

Machine Learning

The Data

Claudio Sartori

DISI

Department of Computer Science and Engineering – University of Bologna, Italy
claudio.sartori@unibo.it

1	Issues on data	2
2	Data Types	5
3	Data Quality	19

Issues on data

- Type
 - Quantitative, qualitative, structured, ...
- Quality
 - Data are never perfect
 - Missing, inconsistent, duplicated, wrong
 - Outliers
 - Small amount of data which are different from the rest, due to anomalies or errors
 - Some mining techniques are more robust w.r.t. errors than others
 - Better data quality ⇒ better results
 - trivial: **garbage-in–garbage-out**
- Pre-processing activities transform data to ease the mining activities



Some examples of datasets

UCI Machine Learning Repository

- Iris
- Adult
- Breast Cancer
- Wine Quality
- Car evaluation

Explore them by yourself

- 1 Issues on data 2
- 2 Data Types 5
- 3 Data Quality 19

Data Types and numerical properties

[Tan et al.(2006) Tan, Steinbach, and Kumar, Stevens(1946)]

Data Type		Description	Examples	Descriptive statistics allowed
Categorical	Nominal	The values are a set of labels, the available information allows to distinguish a label from another Operators: = and ≠	zip code, eye color, sex, ...	mode, entropy, contingency, correlation, χ^2 test
	Ordinal	The values provide enough information for a total ordering Operators: <>≤≥	hardness of minerals, non-numerical quality evaluations (bad, fair, good, excellent)	median, percentiles, rank correlations
Numerical	Interval	The difference is meaningful Operators: +-	Calendar dates, temperatures in centigrades and Fahrenheit	average, standard deviation, Pearson's correlation, F and t tests
	Ratio	Have a univocal definition of 0 Allow all the mathematical operations on numbers	Kelvin temperatures, masses, length, counts	geometric mean, harmonic mean, percentage variation

The “description” and “descriptive statistics” columns are *incremental*, i.e. the properties described in a row are added to the properties described in the rows above

Discuss the types of data in the columns

Patient	Treatment	Treatment Day	Temperature	Pain
XXXX	a	2	37	3
XXXX	a	3	37	2
XXXX	a	4	36.5	1
YYYY	b	1	38	3
YYYY	b	2	37.5	2

Example 1

Patient	Weight	BirtyYear	Age	Sex
XXXX	78	1970	50	M
YYYY	56	1980	40	F

Example 2

Allowed transformations

Data Type		Transformation	Comment
Categorical	Nominal	Any one-to-one correspondence	the SSN can be arbitrarily reassigned (masking)
	Ordinal	Any order preserving transformation $\text{new} \leftarrow f(\text{old})$ where f is a monotonic function	(bad, fair, good, excellent) can be substituted by (1,2,3,4)
Numerical	Interval	Linear functions $\text{new} \leftarrow a + b * \text{old}$	centigrades and Fahrenheit temperatures can be converted either way
	Ratio	Allow any mathematical function, <i>standardization</i> , variation in percentage	Kelvin temperatures, masses, length, counts

The transformations above do not change the meaning of the attribute

- e.g. Kelvin temperature increments can be expressed with percentages, since they have a physical meaning related to energy
- Centigrade temperatures can be linearly transformed into Fahrenheit, but percentage increments do not have any physical meaning, since the zero level is arbitrary

Why should we transform them?

Number of values

- Discrete domains
 - allow a finite number of values (or infinitely countable)
 - codes, counts, ...
 - special case: **binary attributes**
 - special case: **identifier**
 - useful for data manipulation, not for analysis
- Continuous domains
 - floating point variables
- nominals and ordinals are discrete, possibly binary
- intervals and ratio are continuous (possibly with approximation)
- counts are discrete and ratio

Asymmetric attributes

- Only presence is considered important (a non null value)
 - e.g. a student record with one attribute per offered exam
 - only passed exams are interesting, in general the exams not passed will be in much greater number, and do not carry much information
- In particular, binary asymmetric attributes are relevant in the discovery of association rules

General characteristics of data sets

- Dimensionality
 - the difference between having a small or a large (hundreds, thousands, ...) of attribute is also **qualitative**
 - see the **curse of dimensionality**, later
- Sparsity
 - when there are many zeros or nulls
- Beware the nulls in disguise
 - a widespread bad habit is to store zero or some special value when a piece information is not available
- Resolution
 - has a great influence on the results
 - the analysis of too detailed data can be affected by noise
 - the analysis of too general data can hide interesting patterns

Record data

- Tables
 - e.g. relational
- Transaction
 - a row is composed by: TID + set of Items
- Data matrix
 - numeric values of the same type
 - a row is a point in a vector space
- Sparse data matrix
 - asymmetric values of the same type

Relational table

The set of attributes is the same for all the records

<i>Tid</i>	<i>Refund</i>	<i>Marital Status</i>	<i>Taxable Income</i>	<i>Cheat</i>
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

Data matrix

- Numeric attributes
- Each row is a point in a vector space
- N rows and D dimensions (attributes, columns, properties)

<i>Projection of x load</i>	<i>Projection of y load</i>	<i>Distance</i>	<i>Load</i>	<i>Thickness</i>
10.23	5.27	15.22	2.7	1.2
12.65	6.25	16.22	2.2	1.1

Document representation

- Each row represents a document
- Each column represents a term
- Each cell contains the absolute frequency of the term in the document
 - the sequence of terms is lost

	team	coach	play	ball	score	game	won	lost	timeout	season
doc1	3	0	5	0	2	6	0	2	0	2
doc2	0	7	0	2	1	0	0	3	0	0
doc3	0	1	0	0	1	2	2	0	3	0

Transactional data

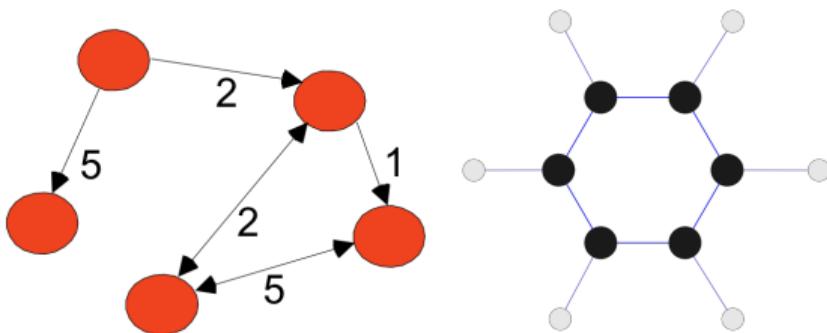
- Each record **contains** a set of objects
 - strictly speaking it isn't a relational table
- The reference example is the *market basket*
 - a commercial transaction

<i>TID</i>	<i>Items</i>
1	bread, coke, milk
2	beer, bread
3	beer, coke, diaper, milk
4	beer, bread, diaper, milk
5	coke, diaper, milk

Graph data

- Web pages
- Set of nodes and (oriented) arcs
- Molecular structures

```
<a href="papers/papers.html#bbbb">  
Data Mining </a>  
<li>  
<a href="papers/papers.html#aaaa">  
Graph Partitioning </a>  
</li>  
<a href="papers/papers.html#aaaa">  
Parallel Solution of Sparse Linear System of Equations </a>  
<li>  
<a href="papers/papers.html#ffff">  
N-Body Computation and Dense Linear System Solvers
```



Ordered data

- Spatial
- Temporal
- Sequence
 - of events, objects, ...
- Genetic bases

```
<a href="papers/papers.html#bbbb">  
Data Mining </a>  
<li>  
<a href="papers/papers.html#aaaa">  
Graph Partitioning </a>  
<li>  
<a href="papers/papers.html#aaaa">  
Parallel Solution of Sparse Linear System of Equations </a>  
<li>  
<a href="papers/papers.html#ffff">  
N-Body Computation and Dense Linear System Solvers
```

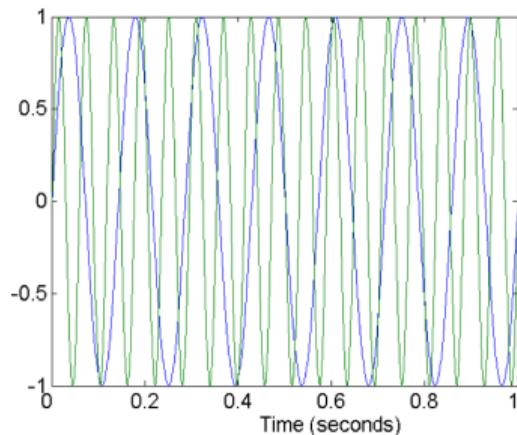
1	Issues on data	2
2	Data Types	5
3	Data Quality	19
	● Outliers	22

Data Quality

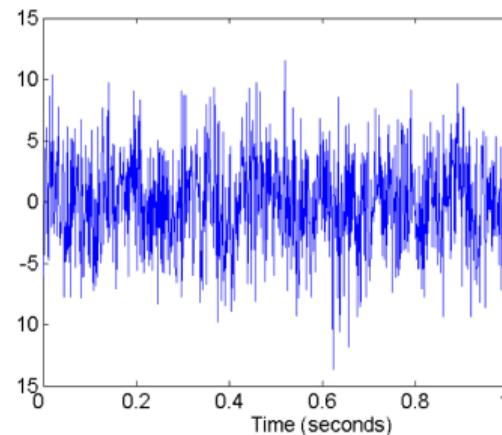
- Which are the problems?
- How can we detect the problems?
- What can we do about these problems?
- Examples
 - noise and outliers
 - missing values
 - duplicates
 - inconsistencies

Noise

- Modification of original values
- Uninteresting mixed to the interesting data
 - noise in transmission
 - web crawler accesses mixed to the human accesses



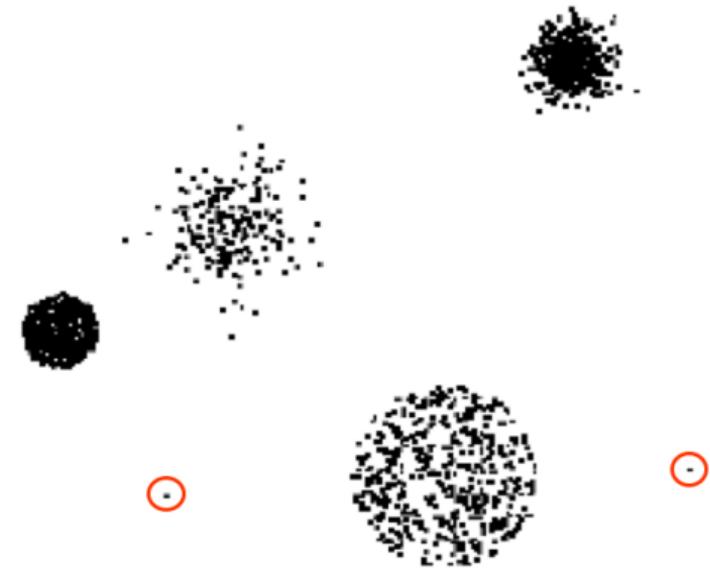
Two sine waves



Two sine waves + noise

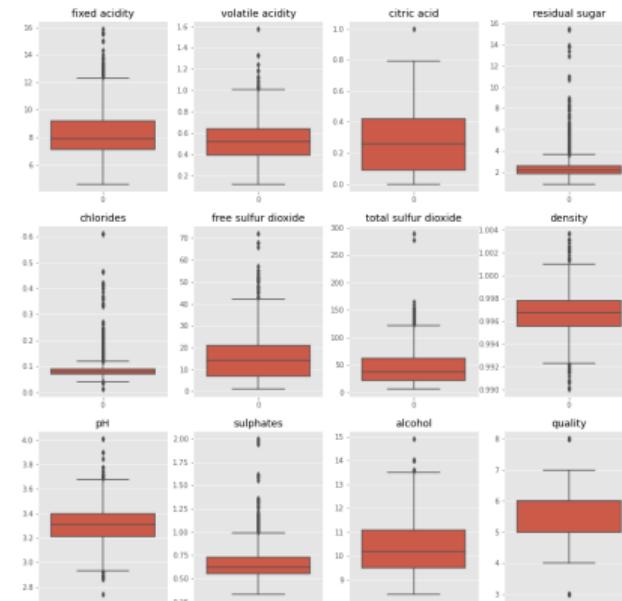
Outliers

- Data whose characteristics are considerably different from most of the data in the dataset
- Can be generated by
 - noise
 - rare events/processes



How to detect outliers: descriptive statistics

- IQR - InterQuartile Range
 - $Q1$: first quartile, $Q3$: third quartile, $IQR = Q3 - Q1$
- Lower boundary
 $= Q1 - IQR * 1.5$
- Upper boundary
 $= Q3 + IQR * 1.5$
- consider **outlier** the values out of the boundaries



Missing values

- Reasons
 - data were not collected
 - e.g. persons are reluctant to communicate income or weight
 - the information is not applicable
 - e.g. children do not have a working annual income
- Management of missing values
 - do not consider objects with missing values
 - sometimes not a good idea
 - estimate/default
 - ignore
 - cannot be done for all the learning schemes
 - insert all the possible values, weighted with probabilities

Duplicated data

- Data objects that are duplicates, or almost duplicated
 - major issue when merging data from different sources
- Data cleaning
 - the (difficult) process of dealing with duplicated/inconsistent data

Bibliography I

- ▶ S. S. Stevens.
On the theory of scales of measurement.
Science, 103(2684):677–680, 1946.
ISSN 0036-8075.
doi: 10.1126/science.103.2684.677.
URL <http://science.scienmag.org/content/103/2684/677>.
- ▶ Pang-Nin Tan, Michael Steinbach, and Vipin Kumar.
Data Mining.
Addison Wesley, 2006.
ISBN 0-321-32136-7.

Machine Learning

Classification - I

Claudio Sartori

DISI

Department of Computer Science and Engineering – University of Bologna, Italy
claudio.sartori@unibo.it

- 1 Introduction to Classification
- 2 Classification model
- 3 C4.5 – Classification with Decision Trees

2

6

17

Unsupervised Classification

- the unsupervised mining techniques which can be in some way related to classification are usually known in literature with names different from classification
- for this reason in this course with the term *classification* we will always mean *supervised classification*

Supervised Classification 1/2

In the following, simply *classification*

Consider the "soybean" example shown in the introduction (link to the [dataset](#))

- The data set \mathcal{X} contains N *individuals* described by D attribute values each
- We have also a \mathcal{Y} vector which, for each individual x contains the *class* value $y(x)$
- The class allows a finite set of different values (e.g. the diseases), say C
- The class values are provided by experts: the supervisors

Supervised Classification 2/2

- We want to learn how to guess the value of the $y(x)$ for individuals which have not been examined by the experts
- We want to learn a *classification model*

- 1 Introduction to Classification
- 2 Classification model
- 3 C4.5 – Classification with Decision Trees

2

6

17

Classification model

- An algorithm which, given an individual for which the class is not known, computes the class
- The algorithm is *parametrized* in order to optimize the results for the specific problem at hand
- Developing a classification model requires
 - choose the *learning algorithm*
 - let the algorithm learn its parametrization
 - assess the quality of the classification model
- The classification model is used by a run-time *classification algorithm* with the developed parametrization

Classification model or, shortly, *classifier*

A bit of formality

- a decision function which, given a **data element** x whose class label $y(x)$ is unknown, makes a *prediction* as

$$\mathcal{M}(x, \theta) = y(x)_{pred}$$

where θ is a set of values of the **parameters** of the decision function

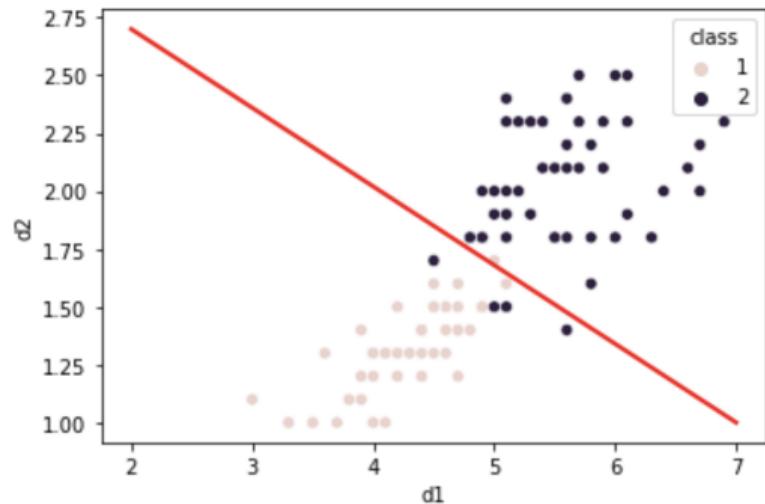
- the prediction can be true or false
- the **learning process** for a given classifier $\mathcal{M}(\cdot, \cdot)$, given the dataset \mathcal{X} and the set of **supervised class labels** \mathcal{Y} determines θ in order to reduce the prediction error as much as possible

Example of decision function

- supervised dataset with two dimensions, two classes
- use as decision function a straight line

$$\theta_1 * d_1 + \theta_2 * d_2 + \theta_3 \geq 0 \Rightarrow c_1$$

$$\theta_1 * d_1 + \theta_2 * d_2 + \theta_3 < 0 \Rightarrow c_2$$



All models are wrong, but some are useful

George Box

- The model (decision function) of the previous page makes some errors
 - even the best choice of parameters cannot avoid errors
- Different models can have different power to shatter the dataset into subsets with homogeneous classes
 - e.g. what about a quadratic function?
$$\theta_1 * d_1^2 + \theta_2 * d_2^2 + \theta_3 * d_1 * d_2 + \theta_4 * d_1 + \theta_5 * d_2 + \theta_6$$

Vapnik-Chervonenkis Dimension

The shattering power of a classification model¹

- Given a dataset with N elements there are 2^N possible different learning problems
- If a model $\mathcal{M}(\cdot, \cdot)$ is able to shatter **all the possible learning problems** with N elements, we say that it has *Vapnik-Chervonenkis Dimension* equal to N
- The straight line has VC dimension 3
 - don't worry, frequently, in real cases, data are arranged in such a way that also a straight line is not so bad

¹ For simplicity, we mention here only the binary case

A workflow for classification - I

1. Learning the model for the given set of classes
 - 1.1 a *training set* is available, containing a number of individuals
 - 1.2 for each individual the value of the class label is available (also named *ground truth*)
 - 1.3 the training set should be *representative* as much as possible
 - 1.3.1 the training set should be obtained by a random process
 - 1.4 the model is fit learning from data the best parameter setting

A workflow for classification - II

1. Estimate the *accuracy* of the model

- 1.1 a *test set* is available, for which the class labels are known
- 1.2 the model is run by a *classification algorithm* to assign the labels to the individuals
 - 1.2.1 the classification algorithm implements the model with the parameters
- 1.3 the labels assigned by the model are compared with the true ones, to estimate the accuracy

2. The model is used to label new individuals

- 2.1 possibly, after the labeling, the true labels may become available and the true accuracy can be compared with the estimated one

A workflow for Learning and Estimation

attribute 1	attribute 2	attribute 3	class
yes	large	24	A
no	medium	31	B
no	large	30	A
yes	large	22	A
yes	small	28	C
no	medium	25	C
no	small	20	A
yes	small	21	B
no	large	30	C

Training Set

Induction

Learning
Algorithm

Learn Model

Model

attribute 1	attribute 2	attribute 3	true class	assigned class
no	small	22	A	A
no	large	30	C	C
yes	large	30	A	B
no	large	26	B	C
yes	small	23	C	C
yes	medium	25	B	B
no	medium	30	A	A

Test Set

Deduction

errors

Apply Model

Classification
Algorithm

Question

OPTIONAL

- *is there a hidden assumption in the description of the soybean example of page 4?*
- *is there a workaround to this hidden assumption?*

Two flavors for classification

Crisp

- the classifier assigns to each individual *one label*

Probabilistic

- the classifier assigns a *probability for each of the possible labels*

1	Introduction to Classification	2
2	Classification model	6
3	C4.5 – Classification with Decision Trees	17
●	Classification algorithm	19
●	Model generation	21
●	Entropy and Information Gain	30
●	Learning a Decision Tree	43
●	Looking at the Information Gains of the predicting attributes	45
●	Learning the Decision Tree	49
●	Errors and Overfitting	57
●	Pruning a Decision Tree	65
●	Impurity functions	68
●	Final remarks on DTs	76
●	Conclusion	81

Decision Trees

C4.5 and beyond [Buntine(1992)]

- Among the most used tools
- History
 - 1966 – ID3 [[Hunt et al.\(1966\)](#)[Hunt, Marin, and Stone](#)]
 - 1979 – CLS [[Quinlan\(1979\)](#)]
 - 1993 – C4.5 [[Quinlan\(1979\)](#)]
- Generate classifiers structured as *decision trees*

Using a Decision Tree 1/2²

- A run-time classifier structured as a decision tree is a tree-shaped set of tests
- the decision tree has *inner nodes* and *leaf nodes*

Using a Decision Tree 2/2

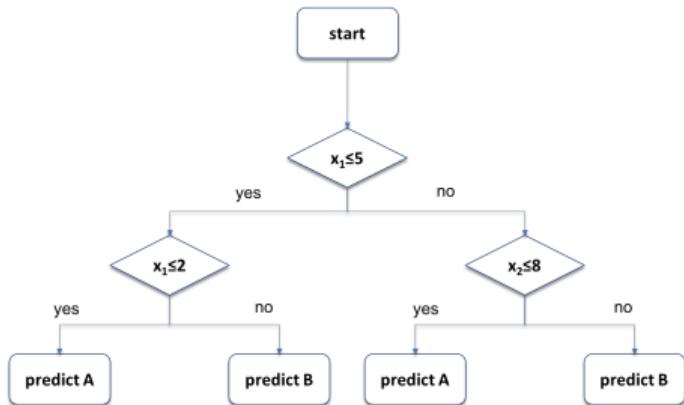
Inner nodes:

if test on attribute d of element x **then**
 execute node'

else
 execute node"

Leaf nodes:

predict class of element x as c''



Learning a decision tree – Model generation

Given a set \mathcal{X} of elements for which the class is known, grow a decision tree as follows

- if all the elements belong to class c or \mathcal{X} is small generate a leaf node with label c
- otherwise
 - choose a test based on a single attribute with two or more outcomes
 - make this test the root of a tree with one branch for each of the outcomes of the test
 - partition \mathcal{X} into subsets corresponding to the outcomes and apply recursively the procedures to the subsets

Learning a decision tree

Problems to solve:

1. which attribute should we test?
2. which kind of test?
 - 2.1 binary, multi-way, . . . , depends also on the domain of the attribute
3. what does it mean \mathcal{X} is small, in order to choose if a leaf node is to be generated also if the class in \mathcal{X} is not unique?

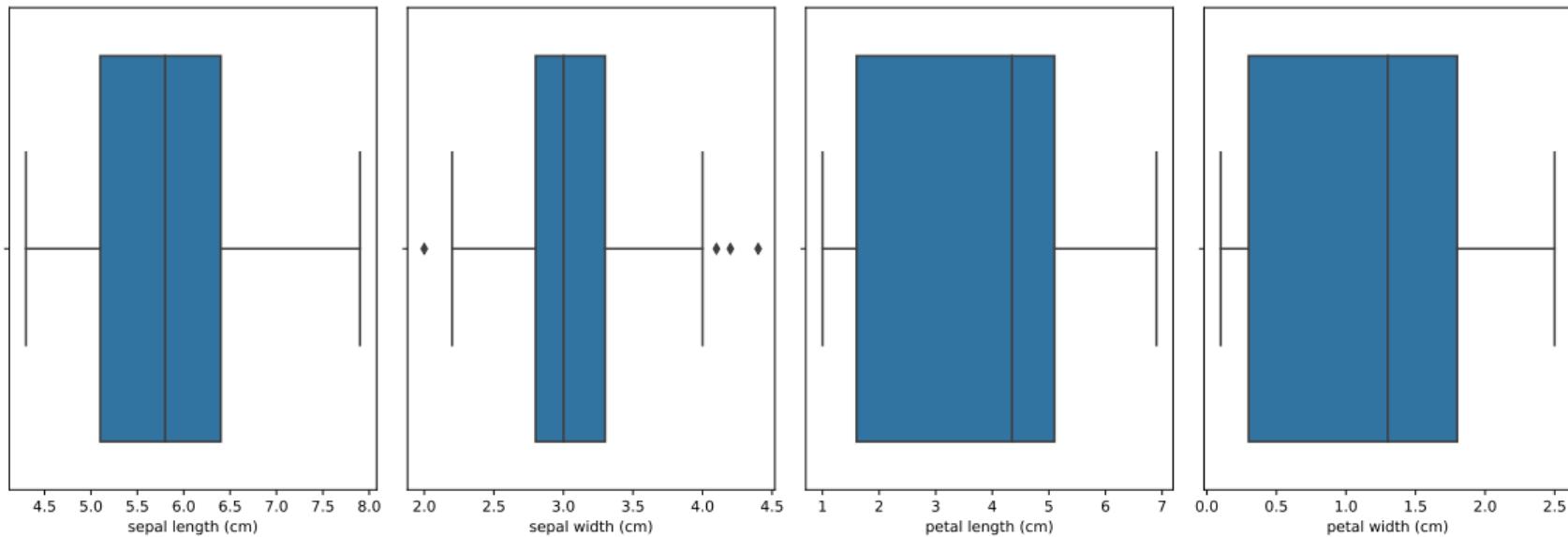
A supervised dataset: Iris

sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	class
6.2	2.2	4.5	1.5	1
5.2	3.5	1.5	0.2	0
5.6	3.0	4.5	1.5	1
6.0	2.9	4.5	1.5	1
7.7	3.0	6.1	2.3	2
5.1	3.8	1.5	0.3	0
5.9	3.2	4.8	1.8	1
5.7	4.4	1.5	0.4	0
6.7	3.1	5.6	2.4	2
6.5	3.2	5.1	2.0	2
...

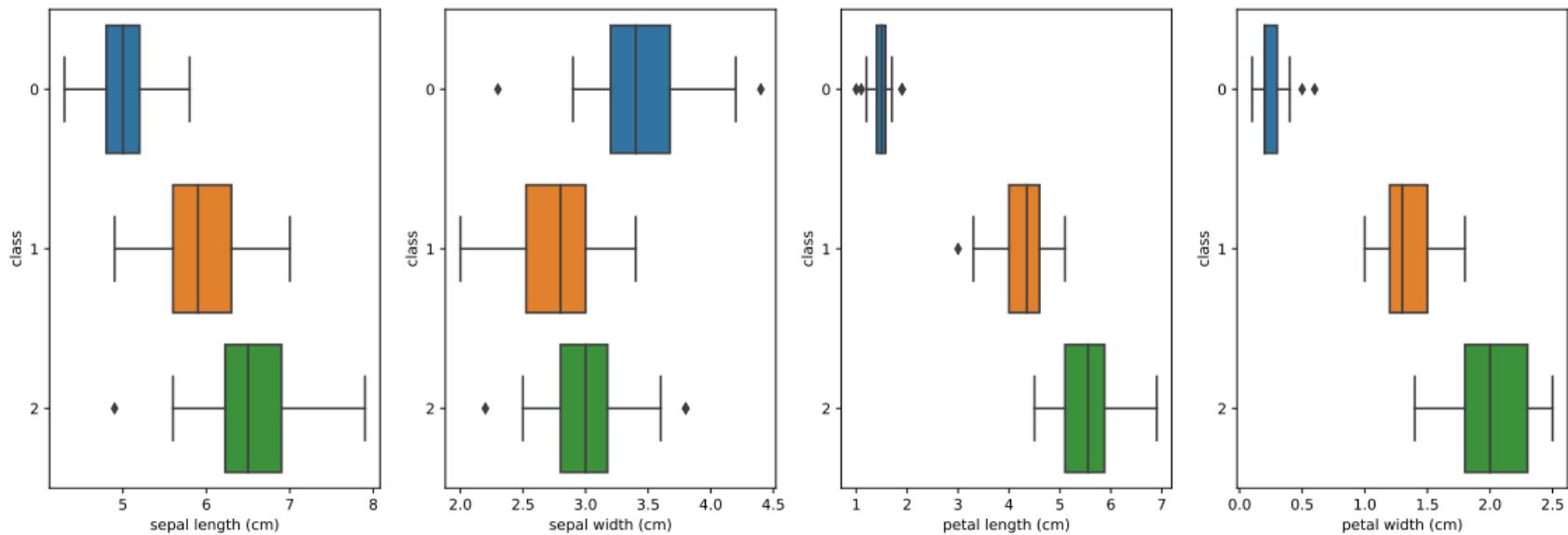
Dataset description

- 150 examples of iris flowers
- 4 attributes describing sizes of petals and sepals, **class** is the target
 - class has three values
- we could be interested in predicting the class for a new individual, given the measures

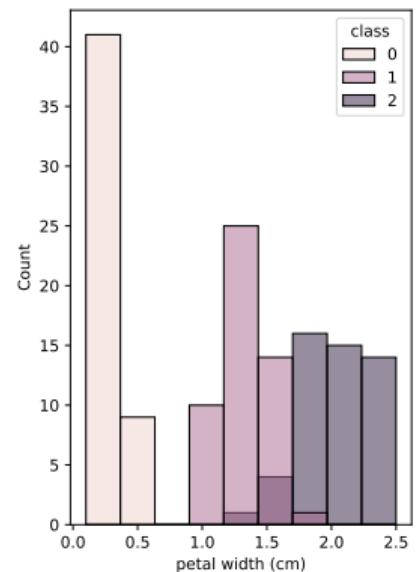
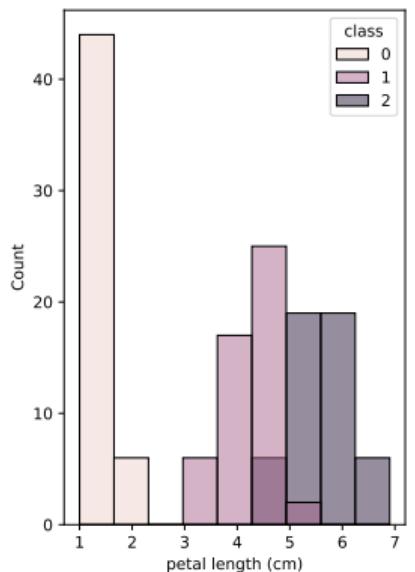
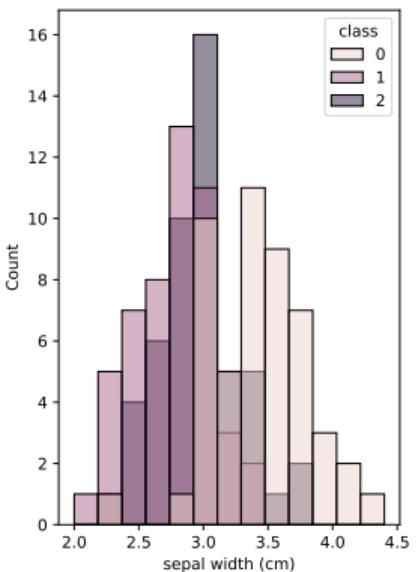
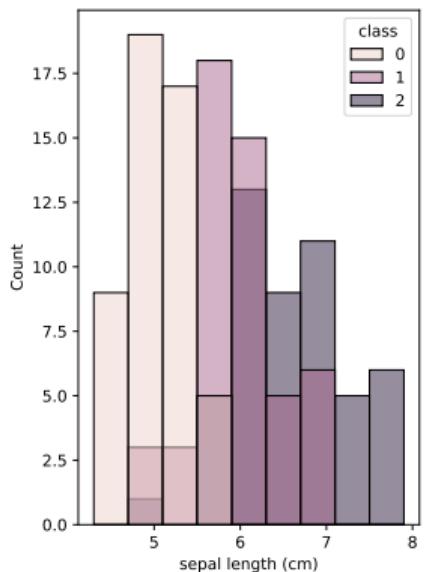
Exploration of the dataset - Boxplot - General



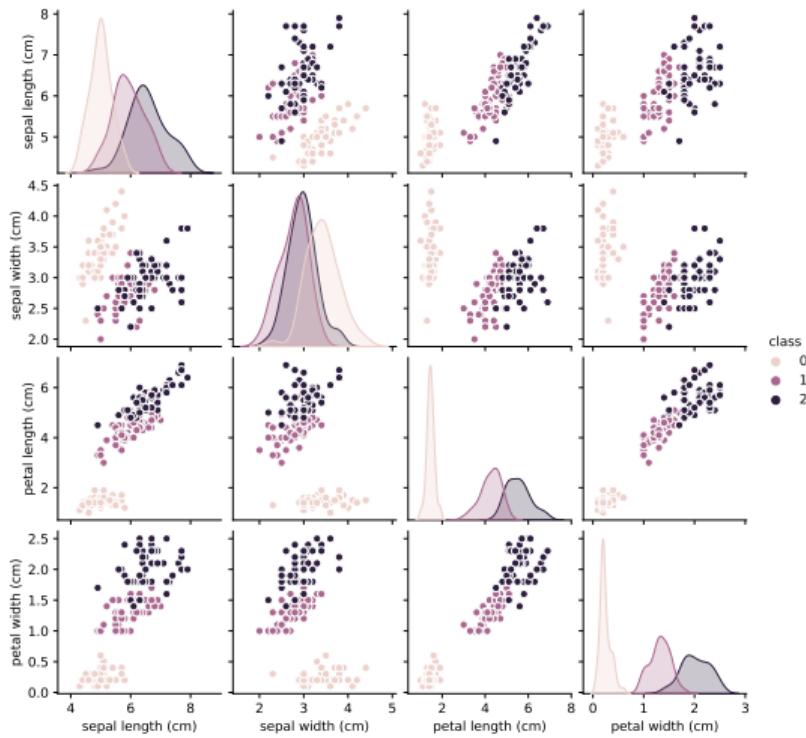
Exploration of the dataset - Boxplot - Inside classes



Exploration of the dataset - Histograms



Exploration of the dataset - Pairplots



Supervised learning goals

- design an algorithm to find interesting patterns, in order to forecast the values of an attribute given the values of other attributes
 - in our case, find patterns to guess the **class** given the other values
- distinguish real patterns from illusions
- choose useful patterns
- in real life, we could have millions of rows and thousands of columns
 - looking at plots could be very hard

Evaluate how much a pattern is interesting

- several methods, one of them is based on *information theory*
 - information theory is primarily used in telecommunications
 - it is based on the concept of *entropy*
 - information content, surprise, ...
 - Claude Shannon, “A Mathematical Theory of Communication”, 1948

The bit transmission example

- given a variable with 4 possible values and a given probability distribution
 $P(A) = 0.25, P(B) = 0.25, P(C) = 0.25, P(D) = 0.25$
- an observation of the data stream could return
BAACBADCADDAA ...
- the observation could be transmitted on a serial digital line with a two-bit **coding**
 $A = 00, B = 01, C = 10, D = 11$
- the transmission will be
0100001001001110110011111100 ...

Less bits

- What if the probability distributions are uneven?
 $P(A) = 0.5, P(B) = 0.25, P(C) = 0.125, P(D) = 0.125$
- of course, the coding shown above is possible, requiring two bits per symbol
- is there a coding requiring only 1.75 bit per symbol, on the average?

Less bits

- What if the probability distributions are uneven?
 $P(A) = 0.5, P(B) = 0.25, P(C) = 0.125, P(D) = 0.125$
- of course, the coding shown above is possible, requiring two bits per symbol
- is there a coding requiring only 1.75 bit per symbol, on the average?

$$A = 0, B = 10, C = 110, D = 111$$

Even less bits

- What if there are only three symbols with equal probability?
 $P(A) = 1/3, P(B) = 1/3, P(C) = 1/3$
- of course, the two-bit coding shown above is still possible
- is there a coding requiring less than 1.6 bit per symbol, on the average?

Even less bits

- What if there are only three symbols with equal probability?
 $P(A) = 1/3, P(B) = 1/3, P(C) = 1/3$
- of course, the two-bit coding shown above is still possible
- is there a coding requiring less than 1.6 bit per symbol, on the average?

$A = 0, B = 10, C = 11$ or any permutation of the assignment

General case

- Given a source X with V possible values, with probability distribution

$$P(v_1) = p_1, P(v_2) = p_2, \dots, P(v_V) = p_V$$

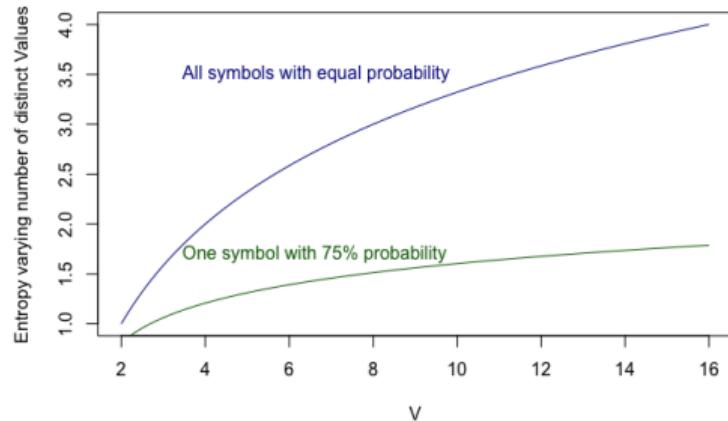
- the best coding allows the transmission with an average number of bits given by

$$H(X) = - \sum_j p_j \log_2(p_j)$$

$H(X)$ is the *entropy* of the information source X

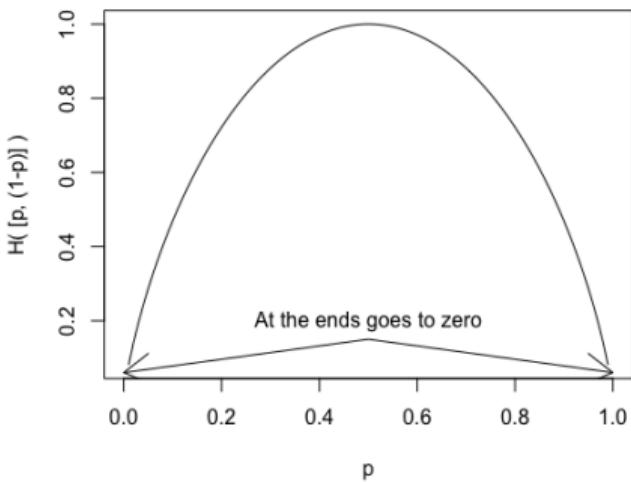
Meaning of entropy of an information source

- high entropy means that the probabilities are mostly similar
 - the histogram would be *flat*
- low entropy means that some symbols have much higher probability
 - the histogram would have *peaks*
- higher number of allowed symbols (i.e. of distinct values in an attribute) gives higher entropy



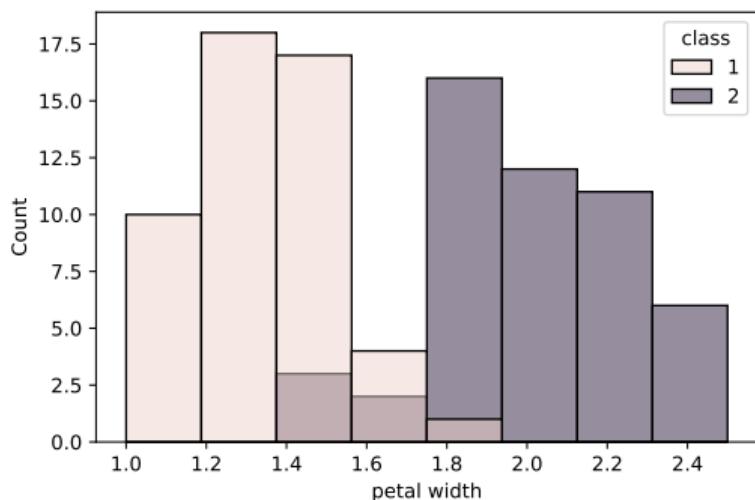
Entropy of a binary source

In a binary source with symbol probabilities p and $(1-p)$ when p is 0 or 1 the entropy goes to 0



Entropy for the target column class in the reduced Iris dataset

A subset: only the fourth data column and the target, only the rows with classes 1 or 2



petal width (cm)	class
2	2
1.7	1
1.3	1
2.2	2
1.5	1
1.5	2
2.3	2
2	2
2.5	2
1.7	2
...	...

$$N = 100 \quad p_{\text{class}=1} = 0.5, p_{\text{class}=2} = 0.5$$

$$H_{\text{class}} = -(p_{\text{class}=1} * \log_2(p_{\text{class}=1}) + p_{\text{class}=2} * \log_2(p_{\text{class}=2})) = 1$$

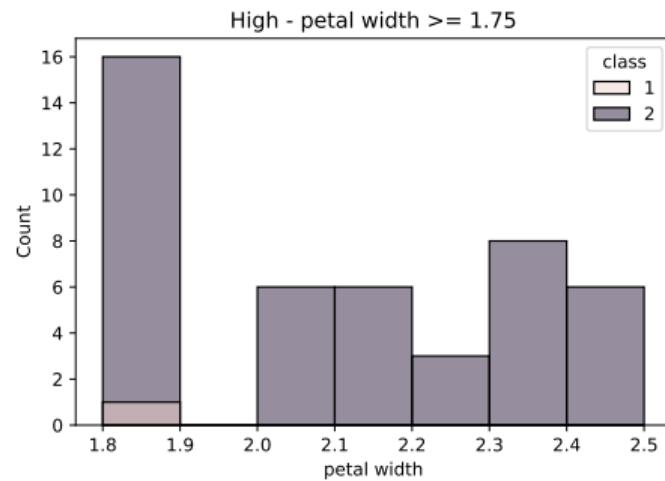
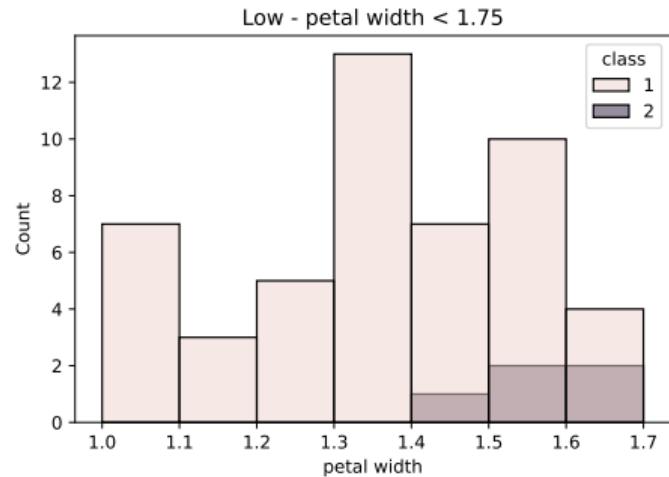
Entropy after a threshold-based split

- Splitting the dataset in two parts according to a threshold on a numeric attribute the entropy changes, and becomes the weighted sum of the entropies of the two parts
 - the weights are the relative sizes of the two parts
- Let $d \in \mathcal{D}$ be a real-valued attribute, let t be a value of the domain of d , let c be the class attribute
- We define the entropy of c w.r.t. d with threshold t as
$$H(c|d : t) = H(c|d < t) * P(d < t) + H(c|d \geq t) * P(d \geq t)$$

Information Gain for binary split

- It is the reduction of the entropy of a target class obtained with a split of the dataset based on a threshold for a given attribute
- We define $IG(c|d : t) = H(c) - H(c|d : t)$
 - it is the information gain provided when we know if, for an individual, d exceeds the threshold t in order to forecast the class value
- We define $IG(c|d) = \max_t IG(c|d : t)$

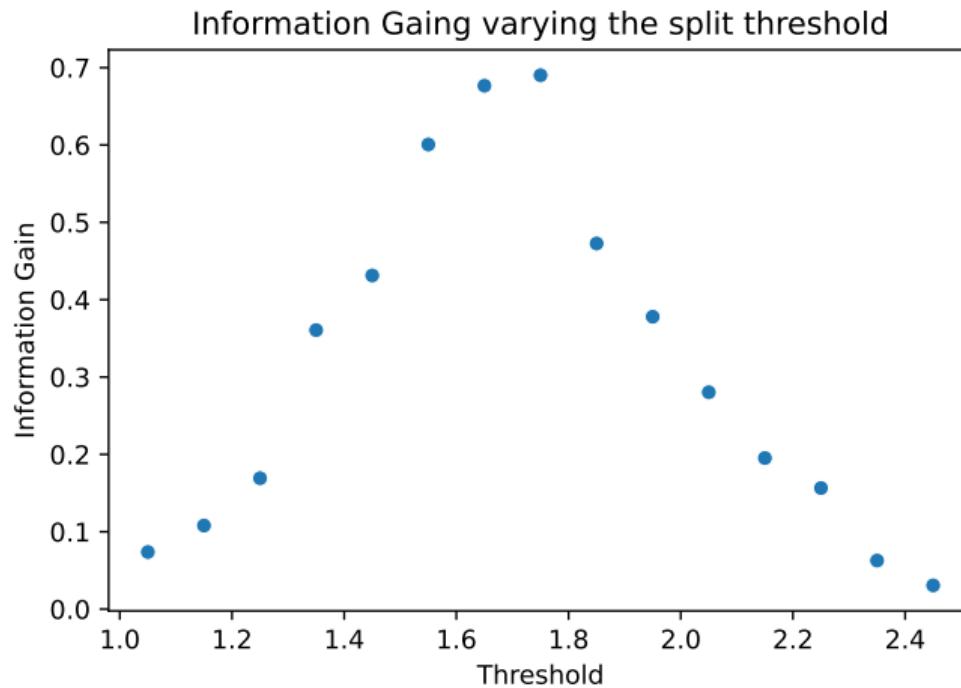
Let's split the reduced Iris with threshold 1.8



	low	high
1	49	1
2	5	45
	54	46

$$\begin{aligned}
 H(class|petalwidth : 1.8) = 0.31 = \\
 - (49/54 * \log_2(49/54) + 5/54 * \log_2(5/54)) * 0.54 + \\
 (1/46 * \log_2(1/46) + 45/46 * \log_2(45/46)) * 0.46
 \end{aligned}$$

Change the threshold to find the best split



How can we use the information gain?

Predict the probability of long life given some historical data on person characteristics and life style

- $IG(LongLife|HairColor) = 0.01$
- $IG(LongLife|Smoker) = 0.2$
- $IG(LongLife|Gender) = 0.25$
- $IG(LongLife|LastDigitSSN) = 0.00001$

Correlations between attributes is an important issue: it is not considered here

Back to DT generation

Choosing the attribute to test

A *decision tree* is a tree-structured plan generating a sequence of tests on the known attributes (*predicting attributes*) to predict the values of an unknown attribute.

Consider question 1 of page 22: *which attribute should we test?*

- test the attribute which guarantees the maximum IG for the class attribute in the current data set \mathcal{X}
- partition \mathcal{X} according to the test outcomes
- recursion on the partitioned data

Train/Test split⁴

- The supervised data set will be split in (at least) two parts:
 - Training set used to learn the model
 - Test set used to evaluate the learned model on fresh data
- The split is done randomly
- Assumption: the parts have similar characteristics
- The proportion of the split is decided by the experimenter
 - Common solutions: 80-20, 67-33, 50-50
- The following slides consider a 50-50 split of the Iris dataset³
 - Entropies for the training and test class columns are both 1.58

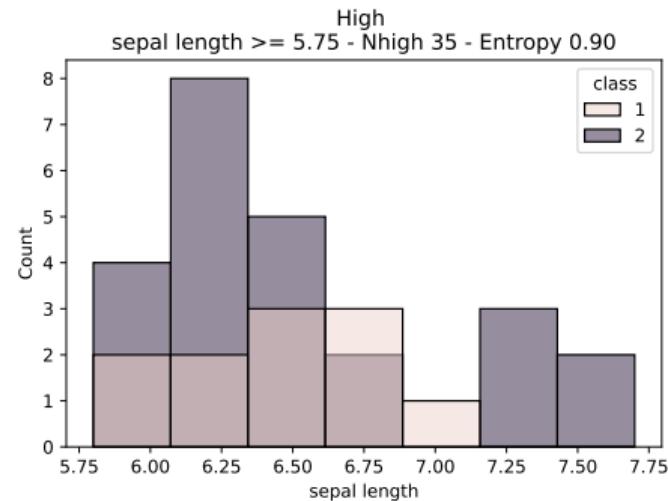
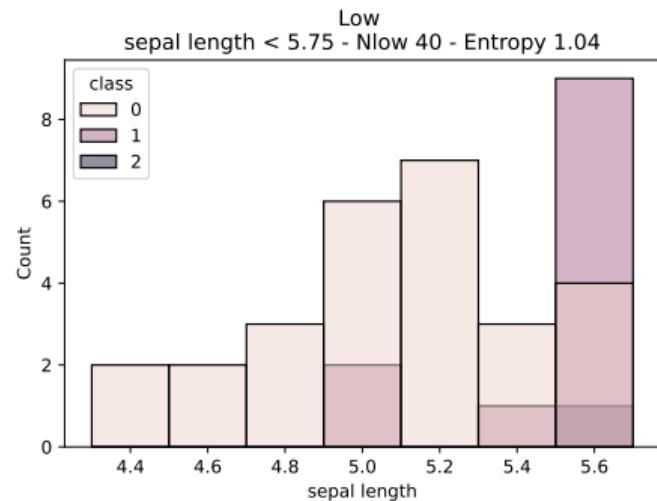
³ In the example, the split has been done using `sklearn.model_selection.train_test_split` and `random_state = 10`

⁴ You can read [this link](#) for a short discussion

Iris Dataset - Predicting attribute: Sepal Length

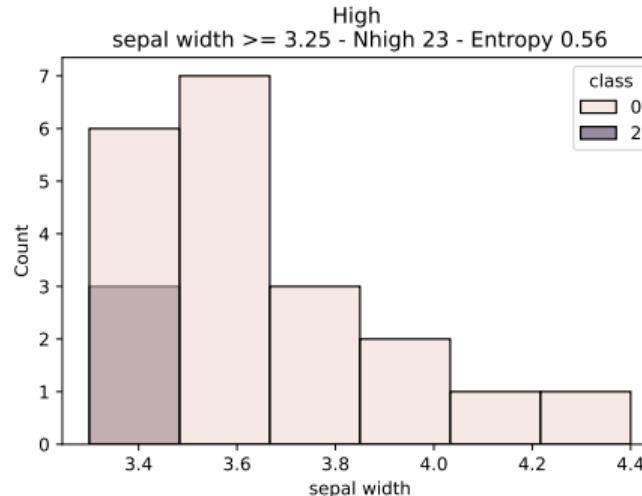
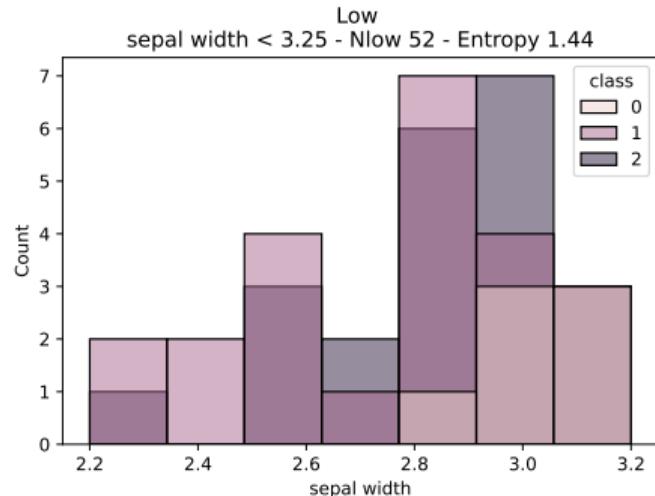
Best threshold: 5.75

$$\text{Information Gain: } 1.58 - (40 * 1.04 + 35 * 0.90)/75 = 0.61$$



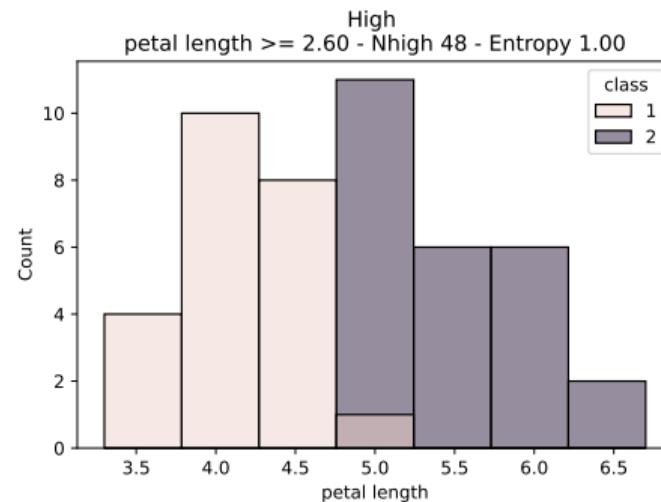
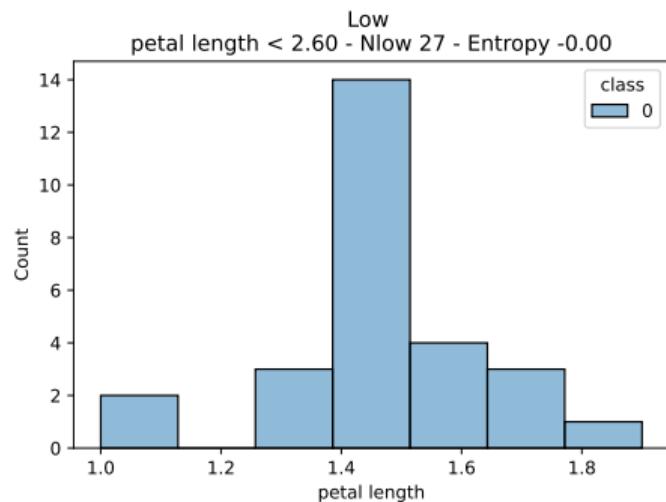
Iris Dataset - Predicting attribute: Sepal Width

Best threshold: 3.25 – Information Gain: $1.58 - (52*1.44 + 23*0.56) / 75 = 0.41$



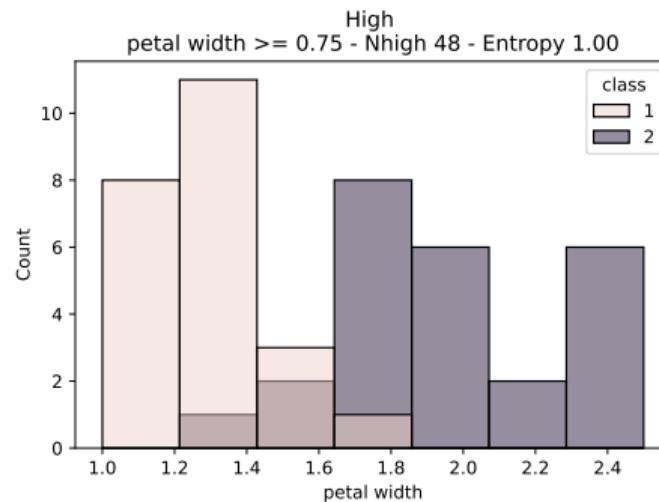
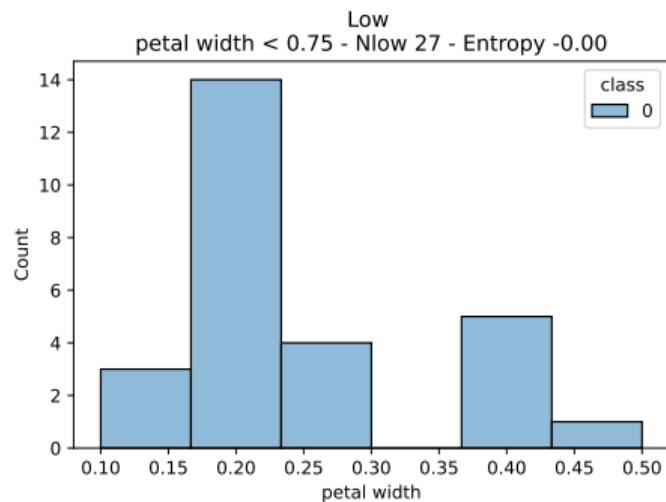
Iris Dataset - Predicting attribute: Petal Length

Best threshold: 2.6 – Information Gain: 0.94



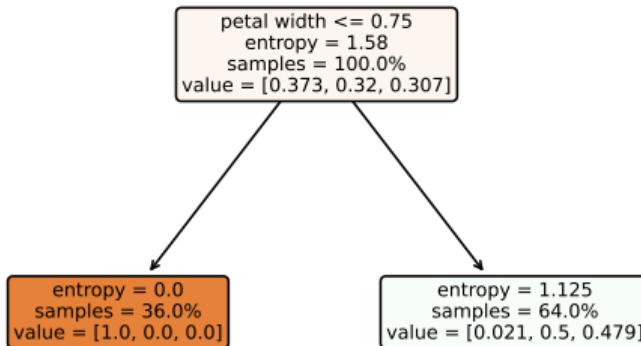
Iris Dataset - Predicting attribute: Petal Width

Best threshold: 0.75 – Information Gain: 0.94

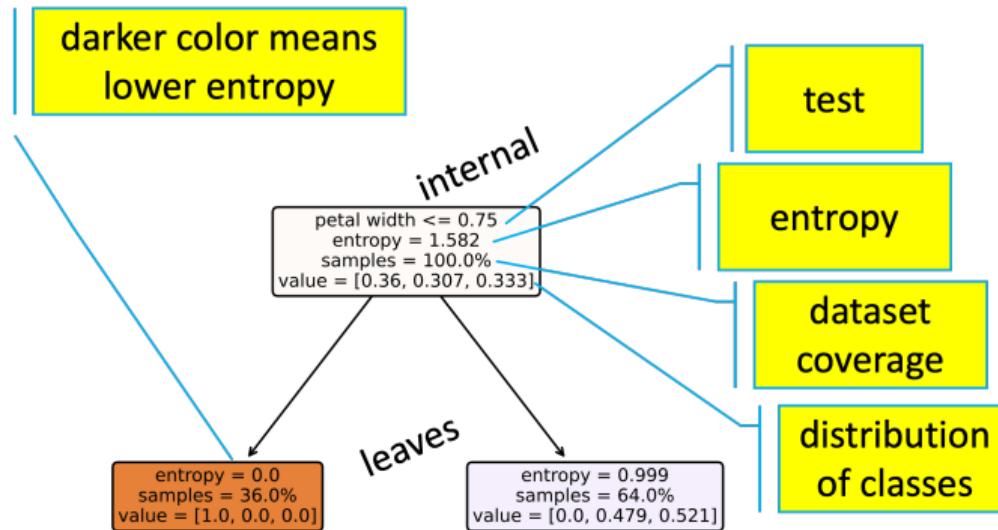


One-stump decision

- choose the attribute giving the highest IG
- partition the dataset according to the chosen attribute
- choose as class label of each partition the majority

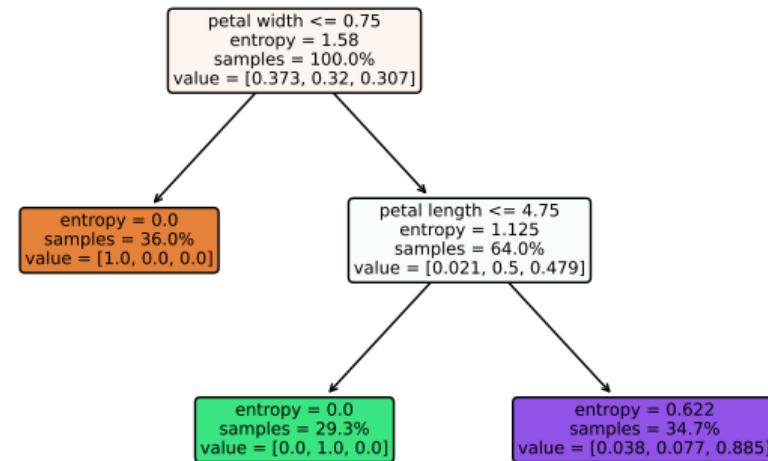


What's in a node



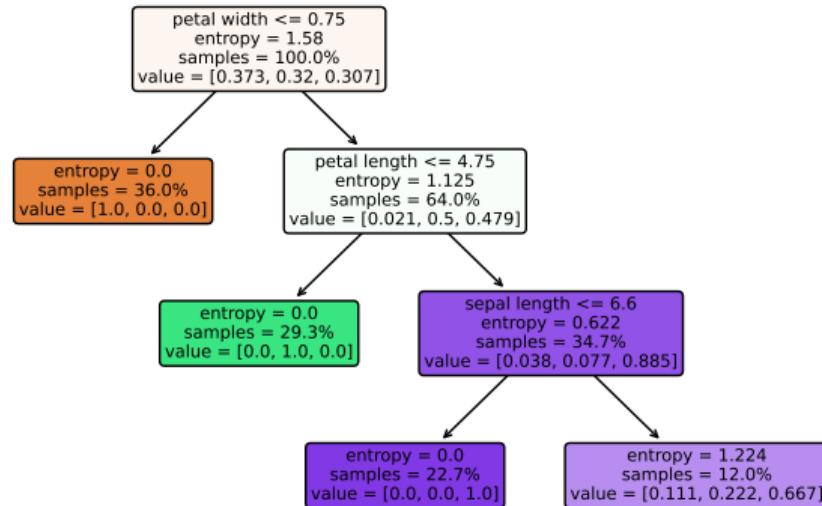
Recursion step

Build a new tree starting from each subset where the minority is non-empty



Recursion step

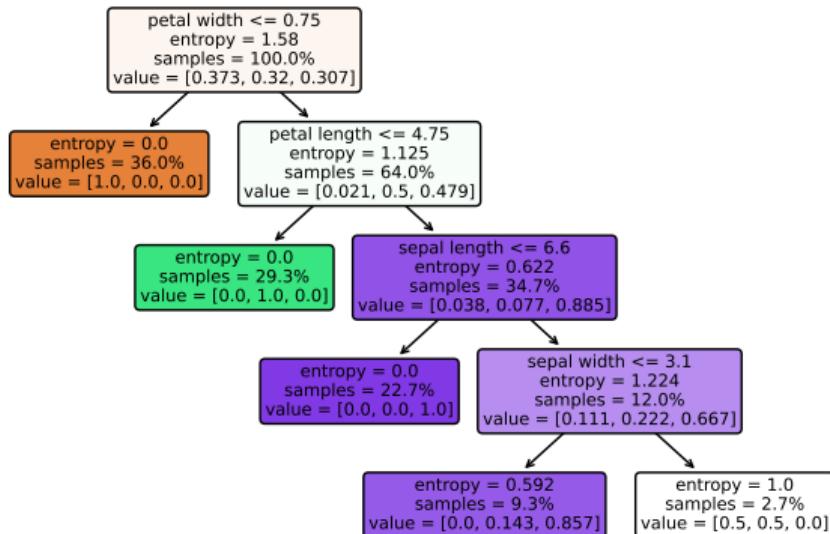
Build a new tree starting from each subset where the minority is non-empty



Recursion step

Observation

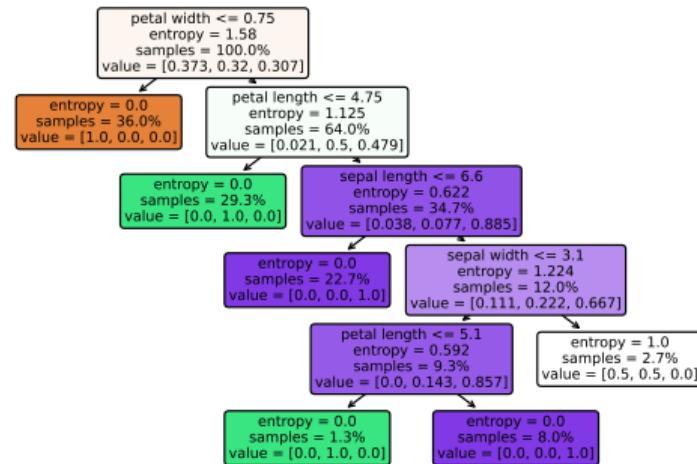
- The weighted sum of the entropy of the descendant nodes is always smaller than the entropy in the ancestor node, even if one of the descendants has higher entropy.
- Consider the three bottom right nodes (a=ancestor, ld=left descendant, rd=right descendant)



$$ent_a = 0.622 > (ent_{ld} * samp_{ld} + ent_{rd} * samp_{rd}) / samp_a = (0 * 22.7 + 1.224 * 12) / 34.7 = 0.39$$

Recursion ends

- Most of the leaves are **pure**, recursion impossible
- One of the leaves is not pure, but no more tests are able to give positive information gain, **recursion impossible**
 - it is labelled with the majority class, or, in case of tie, with one of the non-empty classes
- The error rate on the training set is 1.35%
 - 1 of the 75 examples in the training set is not correctly classified by the learned decision tree
 - it is one of the two items in the rightmost leaf



Building a *Decision Tree* with binary splits

```
procedure BUILDTREE(dataset  $\mathcal{X}$ , node  $p$ )
  if all the class values of  $\mathcal{X}$  are  $c$  then
    return node  $p$  as a leaf, label of  $p$  is  $c$ 
  if no attribute can give a positive information gain in  $\mathcal{X}$  then
    say that the majority of elements in  $\mathcal{X}$  has class  $c$ 
    return node  $p$  as a leaf, label of  $p$  is  $c$ 
  find the attribute  $d$  and threshold  $t$  giving maximum information gain in  $\mathcal{X}$ 
  create two internal nodes descendant of  $p$ , say  $p_{left}$  and  $p_{right}$ 
  let  $\mathcal{X}_{left}$  = selection on  $\mathcal{X}$  with  $d < t$ 
  BUILDTREE( $\mathcal{X}_{left}$ ,  $p_{left}$ )
  let  $\mathcal{X}_{right}$  = selection on  $\mathcal{X}$  with  $d \geq t$ 
  BUILDTREE( $\mathcal{X}_{right}$ ,  $p_{right}$ )
```

Decision tree for the Iris classifier

Internal representation

	ChLeft	ChRight	Feature	Threshold	NNodeSamples	Impurity
0	1	2	petal width (cm)	0.750000	75	1.579659
1	-	-	-	nan	27	0.000000
2	3	4	petal length (cm)	4.750000	48	1.124941
3	-	-	-	nan	22	0.000000
4	5	6	sepal length (cm)	6.600000	26	0.621904
5	-	-	-	nan	17	0.000000
6	7	10	sepal width (cm)	3.100000	9	1.224394
7	8	9	petal length (cm)	5.100000	7	0.591673
8	-	-	-	nan	1	0.000000
9	-	-	-	nan	6	0.000000
10	-	-	-	nan	2	1.000000

Training Set Error

- execute the generated decision tree on the training set itself
 - obviously the class attribute is hidden
- count the number of discordances between the true and the predicted class
- this is the *training set error*
 - it can be non-zero, due to
 - the limits of decision trees in general: a decision tree based on tests on attribute values can fail
 - insufficient information in the predicting attribute

Training Set Error

- Is this 1.35% interesting? What is its *meaning*?

Training Set Error

- Is this 1.35% interesting? What is its *meaning*?
- It is the error we make on the data we used to generate the classification model
- It is probably the **lower limit** of the error we can expect when classifying new data
- We are much more interested to an **upper limit**, or to a more significant value

Test set error

- The test set error is more indicative of the expected behaviour with new data
- Additional statistic reasoning can be used to infer error bounds given the test set error
- We have available 75 additional labelled records in the *Iris* dataset

Iris classification error

	Num Errors	Set Size	% Wrong
Training Set	1	75	1.35
Test Set	13	75	17.33

Iris classification error

	Num Errors	Set Size	% Wrong
Training Set	1	75	1.35
Test Set	13	75	17.33

Why the test set error is so much worse?

Overfitting 1/2

Definition: overfitting happens when the learning is affected by *noise*

When a learning algorithm is affected by noise, the performance on the test set is (much) worse than that on the training set

Overfitting 2/2

More formally

A decision tree is a *hypothesis* of the relationship between the predictor attributes and the class. Some definitions:

- h = hypothesis
- $\text{error}_{\text{train}}(h)$ = error of the hypothesis on the training set
- $\text{error}_{\mathcal{X}}(h)$ = error of the hypothesis on the entire dataset

h overfits the training set if there is an alternative hypothesis h' such that

$$\text{error}_{\text{train}}(h) < \text{error}_{\text{train}}(h')$$

$$\text{error}_{\mathcal{X}}(h) > \text{error}_{\mathcal{X}}(h')$$

Causes for overfitting

1. Presence of noise

- individuals in the test set can have bad values in the predicting attributes and/or in the class label

2. Lack of representative instances

- some situations of the real world can be underrepresented, or not represented at all, in the training set
- this situation is quite common

A good hypothesis has low *generalization* error i.e. it works well on examples different from those used in training

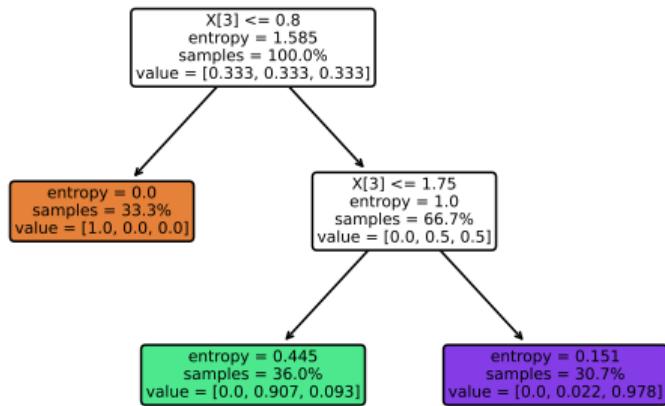
Occam's Razor⁵

*Everything should be made
as simple as possible,
but not simpler*

- all other things being equal, simple theories are preferable to complex ones
- a long hypothesis that fits the data is more likely to be a coincidence
- **pruning** a decision tree is a way to simplify it
 - we need to find precise, quantitative guidelines for effective pruning

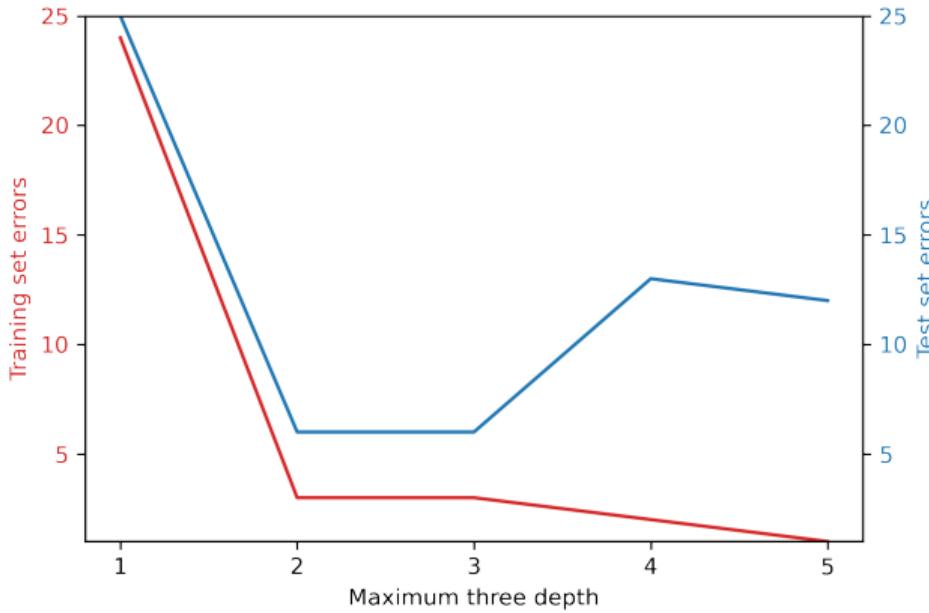
⁵ William of Ockham, an english franciscan philosopher of the 14-th century

Example: Iris classification with pruned tree



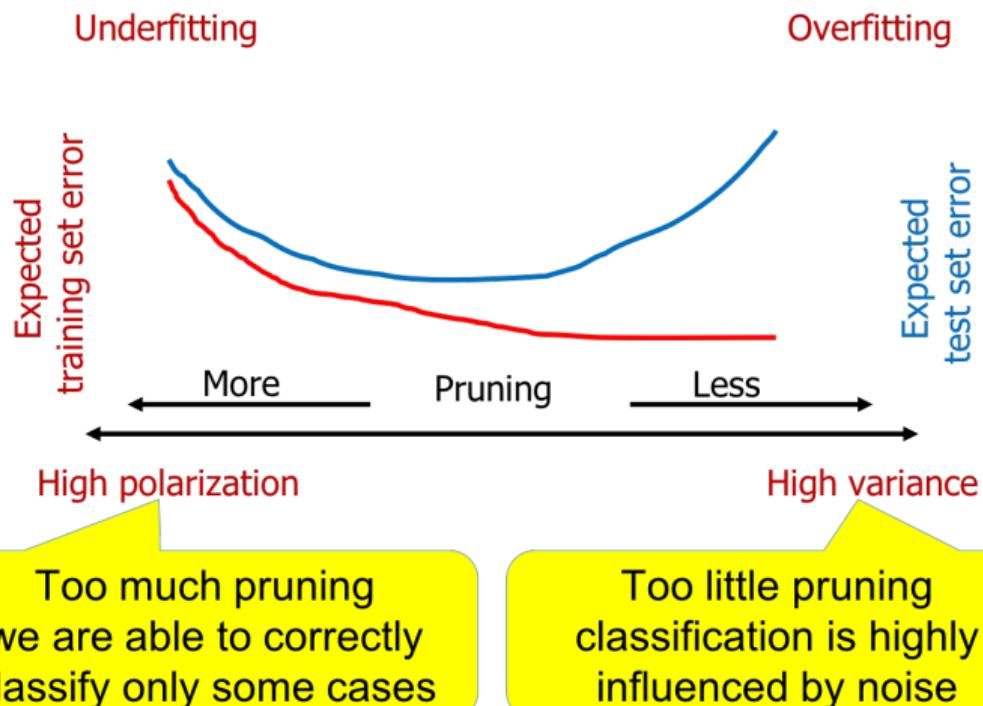
	Num Errors	Set Size	% Wrong
Training Set	3	75	4.00
Test Set	6	75	8.00

Iris: Training and test errors varying tree depth



General effect of model simplification

Pruning is the way to *simplify* the model when you are using a decision tree



Choice of the attribute to split the dataset

- Looking for the split generating the maximum **purity**
- We need a measure for the purity of a node
 - a node with two classes in the same proportion has low purity
 - a node with only one class has highest purity

Impurity functions

Measures of the impurity of a node

Entropy – already seen ⁶

Gini Index

Misclassification Error

OPTIONAL

⁶ it is available in *Scikit-Learn*

Gini Index⁷ – Intuition

- Consider a node p with C_p classes
- Which is the frequency of the wrong classification in class j given by a random assignment based only on the class frequencies in the current node?
- For class j
 - frequency $f_{p,j}$
 - frequency of the other classes $1 - f_{p,j}$
 - probability of wrong assignment $f_{p,j} * (1 - f_{p,j})$
- the Gini Index is the total probability of wrong classification

$$\sum_j f_{p,j} * (1 - f_{p,j}) = \sum_j f_{p,j} - \sum_j f_{p,j}^2 = 1 - \sum_j f_{p,j}^2$$

⁷ This is the default impurity measure in *Scikit-Learn*

Gini Index – Discussion

- the maximum value is when all the records are uniformly distributed over all the classes: $1 - 1/C_p$
- the minimum value is when all the records belong to the same class: 0

Splitting based on the Gini Index

- Used by CART, SLIQ, SPRINT
- When a node p is split into ds descendants, say p_1, \dots, p_{ds}
- Let $N_{p,i}$ and N_p be the number of records in the i -th descendant node and in the root, respectively
- We choose the split giving the maximum reduction of the Gini Index

$$GINI_{split} = GINI_p - \sum_{i=1}^{ds} \frac{N_{p,i}}{N_p} GINI(p_i)$$

Misclassification Error

OPTIONAL

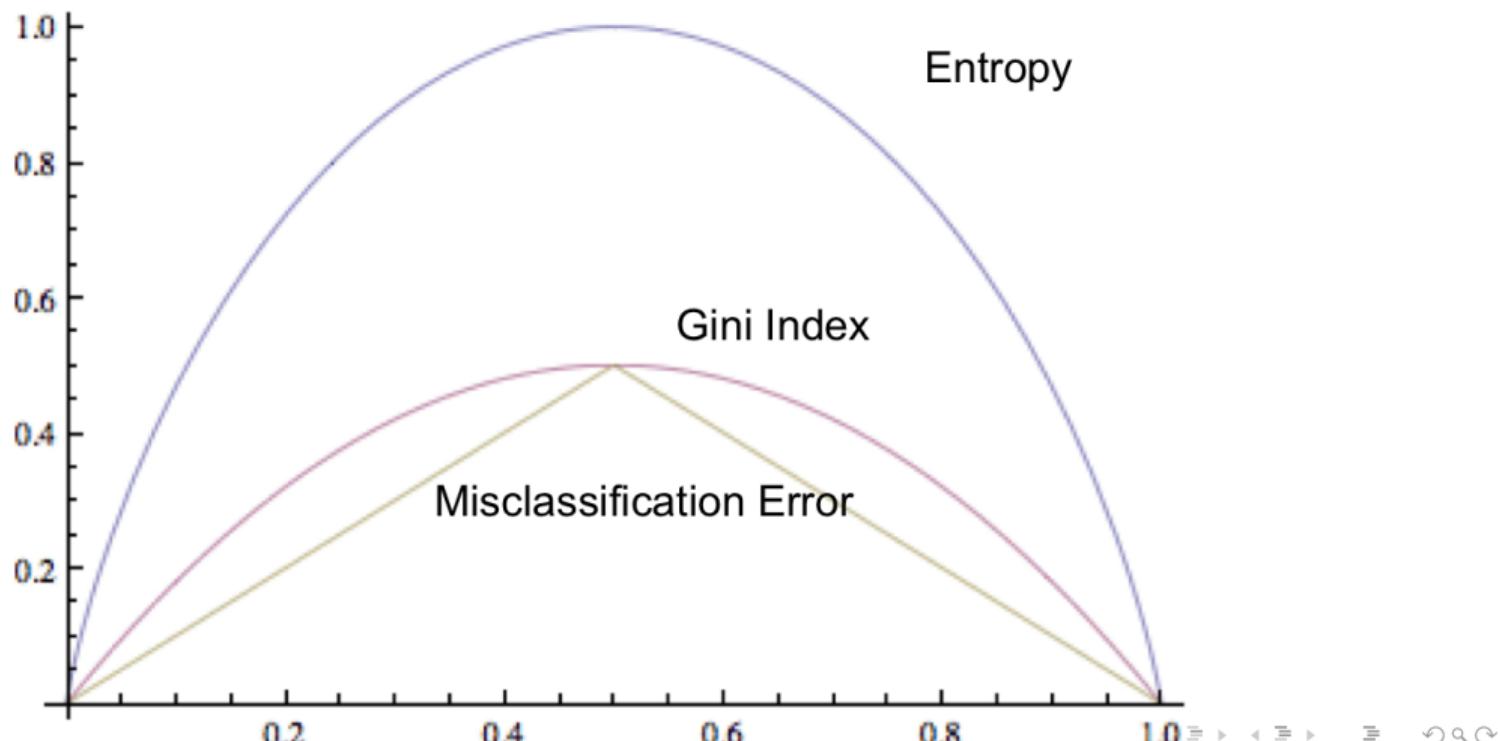
- If a node is a leaf, we find the highest label frequency; this frequency is the **accuracy** of the node and this label is the output of the node
- The **misclassification error** is the complement to 1 of the accuracy
- Since the most frequent class determines the node label, the complement is the error
 - The maximum value is when all the records are uniformly distributed over all the classes: $1 - 1/C_p$
 - The minimum value is when all the records belong to the same class: 0
- The choice of the split is done in the same way as for the Gini index

$$ME(p) = 1 - \max_j f_{p,j}$$

Comparison of the impurity functions

OPTIONAL

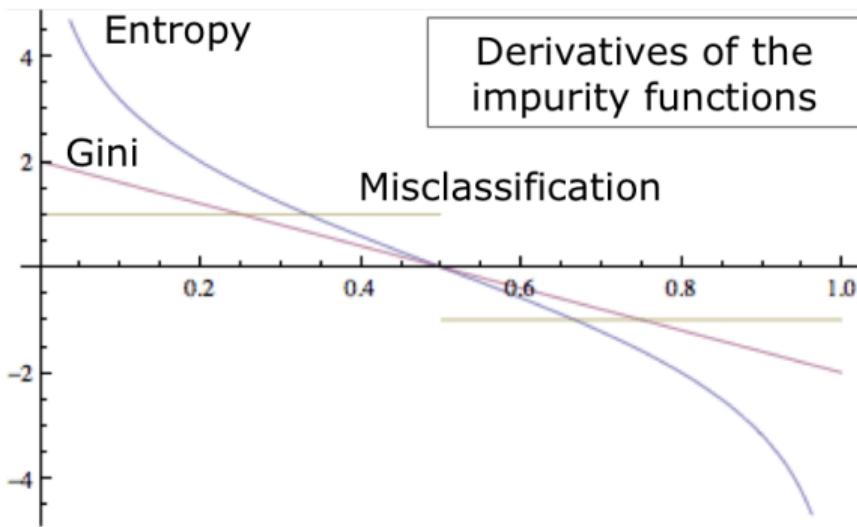
For two classes with frequencies f and $1 - f$



Comparison of the impurity functions – Discussion

OPTIONAL

- The behavior of ME is linear, therefore an error in the frequency is completely transferred into the impurity computation
- Entropy and Gini have varying derivative, with the minimum around the center
 - they are more robust w.r.t. errors in the frequency, when the frequencies of the two



Algorithms for building DTs

- Several variants, depending on
 - tree construction strategy
 - partition strategy
 - pruning strategy
- Tests based on linear combinations of numeric attributes
- Multivariate tests (e.g. $a = x$ and $b = y$)
- ...

Complexity of DT induction I

- N instances and D attributes in \mathcal{X}
 - tree height is $\mathcal{O}(\log N)$
- Each level of the tree requires the consideration of all the dataset (considering all the nodes)
- Each node requires the consideration of all the attributes
 - overall cost is $\mathcal{O}(DN \log N)$

Complexity of DT induction II

- In addition

- binary split of numeric attributes costs $\mathcal{O}(N \log N)$, without increment of complexity
- pruning requires the consideration of each node, but nodes are $2N - 1$, at most⁸
- pruning requires to consider globally all instances at each level, generating an additional $\mathcal{O}(N \log N)$, which does not increase complexity.

8 If the tree is binary and does not degenerate.

Characteristics of DT Induction I

1. It is a **non-parametric** approach to build classification models
 - it does not require any assumption on the probability distributions of classes and attribute values
2. Finding the **best** DT is NP-complete, the heuristic algorithms allow to find sub-optimal solutions in reasonable times
3. The run-time use of a DT to classify new instances is extremely efficient: $\mathcal{O}(h)$, where h is the height of the tree
4. Robust w.r.t. noise in the training set (i.e. wrong class labels), if the overfitting is avoided with appropriate pruning
5. Redundant attributes do not cause any difficulty

Characteristics of DT Induction II

- In case of strong correlation between two attributes, if one is chosen for a split, most likely the other will never provide a good increment of node purity, and will never be chosen
- 6. The nodes at a high depth are easily irrelevant (and therefore pruned), due to the low number of training records they cover
- 7. In practice, the impurity measure has low impact on the final result
- 8. In practice, the pruning strategy has high impact on the final result

Conclusion

- Decision trees are usually the best starting point to learn supervised machine learning
 - easy to understand
 - easy to implement
 - easy to use
- Are prone to overfitting, as all the classification methods
- Are able to predict discrete values (the class) on the basis of continuous or discrete predictor attributes⁹

9 The Scikit-Learn implementation of Decision Trees do not allow discrete attributes, therefore in these cases a *data transformation* is necessary

Important concepts

- Impurity functions: entropy, Gini, misclassification
- The recursive greedy algorithm for building a decision tree
- Training error and test error
- Why the test error can be much greater than the training error
- Why the pruning can improve the performance
- How to deal with continuous attributes

Questions

- Why maximising the Information Gain and the Gini Index gain should be, in general, better than minimising the Misclassification Error?
- Why do we prefer a greedy algorithm instead of trying all the possible trees?
- If the decision tree to predict wealth has the marital status near to the top, can we say that the marital status is a major cause for wealth?
- Can we say that the attributes which are not mentioned in the tree are not a cause for wealth?

Bibliography I

- ▶ Wray Buntine.
Learning classification trees.
Statistics and Computing, 2(2):63–73, Jun 1992.
ISSN 1573-1375.
doi: 10.1007/BF01889584.
URL <https://doi.org/10.1007/BF01889584>.
- ▶ Earl B. Hunt, Janet Marin, and Philip J. Stone.
Experiments in induction.
Academic Press, 1966.
URL <https://books.google.it/books?id=sQoLAAAAMAAJ>.
- ▶ Ross Quinlan.
Discovering rules by induction from large collections of examples.
In *Expert systems in the micro electronic age*. Edinburgh University Press, 1979.

Bibliography II

- ▶ Xindong Wu, Vipin Kumar, J. Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J. McLachlan, Angus F. M. Ng, Bing Liu, Philip S. Yu, Zhi-Hua Zhou, Michael Steinbach, David J. Hand, and Dan Steinberg.
Top 10 algorithms in data mining.
Knowl. Inf. Syst., 14(1):1–37, 2008.
doi: 10.1007/s10115-007-0114-2.
URL <http://dx.doi.org/10.1007/s10115-007-0114-2>.

Machine Learning

Classification - Part II

Claudio Sartori

DISI

Department of Computer Science and Engineering – University of Bologna, Italy
claudio.sartori@unibo.it

1

Model Selection: Evaluation of a classifier

2

- Error estimation 8
- Statistical pruning of DT 14
- Other pruning techniques 17
- Testing a classifier 18
- Performance measures of a classifier 37
- κ statistic 46
- The cost of errors 51

2

Evaluation of a probabilistic classifier

54

Model Selection: Evaluation of a Classifier

Questions to be answered

- Which of the available models for classification is the best one?
- Which of the available algorithms is the best one?
- Which is the best parameters configuration?

⇒

Evaluation

The Oil Slick example I

- Detect oil slick (failures, illegal dumping) from satellite images, for early alarm
- Radar satellite images
 - Dark regions whose size and shape depend on weather and sea conditions
 - Look-alike dark regions can also be caused by local weather conditions, such as high winds
 - Manual detection by experts is definitely expensive and slow
- Scarcity of training data: oil spills are, fortunately, rare
- Unbalanced nature of data: the negative examples (non-spills) are predominant over the positive ones

The Oil Slick example II

- An automatic hazard detection system has been developed and marketed
 - Pre-selection of images for final manual processing
 - Necessary a tradeoff between undetected spills and false alarms
 - Evaluation of performance guides the tradeoff

The training set

- In supervised learning the training set performance is **overoptimistic**
- We need a lower bound for performance obtained by independent tests
- Supervised data are usually scarce, we need to balance the use of them between:
 - train
 - validation, to tune the parameter (sometimes it is omitted)
 - test
- Evaluate how much the **theory fits the data**
- Evaluate the **cost generated by prediction errors**

Learning and evaluation

- Empirically (and intuitively) the more training data we use the best performance we should expect
 - Statistically, we should expect a larger covering of the situation that can occur when classifying new data
- We must consider the effect of random changes
- The evaluation is **independent** from the algorithm used to generate the model

The meaning of the *test error*

- let us suppose that the test set is a good representation, *on the average*, of the entire dataset \mathcal{X} (i.e. run-time)
- the relationship between the training set and \mathcal{X} will be subject to probabilistic variability
- the evaluation can be done either at different levels
 - **general** – the whole performance of the classifier
 - **local** – the local performance of a component of the model, i.e. a **node** of a decision tree
- if the test set error ratio is x , we should expect a run-time error $x \pm ???$

⇒ **confidence interval**

Confidence interval in error estimation

Bernoulli process

- forecasting each element of the test set is like one experiment of a Bernoulli process
 - good prediction \Rightarrow success
 - bad prediction \Rightarrow error
- the same as N independent binary random events of the same type
- $f = S/N =$ empirical frequency of error
- which is p the probability of error?

Empirical frequency and true frequency

- Deviations of the empirical frequency from the true frequency are due to *noise*
- Usually, noise is assumed to have a normal distribution around the true probability (for $N \geq 30$)
- We choose the **confidence level**, i.e. the probability that the true frequency of success is below the pessimistic frequency that we will compute

$$P\left(z_{\alpha/2} \leq \frac{f - p}{\sqrt{p(1-p)/N}} \leq z_{1-\alpha/2}\right) = 1 - \alpha$$

Range error estimate

Wilson score interval

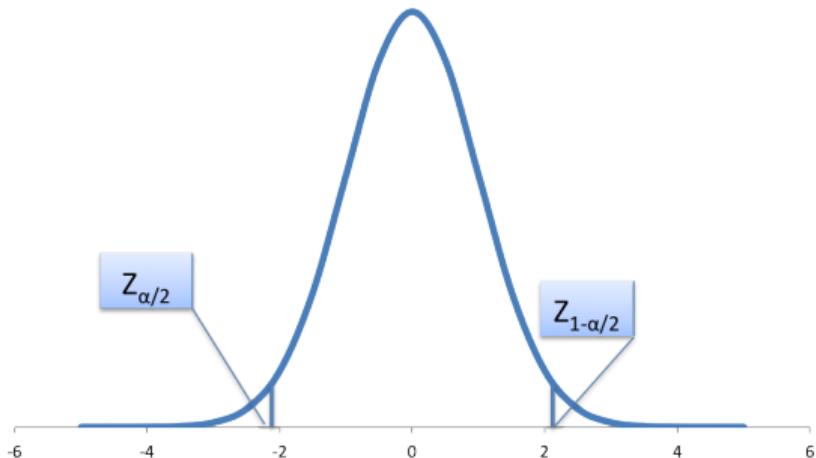
- z depends on the desired confidence level α
- it is the abscissa delimiting the area $1 - \alpha$ for a normal distribution
 - i.e. the inverse of the standard cumulative normal distribution
- with a little of algebra

$$\frac{1}{1 + \frac{1}{N}z^2} \left[f + \frac{1}{2N}z^2 \pm z\sqrt{\frac{1}{N}f(1-f) + \frac{1}{4N^2}z^2} \right]$$

The **pessimistic** error is obtained by substituting \pm with $+$

Confidence level in error estimation

α is the probability of a wrong estimate

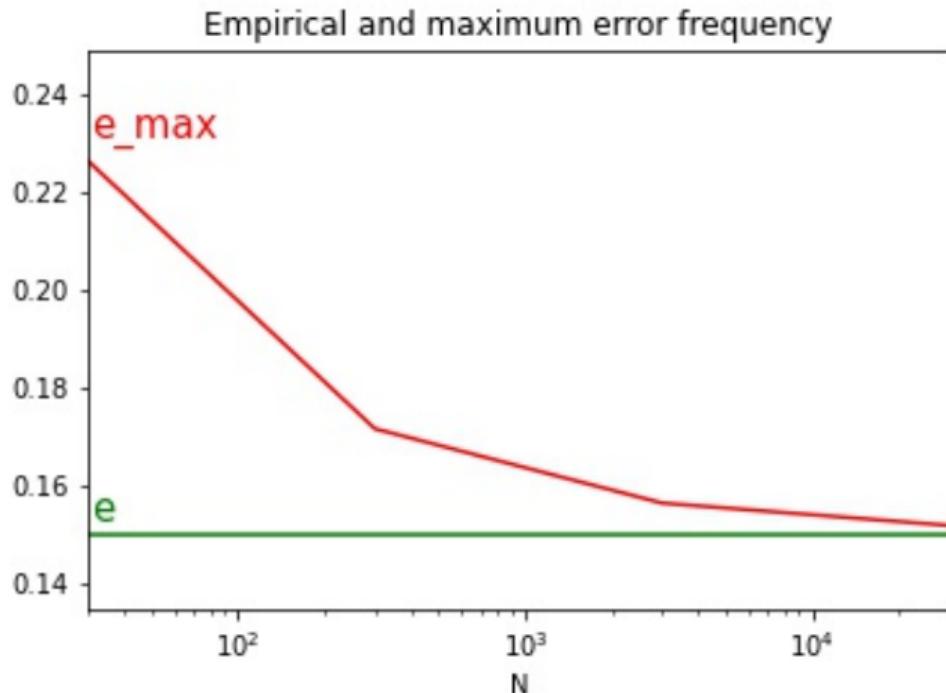


$1 - \alpha$	Z
0.99	2.58
0.98	2.33
0.95	1.96
0.90	1.65
0.75	1.04
0.50	0.67

Confidence interval in error estimation

Increasing N , with constant empirical frequency, the uncertainty for p narrows.

Example:
 $f = 85\%$, $\alpha = 0.05$

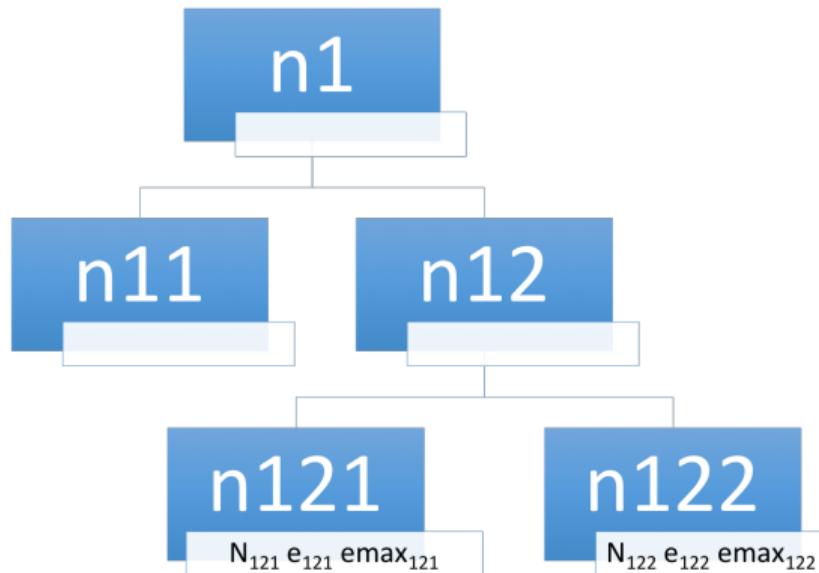


Statistical pruning of DT with error estimation

The C4.5 strategy

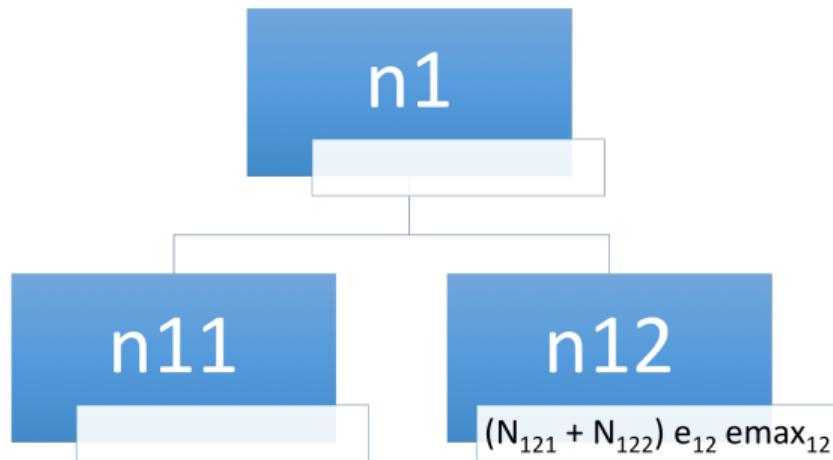
- Consider a subtree near the leaves
- Compute its maximum error e_l as a weighted sum of the maximum error of the leaves
- Compute the maximum error e_r of the root of the subtree transformed into a leaf
- Prune if $e_r \leq e_l$
- With pruning, the error frequency increases, but the number of records in node also increases, therefore the maximum error can decrease

DT before pruning



N_x = Records in node f_x = Error frequency in node e_x = Maximum error frequency in node

DT after pruning



$e_x - f_x$ increases as N_x decreases

Pruning is done if $(N_{121} + N_{122}) * e_{12} < N_{121} * e_{121} + N_{122} * e_{122}$

Other pruning techniques - examples

Scikit-Learn allows to adjust pruning with one or more of the hyperparameters below

- **max_depth** - the maximum depth allowed for the tree; it is a *horizontal cut*, pruning all the branches below a given depth
- **min_samples_split** - either the minimum absolute number of samples or the minimum fraction of samples (with respect to the entire population in the dataset) in a node to make a split, if the threshold is not exceeded the node becomes a leaf
- **min_samples_leaf** - the minimum number of samples (or fraction, as above) required to be at a leaf node
- **min_impurity_decrease** - a node will be split if this split induces a decrease of the impurity greater than or equal to this value; if the weighted sum of the descendant leaves do not decrease from the node under consideration more than this threshold then the node becomes a leaf

Accuracy of a classifier

- The error frequency is the simplest indicator of the quality of a classifier
 - it is the sum of errors **on any class** divided by the number of tested records
- From now on, for simplicity, we will use the empirical error frequencies
 - remember that in real cases the maximum error frequencies should be used instead
- Accuracy and other more sophisticated indicators are used to:
 - compare different classifiers or parameter settings
 - estimate the run-time performance we can expect, and therefore the **cost of errors**

The *hyperparameters*

Optimizing the model learned

- Every machine learning algorithm has one or more parameters than influence its behaviour
 - they are usually called *hyperparameters*
- Several train/test loops are in general necessary to find the best set of values for the hyperparameters
- It is crucial to obtain a highly reliable estimate of the run-time performance
- Sometimes it is necessary to find the best compromise between the optimisation step and the quality of the result

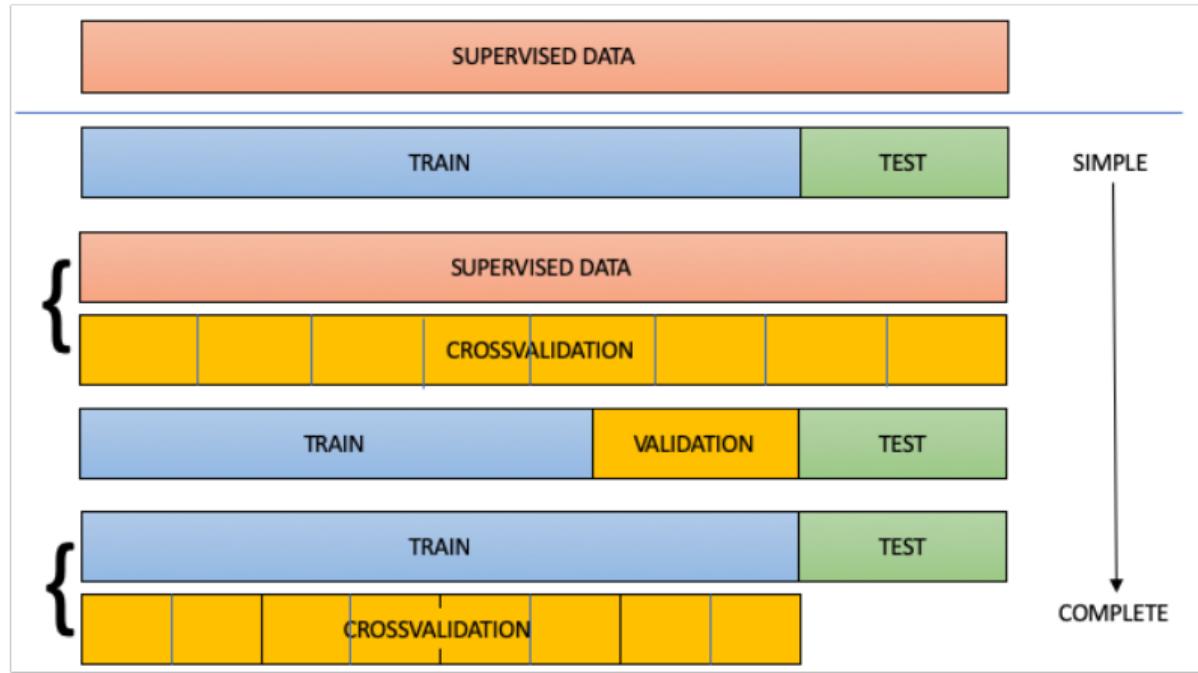
Testing strategies

Getting the most out of the available supervised data

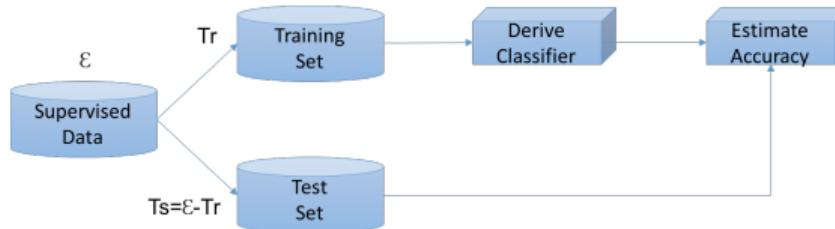
Problem: in every step the data should be *representative* of the data that will be classified run-time

- Holdout
 - splitting data into *training set* and *test set*
 - splitting data into *training set*, *validation set* and *test set*
- Cross validation
 - repeated tests with different splits

The train/test process: some alternatives



Holdout

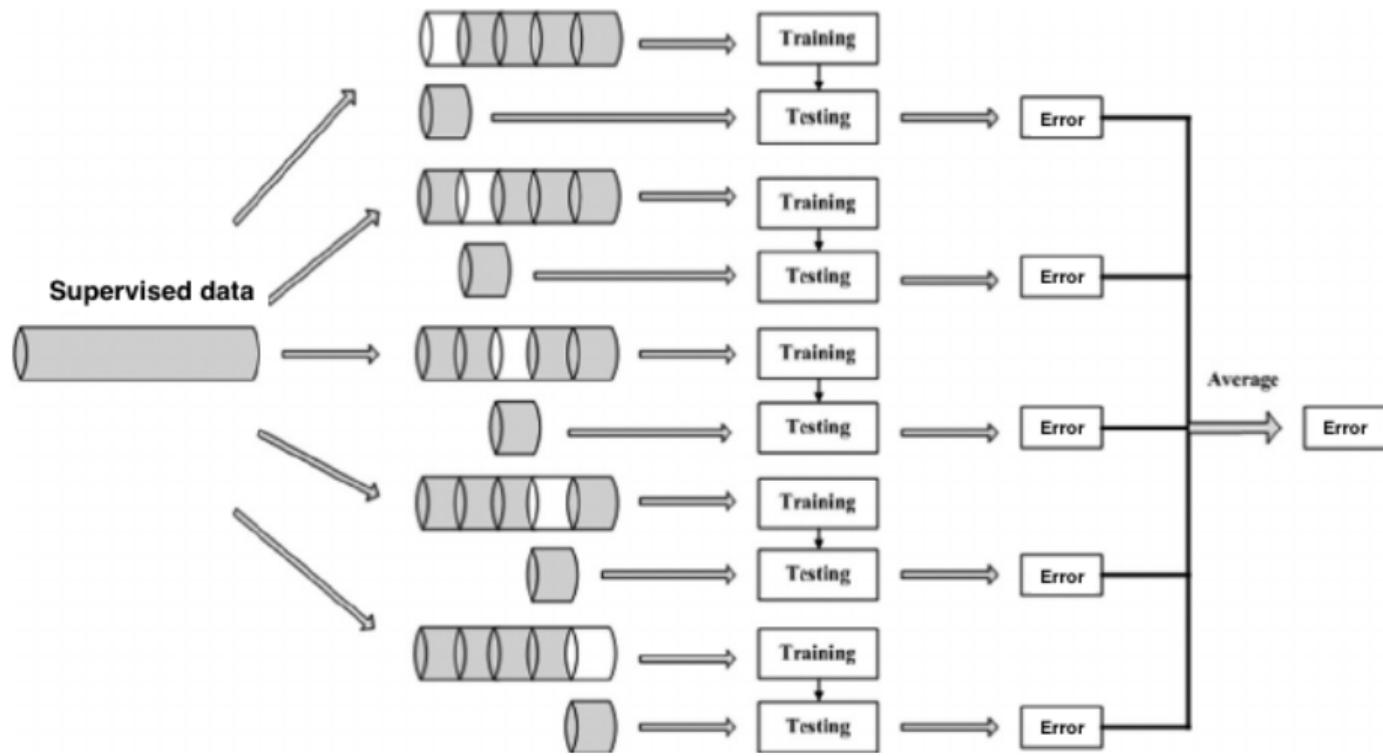


- A typical value of the training/test ratio is 2/1
- The split should be as random as possible
 - It may happen that the proportion of classes in the supervised dataset \mathcal{X} is altered in the Training and Test sets, to prevent such cases the statistical sampling technique of **stratification** ensures the maintenance of the proportion of classes
- In this setting, the test set is used to obtain an *estimation* of the performance measures with new data

Cross Validation (k -fold)

- The training set is randomly partitioned into k subsets
 - If necessary, use stratified partitioning
- k iterations using one of the subsets for test and the others for training
- Combine the result of tests
- Generate the final model using the *entire training set*
- Optimal use of the supervised data
 - each record is used $k - 1$ times for training and once for testing
- Typical value: $k = 10$

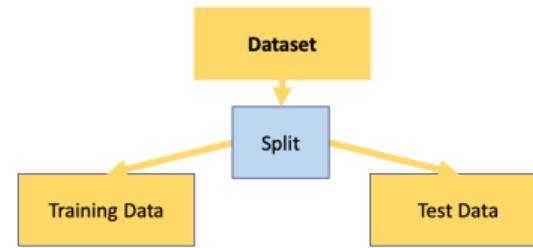
Cross Validation



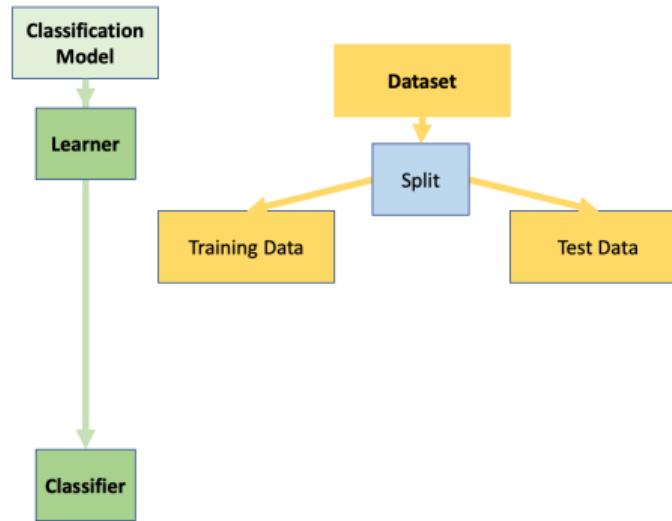
Cross-validation workflow

Dataset

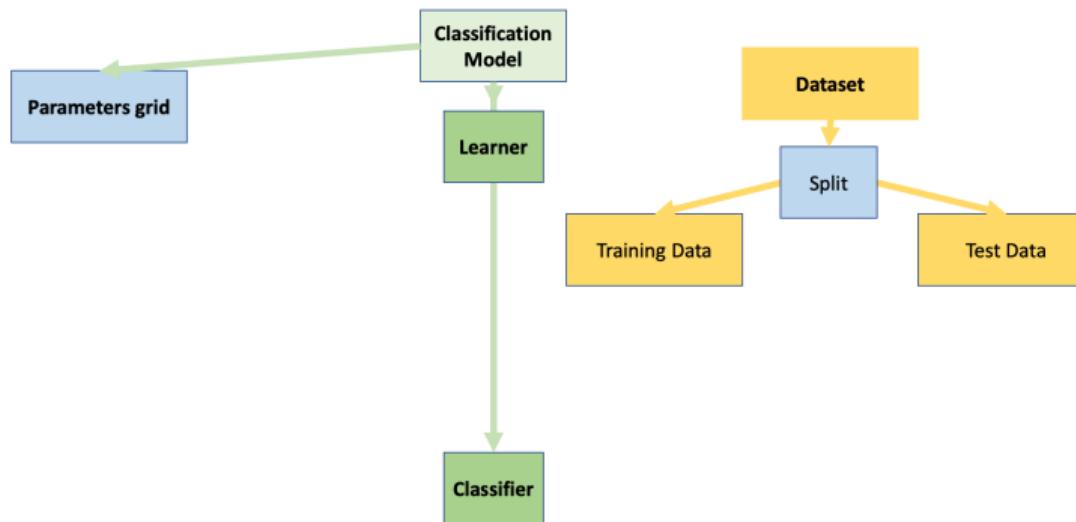
Cross-validation workflow



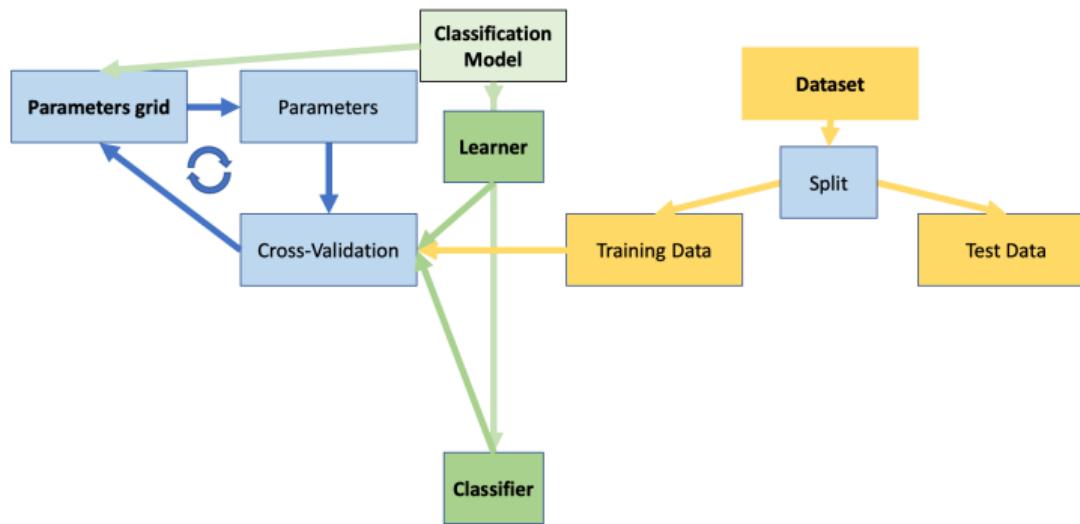
Cross-validation workflow



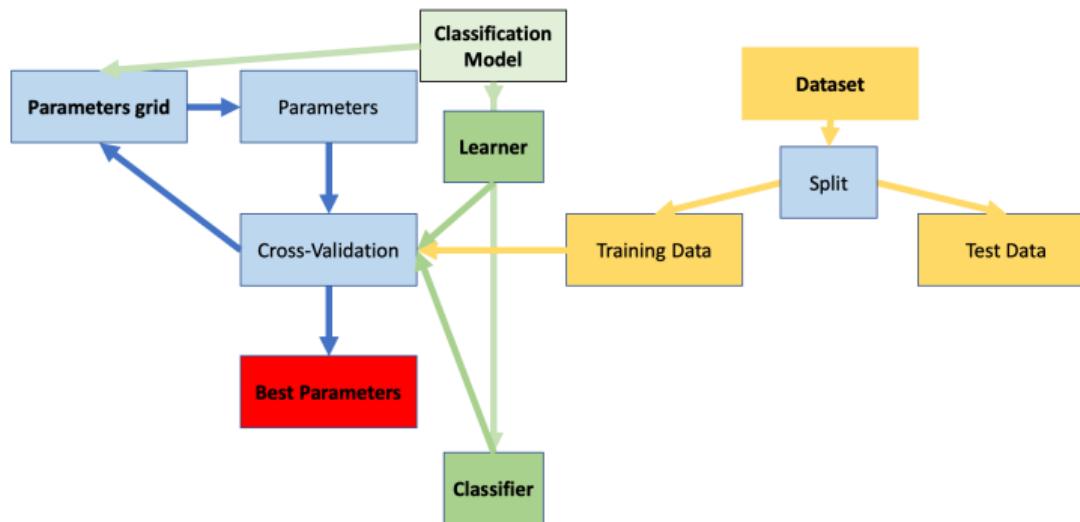
Cross-validation workflow



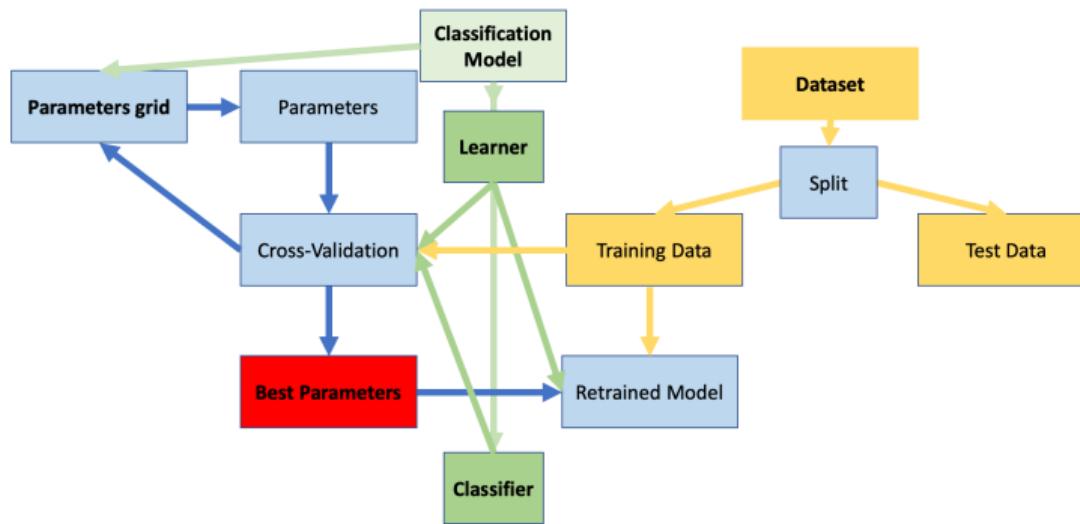
Cross-validation workflow



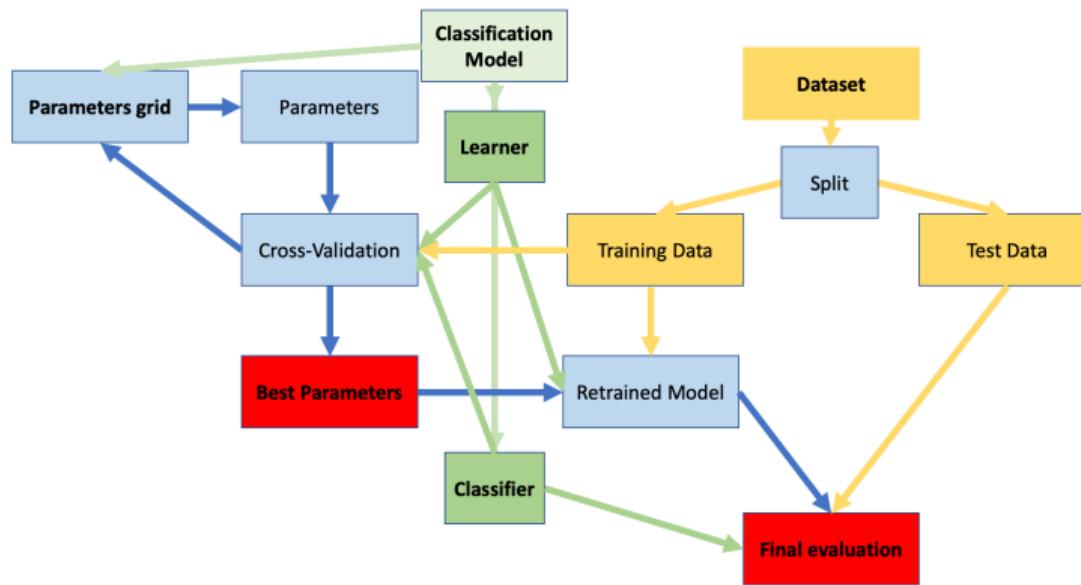
Cross-validation workflow



Cross-validation workflow



Cross-validation workflow



Cross Validation – pros and cons

- :(The train/test loop is repeated k times
- : The estimate of the performance is averaged on k runs
 ⇒ more reliability
- : All the examples are used once for testing
- : The final model is obtained using all the examples
 ⇒ best use of the examples

Leave one out

- Extreme case of cross validation, with $k = N$
- No random partitioning
- It is intrinsically non-stratified

Bootstrap

- A statistical sampling technique
- Sampling of N records **with replacement**
 - each record can be selected, even if it has been selected in previous samples
- Some records will never be selected: they will be used for test

$$e = 0.632e_{test} + 0.368e_{training}$$

$$\left(1 - \frac{1}{N}\right)^N \approx e^{-1} \approx 0.368$$

Train, Validation, Test – pros and cons

- 😊 The train/validation loop is faster than the Cross Validation
- 😊 The optimisation of the hyperparameters is done with the validation set, independent from the final evaluation
 - ⇒ more reliable than the simple holdout
- 😢 The test is done on a portion of the examples
 - ⇒ less reliable with respect to Cross Validation

Performance measures of a classifier

Binary prediction

For simplicity: Positive/Negative

- success rate = accuracy

$$\frac{TP + TN}{N_{test}}$$

- error rate

$$1 - \text{success rate}$$

		Predicted class	
		P	N
True class	P	TP	FN
	N	FP	TN

Accuracy is enough?

- Is the accuracy the only performance indicator for a classifier?
Other possible indicators:
 - Velocity
 - Robustness w.r.t. noise
 - i.e. training data with bad class label
 - Scalability
 - Interpretability
- A classification error can have different consequences, depending on the class of the individual
 - when forecasting an illness a false positive can be less dangerous than a false negative
 - unless the cares or the additional examinations are dangerous or invasive
 - consider the cost of retiring a machinery as damaged, while it is ok (false positive) and the cost of an unpredicted failure (false negative)

A summary of measures I

Precision – $TP/(TP + FP)$

the rate of true positives among the positive classifications

Recall – $TP/(TP + FN)$

the rate of the positives that I can catch (a.k.a.
Sensitivity)

Specificity – $TN/(TN + FP)$

the rate of the negatives that I can catch

A summary of measures II

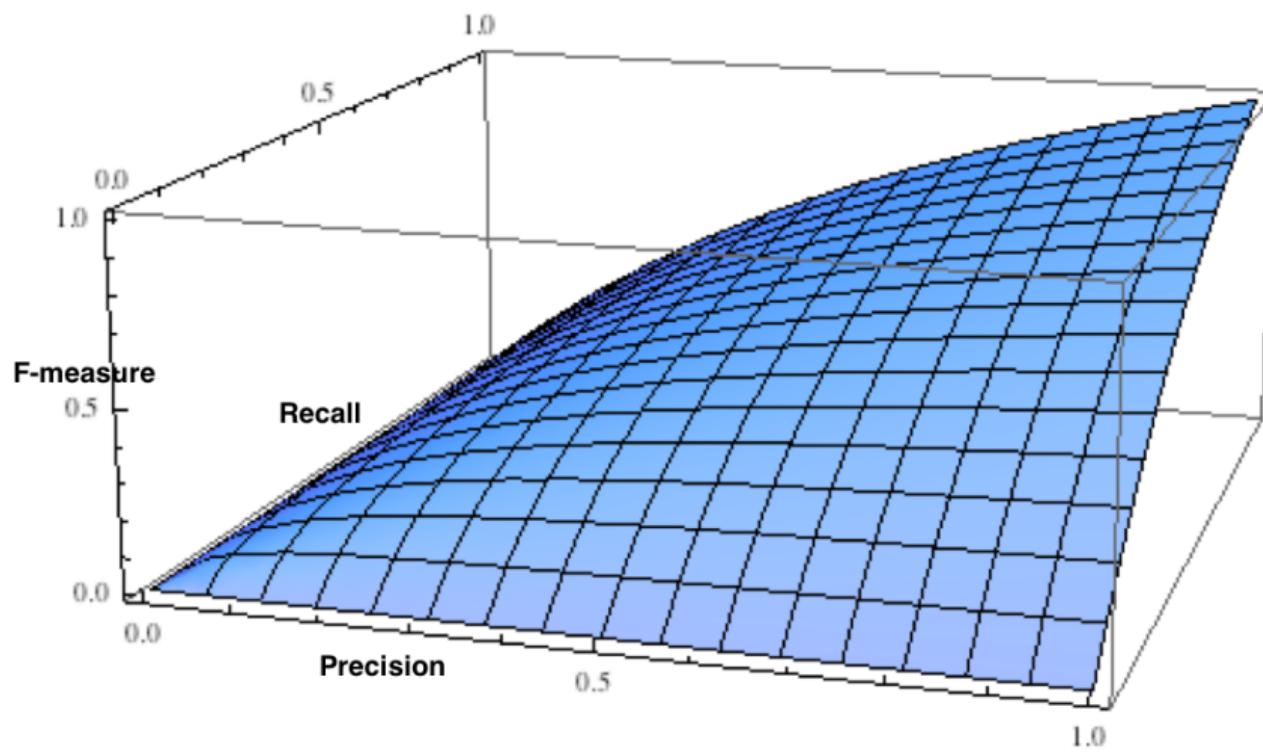
Accuracy – the weighted sum of sensitivity and specificity

$$acc = sens \frac{pos}{N} + spec \frac{neg}{N}$$

F-measure – the armonic mean of precision and recall, a.k.a. F1 score or balanced F-score

$$F = 2 \frac{\text{prec} \cdot \text{rec}}{\text{prec} + \text{rec}}$$

F-measure



Multi-dimensional case

- The table of page 38 is easily extended and is called **confusion matrix**
- Each cell contains the number of test records of class i and predicted as class j
- The numbers in the main diagonal are the “true” predictions

Beyond the accuracy

Taking into account the “a priori” information

- Is it likely to obtain a correct prediction *by chance*?
- Example: early diagnosis
 - let us consider a disease affecting 2% of patients
 - a prediction saying always “no disease” has 98% precision, which, in general would be a good result
 - the evaluation of a prediction should take this as a baseline, and, possibly, look for improvements

Confusion matrix with three classes, say a, b, c

T_x = true number of x labels in the dataset

P_x = total number of x predictions by a given classifier, say \bar{C}

TP_x = number of true predictions for class x given by classifier \bar{C}

FP_{i-j} = number of false prediction for class i predicted as j

$$\text{accuracy} = \frac{\sum_i TP_i}{N}$$

$$\text{precision}_i = \frac{TP_i}{P_i}$$

$$\text{recall}_i = \frac{TP_i}{T_i}$$

		Predicted class			
		a	b	c	Total
True class	a	TP_a	FP_{a-b}	FP_{a-c}	T_a
	b	FP_{b-a}	TP_b	FP_{b-c}	T_b
	c	FP_{c-a}	FP_{c-b}	TP_c	T_c
	Total	P_a	P_b	P_c	N

Example of confusion matrix

- Confusion matrix of classifier \bar{C} on a given dataset
- 140 correct predictions
- The predicted proportion of classes is 100:60:40

		Predicted class			
		a	b	c	Total
True class	a	88	14	18	120
	b	10	40	10	60
	c	2	6	12	20
	Total	100	60	40	200

Confusion matrix of a random classifier $R_{\bar{C}}$

A *virtual* experiment

- A random classifier $R_{\bar{C}}$ producing the same proportion of classes as \bar{C}
 - the horizontal *margin* is the same as \bar{C}
- The rows have all the same proportion as the horizontal margin
- 82 predictions are exact **by chance**
 - the sum of the main diagonal of $R_{\bar{C}}$

		Predicted class			
		a	b	c	Total
True class	a	60	36	24	120
	b	30	18	12	60
	c	10	6	4	20
	Total	100	60	40	200

Taking into account the random component

- The improvement of \bar{C} over $R_{\bar{C}}$ is $140 - 82 = 58$
- The improvement of the perfect classifier is $200 - 82 = 118$
- We define $\kappa(\bar{C})$ the improvement of the classifier at hand w.r.t. the improvement of the perfect classifier

$$\kappa(\bar{C}) = 58/118 = 0.492$$

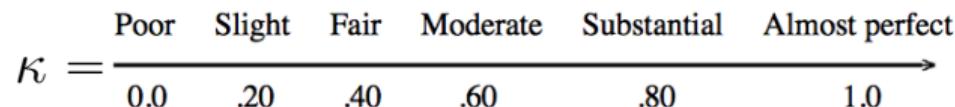
κ statistic [Cohen(1960)]

- Evaluates the concordance between two classifications
 - in our case between the predicted and the true one
- Probability of concordance $\Pr(c) = \frac{TP_a + TP_b + TP_c}{N}$
- Probability of random concordance $\Pr(r) = \frac{T_a * P_a + T_b * P_b + T_c * P_c}{N^2}$
- κ is the ratio between the concordance exceeding the random component and the maximum surplus possible

$$-1 \leq \kappa = \frac{\Pr(c) - \Pr(r)}{1 - \Pr(r)} \leq 1$$

Range of κ

- 1 for perfect agreement
 - $TP_a + TP_b + TP_c = N$
- -1 for total disagreement
 - $TP_a + TP_b + TP_c = 0$ and there is a perfect swap between predictions and true labels
 - if all classes have non-zero counts -1 is possible only if the number of labels is two
- 0 for random agreement



The cost of errors

- Our decisions are driven by predictions
- Bad predictions imply a **cost**
- Examples
 - grant a loan to a person who turns out to be a bad payer costs more than denying a loan to a person that could be a good payer
 - a false “oil spill” alarm is less expensive than an undetected spill
 - a wrong “fault prediction” in an industrial plant is in general less expensive than an unexpected fault disabling the plant and creating damages
 - in direct marketing, sending advertisement material without redemption is less harmful than the loss of business if a promising customer is ignored

Cost sensitive learning I

Weight the errors

- Alternative 1: alterate the proportion of classes in the supervised data, duplicating the examples for which the classification error is higher
 - In this way, the classifier will became **more able** to classify the classes for which the classification error cost is higher
 - This solution is useful also when the classes are *imbalanced*, that is the frequencies of the class labels in \mathcal{X} are not equal

Cost sensitive learning II

- Alternative 2: some learning schemes allow to add weights to the instances
 - e.g. the `DecisionTreeClassifier` of Scikit-Learn has the hyperparameter **class_weight**: it allows to define a dictionary, with one key per distinct class, specifying the relative weight to be assigned to each class, the optimisation of the fitting will be adjusted accordingly
 - the `balance` option balances the classes automatically

- 1 Model Selection: Evaluation of a classifier
- 2 Evaluation of a probabilistic classifier
 - Lift Chart
 - ROC Curve

2

54
57
59

Predicting probabilities of classes I

- Many classifiers produce, rather than a class label (*crisp* prediction), a tuple of probabilities, one for each possible class, (*probabilistic*, or *soft* prediction)
- The adequacy of one output or the other depends on the application domain
 - when an immediate decision is required the crisp output is necessary
 - when the classification is part of a process including several evaluation/action steps the probabilistic output can be more appropriated

Predicting probabilities of classes II

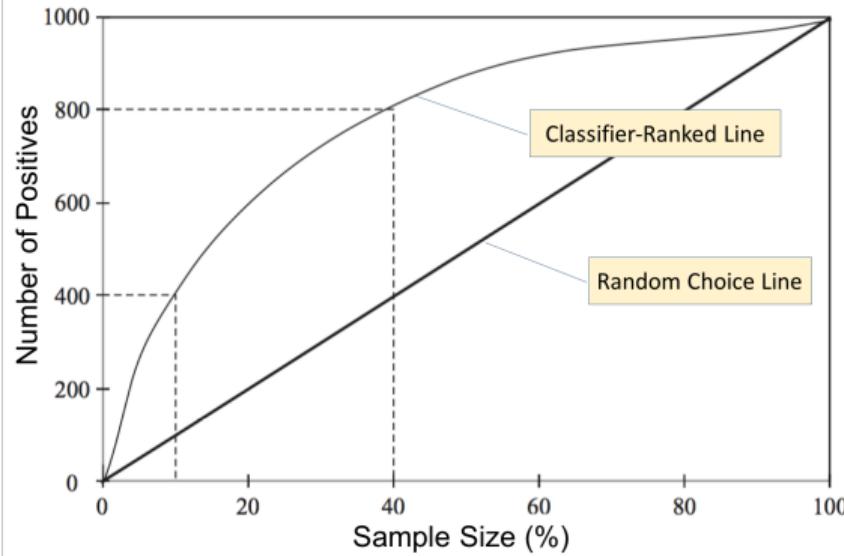
- Crisp values sometimes hide probabilities
 - e.g. when a leaf of a decision tree has non-zero counts for the minority classes a less-than-one probability can be assigned on the basis of the fraction of the elements belonging to the majority class
 - since in this case it is quite common to have leaves with a small number of examples and/or minority classes with frequencies near to zero, **smoothing** techniques are used to adjust the probabilities (e.g. smoothing)
- Probabilities can be converted to a crisp value with different techniques, depending on the number of classes (binary or multiclass)
 - *binary* – set a **threshold** for the positive class
 - *multiclass* – output the class with the **maximum probability**

Binary – Lift Chart

- Used to evaluate various scenarios, depending on the application
- Consider a dataset with 1000 positives and apply a probabilistic classification scheme
- Sort all the classified elements for decreasing probability of positive class
- Make a bi-dimensional chart with axes
 $x = \text{sample size}$, $y = \text{number of positives in sample}$
- Only the rank is important, not the specific probability

Lift Chart

- The straight line plots the number of positives obtained with a random choice of a sample of test data
- The curve plots the number of positives obtained drawing a fraction of test data with decreasing probability
- The larger the area between the two curves, the best the classification model



ROC Curve

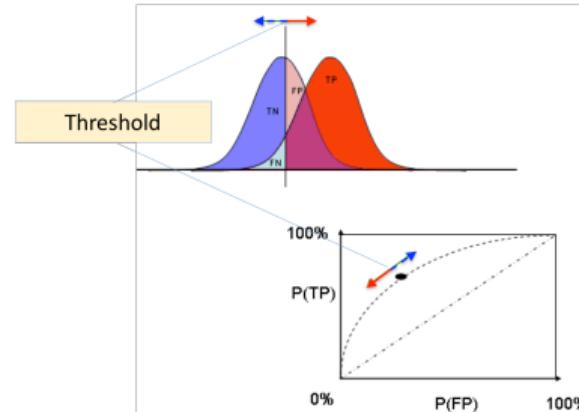
Receiver-Operator Characteristic

History: interpretation of radar signals during WWII

- Tradeoff between hit rate and false alarm in a noisy channel
- The noise can be such that the recognition of the transmission is altered
- The noise alters the two levels according to a gaussian distribution
- Problem: set the positive/negative threshold in order to maximize the tradeoff above, according to application-dependent requirements

ROC Curve

- With less noise the two gaussian curves are better separated
- Moving the threshold towards right increases both the rate of true positives and false positives caught
- The area between the non-discrimination line and the ROC curve is a quality index of the line
- The maximum area is the upper left triangle



a

a Image from Wikipedia

TN = blue + cyan = probability of a negative to be caught

FN = cyan = probability of a negative to be missed

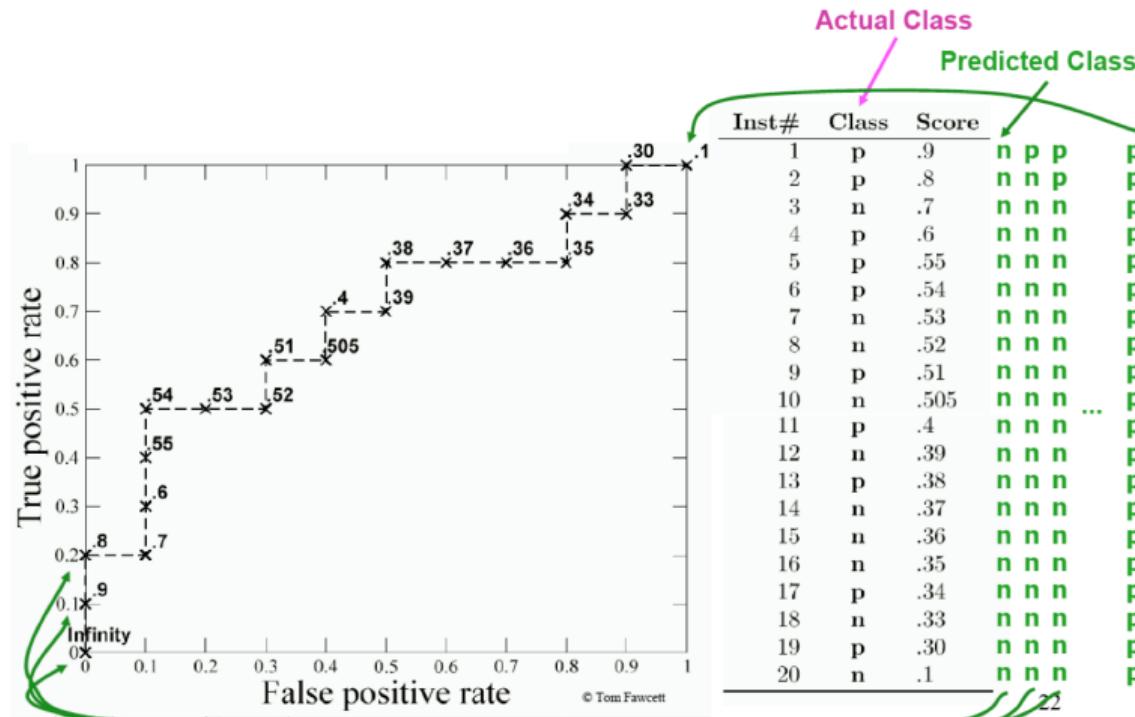
FP = pink + purple = probability of a positive to be missed

TP = red + purple = probability of a positive to be caught

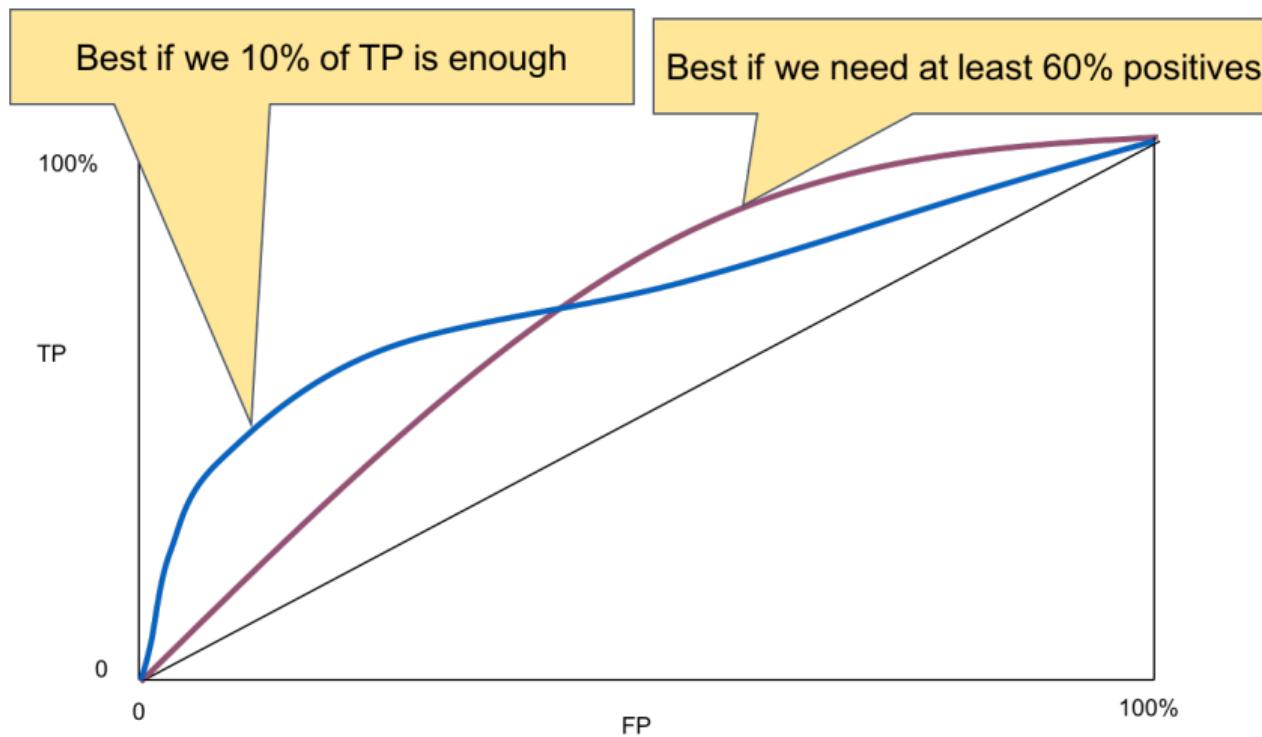
ROC for a soft classifier

- The soft classifier can be converted into a crisp one by choosing a **threshold** \Rightarrow predict *positive* if the probability of the test record exceeds the threshold
- Varying the threshold the behavior of the classifier changes, by changing the ratio of TP and FP
- Threshold steps allow to track the ROC curve
 - sort the test elements by decreasing positive probability
 - set the threshold to the highest probability, set TP and FP to zero
 - repeat
 - update the number of TP and FP with probability from the threshold to 1
 - draw a point in the curve
 - move to next top probability of positive
 - end repeat

Drawing the ROC curve for a soft classifier



Drawing the ROC curve for a soft classifier



Bibliography

- ▶ Jacob Cohen.
A coefficient of agreement for nominal scales.
Educational and Psychological Measurement, 20(1):37–46, 1960.
doi: 10.1177/001316446002000104.
URL <http://dx.doi.org/10.1177/001316446002000104>.
- ▶ Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani.
An Introduction to Statistical Learning.
Springer, 2015.

Machine Learning

Classification - Part III

Claudio Sartori

DISI

Department of Computer Science and Engineering – University of Bologna, Italy
claudio.sartori@unibo.it

1

Statistical modeling – Naive Bayes Classifier

2

- A fictitious example
- The Bayes method
- Missing values
- Numeric values
- Summary

4

6

10

11

16

2

Linear Classification with the Perceptron

17

3

Support Vector Machines

24

4

Neural Networks

42

5

Instance Learning - K Nearest Neighbours Classifier

66

6

From a binary classifier to multi-class classification

68

7

Ensemble methods

73

8

Forest of randomised trees

83

9

Boosting

87

10

Summary

Naive Bayes Classifier²

Main issues

- Based on statistics
 - In particular, on the Bayes' theorem
- Consider the contribution of all the attributes
- Assume that each attribute is **independent** from the others, *given the class*¹
 - This is a very strong assumption, rarely verified, but, nevertheless, the method works!
- Estimate the probabilities with the frequencies, as usual

1 This means

$$\Pr(d_1 = v_1, d_2 = v_2 \mid c = c_x) = \Pr(d_1 = v_1 \mid c = c_x) \cdot \Pr(d_2 = v_2 \mid c = c_x)$$

2 Description based on [Witten et al.(2011)][Witten, Frank, and Hall](#)

The weather/play data

Numbers of cases and fractions of the Weather/Play dataset

Outlook		Temperature		Humidity		Windy		Play			
	yes	no	yes	no	yes	no	yes	no	yes	no	
sunny	2	3	hot	2	2	high	3	4	false	6	2
overcast	4	0	mild	4	2	normal	6	1	true	3	3
rainy	3	2	cool	3	1						
sunny	2/9	3/5	hot	2/9	2/5	high	3/9	4/5	false	6/9	2/5
overcast	4/9	0/5	mild	4/9	2/5	normal	6/9	1/5	true	3/9	3/5
rainy	3/9	2/5	cool	3/9	1/5						

A new sample needs classification

Outlook: sunny, Temperature: cool, Humidity: high, Windy: true, Play: ?

- Treat the five features and the overall likelihood that *play* is yes or no as equally important

- they are independent pieces of evidence, the overall likelihood is obtained by multiplying the probabilities (i.e. the frequencies)

$$\text{likelihood of yes} = 2/9 \times 3/9 \times 3/9 \times 3/9 \times 9/14 = 0.0053$$

$$\text{likelihood of no} = 3/5 \times 1/5 \times 4/5 \times 3/5 \times 5/14 = 0.0206$$

- Normalize to 1

$$\Pr(\text{yes}) = \frac{0.0053}{0.0053 + 0.0206} = 20.5\%$$

$$\Pr(\text{no}) = \frac{0.0206}{0.0053 + 0.0206} = 79.5\%$$

- *no* is more likely than *yes*, about four times

The Bayes' theorem

Given a hypothesis H and an evidence E that bears on that hypothesis

$$\Pr(H|E) = \frac{\Pr(E|H)\Pr(H)}{\Pr(E)}$$

- The hypothesis is the class, say c , the evidence is the tuple of values of the element to be classified
- We can split the evidence into pieces, one per attribute, and, if the attributes are **independent** inside each class,

$$\Pr(c|E) = \frac{\Pr(E_1|c) \times \Pr(E_2|c) \times \Pr(E_3|c) \times \Pr(E_4|c) \times \Pr(c)}{\Pr(E)}$$

The Naive Bayes method

- Compute the conditional probabilities from examples
- Apply the theorem
- The denominator is the same for all the classes, and is eliminated by the normalization step
- It is called *naive* since the assumption of independence between attributes is quite simplistic
 - Nevertheless it works quite well in many cases

Problem

What if value v of attribute d never appears in the elements of class c ?

- In this case $\Pr(d = v | c) = 0$
- This makes the probability of the class for that evidence drop to zero
- In practice, this case is quite common, in particular in a domain with many attributes and many distinct values
- An alternative solution is needed

Values not represented in a class – Laplace smoothing

α – Smoothing parameter, typical value is 1

$af_{d=v_i,c}$ – **Absolute** frequency of value v_i in attribute d over class c

V – number of distinct values in attribute d over the dataset

af_c – **Absolute** frequency of class c in the dataset

$$\text{Smoothed frequency } sf_{d=v_i,c} = \frac{af_{d=v_i,c} + \alpha}{af_c + \alpha V}$$

- With $\alpha = 0$ we obtain the standard, unsmoothed formula
- Higher values of α give more importance to the prior probabilities for the values of d w.r.t. the evidence given by the examples

Missing values

They do not affect the model, it is not necessary to discard an instance with missing value(s)

- Test instance:
 - The calculation of the likelihood simply omits this attribute
 - The likelihood will be higher for all the classes, but this is compensated by the normalization
- Train instance:
 - The record is simply not included in the frequency counts for that attribute
 - The descriptive statistics are based on the number of values that occur, rather than on the number of instances

Numeric values

- The method based on frequencies is inapplicable
- Additional assumption: the values have a *Gaussian* distribution
- Instead of the fractions of counts, we compute, from the examples the mean μ and the variance σ of the values of each numeric attribute **inside each class**
- For a given attribute and a given class, the distribution is supposed to be

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

The weather/play data with numeric values

Numbers of cases, fractions and descriptive statistics of the Weather/Play dataset

Outlook		Temperature		Humidity		Windy		Play			
	yes	no	yes	no	yes	no	yes	no	yes	no	
sunny	2	3	83	85	86	85	false	6	2	9	5
overcast	4	0	70	80	96	90	true	3	3		
	rainy	3	2	68	65	80	70				
			64	72	65	95					
			69	71	70	91					
			75		80						
			75		70						
			72		90						
			81		75						
sunny	2/9	3/5	mean	73	74.6	mean	79.1	86.2	false	6/9	2/5
overcast	4/9	0/5	stdev	6.2	7.9	stdev	10.2	9.7	true	3/9	3/5
rainy	3/9	2/5									

Using numeric values in Naive Bayes

- We are considering a yes outcome when the temperature is 66
- Plug the the value under consideration, the mean and the stdev in the gaussian probability density formula

$$f(\text{temperature} = 66 | \text{yes}) = \frac{1}{\sqrt{2\pi} \cdot 6.2} e^{-\frac{(66-73)^2}{2 \cdot 6.2^2}} = 0.0340$$

$$f(\text{humidity} = 90 | \text{yes}) = 0.0221$$

Using numeric values in Naive Bayes

Probability and probability density are closely related, but are not the same thing

- On a continuous domain, the probability of a variable assuming *exactly* a single real value is zero
- A value of the density function is the probability that the variable lies in a small interval around that value
- The value we use are, of course, rounded at some precision factor
- That precision factor is the same for all the classes, then we can disregard it
- If numeric values are missing, mean and standard deviation are based only on the values that are present

Classification of a sample with numeric values

Outlook: sunny, Temperature: 66, Humidity: 90, Windy: true, Play: ?

$$\text{likelihood of yes} = 2/9 \times 0.0340 \times 0.0221 \times 3/9 \times 9/14 = 0.000036$$

$$\text{likelihood of no} = 3/5 \times 0.0221 \times 0.0381 \times 3/5 \times 5/14 = 0.000108$$

$$\Pr(\text{yes}) = \frac{0.000036}{0.000036 + 0.000108} = 25.0\%$$

$$\Pr(\text{no}) = \frac{0.000108}{0.000036 + 0.000108} = 75.0\%$$

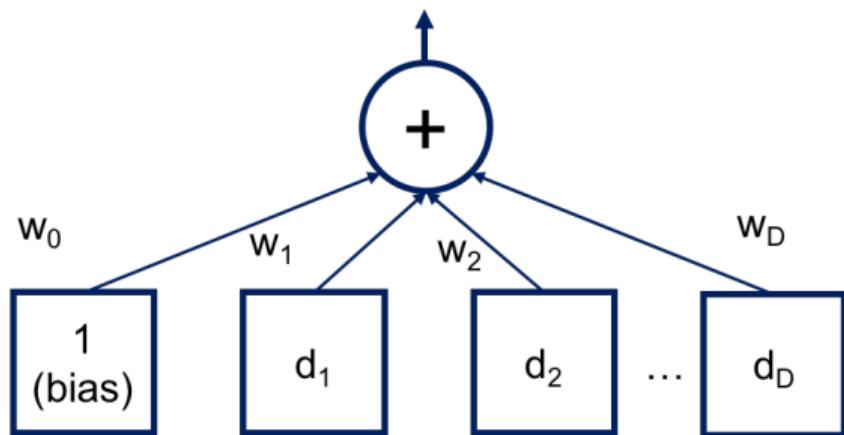
Summary

- Clear semantics for learning probabilistic knowledge
- Excellent results in many cases
- Dramatic degradation when the simplistic conditions are not met
 - Violation of *independence* – for instance, if an attribute is simply a copy of another (or a linear transformation), the weight of that particular feature is enforced (something like squaring the probability)
 - Violation of *gaussian* distribution – use the standard probability estimation for the appropriate distribution, if known, or use estimation procedures, such as *Kernel Density Estimation*

1	Statistical modeling – Naive Bayes Classifier	2
2	Linear Classification with the Perceptron	17
	● Training the perceptron	21
	● Convergence	22
3	Support Vector Machines	24
4	Neural Networks	42
5	Instance Learning - K Nearest Neighbours Classifier	66
6	From a binary classifier to multi-class classification	68
7	Ensemble methods	73
8	Forest of randomised trees	83
9	Boosting	87
10	Summary	92

The linear perceptron³

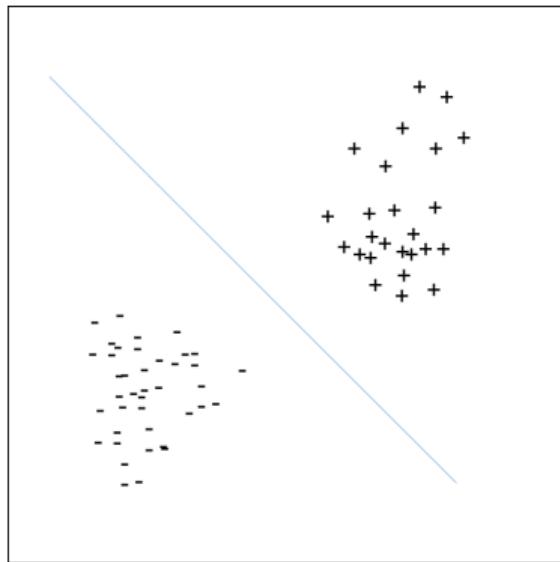
- Often called also **artificial neuron**
- In practice, a linear combination of weighted inputs



3 Description based on [Witten et al.(2011)] Witten, Frank, and Hall]

Separate examples of two classes

- For a dataset with numeric attributes
- Learn a *hyperplane* such that all the positives lay on one side and all the negatives on the other



The hyperplane

- The hyperplane is described by a set of weights w_0, \dots, w_D in a linear equation on the data attributes x_0, \dots, x_D
 - the fictitious attribute $x_0 = 1$ is added to allow a hyperplane that does not pass through the origin
- There are either **none** or **infinite** such hyperplanes

$$w_0 * x_0 + w_1 * x_1 + \dots + w_D * x_D \quad \begin{cases} > 0 \Rightarrow \text{positive} \\ < 0 \Rightarrow \text{negative} \end{cases}$$

Learning the hyperplane

set all weights to zero

while there are examples incorrectly classified **do**

for each training instance x **do**

if x is incorrectly classified **then**

if class of x is positive **then**

 add the x data vector to the vector of weights

else

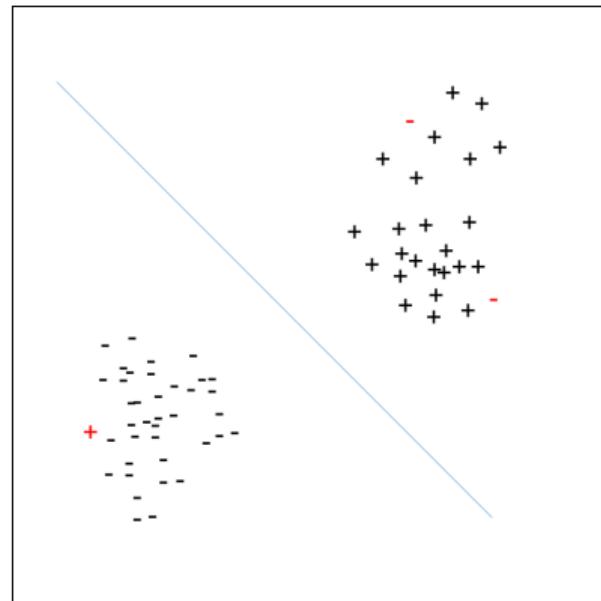
 subtract the x data vector from the vector of weights

Linear perceptron convergence

- Each change of weights moves the hyperplane towards the misclassified instance, consider the equation after the weight change for a positive instance x which was classified as negative
$$(w_0 + x_0) * x_0 + (w_1 + x_1) * x_1 + \dots + (w_D + x_D) * x_D$$
- The result of the equation is increased by a **positive** amount
$$x_0^2 + \dots + x_D^2$$
- Therefore the result will be **less negative** or, possibly, even **positive**
- Analogously for a negative instance which was classified as positive

Linear perceptron algorithm termination

- The corrections are incremental and can interfere with previous updates
- The algorithm converges **if the dataset is linearly separable**, otherwise it does not terminate
- For practical applicability it is necessary to set an upper bound to the iterations



1	Statistical modeling – Naive Bayes Classifier	2
2	Linear Classification with the Perceptron	17
3	Support Vector Machines	24
	● The Maximum Margin hyperplane	28
	● Non-linear class boundaries	34
4	Neural Networks	42
5	Instance Learning - K Nearest Neighbours Classifier	66
6	From a binary classifier to multi-class classification	68
7	Ensemble methods	73
8	Forest of randomised trees	83
9	Boosting	87
10	Summary	92

Support Vector Machines (SVM) for binary classification I

Limitations of the linear models

Description of SVM based on [Witten et al.(2011) Witten, Frank, and Hall]

- What can we do if the data are not *linearly separable*?
 - This means that the boundary between classes is some hyper-surface more complex than a hyperplane
- One possibility would be to give up the linearity, e.g.:

$$w_1 * x_1^3 + w_2 * x_1^2 * x_2 + w_3 * x_1 * x_2^2 + w_4 * x_3^3$$

Support Vector Machines (SVM) for binary classification II

Limitations of the linear models

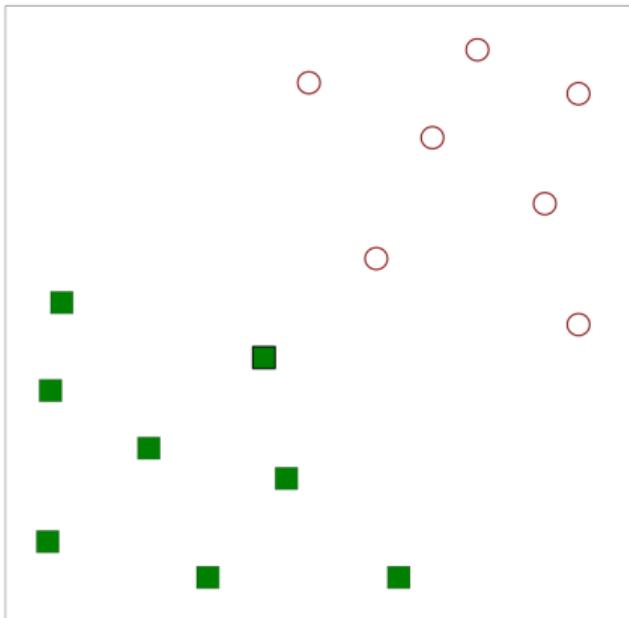
Description of SVM based on [Witten et al.(2011) Witten, Frank, and Hall]

- The method would become soon intractable for any reasonable number of variables
 - with 10 variables and limiting to factors with maximum order 5 we would need something like 2000 coefficient
- The method would be extremely prone to overfitting, if the number of parameters approaches the number of examples

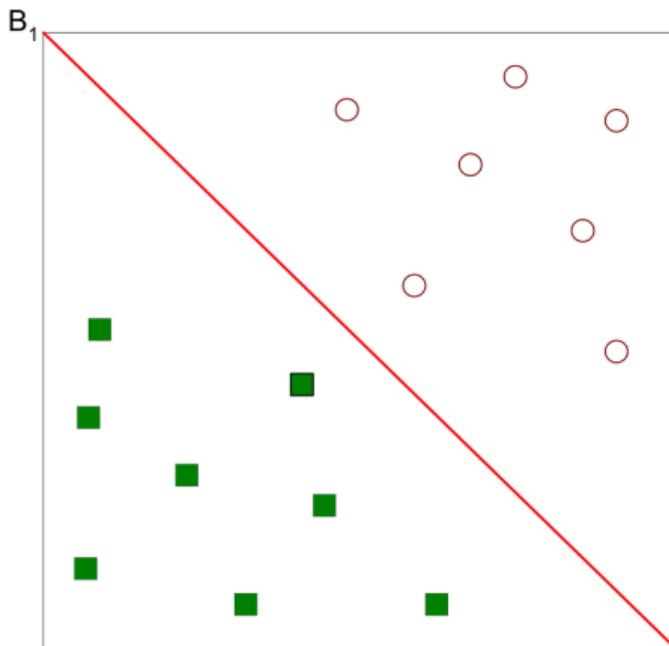
Key ideas

- Computational learning theory
- New efficient separability of non-linear functions that use **kernel functions**
- Optimization rather than greedy search
- Statistical learning
 - The search of a prediction function is modeled as a **function estimation** problem

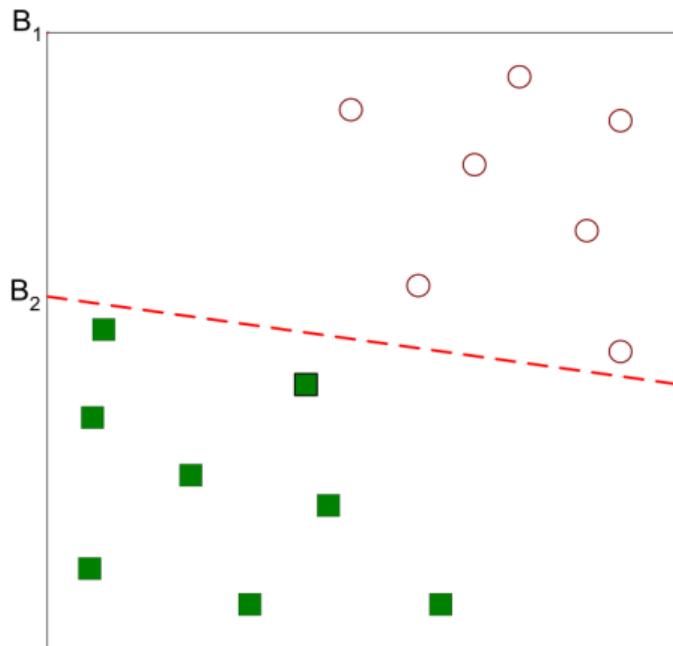
The Maximum Margin hyperplane



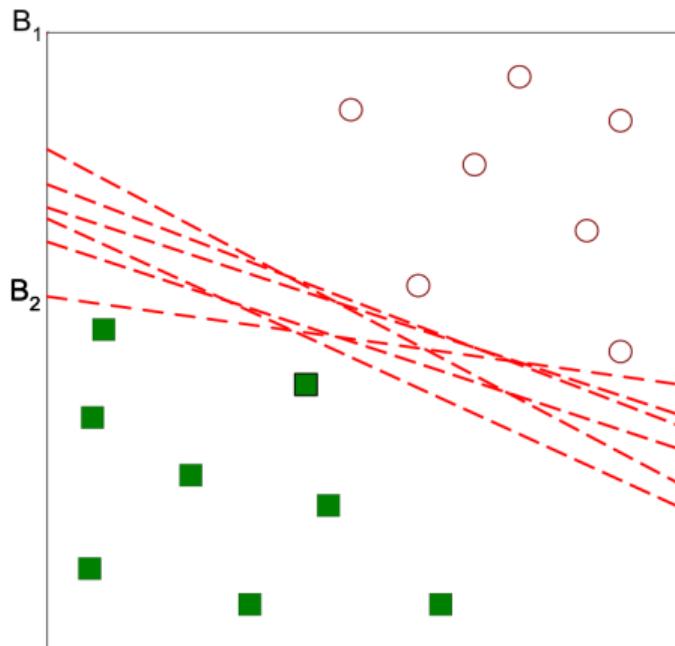
The Maximum Margin hyperplane



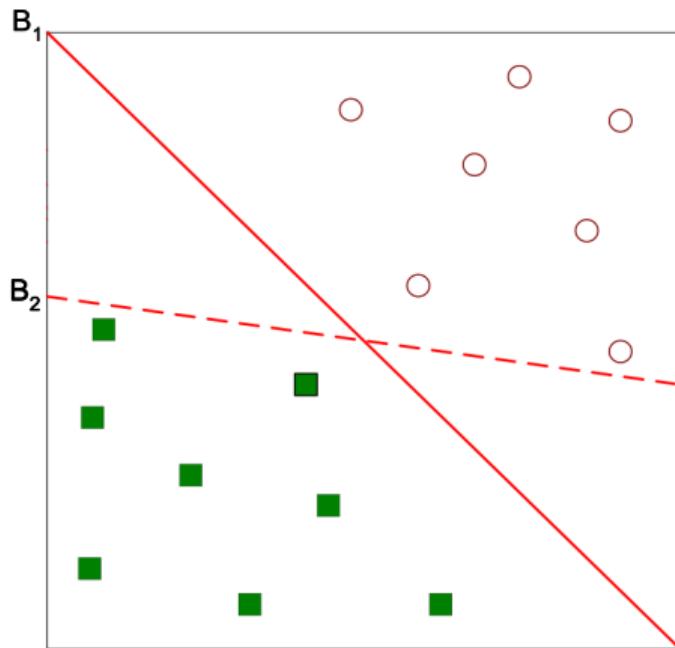
The Maximum Margin hyperplane



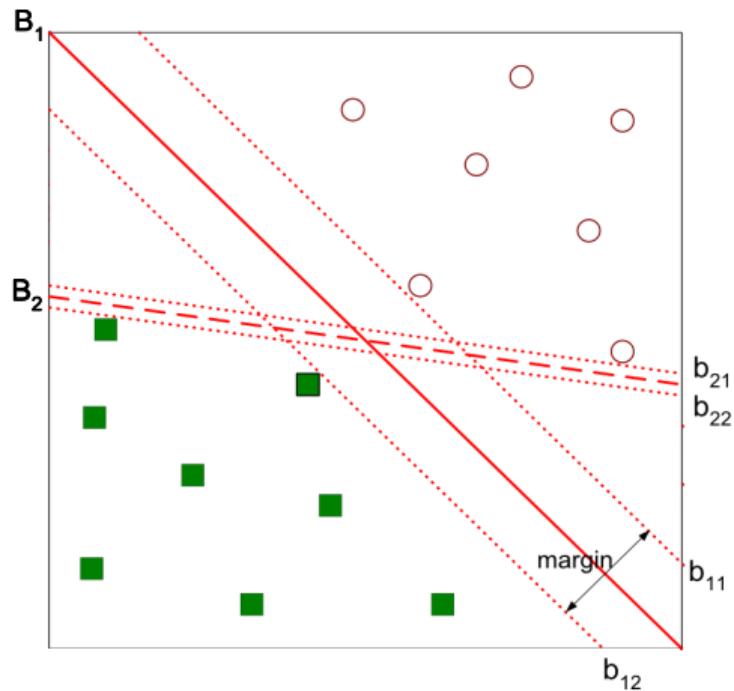
The Maximum Margin hyperplane



The Maximum Margin hyperplane

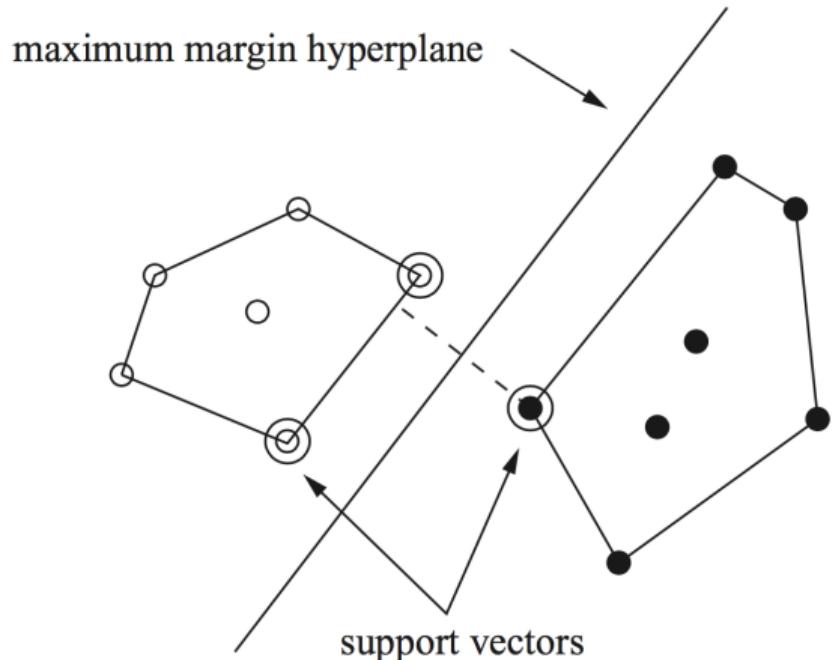


The Maximum Margin hyperplane



Maximum Margin Hyperplane I

- The linear perceptron accepts **any** hyperplane able to separate the classes of the training examples
 - It is conceivable that some hyperplanes are better than others for the classification of **new** items
- The **maximum margin hyperplane** gives the greatest separation between the classes



Maximum Margin Hyperplane II

- The *convex hull* of a set of points is the tightest enclosing convex polygon
 - if the dataset is linearly separable the convex hulls of the classes do not intersect
- The maximum margin hyperplane is as far as possible from both the hulls
 - it is the perpendicular bisector of the shortest line connecting the hulls
- In general a subset of the points is sufficient to define the hull
 - those are the **support vectors** (circled in the figure of the previous page)
- Support vectors are the elements of the training set which **would change**

Maximum Margin Hyperplane III

- Finding the support vectors and the maximum margin hyperplane belongs to the well known class of *constrained quadratic optimization* problems

$$\max_{w_0, w_1, \dots, w_D} M$$

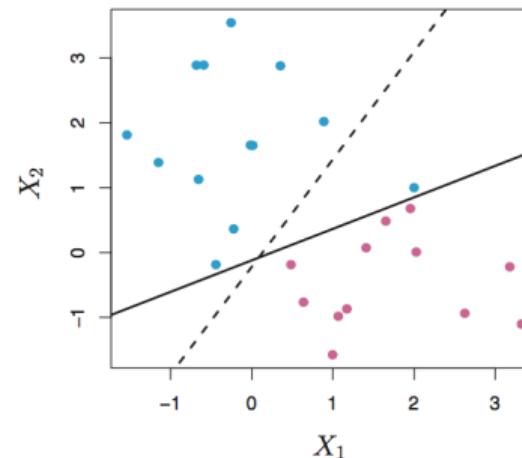
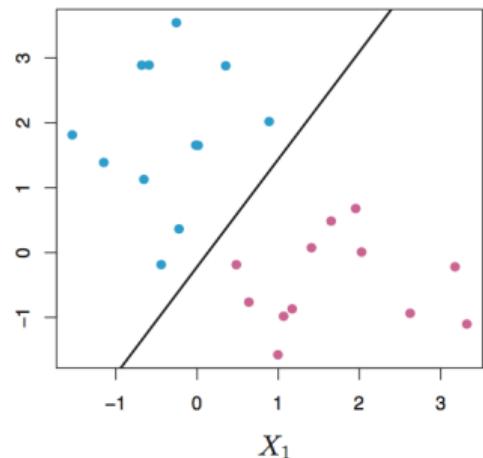
$$\text{subject to } \sum_{j=1}^D w_j^2 = 1$$

$$c_i(w_0 + w_1x_{i1} + \dots + w_Dx_{iD}) > M, \forall i = 1, \dots, N$$

where the class of example i is either -1 or 1 and M is the **margin**

Soft margin

- It is quite common that a separating hyperplane does not exists
 - see figure in slide 23 in the perceptron section
- \Rightarrow find an hyperplane which **almost** separate the classes
- \Rightarrow disregard examples which generate a very narrow margin



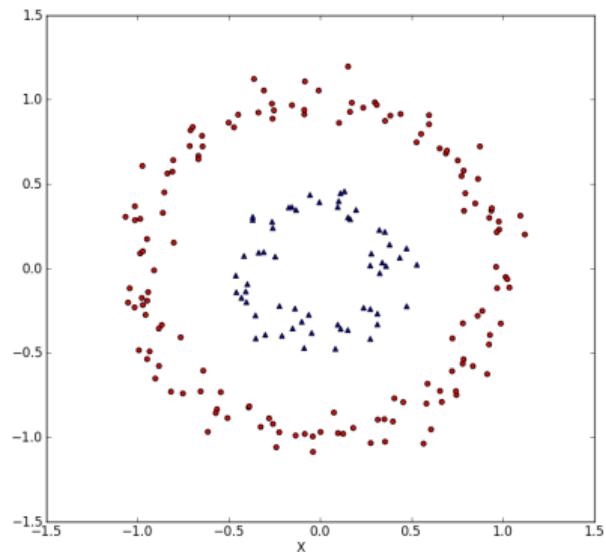
Soft margin Support Vector classifier

- Greater robustness to individual observations
- Better classification of **most** of the training observations
- Obtained by adding a constraint to the optimization problem expressed by a single numeric parameter, usually called C in the literature
 - C , the penalty parameter of the error term, controls the amount of overfitting
 - C tuning is critical

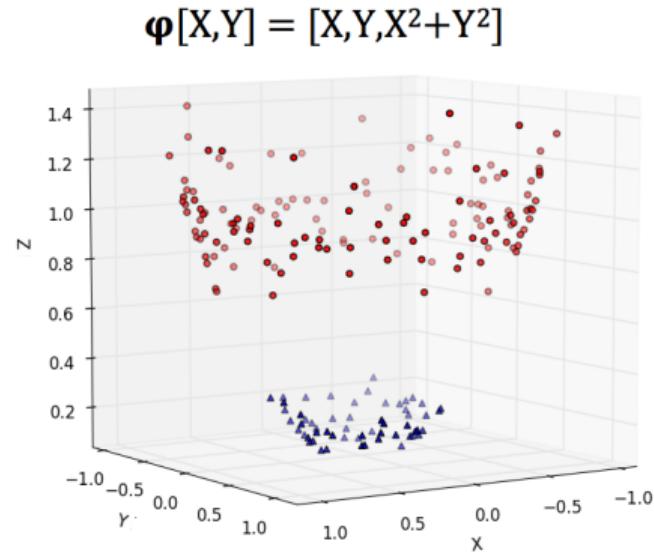
Non-linear class boundaries I

- The support vector method avoids the problem of overfitting
- The nonlinearity of boundaries can be overcome with a **non-linear mapping**
 - the data are mapped into a new space, usually called *feature space*, such that a linear boundary in the feature space can correspond to a non-linear boundary in the original space
 - the feature space can have a number of dimensions higher than the original one

Non-linear class boundaries II

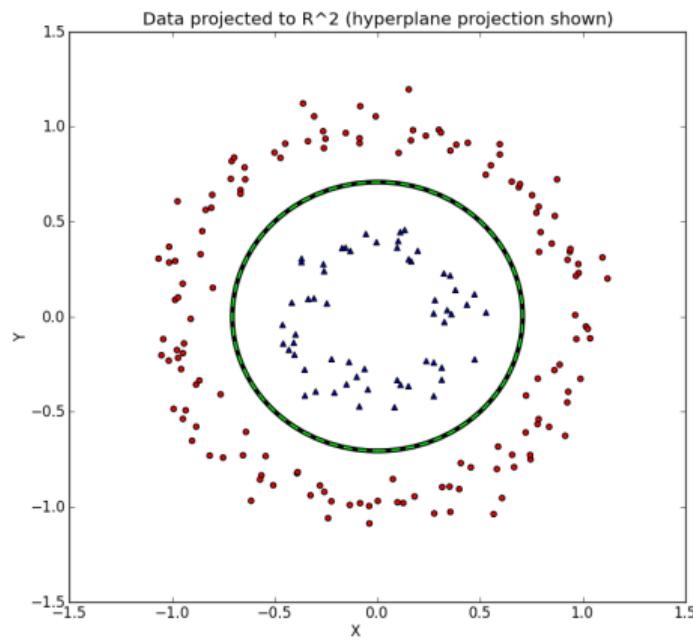
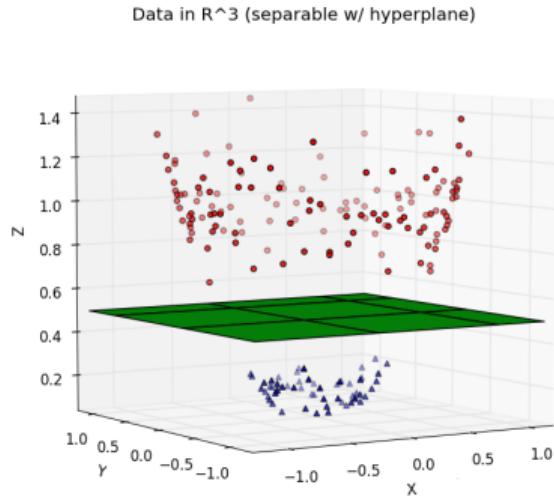


Input space



Feature space

Non-linear class boundaries III



Now linearly separable

The *kernel trick*

- The separating hyperplane computation requires a series of **dot product** computations among the training data vectors
- Defining the mapping on the basis of a particular family of functions, called **kernel functions**, or simply **kernels**, the mapping does not need to be explicitly computed, and the computation is done in the input space
- This avoids an increase in the complexity
- Some kernel functions:

linear $\langle x, x' \rangle$

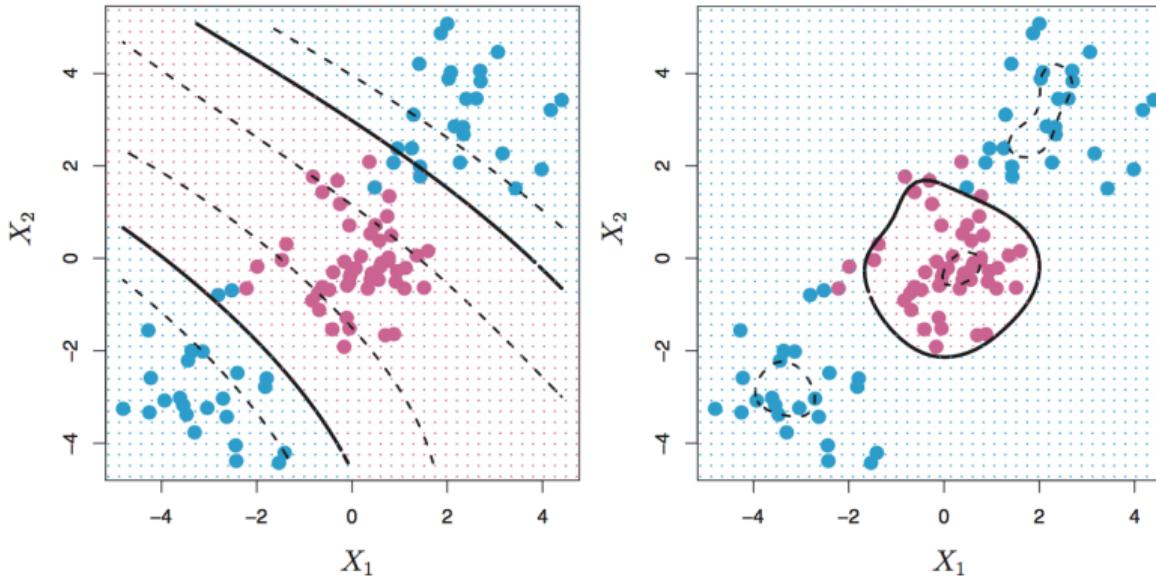
polynomial $(\gamma \langle x, x' \rangle + r)^{dg}$

rbf $\exp(-\gamma \|x - x'\|^2)$

sigmoid $\tanh(\langle x, x' \rangle + r)$

- γ, dg, r are parameters specified in the documentation of the learning tools
- Rule of thumb: start with the simpler and then try with the more complex if necessary

Examples of decision boundaries with kernels



Decision boundaries for polynomial kernel of degree (left) and radial based kernel (RBF, right)
In this case they seem to be both appropriate, but the generalization capabilities change,
depending on where new data could be expected

SVM Complexity

- The time complexity is mainly influenced by the efficiency of the optimization library
- The popular libSVM library scales from $\mathcal{O}(D * N^2)$ to $\mathcal{O}(D * N^3)$, depending on the effectiveness of data caching in the library, which is **data dependent**
 - in case of sparse data it is reduced

Final remarks on SVM

- Learning is in general slower than simpler methods, such as decision trees
- Tuning is necessary to set the parameters (not discussed here)
- The results can be very accurate, because subtle and complex decision boundaries can be obtained
- Explicitly based on a theoretical model of learning
- Are not affected by local minima
- Do not suffer from the curse of dimensionality: do not use any notion of **distance**
- SVMs do not directly provide probability estimates, these can be calculated using rather expensive estimation techniques (see scikit-learn documentation in References below)
 - nevertheless, SVM can produce a **confidence score** related to the distance of an example from the separation hyperplane

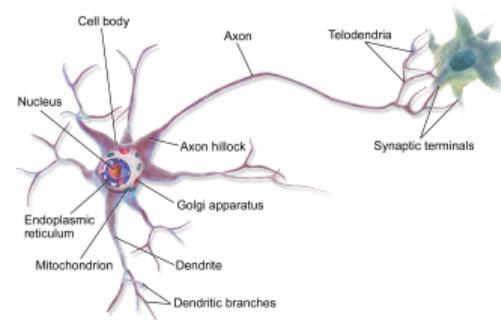
SVM - References

1. See reference [James et al.(2015)James, Witten, Hastie, and Tibshirani] for a more detailed explanation of the maths behind SVM (unfortunately it is not publicly available, ask the teacher for a temporary loan)
2. Jordan, Michael I., and Romain Thibaux. "The Kernel Trick." Lecture Notes. 2004. Web. 5 Jan. 2013.
<https://people.eecs.berkeley.edu/~jordan/courses/281B-spring04/lectures/lec3.pdf>
3. Berwick, Robert. "An Idiot's Guide to Support Vector Machines (SVMs)". Lecture Slides. 2003. Web. 5 Jan. 2013. <http://www.cs.ucf.edu/courses/cap6412/fall2009/papers/Berwick2003.pdf>
4. "Scikit-learn: sklearn.svm.SVC Documentation", Pedregosa et al.
<http://scikit-learn.org/dev/modules/generated/sklearn.svm.SVC.html>
5. Rifkin, Ryan. "Multiclass Classification". Lecture Slides. February 2008. Web. 6 Jan. 2013.
<http://www.mit.edu/ 9.520/spring09/Classes/multiclass.pdf>
6. Hofmann, Martin. "Support Vector Machines – Kernels and the Kernel Trick". Notes. 26 June 2006. Web. 7 Jan. 2013.
http://www.cogsys.wiai.uni-bamberg.de/teaching/ss06/hs_svm/slides/SVM_Seminarbericht_Hofmann.pdf

1	Statistical modeling – Naive Bayes Classifier	2
2	Linear Classification with the Perceptron	17
3	Support Vector Machines	24
4	Neural Networks	42
	● Multi-layer perceptron	44
	● Training the Neural Network	50
	● Computing weight corrections	52
	● Training algorithm	56
	● Example	63
5	Instance Learning - K Nearest Neighbours Classifier	66
6	From a binary classifier to multi-class classification	68
7	Ensemble methods	73
8	Forest of randomised trees	83
9	Boosting	87
10	Summary	92

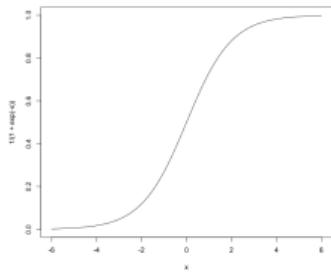
Neural Networks

- Arrange many perceptron-like elements in a hierarchical structure
 - Another way to overcome the limit of linear decision boundary
- Inspired to the complex interconnections of neurons in animal brains
- A neuron is a signal processor with **threshold**
- Signal transmission from one neuron to another is **weighted**
 - weights change over time, also due to **learning**

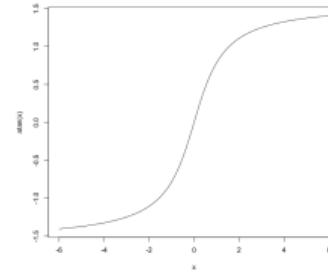


Multi-layer perceptron

- The signals transmitted are modeled as real numbers
- The threshold of the biological system is modeled as a mathematic function
 - continuous and differentiable, superiorly and inferiorly limited
 - the derivative can be expressed in terms of the function itself
 - this simplifies the mathematics
- Several functions available



Sigmoid

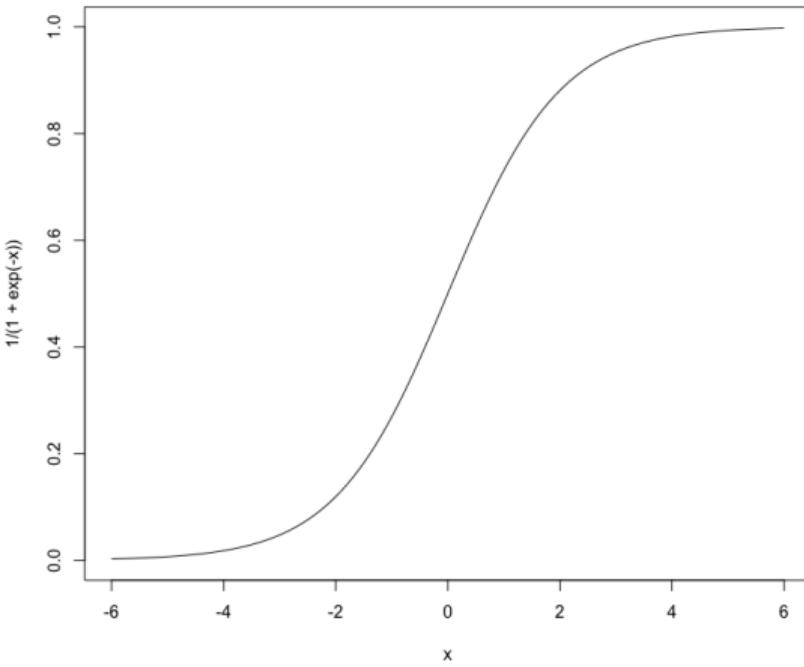


Arctangent

Sigmoid

- Also called *squashing function*
- Maps reals into $]0, 1[$
- Is continuous, differentiable, non-linear

$$\frac{1}{1 + e^{-x}}$$



Importance of non-linearity

- Results with the linear perceptron were non satisfactory because of linearity
 - in addition to the problem of separability
- In a linear system $f(x_1 + x_2) = f(x_1) + f(x_2)$
 - if x_2 is generated by noise, it is completely transferred to the output
- In a non-linear system, in general, $f(x_1 + x_2) \neq f(x_1) + f(x_2)$
- The shape of the function can influence the learning speed

Feed-forward multi-layered network

- Inputs feed an **input layer**
 - one input node for each dimension in the training set
- Input layer feeds (with weights) a **hidden layer**
- Hidden layer feeds (with weights) an **output layer**
 - the number of nodes in the hidden layer are a parameter of the network
 - the number of nodes in the output layer is related to number of different classes in the domain
 - one node if there are two classes
 - one node per class in the other cases⁴

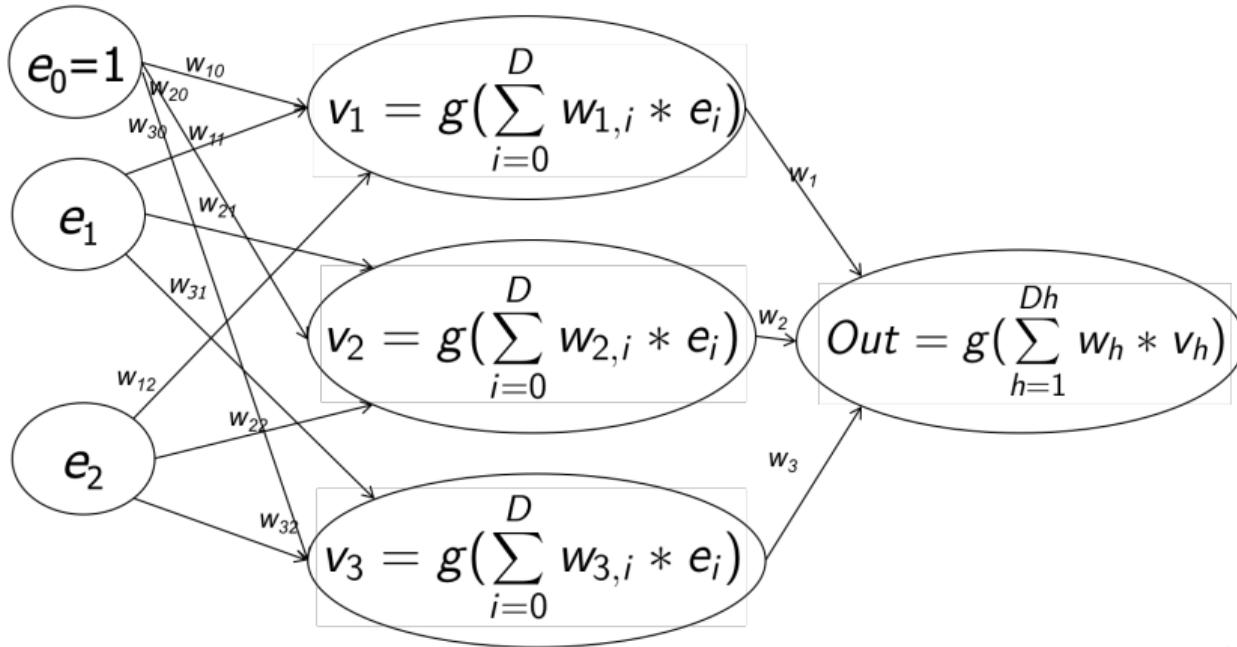
⁴ As an alternative, it is possible to adopt the OVO or OVA architecture, see page 69

Feed-forward multi-layered network – Example

$$D_{in} = D = 2$$

$$D_h = 3$$

$$D_{Out} = 1 \text{ (binary)}$$



1

Details

- $g(\cdot)$ is the transfer function of the node, e.g. the sigmoid
- The unitary input x_0 is added for dealing with the bias, as in the case of the linear perceptron
- The weights are for each edge connecting two nodes
- **Feed-forward** defines which oriented edges are present
 - edges connect only a node in a layer to a node in the following layer
 - input to hidden and hidden to output
 - each node of one layer is connected to all the nodes of the following layer
- In this way the signal flows from the input to the output, without loops

Training the Neural Network I

set all weights to random values

while termination condition is not satisfied **do**

for each training instance x **do**

 feed the network with x and compute the output $nn(x)$

 compute the weight corrections for $nn(x) - x_{out}$

 propagate back the weight corrections

Training the Neural Network II

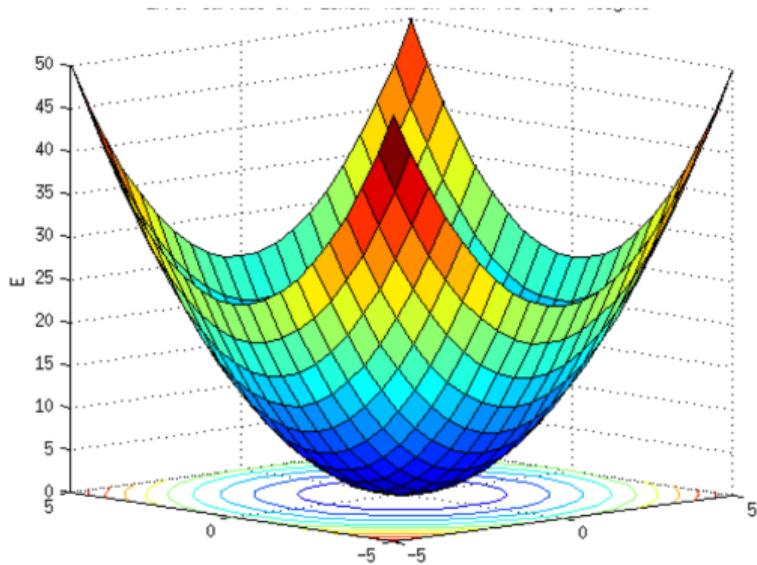
- In analogy with learning in the animal domain, the examples must repeatedly feed the network
- The weights **encode** the knowledge given by the supervised examples
- The encoding is not easily understandable: it looks like a structured set of real numbers
- Convergence is not guaranteed
- Important issues:
 - computing the weight corrections
 - preparation of the training examples
 - standardize the attributes to have zero mean and unit variance
 - termination condition

Computing the error

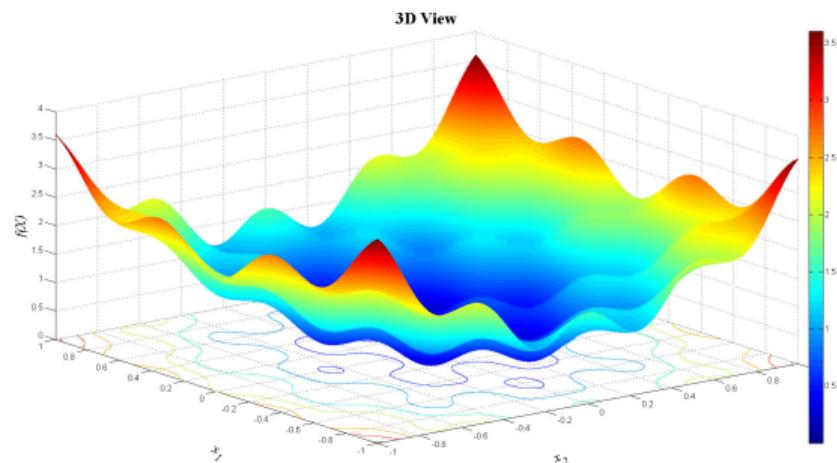
- Let \mathbf{x} and y be the input vector and the desired output of a node, respectively
- Let \mathbf{w} be the input weight vector of a node
- The error is:

$$E(\mathbf{w}) = \frac{1}{2}(y - \text{Transfer}(\mathbf{w}, \mathbf{x}))^2$$

Error functions



Convex error function



Non convex error function

Computing the gradient I

- Move towards a (local) minimum of the error
 - follow the **gradient**
 - compute the partial derivatives of the error as a function of the weights

$$\text{sgm}(x) = \frac{1}{1 + e^{-x}}$$

$$\frac{d}{dx} \text{sgm}(x) = \frac{e^{-x}}{(1 + e^{-x})^2} = (1 - \text{sgm}(x))\text{sgm}(x)$$

Computing the gradient II

- The weight is changed subtracting the partial derivative multiplied by a **learning rate** constant
 - the learning rate influences the convergence speed and can be adjusted as a tradeoff between speed and precision
- The subtraction moves towards smaller errors
- The derivatives of the input weights of the nodes of a layer can be computed if the derivatives for the following layer are known
 - *The actual derivatives are omitted here*

$$w_{ij} \leftarrow w_{ij} - \lambda \frac{\partial E(\mathbf{w})}{\partial w_{ij}}$$

Training algorithm – revised

set all weights to random values

while termination condition is not satisfied **do**
for each training instance x **do**

- 1 – feed the network with x and compute the output $nn(x)$
- 2 – compute error prediction at output layer $nn(x) - x_{Out}$
- 3 – compute derivatives and weight corrections for output layer
- 4 – compute derivatives and weight corrections for hidden layer

Steps 1 and 2 are *forward*, steps 3 and 4 are *backward*

Learning modes

- Stochastic** – each forward propagation is immediately followed by a weight update (as in the algorithm of previous slide)
- introduces some *noise* in the gradient descent process, since the gradient is computed from a single data point
 - reduces the chance of getting stucked in a local minimum
 - good for *online* learning
- Batch** – many propagations occur before updating the weights, accumulating errors over the samples within a batch
- generally yields faster and stable descent towards the *local* minimum, since the update is performed in the direction of the average error

Repetitions

- A learning round over all the samples of the network is called **epoch**
- In general, after each epoch the network classification capability will be improved
- Several epochs will be necessary
- After each epoch the starting weights will be different

Design choices

- The structure of input and output layers is determined by the domain (the training set)
- The number of nodes in the hidden layer can be changed
- The learning rate can be changed in different epochs
 - in the beginning a higher learning rate can push faster towards the desired direction
 - in later epochs a lower learning rate can push more precisely towards a minimum

Stop criteria

- All the weight updates in the epoch have been small
- The classification error rate goes below a predefined target
- A timeout condition is reached

Risks

- Local minima are possible, as usual in gradient tracking methods
- Overfitting is possible, if the network is too complex w.r.t. the complexity of the decision problem

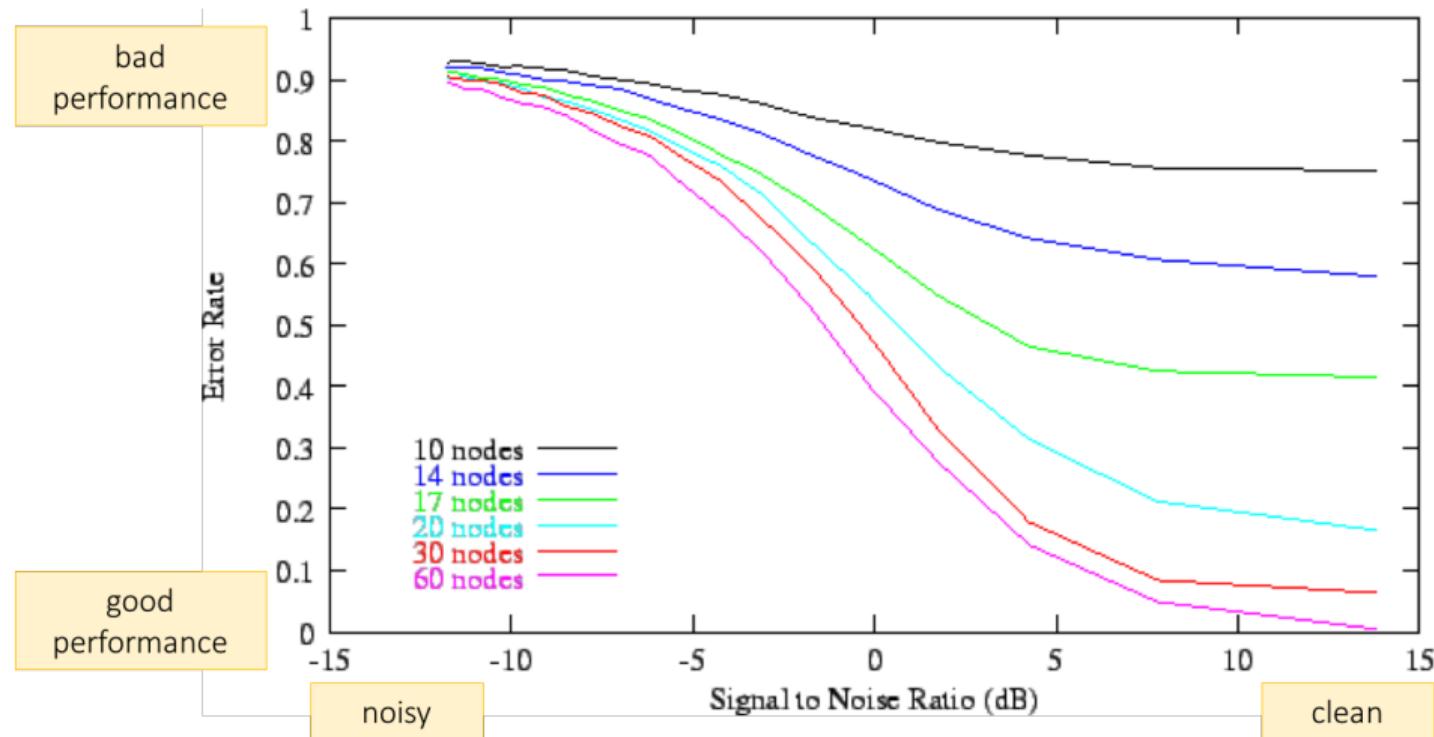
Regularization

- Technique used in many machine learning functions to improve the generalisation capabilities of a model
- Modify the performance function, which is normally chosen to be the sum of squares of the network errors on the training set
- In essence:
 - improvement of performance is obtained by reducing a **loss function** (i.e. the sum of squared errors)
 - regularisation corrects the loss function in order to **smooth** the fitting to the data
 - the amount of regularisation must be tuned

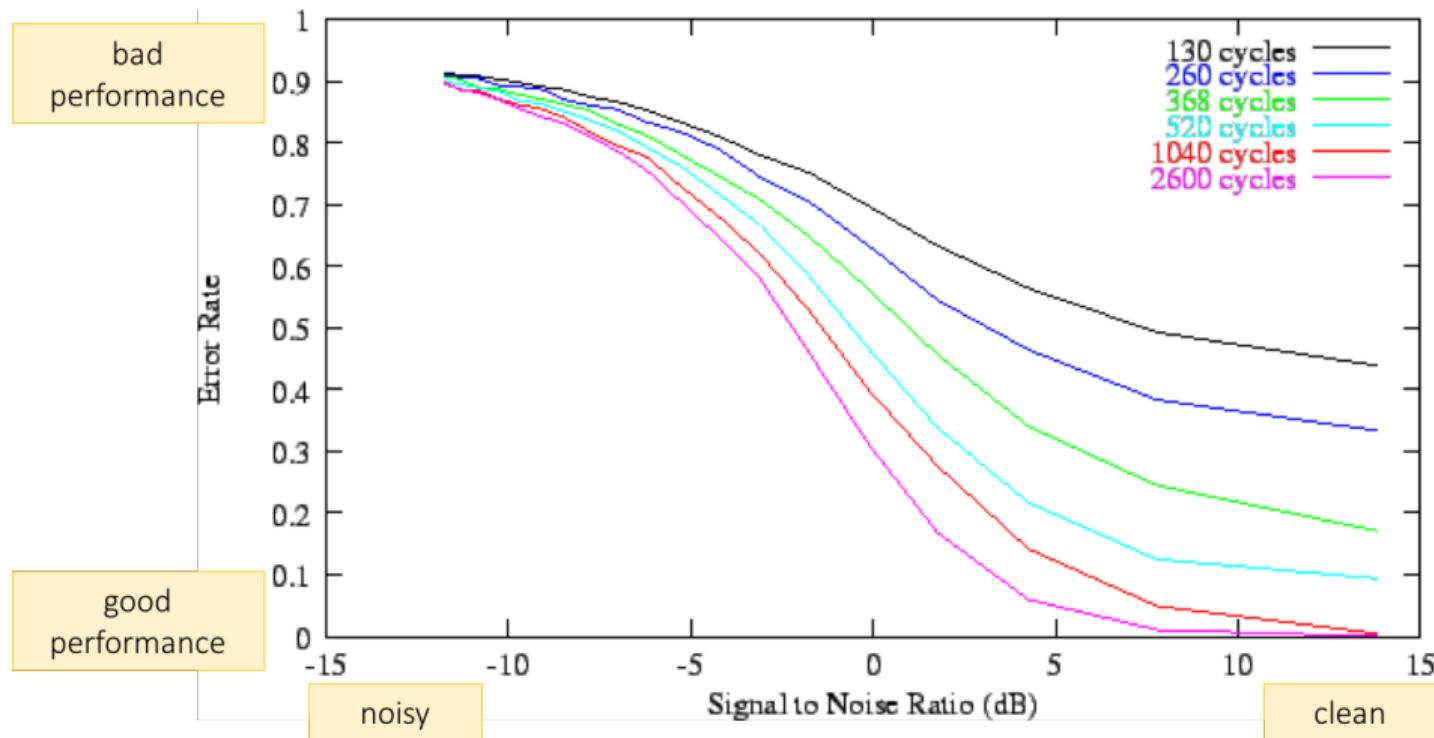
Example

- Recognition of characters with noise
- The images are 7x5 arrays of floating points
 - before noise -1 for black, 1 for white
- 35 input nodes
- up to 60 hidden nodes
- 26 output nodes
 - only one at a time should give a value near 1, all the others should be near 0
 - a good alternative could be a 5 bit coding

Error rate – Varying signal/noise and hidden nodes



Error rate – Varying signal/noise and number of epochs



1	Statistical modeling – Naive Bayes Classifier	2
2	Linear Classification with the Perceptron	17
3	Support Vector Machines	24
4	Neural Networks	42
5	Instance Learning - K Nearest Neighbours Classifier	66
6	From a binary classifier to multi-class classification	68
7	Ensemble methods	73
8	Forest of randomised trees	83
9	Boosting	87
10	Summary	92

K Nearest Neighbours Classifier

- keeps all the training data
 - i.e. the *model* is the entire training set
- future predictions by computing the similarity between the new sample and each training instance
 - can use any similarity function
- picks the K entries in the database which are closest to the new data point
- majority vote
- main parameters
 - the number of neighbours to check
 - the *metric* used to compute the distances
 - the *Mahalanobis* distance has good performance

1	Statistical modeling – Naive Bayes Classifier	2
2	Linear Classification with the Perceptron	17
3	Support Vector Machines	24
4	Neural Networks	42
5	Instance Learning - K Nearest Neighbours Classifier	66
6	From a binary classifier to multi-class classification	68
7	Ensemble methods	73
8	Forest of randomised trees	83
9	Boosting	87
10	Summary	92

From a binary classifier to multi-class classification⁵

- Several classifiers (e.g. SVM and linear perceptron), generate a **binary** classification model
- two ways to deal with multi-class classification
 - transform the training algorithm and the model
 - sometimes at the expenses of an increased size of the problem
 - use a set of binary classifiers and combine the results
 - at the expenses of an increased number of problems to solve
 - **one-vs-one** and **one-vs-all** strategies

One-vs-one strategy (OVO)

- consider all the possible pairs of classes and generate a binary classifier for each pair
 - $C * (C - 1)/2$ pairs
- each binary problem considers only the examples of the two selected classes
- at prediction time apply a **voting** scheme
 - an unseen example is submitted to the $C * (C - 1)/2$ binary classifiers, each winning class receives a +1
 - the class with the highest number of +1 wins

One-vs-rest strategy (OVR)

- consider C binary problems where class c is a positive example, and all the others are negatives
- build C binary classifiers
- at prediction time apply a **voting** scheme
 - an unseen example is submitted to the C binary classifiers, obtaining a confidence score
 - the confidences are combined and the class with the highest global score is chosen

OVO vs OVR

- OVO requires solving a higher number of problems, even if they are of smaller size
- OVR tends to be intrinsically unbalanced
 - if the classes are evenly distributed in the examples, each classifier has a proportion positive to negative 1 to $C - 1$

1	Statistical modeling – Naive Bayes Classifier	2
2	Linear Classification with the Perceptron	17
3	Support Vector Machines	24
4	Neural Networks	42
5	Instance Learning - K Nearest Neighbours Classifier	66
6	From a binary classifier to multi-class classification	68
7	Ensemble methods	73
	● Methods for ensemble classifiers	78
8	Forest of randomised trees	83
9	Boosting	87
10	Summary	92

Ensemble methods

aka Classifier combination

- train a set of **base classifiers**
- the final prediction is obtained taking the votes of the base classifiers
- ensemble methods tend to perform better than a single classifier

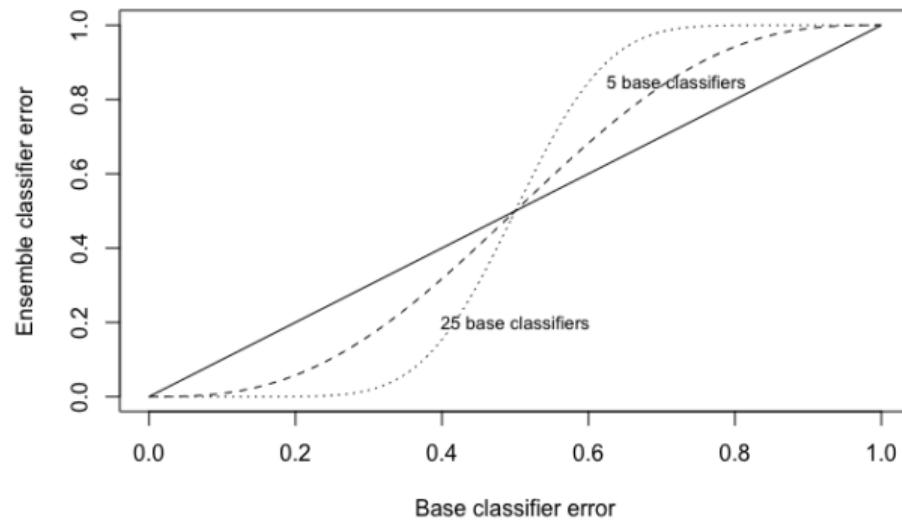
Why should this be better?

A hypothetical, extreme case

- let us consider 25 binary classifiers, each of which has **error rate** $\epsilon = 0.35$
- the ensemble classifier output is the majority of the predictions (13 or more)
- if the base classifiers are equal, the ensemble error rate is still ϵ
- if the base classifiers have all error rate ϵ , but they are independent, i.e. their errors are uncorrelated, then the ensemble will be wrong only when the majority of the base classifier is wrong

$$\epsilon_{ensemble} = \sum_{i=13}^{25} \binom{25}{i} \epsilon^i (1 - \epsilon)^{25-i} = 0.06$$

Ensemble error – a hypothetical extreme case



Comparison between error of the base classifiers and error of the ensemble classifier

Rationale for ensemble method

Ensemble methods are useful if:

1. the base classifiers are independent
2. the performance of the base classifier is better than random choice

Methods for ensemble classifiers I

By manipulating the training set -

Data are resampled according to some sampling strategy

Bagging repeatedly samples with replacement according to a uniform probability distribution

Boosting iteratively changes the distribution of training examples so that the base classifier focus on examples which are hard to classify

Adaboost the importance of each base classifier depends on its error rate

Methods for ensemble classifiers II

By manipulating the input features -

Subset of input features can be chosen either random or according to suggestions from domain experts

Random forest uses decision trees as base classifiers

- frequently produces very good results

Methods for ensemble classifiers III

By manipulating the class labels -

- Useful when the number of classes is high

- For each base classifier, randomly partition class labels into two subsets, say A_1, A_2 and re-label the dataset

- Train binary classifiers with respect to the two classes

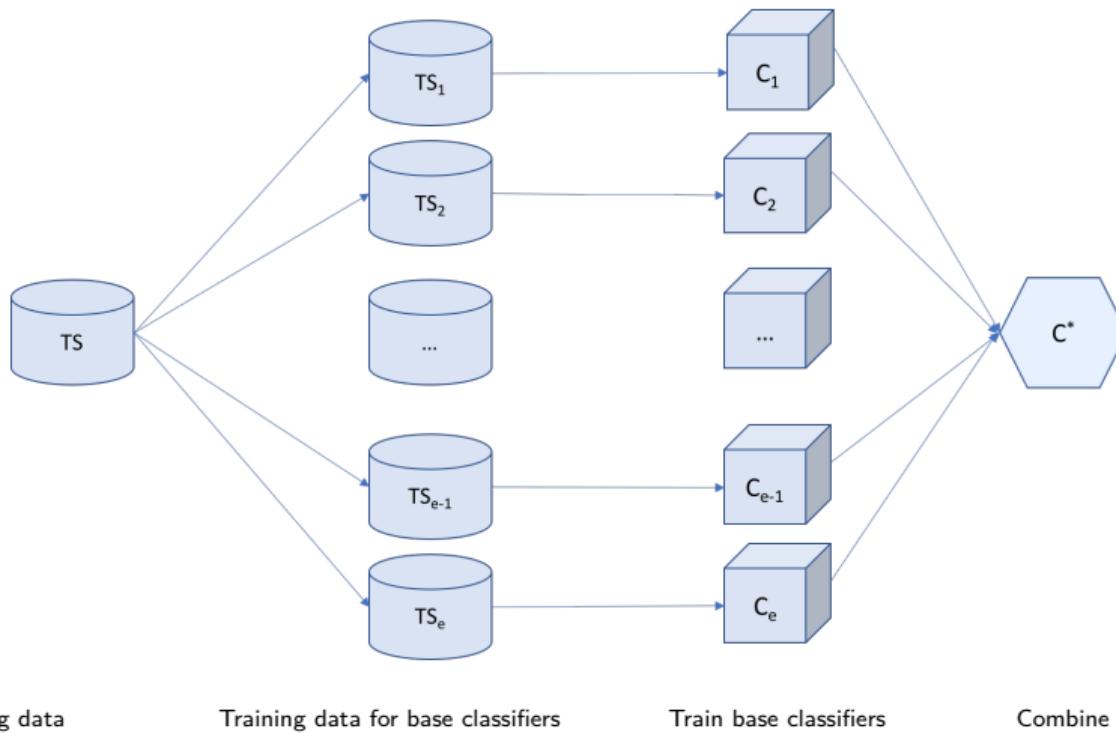
- At testing time, when a subset is selected all the classes it includes receive a vote

- The class with the top score will win

Methods for ensemble classifiers IV

Error-correcting output coding

General schema for ensemble methods



1	Statistical modeling – Naive Bayes Classifier	2
2	Linear Classification with the Perceptron	17
3	Support Vector Machines	24
4	Neural Networks	42
5	Instance Learning - K Nearest Neighbours Classifier	66
6	From a binary classifier to multi-class classification	68
7	Ensemble methods	73
8	Forest of randomised trees	83
9	Boosting	87
10	Summary	92

Forest of randomised trees

- perturb-and-combine techniques specifically designed for trees
- a diverse set of classifiers is created by introducing randomness in the classifier construction
 - the classifiers must be independent
 - each tree in the ensemble is built from a sample drawn with replacement (i.e., a bootstrap sample) from the training set
 - furthermore, when splitting each node during the construction of a tree, the best split is found either from all input features or a random subset of size `max_features`
- the prediction of the ensemble is given as the averaged prediction of the individual classifiers

Bias-vs-Variance tradeoff⁶

- Bias is the simplifying assumptions made by the model to make the target function easier to approximate
- Variance is the amount that the estimate of the target function will change, given different training data
- Bias-variance trade-off is the sweet spot where our machine model performs between the errors introduced by the bias and the variance.

If Bias vs Variance was the act of reading, it could be like

Skimming a Text vs Memorizing a Text

6 <https://www.kdnuggets.com/2020/09/understanding-bias-variance-trade-off-3-minutes.html>

Random Forest

the purpose of the two sources of randomness is to decrease the variance of the forest estimator

- individual decision trees typically exhibit high variance and tend to overfit
- the injected randomness in forests yield decision trees with somewhat decoupled prediction errors
- by taking an average of those predictions, some errors can cancel out
- random forests achieve a reduced variance by combining diverse trees, sometimes at the cost of a slight increase in bias.
- In practice the variance reduction is often significant hence yielding an overall better model.

1	Statistical modeling – Naive Bayes Classifier	2
2	Linear Classification with the Perceptron	17
3	Support Vector Machines	24
4	Neural Networks	42
5	Instance Learning - K Nearest Neighbours Classifier	66
6	From a binary classifier to multi-class classification	68
7	Ensemble methods	73
8	Forest of randomised trees	83
9	Boosting	87
10	Summary	92

Ensemble learning with *boosting*

- a weight is associated to each training instance
- different classifiers are trained with those weights
- the weights are modified iteratively according to classifier performance

AdaBoost I

- fit a sequence of weak learners on repeatedly modified versions of the data
- the predictions from all of them are then combined through a weighted majority vote (or sum) to produce the final prediction
- the data modifications at each so-called boosting iteration consist of applying and modifying weights to each of the training samples

AdaBoost II

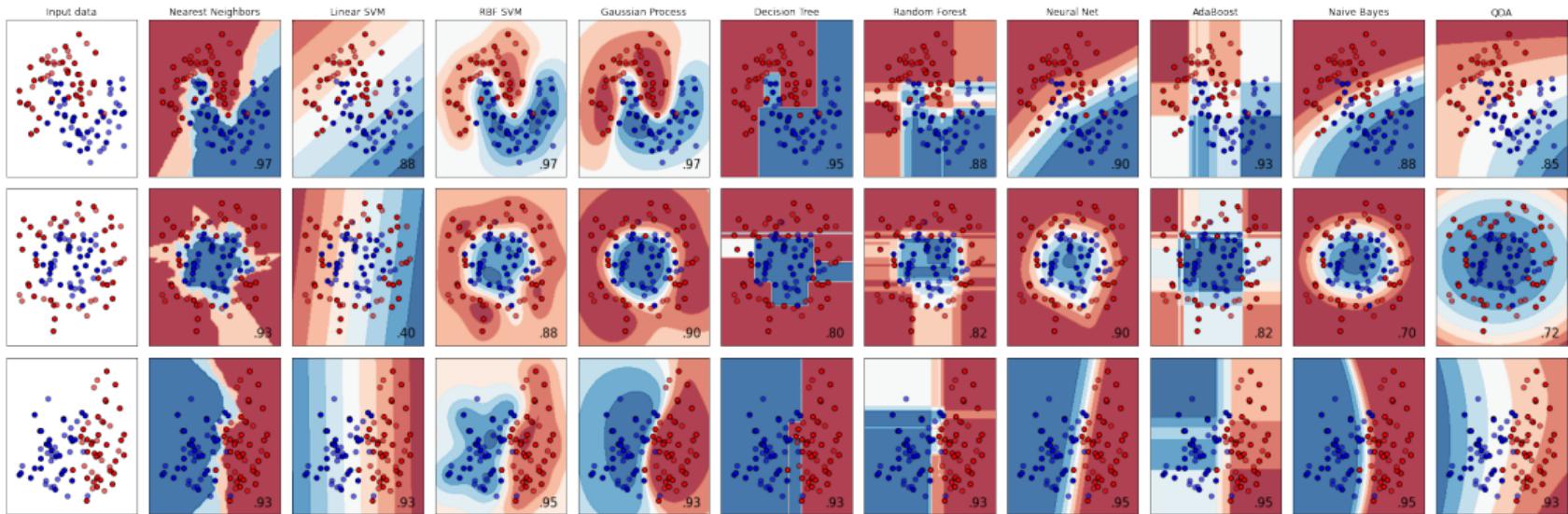
- initially, the weights w_1, w_2, \dots, w_N are all set to $1/N$, so that the first step simply trains a weak learner on the original data
- for each successive iteration, the sample weights are individually modified and the learning algorithm is reapplied to the *reweighted* data
 - at a given iteration, the training examples that were incorrectly predicted by the boosted model induced at the previous step have their weights increased, whereas the weights are decreased for those that were predicted correctly

AdaBoost III

- as iterations proceed, examples that are difficult to predict receive ever-increasing influence; each subsequent weak learner is thereby forced to concentrate on the examples that are missed by the previous ones in the sequence

1	Statistical modeling – Naive Bayes Classifier	2
2	Linear Classification with the Perceptron	17
3	Support Vector Machines	24
4	Neural Networks	42
5	Instance Learning - K Nearest Neighbours Classifier	66
6	From a binary classifier to multi-class classification	68
7	Ensemble methods	73
8	Forest of randomised trees	83
9	Boosting	87
10	Summary	92

Classifiers comparison



Bibliography

- ▶ Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani.
An Introduction to Statistical Learning.
Springer, 2015.
- ▶ Ian H. Witten, Eibe Frank, and Mark Hall.
Data Mining – Practical Machine Learning Tools and Techniques.
Morgan Kaufman, 2011.
ISBN 978-0-12-374856-0.

Machine Learning

Regression - Forecasting continuous values

Claudio Sartori

DISI

Department of Computer Science and Engineering – University of Bologna, Italy

claudio.sartori@unibo.it

Regression – Forecasting continuous values

- Supervised task
- The **target** variable is numeric
- **Minimize** the **error** of the prediction with respect to the target

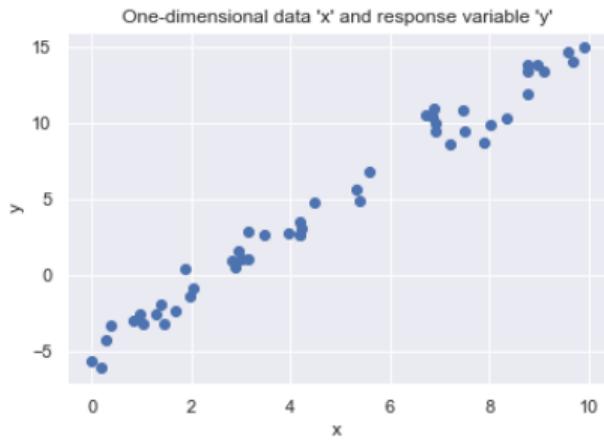
Linear Regression

- data set \mathcal{X} with N rows and D columns
 - x_i is a D dimensional **data element**
- response vector \bar{y} with N values y_i
- w is a D -dimensional vector of coefficients that needs to be learned
- we model the dependence of each response value y_i from the corresponding independent variables x_i as

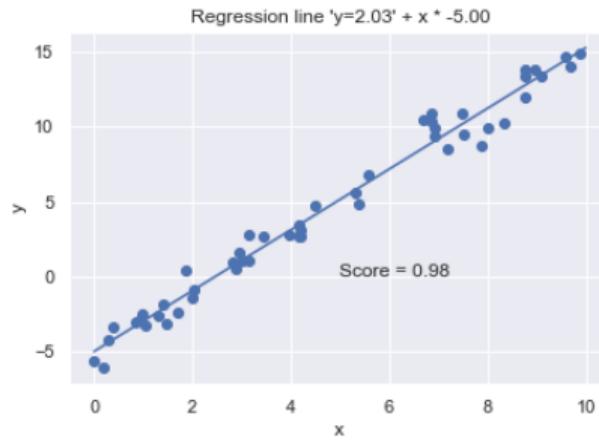
$$y_i \approx w^T \cdot x_i \quad \forall i \in [1 \dots N]$$

- such that the **error of modelling** is minimised
- Classical statistic method (1805)

Data and regression line



One-dimensional data and response variable



Regression and score - Score range (- inf : 1)

Objective function and minimisation |

$$\begin{aligned}\mathcal{O} &= \sum_{i=1}^N (w^T \cdot x_i - y_i)^2 = \|Xw^T - y\|^2 \\ &= (Xw^T - y)^T \cdot (Xw^T - y)\end{aligned}$$

Gradient of \mathcal{O} with respect to w

$$2X^T(Xw^T - y)$$

Constraining the gradient to 0 we obtain the optimisation condition

$$X^T X w^T = X^T y$$

Objective function and minimisation II

If the symmetric matrix $X^T X$ is *invertible* the solution can be derived as

$$w = (X^T X)^{-1} X^T y$$

and the forecast is given by

$$y^f = X \cdot w^T$$

Matrix calculus

- Issues related to matrix calculus if $\bar{x}^T \bar{x}$ is not invertible
- *Moore–Penrose pseudoinverse*
- *Tikonov regularisation* (also known as *ridge regression*)
- *Lasso regularisation*

Quality of the fitting - R^2

Mean of the observed data

$$y^{avg} = \frac{1}{N} \sum_i y_i$$

Sum of squared residuals

$$SS_{res} = \sum_i (y_i - y_i^f)^2$$

Total sum of squares

$$SS_{tot} = \sum_i (y_i - y^{avg})^2$$

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

Intuition about R^2

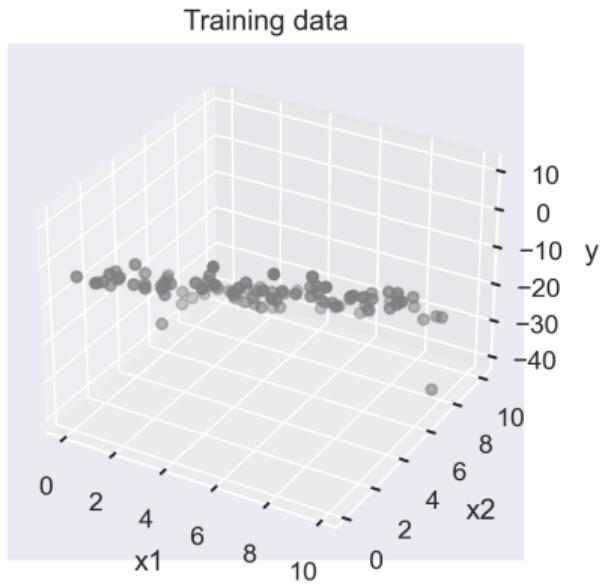
- It compares the fit of the chosen model with that of a horizontal straight line
- With perfect fitting the numerator of the second term is zero and $R^2 = 1$
- If the model does not follow the trend of the data the numerator of the second term can reach or exceed the denominator, and R^2 can also be negative
- Despite the name, R^2 isn't the square of anything

R² and Mean Squared Error

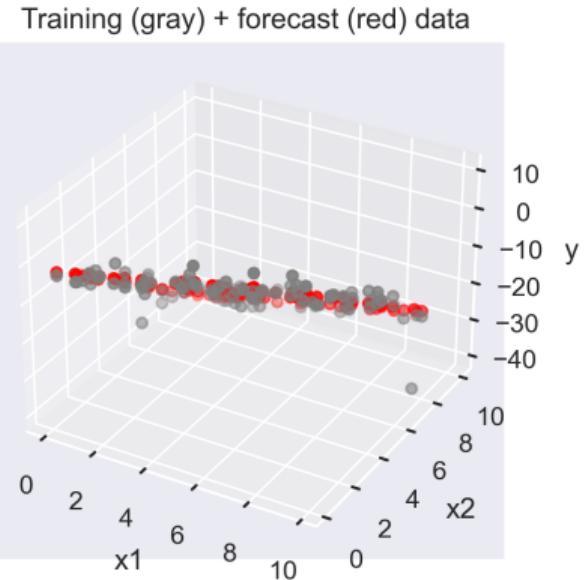
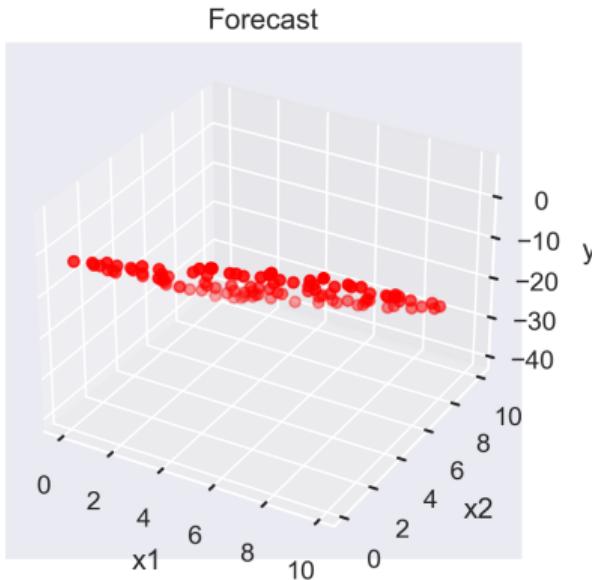
- Both refer to the error of the predictions
- R^2 is a standardised index,
- MSE measures the mean error, this it is influenced by the order of magnitude of the data

Multiple regression

- The response variable depends by more than one features
- The regression technique is quite similar to that of simple regression
- In scikit-learn the estimator is the same



Multiple regression - forecast

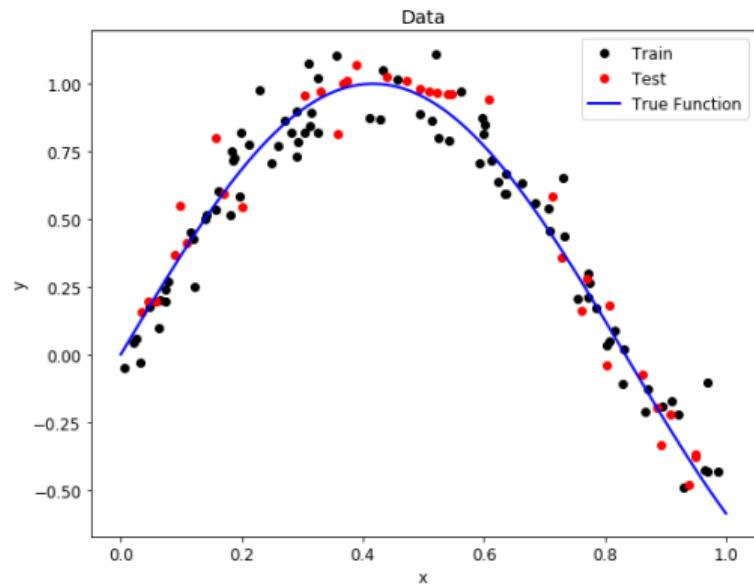


Overfitting and Regularisation

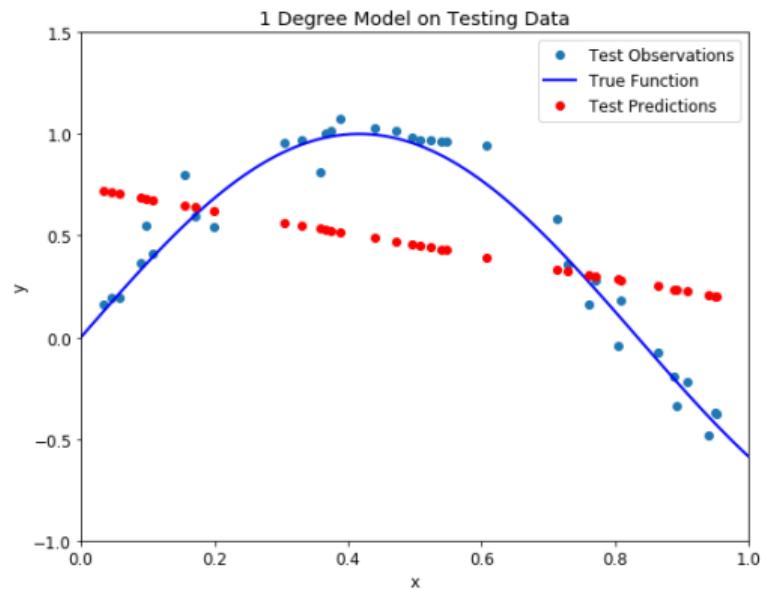
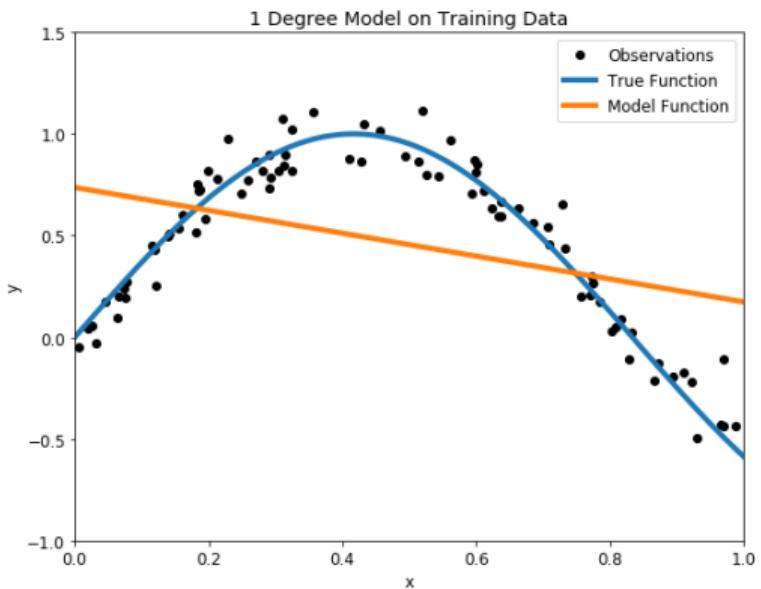
- In presence of high number of features **overfitting** is possible
 - performance on test data becomes much worse
- Regularisation reduces the influence of less interesting attributes and therefore reduces overfitting

Polynomial regression

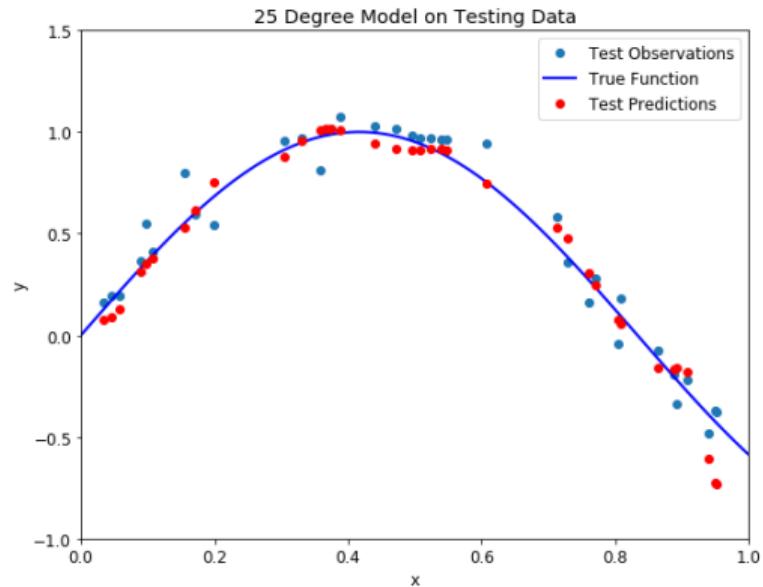
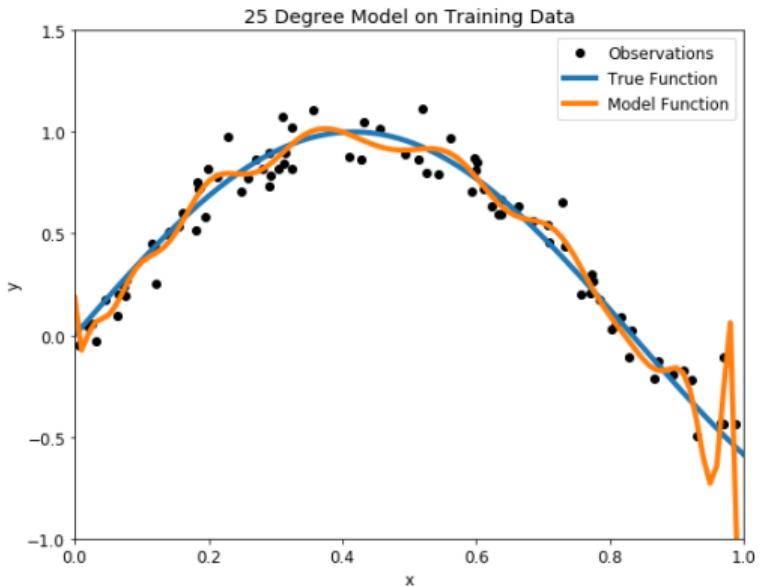
- Target is influenced by a single feature
- The relationship cannot be described by a straight line



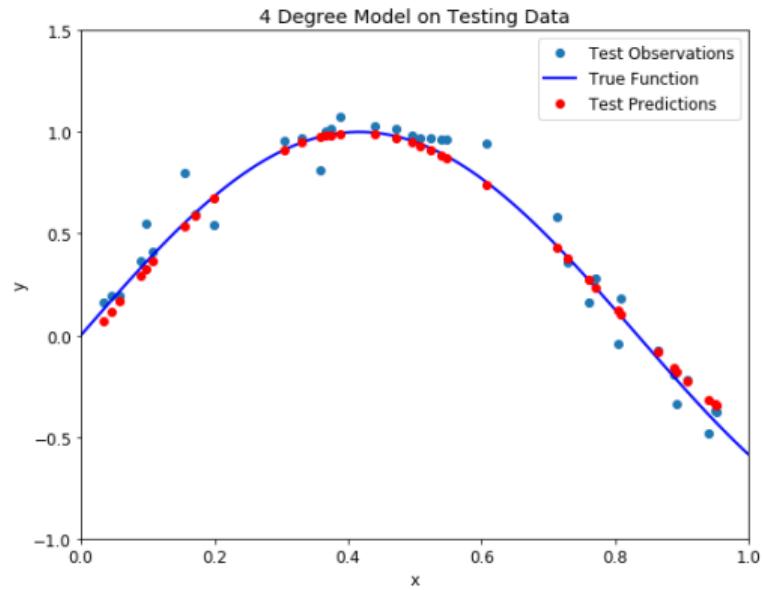
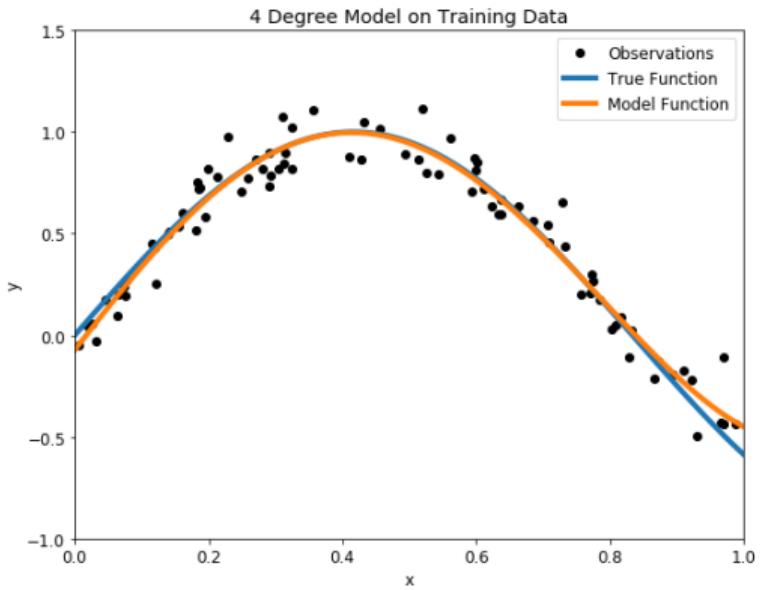
Underfitting



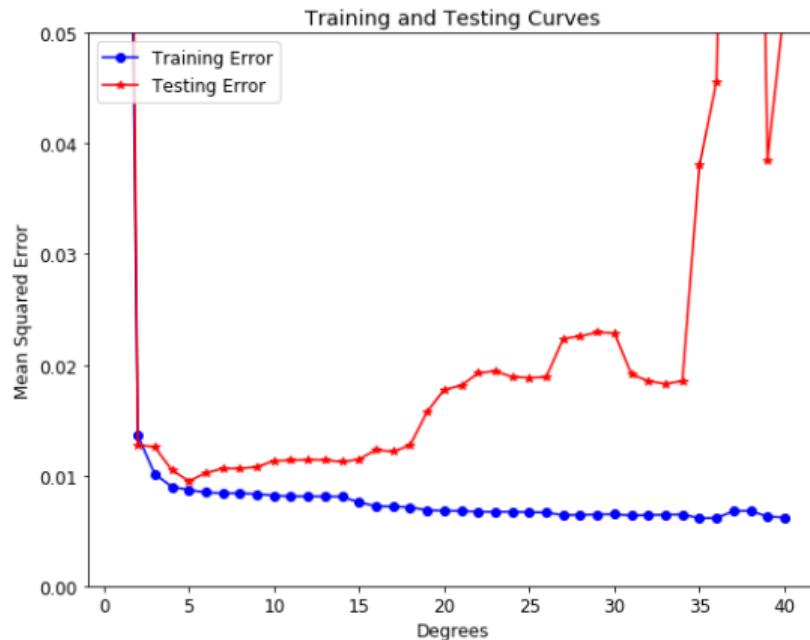
Overfitting



Good fitting



Model complexity vs fitting



Summary of Regression Models available in scikit-learn

Gradient Boosting Regression

Elastic Net Regression

Stochastic Gradient Descent Regression

Support Vector Machine

Bayesian Ridge Regression

CatBoost Regressor

Kernel Ridge Regression

Linear Regression

XGBoost Regressor

LGBM Regressor

```
from sklearn.ensemble import GradientBoostingRegressor  
from sklearn.linear_model import ElasticNet  
from sklearn.linear_model import SGDRegressor  
from sklearn.svm import SVR  
from sklearn.linear_model import BayesianRidge  
from catboost import CatBoostRegressor  
from sklearn.kernel_ridge import KernelRidge  
from sklearn.linear_model import LinearRegression  
from xgboost.sklearn import XGBRegressor  
from lightgbm import LGBMRegressor
```

Machine Learning

The Data - Pre-processing and dissimilarities

Claudio Sartori

DISI

Department of Computer Science and Engineering – University of Bologna, Italy

claudio.sartori@unibo.it

1	Pre-processing	2
●	Sampling	6
●	Dimensionality	14
●	Dimensionality reduction	16
●	Feature creation	20
2	Data type conversions	22
3	Similarity and dissimilarity	32
4	Data transformations	53
5	Imbalanced data in classification	65

Data pre-processing

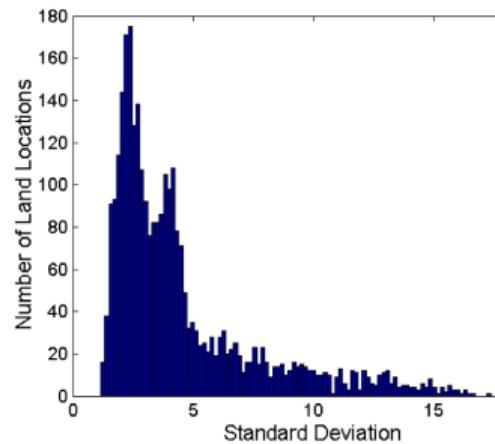
- Aggregation
- Sampling
- Dimensionality Reduction
- Feature subset selection
- Feature creation
- Discretization and Binarization Attribute Transformation

Aggregation

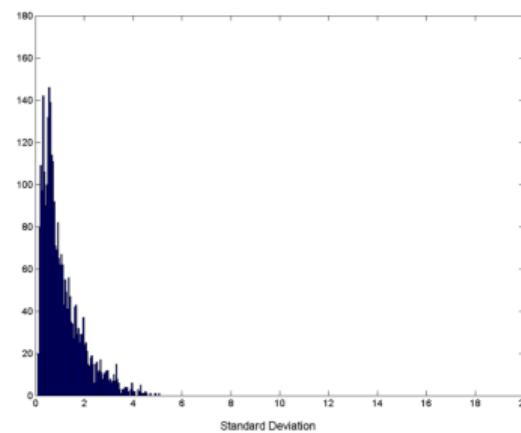
- Combining two or more attributes (or objects) into a single attribute (or object)
- Purpose
 - Data reduction
 - Reduce the number of attributes or objects
 - Change of scale
 - Cities aggregated into regions, states, countries, etc
 - *More stable data*
 - Aggregated data tends to have less variability

Aggregation – Example

Variation of precipitation in Australia



Standard Deviation of
Average Monthly Precipitation



Standard Deviation of
Average Yearly Precipitation

Sampling I

- For both **preliminary** investigation and final data analysis
- Statistician perspective
 - **obtaining** the entire data set could be impossible or too expensive
- Data processing perspective
 - **processing** the entire data set could be too expensive or time consuming

Sampling II

1. using a sample will work almost as well as using the entire data sets, *if the sample is representative*
2. A sample is representative if it has approximately the same property (of interest) as the original set of data

Types of sampling

1. Simple random
 - a single random choice of an object with given probability distribution
2. With replacement
 - repetition of independent extractions of type 1
3. Without replacement
 - repetition of extractions, extracted element is removed from the population
4. Stratified
 - split data into several partitions according to some criteria, then draw the random samples from each partition
 - used when the data set is split into subsets with homogeneous characteristics
 - the representativity is guaranteed inside each subset

Sample size

- Statistics provides techniques to assess
 - optimal sample size
 - sample signifiativity
- Tradeoff between data reduction and precision

Sampling with/without replacement

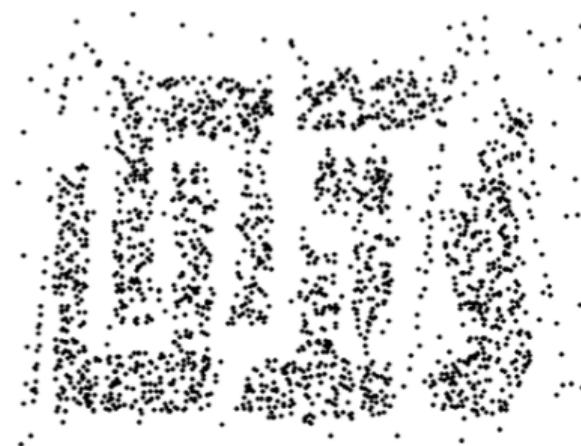
- they are nearly equivalent if sample size is a small fraction of data set size
- with replacement, in a small population a small subset could be underestimated
- sampling with replacement is
 - much easier to implement
 - much easier to be interpreted from a statistical point of view
 - extractions are statistically independent

Sample size – Example

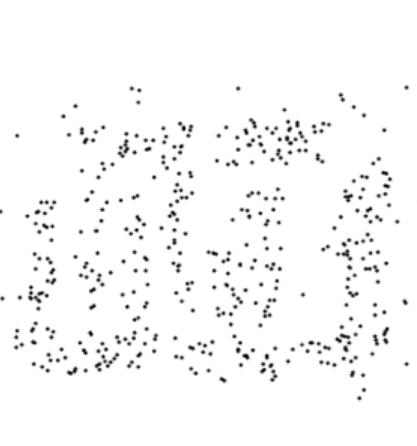
Loss of information



8000 points



2000 points



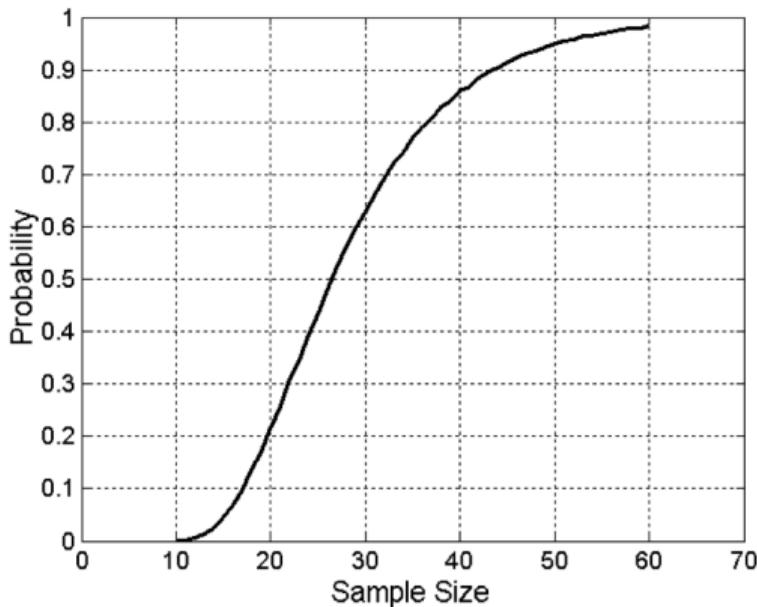
500 points

Sample size – Missing class I¹

- Probability of sampling at least one element for each class (with replacement)
 - it is independent from the size of the data set!

A B C D E
 F G H I J

Ten classes



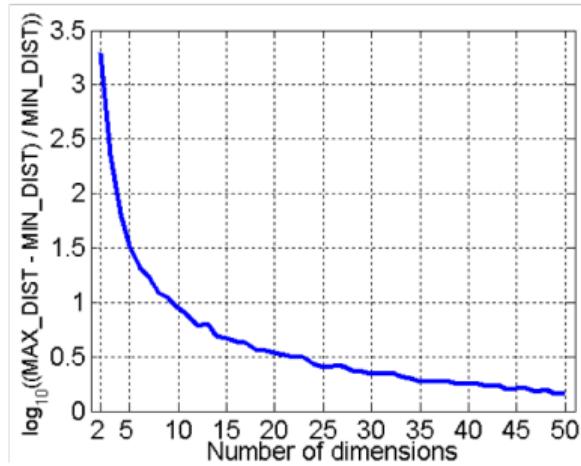
1 It is an instance of the Coupon Collector's Problem

Sample size – Missing class - II

- This aspect becomes relevant, for example, in a supervised dataset with a high number of different values of the target
- If the number data elements is not big enough, it can be difficult to guarantee a stratified partitioning in train/test split or in cross-validation split
- Example:
 - $N = 1000$, $C = 10$, test-set-size = 300, cross-validation-folds = 10
 - the probability of folds without adequate representation of some classes becomes quite high
- When designing the training processes it is necessary to consider those aspects
 - in the example, one could use only 3 folds in cross-validation

The curse of dimensionality

- When dimensionality is very high the occupation of the space becomes very sparse
- Discrimination on the basis of the distance becomes uneffective
- Experiment:
 - random generation of 500 points
 - plot the relative difference between the maximum and the minimum distance between pairs of points



Dimensionality reduction

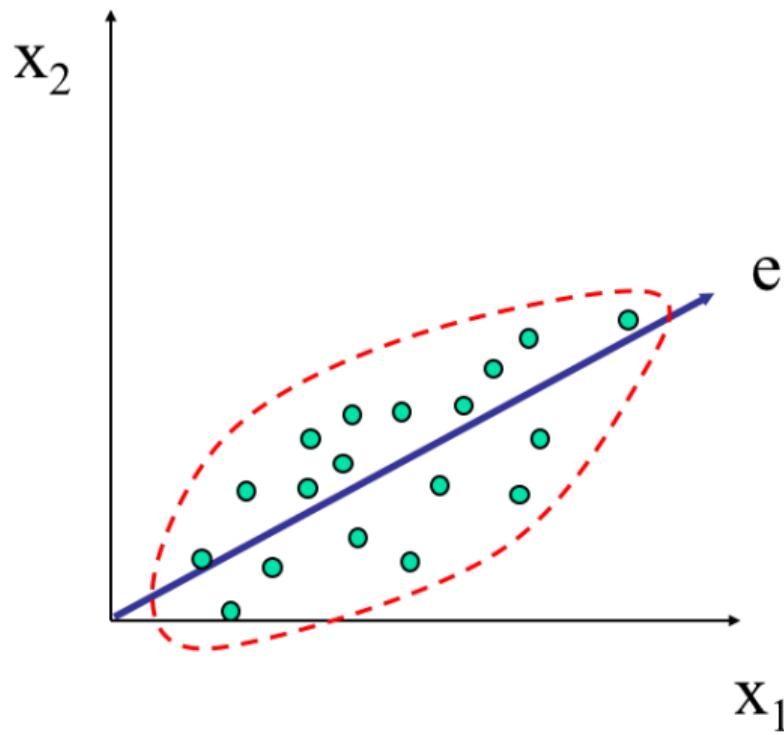
- Purposes:
 - avoid the *curse of dimensionality*
 - noise reduction
 - reduce time and memory complexity of the mining algorithms
 - visualization
- Techniques:
 - principal component analysis
 - singular values decomposition
 - supervised techniques
 - non-linear techniques

PCA

Find projections that capture most of the data variation

- Find the eigenvector of the covariance matrix
- the eigenvectors define the new space

The new dataset will have *only the attributes* which capture most of the data variation



Feature subset selection I

A *local* way to reduce dimensionality

- Redundant attributes
 - duplicate most of the information contained in other attributes
 - e.g. price and v.a.t. amount
- Irrelevant attributes
 - do not contain any information useful for analysis
 - e.g. SSN is not relevant to predict wealth

Feature subset selection II

1. Brute force

- try all possible feature subsets as input to data mining algorithm and measure the effectiveness of the algorithm with the reduced dataset

2. Embedded approach

- Feature selection occurs naturally as part of the data mining algorithm
 - e.g. decision trees

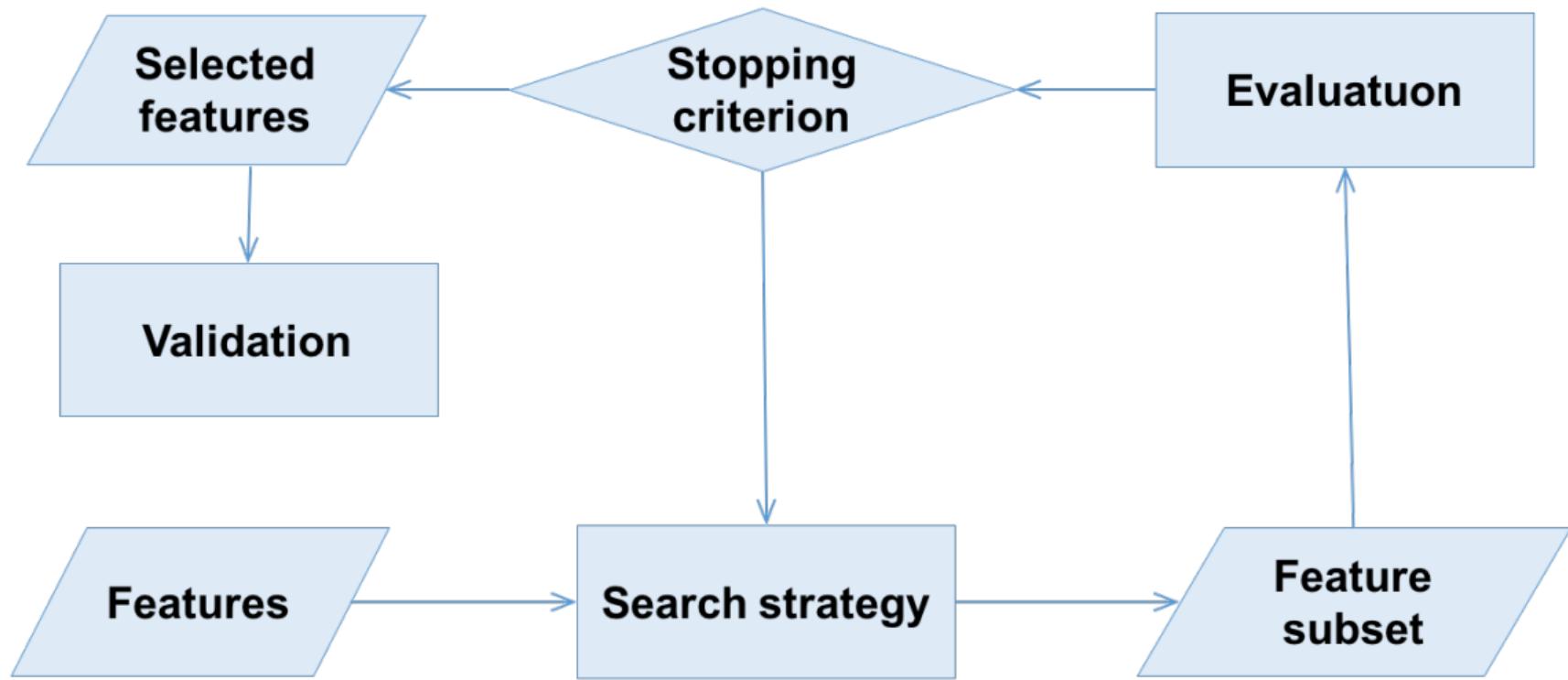
3. Filter approach

- Features are selected before data mining algorithm is run

4. Wrapper approaches

- A data mining algorithm can choose the best set of attributes
 - try as in 1, but without exhaustive search

An Architecture for Feature Subset Selection

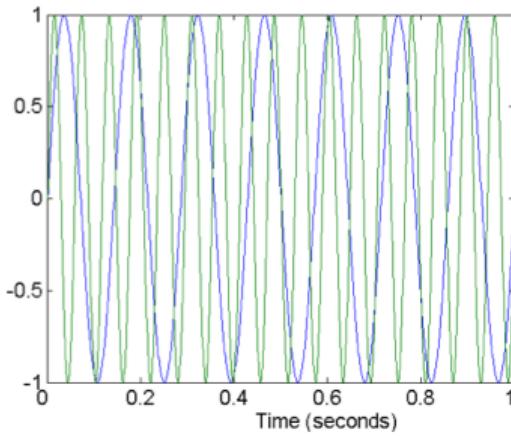


Feature creation

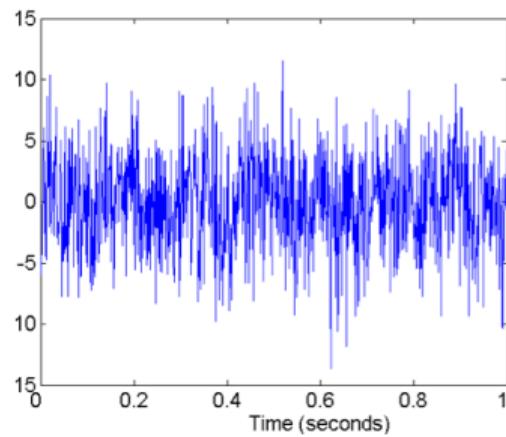
New features can capture more efficiently data characteristics

- Feature extraction
 - pixel picture with a face \Rightarrow eye distance, ...
- Mapping to a new space
 - e.g. signal to frequencies with Fourier transform
- New features
 - e.g. volume and weight to density

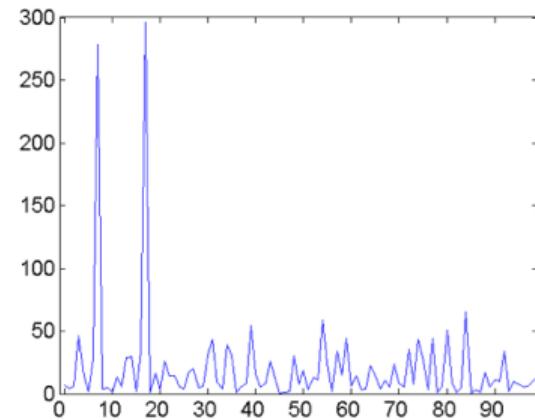
Space transformation example



Two sinusoids



Two sinusoids with noise



Transformation in the domain of frequencies

1	Pre-processing	2
2	Data type conversions	22
	• The scikit-learn solution for type conversions	24
3	Similarity and dissimilarity	32
4	Data transformations	53
5	Imbalanced data in classification	65

Why do we need type conversion?

- Many algorithms require numeric features
 - categorical features must be transformed into numeric
 - ordinal features must be transformed into numeric, and the order must be preserved
- Classification requires a target with nominal values
 - a numerical target can be **discretised**
- Discovery of association rules require boolean features
 - a numerical feature can be discretised and transformed into a series of boolean features

Binarization of discrete attributes

Attribute d allowing V values $\Rightarrow V$ binary attributes.

<i>Quality</i>	<i>Quality-Awful</i>	<i>Quality-Poor</i>	<i>Quality-OK</i>	<i>Quality-Good</i>	<i>Quality-Great</i>
Awful	1	0	0	0	0
Poor	0	1	0	0	0
Ok	0	0	1	0	0
Good	0	0	0	1	0
Great	0	0	0	0	1

Nominal to numeric

- One-Hot-Encoding

- a feature with V unique values is substituted by V binary features each one corresponding to one of the unique values
- if object x has value v in feature d then the binary feature corresponding to v has True for x , all the other binary features have value False
- True and False are represented as 1 and 0, therefore can be processed by also by procedures working only on numeric data, as is the case for the estimators available in scikit-learn

- `sklearn.preprocessing.OneHotEncoder`

[link to manual page](#)

Ordinal to numeric

- The ordered sequence is transformed into consecutive integers
 - by default the lexicographic order is assumed
 - The user can specify the proper order of the sequence
- `sklearn.preprocessing.OrdinalEncoder`

[link to manual page](#)

Numeric to binary with threshold

- Not greater than the threshold becomes zero
- Greater than the threshold becomes one
- `sklearn.preprocessing.Binarizer`

[link to manual page](#)

Discretization/Reduction of the number of distinct values

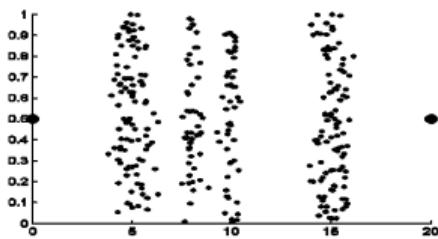
- Some algorithms work better with categorical data
- A small number of distinct values can let patterns emerge more clearly
- A small number of distinct values let the algorithms to be less influenced by noise and random effects

Discretization

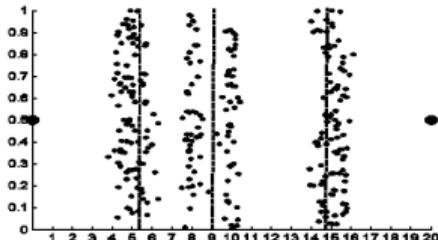
- Continuous \Rightarrow Discrete
 - thresholds
 - many options
 - binarization \Rightarrow single threshold
- Discrete with many values \Rightarrow Discrete with less values
 - guided by domain knowledge

Continuous \Rightarrow Discrete

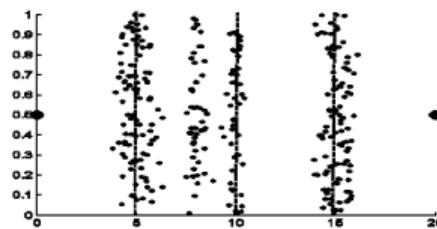
Boundaries on x axis – Unsupervised



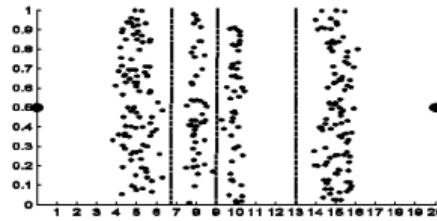
Data



Equal frequency



Equal width



K-means

Numeric to k values

- The numbers are discretised into a sequence of integers 0 to $k - 1$
- Several strategies are available
 - {'uniform', 'quantile', 'means'}
- `sklearn.preprocessing.KBinsDiscretizer`

[link to manual page](#)

1	Pre-processing	2
2	Data type conversions	22
3	Similarity and dissimilarity	32
	● Distance	35
	● Similarity	44
	● Correlation	49
4	Data transformations	53
5	Imbalanced data in classification	65

Similarity and dissimilarity

- Similarity
 - Numerical measure of how alike two data objects are
 - Is higher when objects are more alike
 - Often falls in the range [0,1]
- Dissimilarity
 - Numerical measure of how different are two data objects
 - Lower when objects are more alike
 - Minimum dissimilarity is often 0
 - Upper limit varies
- Proximity refers to a similarity or dissimilarity

Similarity and Dissimilarity by Attribute type

p and q are the values of an attribute for two data objects

Attribute type	Dissimilarity	Similarity
Nominal	$d = \begin{cases} 0 & \text{if } p = q \\ 1 & \text{if } p \neq q \end{cases}$	$s = \begin{cases} 1 & \text{if } p = q \\ 0 & \text{if } p \neq q \end{cases}$
Ordinal Values mapped to integers 0 to $V-1$	$d = \frac{ p-q }{V-1}$	$s = 1 - \frac{ p-q }{V-1}$
Interval or Ratio	$d = p - q $	$s = \frac{1}{1+d}$ or $s = 1 - \frac{d - \min(d)}{\max(d) - \min(d)}$

Euclidean distance – L_2

$$\text{dist} = \sqrt{\sum_{d=1}^D (p_d - q_d)^2}$$

- Where D is the number of dimensions (attributes) and p_d and q_d are, respectively, the d -th attributes (components) of data objects p and q
- Standardization/Rescaling is necessary if scales differ

Minkowski distance – L_r

$$\text{dist} = \left(\sum_{d=1}^D |p_d - q_d|^r \right)^{\frac{1}{r}}$$

- Where D is the number of dimensions (attributes) and p_d and q_d are, respectively, the d -th attributes (components) of data objects p and q
- Standardization/Rescaling is necessary if scales differ
- r is a *parameter* which is chosen depending on the data set and the application

Minkowski distance – Cases

$r = 1$ also named *city block*, *Manhattan*, L_1 norm

- it is the best way to discriminate between zero distance and *near zero* distance
- a ϵ change on any coordinate causes a ϵ change in the distance
- works better than euclidean in very high dimensional spaces

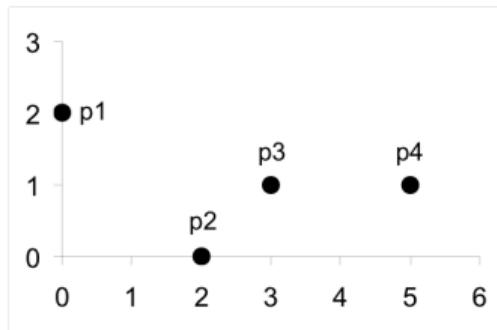
$r = 2$ euclidean, L_2 norm

$r = \infty$ also named Chebyshev, *supremum*, L_{max} norm, L_∞ norm

- considers only the dimension where the difference is maximum
- provides a simplified evaluation, disregarding the dimensions with lower differences

$$\text{dist}_\infty = \lim_{r \rightarrow \infty} \left(\sum^D |p_d - q_d|^r \right)^{\frac{1}{r}} = \max_d |p_d - q_d|$$

Minkowski distances – Example



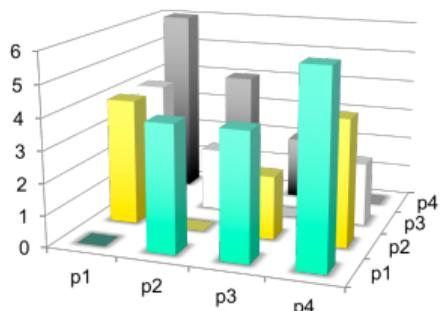
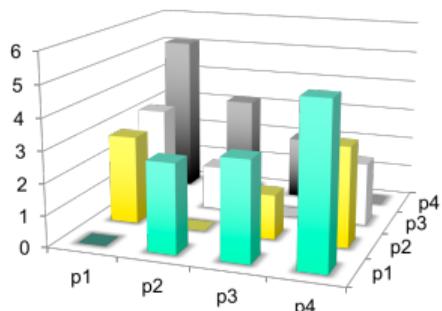
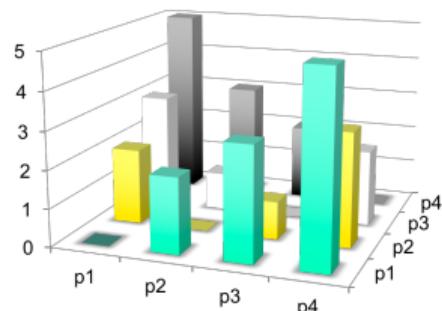
point	x	y
p1	0	2
p2	2	0
p3	3	1
p4	5	1

L_1	p1	p2	p3	p4
p1	0	4	4	6
p2	4	0	2	4
p3	4	2	0	2
p4	6	4	2	0

L_2	p1	p2	p3	p4
p1	0	2.828	3.162	5.099
p2	2.828	0	1.414	3.162
p3	3.162	1.414	0	2
p4	5.099	3.162	2	0

L_∞	p1	p2	p3	p4
p1	0	2	3	5
p2	2	0	1	3
p3	3	1	0	2
p4	5	3	2	0

Comparison

 L_1  L_2  L_∞

Mahalanobis Distance

- Considers **data distribution**
- The Mahalanobis distance between two points p and q decreases if, keeping the same euclidean distance, the segment connecting the points is stretched along a direction of greater variation of data
- The distribution is described by the **covariance matrix** of the data set

$$\Sigma_{ij} = \frac{1}{N-1} \sum_{k=1}^N (x_{ki} - \bar{x}_i)(x_{kj} - \bar{x}_j)$$

$$\text{dist}_m = \sqrt{(p - q)\Sigma^{-1}(p - q)^T}$$

Mahalanobis Distance – Example

$$\Sigma = \begin{bmatrix} 0.3 & 0.2 \\ 0.2 & 0.3 \end{bmatrix}$$

$$A = (0.5, 0.5)$$

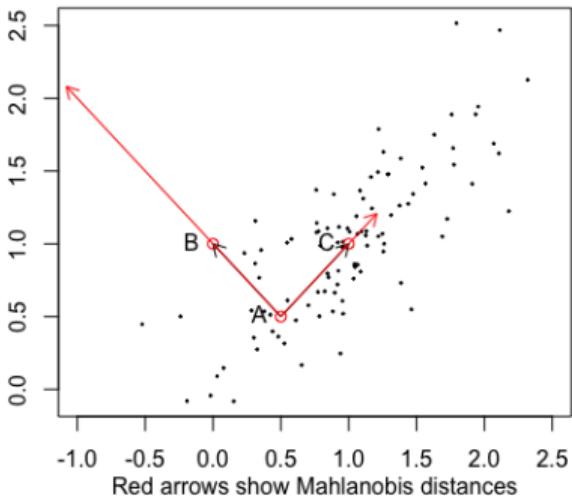
$$B = (0, 1)$$

$$C = (1, 1)$$

The euclidean distances AB and AC are the same

$$\text{dist}_m(A, B) = 2.236068$$

$$\text{dist}_m(A, C) = 1$$



Covariance matrix

- Variation of pairs of random variables
- The summation is over all the observations
- The main diagonal contains the variances
- The values are positive if the two variables grow together
- If the matrix is diagonal the variables are non-correlated
- If the variables are standardised the diagonal contains “one”
- If the variables are standardised and non correlated, the matrix is the identity and the Mahalanobis distance is the same as the euclidean

Common properties of a distance

1. **Positive definiteness:** $\text{Dist}(p, q) \geq 0 \quad \forall p, q$
and $\text{Dist}(p, q) = 0$ if and only if $p = q$
2. **Symmetry:** $\text{Dist}(p, q) = \text{Dist}(q, p)$
3. **Triangle inequality:** $\text{Dist}(p, q) \leq \text{Dist}(p, r) + \text{Dist}(r, q) \quad \forall p, q, r$

A distance function satisfying all the properties above is called a **metric**

Common properties of a Similarity

1. $\text{Sim}(p, q) = 1$ only if $p = q$
2. $\text{Sim}(p, q) = \text{Sim}(q, p)$

Similarity between binary vectors

- Consider the counts below

M_{00} the number of attributes where p is 0 and q is 0

M_{01} the number of attributes where p is 0 and q is 1

M_{10} the number of attributes where p is 1 and q is 0

M_{11} the number of attributes where p is 1 and q is 1

- Simple Matching Coefficient

$$\text{SMC} = \frac{\text{number of matches}}{\text{number of attributes}} = \frac{M_{00} + M_{11}}{M_{00} + M_{01} + M_{10} + M_{11}}$$

- Jaccard Coefficient

$$\text{JC} = \frac{\text{number of 11 matches}}{\text{number of non-both-zero attributes}} = \frac{M_{11}}{M_{01} + M_{10} + M_{11}}$$

Cosine similarity

- It is the cosine of the angle between two vectors

$$\cos(p, q) = \frac{p \cdot q}{\|p\| \|q\|}$$

- Example

$$p = 3 \ 2 \ 0 \ 5 \ 0 \ 0 \ 0 \ 2 \ 0 \ 0$$

$$q = 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 2$$

$$p \cdot q = 3 * 1 + 2 * 0 + 0 * 0 + 5 * 0 + 0 * 0 + 0 * 0 + 0 * 0 + 2 * 1 + 0 * 0 + 0 * 2 = 5$$

$$\|p\| = \sqrt{3 * 3 + 2 * 2 + 0 * 0 + 5 * 5 + 0 * 0 + 0 * 0 + 0 * 0 + 2 * 2 + 0 * 0 + 0 * 0} = 6.481$$

$$\|q\| = \sqrt{1 * 1 + 0 * 0 + 0 * 0 + 0 * 0 + 0 * 0 + 0 * 0 + 0 * 0 + 1 * 1 + 0 * 0 + 2 * 2} = 2.245$$

$$\cos(p, q) = .3150$$

Extended Jaccard Coefficient (Tanimoto)

- Variation of Jaccard for continuous or count attributes
 - reduces to Jaccard for binary attributes

$$T(p, q) = \frac{p \cdot q}{\|p\|^2 + \|q\|^2 - p \cdot q}$$

Choose the right proximity measure

It depends on data

- Dense, continuous
 - a **metric** measure, such as the euclidean distance
- Sparse, asymmetric data
 - cosine, jaccard, extended jaccard

Correlation of quantitative data (Pearson's)

Measure of the linear relationship between a pair of attributes

- Standardize the values
- For two given attributes p and q , consider as vectors the ordered lists of the values over all the data records
- Compute the dot product of the vectors

$$\mathbf{p} = [p_1, \dots, p_N] \xrightarrow{\text{standardize}} \mathbf{p}'$$

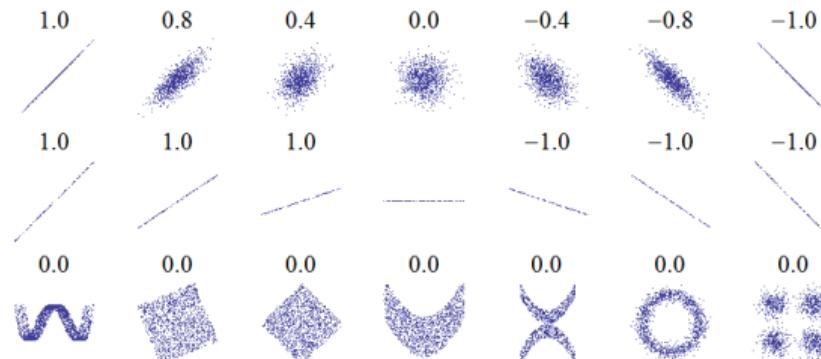
$$\mathbf{q} = [q_1, \dots, q_N] \xrightarrow{\text{standardize}} \mathbf{q}'$$

$$\text{corr}(p, q) = \mathbf{p}' \bullet \mathbf{q}'$$

There is an alternative definition based on covariance and variances

Correlation – Discussion

- Independent variables \Rightarrow correlation is zero
 - the inverse is not valid *in general*
- Correlation zero \Rightarrow absence of *linear relationship* between the variables
- Positive values imply positive linear relationship



From “Correlation and Dependence” on Wikipedia

Correlation between nominal attributes

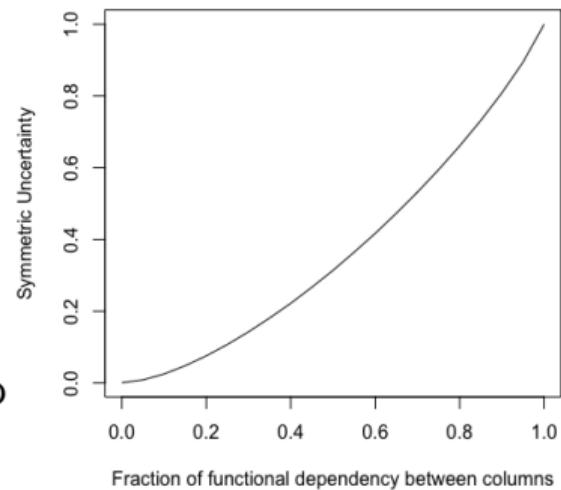
Symmetric Uncertainty [Witten et al.(2011) Witten, Frank, and Hall]

$$U(p, q) = 2 \frac{H(p) + H(q) - H(p, q)}{H(p) + H(q)}$$

- where $H()$ is the entropy of a single attribute while $H(,)$ is the joint entropy, computed from the joint probabilities
- is always between 0 and 1

Correlation between nominal attributes – Experiment

- Behavior of SU for two independent uniformly distributed discrete attributes, say p and q
 - in a variable fraction of records the value of p is copied to q
- from complete independence (left) to complete biunivocal correspondence (right)
- when there is independence, the joint entropy is the sum of the individual entropies, and SU is zero
- when there is complete correspondence, the individual entropies and the joint entropy are equal and SU is one



1	Pre-processing	2
2	Data type conversions	22
3	Similarity and dissimilarity	32
4	Data transformations	53
	● Attribute transformation	56
	● Distance-based algorithms	59
5	Imbalanced data in classification	65

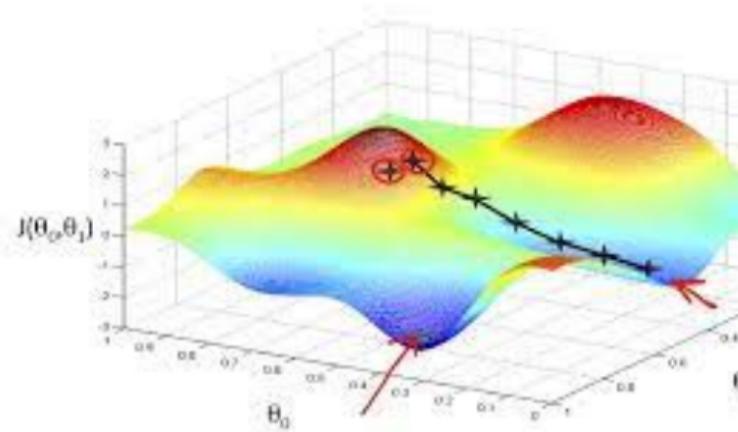
Why Data Transformation

- the features may have different scales
 - this can alterate the results of many learning techniques
 - some machine learning algorithms are sensitive to feature scaling while others are virtually invariant to it
- there can be outliers

Gradient descent

Machine learning algorithms that use *gradient descent* as an optimization technique require data to be scaled

- e.g. linear regression, logistic regression, neural network, etc.
- The presence of feature value X in the formula will affect the step size of the gradient descent
- The difference in ranges of features will cause different step sizes for each feature.
- Similar ranges of the various features ensure that the gradient descent moves smoothly towards the minima and that the steps for gradient descent are updated at the same rate for all the features



Attribute transformation

- Map the entire set of values to a new set according to a function
 - $x^k, \log(x), e^x, |x|$
 - in general they change the *distribution of values*
- Standardization: $x \rightarrow \frac{x-\mu}{\sigma}$
 - if the original values have a *gaussian* distribution, the transformed values will have a *standard gaussian* distribution ($\mu = 0, \sigma = 1$)
 - translation and shrinking/stretching, no change in distribution
- MinMax scaling (a.k.a. Rescaling): the domains are mapped to standard ranges

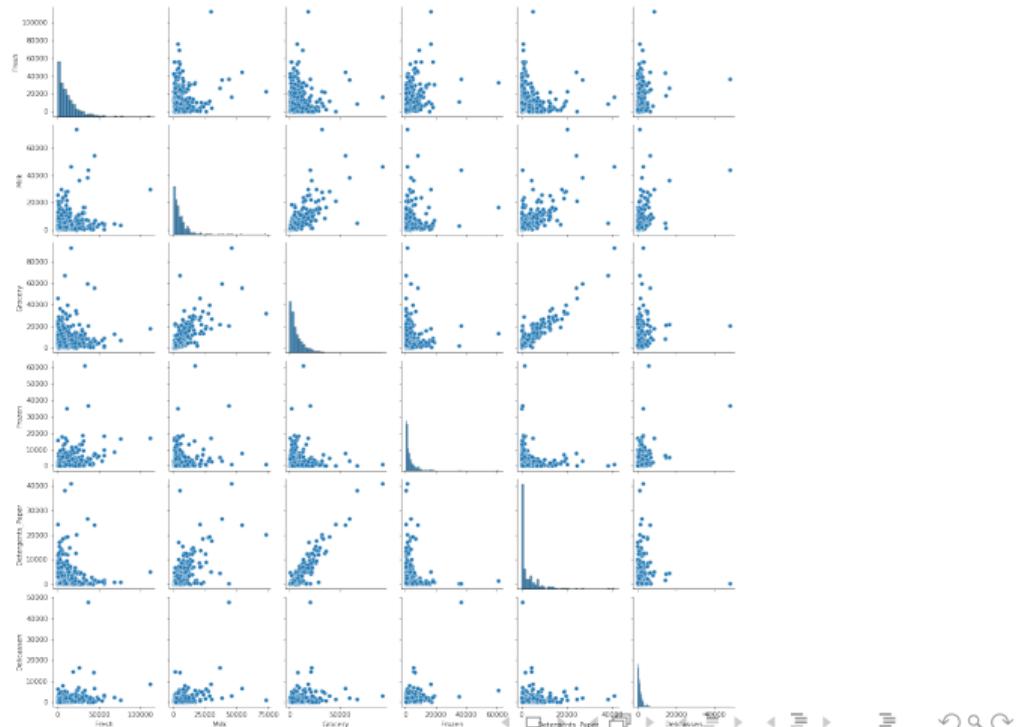
$$x \rightarrow \frac{x - x_{min}}{x_{max} - x_{min}} \quad (0 \text{ to } 1)$$

$$x \rightarrow \frac{x - \frac{x_{max} + x_{min}}{2}}{\frac{x_{max} - x_{min}}{2}} \quad (-1 \text{ to } 1)$$

- translation and shrinking/stretching, no change in distribution

Attribute transformation – Example I

Data with skewed distribution

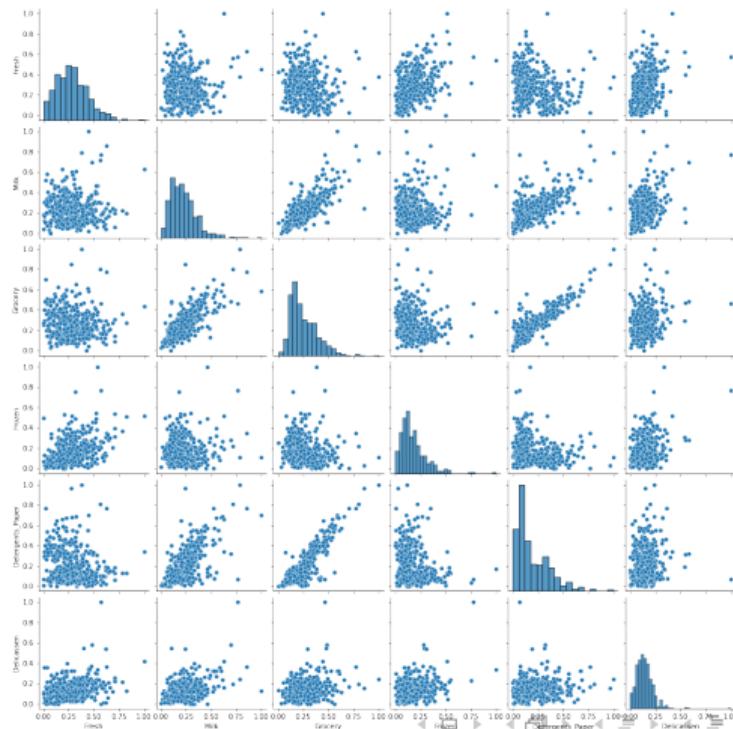


Attribute transformation – Example II

Python code

```
from sklearn.preprocessing  
      import PowerTransformer  
pt = PowerTransformer(method='box-cox')  
X = pd.DataFrame(pt.fit_transform(X0)  
                  , columns = X0.columns)
```

After the transformation the data are
less skewed



Distance-based algorithms

- KNN, K-Means, SVM, ...
- distances between points are used to determine their similarity

Example

Original data		
Student	CGPA	Salary
A	3	60
B	3	40
C	4	40
D	4.5	50
E	4.2	52

Scaled data		
Student	CGPA	Salary
A	-1.18431	1.520013
B	-1.18431	-1.100699
C	0.41612	-1.100699
D	1.21635	0.209657
E	0.736212	0.471728

Distances before and after scaling

$$\text{distance}(A, B) = \sqrt{(40 - 60)^2 + (3 - 3)^2} = 20$$

$$\text{distance}(B, C) = \sqrt{(40 - 40)^2 + (4 - 3)^2} = 1$$

$$\text{distance}(A_s, B_s) = \sqrt{(1.1 + 1.5)^2 + (1.18 - 1.18)^2} = 2.6$$

$$\text{distance}(B_s, C_s) = \sqrt{(1.1 - 1.1)^2 + (0.41 + 1.18)^2} = 1.59$$

Before the scaling the two distances seemed to be very different, due to the a big numeric difference in the Salary attribute, now they are comparable

Range-based scaling and standardization

operate on single features

- Range-based scaling stretches/shrinks and translates the range, according to the range of the feature (there are some variants)
 - good when we know that the data are not gaussian, or we do not make any assumption on the distribution
 - the base variant, the MinMax scaler, remaps to 0, 1
- Standardization subtracts the mean and divides by the standard deviation
 - the resulting distribution has mean zero and *unitary* standard deviation
 - good when the distribution is gaussian
 - StandardScaler

Range-based scalers in Scikit-Learn

affine transformations: linear transformation plus translation

- MinMaxScaler – remaps the feature to $[0, 1]$
- RobustScaler – centering and scaling statistics is based on percentiles
 - not influenced by a few number of very large marginal outliers
 - the resulting range of the transformed feature values is larger than the one given by MinMaxScaler and StandardScaler

Normalization

- **Normalization** is mentioned sometimes with different meanings
 - frequently it refers to MinMaxScaler
- in Scikit-learn the Normalizer normalizes each data row to **unit norm**

Workflow

1. transform the features as required both for the train and test data
2. fit and optimize the model(s)
3. test
4. possibly, use the original data to plot relevant views (e.g. to plot cluster assignments)

1	Pre-processing	2
2	Data type conversions	22
3	Similarity and dissimilarity	32
4	Data transformations	53
5	Imbalanced data in classification	65

Imbalanced data in classification

- The performance minority class (classes) has little impact on standard performance measures
- The optimised model could be less effective on minority class (classes)
- Some estimators allow to **weight** classes
- Some performance measures allow to take into account the contribution of minority class (classes)

Cost Sensitive learning

Already introduced in *Machine-Learning-03-classification*

- several classifiers have the parameter `class_weight`
- it changes the **cost function** to take into account the imbalancing of classes
- in practice it is equivalent to **oversampling** the minority class, (repeating random examples) in order to produce a **balanced training set**

Undersampling

- Obtains a balanced training set by randomly reducing the number of examples of the majority class
- Obviously part of the knowledge embedded in the training set is dropped out

Oversampling with SMOTE

Synthetic Minority Oversampling Technique

- synthesize new examples from the minority class
- a type of **data augmentation**
- a random example from the minority class is first chosen
- k of the nearest neighbors are found
- a randomly selected neighbor is chosen and a synthetic example is created at a randomly selected point between the two examples in feature space

Workflow for *undersampling/oversampling*

- resample the training set
- fit and optimise the estimator
- test the fitted estimator on the test set (untouched)

Bibliography I

- ▶ Ian H. Witten, Eibe Frank, and Mark Hall.
Data Mining – Practical Machine Learning Tools and Techniques.
Morgan Kaufman, 2011.
ISBN 978-0-12-374856-0.

Machine Learning

Clustering

Claudio Sartori

DISI

Department of Computer Science and Engineering – University of Bologna, Italy

claudio.sartori@unibo.it

1	Introduction to clustering	2
2	K-means	12
3	Evaluation of a clustering scheme	48
4	Hierarchical clustering	73
5	Density based clustering	97
6	Model based clustering	117
7	Final remarks	126

The problem of clustering

For a comprehensive review see [Jain et al.(1999) Jain, Murty, and Flynn]

- **Given** – a set of N objects x_i , each described by D values x_{id}
- **Task** – find a *natural* partitioning in K clusters and, possibly, a number of *noise* objects
- **Result** – a *clustering scheme*, i.e. a function mapping each data object to the sequence $[1 \dots K]$ (or to noise)

Desired property of clusters

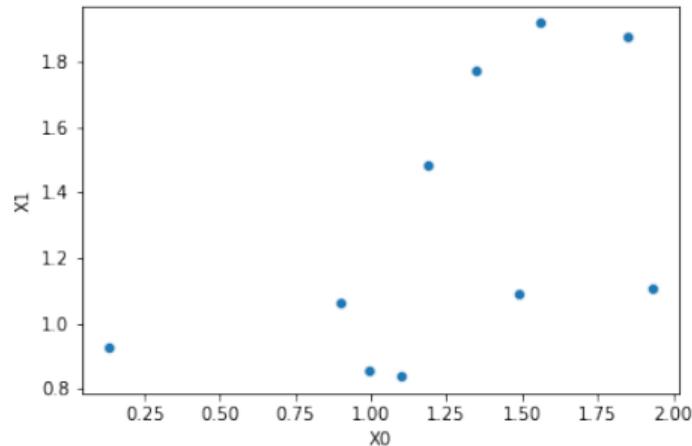
- **objects in the same cluster are similar**
 - look for a clustering scheme which maximizes intra-cluster similarity

A little bit of formality

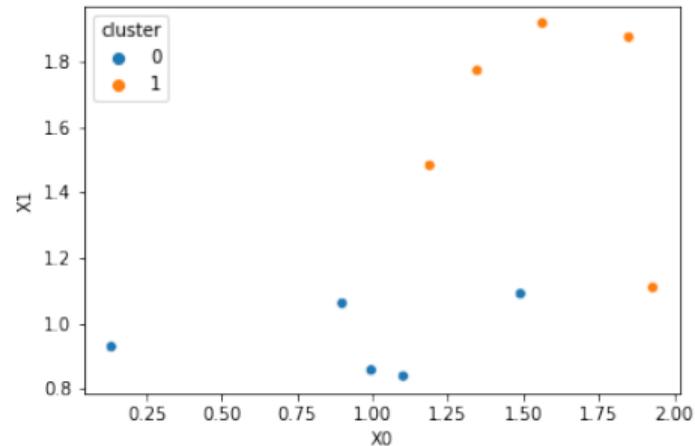
- find a function $clust()$ from \mathcal{X} to $1..K$ such that:
 - $\forall x_1, x_2 \in \mathcal{X}$, $clust(x_1) = clust(x_2)$ when x_1 and x_2 are *similar*
 - $\forall x_1, x_2 \in \mathcal{X}$, $clust(x_1) \neq clust(x_2)$ when x_1 and x_2 are *not similar*

Clustering

2-d Data



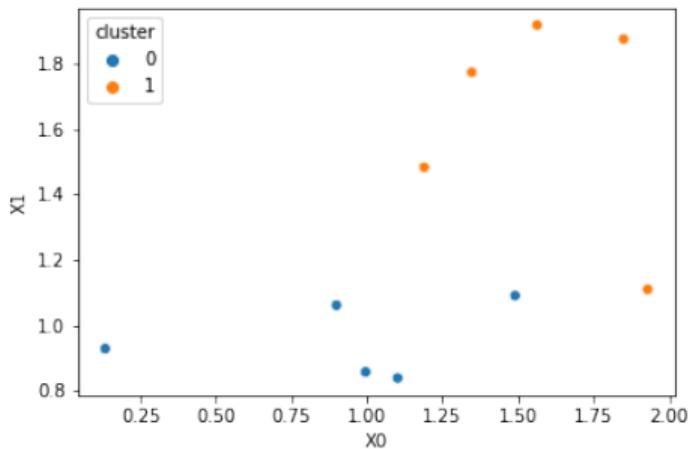
2-d Data clustered



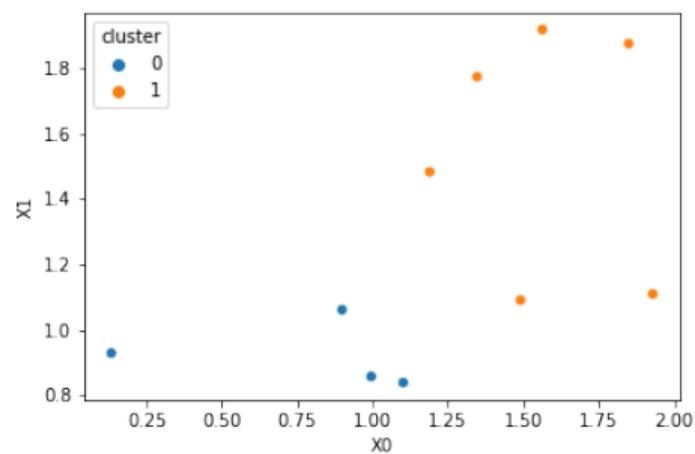
The clustering function maps points to clusters

Clustering - two different clustering functions

$clust^a$



$clust^b$



Which of the two is better, i.e. maximises intra-cluster similarity?
A measure is needed

Ideas for a measure?

- the sum of the distances between a point and the others in the same cluster?
 -
- the average of the distances between a point and the others in the same cluster?
 -
- the sum of the squared distances?
 -
- we could choose a point which, in some sense, **represents** the points of the cluster
 -
- ...
 -

Ideas for a measure? Discussion

- the sum of the distances between a point and the others in the same cluster?
 - bigger for bigger clusters
- the average of the distances between a point and the others in the same cluster?
 - not influenced by cluster size
- the sum of the squared distances?
 - more penalisation for **sparsity**
- we could choose a point which, in some sense, **represents** the points of the cluster
 - what about the **average of the coordinates?**
- ...

Centroid

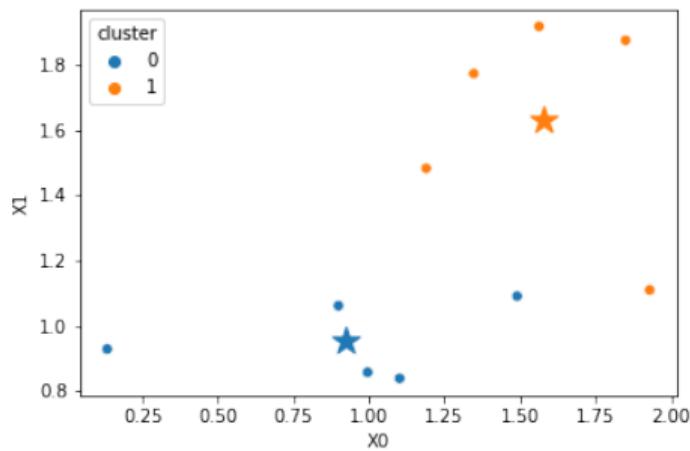
- a point with coordinates computed as the average of the coordinates of all the points in the cluster
- in physics it is the **center of gravity** of a set of points of equal mass
- for each cluster k and dimension d , the d coordinate of the **centroid** is

$$\text{centroid}_d^k = \frac{1}{|x_i : \text{clust}(x_i) = k|} \sum_{x_i : \text{clust}(x_i) = k} x_{id}$$

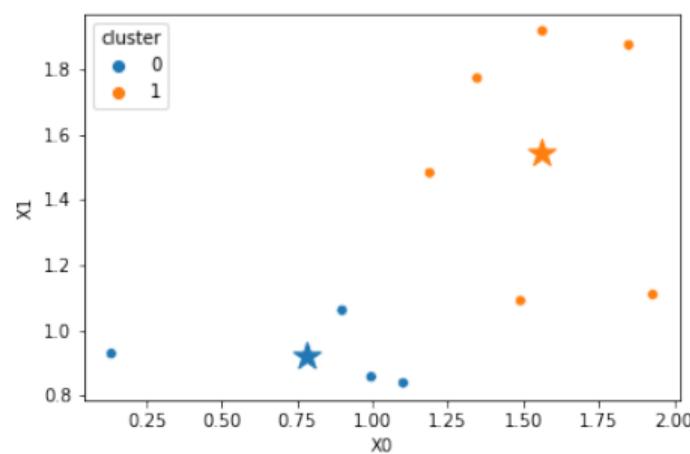
Clustering - two different clustering functions

The stars represent the *centroids*

$clust^a$



$clust^b$



Which of the two is better, i.e. maximises intra-cluster similarity?
A measure is needed

Taxonomy of the clustering methods

- Partitioning
 - K-means (MacQueen 67), expectation maximization (Lauritzen 95), CLARANS (Ng and Han 94)

- Hierarchic
 - agglomerative/divisive, BIRCH (Zhang et al 96), CURE (Guha et al 98)

- Based on linkage
- Based on density

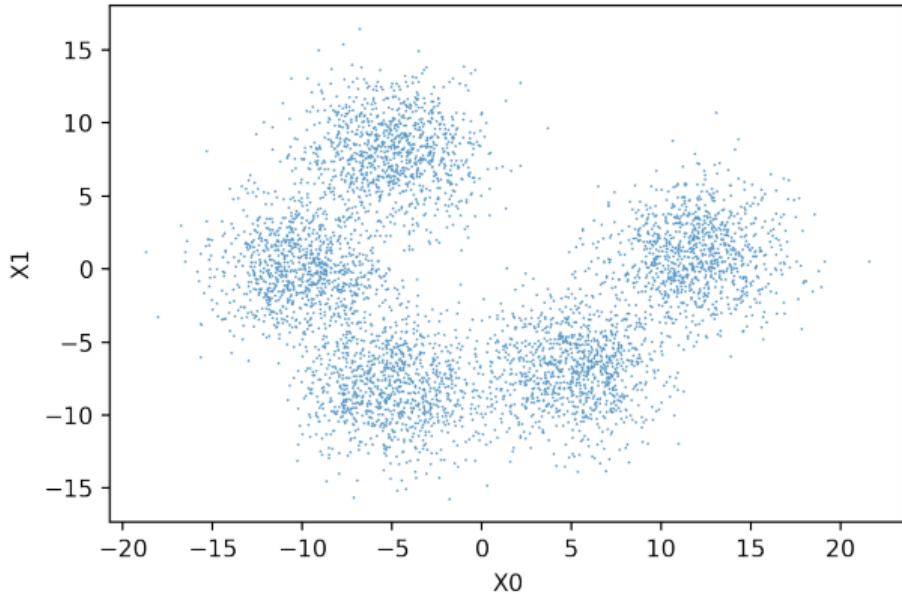
- DBSCAN (Ester et al 96), DENCLUE (Hinnenburg and Keim 98)

- Statistics
 - IBM-IM demographic clustering, COBWEB (Fisher 87), Autoclass (Cheeseman 96)

1	Introduction to clustering	2
2	K-means	12
	● Minimize distortion	29
	● Issues about K-means	35
3	Evaluation of a clustering scheme	48
4	Hierarchical clustering	73
5	Density based clustering	97
6	Model based clustering	117
7	Final remarks	126

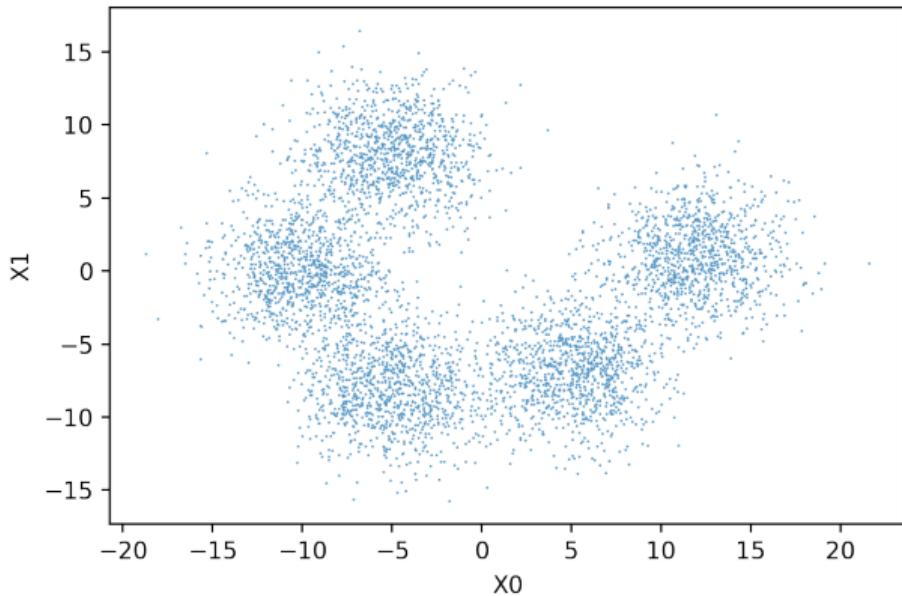
Some data

- Could be modeled as a five component gaussian mixture
 - ... *statistician voice*...
- How do we guess the number five in a D -dimensional space (with $D \geq 2$)?



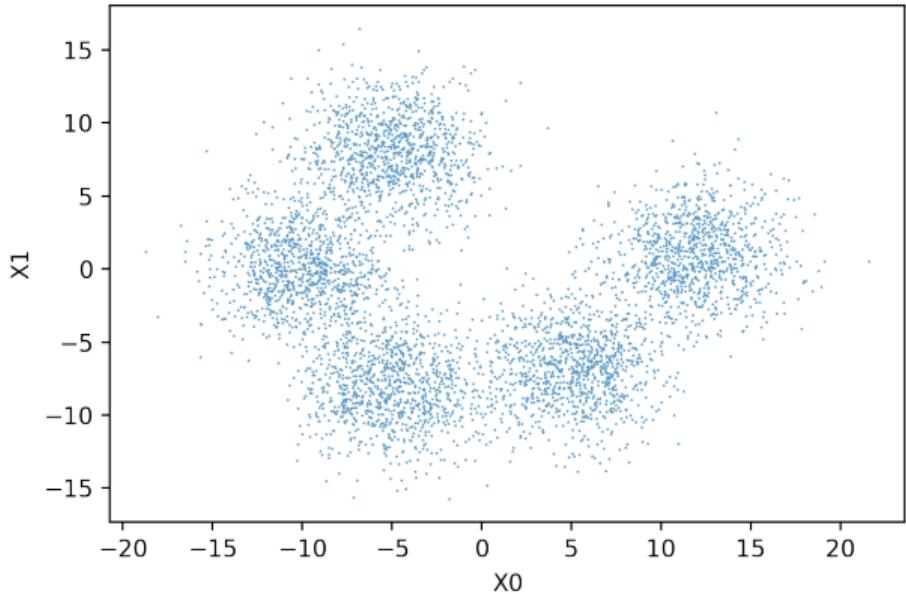
Transmission

- Transmit the coordinates of points
- Allow only two bits per point
 - the transmission will be **lossy**
- Need a coding/decoding mechanism



How much loss?

- Sum of the squared errors between the real points and their encoding/decoding
- Which encoding/decoding minimizes the loss?

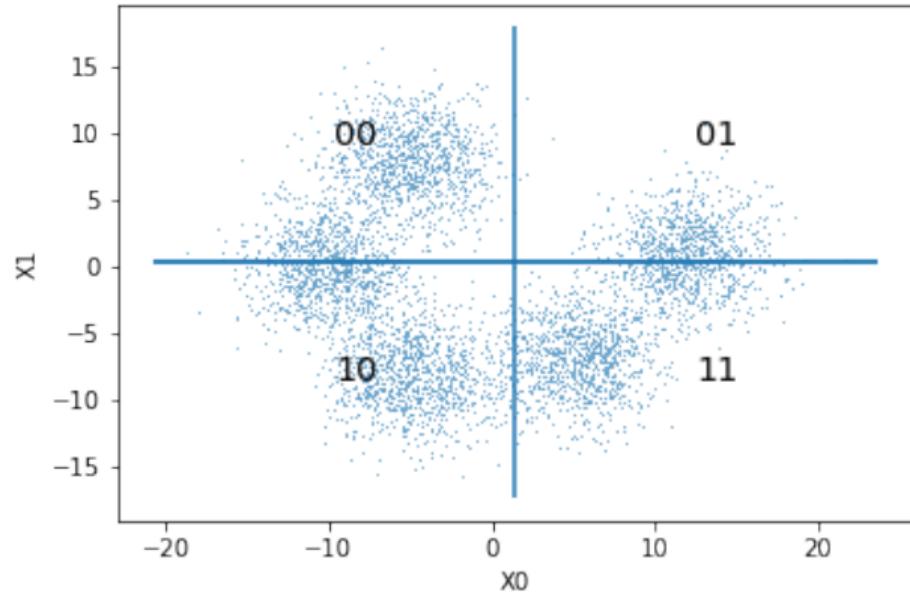


How much loss?

- Sum of the squared errors between the real points and their encoding/decoding

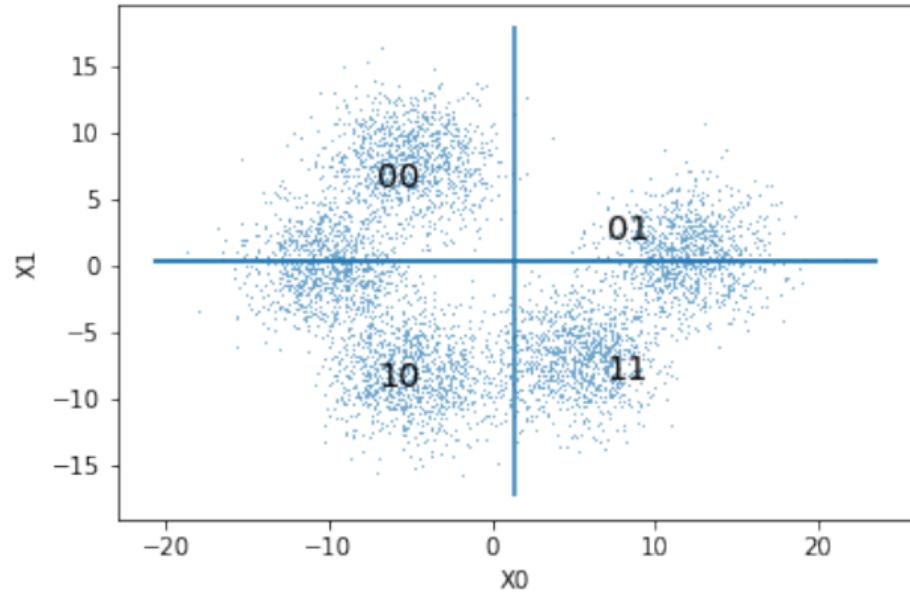
First idea

- partition the space into a grid of cells
- decode each pair of bits with the center of the grid cell



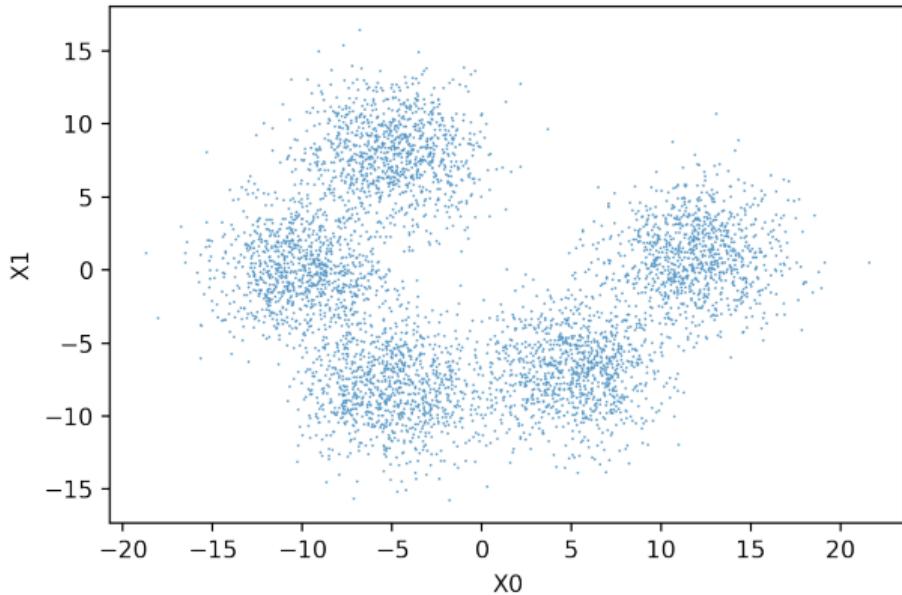
Improvement

- partition the space into a grid of cells
- decode each pair of bits with the centroid of the points in the grid cell



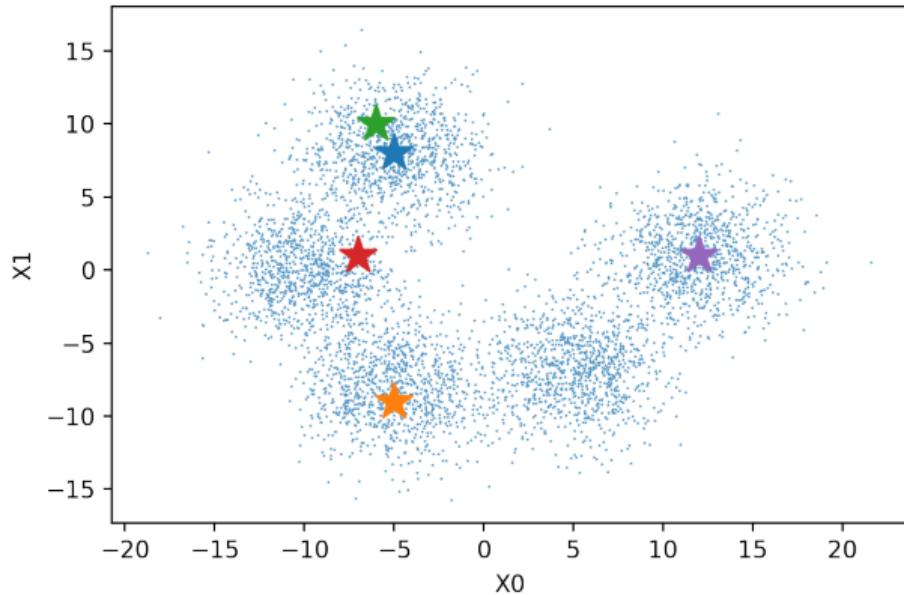
K-means

1. Ask the user the number of clusters K
1.1 $\Rightarrow 5$



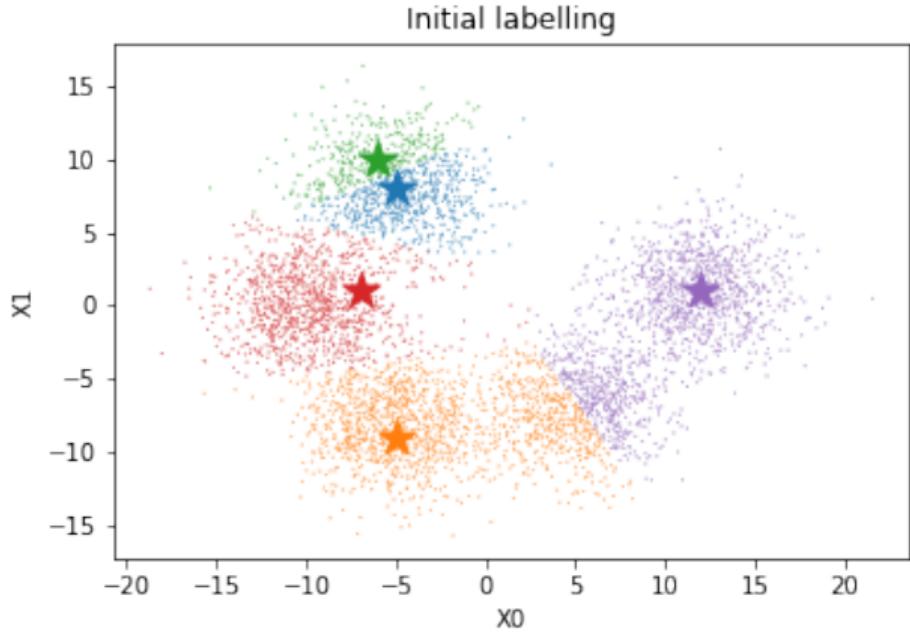
K-means

1. Ask the user the number of clusters K
2. Random choice of K points as **temporary centers**



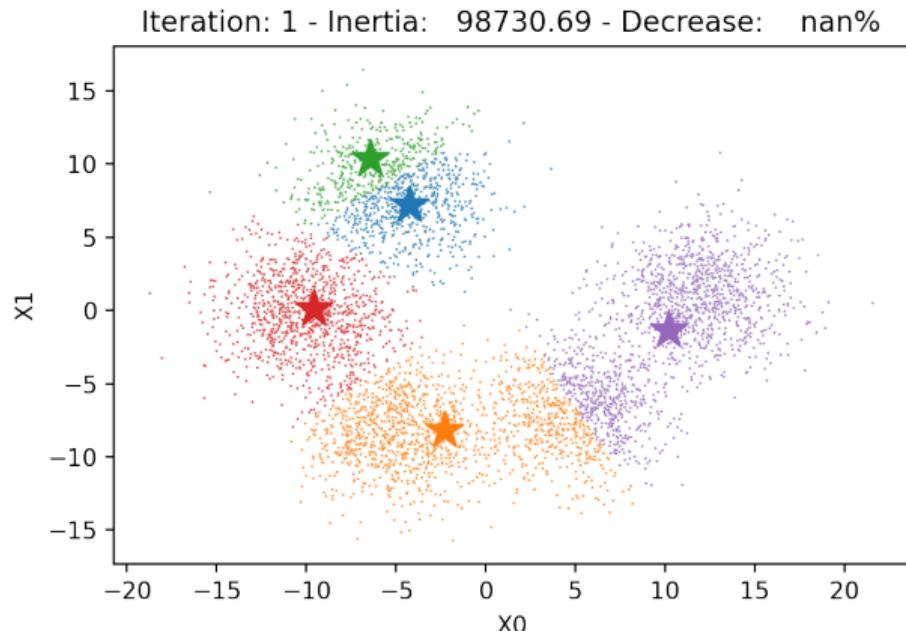
K-means

1. Ask the user the number of clusters K
2. Random choice of K points as **temporary centers**
3. Each point finds his nearest center and is labelled (i.e. colored) accordingly

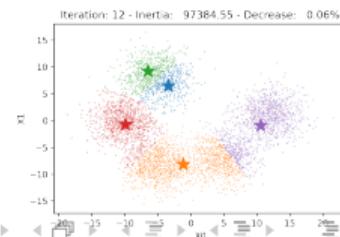
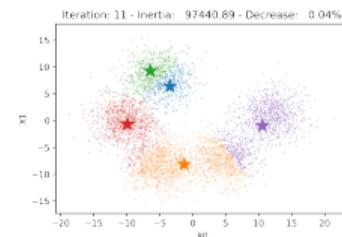
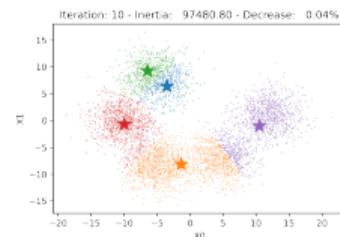
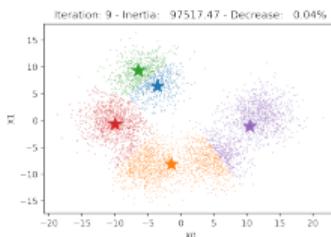
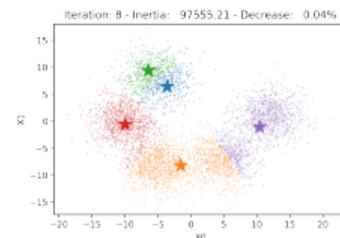
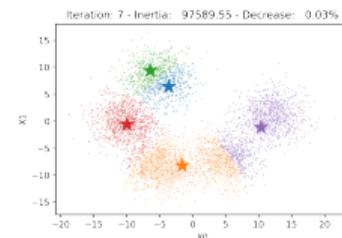
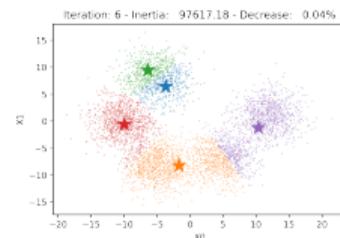
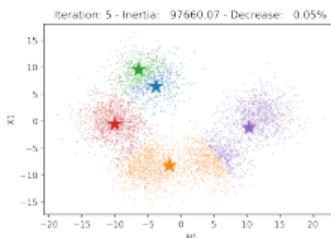
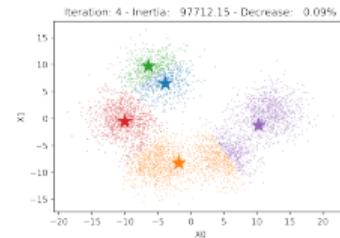
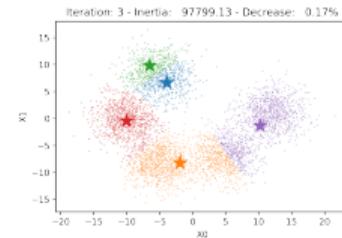
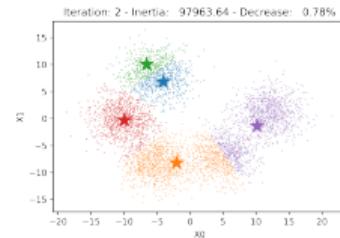
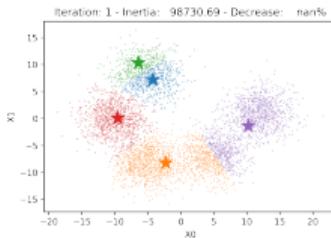


K-means

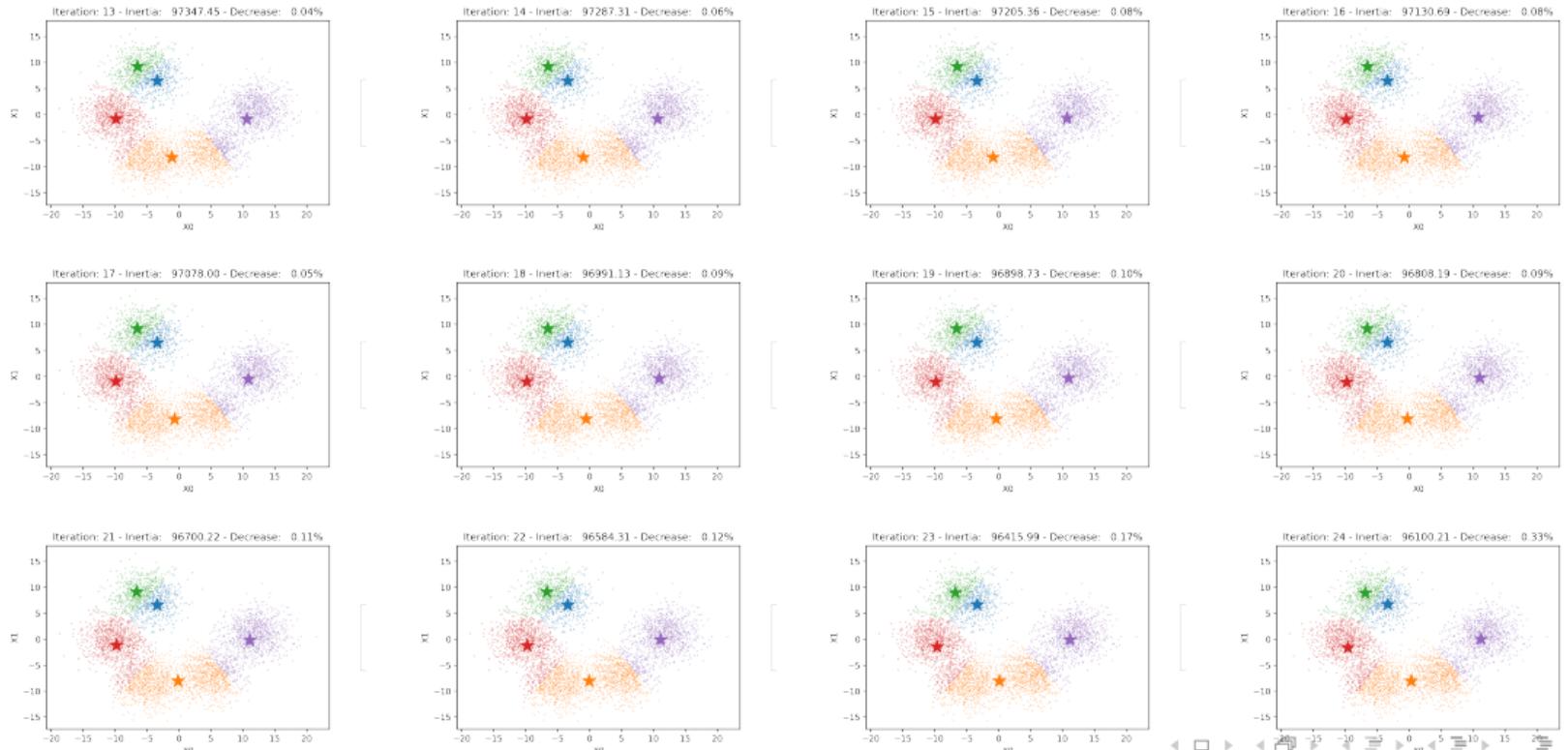
1. Ask the user the number of clusters K
2. Random choice of K points as **temporary centers**
3. Each point finds his nearest center
4. for each center finds the centroid of its points ...
5. ... and move there the center



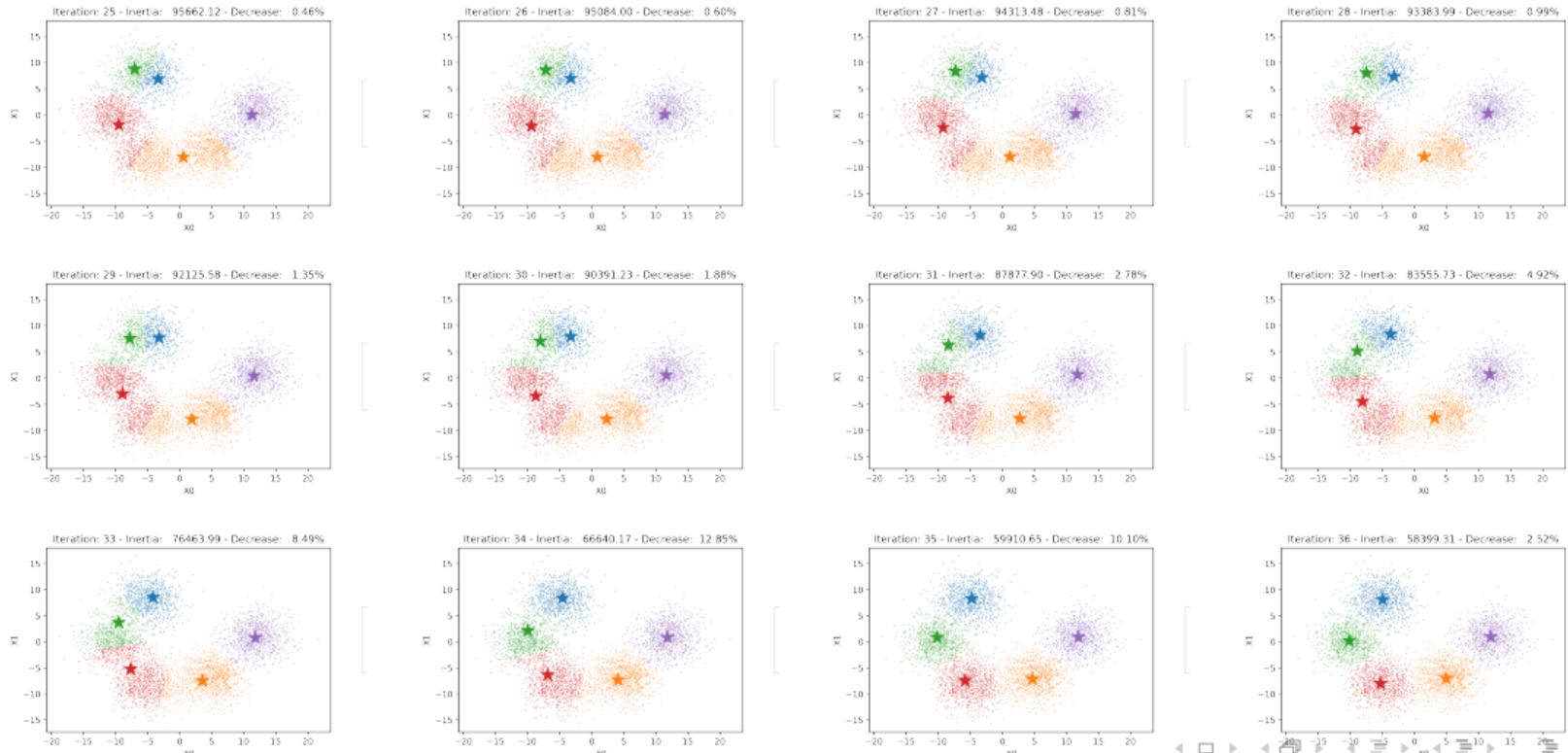
KMeans working . . .



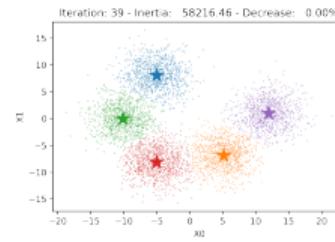
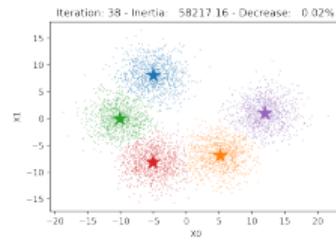
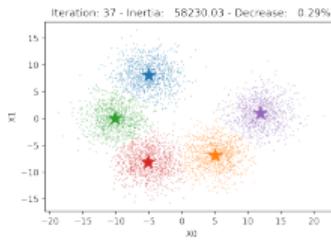
KMeans working . . .



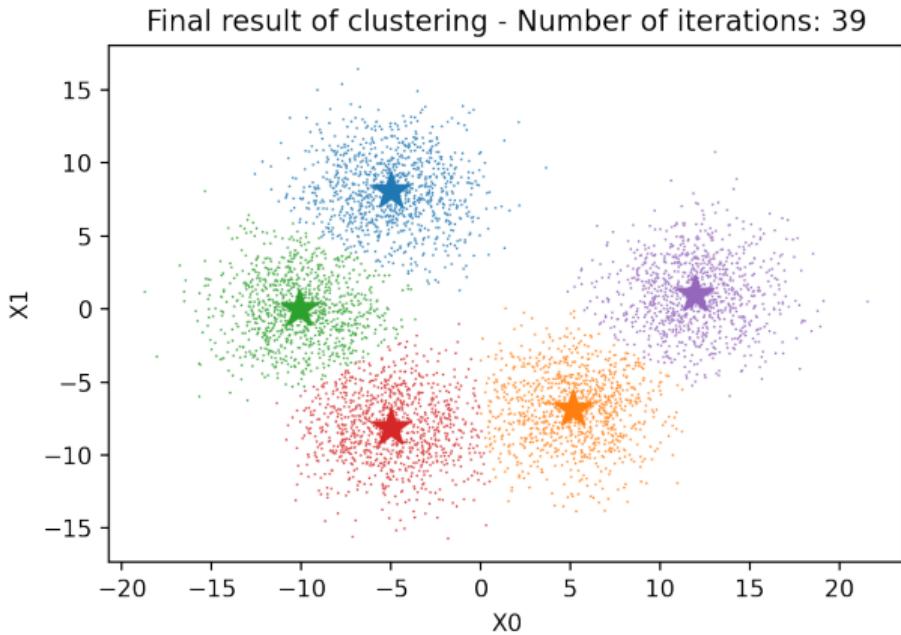
KMeans working . . .



KMeans working . . .



K-means ends



Questions

1. What are we trying to optimize?
2. Is termination guaranteed?
3. Are we sure that the best clustering scheme is found?
 - 3.1 Which is the definition of best clustering scheme?
4. How should we start?
5. How can we find the number of clusters?

Question 1: Distortion

Frequently called in the literature **Inertia**

Given:

- a dataset $\{x_i, i = 1 \dots N\}$
- a coding function $\text{Encode} : \{R\}^D \rightarrow [1..K]$
- a decoding function $\text{Decode} : [1..K] \rightarrow \mathbb{R}^D$
- define $\text{Distortion} = \sum_{i=1}^N (x_i - \text{Decode}(\text{Encode}(x_i)))^2$
- shortcut $\text{Decode}(k) = \mathbf{c}_k$

then

$$\text{Distortion} = \sum_{i=1}^N (x_i - \mathbf{c}_{\text{Encode}(x_i)})^2$$

Minimal distortion I

$$\text{Distortion} = \sum_{i=1}^N (x_i - \mathbf{c}_{\text{Encode}(x_i)})^2$$

Which properties are requested to $\mathbf{c}_1, \dots, \mathbf{c}_K$ for the minimal distortion?

1. x_i must be encoded with the nearest center

Why?

Because otherwise the distortion could be reduced by substituting $\text{Encode}(x_i)$ with the nearest center

$$\mathbf{c}_{\text{Encode}(x_i)} = \operatorname*{argmin}_{\mathbf{c}_j \in \{\mathbf{c}_1, \dots, \mathbf{c}_K\}} (x_i - \mathbf{c}_j)^2$$

Minimal distortion II

$$\text{Distortion} = \sum_{i=1}^N (x_i - \mathbf{c}_{\text{Encode}(x_i)})^2$$

Which properties are requested to $\mathbf{c}_1, \dots, \mathbf{c}_K$ for the minimal distortion?

2. The partial derivative of distortion w.r.t. the position of each center must be zero

Why?

Because in that case the function has either a maximum or a minimum

Step 2. The partial derivative of distortion w.r.t. the position of each center must be zero

$$\begin{aligned}\text{Distortion} &= \sum_{i=1}^N (x_i - \mathbf{c}_{\text{Encode}(x_i)})^2 \\ &= \sum_{j=1}^K \sum_{i \in \text{OwnedBy}(\mathbf{c}_j)} (x_i - \mathbf{c}_j)^2\end{aligned}$$

$$\begin{aligned}\frac{\partial \text{Distortion}}{\partial \mathbf{c}_j} &= \frac{\partial}{\partial \mathbf{c}_j} \sum_{i \in \text{OwnedBy}(\mathbf{c}_j)} (x_i - \mathbf{c}_j)^2 \\ &= -2 \sum_{i \in \text{OwnedBy}(\mathbf{c}_j)} (x_i - \mathbf{c}_j)\end{aligned}$$

2. The partial derivative of distortion w.r.t. the position of each center must be zero

When distortion is minimal

$$\mathbf{c}_j = \frac{1}{|\text{OwnedBy}(\mathbf{c}_j)|} \sum_{i \in \text{OwnedBy}(\mathbf{c}_j)} x_i$$

Minimal distortion III

$$\text{Distortion} = \sum_{i=1}^N (x_i - \mathbf{c}_{\text{Encode}(x_i)})^2$$

Which properties are requested to $\mathbf{c}_1, \dots, \mathbf{c}_K$ for the minimal distortion?

1. x_i must be encoded with the nearest center
2. each center must be the **centroid** of the points it owns

Algorithm: Improving a sub-optimal solution

$$\text{Distortion} = \sum_{i=1}^N (x_i - \mathbf{c}_{\text{Encode}(x_i)})^2$$

Which properties are requested to $\mathbf{c}_1, \dots, \mathbf{c}_K$ for the minimal distortion?

1. x_i must be encoded with the nearest center
2. each center must be the **centroid** of the points it owns

⇒ Alternately perform steps 1 and 2

It can be proven that after a finite number of steps the system reaches a state where neither of the two operations changes the state

Why?

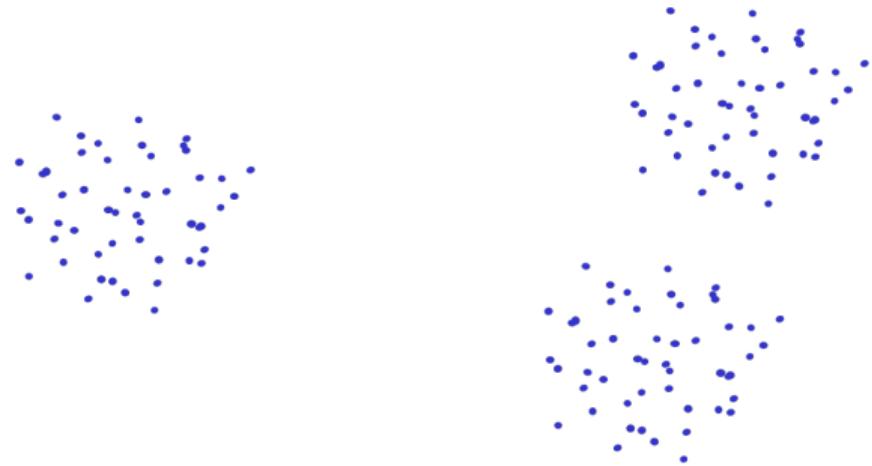
Question 2: Algorithm termination

- There is only a finite number of ways to partition N objects into K groups
- The state of the algorithm is given by the two encode/decode functions
- The number of configurations where all the centers are the centroids of the points they own is **finite**
- If after one iteration the state changes, the distortion is **reduced**
- Therefore each change of state bring to a state which was never visited before
- In summary, sooner or later the algorithm will stop because there are no new states reachable

Question 3: Local or global minimum?

Is the ending state the best possible?

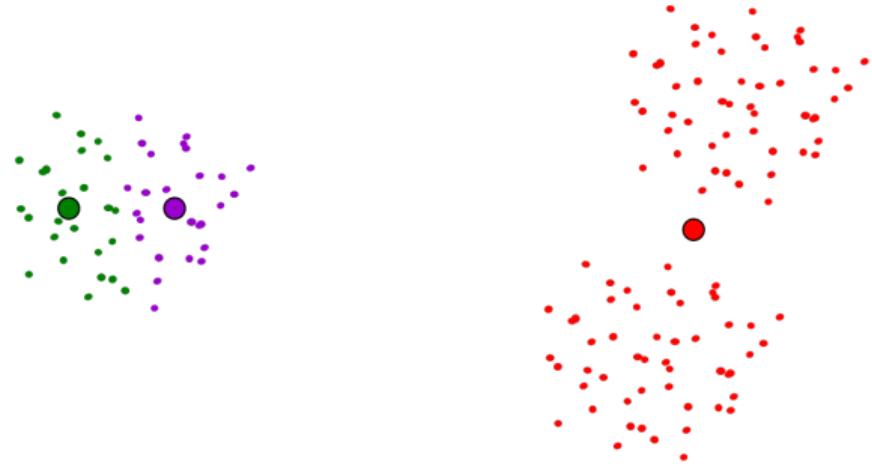
- Not necessarily
- An example



Question 3: Local or global minimum?

Is the ending state the best possible?

- Not necessarily
- An example



Question 4: Looking for a good ending state

- The starting point is important
 - choose randomly the first starting point
 - choose in sequence the $2..K$ starting points as far as possible from the preceding ones
- Re-run the algorithm with different starting points

Question 5: Choose the number of clusters

not so easy...

- try various values
- use a **quantitative evaluation** of the quality of the clustering scheme to decide among the different values
- the best value finds the optimal compromise between the minimization of intra-cluster distances and the maximization of the inter cluster distances

The proximity function

- The most obvious solution, used in the previous formulas is the **euclidean distance**
 - good choice, in general, for vector spaces
- Several alternative solutions for specific data types and data sets
 - see the "Data" module for additional discussions

Sum of Squared Errors I

The official name of the distortion

$$\begin{aligned} \text{SSE} &= \sum_{j=1}^K \sum_{i \in \text{OwnedBy}(\mathbf{c}_j)} (x_i - \mathbf{c}_j)^2 \\ &= \sum_{j=1}^K \text{SSE}_j \end{aligned}$$

Sum of Squared Errors II

- A cluster j with high SSE_j has low quality
- $SSE_j = 0$ if and only if all the points are coincident with the centroid
- SSE decreases for increasing K , is zero when $K = N$
- \Rightarrow minimizing SSE is not a viable solution to choose the best K
 - more discussions on this in the section on “Evaluation of the quality of a clustering scheme”

Empty clusters

- It may happen, at some step, that a centroid does not own any point
- This changes the initial requirement of K clusters
- ⇒ choose a new centroid
 - choose a point far away from the empty centroid or
 - choose as new centroid a random point in the cluster with the maximum SSE
 - in this way the cluster with the lowest quality will be split in two

Outliers

- are points with high distance from their centroid
 - high contribution to SSE
- have a bad influence on the clustering results
 - sometimes it is a good idea to remove them
 - the choice is related to the application domain

Common uses of K-means

- It can be easily used in the beginning, for the exploration of data
- In a one-dimension space it is a good way to discretize the values of a domain in non-uniform buckets
- It is the basis for vector quantization, a classical technique for signal processing and compression
- Used for choosing the color palettes
 - gif compressed images: color quantization

Complexity

Given:

- T number of iterations
- K number of clusters
- N number of data points
- D number of dimensions

the time complexity is

$$\mathcal{O}(TKND)$$

Pros and cons of K-means

- Strong points
 - fairly efficient, nearly linear in the number of data points
 - in general $T, K, D \ll N$
- Weak points
 - in essence it is defined for spaces where the centroid can be computed
 - e.g. when the Euclidean distance is available, also other distance functions work well
 - cannot work with nominal data
 - requires the K parameter
 - nevertheless the best K can be found with iterations
 - it is very sensitive to outliers
 - does not deal with **noise**
 - does not deal properly with **non convex** clusters

1	Introduction to clustering	2
2	K-means	12
3	Evaluation of a clustering scheme	48
	● Cohesion and separation	53
	● Silhouette	59
	● Choice of K	64
	● Supervised measures	68
4	Hierarchical clustering	73
5	Density based clustering	97
6	Model based clustering	117
7	Final remarks	126

Evaluation of a clustering scheme

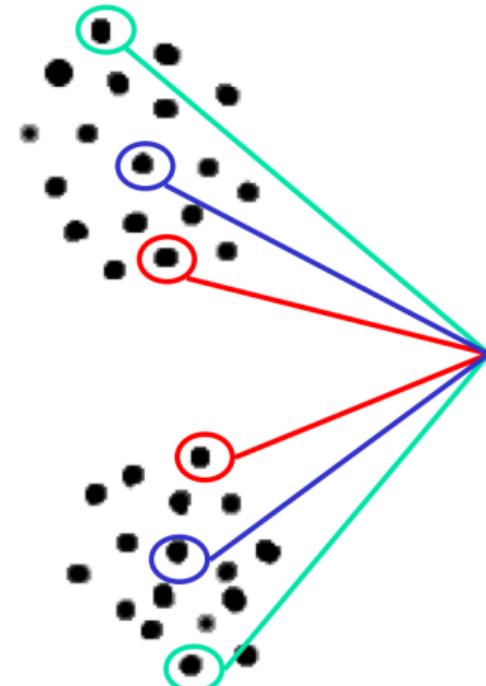
- It is related only to the result, not to the clustering technique
- Clustering is a non supervised method
 - the evaluation is critical, because whenever there is very little apriori information, such as class labels
 - we need one or more **score** function to measure various properties of the clusters and of the clustering scheme as a whole
 - in the literature the words *score* and *index* are considered synonyms in this context
 - if some supervised data are available, they can be used to evaluate the clustering scheme
- In 2D the clusters can be examined visually
- In higher order spaces the 2D projections can help, but in general it is better to use more formal methods

Issues on the evaluation of clustering

- Distinguish patterns from random apparent regularities
- Find the best number of clusters
- Non supervised evaluation
- Supervised evaluation
- Relative comparison of clustering schemes

Measurement criteria

- Cohesion
 - proximity of objects in the same cluster should be high
- Separation between two clusters
 - how to measure it? several choices
 - distance between the **nearest** objects in the two clusters
 - distance between the **most distant objects** in the two clusters
 - distance between the **centroids** of the two clusters

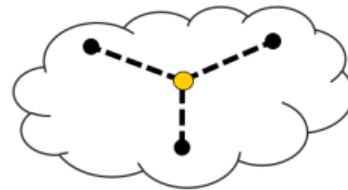


Proximity and others

- Similarity – Proximity
 - a two variable function measuring how much two objects are **similar**, according to the values of their properties
- Dissimilarity
 - a two variable function measuring how much two objects are **different**, according to the values of their properties
 - e.g. the **Euclidean distance**

Cohesion – Prototype based

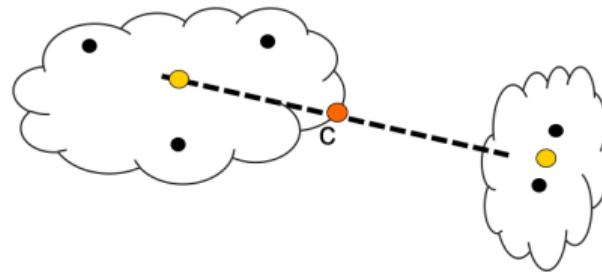
- The sum of the proximities between the elements of the cluster and the geometric center (the prototype)
 - *centroid*
 - a point in the space whose coordinates are the means of the dataset
 - *medoid*
 - an element of the dataset whose average dissimilarity with all the elements of the cluster is minimal
 - not necessarily unique, used in contexts where the mean is not defined, e.g. 3D trajectories or gene expressions



$$\text{Coh}(k_i) = \sum_{x \in k_i} \text{Prox}(x, \mathbf{c}_i)$$

Separation – Prototype based¹

- Separation between two clusters
 - proximity between the prototypes



$$\text{Sep}(k_i, k_j) = \text{Prox}(\mathbf{c}_i, \mathbf{c}_j)$$

¹ It is the blue case in page 51

Global separation of a clustering scheme

SSB – Sum of Squares Between clusters

\mathbf{c} = global centroid of the dataset

$$\text{SSB} = \sum_{i=1}^K N_i \text{Dist}(\mathbf{c}_i, \mathbf{c})^2$$

Link between cohesion and separation – I

- TSS = Total Sum of Squares
 - sum of squared distances of the points from the global centroid
- $TSS = SSE + SSB$
- the total sum of squares is a global property of the dataset, independent from the clustering scheme
- for a given dataset, minimise SSE \Leftrightarrow maximise SSB

Link between cohesion and separation – II

$$\begin{aligned} \text{TSS} &= \sum_{i=1}^K \sum_{x \in k_i} (x - \mathbf{c})^2 = \sum_{i=1}^K \sum_{x \in k_i} ((x - \mathbf{c}_i) - (\mathbf{c} - \mathbf{c}_i))^2 \\ &= \sum_{i=1}^K \sum_{x \in k_i} (x - \mathbf{c}_i)^2 - 2 \sum_{i=1}^K \sum_{x \in k_i} (x - \mathbf{c}_i)(\mathbf{c} - \mathbf{c}_i) + \sum_{i=1}^K \sum_{x \in k_i} (\mathbf{c} - \mathbf{c}_i)^2 \\ &= \sum_{i=1}^K \sum_{x \in k_i} (x - \mathbf{c}_i)^2 + \sum_{i=1}^K \sum_{x \in k_i} (\mathbf{c} - \mathbf{c}_i)^2 \\ &= \sum_{i=1}^K \sum_{x \in k_i} (x - \mathbf{c}_i)^2 + \sum_{i=1}^K |k_i|(\mathbf{c} - \mathbf{c}_i)^2 = \text{SSE} + \text{SSB} \end{aligned}$$

since $\sum_{x \in k_i} (x - \mathbf{c}_i) = 0$ by definition of \mathbf{c}_i

Evaluation of specific clusters and objects

- Each cluster can have its own evaluation
 - the worst clusters can be considered for additional split
- A weakly separated pair of clusters could be considered for merging
- Single objects can give negative contribution to the cohesion of a cluster or to the separation between two clusters
 - border objects

Silhouette score of a cluster – I

Requirements for a clustering quality score

- values are in a standard range, e.g. $-1, 1$
- increases with the separation between clusters
- decreases for clusters with low *cohesion*

$$\begin{aligned} \text{SSE} &= \sum_{j=1}^K \sum_{i \in \text{OwnedBy}(\mathbf{c}_j)} (x_i - \mathbf{c}_j)^2 \\ &= \sum_{j=1}^K \text{SSE}_j \end{aligned}$$

Silhouette score of a cluster – I

Requirements for a clustering quality score

- values are in a standard range, e.g. $-1, 1$
- increases with the separation between clusters
- decreases for clusters with low *cohesion*, or, in other words, with high *sparsity*

Silhouette score of a cluster – II

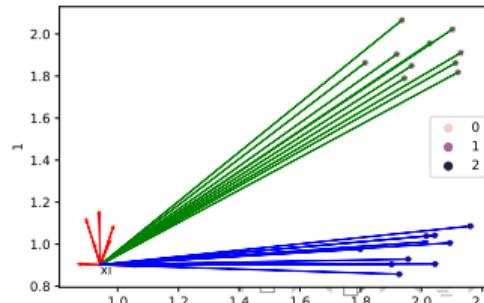
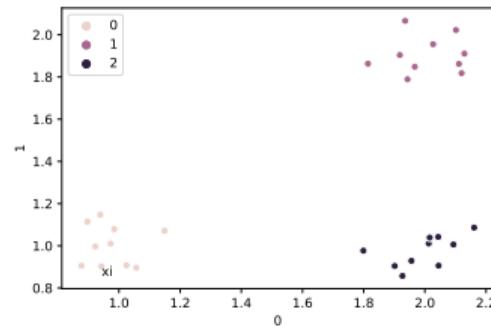
Consider the individual contribution of each object, say x_i

Contribution to cluster **sparsity**: the average of red distances

$$a_i = \text{average } \underset{j, y(x_j) = y(x_i)}{\text{dist}}(x_i, x_j)$$

Contribution to **separation** from other clusters: the minimum of the two averages of green and blue distances

$$b_i = \min_{k \in \mathcal{Y}, k \neq y(x_i)} (\text{average } \underset{j, y(x_j) = k}{\text{dist}}(x_i, x_j))$$



Silhouette score of a cluster – III

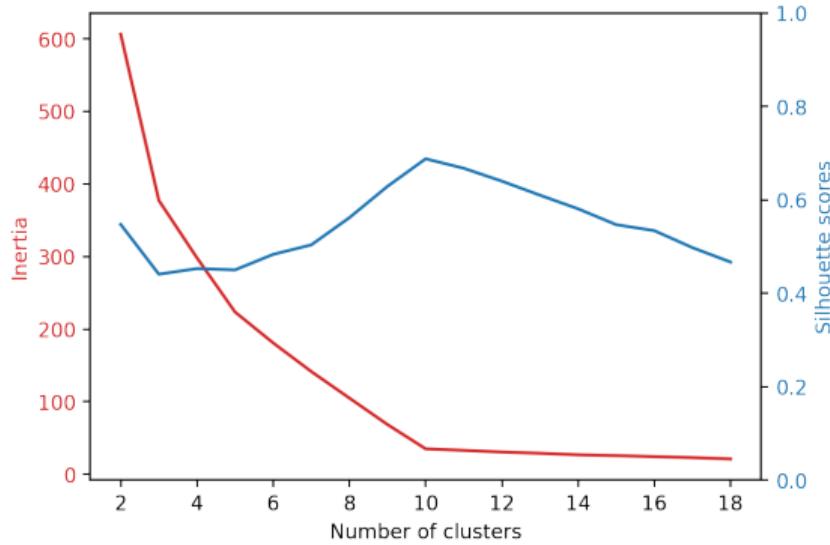
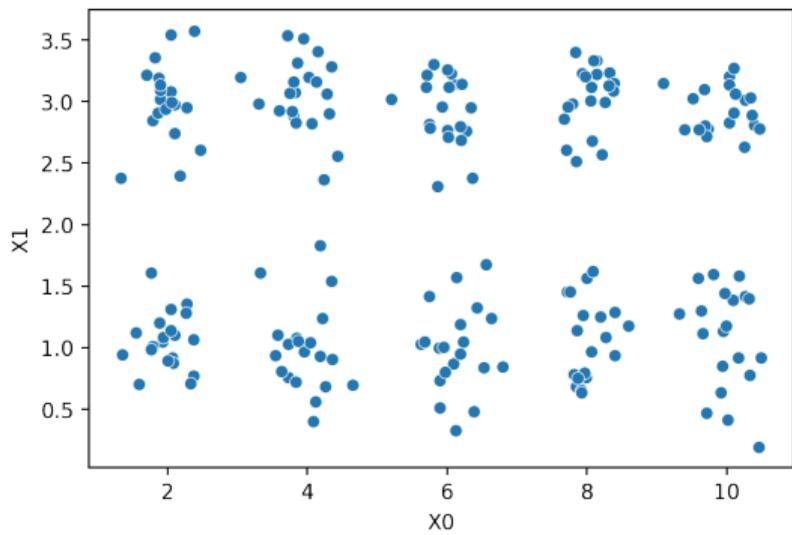
- Silhouette score of x_i

$$s_i = \frac{b_i - a_i}{\max(a_i, b_i)} \in [-1, 1]$$

- For the global score of a cluster/clustering scheme compute the average score over the cluster/dataset
- Intuition
 - when the score is less than zero for an object it means that there is a dominance of objects in other clusters at a distance smaller than objects of the same cluster

Example: Inertia and silhouette scores

Testing K-means with different numbers of clusters



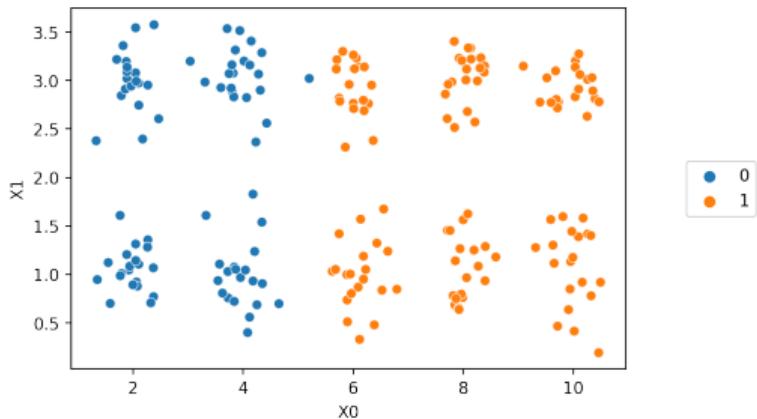
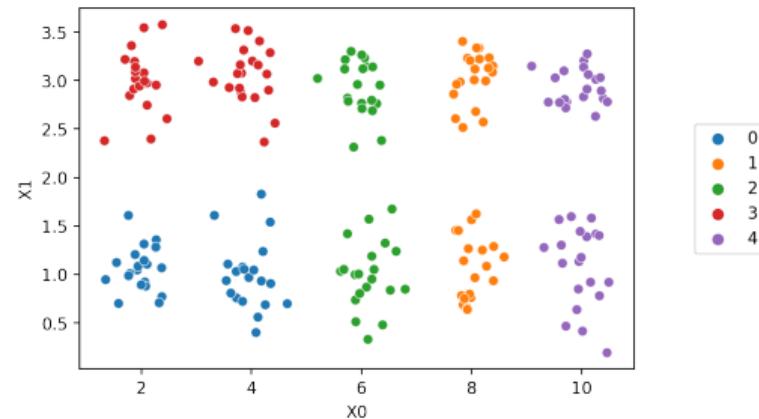
Looking for the best number of clusters – I

- Some algorithms, such as K-means, require the number of clusters as a parameter
- Measures, such as SSE and Silhouette, are obviously influenced by the number of clusters
 - they can be used to optimize K
- Computation of Silhouette score is expensive
- SSE decreases monotonically for increasing K
 - is equal to TSS for $K = 1$
 - goes to zero when $K = N$

Looking for the best number of clusters – II

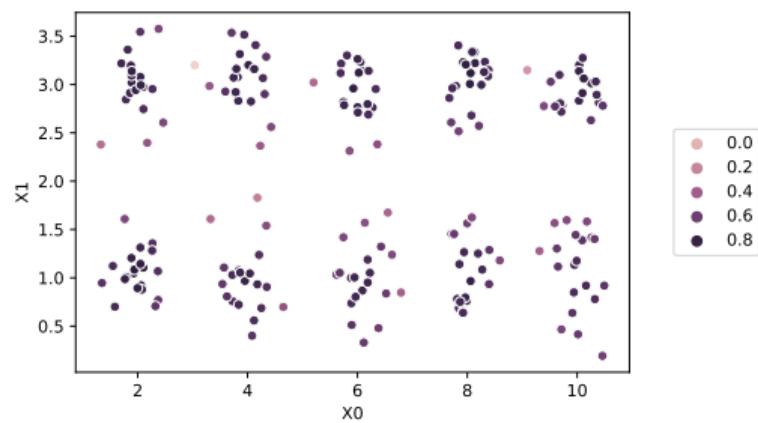
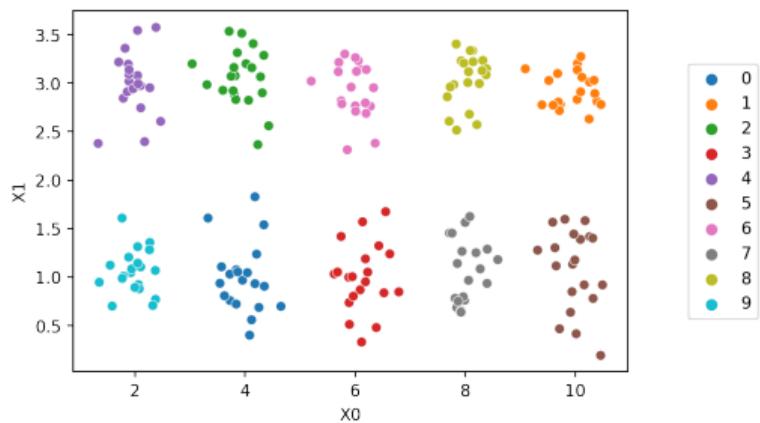
- The *inertia* varying K has frequently one or more points where the **slope decreases**: one of this points is frequently a plausible value for K
 - this is called **elbow method**
- The *silhouette* score varying K has frequently a maximum, in this case it indicates the best value for K

K-means results on the dataset of page 63

 $K = 2$  $K = 5$

K-means results on the dataset of page 63

Best *silhouette* for $K = 10$



Supervised measures: Gold Standard I

- Let be available a partition of a dataset similar to the data to be clustered, which we call **gold standard**, and defined by a labelling scheme $y_g(\cdot)$
 - it is the same as the labels attached to supervised data for training a classifier
- Consider a clustering scheme $y_k(\cdot)$
 - the cardinalities of the sets of distinct labels generated by the two schemes \mathcal{V}_g and \mathcal{V}_k can be different, and also in case of identity of the two grouping schemes, a permutation of labels could be necessary to make them equal

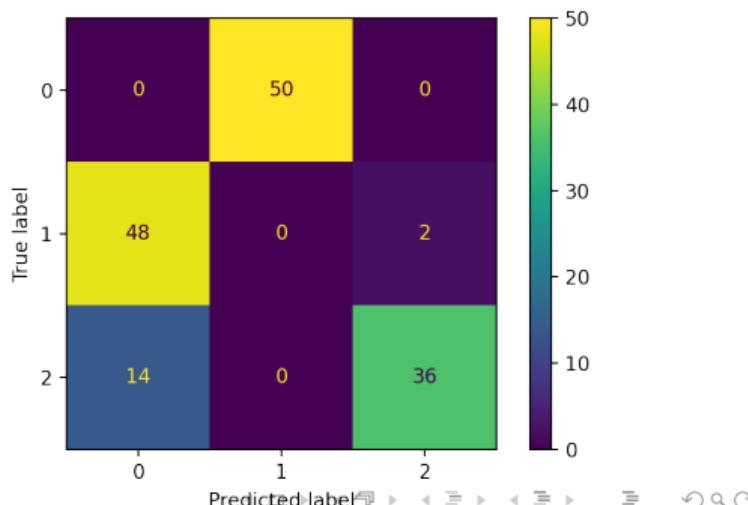
Why should we compare it with the gold standard?

- validate a clustering technique which can be applied later to new, unlabelled data
- the purpose is quite similar to testing a classifier
- the difference is that in this case we are more interested in *grouping* new data than in *labelling them* following the Gold Standard scheme

Classification-oriented measures

- Measure how the gold standard classes are distributed among the clusters
 - confusion matrix, precision, recall, f-measure
- On the right the confusion matrix for the **Iris** dataset
 - best match with permutation¹ of the predicted labels
 $0 \rightarrow 1, 1 \rightarrow 0, 2 \rightarrow 2$

```
X,y = load_iris(return_X_y=True)
estimator = KMeans(n_clusters=3
                    , random_state=363)
y_km = estimator.fit_predict(X)
disp = ConfusionMatrixDisplay(confusion_matrix(y,y_km))
disp.plot()
```



Similarity oriented measures - I

- Analogous to compare binary data
- Consider a clustering scheme $y_k(\cdot)$ and compare it with the **Gold Standard** $y_g(\cdot)$
- Any pair of objects can be labelled as
 - *SGSK* if they belong to the same set in $y_g(\cdot)$ and $y_k(\cdot)$
 - *SGDK* if they belong to the same set in $y_g(\cdot)$ and not in $y_k(\cdot)$
 - *DGSK* if they belong to the same set in $y_k(\cdot)$ but not in $y_g(\cdot)$
 - *DGDK* if they belong to different sets both in $y_g(\cdot)$ and $y_k(\cdot)$

Similarity oriented measures - II

Results given by
`pair_confusion_matrix(y_g,y_k)`

	SK	DK
SG	13512	1488
DG	1200	6150

Rand Score $\frac{SGSK + DGDK}{SGSK + DGDK + SGDK + DGSK} = 0.88$

Adjusted Rand Score Excludes the count of matches expected by chance² = 0.73

Jaccard Coefficient for label c $\frac{SG_c SK_c}{SG_c SK_c + SG_c DK_c + DG_c SK_c} = (1, 0.75, 0.69)$

- it requires remapping of $y_g(\cdot)$ to obtain the best match

2 See the [Wikipedia page](#) for a reference

1	Introduction to clustering	2
2	K-means	12
3	Evaluation of a clustering scheme	48
4	Hierarchical clustering	73
	● Separation	76
	● Single linkage hierarchical clustering	78
	● Examples	86
5	Density based clustering	97
6	Model based clustering	117
7	Final remarks	126

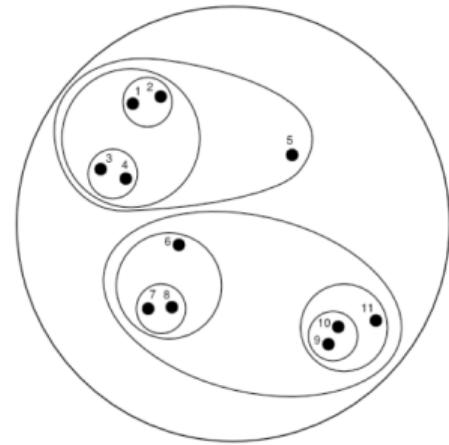
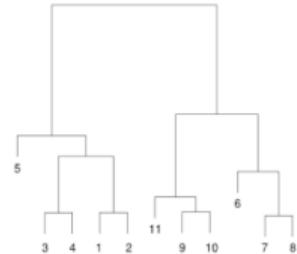
Hierarchical clustering

Generates a **nested structure** of clusters

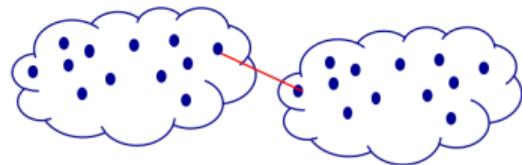
- Agglomerative (bottom up)
 - as a starting state, each data point is a cluster
 - in each step the two **less separated** clusters are merged into one
 - a measure of **separation between clusters** is needed
- Divisive (top down)
 - as a starting state, the entire dataset is the only cluster
 - in each step, the cluster with the lowest cohesion is split
 - a measure of cluster cohesion and a split procedure are needed

Hierarchical clustering output

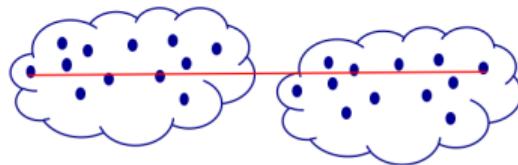
- Dendrogram (left)
- Nested cluster diagram (right)
- They represent the same structure
- The representation is the same for agglomerative and divisive
- The agglomerative methods are the most used



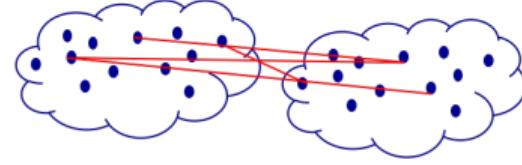
Separation between clusters – Graph based



Single Link



Complete Link



Average Link

$$\text{Sep}(k_i, k_j) = \min_{x \in k_i, y \in k_j} \text{Dist}(x, y)$$

$$\text{Sep}(k_i, k_j) = \max_{x \in k_i, y \in k_j} \text{Dist}(x, y)$$

$$\text{Sep}(k_i, k_j) = \frac{1}{|k_i||k_j|} \sum_{x \in k_i, y \in k_j} \text{Dist}(x, y)$$

The distance between sets is based on the distances between objects belonging to the two sets, respectively

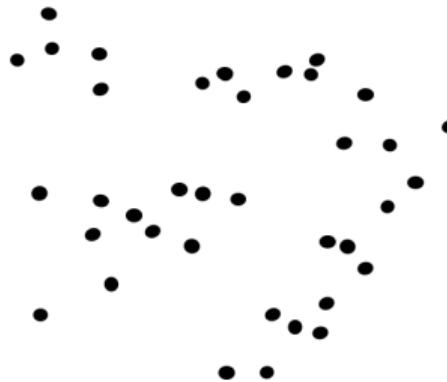
Separation between clusters – Prototype based

Several alternatives, among them

- Distance between the centroids
- Ward's method
 - Given two sets with the respective SSE, the separation between the two is measured as the difference between the total SSE resulting in case of merge and the sum of the original SSE; smaller separation implies a lower increase in the SSE after merging

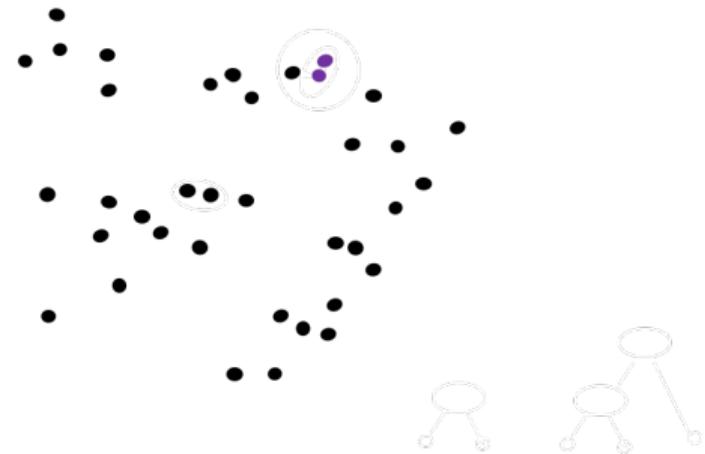
Single linkage hierarchical clustering I

1. Initialization: every object is a cluster



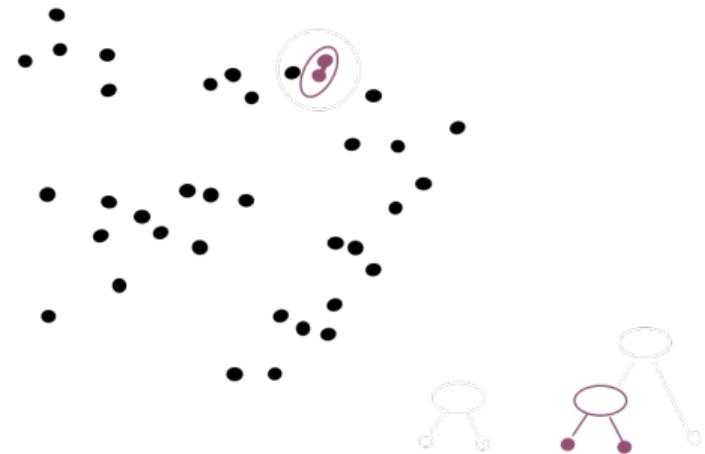
Single linkage hierarchical clustering II

1. Initialization: every object is a cluster
2. Find the **less separated** pair



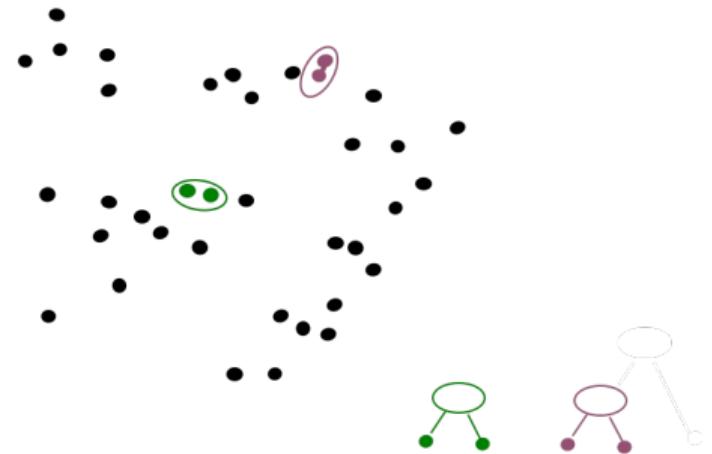
Single linkage hierarchical clustering III

1. Initialization: every object is a cluster
2. Find the **less separated** pair
3. Merge in a single cluster



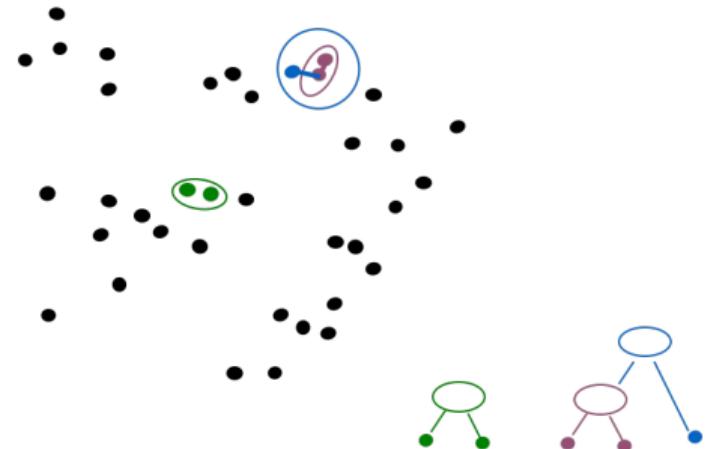
Single linkage hierarchical clustering IV

1. Initialization: every object is a cluster
2. Find the **less separated** pair
3. Merge in a single cluster
4. Repeat



Single linkage hierarchical clustering V

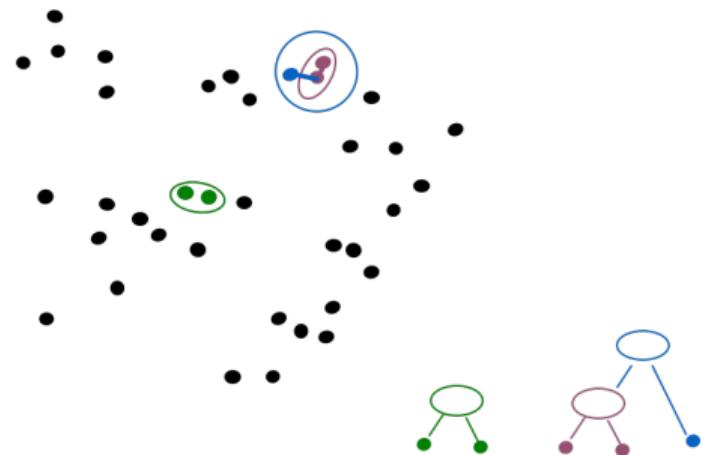
1. Initialization: every object is a cluster
2. Find the **less separated** pair
3. Merge in a single cluster
4. Repeat ...



Single linkage hierarchical clustering VI

1. Initialization: every object is a cluster
2. Find the **less separated** pair
3. Merge in a single cluster
4. Repeat ...

The result is a **dendrogram** (taxonomy, object hierarchy)



Single linkage algorithm

- Initialize the clusters, one for each objects
- Compute the **distance matrix** between the clusters, squared, symmetric, the size is the number of objects N , the main diagonal is null
- While the number of clusters is greater than 1
 - find the two clusters with lowest separation, say k_r and k_s
 - merge them in a cluster
 - delete from the distance matrix the rows and columns r and s and insert one new row and column with the distances of the new cluster from the others

$$\text{Dist}(k_k, k_{(r+s)}) = \min(\text{Dist}(k_k, k_r), \text{Dist}(k_k, k_s)) \forall k \in [1, K]$$

Time and space complexity

- Space and time: $\mathcal{O}(N^2)$ for the computation and the storage of the distance matrix
- Worst case $N - 1$ iterations to reach the final single cluster
- For the i -th step of the main iteration:
 - search of the pair to merge $\mathcal{O}((N - i)^2)$
 - recomputation of the distance matrix $\mathcal{O}((N - i))$
- Time, in summary: $\mathcal{O}(N^3)$
 - can be reduced to $\mathcal{O}(N^2 \log(N))$ with indexing structures

Italian cities example I

	BA	FI	MI	NA	RM	TO
BA	0	662	877	255	412	996
FI	662	0	295	468	268	400
MI	877	295	0	754	564	138
NA	255	468	754	0	219	869
RM	412	268	564	219	0	669
TO	996	400	138	869	669	0



Italian cities example II

	BA	FI	MI	NA	RM	TO
BA	0	662	877	255	412	996
FI	662	0	295	468	268	400
MI	877	295	0	754	564	138
NA	255	468	754	0	219	869
RM	412	268	564	219	0	669
TO	996	400	138	869	669	0



Italian cities example III

	BA	FI	MI/TO	NA	RM
BA	0	662	877	255	412
FI	662	0	295	468	268
MI/TO	877	295	0	754	564
NA	255	468	754	0	219
RM	412	268	564	219	0



Italian cities example IV

	BA	FI	MI/TO	NA/RM
BA	0	662	877	255
FI	662	0	295	268
MI/TO	877	295	0	564
NA/RM	255	268	564	0



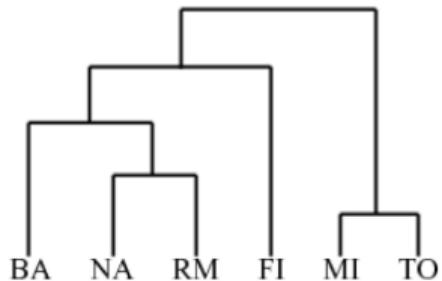
Italian cities example V

	BA/NA/RM	FI	MI/TO
BA/NA/RM	0	268	564
FI	268	0	295
MI/TO	564	295	0



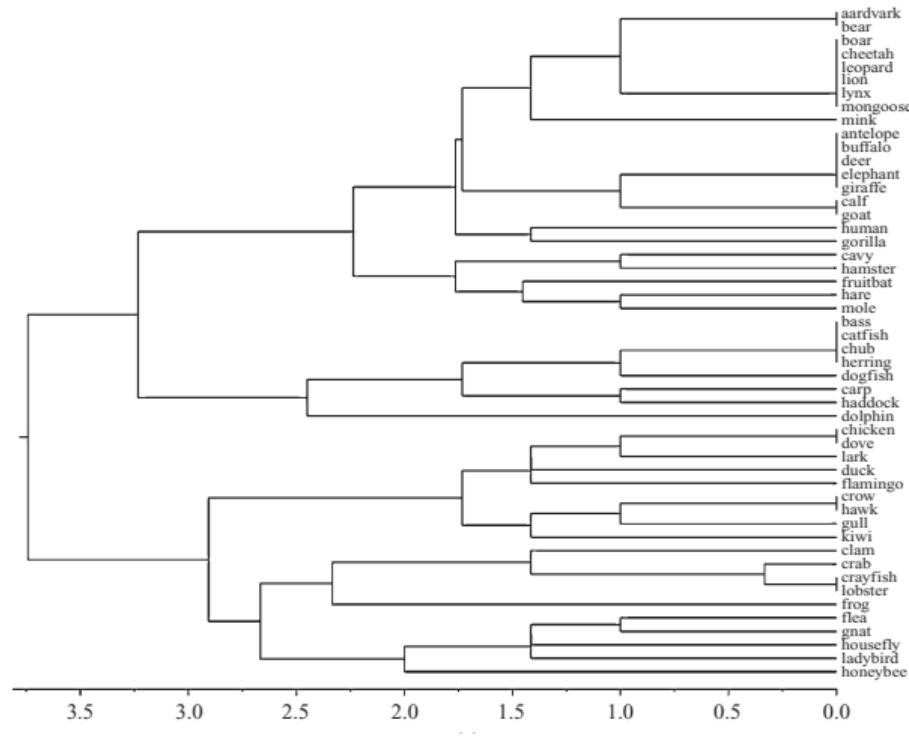
Italian cities example VI

	BA/FI/NA/RM	MI/TO
BA/FI/NA/RM	0	295
MI/TO	295	0



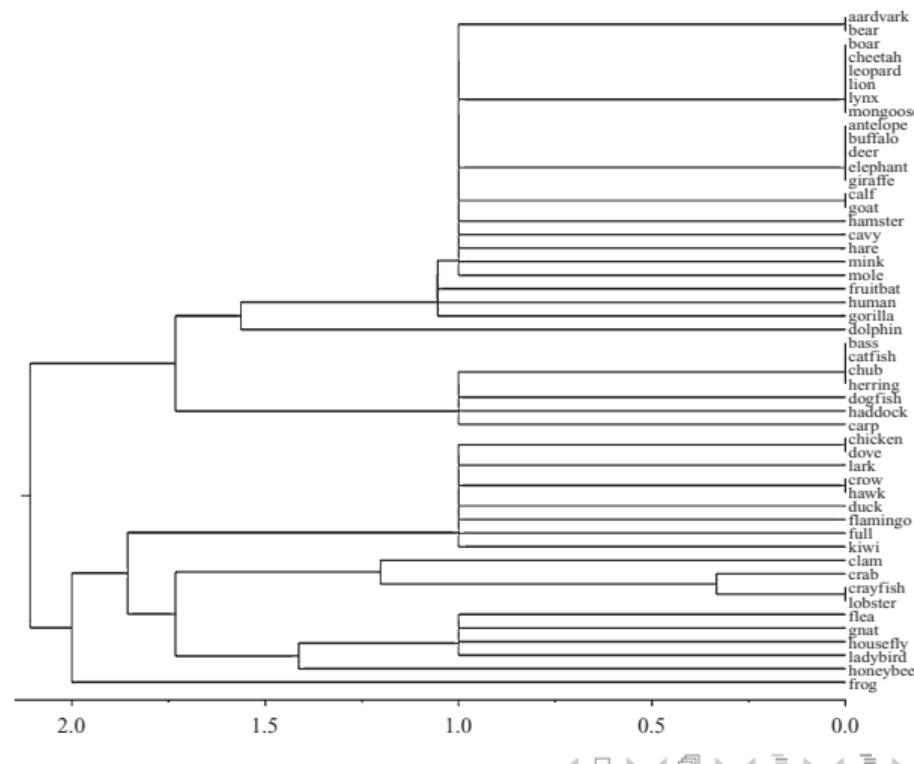
The animals example – Complete Linkage

50 animals described with a numeric attribute (number of legs, scaled to [0,1]) and 15 boolean attributes, such as *has feathers, lays eggs, ...*



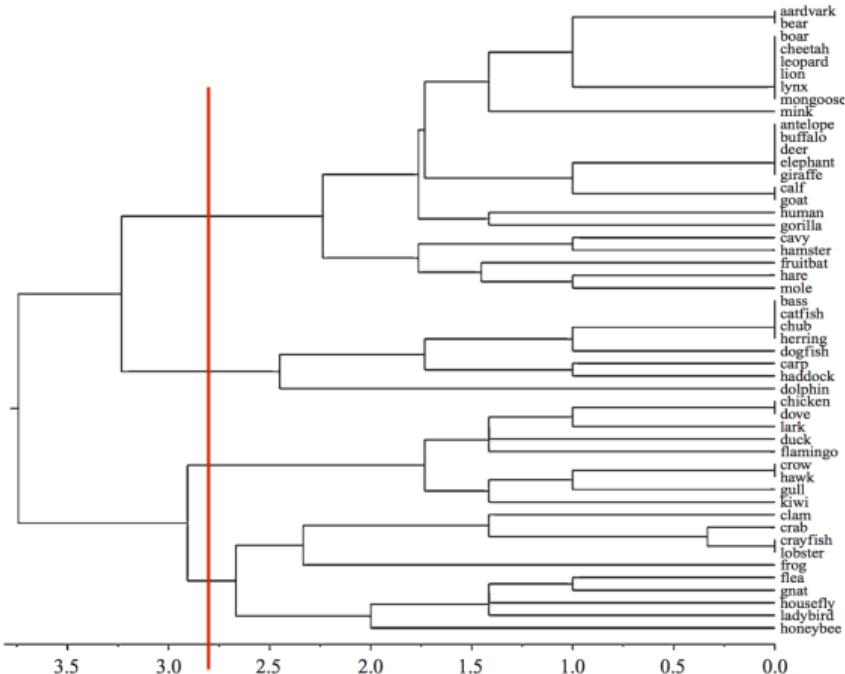
The animals example – Single Linkage

50 animals described with a numeric attribute (number of legs, scaled to [0,1]) and 15 boolean attributes, such as *has feathers, lays eggs, ...*



Generating the clustering scheme

- The desired clustering scheme is obtained by **cutting** the dendrogram at some level
- The choice of the level is application dependent, and can also be guided by indexes, as in the case of K-means



Discussion I

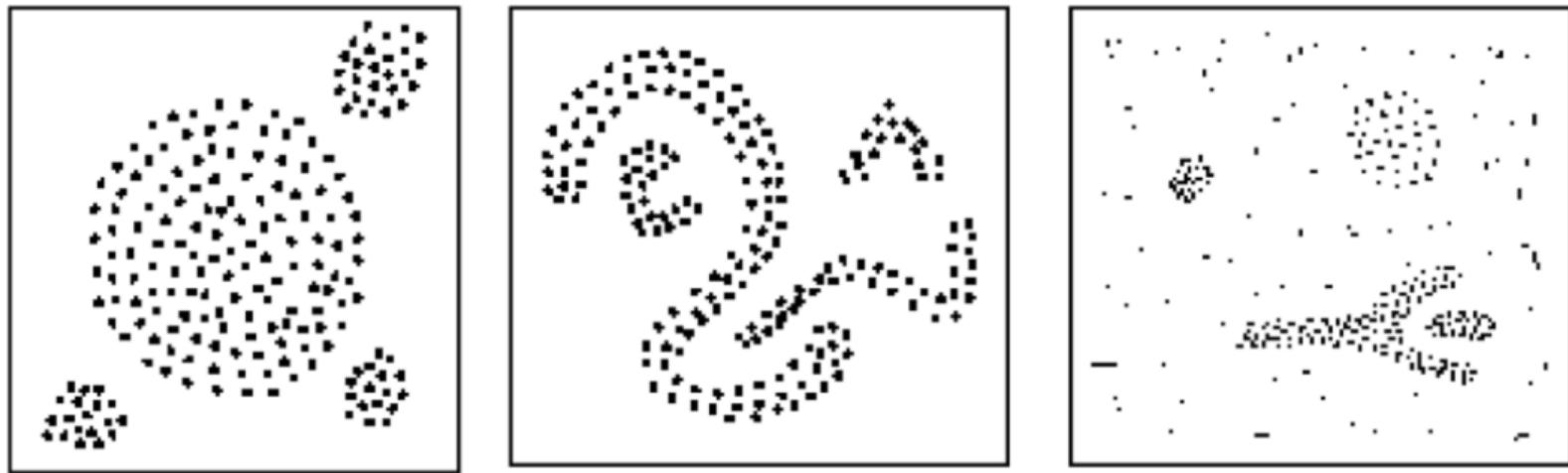
- The horizontal axis in the dendrogram is the **total dissimilarity** inside the clusters, which obviously increases for decreasing number of clusters
- The **diameter** of a cluster is the distance among the most separated objects
 - Single linkage tends to generate clusters with larger diameters also at low levels
 - Complete linkage tends to generate more compact clusters

Discussion II

- (?): The scaling is poor, due to the high complexity
- (?): There isn't a global objective function, the decision is always local and cannot be undone
- (?): The dendrogram structure is of great help for the interpretation of the result
- (?): Empirically, the result is frequently good

1	Introduction to clustering	2
2	K-means	12
3	Evaluation of a clustering scheme	48
4	Hierarchical clustering	73
5	Density based clustering	97
	● DBSCAN	100
	● Kernel Density Estimation	113
6	Model based clustering	117
7	Final remarks	126

Density based clustering



Clusters are high-density regions separated by low-density regions

Computing density

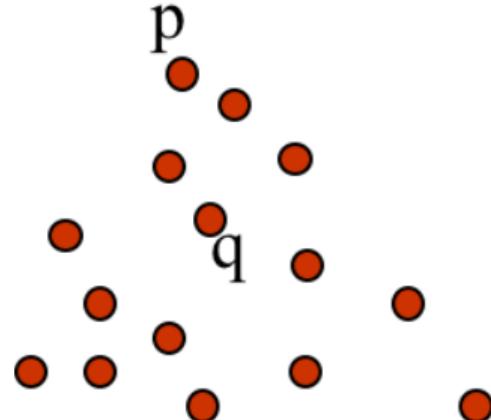
The two most obvious solutions

- Grid-based
 - split the (hyper)space into a regularly spaced grid
 - count the number of objects inside each grid element
- Object-centered
 - define the radius of a (hyper)sphere
 - attach to each object the number of objects which are inside that sphere

DBSCAN – Density Based Spatial Clustering of Applications with Noise³

Intuition

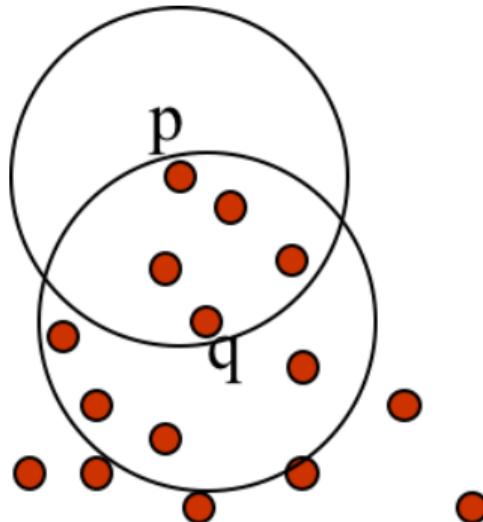
- intuitively, p is a **border** point, while q is a **core** point



3 [Ester et al.(1996)]

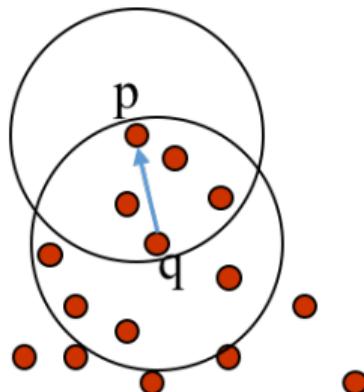
Neighborhood

- define a radius ϵ and define as **neighborhood** of a point the ϵ -hypersphere centered at that point
- points p and q are one in the neighborhood of the other
 - neighborhood is *symmetric*



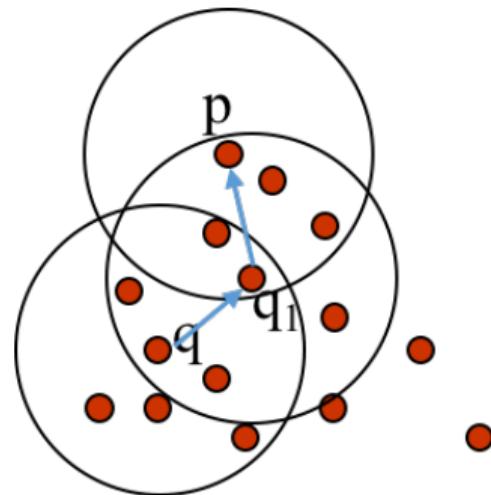
Direct Density Reachability

- define a threshold `minPoints` and define as **core** a point with at least `minPoints` points in its neighborhood, as **border** otherwise
 - with $\text{minPoints} = 5$, q is core, p is border
- define that a point p is **directly density reachable** from point q iff
 - q is core
 - q is in the neighborhood of p
- direct density reachability is not symmetric
 - in the example q is not directly density reachable from p , since p is border



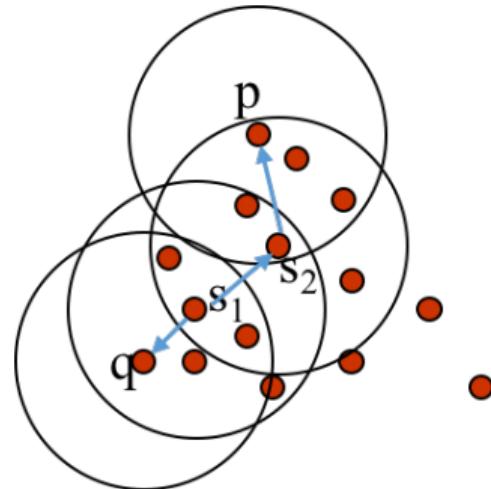
Density Reachability

- a point p is **density reachable** from point q iff
 - q is core
 - there is a sequence of points point q_i such that q_{i+1} is directly density reachable from q_i , $i \in [1, nq]$, q_1 is directly reachable from q and p is directly density reachable from q_{nq}
- reachability is not symmetric
 - in the example q is not density reachable from p , since p is border



Density Connection

- a point p is **density connected** to point q iff there is a point s such that p and q are density reachable from s
- density connection is symmetric



Generation of clusters

- A *cluster* is a maximal set of points connected by *density*
- Border points which are not connected by density to any core point are labelled as **noise**

Algorithm I

Algorithm DBSCAN

Require: SetOfPoints: UNCLASSIFIED points

Require: Eps, MinPts

ClusterId \leftarrow nextId(NOISE);

for i = 1 to SetOfPoints.size **do**

 Point \leftarrow SetOfPoints.get(i)

if Point.CId = UNCLASSIFIED **then**

if ExpandCluster(SetOfPoints, Point, ClusterId, Eps, MinPts)

then

 ClusterId \leftarrow nextId(ClusterId)

Ensure: SetOfPoints

Algorithm II

```
ExpandCluster(SetOfPoints, Point, ClId, Eps, MinPts) : Boolean;  
    seeds:=SetOfPoints.regionQuery(Point,Eps);  
    If seeds.size<MinPts THEN // no core point  
        SetOfPoint.changeClId(Point,NOISE);  
        RETURN False;  
    ELSE ...
```

Algorithm III

```
// all points in seeds are density-reachable from Point
SetOfPoints.changeC1Ids(seeds,C1Id);
seeds.delete(Point);
WHILE seeds <> Empty DO
    currentP := seeds.first();
    result := SetOfPoints.regionQuery(currentP,Eps);
    If result.size >= MinPts THEN
        For i FROM 1 TO result.size DO
            resultP := result.get(i);
            If resultP.C1Id IN {UNCLASSIFIED, NOISE} THEN
                If resultP.C1Id = UNCLASSIFIED THEN seeds.append(resultP);
            END If;
            SetOfPoints.changeC1Id(resultP,C1Id);
        END If; // UNCLASSIFIED or NOISE
        END For;
    END If; // result.size >= MinPts
    seeds.delete(currentP);
END WHILE; // seeds <> Empty
RETURN True;
END If
END; // ExpandCluster
```

How to set ϵ and minPoints?

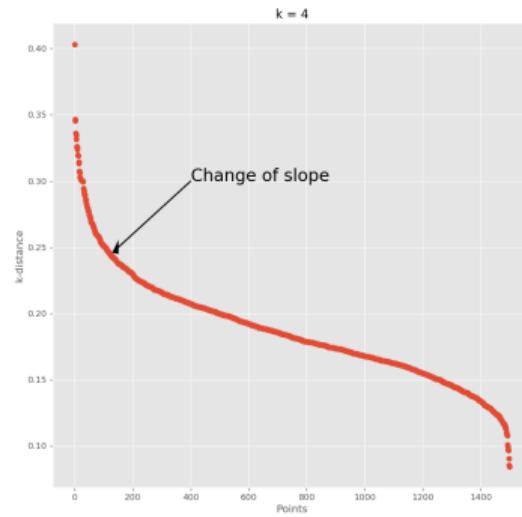
- As in many other machine learning algorithms, a *grid search* over several combination of hyperparameters can be useful
- As a *rule of thumb*, you can try $\text{minPoints} = 2 * D$, the number of dimensions
- Noise suggest an increase in minPoints
- A guess for ϵ requires more effort, considering the distance of the k -nearest neighbour, with $k = \text{minPoints}$

Good guess for ϵ !

- Consider the vector of the *k-distances*
 - choose k
 - for each point we compute the distance of its k -nearest neighbour and we sort the points for decreasing k -distance
- Choosing a given k -distance as ϵ , it turns out that all the points with a k -distance bigger than ϵ will be considered as *border*
 - in the figure of next page they are the points to the left of the vertical of the chosen ϵ

Good guess for $\epsilon \parallel$

- Usually, datasets which exhibit some tendency to clustering exhibit also a *change of slope*
- The best ϵ can be found with a grid search in the *area of the change of slope*
 - the figure refers to a dataset with 1500 points and with minPoints=4
 - this figure suggests a fine tuning of ϵ in the interval 0.2–0.3



Comments

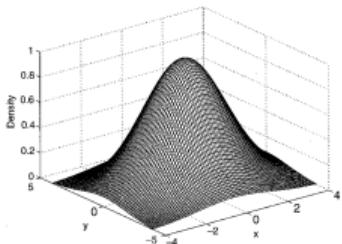
- ☺ Finds clusters of any shape
- ☺ Is robust w.r.t. noise
- ☹ Problems if clusters have widely varying densities
 - Being based on distances between points, the complexity is $\mathcal{O}(N^2)$
 - reduced to $\mathcal{O}(N \log(N))$ if spatial indexes, such as R*, are available
 - Very sensitive to the values of ϵ and minPoints
 - Decreasing ϵ and increasing minPoints reduces the cluster size and increases the number of noise points



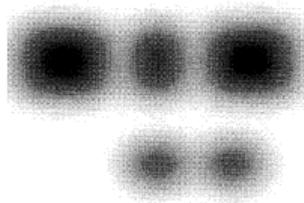
Kernel Density Estimation ⁴

- A technique developed in statistics and pattern mining
- Describe the distribution of the data by a function
- The overall density function is the sum of the **influence functions** (or **kernel functions**) associated with each point
- The kernel function
 - must be **symmetric** and monotonically decreasing
 - usually has a *parameter* to set the decreasing rate

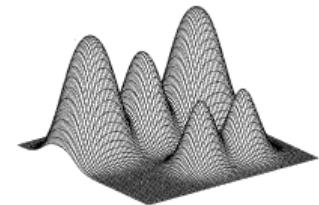
KDE – Example



Gaussian kernel



Overall density – Grey scale plot



Sample set of 12 points
Overall density – Surface plot

DENCLUE algorithm

1. Derive a density function for the space occupied by the data points
2. Identify the points that are local maxima
3. Associate each point with a density attractor by moving in the directions of maximum increase in density
4. Define clusters consisting of points associated with a particular density attractor
5. Discard clusters whose density attractor has a density less than a user-specified threshold ξ
6. Combine clusters that are connected by a path of points that all have a density of ξ or higher

DENCLUE comments

- ☺ It has a strong theoretical foundation on statistics
 - precise computation of density
 - DBSCAN is a special case of DENCLUE where the influence is a step function
- ☺ Good at dealing with noise and clusters of different shapes and sizes
- ☹ expensive computation $\mathcal{O}(N^2)$
 - can be optimized with approximated *grid based* computation
- ☹ Troubles with high dimensional data and clusters with different densities

1	Introduction to clustering	2
2	K-means	12
3	Evaluation of a clustering scheme	48
4	Hierarchical clustering	73
5	Density based clustering	97
6	Model based clustering	117
	● Gaussian Mixture	119
7	Final remarks	126

Model based (or statistic based) clustering⁵

- Estimate the parameters of a statistical model to maximize the ability of the model to **explain the data**
- The main technique is to use the **mixture models**
 - view the data as a set of observation from a mixture of different probability distributions
- Usually, the base model is a multivariate normal
 - well-known, easy to work with, good results
- The estimation is usually done using the **maximum likelihood**
 - given a set of data \mathcal{X} , the probability of the data, regarded as a function of the parameters, is called a **likelihood function**
- Attributes are assumed to be random independent variables

5 [Tan et al.(2006) Tan, Steinbach, and Kumar], Section 9.2.2

Gaussian Mixture

a.k.a. Expectation Maximization – EM

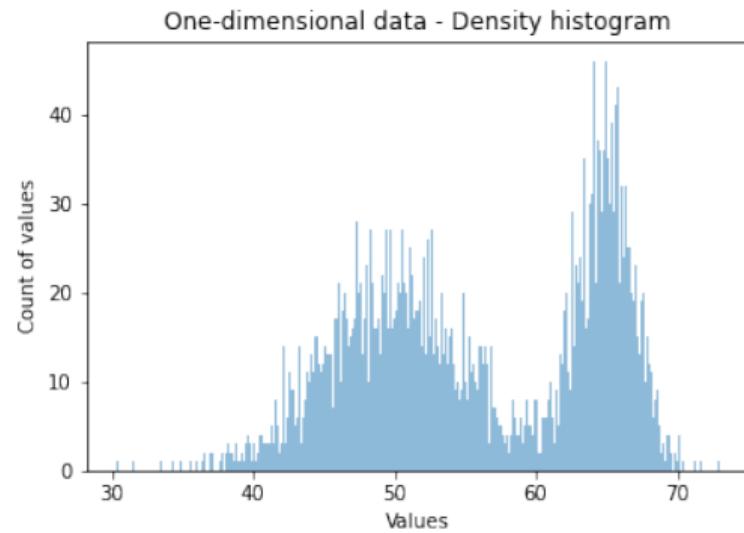
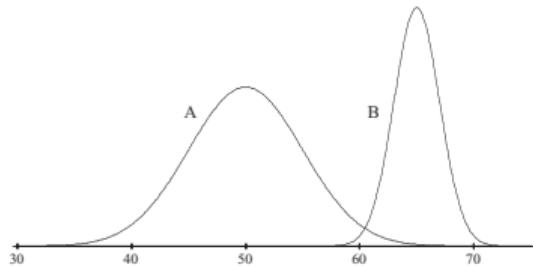
- If the data can be approximated by a single distribution, the derivation of the parameters is straightforward
- In the general case, with many mixed distributions, the EM algorithm is used

EM algorithm

1. Select an initial set of model parameters
2. **repeat**
 - 2.1 **Expectation Step** – For each object, calculate the probability that each object belongs to each distribution
 - 2.2 **Maximization Step** – Given the probabilities from the expectation step, find the new estimates of the parameters that maximize the expected likelihood
3. **until** – the parameters do not change (or the change is below a specified threshold)

One dimension mixture example

- Case with one dimension, two components
- Synthetic data randomly generated with two gaussians
 $\mu_A = 50, \sigma_A = 5, p_A = 0.6$
 $\mu_B = 65, \sigma_B = 2, p_B = 0.4$



EM – one dimension, two clusters example I

- Need to estimate 5 parameters
 - mean and standard deviation for cluster A
 - mean and standard deviation for cluster B
 - sampling probability p for cluster A

$$\Pr(A|x) = \frac{\Pr(x|A)\Pr(A)}{\Pr(x)} = \frac{f(x; \mu_A, \sigma_A)p_A}{\Pr(x)}$$
$$f(x; \mu_A, \sigma_A) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{(x-\mu)^2}{2\sigma^2}}$$

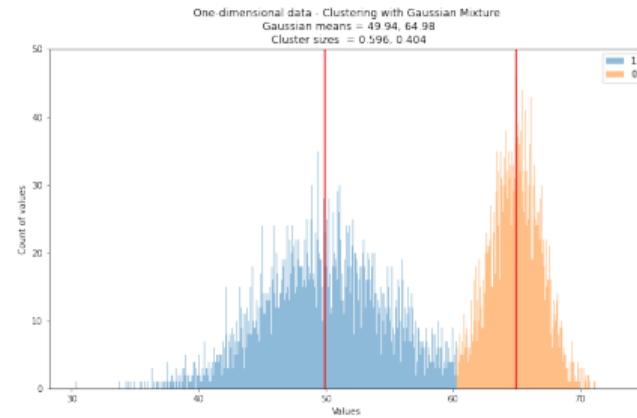
EM – one dimension, two clusters example II

- Repeat until convergence
 - Expectation: Compute p_A and p_B using the current distribution parameters
 - Compute the numerators for $\Pr(A|x)$ and $\Pr(B|x)$ and normalize dividing by their sum
 - Maximization of the distribution likelihood given the data
 - Compute the new distributions parameters, weighting the probabilities according to the current distribution parameters
- After convergence label each object with A or B according to the maximum probability, given the last distribution parameters

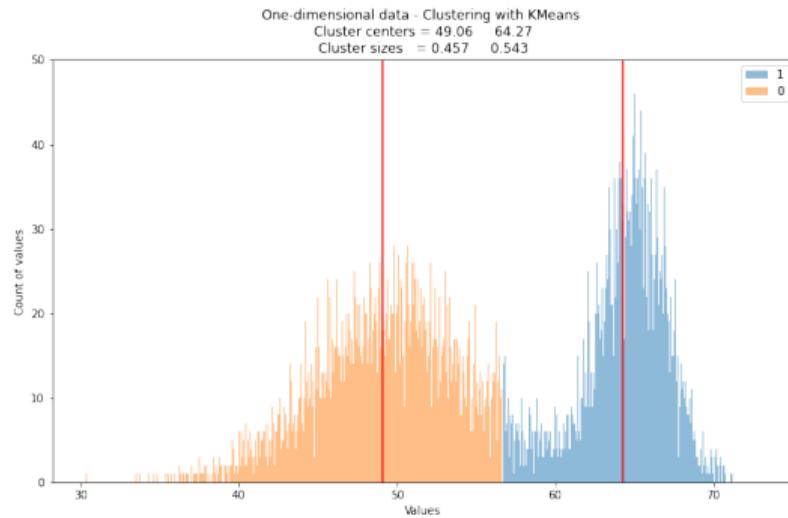
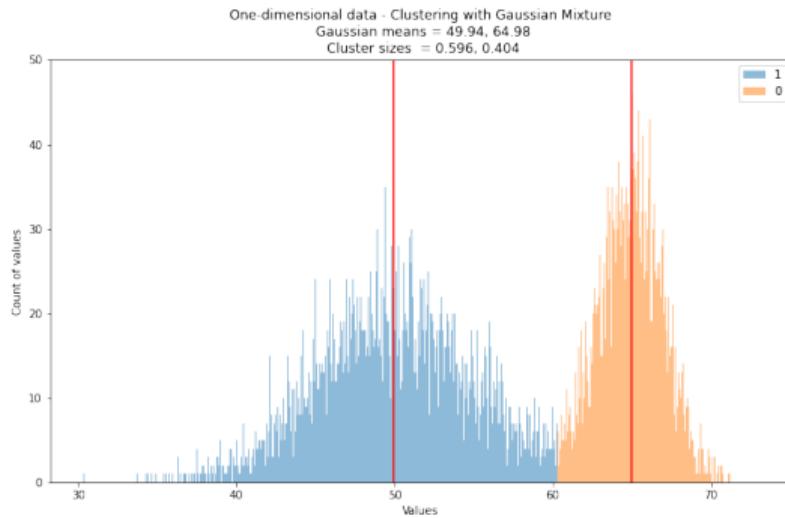
One dimension example - Gaussian Mixture result

Estimated parameters of the distributions

	weight	mean	deviation
0	0.4037	64.978754	2.030968
1	0.5963	49.939990	4.999235



Gaussian Mixture and KMeans – Comparison of results

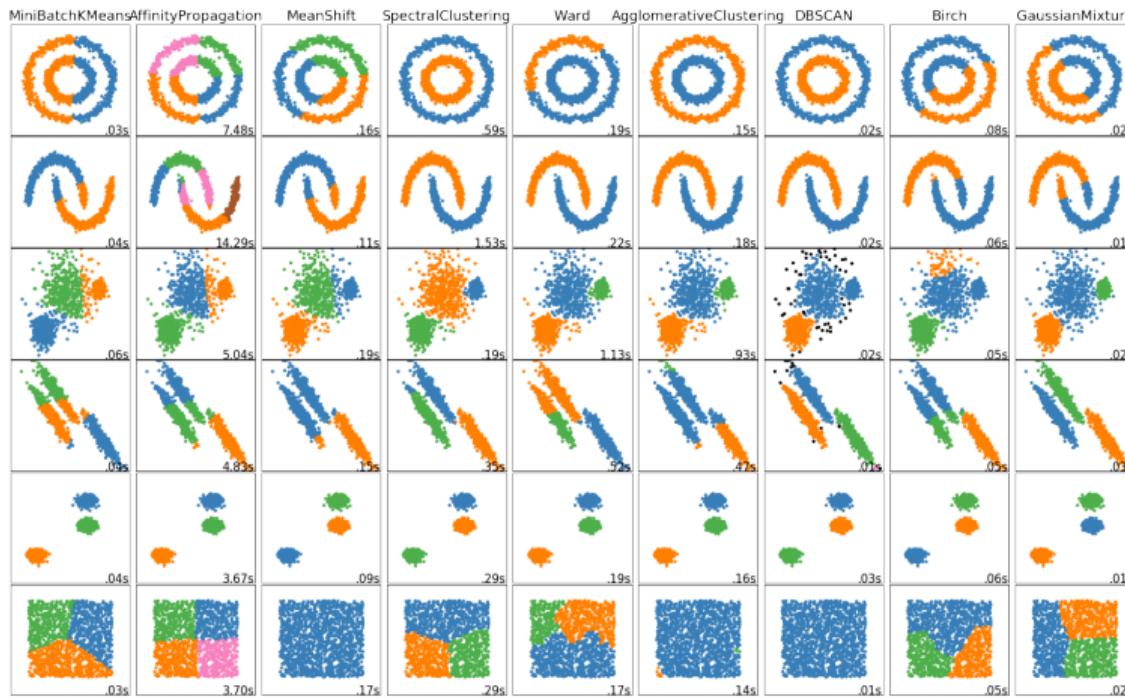


- These data have a *bimodal, gaussian-like* distribution
- The EM algorithm is founded on the hypothesis of modelling data with gaussians
- KMeans is *non-parametric*, and in this case the performance is *worse*

1	Introduction to clustering	2
2	K-means	12
3	Evaluation of a clustering scheme	48
4	Hierarchical clustering	73
5	Density based clustering	97
6	Model based clustering	117
7	Final remarks	126

Comparison of results for selected algorithms

(from [Scikit-Learn documentation](#))



A summary of selected clustering algorithms – I

Method name	Parameters	Scalability	Use case	Geometry (metric used)
K-Means	number of clusters	Very large n_samples, medium n_clusters with MiniBatch code	General-purpose, even cluster size, flat geometry, not too many clusters, inductive	Distances between points
Affinity propagation	damping, sample preference	Not scalable with n_samples	Many clusters, uneven cluster size, non-flat geometry, inductive	Graph distance (e.g. nearest-neighbor graph)
Mean-shift	bandwidth	Not scalable with n_samples	Many clusters, uneven cluster size, non-flat geometry, inductive	Distances between points
Spectral clustering	number of clusters	Medium n_samples, small n_clusters	Few clusters, even cluster size, non-flat geometry, transductive	Graph distance (e.g. nearest-neighbor graph)
Ward hierarchical clustering	number of clusters or distance threshold	Large n_samples and n_clusters	Many clusters, possibly connectivity constraints, transductive	Distances between points

A summary of selected clustering algorithms – II

Method name	Parameters	Scalability	Use case	Geometry (metric used)
Agglomerative clustering	number of clusters or distance threshold, linkage type, distance	Large n_samples and n_clusters	Many clusters, possibly connectivity constraints, non Euclidean distances, transductive	Any pairwise distance
DBSCAN	neighborhood size	Very large n_samples, medium n_clusters	Non-flat geometry, uneven cluster sizes, outlier removal, transductive	Distances between nearest points
OPTICS	minimum cluster membership	Very large n_samples, large n_clusters	Non-flat geometry, uneven cluster sizes, variable cluster density, outlier removal, transductive	Distances between points
Gaussian mixtures	many	Not scalable	Flat geometry, good for density estimation, inductive	Mahalanobis distances to centers
BIRCH	branching factor, threshold, optional global clusterer.	Large n_clusters and n_samples	Large dataset, outlier removal, data reduction, inductive	Euclidean distance between points

Clustering types

- Partitioning
 - iteratively find partitions in the dataset, optimizing some quality criterion
- Hierarchic
 - recursively compute a structured hierarchy of subsets
- Density based
 - compute densities and aggregates clusters in high density areas
- Model based
 - assume a model for the distribution of the data and find the model parameters which guarantee the best fitting to the data

Clustering scalability

- Effectiveness decreases with
 - dimensionality D
 - noise level
- Computational cost increases with
 - dataset size N , at least linearly
 - dimensionality D

Research perspective

- From the past
 - well-known problem in statistics
 - recent research
 - machine learning
 - databases
 - visualization
- For the future
 - effective and efficient algorithms for big data clustering, with a large number of dimensions, noise requested scalability w.r.t.:
 - size (N)
 - dimensionality (D)
 - noise level
 - rate of new data arrivals

Uses of clustering – data comprehension

- Biology
 - Creation of taxonomies
 - Genetics
- Information Retrieval
 - Grouping documents
- Climatology
 - Repetition patterns
- Psychology and medicine
 - Identification of illness types in front of partial variation of evidence
- Business
 - Customer grouping

Uses of clustering – utilities

- Summarization
 - Reasoning with groups representatives instead of with the entire population
- Data compression
 - Reduce the amount of data
 - Find cluster prototypes and substitute data with the indexes of the prototypes
 - Vector quantization
- Find the nearest neighbours
 - Each object refers to his prototypes
 - Near neighbours refer to the same prototype

Bibliography I

- Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu.
A density-based algorithm for discovering clusters in large spatial databases with noise.
pages 226–231. AAAI Press, 1996.
- A. K. Jain, M. N. Murty, and P. J. Flynn.
Data clustering: A review.
ACM Comput. Surv., 31(3):264–323, September 1999.
ISSN 0360-0300.
doi: 10.1145/331499.331504.
URL <http://doi.acm.org/10.1145/331499.331504>.

Bibliography II

- ▶ Pang-Nin Tan, Michael Steinbach, and Vipin Kumar.
Data Mining.
Addison Wesley, 2006.
ISBN 0-321-32136-7.

Machine Learning

Association Rules

Claudio Sartori

DISI

Department of Computer Science and Engineering – University of Bologna, Italy
claudio.sartori@unibo.it

1	Introduction to Market Basket Analysis	2
	● Support and confidence	6
2	Frequent Itemset Generation	10
3	Rule Generation	26
4	Multidimensional association rules	45
5	Multilevel Association Rules	50

Association Rules – Discovering co-occurrences in a market basket

- Given a set of commercial transactions, find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction

- Example of Association Rules

- $\{\text{Diaper}\} \rightarrow \{\text{Beer}\}$,
- $\{\text{Bread, Milk}\} \rightarrow \{\text{Coke, Eggs}\}$,
- $\{\text{Beer, Bread}\} \rightarrow \{\text{Milk}\}$
- Implication means co-occurrence, not causality!
- The implication of Association Rules is different from that of logic (boolean): it can be true *with some level of truth*

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

Market Basket Transactions

Definition: Frequent Itemset

- **Itemset**
 - A collection of one or more items
 - Example: {Bread, Diaper, Milk}
- **k-itemset**
 - An itemset that contains k items
- **Support count (σ)**
 - Frequency of occurrence of an itemset
 - E.g. $\sigma(\{\text{Bread, Diaper, Milk}\}) = 2$
- **Support**
 - Fraction of transactions that contain an itemset
 - E.g. $\sigma(\{\text{Bread, Diaper, Milk}\}) = 2/5$
- **Frequent Itemset**
 - An itemset whose support is greater than or equal to a *minsup* threshold

<i>TID</i>	<i>Items</i>
1	Bread, Milk
2	Beer, Bread, Diaper, Eggs
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Bread, Coke, Diaper, Milk

Market Basket
Transactions

Definition: Association Rule

- Association Rule

- An expression of the form $A \Rightarrow C$, where A and C are itemsets
 - A = Antecedent and C = Consequent
 - Example: $\{\text{Diaper}, \text{Milk}\} \rightarrow \{\text{Beer}\}$

- Rule Evaluation Metrics

- Support (sup)
 - Fraction of the N transactions that contain both A and C
- Confidence (conf)
 - Measures how often all the items in C appear in transactions that contain A

TID	Items
1	Bread, Milk
1	Beer, Bread, Diaper, Eggs
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Bread, Coke, Diaper, Milk

Market Basket Transactions

$$\begin{aligned} \text{sup} &= \frac{\sigma(\text{Beer}, \text{Diaper}, \text{Milk})}{N} = \frac{2}{5} = 0.4 \\ \text{conf} &= \frac{\sigma(\text{Beer}, \text{Diaper}, \text{Milk})}{\sigma(\text{Milk}, \text{Diaper})} \end{aligned}$$

Why support and confidence?

- Rules with low support can be generated by random associations
- Rules with low confidence are not really reliable
- Nevertheless a rule with relatively low support but high confidence can represent an uncommon but interesting phenomenon

Association Rule Mining Task

- Given a set of transactions N , the goal of association rule mining is to find all rules having
 - $\text{support} \geq \text{minsup}$ threshold
 - $\text{confidence} \geq \text{minconf}$ threshold
- Brute-force approach:
 - List all possible association rules
 - Compute the support and confidence for each rule
 - Prune rules that fail the minsup and minconf thresholds
⇒ Computationally prohibitive!

Mining Association Rules

Example of Rules:

$\{Diaper, Milk\} \Rightarrow \{Beer\}$	$(s = 0.4, c = 0.67)$
$\{Beer, Milk\} \Rightarrow \{Diaper\}$	$(s = 0.4, c = 1.0)$
$\{Beer, Diaper\} \Rightarrow \{Milk\}$	$(s = 0.4, c = 0.67)$
$\{Beer\} \Rightarrow \{Diaper, Milk\}$	$(s = 0.4, c = 0.67)$
$\{Diaper\} \Rightarrow \{Beer, Milk\}$	$(s = 0.4, c = 0.5)$
$\{Milk\} \Rightarrow \{Beer, Diaper\}$	$(s = 0.4, c = 0.5)$

- All the rules above are binary partitions of the same itemset:
 $\{\text{Beer}, \text{Diaper}, \text{Milk}\}$
- Rules originating from the same itemset have identical support but can have different confidence
 - ⇒ we may decouple the support and confidence requirements

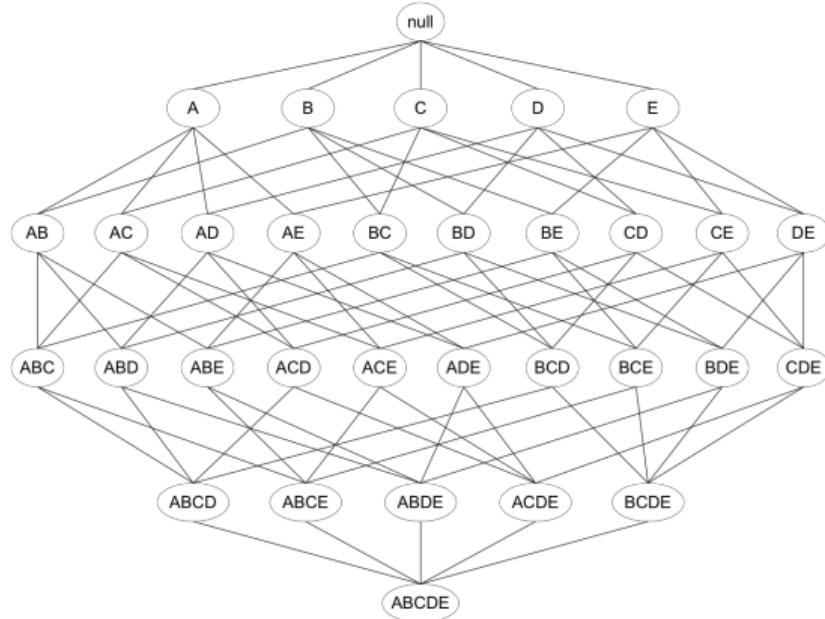
Mining Association Rules

- Two-step approach:
 - 1. Frequent Itemset Generation
 - Generate all itemsets whose support is greater than minsup
 - 2. Rule Generation
 - Generate high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset
- Frequent itemset generation is still computationally expensive

1	Introduction to Market Basket Analysis	2
2	Frequent Itemset Generation	10
	● The Apriori principle	16
	● The Apriori algorithm	18
3	Rule Generation	26
4	Multidimensional association rules	45
5	Multilevel Association Rules	50

Frequent Itemset Generation

Given D items, there are $M = 2^D$ possible candidate itemsets



Frequent Itemset Generation

Brute-force approach:

- Each itemset in the lattice is a **candidate** frequent itemset
- Count the support of each candidate by scanning the database
- Match each transaction against every candidate
- Complexity: $\mathcal{O}(NWM)$ \Rightarrow **Expensive**

TID	Items
1	Bread, Milk
↑ 1	Beer, Bread, Diaper, Eggs
N 3	Beer, Coke, Diaper, Milk
↓ 4	Beer, Bread, Diaper, Milk
5	Bread, Coke, Diaper, Milk

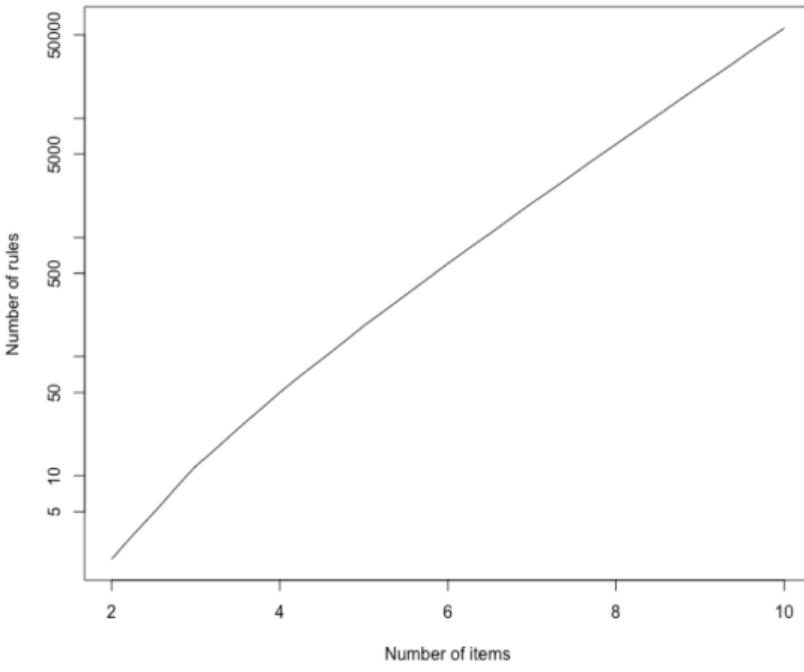
← W →

Brute Force - Computational Complexity

OPTIONAL

- Given D unique items:
 - Total number of itemsets = 2^D
 - Total number of possible association rules:

$$\begin{aligned}
 R &= \sum_{k=1}^{D-1} \left(\binom{D}{k} \times \sum_{j=1}^{D-k} \binom{D-k}{j} \right) \\
 &= 3^D - 2^{D+1} + 1
 \end{aligned}$$



Explanation of the formula

OPTIONAL

- count the number of ways to create an itemset that forms the left hand side of the rule
- for each size k itemset selected for the left-hand side, count the number of ways to choose the remaining $D - k$ items to form the right-hand side of the rule

Going deeper

- choose k of the D items for the left hand side of the rule, there are $\binom{D}{k}$ ways to do this
- there are $\binom{D-k}{i}$ ways to choose the right hand side of the rule, $1 \leq i \leq D - k$
- the double summation derives from the two points above
- the *binomial theorem* states that $\sum_{i=0}^n \binom{n}{i} x^i = (1+x)^n$
- using the theorem for $x = 1$ and $x = 2$ leads to the final result (pay attention to the starting value of the summation)

Frequent Itemset Generation Strategies

- Reduce the number of candidates M
 - Complete search: $M = 2^D$
 - Use pruning techniques to reduce M
- Reduce the number of comparisons NM
 - Use efficient data structures to store the candidates or transactions
 - No need to match every candidate against every transaction

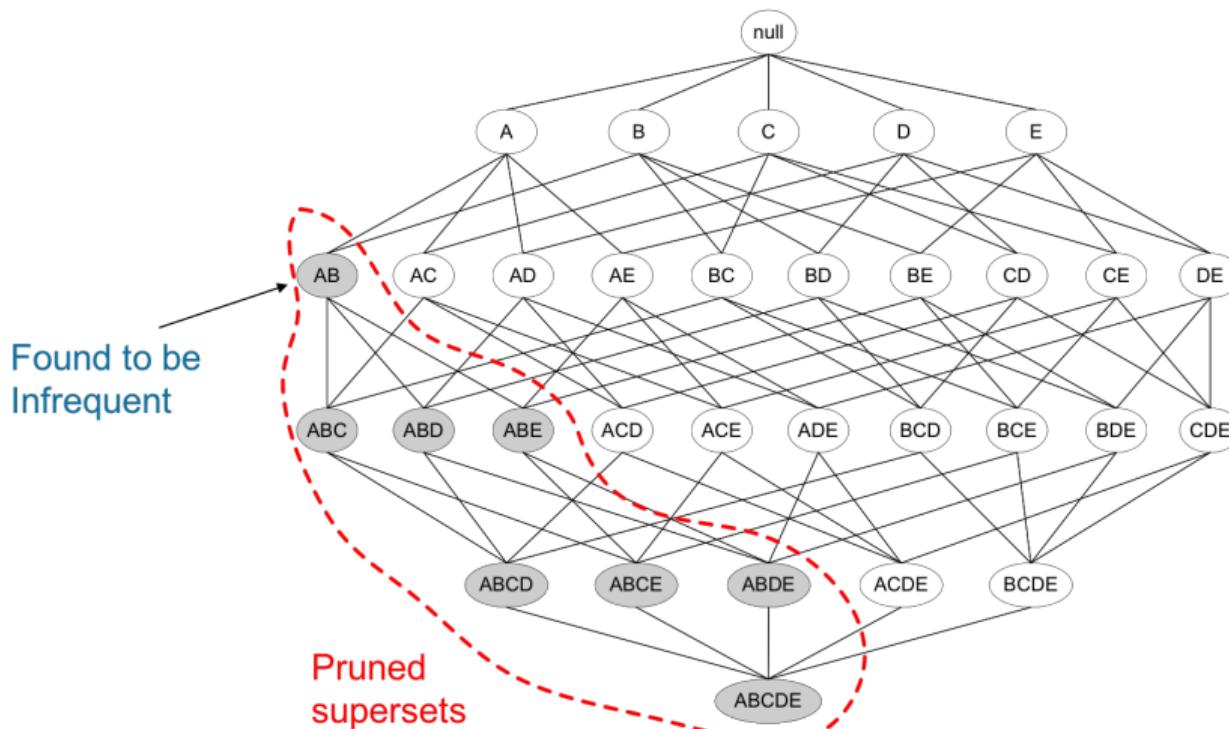
Reducing Number of Candidates

- **Apriori principle**
 - If an itemset is frequent, then all of its subsets must also be frequent
- It holds due to the following property of the support measure:

$$\forall X, Y : (X \subseteq Y) \Rightarrow sup(X) \geqslant sup(Y)$$

- The Support of an itemset never exceeds the support of its subsets
- This is known as the **anti-monotone** property of support

Pruning strategy



Apriori algorithm - Candidate generation

Definitions

C_k : candidate itemsets of size k

L_k : frequent itemsets of size k

$\text{subset}_k(c)$: set of the subsets of c with k elements

Candidate generation – Join Step

- Let L_k be represented as a table with k columns where each row is a frequent itemset
- Let the items in each row of L_k be in lexicographic order
- C_{k+1} is generated by a self join of L_k

```
insert into Ck+1
select p.item1, p.item2, ..., p.itemk, q.itemk
from Lk p, Lk q
where p.item1=q.item1 and ... and p.itemk-1=q.itemk-1
and p.itemk < q.itemk;
```

Candidate generation – Prune Step

Each $(k + 1)$ -itemset which includes a k -itemset which is not in L_k is deleted from C_{k+1}

```
for all  $c \in C_k$  do
    for all  $s \in \text{subset}_{k-1}(c)$  do
        if  $s \notin L_{k-1}$  then
            delete  $c$  from  $C_k$ 
return  $C_k$ 
```

Frequent itemset generation

$L_1 \leftarrow$ frequent 1-itemsets

$k \leftarrow 1$

while $L_k \neq \emptyset$ **do**

C_{k+1} = candidates generated from L_k

for all t transaction in database **do**

 increment candidate count in C_{k+1} for candidates found in t

$L_{k+1} \leftarrow \{c\} \in C_{k+1} : sup(c) \geq minsup$

$k \leftarrow k + 1$

return k, L_k

Pruning example – minsup=3

<i>C₁</i>	<i>Item</i>	<i>Count</i>
Beer	3	
Bread	4	
Coke	2	
Diaper	4	
Eggs	1	
Milk	4	

The support of {Coke} and {Eggs} is below minsupp, therefore they do not generate *C₂* candidates



<i>C₂</i>	<i>Item</i>	<i>Count</i>
Beer,Bread	2	
Beer,Diaper	3	
Beer,Milk	2	
Bread,Diaper	3	
Bread,Milk	3	
Diaper,Milk	3	

No *C₃* candidate will include {Beer, Bread} or {Beer, Milk}



<i>C₃</i>	<i>Item</i>	<i>Count</i>
Bread,Diaper,Milk	2	

Number of itemsets to evaluate:

$$\text{No pruning} = \binom{6}{1} + \binom{6}{2} + \binom{6}{3} = 41$$

Support based pruning = 13

Origin of the name *Apriori*

- Level-wise computation
 - the level is the cardinality of the itemsets under evaluation
- The evaluations at level k use the *prior knowledge* acquired for the previous levels to reduce the search space

Factors Affecting Complexity I

- Choice of minimum support threshold
 - lowering support threshold results in a greater number of frequent itemsets
 - this may reduce pruning and increase the maximum length of frequent itemsets
 - the number of complete reads of the dataset is given by the maximum length of frequent itemsets plus one
- Dimensionality (number of items) of the data set
 - more space is needed to store support count of each item
 - if number of frequent items also increases, both computation and I/O costs may also increase

Factors Affecting Complexity II

- Size of database
 - since Apriori makes multiple passes, run time of algorithm may increase with number of transactions
- Average transaction width
 - transaction width increases with denser data sets
 - This may increase max length of frequent itemsets and traversals of data structures (number of subsets in a transaction increases with its width)

1	Introduction to Market Basket Analysis	2
2	Frequent Itemset Generation	10
3	Rule Generation	26
	● Pattern evaluation	32
4	Multidimensional association rules	45
5	Multilevel Association Rules	50

Confidence

From [Agrawal et al.(1993) Agrawal, Imielinski, and Swami]

- The confidence of a rule can be computed from the supports
⇒ for confidence based pruning of rules it is sufficient to know the supports of frequent itemsets

$$\text{conf}(A \Rightarrow C) = \frac{\text{sup}(A \Rightarrow C)}{\text{sup}(A)}$$

Rule Generation I

Give a frequent itemset L

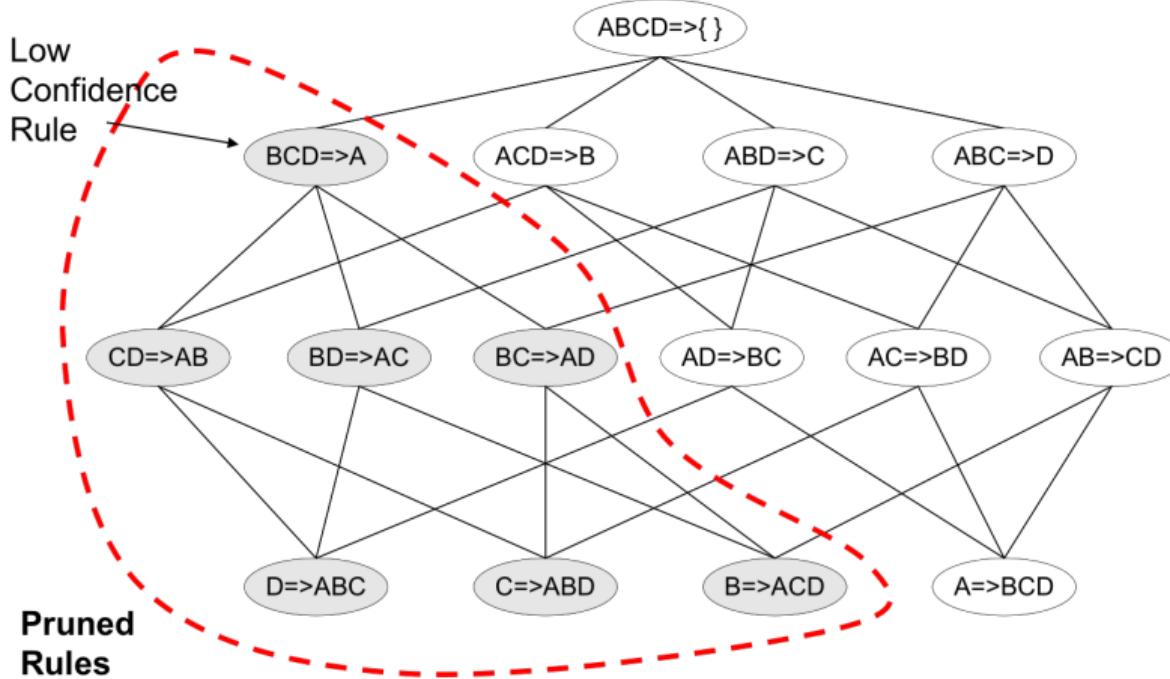
- find all the non-empty subsets $f \in L$ such that the confidence of rule $f \Rightarrow (L - f)$ is not less than the minimum confidence (set by the experiment designer)
 - from $\{Beer, Diaper, Milk\}$ the possible rules are
 $Beer, Diaper \Rightarrow Milk$, $Beer \Rightarrow Diaper, Milk$,
 $Beer, Milk \Rightarrow Diaper$, $Milk \Rightarrow Beer, Diaper$,
 $Diaper, Milk \Rightarrow Beer$, $Diaper \Rightarrow Beer, Milk$
- if $|L| = k$ then there are $2^k - 2$ candidate rules
 - $L \Rightarrow \emptyset$ and $\emptyset \Rightarrow L$ can be ignored

Rule Generation II

- How to efficiently generate rules from frequent itemsets?
 - In general, confidence does not have an anti-monotone property
 - $\text{conf}(ABC \rightarrow D)$ can be larger or smaller than $\text{conf}(AB \rightarrow D)$
 - But let us consider rules generated from the same itemset
 - e.g., $i = \{A, B, C, D\} \in L$:
$$\text{conf}(ABC \rightarrow D) \geq \text{conf}(AB \rightarrow CD) \geq \text{conf}(A \rightarrow BCD)$$
- Confidence of rules generated from the same itemset is anti-monotone w.r.t. the number of items on the RHS of the rule
 - i.e. it decreases when we move an item from the left hand to the right hand

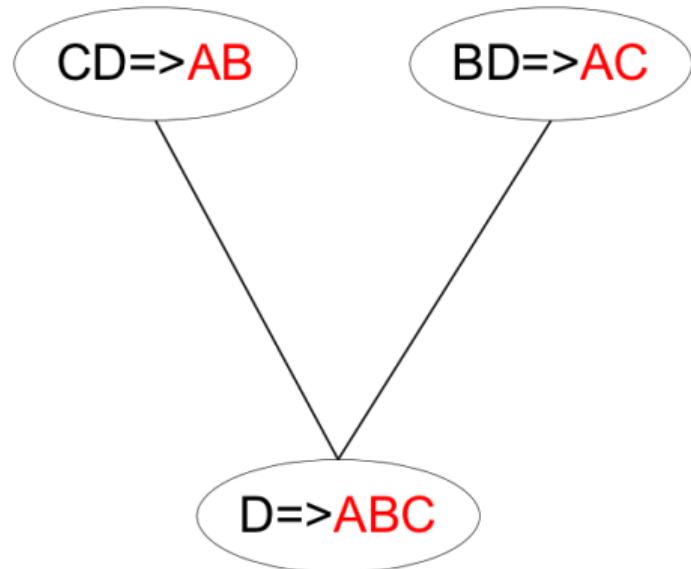
Rule Pruning

Lattice of rules



Rule Generation in Apriori

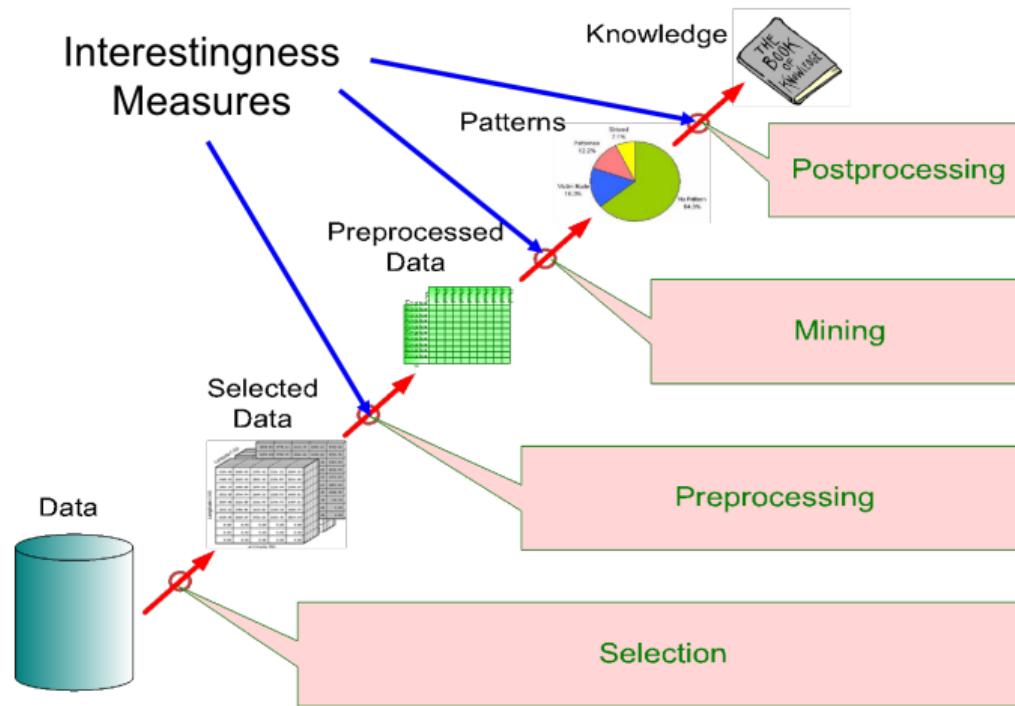
- Candidate rule is generated by merging two rules that share the same prefix in the rule consequent
- $\text{join}(CD \Rightarrow AB, BD \Rightarrow AC)$ would produce the candidate rule $D \Rightarrow ABC$
- Prune rule $D \Rightarrow ABC$ if its subset $AD \Rightarrow BC$ does not have high confidence



Pattern Evaluation

- Association rule algorithms tend to produce too many rules
 - many of them are uninteresting or redundant
 - Redundant if $\{A, B, C\} \Rightarrow \{D\}$ and $\{A, B\} \Rightarrow \{D\}$ have same support and confidence
- Interestingness measures can be used to prune/rank the derived patterns
- In the original formulation of association rules, support and confidence are the only measures used

Application of Interestingness Measure



Computing Interestingness Measures

- Given a rule $A \Rightarrow C$,
the information needed
to compute rule
interestingness can be
obtained from a
contingency table
- The elements of the
contingency table are
the basis for most of the
interestingness measures

	C	\bar{C}	
A	f_{11}	f_{10}	f_{1+}
\bar{A}	f_{01}	f_{00}	f_{0+}
	f_{+1}	f_{+0}	

Drawback of Confidence

- $conf(Tea \Rightarrow Coffee) = \frac{sup(Tea, Coffee)}{sup(Tea)} = \frac{15}{20} = 0.75$
 - fairly high
- $Pr(Coffee) = 0.9$ and
 $Pr(Coffee | \overline{Tea}) = \frac{75}{80} = 0.9375$
 - despite the high confidence of $Tea \Rightarrow Coffee$,
the absence of Tea increases the probability of $Coffee$
 - for this rule the confidence is misleading

	<i>Coffee</i>	<i>Coffee</i>	
<i>Tea</i>	15	5	20
<i>Tea</i>	75	5	80
	90	10	100

$Tea \Rightarrow Coffee$

Statistical Independence

- Population of 1000 students
 - 600 students know how to swim (S)
 - 700 students know how to bike (B)
 - 420 students know how to swim and bike (S, B)
- $\Pr(S \wedge B) = \frac{420}{1000} = 0.42$
- $\Pr(S) * P(B) = 0.6 * 0.7 = 0.42$
- $\Pr(S \wedge B) = P(S) * P(B) \Rightarrow$ Statistical independence
- $\Pr(S \wedge B) > P(S) * P(B) \Rightarrow$ Positively correlated
- $\Pr(S \wedge B) < P(S) * P(B) \Rightarrow$ Negatively correlated

Statistical-based Measures I

Measures that take into account the deviation from statistical independence

$$\text{lift}(A \Rightarrow C) = \frac{\text{conf}(A \Rightarrow C)}{\text{sup}(C)} = \frac{\Pr(A, C)}{\Pr(A) \Pr(C)}$$

- **lift** evaluates to 1 for independence
- insensitive to rule direction
- it is the ratio of true cases w.r.t. independence

Statistical-based Measures II

Measures that take into account the deviation from statistical independence

$$\text{leve}(A \Rightarrow C) = \mathbf{Pr}(A, C) - \mathbf{Pr}(A) * \mathbf{Pr}(C)$$
$$= \text{sup}(A \cup C) - \text{sup}(A)\text{sup}(C)$$

- leverage evaluates to 0 for independence
- insensitive to rule direction
- it is the number of additional cases w.r.t. independence

Statistical-based Measures III

Measures that take into account the deviation from statistical independence

$$\text{conv}(A \Rightarrow C) = \frac{1 - \text{sup}(C)}{1 - \text{conf}(A \Rightarrow C)} = \frac{\Pr(A)(1 - \Pr(C))}{\Pr(A) - \Pr(A, C)}$$

- **conviction** is infinite if the rule is always true
- sensitive to rule direction
- it is the ratio of the expected frequency that A occurs without C (that is to say, the frequency that the rule makes an incorrect prediction) if A and C were independent divided by the observed frequency of incorrect predictions
- also called **novelty**

Intuition about Measures

- higher support \Rightarrow rule applies to more records
- higher confidence \Rightarrow chance that the rule is true for some record is higher
- higher lift \Rightarrow chance that the rule is just a coincidence is lower
- higher conviction \Rightarrow the rule is violated less often than it would be if the antecedent and the consequent were independent

Example of page 35 – Interestingness measures

Tea \Rightarrow *Coffee*

$$\text{conf} = \frac{0.15}{0.20} = 0.75$$

in a 0 to 1 scale it is apparently high

$$\text{lift} = \frac{0.15}{0.90 * 0.20} = 0.83$$

is less than 1, therefore not interesting

$$\text{leve} = 0.15 - 0.90 * 0.20 = -0.03$$

is less than 0, therefore not interesting

$$\text{conv} = \frac{1-0.9}{1-0.75} = 0.4$$

is low, remembering that absolute truth gives
infinite

Comparison of measures

	$C1$	$\bar{C1}$	
$A1$	88	5	93
$\bar{A1}$	5	2	7
	93	7	100

Rule ($A1 \Rightarrow C1$)

$$conf = 0.88/0.93 = 0.946$$

$$lift = 0.88/(0.93 * 0.93) = 1.017$$

$$leve = 0.88 - 0.93 * 0.93 = 0.015$$

$$conv = (1 - 0.93)/(1 - 0.946) = 1.302$$

A high confidence rule can have small lift if both sides are very frequent

	$C2$	$\bar{C2}$	
$A2$	2	5	7
$\bar{A2}$	5	88	93
	7	93	100

Rule ($A2 \Rightarrow C2$)

$$conf = 0.02/0.07 = 0.286$$

$$lift = 0.02/(0.07 * 0.07) = 4.082$$

$$leve = 0.02 - 0.07 * 0.07 = 0.015$$

$$conv = (1 - 0.07)/(1 - 0.286) = 1.302$$

A low confidence rule can have high lift if both sides are very infrequent

Properties of a Good Measure – Piatetsky-Shapiro

Three properties a good measure M must satisfy:

- $M(A, B) = 0$ (or 1) if A and B are statistically independent
- $M(A, B)$ increases monotonically with $\Pr(A, B)$ when $\Pr(A)$ and $\Pr(B)$ remain unchanged
- $M(A, B)$ decreases monotonically with $\Pr(A)$ (or $\Pr(B)$) when $\Pr(A, B)$ and $\Pr(B)$ (or $\Pr(A)$) remain unchanged

Conclusion on measures

- There are lots of measures proposed in the literature, beyond the four presented here
- Confidence is usually the base tool
- Other measures can be used to test the results given by confidence and for additional filtering

1	Introduction to Market Basket Analysis	2
2	Frequent Itemset Generation	10
3	Rule Generation	26
4	Multidimensional association rules	45
	● Equivalence mono/multi	48
5	Multilevel Association Rules	50

Multidimensional association rules

Let's consider a dataset deriving from sensors measuring the concentration of air pollutants

TID	CO	Tin_Oxide	Titanium
1	high	medium	high
2	medium	low	medium
3	medium	high	low
4	low	medium	medium

- Look for rules such as $CO = \text{high}$ and $\text{Tin Oxide} = \text{high}$ then $\text{Titanium} = \text{high}$ (support 0.25 and confidence 1)
- This can be used for example, if one of the sensor is not available, to guess its qualitative value given the others
- Useful for a qualitative analysis, in substitution of regression

Comparison mono– vs multi–dimensional

- Mono–dimensional (intra-attribute)
 - event: **transaction**
 - event description:
 - items A, B, and C are together in a transaction
- Multi–dimensional (inter–attribute)
 - event: **tuple**
 - event description:
 - attribute A has value a, attribute B has value b and attribute C has value c in a tuple

Equivalence mono/multi-dimensional

Multi-dimensional

Schema: $(TID, CO, \text{Tin_Oxide}, \text{Titanium})$

- 1, high, medium, high
- 2, medium, low, medium

Mono-dimensional

- 1, {CO/high, Tin_Oxide/medium, Titanium/high}
2, {CO/medium, Tin_Oxide/low, Titanium/medium}

Schema: $(TID, a?, b?, c?, d?)$

- 1, yes, yes, no, no
- 2, yes, no, yes, no

- ← 1, {a, b}
2, {a, c}

Quantitative attributes

<i>TID</i>	<i>CO</i>	<i>Tin Oxide</i>	<i>Titanium</i>
1	2.6	1360	1046
2	2.0	1292	955
3	2.2	1402	939
4	1.6	1376	948

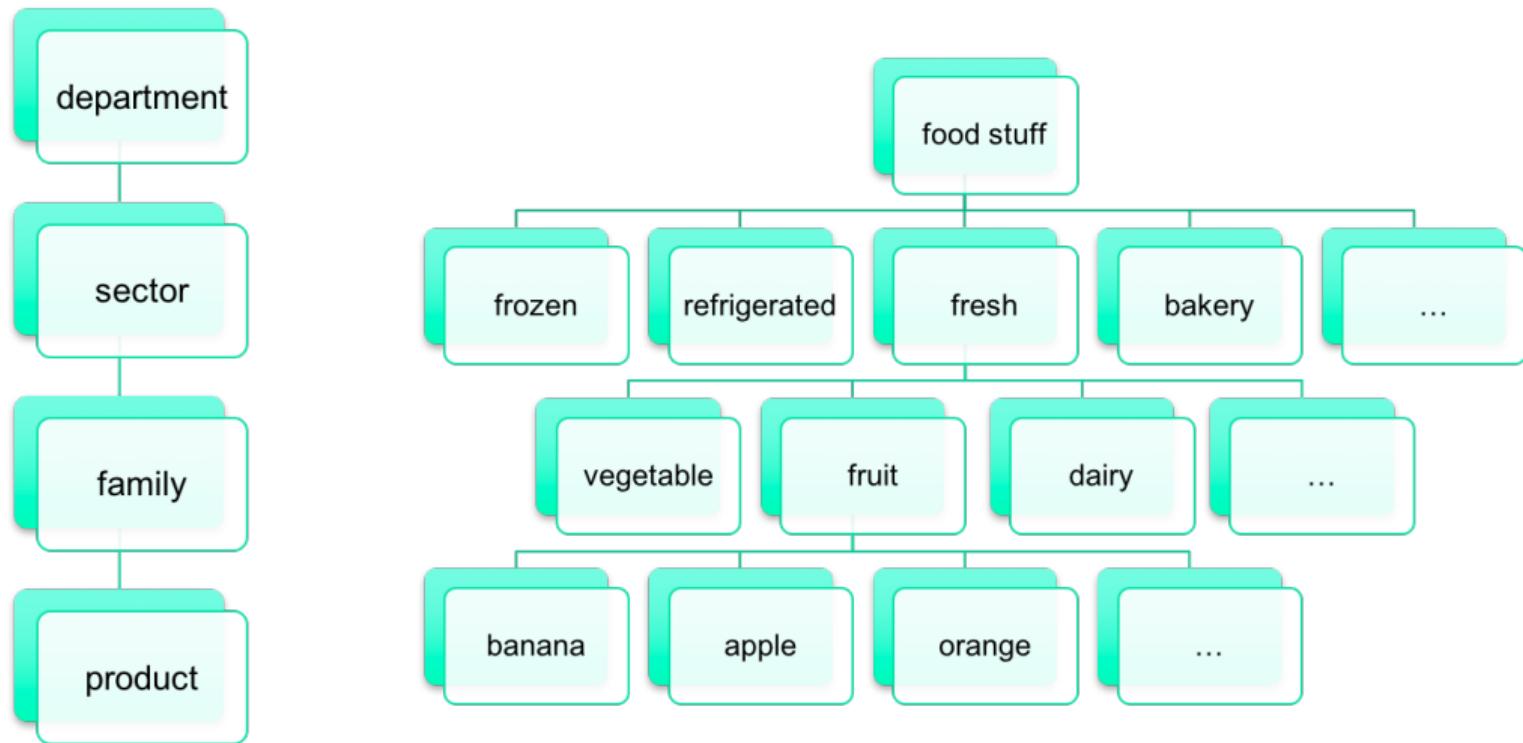
- Too many distinct values for the multi/mono transformation
 - Most software packages for association rules discovery do not deal with quantitative attributes
- ⇒ **discretization**
- possibly *equifrequency* or with *mono-dimensional clustering*, for optimal covering of the original value domains
 - discretisation leads to a dataset like that of page 46
- Association rules can involve items at different qualitative levels

1	Introduction to Market Basket Analysis	2
2	Frequent Itemset Generation	10
3	Rule Generation	26
4	Multidimensional association rules	45
5	Multilevel Association Rules	50
	● Support and Confidence in Multilevel AR	53

Multilevel Association Rules

- A real MBA database can include tens of thousands of distinct items
- Frequently it is necessary to find a tradeoff between general and detailed reasoning
 - choose the right level of abstraction
- A common **background knowledge** is the organization of the items into a hierarchy of concepts
 - it can be easily coded in the transactions
 - it can help the choice of the right level of abstraction

Concept Hierarchy



Support in Multilevel AR

- From specialized to general
 - (apple \Rightarrow milk) \rightarrow (fruit \Rightarrow dairy)
 - the support of rules increases, in general
 - new rules can become interesting
- From general to specialized
 - (fruit \Rightarrow dairy) \rightarrow (apple \Rightarrow milk)
 - the support of rules decreases, in general
 - the support of rules can go under the threshold

Confidence in Multilevel AR

- A level change can influence the confidence in any direction
- If the specialized rule has (approximately) the same confidence as the general one, then it is **redundant**

Example

Low-fat milk is a subclass of milk

- 1000 transactions, 80 with milk and bread, 114 with milk, 20 with low-fat milk and bread, 28 with low-fat milk
 - a) $\text{milk} \Rightarrow \text{bread}$ (support = 8%, confidence = 70%)
 - b) $\text{low-fat milk} \Rightarrow \text{bread}$ (support = 2%, confidence = 71%)
 - rule b) has almost the same confidence as rule a)
 - rule b) is a descendant of rule a)
- ⇒ rule b) is **redundant**

Mining Multilevel Association Rules

- Look for frequent itemsets at each level of abstraction, top down
 - Each level requires a new run of the rule discovery algorithm
- Decrease the support threshold in lower levels

Bibliography I

- Rakesh Agrawal, Tomasz Imieliński, and Arun Swami.
Mining association rules between sets of items in large databases.
SIGMOD Rec., 22(2):207–216, June 1993.
ISSN 0163-5808.
doi: 10.1145/170036.170072.
URL <http://doi.acm.org/10.1145/170036.170072>.

Bibliography II

- ▶ Jiawei Han, Jian Pei, Yiwen Yin, and Runying Mao.
Mining frequent patterns without candidate generation: A frequent-pattern tree approach.
Data Min. Knowl. Discov., 8(1):53–87, 2004.
doi: 10.1023/B:DAMI.0000005258.31418.83.
URL <https://doi.org/10.1023/B:DAMI.0000005258.31418.83>.

Machine Learning

The CRISP-DM methodology

Claudio Sartori

DISI

Department of Computer Science and Engineering – University of Bologna, Italy

claudio.sartori@unibo.it

Standard Process Model

- Can Data Mining be a **push-button** technology?

Standard Process Model

- Can Data Mining be a push-button technology? No

Standard Process Model

- Can Data Mining be a **push–button** technology? No
- Data Mining is a process

Standard Process Model

- Can Data Mining be a **push-button** technology? No
- Data Mining is a process
- The process has **steps** and **complex choices**

Standard Process Model

- Can Data Mining be a **push-button** technology? No
- Data Mining is a process
- The process has **steps** and **complex choices**
- The standard defines the steps in a precise way

Benefits of a Standard Process Model I

DM requires

- a mix of good tools and skilled analysts
- a sound methodology
- project management
- a process model to manage interactions along the process

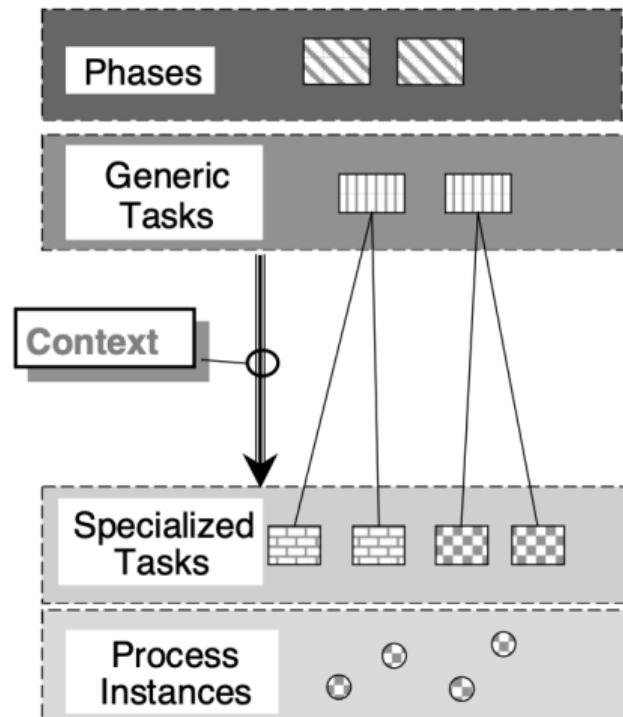
Benefits of a Standard Process Model II

Standardisation provides

- a common reference point for discussions
- a common understanding between the designers and the customers
- a basis for good engineering practice
- checklists
- clarity for expectations

Four Level Breakdown of CRISP-DM

Reference Model

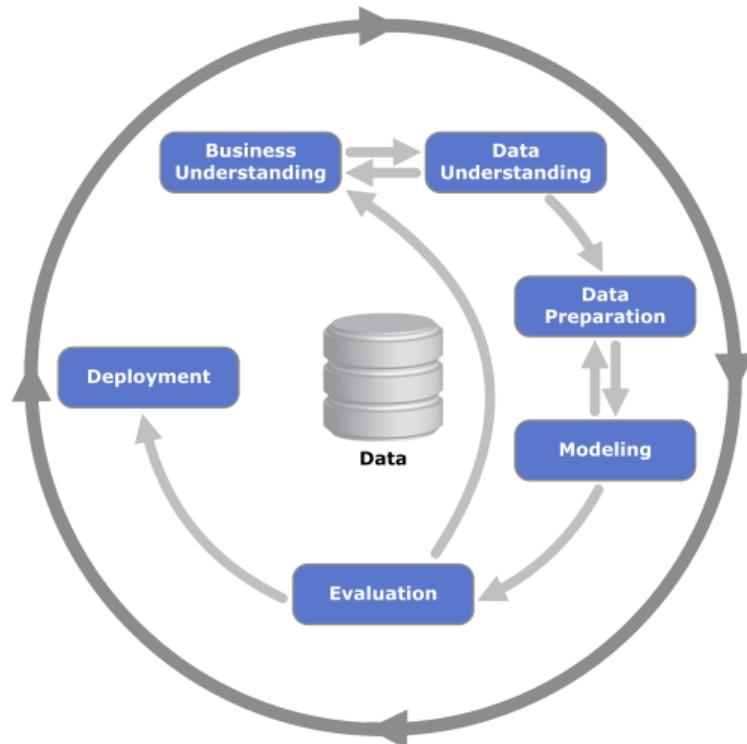


User Guide

- Check lists
- Questionnaires
- Tools and techniques
- Sequence of steps
- Decision Points
- Pitfalls

The CRISP-DM methodology

From the problem to the application - https://en.wikipedia.org/wiki/Cross_Industry_Standard_Process_for_Data_Mining



Business understanding

- reformulate the problem in many ways, as necessary
- think about the scenario
- iterative refinement of problem formulation and scenario

Business understanding – Tasks I

- Determine
 - Business Objectives
 - Background Business Objectives
 - Business Success Criteria
- Assess Situation
 - Inventory of Resources
 - Requirements, Assumptions, and Constraints
 - Risks and Contingencies Terminology
 - Costs and Benefits

Business understanding – Tasks II

- Determine Goals
 - Data Mining Goals
 - Data Mining Success Criteria
- Produce Plan
 - Project Plan
 - Initial Assessment of Tools and Techniques

Data understanding

- which raw data are available?
 - they match rarely the problem needs
 - they are usually collected for different purposes (or for no purpose at all)
 - a customer database, a transaction database, and a marketing response database contain different information, may cover different intersecting populations, and may have varying degrees of reliability
- at which cost?
 - internal data are for free, external data may be not
 - interesting information may need to be collected with ad-hoc campaign
- possible forks in the project choices, according to the collected data

Data Understanding – Tasks

- Collect Initial Data
 - Initial Data Collection Report
- Describe Data
 - Data Description Report
- Explore Data
 - Data Exploration Report
- Verify Data Quality
 - Data Quality Report

Data preparation

- some analysis technique may require data transformations
 - converting to tabular format
 - converting between data types
 - e.g. from numeric to symbolic and viceversa
- some transformation can improve the quality of the results
 - normalization, scaling, guessing missing data, cleaning wrong data
 - ...
- *data leaks*
 - it is the case for supervised cases: the information necessary for the decision is not available at the decision time
- this task is usually very expensive and time consuming

Data Preparation – Tasks

- Data Set
 - Data Set Description
- Select Data
 - Rationale for Inclusion / Exclusion
- Clean Data
 - Data Cleaning Report
- Construct Data
 - Derived Attributes
 - Generated Records
- Integrate Data
 - Merged Data
- Format Data
 - Reformatted Data

Modeling

Capture patterns hidden in data



Modeling – Tasks

- Select Modeling Technique
 - Modeling Technique
 - Modeling Assumptions
- Generate Test Design
 - Test Design
- Build Model
 - Parameter Settings
 - Models
 - Model Description
- Assess Model
 - Model Assessment
 - Revised Parameter Settings

Evaluation

- rigorous assessment of the results of the data mining process
- compare different choices on a *qualitative* and *quantitative* basis
- evaluate the confidence of the derived models
- estimate the expected impact on the business
 - e.g. how many wrong decisions can we expect?
which will be the cost of wrong decisions?



Evaluation – Tasks

- Evaluate Results
 - Assessment of Data Mining results w.r.t Business Success Criteria
 - Approved models
- Review Process
 - Review of Process
- Determine next steps
 - List of possible actions
 - Decisions

Deployment

The results of the DM process (i.e. the models) are used in software systems to obtain some return of investments

- e.g. in *churn* analysis the model for predicting likelihood of churn can be integrated with a package for churn management, for instance sending special offers to selected customers considered *high-risk of churn*

Deployment – Tasks

- Plan Deployment
 - Deployment Plan
- Plan Monitoring and Maintenance
 - Monitoring and Maintenance Plan
- Produce Final Report
 - Final Report Final Presentation
- Review Project
 - Experience Documentation

Bibliography

- ▶ Shearer, C. (2000).
The CRISP-DM model: The new blueprint for data mining.
Journal of Data Warehousing, 5:13–22.
- ▶ Wirth, R. and Hipp, J. (2000).
CRISP-DM: Towards a standard process model for data mining.
Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining.