

## Content

1	Organisation .....	3
1.1	Topics.....	3
1.2	PÜ & PL.....	3
1.3	Schedule.....	4
2	Introduction.....	5
2.1	Overview .....	5
2.2	Origin .....	5
2.3	Tasks and applications .....	5
2.4	The Data Mining Process .....	7
3	Cluster Analysis.....	9
3.1	Overview .....	9
3.2	K-Means Clustering.....	9
3.3	Density-based Clustering.....	11
3.4	Hierarchical Clustering.....	12
3.5	Proximity Measures.....	13
4	Classification .....	16
4.1	Overview .....	16
4.2	K-Nearest-Neighbors.....	16
4.3	Decision Trees.....	17
4.4	Model Evaluation .....	21
4.5	Rule Learning .....	24
4.6	Naïve Bayes .....	26
4.7	Support Vector Machines (SVMs) .....	28
4.8	Artificial Neural Networks (ANNs).....	29
4.9	Hyperparameter Selection.....	29
5	Regression.....	32
5.1	Overview .....	32
5.2	KNN for Regression.....	32

5.3	Model Evaluation .....	32
5.4	Regression Trees.....	33
5.5	Linear Regression.....	33
5.6	Polynomial Regression.....	34
5.7	Local Regression .....	35
5.8	ANNs for Regression.....	35
5.9	Time Series Forecasting.....	35
5.10	The Bias/Variance-Tradeoff.....	36
6	Text Mining .....	38
6.1	Overview .....	38
6.2	Text Preprocessing.....	38
6.3	Feature Creation / Generation .....	39
6.4	Feature Selection .....	39
6.5	Pattern Discovery .....	40
7	Association Analysis.....	42
7.1	Correlation Analysis .....	42
7.2	Association Analysis .....	42

# 1 Organisation

## 1.1 Topics

<b>1. Introduction to Data Mining</b>	What is Data Mining? Tasks and Applications The Data Mining Process
<b>2. Cluster Analysis</b>	K-means Clustering, Density-based Clustering, Hierarchical Clustering, Proximity Measures
<b>3. Classification</b>	Nearest Neighbor, Decision Trees and Forests, Rule Learning, Naïve Bayes, SVMs, Neural Networks, Model Evaluation, Hyperparameter Selection
<b>4. Regression</b>	Linear Regression, Nearest Neighbor Regression, Regression Trees, Time Series
<b>5. Text Mining</b>	Preprocessing Text, Feature Generation, Feature Selection
<b>6. Association Analysis</b>	Frequent Item Set Generation, Rule Generation, Interestingness Measures

## 1.2 PÜ & PL

- Project work: teams of six realize a data mining project. 10 page summary, presentation
- Grading: 75% written exam. 20 % project report, 5% presentation of project results
- In person exam; 60 min; 6 open questions, not about python – only conceptual

### **Lecture:** video recordings + Q&A **Exercise:**

sessions due to Corona

- Wednesday, 10.15 - 11.45, ZOOM

- Thursday, 10.15 - 11.45,  
Room A 104 (B6 , Bauteil A) – 16 places
- Thursday, 12.00 - 13.30,  
ZOOM (online) – unrestricted places
- Thursday, 13.45 - 15.15,  
Room A 104 (B6 , Bauteil A) – 16 places

## 1.3 Schedule


Week	Wednesday	Thursday
16.02.2022	<b>Lecture: Introduction to Data Mining</b> <b>Tutorial: Introduction to Python</b> (see below table)	Exercise: Preprocessing/Visualization
23.02.2022	Video Lecture: Cluster Analysis	Exercise: Cluster Analysis
02.03.2022	Video Lecture: Classification 1	Exercise: Classification
09.03.2022	Video Lecture: Classification 2	Exercise: Classification
16.03.2022	Video Lecture: Classification 3 <b>Question and Answer Session 1</b>	Exercise: Classification
23.03.2022	Video Lecture: Regression	Exercise: Regression
30.03.2022	Video Lecture: Text Mining	Exercise Text Mining
06.04.2022	Video Lecture: Association Analysis <b>Introduction to the Student Projects</b> <b>and Group Formation</b> <b>Question and Answer Session 2</b>	Exercise Association Analysis Preparation of Project Outlines
	- Easter Break -	
27.04.2022	<b>Feedback on Project Outlines</b>	Project Work
04.05.2022	<b>Feedback on demand</b>	Project Work
11.05.2022	<b>Feedback on demand</b>	Project Work
25.05.2022	<b>Feedback on demand</b>	Project Work
29.05.2022	Submission of project reports (Deadline: 23:59)	
01.06.2022	<b>Presentation of project results</b>	<b>Presentation of project results</b>
	Final exam (offline)	


**Bold** = session takes place live via zoom

## 2 Introduction

### 2.1 Overview

- Data mining helps us discover patterns in large quantities of data and use them for decision making

 Data Mining is the exploration & analysis of large quantities of data in order to discover meaningful patterns.

 It is the non-trivial extraction of implicit, previously unknown and potentially useful information from data.

Data Mining methods:

- Detect interesting patterns in large quantities of data
- **Support** human decision making by providing such patterns
- **Predict** the outcome of a future observation based on the patterns


### 2.2 Origin

Data Mining combines ideas from statistics, machine learning, artificial intelligence and database systems. It tries to overcome shortcomings of traditional techniques concerning:

- Large amount of data
- High dimensionality of data
- Heterogeneous and complex nature of data
- Explorative analysis beyond hypothesize-and-test paradigm


### 2.3 Tasks and applications

- Descriptive tasks: find patterns in the data (which products are often bought together?)
- Predictive tasks: predict unknown values of a variable (will sb. click on a online ad?)

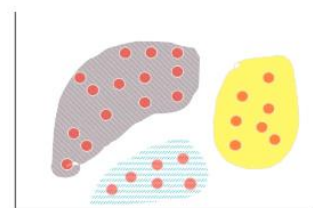
 descriptive = unsupervised; predictive = supervised

1. Cluster Analysis: descriptive
2. Classification: predictive
3. Regression: predictive
4. Association analysis: descriptive

#### 2.3.1 Cluster Analysis

 Given a set of data points, each having a set of attributes, and a similarity measure among them, find groups that:

- Data points in one group are more similar to one another
- Data points in separate groups are less similar to one another



💡 Goals are to minimize **intra**-cluster distances and maximize **inter**-cluster distances.

The result is a descriptive grouping of data points without knowing anything about the data itself.

A possible application could be in **market segmentation** where the goal is to find groups of similar customers. Groups may be conceived as a marketing target to be reached with a distinct marketing mix.

Another possible application could be **document clustering** where the goal is to find groups of documents that are similar to each other based on terms appearing in them (e. g. grouping articles in Google News).

## 2.3.2 Classification

📄 Previously **unseen** records should be assigned a class from a given set of classes as accurately as possible.

- Given a collection of records (**training set**)
  - Each record contains a set of attributes
  - One attribute is the class attribute (**label/target attribute**) that should be predicted
- Find a **model** for predicting the class attribute (label/target attribute) as a function of the values of other attributes.



"tree"



"tree"



"tree"



"not a tree"



"not a tree"



"not a tree"

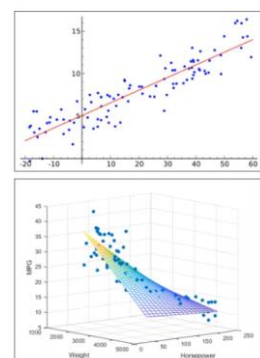
A possible application could be in **fraud detection** where the goal is to predict fraudulent cases in credit card transactions.

## 2.3.3 Regression

📄 Predict a value of a **continuous variable** based on the values of other variables, assuming a linear or nonlinear model of dependency.

- Predicting sales amount based on advertising expenditure
- Pred. miles per gallon of a car as a function of its weight and HP
- Pred. wind veloc. as a function of temp., hum., air pressure, etc.

Difference to classification: target attribute is continuous and not nominal!



## 2.3.4 Association analysis

Given a set of records each of which contain some number of items form a given collection:

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

Frequent Itemsets  
 {Diaper, Milk, Beer}  
 {Milk, Coke}

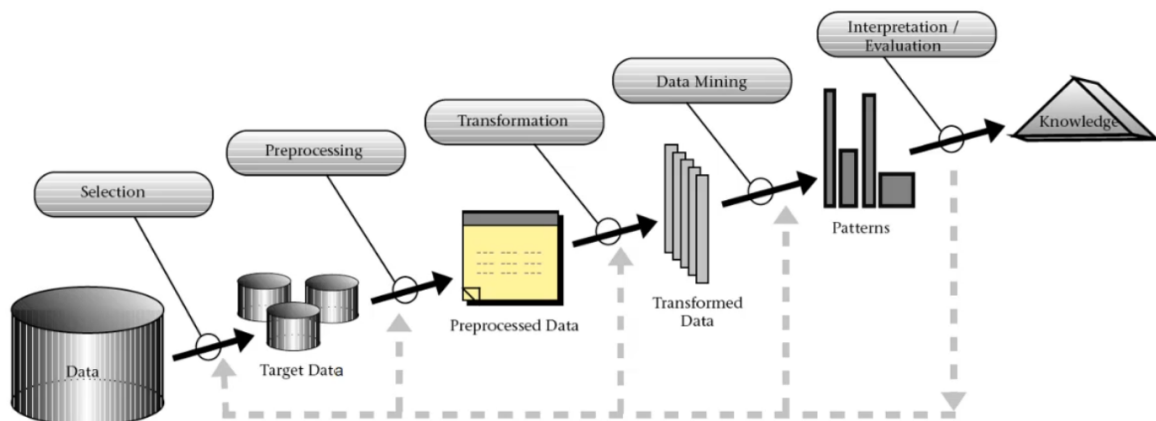
Association Rules  
 {Diaper, Milk} --> {Beer}  
 {Milk} --> {Coke}

Discover **frequent itemsets** and produce **association rules** which will predict occurrence of an item based occurrences of other items.

A possible application could be in **shelf management (supermarket)** where the goal is to identify items that are frequently bought together.

Another possible application could be **inventory management** where the goal is that for example a consumer appliance repair company wants to anticipate the nature of repairs on its consumer products and keep the service vehicles equipped with right parts to reduce on number of visits to consumer households.

## 2.4 The Data Mining Process



Source: Fayyad et al. (1996)

### 2.4.1 Selection and Exploration

Selection:

- What data is potentially useful for the task at hand?
- What data is available?
- What do I know about the quality of the data?

Exploration:

- Get an initial understanding of the data
- Calculate basic summarization statistics
- Visualize the data
- Identify data problems such as outliers, missing values, duplicate records

### 2.4.2 Preprocessing and Transformation

- Transform data into a representation that is suitable for the chosen data mining methods
  - Scales of attributes (nominal, ordinal, numeric)
  - Number of dimensions (represent relevant information using less attributes)
  - Amount of data (determines hardware requirements)
- Methods
  - Discretization and binarization
  - Feature subset selection / dimensionality reduction
  - Attribute transformation / text to term vector / embeddings

- Aggregation, sampling
- Integrate data from multiple sources

💡 Data integration and prep. Is estimated to take 70-80% of the time & effort of a DM project

### **2.4.3 Data Mining**

Input is the preprocessed data and output is a model and/or patterns.

### **2.4.4 Deployment**

- Use model in the business context
- Keep iterating to maintain and improve model



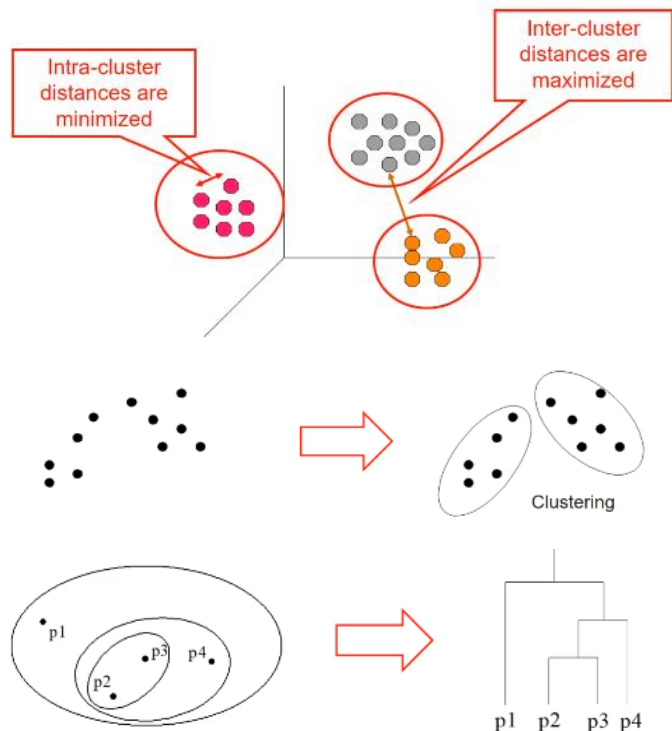
## 3 Cluster Analysis

### 3.1 Overview

Cluster analysis aims to find the objects in a group that are similar to one another and different from the objects in other groups.

There are two general types to be distinguished:

- **Partitional clustering**, where data objects are divided into non-overlapping subsets (clusters) such that each data object is in exactly one subset
- **Hierarchical clustering**, where a set of nested clusters are organized as a hierarchical tree



When performing cluster analysis there are three aspects to consider:

- The clustering algorithm
- A proximity/similarity/dissimilarity measure
- Clustering quality

Supervised learning aims to discover patterns in the data that relate data attributes with a target (class) attribute. These patterns are then utilized to predict the values of the target attribute in unseen data instances.

- Set of classes is known before
- Training data is often provided by human annotators

**CA is unsupervised learning.** It aims to explore the data and to find intrinsic patterns in it.

- Set of classes is not known before
- No training data is used

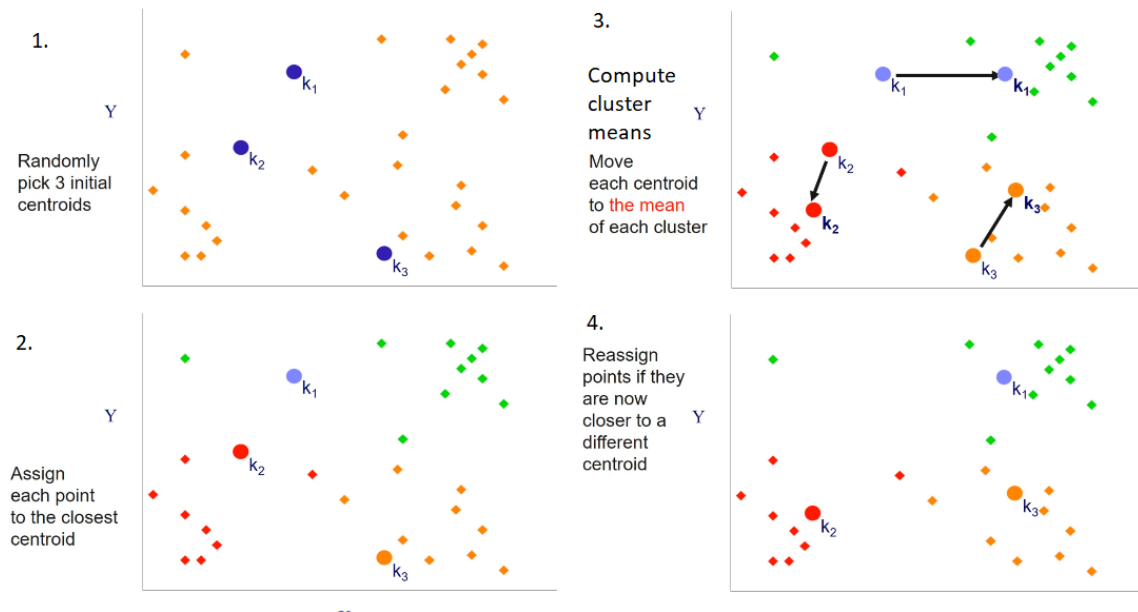
### 3.2 K-Means Clustering

- Is a partitional clustering algorithm
- Number of clusters  $k$  must be specified manually
- Each cluster is associated with a randomly selected centroid (center point, is not an actual data point)
- Each point is assigned to the cluster with closest centroid

**Algorithm:**

1. select  $K$  points randomly as the initial centroids
2. repeat:

3. form  $K$  clusters by assigning all points to closest centroid
4. Recompute the centroid of each cluster (by comp. cluster mean)
5. **until:** the centroids don't change (**convergence criteria**)



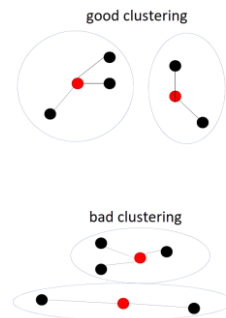
### Convergence criteria:

- No or minimum change of centroids
- No or minimum re-assignments of data points to different clusters
- Stop after  $x$  iterations
- Minimum decrease in the sum of squared error (SSE)

### 3.2.1 Sum of squared error (SSE)

- For each point  $x$  in the respective cluster  $C$  the error is the distance to the nearest centroid  $m$
- Number of clusters is  $k$
- To get SSE, square (puts more weight on longer distances) and sum the error of each point

$$SSE = \sum_{j=1}^k \sum_{x \in C_j} dist(x, m_j)^2$$



### 3.2.2 Weaknesses of K-Means & their solutions

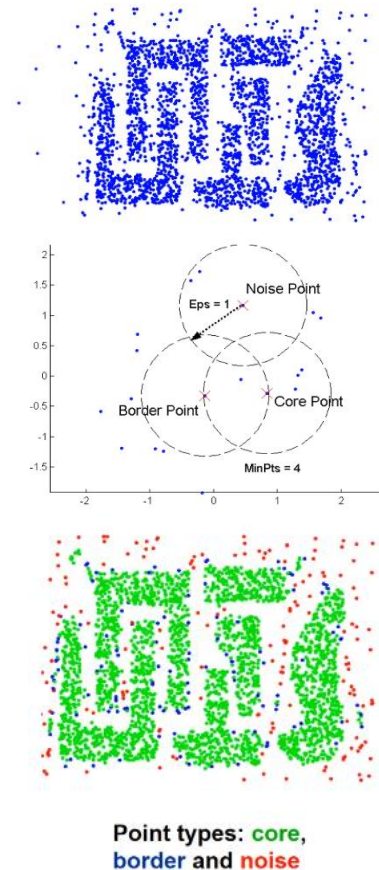
- Clustering results may vary depending on initial choice of seeds (number & position)
  - Restart several times with different random seeds and choose clustering with smallest SSE
  - Run k-means with different values of  $k$ 
    - SSE for different values of  $k$  can't be compared directly !
    - Choose  $k$  where SSE improvement decreases or employ X-Means
- Outliers lead to undesirable clusters
  - Use K-Medoids, a variation that uses median instead of mean
  - Use DBSCAN, a density-based clustering method that removes outliers

- Globular shapes are hard to correctly detect
  - Use density-based clustering

### 3.3 Density-based Clustering

#### 3.3.1 DBSCAN

- Density refers to the number of points within a specified radius  $\epsilon$  (**Eps**/Epsilon)
- It divides data points into three classes
  - **Core Point**: has at least a specified number of neighboring points (**MinPts**) within the radius Eps.
    - The point itself is counted as well
    - These points form the interior of a dense region (cluster)
  - **Border Point**: has fewer points than **MinPts** within Eps, but is in the neighborhood of a core point
  - **Noise Point**: any point that is not a core or border point



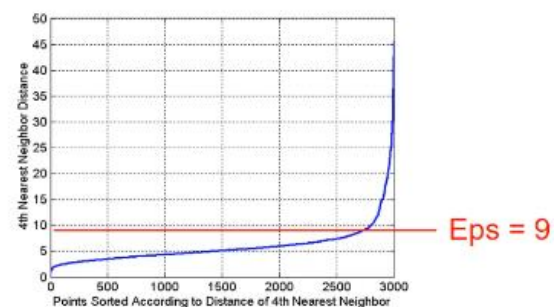
#### Algorithm:

1. label all points as core, border, or noise points
2. eliminate all noise points
3. put an edge between all core points that are within Eps of each other
4. make each group of connected points into a separate cluster
5. assign each border point to one of the clusters of its associated core points
  - a border point can be at the border of multiple clusters
  - use voting if nearby core points belong to different clusters
  - if equal vote, assign border point randomly

#### Determining suitable Eps and MinPts values:

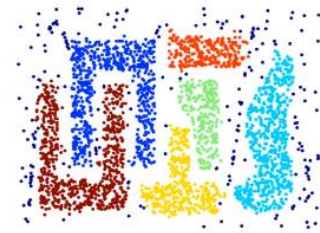
For points in a cluster, their  $k^{\text{th}}$  nearest neighbor (single point) should be at roughly the same distance. Noise points have their  $k^{\text{th}}$  nearest neighbor at farther distances.

1. Start with setting **MinPts**=4 (rule of thumb)
2. Plot sorted distance of every point to its  $k^{\text{th}}$  nearest neighbor
3. Set Eps to the sharp increase of the distances (start of noise points)
4. Decrease  $k$  if small clusters are labeled as noise (subjective decision)



5. Increase  $k$  if outliers are included into the clusters (subjective decision)

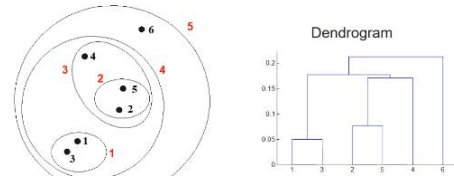
! DBSCAN has problems with datasets of varying densities as Epsilon is defined globally.



Clusters

### 3.4 Hierarchical Clustering

- Produces a set of nested clusters organized as a hierarchical tree
- Can be visualized as a dendrograms; a tree-like diagram that records the sequences of merges or splits. The y-axis displays the former distance between merged clusters



#### Advantages:

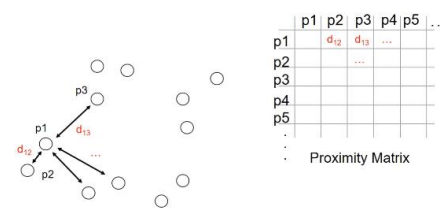
- $k$  does not need to be specified beforehand; after the hierarchical clustering is done the desired amount can be obtained by “cutting” the dendrogram at the proper level
- may be used to discover meaningful taxonomies (of biological species or customer groups)

Two main types of hierarchical clustering:

- Agglomerative
  - Starts with the points as individual clusters
  - At each step, merge the closest pair of clusters until only one cluster (or  $k$ ) is left
- Divisive
  - Start with one, all-inclusive cluster
  - At each step, split a cluster until each cluster contains a single point (or there are  $k$  clusters)

#### 3.4.1 Agglomerative Clustering Algorithm

1. compute the proximity matrix
2. make each data point be a cluster
3. repeat:
  4. Merge the closest two clusters
  5. Update the proximity matrix
6. until: only a single cluster remains



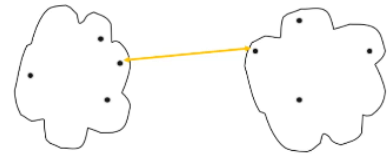
- the key operation is the computation of the proximity of two clusters (step 4)
- different approaches to defining the distance between clusters distinguishes the different algorithms

#### 3.4.2 Defining Inter-Cluster Similarity

Approaches:

### 1. Single Link

Similarity of two clusters is based on the *two most similar/closest points* in the different clusters. Is determined by *one pair of points*, i.e., one link in the proximity graph.

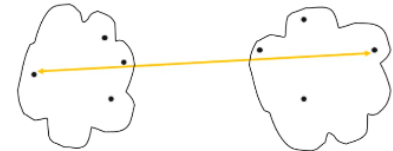


Strength: Can handle non-elliptic shapes

Limitation: Sensitive to noise and outliers

### 2. Complete Link

Similarity of two clusters is based on the *two least similar/most distant points* in the different clusters. Is determined by all pairs of points in the two clusters.



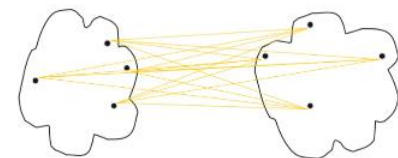
Strength: Less sensitive to noise and outliers

Limitation: Biased towards globular clusters

Limitation: Tends to break large clusters, as decisions cannot be undone

### 3. Group Average

Proximity of two clusters is the average of pair-wise proximity between all points in the two clusters. Compromises between single and complete link.



$$proximity(Cluster_i, Cluster_j) = \frac{\sum_{p_i \in Cluster_i} \sum_{p_j \in Cluster_j} proximity(p_i, p_j)}{|Cluster_i| * |Cluster_j|}$$

Strength: Less sensitive to noise and outliers than single link

Limitation: Biased towards globular clusters

### 4. Distance between Centroids

See: 3.2 K-Means Clustering

## 3.4.3 Problems and Limitations

Different schemes have problems with one or more of the following:

- Sensitivity to noise and outliers
- Difficulty handling non-elliptic shapes
- Breaking large clusters

High space and time complexity:

- $O(N^2)$  or more space since it uses the proximity matrix ( $N$  is the number of points)
- Workaround: apply hierarchical clustering to a random sample of the original data

## 3.5 Proximity Measures

Different proximity measures are used depending on the requirements of the application.

- Similarity
  - Numerical measure of *how alike* two data objects are
  - Often falls in the range of [0, 1]
- Dissimilarity/Distance
  - Numerical measure of *how different* two data objects are
  - Minimum dissimilarity is often 0, upper limit varies

We distinguish proximity measures for single attributes and measures for multidimensional data points (records).

### 3.5.1 Proximity of single attributes

Attribute Type	Dissimilarity	Similarity
Nominal	$d = \begin{cases} 0 & \text{if } p = q \\ 1 & \text{if } p \neq q \end{cases}$	$d = \begin{cases} 1 & \text{if } p = q \\ 0 & \text{if } p \neq q \end{cases}$
Ordinal	$d = \frac{ p - q }{n - 1}$ $p$ and $q$ are translated to nominal values $n$ is the number of values	$s = 1 - \frac{ p - q }{n - 1}$
Interval or ration	$d =  p - q $	$s = -\frac{ p - q }{1 + d}$ $s = \frac{1}{1 + d}$ $s = 1 - \frac{d - \min\_d}{\max\_d - \min\_d}$

$p$  and  $q$  are attribute values for two data objects.

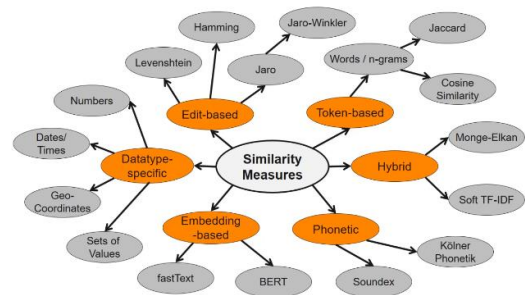
#### Levenshtein Distance

Measures the dissimilarity of two strings.  
Measures the minimum number of edits needed to transform one string into the other. Allowed edit operations:

- Insert a character
- Delete a character
- Replace a character

Examples:

- $\text{levenshtein}(\text{'table'}, \text{'cable'}) = 1$  (1 substitution)
- $\text{levenshtein}(\text{'Bizer, Chris'}, \text{'Chris Bizer'}) = 11$  (10 substitution, 1 deletion)



### 3.5.2 Proximity of multidimensional data points

#### Euclidian Distance

Only works for numeric data: 
$$\text{dist} = \sqrt{\sum_{k=1}^n (p_k - q_k)^2}$$

$n$  is the number of dimensions/attributes and  $p_k$  or  $q_k$  are the attributes of data points  $p$  and  $q$

#### Normalization

Attributes should be normalized so that all attributes can have equal impact on the computation of distances. For example,  $x_1 = (0.1, 20)$ ,  $x_2 = (0.9, 720)$  have Euclidian distance



of 700 because the second attribute dominated the distance. Therefore, attributes should all have a common range, for instance [0, 1].

### Similarity of binary attributes

A binary attribute is *symmetric* if both of its states (0 and 1) have equal importance and carry the same weights (e.g., male and female). → Similarity measure: simple matching coefficient

$$SMC = \frac{\text{number of matching attributes}}{\text{number of attributes}} \quad SMC(x_i, x_j) = \frac{M_{11} + M_{00}}{M_{01} + M_{10} + M_{11} + M_{00}}$$

A binary attribute is *asymmetric* if one of the states is more important than the other.

- By convention, state 1 represents the more important state
- 1 is typically the rare or infrequent state
- Similarity measure: **Jaccard coefficient**

$$J(x_i, x_j) = \frac{M_{11}}{M_{01} + M_{10} + M_{11}}$$

<p><math>p = 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0</math>  <math>q = 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1</math></p> <p> <math>M_{11} = 0</math> (the number of attributes where p was 1 and q was 1)  <math>M_{00} = 7</math> (the number of attributes where p was 0 and q was 0)  <math>M_{01} = 2</math> (the number of attributes where p was 0 and q was 1)  <math>M_{10} = 1</math> (the number of attributes where p was 1 and q was 0) </p> <p> <math>SMC = (M_{11} + M_{00}) / (M_{01} + M_{10} + M_{11} + M_{00}) = (0+7) / (2+1+0+7) = 0.7</math>  <math>J = (M_{11}) / (M_{01} + M_{10} + M_{11}) = 0 / (2 + 1 + 0) = 0</math> </p>	<p>Interpretation of the example:  Customer p bought item 1  Customer q bought item 7 and 10</p>
---	--

When not all attributes should be treated equally the use of a weighted distance measure is useful. For example, the Weighted Euclidian Distance:

$$dist(q, p) = \sqrt{\sum_{k=1}^n w_k * (p_k - q_k)^2}$$

## 4 Classification

### 4.1 Overview

Previously unseen records should be assigned a class from a given set of classes as accurately as possible. Given a collection of records (*training set*) where each record contains a set of attributes and one of the attributes is the *class attribute (label)* that should be predicted, a *model* should be trained to learn the class attributes as a function of the values of other attributes.

Variants are binary classification (e.g., fraud detection or true/false), multi-class classification (e.g., low-medium-high) and multi-label classification (more than one class per record, e.g., user interests).

### 4.2 K-Nearest-Neighbors

Example Problem:

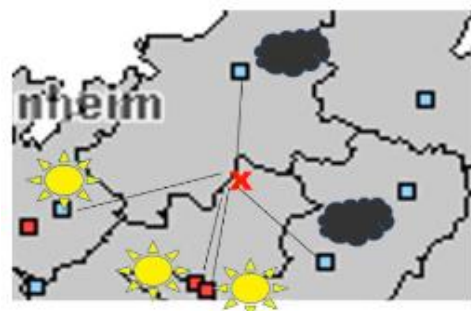
- Predict the current weather in a certain place where there is no weather station.

Basic Idea:

- Use the average forecast of the nearest stations

Approach:

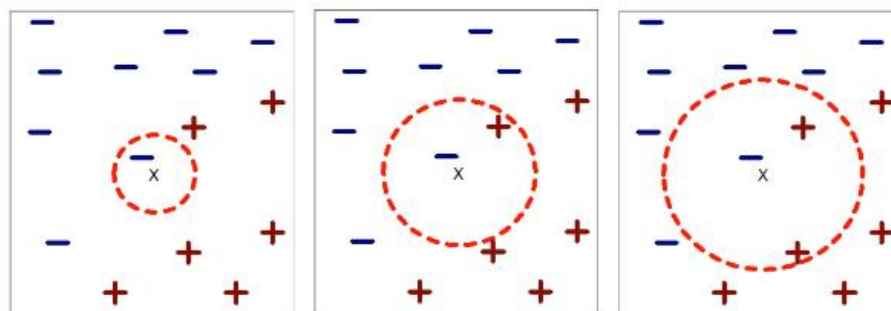
- K-Nearest-Neighbors, where  $k$  is the number of neighbors to consider.



Algorithm:

```
1. Compute distance to each training record
2. Identify k-nearest neighbors
3. Use the class labels of nearest neighbors to determine class label of
unknown record
```

The  $k$ -nearest neighbors of a record  $x$  are data points that have the  $k$  smallest distances to  $x$



(a) 1-nearest neighbor

(b) 2-nearest neighbor

(c) 3-nearest neighbor

If  $k$  is too small, single noise points can influence classification. If  $k$  is too large, more remote points may have a too strong influence. Rule of thumb: test  $k$  values between 1 and 20.



- Often very accurate but slow because all unseen records need to be compared to all training examples
- Results depend on choosing a good proximity measure
- KNN can handle decision boundaries which are not parallel to the axes (unlike decision trees)

## 4.2.1 Learning types

Learning methods are divided into two categories:

- Lazy learning
  - Single goal: Classify unseen records as accurately as possible
  - Instance-based learning approaches, like KNN. are also called lazy learning as *no explicit knowledge (model) is learned*
- Eager learning
  - Goals: classify unseen records; understand the application domain as a human
  - Eager learning approaches generate models that are (might be) interpretable by humans
  - Examples of eager techniques: decision tree learning, rule learning

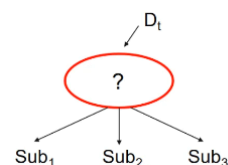
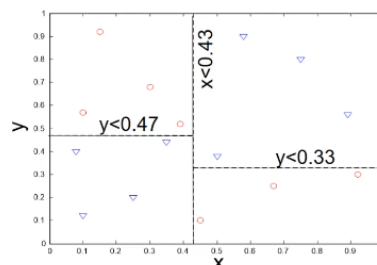
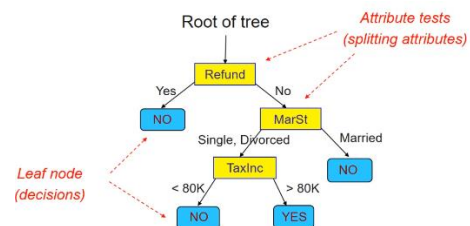
## 4.3 Decision Trees

Encodes a decision into a sequence of single-attribute splits/tests.

Decision boundaries are parallel to the axes because the test condition involves a single attribute at-a-time.

Finding an optimal decision tree is NP-hard. Tree building algorithm thus use a greedy, top-down, recursive partitioning strategy to induce a reasonable solution. Proposed algorithms:

- Hunt's Algorithm
- ID3
- C4.5
- CHAID



**Hunt's Algorithm:**

1. Let  $D_t$  be the set of training records that reach a node  $t$
2. Generate leaf node or attribute test:
  3. If  $D_t$  only contains records that belong to the *same class*  $y_t$ , then  $t$  is a *leaf node* labeled as  $y_t$
  4. If  $D_t$  contains records that belong to *more than one class*, use an attribute test to split the data into subsets having a higher *purity*.
5. Recursively apply this procedure to each subset

## Design Issues

How should the training records be split?

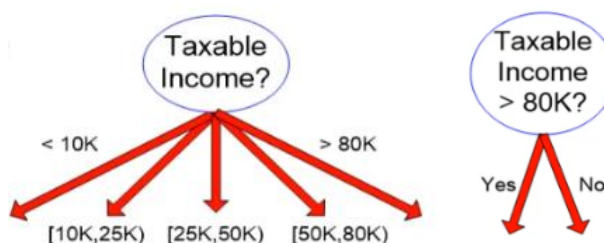
- How to specify the attribute test conditions?
  - Depends on number of ways to split: 2-way or multi-way
  - Depends on attribute data type: nominal, ordinal, continuous
- How to determine the best split
  - Different purity measures

When should the splitting procedure stop?

- Shallow trees might generalize better to unseen records
- Fully grown trees might overfit training data

Splitting based on continuous attributes:

- **Discretization** to form an ordinal categorical attribute
  - Equal interval binning (bin width 10)
  - Equal-frequency binning (bin density 3)
- Binary decision ( $x < y$  or  $x \geq y$ )



## Finding the best split

- Greedy approach: Test all possible splits and use the one that results in the most homogenous/pure nodes
- Need a measure of impurity
  - GINI index
  - Entropy



Measure impurity *before* and *after* splitting and calculate the purity *gain* from the subtraction.

### 4.3.1 GINI index

Calculate the GINI index for a given node  $t$ :

$$GINI(t) = 1 - \sum_j [p(j | t)]^2 \quad \text{where } p(j | t) \text{ is the relative frequency of class } j \text{ at node } t$$

- Minimum (0) when all records belong to one class
- Maximum ( $1 - \frac{1}{n_c}$ ) when records are equally distributed among all classes,  $n_c$  being the number of classes

$$GINI(t) = 1 - p(C_1 | t)^2 - p(C_2 | t)^2$$

$$= 1 - \left( \left( \frac{1}{6} \right)^2 + \left( \frac{5}{6} \right)^2 \right) = 0.278$$

C1	0
C2	6
Gini=0.000	

C1	1
C2	5
Gini=0.278	

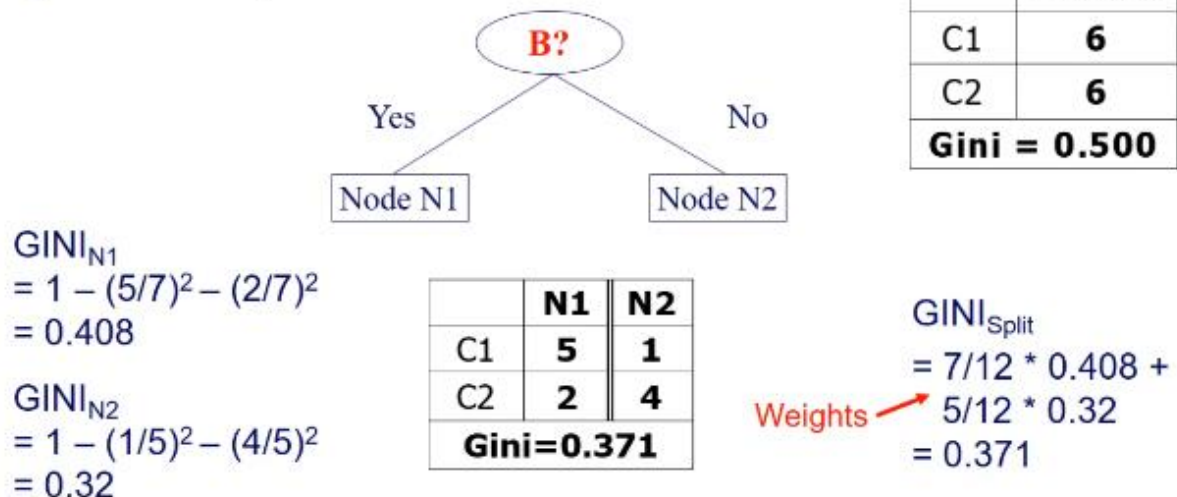
## Splitting into partitions

When a node  $p$  is split into partitions/subsets the GINI index of each partition is weighted according to the partitions size:

$$GINI_{split} = \sum_{i=1}^k \frac{n_i}{n} * GINI(i)$$

where  $n_i$  is the number of records at child  $i$  and  $n$  is the number of records at node  $p$ .

## Split into two partitions



**Purity Gain** =  $0.5 - 0.371 = 0.129$

## GINI index for continuous attributes

Finding the best binary split (e.g.:  $income \geq 80k$ ) for a continuous attribute ( $income [€]$ ):

1. Take all values (income) and sort them by their value
2. Place boundaries/split positions in the middle of each adjacent value
3. Choose split position that has the smallest GINI index

		Taxable Income																					
Sorted Values →		60		70		75		85		90		95		100		120		125		220			
Split Positions →		55		65		72		80		87		92		97		110		122		172		230	
		<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>
Yes		0	3	0	3	0	3	0	3	1	2	2	1	3	0	3	0	3	0	3	0	3	0
No		0	7	1	6	2	5	3	4	3	4	3	4	3	4	4	3	5	2	6	1	7	0
Gini		0.420		0.400		0.375		0.343		0.417		0.400		<u>0.300</u>		0.343		0.375		0.400		0.420	

### 4.3.2 Alternative impurity measure: information gain (entropy)

Information gain relies on the *entropy* of each node:

$$Entropy(t) = - \sum_j p(j | t) * \log_2 p(j | t)$$

$p(j | t)$  is the relative frequency of class  $j$  at node  $t$ .

Entropy measures homogeneity of a node:

- Minimum (0.0) when all records belong to one class
- Maximum ( $\log_2 n_c$ ) when records are equally distributed among all classes

### Splitting into partitions

Entropy values need to be weighted by the size of the emerging partitions. Disadvantage is that it tends to prefer splits that result in large number of partitions, each being small but pure.

GainRATIO is designed to overcome this tendency. It adjusts the information gain by the entropy of the partitioning, so a large number of small partitions is penalized.

### 4.3.3 Overfitting

Learned models can fit the training data too closely and thus work poorly on unseen data. An overfitted model does not generalize well to unseen data.

#### Symptoms

- Model works well on training data but performs poorly on test data
- Decision tree is too deep
- Too many branches

#### Causes

- Too little training data
- Noise/outliers in training data
- High model complexity

💡 If training data is *under-representative*, training errors decrease but testing errors increase. Increasing the size of the training set reduces the difference between training and testing errors.

### Preventing Overfitting

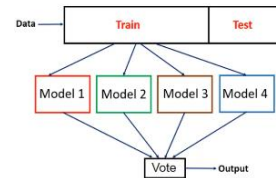
Pre-Pruning:

- Stop the algorithm before tree becomes fully grown (shallow tree generalizes better)
- Early stopping conditions
  - Leaf nodes less than threshold (e.g., leaf\_size=4)
  - Stop expanding node if impurity measure only slightly improves (gain<0.01)

Post-Pruning:

1. Grow decision tree to its entirety
2. Trim the nodes of the decision tree in a bottom-up fashion

- a. Estimate generalization error before and after trimming
  - b. Using a validation set
3. If gen. error improves after trimming:
  - a. Replace sub-tree by a leaf node or
  - b. Replace sub-tree by most frequently used branch

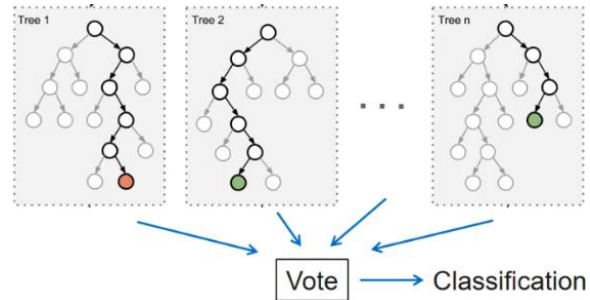


Ensembles:

- Learn different models and have them vote on the final classification decision

### 4.3.4 Random Forest

- Ensemble consisting of many different decision trees (50 to 200)
- Independence of trees achieved by introducing randomness into the learning process
  - Only use a random subset of the attributes at each split
  - Learn on different random subsets of the data (bagging)
- Random forest usually outperforms decision trees



### 4.3.5 Conclusion

#### Advantages

- Inexpensive to construct
- Extremely fast at classifying unknown records
- Easy to interpret by humans for small-sized trees
- Can easily handle redundant or irrelevant attributes
- Accuracy is comparable to other classification techniques for many low dimensional data sets

#### Disadvantages

- Space of possible decision trees is exponentially large. Greedy approaches are often unable to find the best tree
- Trees do not consider interactions between attributes

## 4.4 Model Evaluation

*How good is a model at classifying unseen records?*

### 4.4.1 Metrics for evaluation

#### Confusion Matrix

Counts the correct and false classifications as basis for calculating different performance metrics.

$$\text{accuracy} = \frac{\text{correct predictions}}{\text{all predictions}} = \frac{TP+TN}{TP+TN+FP+FN}$$

Confusion Matrix

	PREDICTED CLASS	
	Class=Yes	Class=No
	Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	Class=No
	Class=No	Class=No

True Positives  
False Negatives  
False Positives  
True Negatives

$$\text{errorRate} = 1 - \text{accuracy}$$

### Class imbalance problem

Sometimes classes have very unequal frequency, e.g., in fraud detection (98% ok vs. 2% fraud). The class of interest is commonly called the *positive class* and the rest *negative classes*.

### Precision $p$

The number of *correctly classified positive* examples divided by the number of examples that are classified as positive.

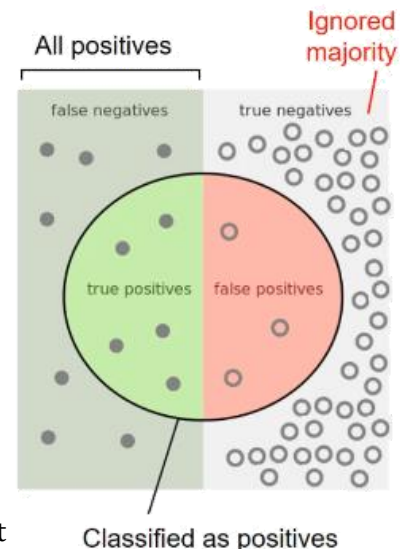
$$p = \frac{TP}{TP + FP}$$

### Recall $r$

The number of *correctly classified positive* examples divided by the total number of actual positive examples in the test set.

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

True Negatives get ignored as they are the majority that dilute the accuracy. These metrics (both equally important) should always be applied if there is a class imbalance problem.



### F<sub>1</sub>-Score

Combines *precision* and *recall* into one measure; it is the harmonic mean of both. It tends to be closer to the smaller of the two, so both must be large for a large F<sub>1</sub>-Score.

$$F_1 = \frac{2pr}{p + r} = \frac{2TP}{2TP + FP + FN}$$

All measures above basically just count through the confusion matrix. This implies that every case has the same weight. So, errors, like false positives and false negatives are regarded as equally bad as well as the correct cases (true positives/negatives).

### Cost-Sensitive model evaluation

For example, in fraud detection, a *misclassified fraudulent transaction* (FN) is worse than a *misclassified honest/integer transaction* (FP).

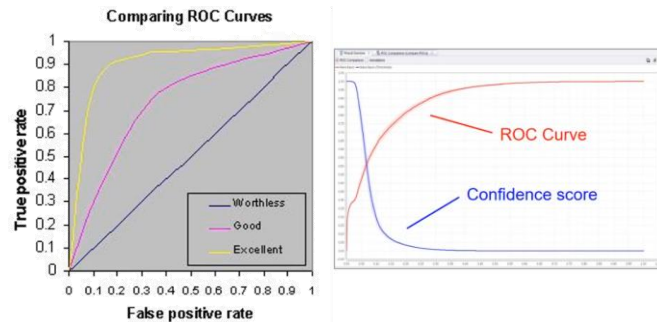
Therefore, each case in the confusion matrix gets an assigned cost  $c(i | j)$  for misclassifying a class  $j$  as class  $i$ .

	PREDICTED CLASS		
	C(i j)	Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	C(Yes Yes)	C(No Yes)
	Class=No	C(Yes No)	C(No No)



## ROC Curves

Graphical approach for displaying trade-offs between detection rate and false alarm rate. ROC curves visualize the *TP-rate* and *FP-rate* in relation to the algorithms confidence.



Interpreting ROC Curves:

"The steeper, the better". Random guessing results in diagonal, so a decent classification model should result in a curve above the diagonal.

## 4.4.2 Methods for evaluation

- How to obtain a reliable estimate of the *generalization performance*?

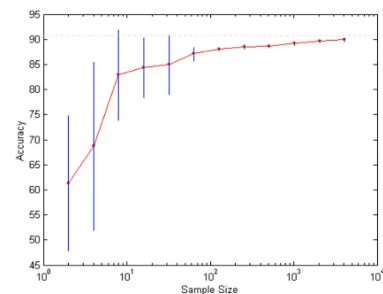
The general approach would be to split labeled records into a training set and a test set. Alternative splitting approaches:

- Holdout method
- Random subsampling
- Cross-Validation

## Learning Curve

💡 Shows how accuracy changes with growing training set size.

If model performance is low and unstable, get more training data. Use as much labeled data for training as possible rather than testing. Labeling additional data is often expensive due to manual effort involved.



## Holdout Method

Reserves a certain amount of labeled data for testing and uses the remainder for training.

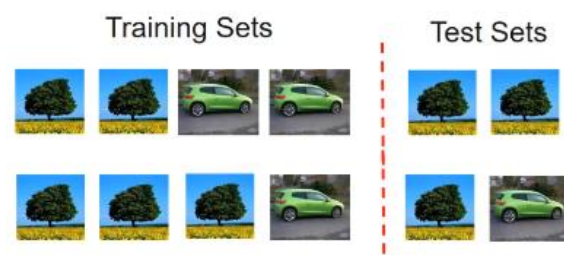
❗ For imbalanced datasets, random samples might not be representative. So, sampling each class independently guarantees the presence of the minority class (*Stratified sampling*).

– Usually: 1/3 for testing, 2/3 for training (or even better 20% / 80%)



## Random Subsampling

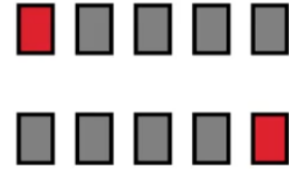
Holdout estimate can be made more reliable by repeating the process with different subsamples. In each iteration a certain proportion is randomly selected for training. The performance of the different iterations is averaged.



! Still not optimal as the different test sets may overlap. Some outliers might always end up in the test sets. Important records for learning (e.g., red tree) might always be in test sets.

### (K-Fold) Cross-Validation

CV avoids overlapping test sets. The data is split into  $k$  subsets/folds of equal size. Per iteration one subset is used for testing and the remainder for training so every record is used *exactly once* for testing. The performance estimates of all runs are averaged to yield the overall performance estimate. Often the subsets are generated using stratified sampling to deal with class imbalance.



## 4.5 Rule Learning

Classify records by using a collection of “if x then y” rules:  $Condition \rightarrow y$

- $Condition$  is a conjunction of attribute tests (rule antecedent)
- $y$  is the class label (rule consequent)

### 4.5.1 Applying a rule-based classifier

A rule  $r$  covers a record  $x$  if the attributes of the record satisfy the condition of the rule.

Example:

Rule 1 *covers* hawk (bird) & Rule 3 covers grizzly bear (mammal).

**R1:** (Give Birth = no)  $\wedge$  (Can Fly = yes)  $\rightarrow$  Birds  
**R2:** (Give Birth = no)  $\wedge$  (Live in Water = yes)  $\rightarrow$  Fishes  
**R3:** (Give Birth = yes)  $\wedge$  (Blood Type = warm)  $\rightarrow$  Mammals  
**R4:** (Give Birth = no)  $\wedge$  (Can Fly = no)  $\rightarrow$  Reptiles  
**R5:** (Live in Water = sometimes)  $\rightarrow$  Amphibians

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
hawk	warm	no	yes	no	?
grizzly bear	warm	yes	no	no	?

### Coverage and Accuracy

Coverage:

Fraction of all records that satisfy the condition of a rule.

Accuracy:

Fraction of covered records that satisfy the consequent of a rule.

**R1:** (Status=Single)  $\rightarrow$  No  
 Coverage = 40%  
 Accuracy = 50%



## Characteristics of rule-based classifiers

Mutually exclusive rule set:

- Rules in a rule set are mutually exclusive if no two rules are triggered by the same record.
- Ensures that every record is covered by *at most* one rule

Exhaustive rule set:

- Rule sets have exh. coverage if there is a rule for every combination of attribute values
- Ensures that every record is covered by *at least* one rule

R1: (Give Birth = no)  $\wedge$  (Can Fly = yes)  $\rightarrow$  Birds  
 R2: (Give Birth = no)  $\wedge$  (Live in Water = yes)  $\rightarrow$  Fishes  
 R3: (Give Birth = yes)  $\wedge$  (Blood Type = warm)  $\rightarrow$  Mammals  
 R4: (Give Birth = no)  $\wedge$  (Can Fly = no)  $\rightarrow$  Reptiles  
 R5: (Live in Water = sometimes)  $\rightarrow$  Amphibians

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
lemur	warm	yes	no	no	?
turtle	cold	no	no	sometimes	?
dogfish shark	cold	yes	no	yes	?

- A turtle triggers both R4 and R5  $\rightarrow$  not mutually exclusive
- A dogfish shark triggers none of the rules  $\rightarrow$  not exhaustive

## 4.5.2 Learning rule-based classifiers

### Direct Method

Extracts rules directly from data using an algorithm such as RIPPER. For a 2-class problem:

- Choose the less frequent class as positive class and the other as negative class
- Learn rules for the positive class & negative will be default

For a multi-class problem:

- Order the classes according to increasing class prevalence
- Learn the rule set for the smallest class first, treat the rest as negative class
- Repeat with next smallest class as positive class

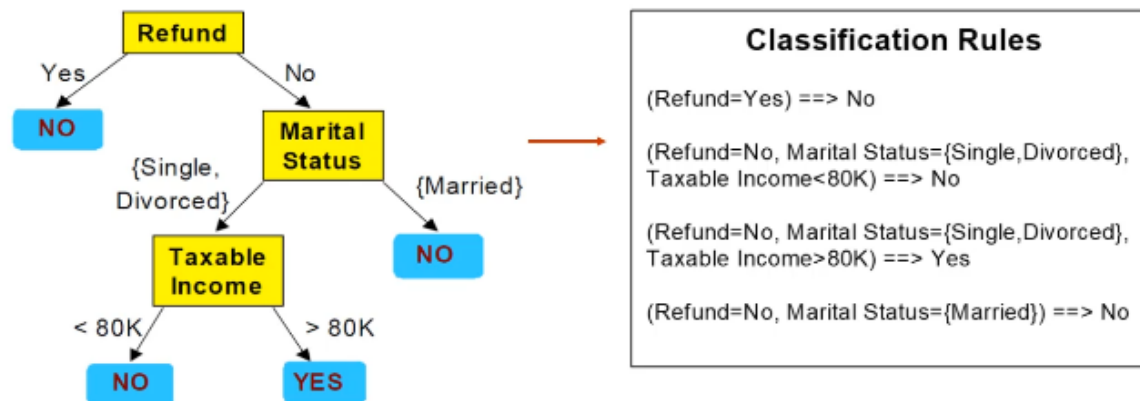
RIPPER uses *sequential covering* to learn a rule list for each class:

- Start from an empty rule list
- Grow a rule that covers as many positive examples as possible and is rather accurate
- Remove training records covered by the rule
- Repeat steps 2 & 3 until stopping criterion is met

### Indirect Method

Extract rules from other classification models (e.g., decision trees) using for example C4.5rules.

Approach: Generate a rule *for every path* from the root to one of the leaf nodes in the decision tree. Therefore, the rule set contains as much information as the tree. The generated rules are mutually exclusive and exhaustive.



Rules can end up being overly complicated. The generated rule:

$$(refund = "no") \wedge (Status = "married") \rightarrow "no"$$

can be simplified to

$$(Status = "married") \rightarrow "no"$$

## 4.6 Naïve Bayes

Probabilistic classification technique based on Bayes theorem. The goal is to *estimate* the *most probable* class label for a given record.

Probabilistic formulation of the classification task:

Consider each attribute and class label as random variables. Given a record with attributes  $(A_1, A_2, \dots, A_n)$ , the goal is to find the class  $C$  that maximizes the conditional probability:

$$P(C | A_1, A_2, \dots, A_n)$$

Example: Should we play golf?

- $P(\text{Play} = \text{yes} | \text{outlook} = \text{rainy}, \text{temperature} = \text{cool})$
- $P(\text{Play} = \text{no} | \text{outlook} = \text{rainy}, \text{temperature} = \text{cool})$
- How to estimate these probabilities given training data?

### 4.6.1 Bayes' theorem

*Prior probability*  $P(C)$ , is the probability of an event  $C$  before evidence  $A$  is seen.  
E.g., we play golf in 70% of all cases > means  $P(C) = 0.7$

*Posterior probability*  $P(C|A)$ , refers to the probability of event  $C$  *after* evidence  $A$  is seen.  
E.g., it is windy and raining > means  $P(C|A) = 0.2$

Theorem:

$$P(C|A) = \frac{P(A|C) * P(C)}{P(A)}$$

Useful in situations where  $P(C|A)$  is unknown while the other factors are known or easy to estimate.

### Applying Bayes Theorem to classification

1. Compute the probability  $P(C|A)$  for all values of  $C$  using Bayes theorem.  $P(A)$  is the same for all cases. Thus, we just need to estimate  $P(C)$  and  $P(A|C)$ .
2. Choose value of  $C$  that maximizes  $P(C|A)$

$$P(C|A) = \frac{P(A|C)P(C)}{P(A)}$$

Evidence = record (points to  $P(A|C)$ )  
Class-conditional probability of evidence (points to  $P(A|C)$ )  
Class (points to  $C$ )  
Prior probability of class (points to  $P(C)$ )  
Prior probability of evidence (points to  $P(A)$ )

Example:

$$P(\text{play} = \text{yes} | \text{outlook} = \text{rainy}, \text{temp} = \text{cool}) = \frac{P(\text{outlook} = \text{rainy}, \text{temp} = \text{cool} | \text{Play} = \text{yes}) * P(\text{play} = \text{yes})}{P(\text{outlook} = \text{rainy}, \text{temp} = \text{cool})}$$

Estimating prior probability  $P(C)$ :

- $P(\text{Play} = \text{no}) = \frac{5}{14}$
- $P(\text{Play} = \text{yes}) = \frac{9}{14}$

Estimating the class-conditional probability  $P(A|C)$ :

- Naïve Bayes assumes that all attributes are statistically independent (knowing the value of one attribute says nothing about the value of another); this independence is almost never correct, but works well in practice
- The independence assumption allows the joint probability  $P(A|C)$  to be reformulated as the product of the individual probabilities  $P(A_i|C_j)$ :

Training Data

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

$$P(\text{outlook} = \text{rainy}, \text{temp} = \text{cool} | \text{Play} = \text{yes}) = P(\text{temp} = \text{cool} | \text{Play} = \text{yes}) * P(\text{outlook} = \text{rainy} | \text{Play} = \text{yes})$$

- The probabilities  $P(A_i|C_j)$  for all  $A_i$  and  $C_j$  can be estimated directly from training data!

### Handling numerical attributes

Option 1: Discretize numerical attributes before learning classifier (37°C="hot", 21°C="mild")

Option 2: Assume that numerical attributes have a normal distribution given the class.

- Use training data to estimate parameters of the distribution (e.g., mean and SD)
- Once the distribution is known, it can be used to estimate the cond. prob.  $P(A_i|C_j)$

The probability density function for the normal distribution is  $f(x) = \frac{1}{\sqrt{2\pi}\sigma} * e^{\frac{(x-\mu)^2}{2\sigma^2}}$  It is defined by two parameters: Sample mean  $\mu = \frac{1}{n} * \sum_{i=1}^n x_i$  and standard deviation  $\sigma = \sqrt{\frac{1}{n-1} * \sum_{i=1}^n (x_i - \mu)^2}$ . Both parameters can be estimated from the training data.

### Handling missing values

Training: Record is not included into frequency count for attribute value-class combination

Classification: Attribute will be omitted from calculation

### Zero-Frequency Problem

What if an attribute value doesn't occur with every class value? E.g., no outlook="overcast" for class "no"?  $P(outcome = "overcast"|"no") = \frac{0}{5} = 0$ . Posterior probability would also be zero.


Remedy: Add 1 to the count for every value-class combination (*Laplace estimator*)

Result: Probability will never be zero.


### Characteristics of NB

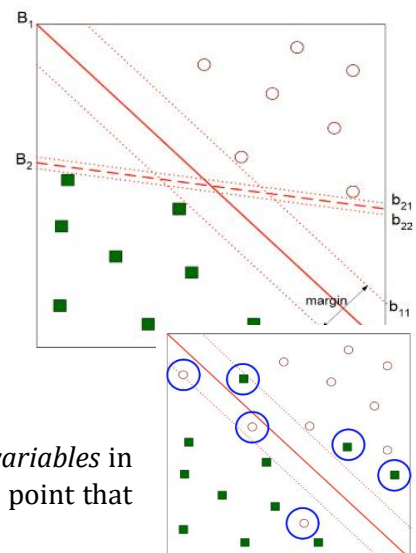
- Robust to isolated noise points, as they will be averaged out
- Robust to irrelevant attributes as  $P(A_i|C)$  is distributed uniformly for  $A_i$
- Adding too many redundant attributes can cause problems
- Learning NB is computationally cheap
- Storing the probabilities does not require a lot memory

## 4.7 Support Vector Machines (SVMs)

 SVMs are algorithms for learning linear classifiers for *two class problems* from examples described by *continuous attributes*. SVMs achieve very good results especially for high dimensional data.

SVMs find a linear *hyperplane* (decision boundary) that will separate the data.

 To avoid overfitting and to generalize for unseen data, SVMs find the hyperplane that maximizes the margin to the closest points (support vectors).



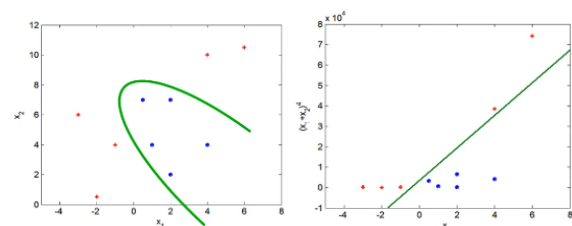
### 4.7.1 Dealing with not linearly separable data

To deal with linearly inseparable data, we introduce *slack variables* in margin computation which result in a penalty for each data point that violates the decision boundary.

### 4.7.2 Dealing with non-linear decision boundaries

To deal with non-linear separation, transform data into a higher *dimensional space* where there is a linear separation.

Different types of *kernel functions* are used for this transformation.

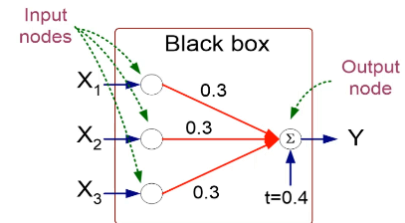


## Characteristics

Performance depends strongly on hyperparameter selection; therefore, running a *hyperparameter optimization* is important.

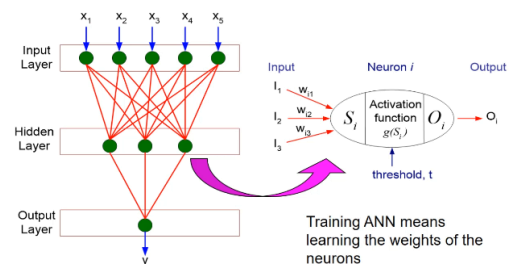
## 4.8 Artificial Neural Networks (ANNs)

ANN Models are assemblies of inter-connected nodes (*neurons*) and weighted links. The output/hidden node/s sums up each of its input values according to the weights of its links.



### 4.8.1 Algorithm for training ANNs

1. Initialize the weights randomly
2. Adjust the weights in such a way that the output of the ANN is as consistent as possible with the class labels of the training example.




### 4.8.2 Deep Neural Networks (DNNs)


Are ANNs but have many hidden Layers therefore are larger. They are very successful in computer vision, speech recognition and natural language processing. They require lots of training data and processing power.

### 4.8.3 Characteristics

- Can be used for classification as well as numerical regression tasks
- ML neural networks are universal approximators
- Very important to choose the right network topology
  - Expressive hypothesis space often leads to overfitting
- Model building is very time consuming, application is fast
- Can handle redundant attributes, difficult to handle missing values

## 4.9 Hyperparameter Selection

 A *hyperparameter* is a parameter which influences the learning process and whose value is set before the learning begins. In contrast, regular parameters are learned from training data.

 To determine good hyperparameters have a machine automatically test many different settings. This is called *hyperparameter optimization*.

### 4.9.1 Hyperparameter Optimization

Goal: Find the combination of hyperparameter values that result in learning the model with the lowest generalization error. How to determine the parameter value combinations to be tested?

- Grid Search: Test all combinations in user-defined ranges
- Random Search: Test combinations of random parameter values

- Evolutionary Search: Keep specific parameter values that worked well

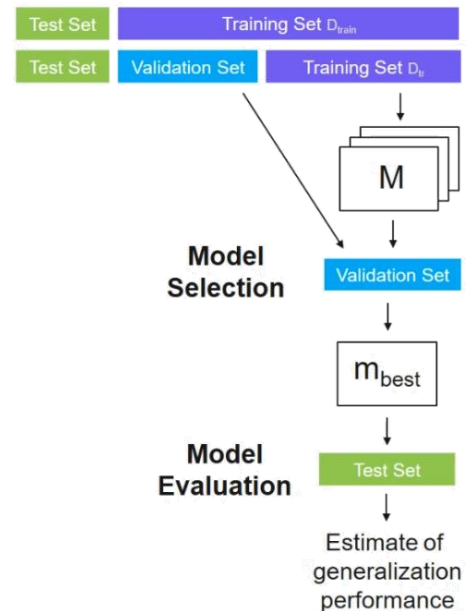
From all learned models  $M$ , select the model  $m_{best}$  that is expected to generalize best to unseen records.

## 4.9.2 Model Selection using a Validation Set

! Keep data used for model selection strictly separate from data used for model evaluation, otherwise  $m_{best}$  will overfit to patterns in test set and the estimate of generalization error will be too optimistic.

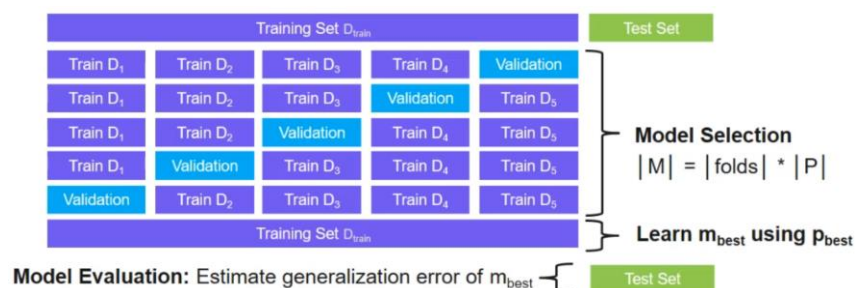
Method to find the best model:

1. Split training set  $D_{train}$  into *validation set*  $D_{val}$  and training set  $D_{tr}$
2. Learn models  $M$  on  $D_{tr}$  using different hyperparameter value combinations  $P$
3. Select best parameter values  $p_{best}$  by evaluating each model  $m_i$  on the *validation set*  $D_{val}$
4. Learn final model  $m_{best}$  on complete  $D_{train}$  using the parameter values  $p_{best}$
5. Evaluate  $m_{best}$  on *test set* to get an unbiased estimate of its generalization performance



## 4.9.3 Model Selection using Cross-Validation

To make sure that all examples are used for validation once and to use as much labeled data as possible for training (see: *Learning Curve*) Cross-Validation can be used.



## 4.9.4 Model Evaluation using Nested Cross-Validation

Nests two cross-validation loops into each other to find the best hyperparameter setting model (model selection) and to get a reliable estimate of the generalization error (model evaluation).

Outer cross-validation estimates error of  $m_{best}$ . The training set is passed on to the inner cross-validation in each iteration.

The inner cross-validation searches for the best parameter combination. It splits the outer training set into inner-training and -validation set and learns model  $m_{best}$  using all outer training data.



## 4.9.5 Feature Selection

Some classification methods automatically select the relevant feature subset as part of the learning process (e.g., decision trees, random forests, ANNs, SVMs). The performance of other methods depends on the subset of the features provided (e.g., KNN, Naïve Bayes).




Approaches for automated feature selection:

- Forward selection: find the best single feature, add further features, test again
  - Backward selection: start using all features, remove some, test again
- Use nested cross-validation to estimate the generalization error.



## 5 Regression

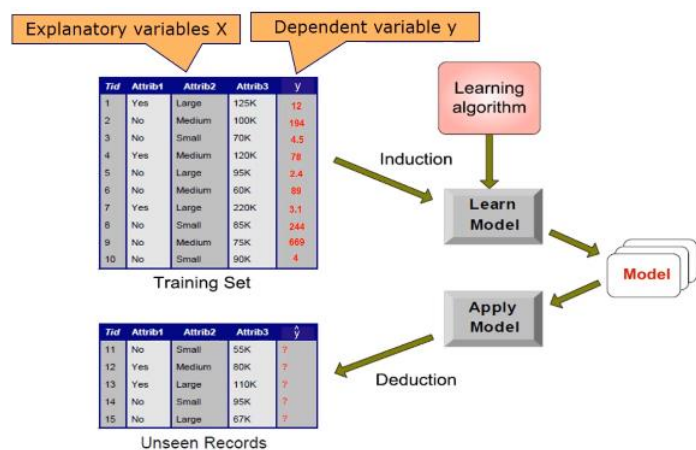
### 5.1 Overview

 Predict the value of a *continuous variable* based on the value of other variables assuming a linear or nonlinear model of dependency. The predicted variable is called *dependent*  $\hat{y}$ . The other variables are called *explanatory variables* or *independent variables* and are denoted  $X = x_1, x_2, \dots, x_n$ . Given training examples  $(X_i, y_i)$ , learn a model  $f$  to predict  $\hat{y}$  from  $X_{unseen}$ .

! Difference to classification is that the predicted attribute  $\hat{y}$  is continuous, while classification is used to predict nominal class attributes.

Application Examples:

- Weather forecasting
- Gasoline consumption
- House Market
- Stock Market



### 5.2 KNN for Regression

? Problem: Predict the temperature in a certain place, where there is no weather station.

 Original Solution: Let the  $k$  nearest stations vote on predicted weather.

💡 Regression Solution: Use the numeric average of the  $k$  nearest Stations.

### 5.3 Model Evaluation

How good is a model at classifying unseen records?

#### 5.3.1 Methods for Model Evaluation

- (Nested) Cross Validation
- Holdout Method

#### 5.3.2 Metrics for Model Evaluation

All metrics are based on the deviation between the real value  $p_i$  and the predicted value  $r_i$ .

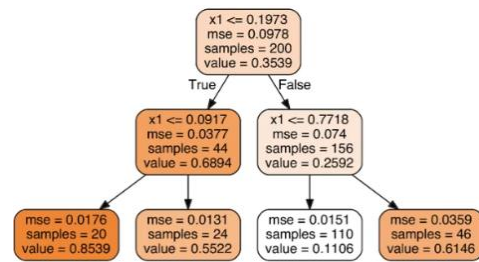
- Mean Absolute Error (MAE)  $MAE = \frac{1}{n} * \sum_{i=1}^n |p_i - r_i|$
- Mean Squared Error (MSE)  $MSE = \frac{1}{n} * \sum_{i=1}^n (p_i - r_i)^2$
- Root Mean Squared Error (RMSE)  $RMSE = \sqrt{\frac{1}{n} * \sum_{i=1}^n (p_i - r_i)^2}$
- Pearsons Correlation Coefficient (PCC)
- R<sup>2</sup>: Coefficient of Determination



## 5.4 Regression Trees

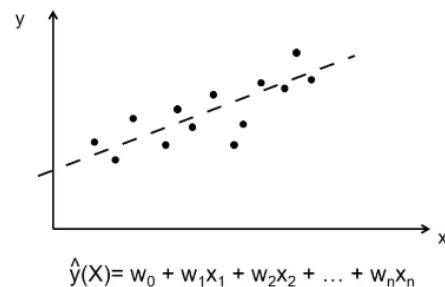
Basic idea of how to learn and apply decision trees can also be used for regression. Differences are that 1) splits are selected by maximizing the MSE reduction (not GINI or entropy) and 2) prediction is average value of the training examples in a specific leaf.

An overfitted tree models specific outliers which is likely to happen when the depth is too high.



## 5.5 Linear Regression

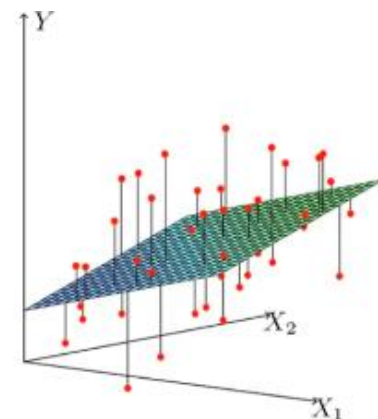
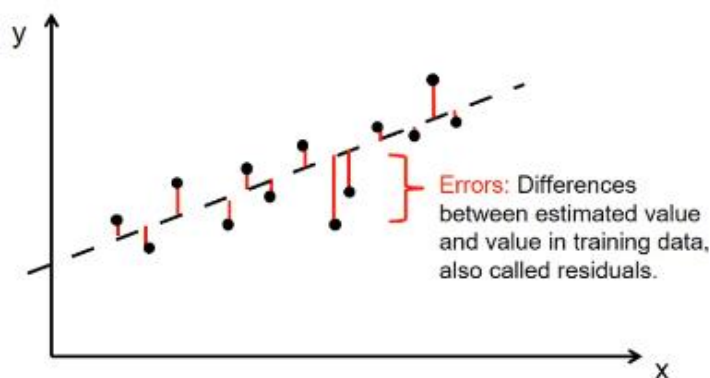
Assumption of Linear Regression: the target variable  $\hat{y}$  is (approximately) linearly dependent on explanatory variables  $X$ . In reality: vector  $X = x_1, x_2, \dots, x_n$  (multiple linear regression).



### 5.5.1 Fitting a regression function

Least-Squares Approach: Find the weight vector  $W = (w_0, w_1, \dots, w_n)$  that minimizes the sum of squared error (SSE) for all training examples.

$$SSE = \sum_{i=1}^n (y_i - \hat{y}(X_i, W))^2$$

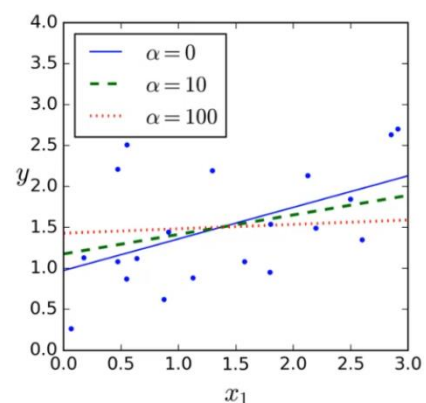


### 5.5.2 Ridge regularization

Variation of least squares approach which tries to avoid overfitting by keeping the weights  $W$  small. Ridge regression cost function to minimize:

$$C(W) = MSE(W) + \alpha * \sum_{i=1}^n w_i^2$$

$\alpha = 0$  is normal least squares regression while  $\alpha = 100$  is a strongly regularized flat curve.



### 5.5.3 Feature Selection

? Are all explanatory values helpful to explain  $\hat{y}$  or is only a subset of the variables useful?

Problem 1: Highly correlated variables > weights are meaningless, and one variable should be removed for the better interpretability of the weights.

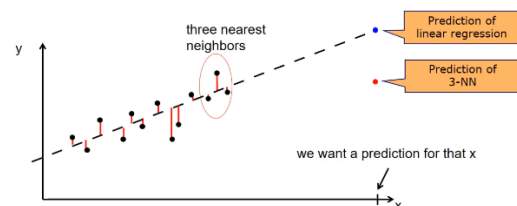
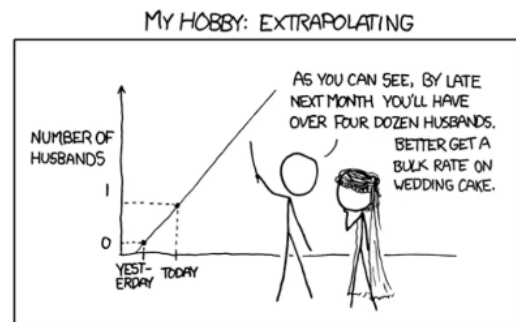
Problem 2: Insignificant variables > uncorrelated variables get  $w = 0$  or other small weights assigned > perform statistical test to prove.

### 5.5.4 Interpolation vs. Extrapolation

Given some training data (e.g., temperature, with values between  $-15^{\circ}\text{C}$  and  $32^{\circ}\text{C}$ ), an *interpolating regression* only predicts values from the min-max-interval of its training examples (here:  $[-15^{\circ}\text{C}, 32^{\circ}\text{C}]$ ), while *extrapolating regression* may also predict values outside of this interval.

Interpolating regression is regarded as “safe” as only reasonable/realistic values are predicted.

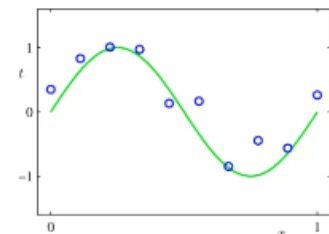
KNN and regression trees only interpolate.



### 5.5.5 Non-linear problems

One possibility to handle non-linear problems, is to apply transformations to the explanatory variables  $X$  within the regression function. E.g., log, exp, square root, square, etc.

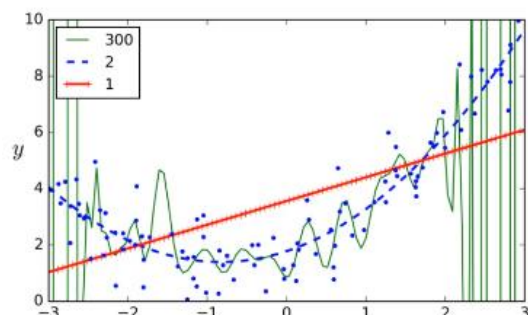
Example:  $\hat{y} = \omega_0 + \omega_1 * x_1^2 + \omega_2 * x_2^2$



## 5.6 Polynomial Regression

Instead of working with the original explanatory variables, we work with different powers of them + an respective weight. This is a widely used extension of linear regression but tends to overfit training data for large degrees  $M$ .

$$\begin{aligned}\hat{y}(x, w) &= \omega_0 + \omega_1 * x^1 + \omega_2 * x^2 \dots + \omega_M * x^M \\ &= \sum_{j=0}^M \omega_j x^j\end{aligned}$$



! Overfitting often happens in sparse regions (regarding data). A workaround is local regression.

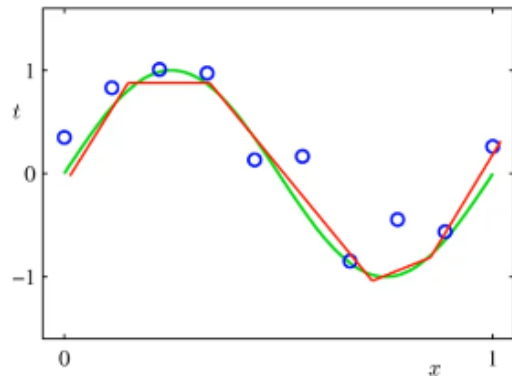
## 5.7 Local Regression

? Assumption: non-linear problems are approximately linear in local areas

💡 Idea: use linear regression locally for the data point at hand (lazy learning)

A combination of KNN and linear regression; given a data point:

1. Retrieve the  $k$  nearest neighbors
2. Learn a regression model using those neighbors
3. Use the learned model to predict  $y$  value



Advantages	Disadvantages
Good local approximation	Slow at runtime
Often better than pure KNN	

## 5.8 ANNs for Regression

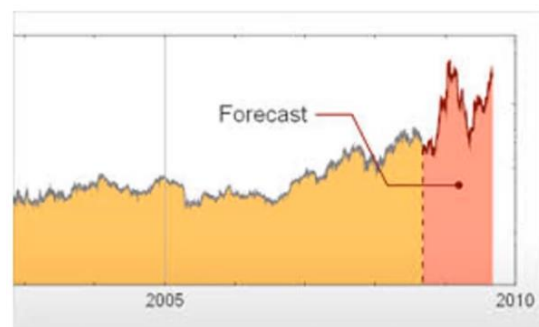
Removing the activation function of the output-layer-neuron and just using the sum allows using ANNs for regression. More complex regression problems can be approximated by using multiple hidden layers; this allows for arbitrary functions.

## 5.9 Time Series Forecasting

📄 A *time series* is a series of data points indexed in *time order* (stock market prices, temperature, etc.). Forecasting: Given a time series predict data points that continue the series into the future. Explicitly deals with time, which is not explicitly considered by other regression techniques. Aims to predict future values of the same variable.

Approaches:

- Data-driven: smoothing
- Model-driven
  - Component models of time series
  - Other regression techniques



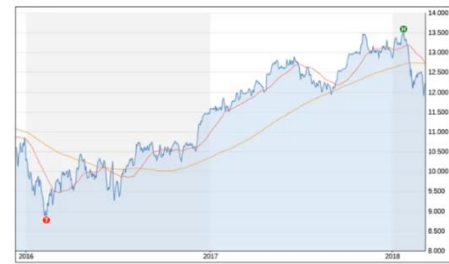
### 5.9.1 Smoothing

Simple Moving Average (SMA): average of the last  $n$  values, as more recent values might matter more.

$$m_{MA}^{(n)}(t) = \frac{1}{n} * \sum_{i=0}^{n-1} x * (t - i)$$

Exponential Moving Average (EMA): exponentially decrease weight of older values.

$$m_{EMA}^{(n)}(t) = \alpha * x(t) + (1 - \alpha) * m_{EMA}^{(n)}(t - 1)$$

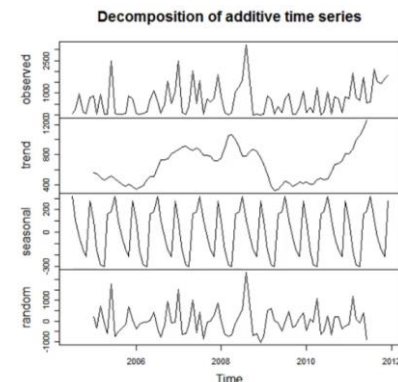


DAX: red = SMA(38 days), yellow = SMA(200 days)

## 5.9.2 Component Models of time series

1. Long-term trend ( $T_t$ )
2. Cyclical effect ( $C_t$ )
3. Seasonal effect ( $S_t$ )
4. Random variation ( $R_t$ )

$$series = T_t + C_t + S_t + R_t$$



## 5.9.3 Windowing

💡 Transforming a forecasting problem into a “classical” learning problem (either classification or regression) by only taking the *last k time periods* into account.

E.g., weather forecasting; using the weather from the three previous days; possible model:

- sunny, sunny, sunny > sunny
- sunny, cloudy, rainy > rainy
- sunny, cloudy, cloudy > rainy

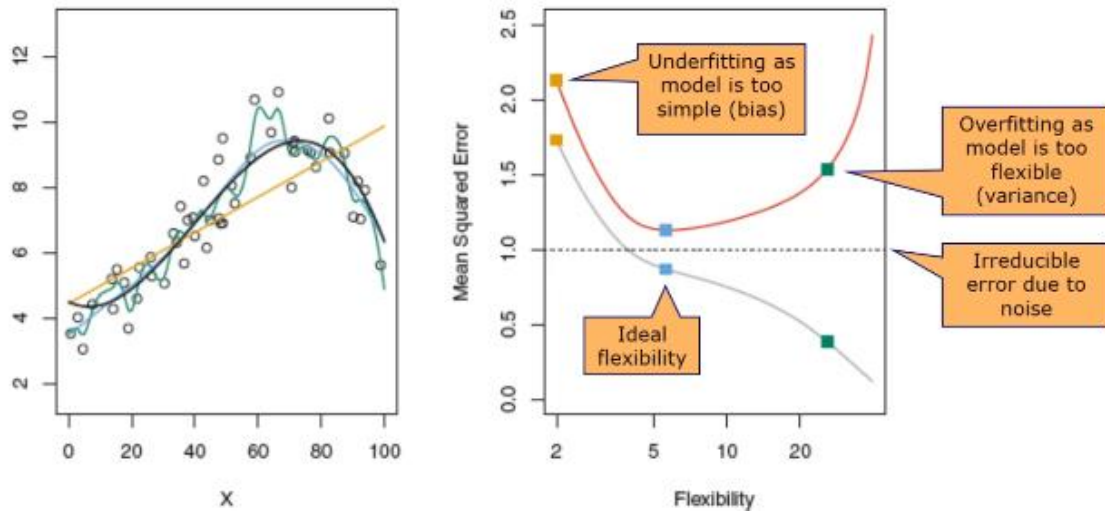
❗ Assumption: only the *last k time periods* matter for the forecast and the older past is irrelevant.

## 5.10 The Bias/Variance-Tradeoff

The goal is to learn regression as well as classification models that generalize well to unseen data. The generalization error of any model can be understood as a sum of three errors:

1. **Bias**; due to wrong model complexity. Simple models used for complex real world phenomena *underfits* the training and test data.
2. **Variance**; due to a model’s excessive sensitivity to small variations in the training data. Models with high degree of freedom/flexibility (like polynomial regression models or deep trees) are likely to *overfit* the training data.
3. **Irreducible error**; due to noisiness of the data itself. To reduce this part of the error the training data needs to be cleansed (by removing outliers, fixing broken sensors).

- Left: Three models with **different flexibility** trying to fit a function
  - **Orange**: Linear regression. **Green, blue**: Polynomials of different degrees
- Right: Training error (**gray**) and test error (**red**) in relation to model flexibility

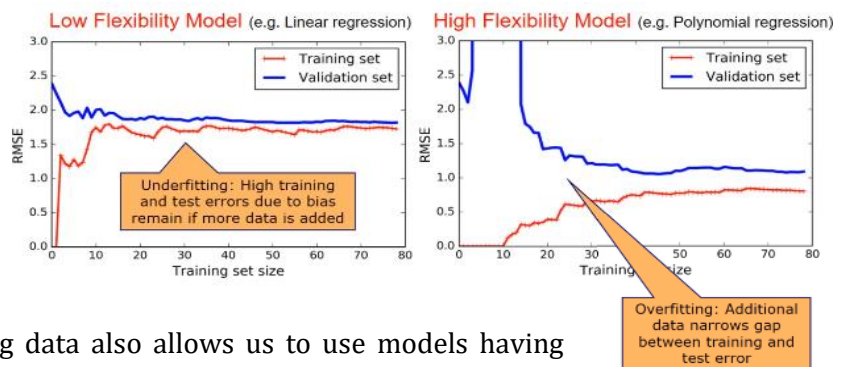


### 5.10.1 Learning Curves

Visualize the training error and test error for different training set sizes.


For overfitting models, the gap between training and test error can often be narrowed by adding more training data.

Thus, having more training data also allows us to use models having higher flexibility, e.g., Deep Learning.



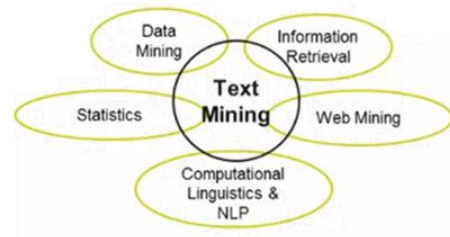
## 6 Text Mining

### 6.1 Overview

 The extraction of implicit, previously unknown, and potentially useful information from a large number of textual resources.

Application examples:

- Classification of news stories or web pages
- Email and news filtering / Spam detection
- Sentiment analysis
- Clustering documents or web pages
- Search term auto-completion
- Information extraction



### 6.2 Text Preprocessing

#### Syntactic Processing (simple)

Text Cleanup (remove punctuation and HTML-tags)

**Tokenization** (break text into single words or n-grams)

#### Linguistic Processing (advanced)


Word sense disambiguation

- Determine which sense a word is having
- Normalize synonyms (United States, US)
- Normalize pronouns

#### Part-of-Speech (POS) Tagging

- Parse sentences according to grammar
- Determine grammatic function of each term, e.g., "John (noun) gave (verb) the (det) ball (noun)."

#### 6.2.1 Stop word removal

 Many of the most frequently used words in English are likely to be useless for text mining; these words are called stop words (e.g., the, of, and, to, is, that, ...).

For an application, an additional domain specific stop word list may be constructed. It reduces data size (20-30%) and might improve effectivity of data mining methods.

#### 6.2.2 Stemming

Techniques to find the *stem of a word*. Improves effectivity of text mining methods – matching of similar words and reduces term vector size.

Basic Rules:

- If a word ends in "es", drop the "s"
- If a word ends with a consonant other than "s", followed by an "s", then delete "s"
- ...

#### 6.2.3 Lemmatization

The process of grouping together the inflected forms of a word so they can be analyzed as a single item, identified by the word's lemma, or dictionary form.





## 6.3 Feature Creation / Generation

### 6.3.1 Vectorization

Transform the text into a vector. This results in a term-document matrix, where each document is represented by a term vector.

Term	Document																			
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
oil	5	12	2	1	1	7	3	3	5	9	5	4	5	4	3	4	5	3	1	85
price	5	6	2	2	0	8	1	2	2	6	5	1	5	2	2	4	0	0	0	63
spec	0	15	0	0	0	8	1	2	2	5	5	2	4	0	0	0	0	0	0	47
info	0	4	0	0	2	4	1	0	0	3	9	0	0	0	0	3	3	0	0	31
market	2	5	0	0	0	3	0	2	0	10	1	2	2	0	0	0	0	0	3	30
land	2	0	1	1	0	4	0	0	1	3	3	0	1	1	0	3	3	0	0	28
bad	0	4	0	0	0	7	0	0	0	2	8	0	0	2	0	0	0	0	0	23
dry	2	0	1	2	2	2	1	0	0	4	2	0	0	0	0	1	1	0	0	23
crude	2	0	2	3	0	2	0	0	0	0	5	2	0	2	0	0	0	0	0	21
invest	0	0	0	0	0	0	0	1	0	5	7	1	4	0	0	0	0	0	0	18
invest	0	0	0	0	0	10	0	1	0	3	3	0	1	0	2	0	0	0	0	17
office	0	0	0	0	0	5	1	1	0	1	4	3	1	0	0	0	0	0	0	17
invest	0	0	0	0	0	1	0	1	0	1	0	1	0	2	0	0	0	0	0	13
act	0	0	0	0	2	0	2	2	2	1	0	0	1	0	0	1	1	0	0	14
invest	0	2	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	14
product	0	0	0	0	0	0	0	0	0	5	1	0	2	0	0	0	0	0	0	12
future	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	12
invest	0	0	0	0	0	3	0	0	0	1	3	1	2	1	1	0	0	0	0	12
invest	0	0	0	0	0	0	5	0	6	0	0	0	0	0	0	0	0	0	0	11
month	0	1	0	0	0	2	2	0	1	0	5	0	0	0	0	0	0	0	0	11
report	0	1	0	0	0	1	8	0	0	0	0	1	0	0	0	0	0	0	0	11
invest	0	0	0	0	0	3	0	0	5	2	0	0	0	0	1	0	0	0	0	11
invest	0	2	0	0	0	1	1	1	0	0	0	0	0	0	0	1	2	0	0	10
product	0	0	0	0	0	4	0	0	1	1	0	0	1	0	0	0	0	0	0	10
invest	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	3	3	0	0	10
world	0	1	0	0	0	1	3	0	1	1	0	0	1	1	0	0	0	0	1	10

T	48	50	34	30	710	710	710	710	710	710	710	710	710	710	710	710	710	710	710	710
Σ	48	50	34	30	710	710	710	710	710	710	710	710	710	710	710	710	710	710	710	710

Term-Document matrix where columns represent documents.

### 6.3.2 Bag-of-Words

In general feature generation, documents are treated as “bags-of-words”. Each word/term becomes a feature, and the order of terms is ignored. Each document is represented by a vector. Different techniques for vector creation:

1. Binary term occurrence: Boolean attributes describe whether a term appears in the document.
2. Term occurrence: Number of occurrences of a term in the document
3. Terms frequency: Attributes represent the frequency in which a term appears in the document (number of occurrences/number of words in document)
4. TF-IDF: see below

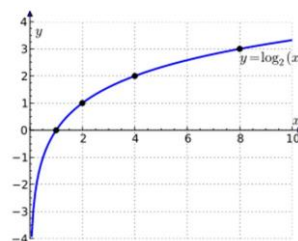
### 6.3.3 TF-IDF term weighting scheme

The TF-IDF weight is used to evaluate how important a word is to a corpus of documents.

- $tf$ : Term frequency (see later)
- $idf$ : inverse document frequency
- $N$ : total number of docs in corpus
- $df_i$ : number of docs in which  $t_i$  appears

$$w_{ij} = tf_{ij} \times idf_i \quad idf_i = \log \frac{N}{df_i}$$

💡 Gives more weight to rare words and gives less weight to common words (domain-specific stop words).



## 6.4 Feature Selection

⚠ Not all features help; learners might have difficulty with high dimensional data.

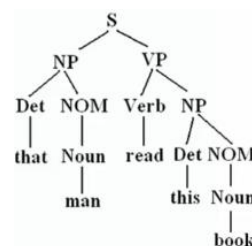
### 6.4.1 Pruning document vectors

Prune Methods; specify if and how “too frequent” or “too infrequent” words should be ignored. Different options:

- Percentual, ignores words that appear in less/more than percentage  $x$  of all documents
- Absolute, ignores words that appear in less/more than  $x$  documents
- By Rank, specifies how many % of the most frequent/infrequent words are ignored.

### 6.4.2 POS Tagging

May be helpful for feature selection. Sometimes a focus on certain classes if words is relevant (adjectives for sentiment analysis, nouns for text clustering).



## 6.5 Pattern Discovery

### 6.5.1 Document Clustering

Given a set of documents and a similarity measure among documents, find clusters such that documents in one cluster are more similar to one another and documents in separate clusters are less similar to one another.

Applications:

- Topical clustering of news stories
- Email message thread identification
- Grouping of document versions

? Which similarity measures are a good choice for comparing document vectors?

💡 Similarity measure to select depends on the vector creation method chosen beforehand.

### 6.5.2 Similarity measures

#### Jaccard Coefficient

The Jaccard coefficient is a popular similarity measure for vectors consisting of asymmetric binary attributes.

$$dist(x_i, x_j) = \frac{M_{11}}{M_{01} + M_{10} + M_{11}}$$

*Number of 1-1 matches/number of not-both-zero attribute values.*

💡 Used for e.g., a binary term occurrence vector, where 1 represents occurrence of specific word and 0 represent the abundance of said word.

	Saturn	is	the	gas	planet	with	rings	Jupiter	largest	Roman	god	of	sowing
d1	1	1	1	1	1	1	1	0	0	0	0	0	0
d2	0	1	1	1	1	0	0	1	1	0	0	0	0
d3	1	1	1	0	0	0	0	0	0	1	1	1	1

*Example document set ( $d_1$ ="Saturn is the gas planet with rings.",  $d_2$ ="Jupiter is the largest gas planet",  $d_3$ ="Saturn is the Roman god of sowing.") where documents are represented as binary term occurrence vectors.*

Jaccard similarity between the documents:

$$sim(d_1, d_2) = 0,44$$

$$sim(d_1, d_3) = 0,27$$

$$sim(d_2, d_3) = 0.18$$

#### Cosine Similarity

$$dotprod(a, b) = \sum_{i=1}^n a_i b_i = a_1 b_1 + a_2 b_2 + \dots + a_n b_n$$

Popular similarity measure for comparing weighted document vectors such as term-frequency or TF-IDF vectors.

$$cos(d_1, d_2) = \frac{dotproduct(d_1, d_2)}{len(d_1) * len(d_2)}$$

Cosine similarity for example above:

$$cos(d_1, d_2) = 0,13$$


$$sim(d_1, d_3) = 0,05$$

$$sim(d_2, d_3) = 0,00$$

$$len(a) = \sqrt{\sum_{i=1}^N a_i^2}$$



### 6.5.3 Document Classification

 Given a collection of labeled documents (training set), find a model to assign previously unseen documents a class as accurately as possible.

Applications:

- Topical clustering of news stories or web pages
- SPAM detection
- Sentiment analysis

Classification methods commonly used for text

1. Naïve Bayes
2. Support Vector Machines
3. (KNN and Decision Trees also work)

#### Example application: sentiment analysis

Given a text, assign a class of sentiment to text (sad, happy, angry, surprised, or positive, neutral, negative). This can be implemented as a supervised classification task (requires training data, i.e., <text; sentiment> pairs).

As labeling data is expensive, reviews (amazon, Gmaps, etc.) from the internet may be used as labeled data.

In preprocessing for sentiment analysis, simple syntactic analysis is useful but reasonable features for sentiment analysis might include punctuation (use of “!”, “?”, “?!”), emojis, amount of capitalization (“SCREAMING”).

#### Text Classification Tricks

Finding selective words: weight words according to their correlation with label, select top  $k$  words with highest correlation.

Sentiment lexicons: use external dictionary of opinion words.

## 7 Association Analysis

💡 We are often interested in co-occurrence relationships (e.g., words that appear together in search queries or web shop items that are frequently bought together).

### 7.1 Correlation Analysis

📄 Correlation analysis measures the degree of dependency between (*exactly*) *two* variables. For continuous variables the Person correlation coefficient (PCC) is used while for binary variables the Phi coefficient is relevant.

$$PCC(x, y) = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2} \cdot \sqrt{\sum(y_i - \bar{y})^2}} \quad \Phi(x, y) = \frac{f_{11}f_{00} - f_{01}f_{10}}{\sqrt{f_{1+}f_{+1}f_{0+}f_{+0}}}$$

Result is in a value range of [-1, 1], where 1 is a positive correlation, 0 means independent and -1 a negative correlation.

### 7.2 Association Analysis

📄 Association analysis can find *multiple item* co-occurrence relationships. It focuses on occurring items, not absent ones.

#### Example

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

Shopping Transactions

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

Examples of Association Rules

{Diaper} → {Beer}  
 {Milk, Bread} → {Eggs, Coke}  
 {Beer, Bread} → {Milk}

**Implication means  
co-occurrence,  
not causality!**

#### Terminology

- (*k*-) **Itemset**; a collection of one or more (*k*) items. E.g., {Milk, Bread, Diaper}
- **Support count ( $\sigma$ )**; the frequency of occurrence of an itemset
- **Support (*s*)**; fraction of transactions that contain an itemset
- **Frequent itemset**; an itemset whose support is greater than or equal to a minimal support (*minsup*) threshold specified by the user.

$$\sigma(\{\text{Milk, Bread, Diaper}\}) = 2 \quad s(\{\text{Milk, Bread, Diaper}\}) = \frac{2}{5}$$

#### Association Rule

📄 An implication expression of the form  $X \rightarrow Y$ , where *X* and *Y* are itemsets. An association rule states that when *X* occurs, *Y* occurs with a certain probability.

Example: {Milk, Diaper} → {Beer}

Rule Evaluation Metrics:

- Support (see above)
- Confidence (*c*) measures how often items in *Y* appear in transactions that contain *X*.

$$s = \frac{\sigma(\text{Milk, Diaper, Beer})}{|T|} = \frac{2}{5} = 0.4$$

$$c = \frac{\sigma(\text{Milk, Diaper, Beer})}{\sigma(\text{Milk, Diaper})} = \frac{2}{3} = 0.67$$

Challenges:

- Mining associations from large amounts of data can be computationally expensive.
- Algorithms often discover many associations.

## Association Rule Mining Task

Given a set of transactions  $T$ , the goal of association rule mining is to find all rules having  $support \geq minsup$  threshold and  $confidence \geq minconf$  threshold.

1. List all possible association rules
2. Compute the support and confidence for each rule
3. Remove rules that fail the thresholds

! Brute Force Approach is computationally prohibitive due to large number of candidates!

Several rules can be mined from one itemset (e.g.,  $\{Milk, Bread, Diaper\}$ ). As they all share an identical support but have different confidence, we may decouple the support and confidence requirements.

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

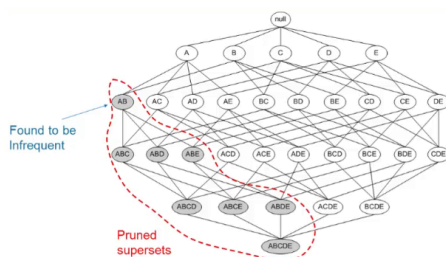
### 💡 Idea: Two-Step Approach

1. Frequent itemset generation: generate all itemsets whose support  $\geq minsup$
2. Rule generation: generate high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset.

! Given  $d$  items, there are  $2^d$  candidate itemsets!

## 7.2.1 Frequent Itemset Generation

### Apriori Principle



📄 If an itemset is frequent, then all its subsets must also be frequent (support is equal or even greater than). This is known as the *anti-monotone* property of support.

💡 As the support of a *superset* never exceeds the support of its *subsets*, finding an infrequent *subset* lets us eliminate all supersets.

Algorithm:

```

let k=1
Generate frequent itemsets of len()==1
repeat:
  generate len()==k+1 candidate itemsets from len()==k frequent itemsets
  prune candidate itemsets that cannot be frequent (apriori principle)
  count the support of each candidate by scanning the DB
  eliminate candidates that are infrequent
until: no new frequent itemsets are identified
  
```

## 7.2.2 Rule Generation

Given a frequent itemset  $L$ , find all non-empty subsets  $f \subset L$  such that  $f \rightarrow L - f$  satisfies the minimum confidence requirement.

Example frequent itemset:  $\{Milk, Beer, Diaper\}$ ,  
example rule:  $\{Milk, Diaper\} \Rightarrow Beer$

$$c = \frac{\sigma(Milk, Diaper, Beer)}{\sigma(Milk, Diaper)} = \frac{2}{3} = 0,67$$

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

In general, confidence does not have an anti-monotone property, but confidence of rules generated from the *same itemset* have an anti-monotone property.

– If  $\{A, B, C, D\}$  is a frequent itemset, then the candidate rules are:

ABC $\rightarrow$ D,	ABD $\rightarrow$ C,	ACD $\rightarrow$ B,	BCD $\rightarrow$ A,
A $\rightarrow$ BCD,	B $\rightarrow$ ACD,	C $\rightarrow$ ABD,	D $\rightarrow$ ABC,
AB $\rightarrow$ CD,	AC $\rightarrow$ BD,	AD $\rightarrow$ BC,	BC $\rightarrow$ AD,
BD $\rightarrow$ AC,	CD $\rightarrow$ AB,		

Confidence is anti-monotone with respect to the number of items on the right-hand side (condition) of the rule.

– If  $|L| = k$ , then there are  $2^k - 2$  candidate association rules (ignoring  $L \rightarrow \emptyset$  and  $\emptyset \rightarrow L$ )

(...)

## 7.2.3 Handling Continuous & Categorical Attributes

### Categorical attributes

Transform categorical attribute into asymmetric binary variables. Introduce a new item/column for each distinct attribute-value pair.

Example: replace browser-type attribute with:

- “Browser type = IE” : 0
- “Browser type = “Mozilla” : 1

Session Id	Country	Session Length (sec)	Number of Web Pages viewed	Gender	Browser Type	Buy
1	USA	982	8	Male	IE	No
2	China	811	10	Female	Netscape	No
3	USA	2125	45	Female	Mozilla	Yes
4	Germany	596	4	Male	IE	Yes
5	Australia	123	9	Male	Mozilla	No
...	...	...	...	...	...	...

! Potential issues: What if attribute has many possible values? What if distribution of attribute values is highly skewed?

💡 Potential solutions:

- Aggregate low-support attribute values.
- Drop the highly frequent item


### Continuous attributes

Transform continuous attributes using discretization (see *Design Issues*).

- equal-width binning
- equal-frequency binning



## 7.2.4 Interestingness Measures

 Association rule algorithms tend to produce too many rules and many of them are uninteresting or redundant. Interestingness of patterns depends on application. They can be used to prune or rank the previously derived rules. There are other measures than support and confidence.

### Drawback of Confidence

Association rule:  $tea \rightarrow coffee$

- $c(tea \rightarrow coffee) = 0,75$
- $\sigma(coffee) = 0,9$

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

! Although confidence is high, rule is misleading as the fraction of coffee drinkers is higher than the confidence of the rule.

### Lift

The *lift* of an association rule  $X \rightarrow Y$  is defined as:

$$lift = \frac{c(X \rightarrow Y)}{s(Y)}$$

The confidence is normalized by support of consequent. Interpretation:

- If  $lift > 1$ , then  $X$  and  $Y$  are positively correlated
- If  $lift = 1$ , then  $X$  and  $Y$  are independent
- If  $lift < 1$ , then  $X$  and  $Y$  are negatively correlated

$lift(tea \rightarrow coffee) = 0.8333$ , therefore, its negatively correlated and we don't want to use this rule.