Classification

Contents of this Chapter

Introduction

Evaluation of classifiers [Aggarwal 2015, section 10.9]

Decision Trees [Section 10.3]

Bayesian classification [Section 10.5.1]

Logistic regression [Section 10.5.2]

Nearest-neighbor classification [Section 10.8]

Support Vector Machines [Section 10.6]

Regression analysis [Section 11.5]

Ensemble classifiers [Section 11.8]

Introduction

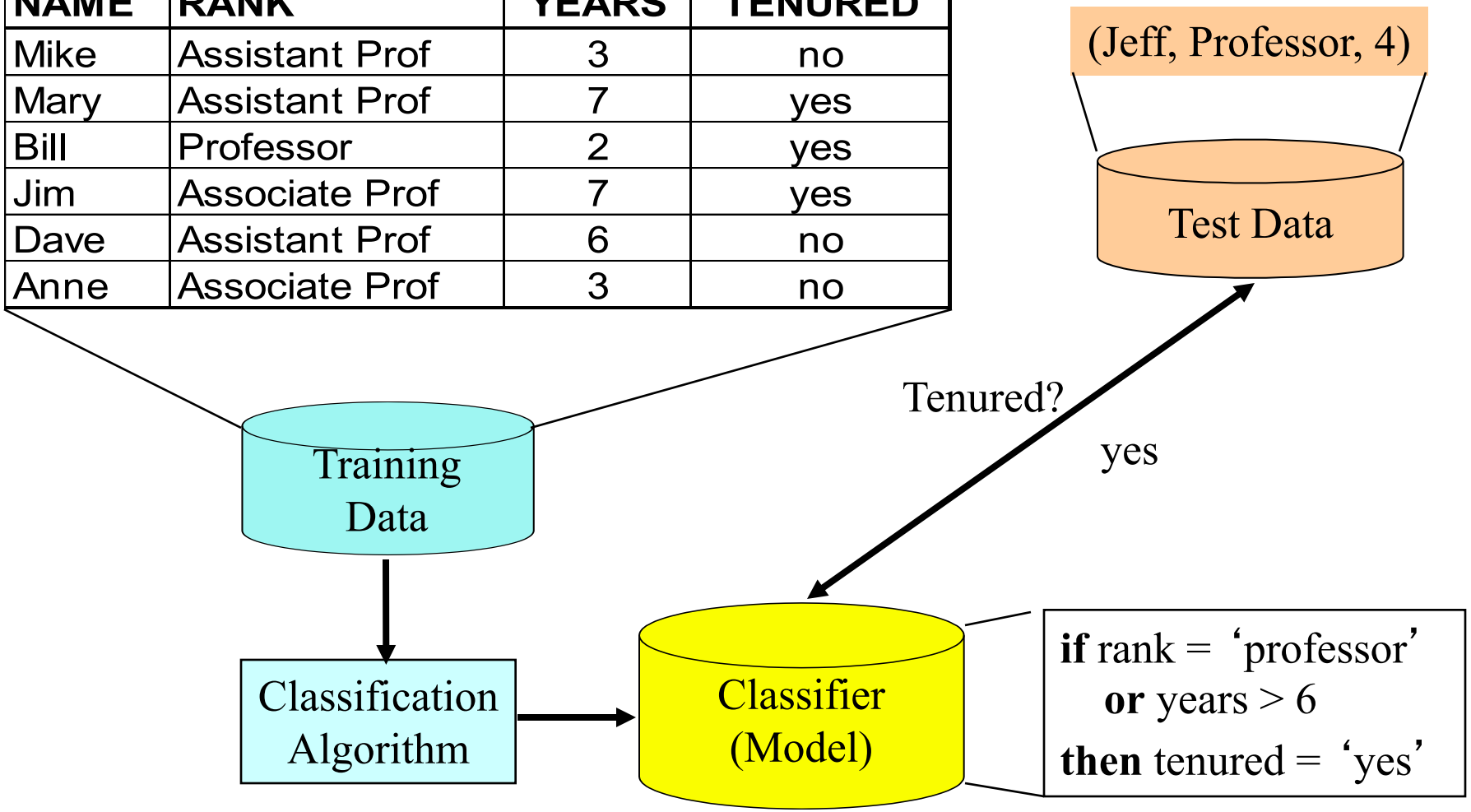
The Classification Problem

- Let $O \subseteq D$ be a set of objects of the form (o_1, . . ., o_d) with *attributes* (*features*) A_i, $1 \le i \le d$, and class label c, $c \in C = \{c_1, \ldots, c_k\}$.
- D denotes the set of all objects, the population, for which we may want to know the class label.
- O is a sample from D which we have observed.
- We assume that O is a representative sample of D.
- Goal:
  - Predict class label for objects from $D \setminus O$.
  - *Classifier* $K: A_1 \times A_2 \times \ldots A_d \rightarrow C$.
- Related problem: regression analysis

Predict the value of a *numerical* attribute.

Introduction

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



Evaluation of Classifiers

Introduction

- Given a sample of labeled data O.
- Want to learn a classifier that labels the entire population

  in particular, the unseen data $D \setminus O$.
- Can only estimate the performance of the classifier on unseen data.
- To estimate it, split labeled data into two disjoint subsets:
  - *Training data*

    for training the classifier (model learning),
  - *Test data*

    to evaluate the trained classifier.

Evaluation of Classifiers

Introduction

- Do not use the test set to tune the parameters of the classification algorithm or make other choices about classifier design.
- This would overestimate the true accuracy because knowledge of the test set has been implicitly used in the training process.
- To avoid this problem, split the training set further into training and validation set. Use validation set for parameter tuning etc.
- Approaches for creating training and test datasets:

- Hold out,
- Cross-validation.

Evaluation of Classifiers

Hold out

- Partition set *O* randomly into two (disjoint) subsets: training data and test data.
- Only a subset of the available labeled data is used for training.
- Full power of the labeled data is not reflected in the error estimate.

Not recommended for small *O*

- If class distributiuon is very skewed, take a stratified sample: sample the same percentage of each class separately.

Evaluation of Classifiers

Cross-validation

- Partition set *O* randomly into *m* same size subsets.
- Train *m* different classifiers using a different one of these *m* subsets as test data and the other *m*-1 subsets for training.

Evaluation of Classifiers

Cross-validation

- Average the evaluation results of the m classifiers.
- Typically, $m = 5$ or $m = 10$.
- Obtains better estimate of classification performance than Hold Out validation.
- Appropriate also for small O.
- Increases training time roughly by a factor of m.

Evaluation of Classifiers

Evaluation Criteria

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

Evaluation of Classifiers

Classification Accuracy

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

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

- *Classification error*

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

Aggregates over all classes $c_i \in C$.

Not appropriate if minority class is most important.

Evaluation of Classifiers

Confusion Matrix

- Let $c_1 \in C$ be the *target (positive) class*, the union of all other classes the *contrasting (negative) class*.
- For the target 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

Evaluation of Classifiers

Precision and Recall

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

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

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

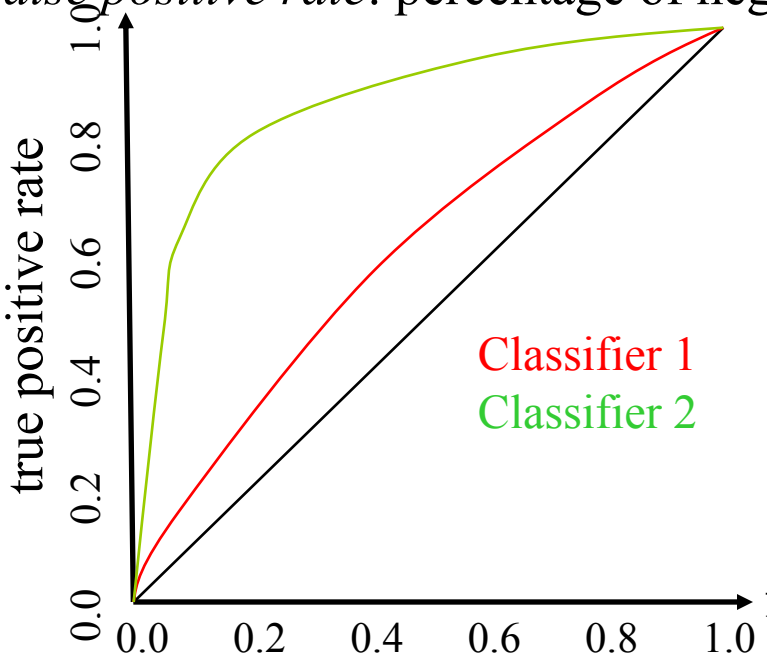
- There is a trade-off between precision and recall.
- Therefore, we also define a measure combining precision and recall:

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

Evaluation of Classifiers

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 vs. false positive rate
- *Area under ROC* as quantitative measure ideally = 1

Evaluation of Classifiers

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.

- Is difference in classification accuracy really significant?

Evaluation of Classifiers

Model Selection

- We measure the classification error on a (small) test dataset $O \subseteq D$.
- Questions:
  How to estimate the *true classification error* on the whole population *D*?
  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 *D*.
  (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).

Evaluation of Classifiers

Binomial distribution

- n independent 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:

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

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

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

Evaluation of Classifiers

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 (i.e. for Y/n):

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

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

Evaluation of Classifiers

Estimating the True Classification Error

- For n not too small and p neither close to 0 or to 1, the Binomial distribution can be approximated by a Normal distribution with the same mean and standard deviation.
- Random variable Y follows Normal distribution with mean m and standard deviation σ. Let y be the observed mean of Y. Then the true mean m 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:

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

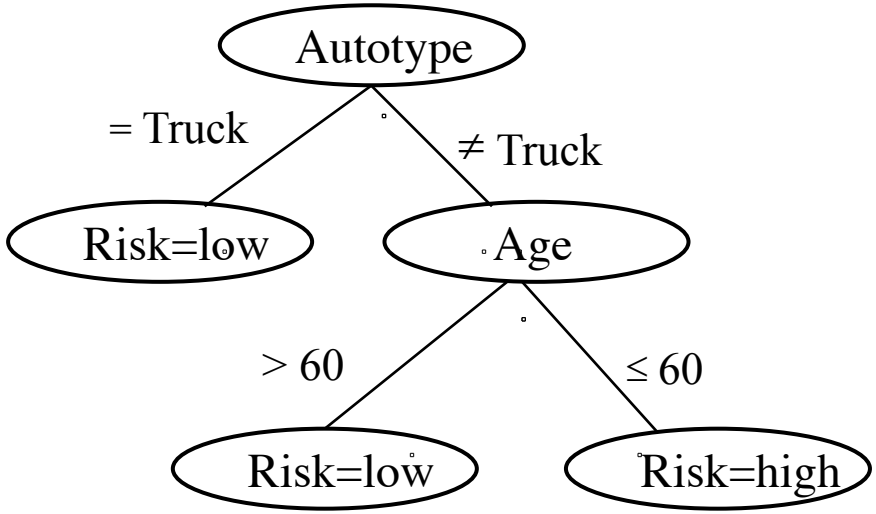
Interval size decreases with increasing n.

Interval size increases with increasing N.

Decision Trees

Introduction [Quinlan 1983]

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

Hierarchical structure

Decision Trees

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 parent node.
  - A leaf represents one of the classes of *C*.
  - The size of the tree is the number of its nodes.
- 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.

  Assignment of the object to class of the resulting leaf.

  ⇒ Unique path.

Decision Trees

Construction of Decision Trees

Base algorithm

- Initially, all training data objects belong to the root.
- Next attribute is selected and split (split strategy).
- Training data objects 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 objects of the node belong to the same class.

Decision Trees

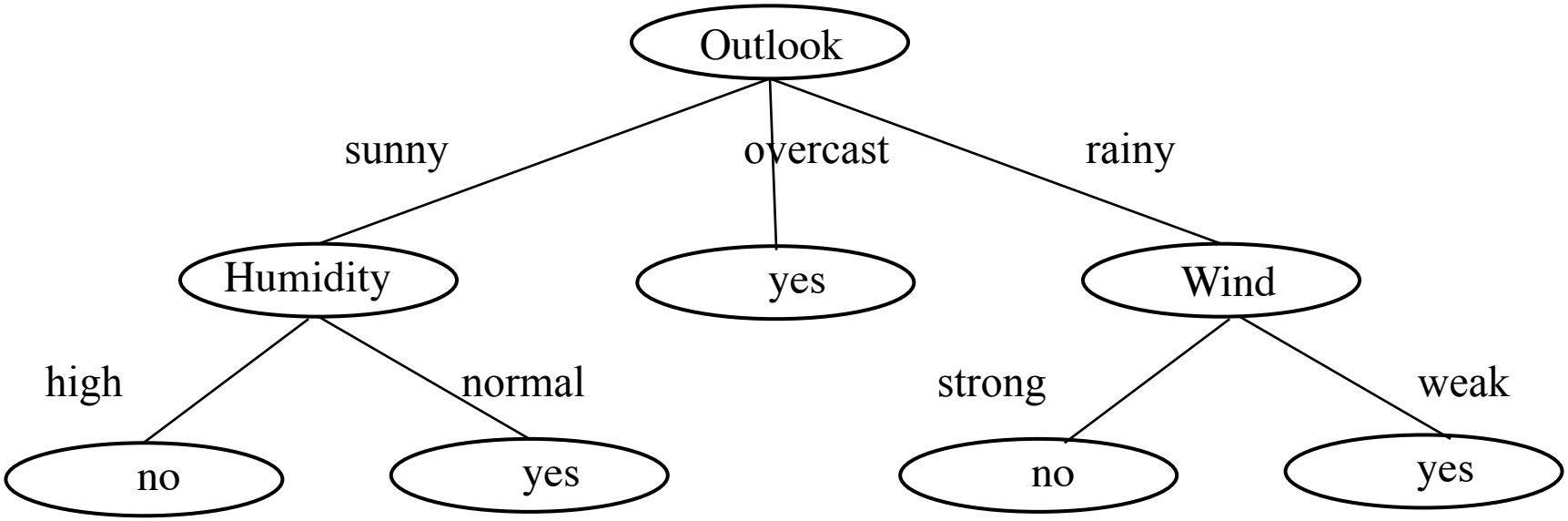
Example

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?

Decision Trees

Example

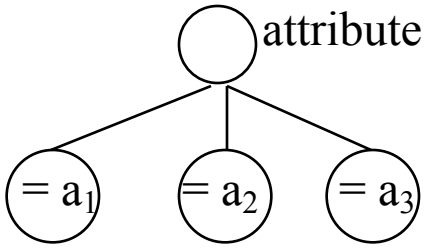


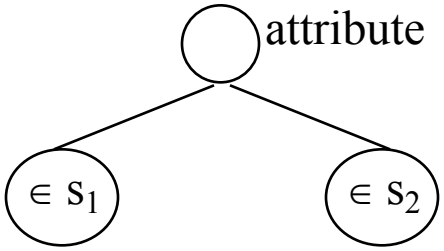
Decision Trees

Types of Splits

Categorical attributes

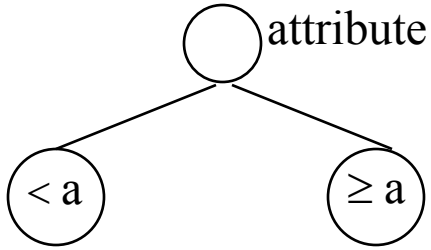
- Conditions of the form „*attribute = a*“ or „*attribute ∈ set*“ .
- Many possible subsets.





Numerical attributes

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



Decision Trees

Quality Measures for Splits

Given

- Set T of training data objects.

Wanted

- Measure of the impurity of any set S of labeled data w.r.t. class labels based on the p_i, the relative frequency of class c_i in S.
- Split of T in $T_1, T_2, \ldots, T_m$ *minimizing* this impurity measure. Split is a partitioning, i.e. a disjoint cover of T.

⇒ information gain, gini-index

Decision Trees

Information Gain

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

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

$entropy(T) = 0$, 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

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

Decision Trees

Gini-Index

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

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

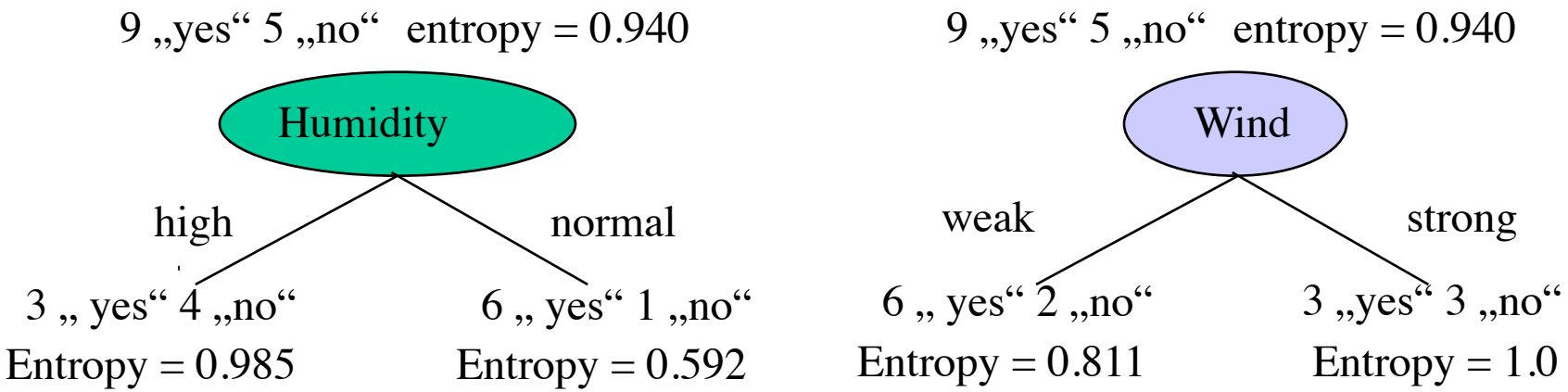
low gini index ⇔ low impurity,
high gini index ⇔ high impurity

- 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

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

Decision Trees

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.

Decision Trees

Generic Algorithm of Decision Tree Construction

Input

- Training dataset D
- Set of attributes A
- „Empty“ root node r

Method

- Recursive calls for each node to grow subtree rooted at this node.
- For the sake of clarity, assume that each attribute can be used only once as split attribute.
- Initial call `LearnDecisionTree(r, D, A).`

Decision Trees

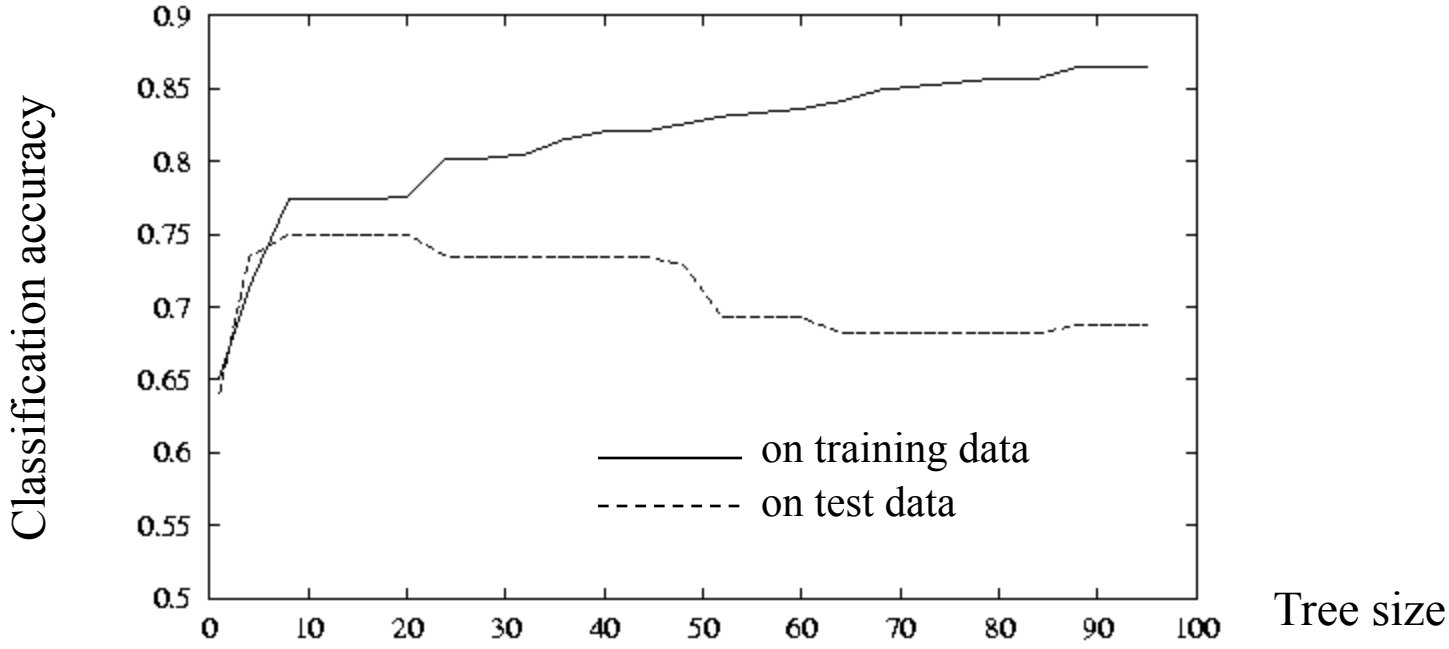
```
LearnDecisionTree(node n, dataset D, attributes A)
if termination condition(A,D) then
   n.label:=majority label in D;
else
   for all a in A do
      q(a):=quality of splitting D using a;
   b:= attribute a with maximum q(a);
   n.attribute := b;
   p:=number of partitions for attribute b;
   for i from 1 to p do
      Di:={d in D|d belongs to partition i of b}
      create new node ni as child of n;
      LearnDecisionTree(ni,Di,A-{b})
```

Decision Trees

Overfitting

Overfitting occurs if there are two decision trees *T* and *T'* with

- *T* has better accuracy than *T'* on the *training* data, but
- *T*' has a better accuracy than *T* on the *test* data.



Decision Trees

Approaches to Avoid 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 objects belonging to a leaf node.

minimum support >> 1

Decision Trees

Approaches to Avoid Overfitting

- Choice of appropriate minimum confidence

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

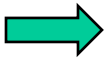
  

  Minimum confidence << 100%

  Leaves can also absorb noisy / erroneous training data objects.

- Subsequent pruning of the decision tree

  remove overfitting branches

  see next section

Decision Trees

Error Reduction-Pruning [Mitchell 1997]

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

Only applicable if enough labled data available.

Decision Trees

Minimal Cost Complexity Pruning

[Breiman 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.

Decision Trees

Notions

- *Size* $|T|$ of decision tree T: number of nodes.
- *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)

Decision Trees

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

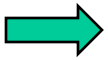
  ⇒ For $\alpha \geq \alpha_{crit}$ the subtree of node e should be pruned.

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

Decision Trees

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 $T(\alpha_1)$, $T(\alpha_2)$, . . ., $T(\alpha_m)$ performing l-fold cross-validation on the training data set.

Decision Trees

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	. . .	. . .	. . .	. . .

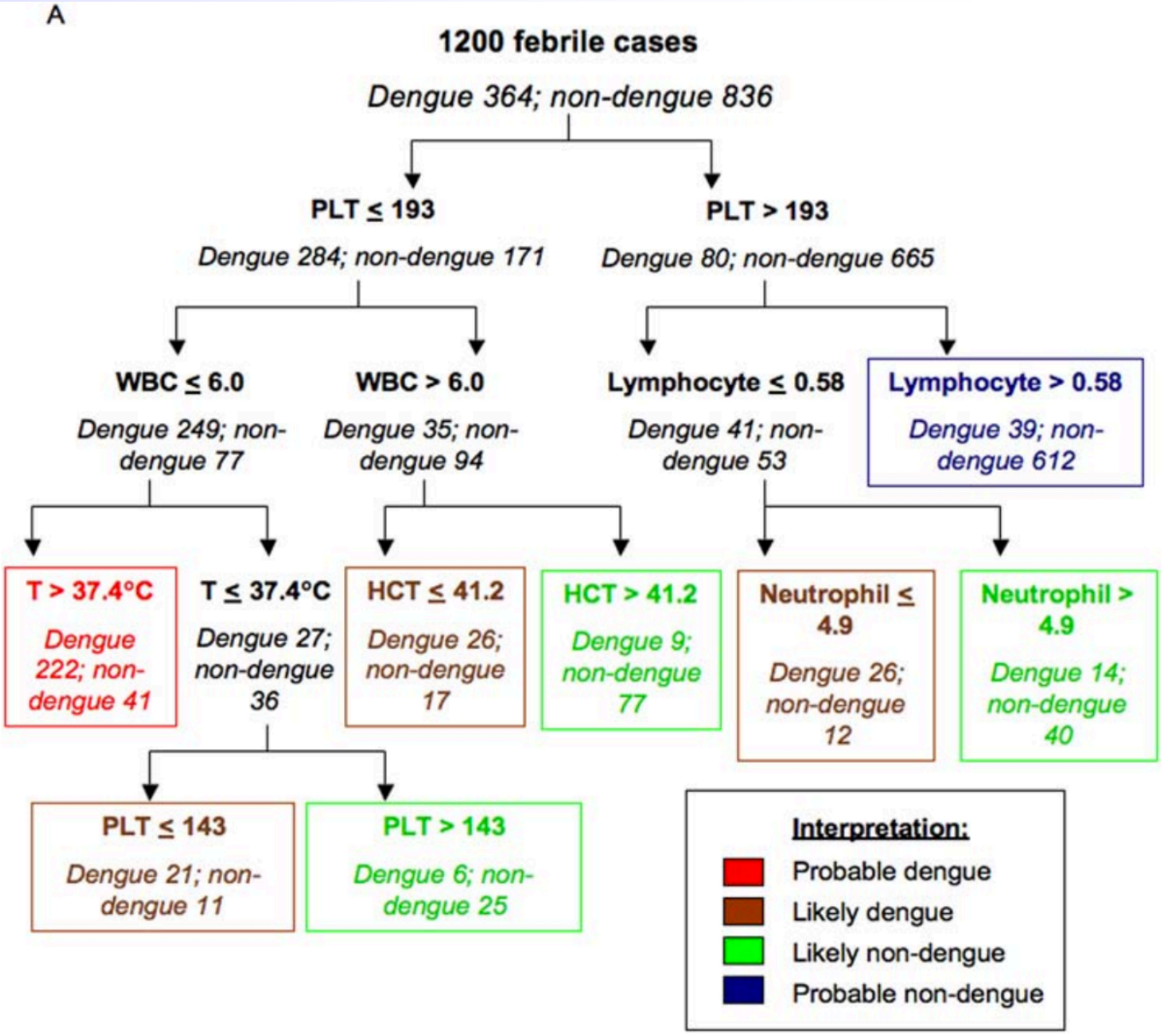
T_7 has the lowest estimated error and the lowest true (test) error.

Decision Trees

Example

Goal: early diagnosis of Dengue feaver.

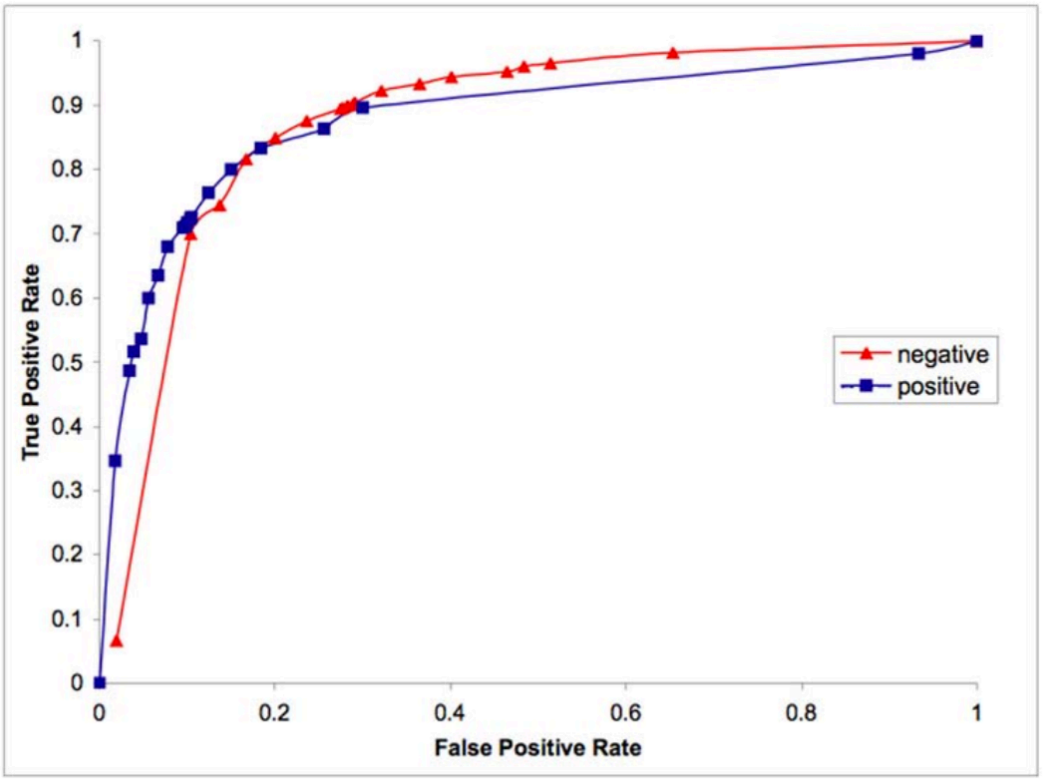
Use body temperature and the results of simple laboratory tests on blood samples as features.



Decision Trees

Example

ROC



Confusion matrix

B

Overall	Value (n=1200)
Total misclassifications	188
Overall error rate	15.7%
AUC negative	0.88
AUC positive	0.88

Sensitivity and specificity

		Predicted	
		Pos	Neg
Actual	Pos	259 (71.2%)	105 (28.8%)
	Neg	83 (9.9%)	753 (90.1%)

Decision Trees

Discussion

Pros

Cons

Bayesian Classification

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.

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

- 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

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

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

Bayesian Classification

Introduction

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

$P(h \mid D)$: 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 $P(D \backslash h)$.
- $P(h)$ 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.

Bayesian Classification

Bayesian Classifier

- When applying a learned hypothesis h to classify an object x, we use the following decision rule: $\underset{c_j \in C}{argmax}\, P(c_j \mid h)$
- h depends on the attribute values of x, i.e. $x_1, \ldots, x_d$.
- Therefore we determine $\underset{c_j \in C}{argmax}\ P(c_j \mid x_1, \ldots, x_d)$
- Applying Bayes theorem, we obtain

$$\underset{c_j \in C}{argmax}\ P(c_j \mid x_1, \ldots, x_d) = \underset{c_j \in C}{argmax}\ \frac{P(x_1, \ldots, x_d \mid c_j) \cdot P(c_j)}{P(x_1, \ldots, x_d)}$$

$$= \underset{c_j \in C}{argmax}\ P(x_1, \ldots, x_d \mid c_j) \cdot P(c_j)$$

Bayesian Classification

Naive Bayes Classifier [Domingos & Pazzani 1997]

- Estimate the $P(c_j)$ using the observed frequencies of the individual classes.
- How to estimate the $P(x_1, \ldots, x_d \mid c_j)$?
- Assumption:
  - Attribute values x_i are conditionally independent, given class c_j.
  - $P(x_i \mid c_j)$ are easier to estimate from the training data than $P(x_1, \ldots, x_d \mid c_j)$.

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

- Decision rule of the *Naive Bayes-Classifier*

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

Bayesian Classification

Naive Bayes Classifier

- How to estimate the $P(x_i | c_j)$?
- Maximum likelihood estimate: observed frequency.

$$P(x_i = a_j \mid C = c) = \frac{q(a_j, c)}{r(c)}$$

where $q(a_j, c)$ is the number of training examples of class c with value a_j in i-th feature

and $r(c)$ is the number of training examples in class c.

Bayesian Classification

Naive Bayes Classifier

- Problem if value of i-th feature in test case was not observed for class c in training data: $q(a_j, c) = 0$

  $\prod_{i=1}^{d} P(x_i \mid c) = 0$ even if the other features are very probable for class c.

- Solution: Laplacian smoothing

  add small α to numerator and normalize denominator.

$$P(x_i = a_j \mid C = c) = \frac{q(a_j, c) + \alpha}{r(c) + \alpha \cdot m_i}$$

where m_i is the number of distinct values of feature i

⇒ For $r(c) = 0$, all feature values have probability $1/m_i$.

Bayesian Classification

Bayesian Networks [Pearl 2000]

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

  ⇒ suboptimal classification accuracy

- Often, only some attributes are dependent, most are conditionally independent (given some class).
- Bayesian networks (Bayesian belief networks / Bayes nets)

  Allow you to specify potential dependencies and conditional independencies.
- Network respresents subjective, a-priori beliefs.
- Bayesian network specifies $P(x_1, \ldots, x_d \mid c_j)$.

Bayesian Classification

Bayesian Networks

• Directed graph with node = *random variable* (attribute) and
edge = *potential dependency.*

→ Graph is acyclic.

•For each node (random variable): conditional probability distribution given values of the parent variables.

• The parameters of these conditional probability distributions are the parameters of the Bayesian network that are learned during the training phase.

•Bayesian network specifies a factorization of the joint probability of all random variables *X*1, . .., *Xn*.

$$P(X_1,\ldots X_n) = \prod_{i=1}^{n} P(X_i \mid Pa(X_i))$$

where *Pa*(*Xi*) denotes the parents (direct predecessors) of *Xi.*

Bayesian Classification

Bayesian Networks

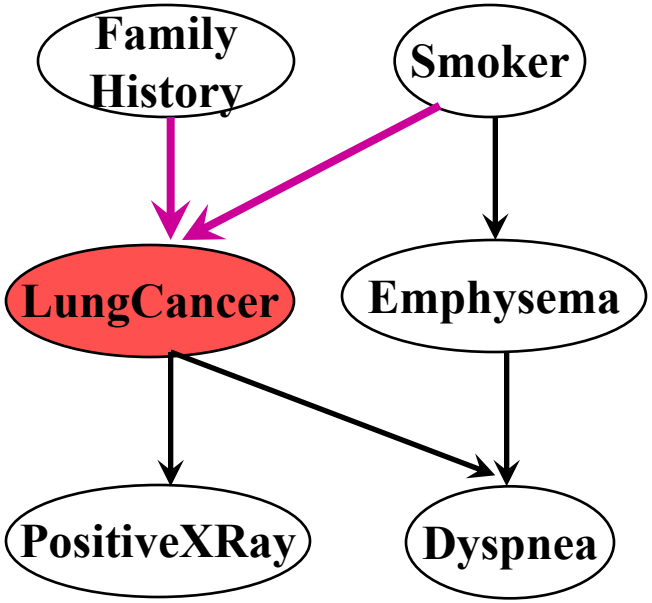
- Bayesian network does not specify dependencies between random variables.
- It specifies a set of conditional independence assumptions:
  every random variable is conditionally independent from all of its non-descendants given its parent variables.

Most classification algorithms learn correlations between features and class labels.

Bayesian network can represent *causal knowledge.*

Bayesian Classification

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.

Bayesian Classification

Training Bayesian Networks

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

Bayesian Classification

Learning the Structure of a Bayesian Network

- Greedy algorithm

  Start with a network of all nodes but no edges.

  In every iteration, add the edge that improves the network score the most.

  Terminate as soon as there is no more such edge.

- Network score

  Trade-off between training accuracy and network complexity.

  Too few edges: miss actual dependencies.

  Too many edges: create spurious dependencies.

- Bayesian score for network G

$$score(G:D) = \log P(D \mid G) + \log p(G)$$
$$= \log P(X_1, \ldots X_n \mid G) + \log p(G)$$

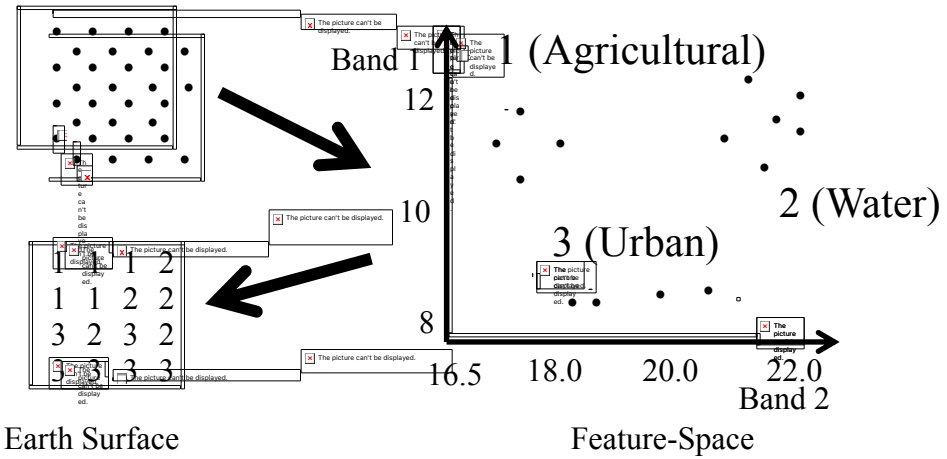
Bayesian Classification

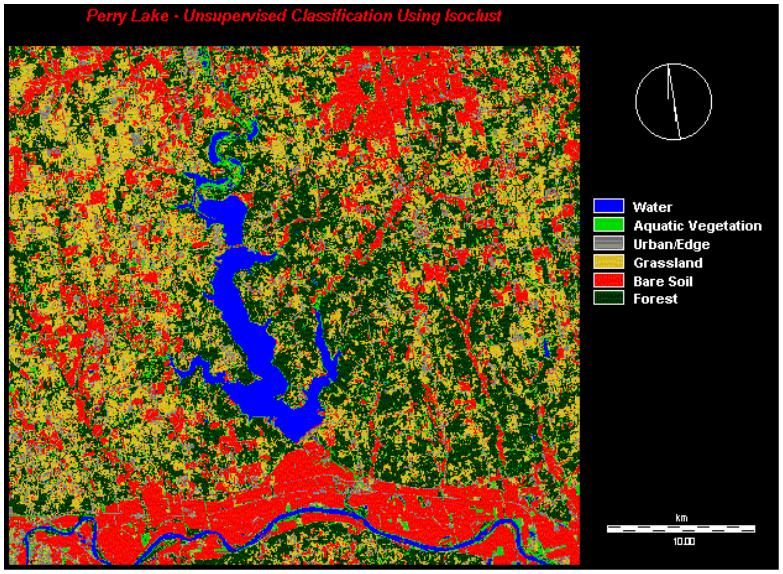
Interpretation of Raster Images

- *Automatic* interpretation of *d* raster images of a given region

  for each pixel: a *d*-dimensional feature vector of grey values $(x_1, \ldots, x_d)$.

- Assumption: different kinds of landuse exhibit characteristic behaviors of reflection / emission.

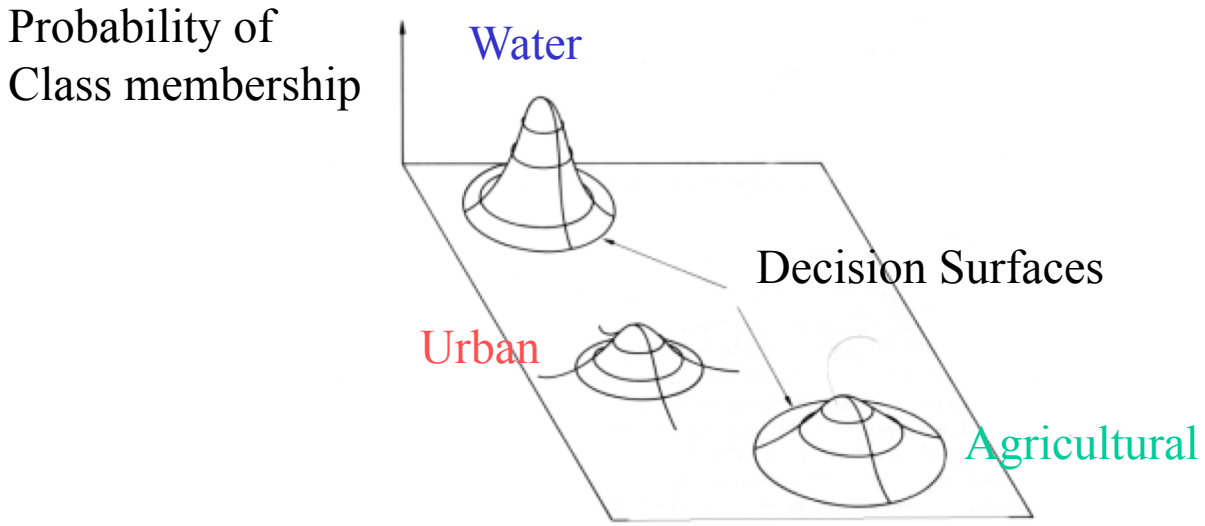




Bayesian Classification

Interpretation of Raster Images

- Application of the (general, Non-Naive) Bayes classifier.
- Estimate the $P(x_1, \ldots, x_d \mid c_j)$ without assuming conditional independency.
- Assume a d-dimensional Normal distribution of the feature vectors of a given class.



Bayesian Classification

Method

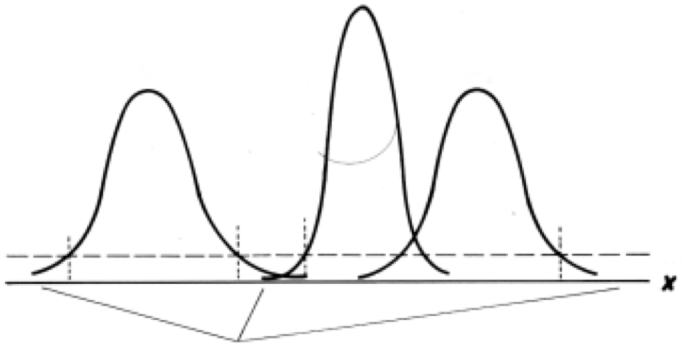
- Estimate from the training data

  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 with the decision rule

  - Likelihood for the chosen class very small.
  - Likelihood for several classes similar.



Bayesian Classification

Discussion

Pros

Cons

Logistic Regression

Introduction [Walker & Duncan 1967]

- Probabilistic classifier
  probabilistic model of the relationship between features (feature variables) and class label (class variable).
- Goal: learn $P(c_j \mid x_1, \ldots, x_d)$.
- Naïve Bayes classifier models the conditional probability distribution of features, given the class.

Generative model

- Logistic regression directly models the relationship between features and class label.

Discriminative model

Logistic Regression

Model

- Binary class variable $C \in \{-1, +1\}$
- Feature variables $X = (x_1, \ldots, x_d)$
- Parameters $\Theta = (\theta_0, \theta_1, \ldots, \theta_d)$

- θ_i: coefficient / weight of the *i*-th feature
- θ_0: offset parameter

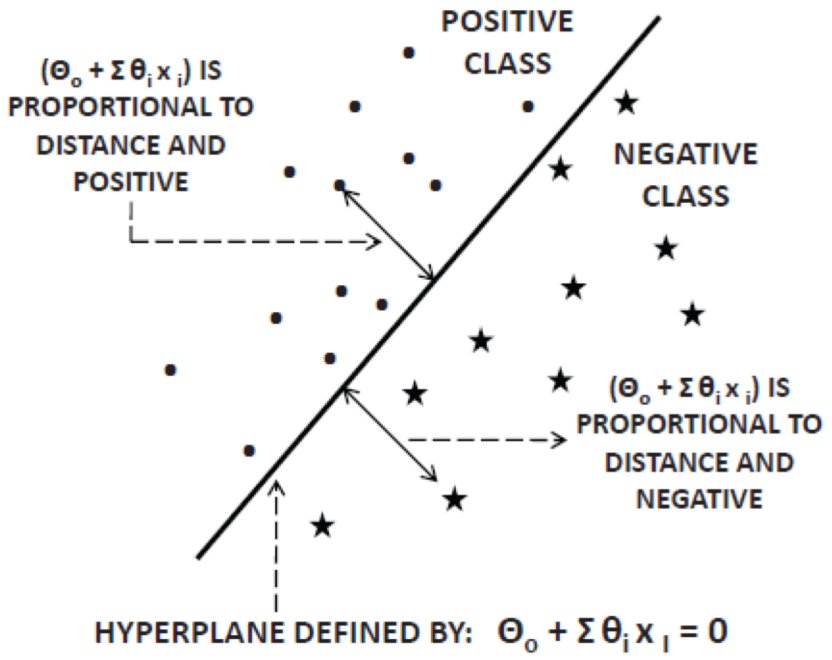
$$P(C = +1 \mid X) = \frac{1}{1 + e^{-(\theta_0 + \sum_{i=1}^{d} \theta_i x_i)}} \qquad P(C = -1 \mid X) = \frac{1}{1 + e^{(\theta_0 + \sum_{i=1}^{d} \theta_i x_i)}}$$

$S(t) = \frac{1}{1 + e^{-t}}$ is the sigmoid function, a special case of the logistic function

Logistic Regression

Model

Parameters Θ define a hyperplane separating the two classes: $\theta_0 + \sum_{i=1}^{d} \theta_i x_i = 0$



$\theta_0 + \sum_{i=1}^{d} \theta_i x_i$ proportional to distance from hyperplane

$\theta_0 + \sum_{i=1}^{d} \theta_i x_i > 0$ probability for positive class greater than for negative class

Logistic Regression

Parameter Learning

- Maximum likelihood approach

  Learn parameters Θ that maximize the likelihood of the training data.

$$L(\Theta) = \prod_{X \in D_+} \frac{1}{1 + e^{-(\theta_0 + \sum_{i=1}^{d} \theta_i \cdot x_i)}} \cdot \prod_{X \in D_-} \frac{1}{1 + e^{(\theta_0 + \sum_{i=1}^{d} \theta_i \cdot x_i)}}$$

  D_+ : positive training examples, D_- : negative training examples

- For numerical convenience, maximize log likelihood:

$$LL(\Theta) = -\sum_{X \in D_+} \log(1 + e^{-(\theta_0 + \sum_{i=1}^{d} \theta_i \cdot x_i)}) - \sum_{X \in D_-} \log(1 + e^{(\theta_0 + \sum_{i=1}^{d} \theta_i \cdot x_i)})$$

Logistic Regression

Parameter Learning

Gradient ascent method for optimization

$$\nabla LL(\Theta) = \left(\frac{\partial LL(\Theta)}{\partial \theta_0}, \ldots, \frac{\partial LL(\Theta)}{\partial \theta_d} \right)$$

$$\frac{\partial LL(\Theta)}{\partial \theta_i} = \sum_{X \in D_+} \frac{x_i}{1 + e^{(\theta_0 + \sum_{i=1}^{d} \theta_i \cdot x_i)}} - \sum_{X \in D_-} \frac{x_i}{1 + e^{-(\theta_0 + \sum_{i=1}^{d} \theta_i \cdot x_i)}}$$

$$= \sum_{X \in D_+} P(X \in D_-)\, x_i - \sum_{X \in D_-} P(X \in D_+)\, x_i$$

$$= \sum_{X \in D_+} P(Mistake\ on\ X)\, x_i - \sum_{X \in D_-} P(Mistake\ on\ X)\, x_i$$

⇨ Use mistakes of current model to identify direction of steepest ascent.

Logistic Regression

Parameter Learning

- In every step $\theta_i \leftarrow \theta_i + \alpha \frac{\partial LL(\Theta)}{\partial \theta_i}$

  α : *step size*

- Global optimum found, since likelihood function is concave.
- To avoid overfitting, add regularization term to likelihood function, e.g.

$$-\lambda \cdot \sum_{i=1}^{d} \frac{\theta_i^2}{2}, \text{ where } \lambda \text{ is a balancing parameter}$$

- Add the following term to the derivative $\frac{\partial LL(\Theta)}{\partial \theta_i}$

$$-\lambda \cdot \theta_i$$

Logistic Regression

Discussion

Pros

Cons

Nearest-Neighbor Classification

Motivation

- Bayesian classifier assuming a *d*-dimensional Normal distribution of features

  Requires estimates for μ_i and Σ_i for each class.

  Estimate for μ_i needs much less training data.

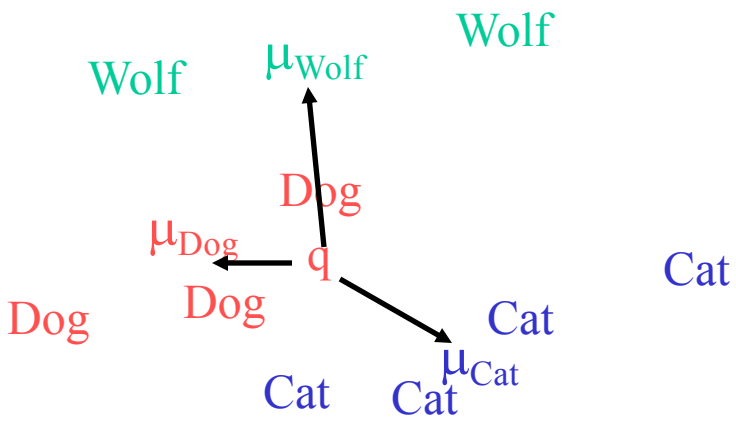
- Goal

  Classifier using only the mean vectors per class.

  ⇒ Nearest-neighbor classifier

Nearest-Neighbor Classification

Example



Cat

Classifier:

q is a dog!

Lazy learning method, does not learn a model.

Related to Case-Based Reasoning.

Nearest-Neighbor Classification

Overview [Altman 1992]

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 according to their distance.

Nearest-Neighbor Classification

Concepts

- Distance function

  defines similarity (dissimilarity) for pairs of objects.

- *k*: number of neighbors considered.

- *Decision Set*

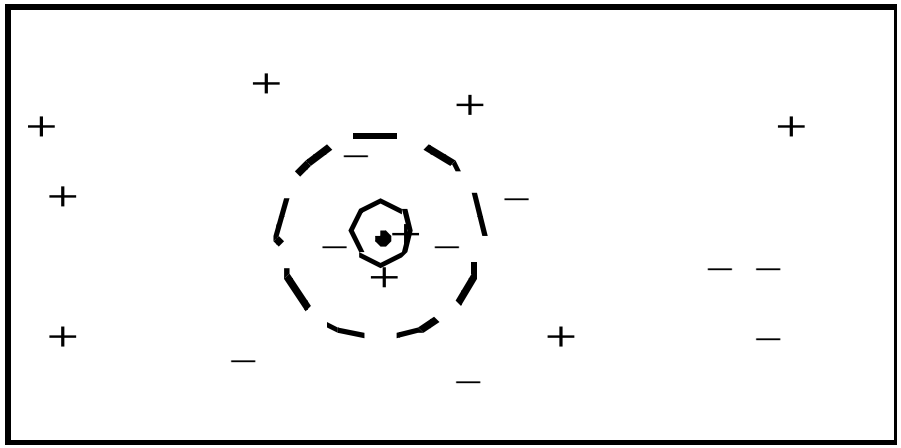
  set of *k*-nearest neighbors considered for classification.

- *Decision rule*

  How to determine the class of the unseen object from the classes of the decision set?

Nearest-Neighbor Classification

Example



classes „+“ and „-“

Decision set for k = 1

Decision set for k = 5

Uniform weight for the decision set

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

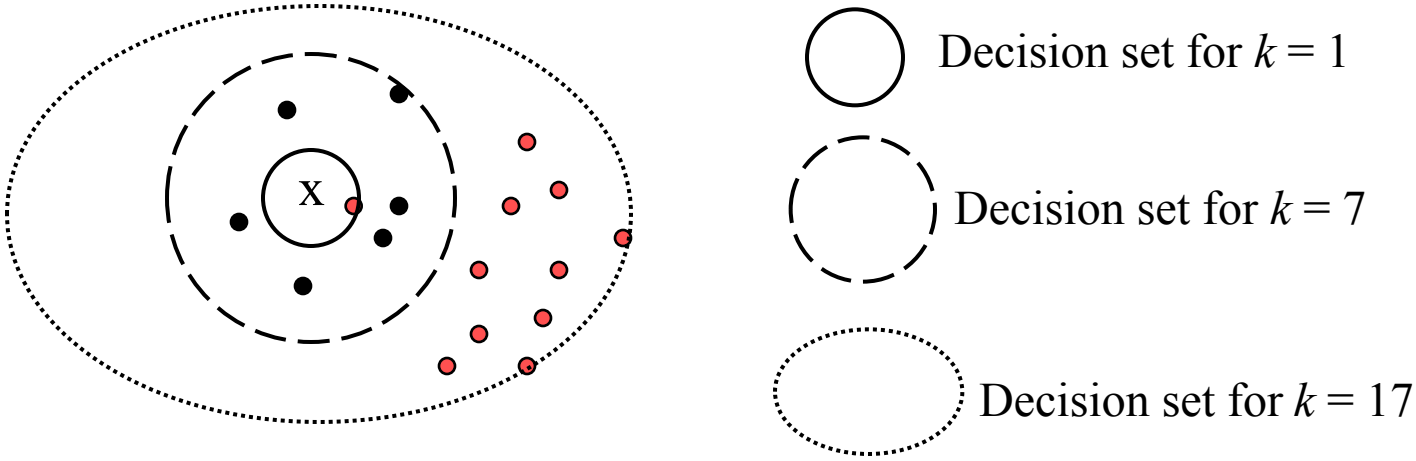
Inverse squared distance as weight for the decision set

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

Nearest-Neighbor Classification

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 $1 << k < 10$.



x: to be classified

Nearest-Neighbor Classification

Decision Rule

Standard rule

Choose the majority class within the decision set.

Weighted decision rule

Weight the classes of the decision set:

- By distance (weight inversely proportional to distance)

  close neighbors have higher weight than more distant neighbors

Nearest-Neighbor Classification

Decision Rule

Weight the classes of the decision set:

- By class distribution (weight inversely proportional to class frequency)
- Imbalanced (skewed) class distribution
  one dominant majority class, one infrequent minority class

  minority class is often more important

  e.g. cancer in classification of patients as cancer vs. normal

  or fraud in classification of transactions as fraud vs. normal
- Example

  class A: 95 %, class B 5 %

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

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

Nearest-Neighbor Classification

Index Support for k-Nearest-Neighbor Queries

- In Nearest Neighbor classification, the test phase is computationally expensive.
- Without index support, the runtime is O(n).
- Index can reduce it to O(log n).
- Balanced tree structure (such as X-tree or M-tree)

  multi-dimensioal generalization of (one-dimensional) B-trees.
- B-tree manages one-dimensional intervals of the atribute domain.

  Secondary storage data structure.

  Nodes of the tree correspond to disk pages.
- Multi-dimensional index structure manages hyper-rectangles (bounding boxes).

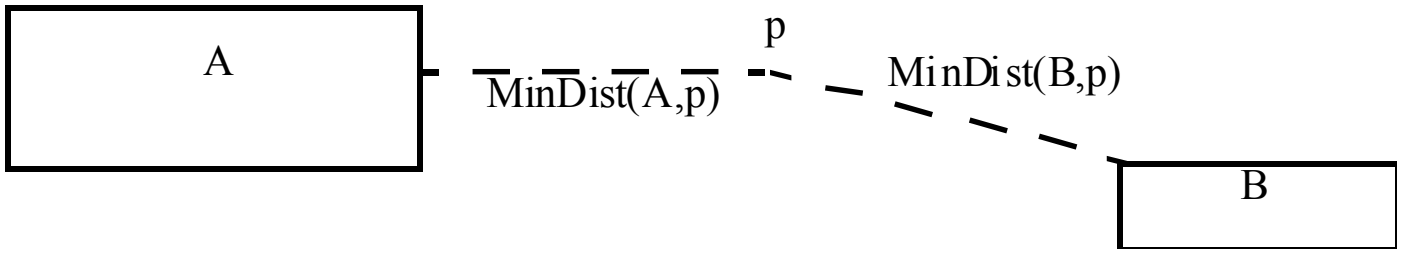
Nearest-Neighbor Classification

Index Support for k-Nearest-Neighbor Queries

- Query point p
- PartitionList

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

  Nearest neighbor of p in the data pages read so far.



Nearest-Neighbor Classification

Index Support for k-Nearest-Neighbor Queries

- MinDist (A,p) is a lower bound of dist(q,p) for all points q in BB A.
- Remove all BBs from PartitionList that have a larger MinDist 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.
- Algorithm is optimal: does not read any unnecessary disk pages.
- Query processing (typically) limited to a few paths of the index structure.

Runtime O(log *n*) for „not too many“ attributes.

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

Nearest-Neighbor Classification

Discussion

Pros

Cons

Support Vector Machines

Introduction [Burges 1998]

Input

Training set $S = \{(x_1, y_1), ..., (x_n, y_n)\}$

of objects $x_i \in X$ and their known classes $y_i \in \{-1, +1\}$.

Output

Classifier

$$K: \; A_1 \times A_2 \times \ldots A_d \rightarrow \{-1, +1\}$$

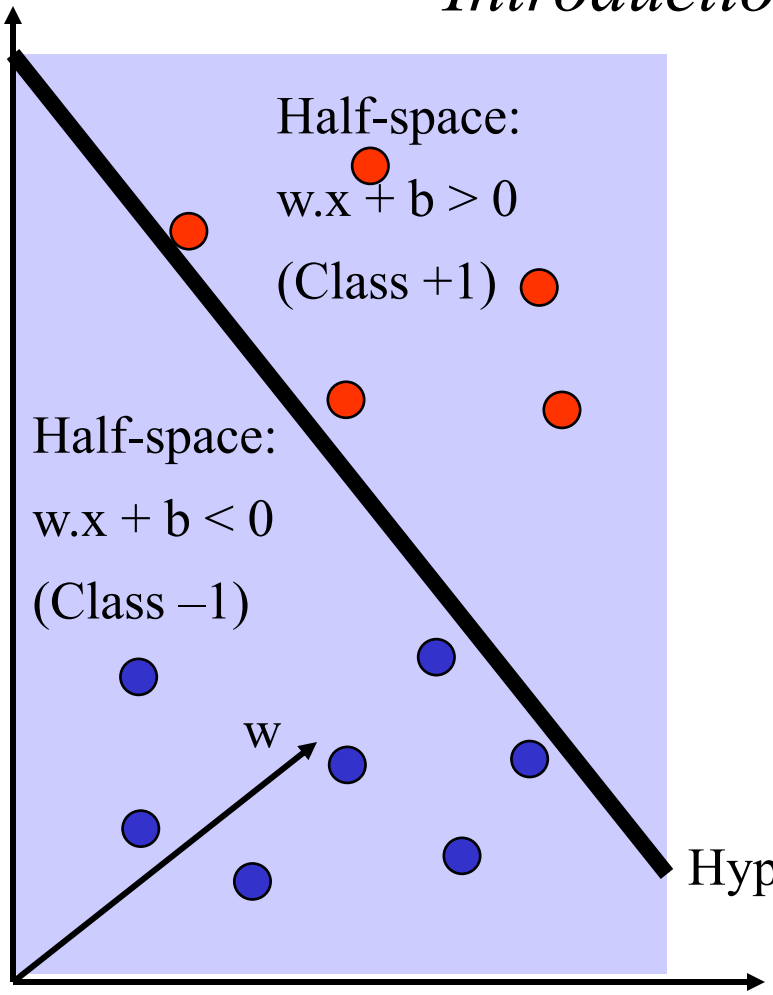
Goal

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

Two-class problem

Support Vector Machines

Introduction



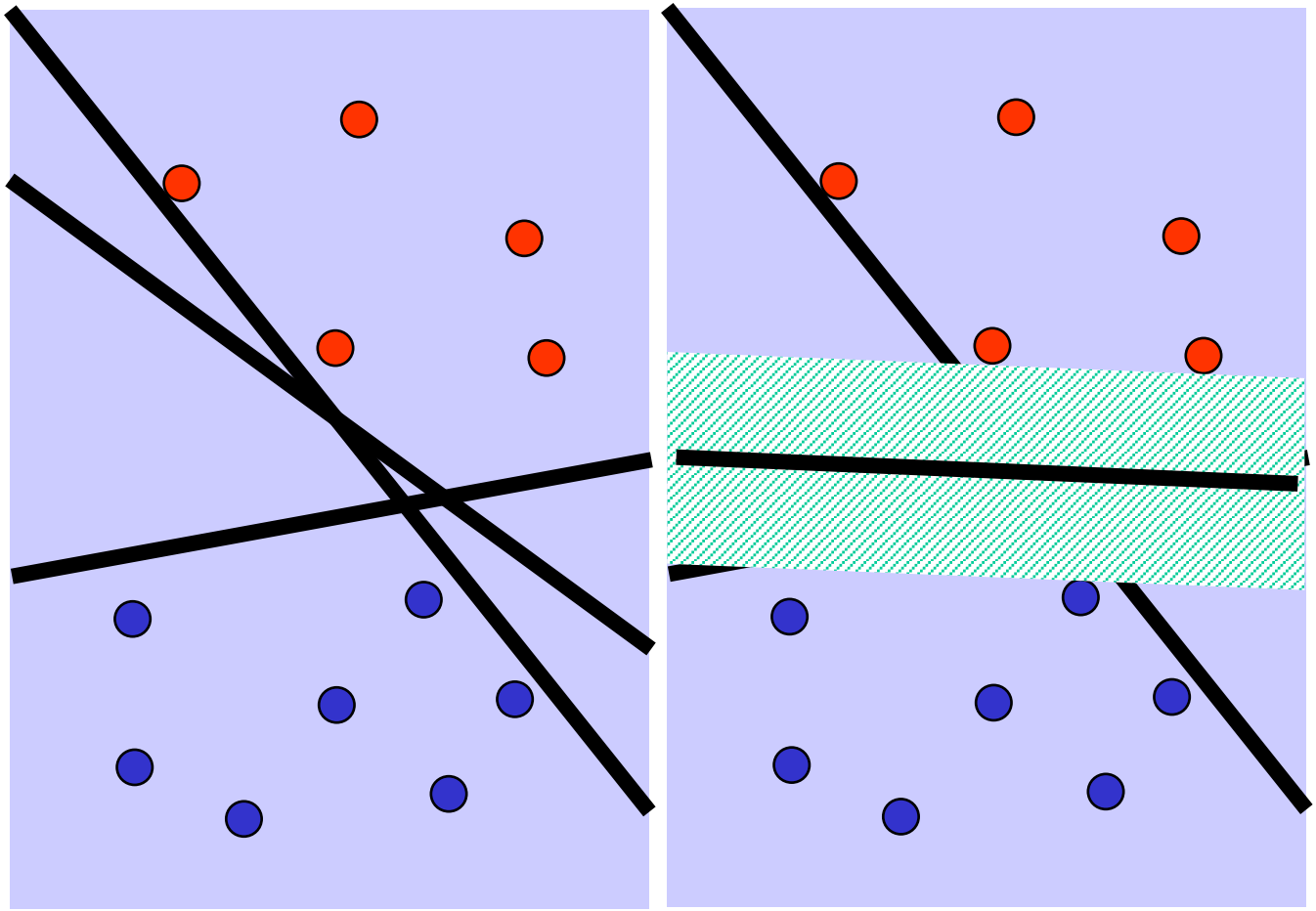
- Classification based on the *sign* of the *decision function:*

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

- "." denotes the inner product of two vectors.

Support Vector Machines

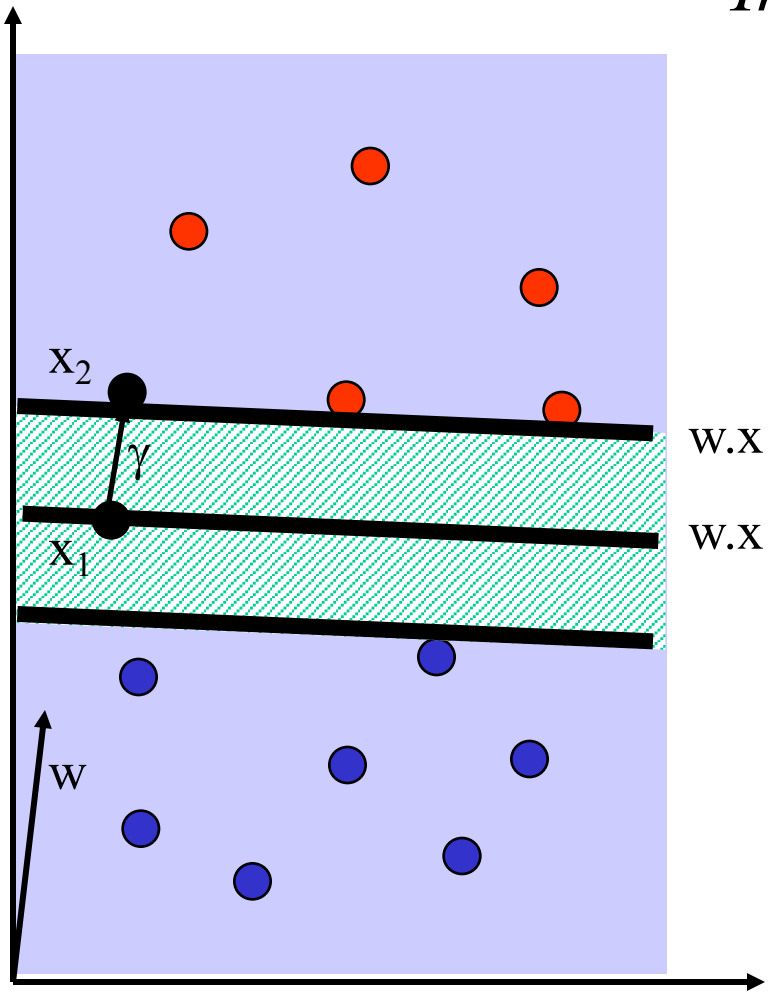
Introduction



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

Support Vector Machines

Introduction



w.x + b = +1

w.x + b = 0

w. x_1 + b = 0

w. x_2 + b = 1

w. (x_2 - x_1) = 1

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

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

γ: margin

Support Vector Machines

Method

Problem

- Maximize margin, i.e. minimize $||w||^2$.
- Under the constraints $\forall i = 1,\ldots,n:\ y_i(w.x_i + b) - 1 \geq 0$.

Dual problem

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

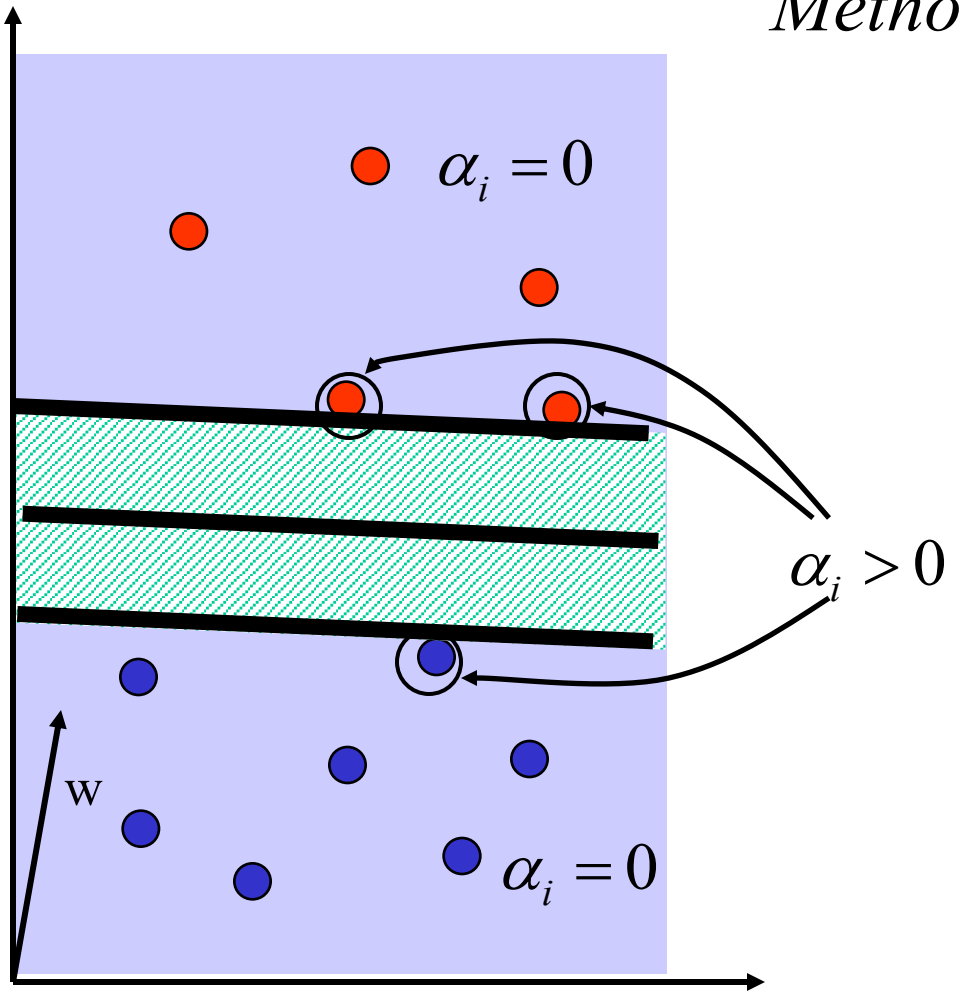
$$L(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \cdot \alpha_j \cdot y_i \cdot y_j \cdot x_i . x_j$$

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

Quadratic programming problem

Support Vector Machines

Method

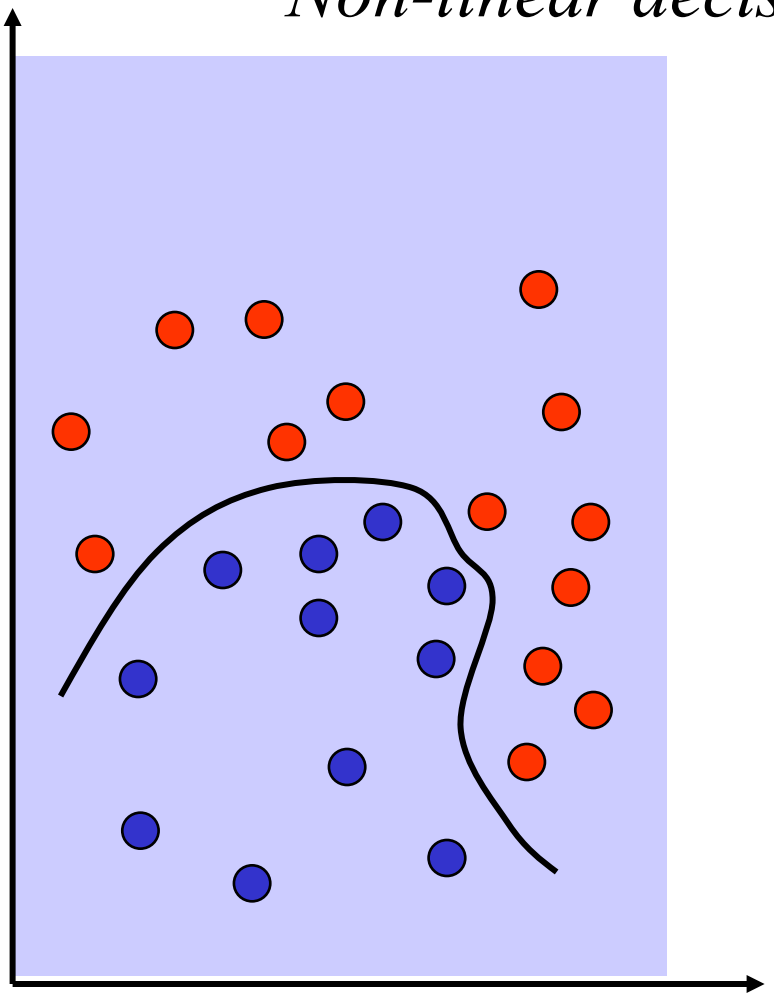


- Only training objects with $\alpha_i > 0$ contribute to w.
- These training objects are the *support vectors*.

Typically, number of support vectors << n.

Support Vector Machines

Non-linear decision surfaces



There are cases where classes cannot be separated by any linear function (hyperplane).

Two approaches

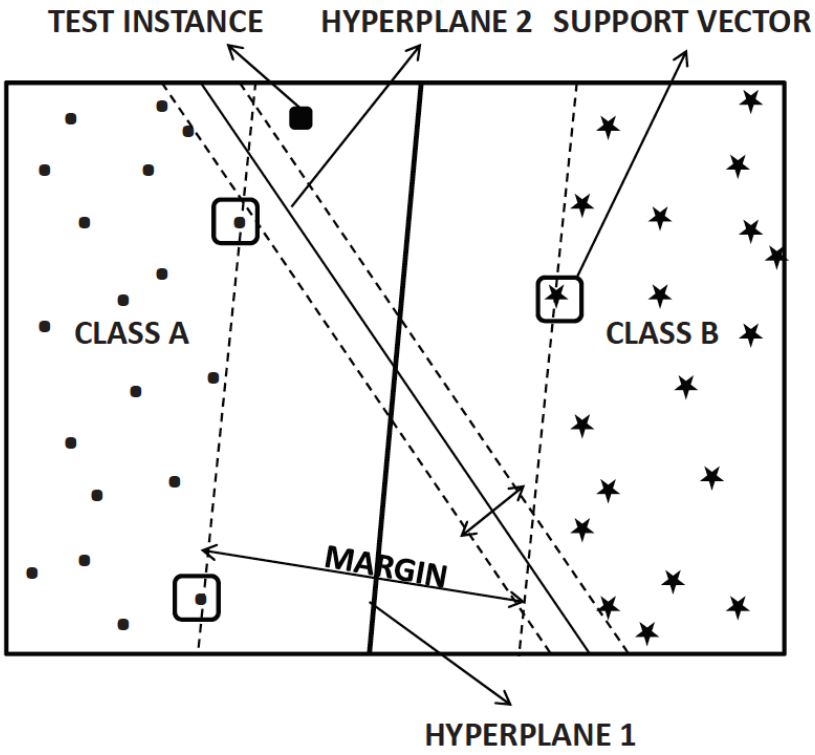
- Allow classification errors on training dataset.
- Map data into high-dimensional space where classes are linearly separable.

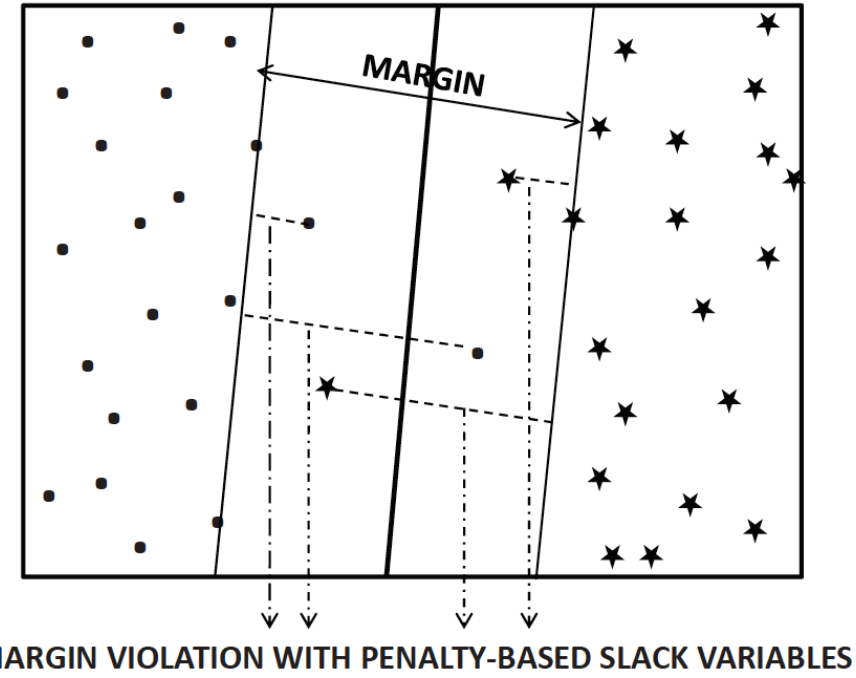
Support Vector Machines

Soft Support Vector Machines

Idea

- So far, we have considered the constraints as hard.
- Now we consider constraints as soft and add a penalty for violations.





Support Vector Machines

Soft Support Vector Machines

Soft constraints

- Instead of

$$\forall i \text{ with } y_i = 1 : w.x_i + b \geq 1$$

$$\forall i \text{ with } y_i = -1 : w.x_i + b \leq -1$$

- Now

$$\forall i \text{ with } y_i = 1 : w.x_i + b \geq 1 - \xi_i$$

$$\forall i \text{ with } y_i = -1 : w.x_i + b \leq -1 + \xi_i$$

$$\text{where } \xi_i \geq 0$$

New objective

$$\text{minimize } \|w\|^2 + C\sum_{i=1}^{n} \xi_i^r \text{ where } \xi_i \geq 0$$

Support Vector Machines

Soft Support Vector Machines

Parameters

- C
- r: often set to 1(hinge loss)

For r=1, optimization using Lagrangian methods

Unconstrained optimization

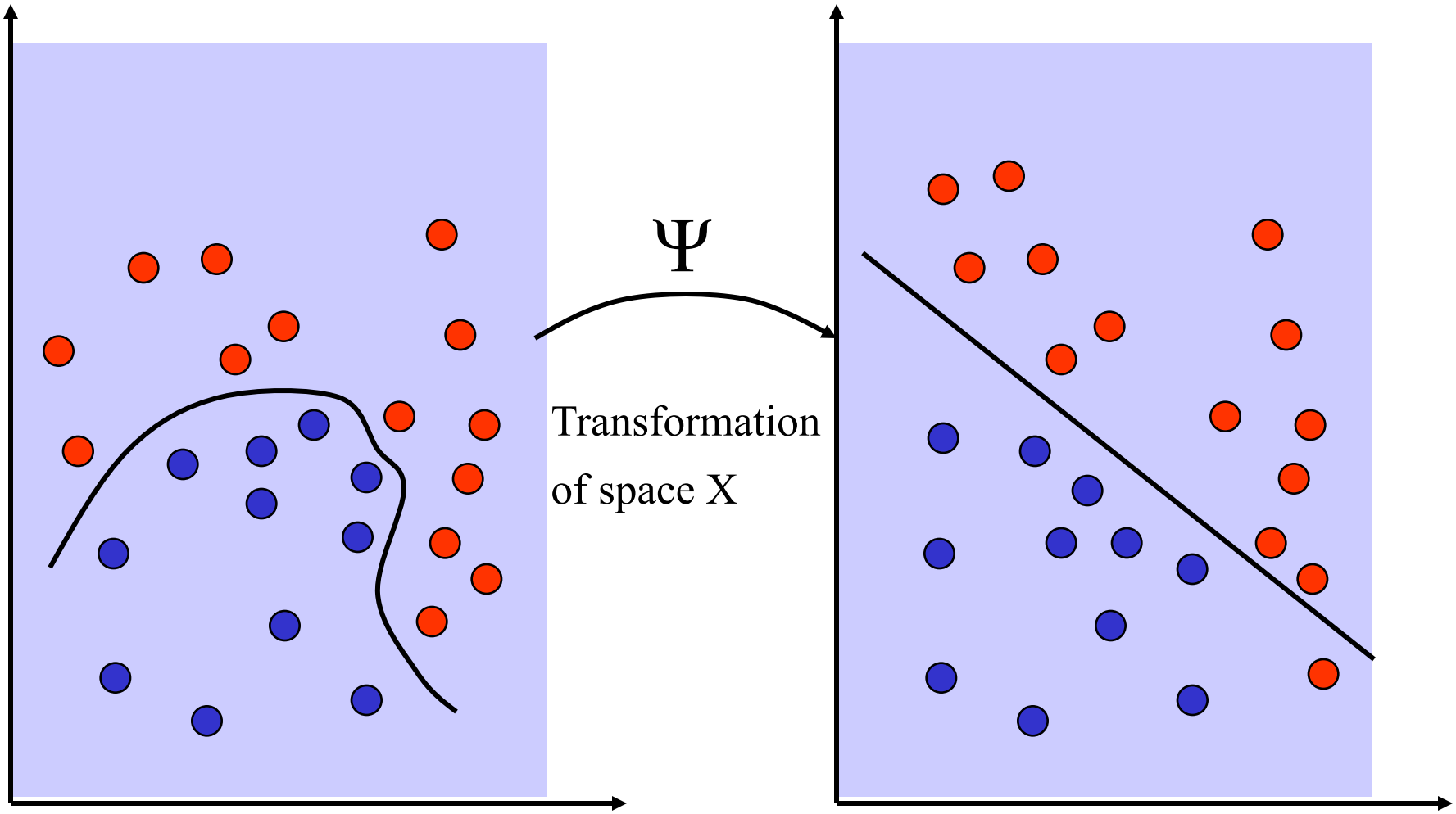
- For r=1, can replace the constrained optimization problem by the following unconstrained optimization problem:

$$\text{minimize } \|w\|^2 + C\sum_{i=1}^{n} \max\{0, 1 - y_i(w.x_i + b)\}$$

- Can be solved using gradient descent.

Support Vector Machines

The Kernel Trick



Support Vector Machines

The Kernel Trick

- Objective function $L(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \cdot \alpha_j \cdot y_i \cdot y_j \cdot x_i . x_j$
- Map x into higher-dimensional feature space: $\Psi(x)$
- Replace $x_i \cdot x_j$ by $\Psi(x_i) \cdot \Psi(x_j)$.
- Explicit computation of $\Psi(x)$ is not necessary.
- *Kernel* of two objects

$$K(x, x') = \Psi(x).\Psi(x')$$

- New objective function

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

Support Vector Machines

Kernels

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

$$S = \begin{pmatrix} K(x_1,x_1)\ K(x_1,x_2)\cdots K(x_1,x_n) \\ \cdots \\ K(x_n,x_1)K(x_n,x_2)\cdots K(x_n,x_n) \end{pmatrix}$$

is a symmetric and positive semi-definite matrix.

- If $S = AA^T$ for some r-dimensional feature transformation A of the original x_i, then for any n -dimensional column vector V

$$V^T SV = (AV)^T AV \geq 0$$

i.e., S is positive semi-definite.

Support Vector Machines

Kernels

- Example

$$\Psi(x) = (x_1^2, x_2^2, \sqrt{2}x_1 x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1)$$

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

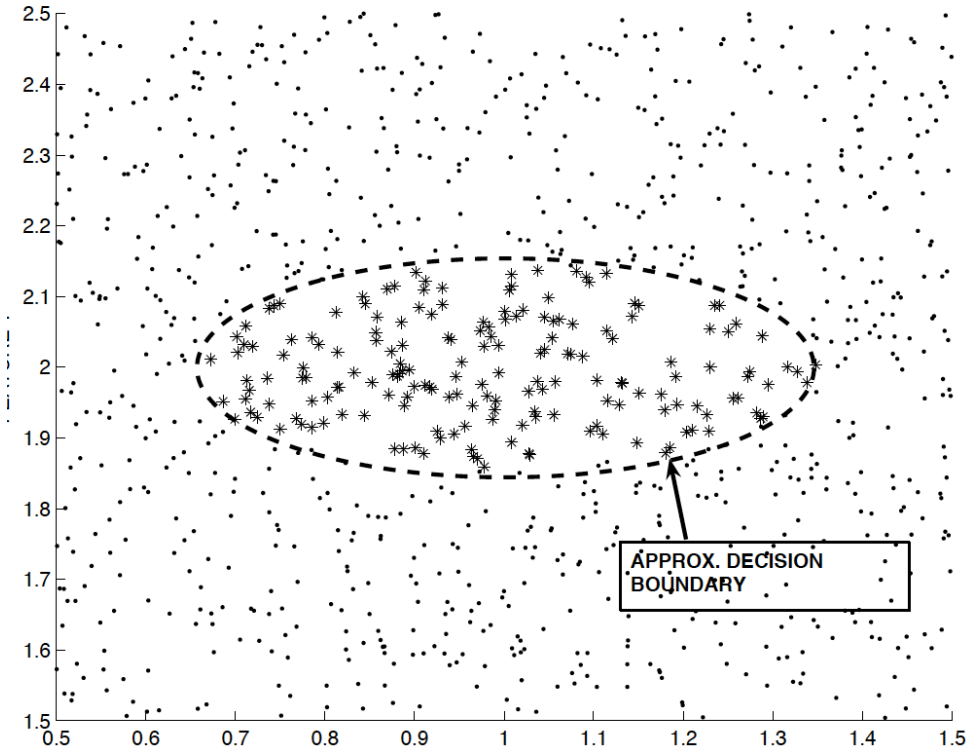
- Using appropriate kernels, arbitrary non-linear decision surfaces can be approximated.

Support Vector Machines

Example

- Consider two classes separated by the decision surface

$$8(x_1 - 1)^2 + 50(x_2 - 2)^2 = 1$$



Support Vector Machines

Example

- The following transformation into a 4-dimensional space makes the two classes linearly separable:

$$z_1 = x_1^2, z_2 = x_1$$

$$z_3 = x_2^2, z_4 = x_2$$

- The non-linear decision surface

$$8x_1^2 - 16x_1 + 50x_2^2 - 200x_2 + 207 = 0$$

can be expressed as a linear function in the transformed space:

$$8z_1 - 16z_2 + 50z_3 - 200z_4 + 207 = 0$$

Support Vector Machines

Common Kernels

- Gaussian Radial Basis Kernel

$$K(x_i, x_j) = e^{-\|x_i - x_j\|^2 / 2\sigma^2}$$

- Polynomial Kernel

$$K(x_i, x_j) = (x_i \cdot x_j + c)^h$$

- Sigmoid Kernel

$$K(x_i, x_j) = \tanh(\kappa x_i \cdot x_j - \delta)$$

Support Vector Machines

Kernels

- How to choose the kernel parameters?
  No general rules
- E.g., for Gaussian Radial Basis Kernel:
  smaller value of σ: greater ability to model complex surfaces
  but also risk of overfitting.
- The optimal values of kernel parameters depend not only on the shape of the decision surface, but also on the size of the training data set.

→ Kernel engineering

- Parameters are tuned on a validation dataset.
- Kernels are especially useful for unstructured data, such as text, images, sequences, and graphs.

Do not have to extract features.

Support Vector Machines

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

Support Vector Machines

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

  → Feature vectors of 1000 bits

Support Vector Machines

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.

Support Vector Machines

Discussion

Pros

Cons

Regression Analysis

Introduction

Commonality with classification

- In training phase, construct a model.
- In test phase, use model to predict unknown value.
- Major method for prediction is regression:
  - Simple and multiple regression,
  - Linear and non-linear regression.

Difference compared to classification

- Classification refers to prediction of categorical class label.
- Prediction models continuous-valued functions.

Regression Analysis

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

Regression Analysis

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:

$$y = Xa + e$$

Regression Analysis

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 can be solved e.g. using LU decomposition or SVD.

Runtime complexity $O(d^2 n + d^3)$.

Regression Analysis

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 (test) object xq*.

Weight the neighboring objects based on their distance to *xq*
distance-decreasing weight *K*.

Related to nearest neighbor classification.

→ Minimize the squared *local weighted* error.

Regression Analysis

Locally Weighted Regression

Local weighted error

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

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

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

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

Regression Analysis

Discussion

Pros

Cons

Ensemble Classifiers

Introduction

- Different classifiers make different predictions on the same test objects.
- Learn an ensemble of classifiers and aggregate their predictions to obtain a more robust prediction.
- Data-centered ensembles:
  - single base classifier (classification algorithm),
  - applied to multiple, different subsets of the training dataset.
- Model-centered ensembles:
  - multiple base classifiers (classification algorithms),
  - applied to the same training dataset.
- Hybrid ensembles:
  - multiple datasets,
  - multiple base classifiers.

Ensemble Classifiers

Generic ensemble classifier

```
EnsembleClassify(training data D, base algorithms
                 A1, . . ., Ar, test data T)
  j=1;
  repeat
     Select an algorithm Aj from A1, . . ., Ar;
     Create a training dataset Dj from D;
     Apply Aj to Dj to learn model Mj;
     j = j+1;
  until termination;
  return labels for each t in T by aggregating
     predictions from all models Mj;
```

Ensemble Classifiers

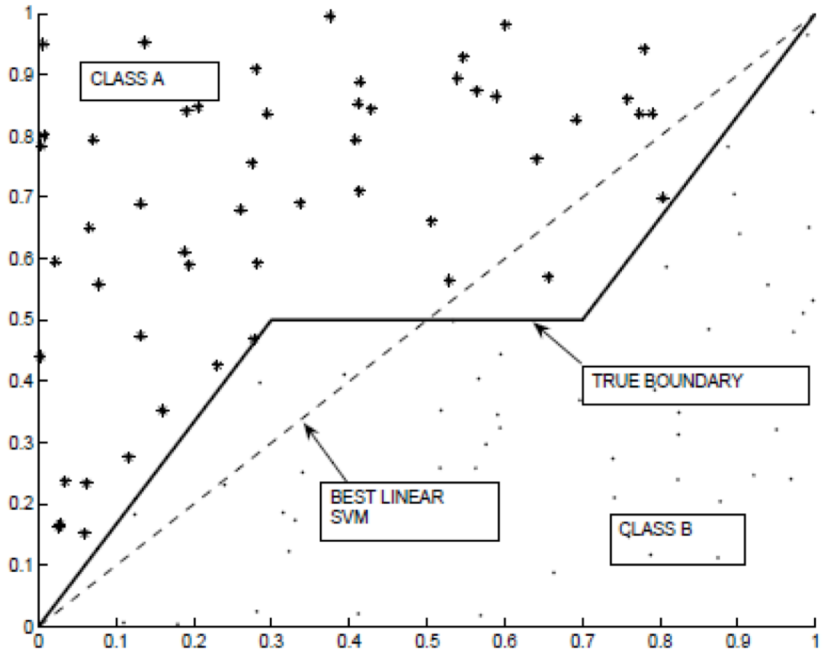
Why do Ensembles Work?

- Bias
  - Classifier makes modeling assumptions about the form of the decision boundary between classes.
  - Datasets violating these assumptions cannot be accurately classified.
- Variance
  - Random variations in the choice of training data mean that different models will be learnt, even when using the same classification algorithm.
  - Different models may disagree on some classifications.
  - Variance is closely related to overfitting.
- Ensemble classifiers reduce bias and/or variance.

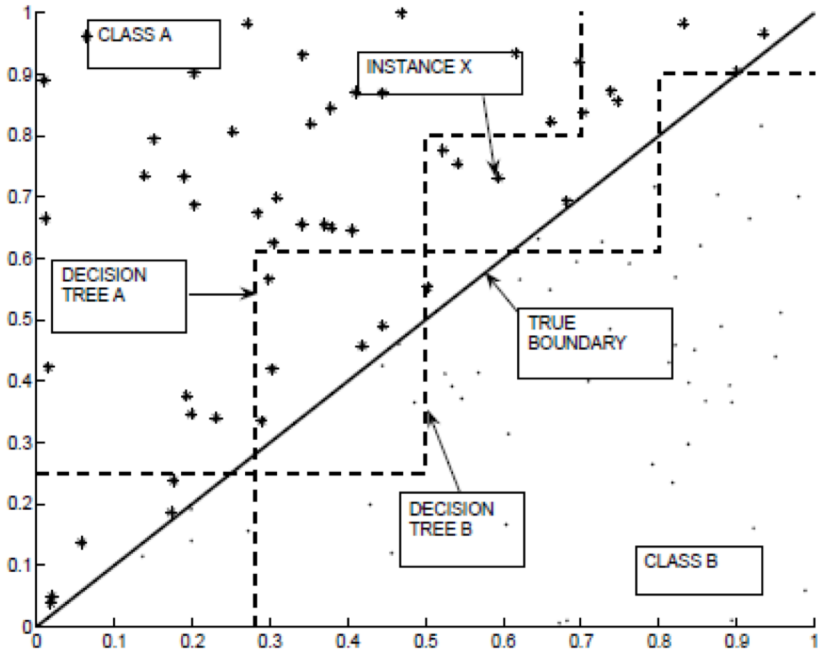
Trade-off between bias and variance:
More complex decision boundary reduces the bias but increases the variance.

Ensemble Classifiers

Bias and Variance



(a) bias



(b) variance

Ensemble Classifiers

Bagging [Breiman 1996]

- Goal: reduce the variance.
- Bagging = bootstrapped aggregating
- Basis

  If the variance of a prediction is σ^2 , then the variance of the average of k independent and identically distributed predictions is reduced to σ^2 / k.
- k different training datasets of size $|D|$ are sampled independently.
- Each training dataset is sampled uniformly from the original dataset D with replacement (bootstrapping).
- Predictions of the k classifiers are aggregated through majority vote.

Does not reduce the bias.

Does not increase it, either.

Ensemble Classifiers

Random Forests [Breiman 2001]

- Weakness of Bagging

  When using decision trees as base algorithms,

  the decision trees learnt from the k different training datasets are correlated, limiting the effect of variance reduction.
- The reason is that the split selections at the top tree levels are likely the same in the different training datasets.
- Idea

  Reduce correlation by introducing randomness into the split selection.
  Predictions of the k classifiers are aggregated through majority vote.
- Restrict the selection of a split attribute to a random subset of size q of the set of all attributes. $q \leq d$

Applicable only if d is large enough.

Ensemble Classifiers

Random Forests

- Choice of q
  larger values lead to more correlation,
  but also improve the accuracy of individual trees.
- Best trade-off

$$q = \log_2 d + 1$$

- Individual trees can be grown deep, i.e. up to 100% training accuracy.
- This increases their variance, but reduces their bias.

Random Forests typically outperform Bagging,
and often achieve top accuracy.

Ensemble Classifiers

Boosting [Breiman 1998]

- Goal: reduce the bias.
- Learn a sequence of classifiers using the same algorithm *A* on a weighted training data set.
- Weights in round $t+1$ depend on classification performance of the classifier of round t.
- Assumption: classification errors are caused by classifier bias.
- Increasing the weight of misclassified training examples will result in a new classifier that corrects for the bias on these particular examples.
- Number of rounds *T* is predefined or until the classifier achieves 100% accuracy on the training dataset.

Works best with simple, high-bias and low-variance base algorithms.

Ensemble Classifiers

AdaBoost

- Initialize all weights $W_1 = \frac{1}{n}$ *where* $n = |D|$
- Update weights

  for incorrectly classified examples $W_{t+1} = W_t \cdot e^{\alpha_t}$

  for correctly classified examples $W_{t+1} = W_t \cdot e^{-\alpha_t}$

  where $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$

  and ε_t is the fraction of incorrectly classified examples
- Aggregate prediction

$sign(\sum_{t=1}^{T} p_t \alpha_t)$ *where* $p_t \in \{+1, -1\}$ *is the prediction in round* t

Ensemble Classifiers

Stacking [Wolpert 1992]

- Training data D divided into two subsets D_A and D_B.
- Two levels of classification
  1) Train the (first-level) ensemble algorithm(s) on subset D_A to learn models $M_1, \ldots, M_k$.
     Apply the models $M_1, \ldots, M_k$ to the examples in subset D_B to create k new features for these examples, where the value of the i-th feature is the class label predicted by model M_i .
  2) Train a (second-level) classifier on subset D_B to train a model M that combines the first level classifiers.
- Second-level classifier uses the new and possibly also the original features.

Ensemble Classifiers

Stacking

- Example first level classifiers
  - Bagging on subset D_A.
  - Boosting on subset D_A.
  - Train k different base algorithms on D_A to create first level classifiers.
  - Train k different base algorithms on D_A, using different subsets of the set of all features, to create first level classifiers.
- Second-level classifier
  learns an classifier-specific way of aggregating the predictions of the first-level classifiers.

Can reduce both bias and variance.

References

[Altman 1992]

Altman, N. S.: An introduction to kernel and nearest-neighbor nonparametric regression. The American Statistician 46 (3): 1992.

[Breiman 1984]

Breiman, Leo: Classification and regression trees, 1984.

[Breiman 1996]

Breiman, L.: Bagging predictors, Machine Learning, 24(2), 1996.

[Breiman 1998]

Breiman, Leo: Arcing classifier. Ann. Statist. 26 (3), 1998.

[Breiman 2001]

Breiman, Leo: Random Forests. Machine Learning 45 (1), 2001.

[Burges 1998]

Christopher J. C. Burges: A Tutorial on Support Vector Machines for Pattern Recognition. Data Mining and Knowledge Discovery 2(2), 1998.

References

[Domingos & Pazzani 1997]

Domingos, Pedro; Pazzani, Michael: On the optimality of the simple Bayesian classifier under zero-one loss, Machine Learning, 1997.

[Leslie et al 2002]

Christina S. Leslie, Eleazar Eskin, William S. Noble: The Spectrum Kernel: A String Kernel for SVM Protein Classification. PSB 2002.

[Mitchell 1997]

Tom M. Mitchell: Machine learning. McGraw-Hill 1997.

[Pearl 2000]

Pearl, Judea: Causality: Models, Reasoning, and Inference, Cambridge University Press, 2000.

[Quinlan 1983]

R. Quinlan: Learning efficient classification procedures, Machine Learning: an artificial intelligence approach, Michalski, Carbonell & Mitchell (eds.), Morgan Kaufmann, 1983.

References

[Walker & Duncan 1967]

Walker, SH; Duncan, DB (1967). Estimation of the probability of an event as a function of several independent variables. Biometrika 54, 1967.

[Wolpert 1992]

Wolpert, D.: Stacked Generalization. Neural Networks, 5(2), 1992.

[Zien et al 2000]

Alexander Zien, Gunnar Raesch, Sebastian Mike, Bernhard Schoelkopf, Thomas Lengauer, Klaus-Robert Mueller: Engineering support vector machine kernels that recognize translation initiation sites. Bioinformatics 16 (9), 2000.