Ludwig-Maximilians-Universität München
Lehrstuhl für Datenbanksysteme und Data Mining
Prof. Dr. Gabriel Marques Tavares

Data Mining Algorithms 1

(Knowledge Discovery and Data Mining 1)

Winter Semester 2024/25

Supervised Learning

Agenda

1. Supervised Learning

1.1 Introduction: Classification

1.2 Bayesian Classifiers

1.3 Linear Discriminant Functions

1.4 Support Vector Machines

1.5 Kernel Methods

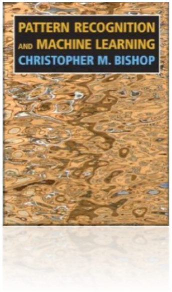
1.6 Decision Tree Classifiers

1.7 Nearest Neighbor Classifiers

1.8 Ensemble Classification

Additional Literature for this Chapter

Christopher M. Bishop: *Pattern Recognition and Machine Learning*. Springer, Berlin 2006.



Introduction: Toy Example Car Insurance

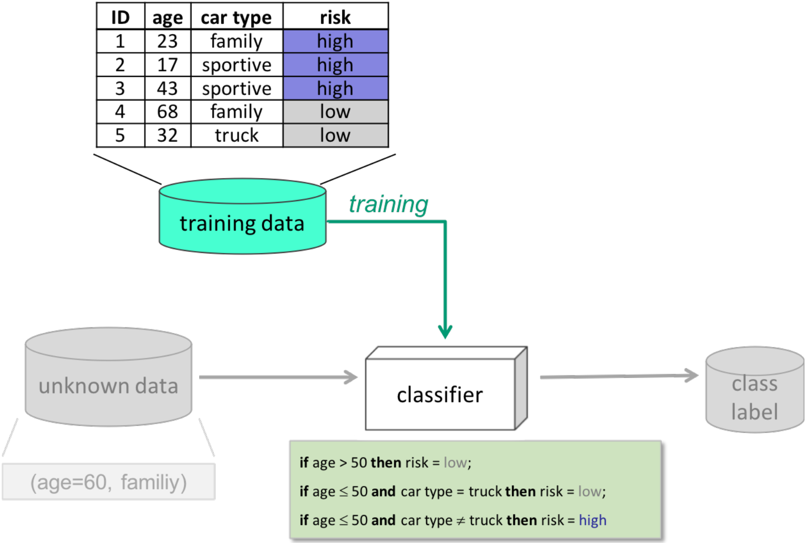
- Training data

age	car_type	max_speed	risk
23	family	180	high
17	sportive	240	high
43	sportive	246	high
68	family	183	low
32	truck	110	low

- Simple classifier

if age $>$ 50 **then** risk $=$ low
if age $\leq$ 50 **and** car type $=$ truck **then** risk $=$ low
if age $\leq$ 50 **and** car type $\neq$ truck **then** risk $=$ high

Classification: Training Phase (Model Construction)



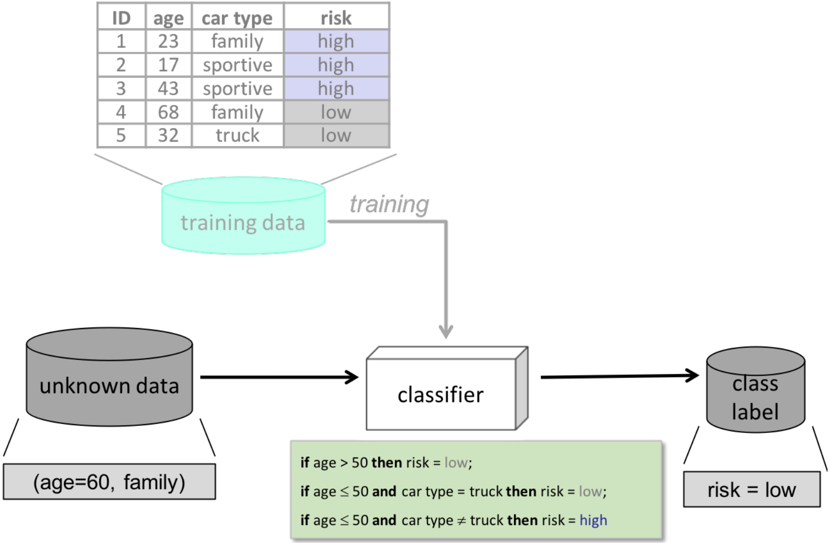
ID	age	car type	risk
1	23	family	high
2	17	sportive	high
3	43	sportive	high
4	68	family	low
5	32	truck	low

if age > 50 **then** risk = low;

if age $\leq$ 50 **and** car type = truck **then** risk = low;

if age $\leq$ 50 **and** car type $\neq$ truck **then** risk = high

Classification: Prediction Phase (Application)



ID	age	car type	risk
1	23	family	high
2	17	sportive	high
3	43	sportive	high
4	68	family	low
5	32	truck	low

if age > 50 **then** risk = low;

if age ≤ 50 **and** car type = truck **then** risk = low;

if age ≤ 50 **and** car type ≠ truck **then** risk = high

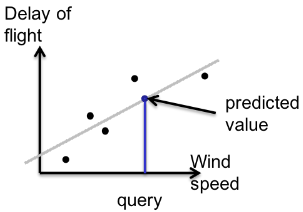
Classification

> *The systematic assignment of new observations to known categories according to criteria learned from a training set.*

Formal Setup

- A *classifier K* for a *model* $M(\theta)$ is a function $K_{M(\theta)} : D \to Y$, where
  - D: data space
    - Often d-dim. space with attributes $a_1, \ldots, a_d$ (not necessarily a vector space)
    - Some other space, e.g. metric space
  - $Y = \{y_1, \ldots, y_k\}$: set of k distinct *class labels*
  - $O \subseteq D$: set of *training objects o* with known class labels $y \in Y$
- *Classification*: Application of classifier K on objects from D (particularly $D \setminus O$)
- Model $M(\theta)$ is the "type" of the classifier, and θ are the model parameters
- *Supervised learning*: find/learn optimal parameters θ for $M(\theta)$ given O

Numerical Prediction



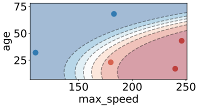
- Related problem to classification: numerical prediction
  - Determine the numerical value of an object
  - Method: e.g., regression analysis
  - Example: Prediction of flight delays

- Numerical prediction is *different* from classification
  - Classification refers to predict categorical class label
  - Numerical prediction models continuous-valued functions
- Numerical prediction is *similar* to classification
  - First, construct a model
  - Second, use model to predict unknown value
  - Major method for numerical prediction is regression:
    - Linear and multiple regression
    - Non-linear regression

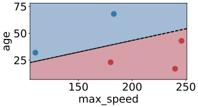
Goals for this Section

1. Introduction of different classification models
2. Learning techniques for these models

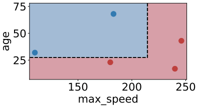
age	car_type	max_speed	risk
23	family	180	high
17	sportive	240	high
43	sportive	246	high
68	family	183	low
32	truck	110	low



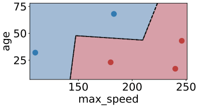
Bayes classifier



Linear classifier



Decision tree



k-NN classifier

Quality Measures for Classifiers

- Classification accuracy or classification error (complementary)
- Compactness of the model
  - Decision tree size, number of decision rules, ...
- Interpretability of the model
  - Insights and understanding of the data provided by the model
- Efficiency
  - Time to generate the model (training time)
  - Time to apply the model (prediction time)
- Scalability for large databases
  - Efficiency in disk-resident databases
- Robustness
  - Robust against noise or missing values

Quality Measures: Accuracy and Error

- Let K be a classifier
- Let $C(o)$ denote the correct class label of an object o
- Measure the quality of K:
  - Predict the class label for each object o from a data set $T \subseteq O$
  - Determine the fraction of correctly predicted class labels

Classification Accuracy of K

$$G_T(K) = \frac{|\{o \in T \mid K(o) = C(o)\}|}{|T|}$$

Classification Error of K

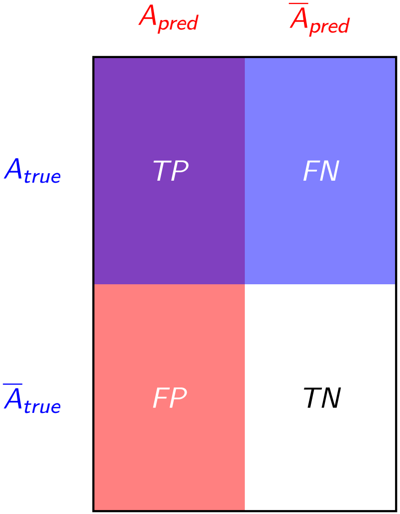
$$F_T(K) = \frac{|\{o \in T \mid K(o) \neq C(o)\}|}{|T|} = 1 - G_T(K)$$

- For a data set with classes of different size accuracy is misleading.
  - Therefore also other quality measures as accuracy and error are needed.

Task: Retrieve all Objects of a Single Class

Define

- True Positives (TP)
  It is true that the object is in the predicted class A
- False Positives (FP)
  It is false that the object is in the predicted class A
- True Negatives (TN)
  It is true that the object is in the predicted class $\overline{A}$ ($=$ it is in another class than A)
- False Negatives (FN)
  It is false that the object is in the predicted class $\overline{A}$ / Classifier predicts a class other than A, but this is false



	A_{pred}	$\overline{A}_{pred}$
A_{true}	TP	FN
$\overline{A}_{true}$	FP	TN

External Measures for a Single Classes

- *Recall* ($0 \leq rec \leq 1$, larger is better)

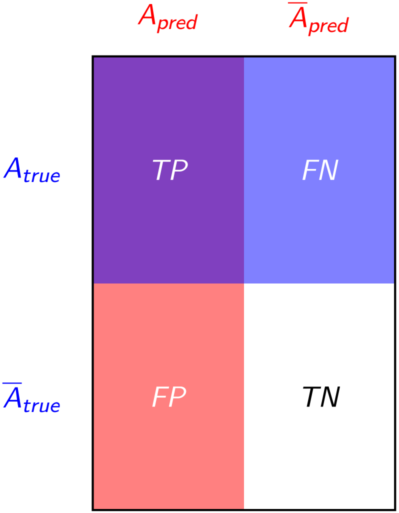
$$rec = \frac{TP}{TP + FN}$$

- *Precision* ($0 \leq prec \leq 1$, larger is better)

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

- F_1-*Measure* ($0 \leq F_1 \leq 1$, larger is better)

$$F_1 = \frac{2 \cdot rec \cdot prec}{rec + prec} = \frac{2TP}{2TP + FN + FP}$$



	A_{pred}	$\overline{A}_{pred}$
A_{true}	TP	FN
$\overline{A}_{true}$	FP	TN

Quality Measures: Confusion Matrix

- Results on the test set: Confusion matrix

		classified as ...				
		class 1	class 2	class 3	class 4	class 5
correct label	class 1	35	1	1	1	4
	class 2	0	31	1	1	5
	class 3	3	1	50	1	2
	class 4	1	0	1	10	2
	class 5	3	1	9	16	13

(correctly classified in green)

- Based on the confusion matrix, we can compute several accuracy measures, including:
  - Classification Accuracy/Error
  - Precision and Recall

Quality Measures: Computation

Recall

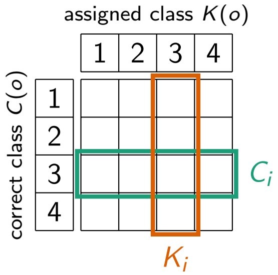
Fraction of test objects of class i, which have been identified correctly.

$$Recall_{TE}(K, i) = \frac{|\{o \in C_i \mid K(o) = C(o)\}|}{|C_i|}$$

Precision

Fraction of test objects assigned to class i, which have been identified correctly.

$$Precision_{TE}(K, i) = \frac{|\{o \in C_i \mid K_i(o) = C(o)\}|}{|K_i|}$$



$$C_i = \{o \in TE \mid C(o) = i\}$$
$$K_i = \{o \in TE \mid K(o) = i\}$$

Quality Measures for Data Sets

- So far recall, precision and F_1-Measure are just defined for a single class.
- To define these measures for the whole data set is not trivial.
- There are different approaches to define them with different meanings.
- You can determine the measures via
  - a micro approach, or
  - a macro approach.

Evaluation of Classifiers: Notions

- Using training data to build a classifier and to estimate the model's accuracy may result in misleading and overoptimistic estimates
  - $\rightsquigarrow$ Overspecialization of the learning model to the training data
- *Train-and-Test*: Decomposition of labeled data set O into two partitions
  - Training data is used to train the classifier
    - Construction of the model by using information about the class labels
  - Test data is used to evaluate the classifier
    - Temporarily hide class labels, predict them anew and compare with original class labels
- Train-and-Test is not applicable if the set of objects for which the class label is known is very small

Evaluation of Classifiers: Cross Validation

m-fold Cross Validation

- Decompose data set evenly into m subsets of (nearly) equal size
- Iteratively use $(m-1)$ partitions for training data and the remaining single partition as test data
- Combine the m classification accuracy values to an overall classification accuracy

Leave-one-out: Special case of cross validation ($m = n$)

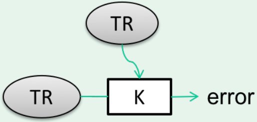
- For each of the objects o in the data set O:
  - Use set $O \setminus \{o\}$ as training set
  - Use the singleton set $\{o\}$ as test set
  - Compute classification accuracy by dividing the number of correct predictions through the database size $|O|$
- Particularly well applicable to nearest-neighbor classifiers

Quality Measures: Accuracy and Error

- Let K be a classifier
- Let $TRAIN \subseteq D$ be the training set: Used to build the classifier
- Let $TEST \subseteq D$ be the test set: Used to test the classifier

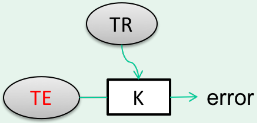
Resubstitution Error of K

$$F_{TRAIN}(K) = \frac{|\{d \in TRAIN \mid K(d) \neq C(d)\}|}{|TRAIN|}$$



(True) Classification Error of K

$$F_{TEST}(K) = \frac{|\{d \in TEST \mid K(d) \neq C(d)\}|}{|TEST|}$$

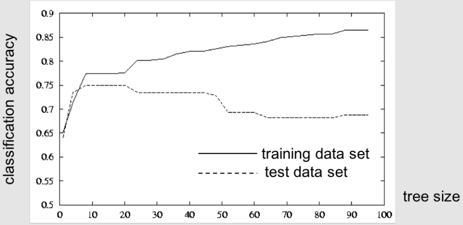


Overfitting

Characterization of Overfitting

The classifier adapts too closely to the training dataset and may therefore fail to accurately predict class labels for test objects unseen during training.

Example: Decision Tree



Generalization ← ***classifier*** → **specialization**
"overfitting"

Overfitting

Overfitting

- Occurs when the classifier is too optimized to the (noisy) training data
- As a result, the classifier yields worse results on the test data set
- Potential reasons:
  - Bad quality of training data (noise, missing values, wrong values)
  - Different statistical characteristics of training data and test data

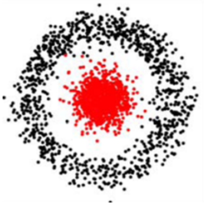
Overfitting Avoidance

- Removal of *noisy/erroneous/contradicting* training data
- Choice of an appropriate *size* of the training set
  - Not too small, not too large
- Choice of appropriate sample
  - Sample should describe all aspects of the domain and not only parts of it

Underfitting

Underfitting

- Occurs when the classifiers model is too simple, e.g. trying to separate classes linearly that can only be separated by a quadratic surface



⇝ *Trade-off*: Usually one has to find a good balance between over- and underfitting.

Agenda

1. Supervised Learning

1.1 Introduction: Classification

1.2 Bayesian Classifiers

1.3 Linear Discriminant Functions

1.4 Support Vector Machines

1.5 Kernel Methods

1.6 Decision Tree Classifiers

1.7 Nearest Neighbor Classifiers

1.8 Ensemble Classification

Bayes Classification

- Basic idea
  - Probability based classification: given a query object q, assign to q the class with the highest probability:
    $$K(q) = \underset{c_j \in \mathcal{C}}{\operatorname{argmax}} \left(P(c_j \mid q) \right)$$
  - The conditional probabilities $P(c_j \mid q)$ are hard to estimate, so turn the rule by applying Bayes' theorem to a formula based on $P(q \mid c_j)$.
  - Estimate the required probability density functions by using distribution models learned from the training data.
- Good classification results in many applications

Bayes' Theorem

- Conditional probabilities $P(A \mid B)$ ("probability of A given B") may be expressed in terms of joint probabilites $P(A \wedge B)$:

$$P(A \mid B) = \frac{P(A \wedge B)}{P(B)}$$

- This directly translates to the product rule $P(A \wedge B) = P(A \mid B) \cdot P(B)$
- By exploiting the symmetry $A \wedge B = B \wedge A$, we obtain:

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

- which leads us to the key theorem

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)}$$

Bayes Classifier

- Bayes' rule: $P(c_j \mid x) = \frac{P(x|c_j) \cdot P(c_j)}{P(x)}$ for object x and class $c_j \in \mathcal{C}$.
- We are interested in maximizing this, i.e.

$$\operatorname*{argmax}_{c_j \in \mathcal{C}} P(c_j \mid x) = \operatorname*{argmax}_{c_j \in \mathcal{C}} \frac{P(x \mid c_j) \cdot P(c_j)}{p(x)} \overset{(*)}{=} \operatorname*{argmax}_{c_j \in \mathcal{C}} \left(P(x \mid c_j) \cdot P(c_j)\right)$$

  where $(*)$ exploits the value of $p(x)$ is constant and does not change the result.
- Final decision rule:

$$K(x) = c_{\text{max}} = \operatorname*{argmax}_{c_j \in \mathcal{C}} \left(P(x \mid c_j) \cdot P(c_j)\right)$$

- Question: how to estimate the apriori probabilities $P(c_j)$ and the conditional probabilities $P(x \mid c_j)$?

Bayes Classifier: Density Estimation

A-Priori Class Probabilities, $P(c_j)$

Estimate the a-priori probabilities $P(c_j)$ of classes $c_j \in \mathcal{C}$ by using the observed relative frequency of the individual class labels c_j in the training set, i.e.,

$$P(c_j) = \frac{N_{c_j}}{N}$$

Conditional Probabilities, $P(x \mid c_j)$

- *Non-parametric* methods: Kernel methods Parzen's window, Gaussian kernels, etc.
- *Parametric* methods, e.g.
  - Single Gaussian distribution: Computed by maximum likelihood estimators (MLE)
  - *Mixture* models: e.g. Gaussian Mixture Model computed by EM algorithm

Bayes Classifier: Density Estimation

Multivariate objects

- If objects are represented by d components, $x = (x_1, \dots, x_d)$, we need to estimate $P(x \mid c_j) = P(x_1, \dots, x_d \mid c_j)$

Problem

- Correlations of attributes are hardly available if training sets are small
- Curse of dimensionality may cause problems in high-dimensional data

Solutions

- Dimensionality reduction
- Assume statistical independence of single attributes → naïve Bayes classifiers

Naïve Bayes Classifier

Independence Assumption

For any given class c_j the attribute values x_i are distributed independently, i.e.

$$P(x_1, \ldots, x_d \mid c_j) = \prod_{i=1}^{d} P(x_i \mid c_j) = P(x_1 \mid c_j) \cdot \ldots \cdot P(x_d \mid c_j)$$

Decision Rule

$$K_{\text{naïve}}(x) = \operatorname*{argmax}_{c_j \in \mathcal{C}} \left(P(c_j) \cdot \prod_{i=1}^{d} P(x_i \mid c_j) \right)$$

Naïve Bayes Classifier

Categorical Attribute x_i

$P(x_i \mid c_j)$ can be estimated as the relative frequency of samples having value v_i as the ith attribute in class c_j in the training set.

Continuous Attribute x_i

$P(x_i \mid c_j)$ can, for example, be estimated through a Gaussian distribution determined by μ_{ij}, σ_{ij}.

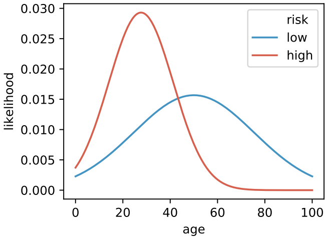
⇝ Computationally easy in both cases.

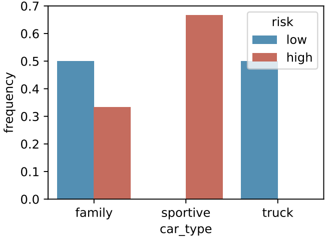
Naïve Bayes Classifier: Example

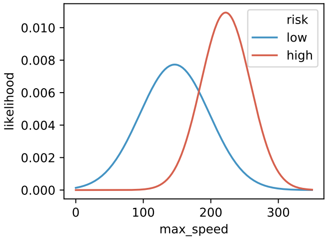
age	car_type	max_speed	risk
23	family	180	high
17	sportive	240	high
43	sportive	246	high
68	family	183	low
32	truck	110	low

Model Setup

- *age* $\sim N(\mu, \sigma^2)$ (normal distribution)
- *car_type* $\sim$ relative frequencies
- *max_speed* $\sim N(\mu, \sigma^2)$ (normal distribution)







Naïve Bayes Classifier: Example (cont'd)

Query

$q = (age = 60; car\_type = family; max\_speed = 190)$

Example

We have:

- $P(high) = \frac{3}{5}$
- $\mu_{age,high} = \frac{83}{3}, \sigma^2_{age,high} = \frac{1112}{3} \implies P\left(age = 60 \mid high\right) \approx 0.00506$
- $P(car\_type = family \mid high) = \frac{1}{3}$
- $\mu_{max\_speed,high} = 222, \sigma^2_{max\_speed,high} = 2664 \implies P\left(max\_speed = 190 \mid high\right) \approx 0.00638$

and hence

$$\begin{aligned} P(high)P(q \mid high) &= P(high)P(age = 60 \mid high)P(car\_type = family \mid high)P(max\_speed = 190 \mid high) \\ &\approx 6.45166 \cdot 10^{-6} \end{aligned}$$

Analogously, we obtain $P(low)P(q \mid low) = 15.72290 \cdot 10^{-6} \implies K_{\text{naïve}}(q) = low$.

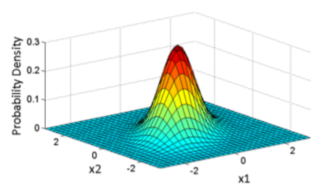
Bayesian Classifier

- Assuming dimensions of $x = (x_1, \ldots, x_d)$ are *not* independent
- Assume multivariate normal distribution (i.e. Gaussian)

$$P(x \mid C_j) = \frac{1}{\sqrt{(2\pi)^d \det(\Sigma_j)}} \exp\left(-\frac{1}{2}(x - \mu_j)\Sigma_j^{-1}(x - \mu_j)^T\right)$$

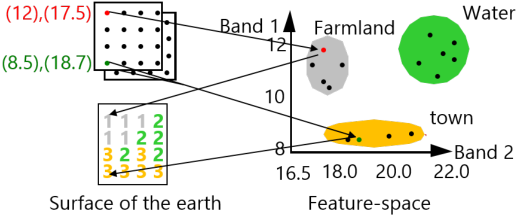
with

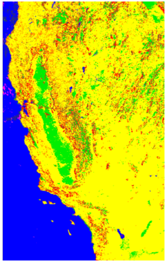
- μ_j: mean vector of class C_j
- Σ_j is the $d \times d$ covariance matrix
- $\det(\Sigma_j)$ is the determinant of Σ_j, and Σ_j^{-1} its inverse



Example: Interpretation of Raster Images

- Scenario: Automated interpretation of raster images
  - Take an image from a certain region (in d different frequency bands, e.g., infrared, etc.)
  - Represent each pixel by d values: $(x_1, \ldots, x_d)$
- Basic assumption: different surface properties of the earth ("landuse") follow a characteristic reflection and emission pattern



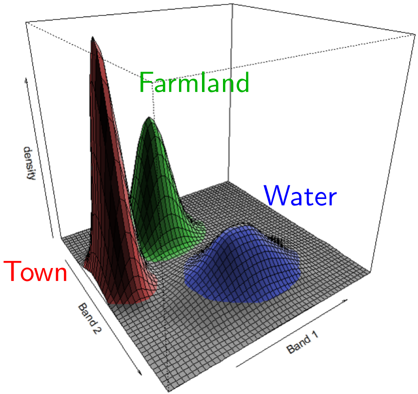


Example: Interpretation of Raster Images

Application of the Bayes classifier:

- Estimation of the $P(x \mid c)$ without assumption of conditional independence
- Assumption of d-dimensional normal (= Gaussian) distributions for the value vectors of a class

Probability of class membership



Example: Interpretation of Raster Images

Method

Estimate the following measures from training data

- μ_j: d-dimensional mean vector of all feature vectors of class C_j
- Σ_j: $d \times d$ covariance matrix of class C_j

Problems

- if likelihood of respective class is very low
- if several classes share the same likelihood

$\rightsquigarrow$ Mitigate e.g. by applying some minimum likelihood threshold; do not classify regions below.

Bayesian Classifiers: Discussion

Pro

- High classification accuracy for many applications if density function defined properly
- Incremental computation: many models can be adopted to new training objects by updating densities
  - For Gaussian: store count, sum, squared sum to derive mean, variance
  - For histogram: store count to derive relative frequencies
- Incorporation of expert knowledge about the application in the prior $P(C_i)$

Contra

- Limited applicability: often, required conditional probabilities are not available
- Lack of efficient computation: in case of a high number of attributes (particularly for Bayesian belief networks)

The Independence Hypothesis

The Independence Hypothesis . . .

- . . . makes efficient computation possible
- . . . yields optimal classifiers when satisfied
- . . . but is seldom satisfied in practice, as attributes (variables) are often correlated.

Attempts to overcome this limitation

- *Bayesian networks*, that combine Bayesian reasoning with causal relationships between attributes
- *Decision trees*, that reason on one attribute at the time, considering most important attributes first

Agenda

1. Supervised Learning

1.1 Introduction: Classification

1.2 Bayesian Classifiers

1.3 Linear Discriminant Functions

1.4 Support Vector Machines

1.5 Kernel Methods

1.6 Decision Tree Classifiers

1.7 Nearest Neighbor Classifiers

1.8 Ensemble Classification

Linear Discriminant Function Classifier

Idea

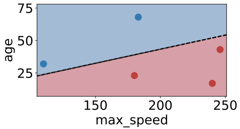
Separate points of two classes by a hyperplane

- I.e., classification model is a hyperplane
- Points of one class in one half space, points of second class in the other half space

Questions

- How to formalize the classifier?
- How to find optimal parameters of the model?

age	car_type	max_speed	risk
23	family	180	high
17	sportive	240	high
43	sportive	246	high
68	family	183	low
32	truck	110	low



Basic Notions

Recall some general algebraic notions for a vector space V:

- $\langle x, y \rangle$denotes an inner product of two vectors $x, y \in V$
  - E.g., the scalar product $\langle x, y \rangle = x^T y = \sum_{i=1}^{d} x_i y_i$
- $H(w, w_0)$ denotes a hyperplane with normal vector w and constant term w_0:

$$x \in H \Leftrightarrow \langle x, w \rangle + w_0 = 0$$

- The normal vector w may be normalized to w':

$$w' = \frac{1}{\sqrt{\langle w, w \rangle}} w \implies \langle w', w' \rangle = 1$$

- Distance of a point x to the hyperplane $H(w', w_0)$:

$$dist(x, H(w', w_0)) = |\langle w', x \rangle + w_0|$$

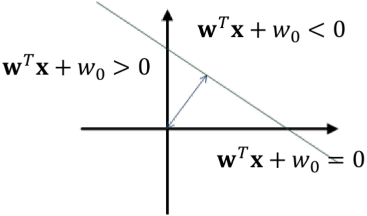
Formalization

- Consider a two-class example (generalizations later on):
  - D: d-dimensional vector space with attributes $a_1, \ldots, a_d$
  - $Y = \{-1, 1\}$ set of 2 distinct class labels y_j
  - $O \subseteq D$: Set of objects $o = (o_1, \ldots, o_d)$ with known class labels $y \in Y$ and cardinality $|O| = N$
- A hyperplane $H(w, w_0)$ with normal vector w and constant term w_0

$$x \in H \Leftrightarrow w^T x + w_0 = 0$$

Classification Rule

$$K_{H(w,w_0)}(x) = \text{sign}(w^T x + w_0)$$


$\mathbf{w}^T\mathbf{x} + w_0 < 0$
$\mathbf{w}^T\mathbf{x} + w_0 > 0$
$\mathbf{w}^T\mathbf{x} + w_0 = 0$

Optimal Parameter Estimation

How to estimate optimal parameters w, w_0?

1. Define an objective/loss function $L(\cdot)$ that assigns a value (e.g. the error on the training set) to each parameter-configuration
2. Optimal parameters minimize/maximize the objective function

How does an objective function look like?

- Different choices possible
- Most intuitive: Each misclassified object contributes a constant (loss) value $\rightsquigarrow$ 0-1 loss

Optimal Parameter Estimation

0-1 Loss Objective for Linear Classifier

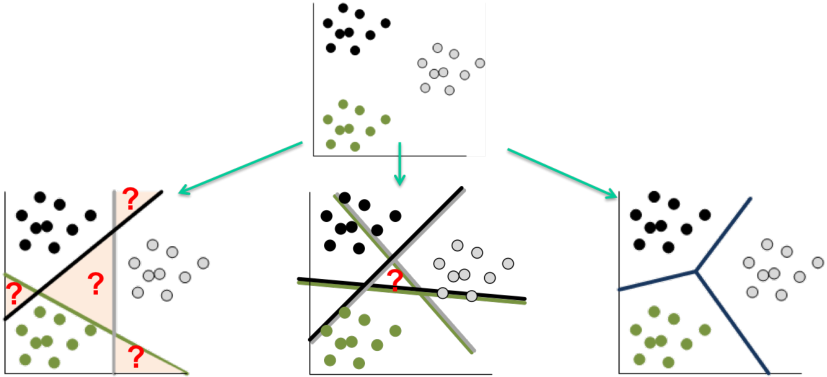
- $L(w, w_0) = \sum_{i=1}^{N} I(y_i \neq K_{H(w,w_0)}(x_i))$
- $\min_{w,w_0} L(w, w_0)$

where $I(condition) = 1$ if condition holds, 0 otherwise

- Minimize the overall number of training errors, but ...
  - NP-hard to optimize in general (non-smooth, non-convex)
  - Small changes of w, w_0 can lead to large changes of the loss

Extension to Multiple Classes

Assume we have more than two ($k > 2$) classes. What to do?



One-vs-the-rest ("one-vs-all")
→k classifiers

One-vs-one (Majority vote of classifiers)
→ $\frac{k(k-1)}{2}$classifiers

Multiclass linear classifier

→ k classifiers

Extension to Multiple Classes

Idea of Multiclass Linear Classifier

- Take k linear functions of the form $H_{w_j, w_{j,0}}(x) = w_j^T x + w_{j,0}$
- Decide for class y_j:

$$y_j = \operatorname*{argmax}_{j=1,\ldots,k} H_{w_j, w_{j,0}}(x)$$

- Advantage: No ambiguous regions except for points on decision hyperplanes
- The optimal parameter estimation is also extendable to k classes $y_1, \ldots, y_k$

Discussion (Linear Discriminant Function)

Advantages

- Simple approach
- Closed form solution for parameters
- Easily extendable to non-linear spaces (later on)

Disadvantages

- Sensitive to outliers – depending on the loss function $\rightsquigarrow$ not a stable classifier
  - How to define and efficiently determine the maximum stable hyperplane?
- Only good results for linearly separable data
- Expensive computation of selected hyperplanes

$\rightsquigarrow$ Approach to solve problems: Support Vector Machines (SVMs) [Vapnik 1979, 1995]

Agenda

1. Supervised Learning

1.1 Introduction: Classification
1.2 Bayesian Classifiers
1.3 Linear Discriminant Functions
1.4 Support Vector Machines
1.5 Kernel Methods
1.6 Decision Tree Classifiers
1.7 Nearest Neighbor Classifiers
1.8 Ensemble Classification

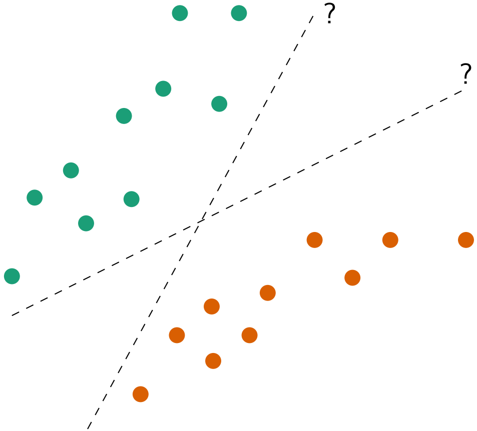
Maximum Margin Hyperplane

Question

How to define the notion of the "best" hyperplane differently?

Criteria

- Stability at insertion
- Distance to the objects of both classes

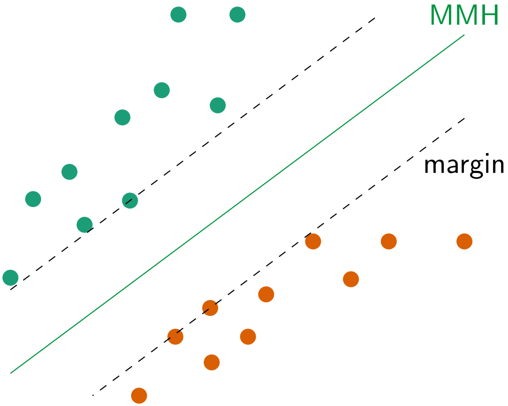


Support Vector Machines: Principle

Basic Idea

Linear separation with the *Maximum Margin Hyperplane (MMH)*:

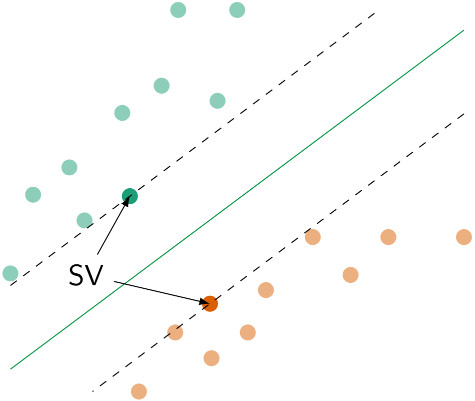
- Distance to points from any of the two sets is maximal, i.e., at least ξ
- Minimal probability that the separating hyperplane has to be moved due to an insertion
  $\rightsquigarrow$ Best generalization behavior; MMH is “maximally stable”



Support Vector Machines: Principle

Support Vectors

MMH only depends on points p_i whose distance to the hyperplane is exactly ξ. These p_i are called *support vectors* (SV).



Formalisation

- Let $\mathbf{x_i} \in \mathbb{R}^d$ denote the data points, and $y_i = +1$, if first class, else $y_i = -1$.
- A hyperplane in Hesse normal form is represented by a normal vector $\mathbf{w} \in \mathbb{R}^d$ of unit length (i.e., $\|\mathbf{w}\|_2 = 1$), and a (signed) distance from the origin $b \in \mathbb{R}$.
- In the following slides, we will define the requirements which the MMH shall fulfil.

Requirements of the MMH

The parameters $(\mathbf{w}, b)$ of the MMH shall fulfil the following two requirements:

No Error

The classification is accurate for all points, i.e.

$$y_i \cdot (\langle \mathbf{w}, \mathbf{x_i} \rangle + b) > 0 \iff \begin{cases} y_i = -1 & \langle \mathbf{w}, \mathbf{x_i} \rangle + b < 0 \\ y_i = +1 & \langle \mathbf{w}, \mathbf{x_i} \rangle + b > 0 \end{cases}$$

Requirement: Maximal Margin

Let $\xi = \min\limits_{\mathbf{x_i} \in TR} |\langle \mathbf{w}, \mathbf{x_i} \rangle + b|$ denote the minimum distance of any training object $\mathbf{x_i}$ to the hyperplane $H(\mathbf{w}, b)$. The margin ξ should be as large as possible.

Computation of the MMH

- *Task*: Maximise ξ subject to $y_i \cdot (\langle \mathbf{w}, \mathbf{x_i} \rangle + b) > \xi$ for all $i \in \{1, \ldots, n\}$.
- Scaling the constraints by ξ^{-1} yields $y_i \cdot (\langle \xi^{-1}\mathbf{w}, \mathbf{x_i} \rangle + \xi^{-1} b) > 1$ for all $i \in \{1, \ldots, n\}$.
- Define $\mathbf{w}' = \xi^{-1}\mathbf{w}$, and $b' = \xi^{-1} b$.
- Maximizing ξ corresponds to minimizing $\langle \mathbf{w}', \mathbf{w}' \rangle = \frac{\langle \mathbf{w}, \mathbf{w} \rangle}{\xi^2}$.

Primal Optimization Problem

$$\begin{array}{lll} \text{min} & \|\mathbf{w}'\|_2^2 & \\ s.t. & y_i \cdot (\langle \mathbf{w}', \mathbf{x_i} \rangle + b') > 1 & i \in \{1, \ldots, n\} \end{array}$$

Computation of the MMH

Primal Optimization Problem

$$\begin{aligned} \min \quad & \|\mathbf{w}'\|_2^2 \\ s.t. \quad & y_i \cdot (\langle \mathbf{w}', \mathbf{x_i} \rangle + b') > 1 \qquad i \in \{1, \ldots, n\} \end{aligned}$$

- Convex optimization problem: Quadratic programming problem with linear constraints
  $\Longrightarrow$ Solution can be obtained by Lagrangian Theory.

Optimization with Lagrange-Multipliers

Wolfe-Dual with Lagrange Multipliers

$$\begin{aligned}
\max \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \mathbf{x_i}, \mathbf{x_j} \rangle \\
s.t. \quad & \sum_{i=1}^{n} \alpha_i y_i = 0 \\
& 0 \leq \alpha_i \leq C \qquad\qquad i \in \{1, \ldots, n\}
\end{aligned}$$

- $\alpha_i = 0$: $\mathbf{x_i}$ is not a support vector
- $\alpha_i = C$: $\mathbf{x_i}$ is support vector with $\xi_i > 0$
- $0 < \alpha_i < C$: $\mathbf{x_i}$ is support vector with $\xi_i = 0$

Optimization with Lagrange-Multipliers

Decision Rule

$$H(q) = sign\left(\sum_{x_i \in SV} \alpha_i y_i \langle \mathbf{x_i}, \mathbf{q} \rangle + b\right)$$

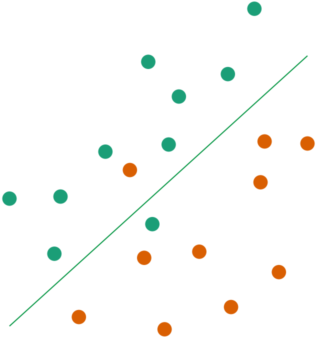
where SV denotes the set of support vectors.

Note that $\sum_{x_i \in SV} \alpha_i y_i \langle \mathbf{x_i}, \mathbf{q} \rangle = \sum_{x_i \in TS} \alpha_i y_i \langle \mathbf{x_i}, \mathbf{q} \rangle$ since $\alpha_i = 0$ for all $x_i \notin SV$

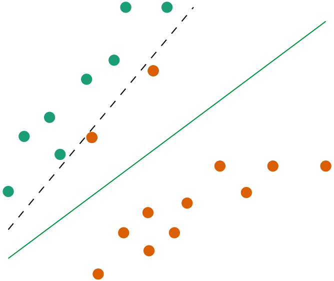
Soft Margin Optimization

- Problem of MMH optimization: How to treat non-(linearly separable) data?
- Two typical problems:

data points not linearly separable



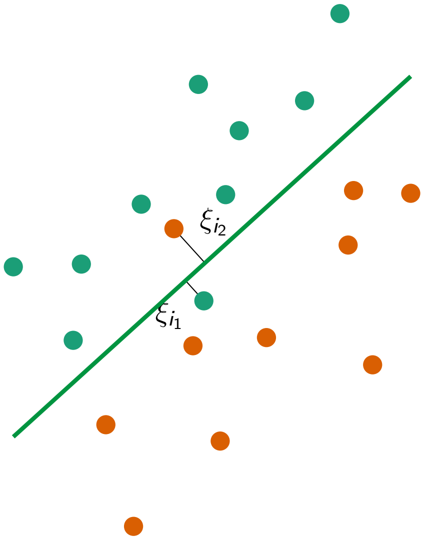
complete separation not optimal (overfitting)



- Trade-off between training error and size of margin

Soft Margin Optimization

- Additionally regard the number of training errors when optimizing:
  - ξ_i is the distance from $\mathbf{x_i}$ to the margin (often called *slack variable*):
    - $\xi_i = 0 \implies \mathbf{x_i}$ on correct side
    - $\xi_i > 0 \implies \mathbf{x_i}$ on wrong side
- Introduce parameter C to weight the misclassification against the size of the margin.



Soft Margin Optimization

Primal Optimization Problem With Soft Margin

$$
\begin{aligned}
\text{min} \quad & \frac{1}{2}\|\mathbf{w}'\|_2^2 + C\sum_{i=1}^{n}\xi_i & & \\
s.t. \quad & y_i \cdot \left(\left\langle \mathbf{w}', \mathbf{x_i} \right\rangle + b'\right) > 1 - \xi_i & & i \in \{1, \ldots, n\} \\
& \xi_i \geq 0 & & i \in \{1, \ldots, n\}
\end{aligned}
$$

SVM: Discussion

Pro

- generate classifiers with a high classification accuracy
- relatively weak tendency to overfitting (generalization theory)
- efficient classification of new objects due to often small number of support vectors
- compact models

Contra

- training times may be long (appropriate feature space may be very high-dimensional)
- expensive implementation

Agenda

1. Supervised Learning

1.1 Introduction: Classification

1.2 Bayesian Classifiers

1.3 Linear Discriminant Functions

1.4 Support Vector Machines

1.5 Kernel Methods

1.6 Decision Tree Classifiers

1.7 Nearest Neighbor Classifiers

1.8 Ensemble Classification

Non-Linearly Separable Data Sets

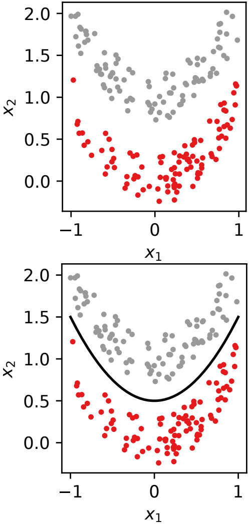
Problem

In many real scenarios, a linear separation causes a very poor classification quality.

Core idea

- Map the data by a non-linear transformation into an (extended) space
- Separate the data in the new space by linear methods

Example for quadratically separable data set



Extension of the Hypotheses Space

Principle

- Mapping ϕ: original space → extended feature space
- Separate data in the new space by linear techniques

Example

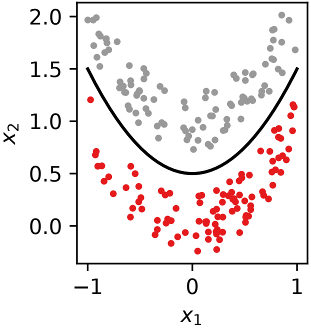
- Polynomial mapping of $x = (x_1, x_2) \in \mathbb{R}^2$:

$$\phi(x) = \phi\left((x_1, x_2)\right) = \left(1, x_1, x_2, x_1^2, x_1 x_2, x_2^2\right)$$

- A (linear) *hyperplane* in the extended feature space is equivalent to a (non-linear) *polynomial of degree 2* in the original space (see below).

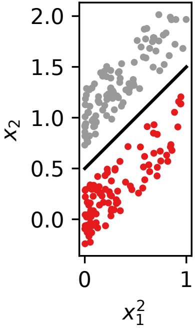
Extension of the Hypotheses Space: Parabola Example

Original space: $x = (x_1, x_2) \in \mathbb{R}^2$



separation by $x_2 = (x_1)^2 + 0.5$

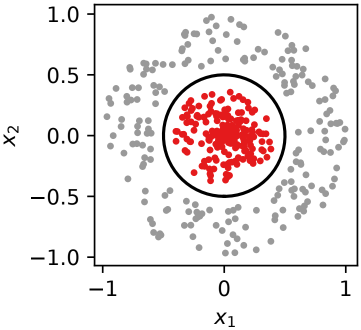
Extended space:
$\phi(x) = \left(1, x_1, x_2, x_1^2, x_1x_2, x_2^2\right) \in \mathbb{R}^6$



separation by $x_2 = (x_1^2) + 0.5$

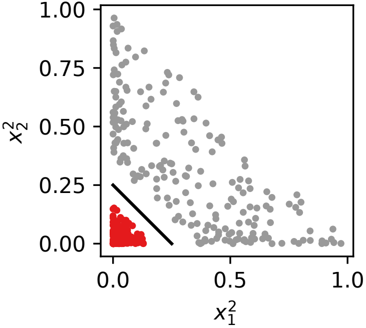
Extension of the Hypotheses Space: Circular Example

Original space: $x = (x_1, x_2) \in \mathbb{R}^2$



separation by $(x_1)^2 + (x_2)^2 = 0.25$

Extended space:
$\phi(x) = \left(x_1^2, x_2^2, x_1 x_2\right) \in \mathbb{R}^3$



separation by $(x_2^2) + (x_1^2) = 0.25$

Extension of Linear Discriminant Function Classifier

- Linear classifier can be easily extended to non-linear spaces
- Recap: linear classifier $K_{H(w,w_0)}(x) = sign(w^T x + w_0)$
- Extend to non-linear case:
  - Transform all data points x to new feature space $\phi(x)$
  - Data Matrix X becomes a matrix Φ
  - The optimal hyperplane vector becomes ...

$$\tilde{w}_{opt,\phi} = (\Phi^T \Phi)^{-1} \Phi^T Y$$

  - ...and that's all!
- New classification rule: $K_{H(w_\phi,w_{0,\phi})}(x) = sign(w_\phi^T \phi(x) + w_{0,\phi})$
- SVM can be extended in a similar way

Non-linear Classification: Discussion

Pro

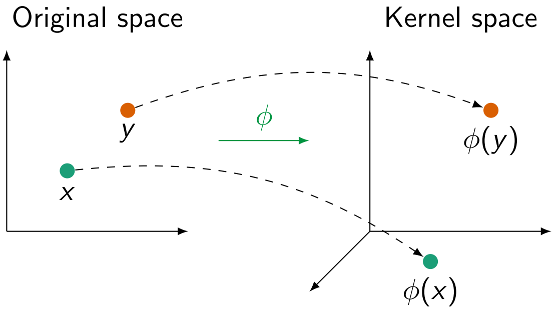
- Explicit feature transformation yields a richer hypotheses space
- Simple extension of existing techniques
- Efficient evaluation, if transformed feature space not too high-dimensional

Contra

- Explicit mapping may run into problems (efficiency, high dimensions)
- Meaningful transformation is usually not known a-priori
- Complex data distributions may require very high-dimensional features spaces $\rightsquigarrow$ high memory consumption, high computational costs

Implicit Mappings: Kernel Trick

- We often need the scalar product of mapped objects only, $K_\phi(x, y) = \langle \phi(x), \phi(y) \rangle$
- If $K_\phi(x, y)$ is represented in the original domain, the mapping ϕ remains *implicit* only, and the problems of mapping *explicitly* are avoided



Kernel: Example

- Let $\phi : \mathbb{R}^2 \to \mathbb{R}^3$ represent the mapping $\phi\left((x_1, x_2)\right) = \left(x_1^2, x_2^2, \sqrt{2}x_1x_2\right)$
- The scalar product of mapped objects x, y is calculated as follows:

$$\begin{aligned}
\langle\phi(x), \phi(y)\rangle &= \left\langle \left(x_1^2, x_2^2, \sqrt{2}\cdot x_1x_2\right), \left(y_1^2, y_2^2, \sqrt{2}\cdot y_1y_2\right)\right\rangle \\
&= x_1^2y_1^2 + x_2^2y_2^2 + 2x_1x_2y_1y_2 \\
&= (x_1y_1 + x_2y_2)^2 \\
&= \langle(x_1, x_2), (y_1, y_2)\rangle^2 \\
&= \langle x, y\rangle^2
\end{aligned}$$

- This $K(x, y) = \langle x, y\rangle^2$ is simply calculated in the *original* space and does not require to map $\phi(x), \phi(y)$ explicitly
- The kernel trick even allows for mappings ϕ to spaces with infinite dimensions
- Example: Radial basis function kernels, $K_{RBF}(x, y) = exp\left(-\gamma \cdot \|x - y\|^2\right), \gamma > 0$

Why are scalar products useful?

- Kernels correspond to scalar products in the corresponding feature space
- Scalar products and, thus, Kernels are used in various definitions:
  - L_2 Norm of a vector, $\|x\|_2 = \sqrt{\langle x, x\rangle}$
    Norm induced by kernel K, $\|x\|_K = \sqrt{K(x, x)}$
  - Distances of points, $\|x - y\|_2 = \sqrt{\langle x - y, x - y\rangle} = \sqrt{\langle x, x\rangle + \langle y, y\rangle - 2\langle x, y\rangle}$
    Kernel distances, $d_K(x, y) = \sqrt{K(x, x) + K(y, y) - 2 \cdot K(x, y)}$
  - Angle between two vectors, $\angle(x, y) = \arccos \frac{\langle x,y\rangle}{\|x\| \cdot \|y\|}$
    Kernel-based angles of vectors: $\angle_K(x, y) = \arccos \frac{K(x,y)}{\|x\|_K \cdot \|y\|_K}$

Formal Definitions

Definition: Kernel Function

A *kernel function* $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ on an input space $\mathcal{X}$ is a symmetric function which maps pairs of objects $x, y \in \mathcal{X}$ to real numbers.

Definition: Mercer Kernel

A kernel function K is called *Mercer kernel, valid kernel, admissible kernel, or positive semi-definite kernel*, if for all finite subsets $X = \{x_1, \ldots, x_n\} \subseteq \mathcal{X}$, the $n \times n$ matrix $M_K(X)$ with $M_K(X)_{i,j} = K(x_i, x_j)$ is positive semi-definite, i.e. for all $c \in \mathbb{R}^n$, it holds

$$c^T \cdot M_K(X) \cdot c \geq 0$$

Formal Definitions (cont'd)

Definition: Scalar Product

A scalar product in a vector space $\mathcal{H}$ is a function $\langle \cdot, \cdot \rangle : \mathcal{H} \times \mathcal{H} \to \mathbb{R}$ satisfying:

- $\langle x, x \rangle = 0$ for $x = 0$
- $\langle x, x \rangle > 0$ for $x \neq 0$
- $\langle x, y \rangle = \langle y, x \rangle$ (symmetry)
- $\langle \alpha x + \beta y, z \rangle = \alpha \langle x, z \rangle + \beta \langle y, z \rangle$ (bi-linearity)

Definition: Hilbert Space

A vector space $\mathcal{H}$ endowed with a scalar product $\langle \cdot, \cdot \rangle : \mathcal{H} \times \mathcal{H} \to \mathbb{R}$ for which the induced norm gives a complete metric space, is called *Hilbert Space*.

Interpretation of Kernel Functions

Theorem

Let $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a valid kernel on $\mathcal{X}$. There exists a possibly infinite-dimensional Hilbert space $\mathcal{H}$ and a mapping $\phi : \mathcal{X} \to \mathcal{H}$ such that $K(x, y) = \langle \phi(x), \phi(y) \rangle_{\mathcal{H}}$ for all $x, y \in \mathcal{X}$ where $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ denotes the scalar product in the Hilbert space $\mathcal{H}$.

$\rightsquigarrow$ every kernel K can be seen as a scalar product in some feature space $\mathcal{H}$.

Advantages

- Feature space $\mathcal{H}$ can be infinite-dimensional
- Not really necessary to know which feature space $\mathcal{H}$ we have
- Computation of kernel is done in original domain $\mathcal{X}$

Kernel SVM

Wolfe-Dual Optimization Problem with Lagrange Multipliers

$$\max \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \quad \rightarrow \quad \max \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

$$s.t. \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

$$0 \leq \alpha_i \leq C, i \in \{1, \ldots, n\}$$

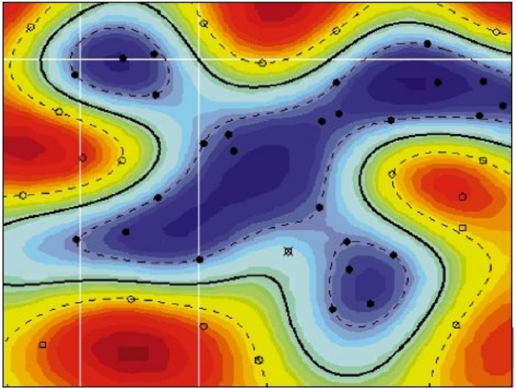
Decision Rule for Kernel Machines

$$H(x) = sign\left(\sum_{x_i \in SV} \alpha_i y_i \langle x_i, x \rangle + b \right) \quad \rightarrow \quad sign\left(\sum_{x_i \in SV} \alpha_i y_i \cdot K(x_i, x) + b \right)$$

Examples for (Mercer) Kernels

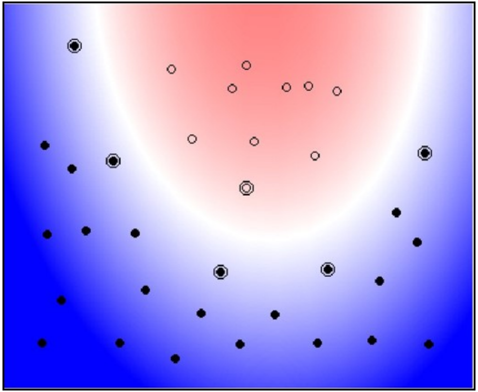
Radial Basis Function Kernel

$K(x, y) = \exp\left(-\gamma \cdot \|x - y\|^2\right), \gamma > 0$



Polynomial Kernel of degree d

$K(x, y) = \left(\langle x, y\rangle + 1\right)^d, d \in \mathbb{N}$



Non-numeric Kernels

General Kernels

- The kernel trick may be applied to non-numerical, structured objects as well.
- Kernels $K(x, y)$ are defined in terms of object properties
- $K(x, y) = \langle \phi(x), \phi(y) \rangle$ remains a scalar product in $\mathbb{R}^d$ but $\phi : \mathcal{X} \rightarrow \mathbb{R}^d$ is not needed explicitly

Examples

- *Graph kernels*: geometric tree kernels; random walk kernels; Wasserstein Weisfeiler-Lehman Graph Kernels for attributed graphs
- *Fisher kernels*: digest complex objects by pooling features and represent them by Gaussian mixture models

Discussion

Pro

- Kernel methods provide a great method for dealing with non-linearity
- Implicit mapping allows for spaces of arbitrary dimensions (even infinite)
- Computational effort of training Kernel machines depends on the number of training examples, but not on the feature space dimension

Contra

- Resulting models may be hard to explain intuitively
- Choice of kernel can be difficult

Agenda

1. Supervised Learning

1.1 Introduction: Classification

1.2 Bayesian Classifiers

1.3 Linear Discriminant Functions

1.4 Support Vector Machines

1.5 Kernel Methods

1.6 Decision Tree Classifiers

1.7 Nearest Neighbor Classifiers

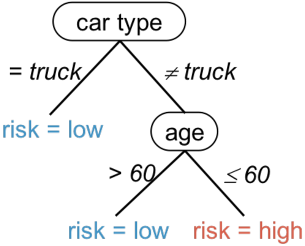
1.8 Ensemble Classification

Decision Tree Classifiers

- Approximating discrete-valued target function
- Learned function is represented as a tree:
  - A flow-chart-like tree structure
  - Internal node denotes a test on an attribute
  - Branch represents an outcome of the test
  - Leaf nodes represent class labels or class distribution
- Tree can be transformed into decision rules:

if age > 60 **then** risk $=$ low
if age ≤ 60 **and** car type $=$ truck **then** risk $=$ low
if age ≤ 60 **and** car type $\neq$ truck **then** risk $=$ high

age	car_type	max_speed	risk
23	family	180	high
17	sportive	240	high
43	sportive	246	high
68	family	183	low
32	truck	110	low



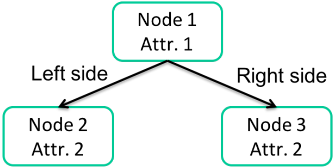
Advantages

- Decision trees represent explicit knowledge
- Decision trees are intuitive to most users

Decision Tree Classifier: Splits

Goal

- Each tree node defines an axis-parallel $(d-1)$-dimensional hyperplane, that splits the data space.
- *Find such splits which lead to as homogeneous groups as possible.*



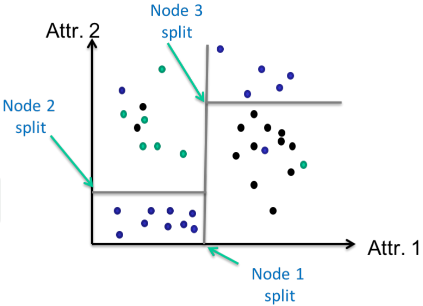
Node 1
Attr. 1

Left side

Right side

Node 2
Attr. 2

Node 3
Attr. 2



Decision Tree Classifiers: Basics

- Decision tree generation (training phase) consists of two phases
  1. Tree construction
     - At start, all the training examples are at the root
     - Partition examples recursively based on selected attributes
  2. Tree pruning
     - Identify and remove branches that reflect noise or outliers
- Use of decision tree: Classifying an unknown sample
  - Traverse the tree and test the attribute values of the sample against the decision tree
  - Assign the class label of the respective leaf to the query object

Algorithm for Decision Tree Construction

- Basic algorithm (a greedy algorithm): ID3 (Iterative Dichotomiser 3)
  - Tree is created in a *top-down recursive divide-and-conquer* manner
  - Attributes may be categorical or continuous-valued
  - At the start, all the training examples are assigned to the root node
  - Recursively partition examples at each node and push them down to the new nodes
  - Select test attributes and determine split points or split sets for the respective values based on a heuristic or statistical measure (*split strategy*, e.g., information gain)
- Conditions for stopping partitioning for ID3
  - All samples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
  - There are no samples left

Algorithm for Decision Tree Construction

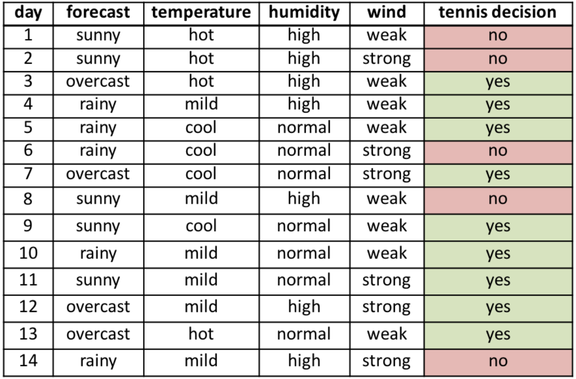
- Most algorithms are versions of this basic algorithm (greedy, top-down)
  - E.g.: ID3, or its successor C4.5

ID3 Algorithm (Iterative Dichotomiser 3)

```
procedure ID3($Examples$, $TargetAttr$, $Attributes$)        ▷ specialized to learn boolean-valued functions
    Create $Root$ node for the tree
    if all $Examples$ are positive then return $Root$ with $label = +$
    else if all $Examples$ are negative then return $Root$ with $label = -$
    else if $Attributes = \emptyset$ then return $Root$ with $label =$ most common value of $TargetAttr$ in $Examples$
    else
        $A =$ "best" decision attribute for next node        ▷ how to determine the "best" attribute?
        Assign $A$ as decision attribute for $Root$
        for each possible value $v_i$ of $A$ do        ▷ how to split the possible values?
            Generate branch corresponding to test $A = v_i$
            $Examples_{v_i} =$ examples that have value $v_i$ for $A$
            if $Examples_{v_i} = \emptyset$ then
                Add leaf node with $label =$ most common value of $TargetAttr$ in $Examples$
            else
                Add subtree ID3($Examples_{v_i}$, $TargetAttr$, $Attributes \setminus \{A\}$)
```

Example: Decision for "playing_tennis"

- Query: How about playing tennis today?
- Training data:



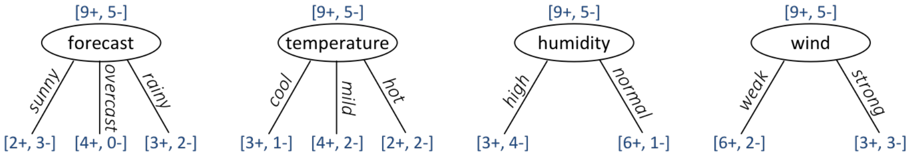
day	forecast	temperature	humidity	wind	tennis decision
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rainy	mild	high	weak	yes
5	rainy	cool	normal	weak	yes
6	rainy	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rainy	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rainy	mild	high	strong	no

- Build decision tree . . .

Split Strategies: Quality of Splits

Given

- A set T of training objects
- A (disjoint, complete) partitioning $T_1, \ldots T_m$ of T
- The relative frequencies p_i of class c_i in T and in the partitions $T_1, \ldots T_m$



Wanted

- A measure for the heterogeneity of a set S of training objects with respect to the class membership
- A split of T into partitions $\{T_1, \ldots, T_m\}$ such that the heterogeneity is minimized

$\rightsquigarrow$ Proposals: *Information gain, Gini index, Misclassification error*

Attribute Selection Measures: Information Gain

- Used in ID3/C4.5

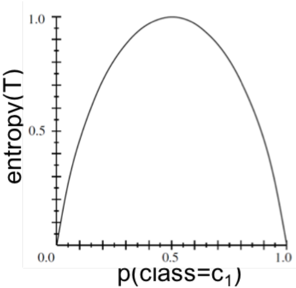
Entropy

- Minimum number of bits to encode a message that contains the class label of a random training object
- The entropy of a set T of training objects is defined as

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

for k classes with frequencies p_i

- $entropy(T) = 0$ if $p_i = 1$ for any class c_i
- $entropy(T) = 1$ if $p_i = \frac{1}{k}$ for all classes c_i



$k = 2$

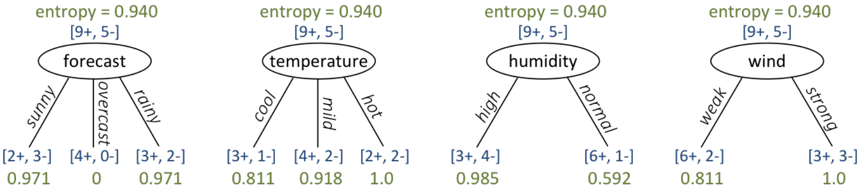
Attribute Selection Measures: Information Gain

Information Gain

Let A be the attribute that induced the partitioning $\{T_1, \ldots, T_m\}$ of T. The information gain of attribute A w.r.t. T is defined as

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

Attribute Selection: Example (Information Gain)

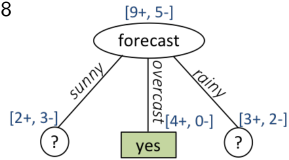


$$information\_gain(T, forecast) = 0.94 - \frac{5}{14}0.971 - \frac{4}{14}0 - \frac{5}{14}0.971 = 0.246$$

$$information\_gain(T, temperature) = 0.94 - \frac{4}{14}0.811 - \frac{6}{14}0.981 - \frac{4}{14}1 = 0.002$$

$$information\_gain(T, humidity) = 0.94 - \frac{7}{14}0.985 - \frac{7}{14}0.592 = 0.151$$

$$information\_gain(T, wind) = 0.94 - \frac{8}{14}0.811 - \frac{6}{14}1 = 0.048$$

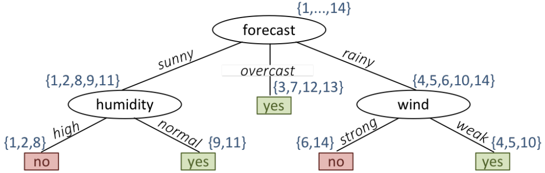


Result: ”forecast” yields the highest information gain

Example: Decision Tree for ”playing_tennis”

day	forecast	temperature	humidity	wind	tennis decision
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	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rainy	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rainy	mild	high	strong	no

Final decision tree:



Attribute Selection Measures: Gini Index

- Used in IBM's IntelligentMiner

Gini Index

The Gini index for a set T of training objects is defined as

$$gini(T) = 1 - \sum_{i=1}^{k} p_i^2$$

- Small value of Gini index $\equiv$ low heterogeneity
- Large value of Gini index $\equiv$ high heterogeneity

Gini Index (of an attribute A)

Let A be the attribute that induced the partitioning $\{T_1, \ldots, T_m\}$ of T. The Gini index of attribute A w.r.t. T is defined as

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

Attribute Selection Measures: Misclassification Error

Misclassification Error

The Misclassification Error for a set T of training objects is defined as

$$Error(T) = 1 - \max_{c_i} p_i$$

- Small value of Error $\equiv$ low heterogeneity
- Large value of Error $\equiv$ high heterogeneity

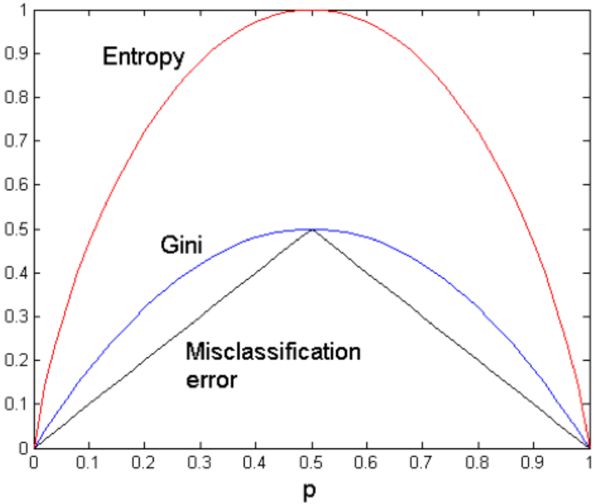
Misclassification Error (of an attribute A)

Let A be the attribute that induced the partitioning $\{T_1, \ldots, T_m\}$ of T. The Misclassification Error of attribute A w.r.t. T is defined as

$$Error_A(T) = \sum_{i=1}^{m} \frac{|T_i|}{|T|} Error(T_i)$$

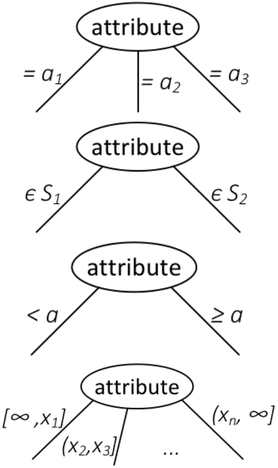
Attribute Selection Measures: Comparison

For two-class problems:



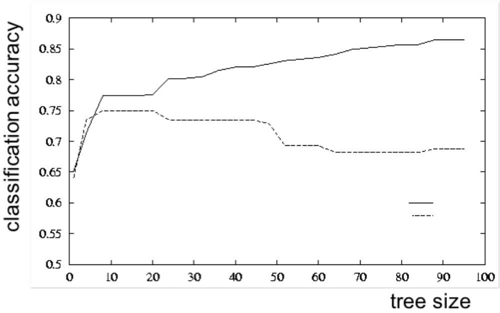
Split Strategies: Types of Splits

- Categorical attributes
  - Split criteria based on equality ”$\textit{attribute} = a$”
  - Based on subset relationships ”$\textit{attribute} \in \textit{set}$”
    $\rightsquigarrow$ many possible choices (subsets)
    - Choose the best split according to, e.g., gini index
- Numerical attributes
  - Split criteria of the form ”$\textit{attribute} < a$”
    $\rightsquigarrow$ many possible choices for the split point
    - One approach: Order test samples w.r.t. their attribute value; consider every mean value between two adjacent samples as possible split point; choose best one according to, e.g., gini index
  - Partition the attribute value into a discrete set of intervals (Binning)



Avoid Overfitting in Classification

- The generated tree may overfit the training data
  - Too many branches, some may reflect anomalies due to noise or outliers
  - Result has poor accuracy for unseen samples



solid line: training data; dashed line: test data

- Two approaches to avoid overfitting for decision trees:
  1. Post-pruning = pruning of overspecialized branches
  2. Pre-pruning = halt tree construction early

Pruning Techniques for Decision Trees

Post-pruning

Pruning of overspecialized branches:

- Remove branches from a "fully grown" tree and get a sequence of progressively pruned trees
- Use a set of data different from the training data to decide which is the “best pruned tree”

Pruning Techniques for Decision Trees

Pre-pruning

Halt tree construction early, do not split a node if this would result in the goodness measure falling below a threshold.

- Choice of an appropriate value for *minimum support*
  - Minimum support: minimum number of data objects a leaf node contains
  - In general, *minimum support* $\gg 1$
- Choice of an appropriate value for *minimum confidence*
  - Minimum confidence: minimum fraction of the majority class in a leaf node
  - Typically, *minimum confidence* $\ll 100\%$
  - Leaf nodes can absorb errors or noise in data records
- Discussion
  - With Pre-pruning it is difficult to choose appropriate thresholds
  - Pre-pruning has less information for the pruning decision than post-pruning $\rightsquigarrow$ can be expected to produce decision trees with lower classification quality
  - Tradeoff: tree construction time vs. classification quality

Avoid Overfitting with Regularization

General: Regularization

Solve the regularized minimization problem

$$\min_{\theta} f(\cdot, \theta) + \lambda g(\theta)$$

where θ denotes the model's parameters, $f(\cdot, \theta)$ is used as a loss function, $g(\theta)$ is a **regularization term** and λ is a trade-off hyperparameter.

- Regularization terms are used to fine-tune the model's complexity
  - Prevents overfitting of a model
- The L_1-norm and L_2-norm, respectively, are commonly used for regularizing the model's parameter

Minimal Cost Complexity Pruning: Notions

- Size $|E|$ of a decision tree E: number of leaf nodes
- Cost-complexity quality measure of E with respect to training set T, classification error F_T and complexity parameter $\alpha \geq 0$:

$$CC_T(E, \alpha) = F_T(E) + \alpha|E|$$

- For the *smallest minimal subtree* $E(\alpha)$ of E w.r.t. α, it is true that:
  1. There is no subtree of E with a smaller cost complexity
  2. If $E(\alpha)$ and B both fulfill (1), then is $E(\alpha)$ a subtree of B
- $\alpha = 0$: $E(\alpha) = E$
  - Only error matters
- $\alpha \to \infty$: $E(\alpha) =$ root node of E
  - Only tree size matters
- $0 < \alpha < \infty$: $E(\alpha)$ is a proper substructure of E
  - The root node or more than the root node

Decision Tree Classifiers: Summary

Pro

- Relatively fast learning speed (in comparison to other classification methods)
- Fast classification speed
- Convertible to simple and easy to understand classification rules
- Often comparable classification accuracy with other classification methods

Contra

- Not very stable, small changes of the data can lead to large changes of the tree

Agenda

1. Supervised Learning

1.1 Introduction: Classification

1.2 Bayesian Classifiers

1.3 Linear Discriminant Functions

1.4 Support Vector Machines

1.5 Kernel Methods

1.6 Decision Tree Classifiers

1.7 Nearest Neighbor Classifiers

1.8 Ensemble Classification

Nearest Neighbor Classifiers

Motivation

- Assume data in a non-vector representation: graphs, forms, XML-files, etc.
- No simple way to use linear classifiers or decision trees

Solutions

- Use appropriate kernel function for kernel machines (e.g. kernel SVM)
  $\rightsquigarrow$ Not always clear how to define a kernel
- Embedding of objects into some vector space (e.g. representation learning)
  $\rightsquigarrow$ Difficult to determine appropriate embedding projection
- *Here*: Nearest neighbor classifier
  $\rightsquigarrow$ Direct usage of (dis-)similarity functions for objects

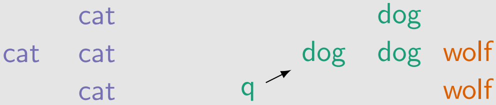
Nearest Neighbor Classifiers

Procedure

Assign query object q to the class c_j of the closest training object $x \in D$:

$$class(q) = class(NN(q)) \qquad NN(q) = \{x \in D \mid \forall x' \in D : d(q,x) \leq d(q,x')\}$$

Example



Classifier decides that query object q is a dog.

Instance-Based Learning

Eager Evaluation

- Examples: Decision tree, Bayes classifier, SVM
- Training phase: Learn parameters for chosen model from training data
- Test phase: evaluate parameterized model for arriving query objects

Lazy Evaluation

- Typical Approach: (k-)nearest neighbor classifiers
- Derive labels from individual training objects: *instance-based learning*
- No training required (= lazy)
- Highly recommended: put data into efficient index structure (e.g., R-Tree)

Nearest Neighbor Classifiers: Notions

Notions

- Distance Function: Defines the (dis-)similarity for pairs of objects
- *Decision Set*: The set of k nearest neighboring objects used in the decision rule

Decision Rule

Given the class labels of the objects from the decision set, how to derive the class label for the query object?

(Plain) Decision Rules

Given a query instance x_q and its k nearest neighboring training objects, $(x_i)_{i=1}^{k}$. Let $\delta(\cdot, \cdot)$ denote the *Kronecker delta* and $C_i = C(x_i)$ be the class label of x_i:

Nearest Neighbor Rule ($k = 1$)

Just inherit the class label of the nearest training object:

$$K(x_q) = C(x_1)$$

Majority Vote ($k \geq 1$)

Choose majority class, i.e. the class with the most representatives in the decision set:

$$K(x_q) = \underset{c_j \in \mathcal{C}}{\operatorname{argmax}} \sum_{i=1}^{k} \delta(C_i, c_j)$$

Weighted Decision Rules

Distance-weighted majority vote

Give more emphasis to closer objects within decision set, e.g.:

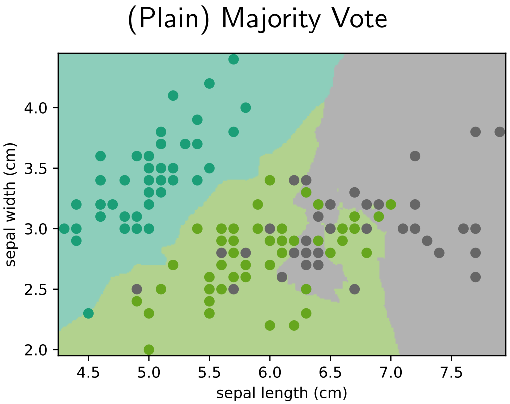
$$K(x_q) = \underset{c_j \in \mathcal{C}}{\operatorname{argmax}} \sum_{i=1}^{k} \frac{\delta(C_i, c_j)}{dist(x_i, x_q)^2}$$

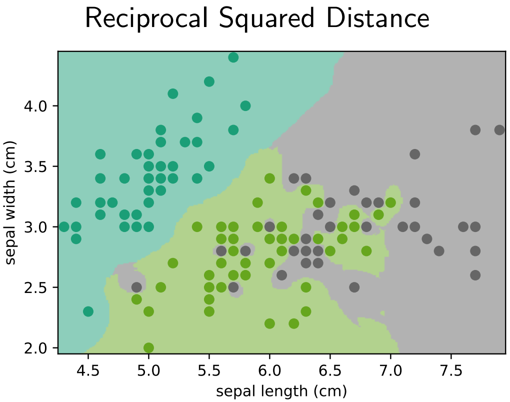
Class-weighted majority vote

Use inverse frequency of classes in the training set (a-priori probabilities):

$$K(x_q) = \underset{c_j \in \mathcal{C}}{\operatorname{argmax}} \frac{\sum_{i=1}^{k} \delta(C_i, c_j)}{\sum_{x \in O_{TR}} \delta(C(x), c_j)}$$

Example: Influence of Weighting (here: $k = 5$)


(Plain) Majority Vote


Reciprocal Squared Distance

NN Classifier: Parameter k

Choosing an appropriate k: Tradeoff between *overfitting* and *generalization*:

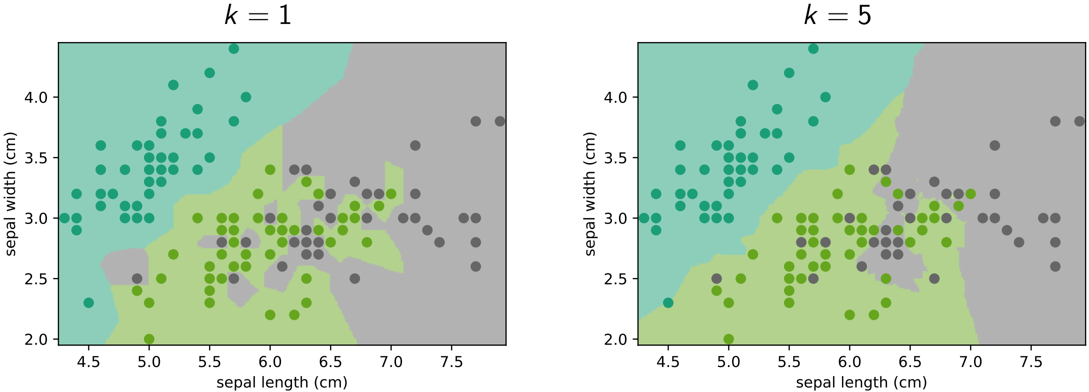
Influence of k

- k too small: High sensitivity against outliers
- k too large: Decision set contains many objects from other classes

Rules of Thumb

- Based on theoretical considerations: Choose k, such that it grows slowly with n, e.g. $k \approx \sqrt{n}$, or $k \approx \log n$
- Empirically, $1 \ll k < 10$ yields a high classification accuracy in many cases

Example: Majority Vote – Influence of k



NN Classifier: Variants

- *k-NN* Classifier: Consider the k nearest neighbors for the class assignment decision
- *Weighted k-NN* Classifier: Use weights for the classes of the k nearest neighbors
- *Mean-based NN* Classifier: Determine mean vector m_i for each class c_j (in training phase); Assign query object to the class c_j of the nearest mean vector m_i
- *Generalization*: Representative-based NN mean classifier; use more than one representative per class (cf. mixture models) – no longer just instance-based

NN Classifier: Discussion

Pro

- *Applicability*: Training data and distance function required only
- High classification *accuracy* in many applications
- Easy *incremental* adaptation to new training objects useful also for prediction
- *Robust* to noisy data by averaging *k*-nearest neighbors

Contra

- Naïve implementation is inefficient: Requires *k*-nearest neighbor query processing $\rightsquigarrow$ support by database techniques may help to reduce from $\mathcal{O}(n)$ to $\mathcal{O}(\log n)$
- Does not produce explicit knowledge about classes, but provides explanations
- Curse of dimensionality: Distance between neighbors could be dominated by irrelevant attributes $\rightsquigarrow$ apply dimensionality reduction first

Agenda

1. Supervised Learning

1.1 Introduction: Classification
1.2 Bayesian Classifiers
1.3 Linear Discriminant Functions
1.4 Support Vector Machines
1.5 Kernel Methods
1.6 Decision Tree Classifiers
1.7 Nearest Neighbor Classifiers
1.8 Ensemble Classification

Ensemble Classification

Problem

- No single classifier performs good on every problem (cf. theorem "There is no free lunch")
- For some techniques, small changes in the training set lead to very different classifiers

Idea

Improve performance by combining different classifiers $\rightsquigarrow$ ensemble classification. Different possibilities exist. Discussed here:

- Bagging (Bootstrap aggregation)
- Boosting

Bagging

How to obtain different classifiers?

Easiest way: Train the *same classifier* K on *different datasets*

Bagging (or Bootstrap Aggregation)

- Randomly select m different subsets from the training set
- On each subset, independently train a classifier K_i $(i = 1, \ldots, m)$
- Overall decision:

$$K(x) = sign\left(\frac{1}{m}\sum_{i=1}^{m} K_i(x)\right)$$

Boosting

Boosting

- Linear combination of several *weak* learners (different classifiers)
- Given m weak learners K_i and weights α_i for $i = 1, \ldots, m$
- Overall decision

$$K(x) = sign\left(\sum_{i=1}^{m} \alpha_i K_i(x)\right)$$

- *Important difference*: classifiers are trained in sequence!
- Repeatedly misclassified points are weighted stronger
- Example: meta-algorithm *AdaBoost* iteratively generates a chain of weak learners

Classification: Summary

	Linear Model	SVM	Decision Tree	kNN	Bayes
Model	hyperplane	hyperplane/ non-linear (kernel)	hierarchy of iso-oriented hyperplanes	no model	probability distribution func's
Data Types	vectors/kernels	vectors/kernels	categorical & vector	metric, kernels	arbitrary
Compactness	good (#dims)	good (#SV)	good (pruned)	no model	depends/model
Interpretability of Model	medium/low	medium/low	good	no model	depends/model
Interpretability of Decision	low	low	good (rules)	medium/good (examples)	medium/good (probabilities)
Training Time	high	medium	low/medium	no training	depends/model
Test Time	low / high (if high-dim)	low/medium	low	low w/ index, high w/o index	depends/model, often low
Robustness	low	high	low	high	high

Classification: Conclusion

- Classification is an extensively studied problem (mainly in statistics and machine learning)
- Classification is probably one of the most widely used data mining techniques with a lot of extensions
- Scalability is an important issue for database applications: thus combining classification with database techniques should be a promising topic
- Research directions: classification of complex data, e.g., text, spatial, multimedia, etc.;
  Example: kNN-classifiers rely on distances but do not require vector representations of data
- Results can be improved by ensemble classification