Machine Learning

Matteo Donati 

July 9, 2021

Contents

Chapter 1

Introduction

Information technology was born around the 1960 with early data collections and databases. In the following decades, database management systems were born and got mature. So people started to realize that data could even be analysed. Around 2000 the Big Data Explosion happened. Nowadays, data are used to improve strategies and learn.

1.1 Machine Learning

The application of methods used to extract patterns from data

Field of study that gives a computer the ability to improve its behaviour, without being explicitly programmed. The computer will learn from new data and examples.

1.1.1 Computing Tasks

Machine learning can be applied to the following computing tasks:

- Classification (i.e. given an individual, targeting it with a label).
- Regression, a linear (continuous, non-discrete) relationship between the input and the output.
- Similarity matching.
- Clustering (i.e. grouping individuals based on their similarities).
- Co-occurrence grouping (i.e. attempting to find associations between entities according to the transactions in which they appear together).
- Profiling (i.e. behaviour description).
- Link analysis (i.e. trying to infer missing connections from the existing ones).
- Data reduction (i.e. reducing data size but maintaining information).
- Casual modelling (i.e. understanding what events or actions actually influence others).

1.1.2 Business Problems

Machine learning can be applied to the following business problems:

- Decision support (e.g. market analysis, risk management, fraud detection.
- Data analysis (e.g. text mining, social mining, image analysis).
- Prediction.
- Advanced diagnosis and predictive maintenance.

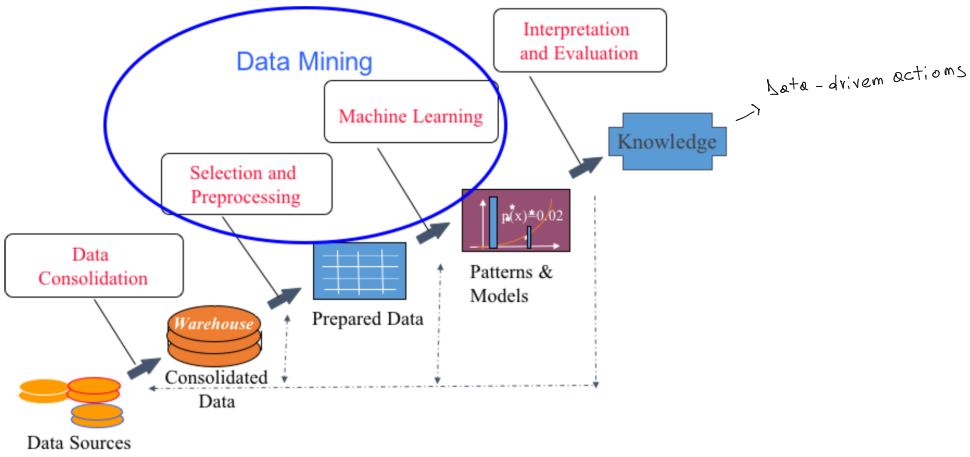
1.1.3 Types of Learning

There exist three main types of learning methods:

- Supervised methods, in which the user associates to each input a specific label, which corresponds to the expected output (e.g. classification and regression problems).
- Unsupervised methods, in which the user do not provide labels (e.g. clustering problems).
- Reinforcement learning methods, which, given a specific actor and a specific environment, are used to select the best set of actions to take in order to get back the maximum amount of reward (e.g. playing games).

1.2 Data Mining

Computational process which we organize in order to discover patterns in data. This science uses concepts and tools from artificial intelligence, machine learning, statistics, database management systems. The data mining process learns from data, thus it is considered a data-driven approach. The discovery process is composed as follows:



Chapter 2

The Data

A dataset is a set of N individuals, where each individual is described by D values. In essence, it can be seen as a relational table with N rows and D columns, or as a $N \times D$ matrix.
Data is raw material, therefore are never perfect. In particular, very often a small amount of data is different from the rest, due to anomalies/errors. Better data quality usually implies better insights, thus pre-processing activities are essential.

2.1 Data Types

Data can be expressed in various ways. It can be made of numbers, letters, classes, symbols. Generally, data contain numbers (quantitative information), and labels (qualitative information). In particular, when speaking about numerical data, there exist two main data types:

- **Interval data type**, where the difference is meaningful (calendar dates, temperatures on centigrades and Fahrenheit, etc.) and on which only the $+$ and $-$ operators can be applied.
- **Ratio data type**, which have a univocal definition of zero (Kelvin temperatures, masses, length, counts, etc.) and on which every mathematical operator can be applied.

Also when dealing with categorical data, it is possible to divide data into two main categories:

- **Nominal data type**, where the information allows us to distinguish a label from another one but not order them (male, female, etc.).
- **Ordinal data type**, where the values provide enough informations for a total ordering (low, medium, high, etc.).

2.1.1 Transformations on Data Types

- For interval data, one can use linear functions.
- For ratio data, one can use any mathematical function.

- For nominal data, one can apply every possible one-to-one correspondence (e.g. the Social Security Number can be arbitrarily reassigned).
- For ordinal data, one must preserve the order (e.g. low, medium, high can become 1, 2, 3).

2.1.2 Number of Values

One can distinguish between discrete domains and continuous domains. Nominal data and ordinal data are discrete, possibly binary. Interval data and ratio data are continuous.

2.1.3 Asymmetric Attributes

In asymmetric attributes, only the presence is considered important (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). Presence is more significant than absence.

2.1.4 General Characteristics of Datasets

Some of the main characteristics of datasets are the following:

- **Dimensionality**. There exist large data ($D \gg N$) and tall data ($N \gg D$). The difference is also qualitative: when D is big, some operations perform badly.
- **Sparsity**. A dataset is sparse when there exist many zero values or null values.
- **Disguised nulls**. Sometimes, the absence of value is represented using special values.
- **Resolution**. Data could be either too detailed and affected by noise or too general, in which case patters are hidden.

2.1.5 Data Representation

Data can be stored, visualized and manipulated by using different representations:

- **Data matrix**, which is a relational table with only numeric attributes. In this case, each row is a point in a vector space.
- **Document representation**, which is used with textual content. In this case, each row represents a document, each colmn represents a term and each cell contains the absolute frequency of the term in the document.
- **Transactional data**, where each record contains a set of objects (e.g. a cell can contain a list of objects).
- **Graph data**, which is useful when the data structure reminds the structure of a graph.
- **Ordered data**, which is useful when the data can be ordered.

2.2 Data Quality

Some examples of problems related to data are the following:

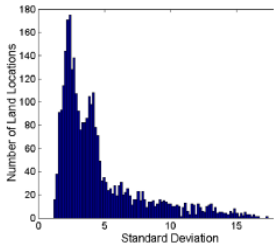
- **Noise**, which is a modification of the original value.
- **Outliers**, which are records that present considerably different characteristics from most of the data in the dataset.
- **Missing values**.
- **Duplicated data**.

2.3 Pre-Processing

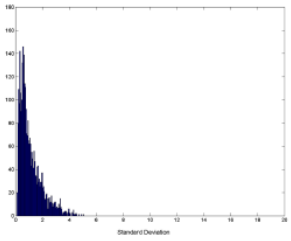
– used to manipulate data in order to obtain the best results when using a machine learning model

Below are the most important aspects related to pre-processing a given dataset:

- **Aggregation**, which consists in combining two or more attributes (or objects) into a single attribute (or object), and allows data reduction, change of scale (e.g. cities aggregated into regions, states, countries, etc.) and more stable data in general.



Standard Deviation of
Average Monthly Precipitation



Standard Deviation of
Average Yearly Precipitation

it is important that any possible sample shows the same property of the entire dataset

- **Sampling**, which is used because obtaining the entire dataset could be impossible or too expensive and because processing the entire dataset could be too expensive or time consuming. If the sample is representative (i.e. if it has approximately the same properties as the original dataset), using a sample will work almost as well as using the entire dataset. In particular, statistics provides techniques to asses the optimal sample size and the how much the sample is significant with respect to the given dataset (i.e. trade-off between data reduction and precision).

  Some of the existing types of sampling are listed below:

Simple random can be either with replacement or without replacement

  - **Simple random**, a single random choice of an object with given probability distribution.
  - **With replacement**, repetition of independent extractions where the extracted objects can be picked at each step.
  - **Without replacement**, repetition of extractions where the extracted element is removed from the population and, thus, can not be extracted in future steps.

subsets have to show the same properties

In this case discrimination on the basis of the distance becomes uneffective

  - **Stratified**, the dataset is split into several partitions according to some criteria and than random samples are picked from each partition.

- **Dimensionality**, when dimensionality is very high the occupation of the space becomes very sparse (curse of dimensionality). In order to avoid the curse of dimensionality, reduce noise, reduce time and memory complexity of mining algorithms and allow visualization, some useful techniques can be used:
  - **Principal Component Analysis (PCA)**, ...
  - **Singular values decomposition**, ...
  - **Supervised techniques**, ...
  - **Non-linear techniques**, ...

this allows dimensionality reduction

- **Feature subset selection**, which allows to remove redundant attributes (i.e. very similar attributes) and irrelevant attributes. This can be done either with brute force, trying every possible feature subset as input to the data mining algorithm, with an embedded approach, when the feature selection process occurs naturally as part of the data mining algorithm, with a filter approach, where the features are selected before the algorithm is run, or with a wrapper approach, where is the algorithm itself that chooses the best set of attributes.
- **Feature creation**, new features can capture more efficiently data characteristics. Feature creation can be achieved by feature extraction (e.g. calculating statistical features), by mapping to a new space (e.g. with transformations) and by adding new features (e.g. from volume and weight to density).

Discretization:
- from continuous to discrete (e.g. to define categories)
- from discrete to discrete (e.g. decreasing number of classes)

- **Discretization and binarization**, wich are used because some algorithms work better with categorical data. An example of binarization is the one-hot encoding used in classification problem with multiple classes.
- **Attribute transformation**, below are the three most important transformations:
  - **Mapping**: $x \rightarrow (x^k, \log(x), e^x, |x|)$. - change of distribution
  - **Standardization**: $x \rightarrow \frac{x-\mu}{\sigma}$. - mo " " "
  - **Normalization**: $x \rightarrow \frac{x-x_{min}}{x_{max}-x_{min}}$ (0 to 1) or $x \rightarrow \frac{x-\frac{x_{max}+x_{min}}{2}}{\frac{x_{max}-x_{min}}{2}}$ (-1 to 1). - mo " " "

MinMax scaling

2.4 Similarity and Dissimilarity

The quantities that can measure how alike two objects are, are the following:

- **Similarity**, which is a numerical measure of how alike two data objects are. The more the objects are alike the higher the similarity.

e.g. distances

- **Dissimilarity**, which is a numerical measure of how different two data objects are. The more the objects are different the higher the dissimilarity.

Below are some examples of similarity and dissimilarity calculated 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	$d = \frac{\lvert p-q \rvert}{V-1}$	$s = 1 - \frac{\lvert p-q \rvert}{V-1}$
Interval or Ratio	$d = \lvert p - q \rvert$	$s = \frac{1}{1+d}$ or $s = 1 - \frac{d - \min(d)}{\max(d) - \min(d)}$

2.4.1 Similarity Between Binary Vectors

Given two binary vectors, it is possibile to determine the following quantites:

- 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.

These quantities can be used to process the Simple **Matching Coeffcient**:

$$\text{SMC} = \frac{\text{number of matches}}{\text{number of attributes}} = \frac{M_{00} + M_{11}}{M_{00} + M_{01} + M_{10} + M_{11}} \tag{2.1}$$

and the **Jaccard Coefficient**:

$$\text{JC} = \frac{\text{number of 11 matches}}{\text{number of non-both-zero attributes}} = \frac{M_{11}}{M_{01} + M_{10} + M_{11}} \tag{2.2}$$

2.4.2 Extended Jaccard Coefficient (Tanimoto)

Given two vectors, p and q, it is possible to calculate the Jaccard Coefficient for continuous or count variables:

non-binary vectors

$$T(p, q) = \frac{p \cdot q}{||p||^2 + ||q||^2 - p \cdot q} \tag{2.3}$$

2.4.3 Cosine Similarity

Given two vectors, p and q:

$$\cos(p, q) = \frac{p \cdot q}{||p||||q||} \tag{2.4}$$

2.4.4 Euclidean Distance - L_2

Given the data objects p_d and q_d, each composed of D attributes, the L_2 distance between these two is:

$$dist = \sqrt{\sum_{d=1}^{D}(p_d - q_d)^2} \tag{2.5}$$

2.4.5 Minkowski Distance - L_r

Given the data objects p_d and q_d, each composed of D attributes, and given a parameter r (chosen depending on the application), the L_r distance between these two is:

$$dist = \left(\sum_{d=1}^{D}|p_d - q_d|^r\right)^{\frac{1}{r}} \tag{2.6}$$

In case $r = 1$, $r = 2$ or $r = \infty$, 2.6 computes respectively the L_1 distance, the L_2 distance or the L_∞ distance ($dist_\infty = \lim_{r\to\infty}\left(\sum_{d=1}^{D}|p_d - q_d|^r\right)^{\frac{1}{r}} = \max_d |p_d - q_d|$).

2.4.6 Mahalanobis Distance

This distance considers data distributions. In particular, 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.

$$\Sigma_{ij} = \frac{1}{N-1}\sum_{k=1}^{N}(e_{ki} - \bar{e}_i)(e_{kj} - \bar{e}_j) \tag{2.7}$$

$$dist_m = \sqrt{(p-q)\Sigma^{-1}(p-q)^T} \tag{2.8}$$

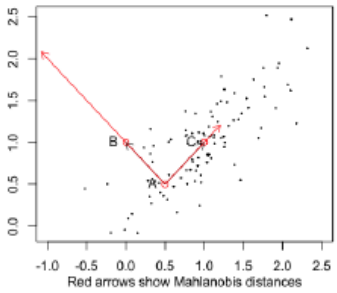
The following figure shows how the Mahalanobis distance relates points to the whole distribution instead of realting a point to another point:

$$\Sigma = \begin{bmatrix} 0.3 & 0.2 \\ 0.2 & 0.3 \end{bmatrix}$$
$$A = (0.5, 0.5)$$
$$B = (0, 1)$$
$$C = (1, 1)$$

The euclidean distances AB and AC are the same

$$\text{dist}_m(A, B) = 2.236068$$
$$\text{dist}_m(A, C) = 1$$

the direction on which B lies there are less points



2.4.7 Properties of a Distance — properties of vector norms

- **Positive definiteness**: $dist(p, q) \geq 0 \; \forall p, q$.
- **Simmetry**: $dist(p, q) = dist(q, p)$.
- **Triangle inequality**: $dist(p, q) \leq dist(p, r) + dist(r, q) \; \forall p, q, r$.

2.4.8 Correlation of Quantitative Data (Pearson's)

The correlation of quantitative data allows to measure the linear relationship between a pair of attributes. In order to obtain this metric it is necessary to:

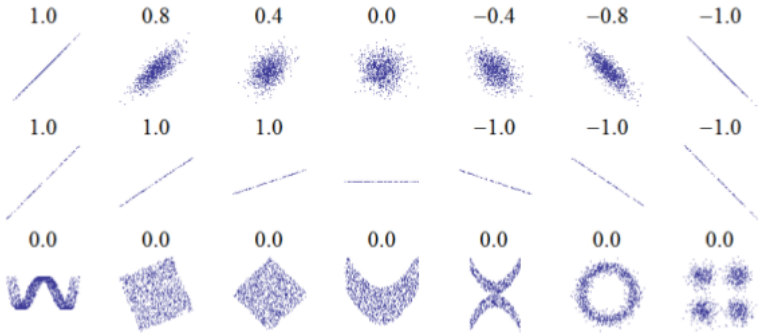
1. Standardize the value.
2. For two given attributes p and q, consider as vectors the ordered lists of the values over all the data records:

$$\mathbf{p} = [p_1, p_2, \ldots, p_N] \xrightarrow{standardize} \mathbf{p}'$$
$$\mathbf{q} = [q_1, q_2, \ldots, q_N] \xrightarrow{standardize} \mathbf{q}'$$

3. Compute the dot product of the vectors:

$$corr(p, q) = \mathbf{p}' \cdot \mathbf{q}' \tag{2.9}$$

The following figure shows an example of correlation between quantitative data:



2.4.9 Correlation Between Nominal Attributes

The correlation between nominal attributes is computed as follows:

$$U(p,q) = 2\frac{H(p) + H(q) - H(p,q)}{H(p) + H(q)} \tag{2.10}$$

In particular, $H()$ is the entropy of a single attribute while $H(,)$ is the joint entropy, computed from the joint probabilities.

Chapter 3

Classification

* : If one has a dataset with N elements, and one considers only two classes, then one has 2^N possible different learning problems. If a model $M(x, \theta)$ is able to shatter (i.e. to separate) all the possible learning problems, then the model has Vapnik-Chervonenkis Dimension equal to N.

3.1 Introduction to Classification

Let DS be a data set of N objects, each composed of D attributes. One of the D attributes is considered the class, i.e. a finite value provided by a supervisor. In a classification problem, the goal is to learn how to guess the value of the class for individuals which have not been examined by experts (not been labelled). *
A typical workflow for classification problems is the following:

1. Learning the model for the given set of classes, using a training set (i.e. a portion of the given data set for which the labels are available).
2. Estimate the accuracy of the model, using a test set (i.e. a portion of the given data set for which the labels are known).
3. The model is used to label new individuals.

Classification can be:

1. **Crisp**, when the classifier assigns to each individual one label.
2. **Probabilistic**, when the classifier assigns a probability for each of the possible labels.

3.1.1 Classification Model

It is an algorithm (a decision function $M(x, \theta)$) which, given an individual for which the class is not known, computes a guess of the class. The algorithm is parameterized in order to optimize the results for the specific problem at hand.

3.2 Classification with Decision Trees

A classifier structured as a decision tree is a tree-shaped set of tests. In particular, a decision tree has inner nodes and leaf nodes.

Inner nodes:

if test on attribute *d* of element *e* **then**

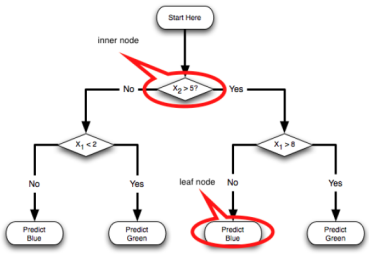
execute node'

else

execute node''

Leaf nodes:

predict class of element *e* as *c*''



Given a set $\mathcal{E}$ of elements for which the class is known, a decision tree is built as follows:

- If all the elements belong to class c or $\mathcal{E}$ is small, generate a leaf node with label c.
- Otherwise:
  - A test is chosen based on a single attribute with two or more outcomes.
  - The specific test becomes the root of a tree with one branch for each of the outcomes of the test.
  - The set $\mathcal{E}$ is partitioned into subsets corresponding to the outcomes and the procedure is recursively applied to all the subsets.

In order to select the attributes on which the test are based, it is necessary to introduce the concept of **entropy** from information theory.

3.2.1 Entropy

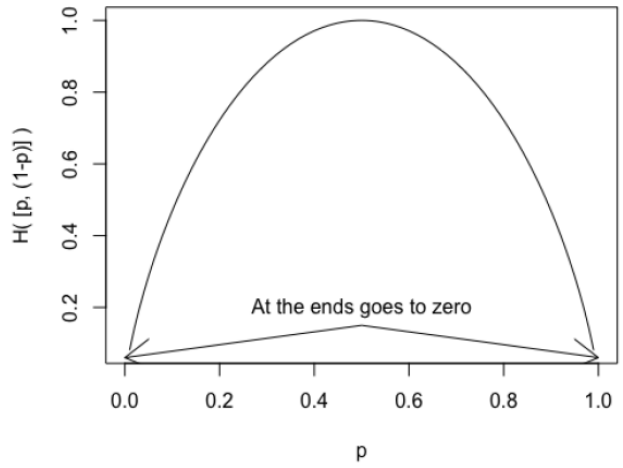
During a bit transmission, given a source X with V possible values, with probability distribution:

$$P(v_1) = p_1, P(v_2) = p_2, \ldots, P(v_V) = p_V \tag{3.1}$$

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

$$H(X) = -\sum_j p_j \log_2(p_j) \tag{3.2}$$

$H(X)$ is the entropy of the information source X. In particular, high entropy means that the probabilities are mostly similar, while low entropy means that some symbols have much higher probability than others. In the binary source case, where the symbol probabilities are p and $(1-p)$, when p is 0 or 1 the entropy goes to 0:



It is also possible to determine the conditional specific entropy $H(Y|X = v)$, i.e. the entropy of Y considering only the cases in which $X = v$. Considering the example below:

X	Y
Math	yes
History	no
CS	yes
Math	no
Math	no
CS	yes
History	no
Math	yes

$$H(Y) = 1$$
$$H(Y|X = \text{Math}) = 1$$
$$H(Y|X = \text{History}) = 0$$
$$H(Y|X = \text{CS}) = 0$$

Lastly, it is possible to define the conditional entropy $H(Y|X)$, i.e. the weighted average of the conditional specific entropy of Y with respect to the values of X:

$$H(Y|X) = \sum_j P(X = v_j)H(Y|X = v_j) \tag{3.3}$$

$H(Y|X)$ represents the average number of bits necessary to transmit the value of Y if both ends of the communication know the values of X.

v_j	$P(X = v_j)$	$H(Y|X = v_j)$
Math	0.5	1
History	0.25	0
CS	0.25	0

In the case above, $H(Y|X) = 0.5$, knowing X the entropy drops from 1 to 0.5. Therefore, X provides insight on Y.

* Let $d \in D$ be a real-valued attribute, let t be a value of the domain of d, let c be the class attribute, then the entropy of c with respect to d with threshold t as:

$H(c|d:t) = H(c|d<t) \cdot P(d<t) + H(c|d \geq t) \cdot P(d \geq t)$

1.2.2 Information Gain

The information gain, $IG(Y|X)$, is the amount of insight that X provides to forecast the values of Y. In particular:

$$IG(Y|X) = H(Y) - H(Y|X) \tag{1.4}$$

$IG(Y|X)$ represents the average number of bits which can be saved to transmit th values of Y if both of the communication ends know the values of X.
Therefore, to build a decision tree, it is necessary to test the attribute which guarantees the maximum IG for the class attribute in the current data set $\mathcal{E}$.

1.2.3 Errors and Overfitting

Given a decision tree, it is possible to execute the specific tree on a training set. The number of discordances between the true and the predicted class is called *training set error*. However, the *test set error* is more indicative of the expected behavior with new data. In particular, when a learning algorithm is affected by noise or when the training set lacks of representative instances, the performance on the test set is worse than that on the training set. This phenomena is called *overfitting*. In particular, considering a decision tree an *hypothesis* h of the relationship between the predictor attributes and the considered class, then h overfits the training set if there exists an alternative hypothesis h' such that:

$$error_{train}(h) < error_{train}(h') \tag{1.5}$$

$$error_{\mathcal{E}} > error_{\mathcal{E}}(h') \tag{1.6}$$

In order to prevent overfitting, it is possible to prune the decision tree. In particular, too much pruning allows one to correctly classify only some samples, while too little pruning classification is highly influenced by noise.

1.2.4 Complexity of Decision Trees

- If a given dataset $\mathcal{E}$ has N samples with D attributes each, then the tree heigh is $\mathcal{O}(\log N)$. Each level of the tree requires the consideration of all the dataset (considering all the nodes). Moreover, the each node requires the consideration of all the attributes. Thus, the overall cost is $\mathcal{O}(DN \log N)$. - Usually $D \ll N \rightarrow \mathcal{O}(N \log N)$
- The run-time use of a decision tree to classify new instances is extremely efficient: $\mathcal{O}(h)$, where h is the height of the tree. $\mathcal{O}(\log N)$

1.3 Model Selection: Evaluation of a Classifier

The model selection process allow to select the best available classification model, hence the best hyperparameters configuration.

In every step the data should be representative of the data that will be classified at run-time.

- Holdout · splitting data into train and test or into train, validation and test
- Cross validation · repeated tests with different splits.

5

```
Holdout (train, test):
- x_train, y_train, x_test, y_test = split_data(data)
- repeat the process changing hyper-parameters:
    - fit(x_train, y_train, model)
    - y_pred = predict(x_test)
    - quality_measure(y_test, y_pred)
- fit(X, y, model) expecting the best quality as lower bound.
```

```
Holdout (train, validation, test):
- x_train, x_val, x_test, y_train, y_val, y_test = split(data)
- repeat process changing h.p.:
  - fit(x_train, y_train, model)
  - y_pred = predict(x_val)
  - quality_measure(y_val, y_pred)
- qm_b = best(quality_measure)
- fit(x_train + x_val, y_train + y_val, model)
- y_pred = predict(x_test). quality_measure(y_test, y_pred)
- fit(X, y, model) expecting qm_b as lower bound
```

1.3.1 Performance Measures of a Classifier

To compare different algorithms it is necessary to define some useful metrics:

- **Error frequency** (e), is the sum of errors on any class divided by the number of tested records.
- **Accuracy** (A), is the complement of the error frequency ($A = 1 - e$) and can be calculated as follows:

$$A = \frac{TP + TN}{TP + TN + FP + FN} \tag{1.7}$$

- **Precision** (P), is the rate of true positives among the positive classifications and can be calculated as follows:

$$P = \frac{TP}{TP + FP} \tag{1.8}$$

- **Recall or Sensitivity** (R), is the rate of observable positives and can be calculated as follows:

$$R = \frac{TP}{TP + FN} \tag{1.9}$$

- **Specificity** (S), is the rate of observable negatives and can be calculated as follows:

$$S = \frac{TN}{TN + FP} \tag{1.10}$$

- **F-measure** (F), is the armonic mean of precision and recall and can be calculated as follows:

$$F = 2\frac{P \cdot R}{P + R} \tag{1.11}$$

Another useful tool which can be used to observe how well a specific classifier performs is the **confusion matrix**. In this matrix, each cell contains the number of test records of class i and predicted as class j, where i and j are the rows and columns indexes respectively:

T_x = true number of x labels in the dataset

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

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

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

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

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

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

$$F_{1_i} = 2\frac{P_i \cdot R_i}{P_i + R_i}$$

		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

Moreover one can define combination strategies:

- Micro:

$$P\_micro = \frac{\sum TP_i}{\sum TP_i + \sum FP_i}, \quad R\_micro = \frac{\sum TP_i}{\sum TP_i + \sum FN_i}$$

- Macro:

$$P\_macro = \frac{\sum P_i}{m.classes}, \quad R\_macro = \frac{\sum R_i}{m.classes}$$

- weighted:

$$P\_weighted = \frac{\sum P_i \cdot T_i}{N}, \quad R\_weighted = \frac{\sum R_i \cdot T_i}{N}$$

Moreover, one can compare two classification models by taking into account the different cost of the the different errors. For example, the false positives cost five times the false negatives, while the correct classification do not imply any cost. In this case, one could apply some cost sensitive learning method, which modifies the proportion of classes in the training data:

- **Random oversampling**, which consists in randomly duplicate examples in the more critical class.
- **Random undersampling**, which consists in randomly delete examples in the less critical class

In this way, the classifier increases the ability to classify the classes for which the classification error cost is higher. In order to assign such costs one can setup a cost matrix which has to be combined with the specific confusion matrix.
Given a specific confusion matrix, it is possible to evaluate the concordance between the model's predictions and the set of true labels given by the supervisor. Being that:

$$\mathrm{P}(c) = \frac{TP_a + TP_b + TP_c}{N} \tag{1.12}$$

is the probability of concordance, and:

$$\mathrm{P}(r) = \frac{TP_a \cdot P_a + TP_b \cdot P_b + TP_c \cdot P_c}{N^2} \tag{1.13}$$

is the probability of random concordance, it is possible to define a new metric, called κ, which represents the ratio between the concordance exceeding the random component and the maximum surplus possible:

$$-1 \leq \kappa = \frac{\mathrm{P}(c) - \mathrm{P}(r)}{1 - \mathrm{P}(r)} \leq 1 \tag{1.14}$$

In particular, $\kappa = 1$ in case of perfect agreement ($TP_a + TP_b + TP_c = N$) and $\kappa = -1$ in case of perfect disagreement ($TP_a + TP_b + TP_c = 0$ and there is a perfect swap between predictions and true labels)

1.3.2 Cross Validation (k-fold)

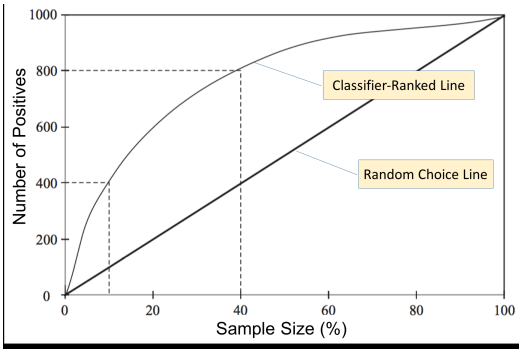
The records contained in the data set should be representative of the data that will be classified at run-time. One can do an holdout, which means that one can split the data into a training set, a testing set and eventually add a validation set. In order to have a cross validation, we can repeat the tests with different splits.
The cross validation method allows to randomly partition the ~~training set~~ supervised data into k subset. k iterations, each of one uses one of the subsets for testing and the others for training, are then used to train a specific model. At the end, one can average the error obtained at each iteration and, during each iteration, eventually compute all the metrics listed in section 1.3.1. This method is k times slower but provides an optimal use of the supervised data.

1.4 Evaluation of a Probabilistic Classifier

Many classifiers produce, rather than a class label, a tuple of probabilities (i.e. a probability distribution) composed of one probability for each possible class. Then, a prediction can be made by, for example, selecting the class with maximum probability. Im a binary environment, one can set a threshold.

1.4.1 Lift Chart

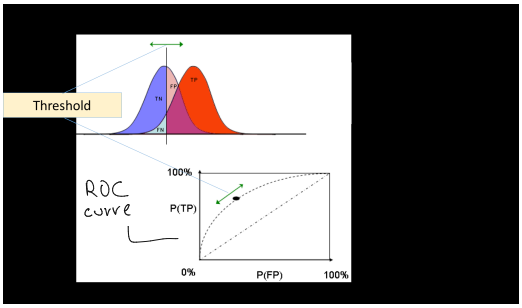
A lift chart can be used to evaluate various scenarios. If, for example, one considers a dataset with one-thousand positives and applies a probabilistic classifier, one can draw a bi-dimensional chart having the x axis reporting the sample size with respect to the total size of the population, and having the y axis reporting the positive samples. In particular, all the classified elements are sorted for decreasing probability of positive class.



In the above figure, the straight line plots the number of positives obtained with a random choice of a sample of test data, while the curve plots the number of positives obtained by drawing a fraction of test data with decreasing probability of positive class. When considering a lift chart, the larger the area between the two curves, the best is the classification model.

1.4.2 ROC Curve

The ROC curve describes a tradeoff between hit rate (i.e. true positives) and false alarm (i.e. false positives) in a noisy channel. Such noise can alter the two levels according to a Gaussian distribution. In order to distinguish between true positives and false positives, one need to set a threshold to best separate the two distributions:



Varying the threshold, the behaviour of the classifier changes (i.e. the ratio between true positives and false positives changes).

1.5 Transforming a Binary Classifier into a Multiclass Classifier

Several classifiers (e.g. SVM and linear perceptron), generate a binary classification model. In general, there exist two ways to deal with multi-class classification:

- Transforming the training algorithm and the model.
- Using a set of binary classifiers and combine the results.

c_1 c_2 c_3
⋮ ⋮ ⋮
I take 3 binary classifiers (one for c_1, c_2, one for c_1, c_3 and one for c_2, c_3).

1.5.1 One-Vs-One Strategy (OVO)

C = number of classes

This specific strategy allows to consider all the possible pairs of classes and generate a binary classifier for each pair ($C \cdot (C-1)/2$ pairs). In particular:

- Each binary problem considers only the examples of the two selected classes.
- At prediction time a voting scheme is applied:
  - An unseen example is submitted to the $C \cdot (C-1)/2$ binary classifiers, each winning class receives a $+1$.
  - The class with the highest number of $+1$ is chosen.

1.5.2 One-Vs-Rest / One-Vs-All Strategy (OVR / OVA)

A single binary classifier predicts either class c_i or every other c_j, $j = 1, \dots, |C|$, $j \neq i$

This specific strategy allows to consider C binary problems where class c is a positive example and all the others examples are negatives. Because of this, this strategy is unbalanced (i.e. there exist many more negatives than positives). In particular:

- C binary classifiers are built.
- At prediction time a voting scheme is applied:
  - An unseen example is submitted to the C binary classifiers, obtaining a confidence score.
  - The confidence are then combined and the class with the highest score is chosen.

× 1.5.1 One-Vs-One Strategy (OVO)

This specific strategy allows to consider all the possible pairs of classes and generate a binary classifier for each pair ($C \cdot (C-1)/2$ pairs). In particular:

- Each binary problem considers only the examples of the two selected classes.
- At prediction time a voting scheme is applied:
  - An unseen example is submitted to the $C \cdot (C-1)/2$ binary classifiers, each winning class receives a $+1$.
  - The class with the highest number of $+1$ is chosen.

× 1.5.2 One-Vs-Rest / One-Vs-All Strategy (OVR / OVA)

This specific strategy allows to consider C binary problems where class c is a positive example and all the others examples are negatives. Because of this, this strategy is unbalanced (i.e. there exist many more negatives than positives). In particular:

- C binary classifiers are built.
- At prediction time a voting scheme is applied:
  - An unseen example is submitted to the C binary classifiers, obtaining a confidence score.
  - The confidence are then combined and the class with the highest score is chosen.

1.6 Ensemble Methods

These methods allow to combine classifiers' results. In particular, instead of training a single classifier, multiple base classifiers are taken into consideration. The final prediction is than obtained considering the results of every classifier.
For example, given 25 independent binary classifiers, each of which has a error rate $e = 0.35$, than the ensemble will be wrong only when the majority of the classifiers are wrong:

$$e_{\text{ensemble}} = \sum_{i=13}^{25} \binom{25}{i} e^i (1-e^i)^{25-i} = 0.06 \tag{1.15}$$

majority (i.e. $\frac{25}{2}+1$)

1.6.1 Methods for Ensemble Classifiers

In order to obtain independent classifiers, different methods can be used:

- Manipulating the training set, i.e. train each of the classifiers on different portions of the original data set.

  Another option, useful when the number of classes is high, is to randomly partition class labels into two subsets, A_1, A_2 and re-label the data set. Than, at testing time, when a subset is selected, all the classes in the specific subset receive a vote; the class with the best score will than be chosen.

- Bagging, i.e. repeatedly samples with replacement according to a uniform probability distribution.
- Boosting, i.e. 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.

1.7 Statistical Modeling with the Naive Bayes Classifier

A Naive Bayes Classifier considers the contribution of all of the attributes of a specific object, assuming that each of these attributes is independent from the others (i.e. $\mathrm{P}(d_1 = v_1, d_2 = v_2 | c = c_x) = \mathrm{P}(d_1 = v_1 | c = c_x) \cdot \mathrm{P}(d_2 = v_2 | c = c_x)$).
In the figure below the weather / play problem is presented:

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	9	5
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	9/14	5/14
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								

Given a new sample, it is necessary to determine the probability of a *yes* and of a *no*. To do so, every feature is treated as equally important as the all the others. Since all the attributes are independent:

$$\text{likelyhood of } yes\text{: } \frac{2}{9} \cdot \frac{3}{9} \cdot \frac{3}{9} \cdot \frac{3}{9} \cdot \frac{9}{14} = 0.0053$$
$$\text{likelyhood of } no\text{: } \frac{3}{5} \cdot \frac{1}{5} \cdot \frac{4}{5} \cdot \frac{3}{5} \cdot \frac{5}{14} = 0.0206$$

These quantities get normalized to 1 to obtain the respective probabilities:

$$\mathrm{P}(yes) = \frac{0.0053}{0.0053 + 0.0206} = 20.5\%$$
$$\mathrm{P}(no) = \frac{0.0206}{0.0053 + 0.0206} = 79.5\%$$

The whole process works thanks to the Bayes Theorem: given a hypothesis H and an evidence E that bears on that hypothesis:

$$\mathrm{P}(H|E) = \frac{\mathrm{P}(E|H) \cdot \mathrm{P}(H)}{\mathrm{P}(E)} \tag{1.16}$$

In the case of a Naive Bayes Classifier, H is the class and E is the new sample to be classified. If the attributes are independent:

$$\mathrm{P}(c|E) = \frac{\mathrm{P}(E_1|c) \cdot \mathrm{P}(E_2|c) \ldots \mathrm{P}(E_D|c)}{\mathrm{P}(E)} \quad (1.17)$$

usually the denominator is not considered

These classifiers are subjects to problems whenever:

- A generic value v never appears in the elements of class c: $\mathrm{P}(d = v) = 0$. This makes the probability of the class, for that evidence, drop to zero. (i.e. the entire numerator is equal to zero)
- There exist missing values.

To deal with the first problem a smoothing technique is used. Given α, the smoothing parameter, $af_{d=v_i,c}$, the absolute frequency of value v_i in attribute d over class c, V the number of distinct values in attribute v_i over the data set, af_c, the absolute frequency of class c in the data set:

$$\text{Smoothed frequency } sf_{d=v_i,c} = \frac{af_{d=v_i,c} + \alpha}{af_c + \alpha V} \quad (1.18)$$

α makes the numerator always different than zero

On the other hand, to deal with the second problem, one can make the assumption that the values of a specific attribute have a Gaussian distribution (i.e. $f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$):

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	9/14	5/14
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											

In this last example, in order to calculate the frequency corresponding to Temperature = 66, it is necessary to apply the Gaussian probability density function:

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

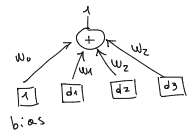
This value can be used to estimate the likelyhood of *yes* and then P(*yes*).

1.8 Linear Classification with the Perceptron

linear perceptron (or artificial neuron)

The linear perceptron implements a linear combination of weighted inputs. It is mostly used to learn a hyperplane that divides two classes. The hyperplane is described by a set of weights, $w_0, w_1, \ldots, w_D$, in a linear equation on the data attributes, $e_0, e_1, \ldots, e_D$ (e_0 is the bias):

$$w_0 \cdot e_0 + w_1 \cdot e_1 + \cdots + w_D \cdot e_D \tag{1.19}$$



The process which allows the learning of the hyperplane is:

```
set all weights to zero
while there are examples incorrectly classified do
  for each training instance e do
    if e is incorrectly classified then
      if class of e is positive then
        add the e data vector to the vector of weights
      else
        subtract the e data vector from the vector of weights
```

> 0 positive
< 0 negative

In order to avoid infinite loop one can set also a number of max iterations

$(w_0 + e_0)\,e_0 + (w_1 + e_1)\,e_1 + \cdots$

e and w have the same size

In particular, each change of weights moves the hyperplane towards the misclassified instance. The algorithm converges if the data set if linearly separable, otherwise it does not terminate.

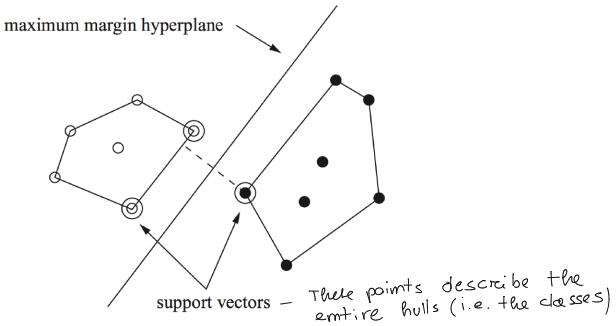
1.9 Support Vector Machines

Support Vector Machines are able to classify data even if the data set is not linearly separable. Moreover, these tools are able to find the *maximum margin hyperplane*, while the linear perceptron accepts any hyperplane able to separate the classes. – linear perceptron can find an infinite number of hyperplanes
In particular, the *convex hull* of a set of points is the tightest enclosing convex polygon (if the data set is linearly separable the convex hulls of the classes do not intersect). The maximum margin hyperplane is located as far as possible from both the hulls and it is the perpendicular bisector of the shortest line connecting the hulls.
If a separating hyperplane does not exist, it is suitable to find one, a *soft margin*, which almost separates the classes, and disregard examples generating a very narrow margin.

maximum margin hyperplane

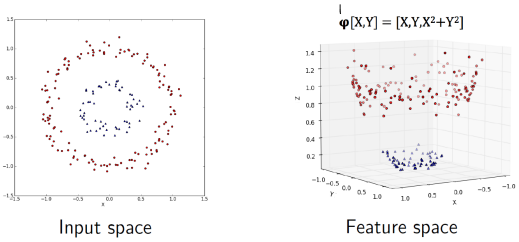
$$\max_{w_0, w_1, \ldots, w_b} M \quad \text{s.t.} \quad \sum_{j=1}^{b} w_j^2 = 1, \quad c^x \cdot (w_0 + w_1 x_1^x + \ldots + w_b x_b^x) > M \quad \forall x = 1, \ldots, N$$

class of example x (can be −1 or +1)

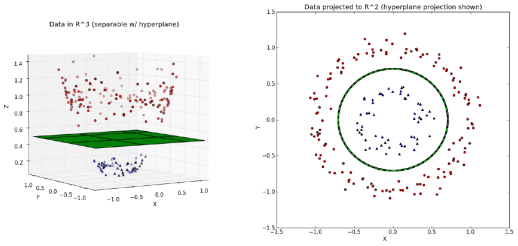


Moreover, if the data set is not linearly separable, the data are mapped into a new space, usually called *feature space*, such that a linear boundary in the feature space corresponds to a non-linear boundary in the original space. In particular, the feature space can have a number of dimensions higher than the original one.

non-linear mapping



another solution would be to disregard examples which generate a very narrow margin (soft margin)
This can be obtained by adding a constraint to the optimization problem expressed by a single numeric parameter C (represents regularization)
Soft margin and kernel concepts can be applied together



The separating hyperplane computation requires a series of dot product computations among the training data vectors. **Kernel functions** allow to do all the computation in the input space.

↳ from which the non-linear mapping is selected

instead of projecting every input point onto the feature space

1.10 Neural Networks – (only MLP one considered)

Neural Networks are hierarchical structures, composed by multiple layers of perceptron-like elements. These networks overcome the limit of linear decision boundary. In particular, these can be seen as function approximators, i.e. they can approximate a target function in order to solve regression and classification problems.

Each neuron of each input and hidden layer, is connected to other neurons, possibly every other neuron in the subsequent layer, by a weighted edge. All these weights are updated during the training steps. The output of the i-th neuron, is calculated as:

continuous and differentiable

$$y_i = f(b + \sum_{j=1}^{D} w_{i,j} x_j) \tag{1.20}$$

In particular, D is the number of neurons, in the previous layer, which are connected to the i-th considered neuron, $w_{i,j}$ is the weight that connects the j-th neuron of the previous layer to the i-th considered neuron, x_j is the output of the j-th neuron of the previous layer and f is a *activation* or *transfer* function.

* compute E(w) (error)
- compute derivatives and weight corrections for output layer.
- ↓ for hidden layer.

set all weights to random values
while termination condition is not satisfied **do**
  for each training instance e **do**
    feed the network with e and compute the output
    * compute the weight corrections
    propagate back the weight corrections

The training step is divided in epochs. In each epoch, every element of the given data set is fed into the neural networks. Moreover, many propagations can occur before updating the weights, accumulating errors over the samples within a batch. – stochastic version (weights are updated after every sample), batch version (weight are computed after N-batch samples by computing E_{avg} and ∇E_{avg})
In order to modify the weights as to reduce the error, given by a selected loss function, it is necessary to apply the backpropagation algorithm. This algorithm is based on the gradient descent method.

1.10.1 Gradient Descent

Gradient descent is an iterative and numerical procedure which allows to find, in the best case scenario, the global minimum of the specific loss function.
Given a loss function $C : \mathbb{R}^N \to \mathbb{R}$ of N real variables, $v = \{v_1, v_2, \ldots, v_N\}$, it is possible to determine how much each of these variables influence the output of C.
In particular, from the definition of a derivative:

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \tag{1.21}$$

If a small h is indeed chosen:

$$\frac{df(x)}{dx} \approx \frac{f(x+h) - f(x)}{h} \quad \Rightarrow \quad f(x+h) - f(x) = \Delta f \approx \frac{df(x)}{dx} h \tag{1.22}$$

Therefore, substituting f with the considered loss function C and h with the variables' increments yields:

$$\Delta C \approx \frac{\partial C}{\partial v_1}\Delta v_1 + \frac{\partial C}{\partial v_2}\Delta v_2 + \cdots + \frac{\partial C}{\partial v_N}\Delta v_N \tag{1.23}$$

If $\Delta v = (\Delta v_1, \Delta v_2, \ldots, \Delta v_N)$ is the vector containing the variables' increments and $\nabla C = (\frac{\partial C}{\partial v_1}, \frac{\partial C}{\partial v_2}, \ldots, \frac{\partial C}{\partial v_N})$ is the gradient of C, than:

$$\Delta C \approx \Delta v \cdot \nabla C \tag{1.24}$$

In order to make ΔC negative (i.e. minimizing the error) it is necessary to consider a specific value for Δv:

$$\Delta v = -\eta \nabla C \quad \Rightarrow \quad \Delta C \approx -\eta \nabla C \cdot \nabla C \tag{1.25}$$

At this point, ΔC has to be negative since $-\eta(\nabla C)^2$ is always negative. η is called *learning rate* and determines the magnitude of the step to take in the opposite direction of the loss function's gradient in order to minimize the error.
Lastly, the value of each variable is updated as follows:

$$v = v - \eta \nabla C \tag{1.26}$$

1.10.2 The Backpropagation Algorithm

The backpropagation algorithm aims at finding the gradient of the loss function with respect to each of the trainable parameters of the network (i.e. the weights and biases) and then updating each parameter according to the 1.26 update rule.

1.11 K Nearest Neighbours Classifier

- The model is the entire training set
- Future predictions by computing the similarity between the new sample and each training instance
  - Pick the K entries in the dataset which are closest to the new sample
  - Majority vote.
- Main parameters:
  - K
  - Metric used to compute the distance

Chapter 4

Clustering

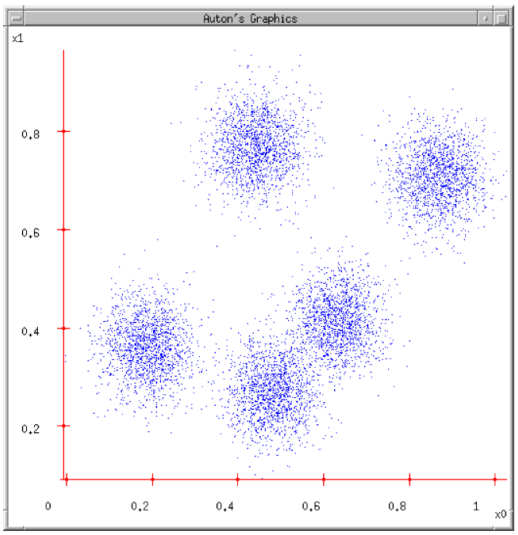
Given a set of N objects, each described by D values, could be necessary to find a natural partitioning, of the given set, in K clusters and, possibly, a number of noise objects. The result of such operation, is a *clustering scheme*, i.e. a function mapping each data object to the corresponding cluster (or to noise).

4.1 Partitioning Clustering

Partitioning clustering methods allow to partition a given data set according to some defined algorithm.

4.1.1 K-means

Given a data set, the K-means algorithm is used to partition the given data set into K clusters, based on the objects' attributes. For example, given the following data set:



the algorithm proceeds as follows:

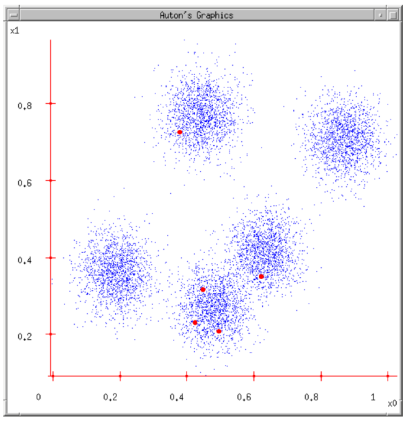
This number is usually guessed

1. The number of clusters, K, to be found, is asked to the user.
2. The coordinates of the K temporary centres are randomly chosen.
3. Each of the K points finds his nearest centre.
4. For each centre, the centroid* (i.e. the centre of gravity) of the specific cluster's points is found.
5. The centre of the cluster is set to coincide with the centroid.

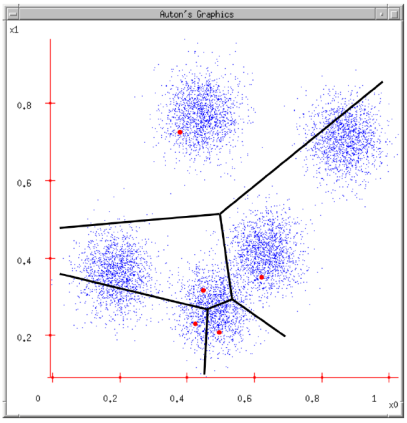
$*\ centroid_d^K = \sum_{x:clust[x]=K} x_d$

The procedure above is then iterated. Inertia = sum of square distances between points and corresponding centroid

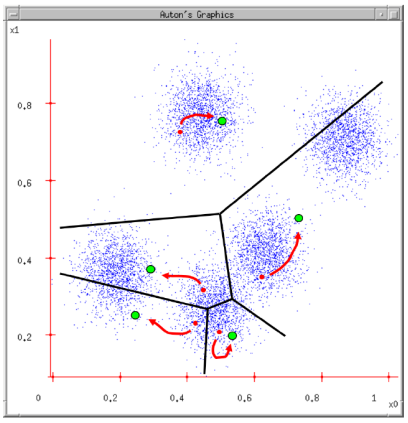
This can be computed at each iteration



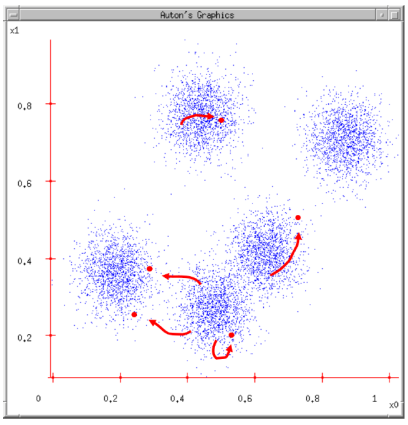
2.



3.



4.



5.

The algorithm uses a coding function and a decoding function:

$$labelling - Encode : \mathbb{R}^D \to \overbrace{[1, \ldots, K]}^{labels} \tag{4.1}$$

$$finds\ centroid - Decode : [1, \ldots, K] \to \mathbb{R}^D \tag{4.2}$$

From these two functions, it is possible to define a distortion function, which is basically the SSE (sum of squared errors):

$$\sum_{i=1}^{N}(e_i - Decode(Encode(e_i)))^2 \Rightarrow \sum_{i=1}^{N}(e_i - c_{Encode(e_i)})^2, \quad \text{with } c_k = Decode(k) \tag{4.3}$$

(point; each point is encoded using the corresponding centroid; inertia; centroid)

In order to obtain minimal distortion, the following properties must hold:

1. e_i must be encoded with the nearest centre:

$$c_{Encode(e_i)} = \text{argmin}_{c_j \in \{c_1, \ldots, c_k\}}(e_i - c_j)^2 \tag{4.4}$$

2. The partial derivative of distortion with respect to the position of each centre must be zero:

$$\begin{aligned}
\text{Distortion (SSE)} &= \sum_{i=1}^{N}(e_i - c_{Encode(e_i)})^2 \\
&= \sum_{j=1}^{K} \sum_{i \in OwnedBy(c_j)} (e_i - c_j)^2 \\
\frac{\partial \text{Distortion}}{\partial c_j} &= \frac{\partial}{\partial c_j} \sum_{i \in OwnedBy(c_j)} (e_i - c_j)^2 \\
&= -2 \sum_{i \in OwnedBy(c_j)} (e_i - c_j) \\
&= 0 \quad \text{i.e. when distortion is minimum.}
\end{aligned} \tag{4.5}$$

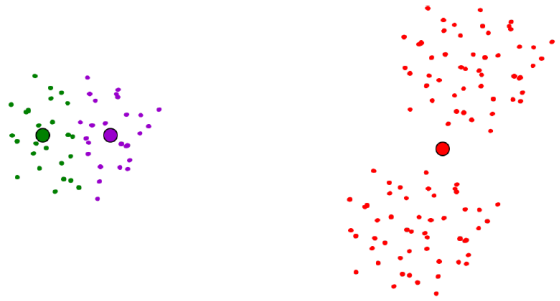
In particular, when distortion is minimal: (solution of $\frac{\partial distortion}{\partial c_j} = 0$)

$$c_j = \frac{1}{|OwnedBy(c_j)|} \sum_{i \in OwnedBy(c_j)} e_i \tag{4.6}$$

Therefore, each centre must correspond with the centroid of the points it owns.

~~The distortion function is a convex function, thus the algorithm is guaranteed to terminate.~~ As a matter of fact, the algorithm converges whenever the configuration reached at some iteration do not change in the consecutive iterations.
It can also happen that, even if the algorithm converges, the ending state is not the best possible:

Complexity:
K-mean : $O(TKND)$
T: number of iterations
K: " " clusters
N: " " data points
D: " " attributes of each data point



4.2 Evaluation of a Clustering Scheme

Evaluating a clustering scheme is done by considering different measurement criteria.

4.2.1 Cohesion

Cohesion is the sum of the proximity between the elements of the cluster and the geometric centre (the higher, the better):

$$Coh(k_i) = \sum_{x \in k_i} Prox(x, c_i) \tag{4.7}$$

4.2.2 Separation Between Clusters

The separation between clusters is the distance between two clusters, i.e. the proximity between the geometric centres:

$$Sep(k_i, k_j) = Prox(c_i, c_j) \tag{4.8}$$

4.2.3 Global Separation of a Clustering Scheme

The global separation of a clustering scheme is computed by considering the global centroid of the dataset, c, and applying the following weighted sum of distances:

$$SSB = \sum_{i=1}^{K} N_i Dist(c_i, c)^2 \tag{4.9}$$

4.2.4 Silhouette Index of a Cluster

The silhouette for the i-th object is computed as follows:

$$s_i = \frac{b_i - a_i}{\max(a_i, b_i)} \in [-1, 1] \tag{4.10}$$

In particular:

- a_i is the average distance, of the i-th object of the cluster, from the other objects of the same cluster.
- b_i is the minimum average distance, of the i-th object of the cluster, from all the other clusters.

When $s_i = 1$ than the object is very well clustered, while if $s_i = -1$ the object is not well clustered. This quantity can be computed for each object of the given data set and than averaged over the entire set. → one could plot inertia and silhouette score with respect to the number of clusters. Moreover, computing the silhouette score can be used to find the best number of clusters → when sil. score is maximum.

4.2.5 Supervised Measures

Let the **gold standard**, $P = \{P_1, \ldots, P_L\}$, be a given partition of the data set, and considering the clustering scheme obtained after using some clustering method, it is then possible to compare the gold standard with the obtained clustering scheme. In particular, any pair of objects can be labelled as:

- ~~*SS*~~ if they belong to the same set in P and K. (SGSK (same G, same K))
- ~~*SD*~~ if they belong to the same set in K but not in K. (SGDK)
- ~~*DS*~~ if they belong to the same set in P but not in K. (DGSK)
- ~~*DD*~~ if they belong to different sets both in P and K. (DGDK)

→ this creates a confusion matrix:

	SK	DK
SG	a	b
DG	c	d

Let a, b, c, d be the number of pairs in the four categories above, then:

$$\textbf{Rand Index } R = \frac{a+d}{a+b+c+d} = \frac{SGSK + DGDK}{SGSK + SGDK + DGSK + DGDK} \quad (4.11)$$

- Adjusted Rand Index: same concept of K-score. Takes into account matches due to chance

$$\textbf{Jaccard Coefficient } J = \frac{a}{a+b+c} = \frac{SGSK}{SGSK + SGDK + DGSK} \quad (4.12)$$

can be computed for each class

4.3 Hierarchical Clustering

Hierarchical clustering generates a nested structure of clusters. This can be achieved in two main ways:

- By using a bottom up approach:
  - 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,

  An example of this is the *single linkage hierarchical clustering* method.

Algorithm:
1. Initialize clusters.
2. Compute distance matrix between clusters.
3. While n. classes > 1:
   a. Find two clusters with lowest separation, K_r and K_s.
   b. Merge them into one cluster.
   c. Update distance matrix.

$dist(K_k, K_{(s+r)}) = \min(dist(K_k, K_r), dist(K_k, K_s))$

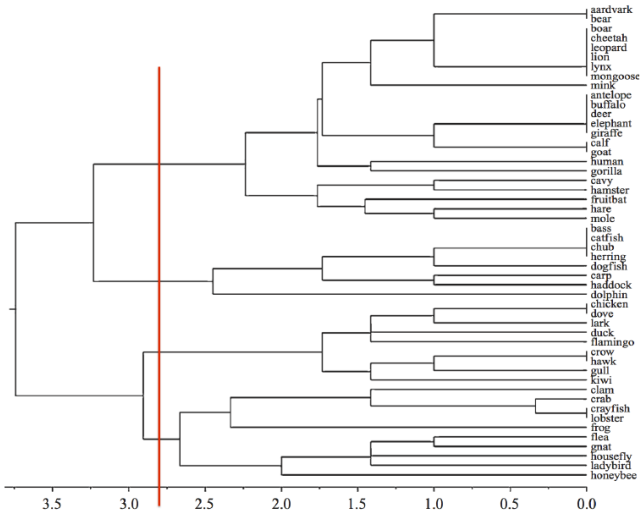
$\forall k \in [1, K]$ — new cluster

- By using a top down approach:
  - As a starting state, the entire data set is the only cluster.
  - In each step, the cluster with the lowest cohesion is split.

Separation between clusters:
- Single link: $Sep(K_i, K_j) = \min_{x \in K_i, y \in K_j} dist(x, y)$
- Complete link: $= \max$
- Average link: $= \frac{1}{|K_i| \cdot |K_j|} \sum_{x \in K_i} \sum_{y \in K_j} dist(x, y)$
- other measures (distance between centroids, Ward's method)

– A measure of cluster cohesion and a split procedure are needed.

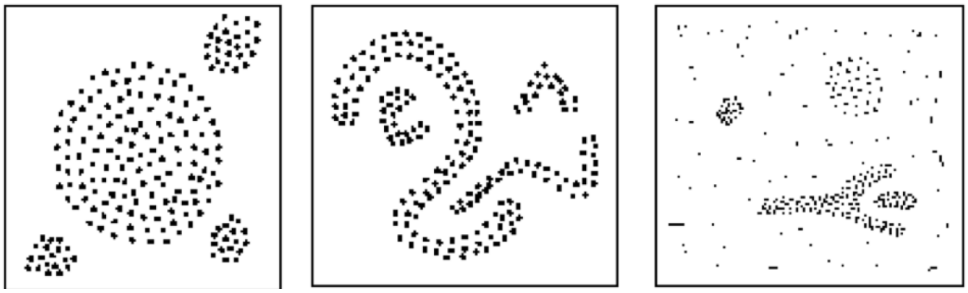
The result of a hierarchical clustering method, such as the *single linkage hierarchical clustering*, is a *dendogram*, which is a hierarchical structure that can be cut at some point to produce the final clustering scheme:



In particular, the horizontal axis represents the total dissimilarity inside the clusters, which increases for decreasing number of clusters, and the diameter of a cluster is the distance among the most separated objects.

4.4 Density Based Clustering

Density based clustering methods are based on the fact that clusters are high-density regions separated by low-density regions.



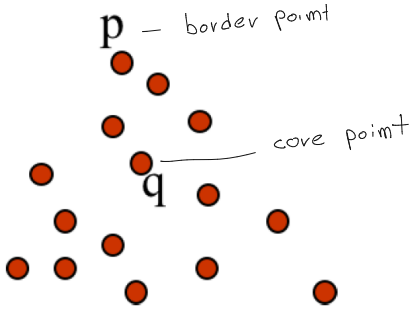
The two most used solutions are the following:

- Grid-based, in which the (hyper)space is split into a regularly spaced grid. For each grid element, the number of objects inside the considered element is then counted.

- Object-centred, in which one defines the radius of a specific (hyper)sphere and attaches to each object the number of objects which are located inside the specific sphere.

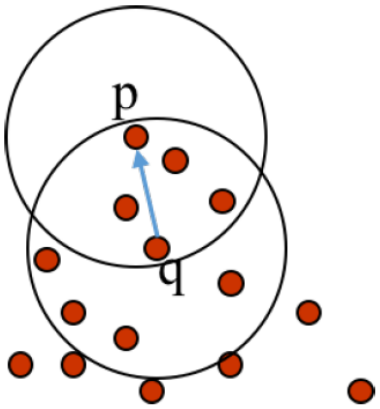
4.4.1 Density Based Spatial Clustering of Applications with Noise (DBSCAN)

Given a set of objects, each located at specific coordinates in some hyperspace:



it is possible to determine which one of these is a border point and which one is a core point. As a matter of fact, given a radius ϵ it is possible to determine the neighbourhood of a specific point, i.e. the ϵ-hypersphere centred at the specific point. Given a specific threshold, *minPoints*, points with at least *minPoints* points in their neighbourhood are defined as core points, as border points otherwise. A particular point p is *directly density reachable* from point q if and only if:

- q is core.
- q is in the neighbourhood of p.

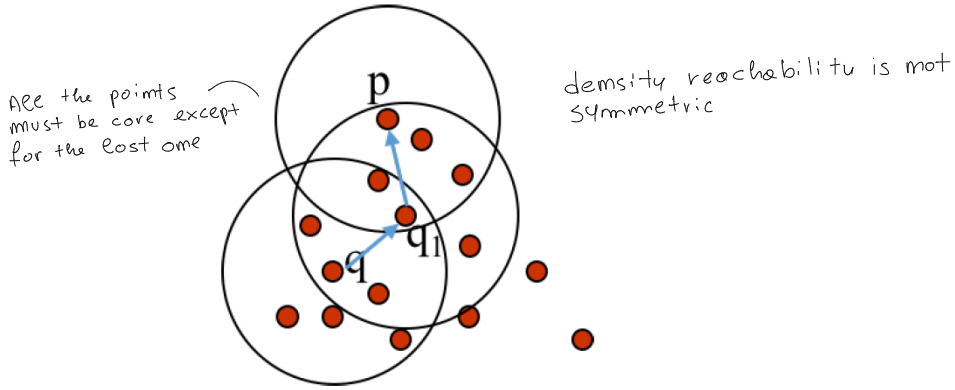


Moreover, a particular point p is *density reachable* from point q if and only if:

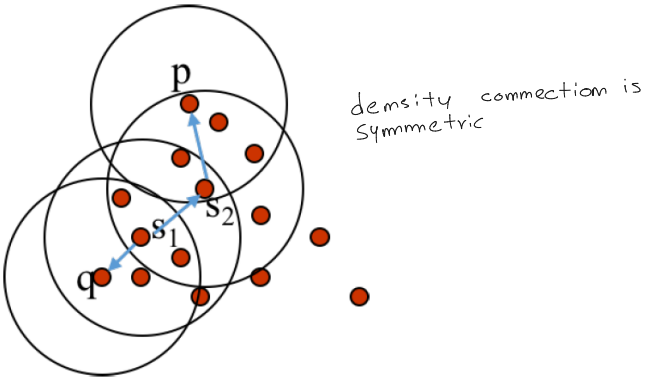
- q is core.

* Kernel Density Estimation:
- Describe the distribution of the data by a function (e.g. a Gaussian)
- The overall density function is the sum of the kernel functions
  Kernel function must be symmmetric and monotonically decreasing
- DENCLUE algorithm (DBSCAN is a special case of DENCLUE)

- There exist a sequence of points, starting from point q_i such that q_{i+1} is directly density reachable from $q_i \in [1, nq]$, q_1 is directly density reachable from q and p is directly density reachable from q_{nq}.



Lastly, a particular point p is density connected to a point q if and only if there exist a point s such that p and q are density reachable from s.



At the end, 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.

*

4.5 Model Based Clustering

Model based clustering estimates the parameters of a statistical model (e.g. Gaussian distribution) in order to maximize the ability of this model to explain the data. These models allow to interpret the data as a set of observations from a mixture of different probability distributions. Usually Gaussian distributions are used. The estimation of these parameters is usually computed by applying the maximum likelihood principle. — $\arg\max_{\theta} P(X \mid \theta)$ (dataset; parameters of the distribution) — parameters which maximize the probability of observing a given dataset

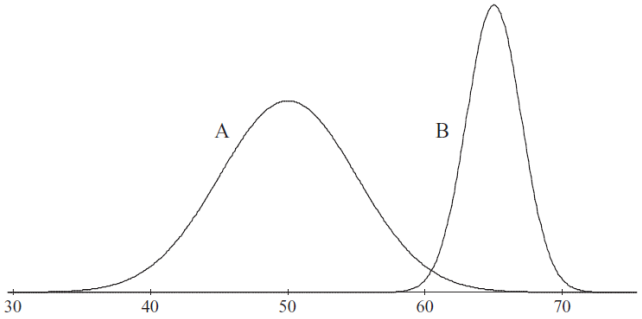
4.5.1 Expectation Maximization (EM) (Gaussian mixture)

This algorithm is used to estimates the models' wanted parameters. In particular, if the data can be approximated by a single distribution, the derivation of the parameters is straightforward, otherwise the following steps need to be applied:

if one has more clusters, than X is modelled through multiple distributions

1. Selection of an initial set of model parameters.
2. Expectation step, in which it is necessary to calculate the probability that each object belong to each distribution.
3. Maximization step, in which, given the calculated probabilities, it is necessary to find the new estimates of the parameters that maximize the expected likelihood.
4. Until the estimated parameters do not change, it is necessary to go back to point 2.

1. For example, given two Gaussian distributions and a set of initial parameters:



$$\mu_A = 50, \sigma_A = 5, p_A = 0.6$$
$$\mu_B = 65, \sigma_B = 2, p_B = 0.4$$

it is necessary to estimate five different parameters: mean and variance for cluster A (μ_A, σ_A), mean and variance for cluster B (μ_B, σ_B), the sampling probability p for cluster A (the sampling probability for cluster B is given by $1 - p$). (how many samples are in cluster A (the proportion of the dataset samples which fall in A))

2. The probability of value A for point x is computed as: (Bayes theorem)

$$P(A|x) = \frac{P(x|A)P(A)}{P(x)} = \frac{f(x; \mu_A, \sigma_A)p_A}{P(x)}, \quad \text{with} \quad f(x; \mu_A, \sigma_A) = \frac{1}{\sqrt{2\pi}\sigma_A} e^{-\frac{(x-\mu_A)^2}{2\sigma_A^2}} \tag{4.13}$$

3. Once $P(A|x)$ and $P(B|x)$ are computed, x is assigned to the cluster with the maximum probability. At this point, p_A in computed and the means and variances are updated by multiplying each data point with a weight (i.e. the probability of belonging to the specific distribution):

P(A|x) and P(B|x) need to be normalized using (P(A|x) + P(B|x))

$$\mu_A = \frac{w_1 e_1 + w_2 e_2 + \ldots w_N e_N}{w_1 + w_2 + \cdots + w_N} \tag{4.14}$$

$$\sigma_A = \frac{w_1(e_1 - \mu_A)^2 + w_2(e_2 - \mu_A)^2 + \cdots + w_N(e_N - \mu_A)^2}{w_1 + w_2 + \cdots + w_N} \tag{4.15}$$

Chapter 5

Association Rules

An association rule is an expression of the form $A \Rightarrow C$, where A is the antecedent, C is the consequant and both are itemsets (collections of items).

5.1 Market Basket Analysis

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.

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

{beer, bread} ⇒ {milk}

Example of association rules are {Diaper} ⇒ {Beer}, ~~{Bread, Milk} ⇒ {Coke, Eggs}.~~

In particular, it is possible to define the following rule evaluation metrics:

- **Support**, which is the fraction of the N transactions that contain both A and C. For example, considering {Milk, Diaper} ⇒ {Beer}.

support count

Independent from direction of the arrow

$$sup = \frac{\sigma(\text{Milk, Diaper, Beer})}{N} = \frac{2}{5} \tag{5.1}$$

- **Confidence**, which measures how often all the items in C appear in transactions that contain A. For example, considering {Milk, Diaper} ⇒ {Beer}.

$$conf = \frac{\sigma(\text{Milk, Diaper, Beer})}{\sigma(\text{Milk, Diaper})} = \frac{2}{3} \tag{5.2}$$

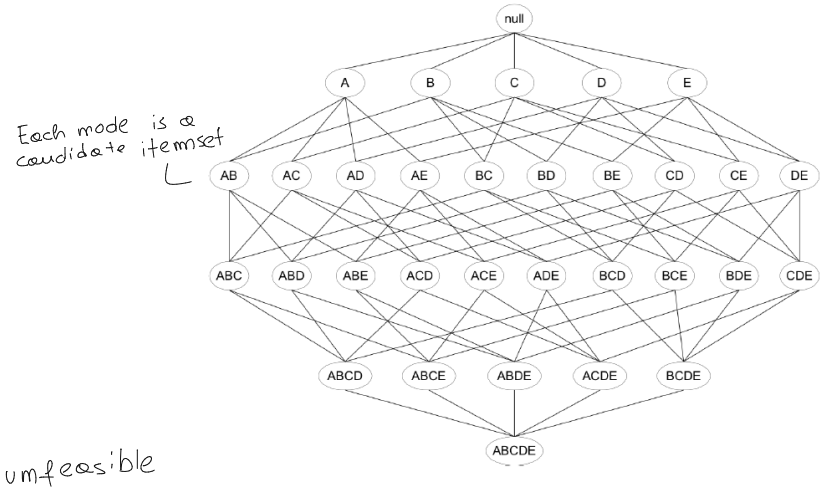
5.2 Association Rules Mining

Given a set of N transactions, the goal of association rule mining is to find every rule which has: $sup \geq minsup$ threshold and $conf \geq minconf$ threshold.
The mining of association rules is done by following these two steps:

1. **Frequent Itemset Generator**, which is a method that generates all itemsets whose support is greater than $minsup$.
2. **Rule Generation**, which generate high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset.

5.2.1 Frequent Itemset Generator

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



A first brute-force approach computes the support for each candidate by scanning the entire database. The complexity of this approach is $\mathcal{O}(NWM)$:

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

← W →

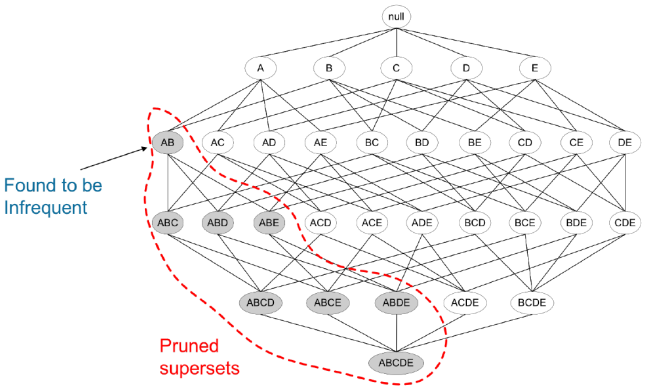
The most useful frequent itemset generation strategies reduce the number of candidates M using pruning techniques and reduce the number of comparisons NM.

* Maximal frequent itemset: frequent itemset with no frequent supersets.

Apriori Principle

This principle states that if an itemset is frequent, then all of its subsets must also be frequent. This holds due to the following anti-monotone property of support:

$$\forall X, Y : (X \subseteq Y) \Rightarrow sup(X) \geq sup(Y) \tag{5.3}$$



This principle allow pruning. For example, minsup=3:

C_1

Item	Count
Beer	3
Bread	4
Coke	2
Diaper	4
Eggs	1
Milk	4

candidates of the first layer

The support of {Coke} and {Eggs} is below minsupp, therefore they do not generate C_2 candidates →

C_2

Item	Count
Beer,Bread	2
Beer,Diaper	3
Beer,Milk	2
Bread,Diaper	3
Bread,Milk	3
Diaper,Milk	3

(repeat process until $|C_k| \neq 0$)

No C_3 candidate will include {Beer, Bread} or {Beer, Milk} →

C_3

Item	Count
Bread,Diaper,Milk	2

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

with pruning = 13

FP-Growth Algorithm

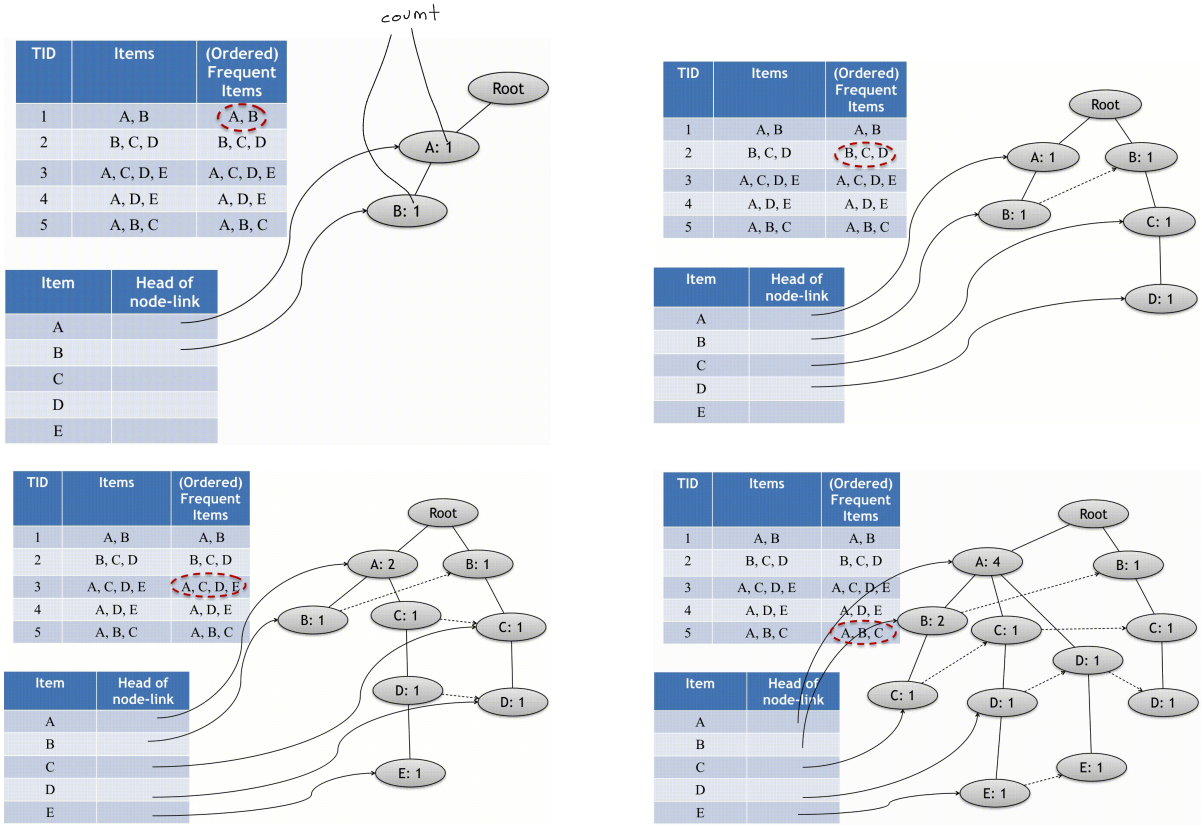
This algorithm is an alternative to the apriori one. In particular, it transforms the problem of finding long frequent patterns into looking for shorter ones and then concatenating the suffix. This is done by using a compressed representation of the database using a FP-tree.

frequent-pattern-tree

1. scan the database to find the support of 1-itemsets

itemsets composed of one element

2. create the root for the FP–tree
3. scan the database and for each transaction
   1. reorder the items for descending support
   2. focus on the root node of the FP-tree
   3. for each item in the transaction
      1. if it matches one of the descendants of the current node, move to it and increment the count, else create a new descendant node with count 1



5.2.2 Rule Generation

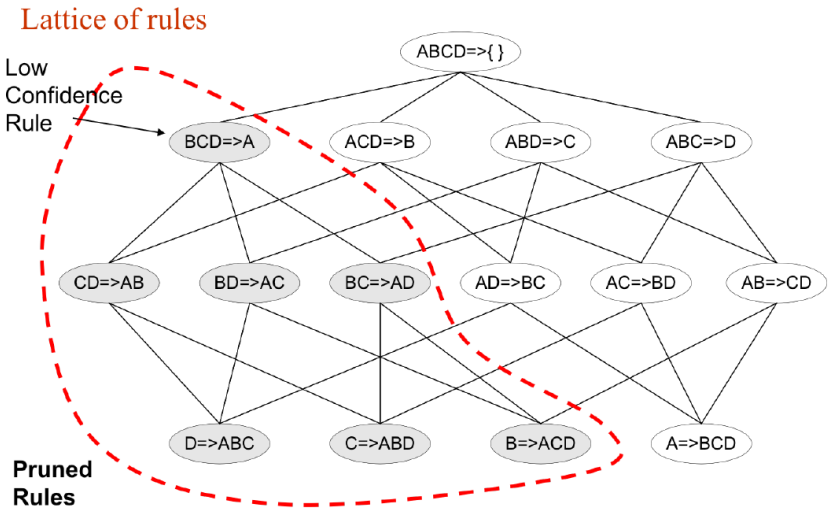
Given a frequent itemset, L, it is possible to find all the non-empty subsets, f, of L such that the confidence of the rule $f \Rightarrow (L - f)$ is not less than the minimum confidence. The confidence of a rule can be computed from the supports:

$$conf(A \Rightarrow C) = \frac{sup(A \Rightarrow C)}{sup(A)} \tag{5.4}$$

For example, from {Beer, Diaper, Milk} the possible rules are: Beer, Diaper $\Rightarrow$ Milk, $Beer \Rightarrow Diaper, Milk$, $Beer, Milk \Rightarrow Diaper$, etc. In particular, if $|L| = k$, there exist $2^k - 2$ candidate rules ($L \Rightarrow \emptyset$ and $\emptyset \Rightarrow L$ can be ignored).

Moreover, the confidence of rules generated from the same itemset is anti-monotone with respect to the number of items on the right hand side of the rule (i.e. decreases when one moves an item from the left hand side to the right hand side). Once a low confidence rule has been found, all of its descendants can be pruned:

The association rule algorithms tend to produce too many rules and many of them are uninteresting or redundant. In particular, if $A, B, C \Rightarrow D$ and $A, B \Rightarrow D$ have same support and confidence, then there exist redundancy. To solve this problem, interestingness measures can be computed. Given a rule, $A \Rightarrow C$, the information needed to compute rule interestingness can be obtained from a contingency table:



f_{xx} = frequency

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

not antecedent

statistical independence

In particular, the *lift* measure represents the ratio of true cases with respect to independence and is computed as follows:

$$lift(A \Rightarrow C) = \frac{conf(A \Rightarrow C)}{sup(C)} = \frac{P(A,C)}{P(A)P(C)} \tag{5.5}$$

if A and C are independent: $P(A,C) = P(A)P(C)$

The *leverage* measure represents the number of additional cases with respect to independence and is computed as follows:

$$leve(A \Rightarrow C) = sup(A \cup C) - sup(A)sup(C) = P(A,C) - P(A)P(C) \tag{5.6}$$

The *conviction* measure represents the deviation of incorrect predictions with respect to independence.

$$conv(A \Rightarrow C) = \frac{1 - sup(C)}{1 - conf(A \Rightarrow C)} = \frac{P(A)(1 - P(C))}{P(A) - P(A,C)} \tag{5.7}$$

For example, given the following contingency table:

	$Coffee$	$\overline{Coffee}$	
Tea	15	5	20
$\overline{Tea}$	75	5	80
	90	10	100

it is possible to compute the afore-mentioned measures from the rule Tea ⇒ Coffee:

$$conf = \frac{0.15}{0.20} = 0.75$$
in a 0 to 1 scale it is apparently high

$$lift = \frac{0.15}{0.90*0.20} = 0.83$$
is less than 1, therefore not interesting

$$leve = 0.15 - 0.90 * 0.20 = -0.03$$
is less than 0, therefore not interesting

$$conv = \frac{1-0.9}{1-0.75} = 0.4$$
is low, remembering that absolute truth gives *infinite*

however, the absence of tea increases the probability of coffe

5.2.3 Multilevel Association Rules

In working with large databases it is important to choose the right level of abstraction (i.e. tradeoff between general and detailed reasoning). If the records are stored in a hierarchical way, then it is possible to generate rules either from a more specialized point of view or from a more general point of view. In general:

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

if specialized rule has almost the same confidence as the general rule, then the specialized rule is redundant.

In this setting, mining multilevel association rules can be done by using a top down strategy, i.e. one can look for frequent itemsets at each level of abstraction and generate new rules.

* Examples of filter methods:
  - Pearsom's correlation: quantifying linear dependence (correlation) between X and Y.
  - LDA: linear discriminant analysis, find linear combination which separate classes.
  - ANOVA: analysis of variance, similar to LDA but operates on categorical and numerical data.
  - CHI-SQUARE: statistical test applied to groups of categorical features to evaluate the likelihood of correlation between them

Chapter 6

Feature Selection, creation

or feature creation

Feature selection can sometimes enable machine learning algorithm to train faster, can reduce the complexity of a model and can improve the accuracy of a model. Feature selection can be either supervised or unsupervised.

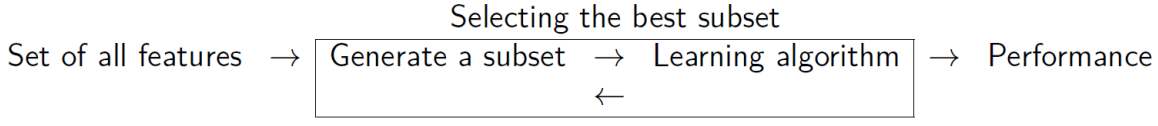
6.1 Filter Methods *

These methods allow the selection of a specific subset (the best subset of features) of attributes independently from the mining model. These methods make use of statistical methods for the evaluation of a subset. These methods try to compute correlation between attributes and target.

Set of all features → Selecting the best subset → Learning algorithm → Performance

6.2 Wrapper Methods

These methods try to use a subset of features to train a specific model. Based on the results of the trained model, one can decide to add or remove specific features from the subset in order to find the best possible subset of features. These methods make use of cross validation.



map dataset into a new space with fewer attributes. Attributes will be ordered based on variance. Attributes with most variance is the most interesting.

6.3 Principal Component Analysis (PCA) — unsupervised

This method allows to find a new (ordered) set of dimensions that better captures the variability of the data. In particular, in order to find the best possible subset of features, new attributes are added so

as to capture most of the remaining variability of the data. The fraction of variance in data captured by each new feature is measured. In most cases, a small number of new features can capture most of the variability.

6.4 Multi-Dimensional Scaling (MDS)

Multi-dimensional scaling is more of a presentation technique. Starting from the distances among the elements of the data set, this method fits the projection of the elements into a m-dimensional space in such a way that the distances among the elements are preserved. Changing the number of dimensions allows the user to interpret the data in a different way. \ can be used also for mominal attributes

6.5 Baseline Estimator

Given a variance threshold, this method allows the removal of features with low variance. This is an unsupervised approach.

6.6 Univariate Feature Selection

In this case, the best set of features is selected based on univariate statistical tests. For each feature, a *score* and a *pvalue* is returned.

6.7 Recursive Feature Elimination (RFE)

Recursive feature elimination works by searching for a subset of features by starting with all features in the training dataset and successfully removing features until the desired number remains. This is achieved by fitting the given machine learning algorithm used in the core of the model, ranking features by importance, discarding the least important features, and re-fitting the model. This process is repeated until a specified number of features remains. Features are scored ~~either~~ using external estimators which assign weights to the features.

Chapter 7

Regression

Regression is a supervised task in which the target variable is numeric. The objective of regression is to minimize the error of the prediction with respect to the target. In particular, given a dataset X with N rows and D columns:

- $\boldsymbol{x}_i$ is a D-dimensional **feature vetor**.
- The **response vector** $\overline{\boldsymbol{y}}$ (i.e. true labels) is a row vector containing N entries, one for each $\boldsymbol{x_i}$.
- $\boldsymbol{w}$ is a D-dimensional row vector of coefficients that needs to be learned.
- The dependence of each response value y_i from the corresponding independent variable $\boldsymbol{x}_i$ is modelled as:

$$y_i \approx \boldsymbol{w} \cdot \boldsymbol{x_i} \; \forall i \in [1, \dots, N] \tag{7.1}$$

  such that the error of modelling is minimized.

- The considered objective function is the following:

$$\mathcal{L} = \sum_{i=1}^{N} (\boldsymbol{w}\boldsymbol{x}_i - y_i)^2 = \|X\boldsymbol{w}^T - \overline{\boldsymbol{y}}\|^2 \tag{7.2}$$

  In particular, the gradient of $\mathcal{L}$ is given by $2X^T(X\boldsymbol{w}^T - \overline{\boldsymbol{y}})$, which, if set equal to zero, allows one to find $\boldsymbol{w}$ by solving a linear system:

$$2X^T(X\boldsymbol{w}^T - \overline{\boldsymbol{y}}) = 0 \quad \Rightarrow \quad X^T X\boldsymbol{w}^T = X^T\overline{\boldsymbol{y}} \tag{7.3}$$

  In particular, $\boldsymbol{w}$ can be obtained if and only if the matrix $X^T X$ is invertible.

- The forecast of the model is computed as:

$$\boldsymbol{y}^f = X \cdot \boldsymbol{w} \tag{7.4}$$

Moreover, the quality of the fitting can be computed using different metrics:

- Mean of the observed data:

$$y_{mean} = \frac{1}{N} \sum_i y_i \tag{7.5}$$

- Sum of squared residuals:

$$SS_{res} = \sum_i (y_i - y_i^f)^2 \tag{7.6}$$

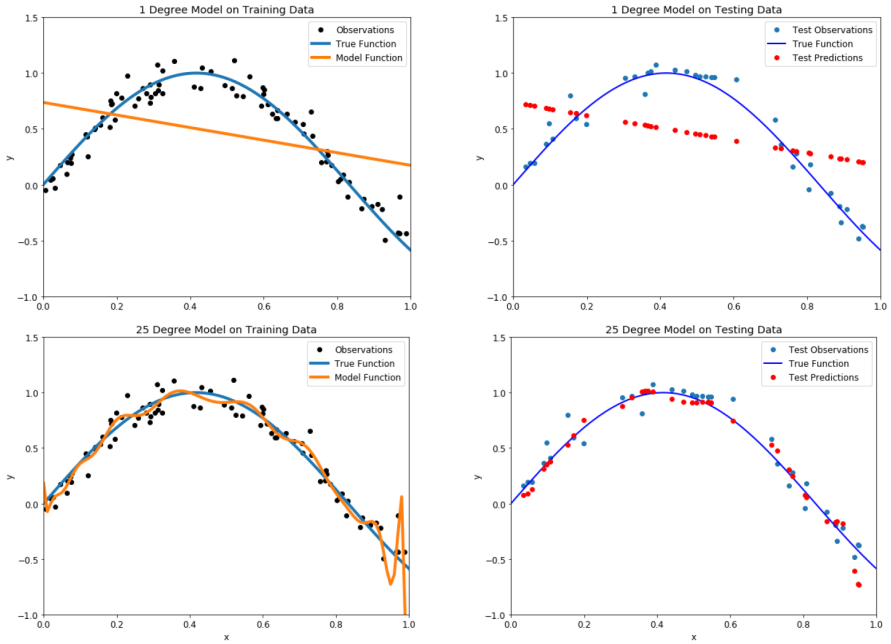
- Total sum of squares:

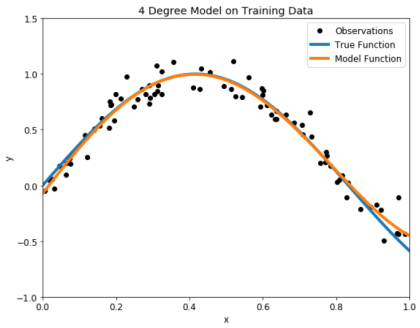
$$SS_{tot} = \sum_i (y_i - y_{mean})^2 \tag{7.7}$$

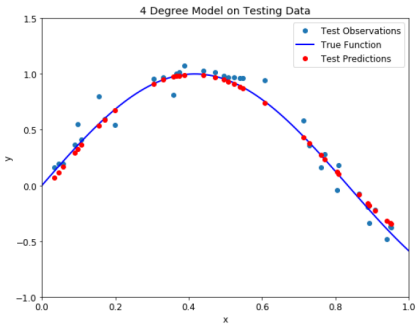
- Coefficient of determination:

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} \tag{7.8}$$

However, it could happen that, in presence of high number of features **overfitting** is possible (i.e. the performance on test data becomes much worse). Regularization reduces the influence of less interesting attributes and therefore reduces overfitting. An example of this problem is given by the polynomial regression task:




4 Degree Model on Training Data
Observations
True Function
Model Function
x
y


4 Degree Model on Testing Data
Test Observations
True Function
Test Predictions
x
y

Chapter 8

Outliers Detection

Outliers (or anomaly) detection is the problem of finding objects that are different from most of the other objects. In particular, an outlier if an observation that differs so much from other observations as to arouse suspicion that it was generated by a different mechanism. Moreover, when data can be modelled by a normal distribution, most objects are near the centre while extreme values have low likelihood. There exist various approaches to anomaly detection:

- Model-based techniques, which consist in building a model of the data (i.e. estimating the parameters of a probability distribution). In this case, outliers will fit poorly in the model.
- Proximity-based techniques, in which anomalous objects are those that are distant from most of the other objects.
- Density-based techniques, in which density is computed using a proximity measure. Objects in low-density regions can be considered outliers.

Moreover, the anomaly detection task can be solved via supervised or unsupervised methods:

- Considering a supervised setting, one usually has a training set where objects are labelled as normal or anomalous.
- Considering an unsupervised setting, it is necessary to learn a way to assign an anomaly score to each object.
- Considering a semi-supervised setting, one usually has a training set containing only normal objects. The anomaly score is computed using only the normal object information.

8.1 Model-Based Detection

From a statistical point of view, an outlier is an object that has a low probability with respect to a probability distribution model of the data. In particular, such probability distribution is usually defined by estimating its parameters (e.g. Gaussian, Poisson, Binomial). In practice, one usually models the data at hand by using a mixture of different distributions, usually of the same type but

with different parameters. In particular, given a Gaussian distribution $\mathcal{N}(0,1)$, an object can be defined as outlier if $|x| \geq c$, where $P(|x|) \geq c = \alpha$. In this case, α is the probability of false positive detection (i.e. is the probability that a regular object is labelled as outlier).

8.2 Proximity-Based Detection

An object is anomalous if it distant from the majority of the points. This techniques rely on proximity measure. In particular:

- The outlier score of an object is the distance to its k-nearest neighbours. This anomaly score is very sensitive to the choice of k.
- Another definition is given by Knorr and Ng. Given a positive $R \in \mathbb{R}$ and $k \in \mathbb{N}$, an object is a **distance-based outlier** if less then k objects lie within distance R from the selected object.

A way of applying such techniques is given by brute-force searching ($\mathcal{O}(N^2)$). However, there exist much more efficient algorithms. An of these algorithms is given by the **Bay's algorithm**:

1. For each example X keep track of the k-nearest neighbours found so far.
2. Determine the **cutoff** value of the score as the distance of the k-th nearest neighbou of the top m-th outlier found so far.
3. When an example achieves a score lower than the cutoff value it is removed, because it can no longer be an outlier.

In particular, later iterations find increasing scores, and the efficiency of pruning increases. If data are in random order, the average complexity is near linear. In the worst case is $\mathcal{O}(N^2)$.

8.3 Density-Based Detection

This techniques are based on the fact that outliers are found in low-density areas. In this case, the outlier score of an object is the inverse of the density around the object:

$$\text{density}(x,k) = \left(\frac{\sum_{y \in \text{Nb}(x,k)} \text{distance}(x,y)}{k} \right)^{-1} \tag{8.1}$$

where $\text{Nb}(x,k)$ is the set containing the k-nearest neighbours of x. Al alternative way of computing density is to count the number of objects that are within a specified distance d (given radius) from the object. However, when data contains regions of different densities, problems related to the identification of outliers arise. In order to avoid so, the average relative density can be computed ($\mathcal{O}(N^2)$):

$$\text{average relative density}(x,k) = \frac{\text{density}(x,k)}{\sum_{y \in \text{Nb}(x,k)} \text{density}(y,k)/k} \tag{8.2}$$

This score represents the density of x normalized with the average density of the objects in the neighbourhood of x.