Classification and Prediction

Introduction

Evaluation of Classifiers

Decision Trees

Bayesian Classification

Nearest-Neighbor Classification

Support Vector Machines

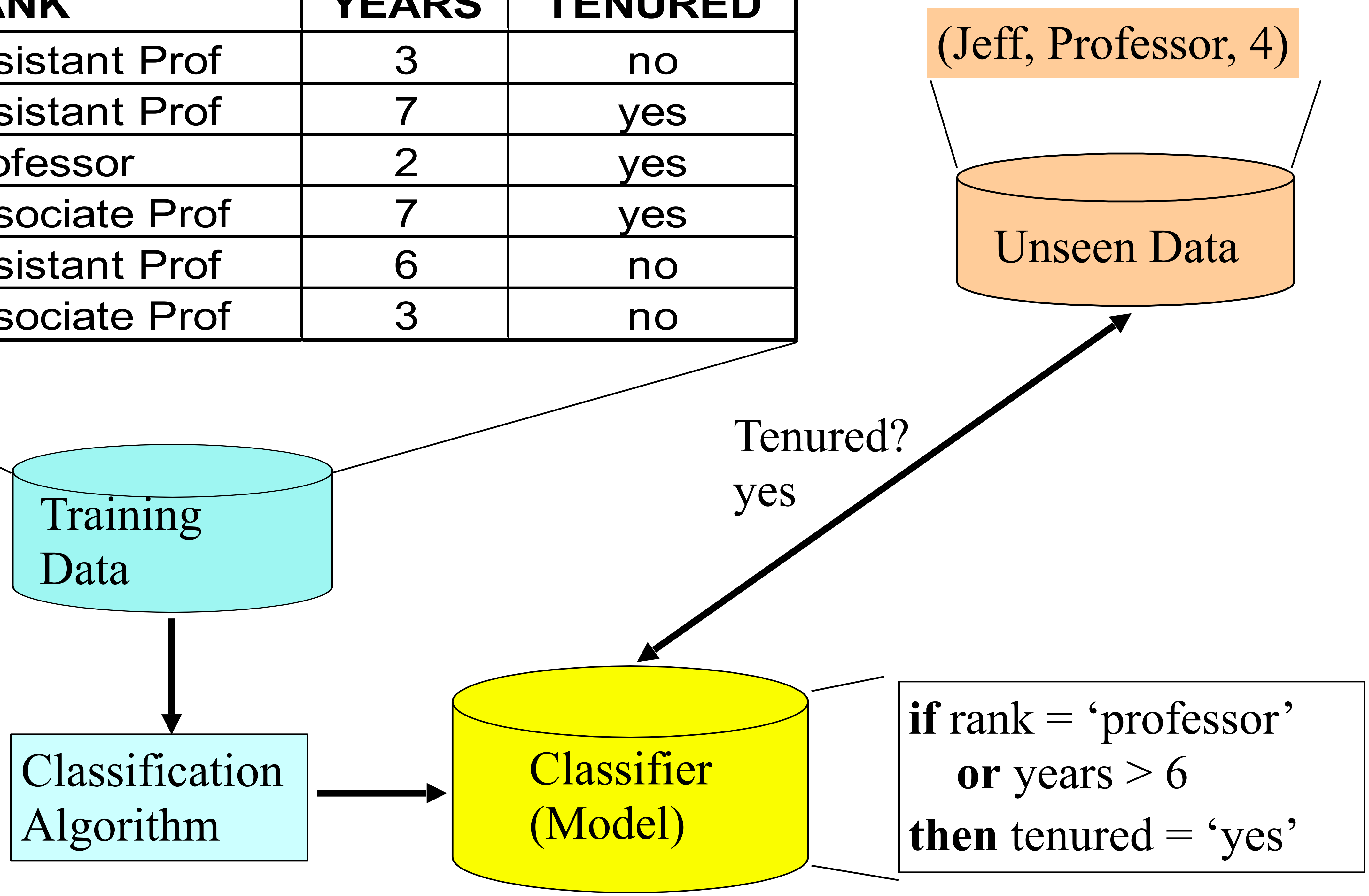
Multi-relational Classification

Regression Analysis

The Classification Problem

- Let O be a set of objects of the form $(o_1, \ldots, o_d)$ with *attributes* A_i, $1 \leq i \leq d$, and class membership c_i, $c_i \in C = \{c_1, \ldots, c_k\}$
- Wanted:
  - class membership for objects from $D \setminus O$
  - a *classifier* $K : D \rightarrow C$
- Difference to clustering
  - classification: set of classes C known apriori
  - clustering: classes are output
- Related problem: prediction
  - predict the value of a *numerical* attribute

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



Introduction

- Given a sample of labeled data (O)
- Want to build a classifier that labels the entire population

  in particular, $D \setminus O$
- Can only estimate the performance of the classifier on unseen data
- Need separate, disjoint training and test data (all labeled)
  - *Training data*

    for training the classifier (model construction)
  - *Test data*

    to evaluate the trained classifier

Approaches

- *Train-and-Test*
  - partition set *O* into two (disjoint) subsets: Training data and Test data
  - not recommended for small *O*
- *m*-fold cross validation
  - partition set *O* into *m* same size subsets
  - train *m* different classifiers using a different one of these *m* subsets as test data and the other subsets for training
  - average the evaluation results of the *m* classifiers
  - appropriate also for small *O*

Evaluation Criteria

- Classification accuracy
- Interpretability
    - e.g. size of a decision tree
    - insight gained by the user
- Efficiency
    - of model construction
    - of model application
- Scalability for large datasets
    - for secondary storage data
- Robustness
    - w.r.t. noise and unknown attribute values

Classification as optimization problem: score of a classifier

Classification Accuracy

- Let K be a classifier, $TR \subseteq O$ the training data, $TE \subseteq O$ the test data.
  $C(o)$: actual class of object o.
- *classification accuracy* of K on TE:

$$Accuracy_{TE}(K) = \frac{|\{o \in TE \mid K(o) = C(o)\}|}{|TE|}$$

- *classification error*

$$Error_{TE}(K) = \frac{|\{o \in TE \mid K(o) \neq C(o)\}|}{|TE|}$$

aggregates over all classes $c_i \in C$

not appropriate if minority class is most important

Confusion Matrix

- Let $c_1 \in C$ be the *target (positive) class*, the union of all other classes the *contrasting (negative) class*.
- Comparing the predicted and the actual class labels, we can distinguish four different cases:

	Predicted as positive	Predicted as negative
Actually positive	True Positive (TP)	False Negative (FN)
Actually negative	False Positive (FP)	True Negative (TN)

Confusion matrix

Precision and Recall

- We define the following two measures of K w.r.t. the given target class:

$$\text{Precision}(K) = \frac{|TP|}{|TP| + |FP|}$$

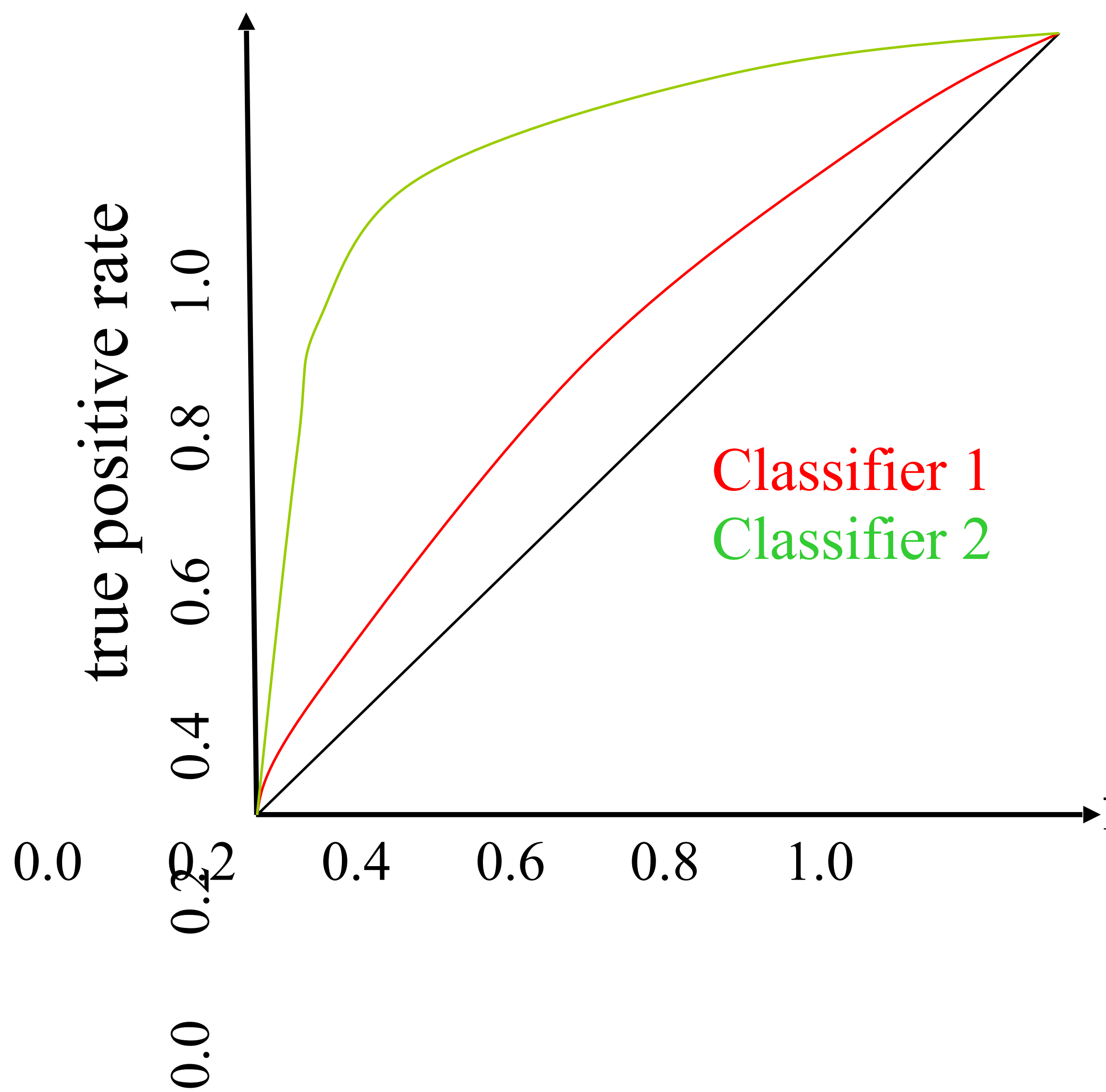
$$\text{Recall}(K) = \frac{|TP|}{|TP| + |FN|}$$

- There is a trade-off between precision and recall.
- Therefore, we also define a measure combining precision and recall:

$$\text{F} - \text{Measure}(K) = \frac{2 \cdot \text{Precision}(K) \cdot \text{Recall}(K)}{\text{Precision}(K) + \text{Recall}(K)}$$

ROC Curves

- F-Measure captures only one of the possible trade-offs between precision and recall (or between TP and FP)
- *True positive rate*: percentage of positive data correctly predicted
- *False positive rate*: percentage of negative data falsely predicted as positive



- *ROC* (Receiver Operating Characteristic) *Curve*: plots true positive rate w.r.t. false positive rate
- *area under ROC* as quantitative measure

  ideally = 1

Model Selection

- Given two classifiers and their (estimated!) classification accuracies

  e.g., obtained from m-fold cross-validation

- Which of the classifiers is really better?
- Naive approach: just take the one with higher mean classification accuracy
- But: classification accuracy may vary greatly among the m folds
- Differences in classification accuracies may be insignificant
  due only to chance

Model Selection

- We measure the classification error on a (small) test dataset $O \subseteq X$.
- Questions:
  How to estimate the *true classification error* on the whole instance space X?
  How does the deviation from the observed classification error depend on the size of the test set?
- Random experiment to determine the classification error on test set (of size n):
  repeat n times
  (1) draw random object from X
  (2) compare predicted vs. actual class label for this object
- Classification error is percentage of misclassified objects
  → observed classification error follows a Binomial distribution
  with mean = true classification error (unknown)

Binomial distribution

- n repeated tosses of a coin with unknown probability p of head
  head = misclassified object
- Record the number r of heads (misclassifications)
- Binomial distribution defines probability for all possible values of r:

$$P(r) = \frac{n!}{r!(n-r)!} p^r (1-p)^{n-r}$$

- Random variable Y counting the number of heads in n coin tosses:

$$E[Y] = n \cdot p \quad expected\ value$$

$$Var[Y] = np(1-p)$$

$$\sigma_Y = \sqrt{np(1-p)}$$

Estimating the True Classification Error

- We want to estimate the unknown true classification error (p).
- Estimator for p: $$E[Y] = n \cdot p = r \quad \Rightarrow \quad p = \frac{r}{n}$$
- We want also confidence intervals for our estimate.
- Standard deviation for the true classification error (Y/n):

$$\sigma_{\frac{Y}{n}} = \frac{\sigma_Y}{n} = \frac{\sqrt{np(1-p)}}{n}$$

$$\sigma_{\frac{Y}{n}} \approx \sqrt{\frac{\frac{r}{n}(1-\frac{r}{n})}{n}} \qquad \text{use } \frac{r}{n} \text{ as estimator for } p$$

Estimating the True Classification Error

- For sufficiently large values of *n*, the Binomial distribution can be approximated by a Normal distribution with the same mean and standard deviation.
- Random variable *Y* Normal distributed with mean *m* and standard deviation *s* and *y* be the observed value of *Y*:

  the mean of *Y* falls into the following interval with a probability of *N* %

$$y \pm z_N \sigma$$

- In our context, *N* % confidence interval for the true classification error:

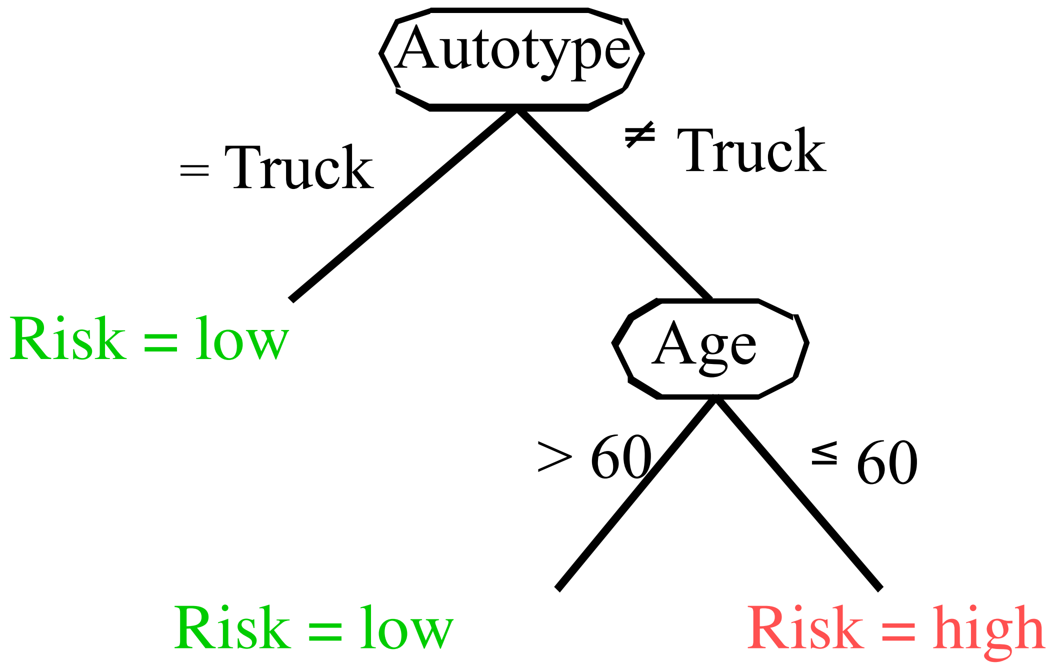
$$\frac{r}{n} \pm z_N \sqrt{\frac{\frac{r}{n}(1-\frac{r}{n})}{n}}$$

interval size decreases with increasing *n*

interval size increases with increasing *N* and z_N

Introduction

ID	Age	Autotype	Risk
1	23	Family	high
2	17	Sports	high
3	43	Sports	high
4	68	Family	low
5	32	Truck	low



disjunction of conjunction of attribute constraints
and hierarchical structure

Introduction

- *A decision tree* is a tree with the following properties:
  - An inner node represents an attribute.
  - An edge represents a test on the attribute of the father node.
  - A leaf represents one of the classes of *C*.
- Construction of a decision tree

  Based on the training data

  Top-Down strategy
- Application of a decision tree

  Traversal of the decision tree from the root to one of the leaves

  Unique path

  Assignment of the object to class of the resulting leaf

Construction of Decision Trees

Base algorithm

- Initially, all training data records belong to the root.
- Next attribute is selected and split (split strategy).
- Training data records are partitioned according to the chosen split.
- Method is applied recursively to each partition.

→ local optimization method (greedy)

Termination conditions

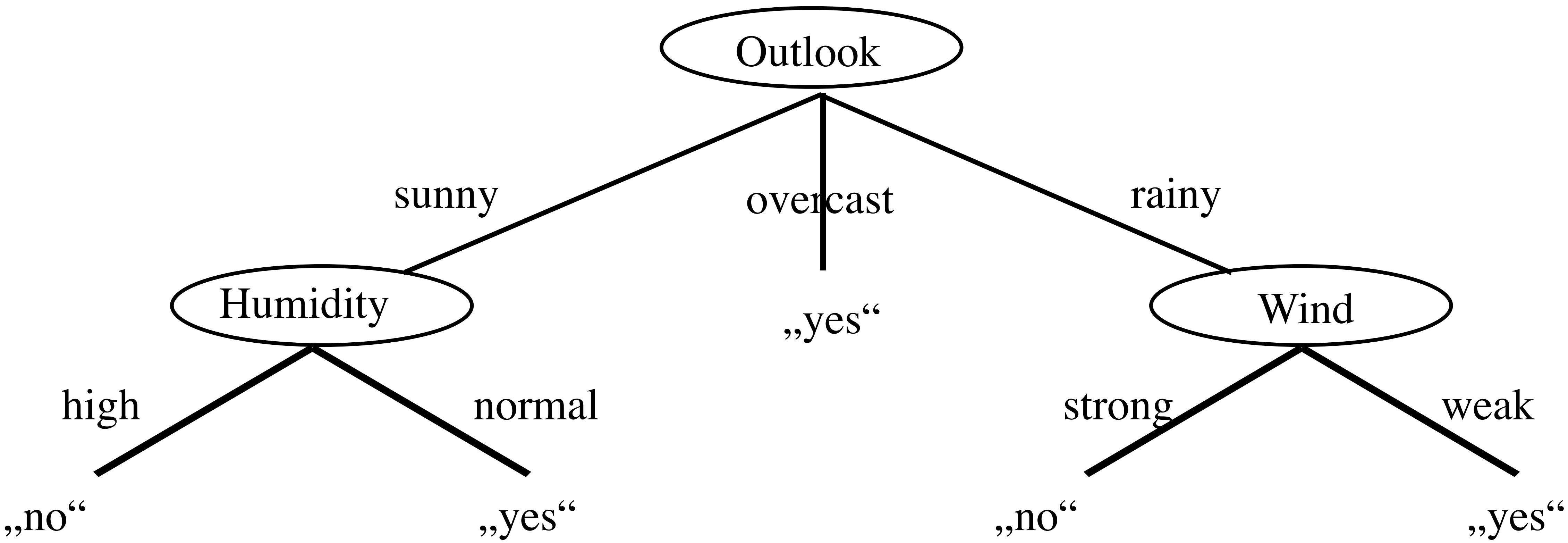
- No more split attributes.
- All (most) training data records of the node belong to the same class.

Example

Day	Outlook	Temperature	Humidity	Wind	PlayTennis?
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rainy	mild	high	weak	yes
5	rainy	cool	normal	weak	yes
6	rainy	cool	normal	strong	no
7	. . .	. . .	. . .	. . .	. . .

Is today a day to play tennis?

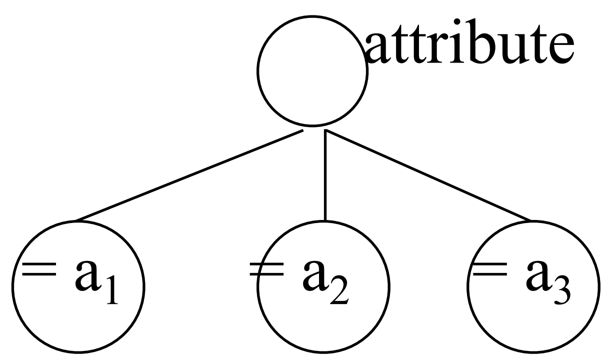
Example

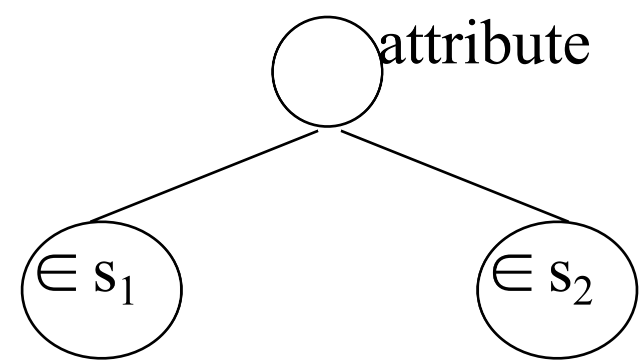

Outlook
sunny
overcast
rainy
Humidity
„yes“
Wind
high
normal
strong
weak
„no“
„yes“
„no“
„yes“

Types of Splits

Categorical attributes

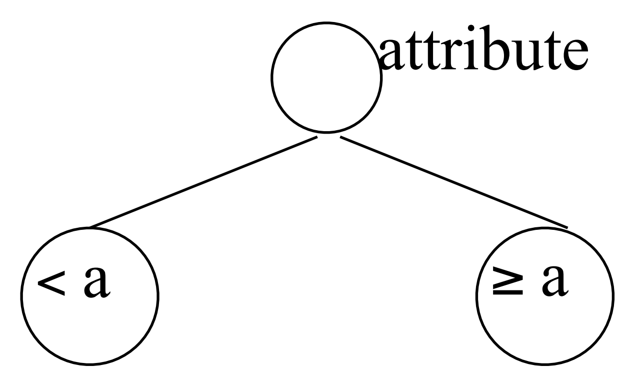
- Conditions of the form „*attribute* = *a*“ or „*attribute* ∈ *set*“
- Many possible subsets





Numerical attributes

- Conditions of the form „*attribute* < *a*“
- Many possible split points



Quality Measures for Splits

Given

- a set T of training data
- a disjoint, exhaustive partitioning $T_1, T_2, \ldots, T_m$ of T
- p_i the relative frequency of class c_i in T

Wanted

- A measure of the impurity of set S (of training data) w.r.t. class labels
- A split of T in $T_1, T_2, \ldots, T_m$ *minimizing* this impurity measure

→ information gain, gini-index

Information Gain

- *Entropy*: minimal number of bits to encode a message to transmit the class of a random training data record
- *Entropy* for a set T of training data:

$$entropy(T) = -\sum_{i=1}^{k} p_i \cdot \log_2 p_i$$

$entropy(T) = 0$, if $p_i = 1$ for some i

$entropy(T) = 1$ for $k = 2$ classes with $p_i = 1/2$

- Let attribute A produce the partitioning $T_1, T_2, \ldots, T_m$ of T.
- The *information gain* of attribute A w.r.t T is defined as

$$InformationGain(T, A) = entropy(T) - \sum_{i=1}^{m} \frac{|T_i|}{|T|} \cdot entropy(T_i)$$

Gini-Index

- *Gini index* for a set T of training data records

$$gini(T) = 1 - \sum_{j=1}^{k} p_j^{\,2}$$

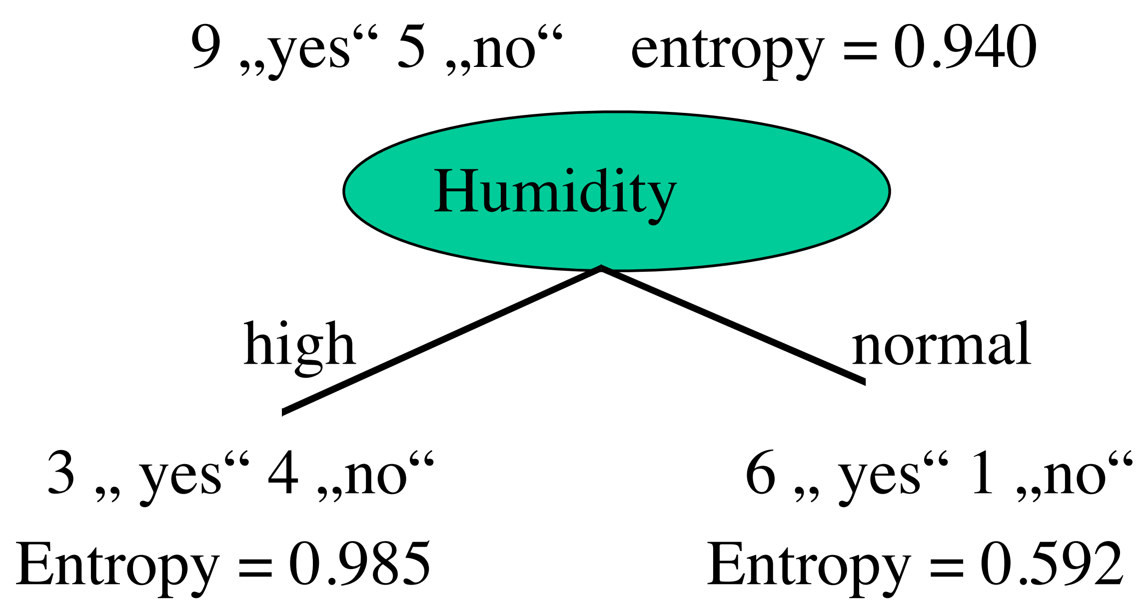
  low gini index ⇔ low impurity,

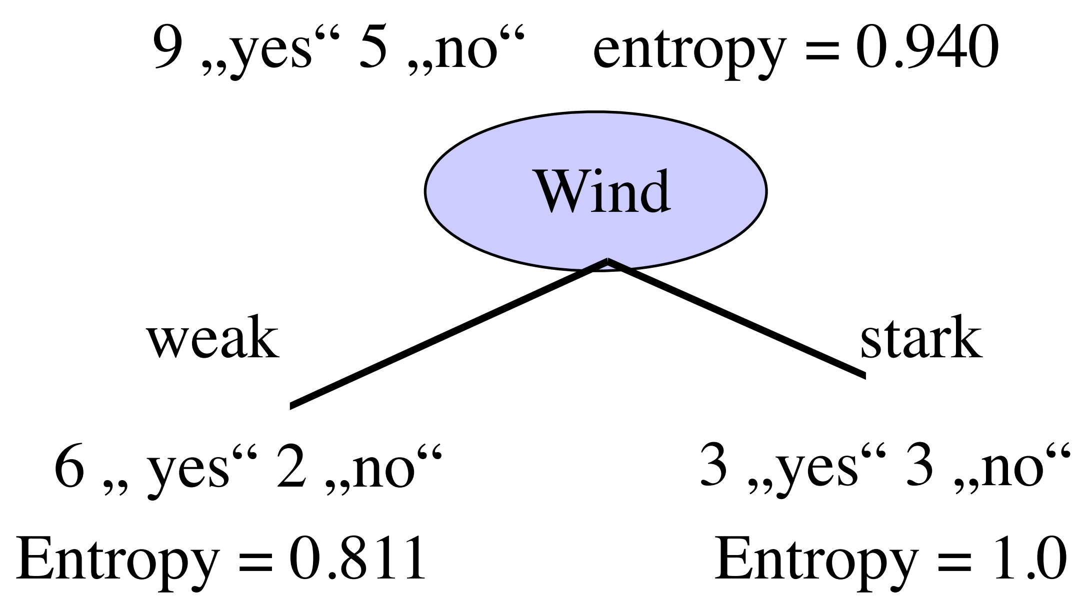
  high gini index ⇔ high impurity

- Let attribute A produce the partitioning $T_1, T_2, \ldots, T_m$ of T.
- *Gini index* of attribute A w.r.t. T is defined as

$$gini_A(T) = \sum_{i=1}^{m} \frac{|T_i|}{|T|} \cdot gini(T_i)$$

Example





$$InformationGain(T, Humidity) = 0.94 - \frac{7}{14} \cdot 0.985 - \frac{7}{14} \cdot 0.592 = 0.151$$

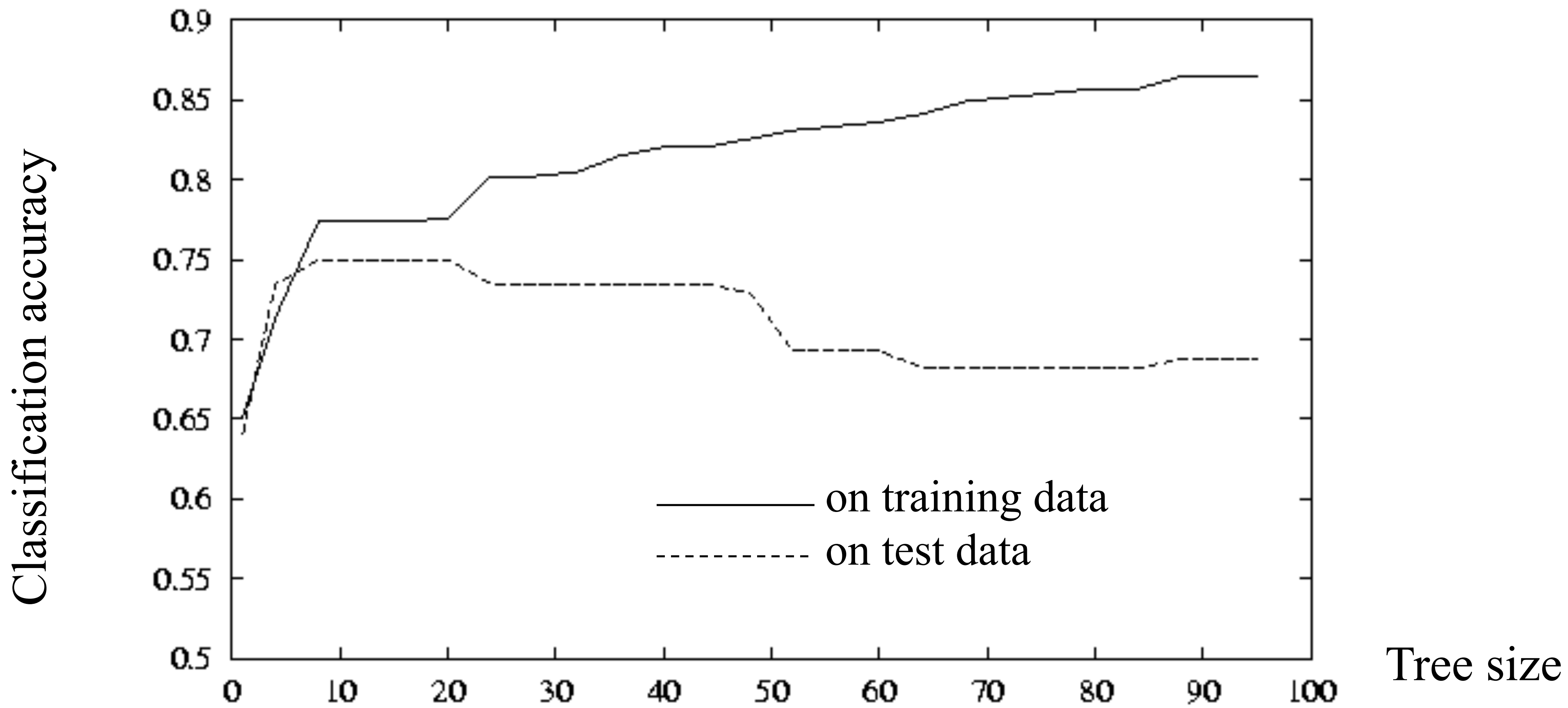
$$InformationGain(T, Wind) = 0.94 - \frac{8}{14} \cdot 0.811 - \frac{6}{14} \cdot 1.0 = 0.048$$

Humidity yields the higher information gain

Overfitting

Overfitting: there are two decision trees *T* and *T'* with

- *T* has a lower error rate than *T'* on the *training* data, but
- *T*' has a lower *test* error rate than *T*.



Approaches for Avoiding Overfitting

- Removal of erroneous training data

  in particular, inconsistent training data

- Choice of appopriate size of training data set

  not too small, not too large

- Choice of appropriate minimum support

  minimum support:

  minimum number of training data records belonging to a leaf node

  minimum support >> 1

Approaches for Avoiding Overfitting

- Choice of appropriate minimum confidence

  minimum confidence: minimum percentage of the majority class of a leaf node

  minimum confidence << 100%

  leaves can also absorb noisy / erroneous training data records

- Subsequent pruning of the decision tree

  remove overfitting branches

  see next section

Error Reduction-Pruning [Mitchell 1997]

- Train-and-Test paradigm
- Construction of decision tree *T* for training data set *TR*.
- Pruning of *T* using test data set *TE*
  - Determine subtree of *T* such that its removal leads to the maximum reduction of the classification error on *TE*.
  - Remove this subtree.
  - Stop, if no more such subtree.

only applicable if enough labled data available

Minimal Cost Complexity Pruning

[Breiman, Friedman, Olshen & Stone 1984]

- Cross-Validation paradigm
  Applicable even if only small number of labled data available
- Pruning of decision tree using training data set
  Cannot use classification error as quality measure
- Novel quality measure for decision trees
  Trade-off between (observed) classification error and tree size
  Weighted sum of classification error and tree size

Small decision trees tend to generalize better to unseen data

Notions

- *Size* $|T|$ of decision tree T: number of leaves
- *Cost complexity* of T *w.r.t.* training data set TR and complexity parameter $\alpha \geq 0$:

$$CC_{TR}(T,\alpha) = error_{TR}(T) + \alpha \cdot |T|$$

- The *smallest minimizing subtree* $T(\alpha)$ of T w.r.t. α has the following properties :

  (1) There is no subtree of T with smaller cost complexity.

  (2) If $T(\alpha)$ and T' satisfy condition (1), then $T(\alpha)$ is a subtree of T'.

- $\alpha = 0$: $T(\alpha) = T$
- $\alpha = \infty$: $T(\alpha)$ = root of T
- $0 < \alpha < \infty$: $T(\alpha)$ = true subtree of T (more than the root)

Notions

- T_e: subtree of T with root e, $\{e\}$: tree consisting only of node e

  $T > T'$: subtree relationship

- For small values of α: $CC_{TR}(T_e, \alpha) < CC_{TR}(\{e\}, \alpha)$,

  for large values of α: $CC_{TR}(T_e, \alpha) > CC_{TR}(\{e\}, \alpha)$.

- *critical value* of α w.r.t. e

  α_{crit}: $CC_{TR}(T_e, \alpha_{crit}) = CC_{TR}(\{e\}, \alpha_{crit})$

  for $\alpha \geq \alpha_{crit}$ the subtree of node e should be pruned

- *weakest link*: node with minimal α_{crit} value

Method

- Start with complete decision tree *T*.
- Iteratively, each time remove the weakest link from the current tree.
- If several weakest links: remove all of them in the same step.

sequence of pruned trees $T(\alpha_1) > T(\alpha_2) > \ldots > T(\alpha_m)$

→ with $\alpha_1 < \alpha_2 < \ldots < \alpha_m$

- Selection of the best $T(\alpha_i)$

estimate the true classification error of all $T(\alpha_1), T(\alpha_2), \ldots, T(\alpha_m)$

performing *l*-fold cross-validation on the training data set

Example

i	|Ti|	training error	estimated error	true error
1	71	0,0	0,46	0,42
2	63	0,0	0,45	0,40
3	58	0,04	0,43	0,39
4	40	0,10	0,38	0,32
5	34	0,12	0,38	0,32
6	19	0,2	0,32	0,31
7	10	0,29	0,31	0,30
8	9	0,32	0,39	0,34
9	7	0,41	0,47	0,47
10	. . .	. . .	. . .	. . .

T_7 has the lowest estimated error

and the lowest true error

Introduction

- When building a probabilistic classifier, we would like to find the classifier (hypothesis) *h* that has the maximum conditional probability given the observed data, i.e.

$$\max_{h \in H} P(h \mid D)$$

- But how to compute these conditional probabilities for all possible classifiers *h*?
- Bayes theorem

$$P(A \mid B) \cdot P(B) = P(B \mid A) \cdot P(A)$$

- Applying Bayes theorem, we obtain

$$P(h \mid D) = \frac{P(D \mid h) \cdot P(h)}{P(D)} \text{ and}$$

$$\max_{h \in H} P(h \mid D) = \max_{h \in H} P(D \mid h) \cdot P(h)$$

Introduction

$$\max_{h \in H} P(h \mid D) = \max_{h \in H} P(D \mid h) \cdot P(h)$$

$P(h \mid D)$: posterior probability of h given the data D

$P(D \mid h)$: likelihood of the data D given hypothesis h

$P(h)$: prior probability of h

- The more training data *D* we have, the higher becomes the influence of *P(D\h)*.
- *P(h)* is subjective.
- *P(h)* can, e.g., favor simpler over more complex hypotheses.
- If there is no prior knowledge, i.e. $P(h)$ uniformly distributed, then we obtain the Maximum Likelihood Classifier as a special case.

Introduction

- When applying a learned hypothesis h to classify an object o, we could use the following decision rule: $\underset{c_j \in C}{argmax}\, P(c_j \mid h)$
- h depends on the attribute values of o, i.e. $o_1, \ldots, o_d$.
- Therefore we determine $\underset{c_j \in C}{argmax}\, P(c_j \mid o_1, \ldots, o_d)$
- Applying Bayes theorem, we obtain

$$\underset{c_j \in C}{argmax}\; P(c_j \mid o_1, \ldots, o_d) = \underset{c_j \in C}{argmax}\; \frac{P(o_1, \ldots, o_d \mid c_j) \cdot P(c_j)}{P(o_1, \ldots, o_d)}$$

$$= \underset{c_j \in C}{argmax}\; P(o_1, \ldots, o_d \mid c_j) \cdot P(c_j)$$

Bayesian Classifier

Naive Bayes Classifier

- Estimate the $P(c_j)$ using the observed frequencies of the individual classes.
- How to estimate the $P(o_1, ..., o_d \mid c_j)$?
- Assumption:
  - Attribute values o_i are conditionally independent, given class c_j
  - $P(o_i \mid c_j)$ are easier to estimate from the training data than $P(o_1, ..., o_d \mid c_j)$

$$\sum_{i=1}^{d} |A_i| \text{ instead of } \prod_{i=1}^{d} |A_i| \text{ parameters to estimate}$$

- Decision rule of the *Naive Bayes-Classifier*

$$\underset{c_j \in C}{argmax} \; P(c_j) \cdot \prod_{i=1}^{d} P(o_i | c_j)$$

Bayesian Networks

- Naive Bayes-Classifier is very efficient, but assumptions may be unrealistic

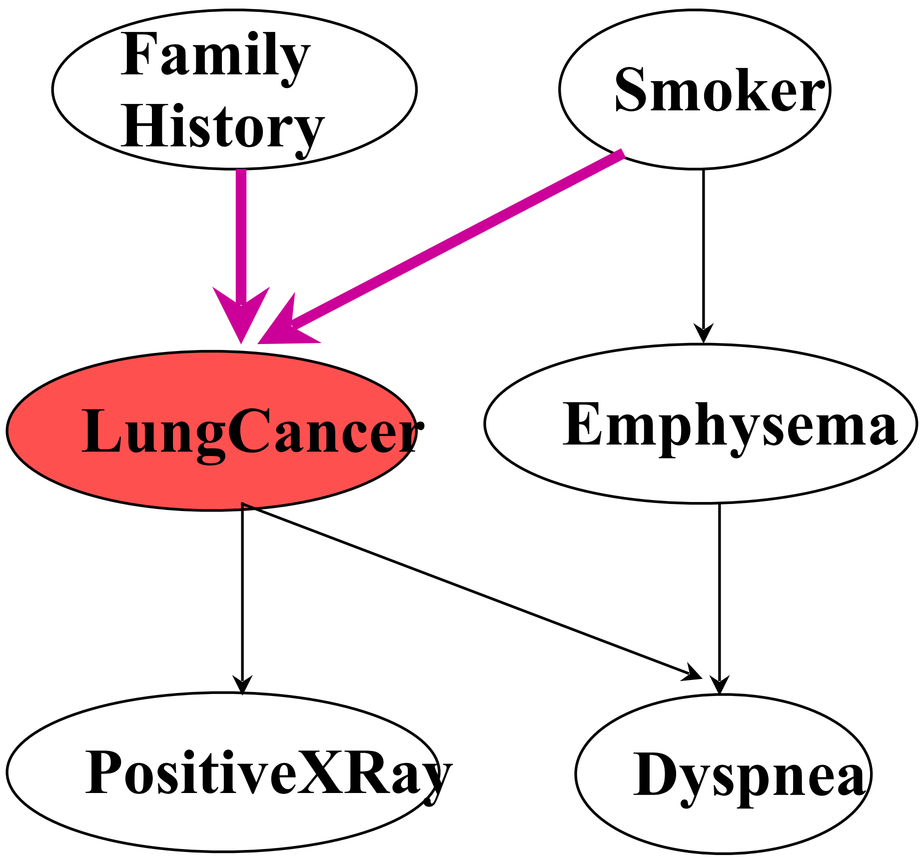
  ⇒ suboptimal classification accuracy
- Often, only some attributes are dependent, most are independent
  (given some class)
- Bayesian networks (Bayesian belief networks / Bayes nets)
  allow you to specify all variable dependencies,
  all other variables are assumed to be conditionally independent
- Network respresents subjective, a-priori beliefs

Bayesian Networks

- Graph with nodes = *random variable* (attribute) and
  edge = *conditional dependency*
- Each random variable is (for given values of the predecessor variables) conditionally independent from all variables that are no successors.
- For each node (random variable): Table of conditional probabilities given values of the predecessor variables

Bayesian network can represent causal *knowledge*

Example



	(FH,S)	(FH,~S)	(~FH,S)	(~FH,~S)
LC	0.8	0.5	0.7	0.1
~LC	0.2	0.5	0.3	0.9

Conditional probabilities for LungCancer

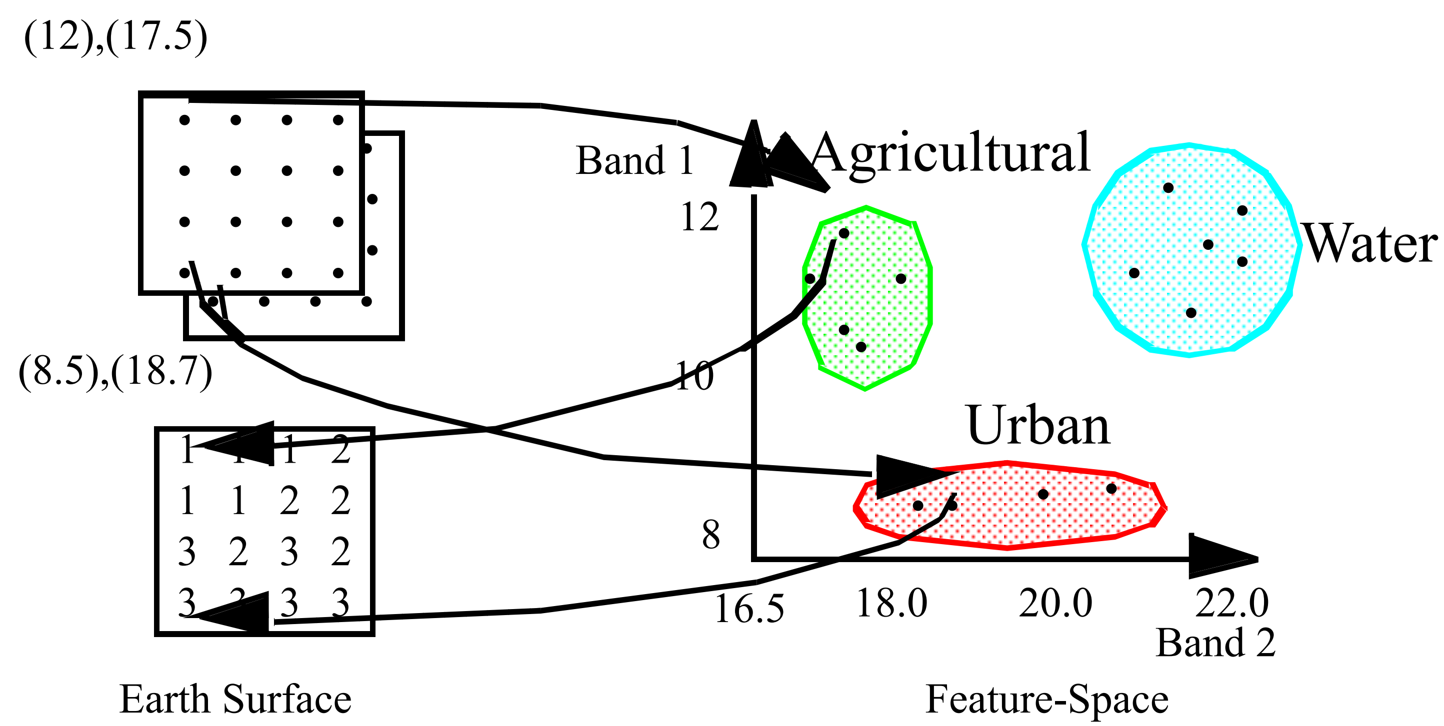
For given values of FamilyHistory and Smoker, the value of Emhysema does not provide any additional information about LungCancer

Training Bayesian Networks

- With given network structure and fully observable random variables
  all attribute values of the training examples known
  estimate conditional probability tables by calculating the relative frequencies
- With given network structure and partially known random variables
  some attribute values of the training examples unknown
  expectation maximization (EM) algorithm
  random initialization of the unknown attribute values
- With apriori unknown network structure (very difficult!)
  assume fully observable random variables
  heuristic scoring functions for alternative network structures

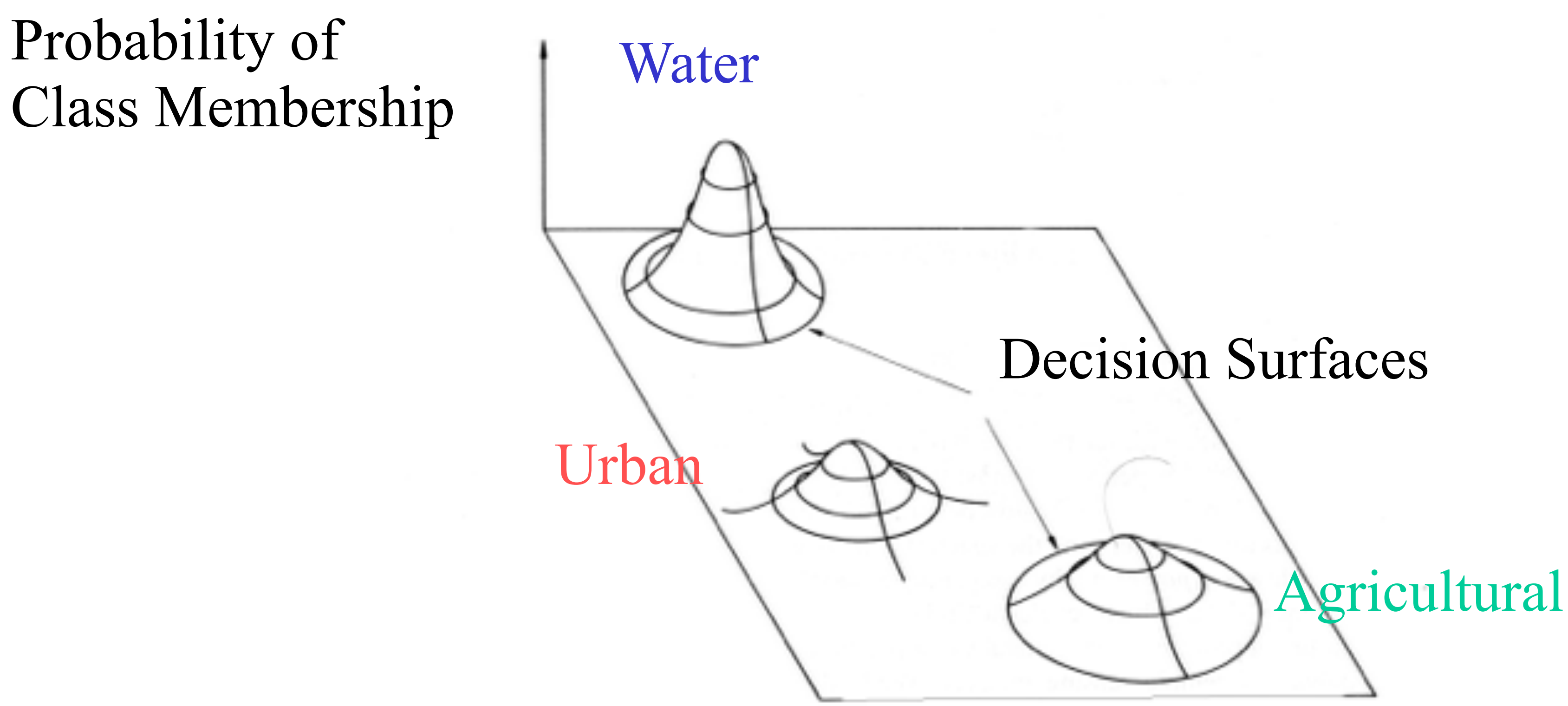
Interpretation of Raster Images

- *Automatical* interpretation of *d* raster images of a given region
  for each pixel: a *d*-dimensional vector of grey values $(o_1, \ldots, o_d)$
- Assumption: different kinds of landuse exhibit characteristic behaviors of reflection / emission



Interpretation of Raster Images

- Application of the (optimal) Bayes classifier
- Estimate the $P(o_1, \ldots, o_d \mid c_j)$ without assuming conditional indepency
- Assume a d-dimensional Normal distribution of the grey value vectors of a given class



Method

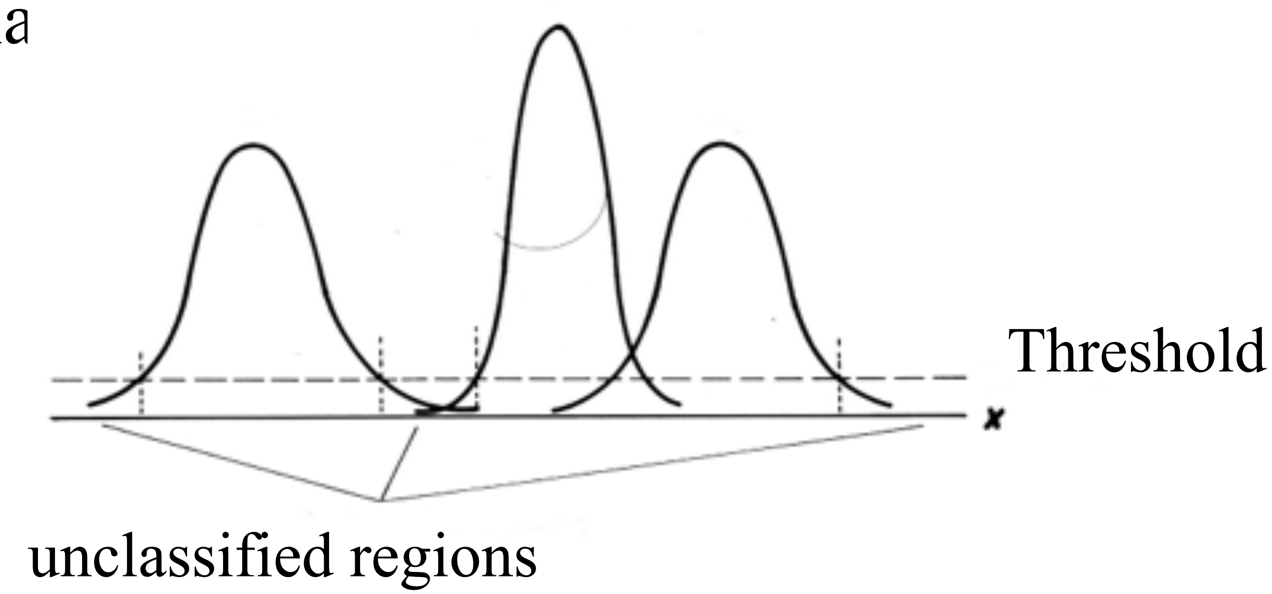
- Estimate from the training data

  μ_i: d-dimensional mean vector of all feature vektors of class c_i

  Σ_i: $d \cdot d$ covariance matrix of class c_i

- Problems of the decision rule

  - Likelihood for the chosen cla
    very small
  - Likelihood for several
    classes similar



Discussion

\+ Optimality property
Standard for comparison with other classifiers

\+ High classification accuracy in many applications

\+ Incrementality
classifier can easily be adapted to new training objects

\+ Integration of domain knowledge

\- Applicability
the conditional probabilities may not be available

\- Maybe inefficient
For high numbers of features
in particular, Bayesian networks

Motivation

- Optimal Bayes classifier assuming a *d*-dimensional Normal distribution

  Requires estimates for μ_i and Σ_i

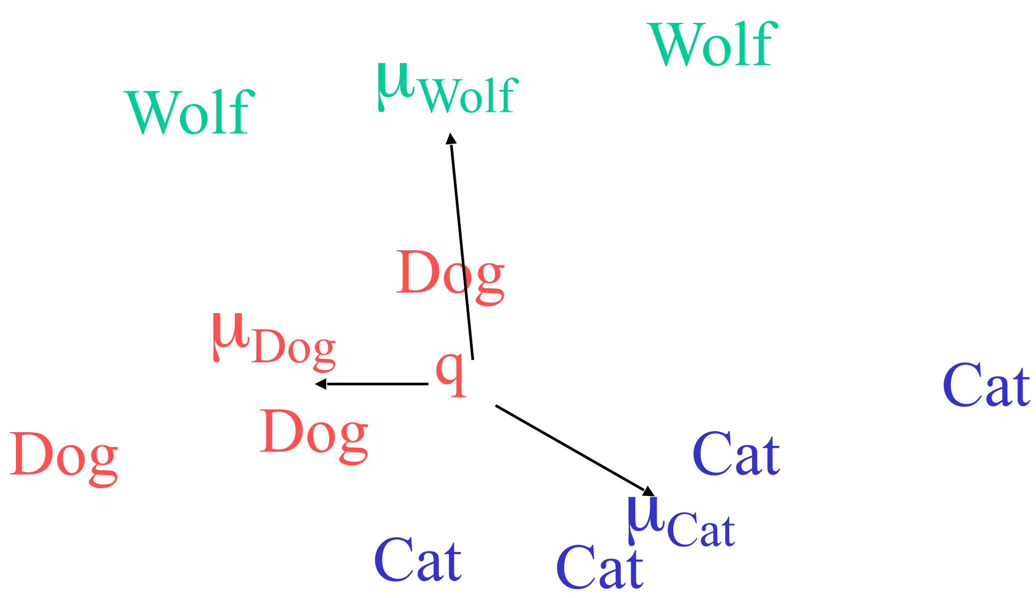
  Estimate for μ_i needs much less training data

- Goal

  classifier using only the mean vectors per class

⇒ Nearest-neighbor classifier

Example



Classifier:
q is a dog!

Instance-Based Learning
Related to Case-Based Reasoning

Overview

Base method

- Training objects o as feature (attribute) vectors $o = (o_1, \ldots, o_d)$
- Calculate the mean vector μ_i for each class c_i
- Assign unseen object to class c_i with nearest mean vector μ_i

Generalisations

- Use more than one representative per class
- Consider $k > 1$ neighbors
- Weight the classes of the k–nearest neighbors

Notions

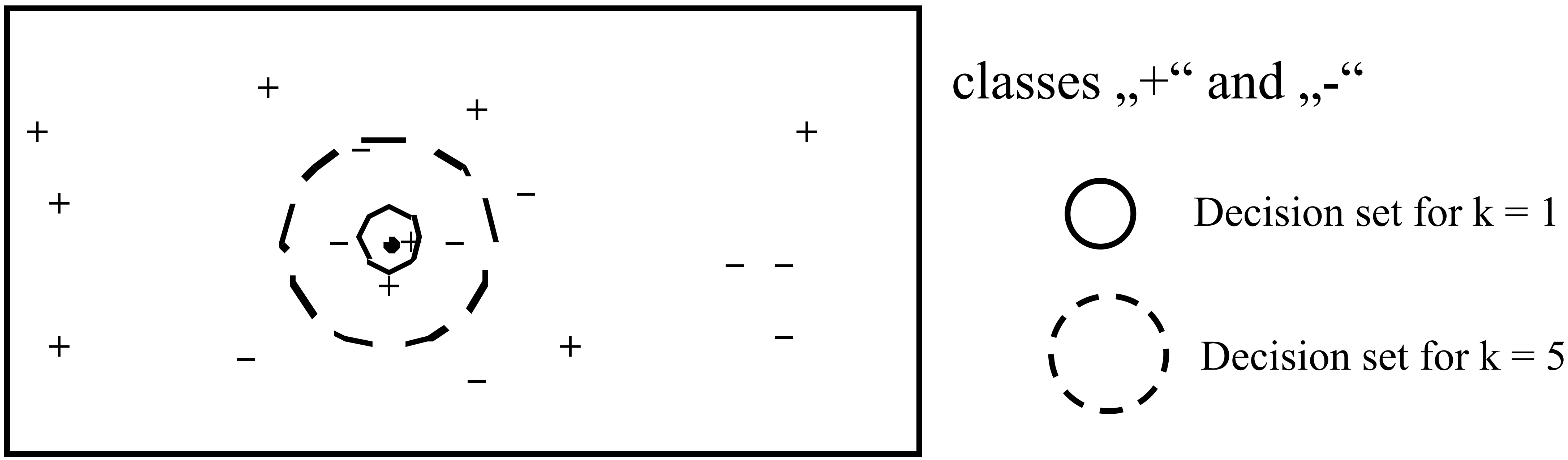
- Distance function
  defines similarity (dissimilarity) for pairs of objects

- *k*: number of neighbors considered

- *Decision Set*
  set of *k*-nearest neighbors considered for classification

- *Decision rule*
  how to determine the class of the unseen object
  from the classes of the decision set?

Example



Uniform weight for the decision set

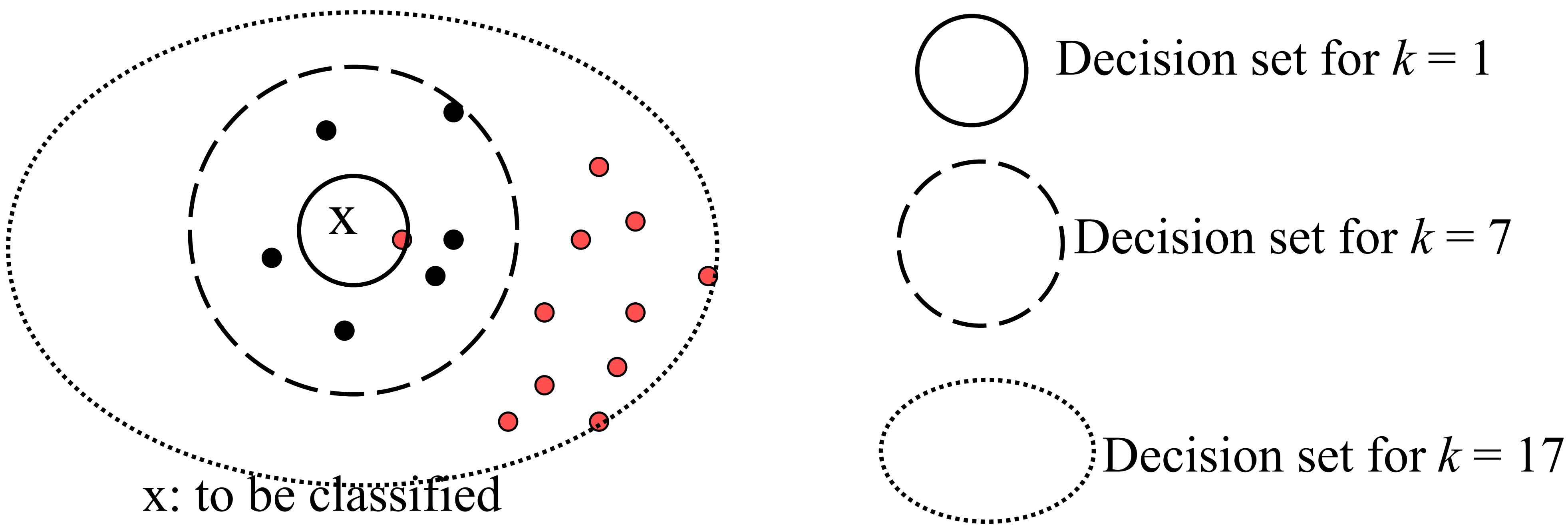
k = 1: classification as „+“, *k* = 5 classification as „–“

Inverse squared distance as weight for the decision set

k = 1 and *k* = 5: classification as „+“

Choice of Parameter k

- „too small“ *k*: very sensitive to outliers
- „too large“ *k*: many objects from other clusters (classes) in the decision set
- medium *k*: highest classification accuracy, often $1 << k < 10$



Decision Rule

Standard rule

Choose the majority class within the decision set

Weighted decision rule

Weight the classes of the decision set

- By distance
- By class distribution (often skewed!)

  class A: 95 %, class B 5 %

  Decision set = {A, A, A, A, B, B, B}

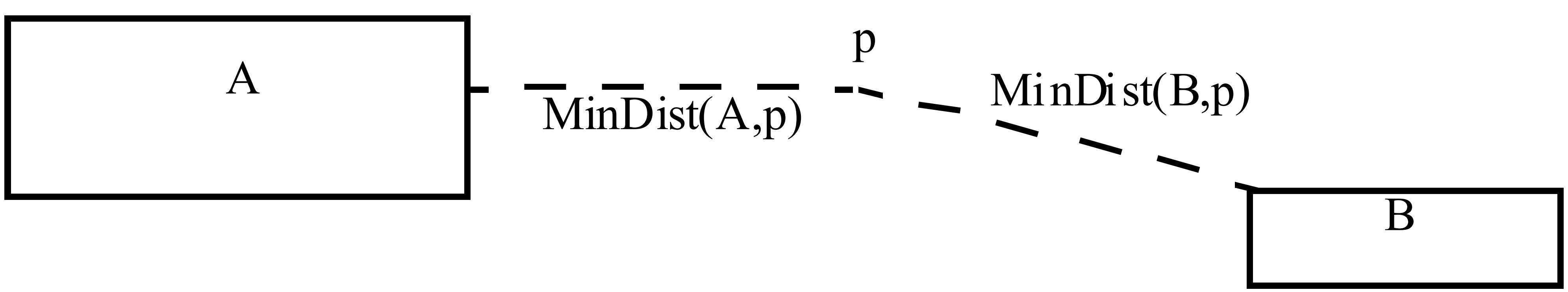
  Standard rule $\Rightarrow$ A, Weighted rule $\Rightarrow$ B

Index Support for k-Nearest-Neighbor Queries

- Balanced index tree (such as X-tree or M-tree)
- Query point p
- PartitionList

  BBs of subtrees that need to be processed, sorted in ascending order w.r.t. MinDist to p
- NN

  Nearest neighbor of p in the data pages read so far



Index Support for k-Nearest-Neighbor Queries

- Remove all BBs from PartitionList that have a larger distance to `p` than the currently best NN of `p`
- PartitionList is sorted in ascending order w.r.t. MinDist to `p`
- Always pick the first element of PartitionList as the next subtree to be explored
  Does not read any unnecessary disk pages!
- Query processing limited to a few paths of the index structure

Average runtime O(log *n*) for „not too many“ attributes


For very large numbers of attributes: O(*n*)

Discussion

+ Local method
  Does not have to find a global decision function (decision surface)
+ High classification accuracy
  In many applications
+ Incremental
  Classifier can easily be adapted to new training objects
+ Can be used also for prediction

- Application of classifier expensive
  Requires *k*-nearest neighbor query
- Does not generate explicit knowledge about the classes

Introduction [Burges 1998]

Input $S = \{(x_1, y_1),\ldots,(x_n, y_n)\}$ $x_i \in X$

a training set of objects and their known classes

$y_i \in \{-1, +1\}$

Output

a classifier

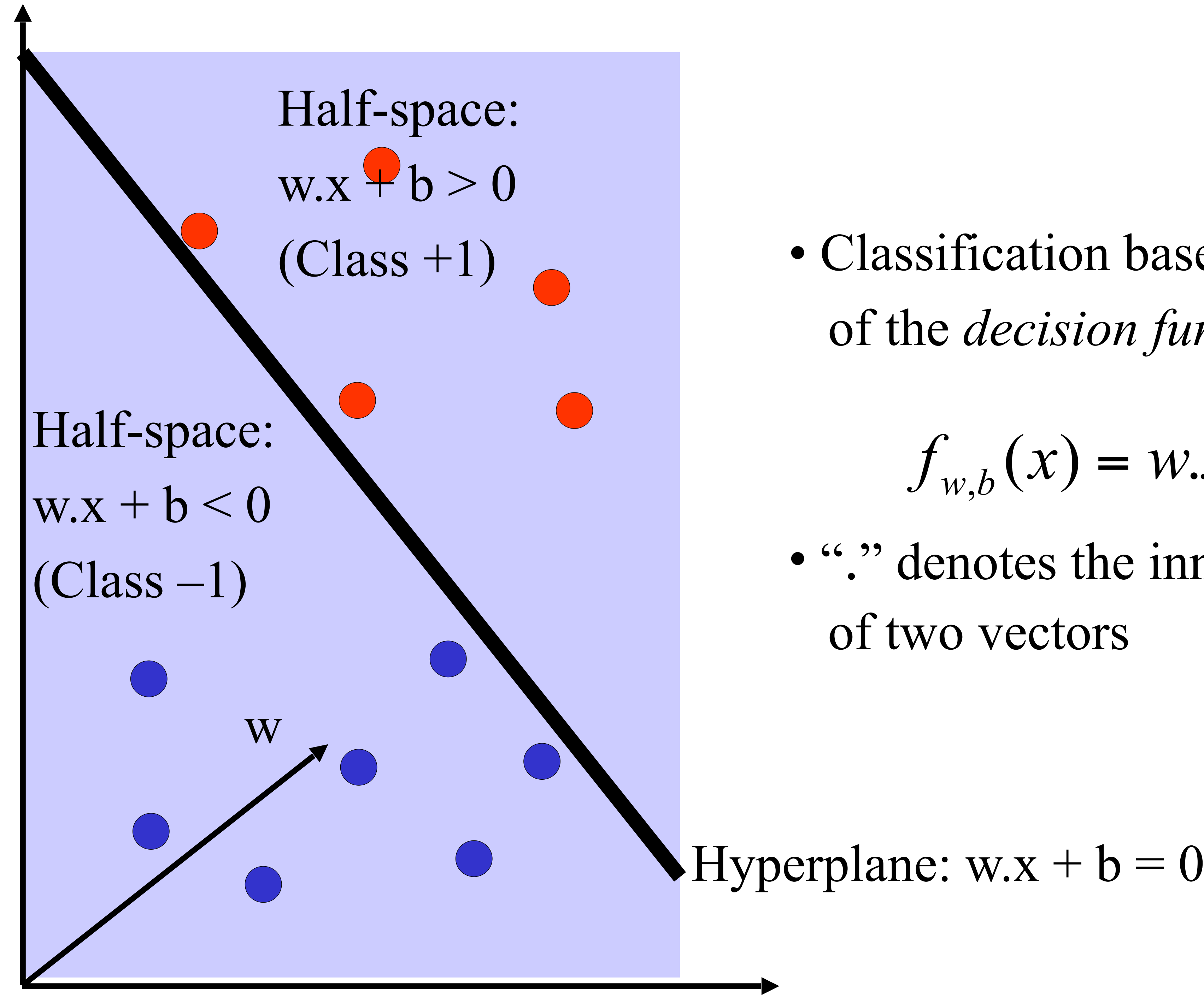
$f : X \rightarrow \{-1, +1\}$

Goal

Find the best separating hyperplane (e.g., lowest classification error)

Two-class problem

Introduction

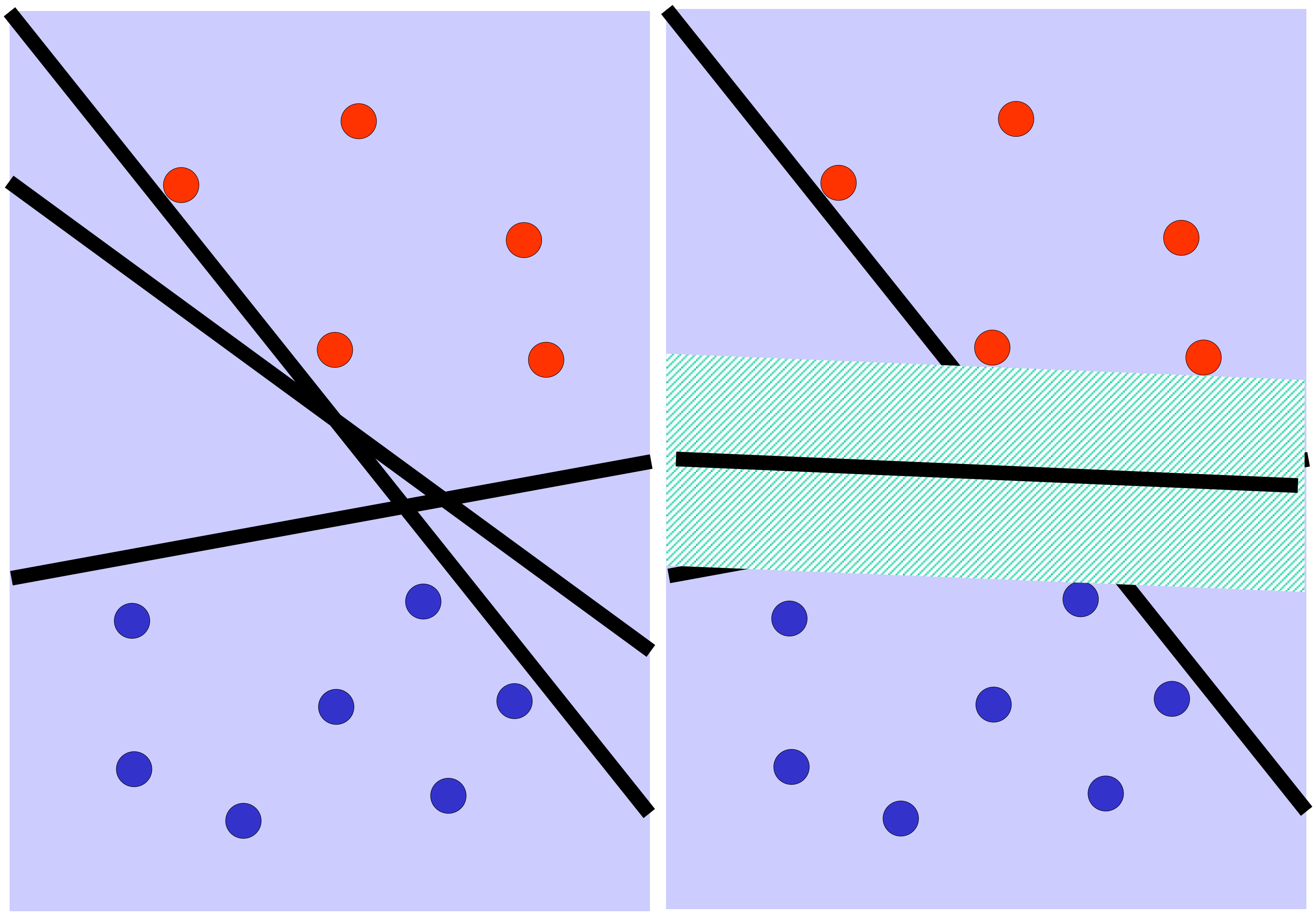


- Classification based on the *sign* of the *decision function*

$$f_{w,b}(x) = w.x + b$$

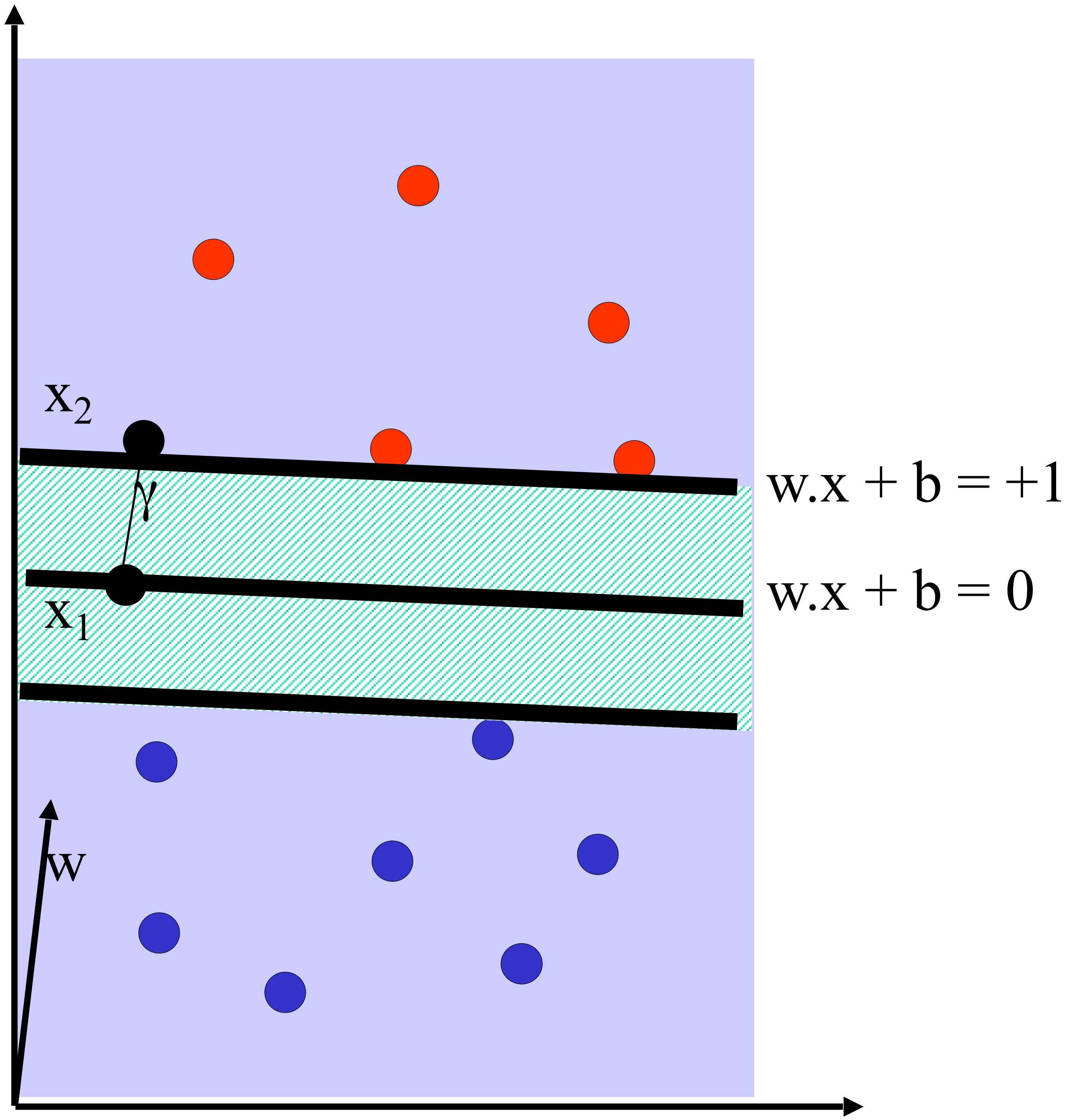
- "." denotes the inner product of two vectors

Introduction



Choose hyperplane with *largest margin* (maximum distance to closest training object)

Introduction



$w.\ x_1 + b = 0$

$w.\ x_2 + b = 1$

à $w.\ (x_2 - x_1) = 1$

à $\|w\|\ \|\ x_2 - x_1\ \|\ \cos 0 = 1$

$$\gamma = \| x_2 - x_1 \| = \frac{1}{\| w \|}$$

γ: margin

Method

Problem

- Minimize $||w||^2$
- Under the constraints $\forall i = 1,\dots,n: \; y_i(w.x_i + b) - 1 \geq 0$

Dual problem

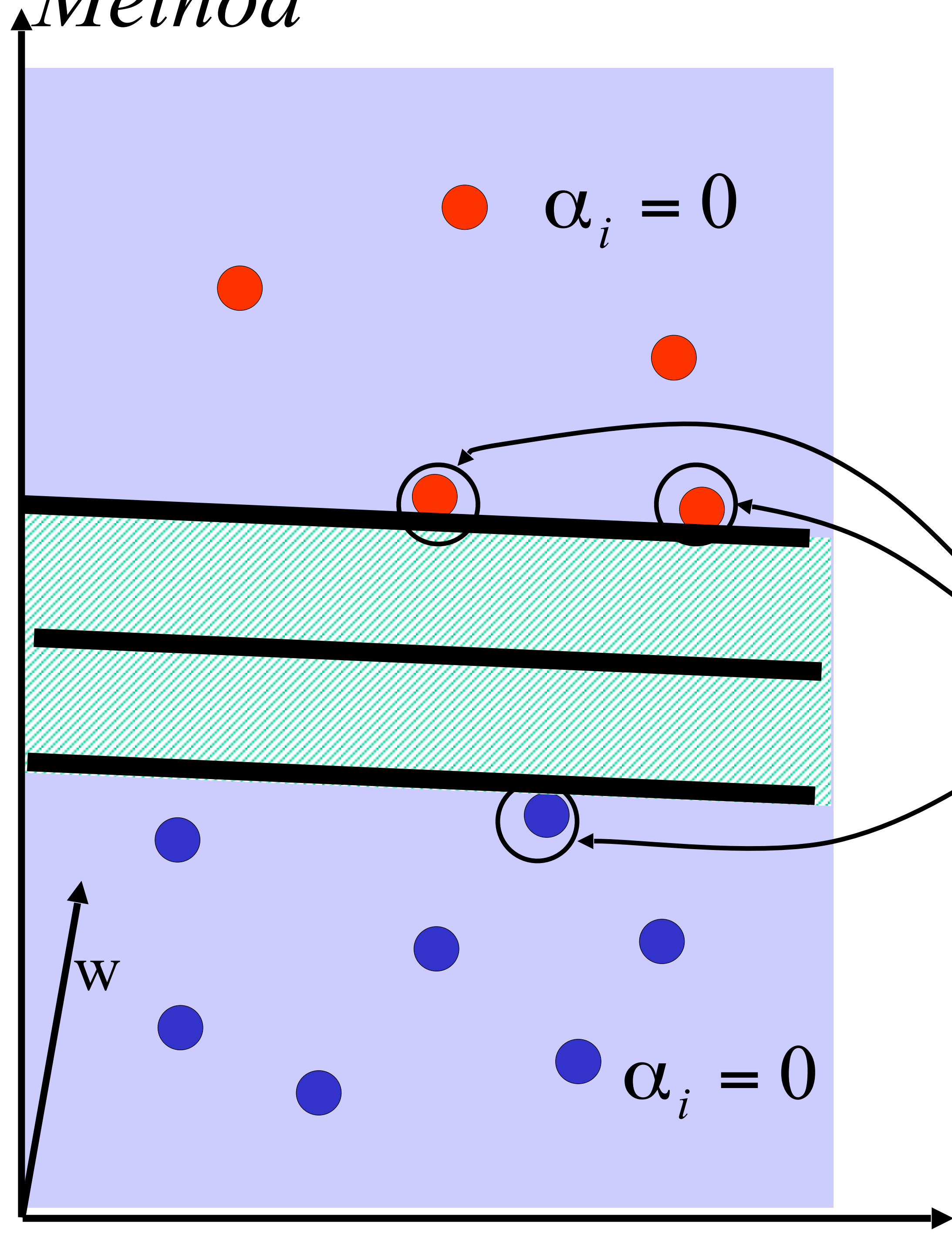
- Introduce dual variables α_i for each training object *i*
- Find α_i maximizing

$$L(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \cdot \alpha_j \cdot y_i \cdot y_j \cdot x_i . x_j$$

under the constraints $\alpha_i \geq 0$ and $\sum_{i=1}^{n} \alpha_i \cdot y_i = 0$

Quadratic programming problem

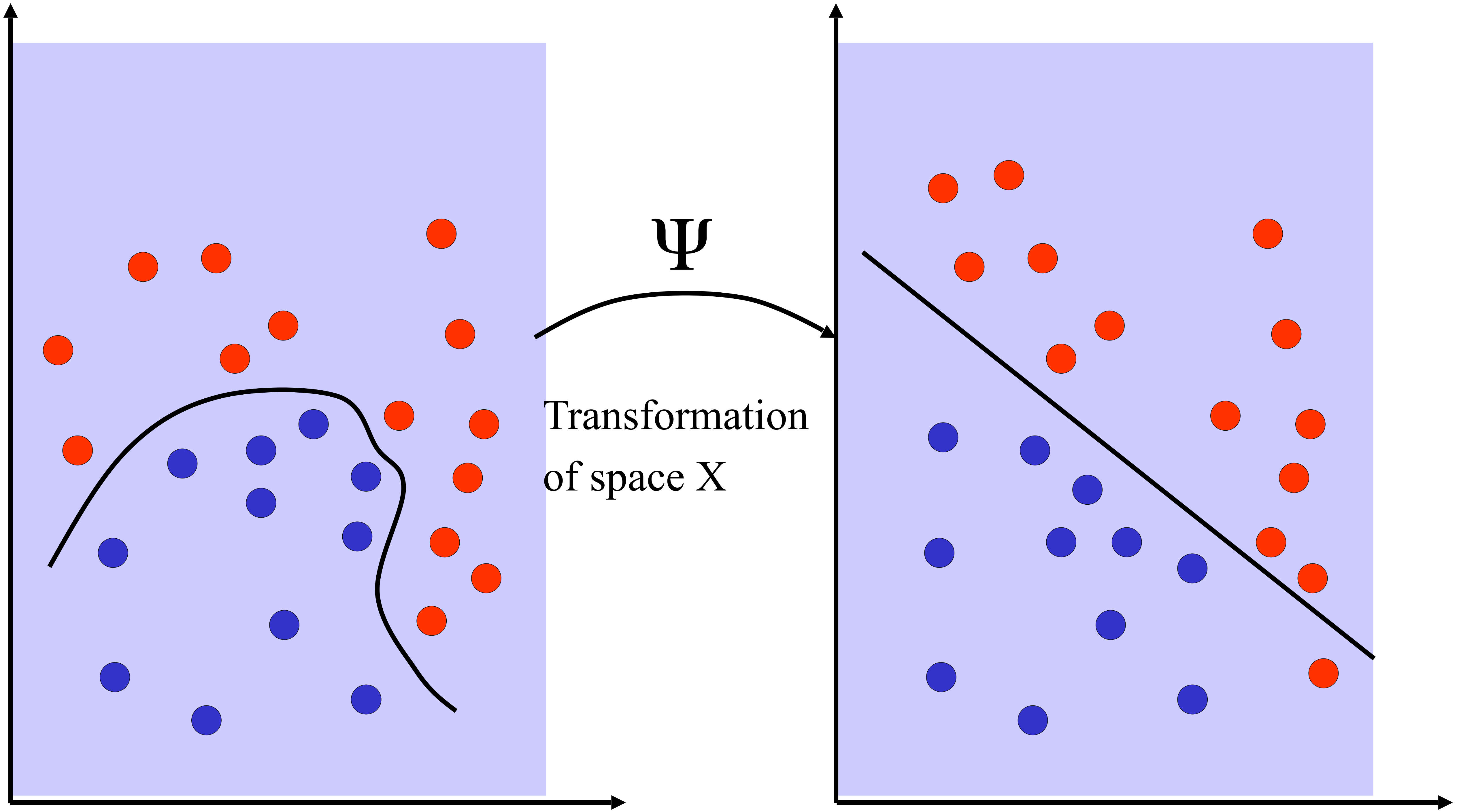
Method



- Only training objects with $\alpha_i > 0$ contribute to w
- These training objects are the *support vectors*

Typically, number of support vectors << n

Non-Linear Classifiers



Non-Linear Classifiers

- Decision function $f_{w,b}(x) = w.\Psi(x) + b$
- *Kernel* of two objects $\forall x, x' \in X : \quad K(x, x') = \Psi(x).\Psi(x')$
- Explicit computation of $\Psi(x)$ is not necessary
- Example: $\Psi(x) = (x_1^2, x_2^2, \sqrt{2}x_1 x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1)$

$$K(x, x') = \Psi(x).\Psi(x') = (x.x'+1)^2$$

Kernels

- Kernel is a similarity measure
- K(x,x') is a *kernel* iff

$$\forall x_i \in X : \begin{pmatrix} K(x_1,x_1) & K(x_1,x_2) & \cdots \\ K(x_2,x_1) & K(x_2,x_2) & \cdots \\ & \vdots & \end{pmatrix}$$

is a symmetric and positive definite matrix

SVM for Protein Classification [Leslie et al 2002]

- Two sequences are similar when they share many common substrings (subsequences)
- $K(x,x') = \sum_{s\ common\ substring} \lambda^{|s|}$ where λ is a parameter

  and |s| denotes the length of string s

- Very high classification accuracy for protein sequences
- Variation of the kernel (when allowing gaps in the matching subsequences)

$$K(x,x') = \sum_{s\ common\ substring} \lambda^{length(s,x)+length(s,x')}$$

length(s,x): length of the subsequence of x matching s

SVM for Prediction of Translation Initiation Sites [Zien et al 2000]

- Translation initiation site (TIS): starting position of a protein coding region in DNA

  All TIS start with the triplet "ATG"

- Problem: given an "ATG" triplet, does it belong to a TIS?
- Representation of DNA

  Window of 200 nucleotides around candidate "ATG"

  Encode each nucleotide with a 5 bit word (00001, 00010, . . ., 10000) for

  A, C, G, T and unknown

  → Vectors of 1000 bits

SVM for Prediction of Translation Initiation Sites

- Kernels

$$K(x,x') = (\mathrm{x}.\mathrm{x}')^{\mathrm{d}}$$

d = 1: number of common bits

d = 2: number of common pairs of bits

. . .

Locally improved kernel: compare only small window around "ATG"

- Experimental results

Long range correlations do not improve performance

Locally improved kernel performs best

Outperforms state-of-the-art methods

Discussion

\+ Strong mathematical foundation

\+ Find global optimum

\+ Scale well to very high-dimensional datasets

\+ Very high classification accuracy

In many challenging applications

\- Inefficient model construction

Long training times (~ O (n^2))

\- Model is hard to interpret

Learn only weights of features

Weights tend to be almost uniformly distributed

The Single Table Assumption

- Existing data mining algorithms expect data in a single table
- But in reality, DBs consist of multiple tables
- Naive solution: join all tables into a single one (*universal relation*) and apply (single-relational) data mining algorithm

Purchases

Client#	**Date**	**Item**	**Quantity**
2765	02/25/2005	A	5
3417	02/26/2005	B	1
1005	02/26/2005	C	12
. . .			

Clients

Client#	**Name**	**Age**
1005	Jones	35
1010	Smith	52
1054	King	27
. . .		

The Single Table Assumption

- Universal relation

Client#	**Date**	**Item**	**Quantity**	**Name**	**Age**
1005	02/26/2005	C	12	Jones	35
1005	02/28/2005	B	2	Jones	35
. . .	. . .	. . .	. . .	. . .	. . .
2765	02/25/2005	A	5	Bornman	23
. . .					

There are no more client entities!
What if rule depends on how many different items were purchased by a client?

Aggregating Related Tables

- Enhancing „target table“ by aggregates of the related tuples in other tables
- Aggregation operators: COUNT, SUM, MIN, AVG, . . .

Client#	**Name**	**Age**	**Overall Quantity of Item A**	**Overall Quantity of Item B**	**. . .**
1005	Jones	35	0	10	. . .
1010	Smith	52	35	0	. . .
. . .					. . .

More meaningful! But what aggregates to consider?
And what if attributes of the other clients that have purchased the same item are relevant?

Multi-Relational Data Mining

- Data mining methods for multi-table databases
- Pattern search space much larger than for single tables
- Testing the validity of a pattern more expensive
- Similar data mining tasks

  classification, clustering, association rules, . . .

  . . . plus some tasks specific to the multi-relational case
- Single table (propositional) algorithms can be upgraded to multiple tables (first order predicate logic)

Inductive Logic Programming (ILP)

- Goal: learn logic programs from example data
- Knowledge representation is expressive and understandable
- Examples: tuples from multiple tables
- Hypotheses: sets of rules
- Use of background knowledge
  also set of rules

Logic Programs and Databases

- *Logic program*: set of clauses
- *Clause*: rule of the form „Head ← Body“
  where Head / Body consist of atoms connected using the logical operators
  $\wedge$, $\vee$ and $\neg$
- *Atom*: predicate applied to some terms
- *Predicate*: boolean function with arguments (terms)
- *Term*: constant (e.g., mary), variable (e.g., X),
  function symbol applied to some term

Logic Programs and Databases

- Example rule

$$father(X,Y) \vee mother(X,Y) \leftarrow parent(X,Y)$$

- Definite clauses: exactly one atom in the head

$$parent(X,Y) \leftarrow father(X,Y) \vee mother(X,Y)$$

- Horn clauses

One (positive) atom in the head, conjunction of body atoms

$$mother(X,Y) \leftarrow parent(X,Y) \wedge female(Y)$$

Classical Rule Induction Task

- Given:
  set P of examples from target relation (positive examples)
  set N of examples not from target relation (negative examples)
  background predicates B
  hypothesis (rule) language
- Find a set of rules that explains all positive and none of the negative examples

consistent and complete set of rules

Example

Training examples

daughter(mary,ann) +
daughter(eve,tom) +
daughter(tom,ann) -
daughter(eve,ann) -

Background knowledge

parent(ann,mary) female(ann)
parent(ann,tom) female(mary)
parent(tom,eve) female(eve)
parent(tom,ian)

Hypothesis language

definite clauses

Resulting rule

$$daughter(X,Y) \leftarrow parent(Y,X) \wedge female(X)$$

The Sequential Covering Algorithm

Hypothesis (H) := {}

Repeat

find a clause c that covers some positive and no negative examples;

add c to H;

delete all positive examples implied by c

Until no more (uncovered) positive examples

$$B \wedge H \cup \{c\}$$

Construction of new clauses:

search of the space of clauses

applying some refinement operator

Structuring the Space of Clauses

- Substitution $\theta = \{V_1 / t_1, \ldots, V_n / t_n\}$
  assignment of terms t_i to variables V_i
- Clauses as sets of atoms (literals)

$$Head \leftarrow Body \quad \Leftrightarrow Head \vee \neg Body$$

$$e.g., \; daughter(X,Y) \leftarrow parent(Y,X):$$

$$\{daughter(X,Y), \neg parent(Y,X)\}$$

-

Clause c $\theta-subsumes$ clause c'

if there exists a substitution θ such that $c\theta \subseteq c'$

Structuring the Space of Clauses

- Examples

$$c = daughter(X,Y) \leftarrow parent(Y,X)$$

$$\theta = \{X / mary, Y / ann\}$$

$$c\theta = daughter(mary, ann) \leftarrow parent(ann, mary)$$

$$c = daughter(X,Y) \leftarrow parent(Y,X)$$

$$c' = daughter(X,Y) \leftarrow female(X) \wedge parent(Y,X)$$

$$\theta = \{\}$$

$$c\theta = c \subseteq c', \text{ i.e. } c\ \theta-\text{subsumes}\ c'$$

Structuring the Space of Clauses

- Syntactic notion of generality

clause c is *at least as general as* clause c' $(c \le c')$ iff

$$c \ \theta - subsumes \ \ c'$$

c is *more general than* clause c' iff

$$c \le c' \wedge \neg(c' \le c)$$

c is a *generalization* of c', c' a *specialization* of c

If c does not cover an example, none of its specializations do
If c covers an example, all of its generalizations do

Searching the Space of Clauses

- Top-down approach:

  start from most general clauses

  recursively apply refinement operators

- Refinement operator

  θ – subsumption – based

  returns all most general specializations of a given clause

- Types of refinements

  apply a substitution to a clause or

  add a literal to the body of the clause

Example

daughter(X,Y) ←

. . .

daughter(X,Y) ← X=Y

daughter(X,Y) ← female(X)

daughter(X,Y) ← parent(Y,X)

. . .

daughter(X,Y) ← female(X), female(Y)

daughter(X,Y) ← female(X), parent(Y,X)

Refinement graph (lattice)

Top-Down Search of Refinement Graphs

Hypothesis (H) := {}

repeat

    clause $c := p(X_1, \ldots, X_n) \leftarrow$

    repeat

        build the set S of all refinements of c;

        c := the best element of S (according to some heuristic)

    until stopping criterion satisfied (c is consistent with $B \cup H$)

    add c to H;

    delete all positive examples implied by c (using $B \cup H$)

until no more (uncovered) positive examples (i.e., H complete)

FOIL [Quinlan 1990]

- Top-down search of refinement graph
- Weighted information gain as heuristic to choose best clause
- Heuristic can be modified to allow clauses covering (some) negative examples

  → handling of noisy data
- Declarative bias to reduce search space

  syntactic restrictions on clauses to be considered

  to be provided by the user

Declarative Bias

- Argument types / domains (relational DBS)
- Input / output modes of arguments
  argument must / must not be instantiated when predicate added
- Parametrized language bias
  e.g., maximum number of variables, literals, . . . per clause
- Clause templates
  Ex.: $P(X,Y) \leftarrow Q(X,Z) \wedge R(Z,Y)$
  where P, Q, R denote predicate variables

Declarative bias difficult to specify for user (syntactic!)

CrossMine [Yin, Han, Yang & Yu 2004]

- Several improvements of FOIL and similar ILP classification methods
- Evaluation of alternative refinement operator requires joins, which are very expensive DB operations

  → TupleID propagation (virtual joins)
  propagate tupleIDs and their class labels from the target table to related tables
- Relationship tables have no attributes and may not yield a high information gain would never been chosen by FOIL

  → Increased look ahead (two instead of one literal)

TupleID Propagation

Loans		
loanID	accountID	class
1	124	+
2	124	+
3	45	-
4	45	+

Target table

Accounts				
accountID	frequency	date	ID set	Class labels
124	monthly	960227	1, 2	2+, 0-
108	weekly	970610		0+, 0-
45	monthly	970611	3,4	1+, 1-
67	weekly	990903		0+, 0-

Related table

Prediction

Commonality with classification

- First, construct a model
- Second, use model to predict unknown value

  Major method for prediction is regression

  - Simple and multiple regression
  - Linear and non-linear regression

Difference from classification

- Classification refers to predict categorical class label
- Prediction models continuous-valued functions

Linear Regression

- Predict the values of the *response variable y* based on a linear combination of the given values of the *predictor variable*(s) x_i

$$\hat{y} = a_0 + \sum_{j=1}^{d} a_j x_j$$

- *Simple regression*: one predictor variable → regression line
- *Multiple regression*: several predictor variables → regression plane
- *Residuals*: differences between observed and predicted values

Use the residuals to measure the model fit

Linear Regression

$$y(i) = \hat{y}(i) + e(i) = a_0 + \sum_{j=1}^{d} a_j x_j(i) + e(i), \quad 1 \le i \le n$$

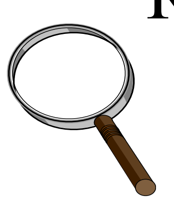
- y: vector of the y values for the n training objects
- X: matrix of the values of the d predictor variables for the n training objects (and an additional column of 1s)
- e: vector of the residuals for the n training objects
- Matrix notation:

$$y = Xa + e$$

Linear Regression

- Optimization goal: minimize $$\sum_{i=1}^{n} e(i)^2 = \sum_{i=1}^{n} [y(i) - \sum_{j=0}^{d} a_j x_j(i)]^2$$
- Solution: $$a = (X^T X)^{-1} X^T y$$
- Computational issues
  - $X^T X$ must be invertible
    Problems if linear dependencies between predictor variables
  - Solution may be unstable
    If predictor variables almost linear dependent

Equation solving e.g. using LU decomposition or SVD
Runtime complexity $O(d^2 n + d^3)$



Locally Weighted Regression

Limitations of linear regression

- Only linear models
- One global model

Locally weighted regression

Construct an explicit approximation to *f* over a local neighborhood of *query instance* x_q

Weight the neighboring objects based on their distance to x_q

Distance-decreasing weight *K*

Related to nearest neighbor classification

→ Minimize the squared *local weighted* error

Locally Weighted Regression

Local weighted error

- W.r.t. query instance x_q
- Arbitrary approximating function $\hat{f}$
- Pairwise distance function d
- Three major alternatives:

$$E(x_q) = \frac{1}{2} \sum_{x \in k\_nearest\_neighbors\_of\_x_q} (f(x) - \hat{f}(x))^2$$

$$E(x_q) = \frac{1}{2} \sum_{x \in D} [f(x) - \hat{f}(x)]^2 \cdot K(d(x_q, x))$$

$$E(x_q) = \frac{1}{2} \sum_{x \in k\_nearest\_neighbors\_of\_x_q} (f(x) - \hat{f}(x))^2 K(d(x_q, x))$$

Discussion

+ Strong mathematical foundation
+ Simple to calculate and to understand
  For moderate number of dimensions
+ High classification accuracy
  In many applications

- Many dependencies are non-linear
  Can be generalized
- Model is global
  Cannot adapt well to locally different data distributions
  But: Locally weighted regression