# Database and Information Systems





# Data Mining

- Data-Mining: The Task
- Data-Mining as a Process
- Data Preprocessing
- Data Mining Tasks



# Data Mining Tasks

- Classification
- Prediction
- Clustering
- Summarization
- Dependency Modelling
- Change and Deviation Detection



# Classification

- well understood
  - decision theory
  - many heuristic solutions
- applications in
  - customer relationship management: tailored marketing
  - banking: credit authorization
  - document management: e-mail routing



# Classification

- given
  - a data base $D = \{t_1, t_2, ..., t_n\}$
  - of tupels $t_i = \vec{x}$ and
  - a set of classes $C = \{c_1, c_2, ..., c_m\}$,
- find a mapping $f : D \to C$
  - such that $f$ partitions $D$.



# Classification

- classes are predefined:
  supervised learning, learning with a teacher
- notation
  - $c_t(t_i)$: class assignment in the training data
  - $c(t_i)$: class assignment by the classifier
- usually: $|D| \gg |C|$
- class is a set of tuples: $c_j = \{t_i | f(t_i) = c_j\}$
- no tuple belongs to several classes



# Classification

- Nearest-Neighbor Classifier
- Threshold-based Classifiers
- Decision Trees
- Neural Networks
- Stochastic Classification
- Evaluation

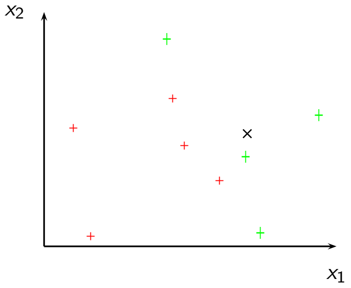


# Nearest-Neighbor Classifier

- direct approach: training data $T = \{(\vec{x_i}, k)\}$ are
  - directly stored in the classifier and
  - used for classification
- nearest neighbor

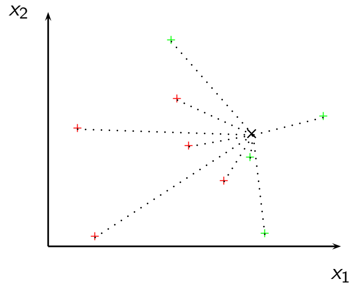
$$c(\vec{x}) = \arg_{c_k}(\vec{x_j}, k), \quad j = \arg\min_i d(\vec{x}, \vec{x_i})$$

## Nearest-Neighbor Classifier



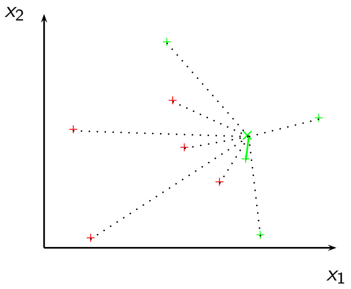



## Nearest-Neighbor Classifier






## Nearest-Neighbor Classifier






## Nearest-Neighbor Classifier

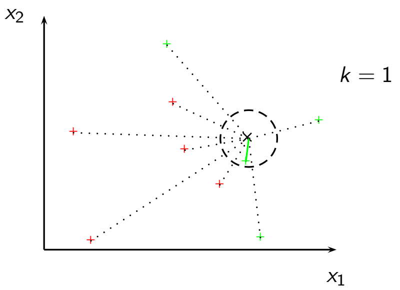
- $k$-nearest neighbors
  - determine the set $N$ of the $k$ nearest neighbors of $\vec{x}$ in $T$
  - choose the class with the maximum number of data points in $N$

$$c(\vec{x}) = \arg\max_{c_k} |\{c_k | c_k \in N\}|$$

  - more robust against singular data points
  - but more expensive

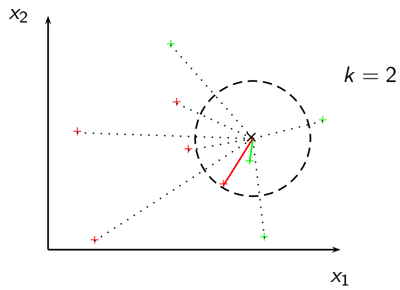


## Nearest-Neighbor Classifier



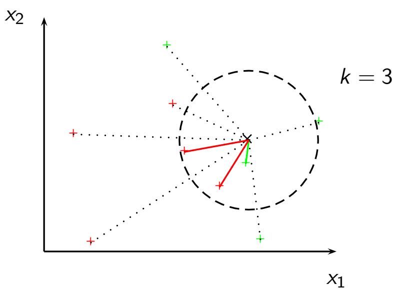



## Nearest-Neighbor Classifier



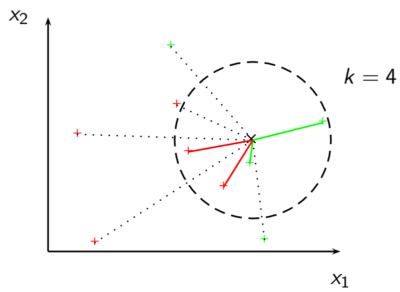



## Nearest-Neighbor Classifier

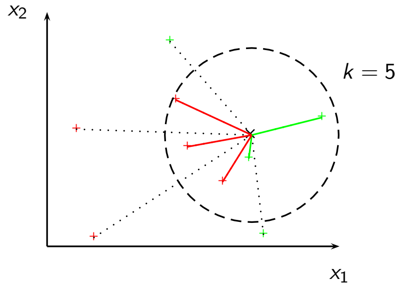





## Nearest-Neighbor Classifier



## Nearest-Neighbor Classifier






## Nearest-Neighbor Classifier

- NN-classifier is instance-based
  - model size and classification effort grow linearly with amount of training data
  - no generalization of the available training data
- generalizing models required
  - use class representatives as data points
  - e.g. mean of class or class-dependent clusters



## Classification

- Nearest-Neighbor Classifier
- Threshold-Based Classifiers
- Decision Trees
- Neural Networks
- Stochastic Classification
- Evaluation



## Threshold-Based Classifiers

- simple generalizing model
- a threshold divides the data space into two subspaces

$$c(\vec{x_i}) = \begin{cases} 1 & x_j > \theta_j \\ 2 & else \end{cases}$$

- analogue separation criteria for non-numeric data



## Threshold-Based Classifiers

- choice of the optimal threshold:
  - minimizing the classification error on the training data

$$\theta = \arg\min_\theta |\{t_i | c(t_i) \neq c_t(t_i)\}|$$

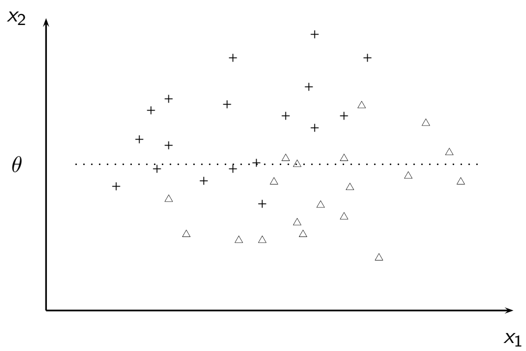
  - for numeric data approximated by minimizing the distance of misclassified samples to the threshold

$$\theta = \arg\min_\theta \sum_{t_i, c(t_i) \neq c_t(t_i)} |x_j - \theta_k|$$



## Threshold-Based Classifiers

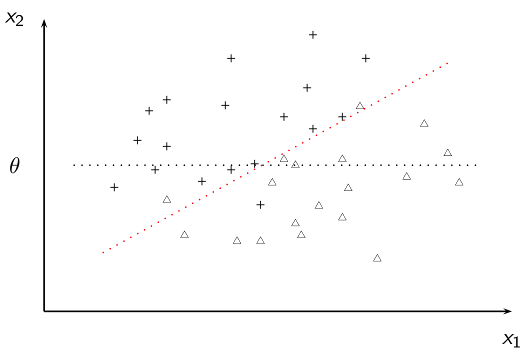
- insufficient to separate more difficult distributions






## Threshold-Based Classifiers

- better class separation






## Threshold-Based Classifiers

- algorithm for finding an optimal threshold
  1. sort the values $[v_1, ..., v_m]$
  2. extract $m - 1$ potential thresholds by either
     - computing the mean of all neighboring values or
     - choosing the smaller one of two neighboring values
  3. evaluate all potential thresholds and select the one with the maximum gain

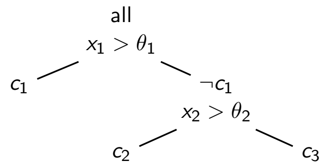
## Classification

- Nearest-Neighbor Classifier
- Threshold-Based Classifiers
- Decision Trees
- Neural Networks
- Stochastic Classification
- Evaluation



## Decision trees

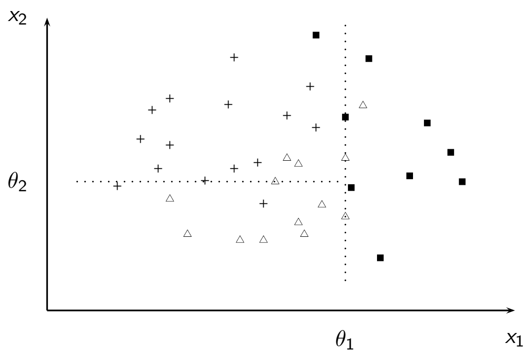
- extension of threshold-based classifiers to multiple classes:
  - multi-branch splits
  - decomposition into a sequence of sub-decisions




- finding the optimal decision tree is NP complete
  $\rightarrow$ deterministic (non-backtracking), greedy algorithms



## Decision trees






## Decision trees

- ID3: split along a dimension as to maximise information gain
- entropy of a set $S$ partitioned into $k$ classes

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

- 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

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



## Decision trees

- C4.5: extension of ID3 to numerical data
  - split along a dimension so that the resulting subsets have lowest class entropy
    i.e. contain data points of as few classes as possible
- problem of overfitting
  - splitting until no data point is misclassified usually means to adapt the classifier too much to the training data
    - "learning off by heart"
    - degrading performance on held out test data
  - cut-off criterion required, or post-pruning



## Decision trees

- decision rules can be extracted from a decision tree
  - IF part: combine all tests on the path from the root node to the leave node
  - THEN part: the final classification



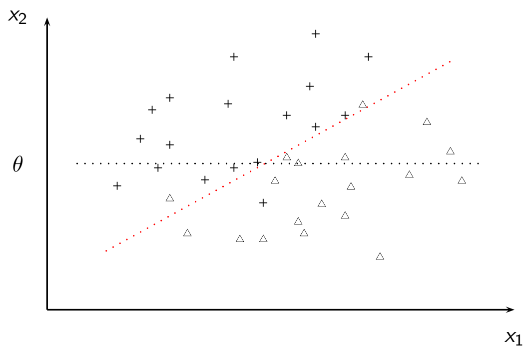
## Classification

- Nearest-Neighbor Classifier
- Threshold-Based Classifiers
- Decision Trees
- Neural Networks
- Stochastic Classification
- Evaluation



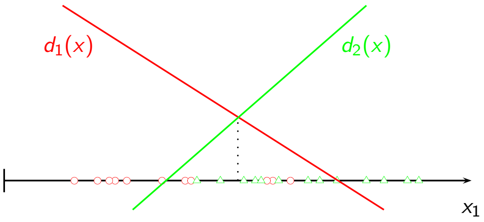
## Neural Networks

- sometimes linear functions can be used to separate two classes



## Neural Networks

- classes are represented by means of linear discrimination functions $d_k(\vec{x})$




$\longrightarrow$ linear discriminance analysis (LDA)



## Neural Networks

- class decision is reduced to a maximum detection

$$c(\vec{x}) = \arg\max_{c_k} d_k(\vec{x})$$

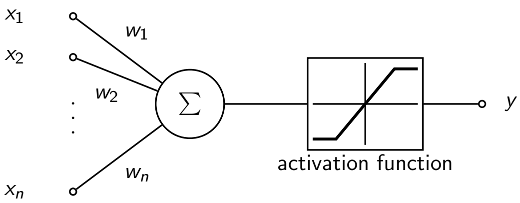
- discriminating functions are (in the simplest case) linear combinations of the components of a data point

$$d_k(\vec{x}) = w_0 + \sum w_i x_i$$



## Neural Networks

- corresponds to first part of a perceptron

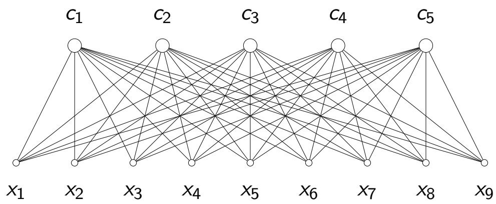



- single perceptron: classification only for
  - two class problems and
  - linear separable classes



## Neural Networks

- extension to multiple classes: single-layer networks

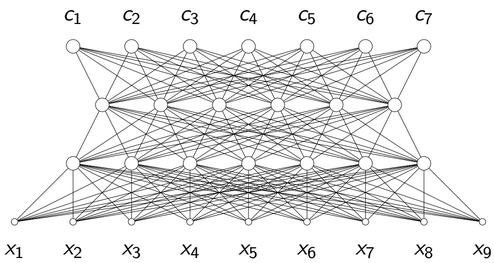



- class decision: maximum detection ("the winner takes all")



## Neural Networks

- multiple perceptrons simulate a piecewise-linear discrimination function
- single layer networks only for simple problems
  $\longrightarrow$ usually multiple-layer networks required





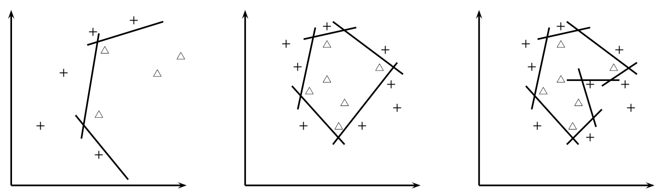

## Neural Networks

- optimal architecture has to be determined experimentally
- only few heuristic criteria available



## Neural Networks

- How many layers are necessary?



single-layer network — two-layer network — three-layer network

no islands — only convex islands — also concave islands



## Neural Networks

- How many nodes per layer are required?
  - the more nodes, the smoother the class separation
  - the more nodes, the more training data and training cycles are required

## Neural Networks

- training of neuronal networks
- error driven learning
  - assume an initial (random) assignment of synaptic weights $w_{ij}$
  - determine the error of the output value of a node $i$: $e_i = \frac{(y_i - d_i)^2}{2}$
  - change the weights $w_{ij}$ according to a learning rule
  - continue with the next training sample
- backpropagation of the error signal from the output layer to the input layer



## Neural Networks

- examples of learning rules
  - Hebb rule

    $$\Delta w_{ij} = \eta \; x_{ij} \; y_j$$

    $\eta$: learning rate (approx $1/|T|$)
    does not consider the desired output
  - delta rule

    $$\Delta w_{ij} = \eta \; x_{ij} \; (d_j - y_j)$$

  - learning rules with momentum

    $$\Delta w_{ij}(n) = \eta \; x_{ij} \; (d_j - y_j) + \alpha \; \Delta w_{ij}(n-1)$$



## Neural Networks

- gradient descent search: all weights are changed until no significant change of the global error measure can be observed
  - high number of training iterations is required
  - local search: optimum is not guaranteed
  - not even convergence of the algorithm is guaranteed



## Neural Networks

- problems
  - many parameters have to be determined empirically
    - number of layers, number of nodes per layer
    - learning rate, momentum
    - initialization
    - termination criterion
  - overfitting may occur
    - stop training early enough
    - choose the most simple architecture possible



## Neural Networks

- decision rules can be extracted from a neural network
  - cluster the node activations
  - generate rules from high synaptic links
  - combine the rules across layers



## Classification

- Nearest-Neighbor Classifier
- Threshold-Based Classifiers
- Decision Trees
- Neural Networks
- Stochastic Classification
- Evaluation



## Stochastic Classification

- Bayesian inference
  - given: a prior data distribution
  - observe data
  - infer a posterior distribution
- Bayes' theorem

  $$p(c_k|\vec{x}) = \frac{p(\vec{x}|c_k) \cdot p(c_k)}{p(\vec{x})}$$

  - $p(\vec{x})$ does not influence a class decision
  - $p(\vec{x}|c_k)$ and $p(c_k)$ have to be estimated using the available training data



## Stochastic Classification

- $p(c_k)$: class probability

  $$p(c_k) = \frac{|c_k|}{|S|}$$

- $p(\vec{x}|c_k)$: data generation (or emission) model
  - more difficult to estimate
  - simplifying assumption:
    conditional independence between attributes

  $\rightarrow$ naïve / simple Bayesian classifier

  $$p(\vec{x}|c_k) = \prod_{i=1}^{n} p(x_i|c_k)$$

- training method: maximum likelihood (ML) estimation

## Stochastic Classification

- classification rule

$$k = \arg\max_k p(c_k|\vec{x}) = \arg\max_k p(\vec{x}|c_k) \cdot p(c_k)$$

- Bayes classifier has optimal error rates
- but: in practice worse because of the independence assumption



## Stochastic Classification

- problems with non-trivial input output dependencies
  - strongly correlated variables
  - time series analysis
- Bayesian networks
- emission probabilities are conditioned on the state of the model
- state of the model is not directly observable → hidden variable
- simplifying assumption: state probabilities depend only on the preceding state
- ML training requires direct counting of observations
- alternative: expectation maximization (EM)
  - start with an initial probability estimation
  - modify the current probabilities as to better fit the training data
- resulting probabilities are only approximations



## Comparison of classifiers

| | Nearest Neighbor | Decision Trees | (Linear) Discriminance | Stochastic Models |
|---|---|---|---|---|
| model | sample set | ranking of tests | class boundaries | probabilistic generation |
| generalization | no | forced | yes | yes |
| robust against | | | | |
| incons. data | no | now | yes | yes |
| outliers | only k-NN | low | yes | yes |
| perspicuity | high | high | low | low |
| scalability | very low | low | good | very good |
| additional assumptions | metrics | no | architecture learning rule | (architecture) distribution |



## Classification

- Nearest-Neighbor Classifier
- Threshold-Based Classifiers
- Decision Trees
- Neural Networks
- Stochastic Classification
- Evaluation



## 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

  → evaluation only on separate data: test set
- but: available test set data is usually limited
  - 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 → repeat the measurement with different partitionings and average
- leave one out
  - use $n-1$ samples for training and evaluate on the $n$-th one
  - repeat with all $n$ samples
  - extremely expensive



## Resampling Methods

- n-fold cross validation
  - combines hold-out and leave-one-out
  - divide data set into $p$ partions
  - use $p-1$ partitions for training; evaluate on the remaning one
- bootstrapping
  - generate artificial training data by replacing data items
  - obtain bootstrap estimations of the error rates on these data sets
  - useful if few data are available



## Quality Measures

- error rate

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

  $S$: test set, $M \subseteq S$: misclassified data
- accuracy

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

- only for atomic data!

## Quality Measures

- contrastive analysis:
  - absolute improvement/degradation: comparison with a baseline case

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

  - relative improvement/degradation

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



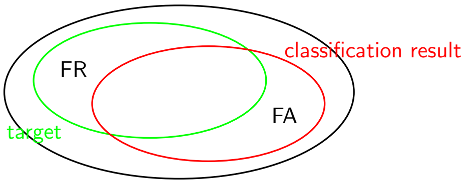
## Quality Measures

- special case: 2 classes (true/false) $\rightarrow$ 2 error cases
- false positives/acceptance: false acception rate (sensitivity)

$$FAR = \frac{|\{x|c(x) = \text{true} \neq c_t(x)\}|}{|\{x|c(x) = \text{true}\}|}$$

- false negatives/rejection: false rejection rate (specificity)

$$FRR = \frac{|\{x|c(x) = \text{false} \neq c_t(x)\}|}{|\{x|c(x) = \text{false}\}|}$$





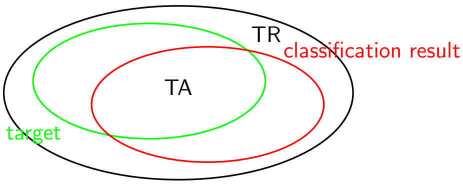

## Quality Measures

- true positives/acceptance: true acception rate

$$TAR = \frac{|\{x|c(x) = \text{true} = c_t(x)\}|}{|\{x|c(x) = \text{true}\}|}$$

- true negatives/rejection: true rejection rate

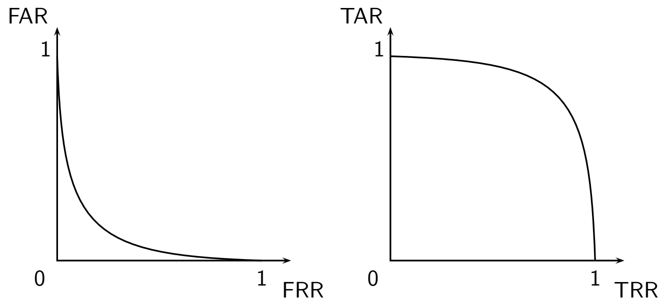
$$TRR = \frac{|\{x|c(x) = \text{false} = c_t(x)\}|}{|\{x|c(x) = \text{false}\}|}$$






## Quality Measures

- trade-off between FAR and FRR / TAR and TRR



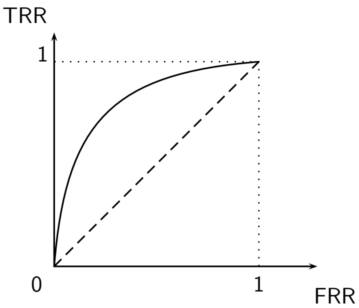

- trivial classifier: upper threshold for the error rate

$$e_{max} = \min(p(\text{true}), p(\text{false}))$$



## Quality Measures

- receiver operating characteristic (ROC): TRR vs. FRR




- quality: area under the ROC-curve



## Quality Measures

- in general $k^2 - k$ ($k$: number of classes) 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 $c_{ij}$

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