Data Mining

Databases and Information Systems

Fabian Panse

panse@informatik.uni-hamburg.de

University of Hamburg

Acknowledgements

These slides are based on slides provided by

- Prof. Dr. Wolfgang Menzel
  University of Hamburg
  https://nats-www.informatik.uni-hamburg.de/

- Prof. Dr. Erhard Rahm
  University of Leipzig
  http://dbs.uni-leipzig.de/

Overview

- **Introduction**
  - What is data mining?
  - Example tasks
  - Data mining process
- **Data Preprocessing**
  - Outlier detection, number of values reduction, ...
- **Classification**
  - Nearest-Neighbor classifier, decision trees, ...
- **Clustering**
  - K-means, canopy clustering, hierarchical clustering, ...
- **Association rules (Market Basket Analysis)**
  - Apriori algorithm, support and confidence

What is Data Mining?

- **Mining as a metaphor:**
  - The search for the precious thing amongst the overburden
  - Goal is not always clearly specified
    - What's precious?
    - Successful even if looking for gold but found diamonds
- **Mining is an explorative activity**
  - Finding cues
  - Making hypotheses
  - Evaluating hypotheses
  - Getting the precious stuff

What is Data Mining? (2)

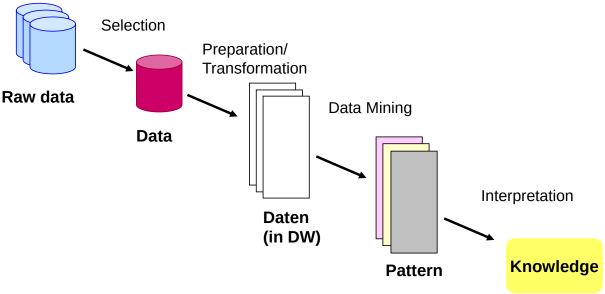
- **“Data mining” is a misleading metaphor!**

	mining	data mining
target	coal, gold, ore, ...	trends, associations
overburden	dirt, rock	“useless” data

- **Alternative notion:**
  - KDD: Knowledge Discovery in Databases
- **Alternative view:**
  - KDD is a complex process, DM only one step in it

Knowledge Discovery in Databases (KDD)

- **(Semi-)automatic extraction of knowledge from databases which is:**
  - valid (in the statistical sense),
  - unknown so far,
  - and potentially useful
- **Combination of approaches from databases, statistics and AI (machine learning)**



Example Tasks

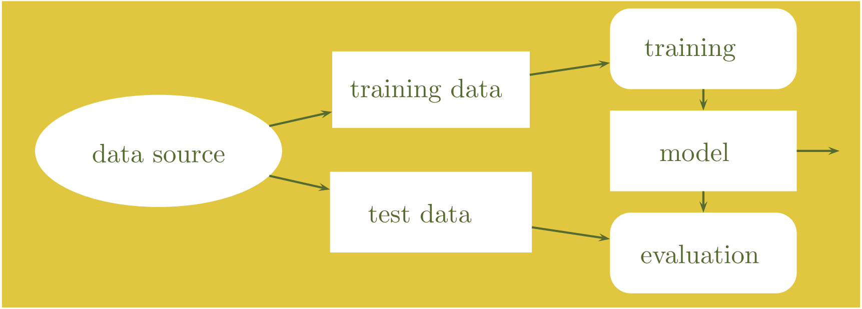
- **Customer relationship management**
  - Grouping of customer populations (tailored marketing)
  - Prediction of customer behaviour (individualized marketing)
  - Risk assessment (risky credits, fraudulent credit card use)
- **Fault analysis**
  - Interdependencies between faults
  - Interdependencies between production processes or maintenance procedures and faults
- **Time-series analysis**
  - Trend detection
  - Stock market development
  - Event prediction (stock market crashes, bankruptcies, natural disasters)
  - Intrusion detection
- **Web Usage and Text Mining**

Why is Data Mining special?

- **There is no silver bullet for data mining!**
  - Many different techniques
  - Extremely laborious parameter tuning
  - Very few clues for performance predictions
- **Data mining aims at partially correct solutions**
  - Vague task descriptions
  - No perfect solution methods
  - Issues: quality, portability to similar application domains
- **Contrast: querying a database**
  - Precise semantics
  - 100% correct results must be guaranteed
  - Issues: performance, maintenance, ...

Evaluation

- Models in data mining are usually developed and evaluated on data collections
- For development purposes a clear separation between training and test data is necessary



Data Mining as a Process

Data Mining as a Process

1. **Problem formulation**
   - Based on domain-specific knowledge and experience
   - Which $\left\{ \begin{array}{c} \text{categories} \\ \text{dependencies} \end{array} \right\}$ are of particular interest?
2. **Selection of a suitable model class**
   - Based on the available and feasible data mining techniques
   - Which kind of model seems most promising given the available data?

Data Mining as a Process

3. **Data collection**
   - Designed experiment (artificial data generation):
     - Data can be generated on demand
     - Sampling distribution is known (but realistic?)
   - Observational approach:
     - Random data collection
     - Sampling distribution is unknown or implicit in the data collection procedure
     - Data can be biased
   - Data collection affects the outcome (garbage in, garbage out)

Data Mining as a Process

4. **Data preparation**
   - Data cleansing
     - Outliers: measurement, coding or recording errors
     - Outliers: abnormal but natural values
     - Missing values
   - Outlier treatment
     - Only if outlier detection is not the goal
     - Either removal as part of preprocessing
       or application of techniques insensitive to outliers
   - Data transformation
     - Rescaling: making features comparable
     - Dimensionality reduction: gaining efficiency
     - Binning: map numerical values to qualitative classes

Data Mining as a Process

5. **Training of model parameters**
   - Estimation of stochastic parameters
   - Adjustment of threshold values
   - Adjustment of weights
   - ...
   - Training is usually an optimization problem
6. **Evaluation and interpretation of results**
   - Defining metrics for quality assessment
   - Measuring the quality of results on held out test data
   - Summarization and visualization of results
   - Usually simple models ...
     - ... are better trainable
     - ... are better interpretable
     - ... but less accurate

CRISP-DM

- Cross industry standard process for data mining
- Life-cycle model

1. **Business understanding phase**
   - Analysis of objectives and requirements
   - Problem definition
   - Initial strategy development
2. **Data understanding phase**
   - Data collection
   - Exploratory data analysis
   - Assessment of data quality
3. **Data preparation phase**
   - Cleansing, transformation etc.

CRISP-DM

4. **Modelling phase**
   - Selection of modelling techniques and tools
   - Parameter tuning / optimization
   - Data analysis
5. **Evaluation Phase**
   - Evaluation of the model
   - Comparison of the outcome to the initial objectives
   - Deployment decision
6. **Deployment phase**
   - Reporting
   - Transfer to other application cases
   - If applicable: introduction into day-to-day business

Data Preprocessing

- **Goals:**
  - Decrease runtime of data mining process
  - Decrease resource requirements of data mining process
  - Increase quality of mining result
- **Important aspects:**
  - Handling of missing data
  - Detecting outliers
  - Reducing the number of dimensions
  - Reducing the number of values
  - Transforming data values (e.g. binning, rescaling)
- Depends on data type
- Uses similarity/distance measures

Data Types

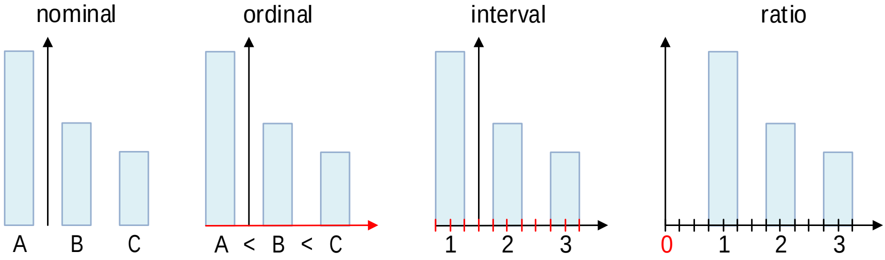
- **Nominal scale**
  - No problem-specific order and distance relation
  - Mathematical Operators: $=, \neq$
  - Central Tendency: mode (most often occurring value)
  - Examples: color, zip-code
- **Ordinal scale**
  - Problem-specific order relation
  - No problem-specific distance relation
  - Mathematical Operators: $=, \neq, >, <$
  - Central Tendency: mode, median
  - Examples: income classes, medal ranks, age

Data Types

- **Interval scale**
  - Problem-specific order and distance relation
  - No problem-specific zero point
  - Differences meaningful, ratios meaningless
  - Mathematical Operators: $=, \neq, >, <, +, -$
  - Central Tendency: mode, median, arithm. mean, deviation
  - Examples: temperature (Celsius), date (B.C./A.D.)

- **Ratio scale**
  - Problem-specific zero point (absence of the feature)
  - Differences and ratios meaningful
  - Mathematical Operators: $=, \neq, >, <, +, -, \cdot, /$
  - Central Tendency: mode, median, arithm./geom. mean, deviation
  - Examples: temperature (Kelvin), speed, length, age, quantity

Data Types - Summary



Scale	Math. Operators	Advanced Operations	Central Tendency
nominal	$=, \neq$	grouping	mode
ordinal	$>, <$	sorting	median
interval	$+, -$	yardstick	mean, deviation
ratio	$\cdot, /$	ratio	geometric mean

Sources: https://de.wikipedia.org/wiki/Skalenniveau and https://en.wikipedia.org/wiki/Level_of_measurement

Metrics

- Many data mining techniques are based on some notion of distance, similarity or dissimilarity
- e.g. MINKOVSKIJ Distance

$$d(\vec{x_1}, \vec{x_2}) = \left(\sum_{i=1}^{n} |x_{1i} - x_{2i}|^m\right)^{1/m}$$

- $m = 1$: Manhattan Distance, City Block Distance
- $m = 2$: EUCLIDIAN Distance
- $m = \infty$: max-Distance, TCHEBYCHEV Distance

- **Properties:**
  - Self identity: $\forall \vec{x}: d(\vec{x}, \vec{x}) = 0$
  - Positivity: $\forall \vec{x_1} \neq \vec{x_2}: d(\vec{x_1}, \vec{x_2}) > 0$
  - Symmetry: $\forall \vec{x_1}, \vec{x_2}: d(\vec{x_1}, \vec{x_2}) = d(\vec{x_2}, \vec{x_1})$
  - Triangle inequation: $\forall \vec{x_1}, \vec{x_2}, \vec{x_3}: d(\vec{x_1}, \vec{x_3}) \leq d(\vec{x_1}, \vec{x_2}) + d(\vec{x_2}, \vec{x_3})$

Metrics

- **Metrics for complex objects**
- **Sets/Bags**
  - JACCARD Coefficient

$$\text{Jacc}(S_1, S_2) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}$$

  - Cosine Similarity
- **Strings**
  - Token-based (e.g. JACCARD Coefficient of q-grams)
  - Sequence-based (e.g. LEVENSHTEIN Distance)
  - Phonetic (e.g. Soundex)
  - Hybrid (e.g. MONGE-ELKAN Similarity)
- **Signatures, histograms, probability distributions**
  - Earth Mover's Distance

Metrics - Levenshtein Distance

- **String edit distance,** LEVENSHTEIN Distance
- **Minimal effort to transform a sequence into another one**
- **Basic operations**
  - Substitution
  - Insertion
  - Deletion
- **Computation rules:**

$$d(x_i, y_0) = i$$

$$d(x_0, y_j) = j$$

$$d(x_i, y_j) = \min \begin{cases} d(x_{i-1}, y_{j-1}) + d(x_1, y_1) \\ d(x_{i-1}, y_j) + 1 \\ d(x_i, y_{j-1}) + 1 \end{cases}$$

Metrics - Levenshtein Distance

- Finding the minimum distance is an optimization problem
  $\Rightarrow$ dynamic programming

local distances

		c	h	e	a	t
	0	1	1	1	1	1
c	1	0	1	1	1	1
o	1	1	1	1	1	1
a	1	1	1	1	0	1
s	1	1	1	1	1	1
t	1	1	1	1	1	0

global distances

		c	h	e	a	t
	0	1	2	3	4	5
c	1	0	1	2	3	4
o	2	1	1	2	3	4
a	3	2	2	2	2	3
s	4	3	3	3	3	3
t	5	4	4	4	4	3

- Space and time requirements $\mathcal{O}(m \cdot n)$

Handling of Missing Data

- **Some data mining tools are insensitive to missing data**
- **Ignore all incomplete objects**
  - Might result in the loss of a substantial amount of data
  - Problem: maybe a systematic correlation between target of mining process and missing values (e.g. customers who do not answer a specific question of a survey)
- **Manual completion**
  - Expensive in time and money
- **Automatic completion**
  - Using a global constant
  - Using the global mean value
  - Using a class-dependent mean value
  - Use a predictive model, e.g. based on feature correlations

Outlier Detection

- **Not applicable if the application is aimed at outlier detection**
  e.g. fraudulent credit card transactions
- **The task:** find *k* out of *n* tuples, which are
  - Considerably dissimilar
  - Exceptional
  - Inconsistent with the remaining data

Outlier Detection

- **Manual detection supported by visualization tools**
  - Only for low dimensional data
- **Statistical methods**
  - Threshold for the variance, e.g. two times variance
  - Only applicable if the distribution is known
- **Using domain knowledge**
  - Value restrictions, e.g. $0 \leq \text{age} < 150$
  - Less applicable for multi-dimensional data
- **Distance-based detection**
  - A sample is an outlier if it has not enough neighbors
- **Deviation-based methods**
  - Measure the dissimilarity of a data set (e.g. variance)
  - Determine the smallest subset of data that if removed results in the largest reduction of dissimilarity
  - Combinatorics of subset selection $\Rightarrow$ extremely expensive

Outlier Detection - Example

- **Sample data set:**

$$S = \{s_1, ..., s_7\} = \{(2,4), (3,2), (1,1), (4,3), (1,6), (5,3), (4,2)\}$$

- **Distance matrix:**

	s_1	s_2	s_3	s_4	s_5	s_6	s_7
s_1		2.236	3.162	2.236	2.236	3.162	2.828
s_2	2.236		2.236	1.414	4.472	2.236	1.000
s_3	3.162	2.236		3.605	5.000	4.472	3.162
s_4	2.236	1.414	3.605		4.242	1.000	1.000
s_5	2.236	4.472	5.000	4.242		5.000	5.000
s_6	3.162	2.236	4.472	1.000	5.000		1.414
s_7	2.828	1.000	3.162	1.000	5.000	1.414	

Outlier Detection - Example

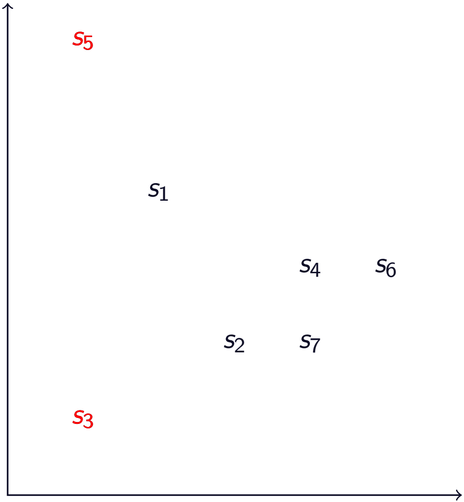
- **Neighborhood:** $d \leq \theta = 3$

	s_1	s_2	s_3	s_4	s_5	s_6	s_7
s_1		2.236	3.162	2.236	2.236	3.162	2.828
s_2	2.236		2.236	1.414	4.472	2.236	1.000
s_3	3.162	2.236		3.605	5.000	4.472	3.162
s_4	2.236	1.414	3.605		4.242	1.000	1.000
s_5	2.236	4.472	5.000	4.242		5.000	5.000
s_6	3.162	2.236	4.472	1.000	5.000		1.414
s_7	2.828	1.000	3.162	1.000	5.000	1.414	

- **Number of points in the neighborhood**

Sample	s_1	s_2	s_3	s_4	s_5	s_6	s_7
	4	5	1	4	1	3	4

Outlier Detection - Example



Dimensionality Reduction

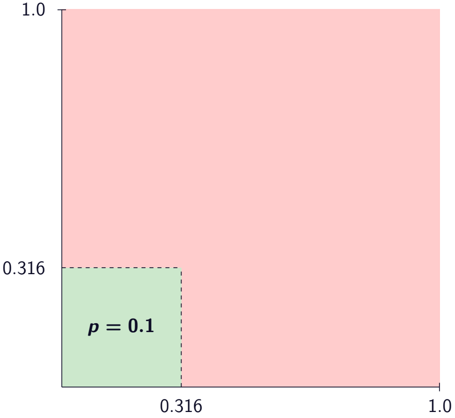
- **All high-dimensional data spaces are sparse:**
  - Keeping the same object density in a space with more dimensions requires exponentially more objects
  - To enclose a prespecified portion of objects, an increasingly large part of the hypercube needs to be "encircled":

portion p	dimensionality n	edge length $p^{1/n}$
0.1	1	0.100
0.1	2	0.316
0.1	3	0.464
	...	...
0.1	10	0.794

  - Almost every object is closer to an edge of the cube than to another sample object
  - Almost every object is an outlier

Dimensionality Reduction

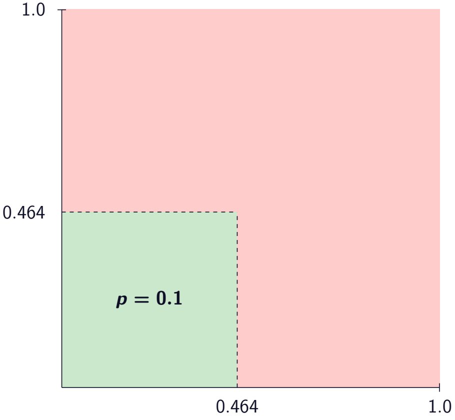
- **Example:** portion $p = 0.1$, dimensionality $n = 2$



Note: $0.316^2 = 0.1$

Dimensionality Reduction

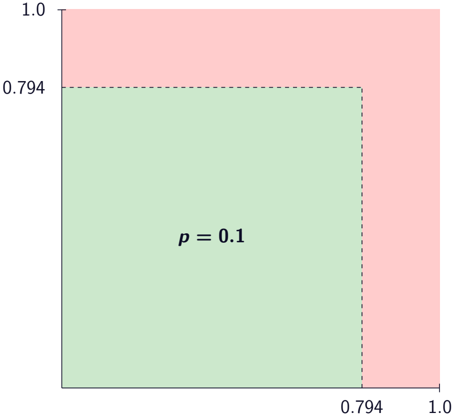
- **Example:** portion $p = 0.1$, dimensionality $n = 3$



Note: $0.464^3 = 0.1$

Dimensionality Reduction

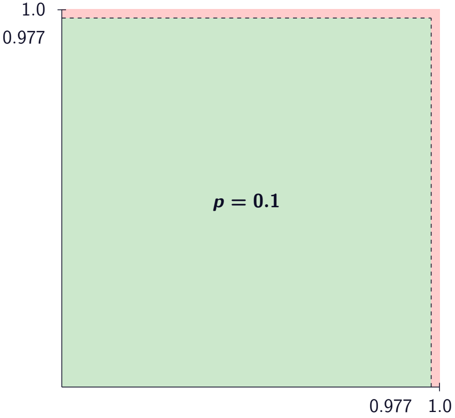
- **Example:** portion $p = 0.1$, dimensionality $n = 10$



Note: $0.794^{10} = 0.1$

Dimensionality Reduction

- **Example:** portion $p = 0.1$, dimensionality $n = 100$



Note: $0.977^{100} = 0.1$

Dimensionality Reduction

- **Question:** Which data can be discarded without sacrificing the quality of the data mining results?
- **Too many dimensions:**
  - Mining results degrade (insufficient amount of data)
  - Resulting model is incomprehensible
  - Problem becomes intractable
- **Too few dimensions:**
  - Data dependencies are lost
  - Mining results degrade (limited expressiveness)

Dimensionality Reduction

- **Feature selection approaches:**
  - Idea: discard features (i.e. attributes, dimensions) which
    - have many inaccurate/inconsistent values
    - have many missing values
    - do not provide much (relevant) information
    - contribute least to the overall class distinction capability (task specific criterion)
  - Approaches: Feature ranking, minimum subset selection

- **Feature composition approaches:**
  - Idea: features are composed into a new feature set with reduced dimensionality

  $\Rightarrow$ Given feature space is transformed into a more compact feature space without losing relevant information
  - Approach: Principal component analysis

Dimensionality Reduction - Minimum Subset Selection

- **Idea:** select a feature set which maximizes the class distinction capability
- Assuming normal distribution, class distinction can be defined as:

$$cd_{A,B} = \frac{|\mu_A - \mu_B|}{\sqrt{\frac{\sigma_A}{n_A} + \frac{\sigma_B}{n_B}}} \qquad \text{where } A \text{ and } B \text{ are the compared classes}$$

- **Class distinction is maximal if:**
  - Means μ_A and μ_B are far away and
  - Variances σ_A and σ_B are low (despite many samples n_A and n_B)
- **Simple case:** features are considered as independent
- Compute all pairwise class distinction values
- Retain a feature, if it is relevant for any of the pairwise comparisons

Dimensionality Reduction - Example

- **Assumption:** normal distribution, independent features
- **Sample data set:**

X	Y	C	X	Y	C
0.3	0.7	A	0.2	0.9	B
0.6	0.6	A	0.7	0.7	B
0.5	0.5	A	0.4	0.9	B

- **Data sets separated by class and dimension**

$$X_A = \{0.3, 0.6, 0.5\}, \qquad X_B = \{0.2, 0.7, 0.4\},$$
$$Y_A = \{0.7, 0.6, 0.5\}, \qquad Y_B = \{0.9, 0.7, 0.9\}$$

Dimensionality Reduction - Example

- **Data sets separated by class and dimension**

$$X_A = \{0.3, 0.6, 0.5\}, \qquad X_B = \{0.2, 0.7, 0.4\},$$
$$Y_A = \{0.7, 0.6, 0.5\}, \qquad Y_B = \{0.9, 0.7, 0.9\}$$

$$cd_{X_A,X_B} = \frac{|\mu_{X_A} - \mu_{X_B}|}{\sqrt{\frac{\sigma_{X_A}}{n_{X_A}} + \frac{\sigma_{X_B}}{n_{X_B}}}} = \frac{|0.4667 - 0.4333|}{0.3671} = 0.0908$$

$$cd_{Y_A,Y_B} = \frac{|\mu_{Y_A} - \mu_{Y_B}|}{\sqrt{\frac{\sigma_{Y_A}}{n_{Y_A}} + \frac{\sigma_{Y_B}}{n_{Y_B}}}} = \frac{|0.6 - 0.8333|}{0.268} = 0.8706$$

$\Rightarrow$ Discard dimension X

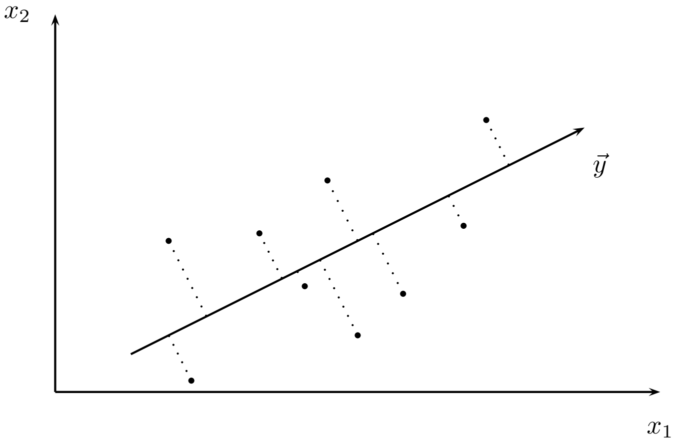
Dimensionality Reduction - Principal Component Analysis

- **Construction of a new feature space:**
  - New features are computed as linear combinations of old ones

    $$\vec{y} = \sum_{i=1}^{n} w_i \cdot x_i$$

  - Each vector of weights is called a principal component
  - Principal components are ranked according to their usefulness
    $\Rightarrow$ components with lowest rank can be discarded
- **Idea:** high information content corresponds to high variance
  - Not always valid (e.g. distribution with many very small and many very large values)
- Choose weights so that $\vec{y}$ has the largest possible variance
  - $\vec{y}$ is a new axis in the direction of maximum variance

Dimensionality Reduction - Principal Component Analysis



Number of Values Reduction

- **Benefits:**
  - Increase of performance (less values to process)
  - Often simplifies the mining process (e.g. finding of rules)

- **One dimensional:** feature discretization (binning)

  $\Rightarrow$ Mapping values to intervals
  - Value exchange techniques
  - Merging techniques

- **Multi dimensional:** clustering of feature vectors
  - Splitting techniques

Number of Values Reduction

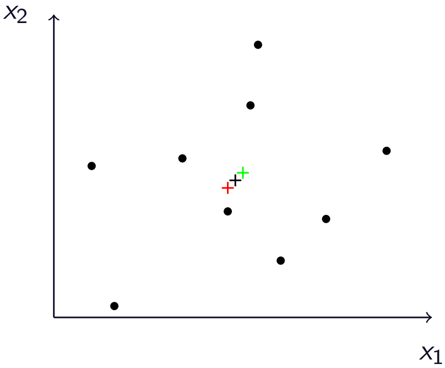
- **Splitting Techniques:** splitting of bins (vector quantization)
  1. Start: All values in a single bin/cluster
  2. Compute the mean of all data values (centroid)
  3. Split each centroid into 2 or more centroids (bins)
  4. Assign data points to the nearest centroid (bin)
  5. Continue until enough bins have been generated

- **Learning without teacher (i.e. no labeled training objects)**
- **Based on distance functions**

  $\Rightarrow$ Problems in high dimensional spaces

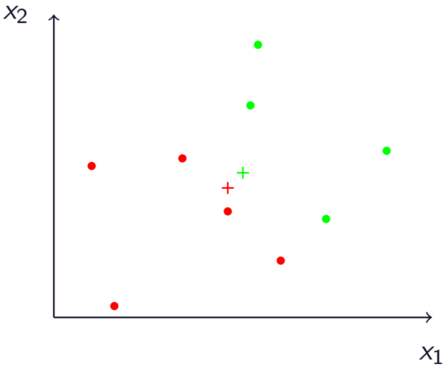
Number of Values Reduction

Input: 9 values
Output: 4 values



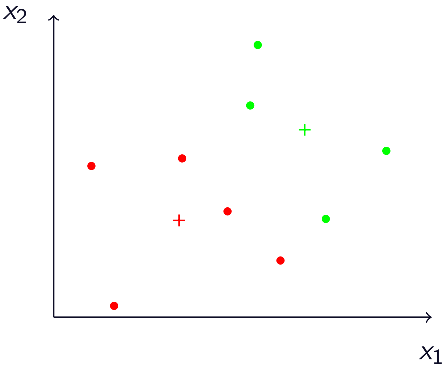
Number of Values Reduction

Input: 9 values
Output: 4 values



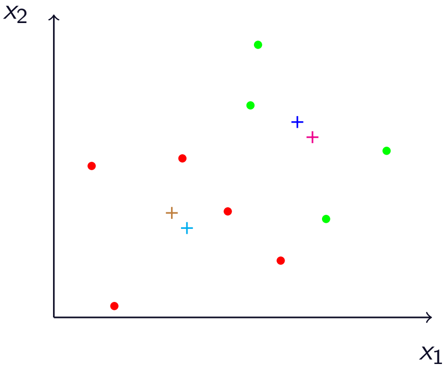
Number of Values Reduction

Input: 9 values
Output: 4 values



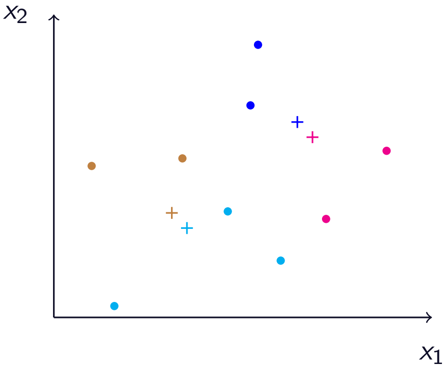
Number of Values Reduction

Input: 9 values
Output: 4 values



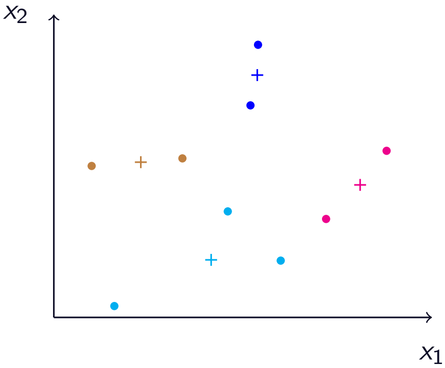
Number of Values Reduction

Input: 9 values
Output: 4 values



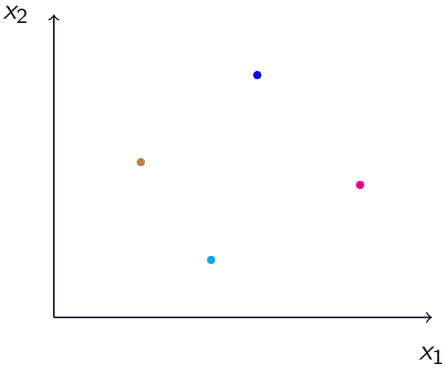
Number of Values Reduction

Input: 9 values
Output: 4 values



Number of Values Reduction

Input: 9 values
Output: 4 values



Classification

- **Applications in**
  - Customer Relationship Management: tailored marketing
  - Banking: credit authorization
- **Given:**
  - Set of objects (database) $D = \{o_1, o_2, \ldots, o_m\}$ where each object o_i corresponds to a k-dimensional vector $\langle o_{i,1}, \ldots, o_{i,k} \rangle$
  - Set of classes $C = \{c_1, c_2, \ldots, c_n\}$ (usually: $|D| \gg |C|$)
  - Set of labeled training objects $T \subset D$
- **Goal:** Find a mapping $f : D \to C$ that assigns objects to classes
  - Class is a set of objects: $c = \{o | o \in D, f(o) = c\}$
  - No object belongs to several classes
  - Classes are predefined $\Rightarrow$ Supervised learning, learning with a teacher
- **Notation:**
  - $c_t(o)$: class assignment in the training data
  - $c(o)$: class assignment by the classifier

Classification

1. Construction of the classifier

Training Data

Classification Algorithm

Classifier

NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

if rank = 'professor'
or years > 6
then tenured = 'yes'

2. Using the classifier to make predicitions

Unknown Data

(Jeff, Professor, 4)

Classifier

Tenured?

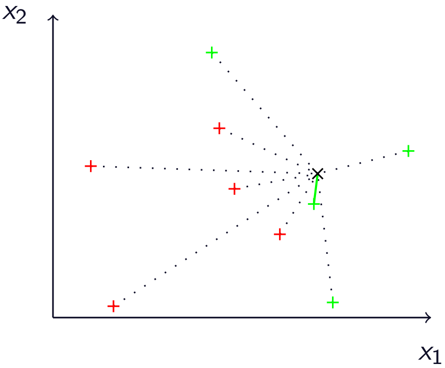
Nearest-Neighbor Classifier

- **Direct approach:** training objects are
  - directly stored in the classifier and
  - used for classification
- **Given:**
  - Training data set $T = \{o_1, o_2, \ldots, o_r\}$ with class labels $c_t : T \rightarrow C$
  - Object distance function d
- **Nearest neighbor:**
  - The class of an object o is set to the class label of its nearest training object o^*, i.e.

$$c(o) = c_t(o^*) \quad \text{where} \quad o^* = \arg\min_{o' \in T} d(o, o')$$

(here we assume that the distance $d(o, o')$ is different for every $o' \in T$)

Nearest-Neighbor Classifier



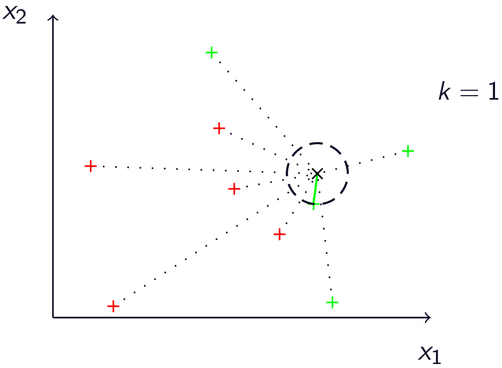
Nearest-Neighbor Classifier

- ***k*-nearest neighbors:**
  - Determine the set N of the k nearest neighbors of o in T
  - Choose the class with the maximum number of training objects in N, i.e.

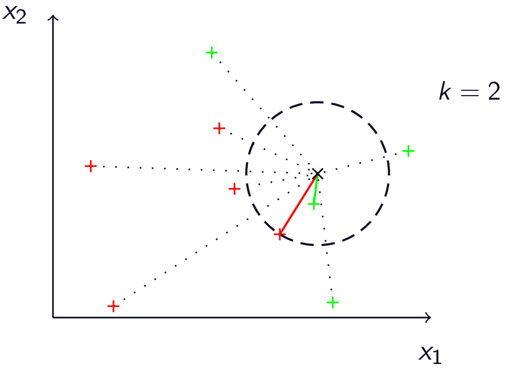
$$c(o) = \arg\max_{c \in C} |\{o' | o' \in N, c_t(o') = c\}$$

  - More robust against singular data points
  - But more expensive

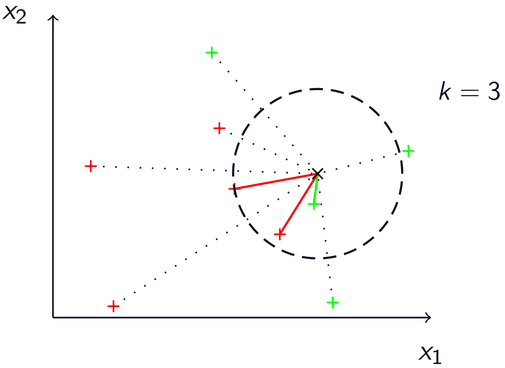
Nearest-Neighbor Classifier



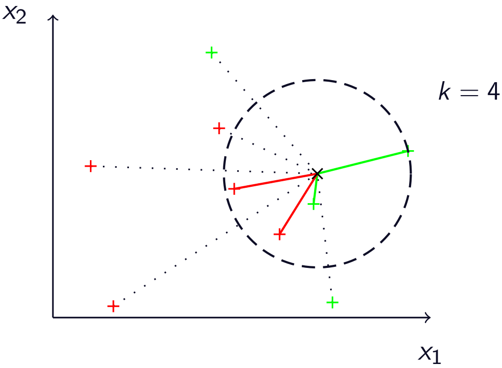
Nearest-Neighbor Classifier



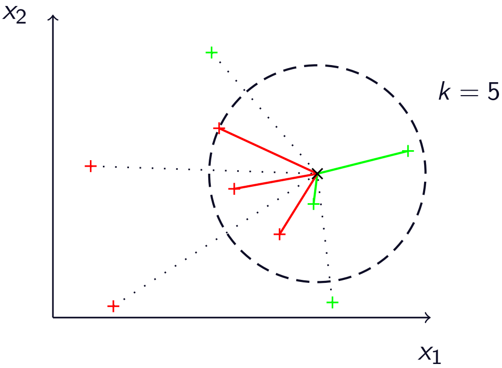
Nearest-Neighbor Classifier



Nearest-Neighbor Classifier



Nearest-Neighbor Classifier



Nearest-Neighbor Classifier

- **NN-classifier is instance-based**
  - Model size and classification effort grow linearly with amount of training data
    - Theoretical classification effort: one distance computation per training object
    - Several techniques for increasing performance of NN-search (e.g. filtering, pruning)
  - No generalization of the available training data
- **Generalizing models required**
  - Use class representatives as training objects
    - Means of classes
    - Means of class-dependent clusters

Threshold-Based Classifiers

- **Simple generalizing model for classification with two classes**
  - Threshold divides the data space into two subspaces along a single dimension

$$c(o_i) = \begin{cases} c_1, & o_{i,j} > \theta \\ c_2, & else \end{cases}$$

  - Analogue separation criteria for non-numeric data

Threshold-Based Classifiers

- **Choice of the optimal threshold:**
  - Minimizing the classification error on the training objects

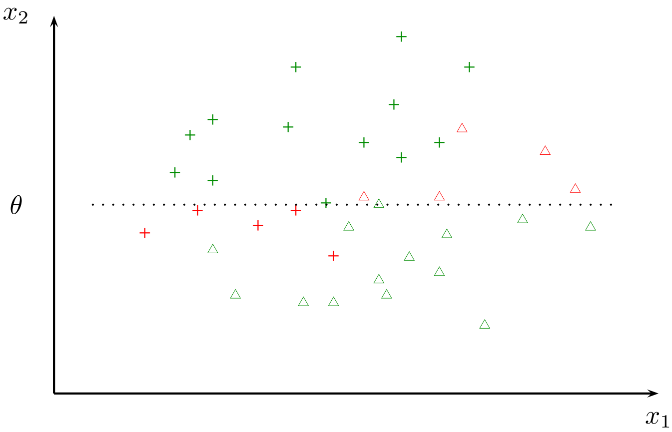
$$\theta = \arg\min_\theta |\{o|o \in T, c(o) \neq c_t(o)\}|$$

  - For numeric data: minimizing the total distance of misclassified samples to the threshold

$$\theta = \arg\min_\theta \sum_{o_i \in T, c(o_i) \neq c_t(o_i)} |o_{i,j} - \theta|$$

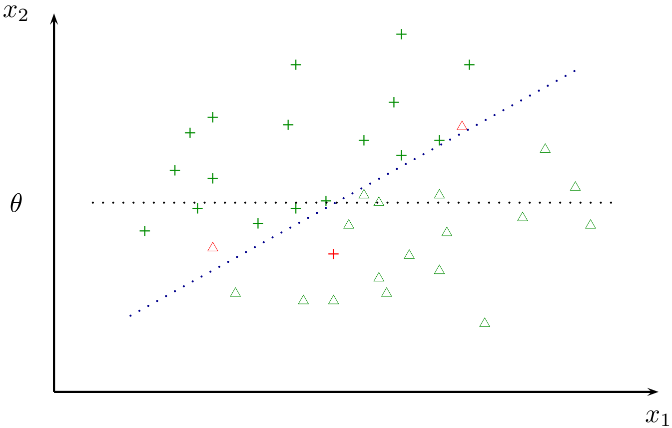
Threshold-Based Classifiers

- **Insufficient to separate more difficult distributions**



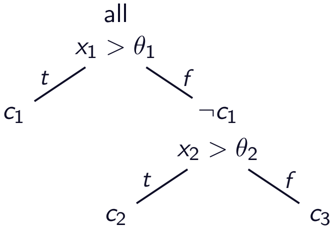
Threshold-Based Classifiers

- **Better (linear) class separation**



Decision Trees

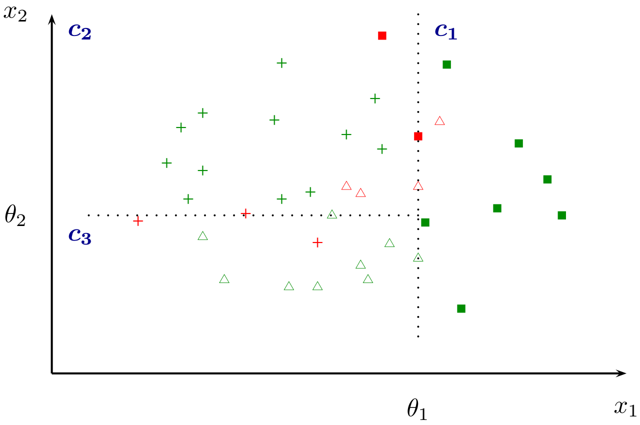
- **Extension of threshold-based classifiers to multiple classes:**
  - Decomposition into a sequence of sub-decisions
  - Multi-branch splits possible
  - Each leaf node represents a class $c \in C$



- **Finding the optimal decision tree is NP complete**

  $\Rightarrow$ Deterministic (non-backtracking), greedy algorithms

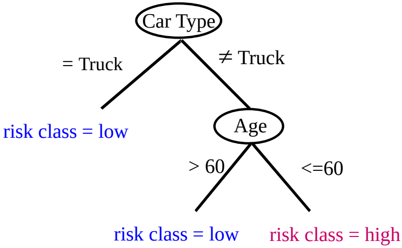
Decision Trees



Decision Trees

- **Explicit, simple to understand representation of classification knowledge**

ID	Age	Car Type	Risk
1	23	Family	high
2	17	Sport	high
3	43	Sport	high
4	68	Family	low
5	32	Truck	low



Representation in the form of rules:

If (Car Type = Truck) then risk class = low
If (Car Type ≠ Truck AND Age > 60) then risk class = low
If (Car Type ≠ Truck AND Age <= 60) then risk class = high

- **Usage of the decision tree to make predictions:**
  - ⇒ Top-down traversing of the tree from the root to one of the leaf nodes
  - ⇒ Assignment of the object to the class of the resultant leaf node

Construction of a Decision Tree

- **Basic algorithm:**
  - Initially, all training objects belong to the root
  - Selection of the next attribute (split strategy), e.g. by maximization of the information gain
  - Partitioning of the training objects with the selected split attribute
  - Algorithm is applied to each partition recursively
- **Stop criterion:**
  - No further split attributes
  - All training objects of a node belong to the same class
- **Types of splits:**
  - Categorical attribute: Split condition of the form "attribute $=$ a" or "attribute $\in$ set" (many possible subsets)
  - Numerical attribute: Split condition of the form "attribute $<$ a" (many possible split points)

Construction of a Decision Tree

- **ID3:** split along a dimension as to maximize information gain
- Entropy of a set S partitioned into k classes

  $$E(S) = -\sum_{i=1}^{k} p(c_i) \cdot \log p(c_i)$$

  where $p(c_i)$ is the relative amount of objects in S that are labeled with class c_i, i.e. $p(c_i) = |\{o \mid o \in S, c_t(o) = c_i\}|/|S|$
- Entropy of a test set T partitioned into n subsets by an attribute test X with n possible outcomes

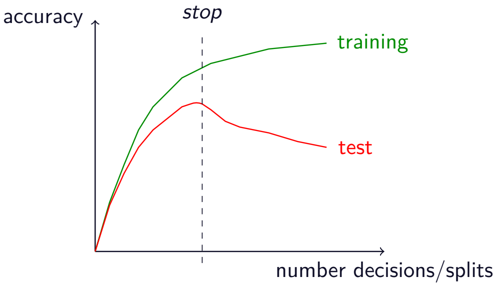
  $$E_X(T) = -\sum_{i=1}^{n} \frac{|T_i|}{|T|} \cdot E(T_i)$$

- Information gain of the attribute test

  $$G(X) = E(T) - E_X(T)$$

Construction of a Decision Tree

- **Problem of overfitting**
  - Splitting until no object is misclassified usually means to adapt the classifier too much to the training data
    - "Learning off by heart" (dt. "auswendig lernen")
    - Degrading performance on held out test data
  - Cut-off criterion required, or post-pruning



Construction of a Decision Tree

- **Decision rules can be extracted from a decision tree**
  - IF part: combine all tests on the path from the root node to the leaf node
  - THEN part: the final classification
- **Enables a simple extraction of 'real' knowledge from the learned classifier**

Evaluation

- **Goal:** predicting future model performance
  - Estimation of an error rate on a sample of test cases
- **Testing on the training data is too optimistic**
  - Error rate is significantly lower compared to a real application scenario

  $\Rightarrow$ Evaluation only on separate test data
- **But: labeled data (test and training) is usually not much available**
  - Manual data cleansing
  - Manual class assignment
- **Using data for training and testing: resampling**

Resampling Methods

- **Held out data:**
  - 30% ... 50% of the data are reserved for testing
  - Training and test data are independent
  - Error estimation is pessimistic and depends on the partitioning
    $\Rightarrow$ repeat the measurement with different partitionings and average the results
- **Leave one out:**
  - Use $n-1$ samples for training and evaluate on the n-th one
  - Repeat with all n samples
  - Extremely expensive
- **n-fold cross validation**
  - Combines hold-out and leave-one-out
  - Divide data set into p partitions
  - Use $p-1$ partitions for training; evaluate on the remaining one
  - Repeat with different partitionings

Quality Measures

- **Given:**
  - Test set S
  - Misclassified test data objects $M \subseteq S$
- **Error rate:**

$$e = \frac{|M|}{|S|}$$

- **Accuracy:**

$$a = 1 - e = \frac{|S| - |M|}{|S|}$$

Quality Measures

- **Contrastive analysis:** comparison with a baseline case
  - Absolute improvement/degradation:

$$\Delta_{abs} a = a_n - a_{n-1}$$

$$\Delta_{abs} e = e_n - e_{n-1}$$

  - Relative improvement/degradation:

$$\Delta_{rel} a = \frac{a_n - a_{n-1}}{a_{n-1}}$$

$$\Delta_{rel} e = \frac{e_n - e_{n-1}}{e_{n-1}}$$

- Essential difference between absolute and relative improvement/degradation

Quality Measures

- **Example:** Test of a new medication
  - Baseline: 4 death in 1000 applications
  - New approach: 3 death in 1000 applications
  - relative improvement/degradation (strong):

$$\Delta_{rel} e = \frac{e_n - e_{n-1}}{e_{n-1}} = \frac{\frac{3}{1000} - \frac{4}{1000}}{\frac{4}{1000}} = -0.25$$

  - absolute improvement/degradation (moderate):

$$\Delta_{abs} e = e_n - e_{n-1} = \frac{3}{1000} - \frac{4}{1000} = -0.001$$

  - However, the single death less can even be pure coincidence
- **Often used manipulation:**
  - Relative improvement for positive effects
  - Absolute improvement for negative (adverse) effects

Quality Measures

- **Special case:** 2 classes (true/false)
  - True positives: $TP = \{o \mid c(o) = \text{true} = c_t(o)\}$
  - True negatives: $TN = \{o \mid c(o) = \text{false} = c_t(o)\}$
  - False positives: $FP = \{o \mid c(o) = \text{true} \neq c_t(o)\}$
  - False negatives: $FN = \{o \mid c(o) = \text{false} \neq c_t(o)\}$

$c(o)$ \ $c_t(o)$	***true***	***false***
true	***True Positives***	***False Positives***
false	***False Negatives***	***True Negatives***

Quality Measures

- **False positive rate (fall-out):**

$$FPR = \frac{|FP|}{|TN| + |FP|}$$

- **False negative rate (miss rate):**

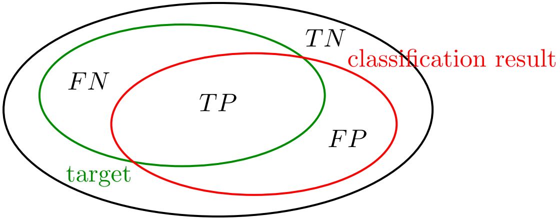
$$FNR = \frac{|FN|}{|TP| + |FN|}$$

- **True positive rate (sensitivity):**

$$TPR = \frac{|TP|}{|TP| + |FN|}$$

- **True negative rate (specificity):**

$$TNR = \frac{|TN|}{|TN| + |FP|}$$



Quality Measures

- **General case:** k classes
- $k^2 - k$ error types
- Description of the error type distribution as a confusion matrix
- **Biased error consequences:** Weighted error measures
  - Error types e_{ij} are associated with costs c_{ij}

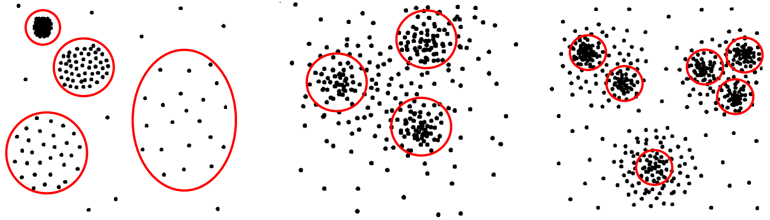
$$e_w = \frac{\sum_{i=1}^{k} \sum_{j=1}^{k} e_{ij} \cdot c_{ij}}{|S|}$$

Quality Measures

- **Further important quality aspects:**
  - Compactness, e.g. size of the decision tree (size determines classification time)
  - Interpretability (how many insights can be imparted by the classifier to the user?)
  - Efficiency of the construction process and application of the constructed classifier
  - Scalability with respect to large data sets (construction as well as application)
  - Robustness against noise and missing values

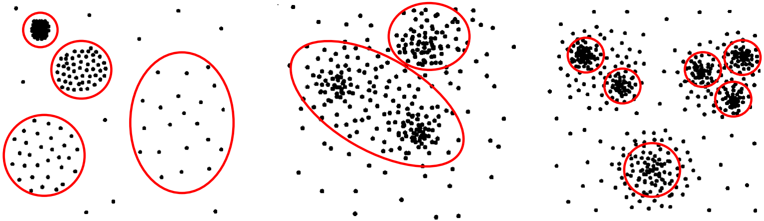
Clustering (or Cluster Analysis)

- **Goal:** Automatic identification of a finite set of categories, classes, or groups (clusters) in the given data
- **Procedure:** Grouping of data points according to their inherent structure
  - Points within the same cluster should be as similar as possible
  - Points in different clusters should be as dissimilar as possible
  - Learning without teacher
- **Clustering approaches:** partitioning, hierarchical, incremental, ...
- **Problem:** What is the optimal clustering? What is the meaning of "optimal"?



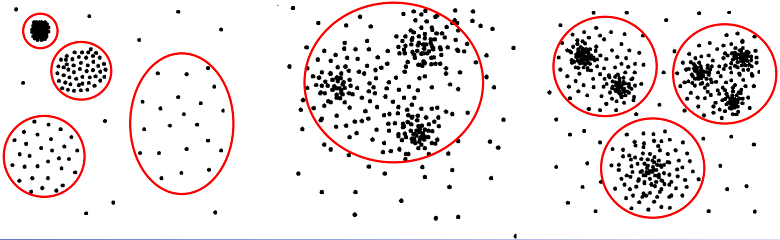
Clustering (or Cluster Analysis)

- **Goal:** Automatic identification of a finite set of categories, classes, or groups (clusters) in the given data
- **Procedure:** Grouping of data points according to their inherent structure
  - Points within the same cluster should be as similar as possible
  - Points in different clusters should be as dissimilar as possible
  - Learning without teacher
- **Clustering approaches:** partitioning, hierarchical, incremental, ...
- **Problem:** What is the optimal clustering? What is the meaning of "optimal"?



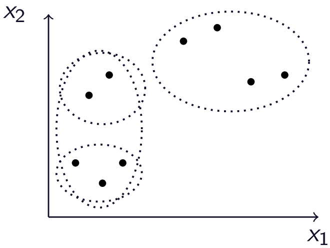
Clustering (or Cluster Analysis)

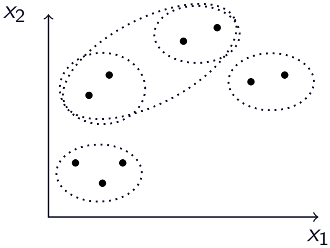
- **Goal:** Automatic identification of a finite set of categories, classes, or groups (clusters) in the given data
- **Procedure:** Grouping of data points according to their inherent structure
  - Points within the same cluster should be as similar as possible
  - Points in different clusters should be as dissimilar as possible
  - Learning without teacher
- **Clustering approaches:** partitioning, hierarchical, incremental, ...
- **Problem:** What is the optimal clustering? What is the meaning of “optimal”?



Clustering (or Cluster Analysis)

- **Definition of the optimal clustering depends on**
  - Application domain
    (i.e. structure/semantics of the data, meaning of distance)
  - Data mining target (i.e. object correlations that should be detected)
- **Computing the optimal clustering is computationally infeasible**
  $\Rightarrow$ greedy, sub-optimal approaches
- **Different clustering algorithms might lead to different clustering results**





K-Means Algorithm

- **Starting Situation:**
  - Distance measure
  - Method that can be used to compute the centroid ("mean") of a cluster
  - Number of clusters k
- **Basic Algorithm:**
  1. (Initialization): k cluster centroids are chosen (randomly)
  2. (Assignment): Each object is assigned to the cluster with the nearest centroid
  3. (Cluster Centroids): For each cluster a new centroid is computed
  4. (Repeat): Stop if the clustering stabilizes (i.e. the assignment does not change) or another termination criterion (e.g. number of iterations) is met, otherwise restart Step 2
- **Problems:**
  - Converges to local minima, i.e. the clustering does not have to be optimal
  - Workaround: Start the algorithm several times
  - Relatively high effort to compute distances and cluster centroids

K-Means Algorithm: Example 1

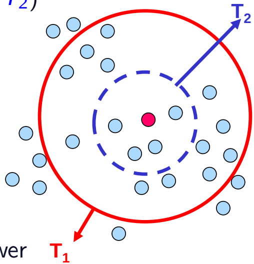
- **Clustering of the numbers $1, 3, 6, 14, 17, 24, 26, 31$ into 3 clusters**
  - (1) Centroids (chosen randomly): $10, 21, 29$
  - (2) Cluster: $10 : \{1, 3, 6, 14\}, 21 : \{17, 24\}, 29 : \{26, 31\}$
  - (3) Centroids (arithmetic mean): $6, 21, 29$
  - (2) Cluster: $6 : \{1, 3, 6\}, 21 : \{14, 17, 24\}, 29 : \{26, 31\}$
  - (3) Centroids (arithmetic mean): $3, 18, 29$
  - (2) Cluster: $3 : \{1, 3, 6\}, 18 : \{14, 17\}, 29 : \{24, 26, 31\}$
  - (3) Centroids (arithmetic mean): $3, 16, 27$
  - (2) Cluster: $3 : \{1, 3, 6\}, 16 : \{14, 17\}, 27 : \{24, 26, 31\}$
  - Stop because clustering did not change.

K-Means Algorithm: Example 2

- **Clustering of the numbers $1, 3, 6, 14, 17, 24, 26, 31$ into 3 clusters**
  - (1) Centroids (chosen randomly): $2, 5, 8$
  - (2) Cluster: $2 : \{1, 3\}, 5 : \{6\}, 8 : \{14, 17, 24, 26, 31\}$
  - (3) Centroids (arithmetic mean): $2, 6, 22$
  - (2) Cluster: $2 : \{1, 3\}, 6 : \{6\}, 22 : \{14, 17, 24, 26, 31\}$
  - Stop because clustering did not change.

Canopy Clustering Algorithm (Non-Partitioning Clustering)

- **Construction of overlapping clusters (canopies)**
  - Often used as a first step in a multi-step analysis approach
  - Scalable for large data sets
  - Can be used for string data
- **Given:** Distance function, two thresholds T_1 and T_2 ($T_1 > T_2$)
- **Algorithm:**
  1. (Initialization): candidate list for canopy centroids is initialized with all objects
  2. (Canopy Centroid): Canopy centroid Z is selected (randomly) from the candidate list
  3. (Assignment): All objects whose distance to Z is lower than threshold T_1 are assigned to canopy Z
  4. (Candidate List): All objects whose distance to Z is lower than threshold T_2, are removed from the candidate list
  5. (End/Repeat): Stop if candidate list is empty, otherwise restart of Step 2

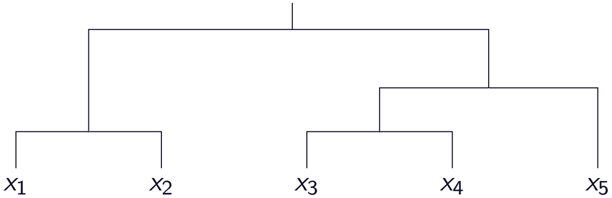


Canopy Clustering Algorithm: Example

- **Clustering of the numbers $1, 3, 6, 11, 13$**
- **Given: Distance function = absolute difference, $T_1 = 8$, $T_2 = 4$**
  - (1) Candidate list $= \{1, 3, 6, 11, 13\}$
  - (2) Canopy centroid $= 11$ (randomly)
  - (3) Constructed Canopy $\{6, 11, 13\}$
  - (4) Remove $\{11, 13\}$ from candidate list
  - (5) Candidate list $= \{1, 3, 6\}$
  - (2) Canopy centroid $= 1$ (randomly)
  - (3) Constructed Canopy $\{1, 3, 6\}$
  - (4) Remove $\{1, 3\}$ from candidate list
  - (5) Candidate list $= \{6\}$
  - (2) Canopy centroid $= 6$
  - (3) Constructed Canopy $\{1, 3, 6, 11, 13\}$
  - (4) Remove $\{3, 6\}$ from candidate list
  - (5) Stop because candidate list is empty

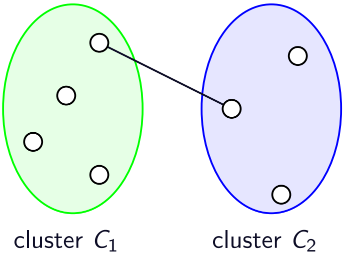
Agglomerative/Hierarchical Clustering

- **Successively merging data sets**
- **Algorithm:**
  - Initially each cluster consists of a single data point
  - Determine all inter-cluster distances
  - Merge the least distant clusters into a new one
  - Continue until the shortest cluster distance exceeds a predefined threshold or all clusters have been merged
- **Result can be displayed as a dendrogram**



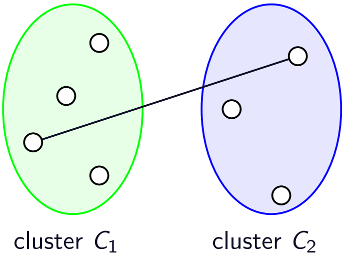
Distance Measures for Clusters

- **Single link:** minimal distance between two data points



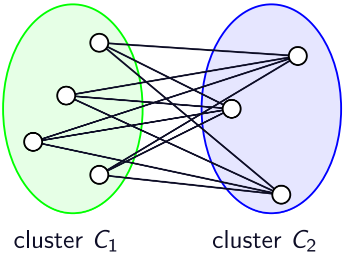
Distance Measures for Clusters

- **Single link:** minimal distance between two data points
- **Complete link:** maximal distance between two data points



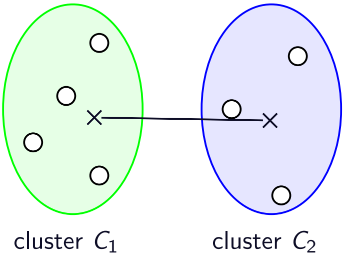
Distance Measures for Clusters

- **Single link:** minimal distance between two data points
- **Complete link:** maximal distance between two data points
- **Average link:** average distance between two data points



Distance Measures for Clusters

- **Single link:** minimal distance between two data points
- **Complete link:** maximal distance between two data points
- **Average link:** average distance between two data points
- **Canonical link:** distance between two cluster representatives (e.g. the centroids)



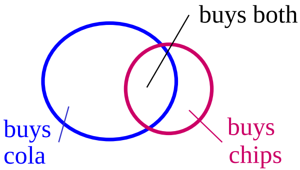
Incremental Clustering

- **Huge data sets cannot be clustered in a single step**
  - Divide-and-conquer: cluster subsets and merge the results
  - Incremental clustering: data points are loaded successively and the cluster representation is updated accordingly
- **Algorithm:**
  1. Assign the first data point to the first cluster
  2. Consider the next data point
     - Assign it to an already existing cluster, or
     - Create a new cluster
  3. Recompute the cluster description for that cluster
  4. Continue until all data points are clustered
- **Cluster description consists at least of**
  - Centroid
  - Number of data points in the cluster
  - "Radius" of the cluster (e.g. the maximal distance of a point to the centroid)
- **Problem:** Result depends on the order in which data points are processed

Dependency Mining

- **Goal:** Prediction of events commonly occurring together
- **Market basket analysis:** Which items are often purchased together
  - Placement of items in a store
  - Layout of mail-order catalogues
  - Targeted marketing campaigns
- **Association rules:** Rules of the form

  $$a \wedge b \wedge \ldots \wedge c \rightarrow d \wedge e$$



- **Example:** buys cola $\rightarrow$ buys chips
- **Challenge:** Finding good combinations of premises and conclusions is a combinatorial problem

Association Rules

- **Example (transaction) database:**

transaction	item
001	cola
001	chips
001	peanuts
002	beer
002	chips
002	cigarettes
...	...

transaction	items
001	{chips, cola, peanuts}
002	{beer, chips, cigarettes}
003	{beer, chips, cigarettes, cola}
004	{beer, cigarettes}

Association Rules

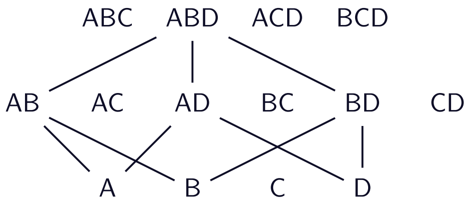
- **Set of n different items:** $I = \{x_1, \ldots, x_n\}$
- **Itemset:** $I_k \subseteq I$
- **i-itemset:** $I_k^i \subseteq I, \ \ |I_k^i| = i$
- **Transaction:** $T_k \subseteq I$
- **Database:** $D = \{(k, T_k) \mid k = 1, \ldots, m\}$
- **Support of an itemset:** share of transactions which contain the itemset

$$s(I_i) = \frac{|\{T_k \mid I_i \subseteq T_k\}|}{|D|}$$

- **Frequent (strong, large) itemset:** $s(I_i) \geq s_{cut}$
- **Minimal support:** s_{cut} is application-specific, i.e. depends on total number of items, size of database, etc.

Association Rules

- **Downward closure:**
  Every subset of a frequent itemset is also a frequent itemset



- Every superset of a non-frequent itemset is also a non-frequent itemset

Association Rules

- **Association rule:** $X \rightarrow Y$ where $X, Y \subseteq I, X \cap Y = \emptyset$
- **Support of a rule:** Share of transactions which contain both, premise and conclusion of the rule

$$s(X \rightarrow Y) = s(X \cup Y) = \frac{|\{T_k | X \cup Y \subseteq T_k\}|}{|D|} = p(XY)$$

- **Confidence of a rule:** Share of transactions supporting the rule from those supporting the premise

$$c(X \rightarrow Y) = \frac{s(X \cup Y)}{s(X)} = \frac{|\{T_k | X \cup Y \subseteq T_k\}|}{|\{T_k | X \subseteq T_k\}|} = p(Y|X)$$

- **Lift of a rule:** Ratio of the observed support to that expected if X and Y were independent

$$lift(X \rightarrow Y) = \frac{s(X \cup Y)}{s(X) \times s(Y)} = \frac{p(Y|X)}{p(Y)}$$

Association Rules

- **Strong rule:** high support + high confidence
- **Detection of strong rules:** Two pass algorithm

1. Find frequent (strong, large) itemsets (Apriori algorithm)
  - Necessary to generate rules with strong support
  - Uses the downward closure
  - Itemsets are ordered

2. Use the frequent itemsets to generate association rules
  - Find strong correlations in a frequent itemset

Association Rules

Apriori: Finding frequent itemsets of increasing size (itemsets are ordered!)

1. Start with all itemsets of size one: I^1
2. Select all itemsets with sufficient support
3. From the selected itemsets I^i generate larger itemsets I^{i+1}

$$\{i_1, \ldots, i_{n-2}, i_{n-1}\} \oplus \{i_1, \ldots, i_{n-2}, i_n\} \rightarrow \{i_1, \ldots, i_{n-2}, i_{n-1}, i_n\}$$

   - Already blocks some of the non-frequent itemsets, but not all of them
4. Remove those itemsets which still contain a non-frequent subset
   - They cannot have enough support (downward closure)
5. Continue with Step 2 until no further frequent itemsets can be generated

Association Rules

- **Example database again**
- **Assumption:** Threshold for the required support $s_{cut} = 0.5$

k	T_k
001	{chips, cola, peanuts}
002	{beer, chips, cigarettes}
003	{beer, chips, cigarettes, cola}
004	{beer, cigarettes}

I_k^1	#	$s(I_k^1)$
{chips}	3	0.75
{cola}	2	0.5
{peanuts}	1	0.25
{beer}	3	0.75
{cigarettes}	3	0.75

- **No non-empty subsets**

Association Rules

- **2-itemsets: I_k^2**

I_k^1	$\#$	$s(I_k^1)$
{chips}	3	0.75
{cola}	2	0.5
{beer}	3	0.75
{cigarettes}	3	0.75

I_k^2	$\#$	$s(I_k^2)$
{chips, cola}	2	0.5
{beer, chips}	2	0.5
{chips, cigarettes}	2	0.5
{beer, cola}	1	0.25
{cigarettes, cola}	1	0.25
{beer, cigarettes}	3	0.75

- **No itemsets to prune**

Association Rules

- **3-itemsets:** I_k^3

I_k^2	#	$s(I_k^2)$
{chips, cola}	2	0.5
{beer, chips}	2	0.5
{chips, cigar.}	2	0.5
{beer, cigar.}	3	0.75

I_k^3	#	$s(I_k^2)$
{beer, chips, cigar.}	2	0.5
{chips, cigar., cola}	1	0.25

- Note: Itemset $\{beer, chips, cola\}$ is not constructed because we only combine sets of I_k^2 that have the first element in common
- However, this set cannot be frequent because otherwise the set $\{beer, cola\}$ would be in I_k^2 (and then we would have constructed $\{beer, chips, cola\}$)
- Itemset $\{chips, cigar., cola\}$ can be pruned because $\{cigar., cola\}$ is not frequent

Association Rules

- **Resulting frequent itemsets:**

  {beer, chips, cigarettes}
  {chips, cola}
  {beer, chips}
  {chips, cigarettes}
  {beer, cigarettes}
  {beer}
  {chips}
  {cigarettes}
  {cola}

Association Rules

- **Generation of strong association rules:**
  - For all frequent itemsets I_j with $|I_j| \geq 2$ determine all non-empty subsets I_k for which

$$c = \frac{s(I_j)}{s(I_k)} \geq c_{min}$$

  - Add rule $I_k \rightarrow Y$ with $Y = I_j - I_k$ to the rule set

- e.g. $s(\{chips\}) = 0.75, s(\{cola\}) = 0.5$, $s(\{chips, cola\}) = 0.5$

rule	confidence
{cola} → {chips}	1.00
{chips} → {cola}	0.67

Association Rules

- **Interesting association rules:** Only those for which the confidence is greater than the support of the conclusion

$$c(X \rightarrow Y) > s(Y) \qquad (\equiv \mathit{lift}(X \rightarrow Y) > 1)$$

- **Rule modification:** Confidence can be increased by shifting items from the conclusion to the premise
- **Negative border:**

$$\{I_j \mid s(I_j) < s_{cut} \wedge \forall I_k \subset I_j : s(I_k) \geq s_{cut}\}$$

used to derive negative association rules (e.g. customers who buy cola and chips do not buy beer)

Association Rules

- **Apriori:** Number of potential itemsets is exponential in the number of items
- **But:**
  - Data is sparse: $|T_i| \ll |I|$
  - Itemsets are generated in separate scans of the database
  - Size of generated itemsets grows monotonically
  - Large itemsets are usually useless
  - Only k scans required ($k \ll |I|$)

Association Rules

- **Modifications / Extensions:**
  - Rule mining on relational data
  - Apriori for hierarchically organized items
  - Partitioned Apriori
  - Sampled transactions
  - Incremental rule mining
  - Non uniform support thresholds
  - Class association rules

Association Rules with Relational Data

- Relational data has to be transformed into transaction data
- Apriori requires categorical data $\Rightarrow$ binning has to be performed
- The same category can appear as value of different attributes

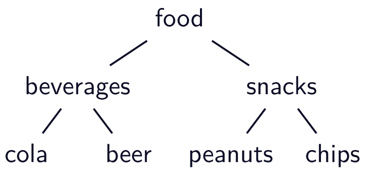
age	income	debt
low	low	low
middle	low	high
high	high	low

- Values have to be combined with their attribute
  - Attribute-value pairs are taken as items

1	(age, low)	(income, low)	(debt, low)
2	(age, middle)	(income, low)	(debt, high)
3	(age, high)	(income, high)	(debt, low)

Hierarchical Apriori

- In addition to the base level of items, determine also frequent itemsets on a higher level in an is-a hierarchy



- Sometimes regularities can only be found at higher levels of abstraction

Partitioned Apriori

Computation in two steps:

1st step:

- Partition the database
- Compute locally frequent itemsets on each partition

2nd step:

- Determine the global support of all locally frequent itemsets
- Remove all itemsets which do not satisfy the minimal support

Heuristics:

- Idea: if an itemset is globally frequent it will be so locally in at least one partition

$\Rightarrow$ Second step deals with a superset of possible frequent itemsets