variability** (common values are 80% to 95%)

	PC 1	PC 2	PC 3	PC 4
Sepal_Length	0.361	0.657	-0.582	0.315
Sepal_Width		0.730	0.598	-0.320
Petal_Length	0.857	-0.173		-0.480
Petal_Width	0.358		0.546	0.754

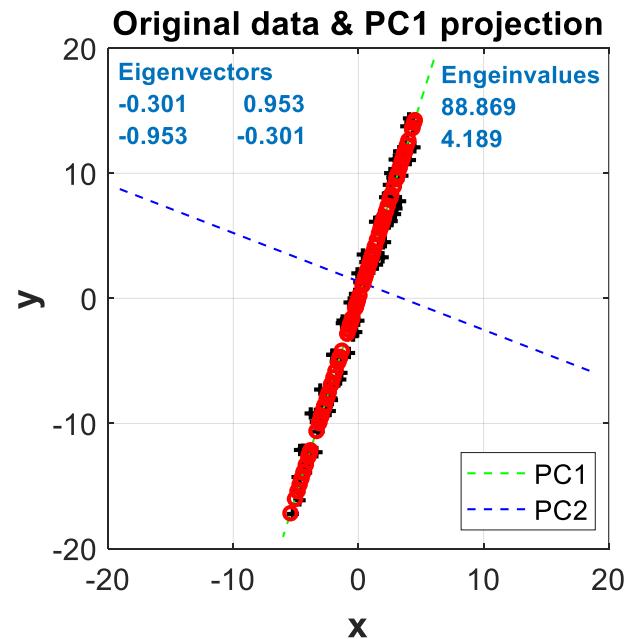
$$\begin{aligned} \text{PC1} = & 0.361 \times \text{Sepal_Length} \\ & + 0.857 \times \text{Petal_Length} \\ & + 0.358 \times \text{Petal_Width} \end{aligned}$$



Dimensionality reduction: PCA model

The **eigenvectors** form a new basis

- Basis vector model for the data: the **eigenvectors** $\mathbf{U} = [\mathbf{u}_1, \dots, \mathbf{u}_D]$
- \mathbf{u}_i is related to the entry (i, i) of the diagonal matrix Λ with **eigenvalues**: λ_i



Basis vector model \mathbf{u}_1 adapted to the spread of the data

Dimensionality reduction: PCA model

The **PCA model** is the $D \times D$ eigenvector matrix **U**

- SVD decomposition of centered data matrix **Z**
- Eigendecomposition of the scatter matrix or covariance matrix or kernel matrix

The **eigenvalues (or singular values)** allow to select the **columns of eigenvector matrix (**U**)** to project data **Z**

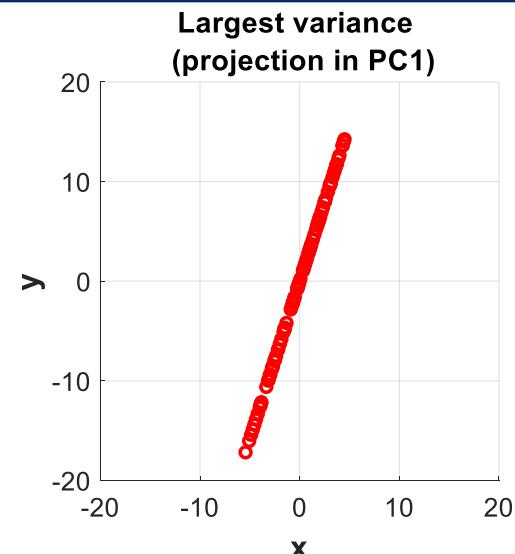
Dimensionality reduction: PCA projections

Projection

Data (the rows of data matrix) can be projected onto m^{th} eigenvector \mathbf{u}_m

$$\mathbf{P}_m = \mathbf{Z} \mathbf{u}_m$$

- \mathbf{P}_m projections

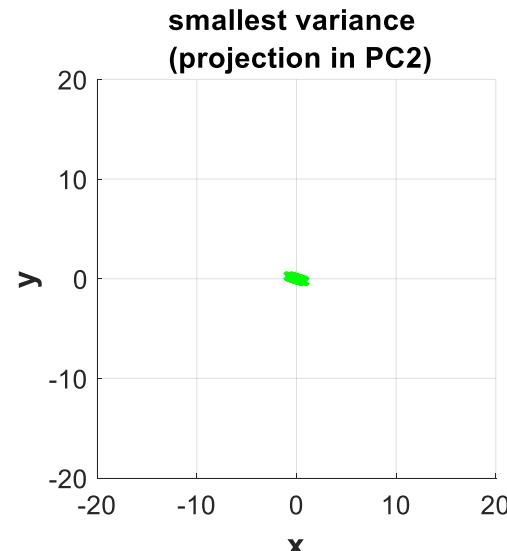


Reconstruction

With projections \mathbf{P}_m and performing

$$\tilde{\mathbf{Z}} = \mathbf{P}_m \mathbf{u}_m^T$$

Is possible to reconstruct to the original dimension, e.g, an approximation $\tilde{\mathbf{Z}}$ of the original centered values \mathbf{Z}



Dimensionality reduction: PCA projections

PCA MODEL: assuming that

- Eigenvalue matrix (or singular value matrix) has diagonal entries (i, i) ordered in decreasing order:

$$\lambda_1 > \lambda_2 > \dots > \lambda_L > \dots > \lambda_D$$

- The columns of eigenvector matrix \mathbf{U} is formed with D eigenvectors
 - i^{th} column is related with the corresponding eigenvalue λ_i

Dimension reduction occurs by projecting the data onto the first L columns of \mathbf{U}

- forming a $D \times L$ matrix \mathbf{U}_L

Note: user assigns L or defines a criterium (like explained variance) to calculate L

Dimensionality reduction: PCA projections

Considering the ordered eigenvalues

$$\lambda_1 > \lambda_2 > \dots > \lambda_L > \dots > \lambda_D$$

The criterium **Explained variance**

$$\frac{\sum_{i=1}^L \lambda_i}{\sum_{i=1}^D \lambda_i} \times 100 \geq \text{threshold}$$

- where D is the total number of non-zero eigenvalues, then
 - user defines the *threshold* (common values are 80% to 95%)
 - and L is calculated according the required threshold

Dimensionality reduction: PCA – number of PC

Dimension reduction

- Using the first L principal directions

$$\mathbf{P} = \mathbf{Z} \mathbf{U}_L$$

- \mathbf{P} is a $N \times L$ matrix -> new representation of the data with small dimension
 - The new feature vector has only $L (< D)$ entries

Recovering to the original dimension

$$\tilde{\mathbf{Z}} = \mathbf{P} \mathbf{U}_L^T$$

- The recovery data can be compared with the original:
 - *Square error or mean square error (related with discarded eigenvalues)*

Dimensionality reduction: LDA

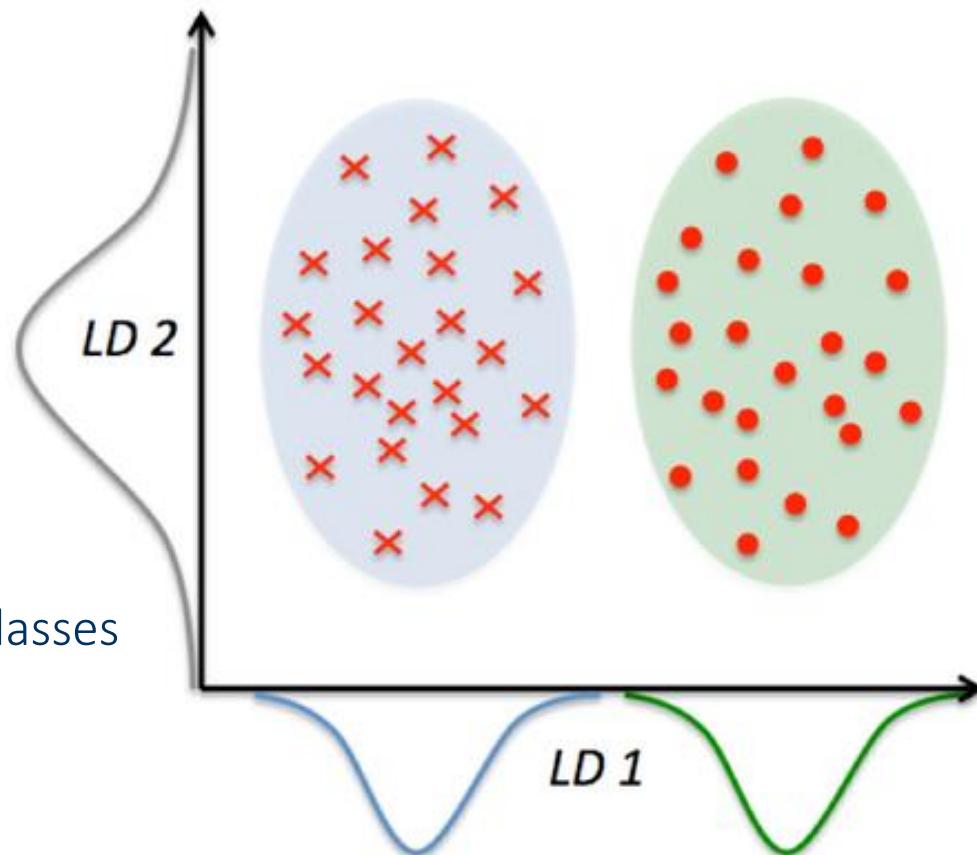
Linear Discriminant Analysis (LDA)

- **Supervised** data reduction technique
- Reduces high dimensions into low dimension subspace
 - dimension = #classes - 1
- Projects the data points onto new axes:
 - It **maximizes** the distance **between** the means of each category (maximizes separability among class categories)
 - It **minimizes** the **variation within** each category
- These new components are a **linear combination of all the features** that separates two or more categories of objects
- The resulting combination may be used as a **linear classifier**

But there are certain **assumptions** Linear Discriminant Analysis makes on the data set...

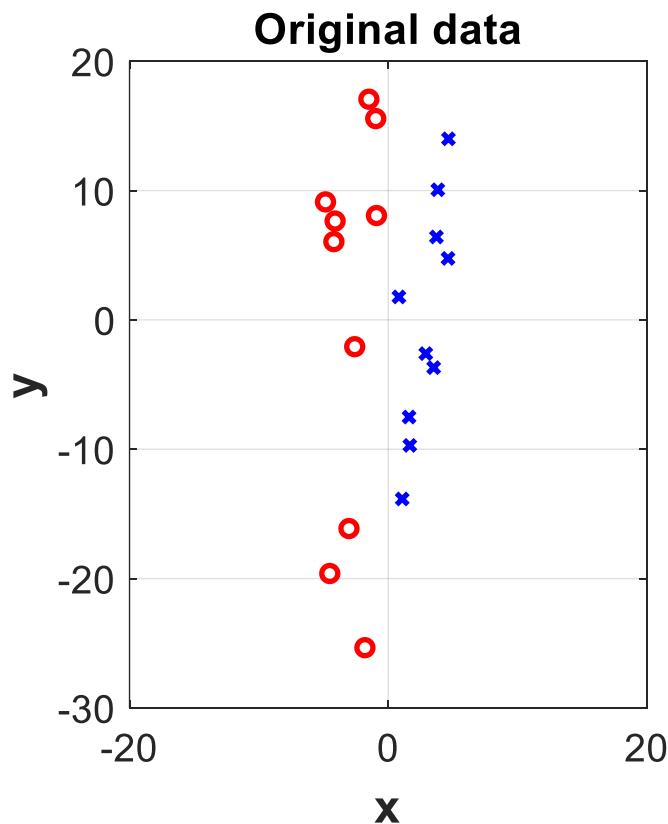
Dimensionality reduction: LDA

Linear Discriminant Analysis (LDA)



Dimensionality reduction: LDA

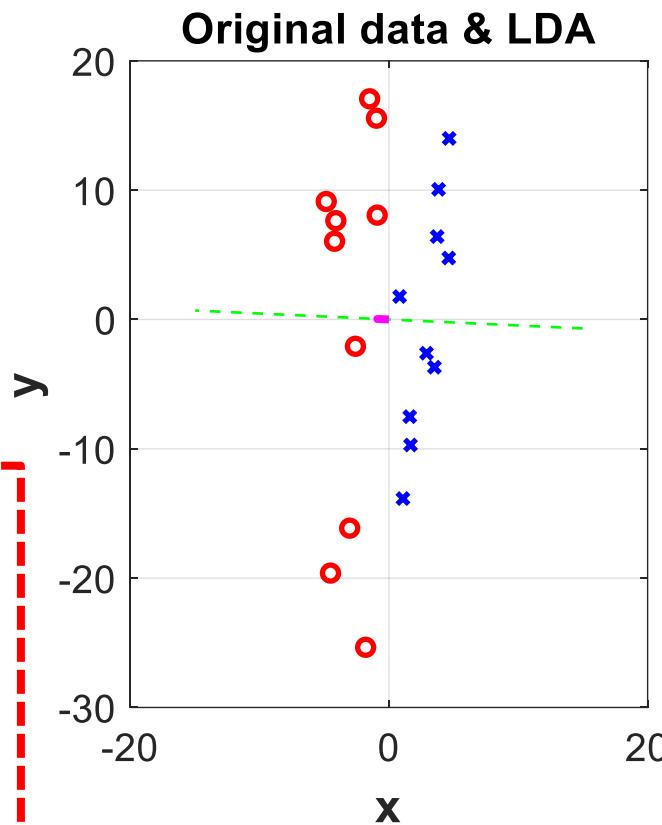
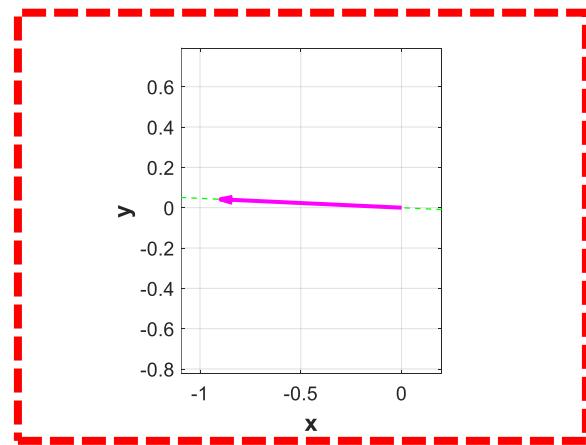
Linear Discriminant Analysis (LDA)



Dimensionality reduction: LDA

Linear Discriminant Analysis (LDA)

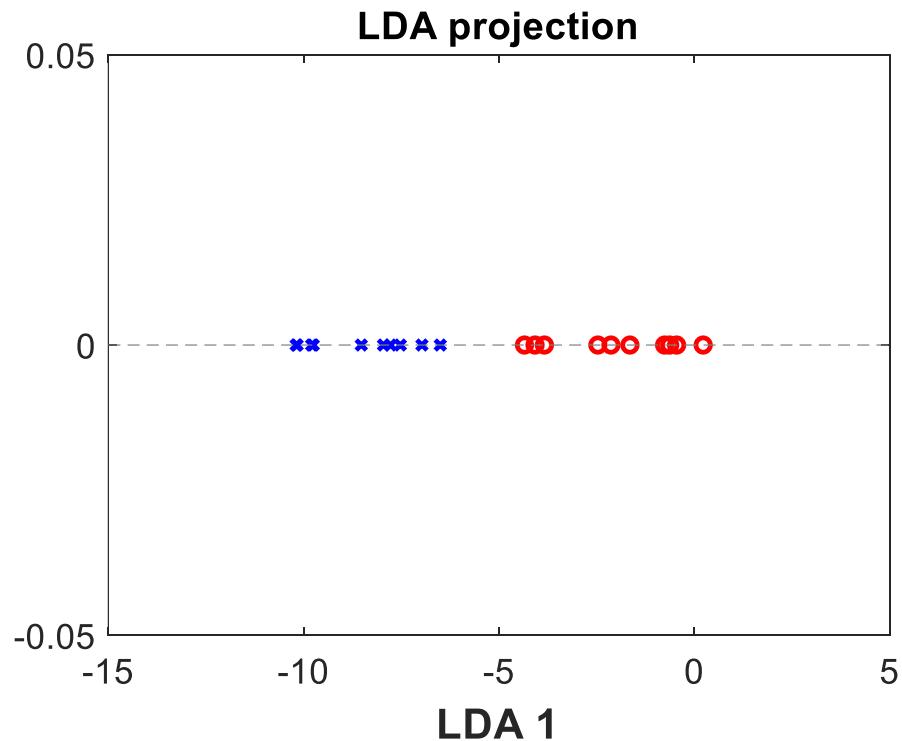
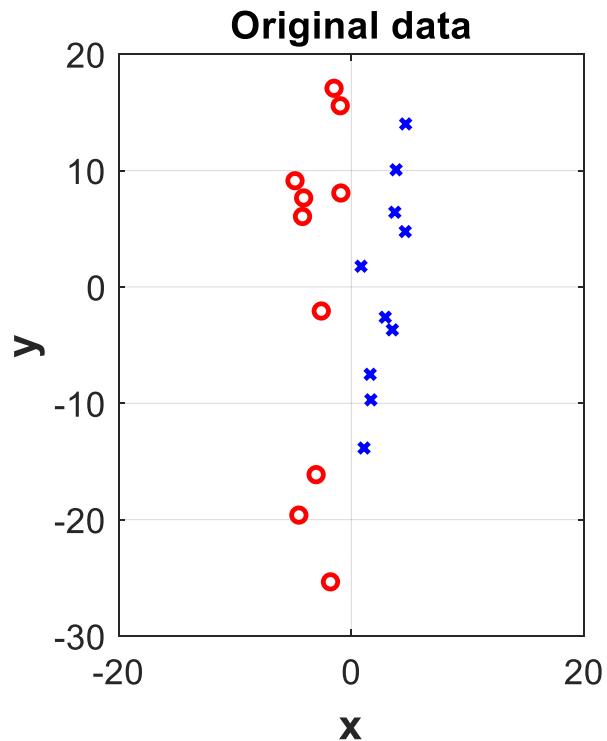
$$\text{PC} = [-0.9989 \quad 0.0462]$$



Dimensionality reduction: LDA

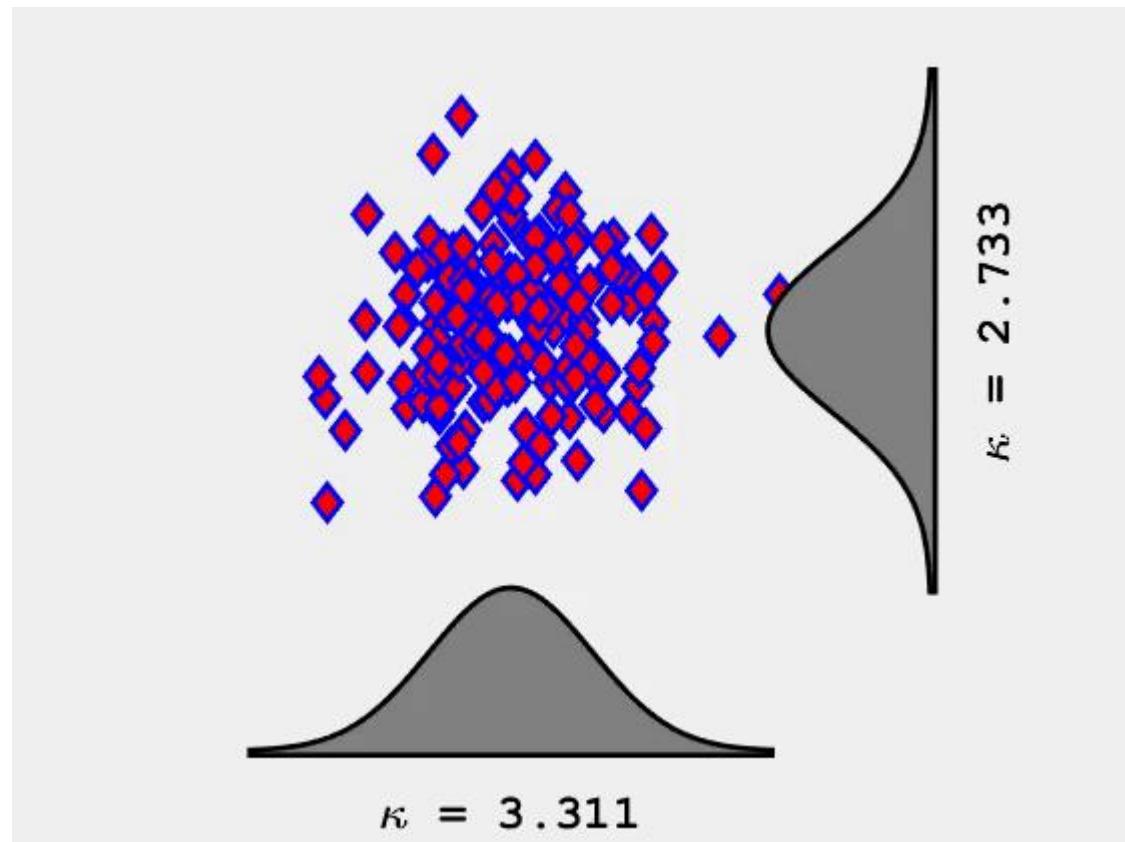
Linear Discriminant Analysis (LDA)

$$\text{LDA1} = [-0.9989 \quad 0.0462]$$



Dimensionality reduction: LDA

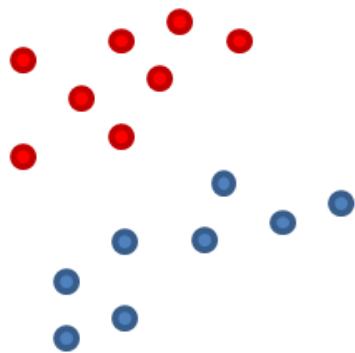
Linear Discriminant Analysis (LDA)



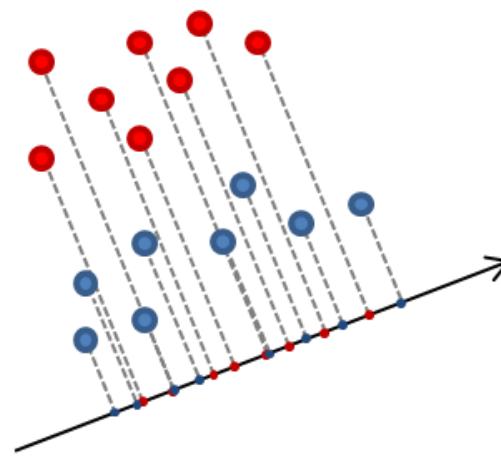
<https://towardsdatascience.com/interesting-projections-where-pca-fails-fe64ddca73e6>

Dimensionality reduction: PCA vs LDA

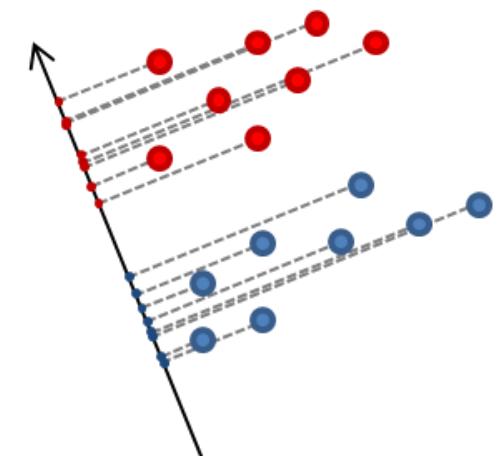
Labelled data



PCA projection:
Maximising the variance of
the whole set



LDA projection:
Maximising the distance
between groups



Contents

- Data Quality
- Data Pre-processing
 - Feature extraction
 - Data integration
 - Data cleaning
 - Feature transformation
 - Feature Engineering
 - Data reduction
- Summary

Summary

- Data quality
- Data pre-processing
 - Feature extraction
 - Data integration
 - Data cleaning
 - Feature transformation
 - Aggregation
 - Scaling
 - Discretization
 - Binarization
 - Feature Engineering
 - Data reduction
 - Numerosity reduction
 - Dimensionality reduction

Homework

- Assignment II (see eLearning)
 - Data Preparation:
 - Hands on: Handling categoric
 - Hands on: Handling missing
 - Hands on: Data Preprocessing

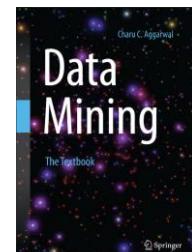
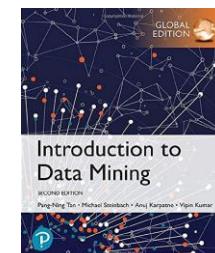
Bibliography

Introduction to Data Mining, Pang-Ning Tan, Michael Steinbach, Anuj Karpatne, Vipin Kumar, *Pearson*, 2019 (chap 2)

Data Mining, the Textbook, Charu C. Aggarwal, *Springer*, 2015 (chap 2)

<https://towardsdatascience.com/smarter-ways-to-encode-categorical-data-for-machine-learning-part-1-of-3-6dca2f71b159>

<https://sebastianraschka.com/faq/docs/lda-vs-pca.html>



Data Mining

Descriptive Modelling

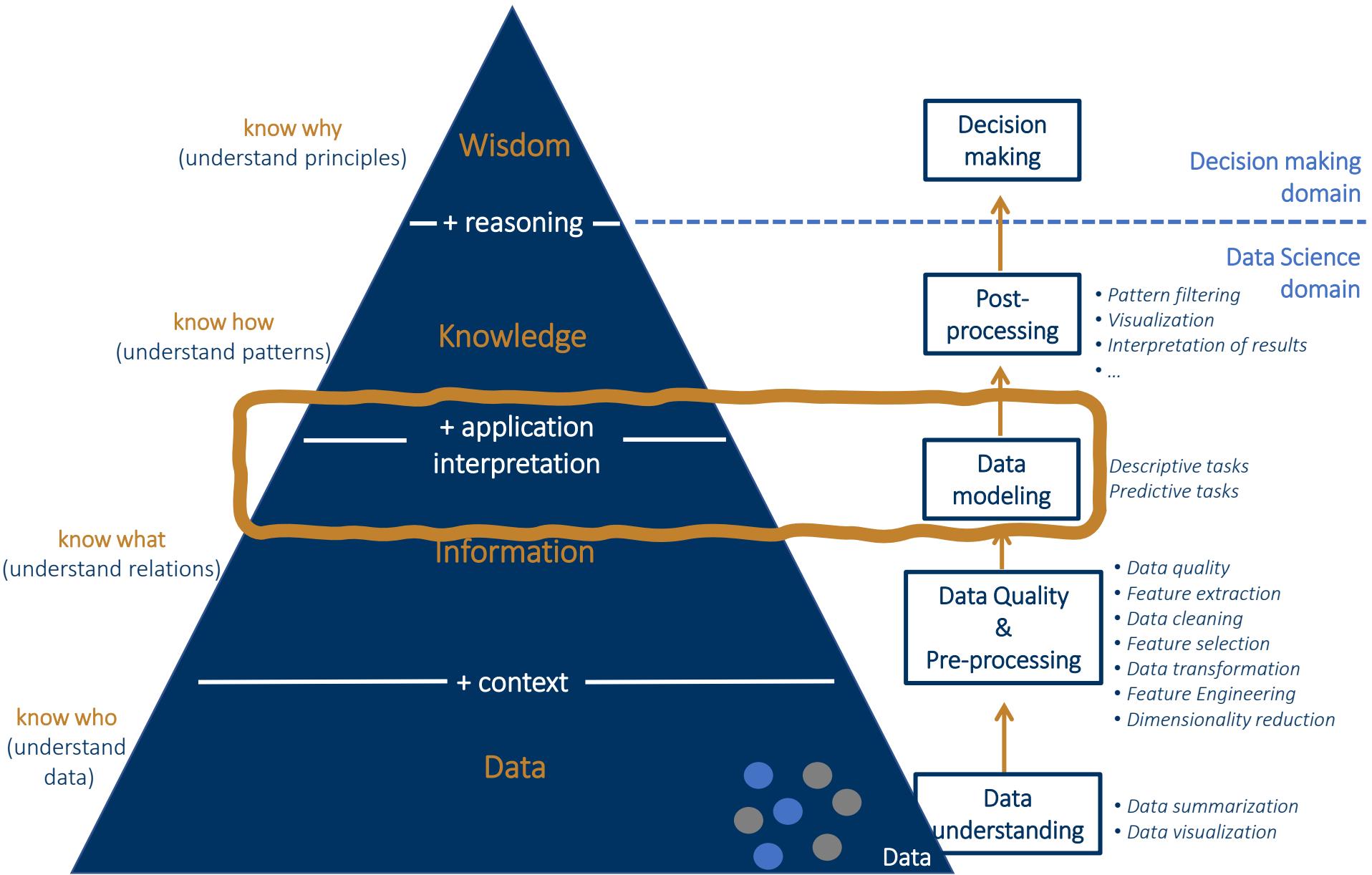
Raquel Sebastião

Departamento de Eletrónica, Telecomunicações e Informática

Universidade de Aveiro

raquel.sebastiao@ua.pt

2022/2023



Contents

- Descriptive analytics
- Cluster analysis
- Main categories of clustering methods
- Clustering validation
- Summary

Descriptive Analytics

Goals:

- **Describe/summarize** or discover **structure** in collections of data
 - Data summarization and visualization are simple forms of descriptive analytics
 - Cluster analysis is frequently used for discovering **structure/groups** in data
 - Clustering the data into similar groups helps greatly in **summarizing the data and understanding it**

Contents

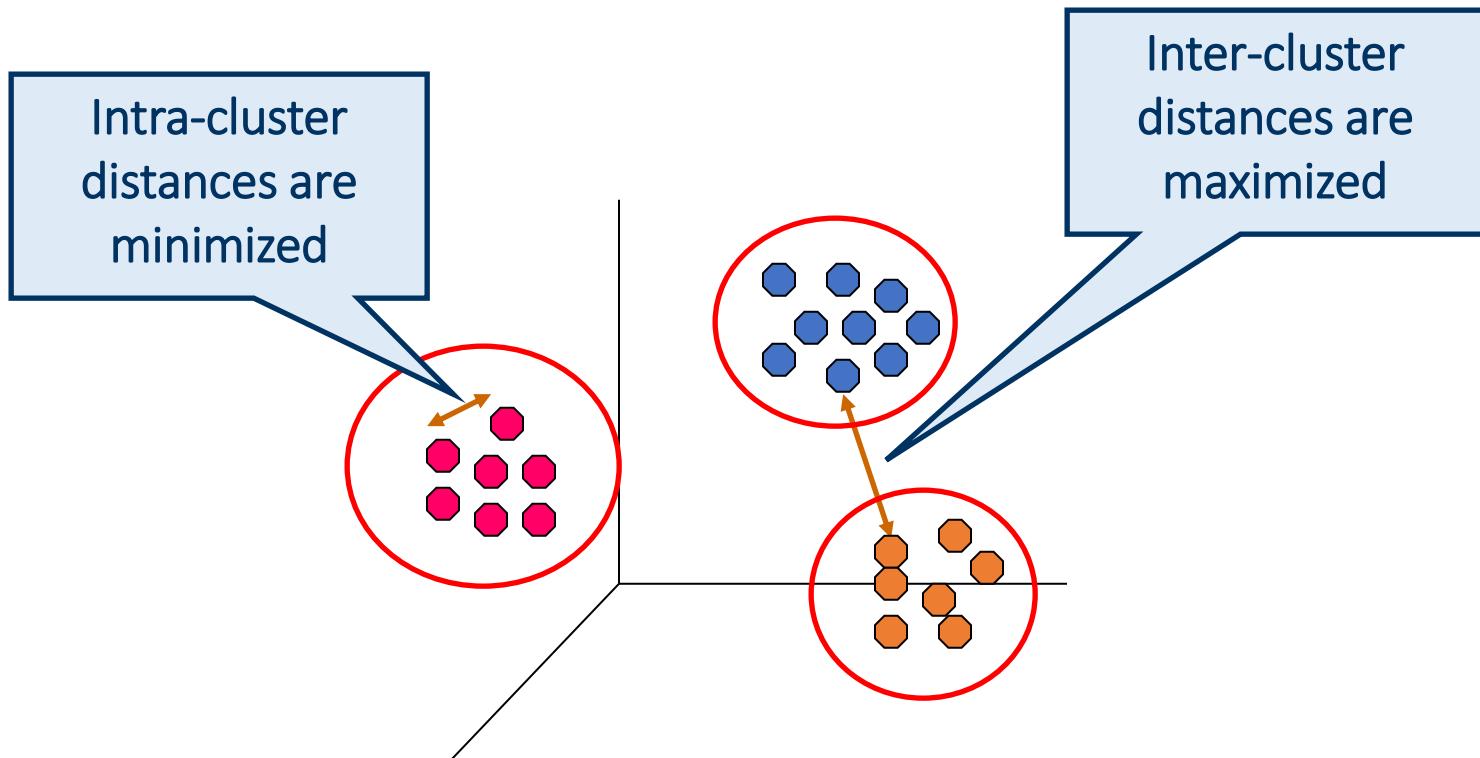
- Descriptive analytics
- Cluster analysis
- Main categories of clustering methods
- Clustering validation
- Summary

Cluster analysis (clustering)

- Process of grouping data objects (or observations) into subsets
 - exploitation of similarities (or differences) between objects(or observations, data points)
 - Objects within a group are similar (or related) to one another and different from (or unrelated to) the objects in other groups
- Unsupervised technique - no labeled data
 - finds groups based only on information in the data that describes the objects and their relationships

Cluster analysis (clustering)

- Process of grouping data objects (or observations) into subsets
- Objects within a group are similar (or related) to one another and different from (or unrelated to) the objects in other groups



Cluster analysis (clustering): goals

- Find some structure on the data set (obtaining “natural” groups)
- Provide some abstraction of the found groups
 - representation of their main features
 - a prototype for each group
- Gain novel insights of data

Cluster analysis (clustering): motivation

- Data reduction
 - All objects within a cluster/group are substituted (represented) by the corresponding cluster representative
- Hypothesis generation
- Hypothesis testing
- Prediction based on groups
 - a cluster/group of data objects can be treated as an implicit class
 - clustering is a form of **learning by observation**, rather than *learning by examples*

Cluster analysis (clustering): applications

- Business and Marketing
 - group clients with similar buying behavior
 - describe different market segments from a set of potential clients
 - group stocks with similar price fluctuations
- Medical
 - find patients with similar symptoms
 - provide treatment recommendations based on groups of similar patients
 - identify groups of diseases
 - Group diagnostic imaging techniques with similar characteristics

Cluster Analysis (clustering): applications

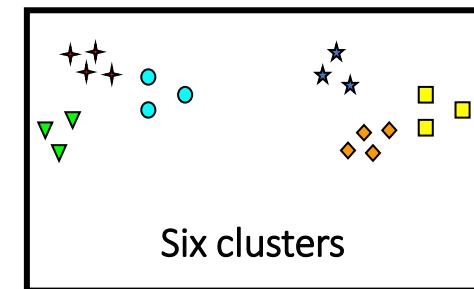
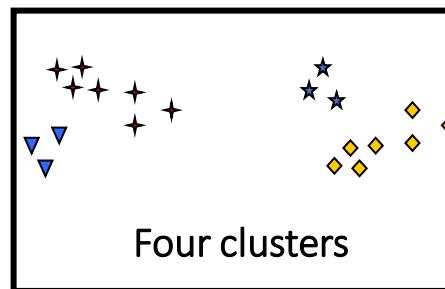
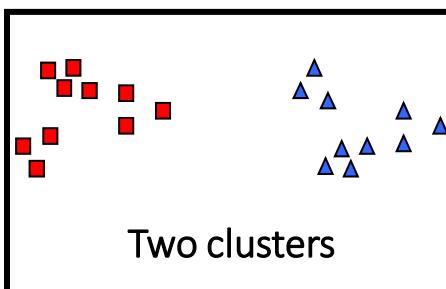
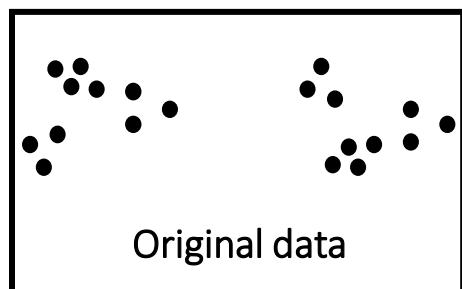
- Document retrieval
 - find documents with similar contents (travels, economy, ...).
- Image retrieval
 - find images with similar contents (sport, landscapes, ...)
- Biology
 - describe spatial and temporal communities of organisms
 - group genes or proteins that have similar functionality
- Web Mining
 - find communities in social networks
 - build recommendation systems

Cluster analysis (clustering): subjectivity

Different clusters may result, depending on

- the proximity measure
- the clustering criterion
- the clustering algorithm

How many clusters?



Contents

- Descriptive analytics
- Cluster analysis
- Main categories of clustering methods
- Clustering validation
- Summary

Clustering: main categories

Partitional: divide the observations in k partitions according to some criterion

- **Representative based:** identify cluster representatives (centroids)
- **Density based¹:** locate regions of high density in the feature space
- **Model based:** assume a probability model for the data
- **Grid based:** discretize the data into p intervals (typically equal-width)
- **Neural-Network Based:** Self Organizing Maps (SOM) - consider an underlying “topology” that relates cluster centroids to one another

Hierarchical: successive development of clusters by generating a hierarchy of groups

- **Agglomerative:** generate a hierarchy from bottom-up (from n to 1 group)
- **Divisive:** create a hierarchy in a top-down way (from 1 to n groups)

¹ can be considered two-level hierarchical agglomerative

Clustering: task stages

- Proximity measure
 - quantifies the term similar or dissimilar
- Clustering criterion
 - cost function or some type of rules
- Clustering algorithm
 - steps followed to reveal the structure, based on the proximity measure and the adopted criterion
- Validation of the results
- Interpretation of the results

Clustering: partitional methods

Representative based

Discovering the groupings in the data by optimizing a specific objective function and iteratively improving the quality of partitions

- Cluster compactness
 - how similar are objects within the same cluster
- Cluster separation
 - how far is the cluster from the other clusters
- A clustering solution assigns all the objects to a cluster
 - hard clustering: an object belongs to a single cluster
 - fuzzy clustering: each object has a probability of belonging to each cluster

K-means clustering

- Partitional clustering method
- Number of clusters, K , must be specified
 - Methods to determine the “best” K
- Each cluster is represented by the centroid/representative (center point)
- Each object is assigned to the cluster with the closest centroid
- Different kind of proximity measures can be used
 - Manhattan distance (L_1 norm), Euclidean distance (L_2 norm), Cosine similarity, Correlation
- Simple iterative algorithm

K-means clustering

Consider the cluster $C_k = \{x_1, x_2, \dots, x_{n_k}\}$, the **centroid** of C_k is given by

$$c_k = \frac{1}{n_k} \sum_{x_i \in C_k} x_i$$

Goal: obtain a set of clusters C that minimize the criterion

$$h(C) = \sum_{j=1}^K \sum_{x_i \in C_j} d(x_i, c_j)$$

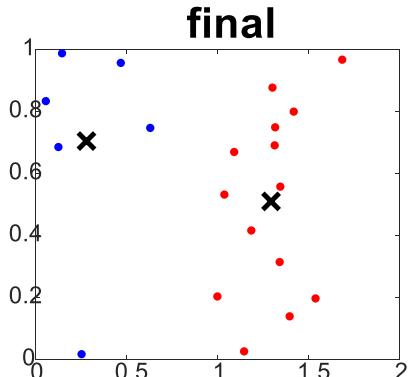
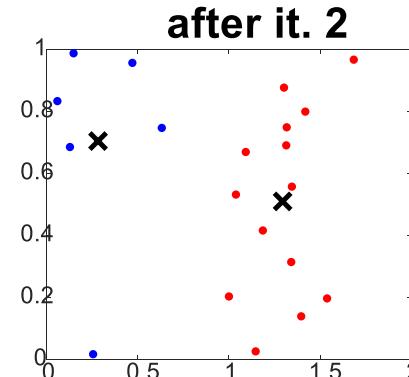
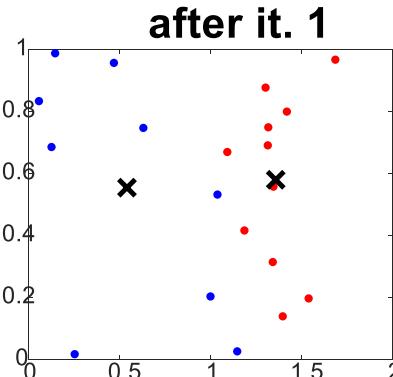
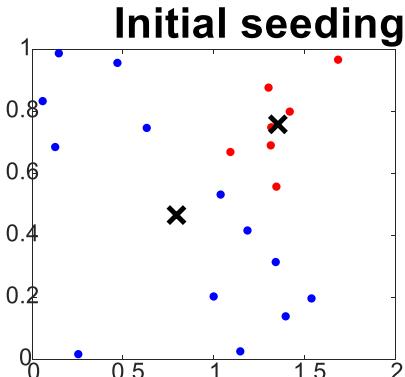
Usual criteria for numerical data

- Sum of square errors (SSE): $d(x_i, c_j) = (x_i - c_j)^2$
- L1 measure: $d(x_i, c_j) = |x_i - c_j|$
- Cosine: $d(x_i, c_j) = 1 - \frac{x_i \cdot c_j}{\|x_i\| \times \|c_j\|}$

K-means clustering

Execution of the K-means clustering algorithm:

- Select K points as the initial centroids/representatives (often randomly chosen)
- Repeat
 - assign each object/observation to the group with the nearest centroid
 - re-compute cluster centroids (i.e., **mean** point) of each cluster
- Until convergence criterion is satisfied (i.e., the centroids stop changing)



K-means clustering: details

- Initial centroids are often chosen **randomly**
- minimize intra-cluster distance and maximize inter-cluster distances

Advantages:

- Stochastic approach that frequently works well. It tends to **identify local minima**
- Most of the convergence happens in the **first few iterations**
 - Often the stopping condition is changed to ‘Until relatively few points change clusters’
- Complexity: $O(n*K*I*d)$ where n : # of objects, K : # of clusters, I : # of iterations, and d : # attributes
 - Normally, $K, I \ll n$; thus, an **efficient** method

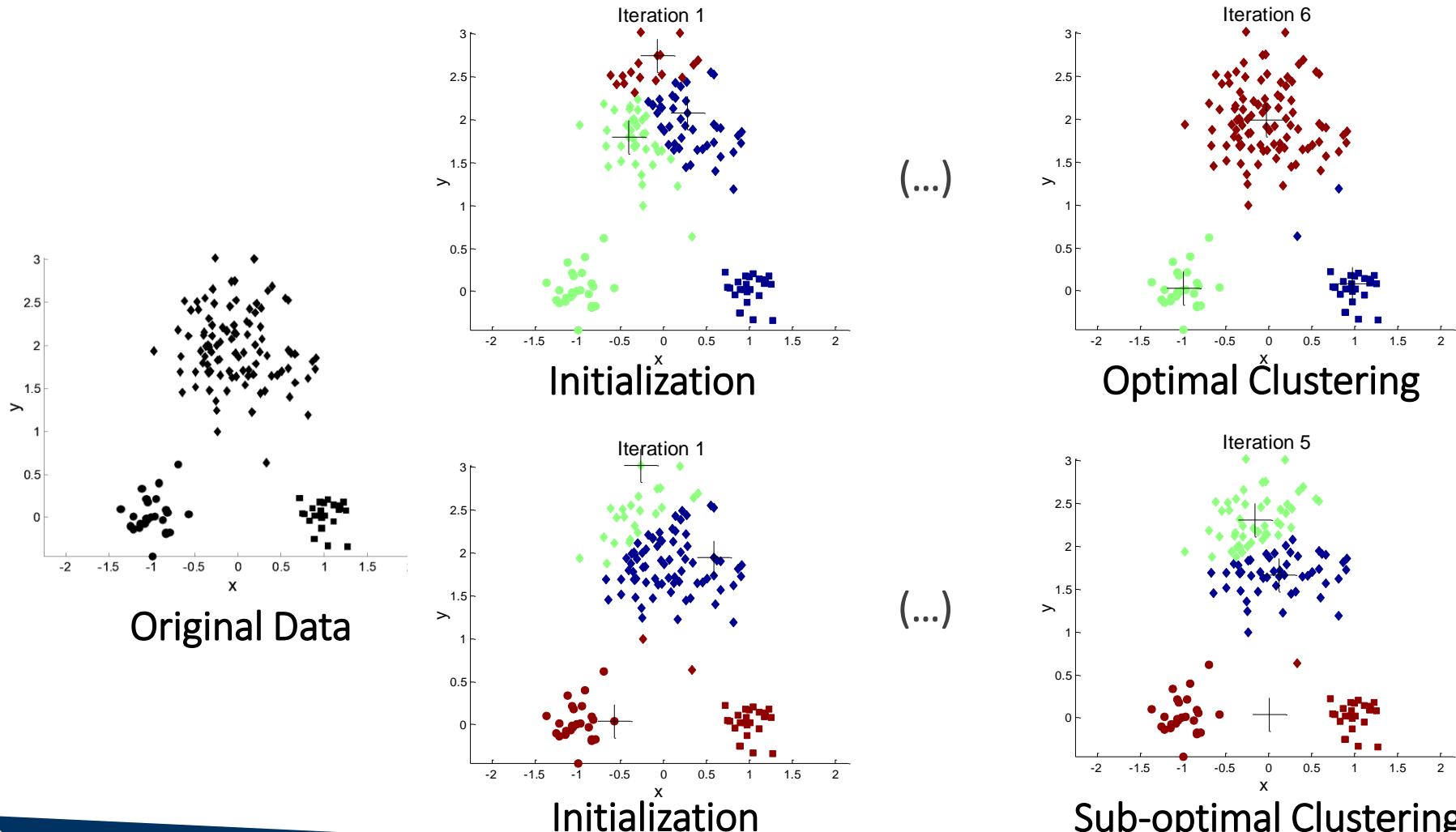
K-means clustering: details

Disadvantages:

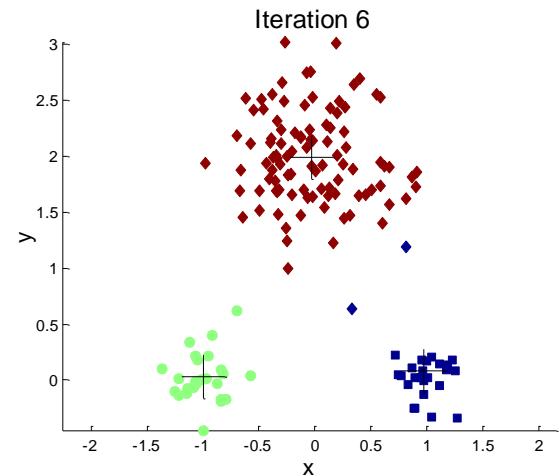
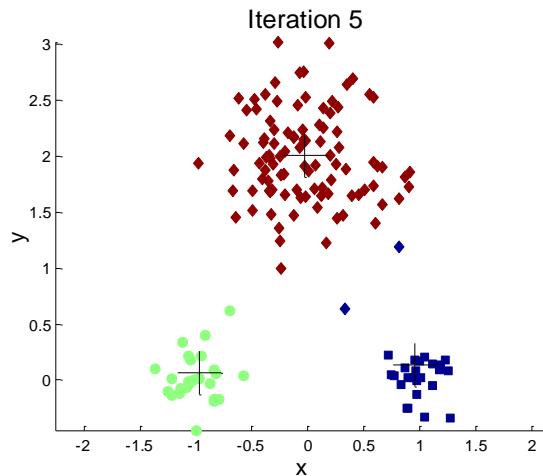
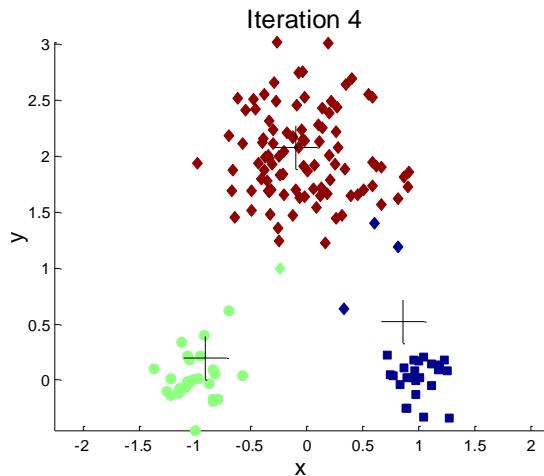
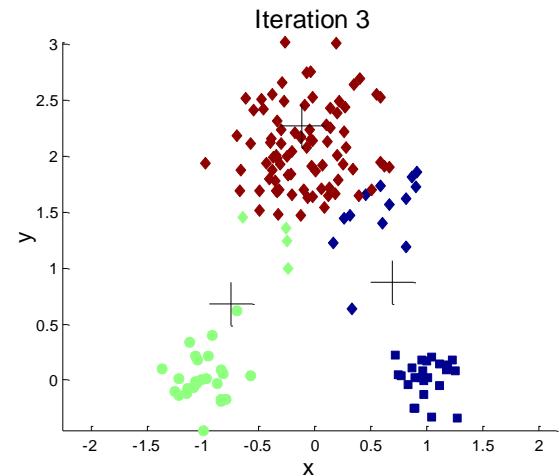
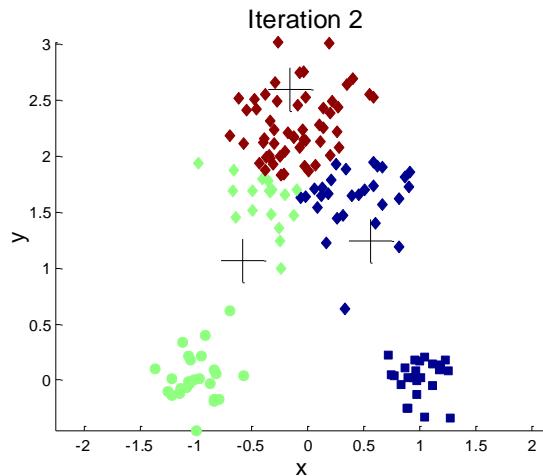
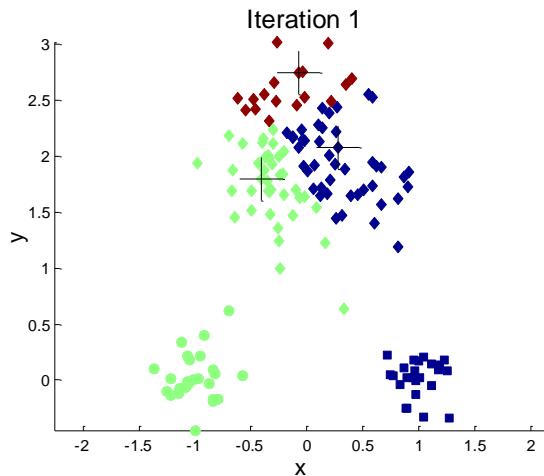
- Different initializations may generate rather different final clusters
- Sensitive to noisy data and outliers
 - Possible solution: remove outliers prior to clustering
 - Alternatives: K-medians, K-medoids, ...
- K-means is applicable only to numerical data
 - Alternatives: K-modes (*categorical data*)
- Not suitable to discover clusters:
 - with different sizes
 - with different densities
 - with non-convex/non-globular shapes
 - Alternatives: kernel K-means, density-based clustering, ...

K-means limitations: poor initialization

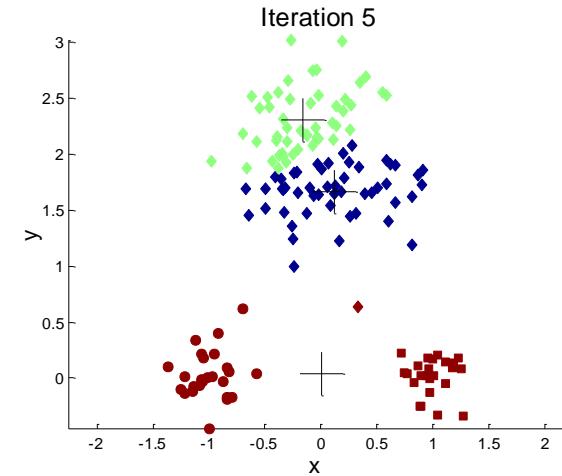
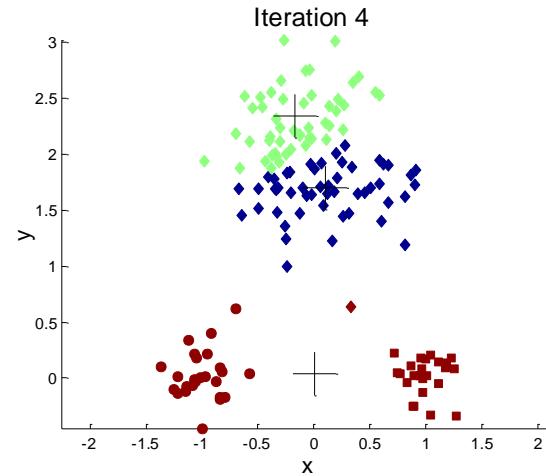
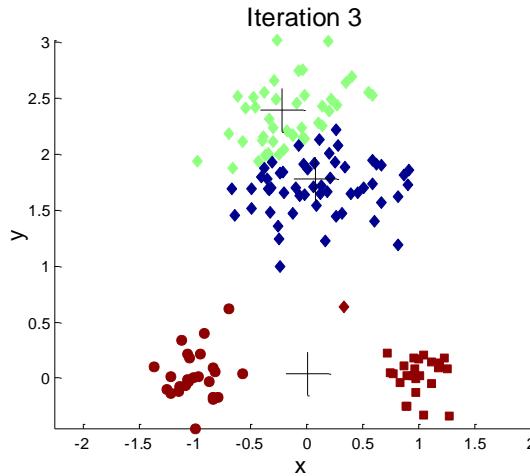
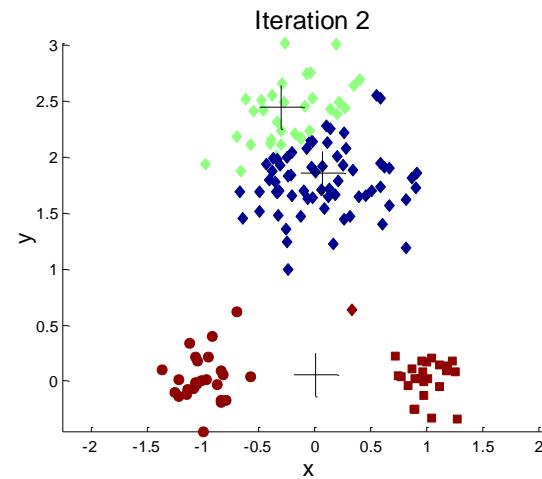
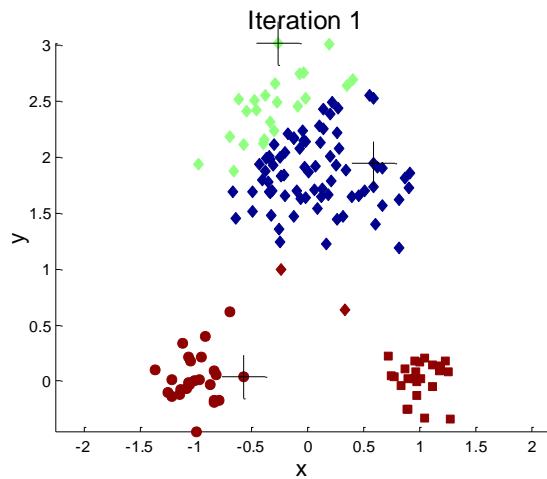
- Different initializations may generate rather different final clusters



K-means limitations: poor initialization



K-means limitations: poor initialization



K-means limitations: poor initialization

Solutions

- Multiple runs with K seeds randomly selected
 - Helps, but probability is not on your side
- K-means++ : select the most widely separated centroids
 - The first centroid is selected at random
 - The next centroid selected is the one that is farthest from the currently selected (selection is based on a weighted probability score)
 - The selection continues until K centroids are obtained
- Use hierarchical clustering to determine initial centroids
- Bisecting K-means
 - Not as susceptible to initialization issues

K-medians clustering: handling outliers

Medians are **less sensitive to outliers** than means

K-medians: Instead of taking the **mean** value of the object in a cluster as a reference point, **medians** are used

Execution of the **K-medians** clustering algorithm:

- Select **K** points as the initial centroids/representatives (i.e., as initial **K** medians)
- Repeat
 - assign each object/observation to the group with the nearest centroid
 - re-compute cluster centroids (i.e., **median** point) of each cluster
- Until convergence criterion is satisfied (i.e., the centroids stop changing)

K-medoids clustering: handling outliers

K-medoids: Instead of taking the **mean** value of the object in a cluster as a representative/centroid, **medoids** are used, which is the most centrally located object in a cluster

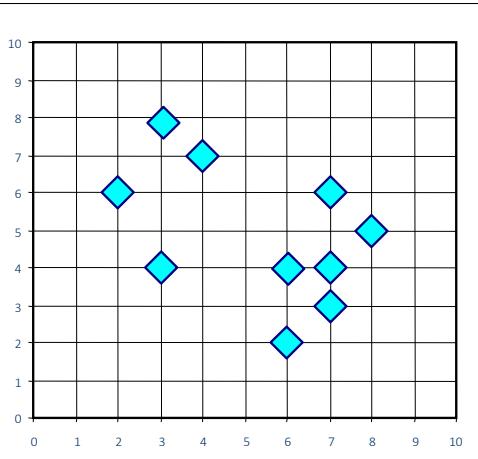
Is more robust to the presence of outliers because it uses original objects as centroids instead of averages that may be subject to the effects of outliers

K-medoids clustering: handling outliers

Execution of the K-medoids clustering algorithm:

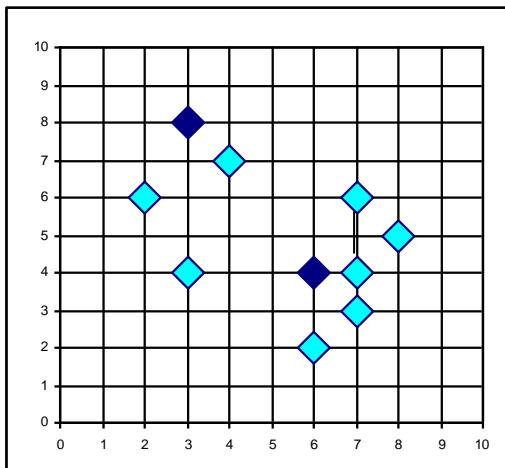
- Select K points as the initial centroids/representatives (i.e., as initial K medoids)
- Repeat
 - assign each object/observation to the group with the nearest medoid
 - Randomly select a non-representative object o_i
 - Compute the total cost S of swapping the medoid m with o_i
 - If $S < 0$, then swap m with o_i to form the new set of medoids
- Until convergence criterion is satisfied (i.e., the centroids stop changing)

K-medoids clustering: PAM (Partitioning Around Medoids)

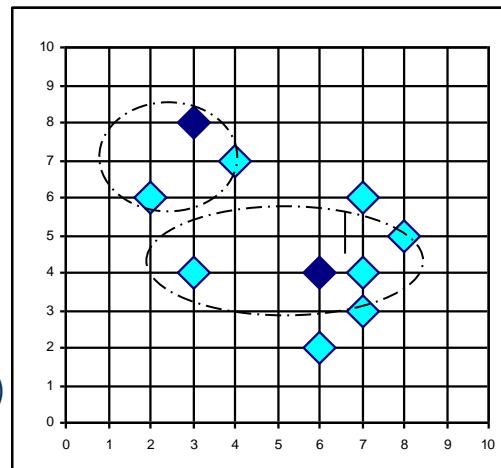


$K = 2$

Arbitrary choose K object as initial medoids



Assign each remaining object to nearest medoid (m)



Randomly select a non-medoid object, O_{random}

Select initial K medoids randomly

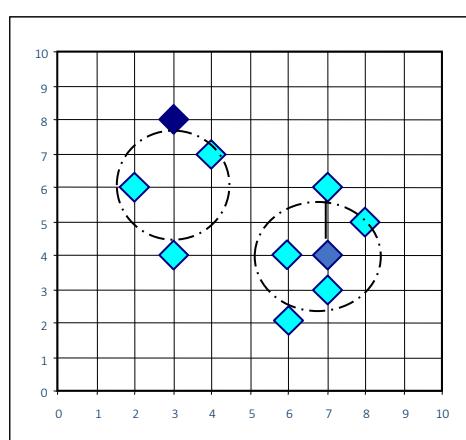
Repeat

Object re-assignment

If quality is improved

Swap medoid m with o_i

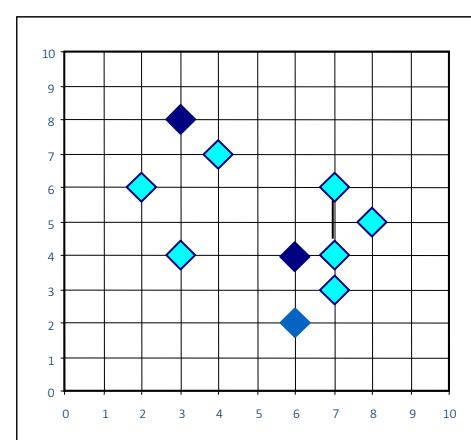
Until convergence criterion is satisfied



Compute total cost of swapping

If quality is improved:

swap m and O_{random}



K-medoids clustering: discussion

PAM (Partitioning Around Medoids)

- Computational complexity: $O(K(n - K)^2)$ (quite expensive!)
- PAM works effectively for small data sets but does not scale well for large data sets (due to the computational complexity)

Efficient improvements on PAM

- CLARA (Clustering Large Applications)
 - PAM on samples; $O(Ks^2 + K(n - K))$, s is the sample size
- CLARANS (Clustering Large Applications based on RANdomized Search)
 - randomized re-sampling
 - ensure efficiency & quality

K-modes clustering: categorical data

- K-Means cannot handle non-numerical (categorical) data
 - Mapping categorical value to 1/0 cannot generate quality clusters
- *K-Modes*: an extension to K-Means by replacing means of clusters with *modes* as representatives/centroids
- Dissimilarity measure for categorical data: frequency-based

Algorithm is still based on iterative *object cluster assignment* and *centroid update*

kernel K-means clustering

K-Means can only detect clusters that are linearly separable

Kernel K-Means can be used to detect non-convex clusters

- A region is **convex** if it contains all the line segments connecting any pair of its points. Otherwise, it is **concave**
- Idea: Project data onto the **high-dimensional kernel space**, and then perform ***K-Means*** clustering

kernel K-means clustering

- Idea: Project data onto the high-dimensional kernel space, and then perform *K-Means* clustering
- Map data points in the input space onto a high-dimensional feature space using the kernel function
- Perform *K-Means* on the mapped feature space
- Computational complexity is higher than K-Means
 - Need to compute and store $n \times n$ kernel matrix generated from the kernel function on the original data, where n is the number of points
- *Spectral clustering* can be considered as a variant of Kernel K-Means clustering

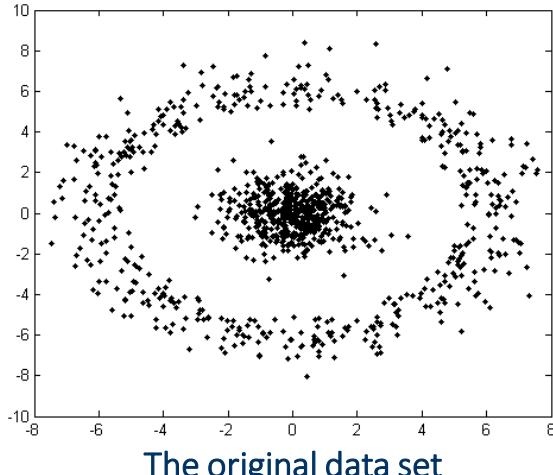
kernel K-means clustering

- Typical Kernel functions
 - Polynomial kernel of degree
 - Gaussian radial basis function (RBF) kernel:
 - Sigmoid kernel
- The formula for **kernel matrix K** for any two points $x_i, x_j \in C_k$:

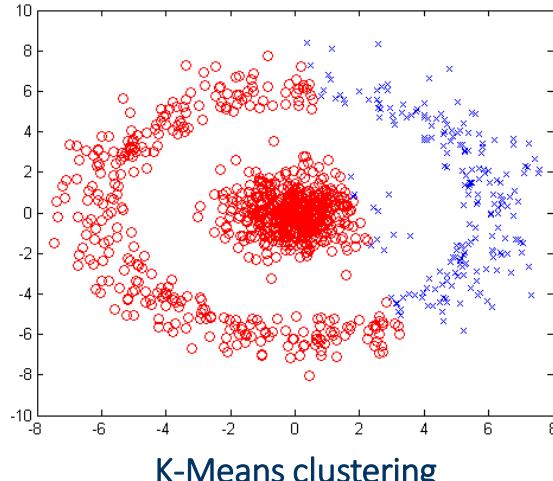
$$K_{x_i x_j} = \phi(x_i) \bullet \phi(x_j)$$

- All steps of K-means are based on dot product, but the centroid is never explicitly computed
- Clustering can be performed without the actual individual projections $\phi(x_i)$ and $\phi(x_j)$ for the data points $x_i, x_j \in C_k$

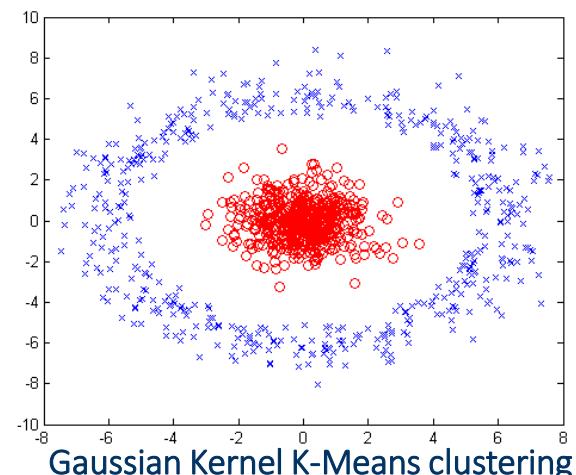
kernel K-means clustering



The original data set



K-Means clustering



Gaussian Kernel K-Means clustering

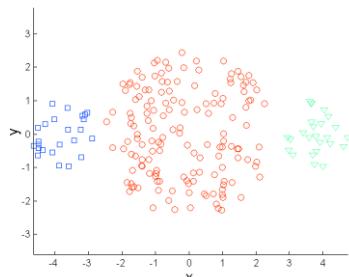
- This data set cannot generate quality clusters by **K-Means** since it contains non-convex clusters
- **Kernel transformation** maps data to a kernel matrix \mathbf{K} for any two points x_i, x_j :

$$K_{x_i x_j} = \phi(x_i) \bullet \phi(x_j)$$

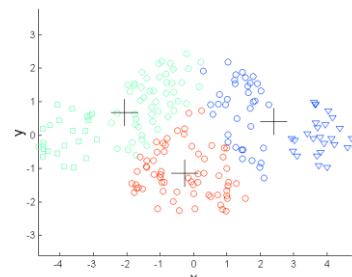
- Gaussian kernel: $K(\mathbf{X}_i, \mathbf{X}_j) = e^{-\frac{\|\mathbf{X}_i - \mathbf{X}_j\|^2}{2\sigma^2}}$
- K-Means clustering is conducted on the mapped data, generating quality clusters

K-means “variations” limitations

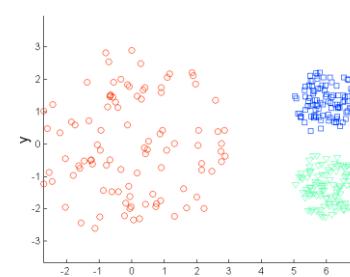
- Clusters of different sizes, densities and with non-convex/non-globular shapes



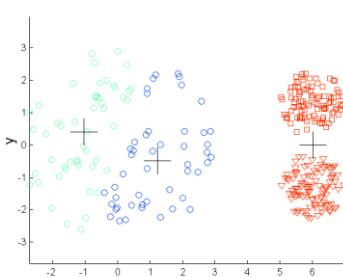
Original Data



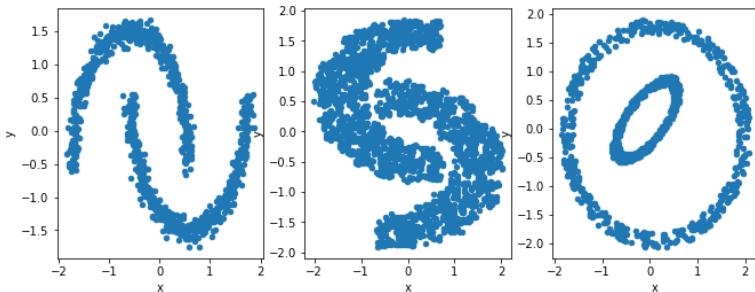
K-means (3 Clusters)



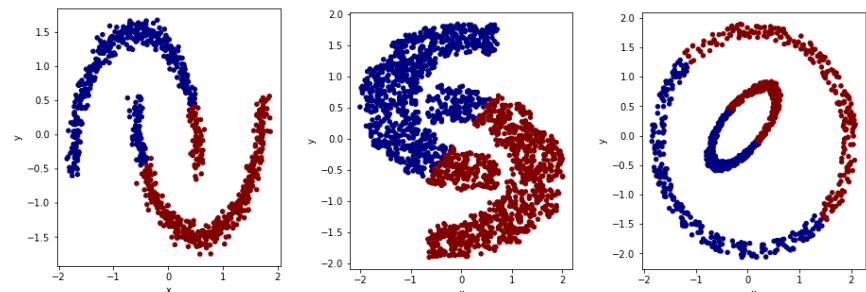
Original Data



K-means (3 Clusters)



Original Data



K-means (2 Clusters)

- Data with outliers/noise

Density based clustering

DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

- Clusters are regions of **high density** that are separated from one another by regions on **low density**
- The **density** of a single observation is **estimated by the number of observations that are within a specified radius** (**eps** - parameter of the method)
- It does not require the user to specify the **number of groups**
- Input the radius (**eps**) and the minimum number of points (**MinPts**)

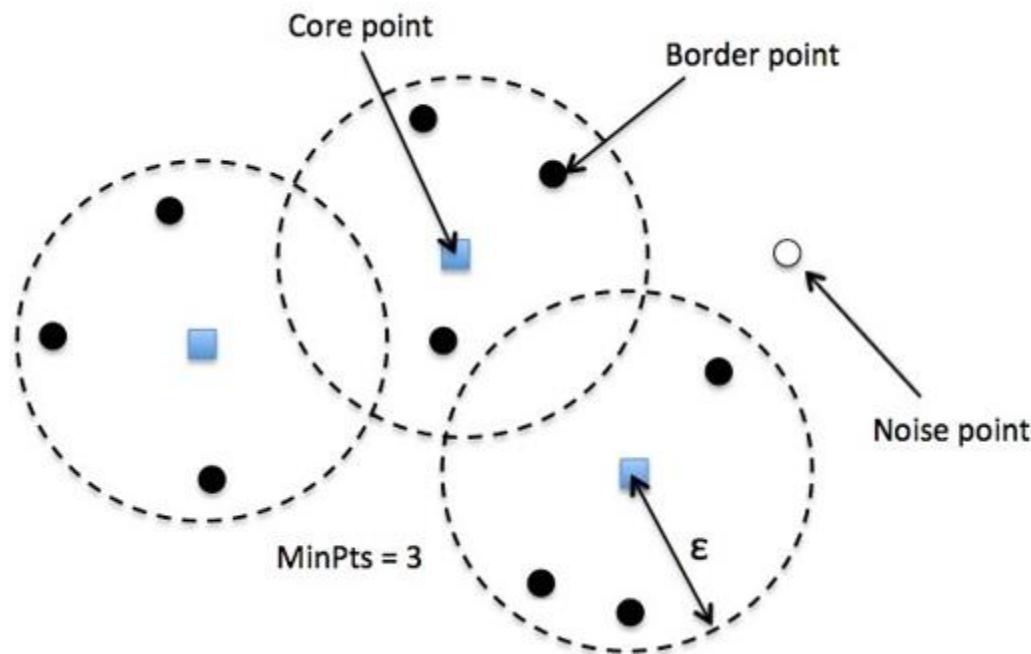
Density based clustering

Observations are identified as:

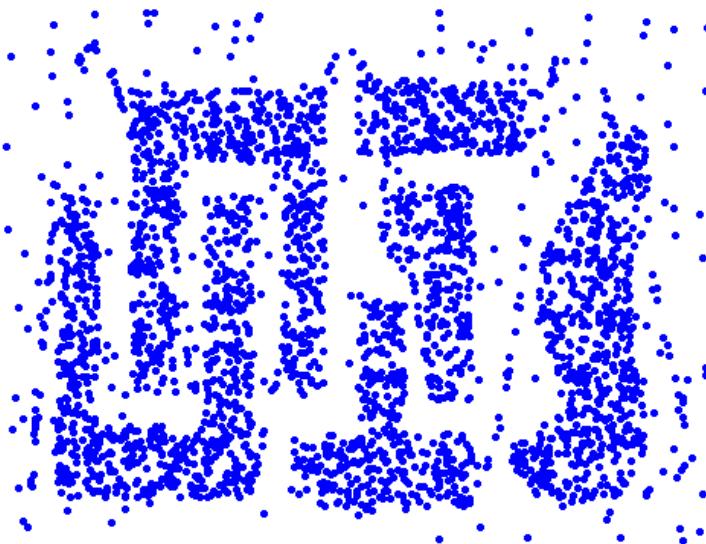
- **core point**: if it has at least a specified number of points (**MinPts**) within **eps** radius
 - These are points that are at the interior of a cluster
 - Counts the point itself
- **border point**: if the number of observations within its radius does not reach the threshold but it is within the radius of a core point
- **noise point**: does not have enough observations within their radius, nor is sufficiently close to any core point (any point that is not a core point or a border point)

Density based clustering: DBSCAN

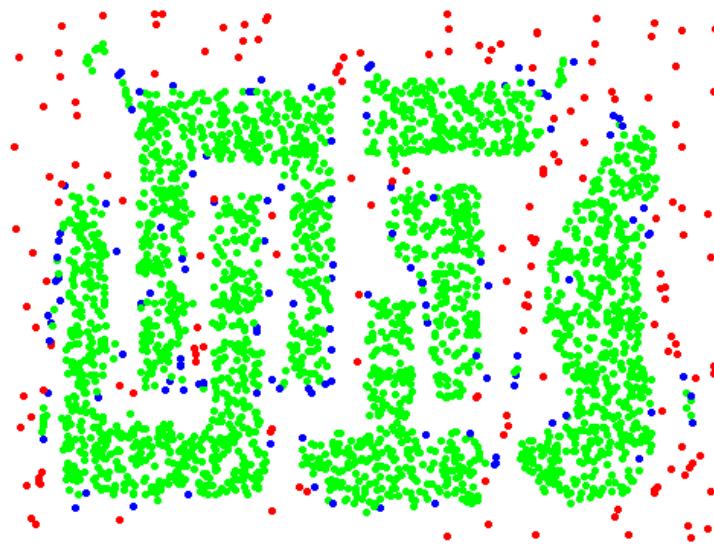
Core, border and noise points



Density based clustering: DBSCAN



Original Points



Point types: core, border and noise

Eps = 10, MinPts = 4

Density based clustering: DBSCAN

Form clusters using core points, and assign border points to one of its neighboring clusters

- Identify all points as core, border, or noise points
- Eliminate noise points
- Put an edge between all core points within a distance Eps of each other
- Make each group of connected core points into a separate cluster
- Assign each border point to one of the clusters of its associated core points

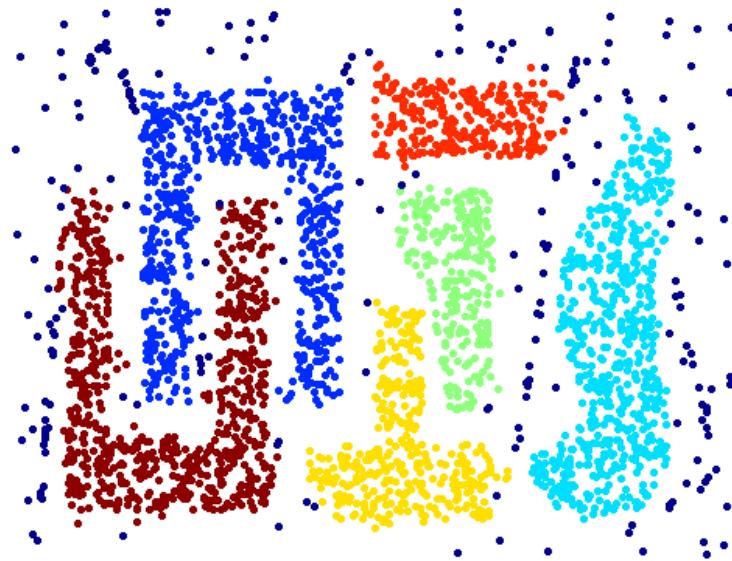
Density based clustering: DBSCAN

Advantages:

- Can handle clusters with different shapes and sizes
- Resistant to noise

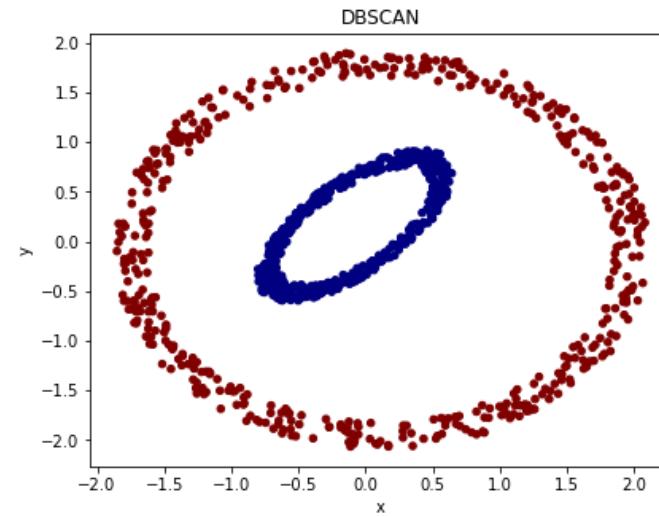
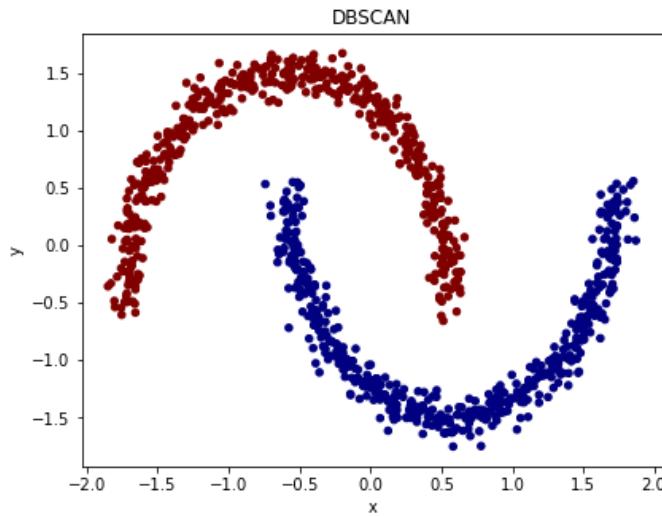
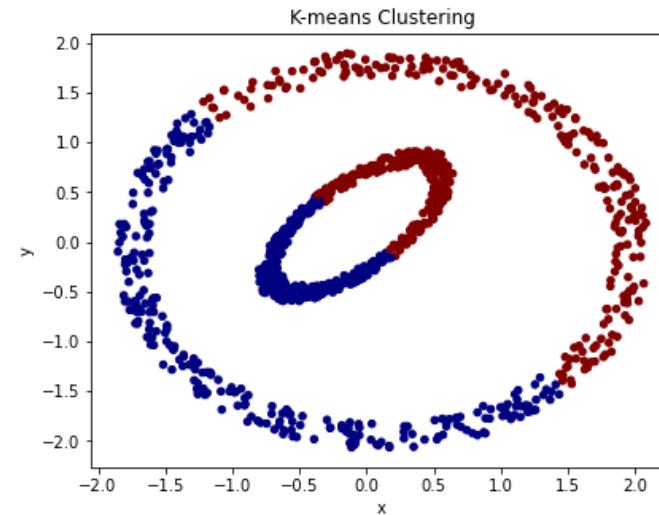
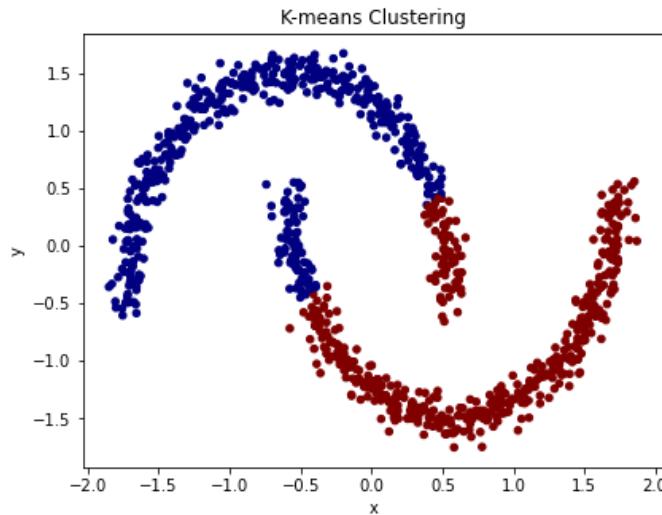
Disadvantages:

- Varying densities
- High-dimensional data



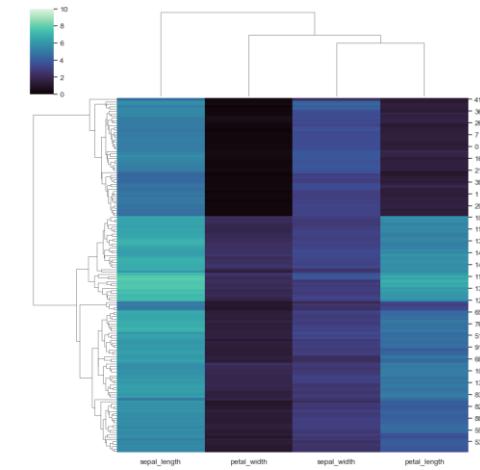
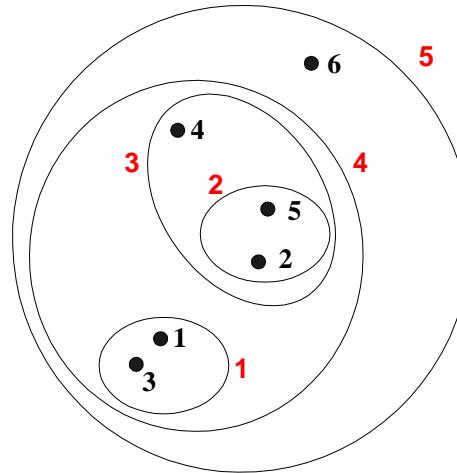
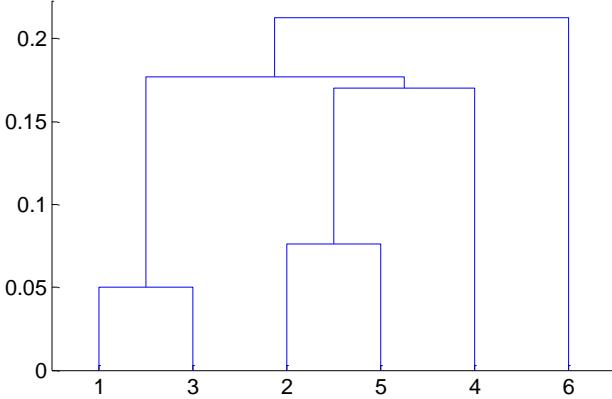
Clusters (dark blue points indicate noise)

Density based clustering: DBSCAN



Hierarchical clustering

- Obtain a set of nested clusters organized as a hierarchical tree
 - each level represents a possible solution with x groups
- Can be visualized as a dendrogram
 - A tree like diagram that records the sequences of merges or splits
- Additional visualization: combine the dendrogram with heat maps

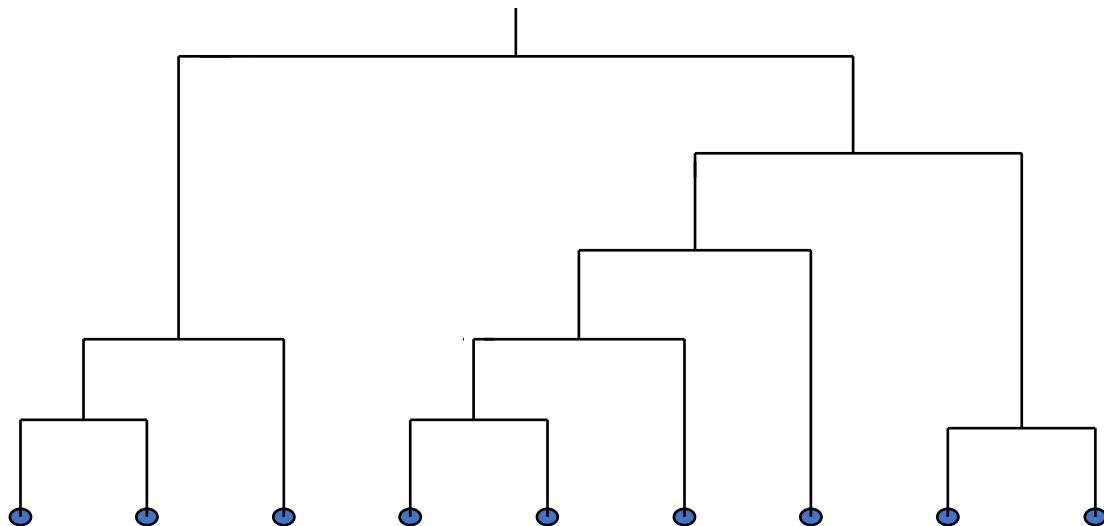


Hierarchical clustering

- Do not have to assume a pre-defined number of clusters
 - Any desired number of clusters can be obtained by ‘cutting’ the dendrogram at the proper level (the user can specify the desired number of clusters)
- More deterministic
- No iterative refinement
- Two categories of algorithms: **Agglomerative** and **Divisive**
- Key operation is the computation of the proximity of two clusters
- Different approaches to defining the distance between clusters distinguish the different algorithms

Hierarchical clustering

- **Dendrogram:** Decompose a set of data objects into a **tree** of clusters by multi-level nested partitioning
- A **clustering** of the data objects is obtained by **cutting** the dendrogram at the desired level, and each **connected component** forms a cluster



Hierarchical clustering generates a dendrogram (a hierarchy of clusters)

Hierarchical clustering

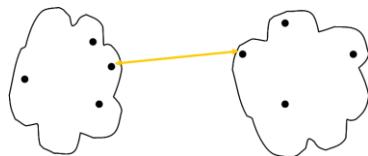
Two main types of hierarchical clustering

- Agglomerative: bottom-up, from n to 1 group
 - Start with as many clusters as there are objects
 - At each step, merge the closest pair of clusters into a single cluster
 - The chosen pair is formed by the groups that are more similar
 - Until only one cluster (or k clusters) left
- Divisive: top-down (less used), from 1 to n groups
 - Start with a single, all-inclusive, cluster
 - At each step, split a cluster into two
 - The selected cluster is the one with smallest uniformity
 - Until each cluster contains only one object (or there are k clusters)

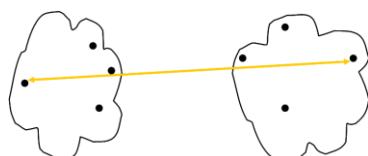
Traditional hierarchical algorithms use a proximity matrix

- Merge or split one cluster at a time

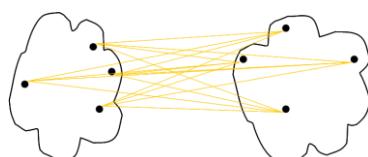
Hierarchical clustering: Inter-cluster proximity



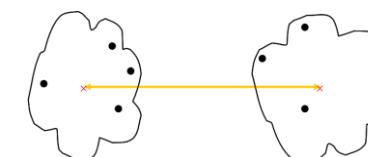
- Single Linkage (MIN): $d(C_1, C_2) = \min_{x_i \in C_1, x_j \in C_2} d(x_i, x_j)$
 - dissimilarity between the two most similar objects of the two clusters



- Complete Linkage (MAX): $d(C_1, C_2) = \max_{x_i \in C_1, x_j \in C_2} d(x_i, x_j)$
 - dissimilarity between the two most dissimilar objects of the two clusters



- Average Linkage: $d(C_1, C_2) = \frac{1}{n_1 n_2} \sum_{x_i \in C_1, x_j \in C_2} d(x_i, x_j)$
 - dissimilarity between all pairs of objects of the two clusters



- Distance between centroids
- Ward's method: takes into account the number of objects of the clusters

Hierarchical clustering: agglomerative methods

Algorithm:

- Compute the proximity matrix: matrix with value of proximity measure between pairs of points
- Let it each data point be a cluster
- **Repeat**
 - Merge the two closest clusters
 - Update the proximity matrix to reflect the proximity between the new cluster and original clusters
- **Until** only a single cluster remains

Hierarchical clustering: agglomerative methods

Example: consider the following distance matrix

- Use Agglomerative Hierarchical Clustering to obtain the single-link dendrogram

	A	B	C	D	E	F
A	0					
B	4	0				
C	25	21	0			
D	24	20	1	0		
E	9	5	16	15	0	
F	7	3	18	17	2	0

Distance Matrix - Stage 0



	A	B	CD	E	F
A	0				
B	4	0			
CD	24	20	0		
E	9	5	15	0	
F	7	3	17	2	0

Distance Matrix - Stage 1

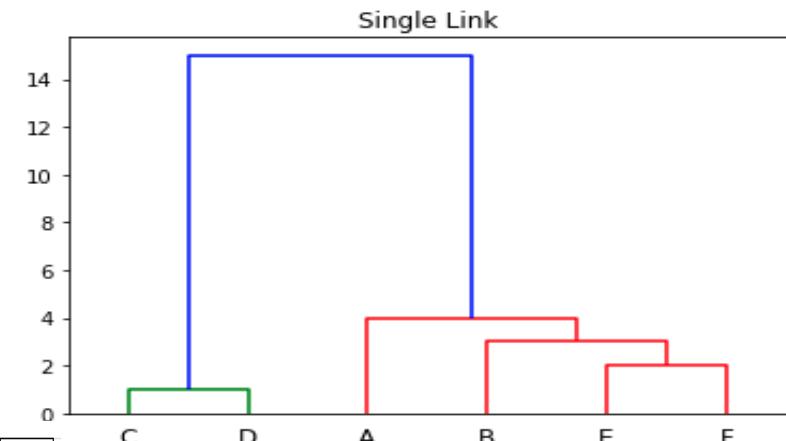
- New cluster with two "objects"
- Updating the proximity matrix
 - the distance to the remaining is the shortest distance from any member i

Single linkage:

$$\min(d_{i,c}, d_{i,p})$$

	A	B	CD	EF
A	0			
B	4	0		
CD	24	20	0	
EF	7	3	15	0

Distance Matrix - Stage 2



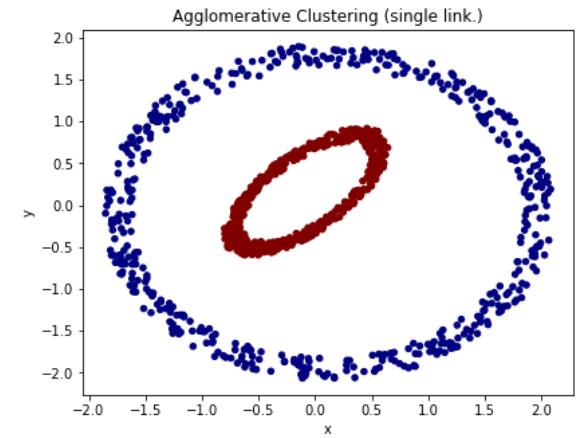
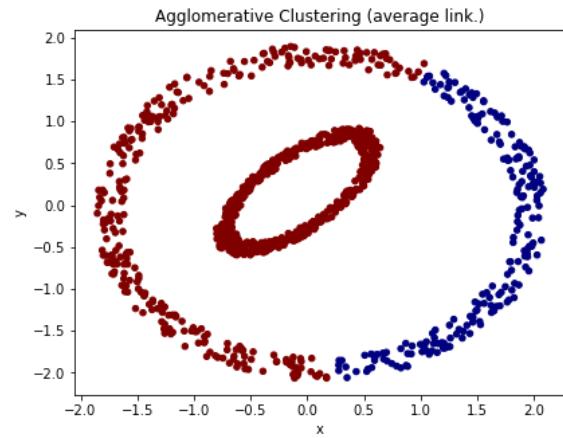
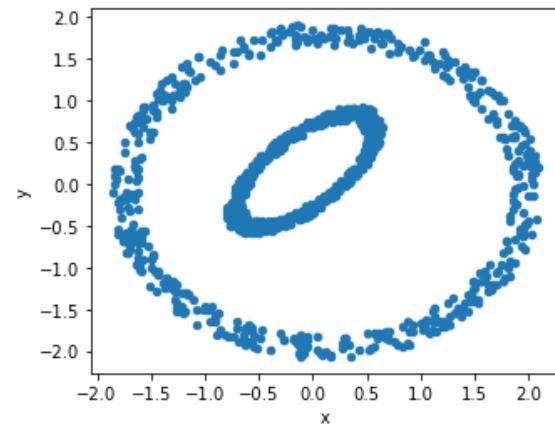
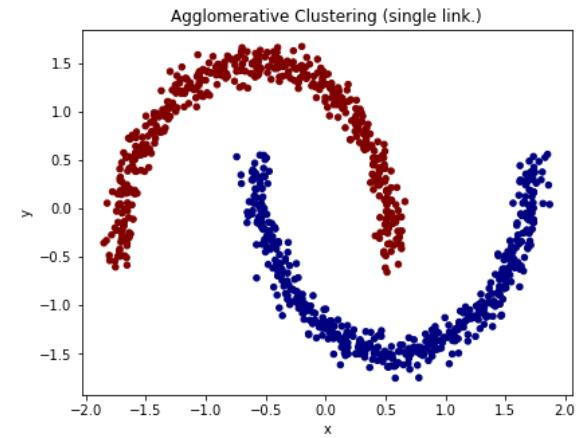
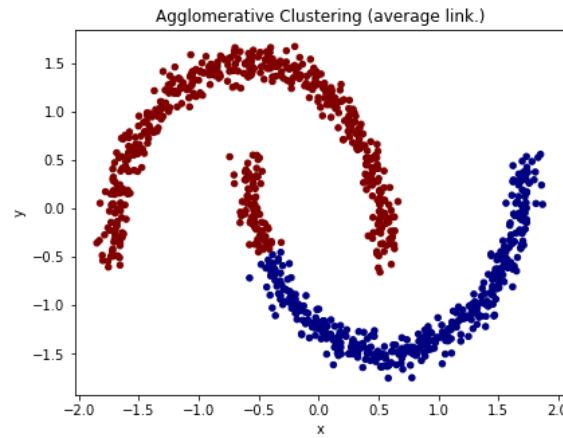
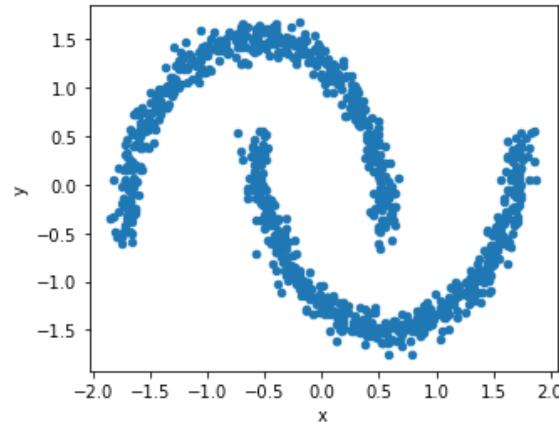
	A	BEF	CD
A	0		
BEF	4	0	
CD	24	15	0

Distance Matrix - Stage 3

	ABEF	CD
ABEF	0	
CD	15	0

Distance Matrix - Stage 4

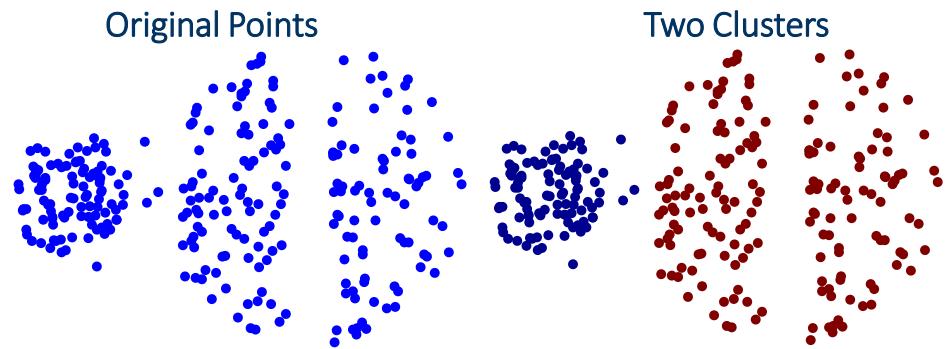
Hierarchical clustering: agglomerative methods



Hierarchical clustering

Different inter-cluster proximity schemes yield to different types of clusters

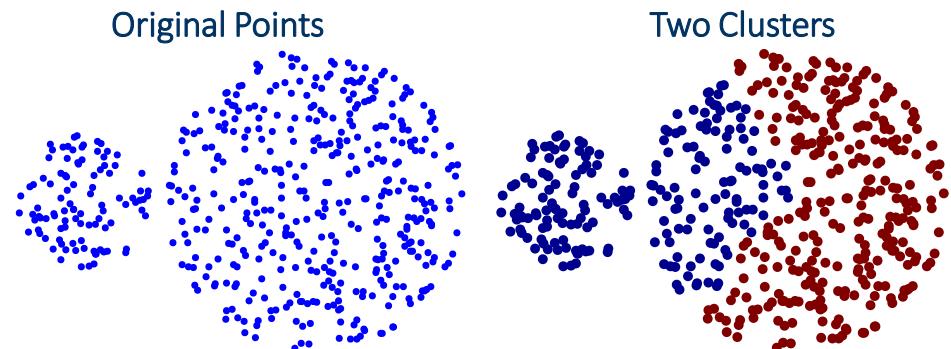
- **Single-linkage**
 - follows chains on the data
 - can handle non-elliptical shapes
 - uses a local merge criterion
 - distant parts of the cluster and the clusters' overall structure are not taken into account
 - sensitive to noise



Hierarchical clustering

Different inter-cluster proximity schemes yield to different types of clusters

- **Complete-linkage**
 - Tends to break large groups in data
 - biased towards globular clusters
 - uses a non-local merge criterion
 - chooses the pair of clusters whose merge has the smallest diameter
 - the similarity of two clusters is the similarity of their most dissimilar members
 - sensitive to outliers
 - less susceptible to noise



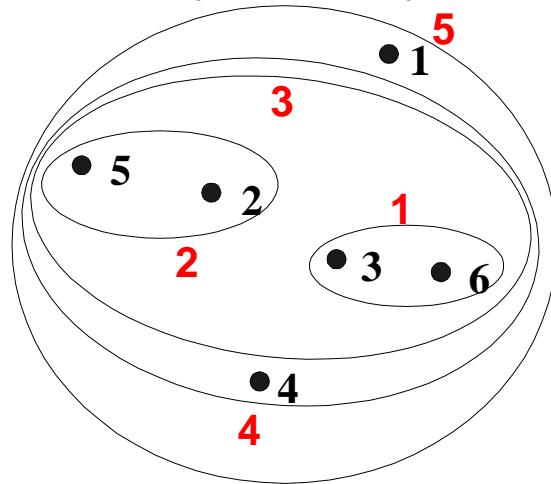
Hierarchical clustering

Different inter-cluster proximity schemes yield to different types of clusters

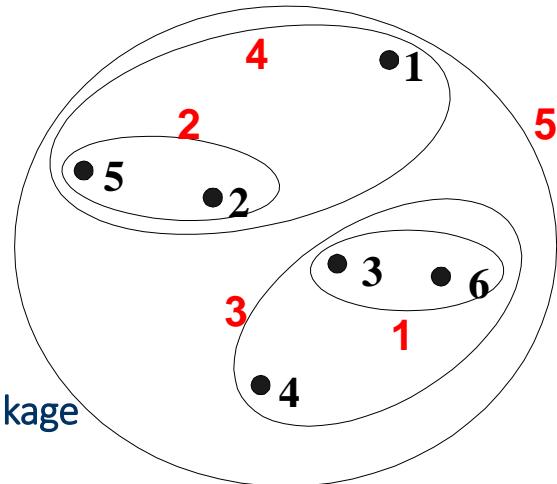
- **Average-linkage**
 - it is a compromise between **single** and **complete linkage**
 - biased towards globular clusters
 - less susceptible to noise
- **Complete** and **Average linkage**
 - lead to compact clusters

Hierarchical clustering: problems and limitations

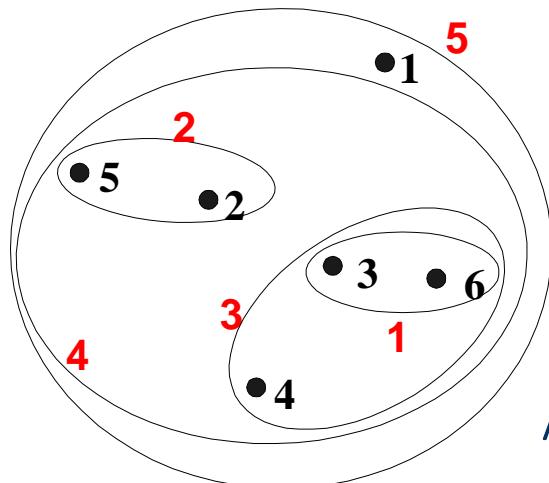
Different proximity measures yield to different types of clusters



Single-linkage

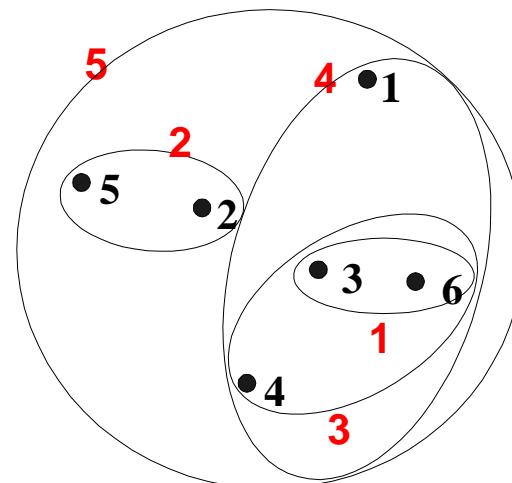


Complete-linkage



Average linkage

Ward's Method



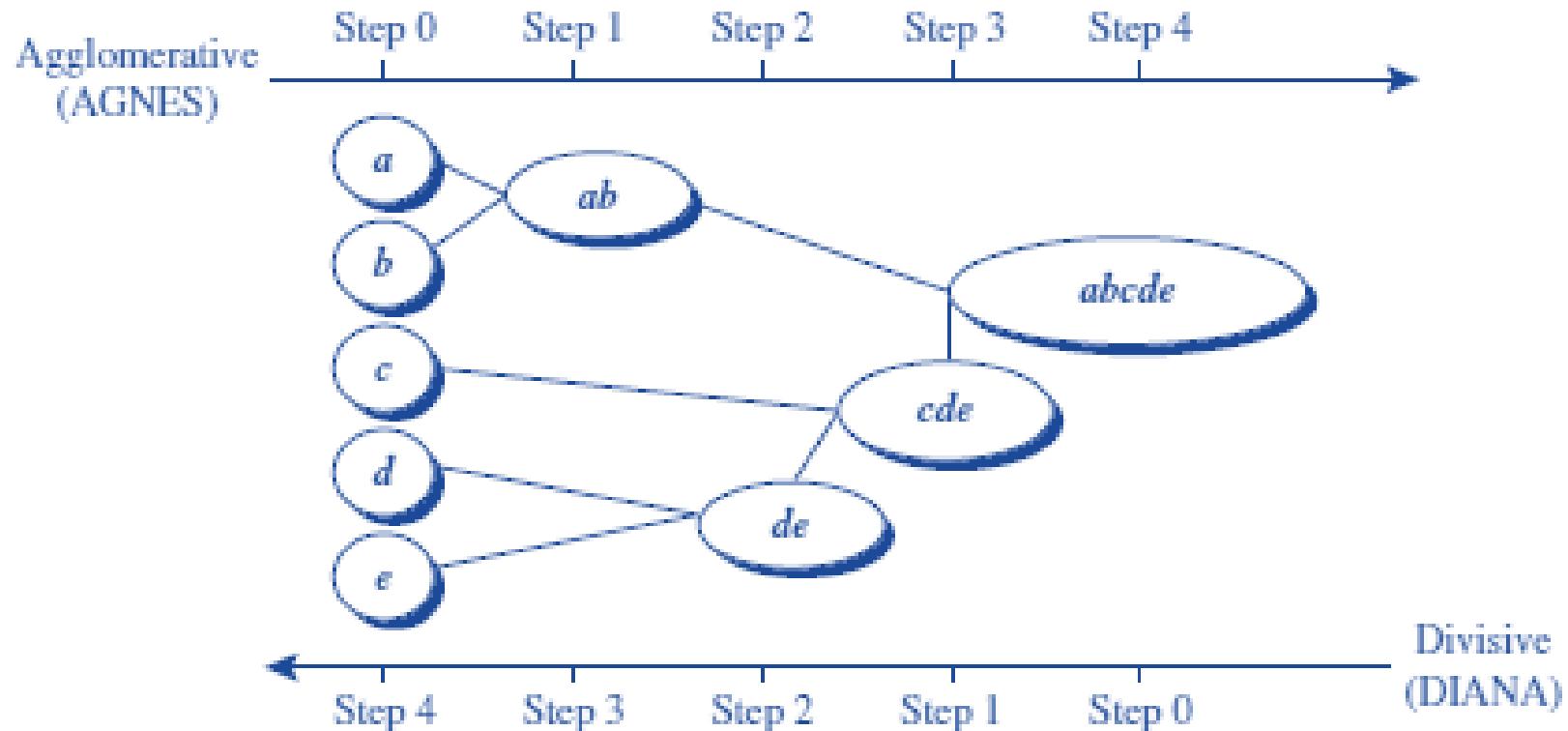
Hierarchical clustering: divisive methods

Algorithm:

- Compute the proximity matrix: matrix with value of proximity measure between pairs of points
- Start with a single cluster that contains all data points
- **Repeat**
 - choose a cluster based on a pre-defined criterion
 - use flat-algorithm A^1 to split the cluster into L clusters
- **Until** each data point constitutes a cluster

¹ algorithm A can be any arbitrary clustering algorithm, and not just a distance-based one

Hierarchical clustering



Hierarchical clustering: remarks and constraints

- Time and space requirements:
 - **Storage:** $O(n^2)$ space, n is the number of points
 - **Time:** $O(n^3)$, there are n steps and at each step the proximity matrix (with size n^2) must be updated and searched
- Once a decision is made to **merge two clusters** or **divide one cluster**, it **cannot be undone**
- **No objective function** is directly minimized
- **Different schemes have problems** with one or more of the following:
 - Sensitivity to noise
 - Difficulty handling clusters of different sizes and non-globular shapes
 - Breaking large clusters

Contents

- Descriptive analytics
- Cluster analysis
- Main categories of clustering methods
- Clustering validation
- Summary

Clustering validation

How to validate/evaluate/compare the results obtained by some clustering method?

1. Clustering tendency

- Assess the suitability of clustering, i.e., whether the data has any inherent grouping structure
 - Is the found grouping structure random?

2. Clustering stability

- Understand the sensitivity of the clustering result to various algorithm parameters, e.g., # of clusters
 - What is the “correct” number of clusters?

3. Clustering evaluation

- Evaluating the goodness of clustering results

Clustering validation: types of evaluation measures

3. Clustering evaluation: evaluating the goodness of clustering results

- Evaluating the entire clustering or just individual clusters

3.1. Unsupervised

How to evaluate the result of a clustering algorithm without external information?

3.2. Supervised

How to compare the results obtained by different methods when external information exists (such as class labels)?

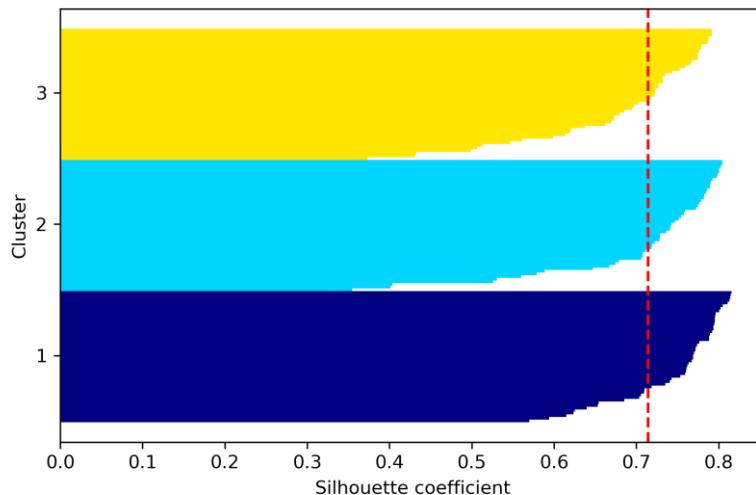
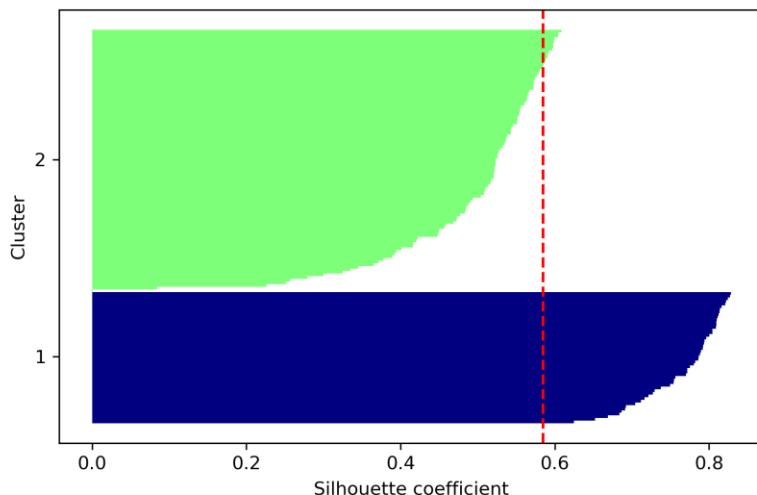
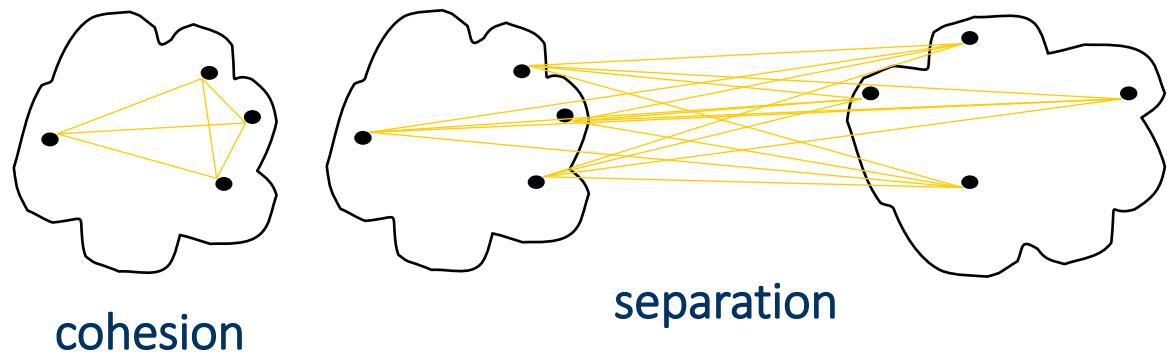
3.3 Relative (supervised or unsupervised)

How to compare clustering or clusters?

Clustering validation: unsupervised

Measure the quality of the clustering without any external information

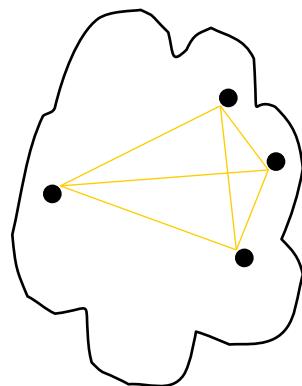
- Cluster Cohesion
- Cluster Separation
- Silhouette coefficient



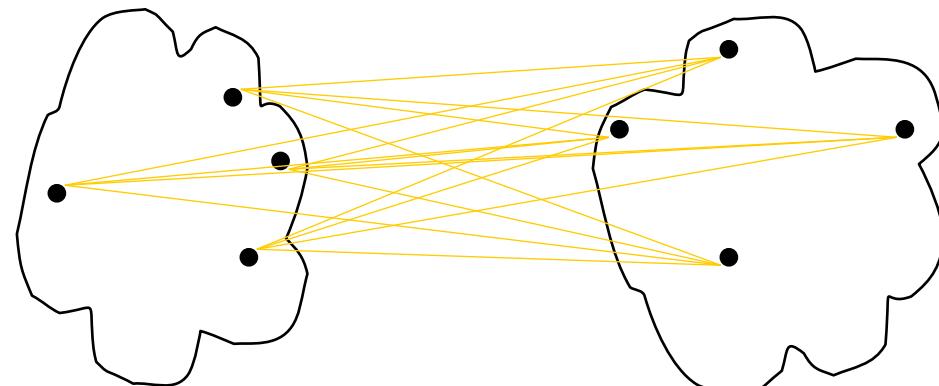
Clustering validation: unsupervised

Cohesion and separation

- **Cluster Cohesion:** Measures how cohesive/close/compact are the elements in a cluster (Intra-cluster distances)
- **Cluster Separation:** Measure how distinct or well-separated a cluster is from other clusters (Inter-cluster distances)



cohesion



separation

Clustering validation: unsupervised

Cohesion and separation

Example: Squared error ($SSW + SSB$ is constant)

- Cluster Cohesion: is measured by the within cluster sum of squares

$$SSW = \sum_{i=1}^n \sum_{x \in C_j} (x_i - m_j)^2$$

- Cluster Separation: is measured by the between cluster sum of squares

$$SSB = \sum_{j=1}^K |C_j| (m - m_j)^2$$

where m is the mean of all points, m_j is the center/centroid of cluster C_j , and $|C_j|$ is the size of cluster C_j

Clustering validation: unsupervised

Example: Cohesion and separation

- K=2 clusters



- Centroids: $m_1 = 15.25, m_2 = 25$
- Mean of all points: $m = 19.43$
- $SSW = 70.75, SSB = 162.97, SST = 233.71^*$

- K=3 clusters



- Centroids: $m_1 = 12, m_2 = 19.3, m_3 = 27$
- Mean of all points: $m = 19.43$
- $SSW = 8.7, SSB = 225.05, SST = 233.71^*$

The case $K=3$ is better:

- higher cohesion (SSW is lower) & higher separation (SSB is higher)

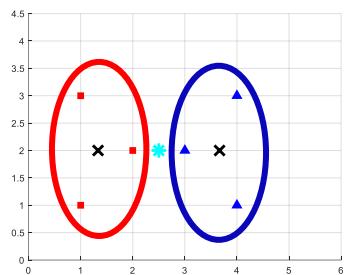
* the sum $SST = SSW + SSB$ is constant

Obj.	K=2	K=3
11	1	1
13	1	1
18	1	2
19	1	2
21	2	2
26	2	3
28	2	3

Clustering validation: unsupervised

Example: Cohesion and separation

- K=2 clusters

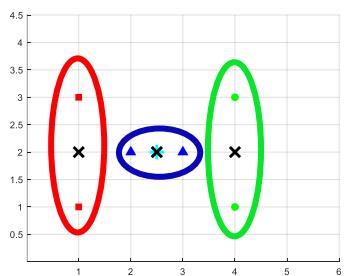


Centroids: $m_1 = (4/3, 2)$, $m_2 = (11/3, 2)$

Mean of all points: $m = (2.5, 2)$

$$SSW = 5.333, \quad SSB = 8.16, \quad SST = 13.5^*$$

- K=3 clusters



Centroids: $m_1 = (1, 2)$, $m_2 = (2.5, 2)$, $m_3 = (4, 2)$

Mean of all points: $m = (2.5, 2)$

$$SSW = 4.5, \quad SSB = 9, \quad SST = 13.5^*$$

The case $K=3$ is better:

- higher cohesion (SSW is lower) & higher separation (SSB is higher)

* the sum $SST = SSW + SSB$ is constant

Clustering validation: unsupervised

Silhouette coefficient

Silhouette coefficient: check cluster cohesion and separation

- For each object x_i :
 - compute a_i = average distance of x_i to the points in its cluster
 - compute b_i = min (average distance of x_i to the points in other clusters)
 - the silhouette coefficient is $s_i = (b_i - a_i) / \max(a_i, b_i)$
- Silhouette coefficient (SC) is the mean values of s_i across all the objects:

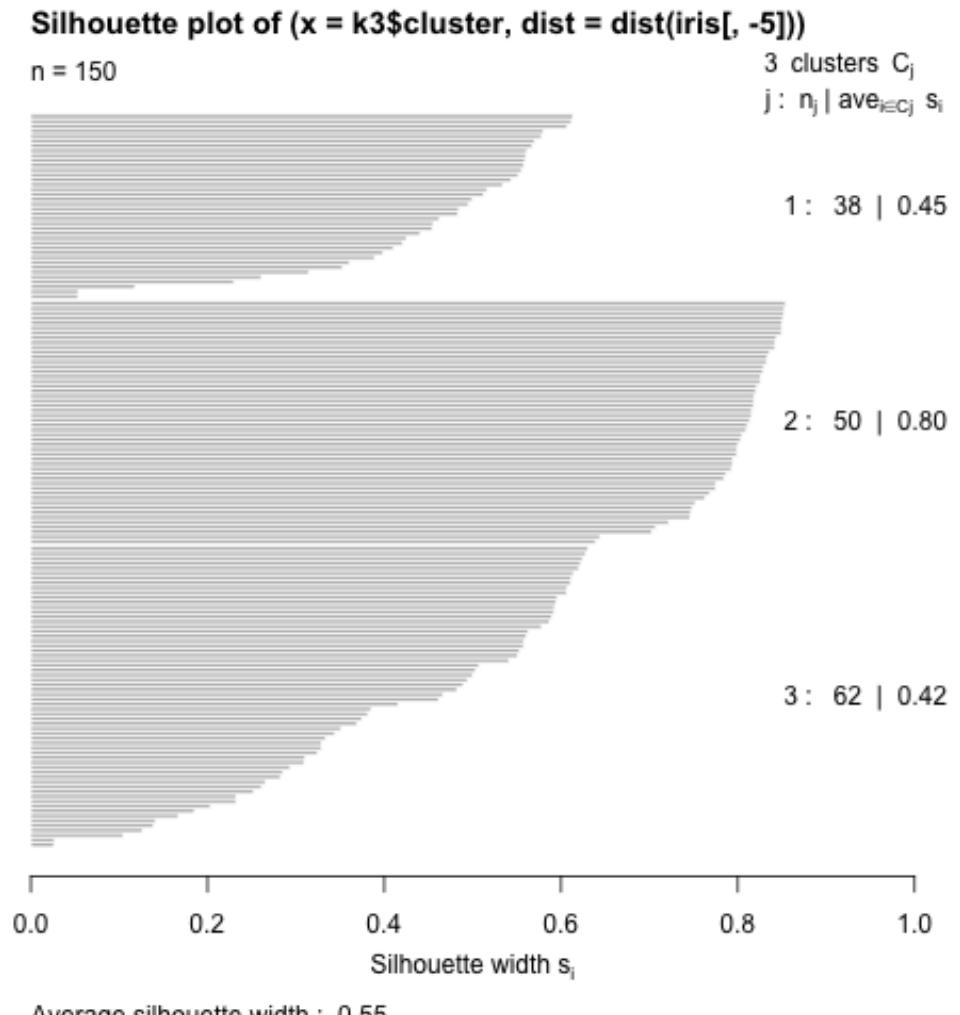
$$SC = \frac{1}{n} \sum_{i=1}^n s_i$$

- SC can vary between -1 and 1
 - close to +1 signifies good clustering: objects are close to their own clusters but far from other clusters

Clustering validation: unsupervised Silhouette coefficient

Example: iris data set silhouette coefficients s_i with $k = 3$ clusters

- Large s_i (almost 1) means that the object is very well clustered
- Small s_i (around 0) means that the object lies between two clusters
- Negative s_i means that the object is probably placed in the wrong cluster

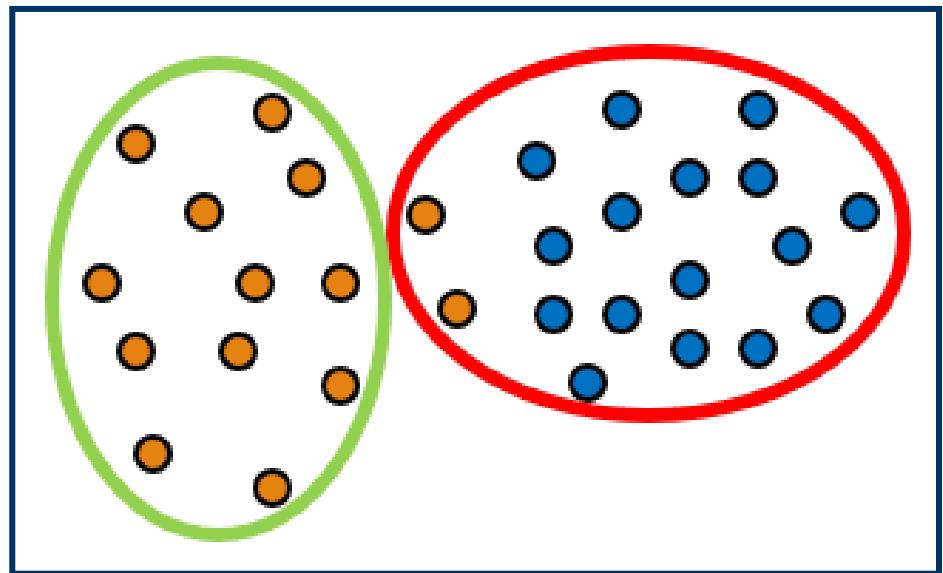


The closer SC to 1, the better

Clustering validation: supervised

Compare the results obtained by different methods when external information exists

- Pairwise measures
- Matching-based measures
- Entropy-Based Measures
- Correlation measures



$class_1$

$class_2$

Cluster C_1

Cluster C_2

Clustering validation: supervised

Pairwise measures

- f_{00} = number of pairs of objects having a different class and a different cluster
- f_{01} = number of pairs of objects having a different class and the same cluster
- f_{10} = number of pairs of objects having the same class and a different cluster
- f_{11} = number of pairs of objects having the same class and the same cluster

Two-way contingency table for determining whether pairs of objects are in the same class and same cluster

	Same cluster	Different cluster
Same class	f_{11}	f_{10}
Different class	f_{01}	f_{00}

Clustering validation: supervised

Matching-based measures

- **Precision:** the fraction of a cluster that consists of objects of a specified class
- **Recall:** the extent to which a cluster contains all objects of a specified class
- **F-measure:** A combination of both precision and recall that measures the extent to which a cluster contains *only* objects of a particular class and *all* objects of that class

Clustering validation: best number of clusters

How to select the right K for k-means?

- An inappropriate choice of K can result in a clustering with poor performance
- What happens when selecting a K that is too high? When the K is too low?

Ideally: some a priori knowledge on the real structure of the data

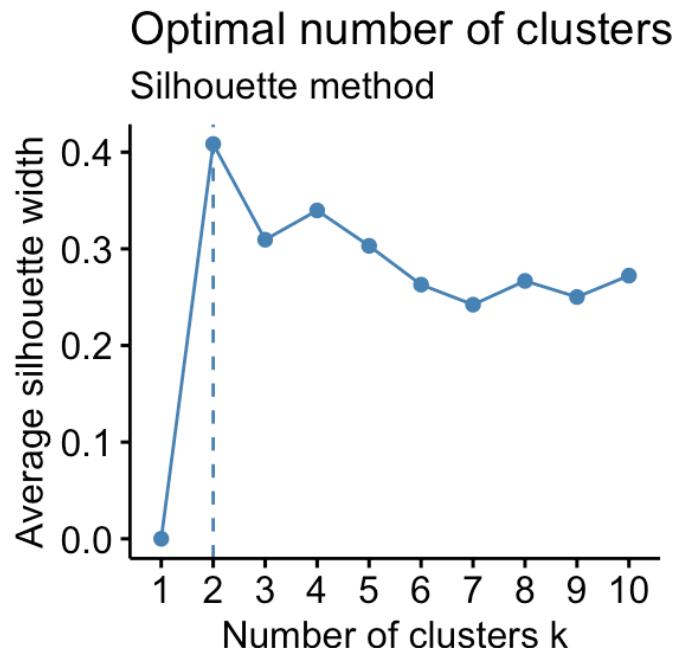
- If no a priori value is known start with $K = \sqrt{n/2}$ as a rule of thumb, where n is the number of attributes.

Clustering validation: best number of clusters

Silhouette coefficient method

For several possible number of clusters K :

- Calculate the SC and choose the K that yields to the highest value

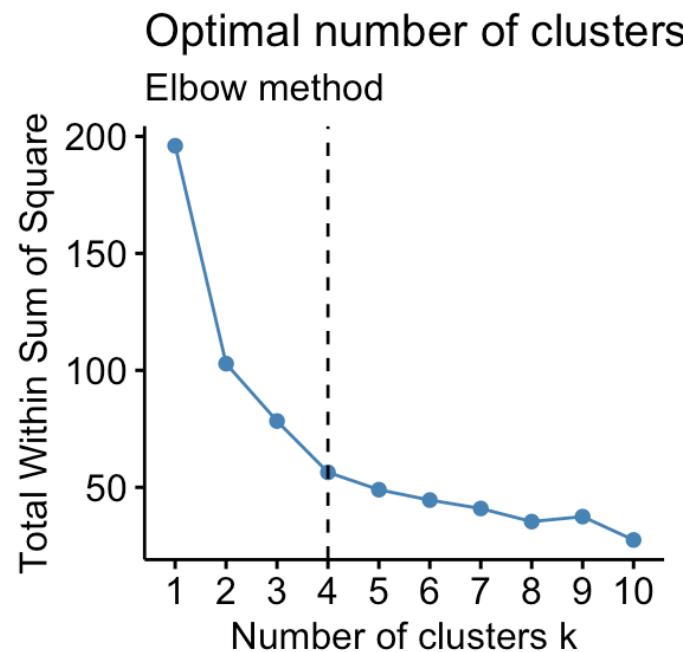


Clustering validation: best number of clusters

Elbow method

For several possible number of clusters K :

- Calculate the **SSE** (Sum of Squared Error), also called distortion, and choose the K so that adding another cluster doesn't yield to a much smaller SSE.



Clustering validation: tendency

Assess if the data has any inherent grouping structure

- Cluster the data set
 - Use multiple algorithms and evaluate the quality of the resulting clusters
 - If the clusters are uniformly poor, then this may indeed indicate that there are no clusters in the data

Focus of measures of clustering tendency

- try to evaluate whether a data set has clusters without clustering
 - use statistical tests for spatial randomness
 - However, choosing the correct model, estimating the parameters, and evaluating the statistical significance of the hypothesis that the data is non-random can be quite challenging
 - many approaches have been developed, most of them for points in low-dimensional Euclidean space
 - Hopkins statistic

Clustering validation: tendency

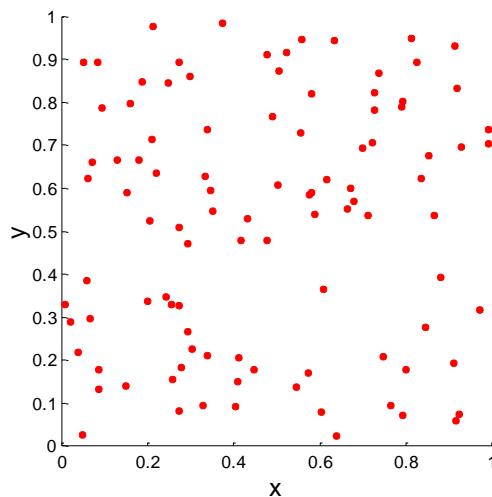
Hopkins statistic H

- generate p points randomly distributed across the data space
- sample p actual data points from the data set
- For both sets of points find the distance to the nearest neighbor in the original data set
 - ui - nearest neighbor distances of the artificially generated points
 - wi - nearest neighbor distances of the samples from the original data set

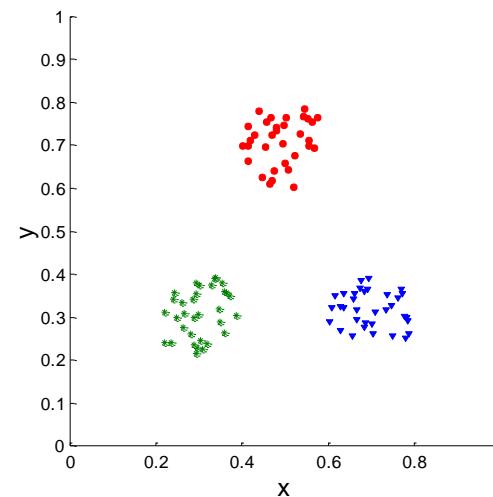
$$H = \frac{\sum_{i=1}^p w_i}{\sum_{i=1}^p u_i \sum_{i=1}^p w_i}$$

Clustering validation: Hopkins statistic H

- $H \sim 0.5$: randomly generated points and the samples of the data set have roughly the same nearest neighbor distances
- $H \sim 0$: whole data (random + sampled) is highly clustered
- $H \sim 1$: whole data is regularly distributed in the data space



$H: 0.56 \pm 0.03$
($p=20, p=100$)



$H: 0.95 \pm 0.006$
($p=20, p=100$)

Contents

- Descriptive analytics
- Cluster analysis
- Main categories of clustering methods
- Clustering validation
- **Summary**

Summary

- Descriptive analytics
- Cluster analysis
- Main categories of clustering methods
 - Partitional
 - Representative based
 - Density based
 - Hierarchical
 - Agglomerative
 - Divisive
- Clustering validation

What is good clustering?

A good clustering method will produce high quality clusters which should have

- **High intra-class similarity:** Cohesive within clusters
- **Low inter-class similarity:** Distinctive between clusters

Clustering methods: comparison

Overall, we can compare clustering methods w.r.t

- Algorithm:
 - complexity and scalability
 - proximity measures that can be employed
 - robustness to noise
 - it is able to find clusters on sub-spaces
 - different runs lead to different results
 - it is incremental

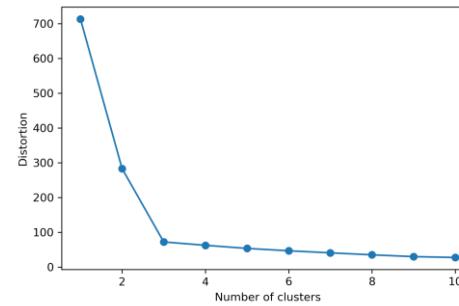
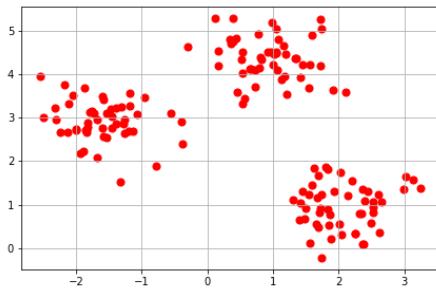
Clustering methods: comparison

- Data:
 - it is able to handle different types of data (continuous, categorical, binary)?
 - is there dependency on the order of data points?
- Domain:
 - does the algorithm finds the number of clusters, or needs it as input?
 - how many parameters are necessary?
 - what is the required domain knowledge for that?
- Results:
 - shape of clusters that is able to find
 - interpretability

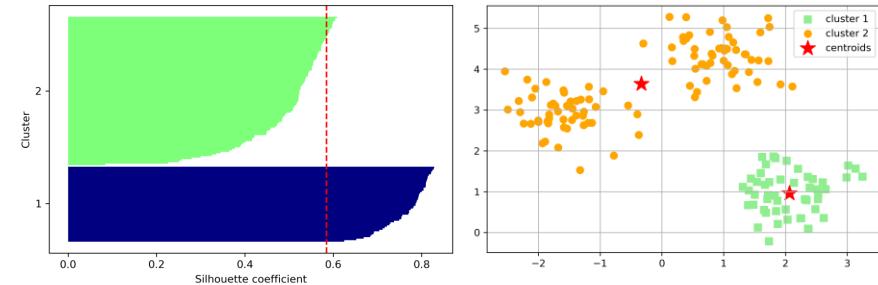
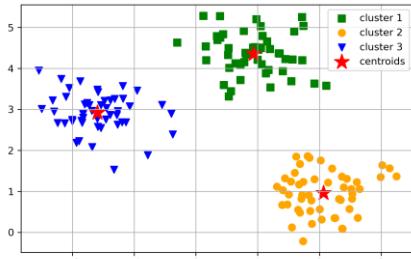
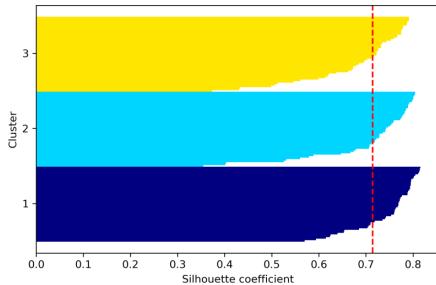
Clustering: toy example

Two versus three clusters

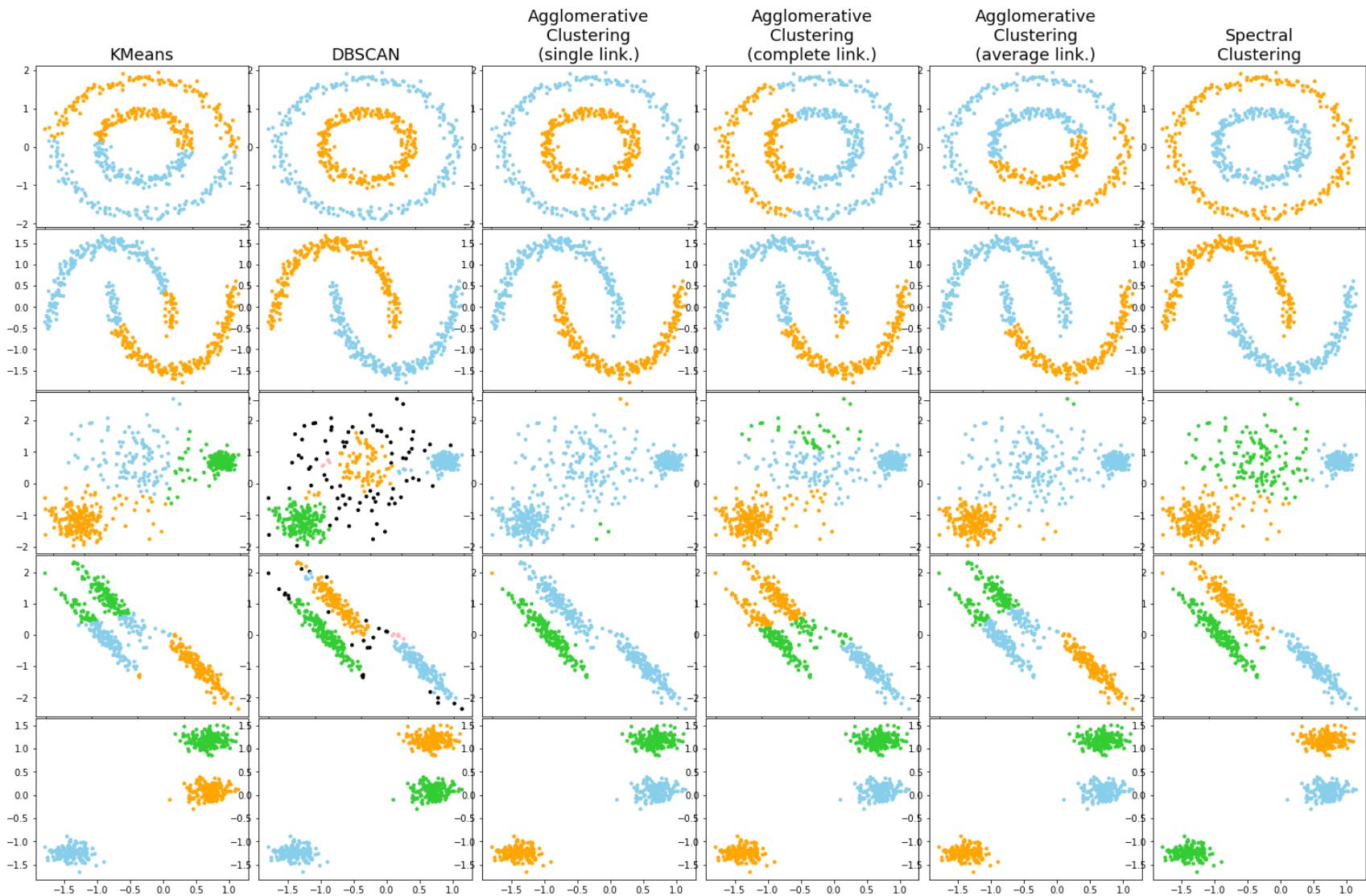
- The **optimal** number of clusters according Elbow method is three



- The **silhouette** values and clusters



Toy exemple: K-mean, DBSCAN, Hierarchical



Bibliography

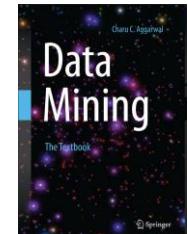
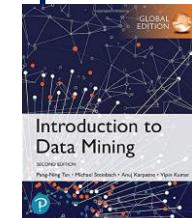
Introduction to Data Mining, Pang-Ning Tan, Michael Steinbach, Anuj Karpatne, Vipin Kumar, *Pearson*, 2019 (chap 5)

Data Mining, the Textbook, Charu C. Aggarwal, *Springer*, 2015 (chap 6)
Xindong Wu, et al.. Top 10 algorithms in data mining. *Knowl Inf Syst*, 14:1-37, 2008

R. Ng and J. Han. **Efficient and Effective Clustering Method for Spatial Data Mining**. *VLDB'94*, 1994

B. Schölkopf, A. Smola, and K. R. Müller. **Nonlinear Component Analysis as a Kernel Eigenvalue Problem**. *Neural computation*, 10(5):1299–1319, 1998

I. S. Dhillon, Y. Guan, and B. Kulis. **Kernel K-Means: Spectral Clustering and Normalized Cuts**. *KDD'04*, 2004



Data Mining

Predictive Modelling

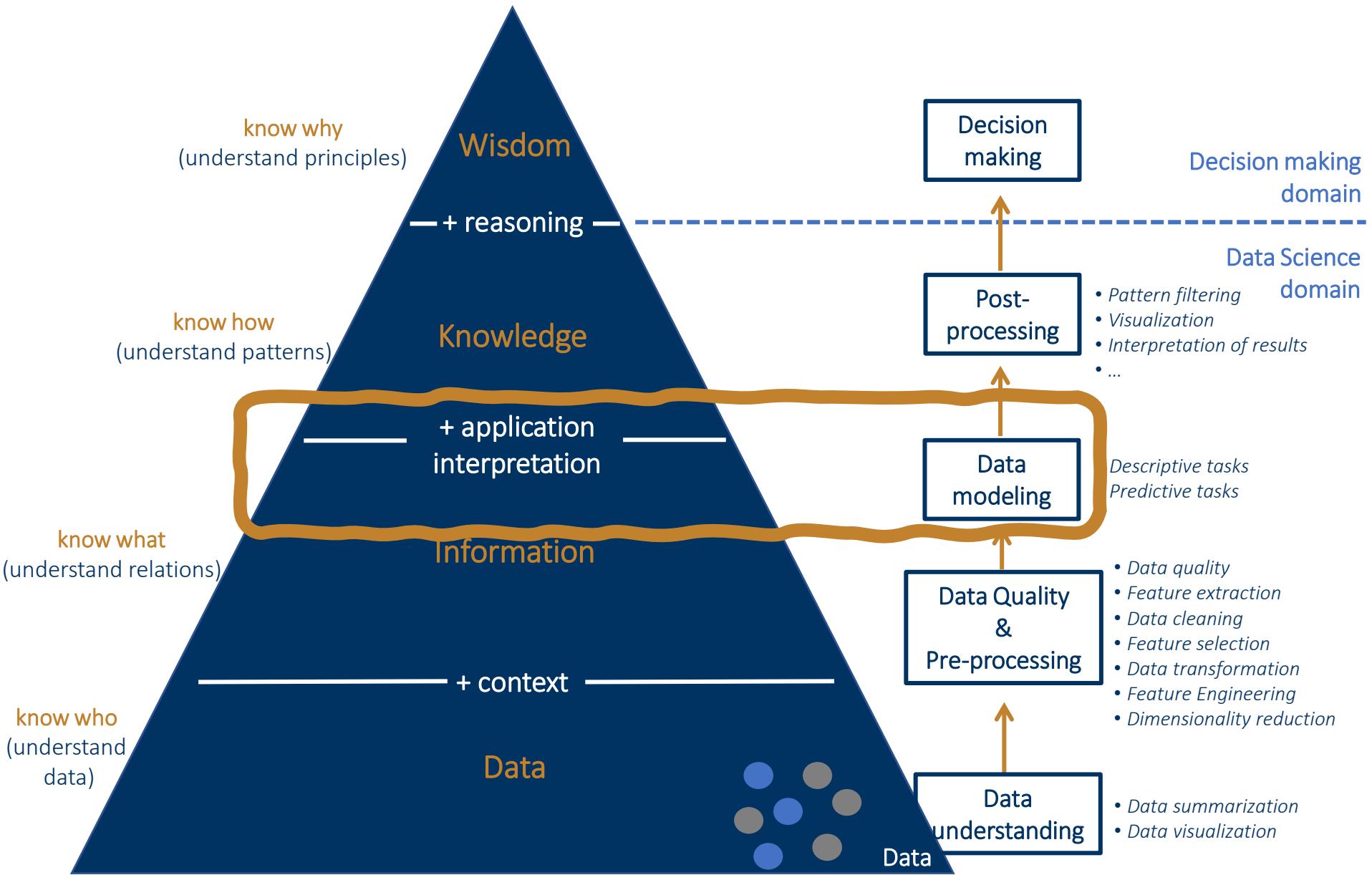
Raquel Sebastião

Departamento de Eletrónica, Telecomunicações e Informática

Universidade de Aveiro

raquel.sebastiao@ua.pt

2022/2023



Contents

- Machine Learning
- Predictive Modelling
- Classification
- Summary

Machine Learning

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T , as measured by P , improves with experience E .”

Mitchell, T. (1997)

“Machine Learning is the systematic study of algorithms and systems that improve their knowledge or performance with experience”

Flach, P. (2012)

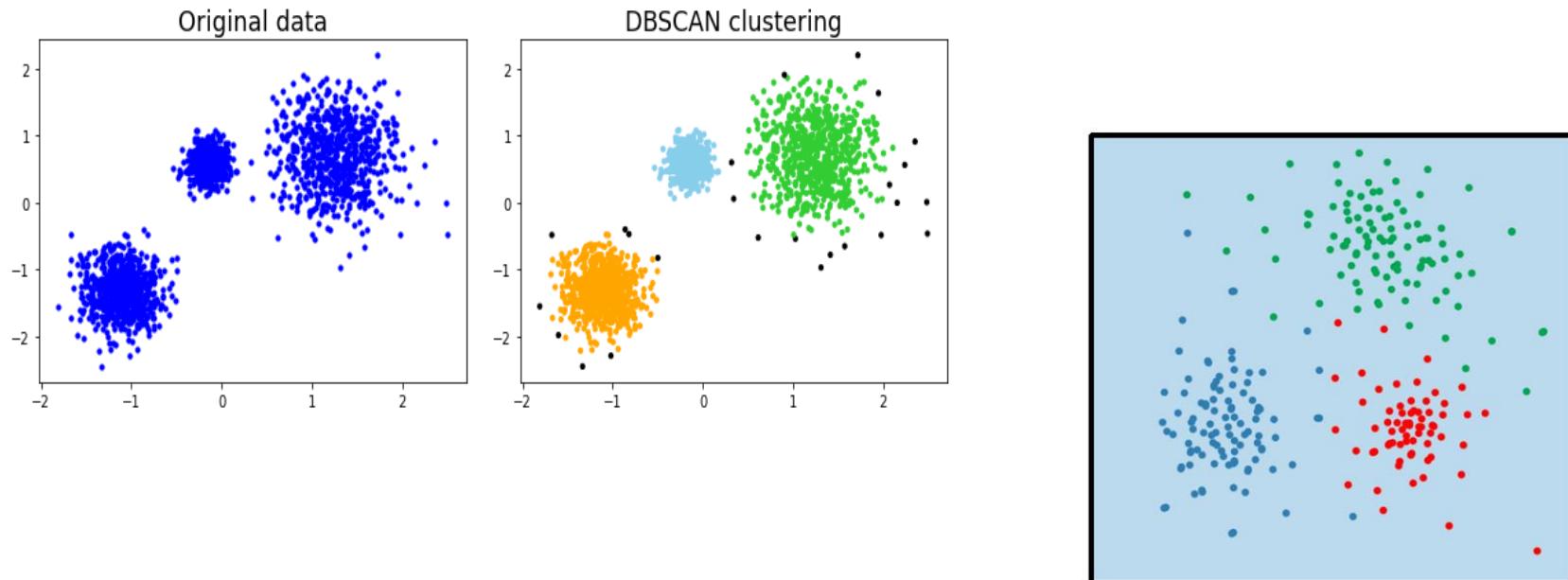
Goal:

- Build models that capture the knowledge from observed cases to make inferences in unseen cases. In principle, more observations should lead to better models!

Machine Learning: Tasks

Unsupervised Learning: no target label/value is associated to each object (class labels of the training data are unknown)

- Given a set of observations/objects, the goal of learning is to obtain possible groups/clusters in the data (structure of the data)

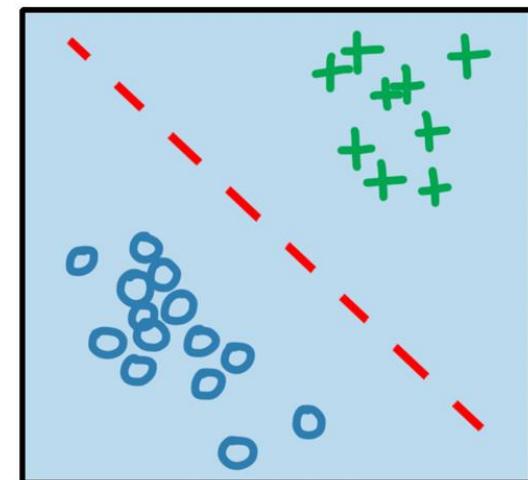
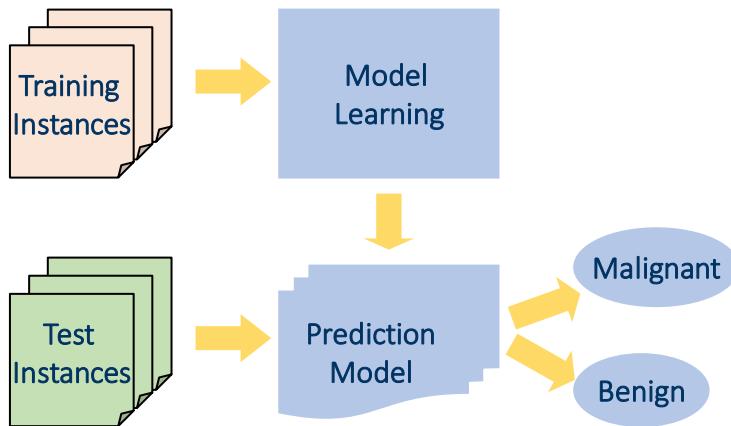


Machine Learning: Tasks

Supervised Learning: there is a target label/value that is associated to each object/example

- the goal of the learning task is to learn a function (model) that maps each example with its target variable

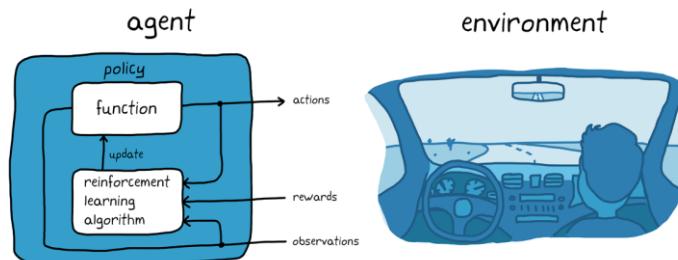
→ **Predictive Modelling:** New data is classified based on the models built from the training set



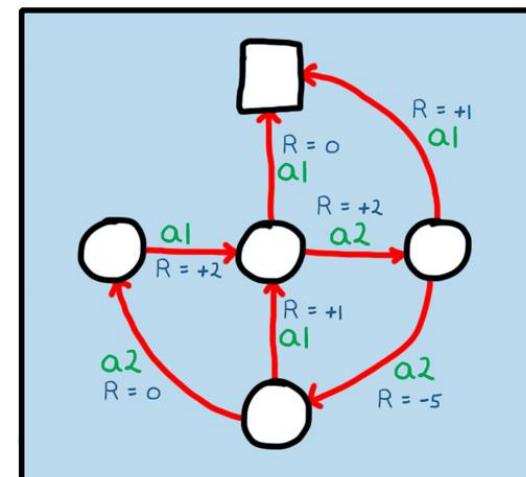
Machine Learning: Tasks

Reinforcement Learning: the learning algorithm builds examples from a set of rules; then an iterative process is used to improve (or “reinforce”) the set of examples until some evaluation criterion is good enough.

- **example:** parking a vehicle using an automated driving system: teach the vehicle computer (agent) to park in the correct parking spot with reinforcement learning



<https://www.mathworks.com/>



Machine Learning

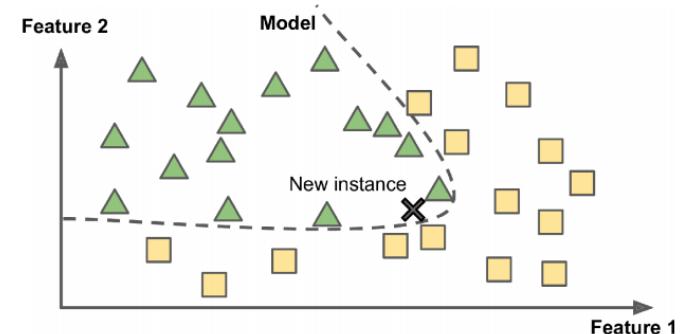
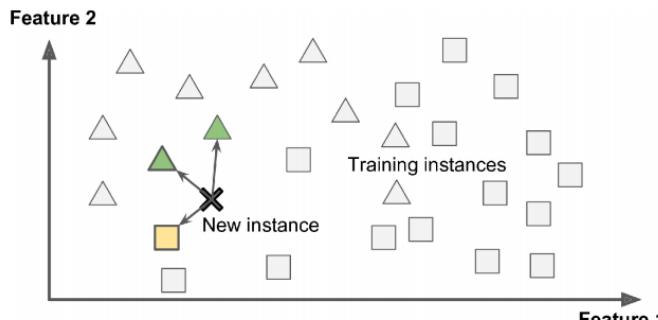
What are the main learning paradigms?

- Batch learning
- Online learning

Is there an assumption on data distribution?

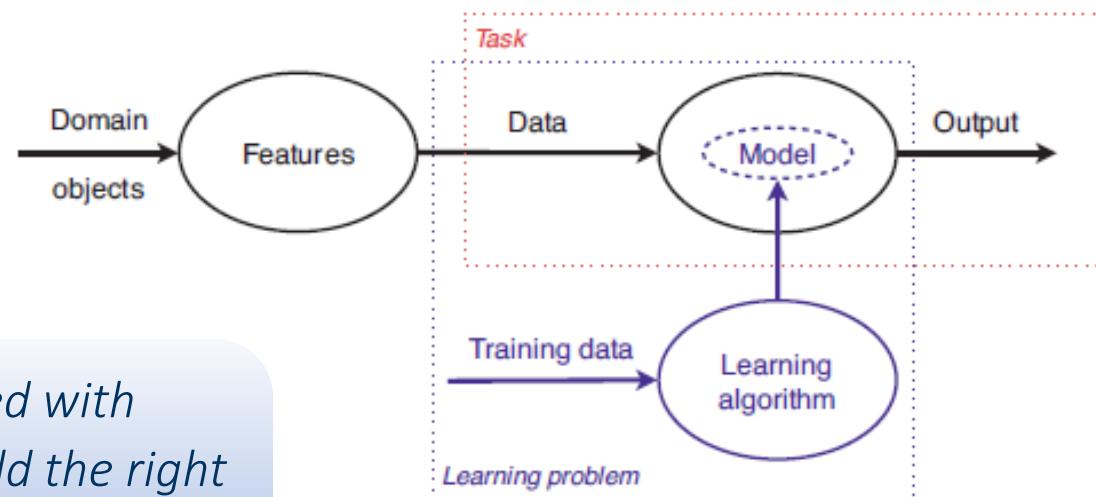
- Parametric
- Non-parametric

What to do when new data points arrive?



Machine Learning

- Tasks are addressed by models
- Learning problems are solved by learning algorithms
- Learning algorithms produce models (when applied to training data)



“machine learning is concerned with using the right features to build the right models that achieve the right tasks”

Flach, P. (2012)

Contents

- Machine Learning
- Predictive Modelling
 - Pipeline
 - Tasks
 - Prediction models – approaches
- Classification
- Summary

Predictive Modelling

Example: Clinical diagnosis

- Given a data set containing diverse features extracted from X-rays or MRI scans of several patients and the diagnosis

ID	r-mean	t-mean	per-mean	ar-mean	sm-mean	cm-mean	cn-mean	nc-mean	(...)	diagnosis
842302	17.99	10.38	122.8	1001	0.1184	0.2776	0.3001	0.1471	(...)	M
842517	20.57	17.77	132.9	1326	0.08474	0.07864	0.0869	0.07017	(...)	M
84300903	19.69	21.25	130	1203	0.1096	0.1599	0.1974	0.1279	(...)	M
84348301	11.42	20.38	77.58	386.1	0.1425	0.2839	0.2414	0.1052	(...)	M
84358402	20.29	14.34	135.1	1297	0.1003	0.1328	0.198	0.1043	(...)	M
843786	12.45	15.7	82.57	477.1	0.1278	0.17	0.1578	0.08089	(...)	M
844359	18.25	19.98	119.6	1040	0.09463	0.109	0.1127	0.074	(...)	M
84458202	13.71	20.83	90.2	577.9	0.1189	0.1645	0.09366	0.05985	(...)	M
844981	13	21.82	87.5	519.8	0.1273	0.1932	0.1859	0.09353	(...)	M
84501001	12.46	24.04	83.97	475.9	0.1186	0.2396	0.2273	0.08543	(...)	B
845636	16.02	23.24	102.7	797.8	0.08206	0.06669	0.03299	0.03323	(...)	B
84610002	15.78	17.89	103.6	781	0.0971	0.1292	0.09954	0.06606	(...)	B
846226	19.17	24.8	132.4	1123	0.0974	0.2458	0.2065	0.1118	(...)	M
846381	15.85	23.95	103.7	782.7	0.08401	0.1002	0.09938	0.05364	(...)	M
84667401	13.73	22.61	93.6	578.3	0.1131	0.2293	0.2128	0.08025	(...)	M
84799002	14.54	27.54	96.73	658.8	0.1139	0.1595	0.1639	0.07364	(...)	M
848406	14.68	20.13	94.74	684.5	0.09867	0.072	0.07395	0.05259	(...)	M
84862001	16.13	20.68	108.1	798.8	0.117	0.2022	0.1722	0.1028	(...)	M

- Predict the correct diagnosis for a new patient for which we know the features of X-ray and MRI scans

Predictive Modelling

Prediction Models are learnt on the basis of the assumption that there is an **unknown mechanism** that **maps the characteristics/features** of the observations into **conclusions**

- The goal of prediction models is to discover this mechanism

Clinical diagnosis

- how features/characteristics of the cells in the X-rays or MRI scans influence the diagnosis
- Use a data set with “examples” of this mapping, e.g., this patient had cells’ characteristics c₁, c₂, ..., c_D and the diagnosis was that tumor cells were benign
- Using the **available data**, obtain a **good approximation** of the unknown **function** that maps the **observation descriptors** into the **conclusions**

Predictive Modelling

- **Descriptors:** set of variables that describe the properties (features, attributes, predictors) of the objects in the data set (X_1, X_2, \dots, X_D)
- **Target variable:** what we want to predict/conclude regarding the observations (Y)
- The **goal** is to obtain an approximation of the function

$$Y = f(X_1, X_2, \dots, X_D)$$

- It is assumed that Y is a variable whose values depend on the values of the variables which describe the objects.

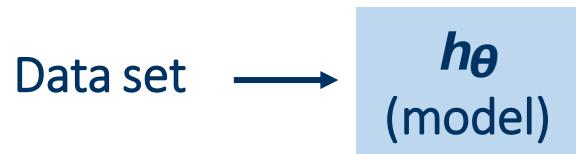
We just do not know how!

Predictive Modelling

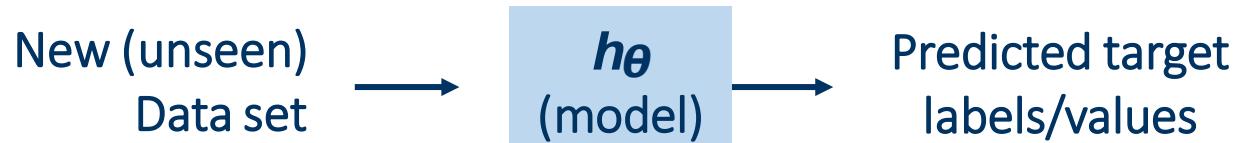
- Given a set of predictor variables X and a target variable Y , there is a function f , such that $f(X) = Y$



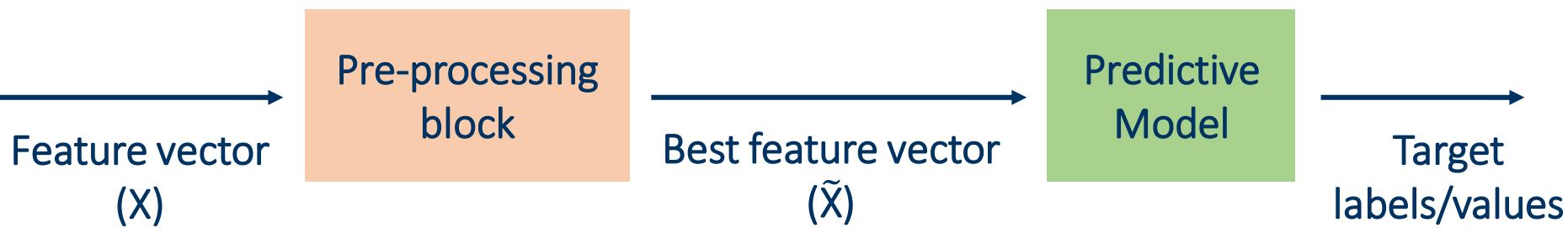
- Since f is unknown, the goal is to learn the best approximation to f , h_{θ} , so that the target labels/values can be obtained from the input data set



- With the built model h_{θ} , it is possible to make predictions for new, unseen observations!



Predictive Modelling: pipeline



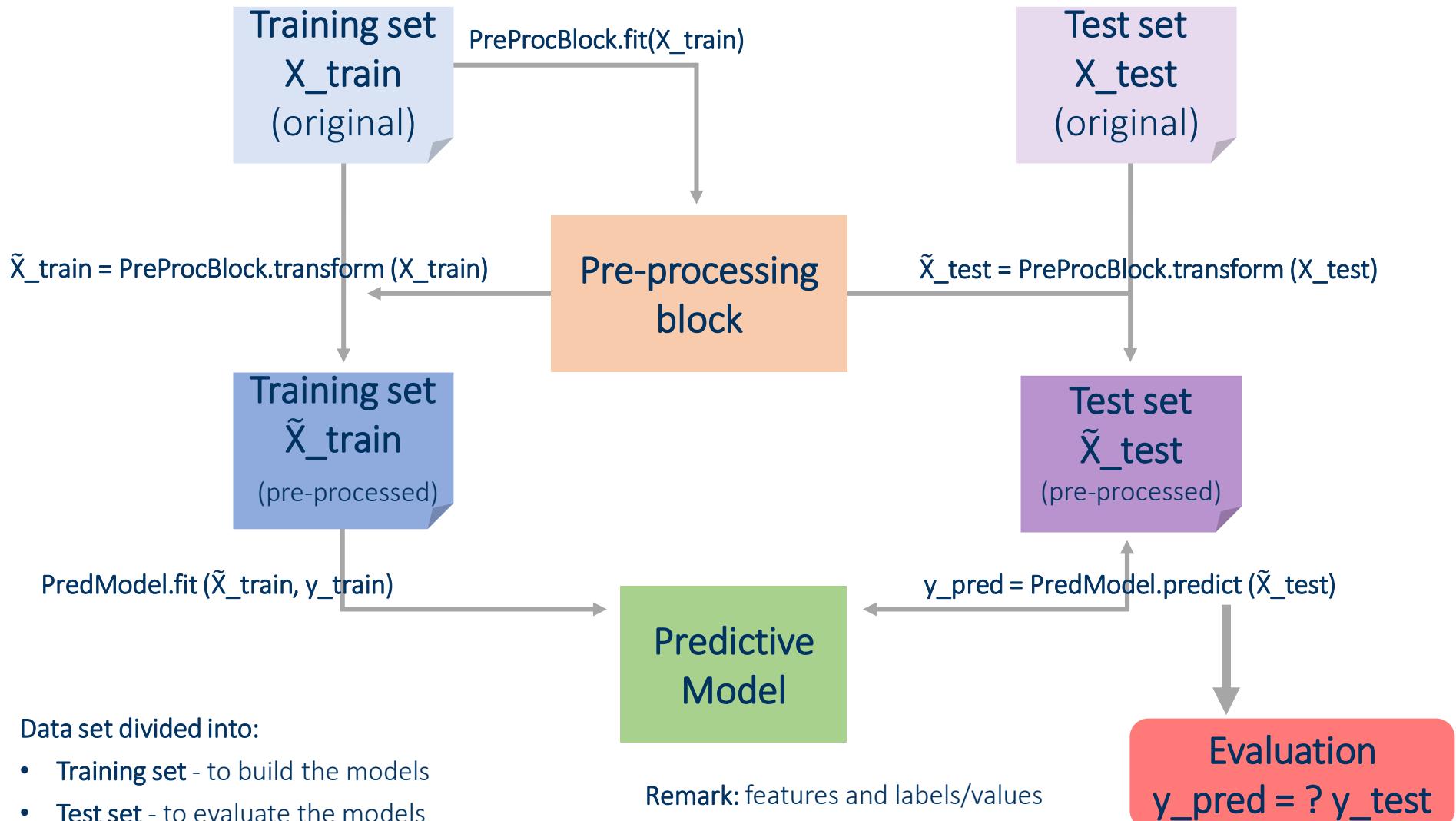
Pre-processing block

- To deal with heterogenous data
 - numerical and categorical
- To deal with different features' ranges
 - Normalization/standardization
- Feature Engineering
- Data reduction
 - Feature selection
 - Projection of feature vector

Predictive model

- Classification model
 - Target labels
- Regression model
 - Target values

Predictive Modelling: pipeline

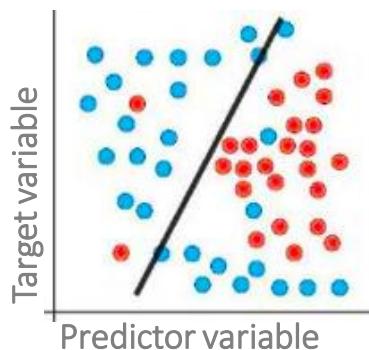


Predictive Modelling

Underfitting: model is too simple to capture patterns in data

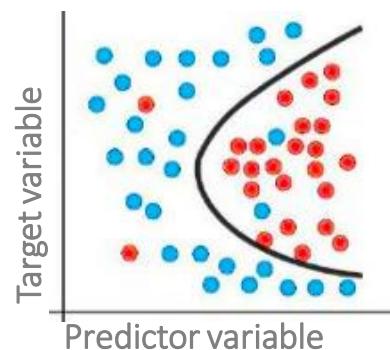
Overfitting: model performs well on training data but does not generalize well to unseen data

Underfitting
(high bias)



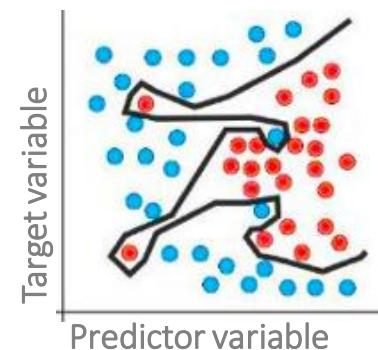
- High training error
- High test error

Optimal
(good compromise)



- Low training error
- Low test error

Overfitting
(high variance)



- Low training error
- High test error

Predictive Modelling

- Predictive models have two main uses:

1. Prediction:

- use the obtained models to make predictions regards the target variable of new cases given their descriptors.

2. Comprehensibility:

- use the models to better understand which are the factors that influence the conclusions.

Predictive Modelling - tasks

Types of Prediction tasks

Depending on the type of the target variable Y we may be facing two different types of prediction models:

- **Classification tasks**
 - the target variable Y is nominal, e.g., medical diagnosis - given the symptoms of a patient try to predict the diagnosis
- **Regression tasks**
 - the target variable Y is numeric e.g., forecast the market value of a certain asset given its characteristics

Prediction Models

There are many approaches that can be used to obtain **prediction models** based on a data set

Independently of the **pros** and **cons** of each alternative, all have some key characteristics:

- They assume a certain **functional form** for the unknown function $f()$
- Given this assumed form, the methods try to obtain the best possible model based on:
 - the given **data set**
 - a certain **preference criterion** that allows comparing the different alternative model variants

Prediction Models – approaches

Geometric approaches

- Distance-based: kNN
- Linear models: linear discriminants, logistic regression, perceptron, SVM (w. linear kernel)

Probabilistic approaches

- naive Bayes, logistic regression

Logical approaches

- classification or regression trees, rules

Optimization approaches

- neural networks, SVM

Sets of models (ensembles)

- random forests, adaBoost

Prediction Models – approaches

Or...

Linear approaches

- linear discriminants, logistic regression, perceptron, SVM (w. linear kernel)

Non-linear approaches

- kNN, naive Bayes, classification trees, (w. non-linear kernel) SVM, neural networks

Sets of models (ensembles)

- random forests, adaBoost

Prediction Models

These different approaches entail **different compromises** in terms of:

- **assumptions** on the unknown form of dependency between the target and the predictors
- **computational complexity** of the search problem
- **flexibility** in terms of being able to approximate **different types of functions**
- **interpretability** of the resulting model
- ...

Contents

- Machine Learning
- Predictive Modelling
- Classification
 - Problem definition
 - Binary and multiclass classification
 - Evaluation metrics
- Summary

Classification: problem definition

Setting

- Given a **data set** $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$, where each **object** is represented by a **D+1–tuple**:
(D-dim) feature vector $\mathbf{x}_i = [x_i^1 \ x_i^2 \ \dots x_i^D]^T \in \mathbb{R}^D$ and the corresponding **label** $y_i \in \mathcal{Y}$
- There is an **unknown** function: $\mathcal{Y} = f(\mathcal{X})$

Goal

Learn the model that yields the best approximation of the unknown function $f()$

Approach

- Assume a functional form $h_{\boldsymbol{\theta}}(\mathbf{x})$ for the unknown function $f()$, where $\boldsymbol{\theta}$ are a set of parameters
- Assume a preference criterion over the space $\boldsymbol{\theta}$ of possible parameterizations of $h()$
- Search for the “best” $h_{\boldsymbol{\theta}}$ (according to the criterion and the data set)

Classification: learn/build a classifier

Given a training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$:

- $(\mathbf{x}_i, \omega_{c_i})$ - the label is **symbolic** or **categorical** (ex.: binary {malignant, benign})
- (\mathbf{x}_i, t_i) - label is **numerical** (ex.: binary {-1, 1}, multiclass {0, 1, 2}))

Learning/Training phase

- find the “best” approximation to f, h_{θ}

Testing phase: Given a test set (data not included in the training set)

- Study the accuracy of the model (performance of the classifier).

Usually, the available data set is divided into training and test sets

Classification: binary classification problem

- when the target variable only assumes **two possible labels/values** (classes), usually referred as positive and negative class
 - e.g., flu: yes/no, credit: yes/no

Output of a classification model:

- class assigned to a case
- score / probability of case belonging to a certain class; a decision threshold is chosen to establish the predicted class
- e.g., if $h_{\theta}(x_i) \geq 0.5$ then is positive example, otherwise is negative

Classification: multiclass classification problem

- when the target variable assumes more than two possible classes
 - e.g., insurance risk: low, medium, high

Some algorithms **cannot handle multiclass**; the alternative is to combine several binary classifiers

- **one-vs-all:** train a model for each class; for k classes, we have k binary classifiers
- **one-vs-one:** train a model for each pair of classes; for k classes, we have $k(k - 1)/2$ classifiers

Classification: performance of models

Metrics for evaluating a model's performance

- How can we measure the performance of a model?
 - Accuracy, precision, recall, F-measure

Use test set (unseen data) of class-labeled tuples instead of training set when assessing performance

Strategies for estimating a model's performance

- How can we evaluate the performance of a model?
 - Holdout method, Cross-validation, Bootstrap method

Comparing models:

- Statistical Hypothesis Testing

Classification: evaluation metrics

Goal: Obtain reliable estimates of performance and compare different classification models

- **Error Rate:** proportion of predictions that are incorrect

$$L_{0/1} = \frac{1}{N} \sum_{i=1}^N l(\hat{y}_i, y_i)$$

- N is the number of cases
- $\hat{y}_i = h_\theta(x_i)$ is the predicted class by the model for the object i
- y_i is the respective true class
- $l()$ is loss-function such that $l(\hat{y}_i, y_i) = 0$, if $\hat{y}_i = y_i$, and 1 otherwise
- **Accuracy** = 1 - Error Rate

Classification: evaluation metrics

- Confusion matrices

- A square $c \times c$ matrix*, where c is the number of class values of the problem
- A special kind of contingency table, with two dimensions (“true class” and “predicted class”)
- Each value reports the number of predictions made by the model of a class for a given true class

		<i>Predicted class</i>		
		C_1	C_2	C_3
<i>True class</i>	C_1	$n_{c1, c1}$	$n_{c1, c2}$	$n_{c1, c3}$
	C_2	$n_{c2, c1}$	$n_{c2, c2}$	$n_{c2, c3}$
	C_3	$n_{c3, c1}$	$n_{c3, c2}$	$n_{c3, c3}$

- $n_{ci, cj}$ indicates # of objects in class i that were predicted by the model as class j

*(may have extra rows/columns to provide totals)

Classification: evaluation metrics

- Confusion matrix for a binary classification problem

		<i>Predicted class</i>	
		<i>P</i>	<i>N</i>
<i>True class</i>	<i>P</i>	TP True Positive	FN False Negative
	<i>N</i>	FP False Positive	TN True Negative

- TP: hit
- FN: miss (type II error)
- FP: false alarm (type I error)
- TN: correct rejection

Classification: evaluation metrics

- Confusion matrix for a binary classification problem

		<i>Predicted class</i>	
		<i>P</i>	<i>N</i>
<i>True class</i>	<i>P</i>	TP True Positive	FN False Negative
	<i>N</i>	FP False Positive	TN True Negative

- Accuracy = $\frac{TP+TN}{TP + FN + FP + TN}$ (proportion of correct predictions)
- Precision = $\frac{TP}{TP + FP}$ (proportion of correct positive predictions)
- Recall = $\frac{TP}{TP + FN}$ (proportion of positive objects correctly predicted)

Classification: evaluation metrics

- Confusion matrix for a binary classification problem

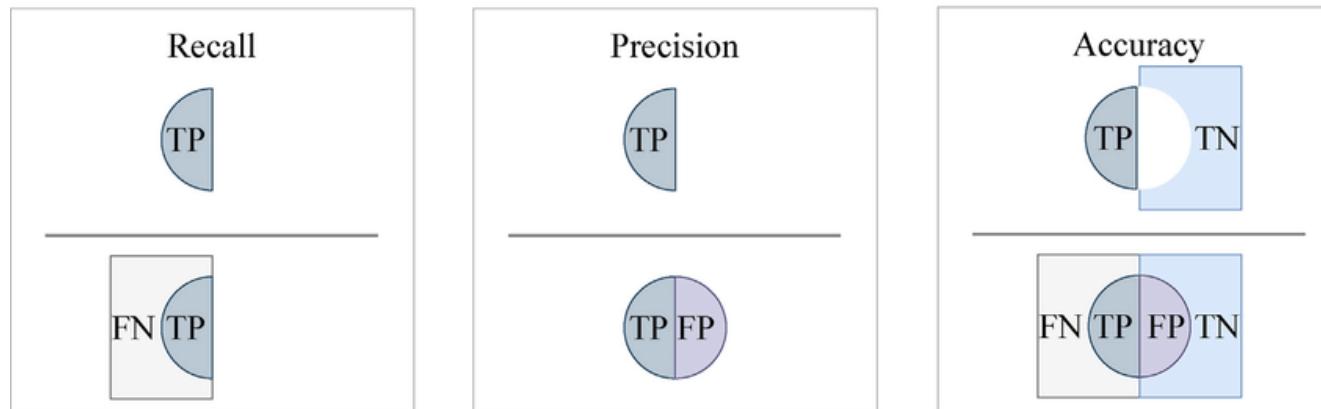
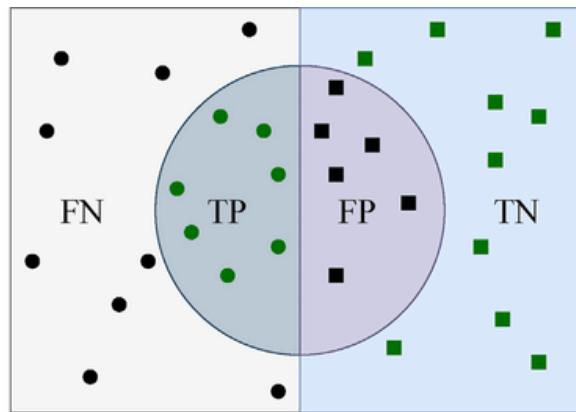
		<i>Predicted class</i>	
		<i>P</i>	<i>N</i>
<i>True class</i>	<i>P</i>	TP True Positive	FN False Negative
	<i>N</i>	FP False Positive	TN True Negative

Precision: A green dashed box highlights the True Positive (TP) and False Positive (FP) cells, representing the proportion of correct positive predictions.

Recall: A purple dashed box highlights the True Positive (TP) and False Negative (FN) cells, representing the proportion of positive objects correctly predicted.

- Precision:** proportion of correct positive predictions
- Recall:** proportion of positive objects correctly predicted

Classification: evaluation metrics



Maleki, F., et al. (2020) Overview of Machine Learning Part 1. *Neuroimaging Clinics of North America*. 30. e17-e32. 10.1016/j.nic.2020.08.007

Classification: evaluation metrics

- Precision/Recall tradeoff
 - increasing precision may reduce recall and vice versa.
- It is easy to obtain **100% Recall**: always predict **Positive**
- **F-measure**: weighted combination of Precision and Recall

$$F_{\beta} = \frac{1}{\alpha \cdot \frac{1}{Precision} + (1 - \alpha) \cdot \frac{1}{Recall}} = \frac{(\beta^2 + 1) \times Precision \times Recall}{\beta^2 Precision + Recall}$$

where β controls the weighted combination

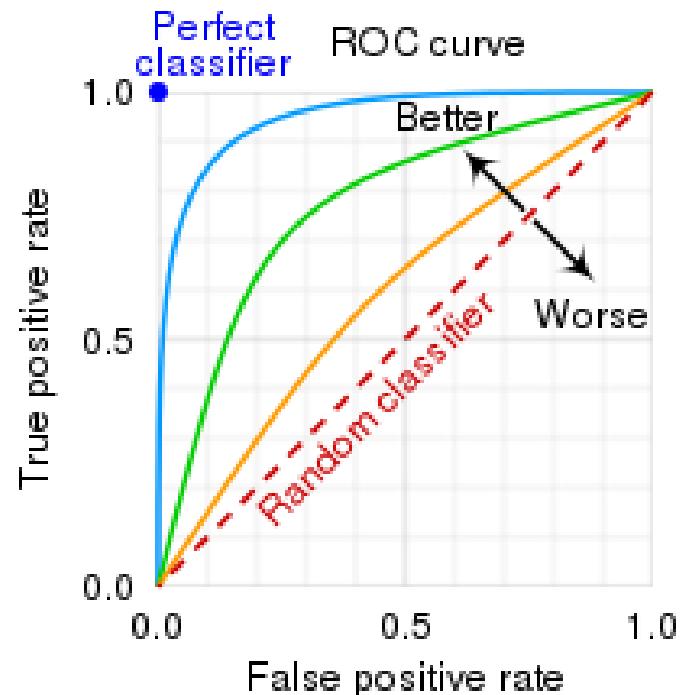
- if $\beta = 1$ then is the harmonic mean of **Precision** and **Recall**
- when $\beta \rightarrow 0$ the weight of **Recall** decreases.
- when $\beta \rightarrow \infty$ the weight of **Precision** decreases

Classification: evaluation metrics

- Some classifiers may require higher precision:
 - e.g., classifier that detects videos that are safer for kids. Keep high precision with only safe videos and may reject other videos that are good (low recall)
- Some classifiers may require higher recall:
 - e.g., classifier that detects disease on image samples. High recall to get all disease samples. Can handle some false positives (lower precision) that later will be double checked by doctors.
- There are several tradeoff measures: e.g., **G-mean**, IBA (Index of Balanced Accuracy)

Classification: evaluation metrics

- Receiver Operator Characteristic (ROC) curve is a common tool for evaluation of binary classifiers.
- Plots TPR (Recall) vs $FPR = FP/(TN + FP)$ for different decision thresholds.
- Area Under the Curve (AUC) allows to compare classifiers
 - A perfect classifier has AUC of 1
 - A random classifier has AUC of 0.5



en.wikipedia.org

Summary

- Machine Learning
- Predictive Modelling
 - Pipeline
 - Tasks
 - Prediction models - approaches
- Classification
 - Problem definition
 - Binary and multiclass classification
 - Evaluation metrics

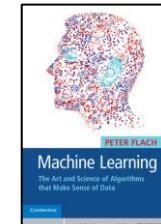
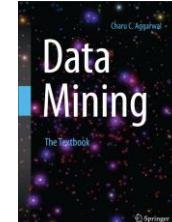
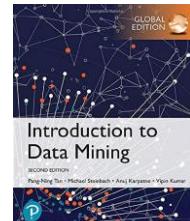
Bibliography

Introduction to Data Mining, Pang-Ning Tan, Michael Steinbach, Anuj Karpatne, Vipin Kumar, *Pearson*, 2019 (chap 3)

Data Mining, the Textbook, Charu C. Aggarwal, *Springer*, 2015 (chap 6)

Machine Learning: The Art and Science of Algorithms That Make Sense of Data, P. Flach, *Cambridge University Press*, 2012 (ch 1, 2, 11)

Data Mining: Practical Machine Learning Tools and Techniques, I. H. Witten, E. Frank, M. A. Hall, C. J. Pal, *Morgan Kaufmann*, 2017 (ch 3, 5)



Predictive Modelling

k-Nearest Neighbors (kNN)

Raquel Sebastião

Departamento de Electrónica, Telecomunicações, Informática
Universidade de Aveiro

raquel.sebastiao@ua.pt

2022/2023

k-Nearest Neighbors (kNN)

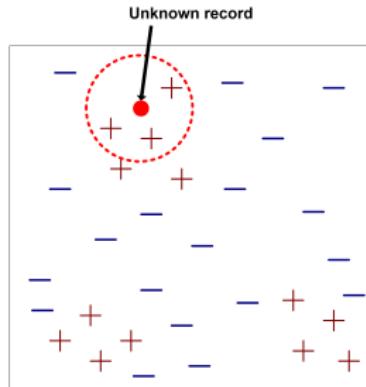
k-Nearest Neighbor belongs to the class of **instance-based** often known as "lazy learner"

- No algorithm to extract information from labeled data.
- The labeled data is stored to classify new data.
- It does not learn a function to map the predictor variables into a target variable
- it does not make any assumption on the unknown functional form we are trying to approximate, it means that with sufficient data they are applicable to any problem

k-Nearest Neighbors (kNN)

The decision about label of new data

- looking for the **most similar examples (neighbors)** within the stored data
- the **label** is decided according to the **label of neighbors**



The hyper-parameters of the classifier are: **k** and **criterium** to find the neighbors.

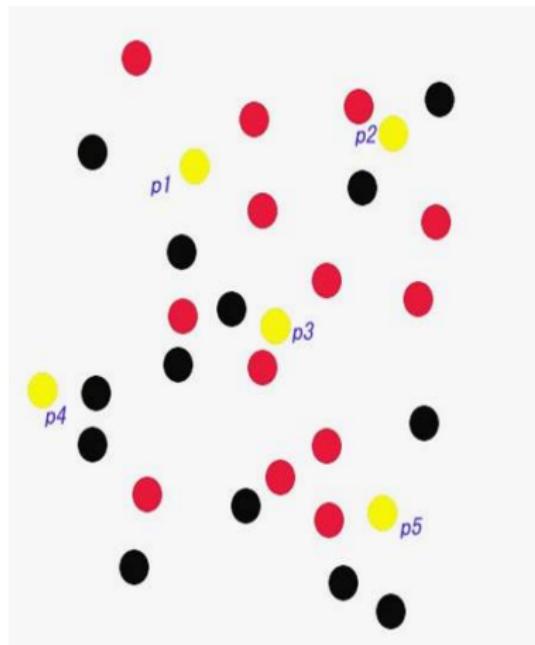
k-Nearest Neighbors (kNN)

Method:

- Choose the number **k** and the distance metric **d**
- For a test case **x**
 - find the **k** nearest cases in the training data according to **d**
 - use the target variable values of these cases to obtain the prediction for **x**
 - the prediction is the majority class

k-Nearest Neighbors (kNN): example

2 – D data set belonging to two classes



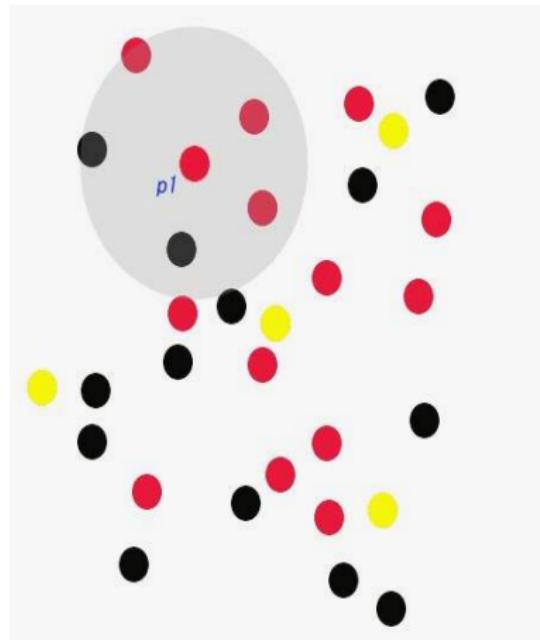
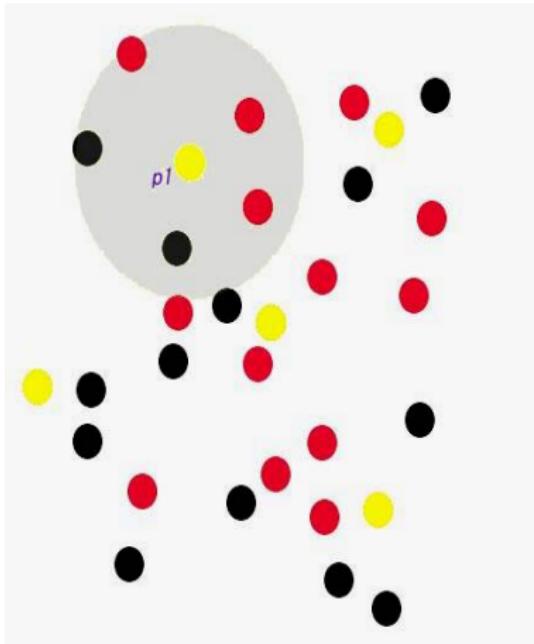
- Class A
- Class B
- New data

What is the label of new (yellow) points
 $p_i, i = 1 \dots 5?$

KNN: choose $K = 5$

k-Nearest Neighbors (kNN): example

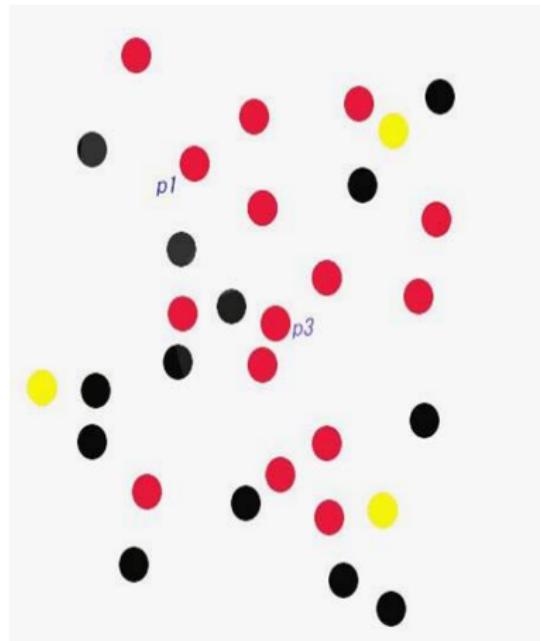
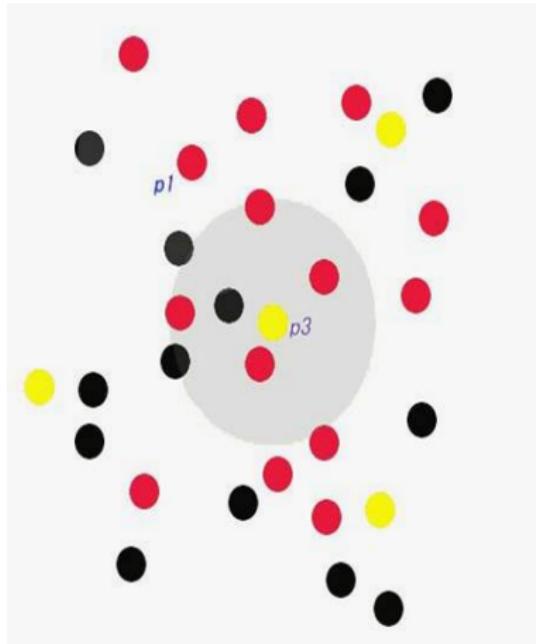
Looking for neighbors of p_1



Within 5 neighbors: 3 are of class A then Class A

k-Nearest Neighbors (kNN): example

Looking for neighbors of p_3



Within 5 neighbors: 3 are of class A then Class A

k-Nearest Neighbors (kNN): basic principles

- The **class membership** of a new object is estimated by **majority vote within nearest neighbors**.
 - Note $k = 1$, is the label of the **nearest neighbor**.
- The definition of neighborhood depends on a proper measure

Different measures to find neighbors

- **Euclidean distance**
- **Manhattan distance**
- **Chebyshev distance**
- **Cosine distance**
- ...

k-Nearest Neighbors (kNN): Choosing k

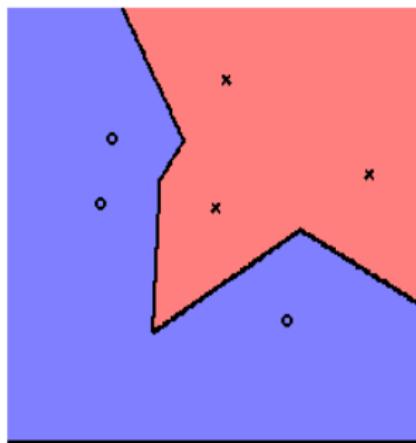
What should be the value of **k** ?

- typically, 3, 5 and 7
- odd numbers to avoid draws
- it can be estimated experimentally
 - **global** estimation searches for the ideal k for a given data set
 - **local** estimation methods try to estimate the ideal **k** for each test case (computationally very demanding!)

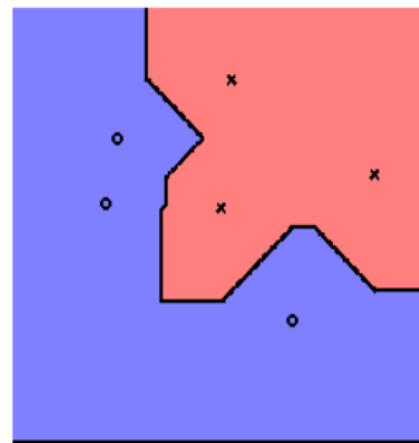
k-Nearest Neighbors (kNN): toy example

$K = 1$ and Euclidian distance versus Manhattan

knn ($K=1$): l2 Distance

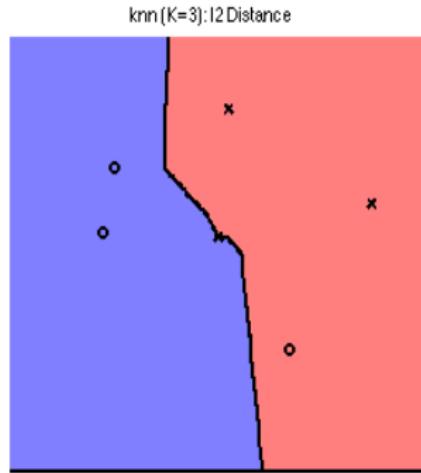
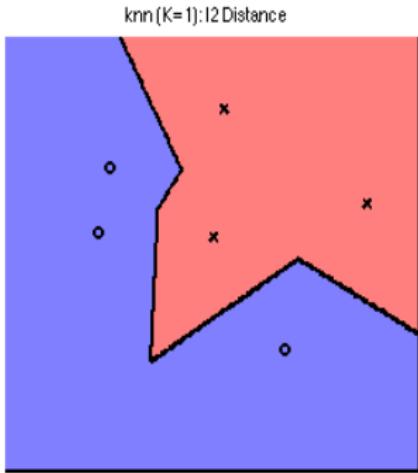


knn ($K=1$): l1 Distance



k-Nearest Neighbors (kNN): toy example

Euclidian distance with $k = 1$ versus $k = 3$



k-Nearest Neighbors (kNN): Advantages

Algorithm provides a highly effective inference method for noisy training data

Algorithm is easy to interpret

New classes can be added without re-training

Different metrics provide flexibility

Works well for online learning as new data is constantly arriving

k-Nearest Neighbors (kNN): Disadvantages

Requires good choices!

- Results depend on choice of k
- Results depend on choice of metric, especially in high-dim spaces
 - normalization, irrelevant variables, unknown values, outliers may have a strong impact on the performance

"Training set" consumes much main memory

- Complexity grows linearly with the number of cases

Classification is time consuming

- Fast training time, but slow testing time

References

Introduction to Data Mining, Pang-Ning Tan, Michael Steinbach, Anuj Karpatne, Vipin Kumar, Pearson, 2019 (chap 6.3)

Data Mining, the Textbook, Charu C. Aggarwal, Springer, 2015 (chap 10.8)

Data Mining

Predictive Modelling

Linear classification models

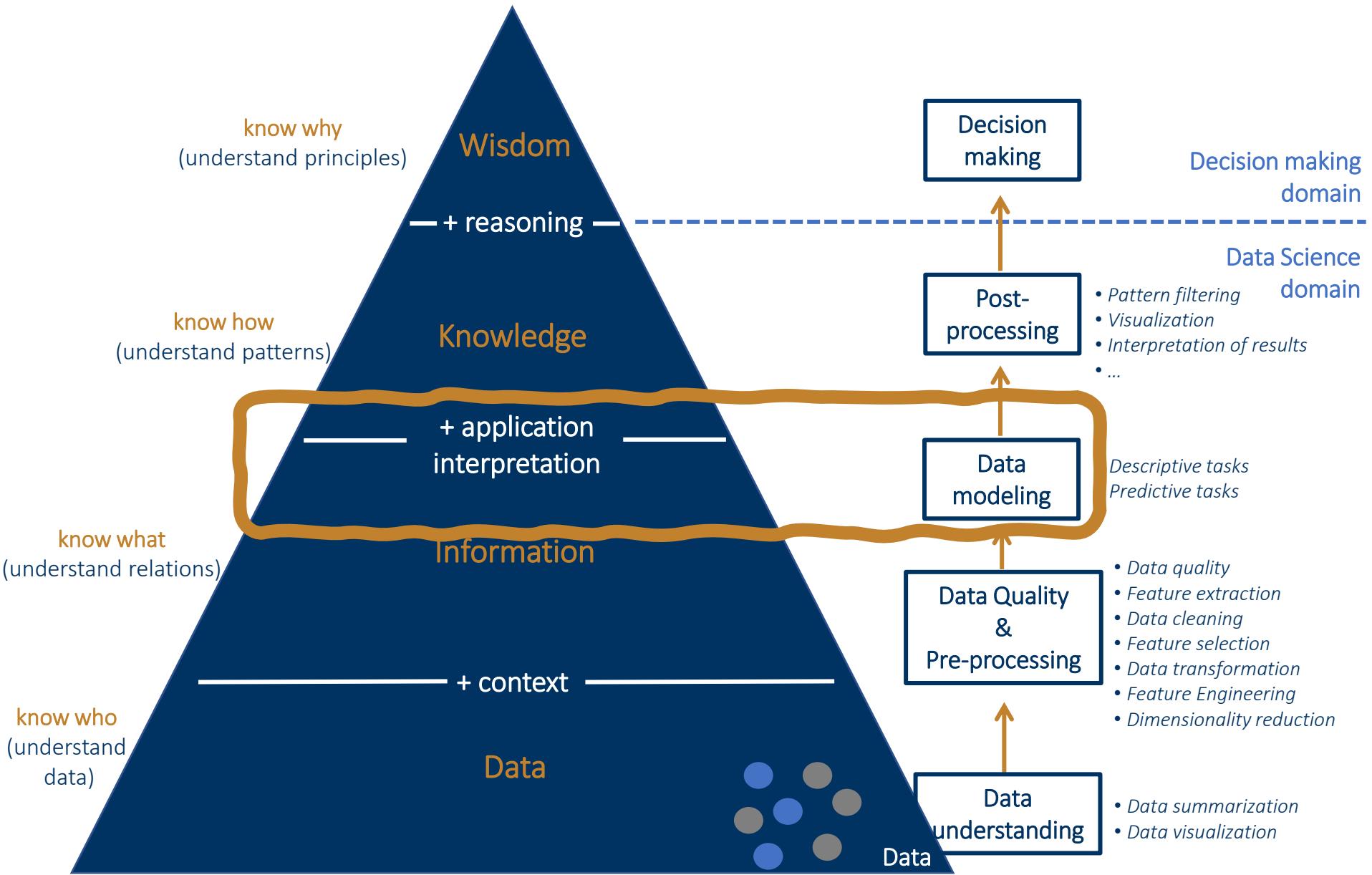
Raquel Sebastião

Departamento de Eletrónica, Telecomunicações e Informática

Universidade de Aveiro

raquel.sebastiao@ua.pt

2022/2023



Contents

- Predictive modelling
- Linear classification models
- Iterative optimization
- Summary

Predictive problems

Classification problems

Id	Sepal length	Sepal width	Petal length	Petal width	Type
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3	1.4	0.2	setosa
51	7	3.2	4.7	1.4	versicolor
101	6.3	3.3	6	2.5	virginica
102	5.8	2.7	5.1	1.9	virginica

Target: class label
(nominal var.)

Regression problems

Id	MSSubClass	LotArea	LotConfig	Neighborhood	Condition1	(...)	YearBuilt	Exterior1st	SalePrice
1	60	8450	Inside	CollgCr	Norm	(...)	2003	VinylSd	208500
67	70	9550	Corner	Crawfor	Norm	(...)	1915	Wd Sdng	140000
106	50	14115	Inside	Mitchel	Norm	(...)	1993	VinylSd	143000
208	60	10382	Corner	NWAmes	PosN	(...)	1973	HdBoard	200000
386	50	6120	Inside	OldTown	Artery	(...)	1931	BrkFace	129900

Target: numeric
(numerical var.)

Predictive problems

Classification vs Regression

- **Classification** assigns input vector \mathbf{x} to one of k discrete classes C_k , $k = 1, \dots, K$
 - Common classification scenario: classes considered disjoint
 - Each input assigned to only one class
 - Input space is thereby divided into decision region
- **Regression** assigns input vector \mathbf{x} to one or more continuous target variables t
 - Linear regression has simple analytical and computational properties

Classification: problem definition

Setting

- Given a **training data set** $\{(x_i, y_i)\}_{i=1}^N$, where each **object** is represented by a **D+1–tuple**: (D-dim) feature vector $x_i \in \mathbb{R}^D$ and the corresponding **label** $y_i \in Y$
- There is an **unknown function**: $Y = f(X)$

Goal

Learn the **model** that yields the best approximation of the unknown function $f()$

Approach

- Assume a functional form $h_{\theta}(x)$ for the unknown function $F()$, where θ are a set of parameters
- Assume a preference criterion over the space Θ of possible parameterizations of $h()$
- Search for the “best” h_{θ} (according to the criterion and the data set)

Classification: learn/build a prediction model

Given a training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$:

- $(\mathbf{x}_i, \omega_{c_i})$ - the label is **symbolic** or **categorical** (ex.: binary {malignant, benign})
- (\mathbf{x}_i, t_i) - label is **numerical** (ex.: binary {-1, 1}, multiclass {0, 1, 2}))

Learning/Training phase

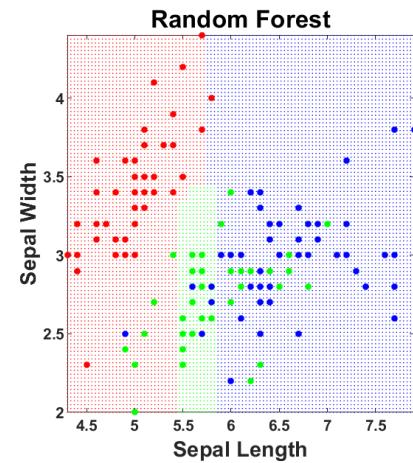
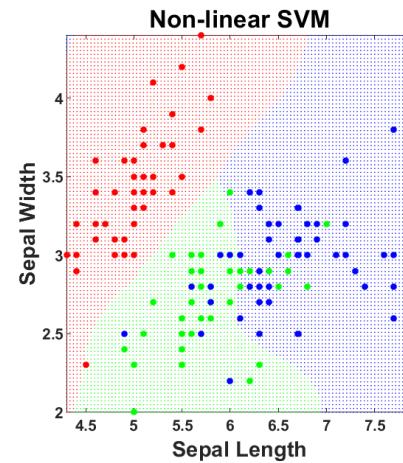
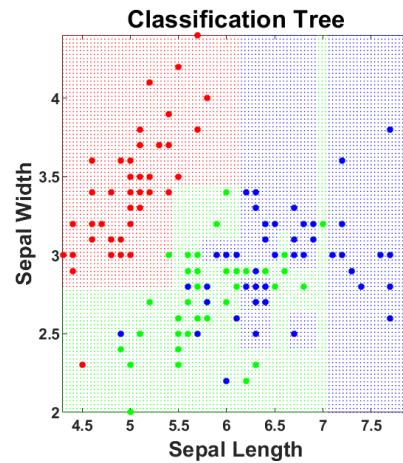
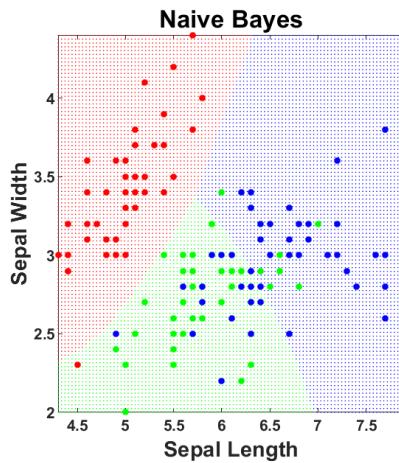
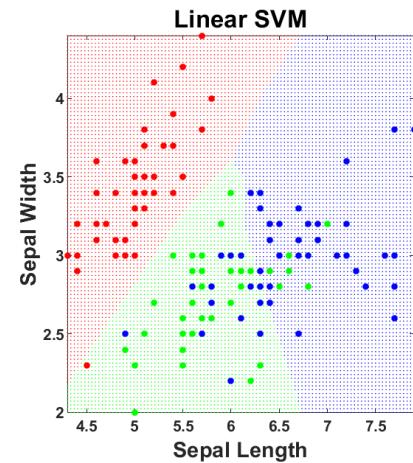
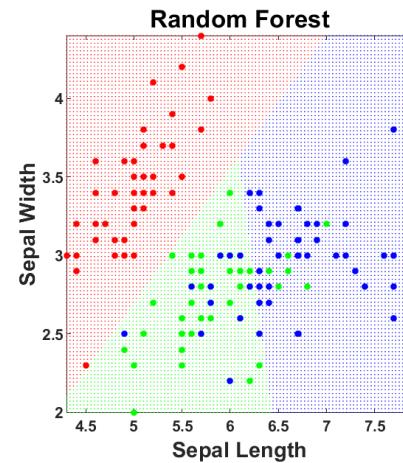
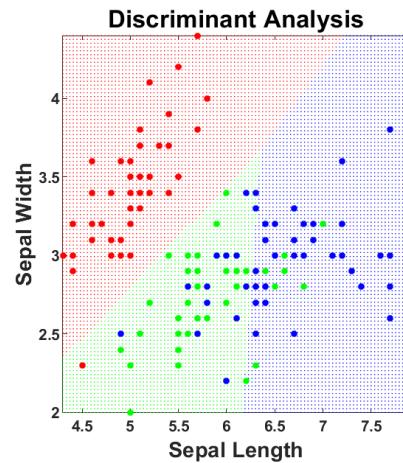
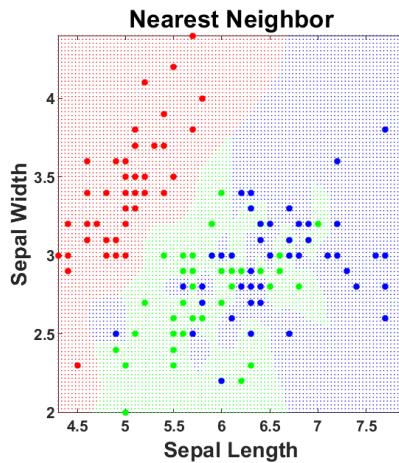
- find the “best” approximation to F, h_{θ}

Testing phase: Given a test set (data not included in the training set)

- Study the performance of the model

Usually, the available data set is divided into training and test sets

Classification: decision surfaces



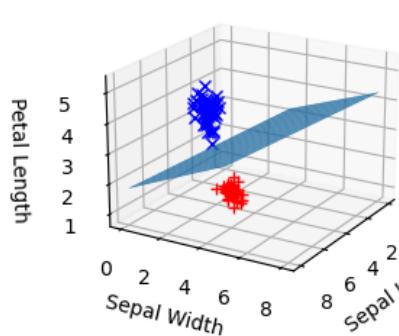
● **setosa**

● **versicolor**

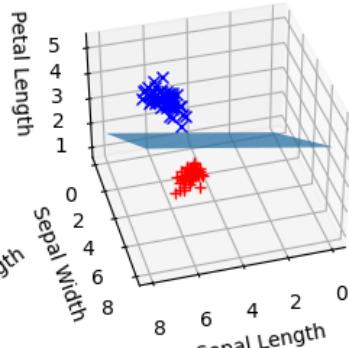
● **virginica**

Classification: decision surfaces

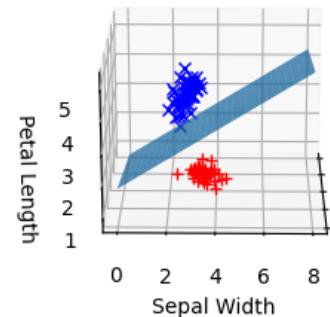
linear SVM



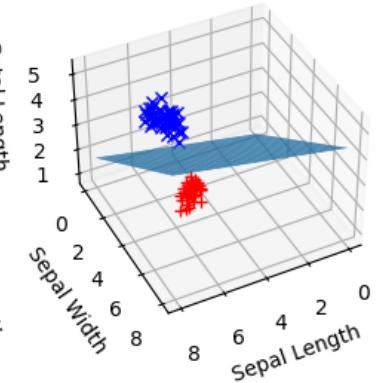
linear SVM



LDA

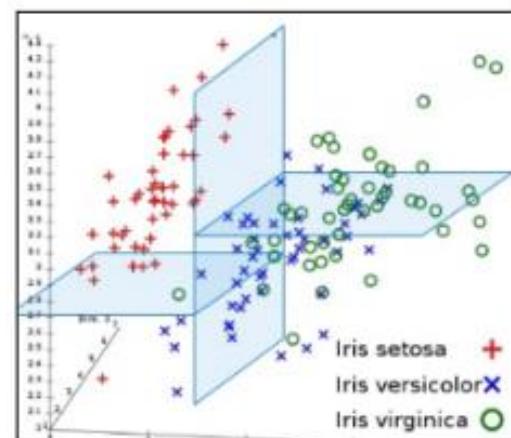


LDA



+ setosa

x versicolor



Prediction Models – approaches

Geometric approaches

- Distance-based: kNN
- Linear models: Fisher's linear discriminant, perceptron, logistic regression, SVM (w. linear kernel)

Probabilistic approaches

- naive Bayes, logistic regression

Logical approaches

- classification or regression trees, rules

Optimization approaches

- neural networks, SVM

Sets of models (ensembles)

- random forests, adaBoost

Contents

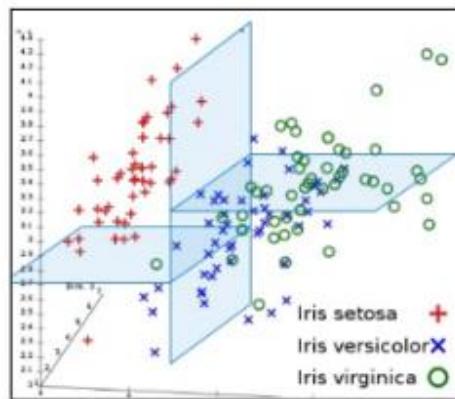
- Predictive modelling
- Linear classification models
 - Discriminant Functions
 - Least squares for classification
 - Fisher's Linear Discriminant (LDA)
 - Perceptron
 - Logistic Regression
 - Comparison
- Iterative optimization
- Summary

Linear classification models

Decision surfaces are linear functions of input x

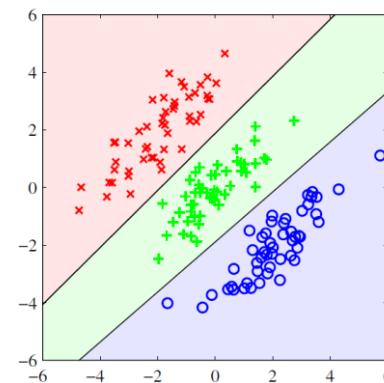
- Defined by $(D - 1)$ -dim hyperplanes within D -dim input space

3-class problem
3-D feature vector



A plane is 2-D surface in 3-D space

3-class problem
2-D feature vector



Straight line is 1-D decision boundary in 2-D space

- Data sets whose classes can be separated exactly by linear decision surfaces are said to be **linearly separable**

Linear classification models

Discriminant functions

- map feature vector x directly into decisions
- probabilities play no role
 - Least Squares for classification
 - Fisher's linear discriminant
 - Perceptron

Probabilistic approaches

- The inference and decision are taken in separated stages
 - Logistic regression (discriminative model)

Support Vector Machines (linear SVM)

Contents

- Predictive modelling
- Linear classification models
 - Discriminant Functions
 - Least squares for classification
 - Fisher's Linear Discriminant (LDA)
 - Perceptron
 - Logistic Regression
 - Comparison
- Iterative optimization
- Summary

Linear discriminant functions

- A **discriminant function** assigns the D-dim feature vector

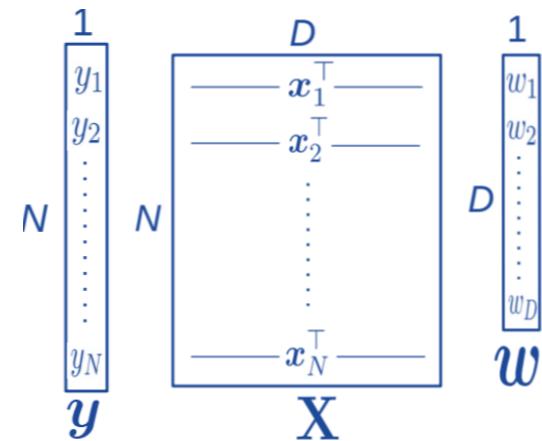
$$\mathbf{x} = [\mathbf{x}_1 \quad \mathbf{x}_2 \quad \dots \mathbf{x}_D]^T \in \mathbb{R}^D \text{ to one of the } k \text{ classes denoted by } C_k$$

- Linear discriminant functions

$$g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0$$

$g(\mathbf{x})$ is linear combination of feature vector \mathbf{x}

- \mathbf{w} is the **weighted vector** ($\mathbf{w} = [w_1 \quad w_2 \quad \dots \quad w_D]^T$)
- w_0 is the **bias** ($threshold = -bias = -w_0$)
- Decision surfaces are hyperplanes



Linear discriminant functions: 2-class

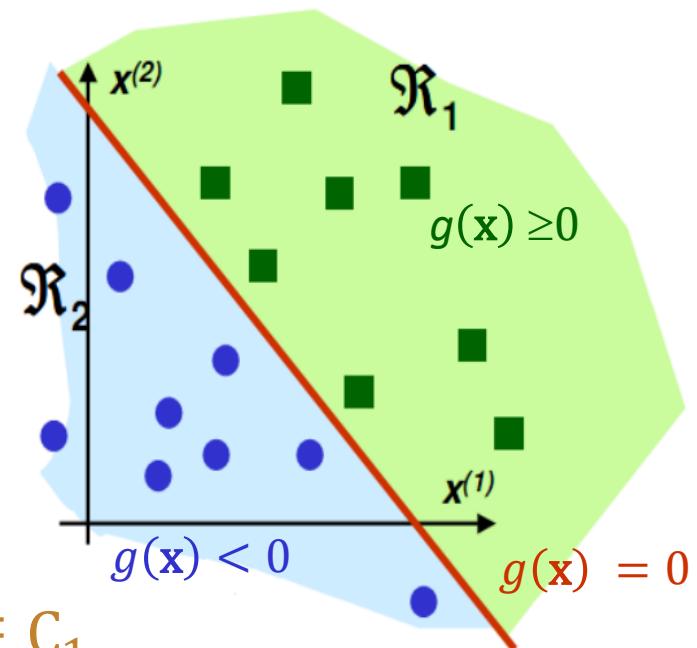
- Linear discriminant functions

$$g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0$$

- \mathbf{w} is the **weighted vector** ($\mathbf{w} = [w_1 \quad w_2 \quad \dots \quad w_D]^T$)
- w_0 is the *bias* (*threshold* = - bias = - w_0)

- Decision surface: $g(\mathbf{x}) = 0$
 - (D-1)-dimensional hyperplane within the D-dimensional feature vector
- Classification

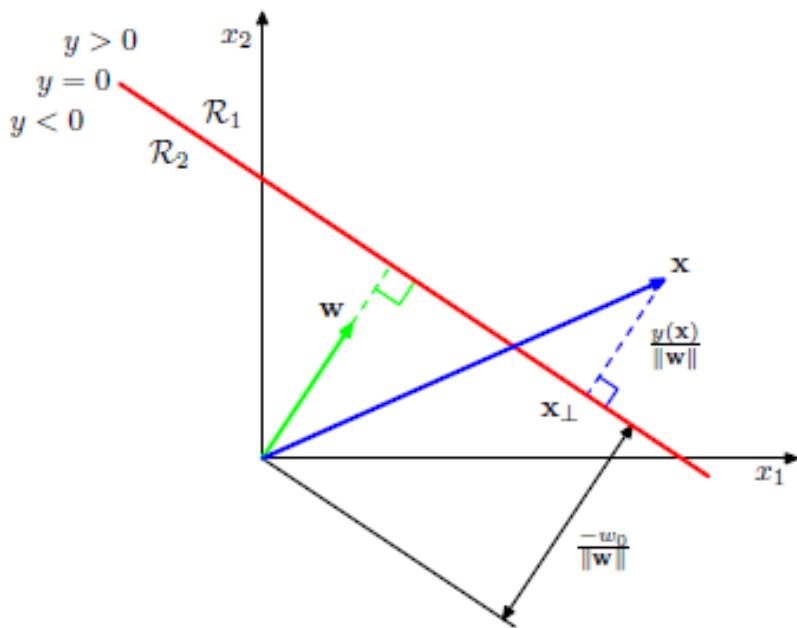
$$g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0 \Rightarrow \begin{cases} g(\mathbf{x}) \geq 0, \mathbf{x} \in C_1 \\ g(\mathbf{x}) < 0, \mathbf{x} \in C_2 \end{cases}$$



Linear discriminant functions: 2-class

- Classification

$$g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0 \Rightarrow \begin{cases} y(\mathbf{x}) > 0, \mathbf{x} \in C_1 \\ y(\mathbf{x}) < 0, \mathbf{x} \in C_2 \end{cases}$$



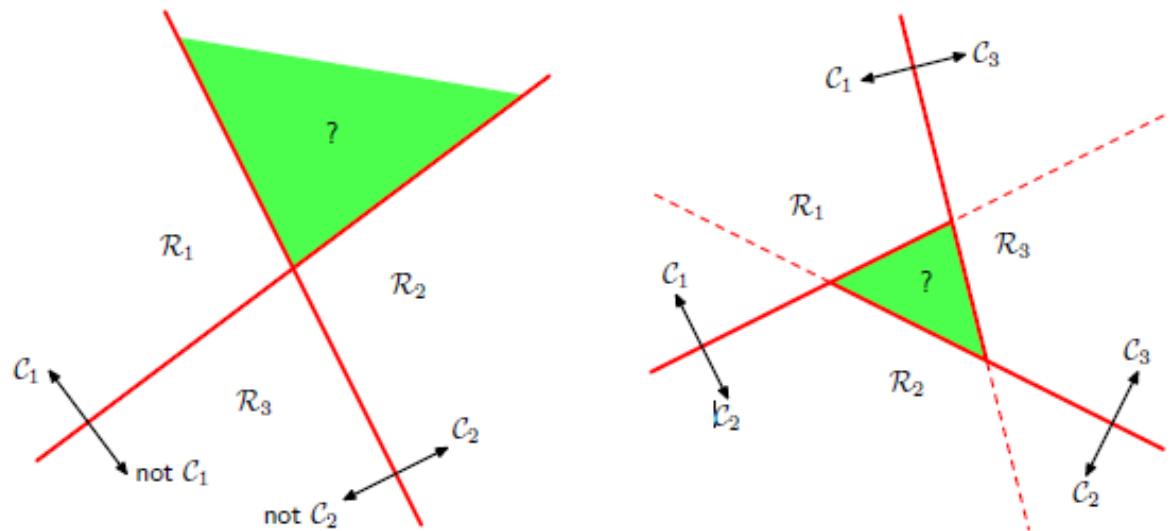
Decision surface: $g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0$

- orientation in space is given by its normal \mathbf{w} ($\mathbf{w} \perp$ decision boundary)
- position in space is given by its distance $\frac{w_0}{\|\mathbf{w}\|}$ to the origin
- Distance of an object \mathbf{x}_a (with label t) to the decision surface $\frac{tg(\mathbf{x})}{\|\mathbf{w}\|}$

Linear discriminant functions: multi-class

Two approaches

- Using several two-class classifiers
 - But leads to serious difficulties
- Use k linear discriminant functions



Linear discriminant functions: learning \mathbf{w}

Linear discriminant function

$$g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0$$

Given a **training data set** (\mathbf{x}, y)

- (D-dim) **feature vector** $\mathbf{x} = [x_1 \quad x_2 \quad \dots x_D]^T \in \mathbb{R}^D$
- corresponding **label** $g(\mathbf{x}) \in Y$

How to learn \mathbf{w} ???

- Least Squares for classification
- Fisher's Linear Discriminant
- Perceptrons

Contents

- Predictive modelling
- Linear classification models
 - Discriminant Functions
 - Least squares for classification
 - Fisher's Linear Discriminant (LDA)
 - Perceptron
 - Logistic Regression
 - Comparison
- Iterative optimization
- Summary

Least squares for classification

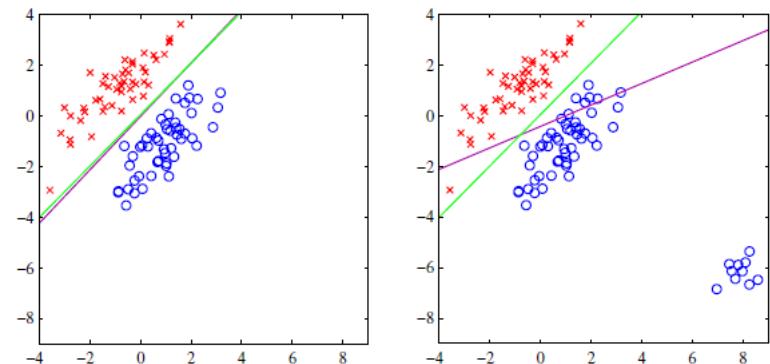
Linear discriminant function

$$g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0$$

Pure geometric-based model

Goal:

- make the model predictions as close as possible to a set of target values
- Find parameters \mathbf{w} that **minimizes the Sum of Square Error** (or **maximizing the likelihood function** under conditional Gaussian distribution)
- **Highly sensitive to outliers**



decision boundary:

- least squares
- logistic regression

Contents

- Predictive modelling
- Linear classification models
 - Discriminant Functions
 - Least squares for classification
 - Fisher's Linear Discriminant (LDA)
 - Perceptron
 - Logistic Regression
 - Comparison
- Iterative optimization
- Summary

Fisher's linear discriminant

Linear discriminant function

$$g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0$$

Basic idea:

- View classification in terms of dimensionality reduction
 - Project D-dim feature vector \mathbf{x} into 1-dim using $\mathbf{w}^T \mathbf{x}$
- The goal is to find \mathbf{w} (direction) to project the data such that:
 - It maximizes the distance between the means of each class (maximizes separability among class)
 - It minimizes the variance within each class

Linear classifier

- Place *threshold* on $g(\mathbf{x})$ to classify $y \geq w_0$ as C_1 and otherwise C_2

Fisher's linear discriminant

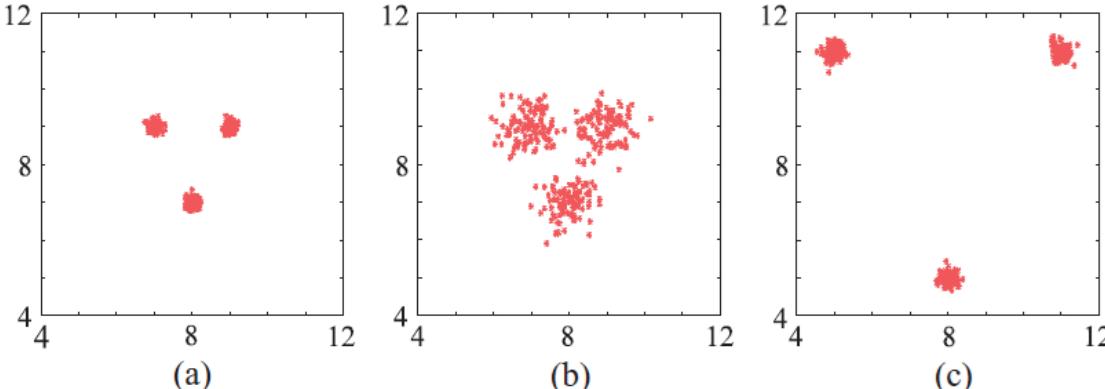
Linear discriminant function

$$g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0$$

The goal is to find \mathbf{w} (direction) to project the data such that:

- It maximizes the distance between the means of each class (maximizes separability among class)
- It minimizes the variance within each class

Example: consider the data as belonging to three classes and projected into two directions



- (c) is the best solution: large distance between classes and small variance within classes

Fisher's linear discriminant

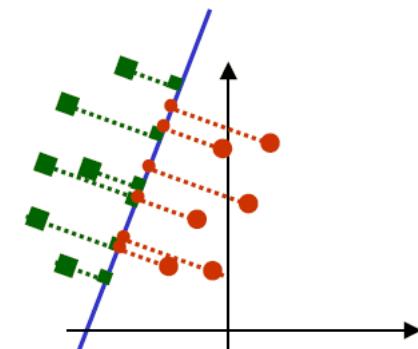
$$g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0$$

The goal is to find directions to project the data such that:

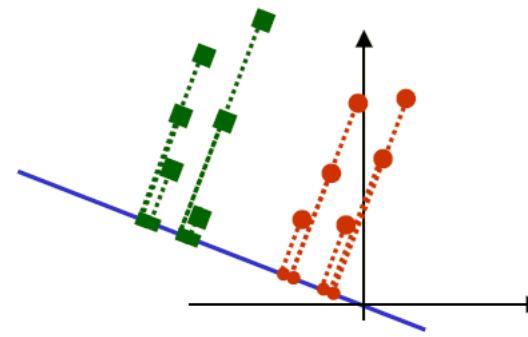
- Large distance between classes
- Small variance within class

Example: Suppose we have 2 classes and 2-dimensional samples

- find projection to a line s.t. samples from different classes are well separated



bad line to project to
(classes are mixed up)



good line to project to
(classes are well separated)

Fisher's linear discriminant: 2-class problems

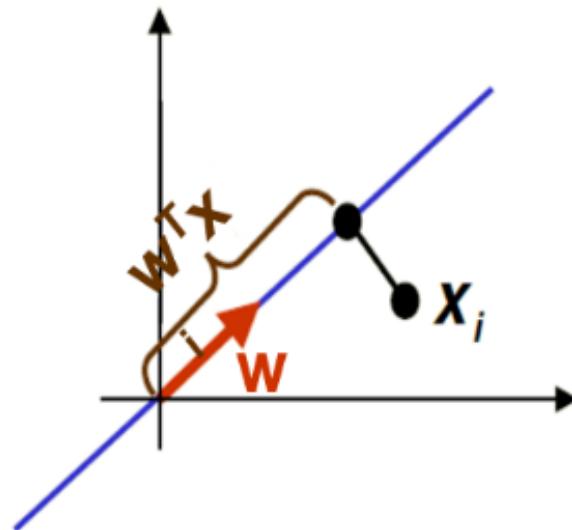
Suppose we have 2 classes and 2-dimensional samples X_1, X_2, \dots, X_N :

- N_1 samples come from **class 1**
- N_2 samples come from **class 2**

Basic idea: Project all samples on a line s.t. different classes are well separated

- Let the **line direction be given by unit vector \mathbf{w}**

- $\mathbf{w}^T \mathbf{x}_i$ is the distance of projection of \mathbf{x}_i from the origin
- Thus $\mathbf{z}_i = \mathbf{w}^T \mathbf{x}_i$ is the projection of \mathbf{x}_i into a one dimensional subspace



Fisher's linear discriminant: learning \mathbf{w}

2-class problems

- The projection of \mathbf{x}_i into a one dimensional subspace with direction \mathbf{w} is given by $\mathbf{z}_i = \mathbf{w}^T \mathbf{x}_i$

How to measure separation between projections of different classes?

- Consider that μ_1 and μ_2 are the means of class 1 and 2 (in the original space)
- Let $\tilde{\mu}_1$ and $\tilde{\mu}_2$ be the means of projections of classes 1 and 2
- thus $|\tilde{\mu}_1 - \tilde{\mu}_2|$ seems like a **good measure**:

$$\tilde{\mu}_c = \frac{1}{N_c} \sum_{i \in \omega_c} \mathbf{w}^T \mathbf{x}_i = \mathbf{w}^T \left(\frac{1}{N_c} \sum_{i \in \omega_c} \mathbf{x}_i \right) = \mathbf{w}^T \mu_c$$

where N_c is the number of samples of class ω_c ($c = 1, 2$)

Fisher's linear discriminant: learning w 2-class problems

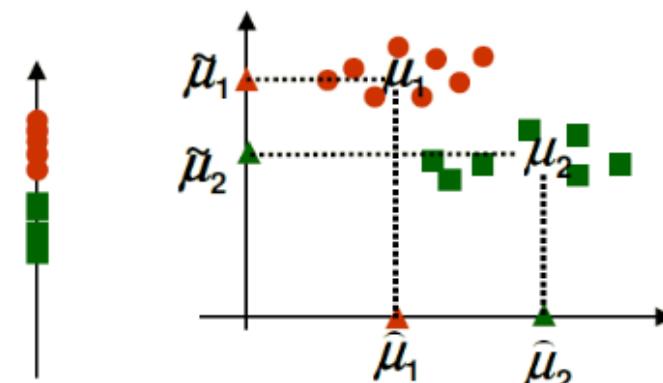
How good is $|\tilde{\mu}_1 - \tilde{\mu}_2|$ as a measure of separation?

- The larger $|\tilde{\mu}_1 - \tilde{\mu}_2|$ the better is the expected separation

- Consider the two directions for projecting the following data:

- $\mathbf{w} = (0, 1)$

- $\mathbf{w} = (1, 0)$



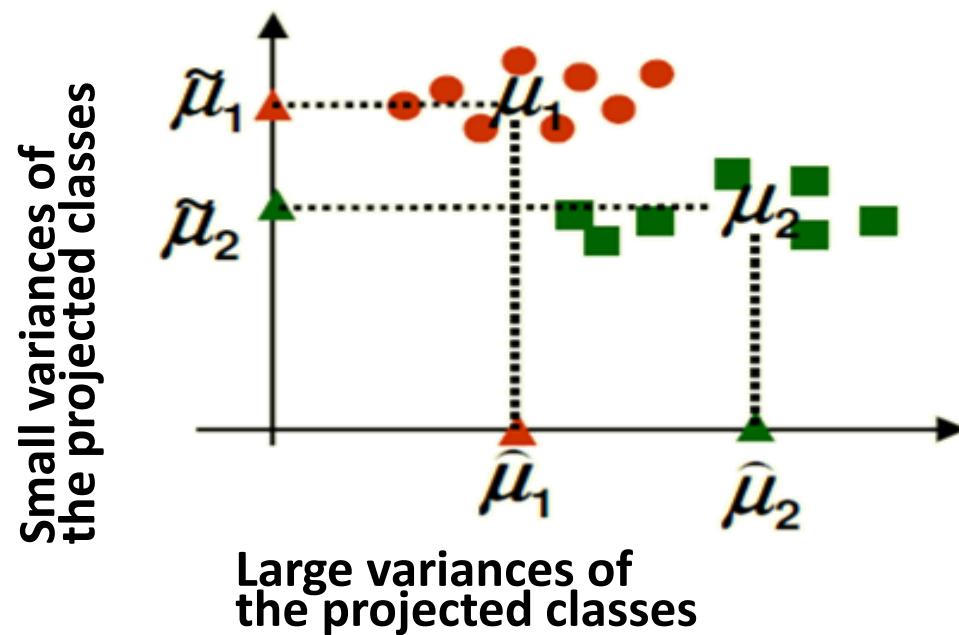
- The distance between the means of the projected data is **higher** in the horizontal axes \Rightarrow **Large distance** between classes

However the vertical axes is a **better** line to project attending the class separability

Fisher's linear discriminant: learning w 2-class problems

How good is $|\tilde{\mu}_1 - \tilde{\mu}_2|$ as a measure of separation?

- Problem: $|\tilde{\mu}_1 - \tilde{\mu}_2|$ does not consider the variance of the classes!!!



Ideally, the projected classes have both **faraway means** and **small variances** !!

Fisher's linear discriminant: learning w 2-class problems

Problem: $|\tilde{\mu}_1 - \tilde{\mu}_2|$ does not consider the variance of the classes!!!

Thus, $|\tilde{\mu}_1 - \tilde{\mu}_2|$ needs to be **normalized** by a factor proportional to **variance**¹

Fisher Solution: normalize $|\tilde{\mu}_1 - \tilde{\mu}_2|$ by **scatters**

- Consider the projected samples: $\mathbf{z}_i = \mathbf{w}^T \mathbf{x}_i$
- Scatters (sample variance multiplied by N) for projected samples:
 - $\tilde{s}_1 = \sum_{\mathbf{z}_i \in C_1} (\mathbf{z}_i - \tilde{\mu}_1)^2$
 - $\tilde{s}_2 = \sum_{\mathbf{z}_i \in C_2} (\mathbf{z}_i - \tilde{\mu}_2)^2$
- The optimal solution can be achieved by the maximization, w.r.t \mathbf{w} , of:

$$J(\mathbf{w}) = \frac{(\tilde{\mu}_1 - \tilde{\mu}_2)^2}{\tilde{s}_1 + \tilde{s}_2}$$

Closed-form optimal solution

The optimal \mathbf{w} should be such that

- $(\tilde{\mu}_1 - \tilde{\mu}_2)^2$: large
- $\tilde{s}_1; \tilde{s}_2$: both small

¹scatter measures the same thing as variance, the spread of data around the mean

Fisher's linear discriminant: learning \mathbf{w} 2-class problems

The optimal solution can be achieved by the maximization, w.r.t \mathbf{w} , of:

$$J(\mathbf{w}) = \frac{(\tilde{\mu}_1 - \tilde{\mu}_2)^2}{\mathbf{s}_1 + \mathbf{s}_2}$$

- express J explicitly as a function of \mathbf{w} and maximize it

- straightforward but need linear algebra and calculus

- (...)

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_b \mathbf{w}}{\mathbf{w}^T \mathbf{S}_w \mathbf{w}}$$

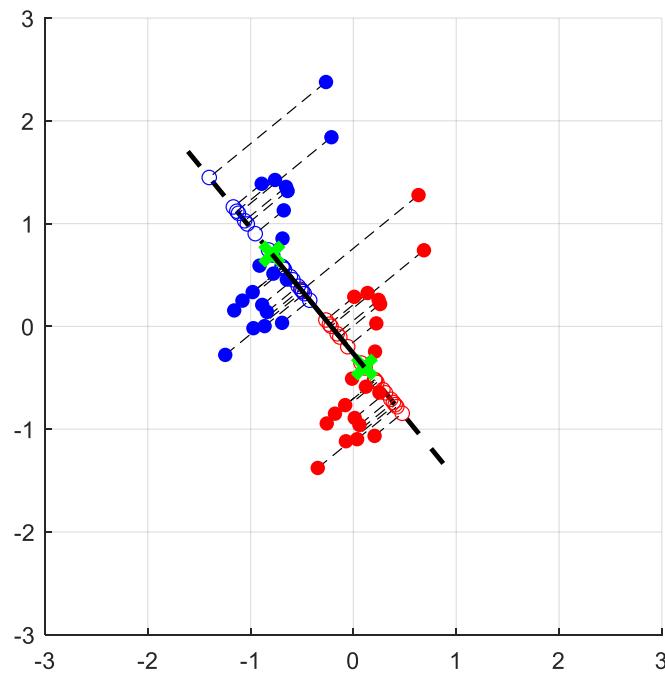
- \mathbf{S}_b is the between class scatter matrix
 - \mathbf{S}_w is the within class scatter matrix
 - Thus, supposing \mathbf{S}_w non-singular, the maximum is equivalent to the largest eigenvector \mathbf{w}_1 of $\mathbf{S}_w^{-1} \mathbf{S}_b$:

$$\mathbf{S}_w^{-1} \mathbf{S}_b \mathbf{w}_1 = \lambda_1 \mathbf{w}_1$$

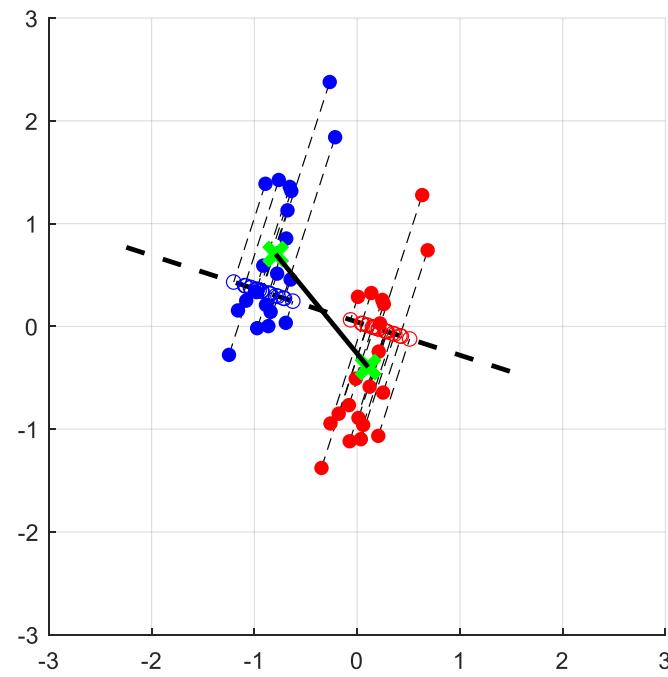
- Moreover, noting that \mathbf{S}_b is in the same direction as $(\tilde{\mu}_1 - \tilde{\mu}_2)$, and multiplying by \mathbf{S}_w^{-1} :

$$\mathbf{w} \propto \mathbf{S}_w^{-1} (\tilde{\mu}_1 - \tilde{\mu}_2)$$

Fisher's linear discriminant: learning w 2-class problems



(a)



(b)

(a) direction to project is parallel to the vector of the difference between means. **Classes overlap**

(b) optimal direction (Fisher's solution): accounts for the difference between means and variance minimization. **Classes do not overlap**

Fisher's linear discriminant: learning w 2-class problems

- A variant for $C = 2$ classes
 - with both classes equiprobable (the same number of elements of each class in the data set)
 - avoids **eigendecompositions**

The **main steps** are

1. Compute **class means**: given N_c elements in each class

$$\mu_c = \frac{1}{N_c} \sum_{i \in \omega_c} \mathbf{x}_i, \quad c = 1, 2$$

2. Compute **within-class scatter matrix**: $\mathbf{S}_W = \mathbf{S}_1 + \mathbf{S}_2$

- The matrices \mathbf{S}_1 and \mathbf{S}_2 are computed with data of class ω_1 and ω_2 , respectively.

3. The **direction to project the data** is $\mathbf{w} = \mathbf{S}_W^{-1}(\mu_2 - \mu_1)$

4. Project data: $y_k = \mathbf{w}^T \mathbf{x}_k, \quad k = 1, 2 \dots N$

Fisher's linear discriminant: learning w 2-class problems

Decision rule: projecting the mean point of the means of two classes

$$p = 0.5\mathbf{w}^T(\mu_2 + \mu_1)$$

p is **the threshold** used to identify the projections y_k of each class.

- $y_k > p$ belongs to class ω_1 otherwise belongs ω_2

Re-writing by incorporating the projection calculation and $b = -p$

$$g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b \Rightarrow \begin{cases} g(\mathbf{x}) > 0 & \mathbf{x} \in \omega_1 \\ g(\mathbf{x}) < 0 & \mathbf{x} \in \omega_2 \end{cases}$$

The **linear discriminant function**: decision is taken with an **weighted sum of the feature values**

Fisher's linear discriminant: learning \mathbf{w} algebraic approach

Given a training set: $(\mathbf{x}_k, y_k), k = 1 \dots N$, the following system of equations

$$\begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1D} \\ 1 & x_{21} & x_{22} & \dots & x_{2D} \\ 1 & x_{31} & x_{32} & \dots & x_{3D} \\ 1 & & \dots & & \\ 1 & x_{N1} & x_{N2} & \dots & x_{ND} \end{bmatrix} \begin{bmatrix} w_0 \equiv b \\ w_1 \\ \dots \\ w_D \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \dots \\ y_K \end{bmatrix} \Leftrightarrow \mathbf{X}\mathbf{w} = \mathbf{y}$$

where

- The k th row of \mathbf{X} has $[1 \mathbf{x}_k^T]$
- The k th row of vector \mathbf{y} has y_k the value that the model should predict with \mathbf{x}_k
- The unknown \mathbf{w} : the vector of parameters of the model

System of equations: more equations than variables? \Rightarrow Minimizing least squares

Fisher's linear discriminant: learning \mathbf{w} algebraic approach

$$\mathbf{X}\mathbf{w} = \mathbf{y}$$

Multiplying both members by \mathbf{X}^T ²

$$\mathbf{X}^T \mathbf{X}\mathbf{w} = \mathbf{X}^T \mathbf{y} \Rightarrow \mathbf{R}_{xx}\mathbf{w} = \mathbf{r}_{xy}$$

The solution is

$$\mathbf{w} = \mathbf{R}_{xx}^{-1} \mathbf{r}_{xy}$$

- \mathbf{R}_{xx} is the correlation matrix (of augmented data vector $[1 \quad \mathbf{x}_k]$)
- The pseudo-inverse of \mathbf{R}_{xx} can substitute the inverse.

²The first column of the data matrix has only ones

Fisher's linear discriminant: learning \mathbf{w}

Toy example 2D

Given the data set

	A_1	A_2	label $\equiv d$
\mathbf{x}_1^T	-1	1	1
\mathbf{x}_2^T	1	-1	1
\mathbf{x}_3^T	-1	-1	1
\mathbf{x}_4^T	1	1	-1

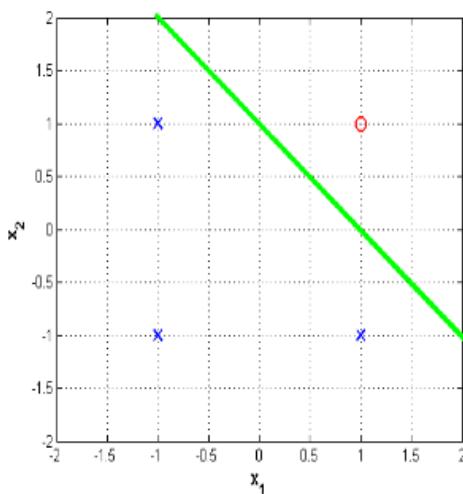
The solution of the system of equations is

$$\mathbf{w} = \begin{pmatrix} w_0 \equiv b \\ w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} 0.5 \\ -0.5 \\ -0.5 \end{pmatrix}$$

Fisher's linear discriminant: learning w

Toy example 2D

The decision surface (**green line**) divides the feature space into two regions



The parameters of the classifier are

- $\mathbf{w} = [-0.5 \quad -0.5]^T$ and $b = 0.5$

The equation of green line is
 $0.5 - 0.5x_1 - 0.5x_2 = 0$

$$\Rightarrow x_2 = 1 - x_1$$

The decision rule is

$$y = [-0.5 \quad -0.5] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + b = -0.5x_1 - 0.5x_2 + 0.5 \Rightarrow \begin{cases} y > 0 & \text{Region } \mathfrak{R}_1 \\ y < 0 & \text{Region } \mathfrak{R}_2 \end{cases}$$

Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA)

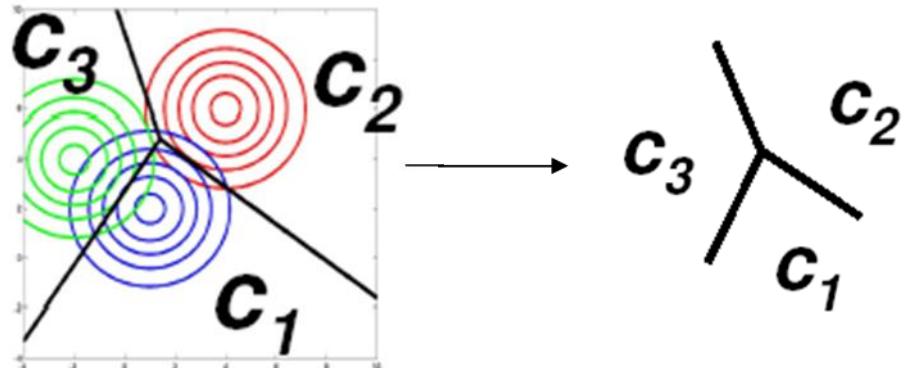
- linear model for multiclass classification (generalization of Fisher's linear disc.)
- dimensionality reduction

Binary classification

- compute w of $g(x)$

Decision:

- Given a new object z , compute $g(z) = w^T z + w_0 \Rightarrow \begin{cases} g(x) \geq 0, x \in C_1 \\ g(x) < 0, x \in C_2 \end{cases}$



Multi-class classification

- compute the parameters of C functions $g_c(x) = w_c^T x + b_c, c = 1, \dots, C$

Decision:

- Given a new object $z : z \in C_c : g_c(z) > g_j(z), \forall j \neq c$

Linear discriminant analysis: learning w multiclass

The solution is based on the definition

- **Within** class scatter matrices

$$\mathbf{S}_W = \sum_{c=1}^C \mathbf{S}_c$$

where \mathbf{S}_c is the **scatter matrix** computed with the **data belonging to c-class**

- **Between** class scatter matrices

$$\mathbf{S}_B = \sum_{c=1}^C N_c \mathbf{S}_{m_c}$$

where N_c is the number of objects in class c and

$$\mathbf{S}_{m_c} = (\boldsymbol{\mu}_c - \boldsymbol{\mu})(\boldsymbol{\mu}_c - \boldsymbol{\mu})^T$$

with $\boldsymbol{\mu}$ is the **global** mean vector and $\boldsymbol{\mu}_c$ is the **mean** vector of the **data belonging to class c**

Linear discriminant analysis: learning w multiclass

- The $C - 1$ optimal directions are computed with the eigendecomposition of

$$\mathbf{S}_T = \mathbf{S}_w^{-1} \mathbf{S}_B$$

- The matrix \mathbf{S}_w should be invertible. For high-dimensional and sparse data it might be a drawback.
- The \mathbf{S}_T is not symmetric it might have non-real eigenvalues.
- The generalized eigendecomposition of $(\mathbf{S}_w, \mathbf{S}_B)$ is an alternative solution.

The chosen $(C - 1)$ **eigenvectors** form the columns of the **projection model w**

- The new representation (reduced dimension)

$$\mathbf{P} = \mathbf{X}\mathbf{w}$$

is formed by $(C - 1)$ **features**

Fisher's and LDA

FLF and LDA assumptions on data:

- normal or Gaussian distribution
- classes with identical covariance matrices
 - However, it performs quite well when assumptions are violated → optimized

FLF and LDA

- fails when the discriminatory information is not in the mean, but rather in the variance of the data

Contents

- Predictive modelling
- Linear classification models
 - Discriminant Functions
 - Least squares for classification
 - Fisher's Linear Discriminant (LDA)
 - Perceptron
 - Logistic Regression
 - Comparison
- Iterative optimization
- Summary

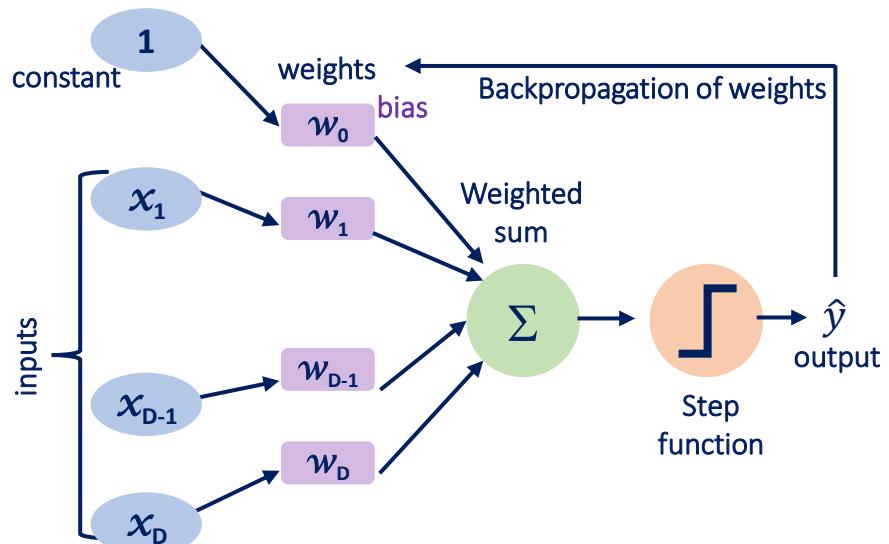
Perceptron

Psychological Review
Vol. 65, No. 6, 1958

THE PERCEPTRON: A PROBABILISTIC MODEL FOR INFORMATION STORAGE AND ORGANIZATION IN THE BRAIN¹

F. ROSENBLATT
Cornell Aeronautical Laboratory

- Proposed by Rosenblatt (1958)
 - introduced the notion of **perceptron networks**
 - correspond to a 2-class model
- Perceptrons are the simplest ANN:
 - Only one input layer
 - Only one output layer
 - Learn linear decision boundaries
 - Binary problems



Perceptron

The processing steps:

- Weighted sum of the inputs:

$$a = \sum_d w_d x_d + w_0 = \mathbf{w}^T \mathbf{x} + w_0$$

- $f(\cdot)$ activation function: step function

$$g(\mathbf{x}) = f(\mathbf{w}^T \mathbf{x} + w_0)$$

$$g(\mathbf{x}) = f(\mathbf{w}^T \mathbf{x} + w_0) = f(a) = \begin{cases} +1, & a > 0 \\ -1, & \text{otherwise} \end{cases}$$

- Output: $f(a)$ - allows to assign a *class* $\in \{-1,1\}$ to feature vector \mathbf{x}

- During the learning phase:

- For each example n ($n = 1, 2, \dots, N$):

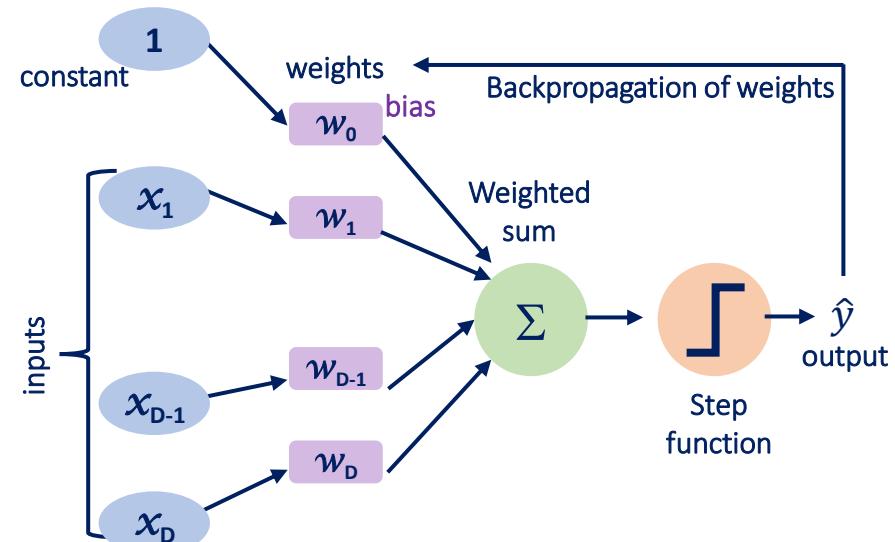
- if $y_n \mathbf{x}_n^T \mathbf{w}^{(i-1)}$:

- update the weights:

$$\mathbf{w}^{(i)} = \mathbf{w}^{(i-1)} + \lambda y_n \mathbf{x}_n$$

(λ is the learning rate)

Perceptron algorithm



Perceptron: learning \mathbf{w}

- Algorithm

Algorithm 18.1 (The online perceptron algorithm).

- Initialization
 - $\theta^{(0)} = \mathbf{0}$.
 - Select μ ; usually it is set equal to one.
 - $i = 0$.
- **Repeat**; Each iteration corresponds to an epoch.
 - counter = 0; Counts the number of updates per epoch.
 - **For** $n = 1, 2, \dots, N$, **Do**; For each epoch, all samples are presented.
 - **If** $(y_n x_n^T \theta^{(i-1)} \leq 0) **Then**
 - $i = i + 1$
 - $\theta^{(i)} = \theta^{(i-1)} + \mu y_n x_n$
 - counter=counter+1$
 - **End For**
 - **Until** counter=0

- It only updates weights¹ when misclassification occurs

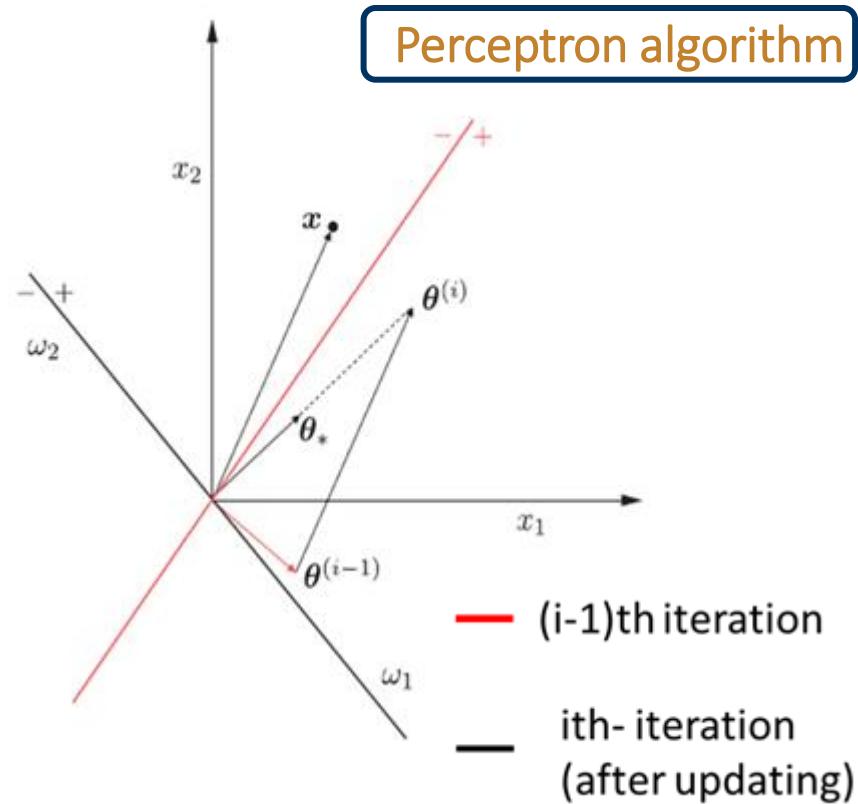
¹ $\mathbf{W} \equiv \theta$, training set (\mathbf{x}_i, y_i) , i iteration (epoch), μ learning rate

Perceptron: learning w

- The decision surface is orthogonal¹ to $\theta = [\theta_1 \ \theta_2]^T$,

At each iteration, only updates weights when misclassification occurs

- Assuming that x is misclassified
 - The weight vector moves towards x



¹ $\theta_0 \equiv b = 0$, the decision lines crosses the origin of the feature space

Perceptron: learning w

Perceptron convergence theorem

- if there exists an exact solution (i.e., if the training data set is linearly separable)
 - perceptron learning algorithm is guaranteed to find an exact solution in a finite number of steps
 - # steps to convergence could still be substantial
- **Limitations of the Perceptron learning algorithm**
 - It only stops when all exemplles are learned
 - Not applicable in real applications
 - Limitation of the weight updating rule
 - It does not guarantee to reduce the total error function at each stage

If data is not linearly separable, the perceptron learning algorithm will never converge

→ Alternative: optimization of the error

Perceptron: learning \mathbf{w}

Perceptron criterion associates zero error with any input correctly classified

- \mathbf{x}_n in C_1 ($y_n = +1$): add to \mathbf{w}
- \mathbf{x}_n in C_2 ($y_n = -1$): subtract from \mathbf{w}
- For each misclassified sample, it tries to minimize

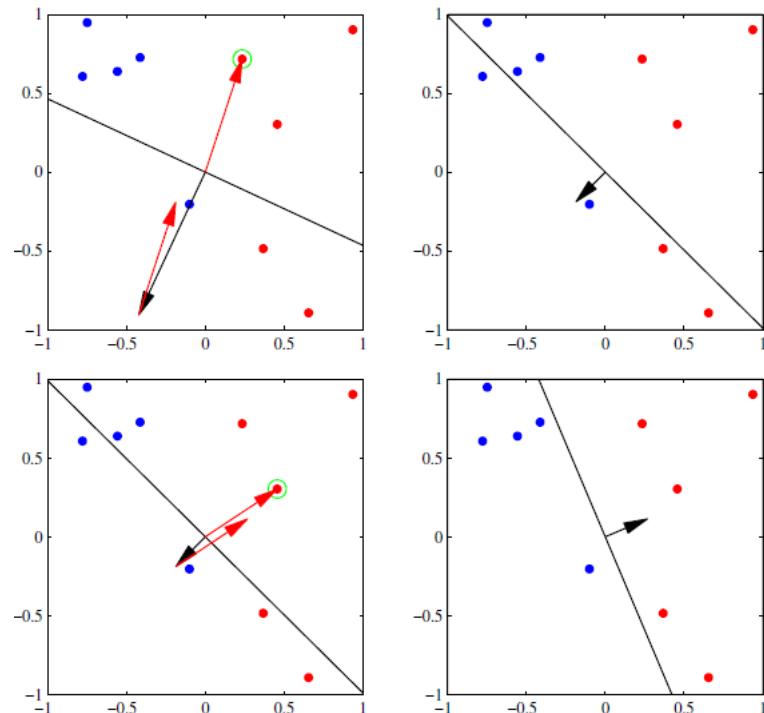
$$E_p(\mathbf{w}) = - \sum_{m \in M} \mathbf{w}^T \mathbf{x}_m y_m = \sum_{m \in M} E_n(\mathbf{w})$$

- M: set of all misclassified samples
- Gradient Descent (GD)
- Weight update formula: for each misclassified \mathbf{x}_m

$$\mathbf{w}^{(i)} = \mathbf{w}^{(i-1)} - \lambda E_n(\mathbf{w}) = \mathbf{w}^{(i-1)} + \lambda(y_n \mathbf{x}_n)$$

GD for Perceptron criterion

$$C_1 : y_n = +1, C_2 : y_n = -1$$

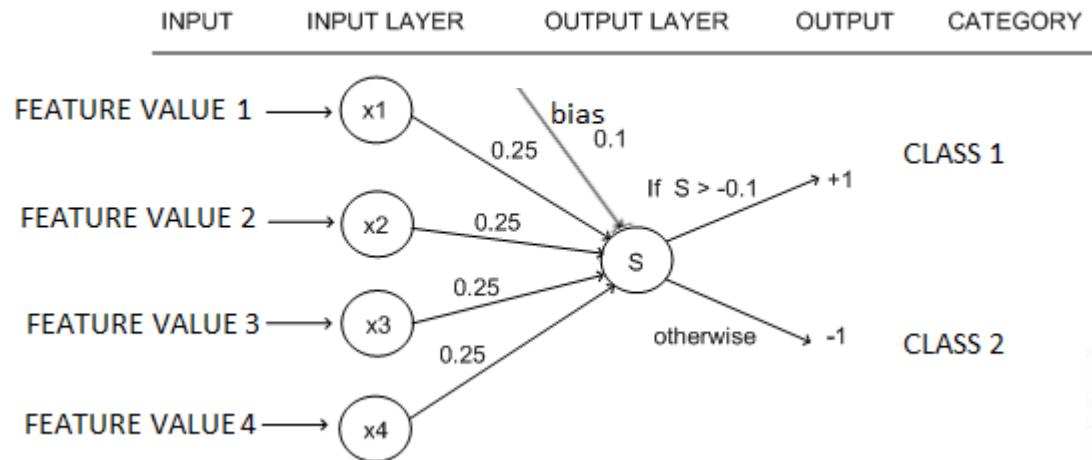


Pattern Recognition and Machine Learning,
C. Bishop, Springer, 2007

Perceptron: limitations

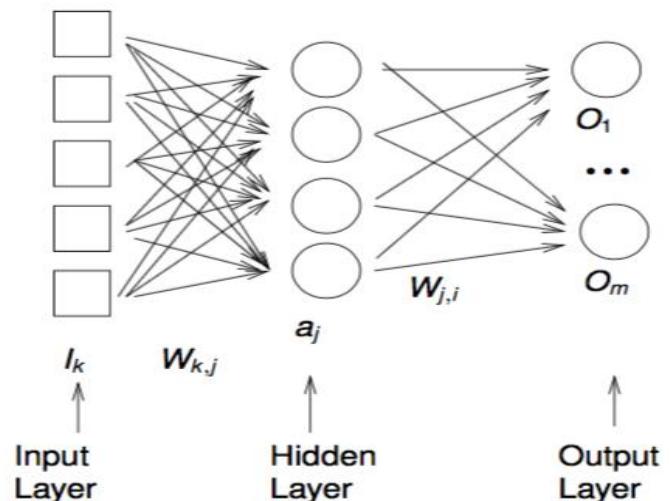
- Perceptron only works for 2 classes
- If dataset is not linearly separable, the **perceptron algorithm will not converge**
- Based on linear combination of fixed basis functions
- **Different solutions** depending on the initialization of **w**
- Online algorithm: **decision boundary** depends on the order of the learning examples
 - Also, the order in which data is presented in GD affects the decision boundaries

Extending the Perceptron ...



Perception: precursor for Neural Networks!

Extending Perceptron: connecting units together into multilayer Neural Networks!

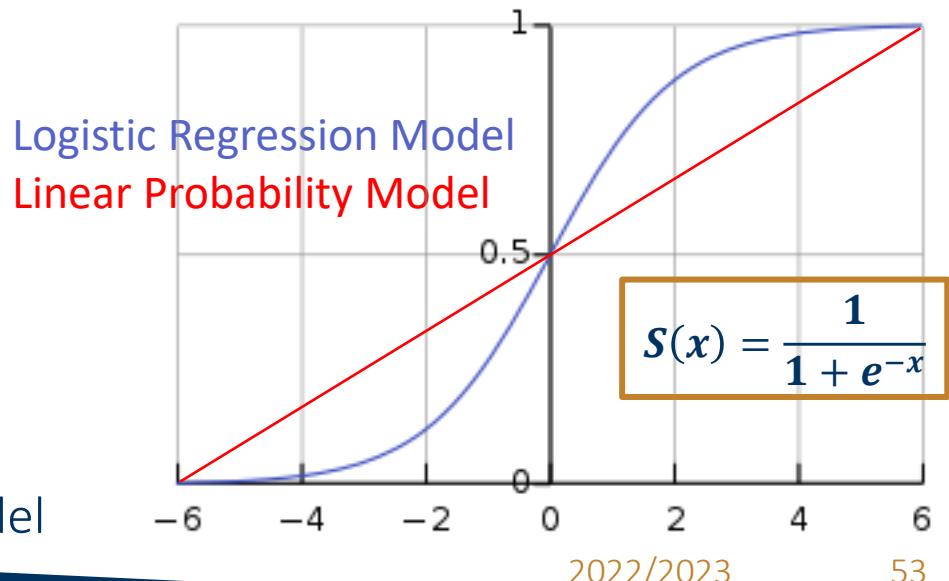


Contents

- Predictive modelling
- Linear classification models
 - Discriminant Functions
 - Least squares for classification
 - Fisher's Linear Discriminant (LDA)
 - Perceptron
 - Logistic Regression
 - Comparison
- Iterative optimization
- Summary

Logistic Regression: a generalized linear model

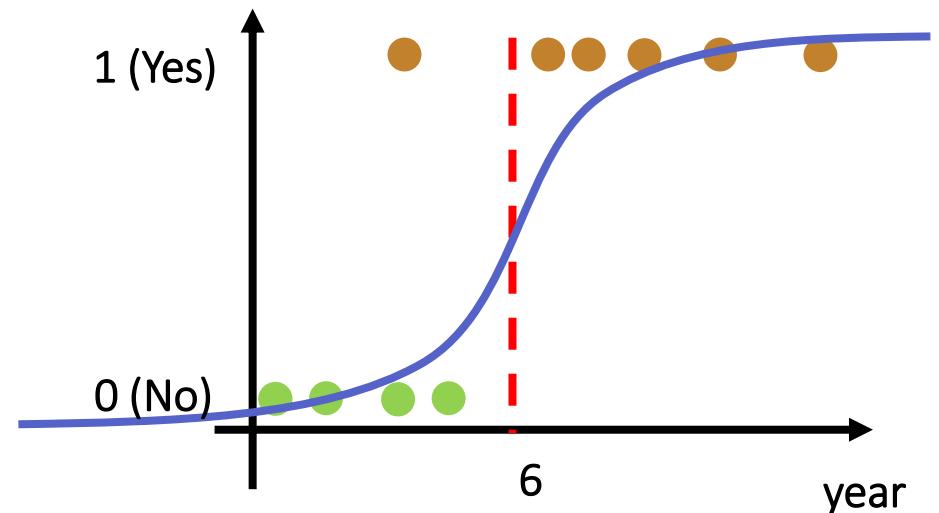
- Probabilistic discriminative linear model for **binary** classification
- **Key Idea:** Turn linear predictions into probabilities
 - The goal is to calculate a **score**
- **Sigmoid function** to define the conditional probability of $y \in \omega_1$
 - $P(y \in \omega_1 | \mathbf{x}) = \frac{1}{1 + \exp(-y)}$, $y = \mathbf{w}^T \mathbf{x} + b$
 - Projects $(-\infty, +\infty)$ to $[0, 1]$
- **Decision rule**
 - $y \in \omega_1$ if $P(y \in \omega_1 | \mathbf{x}) > 0.5$
 - $y \in \omega_2$ if $P(y \in \omega_2 | \mathbf{x}) > 0.5$
- Smoother than linear probability model



Logistic Regression

- Key Idea: Turn linear predictions into probabilities
 - The goal is to calculate a score

$$S(x) = \frac{1}{1 + e^{-x}}$$



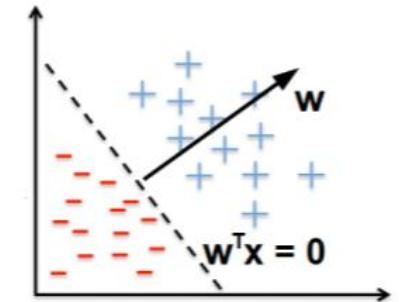
- Decision rule
 - $y \in \omega_1$ if $P(y \in \omega_1 | \mathbf{x}) > 0.5$
 - $y \in \omega_2$ if $P(y \in \omega_2 | \mathbf{x}) > 0.5$

$$y = \mathbf{w}^T \mathbf{x} + b$$

- Probabilistic discriminative linear model for binary classification
 - Learns the PMF of the output label given the input, i.e., $p(y \in \omega_k | \mathbf{x})$, $k = 1, 2$
 - Does not model inputs \mathbf{x} (only relationship between \mathbf{x} and y)
 - $P(y \in \omega_1 | \mathbf{x})$ are nonlinear functions with a linear function of \mathbf{x} as input
 - Linear relation with the parameters (\mathbf{w} , b) of the classifier $g(\mathbf{x})$ despite the non-linear scoring function

Logistic Regression

- Very large positive $\mathbf{w}^\top \mathbf{x}$ means $p(y = 1|\mathbf{x})$ close to 1
- Very large negative $\mathbf{w}^\top \mathbf{x}$ means $p(y = 0|\mathbf{w})$ close to 1
- At decision boundary, $\mathbf{w}^\top \mathbf{x}=0$ implies $p(y = 1|\mathbf{x}) = p(y = 0)$



The logit function gives the logarithm of the probability of one class divided by the probability of the other class

$$\text{logit}(P(y \in \omega_1 | \mathbf{x})) = \ln\left(\frac{P(y \in \omega_1 | \mathbf{x})}{1 - P(y \in \omega_1 | \mathbf{x})}\right) = \mathbf{w}^\top \mathbf{x} + b$$

- It is the log of the odds ratio
- It links the probability to the predictor variables
- Extension **Multiclass Logistic Regression** (aka **Softmax**)

Logistic Regression: learning \mathbf{w}

Given the **training data set** $\chi = \{(\mathbf{x}_n, y_n)\}_{n=1}^N$, where $y_n \in \{0,1\}$ are the values to be assigned to the class labels

Likelihood function*:

$$L(\mathbf{w}, b | \chi) = \prod_{n=1}^N P(y_n | \mathbf{x}_n) = \prod_{n=1}^N p_n^{y_n} (1 - p_n)^{1-y_n}$$

where the probability of \mathbf{x}_n belonging to the class with label:

$$y_n = 1 \text{ is } p_n = \frac{\exp(\mathbf{w}^\top \mathbf{x}_n)}{1 + \exp(\mathbf{w}^\top \mathbf{x}_n)} \quad y_n = 0 \text{ is } 1 - p_n = 1 - \frac{\exp(\mathbf{w}^\top \mathbf{x}_n)}{1 + \exp(\mathbf{w}^\top \mathbf{x}_n)}$$

- The likelihood measures the **goodness of fit** of a statistical model to a sample of data for given values of the parameters (\mathbf{w}, b)

*Bernoulli distribution

Logistic Regression: learning w

Cost function

- The log-likelihood $J(\cdot) = -\log(L(\cdot))$

$$E = J(\mathbf{W}) = - \sum_{n=1}^N (y_n \log(p_n) + (1 - y_n) \log(1 - p_n))$$

- No closed form solution
 - Iterative optimization methods are need to **minimized** the **cost function**
 - **gradient descendente**

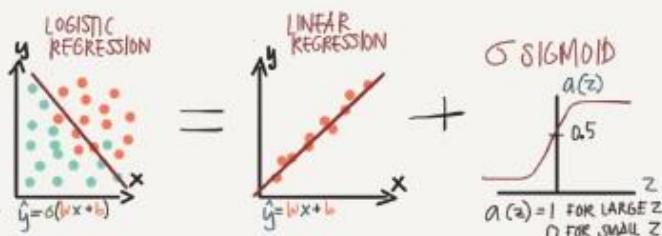
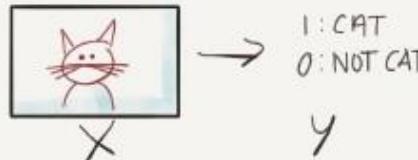
The parameter **updating values*** are

$$\Delta w^d = -\lambda \frac{\partial E}{\partial w_i} = \lambda \sum_{n=1}^N \left(\frac{y_n}{p_n} - \frac{1-y_n}{1-p_n} \right) p_n (1 - p_n) x_n^d$$

* Note that $x_n^d \equiv \langle x_d \rangle_n$, $w^d \equiv w_d$, bias $b \equiv w_0$, $x_0 = 1$

Logistic Regression

BINARY CLASSIFICATION



THE TASK IS TO LEARN w & b BUT HOW?

A: OPTIMIZE HOW GOOD THE GUESS IS BY
MINIMIZING THE DIFF BETWEEN GUESS (\hat{y})
AND TRUTH (y)

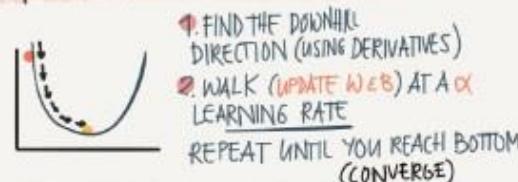
$$\text{LOSS} = \mathcal{L}(\hat{y}, y)$$

$$\text{COST} = J(w, b) = \frac{1}{m} \sum_{i=1}^m \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$$

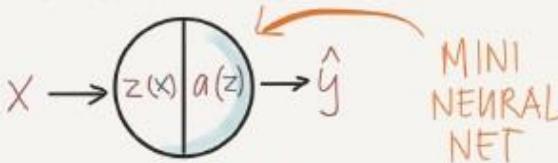
COST = LOSS FOR THE ENTIRE DATASET

LOGISTIC REGRESSION AS A NEURAL NET

FINDING THE MINIMUM
WITH GRADIENT DESCENT



PUTTING IT ALL TOGETHER



$$z(x) = w \cdot x + b$$

$$\hat{y} = a(z) = \text{SIGMOID}(z)$$

1. FORWARD • CALCULATE \hat{y}
 2. BACKWARD • GRADIENT DESCENT
+ UPDATE w & b
- REPEAT UNTIL IT CONVERGES

@TessFernandez

<https://storage.ning.com/topology/rest/1.0/file/get/2408482975?profile=original>

Contents

- Predictive modelling
- Linear classification models
 - Discriminant Functions
 - Least squares for classification
 - Fisher's Linear Discriminant (LDA)
 - Perceptron
 - Logistic Regression
 - Comparison
- Iterative optimization
- Summary

Comparison: Training/Learning in linear approaches

Different algorithms to the binary classification problem

- **FDF and LDA** : Algebraic approach or **optimized criterium**
- **Perceptron**: perceptron learning algorithm or **optimized criterium**
- **Logistic Regression**: defining a cost function (based on likelihood) and **optimization** with iterative optimization methods

Optimized criterium for FDF, LDA and Perceptron

- **Optimization** of the $J(\cdot)$ / error with iterative optimization methods
 - gradient descendent

Comparison: Training/Learning in linear approaches

FDF/LDA:

- fails when the discriminatory information is not in the mean, but rather in the variance of the data

Logistic regression:

- the non-linearity of $f(\cdot)$ gives more flexibility → it can limit the effects of outliers
- w can become unstable:
 - If classes are well-separated
 - Data size (#objects) is small

Perceptron:

- **decision boundary** depends on the order of the learning examples

Comparison: Training/Learning in linear approaches

Logistic Regression & Perceptron: only for binary classification

- Multiclass logistic regression

FDF/LDA vs Logistic regression:

When the normality assumption of data (approximately) holds, its expected that FDF/LDA outperforms Logistic regression

Perceptron vs Logistic regression:

Perceptron only updates weights when misclassification

Contents

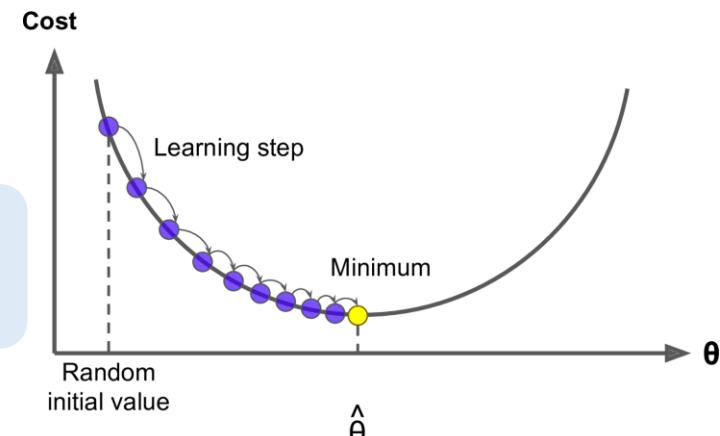
- Predictive modelling
- Linear classification models
- **Iterative optimization**
- Summary

Iterative optimization methods: Gradient-based learning

- Iterative optimization algorithm for finding the minimum of a function, typically a convex **cost/loss** function
- Mostly used when there is no **closed form solution** (the parameters can not be calculated analytically, e.g., using linear algebra)
- Used to fit linear classifiers and regressors under convex loss functions
- For a function $F(x)$ at a point a , $F(x)$ decreases fastest if we go in the direction of the negative gradient of a

$$a_n = a_{n-1} - \lambda \nabla F(a_{n-1})$$

When the **gradient** is zero, we arrive at the local minimum



Iterative optimization methods: Gradient-based learning

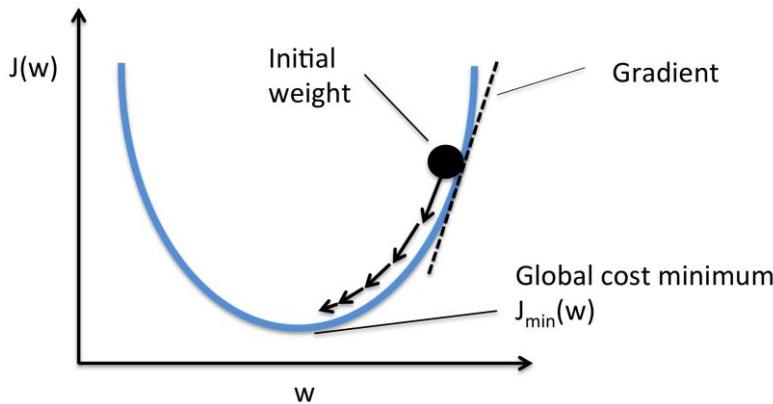
- Cost/loss function $E = J(\mathbf{w})$
- Vector gradient

$$\nabla E = \left(\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \dots, \frac{\partial E}{\partial w_D} \right)$$

- each **weight** is updated according to

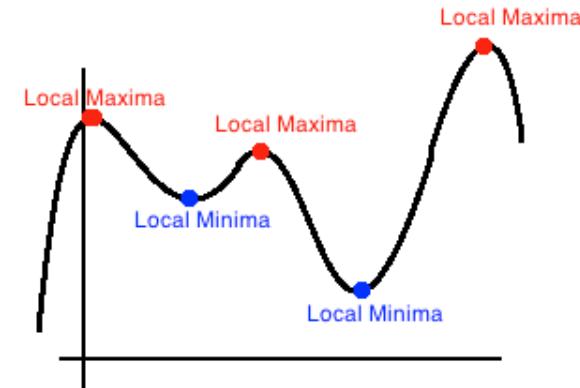
$$\Delta w_i = -\lambda \frac{\partial E}{\partial w_i}$$

Linear activation function



The learning rate λ is the step length in negative gradient direction

Non-linear error function



drawback (to live with): the learning can converge to any local minimum

Iterative optimization methods: Batch vs Stochastic Gradient Descent

Global Cost/loss function $E = \sum_n E^{(n)}$

The gradient can be estimated with

- **Batch Gradient Descent:** with the total (or average) error in training set E

$$\mathbf{w}^{(i)} = \mathbf{w}^{(i-1)} - \lambda \nabla E$$

- the batch size is the number of samples
- **Online/Stochastic Gradient Descent:** with the error of a single training example $E^{(n)}$

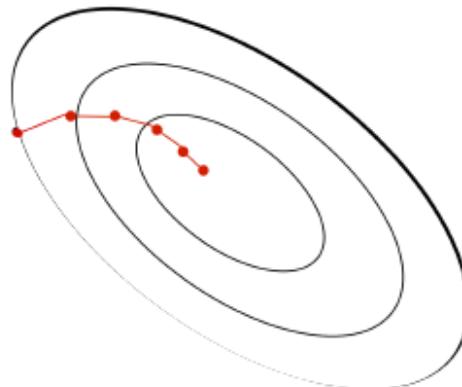
$$\mathbf{w}^{(i)} = \mathbf{w}^{(i-1)} - \lambda \nabla E^{(n)}$$

- instead of calculating the gradient of the full error function (which involves using the full training set), we update the weights one example at a time

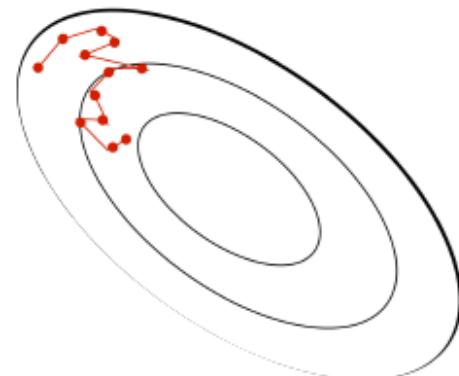
Both are more effective to escape from local minima

Iterative optimization methods: Batch vs Stochastic Gradient Descent

- Batch gradient descent moves directly downhill
- Batch is impractical for large data sets
- SGD takes steps in a noisy direction, but moves downhill on average
- Randomly sampling the training set → SGD converges to Batch solution
- Mixed Strategy: Mini-Batch



batch gradient descent



stochastic gradient descent

Contents

- Predictive modelling
- Linear classification models
- Iterative optimization
- **Summary**

Summary

- Predictive modelling
 - Predictive problems
 - Models approaches for classification
- Linear classification models
 - Discriminant Functions
 - Least squares for classification
 - Fisher's Linear Discriminant (LDA)
 - Perceptron
 - Logistic Regression
 - Comparison
- Iterative optimization
 - Batch Gradient Descent
 - Stochastic Gradient Descent

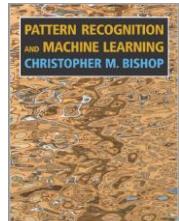
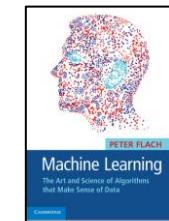
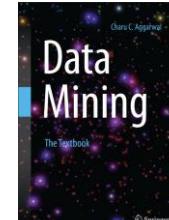
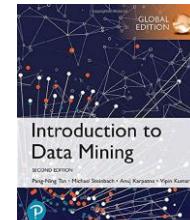
Bibliography

Introduction to Data Mining, Pang-Ning Tan, Michael Steinbach, Anuj Karpatne, Vipin Kumar, *Pearson*, 2019 (chap 3.1, 3.2, 6.1, 6.6, 6.7)

Data Mining, the Textbook, Charu C. Aggarwal, *Springer*, 2015 (chap 10.2.1.4, 10.5.2, 10.7.1)

Machine Learning: The Art and Science of Algorithms That Make Sense of Data, P. Flach, *Cambridge University Press*, 2012 (ch 1, 2, 7.1, 7.4)

Pattern Recognition and Machine Learning, C. Bishop, *Springer*, 2007 (ch 4)



Data Mining

Predictive Modelling

Support Vector Machines

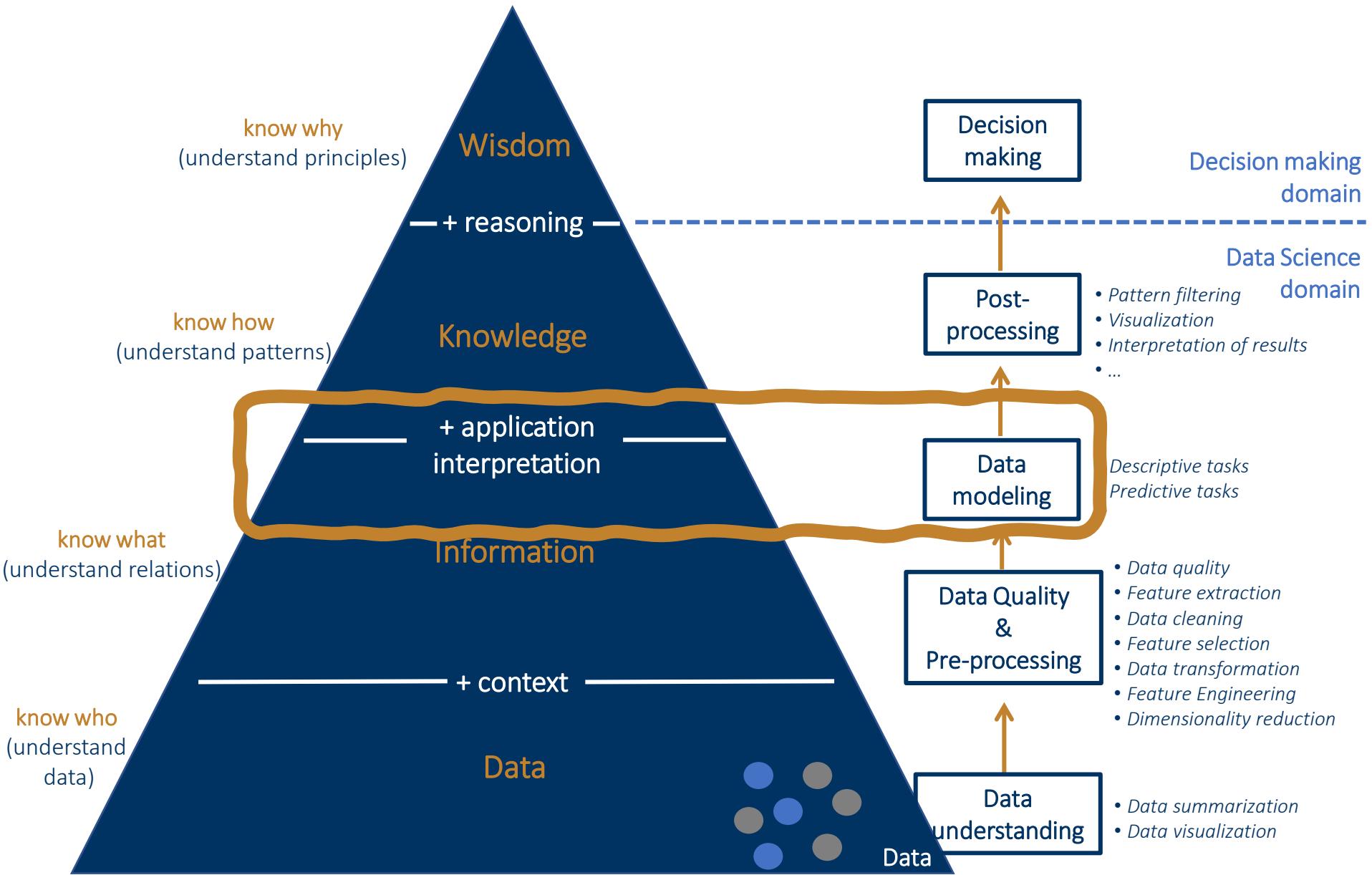
Raquel Sebastião

Departamento de Eletrónica, Telecomunicações e Informática

Universidade de Aveiro

raquel.sebastiao@ua.pt

2022/2023



Contents

- Support Vector Machines (SVM)
 - Support vectors
- Linear support vector machines
- Nonlinear support vector machines
- Multi-class SVM
- Summary

Prediction Models – approaches

Geometric approaches

- Distance-based: kNN
- Linear models: Fisher's linear discriminant, perceptron, logistic regression, SVM (w. linear kernel)

Probabilistic approaches

- naive Bayes, logistic regression

Logical approaches

- classification or regression trees, rules

Optimization approaches

- neural networks, SVM

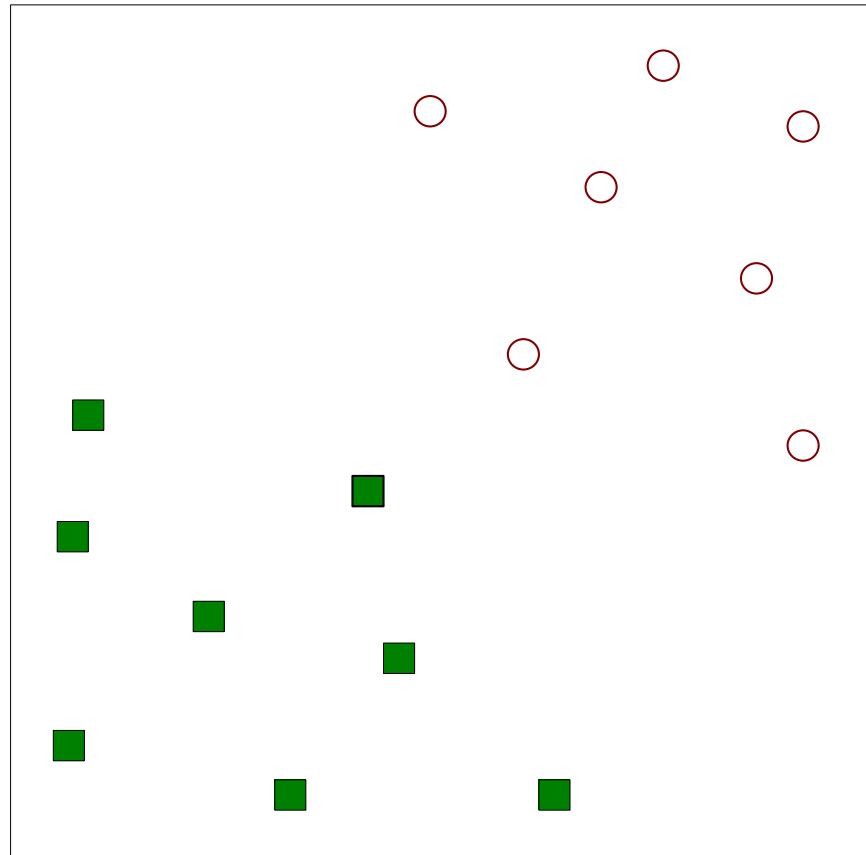
Sets of models (ensembles)

- random forests, adaBoost

Support Vector Machines

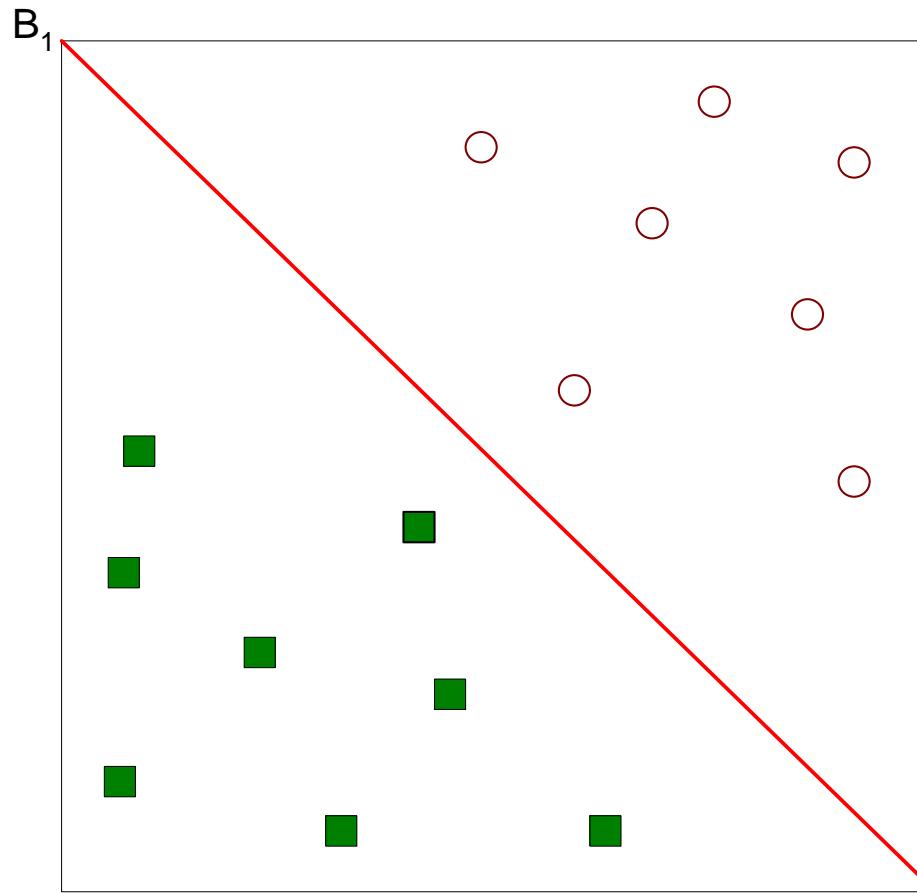
- “Relative recent”
 - introduced in 1992 (@COLT-92 conf)
- Gave origin to a new class of algorithms named kernel machines
- SVMs have been applied with success in a wide range of areas:
 - Bioinformatics
 - text mining
 - hand-written character recognition
 - Biometric recognition

Support Vector Machines



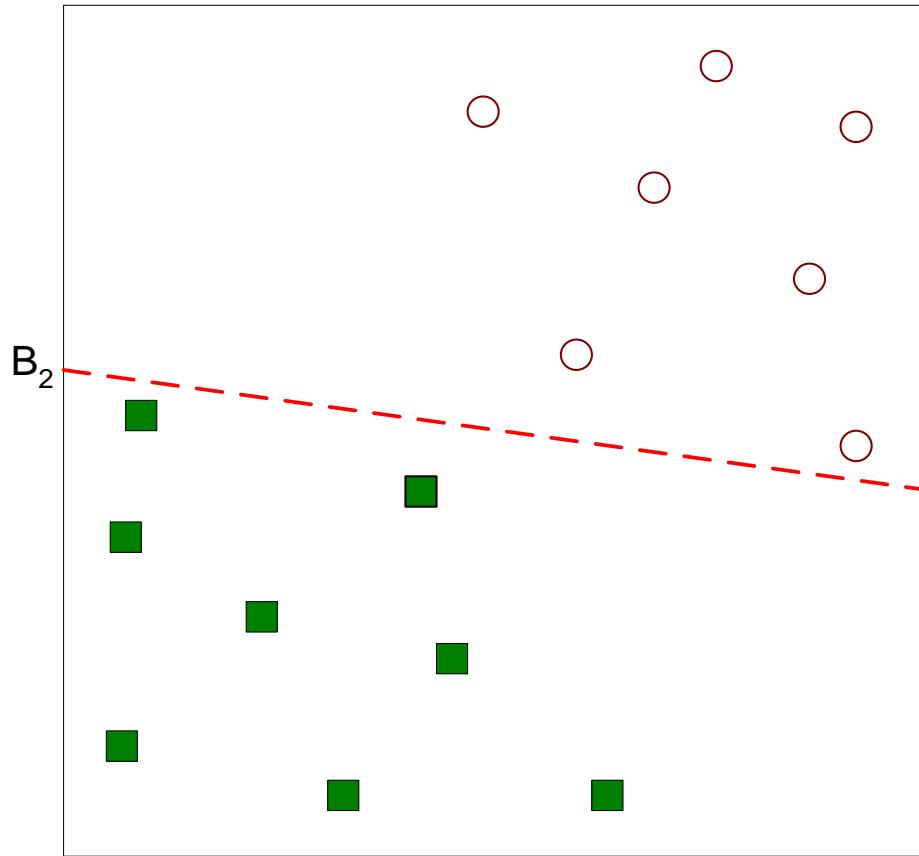
- Find a linear hyperplane (decision boundary) that will separate the data

Support Vector Machines



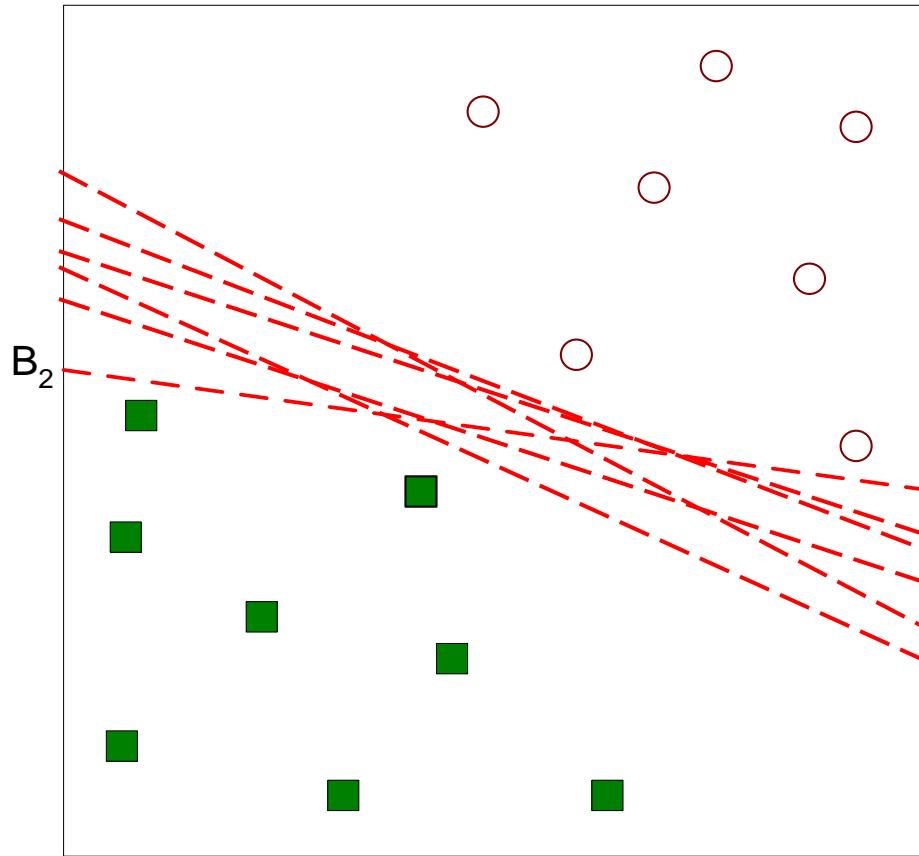
- One Possible Solution

Support Vector Machines



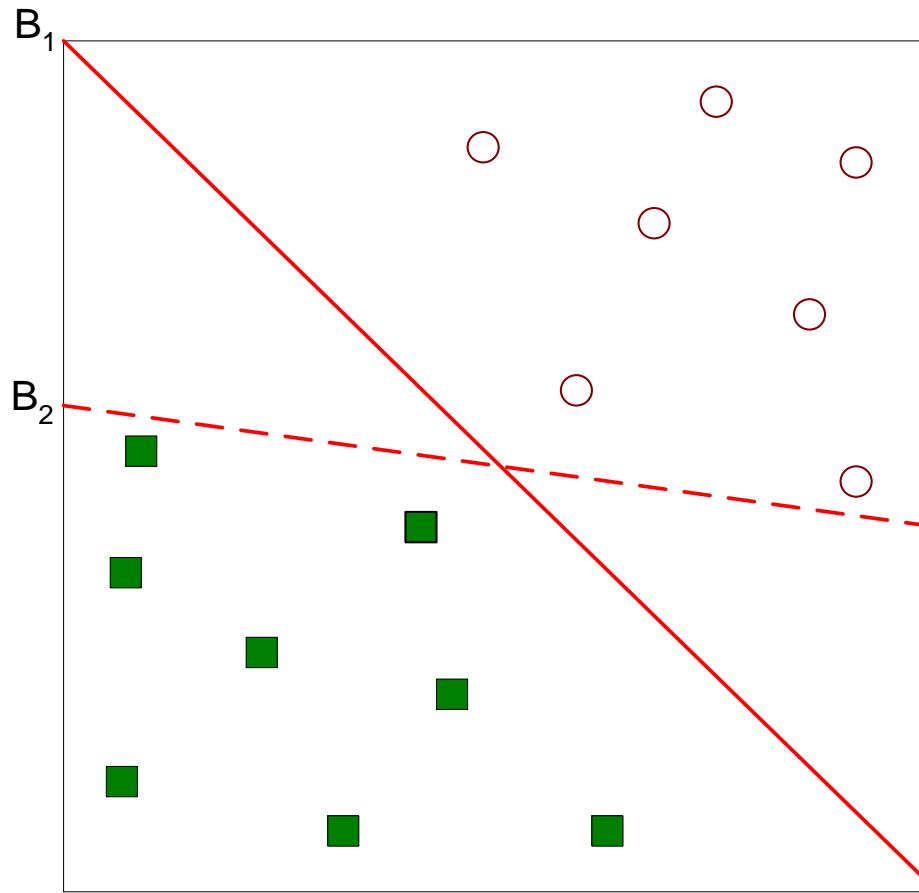
- Another possible solution

Support Vector Machines



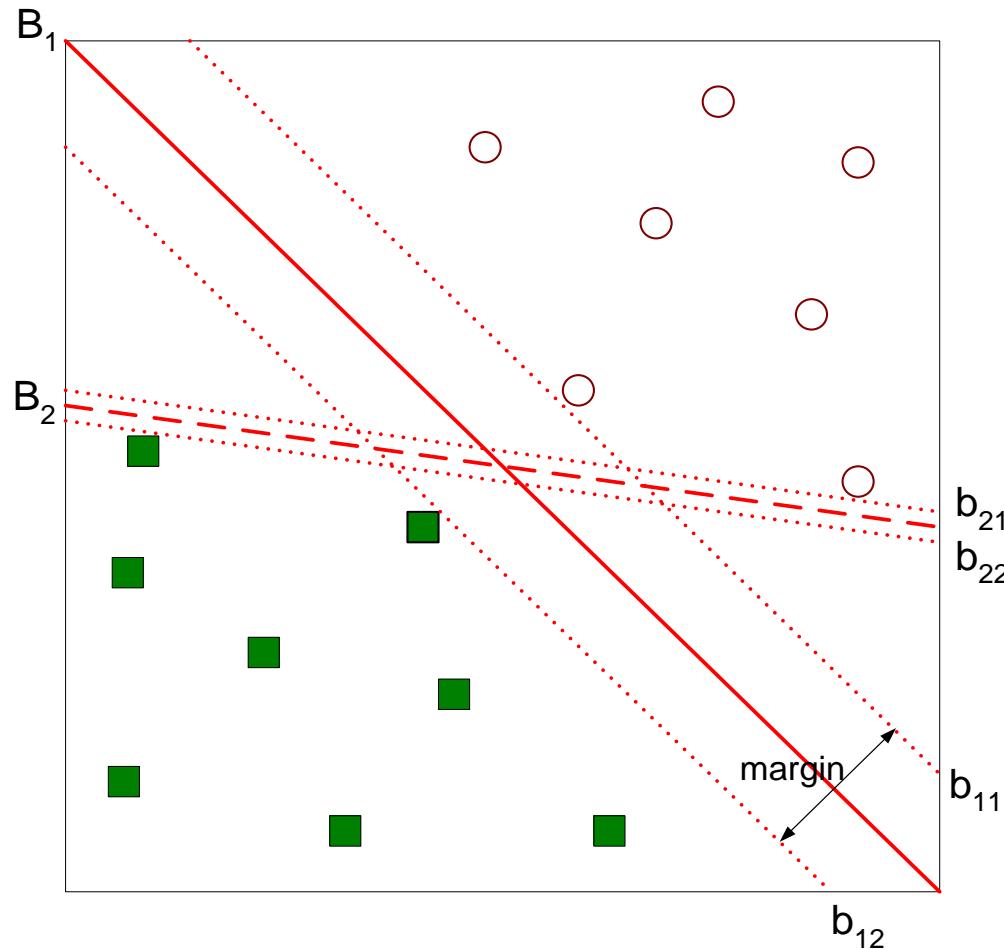
- Other possible solutions

Support Vector Machines



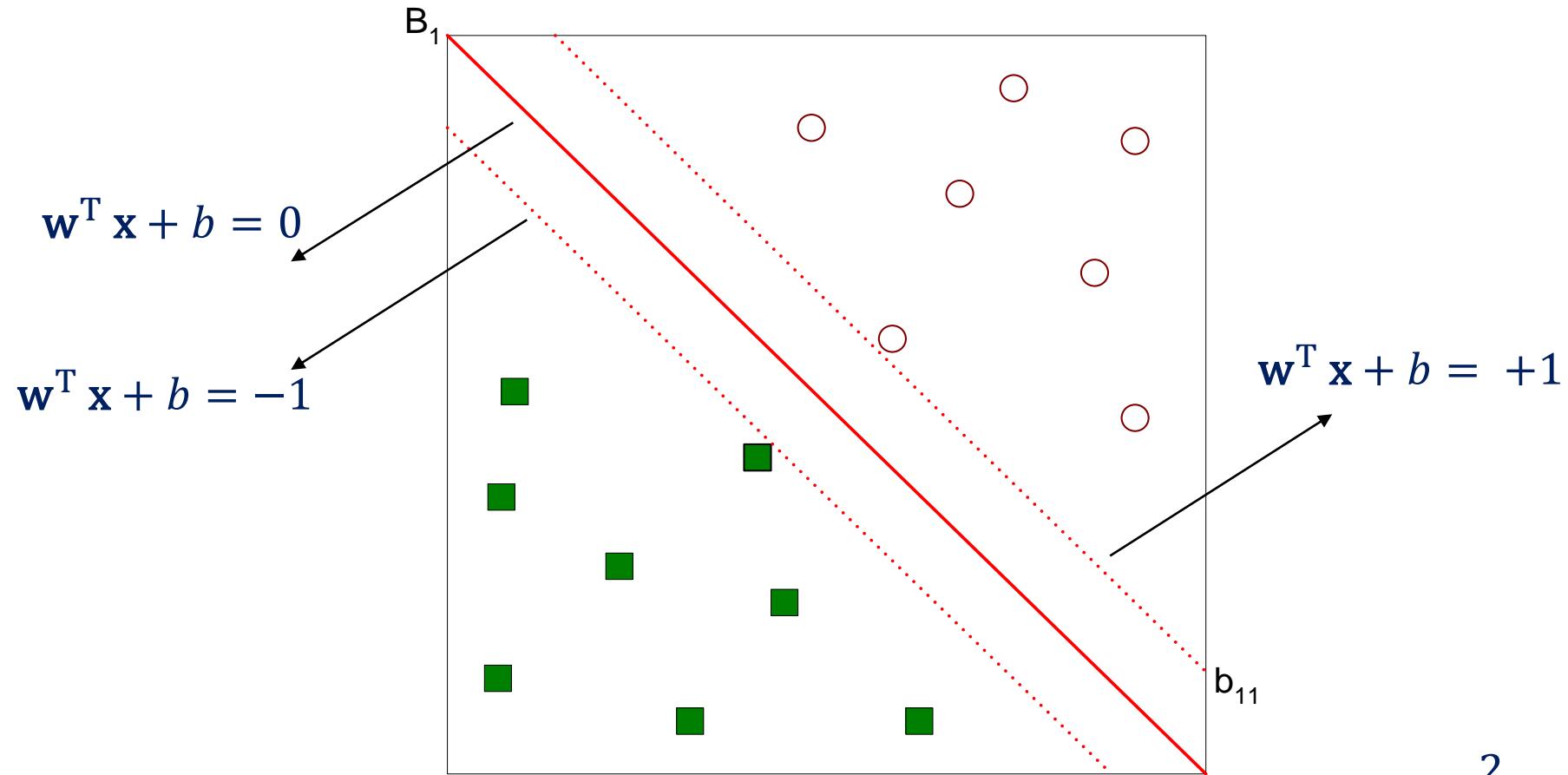
- Which one is **better**? B_1 or B_2 ?
- How to define better?

Support Vector Machines



- Find hyperplane **maximizes** the margin => B_1 is better than B_2

Support Vector Machines



$$g(\mathbf{x}) = \begin{cases} 1 & \text{if } w^T \mathbf{x} + b \geq 1 \\ -1 & \text{if } w^T \mathbf{x} + b \leq -1 \end{cases}$$

$$\text{Margin} = \frac{2}{\|w\|}$$

Contents

- Support Vector Machines (SVM)
 - Support vectors
- **Linear support vector machines**
 - Learning linear SVM
 - Optimization problem and Dual optimization problem
 - Example
 - Soft margin SVM
- Nonlinear support vector machines
- Multi-class SVM
- Summary

Linear Support Vector Machines

- Linear model:

$$g(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w}^T \mathbf{x} + b \geq 1 \\ -1 & \text{if } \mathbf{w}^T \mathbf{x} + b \leq -1 \end{cases}$$

- Target labels = {-1,1}

- Learning the model is equivalent to determining the values of \mathbf{w} and b
 - How to find \mathbf{w} and b from training data?

Linear Support Vector Machines

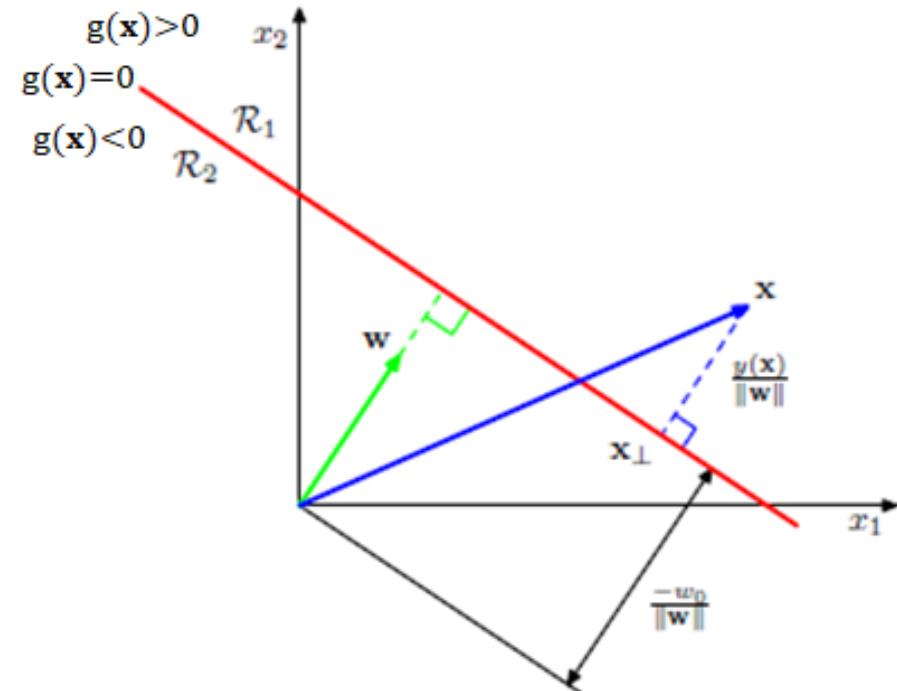
The decision rule is a linear discriminant function

$$g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$$

How to find \mathbf{w} and b from training data?

Separating hyperplane

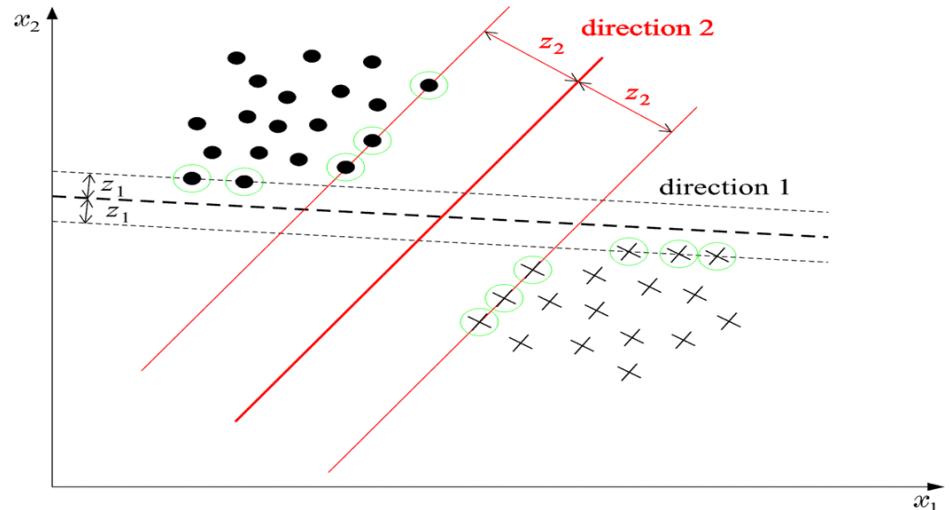
- Direction in space: \mathbf{w}
- Position in space: b
- Distance of an object \mathbf{x} (with label y) to the hyperplane: $\frac{y g(\mathbf{x})}{\|\mathbf{w}\|}$



Linear Support Vector Machines

Optimal hyperplane: $g(\mathbf{x}) = 0$

- for EACH possible direction \mathbf{w} :
 - choose the hyperplane that leaves the SAME distance from the nearest points from each class
 - The margin is twice this distance
- $g(\mathbf{x}) = 1$ and $g(\mathbf{x}) = -1$ define two parallel hyperplane to the optimal hyperplane $g(\mathbf{x}) = 0$
 - Cases that fall on the hyperplanes are the support vectors ($\mathbf{w}^T \mathbf{x} + b = \pm 1$)
 - Removing all other cases would not change the solution!
- The optimal hyperplane classifier of a support vector machine is unique



Contents

- Support Vector Machines (SVM)
 - Support vectors
- Linear support vector machines
 - Learning linear SVM
 - Optimization problem and Dual optimization problem
 - Example
 - Soft margin SVM
- Nonlinear support vector machines
- Multi-class SVM
- Summary

Learning linear SVM

Given a **training data set** $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$, where each **object** is represented by a $D+1$ -tuple (D -dim) feature vector $\mathbf{x}_i \in \mathbb{R}^D$ and the corresponding **label** $y_i \in Y$

- **Goal:** maximize Margin $= \frac{2}{\|\mathbf{w}\|}$ (Largest margin \rightarrow better generalization)

- Which is equivalent to minimizing: $L(\mathbf{w}) = \frac{\|\mathbf{w}\|^2}{2}$

- subject to: \mathbf{w} and b such that:

$$y_i = \begin{cases} 1 & \text{if } \mathbf{w}^T \mathbf{x}_i + b \geq 1 \\ -1 & \text{if } \mathbf{w}^T \mathbf{x}_i + b \leq -1 \end{cases}$$

- Which is equivalent to: $y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1, \forall \{(\mathbf{x}_i, y_i)\}$

Contents

- Support Vector Machines (SVM)
 - Support vectors
- Linear support vector machines
 - Learning linear SVM
 - Optimization problem and Dual optimization problem
 - Example
 - Soft margin SVM
- Nonlinear support vector machines
- Multi-class SVM
- Summary

Learning linear SVM: Optimization problem

Maximum Margin Hyperplane

- The solution is achieved with
 - minimizing : $L(\mathbf{w}) = \frac{\|\mathbf{w}\|^2}{2}$
 - subject to: $y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1, \forall \{(\mathbf{x}_i, y_i)\}$
- This is a constrained optimization problem
 - Solve it using Lagrange multiplier method

Learning linear SVM: Optimization problem

Maximum Margin Hyperplane

- **Minimization** is achieved by writing the **Lagrangian primal problem**

$$L = \frac{\|\mathbf{w}\|^2}{2} - \sum_{i=1}^N \lambda_i (y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1)$$

- Calculating

$$\frac{\partial L}{\partial \mathbf{w}} = 0 \Rightarrow \boxed{\mathbf{w} = \sum_{i=1}^N \lambda_i y_i \mathbf{x}_i}$$

$$\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{i=1}^N \lambda_i y_i = 0$$

and substituting in L the **dual optimization problem** $L_d(\lambda)$ is obtained

Learning linear SVM:

Dual optimization problem

Dual optimization problem: data manipulations are dot products

- Training by **maximizing**:

$$L_d(\lambda) = \sum_{i=1}^N \lambda_i - \frac{1}{2} \sum_{i,j=1}^N \lambda_i \lambda_j y_i y_j ((\mathbf{x}_i)^T \mathbf{x}_j)$$

- Subjected to
 - Lagrangians $\lambda_i \geq 0$
 - $\sum_{i=1}^N \lambda_i y_i = 0$

- **outputs**

- The **Lagrangians** λ_i computed for all the examples in the training data set
 - If $\lambda_i = 0$ the i -th example \mathbf{x}_i is not relevant
 - If $\lambda_i \neq 0$ the i -th example \mathbf{x}_i is a **support vector**

Learning linear SVM: Dual optimization problem

After learning the Lagrangians:

- Compute $\mathbf{w} = \sum_{i=1}^N \lambda_i y_i \mathbf{x}_i^*$
 - when $\lambda_i = 0$ the i -th example \mathbf{x}_i is not relevant (it does not contribute to the sum)
 - when $\lambda_i \neq 0$ the corresponding i -th example \mathbf{x}_i is a **support vector**
 - lies along the hyperplanes parallel to the decision hyperplane (linearly separable problem): $\mathbf{w}^T \mathbf{x} + b = \pm 1$
- b is computed using **support vectors**

Decision boundary depends only on support vectors

$$g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b = 0$$

* Note that only the support vectors contribute to compute \mathbf{w}

Learning linear SVM: Dual optimization problem

Testing

Applying **dual form** of linear classifier, the label of object \mathbf{z} is

$$g(\mathbf{z}) = \sum_{i,j=1}^{K_s} \lambda_i y_i ((\mathbf{x}_i)^T \mathbf{z}) + b \Rightarrow \begin{cases} g(\mathbf{z}) > 0, \mathbf{z} \in \omega_1 \\ g(\mathbf{z}) < 0, \mathbf{z} \in \omega_2 \end{cases}$$

- K_s : the number **support vectors**

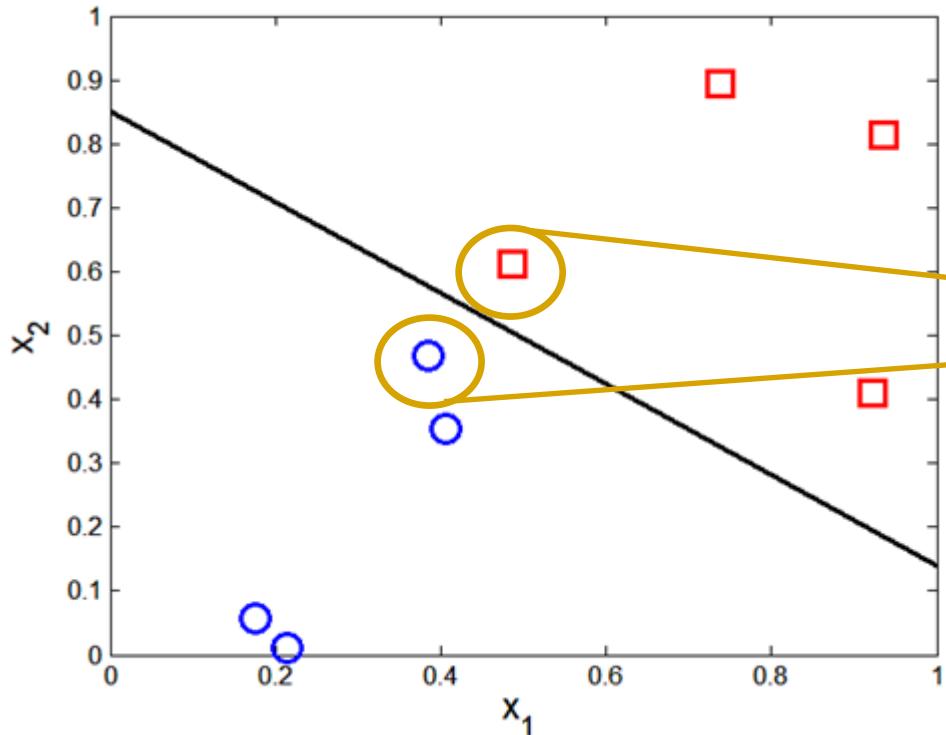
To apply **dual form** during testing phase it is needed:

- the support vectors to perform $(\mathbf{x}_i)^T \mathbf{z}$ and corresponding labels y_i
- the complexity of testing phase is dependent on the **number of support vectors**

Contents

- Support Vector Machines (SVM)
 - Support vectors
- Linear support vector machines
 - Learning linear SVM
 - Optimization problem and Dual optimization problem
 - Example
 - Soft margin SVM
- Nonlinear support vector machines
- Multi-class SVM
- Summary

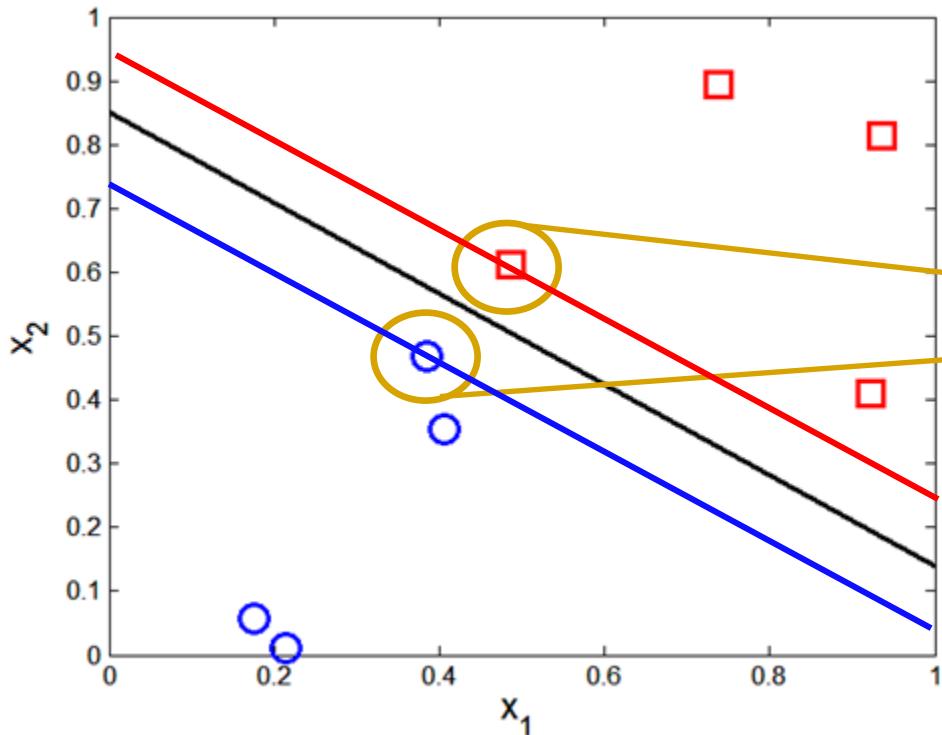
Example of linear SVM



x_1	x_2	y	λ
0.3858	0.4687	1	65.5261
0.4871	0.611	-1	65.5261
0.9218	0.4103	-1	0
0.7382	0.8936	-1	0
0.1763	0.0579	1	0
0.4057	0.3529	1	0
0.9355	0.8132	-1	0
0.2146	0.0099	1	0

How to find \mathbf{w} and b from training data?

Example of linear SVM



How to find \mathbf{w} and b from training data?

x1	x2	y	λ
0.3858	0.4687	1	65.5261
0.4871	0.611	-1	65.5261
0.9218	0.4103	-1	0
0.7382	0.8936	-1	0
0.1763	0.0579	1	0
0.4057	0.3529	1	0
0.9355	0.8132	-1	0
0.2146	0.0099	1	0

Example of linear SVM

How to find \mathbf{w} and b from training data?

$$\mathbf{w} = \sum_{i=1}^N \lambda_i y_i \mathbf{x}_i$$

$$\mathbf{w} = \sum_{i=1}^{K_s} \lambda_i y_i \mathbf{x}_i$$

$$\mathbf{w} = 65.5261 \times 1 \times [0.3858 \ 0.4687] + 65.5261 \times (-1) \times [0.4871 \ 0.611] =$$

$$\mathbf{w} = [-6.6378 \ -9.3244]$$

Support vectors

x1	x2	y	λ
0.3858	0.4687	1	65.5261
0.4871	0.611	-1	65.5261
0.9218	0.4103	-1	0
0.7382	0.8936	-1	0
0.1763	0.0579	1	0
0.4057	0.3529	1	0
0.9355	0.8132	-1	0
0.2146	0.0099	1	0

Example of linear SVM

How to find \mathbf{w} and b from training data?

At the margins: $\mathbf{w}^T \mathbf{x} + b = \pm 1$

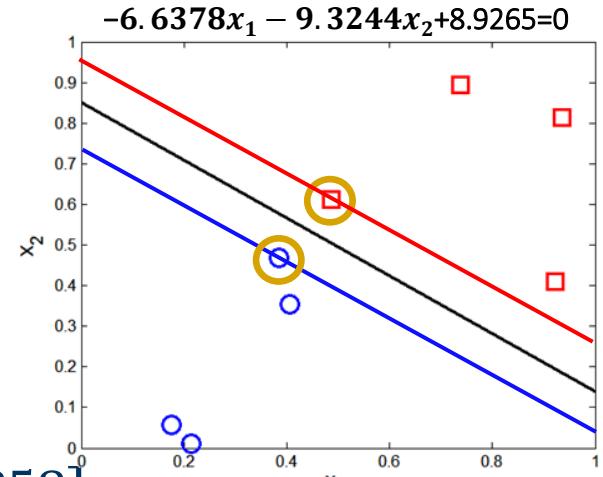
Red hyperplane ($\mathbf{w}^T \mathbf{x} + b = +1$)

$$b = +1 - \mathbf{w}^T \mathbf{x} = +1 - [-6.6378 \ -9.3244] \begin{bmatrix} 0.3858 \\ 0.4687 \end{bmatrix} = 7.9226$$

Blue hyperplane ($\mathbf{w}^T \mathbf{x} + b = -1$)

$$b = -1 - \mathbf{w}^T \mathbf{x} = -1 - [-6.6378 \ -9.3244] \begin{bmatrix} 0.4871 \\ 0.611 \end{bmatrix} = 7.9305$$

$$b = \frac{(7.9226 + 7.9305)}{2} = 7.9265$$



x1	x2	y	λ
0.3858	0.4687	1	65.5261
0.4871	0.611	-1	65.5261
0.9210	0.4103	-1	0
0.7382	0.8936	-1	0
0.1763	0.0579	1	0
0.4057	0.3529	1	0
0.9355	0.8132	-1	0
0.2146	0.0099	1	0

(it is numerically safer to take the mean value of b resulting from all support vectors)

Contents

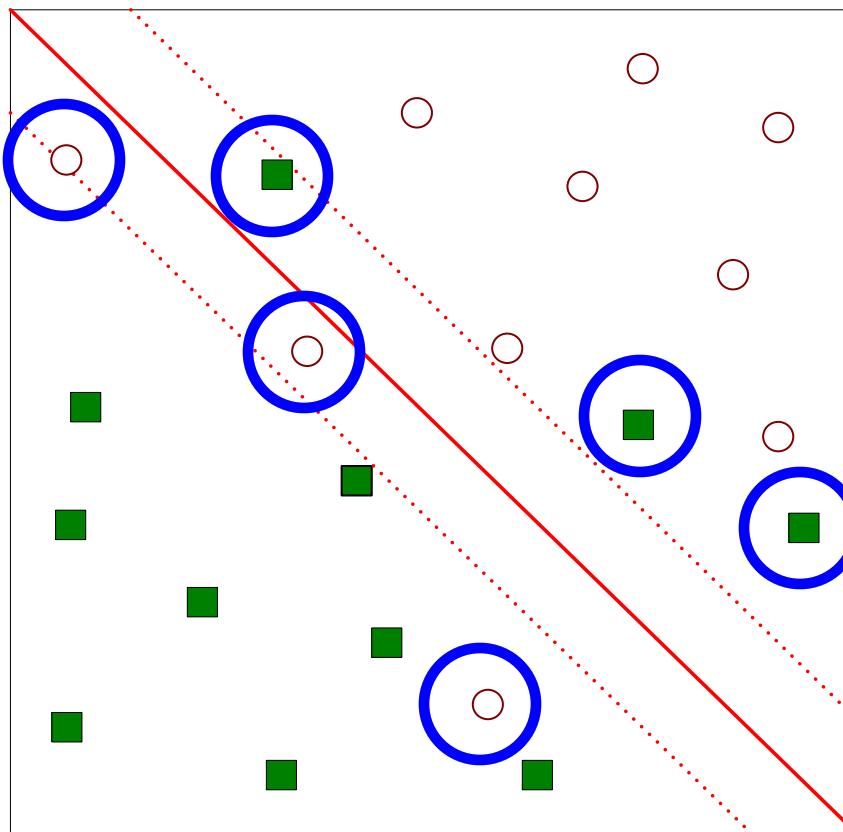
- Support Vector Machines (SVM)
 - Support vectors
- Linear support vector machines
 - Learning linear SVM
 - Optimization problem and Dual optimization problem
 - Example
 - Soft margin SVM
- Nonlinear support vector machines
- Multi-class SVM
- Summary

Learning linear SVM

not linearly separable data

- What if the problem is **not** linearly separable?

Soft-margin SVM



Learning linear SVM

not linearly separable data

- What if the problem is **not linearly separable?**

Soft-margin SVM

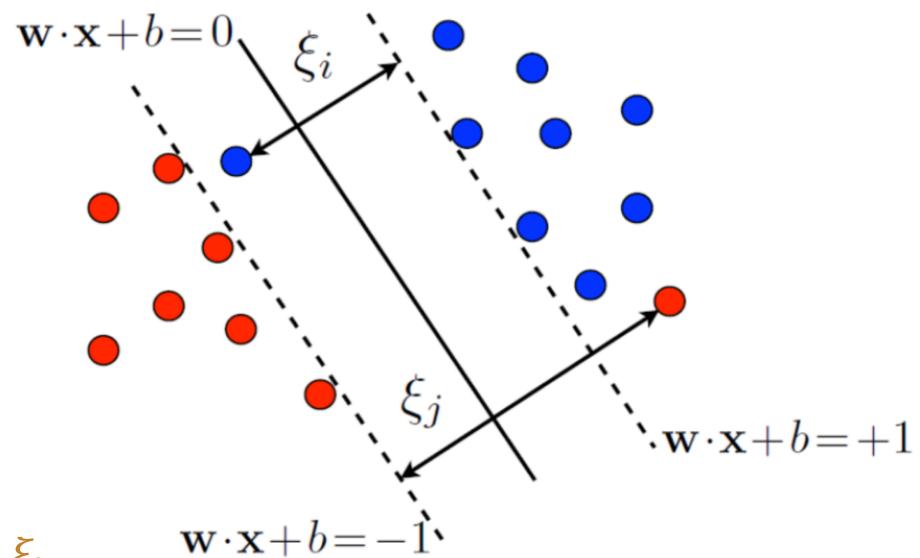
- Introduce **slack variables** to tolerate some misclassification errors controlled by the parameter **C** (regularization term)

- Objective: minimize

$$L(w) = \frac{\|w\|^2}{2} + C \left(\sum_{i=1}^N \xi_i^k \right)$$

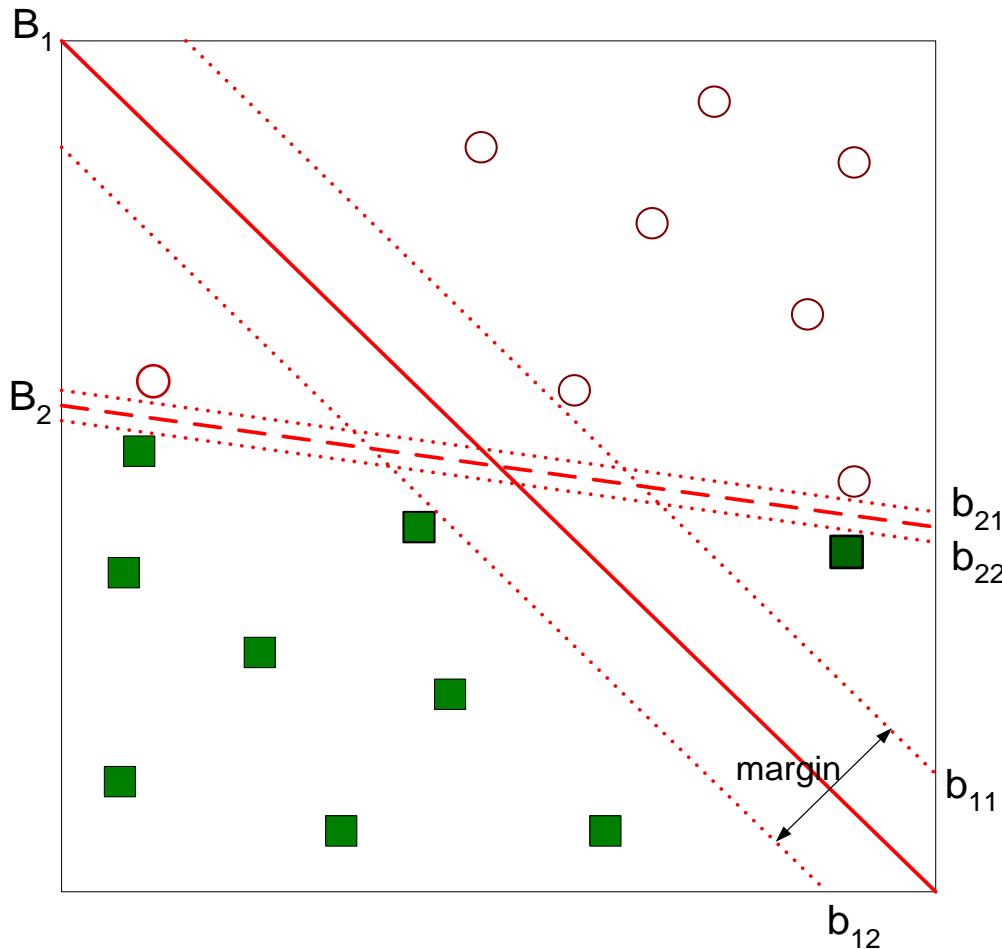
- Goal: find w , b and ξ , given that:

$$y_i = \begin{cases} 1 & \text{if } w^T x_i + b \geq 1 - \xi_i \\ -1 & \text{if } w^T x_i + b \leq -1 + \xi_i \end{cases}$$



Learning linear SVM

not linearly separable data



- Find the hyperplane that optimizes both margins (B_1 and B_2)

Linear Support Vector Machines

Hard Margin SVMs: Linearly separable data

- works well when data is linearly separable
- on real-world data this is hardly the case
- does not take into account presence of noise

Soft Margin SVMs: Not linearly separable data

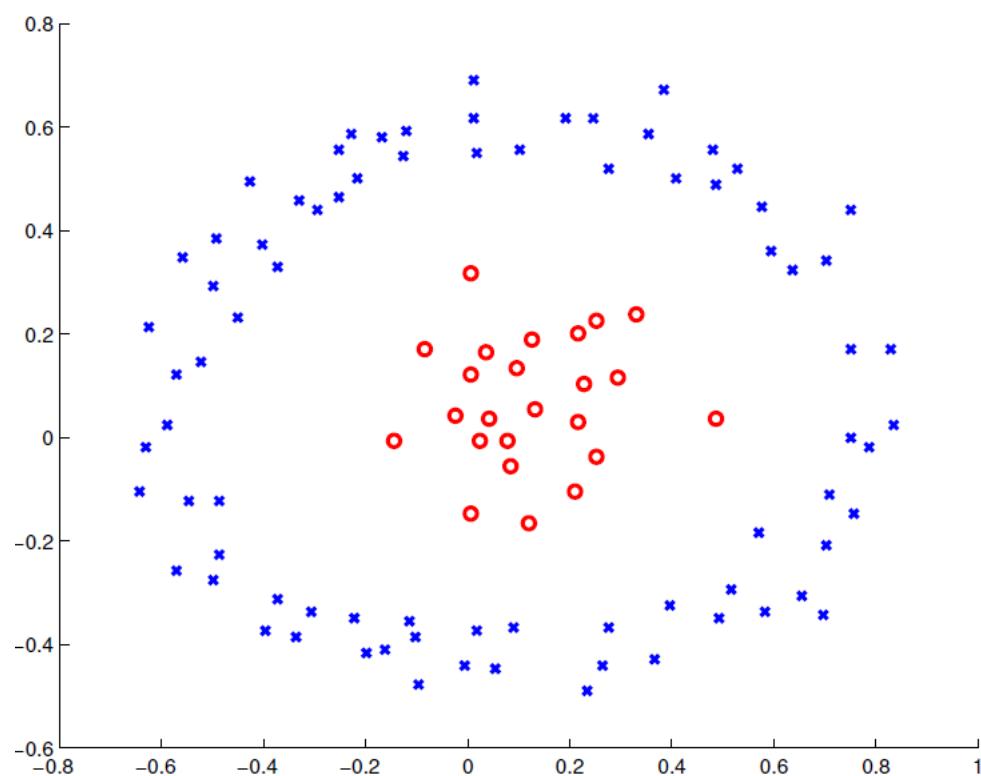
- it tolerates some misclassification
- errors controlled by a parameter C (regularization term)
- introduces slack variables for each example

Contents

- Support Vector Machines (SVM)
 - Support vectors
- Linear support vector machines
- Nonlinear support vector machines
 - Learning nonlinear SVM and the Kernel trick
 - Example
- Multi-class SVM
- Summary

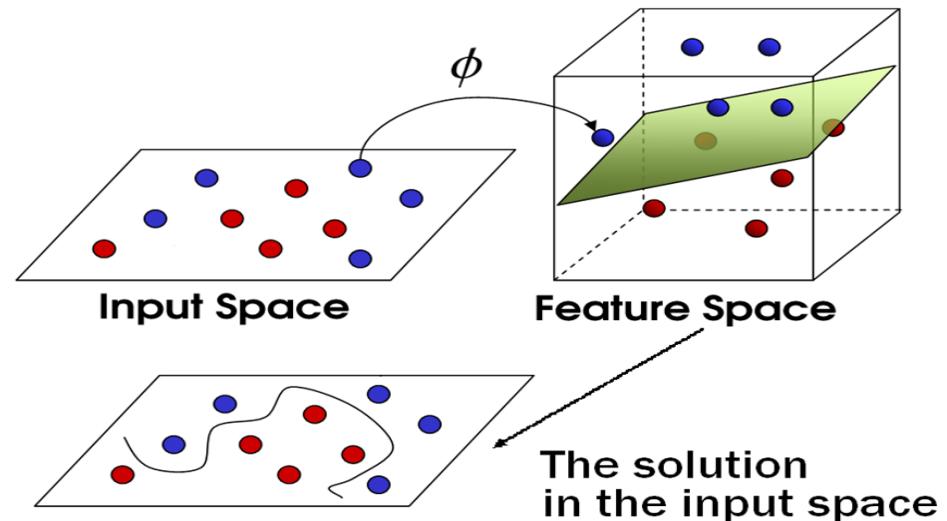
Nonlinear Support Vector Machines

- What if the decision boundary is **not** linear?



Nonlinear Support Vector Machines

- Most real world problems have inherent nonlinearity
- SVMs solve this by “moving” into an **extended input space** where classes are already linearly separable
- This means the **maximum margin hyperplane** needs to be found on this new very high dimension space



Contents

- Support Vector Machines (SVM)
 - Support vectors
- Linear support vector machines
- Nonlinear support vector machines
 - Learning nonlinear SVM and the Kernel trick
 - Example
- Multi-class SVM
- Summary

Learning nonlinear SVM

- Transform data into higher dimensional space with $\Phi(\mathbf{x})$
 - Such that classes are linearly separable
- Same optimization problem $\min_{\mathbf{w}} \frac{\|\mathbf{w}\|^2}{2}$, but involving $\Phi(\mathbf{x})$ instead of \mathbf{x}
- Computations involve dot product $\Phi(\mathbf{x}_i) \Phi(\mathbf{x}_j)$
 - Solution to the optimization equation involves dot products between feature vectors, \mathbf{x}_i and \mathbf{x}_j , that are computationally heavy on high-dimensional spaces
 - Calculate the image of $\Phi(\mathbf{x})$ of each input vector \mathbf{x} and then do the dot product can be quite expensive
- The kernel function defined as $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \Phi(\mathbf{x}_j)$ performs simultaneously the mapping and dot product



Kernel Trick

Learning nonlinear SVM

Kernel Trick

- The result of complex calculations in higher dimensional space is equivalent to the result of applying certain functions (**kernel functions**) in the space of the original variables
 - replace the complex dot products by these simpler and efficient calculations
- The kernel function takes as inputs vectors in the original space and **returns the dot product of the vectors in the higher dimensional space**
 - perform operations in the **original space** (without a feature transformation!)
- Using kernels, we do not need to embed the data into the higher dimensional space explicitly!
- instead of calculating the dot products in a high dimensional space, take advantage of $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \Phi(\mathbf{x}_j)$
- use a **linear optimization solution** to solve a non-linear problem

Learning nonlinear SVM

Dual optimization problem: the same set of equations (but involve $\Phi(\mathbf{x})$ instead of \mathbf{x})

$$L_d(\lambda) = \sum_{i=1}^N \lambda_i - \frac{1}{2} \sum_{i,j=1}^N \lambda_i \lambda_j y_i y_j ((\Phi(\mathbf{x}_i))^T \Phi(\mathbf{x}_j))$$

$$L_d(\lambda) = \sum_{i=1}^N \lambda_i - \frac{1}{2} \sum_{i,j=1}^N \lambda_i \lambda_j y_i y_j (\mathbb{K}(\mathbf{x}_i, \mathbf{x}_j))$$

Kernel Trick

- Subjected to

$$\mathbf{w} = \sum_{i=1}^N \lambda_i y_i \Phi(\mathbf{x}_i)$$

- SVM dual form of decision rule

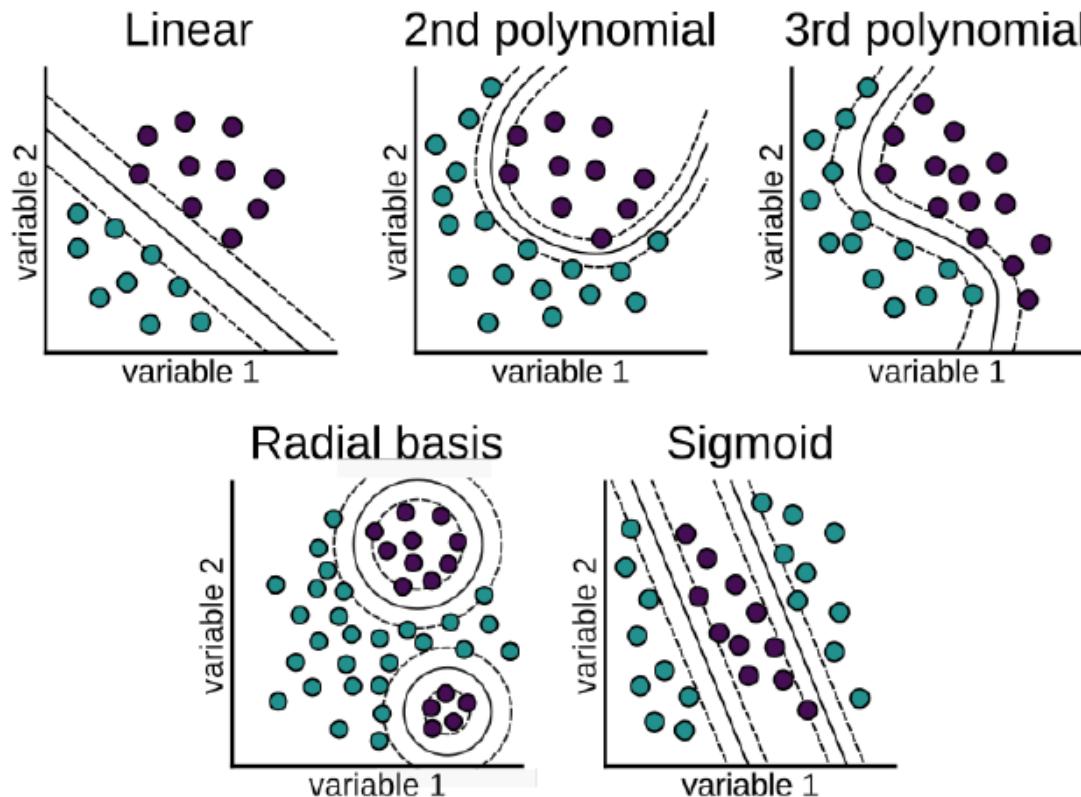
$$g(\mathbf{z}) = \mathbf{w}^T \mathbf{z} + b = \sum_{i,j=1}^{K_s} \lambda_i y_i (\Phi(\mathbf{x}_i))^T \Phi(\mathbf{z}) + b \Rightarrow \begin{cases} g(\mathbf{z}) > 0, \mathbf{z} \in \omega_1 \\ g(\mathbf{z}) < 0, \mathbf{z} \in \omega_2 \end{cases}$$

Learning nonlinear SVM

- Advantages of using kernel:
 - Don't have to know the mapping function Φ
 - Computing dot product $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$ in the original space avoids curse of dimensionality
- Not all functions can be kernels
 - Must make sure there is a corresponding Φ in some high-dimensional space

Learning nonlinear SVM

Examples of different kernel functions:



Contents

- Support Vector Machines (SVM)
 - Support vectors
- Linear support vector machines
- Nonlinear support vector machines
 - Learning nonlinear SVM and the Kernel trick
 - Example
- Multi-class SVM
- Summary

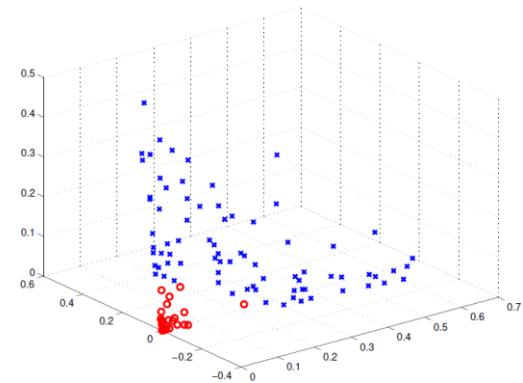
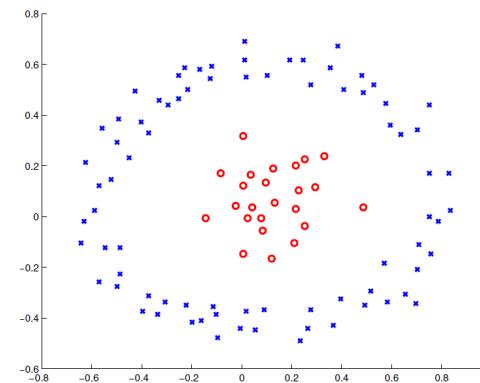
Nonlinear Support Vector Machines: example

Transform data into higher dimensional space (from 2D to 3D)

$$\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2] \rightarrow \Phi(\mathbf{x}) = [\mathbf{x}_1^2 \quad \sqrt{2}\mathbf{x}_1\mathbf{x}_2 \quad \mathbf{x}_2^2]$$

- linear decision boundary with SVM (for example..)
 - Linear hyperplane can be used to separate the instances in the transformed space
- The learned hyperplane can then be projected back to the original attribute space
 - nonlinear decision boundary
- The dot product in the 3D space using the polynomial kernel

$$\Phi(\mathbf{x}_1)^T \Phi(\mathbf{x}_2) = K(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1^T \mathbf{x}_2)^2$$



Nonlinear Support Vector Machines: example

Transform data into higher dimensional space (from 2D to 3D)

$$\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2] \rightarrow \Phi(\mathbf{x}) = [\mathbf{x}_1^2 \quad \sqrt{2}\mathbf{x}_1\mathbf{x}_2 \quad \mathbf{x}_2^2]$$

Data set: 2 examples (2D)

$$\mathbf{x} = \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}$$

Mapped data set: 2 examples (3D)

$$\Phi(\mathbf{x}) = \begin{pmatrix} 1 & \sqrt{2} & 1 \\ 1 & \sqrt{2} & 1 \end{pmatrix}$$

Dot product in the 3D space using kernel

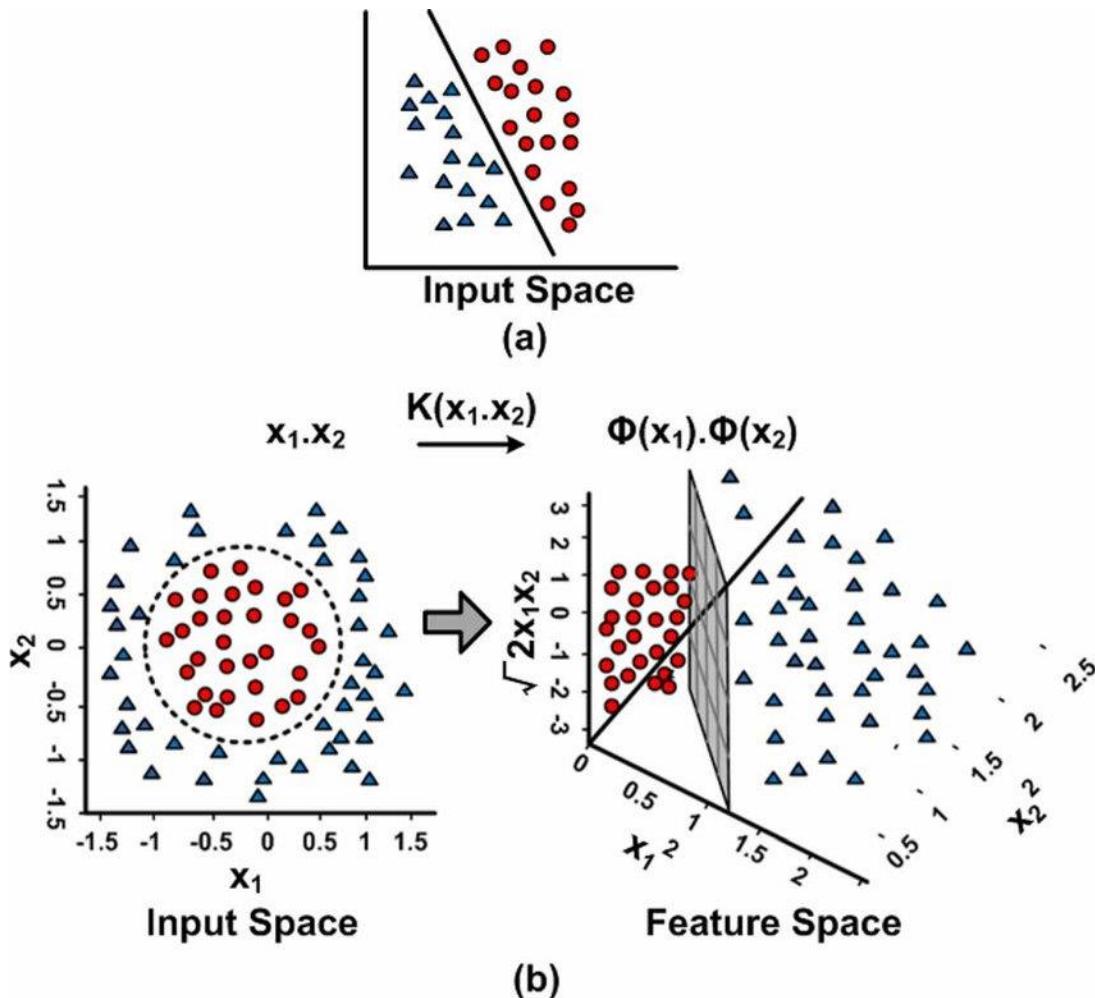
$$\left((1 \quad 1) \begin{pmatrix} -1 \\ -1 \end{pmatrix} \right)^2 = 4$$

Dot product with the 3D data

$$(1 \quad \sqrt{2} \quad 1) \begin{pmatrix} 1 \\ \sqrt{2} \\ 1 \end{pmatrix} = 4$$

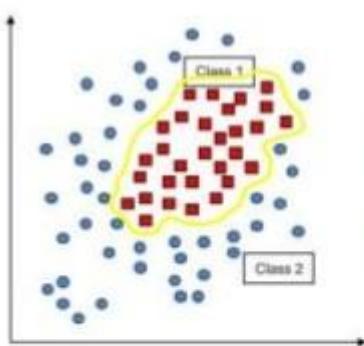
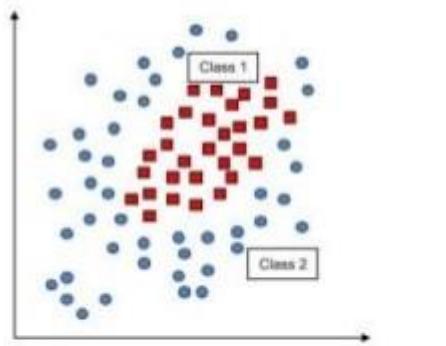
ADVANTAGE: The dot product in the 3D space with the data of the 2D space

Nonlinear Support Vector Machines: example



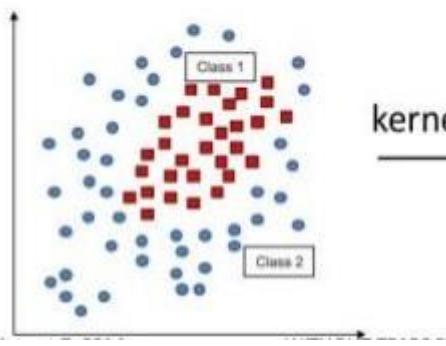
SVM with polynomial degree 2 kernel

Nonlinear Support Vector Machines: example

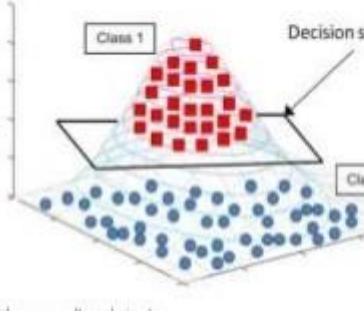


Non Linear
Decision
Boundary

SVM with Gaussian
RBF kernel



kernel



Kernel
method

Contents

- Support Vector Machines (SVM)
 - Support vectors
- Linear support vector machines
- Nonlinear support vector machines
- **Multi-class SVM**
- Summary

Support Vector Machines: multi-class

How to handle more than 2 classes?

- Solve several binary classification tasks
- Essentially, find the support vectors that separate each class from all others

Support Vector Machines: multi-class

Two strategies of training multiple binary classifiers. Considering C classes

- one-against-all (one-against-the rest): C binary classifiers
 - Training set: Positive class (objects of C_i), Negative class (objects of the rest $C_j, j \neq i$)
 - Testing a new object: winner-takes-all strategy, binary classifier with the highest (largest) output function assigns the class
- one-against-one: $c(c - 1)/2$ binary classifiers
 - Training set: Positive class (objects of C_i), Negative class (examples of the other $C_j, j \neq i$)
 - Testing a new object: max-wins voting strategy, in which every classifier assigns the instance to one of the two classes, the class with most votes determines the instance classification

Contents

- Support Vector Machines (SVM)
 - Support vectors
- Linear support vector machines
- Nonlinear support vector machines
 - Learning nonlinear SVM and the Kernel trick
 - Example
- Multi-class SVM
- **Summary**

SVM: summary

The outputs of the training algorithm are

- the values of Lagrangian λ_i : $0 \leq \lambda_i < C$
- the parameter b
- the support vectors (\mathbf{x}_i, y_i) a subset of training set
- Linear SVM: the w can be estimated
- Non-Linear SVM: dual form is a must
- Data set with same support vectors → decision boundary remains

SVM: training and testing

Training SVM

- Choose appropriate kernel function. This implicitly assumes a mapping to a higher dimensional (yet, not known) space
- Assign the parameters of the kernel (e.g., σ if RBF kernel)
- The margin control parameter C

The choice of parameters is usually experimentally driven: k-folder cross-validation

Testing SVM

- The support vectors need to be stored to apply the kernel function (if non-linear)
- The complexity depends on the number of support vectors
- Major limitations of (non-linear) SVM is the computational burden

SVM: characteristics

- The **learning problem** is formulated as a **convex optimization** problem
 - Efficient algorithms are available to find the **global minima** (e.g., SGD)
 - Many of the other methods use greedy approaches and find locally optimal solutions
- SVM is effective on large datasets
 - **Complexity** of trained classifier
 - characterized by the # of support vectors (rather than the size of data)
 - **Support vectors**: essential or critical training examples
 - lie closest to the decision boundary
- SVM with a small number of support vectors can have good generalization, even on large datasets
- **What about categorical variables?**

SVM: advantages and disadvantages

Advantages

- Linear and non-Linear in the same algorithm.
- Good generalization (classification accuracy high)
 - Overfitting: handled by maximizing the margin of the decision boundary
- Robust to noise (works when training examples contain errors)
- Can handle irrelevant and redundant attributes better than many other techniques

Disadvantages

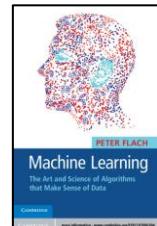
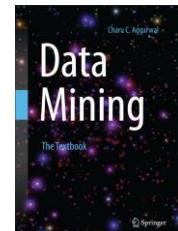
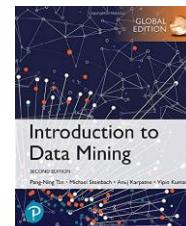
- In training: no criterium to choose of appropriate kernel function (and parameters)
- If number of support vectors is high: complexity (storage and computation) is high
- Not easy to interpret results
- Multiclass problems still need more improvement

Bibliography

Introduction to Data Mining, Pang-Ning Tan, Michael Steinbach, Anuj Karpatne, Vipin Kumar, *Pearson*, 2019 (chap 6.9)

Data Mining, the Textbook, Charu C. Aggarwal, *Springer*, 2015 (chap 10.6)

Machine Learning: The Art and Science of Algorithms That Make Sense of Data, P. Flach, *Cambridge University Press*, 2012 (ch 7.3)



Data Mining

Predictive Modelling

Evaluation Methodologies, Model Selection and Comparison of Models

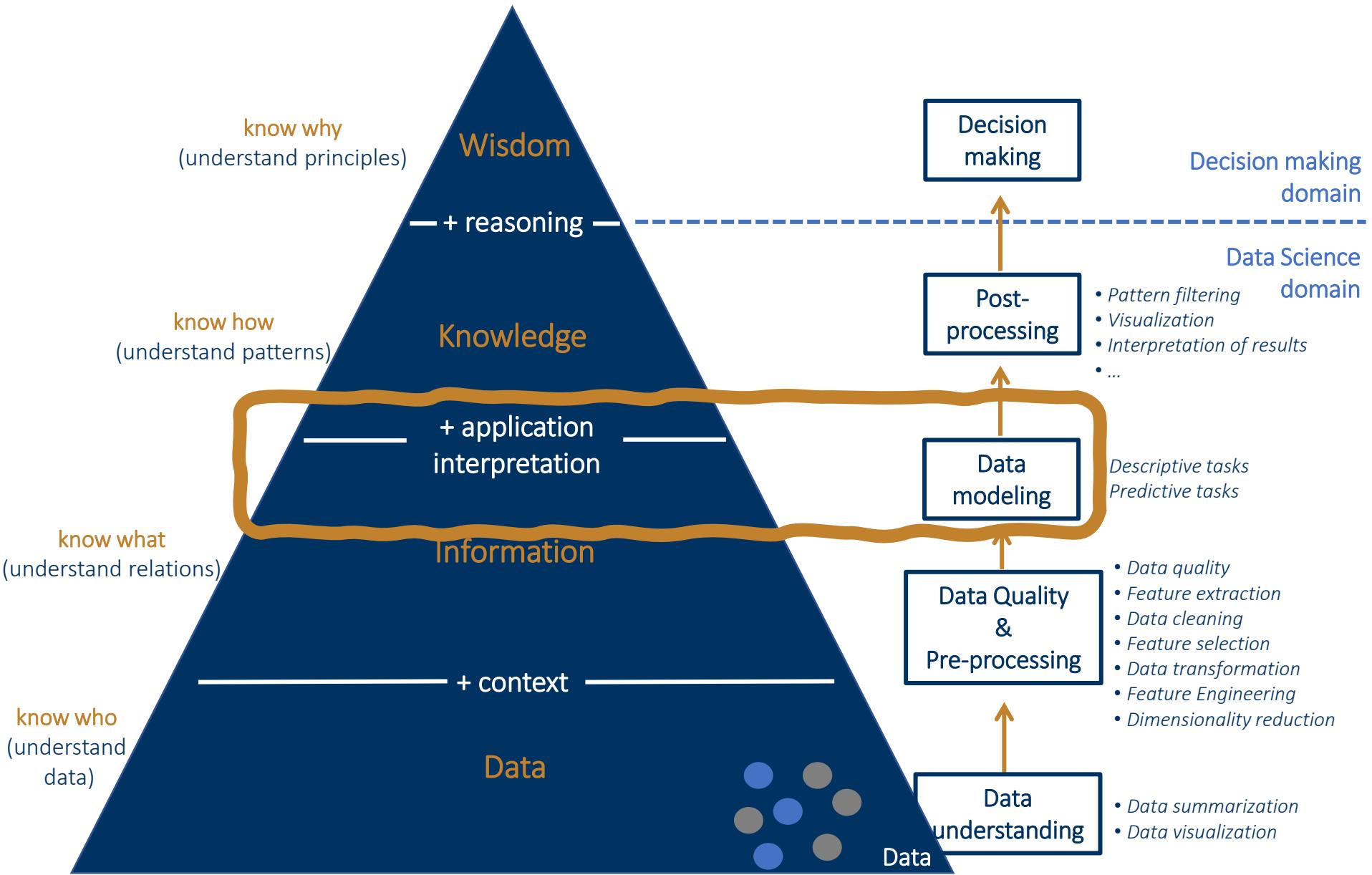
Raquel Sebastião

Departamento de Eletrónica, Telecomunicações e Informática

Universidade de Aveiro

raquel.sebastiao@ua.pt

2022/2023



Contents

- Evaluation methodologies
 - Performance estimation
 - Measures/metrics
 - confusion matrix
 - measures of performance
 - Techniques
 - Holdout
 - K-Fold Cross-Validation
 - Leave-One-Out Cross Validation
- Model selection & hyperparameter tuning
- Comparison of models
- Summary

Performance estimation

Setting

- Given a **data set** $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$, where each **object** is represented by a **D+1–tuple**:
 - Predictors variables: (D–dim) feature vector $\mathbf{x}_i = [x_i^1 \ x_i^2 \ \dots x_i^D]^T \in \mathbb{R}^D$
 - Target variable: corresponding **label** $y_i \in \mathbf{Y}$
- There is an **unknown** function: $\mathbf{Y} = F(\mathbf{X})$ that maps the values of a set of predictors into a target variable value (can be a classification or a regression problem)

Goal: Predictive task

- Learn the **model** that yields the best approximation of the unknown function $F()$
 - Classification problem
 - Regression problem

How to obtain a reliable estimates of the predictive performance of the learned model?

Performance estimation: resubstitution estimate

Estimate of the performance of the model by evaluating on the same data set used for learning the model

- **Unreliable** and should not be used as they tend to be **over-optimistic!**
 - Models are obtained with the goal of optimizing the selected prediction error statistic on the given data set
 - Thus, it is expected to get good scores on the data used for learning
 - The given data set is just a sample of the unknown distribution of the problem being tackled
 - Ideally: compute the performance of the model on this distribution
 - As this is usually impossible, the best we can do is to **evaluate the model on new samples** of this distribution

Use **test set (unseen data)** of class-labeled tuples instead of training set when assessing performance

Performance estimation

- Obtain a **reliable estimate** of the expected prediction error of a model on the unknown data distribution
- In order to be **reliable** it should be based on evaluation on unseen cases: **test set**

The golden rule

The data used for evaluating (or comparing) any models cannot be seen during model development

Contents

- Evaluation methodologies
 - Performance estimation
 - Measures/metrics
 - confusion matrix
 - measures of performance
 - Techniques
 - Holdout
 - K-Fold Cross-Validation
 - Leave-One-Out Cross Validation
- Model selection & hyperparameter tuning
- Comparison of models
- Summary

Evaluation of models: confusion matrix

- Confusion matrix for a binary classification problem

		<i>Predicted class</i>	
		<i>P</i>	<i>N</i>
<i>True class</i>	<i>P</i>	TP True Positive	FN False Negative
	<i>N</i>	FP False Positive	TN True Negative

- TP: hit
- FN: miss (type II error)
- FP: false alarm (type I error)
- TN: correct rejection

Evaluation of models: measures of performance

- **Accuracy** = $\frac{TP+TN}{TP + FN+FP+TN}$ (proportion of correct predictions)
- **Error Rate** = 1- Accuracy (proportion of predictions that are incorrect)
- **Precision** = $\frac{TP}{TP + FP}$ (proportion of correct positive predictions)
- **Recall** = $\frac{TP}{TP + FN}$ (proportion of positive objects correctly predicted)
- **F-measure**: weighted combination of Precision and Recall

$$F_{\beta} = \frac{1}{\alpha \cdot \frac{1}{Precision} + (1 - \alpha) \cdot \frac{1}{Recall}} = \frac{(\beta^2 + 1) \times Precision \times Recall}{\beta^2 Precision + Recall}$$

- **F₁**: $\beta = 1$ then is the harmonic mean of **Precision** and **Recall**
- **Area Under the Curve (AUC)**

Contents

- Evaluation methodologies
 - Performance estimation
 - Measures/metrics
 - confusion matrix
 - measures of performance
 - Techniques
 - Holdout
 - K-Fold Cross-Validation
 - Leave-One-Out Cross Validation
- Model selection & hyperparameter tuning
- Comparison of models
- Summary

Performance estimation techniques

- Holdout
- K-Fold Cross-Validation
- Leave-One-Out Cross Validation
- Bootstrap

Performance estimation techniques: Holdout

It consists of randomly dividing the available data sample in two sub-sets:

- one used for training the model
- the other for testing/evaluating it
 - a frequently used proportion is 70% for training and 30% for testing
 - only one prediction error score is obtained (no average error nor standard error)

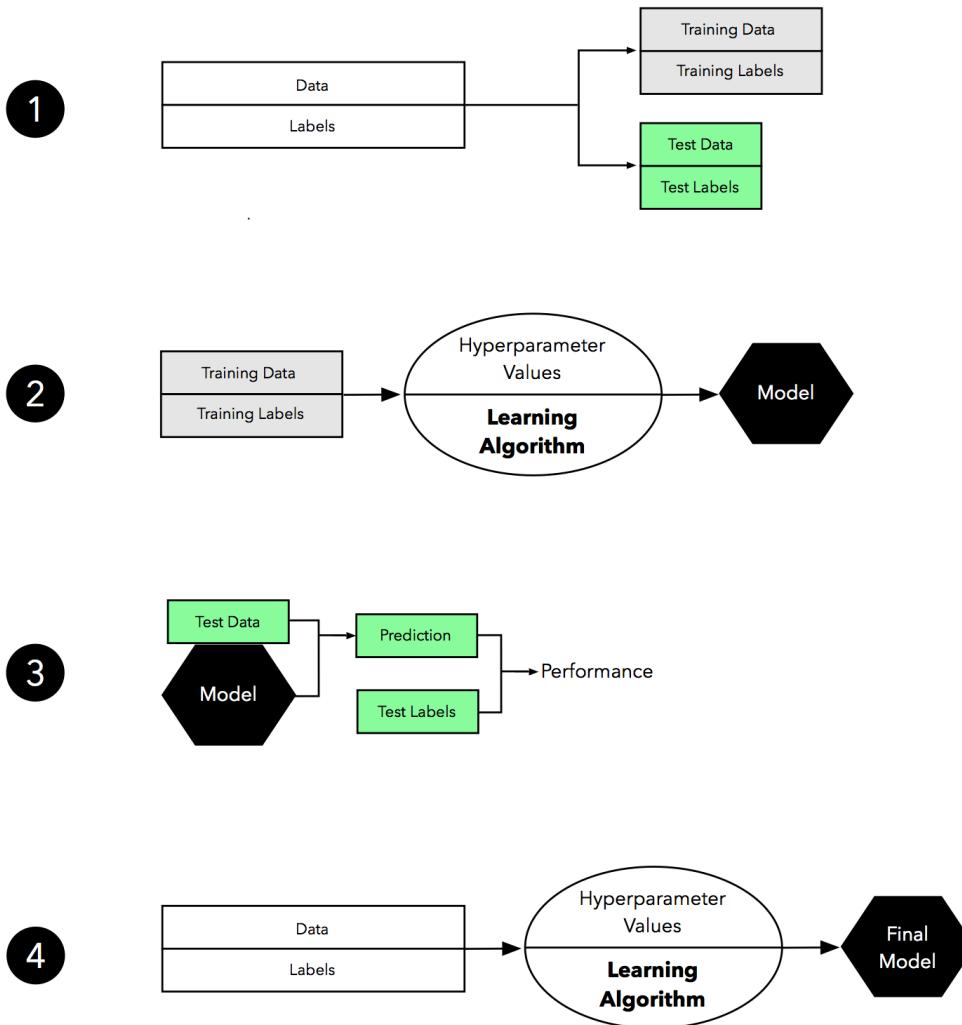


Performance estimation techniques: Holdout

- Very large dataset: preferred evaluation method
- Small dataset
 - too small test set (consequence: unreliable estimates)
 - removing too much data from the training set (worse model than what could be obtained with the available data)



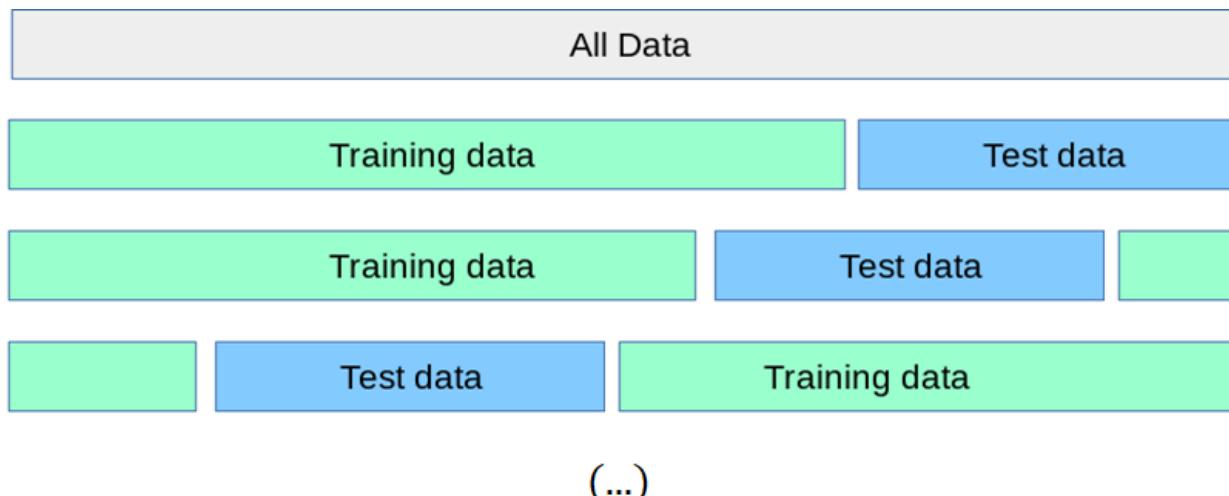
Performance estimation techniques: Holdout



This work by Sebastian Raschka is licensed under a
Creative Commons Attribution 4.0 International License.

Performance estimation techniques: Random subsampling

- Repeated holdout (Monte-Carlo Cross Validation)
 - the holdout process is repeated several times by randomly selecting the train and test sub-sets
- The performance is the average of different training/test
 - several prediction error scores (allow compute average error and standard error)

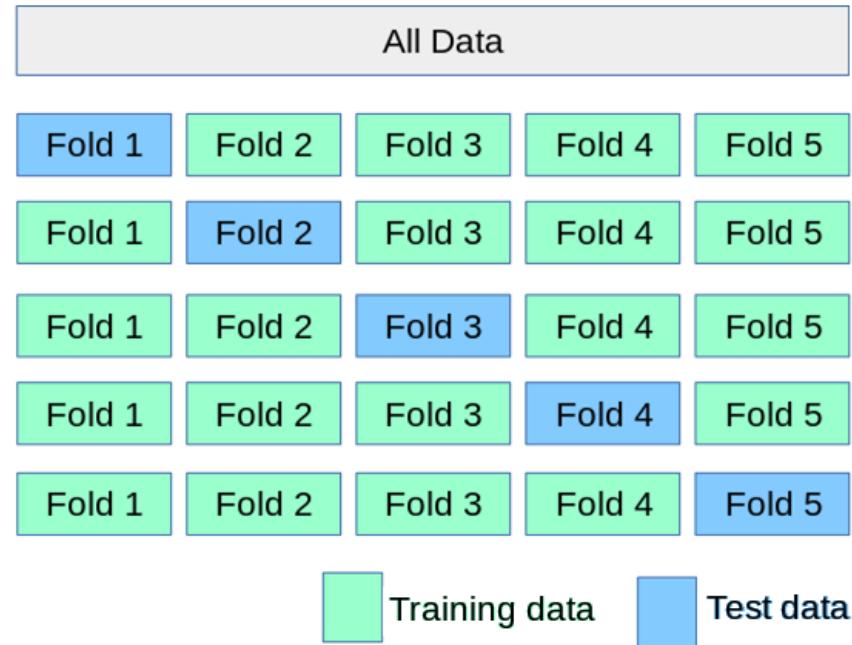


Performance estimation techniques: K-Fold Cross-Validation

Divide the data set into K partitions:

- training set with $(K - 1)$ partitions
- test set with 1 subset

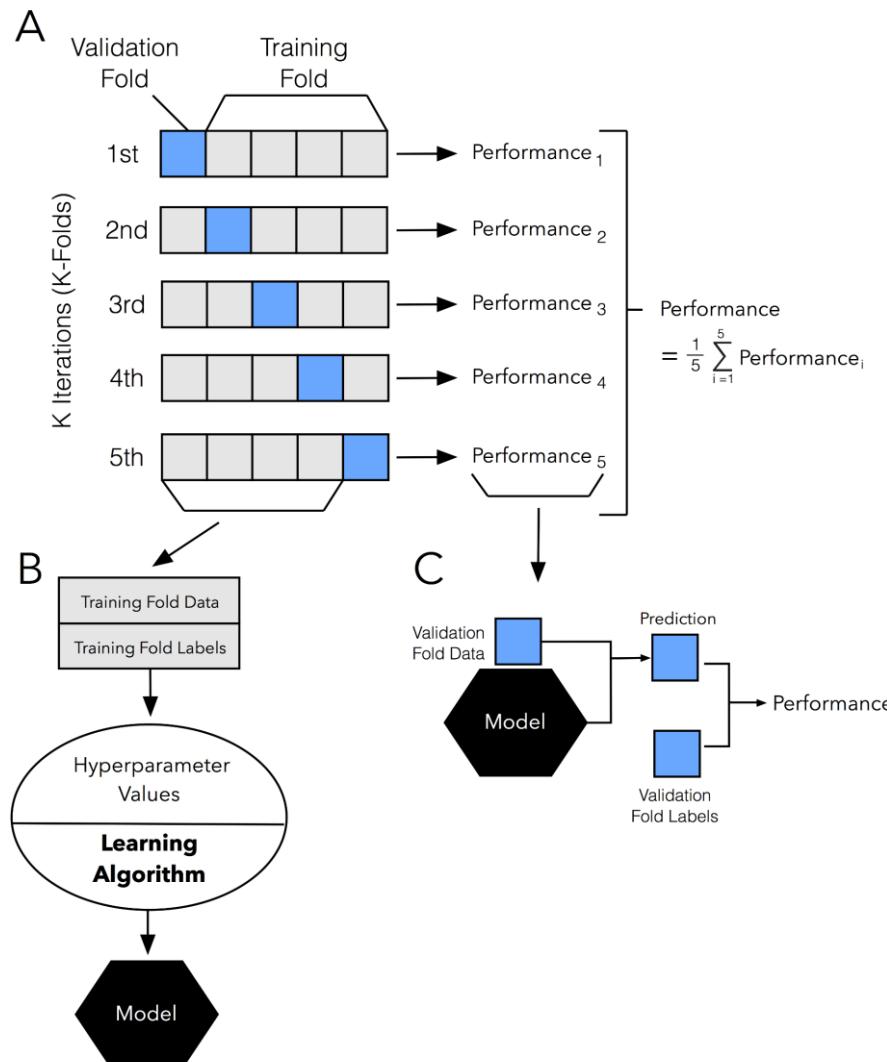
Repeat training/test K times



The performance is the average of different training/test

- several prediction error scores (allow compute average error and standard error)

Performance estimation techniques: K-Fold Cross-Validation



This work by Sebastian Raschka is licensed under a
Creative Commons Attribution 4.0 International License.

Performance estimation techniques: K-Fold Cross-Validation

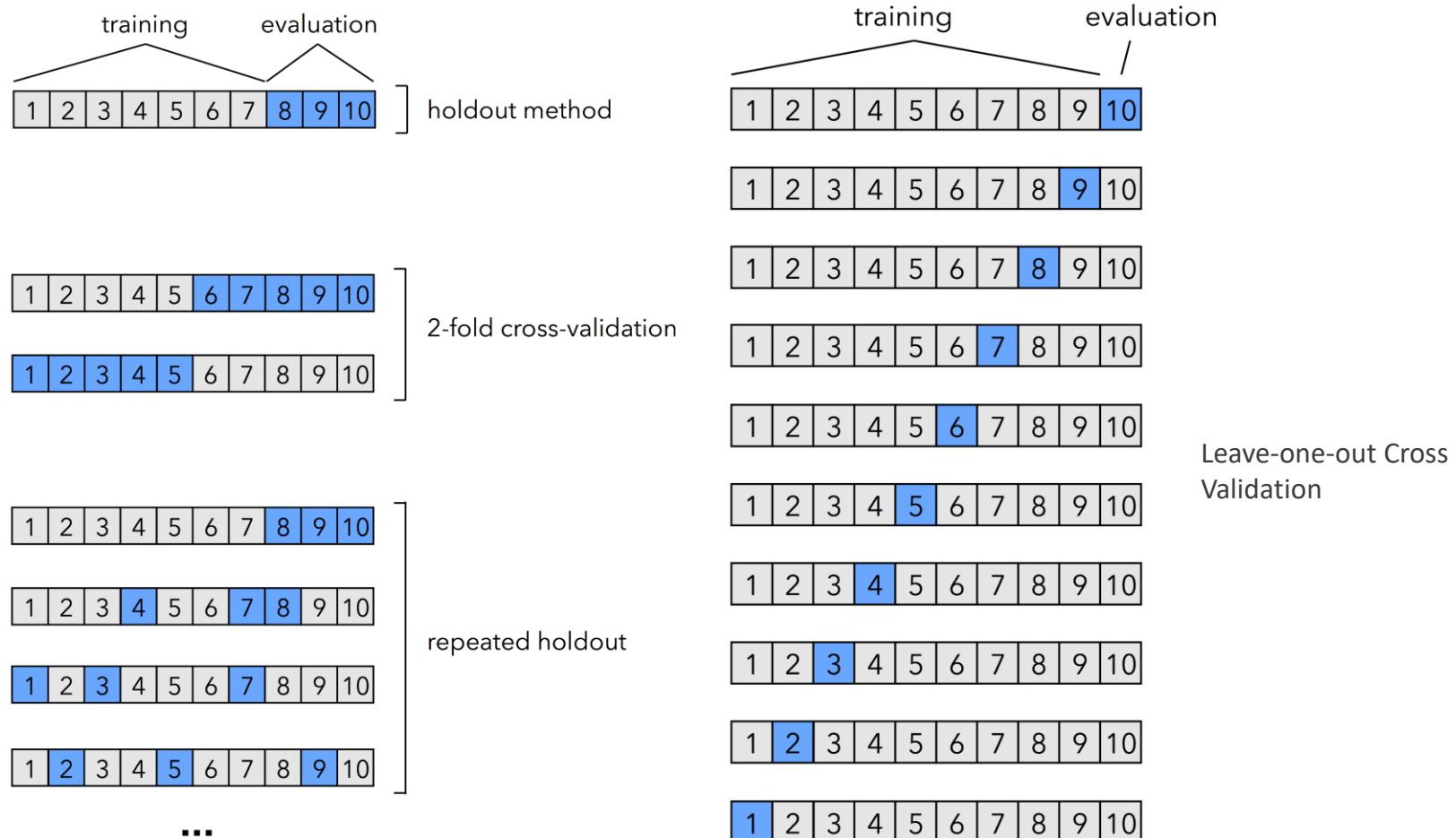
Stratified k-fold Cross Validation

- ensures that, in each fold, each class has roughly same proportion as in full data set
- if it is expected that the learning algorithm to sensitive to the target variable distribution

Leave One Out Cross Validation (LOOCV)

- n-fold CV, where n is the size of the full data set
 - in this case on each iteration, a single case is left out of the training set

Performance estimation techniques: overview



This work by Sebastian Raschka is licensed under a Creative Commons Attribution 4.0 International License.



This work by Sebastian Raschka is licensed under a Creative Commons Attribution 4.0 International License.

Contents

- Evaluation methodologies
 - Performance estimation
 - Measures/metrics
 - Techniques
- Model selection & hyperparameter tuning
 - 3-way Holdout
 - K-Fold Cross-Validation
 - Nested Cross-Validation
- Comparison of models
- Summary

Model selection & optimal hyperparameters

How can we choose the optimal hyperparameters?

Hyperparameter tuning & model selection:

- Meta-optimization task

Learning algorithm

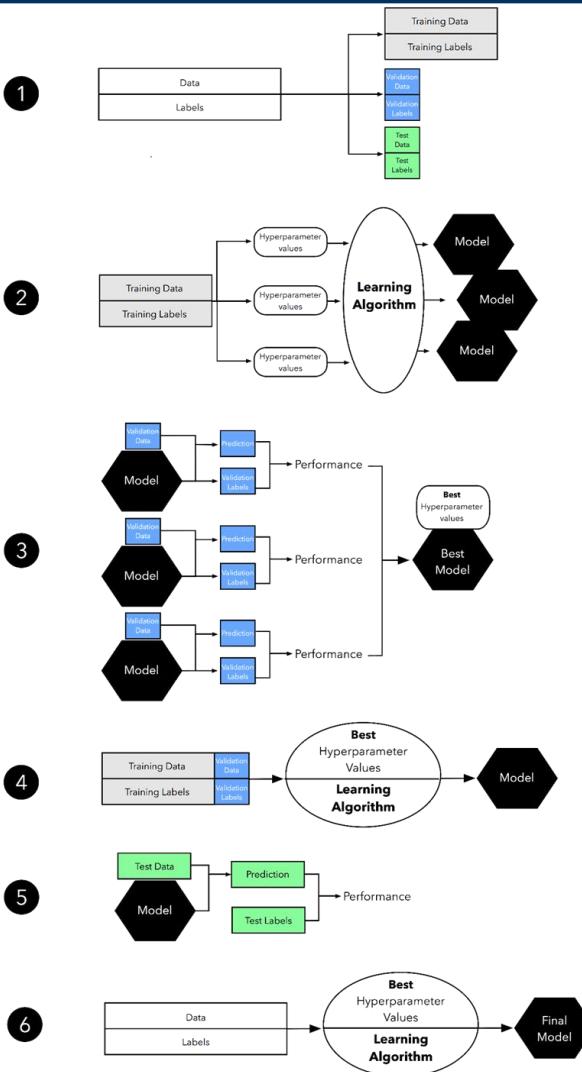
- optimizes an objective function on the training set*

Hyperparameter optimization

- optimize a performance metric
 - another task on top of optimization of the objective function

* with exception of lazy learners

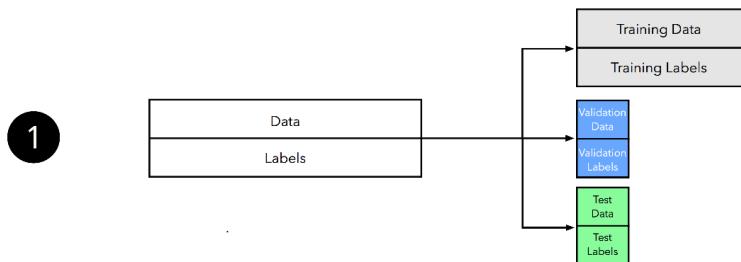
Model selection & optimal hyperparameters: holdout (3-way)



Model selection & optimal hyperparameters: holdout (3-way)

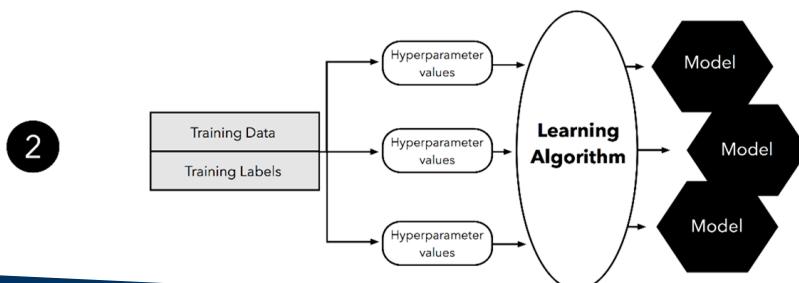
1. The data is split into three subsets (usually random with stratification):

- training set for model fitting
- validation set for model selection
- test set for the final evaluation of the selected model



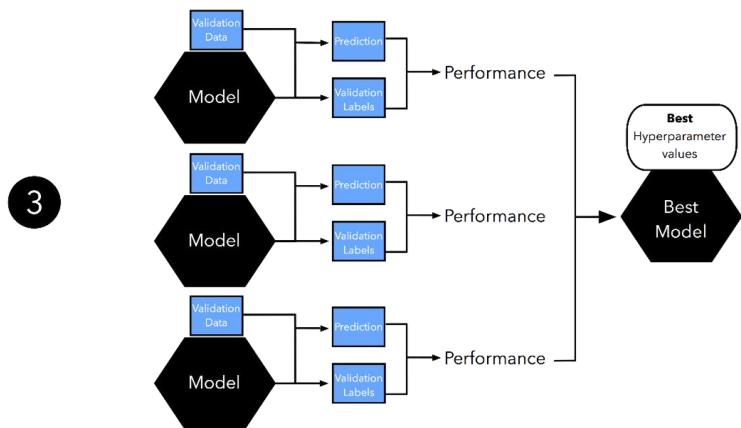
2. Training the model (training set) with different hyperparameters configurations

- For instance: KNN with different values for K, SVM (with different kernels), Neural Networks (with different number of layers/ number of units), and so on

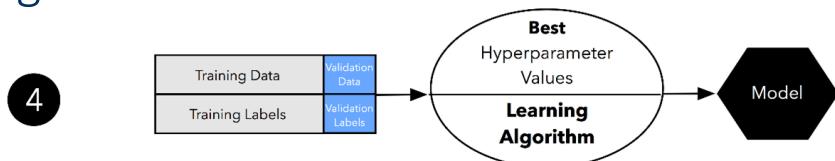


Model selection & optimal hyperparameters: holdout (3-way)

3. Evaluate the performance of each model (for each hyperparameter configuration) on the validation set and choose the hyperparameter conf. associated with the best performance -> **Model selection stage**



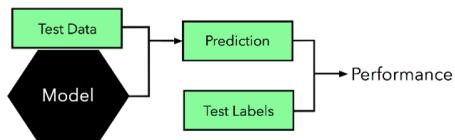
4. After model selection, merge the training and validation set and use the best hyperparameter conf. from the previous step to fit a model to this larger dataset



Model selection & optimal hyperparameters: holdout (3-way)

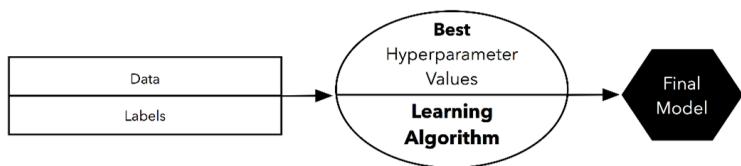
5. Use the independent test set to estimate the generalization performance of the model obtained in step 4

5

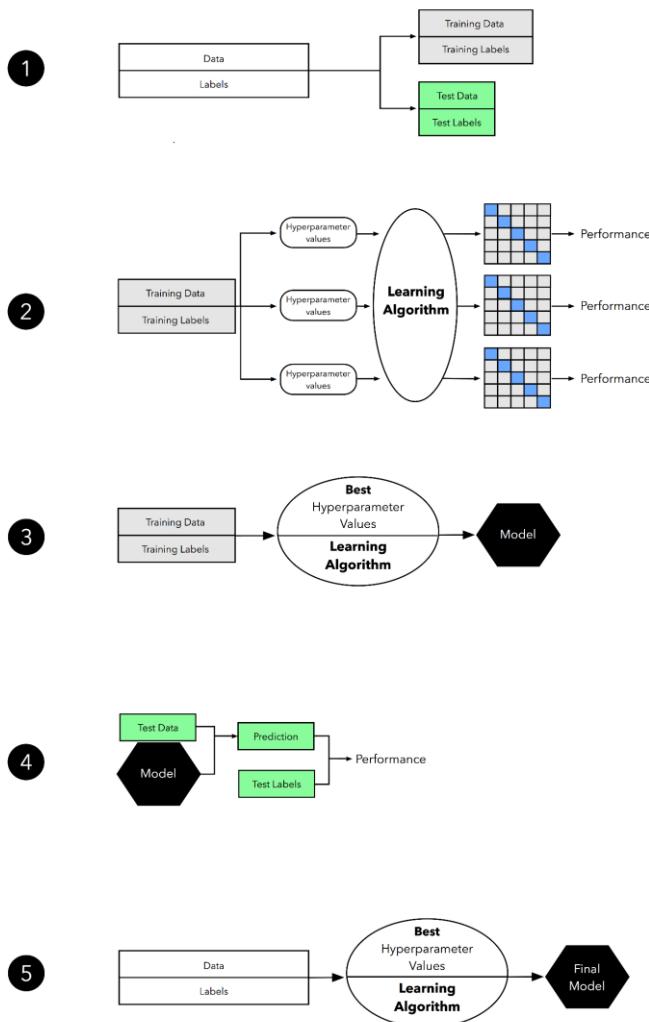


6. Merge training and test set and fit a model (best hyperparameter conf.) to all data points for model deployment

6

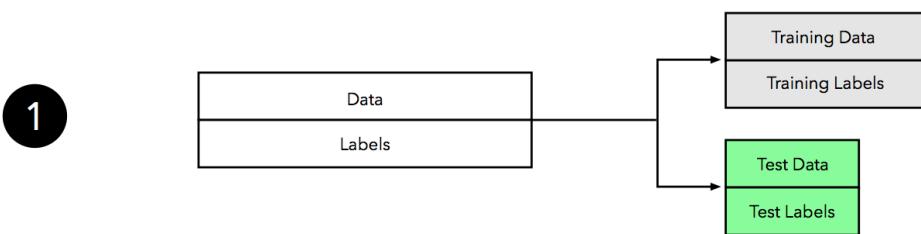


Model selection & optimal hyperparameters: K-Fold Cross Validation

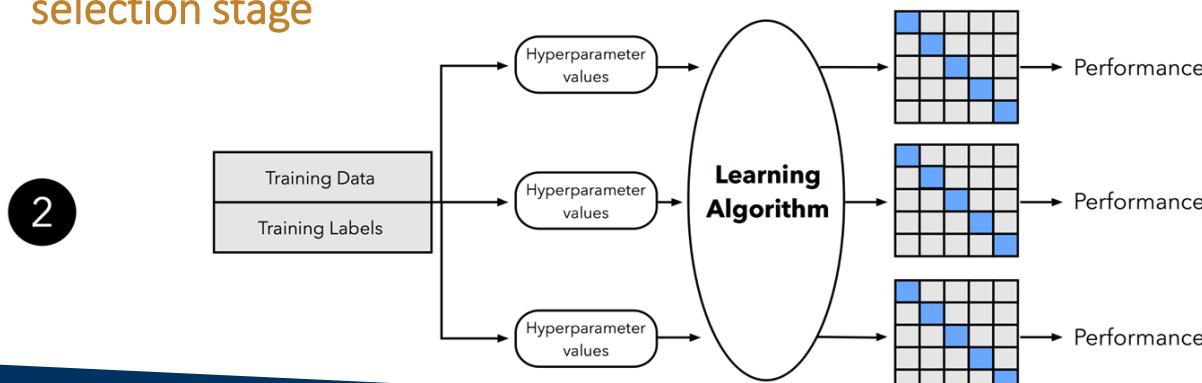


Model selection & optimal hyperparameters: K-Fold Cross Validation

1. The data is split into two subsets (usually random with stratification):
 - training set for model fitting
 - test set for the final evaluation of the selected model



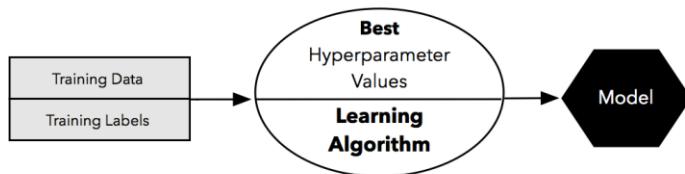
2. Apply k-fold cross-validation on the training set, for each hyperparameter configuration (total number of training run =: $k \times \#$ hyperparameter conf.)
 - choose the hyperparameter conf. associated with the best performance -> **Model selection stage**



Model selection & optimal hyperparameters: K-Fold Cross Validation

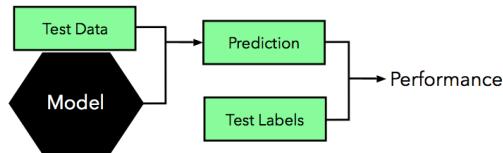
- With the full training set: fit the model with the hyperparameter conf. that correspond to the best-performing model

3



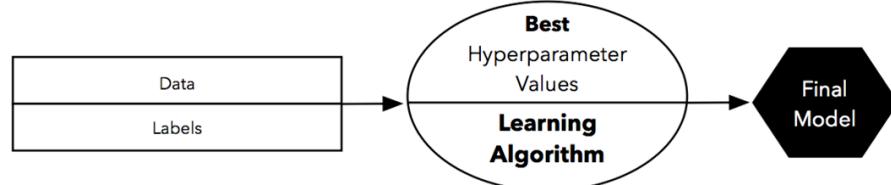
- Use the independent test set to estimate the generalization performance of the model obtained in step 3 (**evaluate on the held-out test set**)

4



- Merge training and test set and fit a model (best hyperparameter conf.) to all data points for model deployment

5



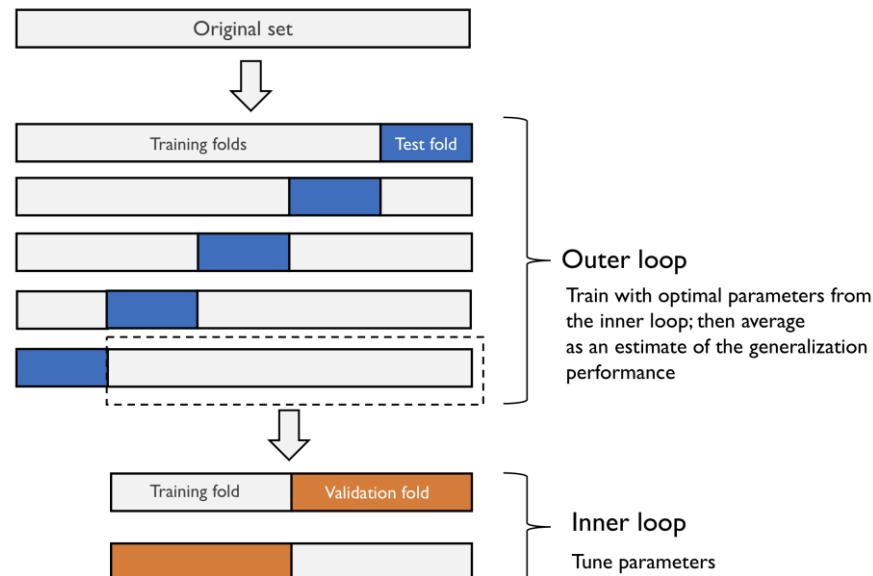
Model selection & optimal hyperparameters: Nested Cross-Validation

Small data sets

- reserving data for independent **test sets** is not feasible

Nesting of two K-Fold Cross-Validation loops:

- inner loop:** for the model selection
- outer loop:** estimating the generalization performance

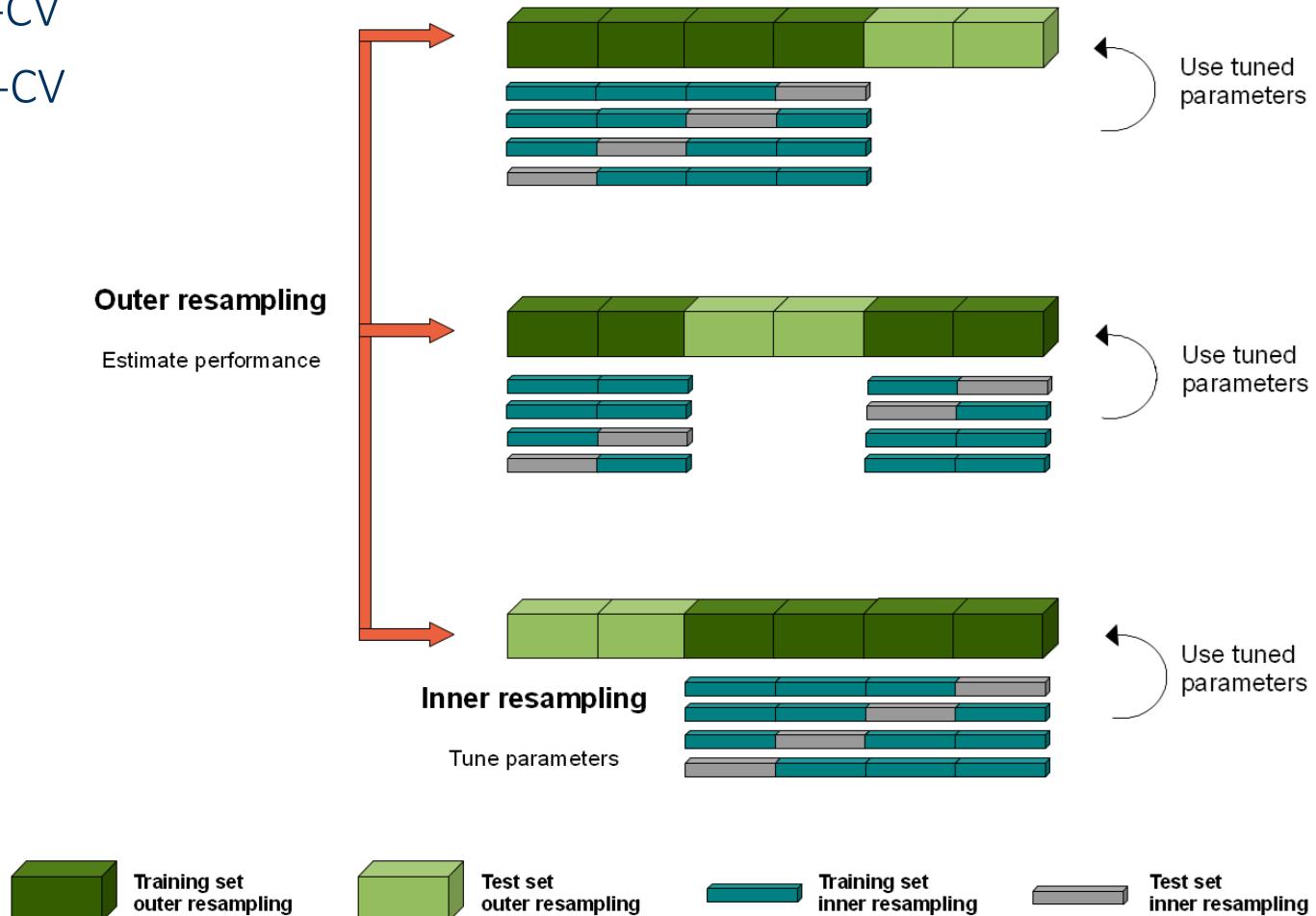


This work by Sebastian Raschka is licensed under a
Creative Commons Attribution 4.0 International License.

Has recently emerged as one of the popular or somewhat recommended methods for comparing machine learning algorithms (Iizuka et al., 2003, Varma and Simon, 2006)

Model selection & optimal hyperparameters: Nested Cross-Validation

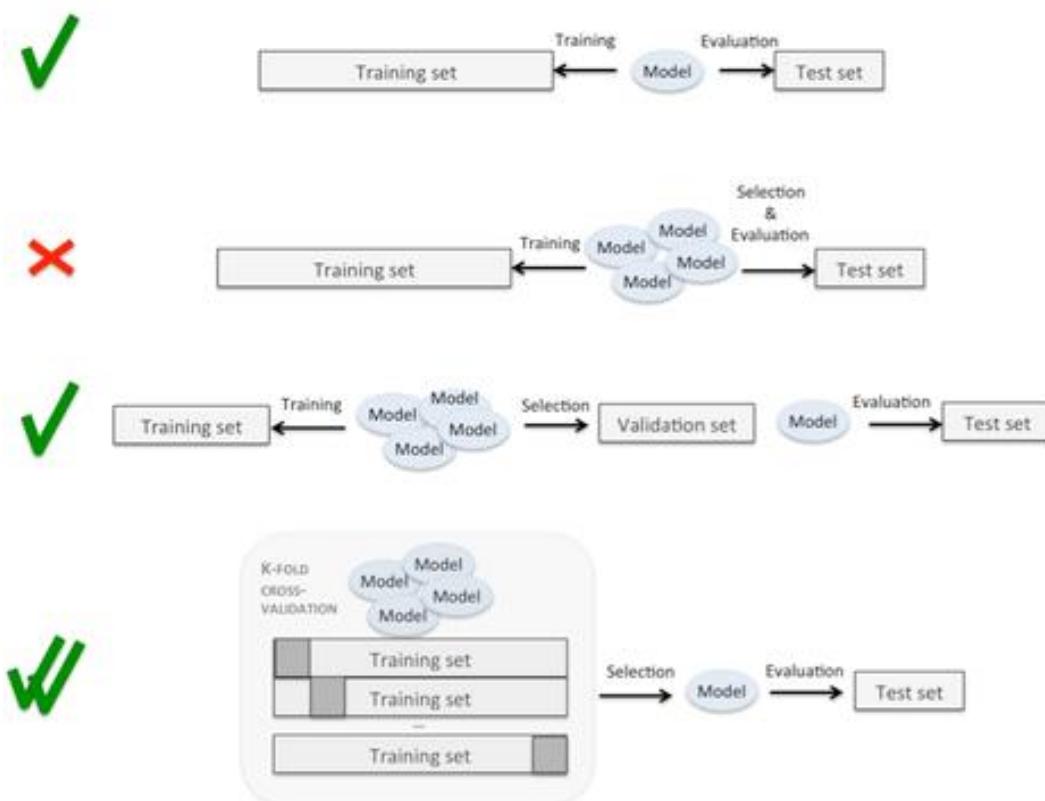
- inner loop: 4-CV
- outer loop: 3-CV



https://mlr.mlr-org.com/articles/tutorial/nested_resampling.html#feature-selection-1

How to evaluate a model?

- Just train a simple model
- Train a model and optimize (tune) its hyperparameters



Contents

- Evaluation methodologies
 - Performance estimation
 - Measures/metrics
 - Techniques
- Model selection & hyperparameter tuning
 - 3-way Holdout
 - K-Fold Cross-Validation
 - Nested Cross-Validation
- Comparison of models
- Summary

Comparison of models

- models were trained and evaluated on the same training and testing sets
- any of the Model evaluation techniques can be used (but the same for all models under comparison)

Comparing the performance of two models:

- Paired t-test
- Mcnemar's test

Comparing the performance of two or more models:

- F-test

Comparison of 2 models: paired t-test

Use the **paired t-test** with the goal of comparing the average performance of both classifiers:

$$H_0: \mu_1 = \mu_2 \quad \text{vs} \quad H_1: \mu_1 \neq \mu_2$$

- H_0 : there is no difference among two models, i.e. the true difference is 0 and any differences in performance are attributed to chance
- H_0 is rejected if the result of the **paired t-test** has a **p-value < α** (α significance level):
 - If **p-value < α** , then H_0 is rejected with **(1 - α) confidence**
- **p-value** is the probability of observing a difference as large as the sample difference given H_0

Comparison of 2 models: paired t-test

Consider that models 1 and 2 were trained with a 10-fold CV technique

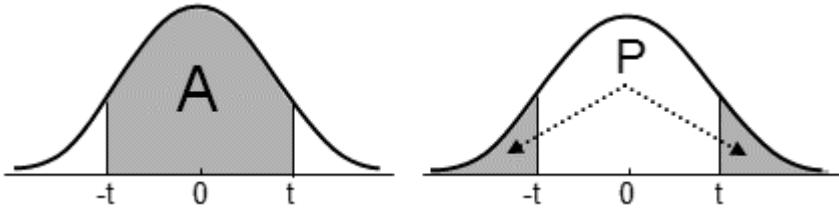
Are the models' performance different?

- accuracy estimates (or error) of classifiers A and B on fold k : $p_k^{(m1)}$ and $p_k^{(m2)}$
- compute the paired difference for each fold k : $p_k = p_k^{(m1)} - p_k^{(m2)}$
- H_0 is whether p_k has mean 0

$$m = \frac{\sum_k p_k}{k} \quad s^2 = \frac{\sum_k (p_k - m)^2}{k - 1}$$

- the **t score** is estimated as $t = \sqrt{k} \frac{m}{s}$
- if $t \in]-t_{\alpha/2,(k-1)}, t_{\alpha/2,(k-1)}[$, then H_0 is not rejected with $(1 - \alpha)$ confidence:
- Reject the null hypothesis that the two models' performances are equal

Comparison of 2 models: paired t-test



- Area $A \equiv (1-\alpha)$ -> confidence level
- Area $P \equiv \alpha$ -> significance level
- $(k-1)$ degrees of freedom (DF)

Example: Difference of two models with mean $m = 0.05$ and standard deviation $s = 0.002$, in 10 folds

DF	A P	0.90 0.10	0.95 0.05	0.99 0.01	0.995 0.005	0.998 0.002
1		6.314	12.706	63.657	127.321	318.309
2		2.920	4.303	9.925	14.089	22.327
3		2.353	3.182	5.841	7.453	10.215
4		2.132	2.776	4.604	5.598	7.173
5		2.015	2.571	4.032	4.773	5.893
6		1.943	2.447	3.707	4.317	5.208
7		1.895	2.365	3.499	4.029	4.785
8		1.860	2.306	3.355	3.833	4.501
9		1.833	2.262	3.250	3.690	4.297
10		1.812	2.228	3.169	3.581	4.144
.....	

- t score: $t = \sqrt{10} \frac{0.05}{0.002} = 79.05$
- t-value: $t_{\frac{\alpha}{2}, 9} = 2.26$ ($\alpha=5\%$)
- $t = 79.05 \notin]-2.26, 2.26[$

The null hypothesis H_0 is rejected

Comparison of 2 models: McNemar's test

- more robust alternative
- also referred to as "within-subjects chi-squared test"
- applied to paired nominal data based on a contingency table
- compares the predictions of two models to each other

		Model 2 correct	Model 2 wrong
		A	B
Model 1 correct	A		
	C	D	

 This work by Sebastian Raschka is licensed under a Creative Commons Attribution 4.0 International License.

Comparison of 2 models: McNemar's test

Compute accuracies of models 1 and 2

- Accuracy_m1 = $(A+B) / n$
- Accuracy_m2 = $(A+C) / n$
- where $n = (A+B+C+D)$ is the total number of test examples
- Cells B and C (the off-diagonal entries), indicate how the models differ
- H_0 is whether $\text{Prob}(B)$ and $\text{Prob}(C)$ are the same
- compute the **p-value**
- if $\text{p-value} < \alpha$, then H_0 is rejected with $(1 - \alpha)$ confidence:
 - Reject the null hypothesis that the two models' performances are equal

		Model 2 correct	Model 2 wrong
Model 1 correct	A	B	
	C	D	

 This work by Sebastian Raschka is licensed under a Creative Commons Attribution 4.0 International License.

Comparison of models: F-test for comparing multiple models

Use the F test with the goal of comparing the performance of multiple models:

$$H_0: p_1 = p_2 = \dots = p_L$$

H_0 : there is no difference among the performance of the models, i.e. the L models don't perform differently

Performance of the classifiers evaluated in the same test set

- H_0 is rejected if the result of the F-test has a p-value $< \alpha$ (α significance level):
 - If p-value $< \alpha$, then H_0 is rejected with $(1 - \alpha)$ confidence
 - There is a difference between the models' performances
 - perform multiple post hoc pair-wise tests (e.g. McNemar tests with a Bonferroni correction)
 - determine which pairs of models have different performances

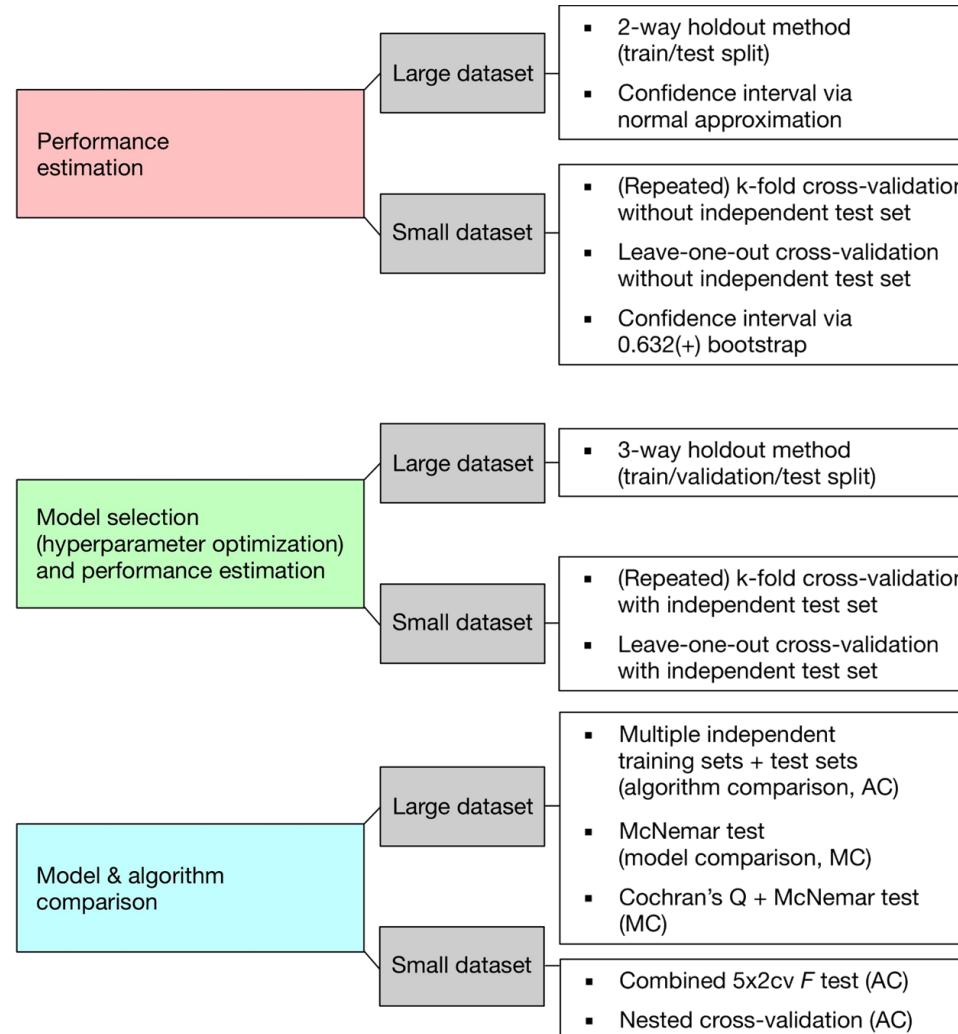
Contents

- Evaluation methodologies
 - Performance estimation
 - Measures/metrics
 - Techniques
- Model selection & hyperparameter tuning
 - 3-way Holdout
 - K-Fold Cross-Validation
 - Nested Cross-Validation
- Comparison of models
- Summary

Summary

- Evaluation methodologies
 - Performance estimation
 - Measures/metrics
 - confusion matrix
 - measures of performance
 - Techniques
 - Holdout
 - K-Fold Cross-Validation
 - Leave-One-Out Cross Validation
- Model selection & hyperparameter tuning
 - 3-way Holdout
 - K-Fold Cross-Validation
 - Nested Cross-Validation
- Comparison of models

Performance estimation, Model selection & Algorithm selection: recommendations



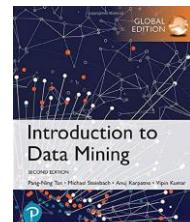
This work by Sebastian Raschka is licensed under a
Creative Commons Attribution 4.0 International License.

Bibliography

Introduction to Data Mining, Pang-Ning Tan, Michael Steinbach, Anuj Karpatne, Vipin Kumar, *Pearson*, 2019 (chap 3.5, 3.6)

Data Mining, the Textbook, Charu C. Aggarwal, *Springer*, 2015 (chap 10.9)

Sebastian Raschka, **Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning**. <https://arxiv.org/abs/1811.12808>



Data Mining

Predictive Modelling

Decision Trees

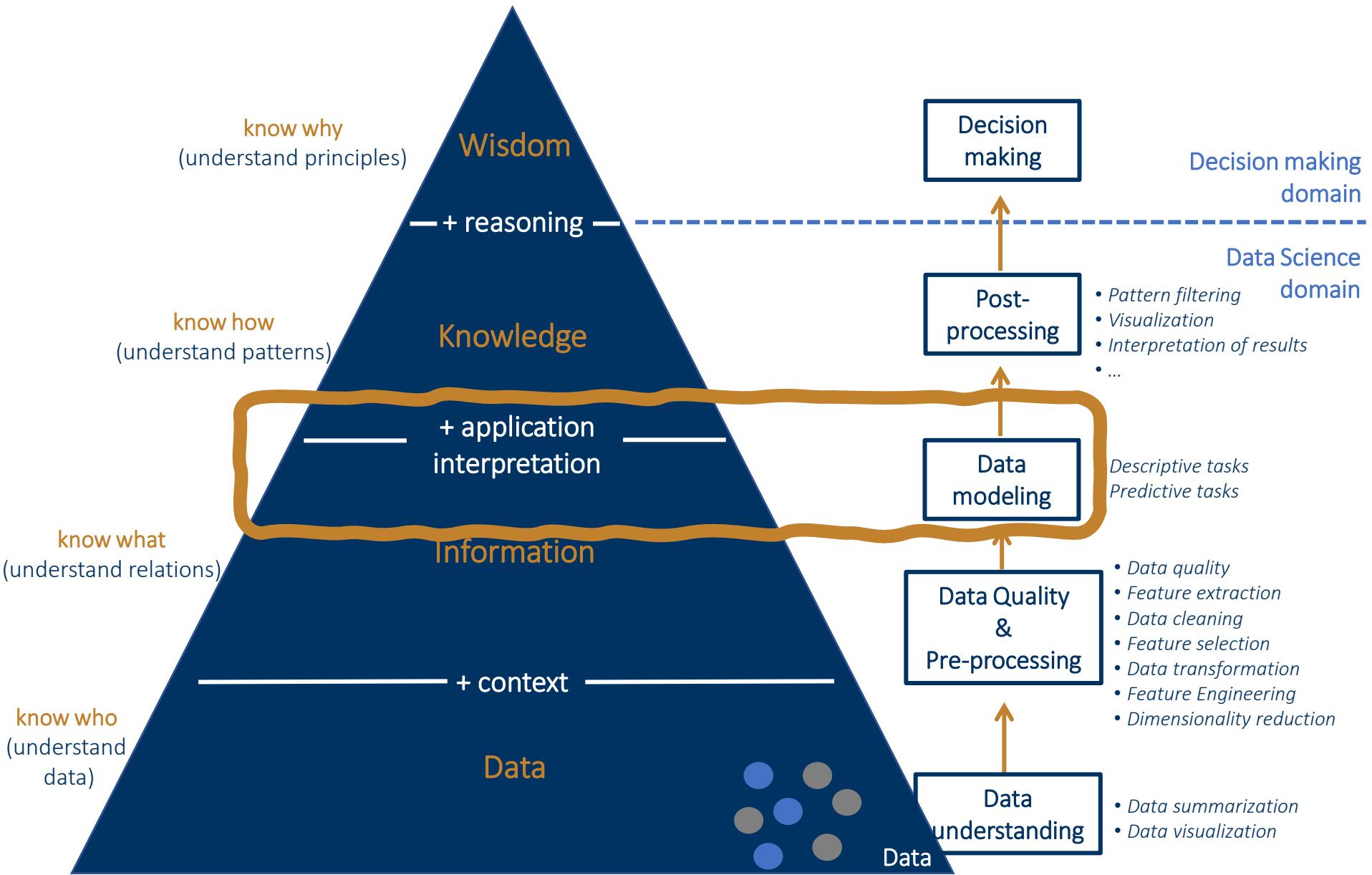
Raquel Sebastião

Departamento de Eletrónica, Telecomunicações e Informática

Universidade de Aveiro

raquel.sebastiao@ua.pt

2022/2023



Prediction Models – approaches

Geometric approaches

- Distance-based: kNN
- Linear models: Fisher's linear discriminant, perceptron, logistic regression, SVM (w. linear kernel)

Probabilistic approaches

- naive Bayes, logistic regression

Logical approaches

- classification or regression trees, rules

Optimization approaches

- neural networks, SVM

Sets of models (ensembles)

- random forests, adaBoost

Decision Trees

Model to predict a response y based on the feature vector

$$\mathbf{x} = [x^1 \quad x^2 \quad \dots x^D]^T \in \mathbb{R}^D$$

If the dependent (target) variable y (predicted) represents:

- Class label: classification tree
- Real number: regression tree

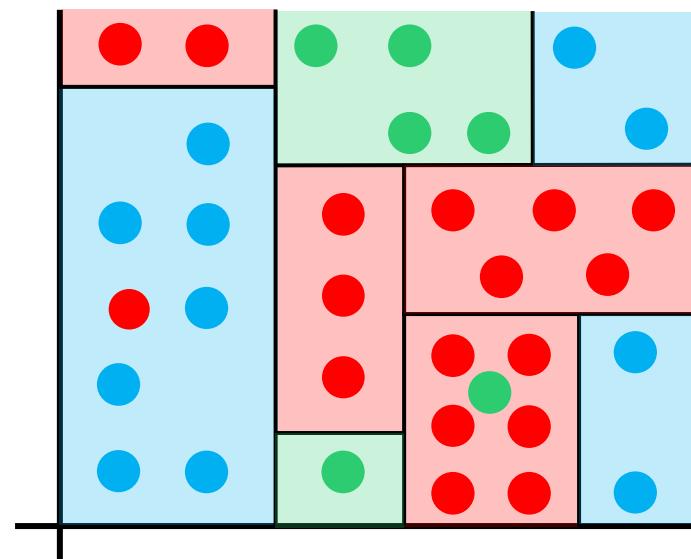
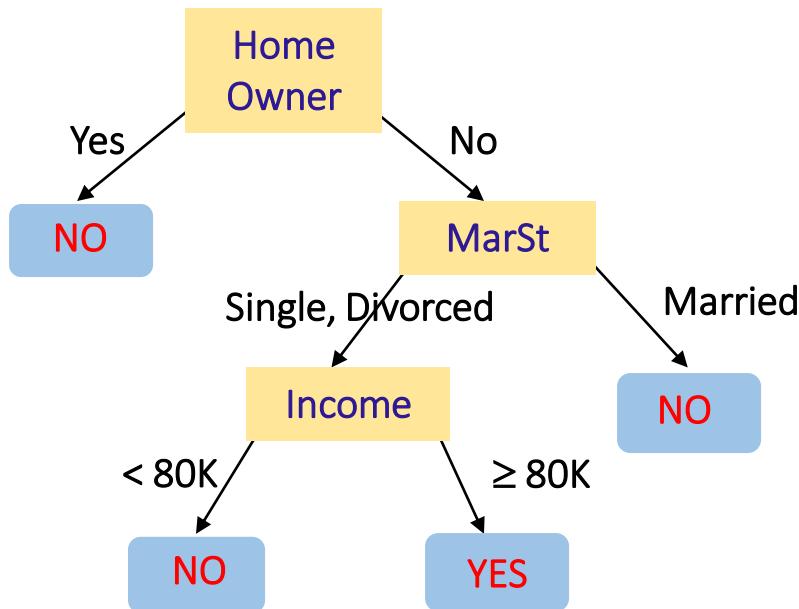
Ensemble algorithms

- combining many decision trees

Decision Trees

Divide-and-conquer method

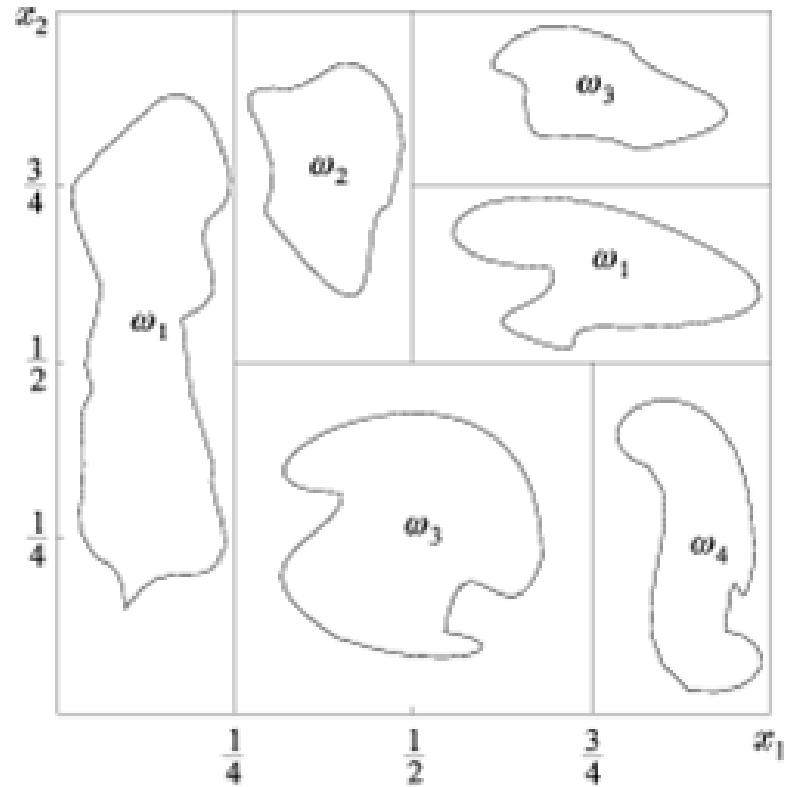
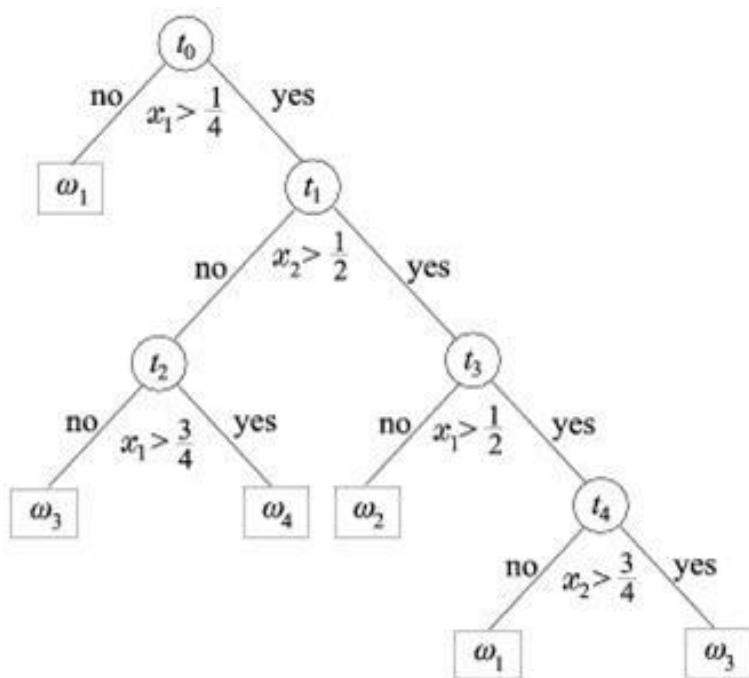
- Set of *splitting rules*
- Partitioning the feature space into smaller (non-overlapping) regions with similar response values
- Each node is a partition of the input space



Decision Trees

The decision surfaces

- hyperplanes, splitting the space into regions
- are parallel to the axis of the spaces



Decision Trees

Tree:

- **Root node** – no income edges
- **Internal node** – with a test on an attribute/predictor variable
- **Branch** – represents an outcome of the test in the respective node
- **Leaf node** – contains the value of the target variable (either class label or a numeric value)

Training phase: at each node, one attribute is chosen **to split training examples** into distinct classes as much as possible

Testing phase: A new case is classified by following a matching **path** to a **leaf node**

Decision Trees

Tree:

- **Root node** – no income edges
- **Internal node** – with a test on an attribute/predictor variable
- **Branch** – represents an outcome of the test in the respective node
- **Leaf node** – contains the value of the target variable (either class label or a numeric value)

Training phase: at each node, one attribute is chosen **to split training examples** into distinct classes as much as possible

Testing phase: A new case is classified by following a matching **path** to a **leaf node**

Decision Trees

Algorithms:

- CLS (Hunt et al., 1966): Concept Learning System – early algorithm for DT construction
- ID3 (Quinlan, 1979): based on information theory
- CART (Breiman et al., 1984)
- C4.5 (Quinlan, 1993): improved extension of ID3
- (...)

CLASSIFICATION AND REGRESSION TREES (CART)



Decision Trees



Decision Trees: learning

Algorithms to grow a decision tree use a **best feature x_i at each node** and the related value

- to split the corresponding subset of **the training set into more homogeneous groups**

The **main issues** of the algorithm

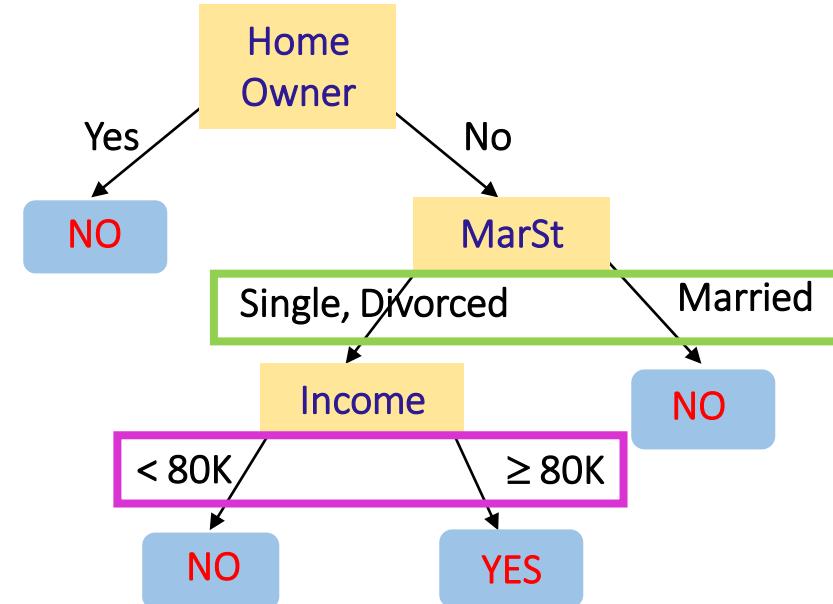
- **Splitting Rule:** split the cases of the sample to form partitions that are more homogeneous (“purer”) than the parent node
- Using a *homogeneity function* (impurity function) based on the distribution of the classes at each node
- **Class Assignment Rule:** Assign a class to each leaf (terminal node)
- **Pre-pruning Rule:** early stop (stop growing the tree)

Post-Pruning: removing branches and nodes after training

Decision Trees: learning

Splitting binary rules

- numerical predictors: $x_i \in \mathbb{R}$
- categorical predictors: $x_i \in \{v_1, \dots, v_m\}$



Each path from the root till a leaf node is a set of logical tests defining a region on the predictors space

All the examples “falling” on a leaf will get the same prediction (either a class label or a numeric value)

Decision Trees: learning

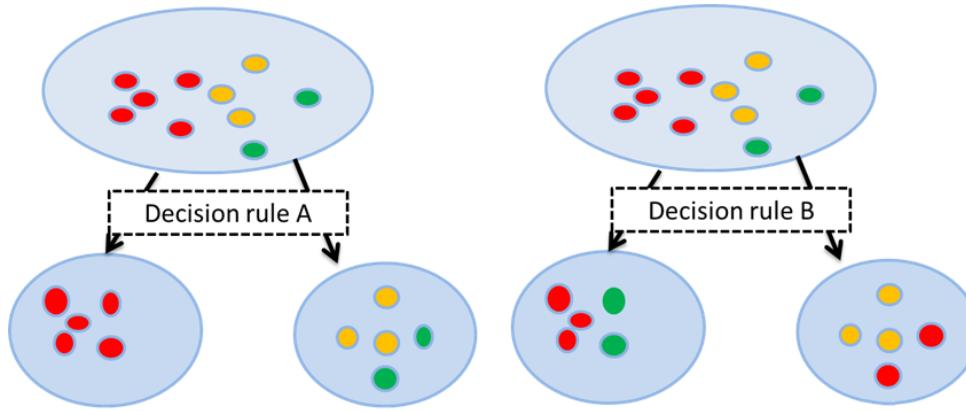
Splitting binary rules

- How to handle categorical attributes*
 - Evaluate all possible combinations of 2 subsets of values
 - Choose the test that produces more homogeneous partitions
- How to handle continuous attributes
 - Discretize continuous values and treat them as categorical values
 - Determine the best split point
 - Sort all values in increasing order
 - Consider all possible binary splits and finds the best cut

* For ordinal attributes: preserve order property among attribute values

Decision Trees: homogeneity function

The **basic idea** is to find out decision rules which turn the **subsets more homogenous**



- Decision rule A: one of the subsets is homogeneous
- Decision rule B: subsets have samples of two classes almost equiprobable

The rule A is preferred over rule B

Decision Trees: designing facts

Each node k is associated with a subset \mathbf{X}_k of the training set \mathbf{X} .

In node k

- \mathbf{X}_k is split into two (binary splits) disjoint descendant subsets:
 - $\mathbf{X}_{k,N} \cap \mathbf{X}_{k,M} = \emptyset$
 - $\mathbf{X}_{k,N} \cup \mathbf{X}_{k,M} = \mathbf{X}_k$
 - the subsets $\mathbf{X}_{k,N}$ and $\mathbf{X}_{k,M}$ correspond to "YES" and "NO" branches

How to measure class-homogeneity of set \mathbf{X}_k and subsets $\mathbf{X}_{k,N}$ and $\mathbf{X}_{k,M}$?

- Computing the variation on entropy (information gain)
- Computing the variation on Gini index

Decision Trees: homogeneity of sets

How to measure class-homogeneity of (sub)set \mathbf{X}_k and subsets $\mathbf{X}_{k,N}$ and $\mathbf{X}_{k,M}$?

- Computing the variation on entropy (information gain)

$$\Delta I(k) = I(k) - \frac{N_{k,N}}{N_k} I(k_N) - \frac{N_{k,M}}{N_k} I(k_M)$$

- Computing the variation on Gini index

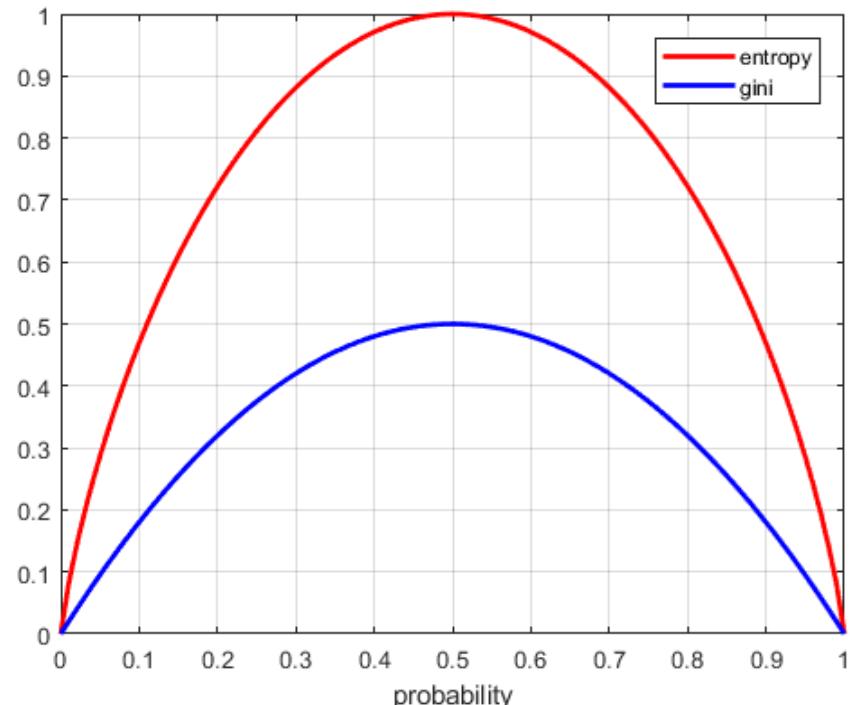
$$\Delta G(k) = G(k) - \frac{N_{k,N}}{N_k} G(k_N) - \frac{N_{k,M}}{N_k} G(k_M)$$

- Entropy: $I = -\sum_i p_i \log_2(p_i)$
- Gini index: $G = 1 - \sum_i (p_i)^2$

where $p_i, i = 1, \dots, C$ are the probabilities of the classes

Decision Trees: homogeneity of sets

Data set with two ($C=2$) classes



- I (or G) is **maximum** when $p_i = p_j, \forall i, j$
 - maximum uncertainty (randomness, **non-homogeneous**)
- I (or G) is **minimum** when $p_i = 1, p_j = 0 \quad (i \neq j) \quad i = 1$
 - the outcome is already known (set is **homogeneous**)

Decision Trees: splitting criteria

Node k : with N_k elements, belonging to classes $\omega_i, i = 1, \dots, C$

- Entropy* of node k

$$I(k) = - \sum_{i=1}^C P(\omega_i|k) \log_2(P(\omega_i|k))$$

where

- $P(\omega_i|k) = \frac{N_k^{(i)}}{N_k}$
- $N_k^{(i)}$ is the number of examples (in node k) that belongs to class ω_i

* The Gini index can be used instead of entropy

Decision Trees: splitting criteria

Assuming that the (sub)set \mathbf{X}_k (at node k) is divided into the subsets

- $\mathbf{X}_{k,N}$ with entropy $I(k_N)$
- $\mathbf{X}_{k,M}$ with entropy $I(k_M)$

Then the **decrease in impurity** with this split is given by the variation on **entropy**

$$\Delta I(k) = I(k) - \frac{N_{k,N}}{N_k} I(k_N) - \frac{N_{k,M}}{N_k} I(k_M)$$

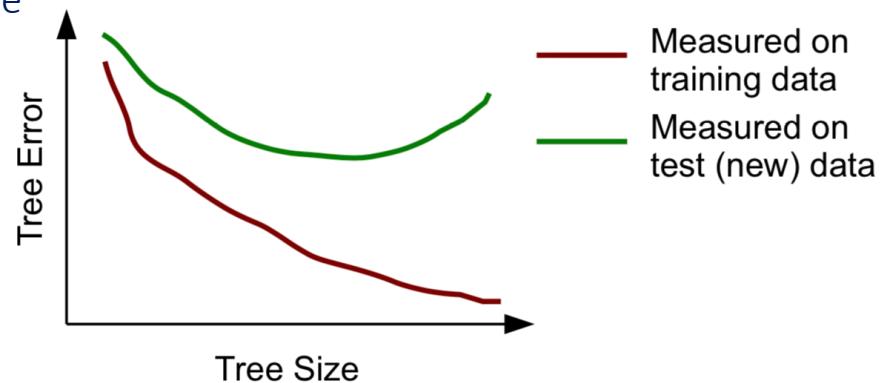
$\Delta I(k)^*$ is computed for **every possible splitting test**

The **splitting rule** is chosen for the **maximum $\Delta I(k)$**

* The **Gini index** can be used instead of **entropy**

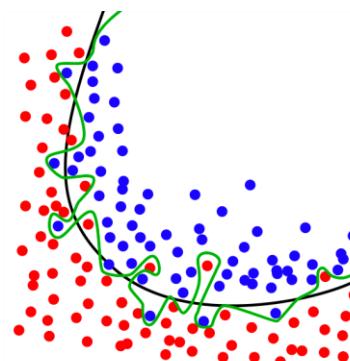
Decision Trees: overfitting and tree pruning

- Overall scores keep improving while growing the DT
- Going down in the tree: the split decisions are made based on smaller and smaller sets
- Thus, potentially less reliable decisions are made



Overfitting

Too large trees tend to overfit the training data and will perform badly on new data



Decision Trees: overfitting and tree pruning

Pre-pruning: stop growing the tree if our estimate of quality indicates that is not worth continuing

- minimum nr. of examples in a node
- minimum nr. of examples in a leaf
- maximum depth of the tree

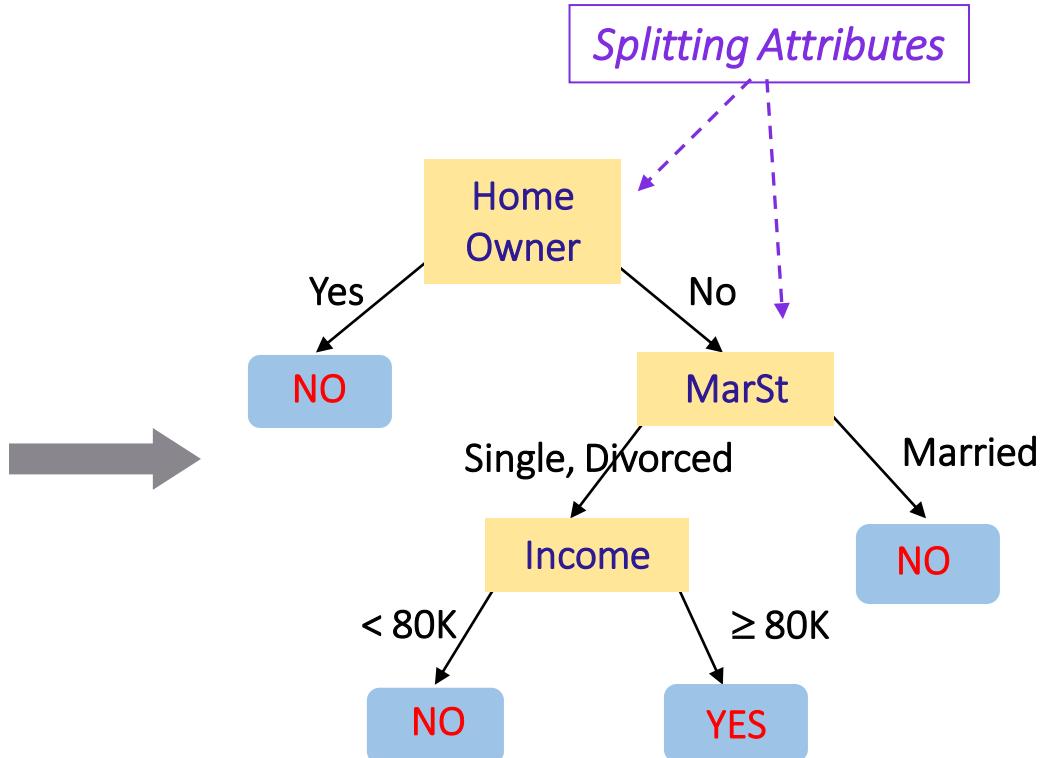
Post-pruning: grow an overly large tree and then use some statistical procedure to prune unreliable branches according to error estimates

- *Minimum error:* The tree is pruned back to the point where the cross-validated error is a minimum
- *Smallest tree:* The tree is pruned back slightly further than the minimum error

Classification Trees: example

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower	
					categorical categorical continuous target
1	Yes	Single	125K	No	
2	No	Married	100K	No	
3	No	Single	70K	No	
4	Yes	Married	120K	No	
5	No	Divorced	95K	Yes	
6	No	Married	60K	No	
7	Yes	Divorced	220K	No	
8	No	Single	85K	Yes	
9	No	Married	75K	No	
10	No	Single	90K	Yes	

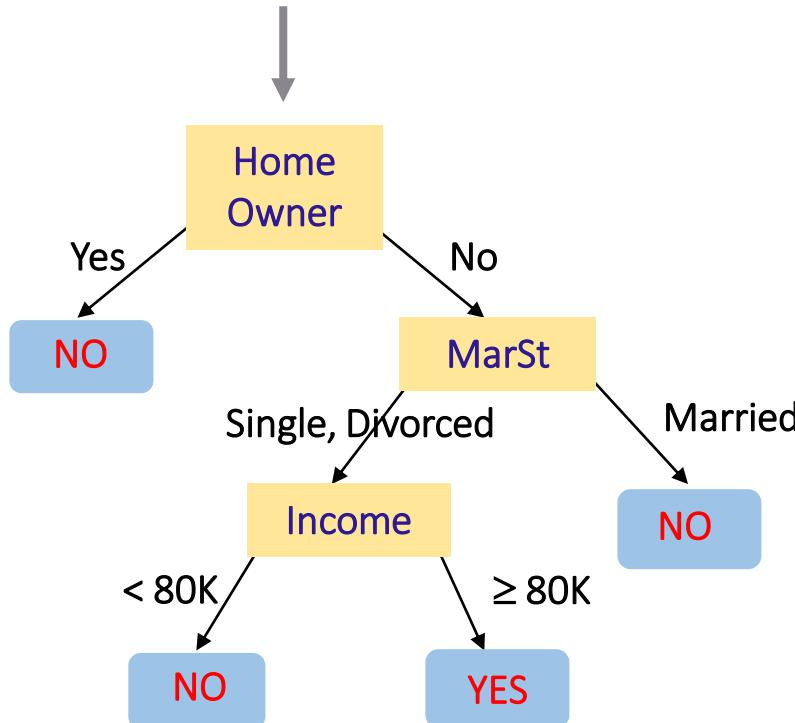
Training Data



Model: Decision Tree

Classification Trees: apply model to test data

Start from the root of tree

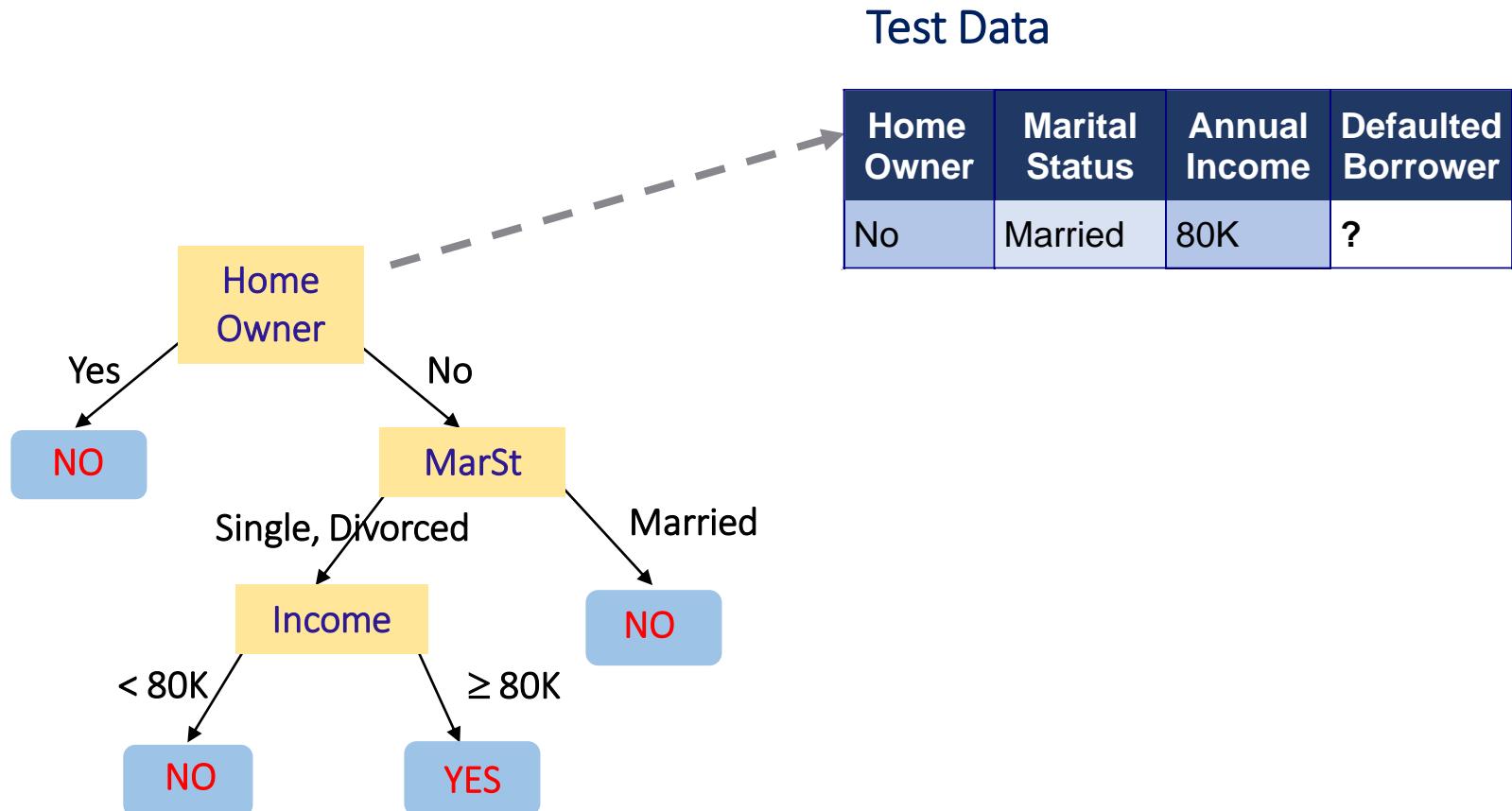


Test Data

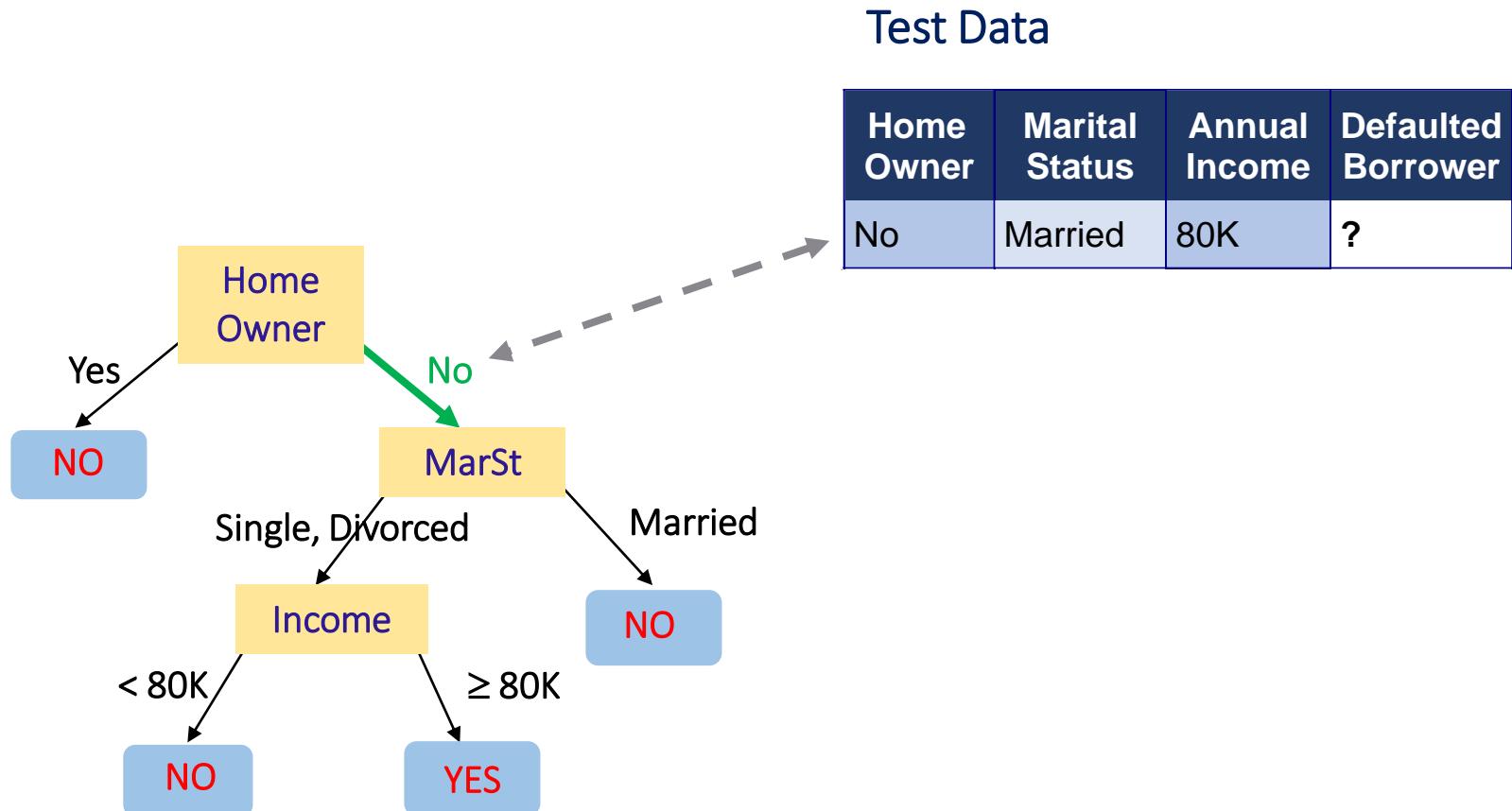
Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?

The prediction for a new test case is obtained by following a path from the root till a leaf according to its predictor values

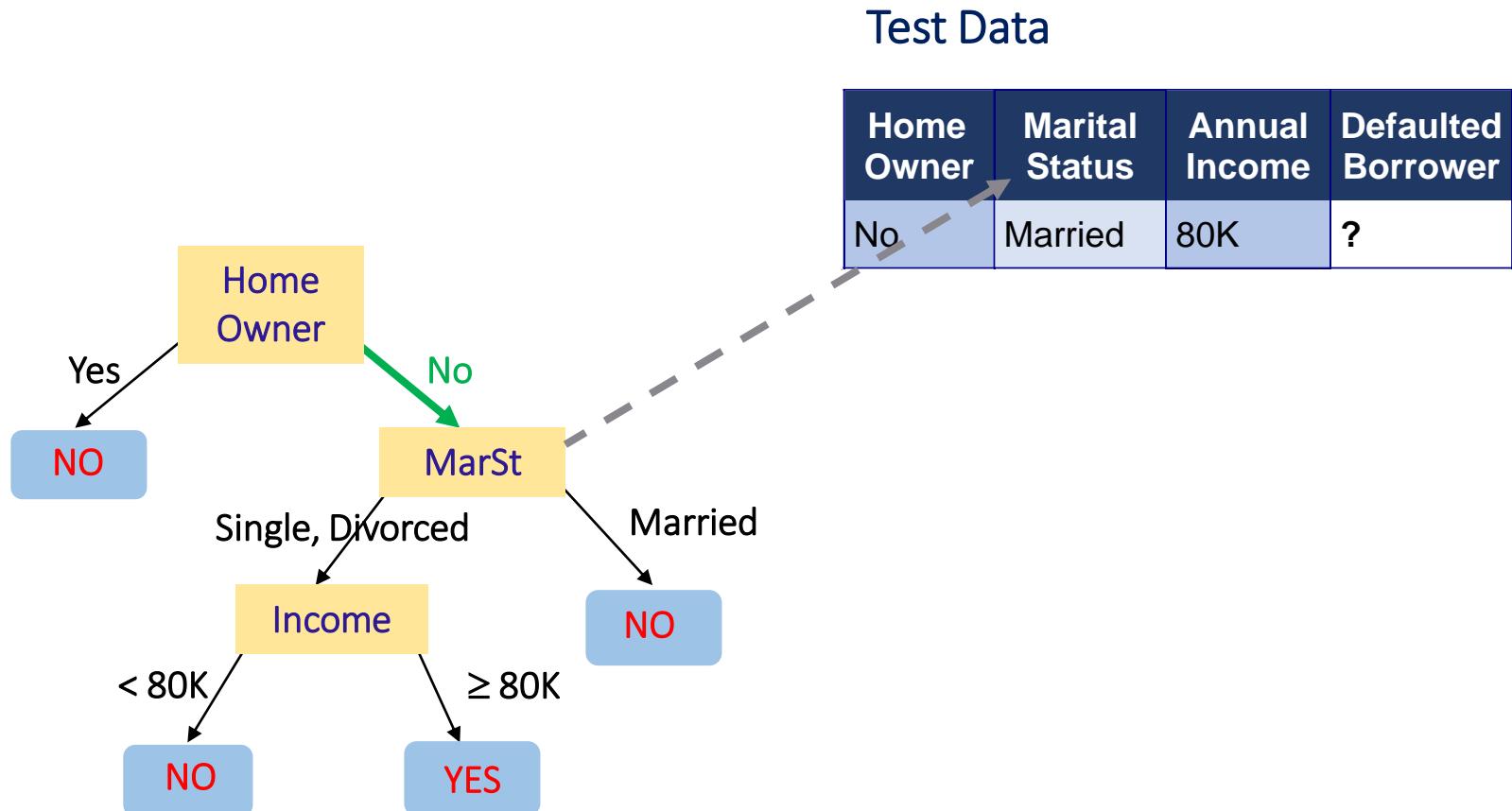
Classification Trees: apply model to test data



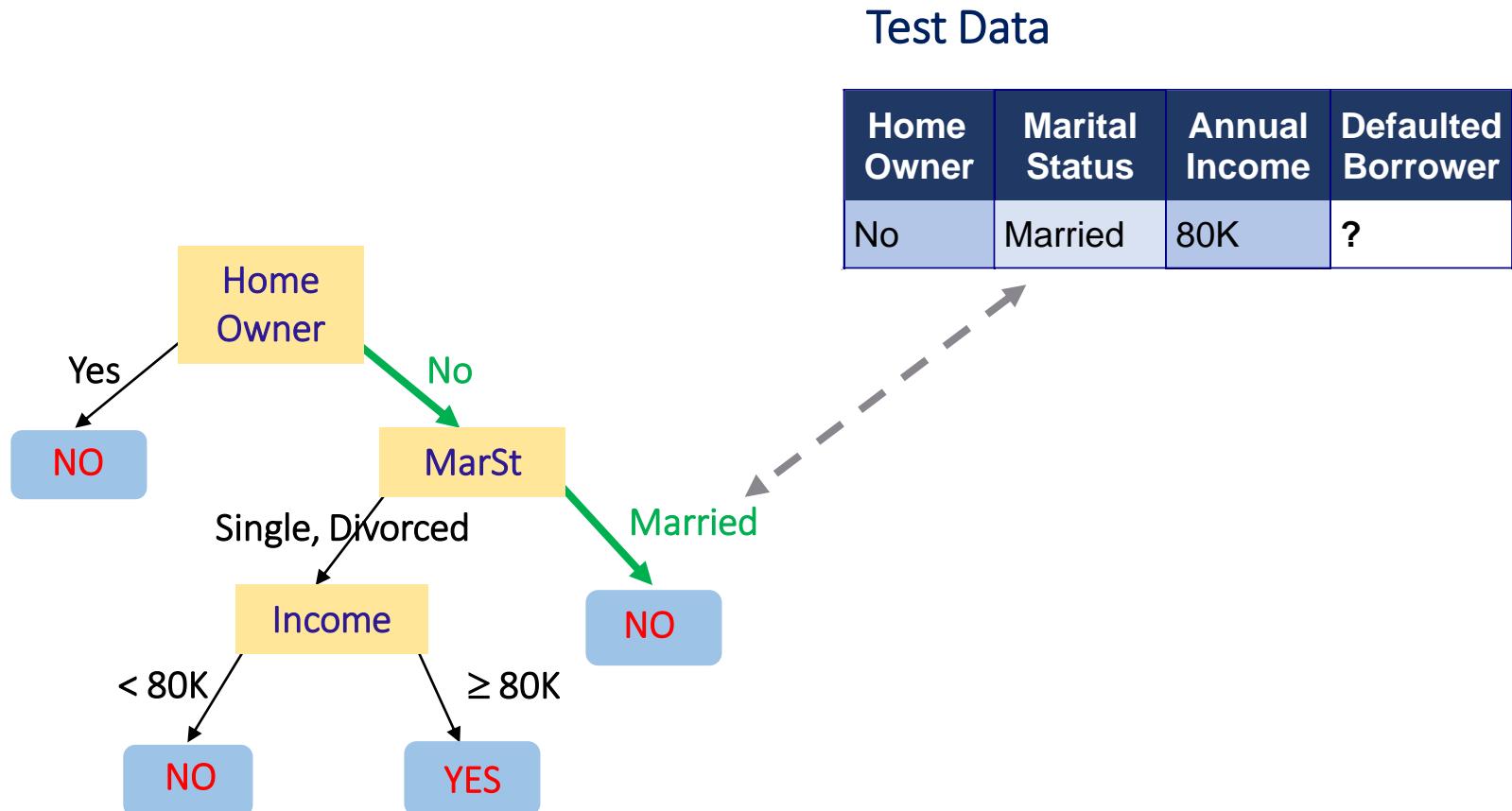
Classification Trees: apply model to test data



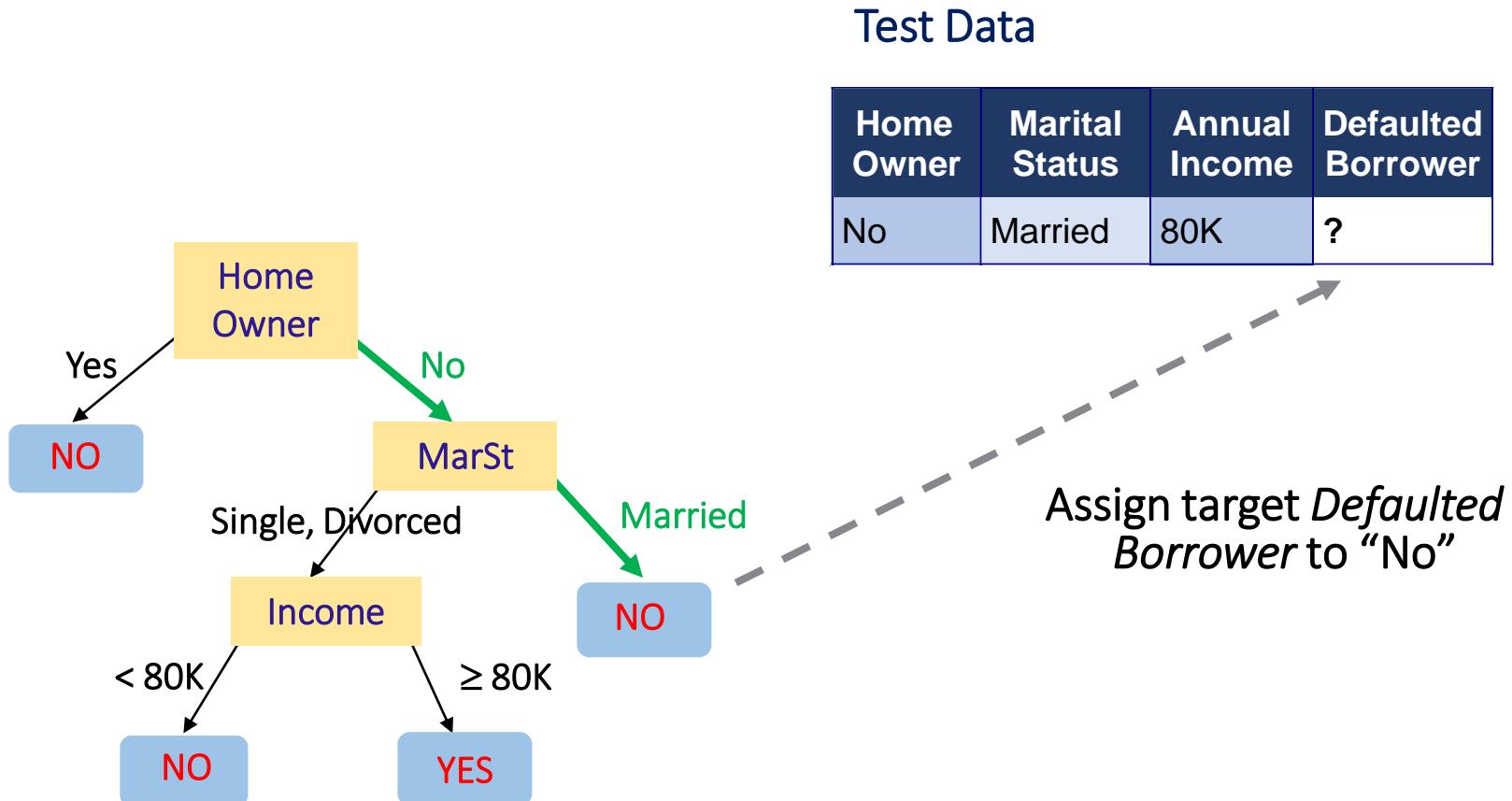
Classification Trees: apply model to test data



Classification Trees: apply model to test data



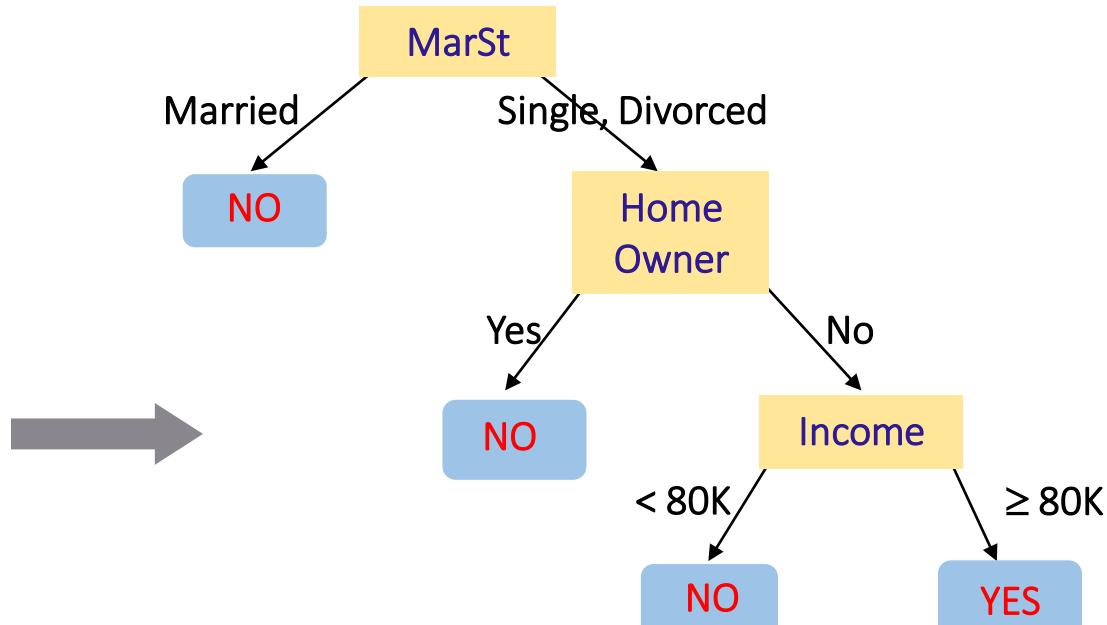
Classification Trees: apply model to test data



Classification Trees: same data, another DT

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower	
	categorical	categorical	continuous		target
1	Yes	Single	125K	No	
2	No	Married	100K	No	
3	No	Single	70K	No	
4	Yes	Married	120K	No	
5	No	Divorced	95K	Yes	
6	No	Married	60K	No	
7	Yes	Divorced	220K	No	
8	No	Single	85K	Yes	
9	No	Married	75K	No	
10	No	Single	90K	Yes	

Training Data



There could be more than one tree that fits the same data!

Decision Trees: advantages and disadvantages

Advantages

- Easy to interpret models
- Can cope with **any data type** (categorical and numeric)
- Don't require **feature scaling**
- Embedded **feature importance** and **handling of missing values**
- Relatively inexpensive to construct
- Extremely fast at classifying unknown examples

Disadvantages

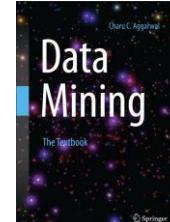
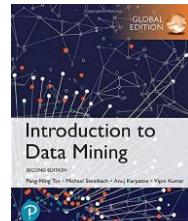
- **Not robust:** can be very sensitive to small variations in the data -> **High variance**
- **Very complex decision trees** usually have low **bias** (difficult to incorporate new data):
 - **Weak learners:** poor classification performance (specially if overfitting occurs)
 - **Key features might be hidden** by more dominant ones: **interacting** attributes (that can distinguish between classes together but not individually) may be passed over in favor of other attributed that are less discriminating.

Bibliography

Introduction to Data Mining, Pang-Ning Tan, Michael Steinbach, Anuj Karpatne, Vipin Kumar, *Pearson*, 2019 (chap 3.3, 3.4)

Data Mining, the Textbook, Charu C. Aggarwal, *Springer*, 2015 (chap 10.2, 10.3)

Xindong Wu, et al., **Top 10 algorithms in data mining**, *Knowl Inf Syst*, 14(1):1-37, 2007. <https://doi.org/10.1007/s10115-007-0114-2>



Data Mining

Predictive Modelling

Ensembles

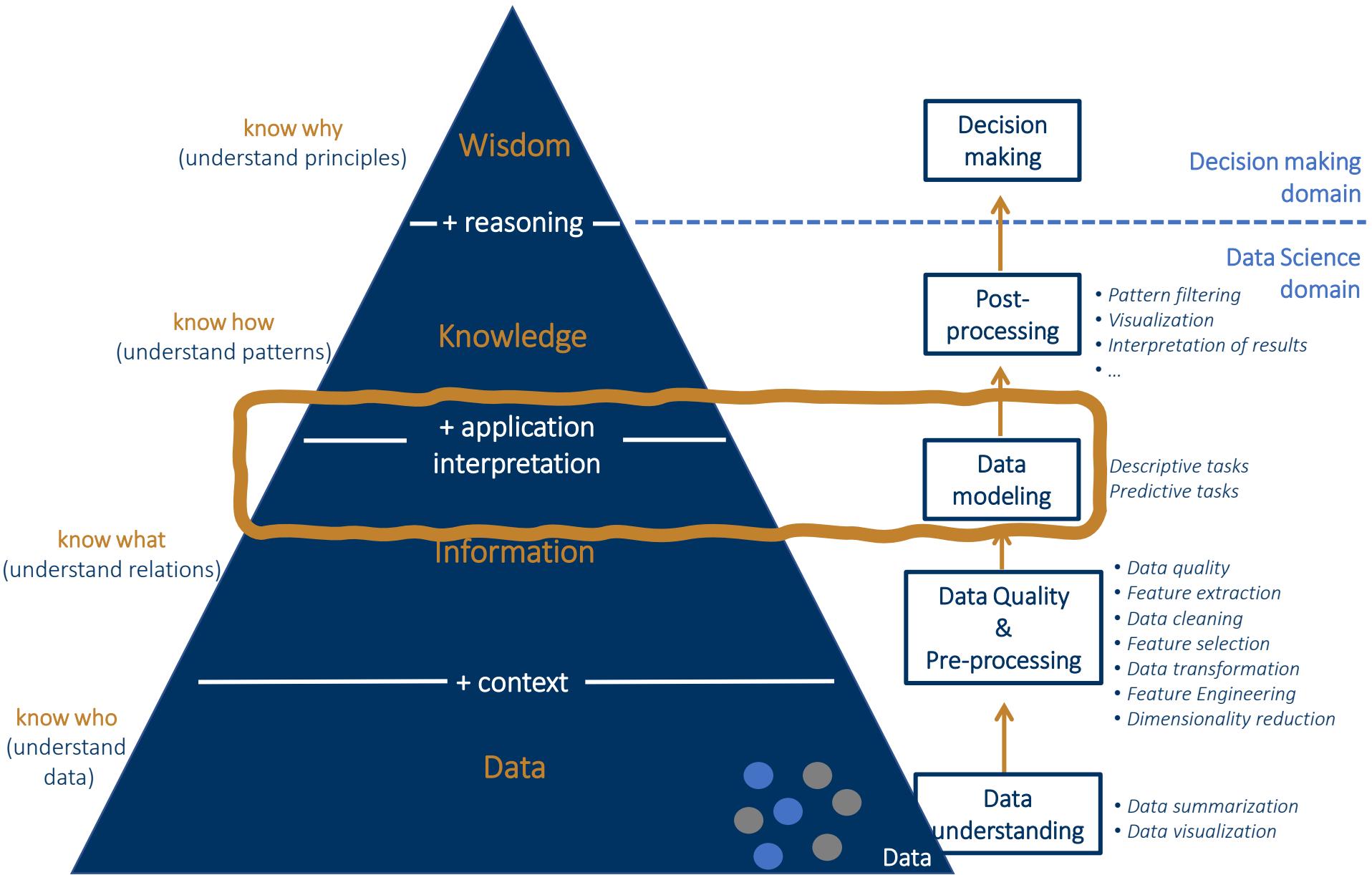
Raquel Sebastião

Departamento de Eletrónica, Telecomunicações e Informática

Universidade de Aveiro

raquel.sebastiao@ua.pt

2022/2023



Prediction Models – approaches

Geometric approaches

- Distance-based: kNN
- Linear models: Fisher's linear discriminant, perceptron, logistic regression, SVM (w. linear kernel)

Probabilistic approaches

- naive Bayes, logistic regression

Logical approaches

- classification or regression trees, rules

Optimization approaches

- neural networks, SVM

Sets of models (ensembles)

- random forests, adaBoost

Contents

- Ensembles
- Types of ensembles
- Ensembles methods
- Summary

Ensemble models

Why?

- For complex problems it is hard to find a model that “explains” all observed data

What?

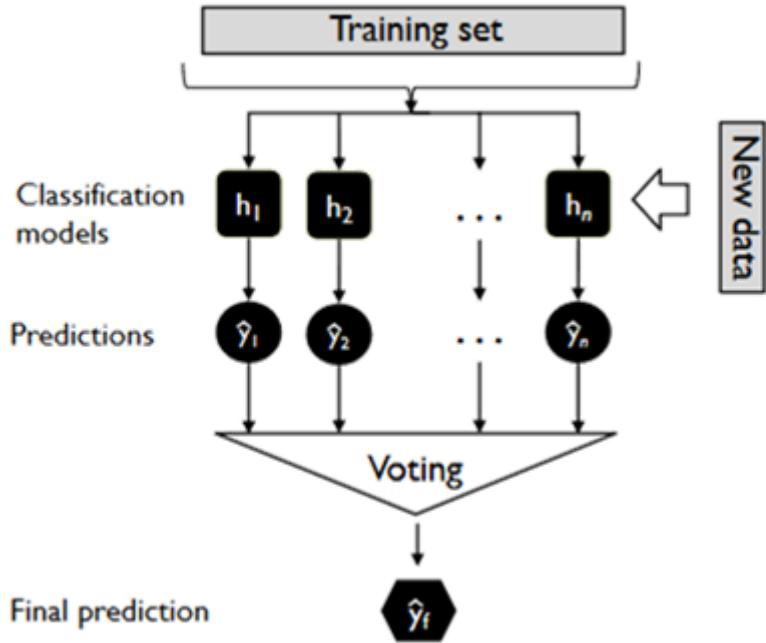
- Ensembles are collections of models that are used together to solve a prediction problem

How?

- Construct a set of base models learned from different samples of the training data
- Use a combination of models to increase accuracy
 - Predict class label/value of new cases by combining the predictions made by multiple models (majority vote/averaging)

Ensemble classifiers

Image credits: Sebastian Raschka



Classification

- Majority vote of the classifiers
- Each classifier has an error rate better than chance. For binary classification:

$$\{\epsilon_1, \epsilon_2, \dots, \epsilon_n\}, \quad \epsilon_t < 0.5$$

If all classifiers are independent (errors are uncorrelated):

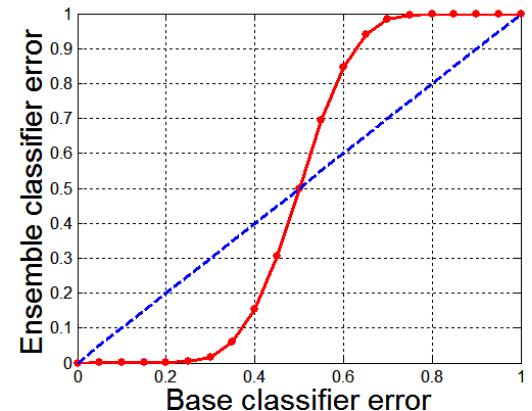
- Error rate of ensemble = probability of having more than half of base classifiers being wrong:

$$\epsilon_{ensemble} = \sum_{i=[n/2]}^n \binom{n}{i} \epsilon^i (1 - \epsilon)^{n-i}$$

Ensemble classifiers

An ensemble of classifiers improves over individual classifiers iif (Dietterich 2002):

- they perform better than random guess
- they have uncorrelated errors
- they commit errors in different regions of the instance space



Classification error for an ensemble of 25 base classifiers, assuming their errors are uncorrelated

Ensemble methods work best with **unstable base classifiers**

- Classifiers that are **sensitive to minor perturbations in training set**, due to *high model complexity*
- Examples: Unpruned decision trees, ANNs, ...

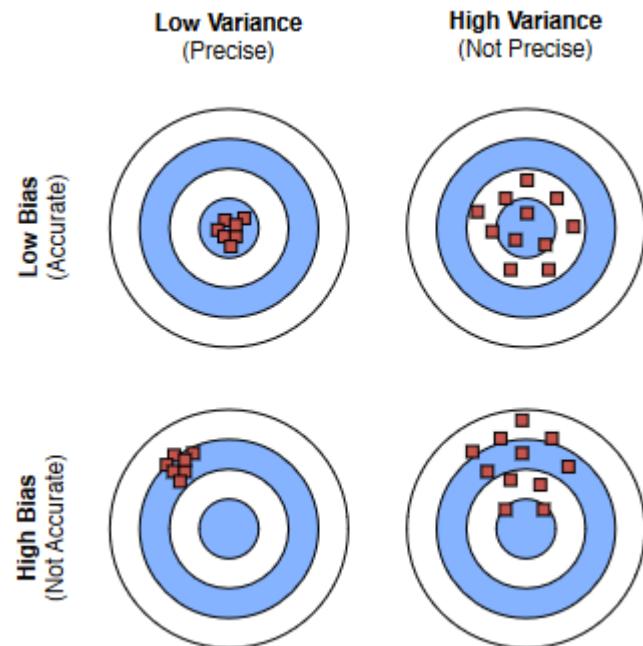
Ensemble classifiers: construction

- By manipulating the training set
 - Homogeneous models (e.g.: Bagging, Boosting)
 - Heterogeneous models (e.g.: Cascading, Stacking)
- By manipulating the input features
 - e.g.: random forests
- By manipulating the class labels
 - E.g.: error-correcting output coding
- By manipulating the learning algorithm
 - Example: injecting randomness in the initial weights of ANN

Bias-variance trade-off

The **generalization error** of a model can be split in two main components:

- the bias and the variance components

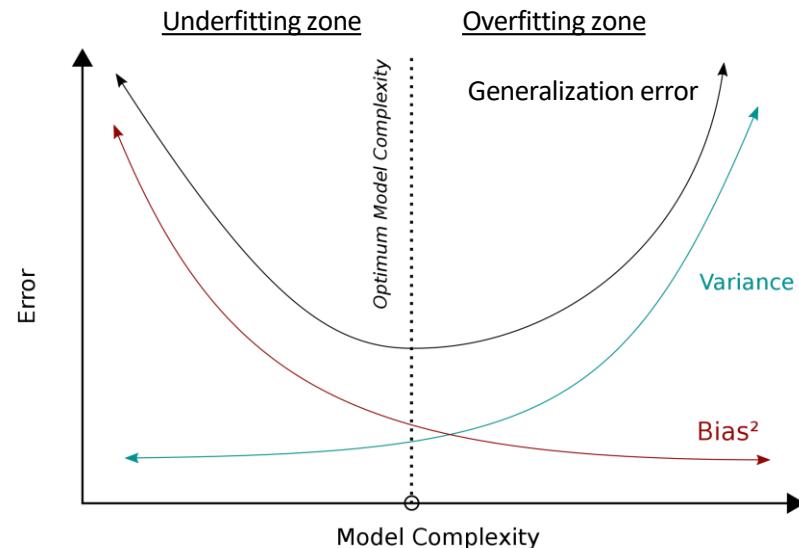


- **Bias:** error that is due to the **poor ability of the model to fit the seen data**
- **Variance:** error related to the **sensibility of the model to the given training data**

Image credits: Sebastian Raschka

Bias-variance trade-off

- Decreasing the bias by adjusting more to the training sample → higher variance: over-fitting
- Decreasing the variance by being less sensitive to the given training data → higher bias



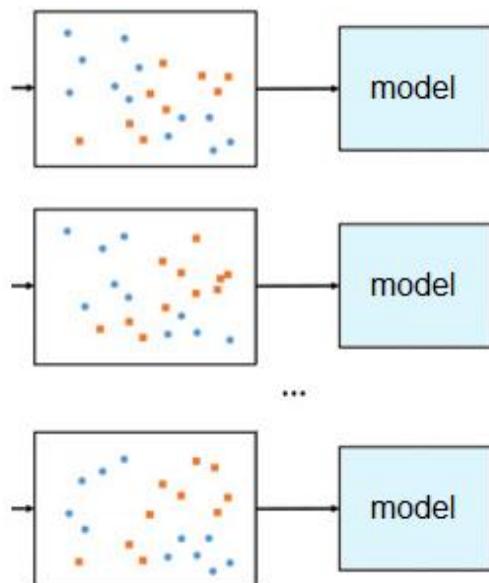
Ensembles can reduce both components of the generalization error!

Contents

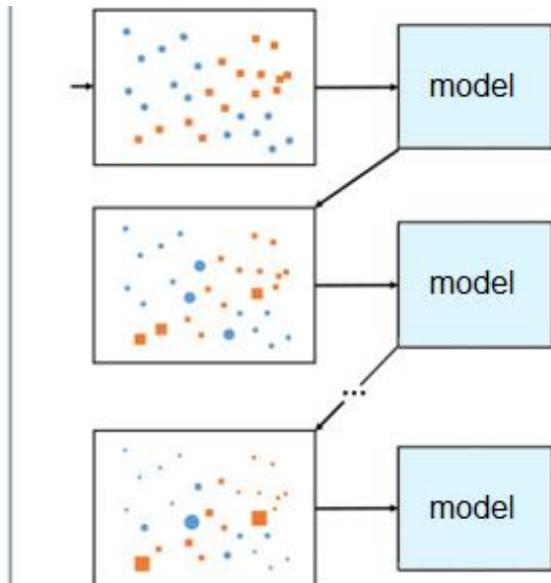
- Ensembles
- Types of ensembles
 - Bagging
 - Boosting
- Ensembles methods
- Summary

Types of ensembles

- Independent or Parallel models
- Iterative or Sequential models



Independent or Parallel models



Iterative or Sequential models

Sergio González, et al., A practical tutorial on bagging and boosting based ensembles for machine learning: Algorithms, software tools, performance study, practical perspectives and opportunities, *Information Fusion*, 64:205-237, 2020.

Types of ensembles

Independent or Parallel models:

- Models are trained in **parallel**
- Each model is trained with a **bootstrap sample** of the data set
- Global decision: the class is predicted by **voting/averaging of the models**

Iterative or Sequential models:

- Models are trained in **sequence**
- Each model is trained by **emphasizing the training samples that previous models misclassified**
 - Initially, all N samples are assigned equal weights
 - weights may change at the end of each boosting round
- Global decision: the class is predicted by **voting/averaging but the models contributions depend in their performance**

Types of ensembles: similarities & differences

Similarities

- Ensemble methods to get n models
- Generate several training data sets by random sampling
- Reduce variance and provide higher stability

Differences

- Bagging: they are built independently; Boosting tries to add new models that do well where previous models fail
- Boosting determines weights for the data to increase (boost) performance for the most difficult examples
- **Gobal decision** – Bagging: **equally weighted voting/averaging of the models**; Boosting: **voting/averaging depending on models's performance** (models with better performance on training data contribute more in predicting)
- **Boosting tries to reduce bias**
- **Bagging may solve the over-fitting problem, while Boosting can increase it**

Contents

- Ensembles
- Types of ensembles
- Ensembles methods
 - Bagging
 - Random Forest
 - Out-Of-Bag Error
 - Variable/Feature importance
 - AdaBoost
 - XGBoost
- Summary

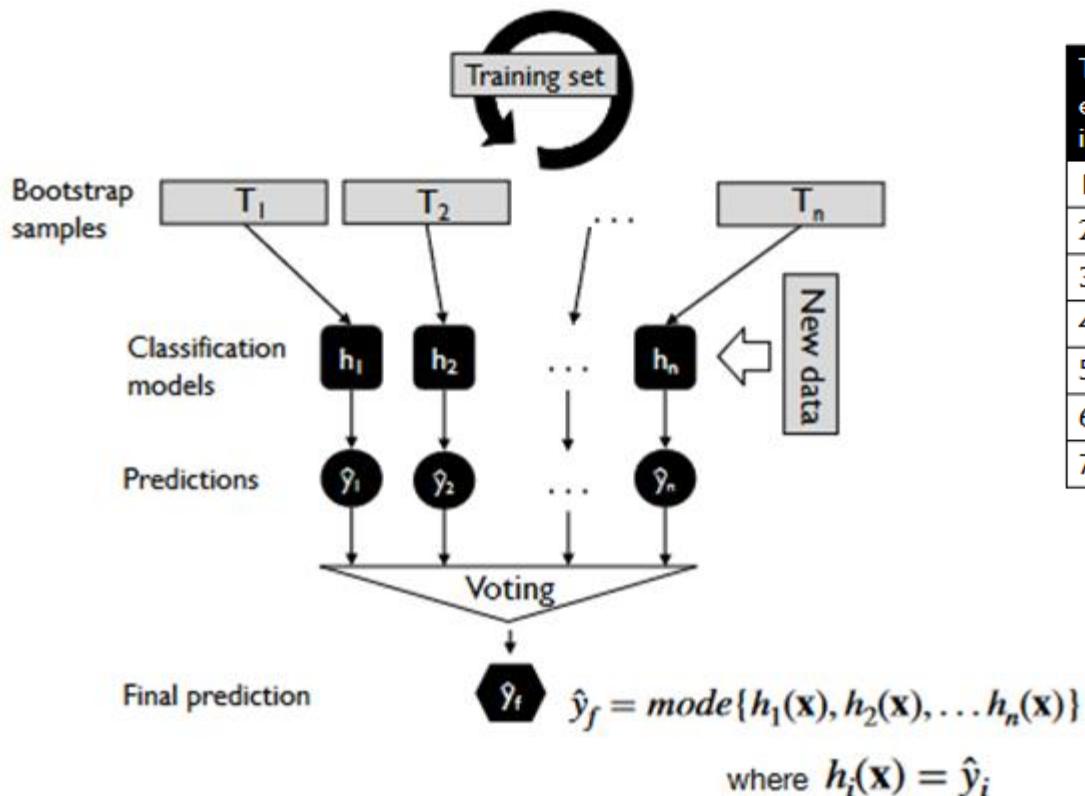
Ensembles methods

- **Bagging** (Breiman 1996): Fit many models to bootstrap-resampled versions of the training data, and classify by majority vote
- **AdaBoost** (Freund and Schapire 1996): Fit many weak learners to reweighted versions of the training data while increase (boosting) the efficiency, and classify by weighted majority vote
- **Gradient Boosting** (Friedman 2000): Employs Gradient Descent algorithm to minimize errors in sequential models, and classify by weighted majority vote
- **Random Forests** (Breiman 2001): Bagging w. trees + random feature subsets
- **XGBoost** (Chen and Guestrin 2016): It is an ideal combination of hardware and software optimization techniques to achieve superior results by utilizing minimal computing resources in short periods

Ensembles using independent models:

Bagging

Bagging or Bootstrap Aggregating (Breiman 1996)



Training example indices	Bagging round 1	Bagging round 2	...
1	2	7	...
2	2	3	...
3	1	2	...
4	3	1	...
5	7	1	...
6	2	7	...
7	4	7	...

Below the table, arrows point from the columns to the corresponding models: h_1 , h_2 , and h_n .

Image credits: Sebastian Raschka

If the base learner has a high variance (i.e. very sensitive to variations on the training sample), this procedure ensures diversity among the n models

Ensembles using independent models:

Bagging

- Obtains a set of n models using **different bootstrap samples** of the given training data
 - for each model a **sample with replacement** of the same size as the available data is obtained
 - this means that for each model there is a small proportion of the examples that will be different
- Bagging should be applied to base learners with **high variance**:
 - Decision trees, Rule learners, etc
 - Easy to implement with any algorithm
 - Easy to implement in parallel environments
- **The bias-variance argument**:
 - error decreases due to reduction in the variance component

Ensembles using independent models: Random Forests

Random Forests (Breiman 2001): Bagging w. trees + random feature subsets

- Ensemble classification (and regression) algorithm based on **decision trees**
- Construct an ensemble of **decision trees** by manipulating **the training set** and the **input features**
 - Use bootstrap sample to train every decision tree (similar to Bagging)
 - At every internal node of DT, **randomly sample feature subsets** (p attributes) for selecting split criterion
- Easy to implement
- Very effective
- **Reduces variance of unstable classifiers without negatively impact the bias** (good generalization properties)
- Algorithm outputs **more information than just class label/value**

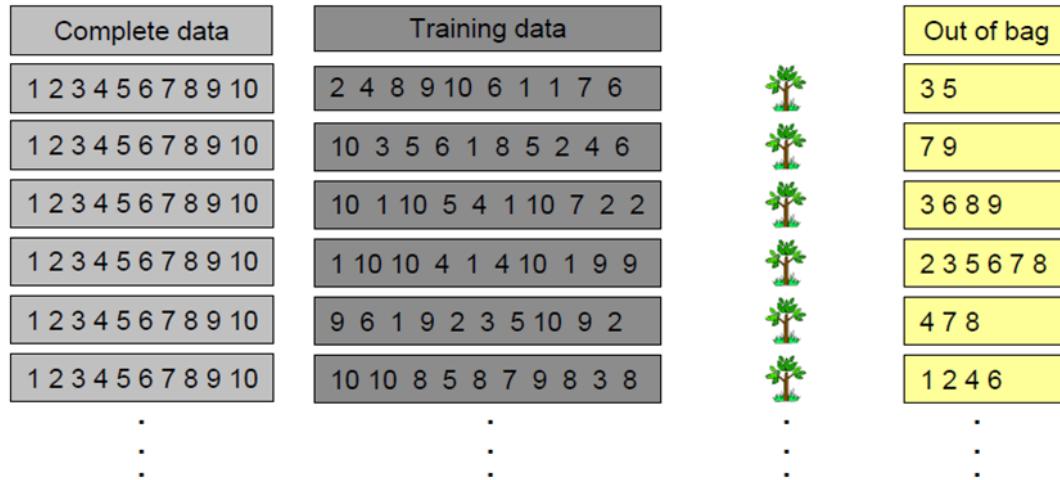
Ensembles using independent models: Random Forests

Combine weak learners (unpruned trees) and generate a classification algorithm with good performance

- Each weak learner is a decision tree trained in **parallel** with
 - Random training set: a **bootstrap sample**
 - Random **selection of features** at each node
 - Typically, the number of selected features is $m = \sqrt{D}$ or $m = \log_2(D)$, where D is the number of features
- For each tree grown on a bootstrap sample, the **error rate** for **examples left out of the bootstrap sample** is monitored
- **Out-of-Bag** (OOB): test set for testing performance

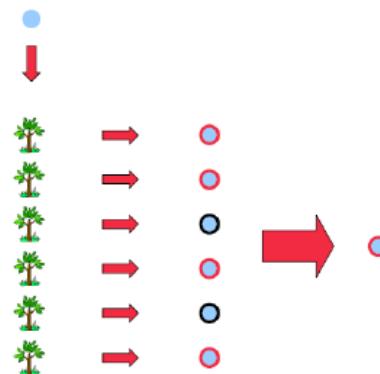
Ensembles using independent models: Random Forests

The examples not included in the bootstrap sample form the Out-Of-BAG (OOB) set



Classification rule:

- each new sample is **classified by all trees**
- final decision by **majority vote**

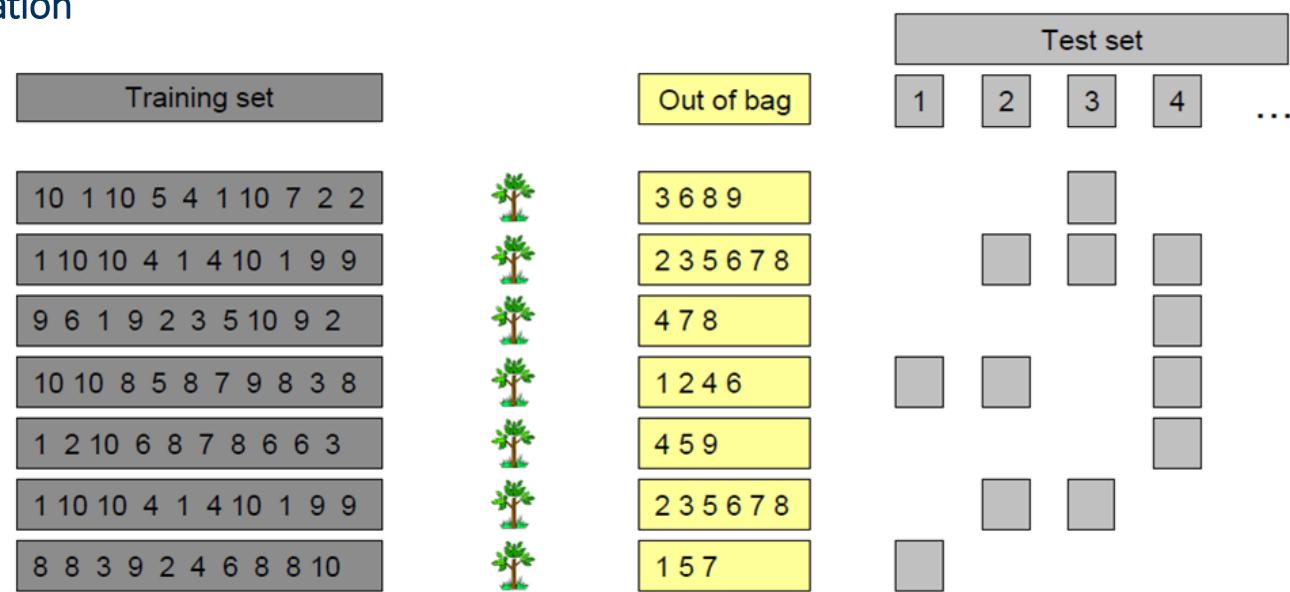


Ensembles using independent models: Random Forests

Out-Of-Bag error:

Estimating the TEST error (during train phase): for each example x in data set

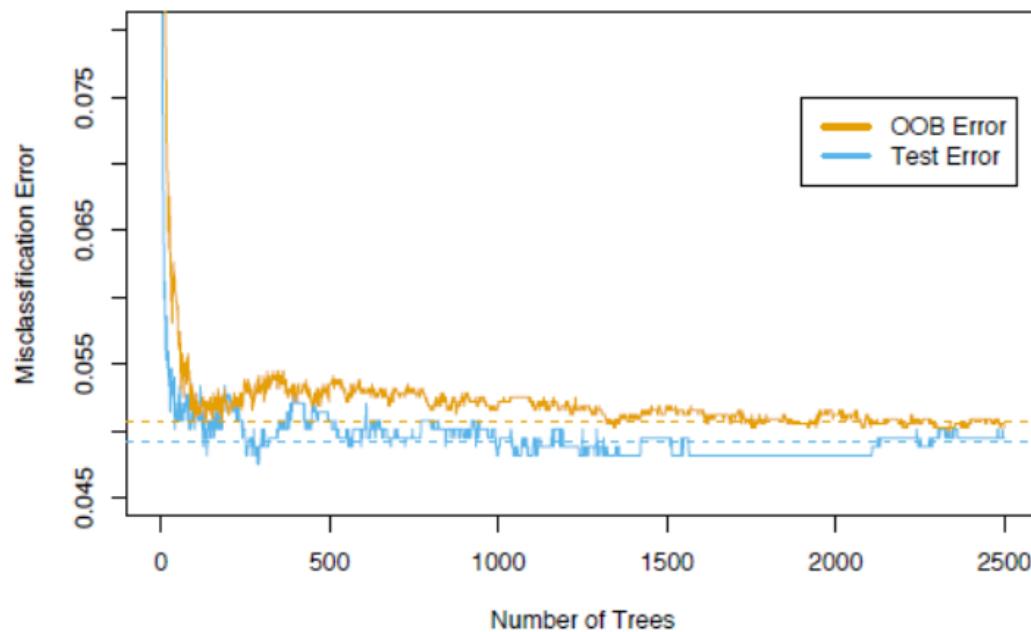
- Predicting the response of the x using the trees in which $x \in$ OOB set
 - Good estimate for the **generalization error** because the information provided by x was not used for building these trees
 - No hold-out or cross-validation



Ensembles using independent models: Random Forests

Out-Of-Bag error:

- Misclassification error Out-Of-Bag error versus test error (hold-out method)



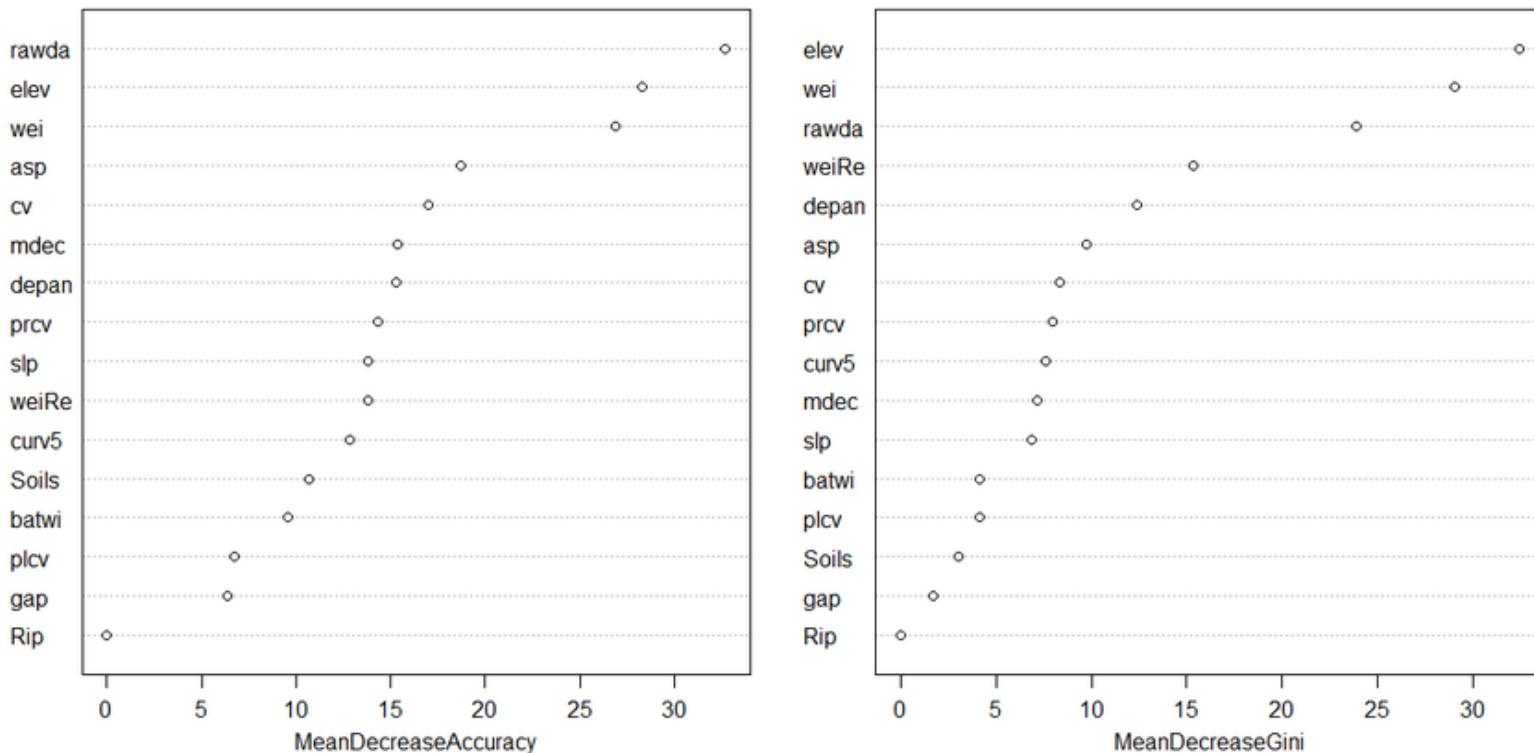
Ensembles using independent models: Random Forests

Feature importance

- The use of forests of trees to **evaluate the importance of features** on a classification task:
 - Which variables have the most predictive power?
- The more often used measures are:
 - **Mean Decrease in Impurity:** the average of the decrease in impurity over all nodes (all trees) where the feature is used for a rule
 - how much the impurity decreases when the variable is chosen to split a node?
 - **Mean decrease on the accuracy/mean square-error increases** in out-of-bag subset. After **randomly permute the feature** in the feature vector the decrease in accuracy is calculated
 - how much the accuracy decreases / mean square error increases when the variable is excluded?

Ensembles using independent models: Random Forests

Feature importance



Sheng-Guo Wang, et al., **Random Forest Classification and Automation for Wetland Identification based on DEM Derivatives**. 2015

Ensembles using independent models: Random Forests

Advantages

- Do not require elaborate tuning of the hyper-parameters. Often these can/should be optimized
- The most important parameter to tune is the **number of trees to grow**, typically the larger the best
- Do not need to worry about creating very complex trees
- Less prone to overfitting (than DT)
- Slower than DT, but faster than typical bagging or boosting
- Out-Of-Bag error (cross-validation is not needed)

Disadvantages

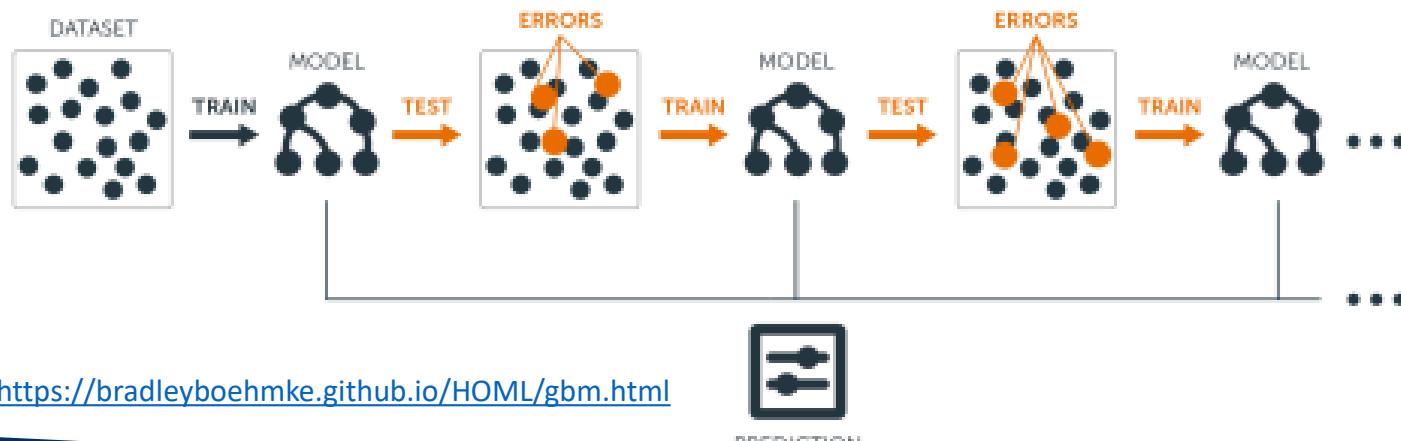
- Do not provide the interpretability level of a Decision Tree

Ensembles using iterative models:

AdaBoost

AdaBoost or Adaptive Boosting (Freund and Schapire 1996)

- AdaBoost was the first successful boosting algorithm developed for binary classification
- AdaBoost is best used to boost the performance of decision trees on binary classification problems
- Is used for classification rather than regression
- Can be used to boost the performance of any machine learning algorithm



Ensembles using iterative models:

AdaBoost

AdaBoost or Adaptive Boosting (Freund and Schapire 1996)

Ensemble of weak classifiers trained sequentially, $f_t = 1, \dots, n$

Goal: Train each classifier given the performance of **previous weak classifiers**

- At each step t
 - **Modify training** sample distribution in order to favor difficult examples (according to previous weak classifiers)
 - Train a **new weak classifier** f_t
 - Select the **new weight** α_t by optimizing a global criterion
 - **Stop** when impossible to find a weak classifier satisfying the simplest condition (being better than chance)
- Final classifier is the **combination** (with weights α_t) of all n classifiers
 - Assigns weights to the N classifiers: a classifier with good a classification result on the training data will contribute more than a poor one

Ensembles using iterative models:

AdaBoost

Most popular algorithm in the family of boosting algorithms

- Boosting: the performance of simple (weak) classifiers is boosted by combining them iteratively (usually **Decision Tree Stumps**)
- **Combination rule**

$$g(\mathbf{x}) = \sum_{t=1}^n \alpha_t f_t(\mathbf{x}), \text{ where } \alpha_t \text{ means the importance of classifier } f_t$$

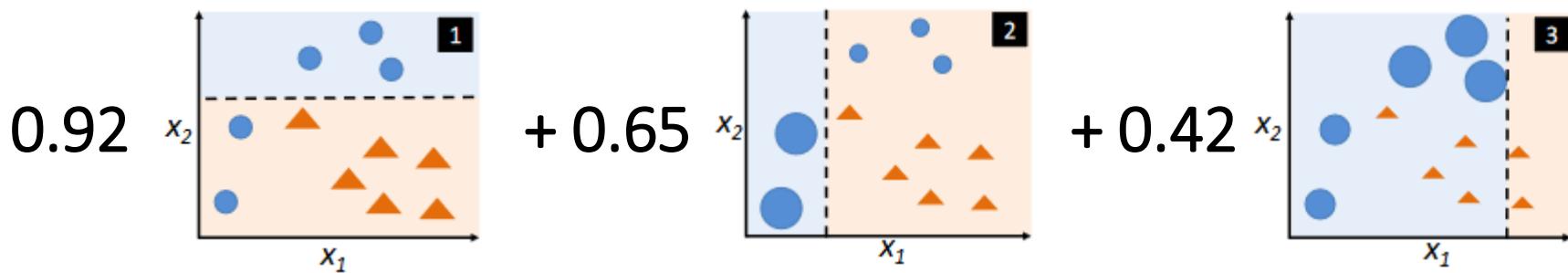
- Simplest framework: binary classification, each $f_1 = \{-1, +1\}$
- The following simplest requirement: **each weak classifier f_t should perform better than chance**

Ensembles using iterative models:

AdaBoost

Binary Classification problem: with 3 trained weak learners

$$g(\mathbf{x}) = \text{sign}(\alpha_1 f_1(\mathbf{x})) + \text{sign}(\alpha_2 f_2(\mathbf{x})) + \text{sign}(\alpha_3 f_3(\mathbf{x}))$$



$$f_i(\mathbf{x}) = \begin{cases} +1, & \mathbf{x} \in \text{blue} \\ -1, & \mathbf{x} \in \text{pink} \end{cases}$$

Image credits: Sebastian Raschka

Ensembles using iterative models:

GBM

Gradient Boosting Machine (Friedman 2000)

- Boosting problem as an **optimization** problem
- **Sequential** ensemble learning
- Base learners are generated **sequentially** in such a way that the **present base learner is always more effective than the previous one**
- **Weights for misclassified** outcomes are **not incremented**
- At each step, adds another weak learner to increase the performance and build a strong learner
- Final classifier is the **equally weighted combination** of all n classifiers, but their predictive capacity is restricted with learning rate to increase accuracy

Ensembles using iterative models:

AdaBoost vs GBM

Adaboost

An additive model where shortcomings of previous models are identified by high-weight data points.

The trees are usually grown as decision stumps.

Each classifier has different weights assigned to the final prediction based on its performance.

It gives weights to both classifiers and observations thus capturing maximum variance within data.

Gradient Boost

An additive model where shortcomings of previous models are identified by the gradient.

The trees are grown to a greater depth usually ranging from 8 to 32 terminal nodes.

All classifiers are weighed equally and their predictive capacity is restricted with learning rate to increase accuracy.

It builds trees on previous classifier's residuals thus capturing variance in data.

Ensembles using iterative models:

XGBoost

eXtreme Gradient Boosting (XGBoost) (Chen and Guestrin 2016)

- An advanced version of Gradient Boosting Method
- Software and hardware optimization
 - a scalable tree boosting system

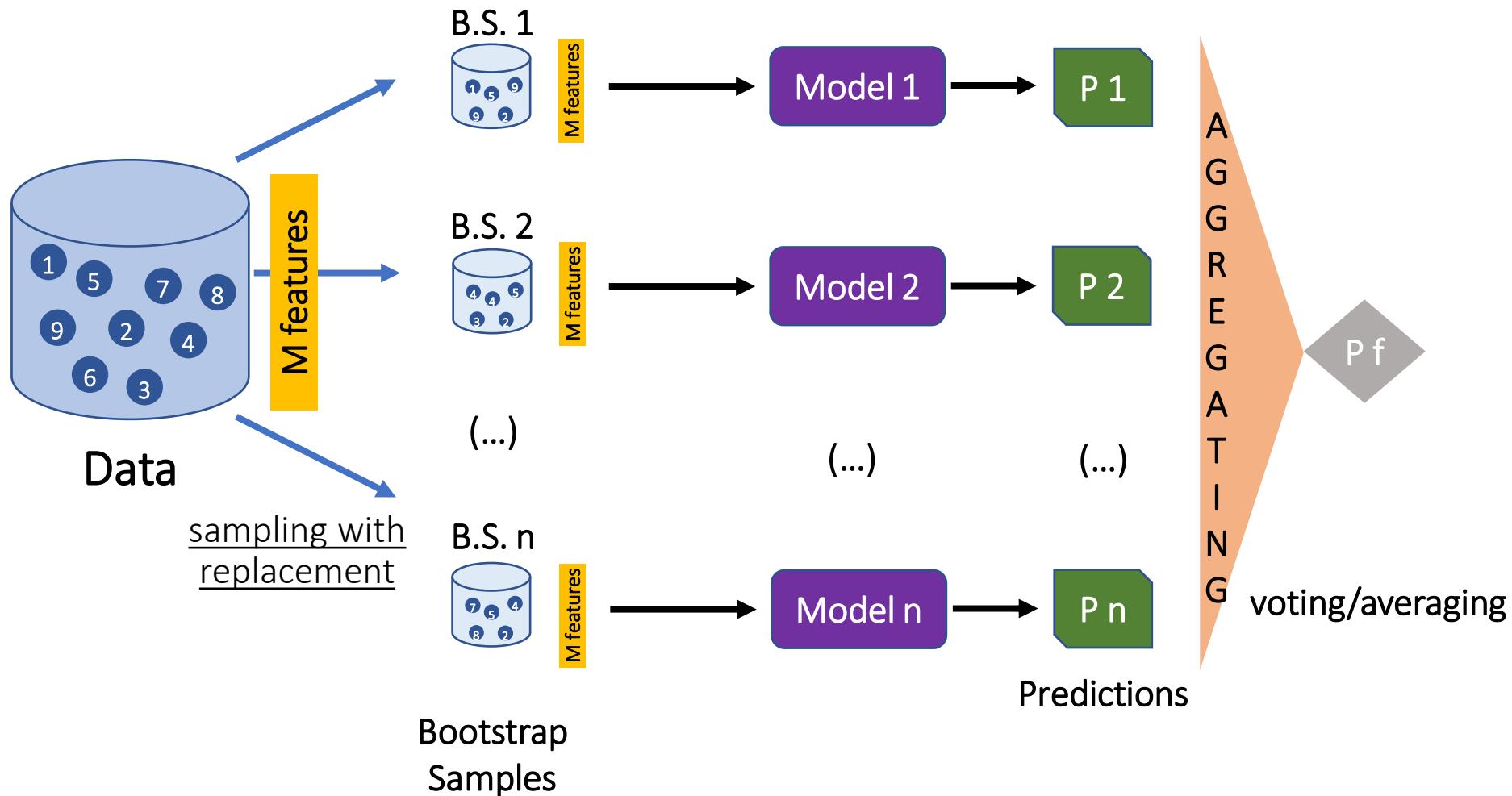
Some features:

- clever penalization of trees: weights of the trees that are calculated with less evidence is shrunk more heavily
- extra randomization parameter to reduce the correlation between the trees
- parallelization, cache optimization, distributed computing, etc

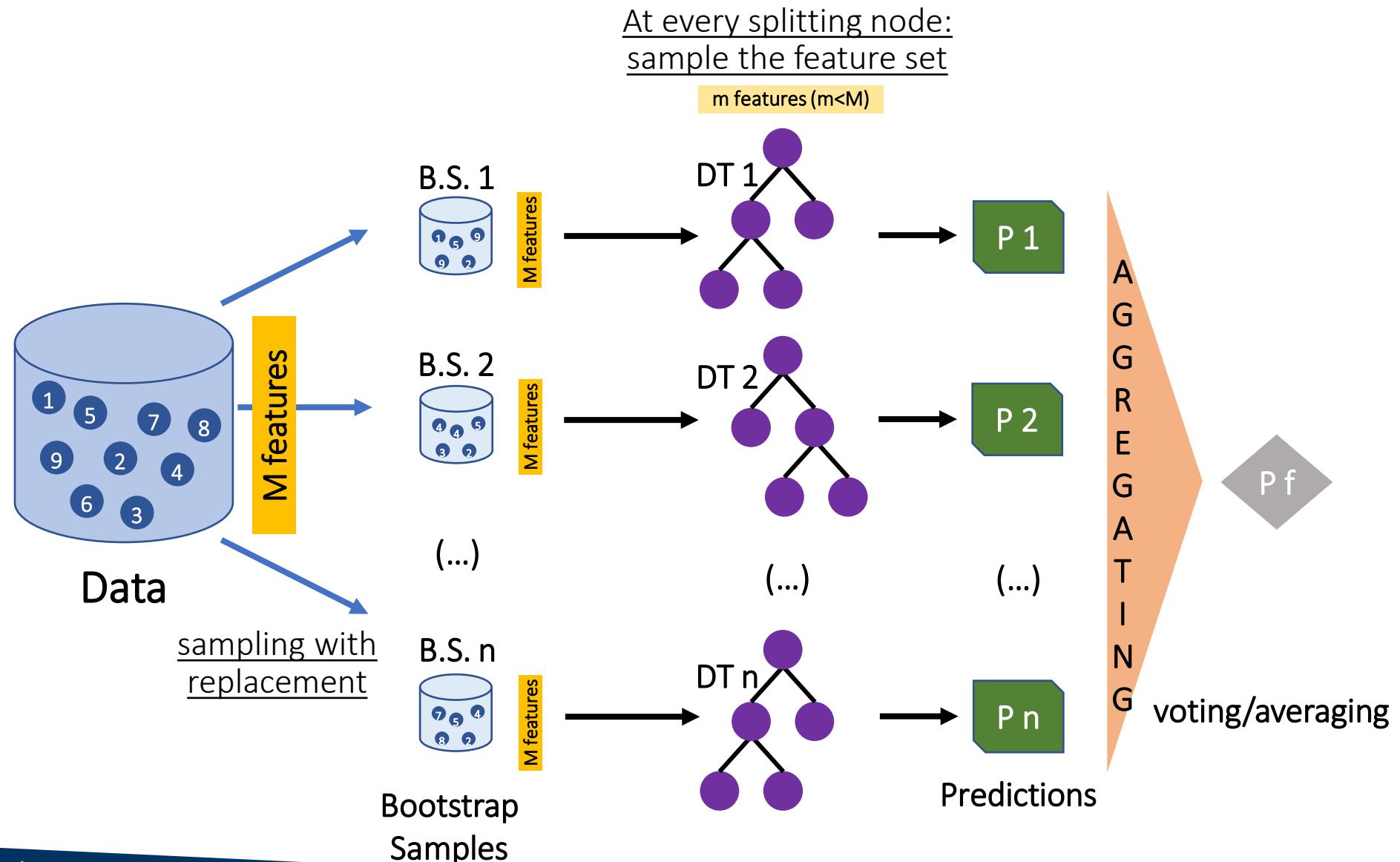
Contents

- Ensembles
- Types of ensembles
- Ensembles methods
- **Summary**

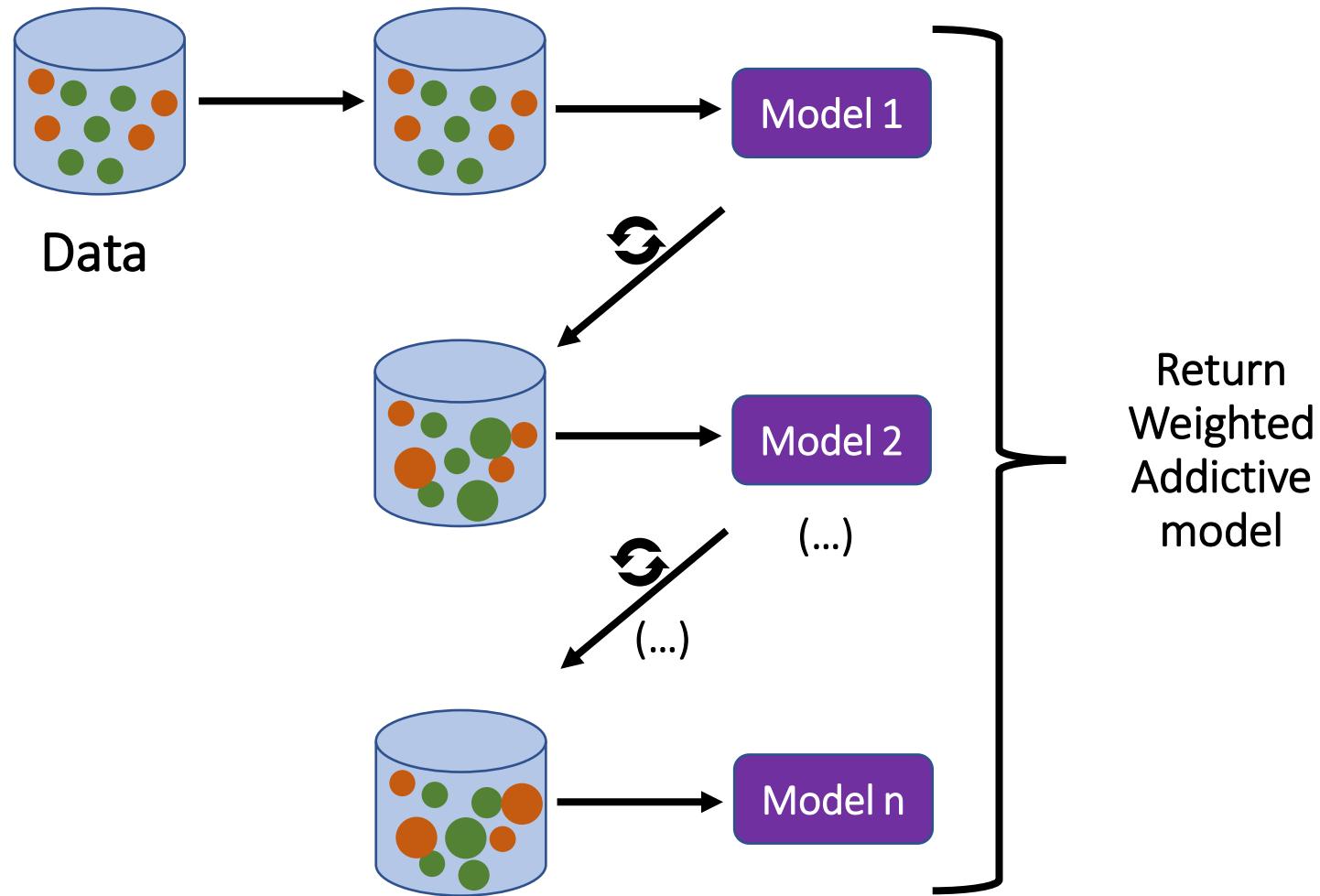
Summary: Bagging



Summary: Random Forests

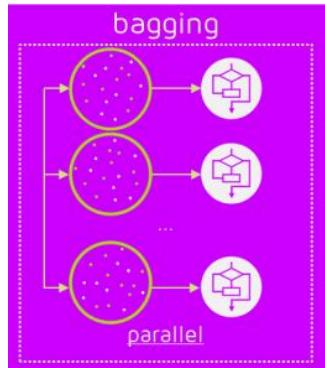


Summary: AdaBoost



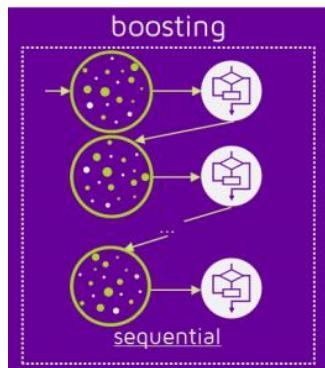
Summary: Bagging & Boosting

Training stage:



Bagging Methods :

- parallel
- bootstrap samples



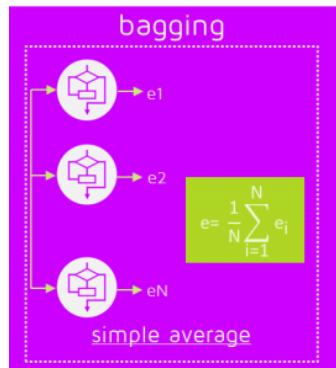
Boosting Methods :

- Sequential
- Increase of the **weights of misclassified data** to emphasize the most difficult example
 - subsequent learners will focus on them during their training

<https://quantdare.com/what-is-the-difference-between-bagging-and-boosting/>

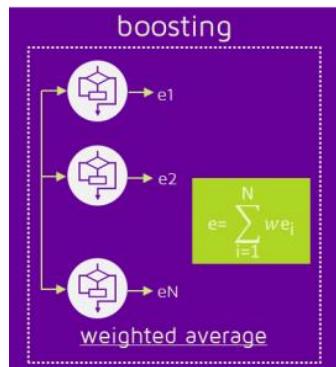
Summary: Bagging & Boosting

Prediction stage: apply the n learners to the new observations



Bagging Methods :

- Prediction is obtained by **equally** voting/averaging the responses of the n learners



Boosting Methods :

- Prediction is obtained by **voting/averaging but the models contributions depend in their performance**

<https://quantdare.com/what-is-the-difference-between-bagging-and-boosting/>

Summary: Bagging & Boosting

Bagging Methods

- Error reduction due to reduction in variance
- Effective with unstable models

⇒ single model is **over-fitting** → Bagging ensembles can get reduced variance

Boosting Methods

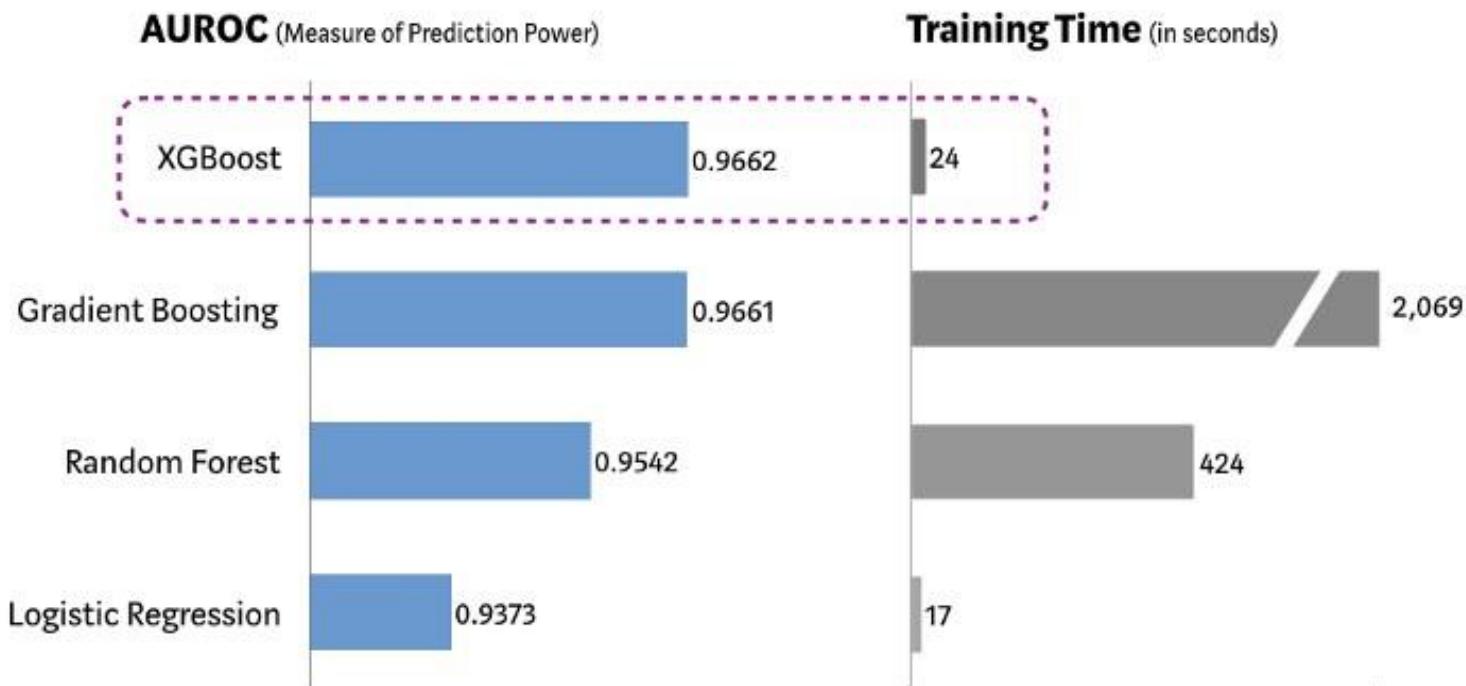
- Error reduction due to reduction in bias and variance
- Risky in problems with noise (increase of the error)
- more prone to over-fitting

⇒ single model with **low performance** → Boosting ensembles can get a **better bias**

Summary: comparision

Performance Comparison using SKLearn's 'Make_Classification' Dataset

(5 Fold Cross Validation, 1MM randomly generated data sample, 20 features)



<https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d>

Bibliography

Introduction to Data Mining, Pang-Ning Tan, Michael Steinbach, Anuj Karpatne, Vipin Kumar, *Pearson*, 2019 (chap 6.10)

Data Mining, the Textbook, Charu C. Aggarwal, *Springer*, 2015 (chap 11)

https://sebastianraschka.com/pdf/lecture-notes/stat451fs20/07-ensembles_slides.pdf

Breiman, L. (1996). Bagging predictors. *Machine learning*, 24(2), 123-140

Breiman, Leo. “Random Forests” *Machine learning* 45.1 (2001): 5-32.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1), 119-139

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, 1189-1232.

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785-794). ACM.

Data Mining

Predictive Modelling

Artificial Neural Networks

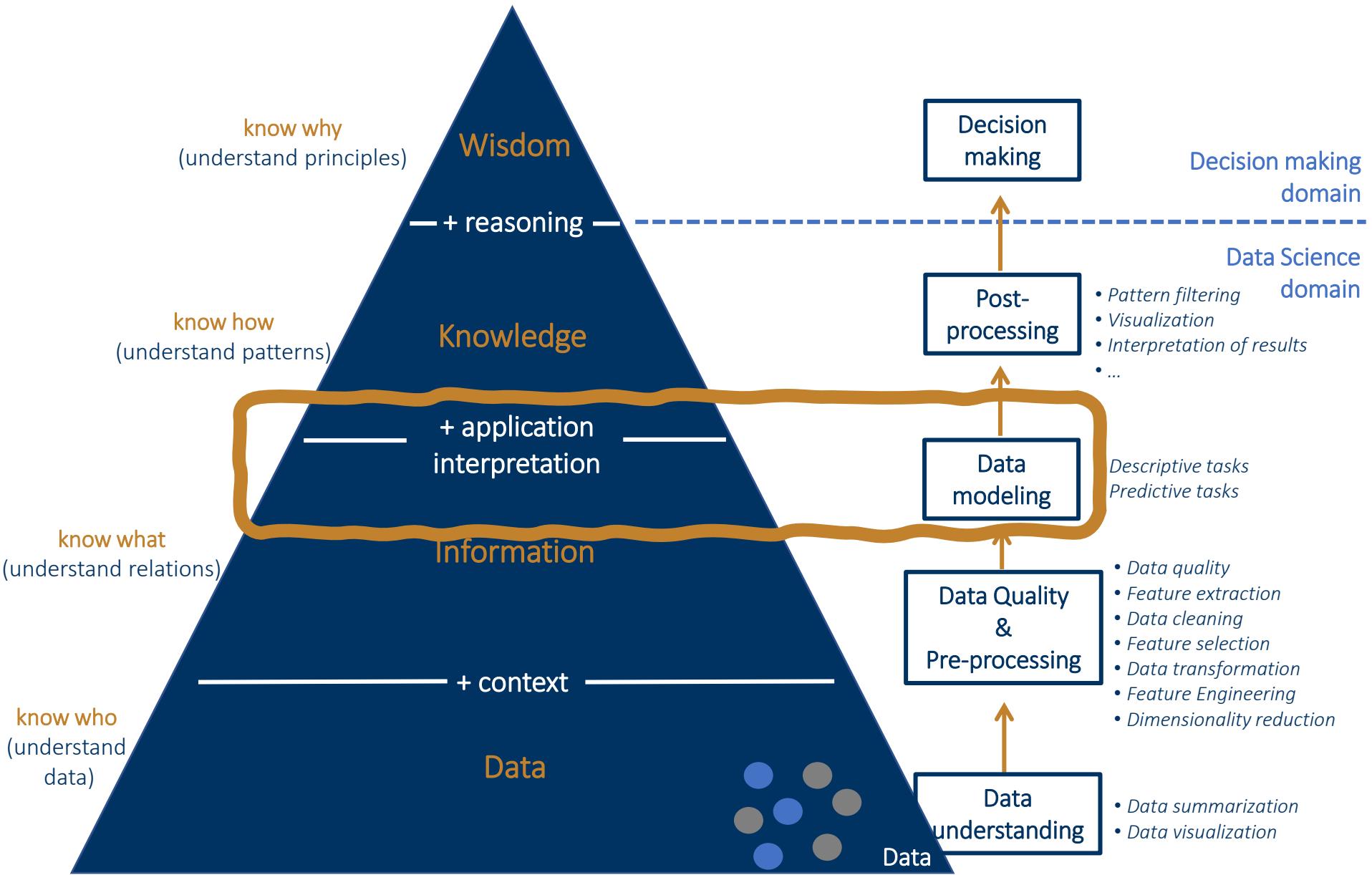
Raquel Sebastião

Departamento de Eletrónica, Telecomunicações e Informática

Universidade de Aveiro

raquel.sebastiao@ua.pt

2022/2023



Prediction Models – approaches

Geometric approaches

- Distance-based: kNN
- Linear models: Fisher's linear discriminant, perceptron, logistic regression, SVM (w. linear kernel)

Probabilistic approaches

- naive Bayes, logistic regression

Logical approaches

- classification or regression trees, rules

Optimization approaches

- neural networks, SVM

Sets of models (ensembles)

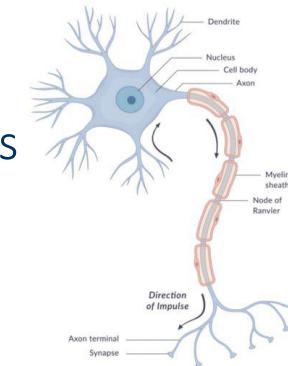
- random forests, adaBoost

Contents

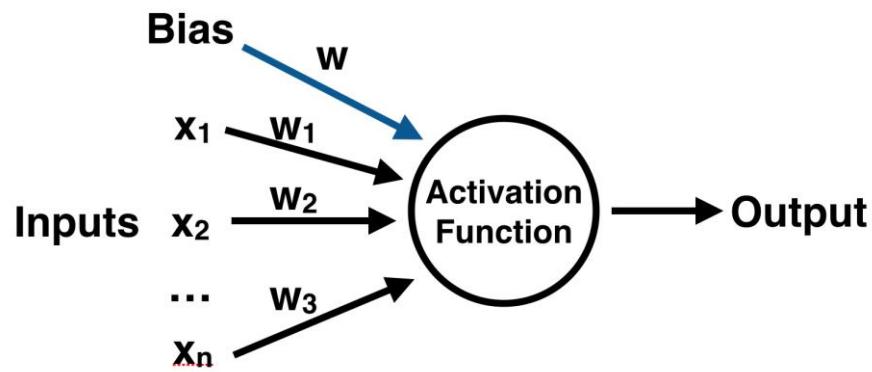
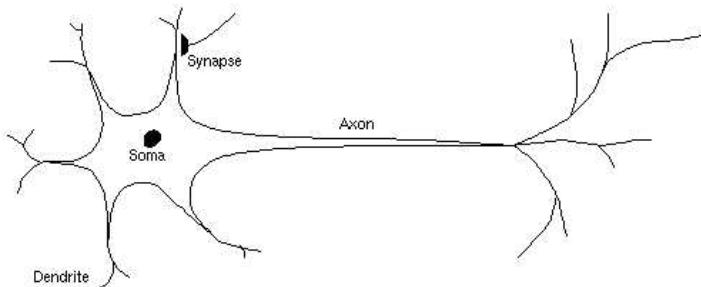
- Artificial Neural Networks
- Deep Learning (very short-introduction)

Artificial Neural Networks (ANN): Biological inspiration

- unit (neuron) has multiple inputs and one output
- network composed of highly interconnected processing units
- the weights of the connections are the adaptive elements
- inspired on brain structure



The computational model of the unit (neuron)



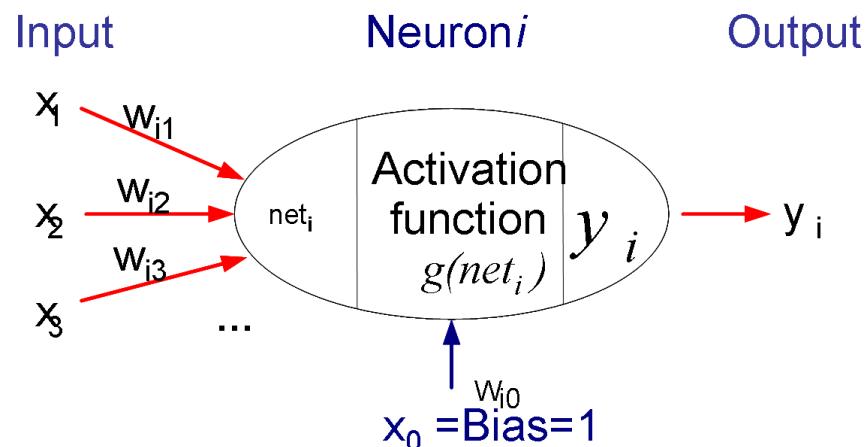
Artificial Neural Networks (ANN)

Each unit

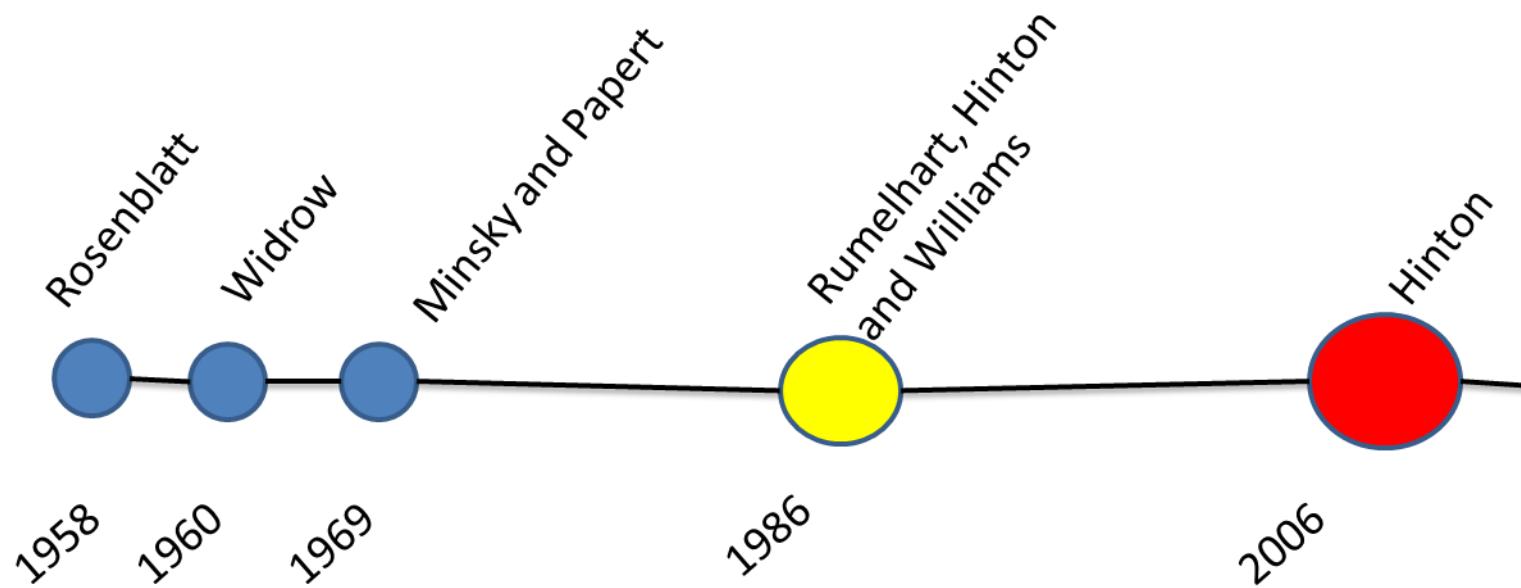
- receive the **input** impulses and calculate its **output** as a function of these impulses

This calculation is divided:

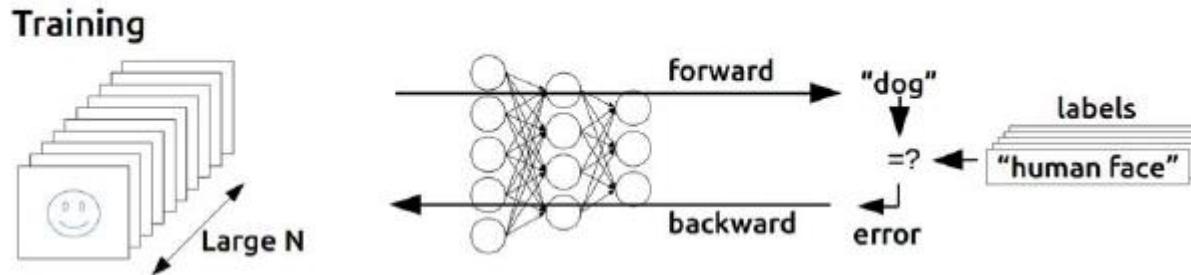
- linear combination of the inputs: $net_i = \sum_i w_{ij}^{(l)} x_i + b$
- application of activation function: $y_i = g(net_i)$ (typically non-linear)



Artificial Neural Networks (ANN): History



Artificial Neural Networks (ANN): History



- Neural Networks need larger amounts of data to optimize
- Experimentally, training multi-layer feedforward networks was not useful
 - the accuracy didn't improve with more layers

Around 1998 SVM and kernel based methods become popular.

- Once again Neural Networks were putted aside.

Artificial Neural Networks (ANN): Perceptron

Psychological Review
Vol. 65, No. 6, 1958

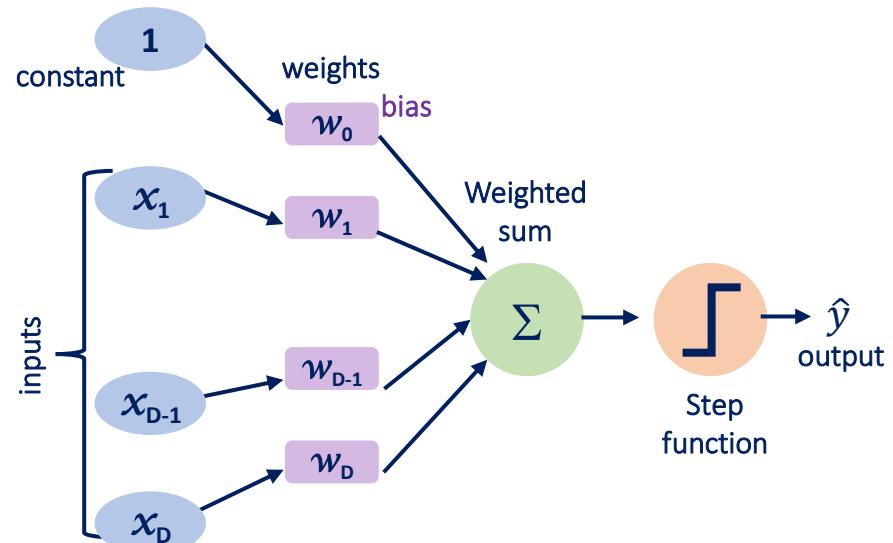
THE PERCEPTRON: A PROBABILISTIC MODEL FOR
INFORMATION STORAGE AND ORGANIZATION
IN THE BRAIN¹

F. ROSENBLATT
Cornell Aeronautical Laboratory

- Proposed by Rosenblatt (1958)

Perceptrons are the simplest ANN:

- Only one input layer
- Only one output layer
- Learn linear decision boundaries
- Binary problems



Artificial Neural Networks (ANN): Perceptron

- Weighted sum of the inputs:

$$a = \sum_d w_d x_d + w_0 = \mathbf{w}^T \mathbf{x} + w_0$$

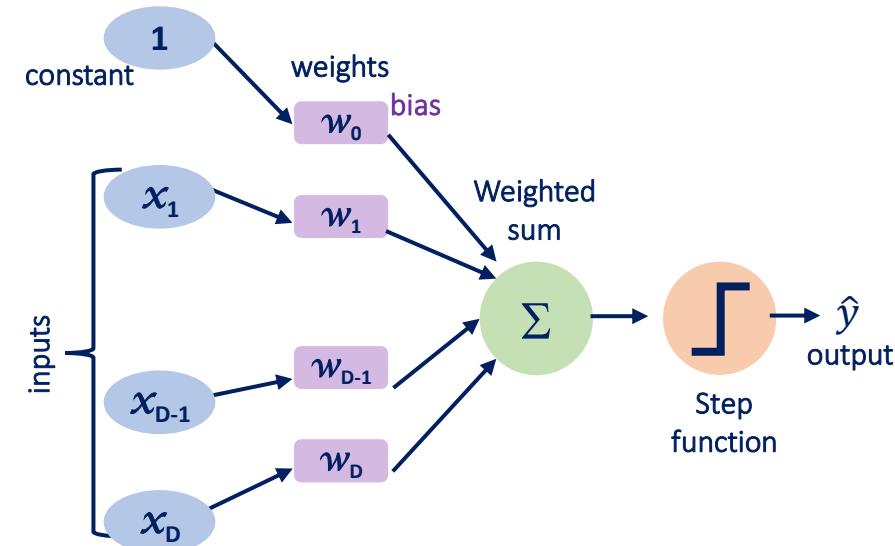
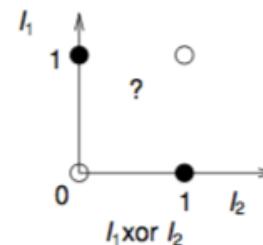
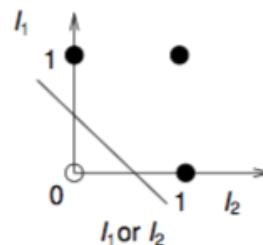
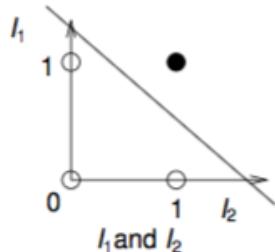
- $f(\cdot)$ activation function: step function

$$f(\mathbf{x}) = \begin{cases} +1, & \mathbf{w}^T \mathbf{x} + w_0 > 0 \\ -1, & \text{otherwise} \end{cases}$$

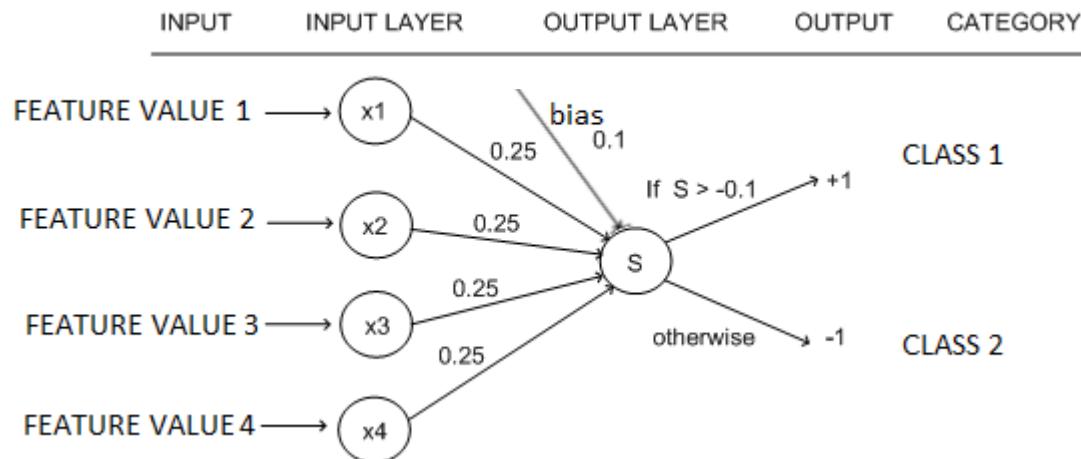
- It learns by updating the weights (only updates when misclassification occurs)

$$\mathbf{W}_i^{(k+1)} = \mathbf{W}_i^{(k)} + \lambda (y_i - \hat{y}_i^{(k)}) \mathbf{X}_i \quad (\lambda \text{ is the learning rate})$$

- Perceptrons are limited to linearly separable problems

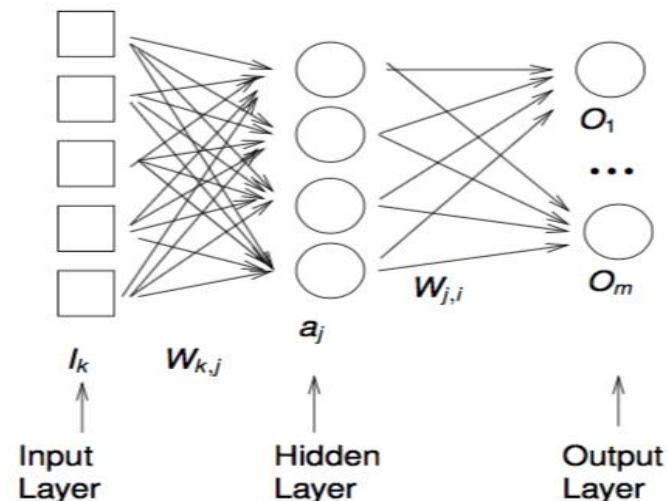


Artificial Neural Networks (ANN): Extending the Perceptron ...



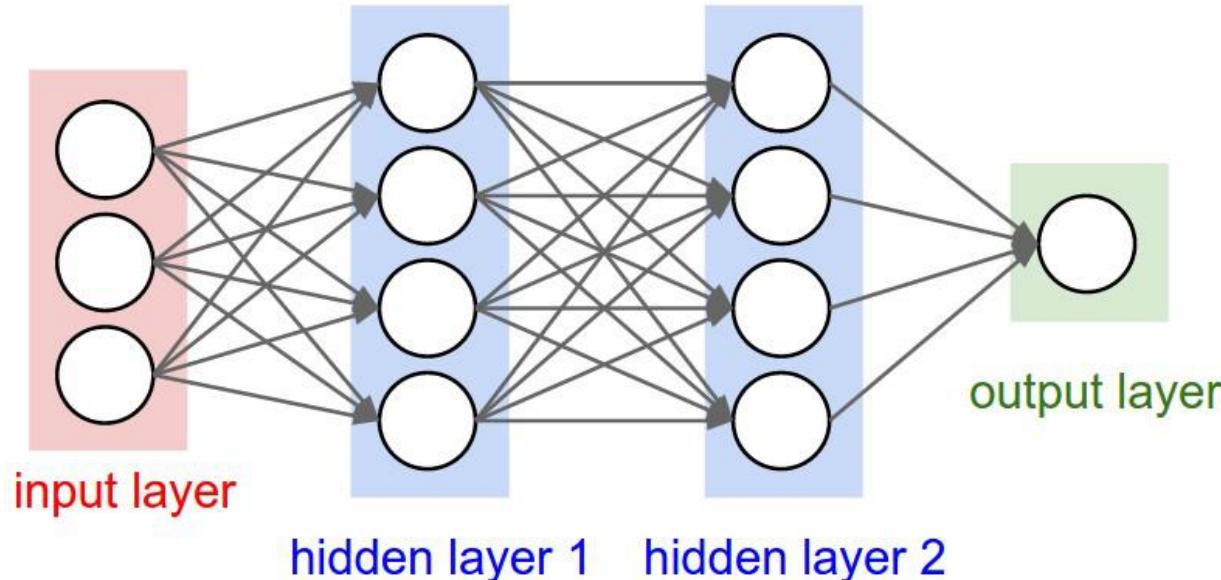
Perception: precursor for Neural Networks!

Extending Perceptron: connecting units together into multilayer Neural Networks!



Artificial Neural Networks (ANN)

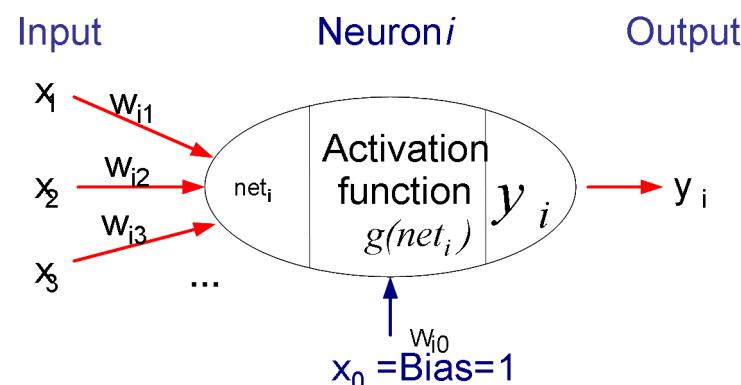
Feedforward Neural Networks (Multilayer perceptrons)



Each unit

$$net_i = \sum_i w_{ij}^{(l)} x_i + b$$

$$y_i = g(net_i)$$



Artificial Neural Networks (ANN)

Input and Output layers are the interface with the "outside"

- Input: reads information. For instance, pixels of one image
- Output: writes information. Digit Classification: 10 units each identifying one digit

Configuration of the network (user's choice)

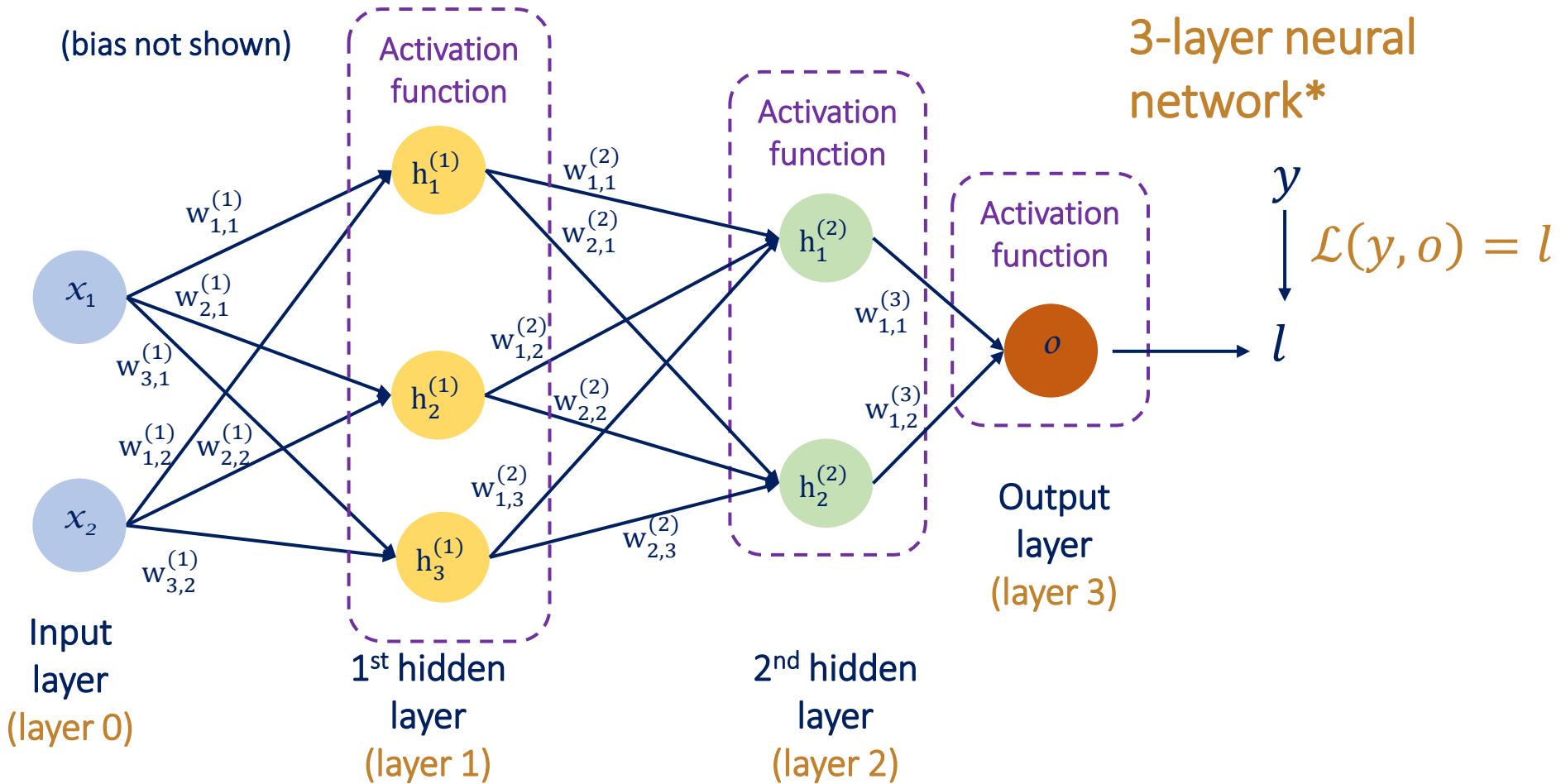
- number of hidden layers: groups of units with the same activation function
- number of units per layer
- connections between layers and units. **Fully connected**: all units of layer L are connected to the following layer $L + 1$
- activation functions of each layer

Feedforward: the information flows from input to output

- Each layer is **feeding** the next layer: output of layer L is the input of layer $L + 1$

The **Weights of all connections** are adapted during **learning phase**

Artificial Neural Networks (ANN): Feedforward NN – Binary classification



Perceptron and logistic regression model can be called a "1-layer neural network"

* number of layers = number of layer of adaptive weights

Artificial Neural Networks (ANN): Feedforward NN

- inputs are combined with the initial weights in a weighted sum

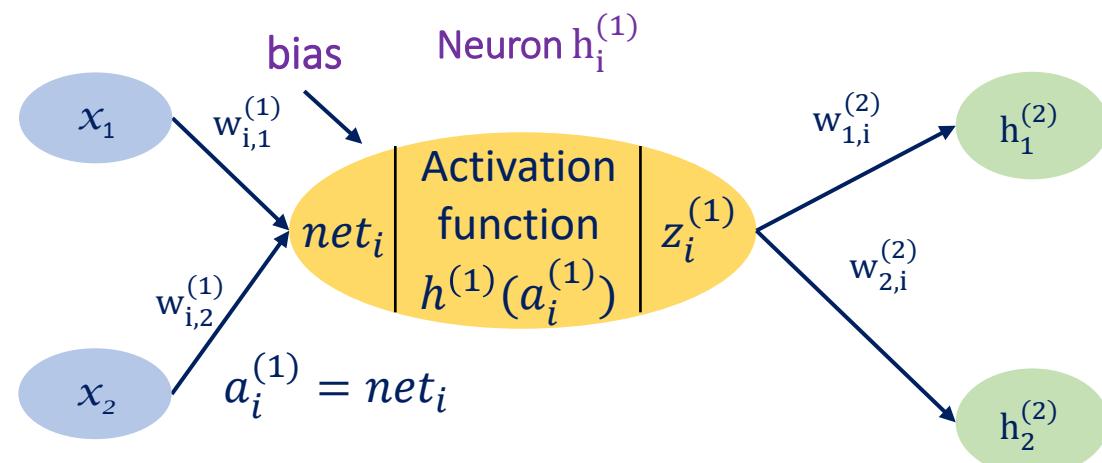
$$net_i = \sum_i w_{ij}^{(l)} x_i + b$$

- application of the activation function: $y_i = g(net_i)$ (typically non-linear)

each linear combination is propagated to the next layer

Each layer is **feeding** the next layer

- The inputs of layer **L+1** are the outputs of layer **L**



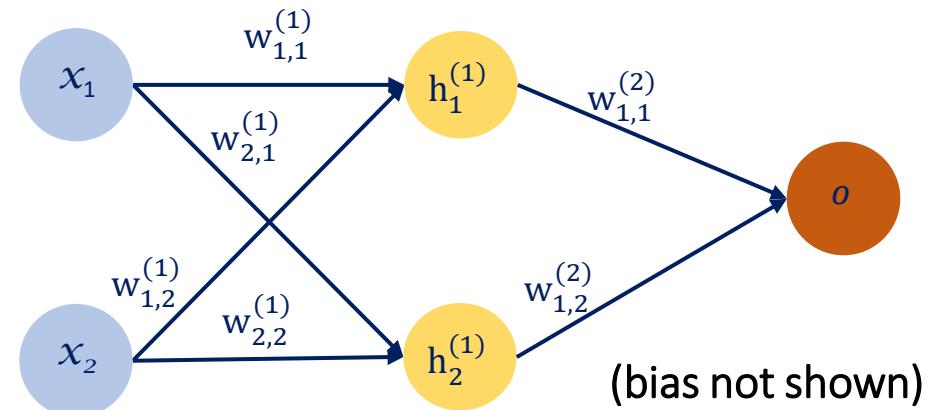
Artificial Neural Networks (ANN): Feedforward NN

2-layer feedforward neural network:

- 1 input layer
- 1 hidden layer with 2 neurons
- 1 output layer (o) with one output variable:

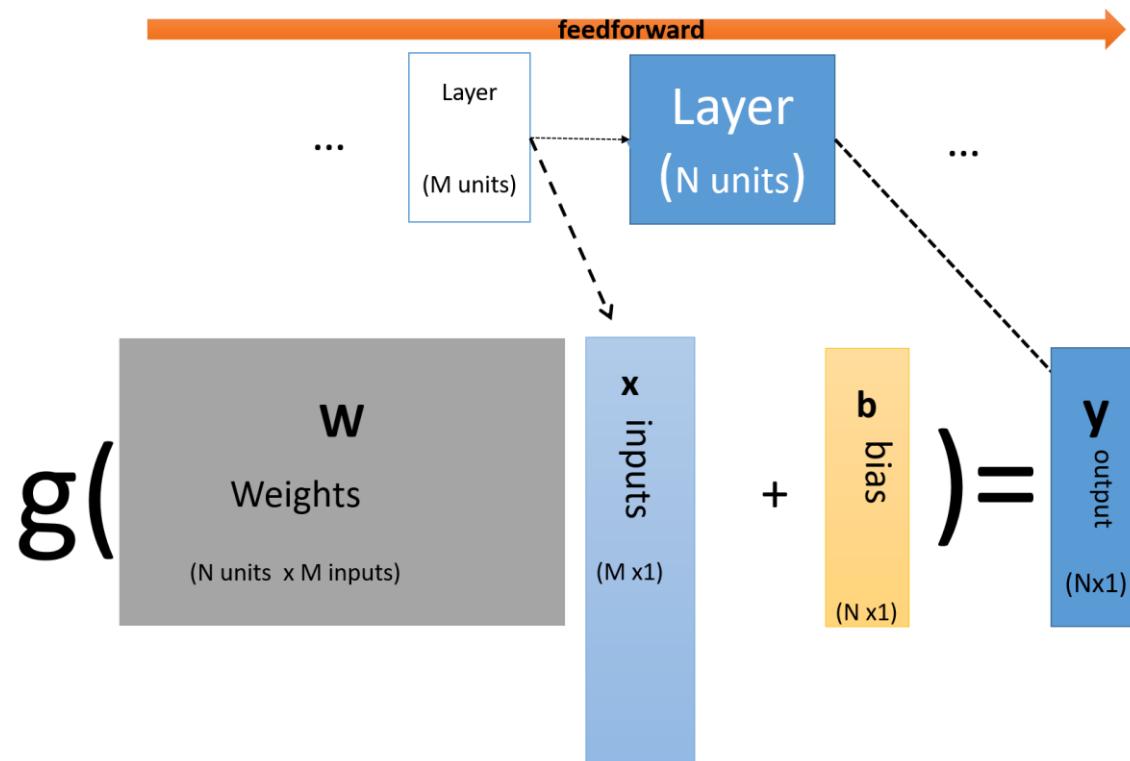
$$\begin{aligned} o = a_3 &= g \left(w_{1,1}^{(2)} a_1 + w_{1,2}^{(2)} a_2 \right) = \\ &= g \left(w_{1,1}^{(2)} g \left(w_{1,1}^{(1)} x_1 + w_{1,2}^{(1)} x_2 \right) + w_{1,2}^{(2)} g \left(w_{2,1}^{(1)} x_1 + w_{2,2}^{(1)} x_2 \right) \right) \end{aligned}$$

- $g()$ is the activation function



Learning Phase: Calculation of the weights of the connections between units (layers)

Artificial Neural Networks (ANN): Feedforward NN



The inputs are propagated from **input to output**

- The units of a layer process the outputs of the previous layer

Artificial Neural Networks (ANN): Activation functions

Activation functions are used to determine the output of each node of the neural network

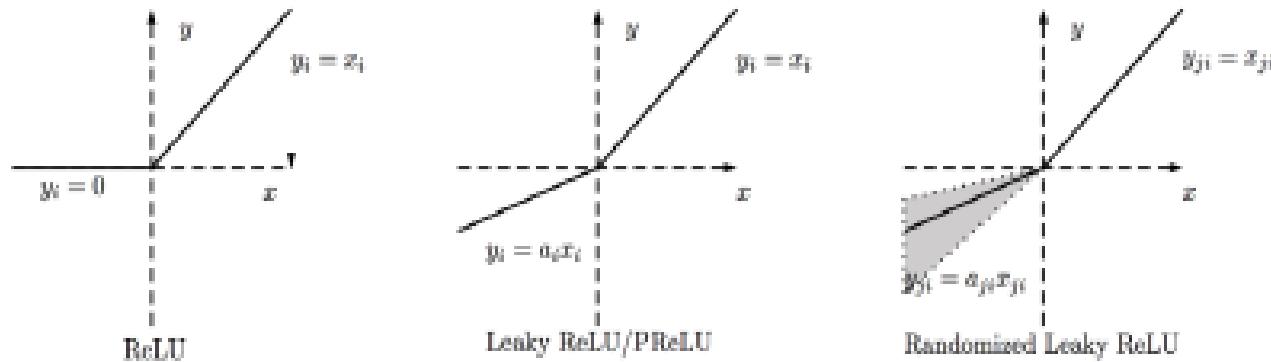
- linear
- non-linear: most commonly used as it allows the model to generalize or adapt with variety of data



$$y = g(a) = a$$

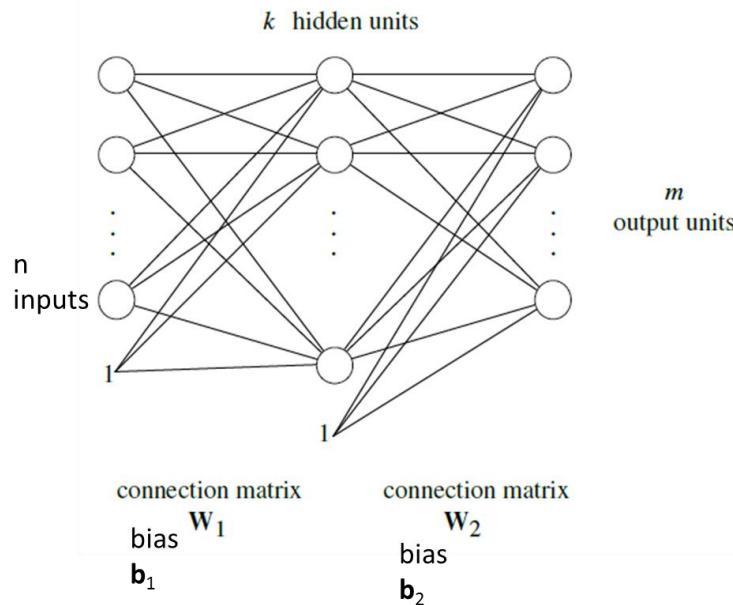
$$y = g(a) = \frac{1}{1+e^{-a}}$$

$$y = g(a) = \tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}}$$



Artificial Neural Networks (ANN): Non-linear activation functions

Non-linear activation functions in **hidden layers** are a must



- hidden layer activation function: **sigmoid**
- output layer activation function: **linear**
- The output of the network

$$\mathbf{o} = \mathbf{W}_2 g(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2$$

where $g()$ is the activation function

With an activation function linear

$$\mathbf{W} = \mathbf{W}_2 \mathbf{W}_1 \quad \mathbf{b} = \mathbf{W}_2 \mathbf{b}_1 + \mathbf{b}_2$$

- Possible to find an equivalent network without hidden layers

Artificial Neural Networks (ANN):

Learning multilayer NN

- Can we apply perceptron learning rule to each node, including hidden nodes?
- Perceptron learning rule computes error term $e = y - \hat{y}$ and updates weights accordingly
 - Problem: how to determine the true value of y for hidden nodes?
- Approximate error in hidden nodes by error in the output nodes
 - Problem:
 - Not clear how adjustment in the hidden nodes affect overall error
 - No guarantee of convergence to optimal solution

Artificial Neural Networks (ANN): Gradient-based learning

- Cost/loss function $E = J(\mathbf{w}) = \sum_{k=1}^n Loss(y_k, \hat{y}_k)$

- Vector gradient: indicate the maximum of the error

$$\nabla E = \left(\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \dots, \frac{\partial E}{\partial w_D} \right)$$

- each weight is updated according to

$$\Delta w_i = -\lambda \frac{\partial E}{\partial w_i}$$

Parameters are updated in the direction of “maximum descent” in the loss function across all points

$$w_{ij}^l \leftarrow w_{ij}^l - \lambda \frac{\partial E}{\partial w_{ij}^l}, \quad \lambda: \text{learning rate}$$
$$b_i^l \leftarrow b_i^l - \lambda \frac{\partial E}{\partial b_i^l},$$

Stochastic gradient descent (SGD): update the weight for every instance

minibatch SGD: update over min-batches of instances

Artificial Neural Networks (ANN): Computing gradients

$$\frac{\partial E}{\partial w_j^l} = \sum_{k=1}^n \frac{\partial \text{Loss}(y_k, \hat{y}_k)}{\partial w_j^l}.$$
$$\hat{y} = a^L$$
$$a_i^l = f(z_i^l) = f\left(\sum_j w_{ij}^l a_j^{l-1} + b_i^l\right)$$

Using chain rule of differentiation (on a single instance):

$$\frac{\partial \text{Loss}}{\partial w_{ij}^l} = \underbrace{\left(\frac{\partial \text{Loss}}{\partial a_i^l} \right)}_{\text{dashed box}} \times \frac{\partial a_i^l}{\partial z_i^l} \times \frac{\partial z_i^l}{\partial w_{ij}^l} \quad \delta_i^l = \frac{\partial \text{Loss}}{\partial a_i^l}$$

How to compute δ_i^l for every layer???

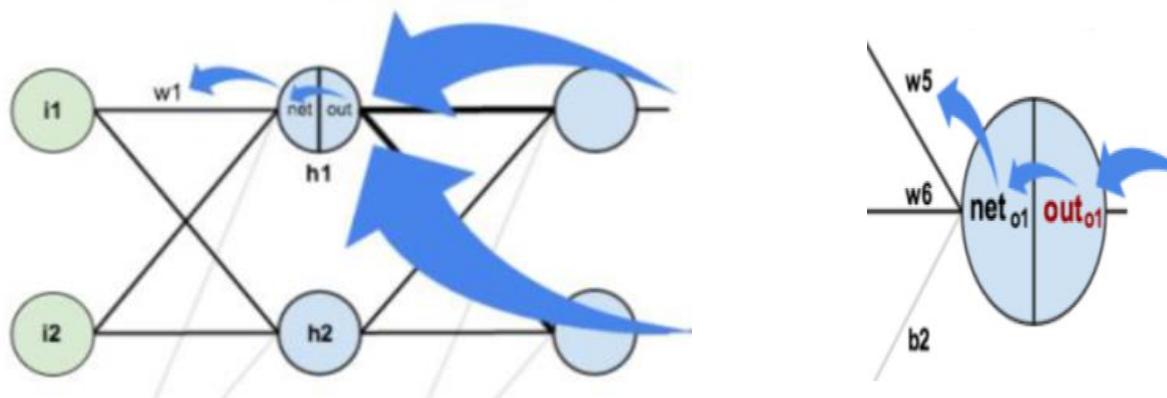
Artificial Neural Networks (ANN): Backpropagation algorithm

Intuition

- each unit is responsible for a certain fraction of the error in the output nodes to which it is connected
- thus, the error is divided according to the weight of the connection between the respective hidden and output units, thus propagating the errors backwards

Backpropagation computes the gradient in weight space of a feedforward neural network, with respect to a loss function

Chain rule



Artificial Neural Networks (ANN): Backpropagation algorithm

- At output layer L :

$$\delta^L = \frac{\partial \text{Loss}}{\partial a^L} = \frac{\partial (y - a^L)^2}{\partial a^L} = 2(a^L - y)$$

- At a hidden layer L (using chain rule):

$$\delta_j^l = \sum_i (\delta_i^{l+1} \times a_i^{l+1} (1 - a_i^{l+1}) \times w_{ij}^{l+1})$$

- Gradients at layer L can be computed using gradients at layer $L + 1$
- Start from layer L and “backpropagate” gradients to all previous layers
- Use gradient descent to update weights at every epoch
- For next epoch, use updated weights to compute loss fn. and its gradient
- Iterate until convergence (loss does not change)

Artificial Neural Networks (ANN): Backpropagation algorithm

The algorithm (for one hidden layer)

- Initialize network weights (often small random values)
- Do
 - For each example in training set
 - predict the output
 - calculate the prediction error by a loss function
 - compute δ_h for all the weights from output layer to hidden layer
 - compute δ_i for all the weights from hidden layer to input layer
 - update network weights
- Until it converges
 - all examples are classified correctly or stopping criterion is satisfied
- Return the network

Artificial Neural Networks (ANN): Backpropagation algorithm

When to stop training?

- If stopping too early: risk of getting a network not yet trained
- If stopping too late: danger of overfitting (adjustment to noise in the data)
- Stopping criteria:
 - maximum number of iterations
 - error based on the training set
 - when the error in the training set is below a certain limit
 - error based on a validation set (independent from the training set)
 - when the error on the validation set has reached a minimum

Artificial Neural Networks (ANN): Some relevant hyperparameters

Network Structure

- number of layers
- number of neurons in each layer
- weights initialization
- activation function

Training Algorithm

- learning rate
- max. number of epochs
- early stopping criterion
- mini-batch size for mini-batch SGD

Artificial Neural Networks (ANN): Design issues

Network Structure

- Number of nodes in input layer:
 - One input node per **binary/continuous** attribute
 - k or $\log_2 k$ nodes for each **categorical** attribute with k values
- Number of nodes in output layer:
 - **One output** for **binary** class problem
 - k or $\log_2 k$ nodes for **k-class** problem
- Number of hidden layers

Artificial Neural Networks (ANN): Design issues

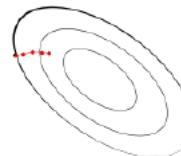
Network Structure

- Nodes per hidden layer:
 - few nodes: underfitting (network is unable to learn problem concept)
 - many nodes: overfitting (training set is memorized, thus making the network useless on new data sets)
 - there are no criteria for defining the number of nodes in the hidden layer
- Initial weights and biases

Artificial Neural Networks (ANN): Design issues

Training Algorithm

- Learning rate (sets the size of the steps to obtain the direction of maximum descendent)
 - a small learning rate has the effect of learning times higher
 - a high learning rate may lead to non-convergence
 - Batch learning typical values are 0.001 to 0.1
 - max. number of epochs, mini-batch size for mini-batch SGD, ...



η too small



η too large

Artificial Neural Networks (ANN): Practical hints

- Preprocessing inputs of the network:
 - Data should be standardized
 - Features with very different distributions of values are not convenient, given the typical activation functions
- Missing values in input features may be represented as zeros, which do not influence the neural net training process
- Output in Multiclass Setting:
 - Use one-hot encoding, there are M output neurons (1 per class)
 - For each case, the class with the highest probability value
- Random initialization of the weights: zero mean and standard deviation equal to number $m^{-1/2}$ where m is the number of weights of the unit

Artificial Neural Networks (ANN)

Advantages

- Linear and Non-Linear in the same algorithm
- Good generalization (classification accuracy high)
- Robust, works when training examples contain errors
- Binary or multiclass problems
- Can handle redundant and irrelevant attributes because weights are automatically learnt for all attributes
- Ability to classify patterns on which they have not been trained

Disadvantages

- No criterium to choose the appropriate architecture
- Long training times (but testing is fast)
- The training can converge to a local minimum
- If the network is large, then a large training set is needed
- Not easy to interpret results (resulting models are essentially black boxes)

Artificial Neural Networks (ANN)

Use ANNs when

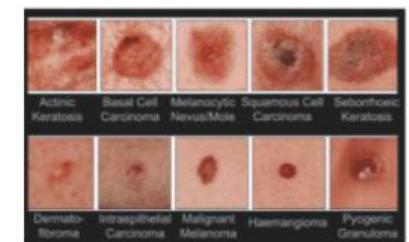
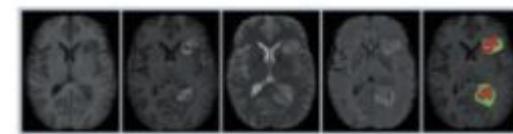
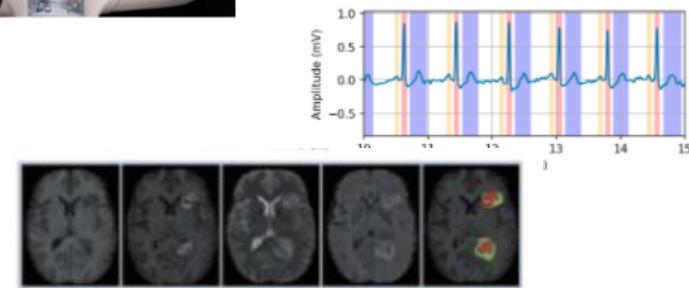
- Input is high-dimensional discrete or real-valued (e.g. raw sensor input)
- Output is a vector of values (classification or regression)
- Possibly noisy data
- Form of target function is unknown
- Human readability of result is unimportant

Contents

- Artificial Neural Networks
- Deep Learning (very short-introduction)

Deep Learning: where?

- Image recognition (e.g. Google, Facebook)
- Automatic text translation (e.g. Google Translator)
- Answers in natural language / digital assistants
- Games (e.g. DeepMind AlphaGo)
- Transcript of handwritten text
- Self-driving cars
- Image colorization, caption generation
- Classification of protein and DNA sequences
- Heart sound: classification and segmentation
- Tumor images detection from MRI, CT, X-rays
- Skin lesion classification from clinical and dermoscopic images
- Parkinson's disease detection from voice recording



Deep Learning

- Deep learning = Deep neural networks
 - Deep = high number of hidden layers
 - Learn a larger number of parameters!
- Training **deep neural networks** (more than **5-10 layers**) could only be possible in recent times with:
 - Faster computing resources (GPU)
 - Larger labeled training sets

Deep Neural Networks

- Algorithmic improvements in Deep Learning
 - Responsive activation functions (e.g., RELU)
 - Regularization (e.g., Dropout)
 - Supervised pre-training
 - Unsupervised pre-training (auto-encoders)
- Specialized ANN Architectures:
 - Convolutional Neural Networks (for image data)
 - Recurrent Neural Networks (for sequence data)
 - Residual Networks (with skip connections)
- Generative Models: Generative Adversarial Networks

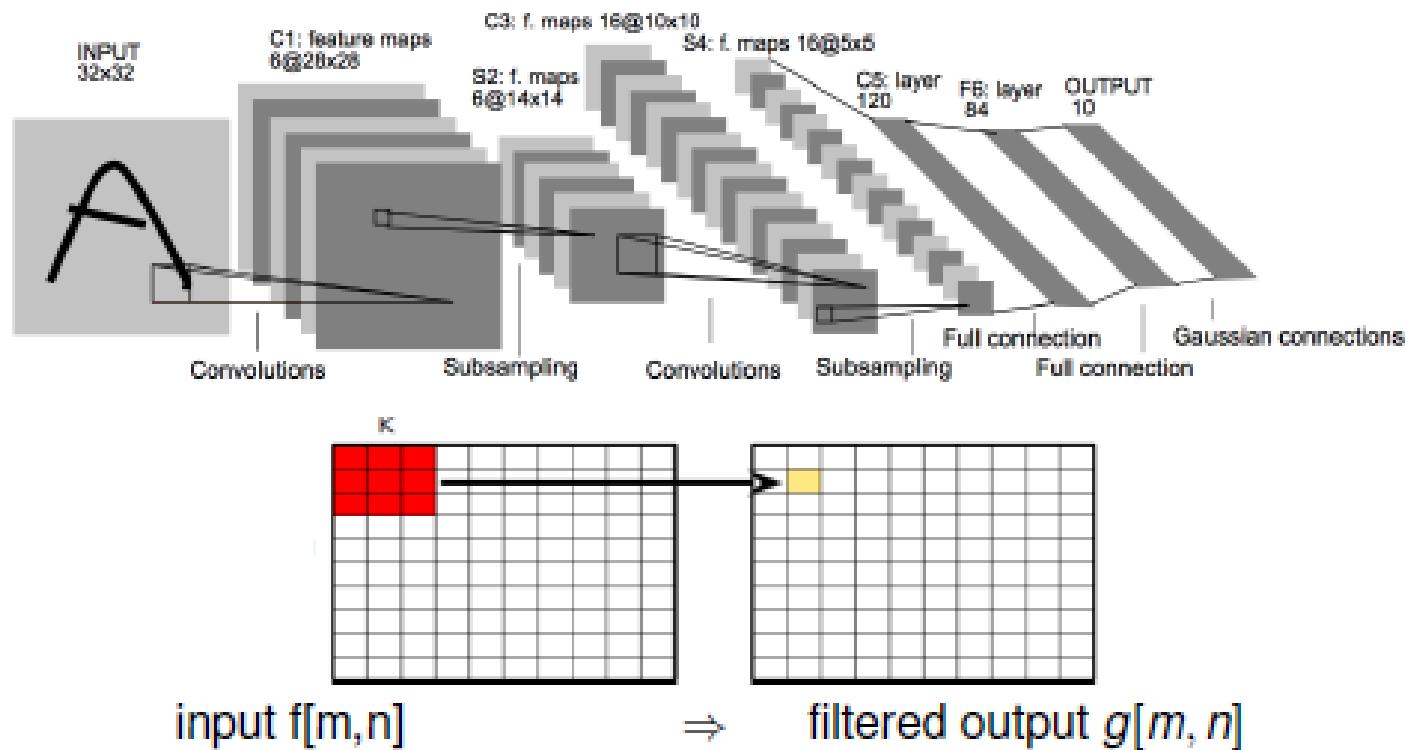
Deep Neural Networks

- Commonly used architectures are **convolutional networks**
- Almost all **CNN** architectures follow the same general design principles of
 - successively applying convolutional layers to the input,
 - periodically downsampling the spatial dimensions while
 - increasing the number of feature maps.
- **Classic network** architectures were comprised simply of stacked convolutional layers
- **Modern** architectures explore innovative ways for constructing convolutional layers for more efficient learning
- Almost all of these architectures are based on a **repeatable unit**

Convolution Neural Networks (CNNs)

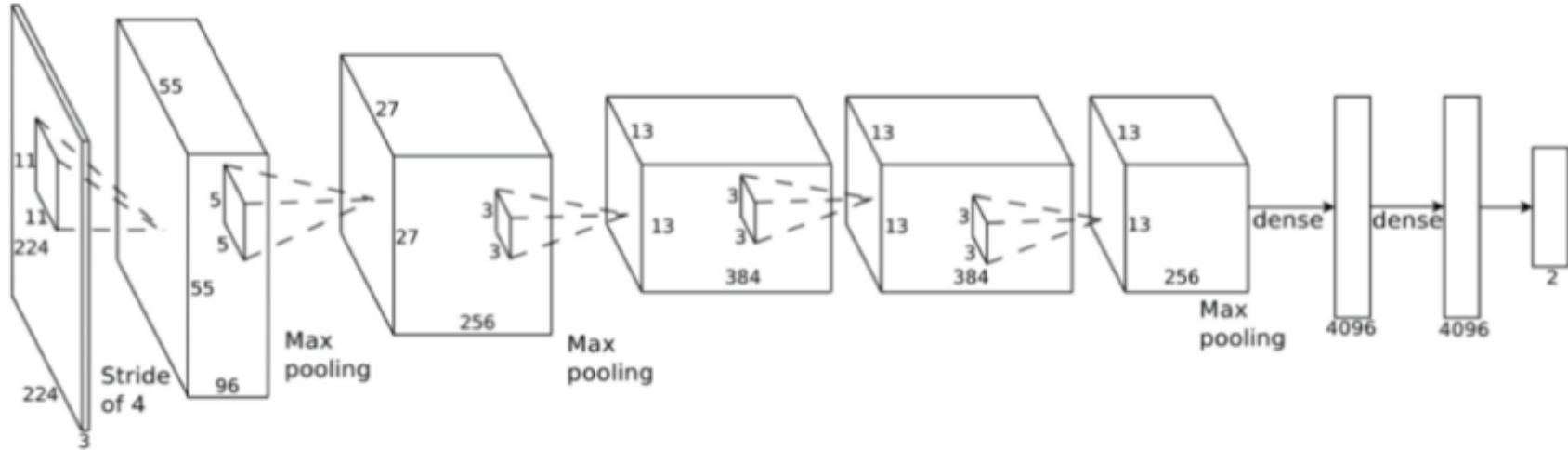
- Feedforward neural networks
- Neurons typically use the ReLU or sigmoid activation functions
- Weight multiplications are replaced by convolutions (filters)
- Change of paradigm: can be directly applied to the raw signal, without computing first ad hoc features
- Features are learnt automatically!!

LeNet-5



- Centering the mask on every pixel (m, n) , $m = 1 \dots M; n = 1 \dots N$
- Convolution yields filtered output image
$$g[m, n] = \sum_{k,l} h[k, l]f[m - k, n - l]$$

AlexNet



Conv Layer I: stride $S = 4$

Conv Layers II, III , IV and V:
stride $S = 1$

Local Normalization: after
Layer I and II

Max Pooling: after Layer I, II e
V.

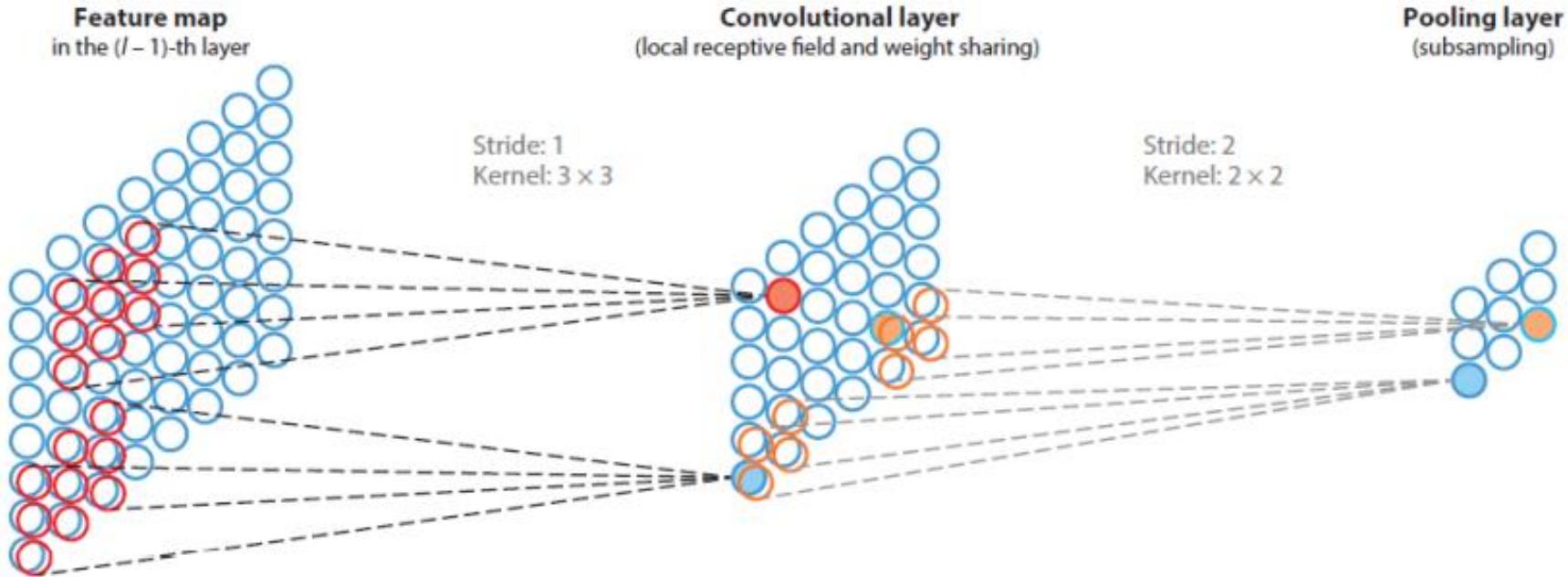
After last conv layer the
outputs $13 \times 13 \times 128$ are
vectorized

AlexNet

The parameters are

- **Filter kernel K:** The kernels have dimension $K \times K \times D$
- **Stride S:** The filter is centered in one out of S pixels
- **Padding pad:** The input **feature maps** are padded at the edges with pad pixels (zero-initialized)
- **Pooling** with overlap

Convolution and Pooling



- CNNs use **Local Receptive Fields** and scan input space
- **Weight Sharing:** All units of a feature map have the same weights
- Dimension reduction: **pooling** (max or average) or **striding**

Deep Learning: summarization

Great results! But...

- Like any other technique, DL does not solve all problems and will not always be the best option for any learning task
- Difficult to select best architecture for a problem
- Require new training for each task/configuration
- (Most commonly) require a large training dataset to generalize well
 - Data augmentation, weight regularization, dropout, transfer learning, etc
- Still not fully understood why it works so well
- Unstable against adversarial examples

Bibliography

Introduction to Data Mining, Pang-Ning Tan, Michael Steinbach, Anuj Karpatne, Vipin Kumar, *Pearson*, 2019 (chap 6.7)

Data Mining, the Textbook, Charu C. Aggarwal, *Springer*, 2015 (chap 10.7)

Pattern Recognition and Machine Learning, C. Bishop, *Springer*, 2007 (ch 5)

Introduction to Machine Learning, Ethem Alpaydin, MIT Press

<https://www.skynettoday.com/overviews/neural-net-history>

https://sebastianraschka.com/pdf/lecture-notes/stat453ss21/L09_mlp_slides.pdf