DATABASE SYSTEMS GROUP

Ludwig-Maximilians-Universität München
Institut für Informatik
Lehr- und Forschungseinheit für Datenbanksysteme

LMU

Knowledge Discovery in Databases

WiSe 2017/18

Kapitel 6: Klassifikation

Vorlesung: Prof. Dr. Peer Kröger

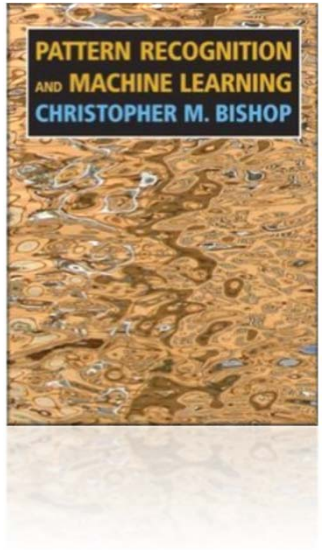
Übungen: Anna Beer, Florian Richter

Chapter 5: Classification

1) Introduction
   - Classification problem, evaluation of classifiers, numerical prediction
2) Bayesian Classifiers
   - Bayes classifier, naive Bayes classifier, applications
3) Linear discriminant functions & SVM
   1) Linear discriminant functions
   2) Support Vector Machines
   3) Non-linear spaces and kernel methods
4) Decision Tree Classifiers
   - Basic notions, split strategies, overfitting, pruning of decision trees
5) Nearest Neighbor Classifier
   - Basic notions, choice of parameters, applications
6) Ensemble Classification

Additional literature for this chapter

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



Introduction: Example

- Training data

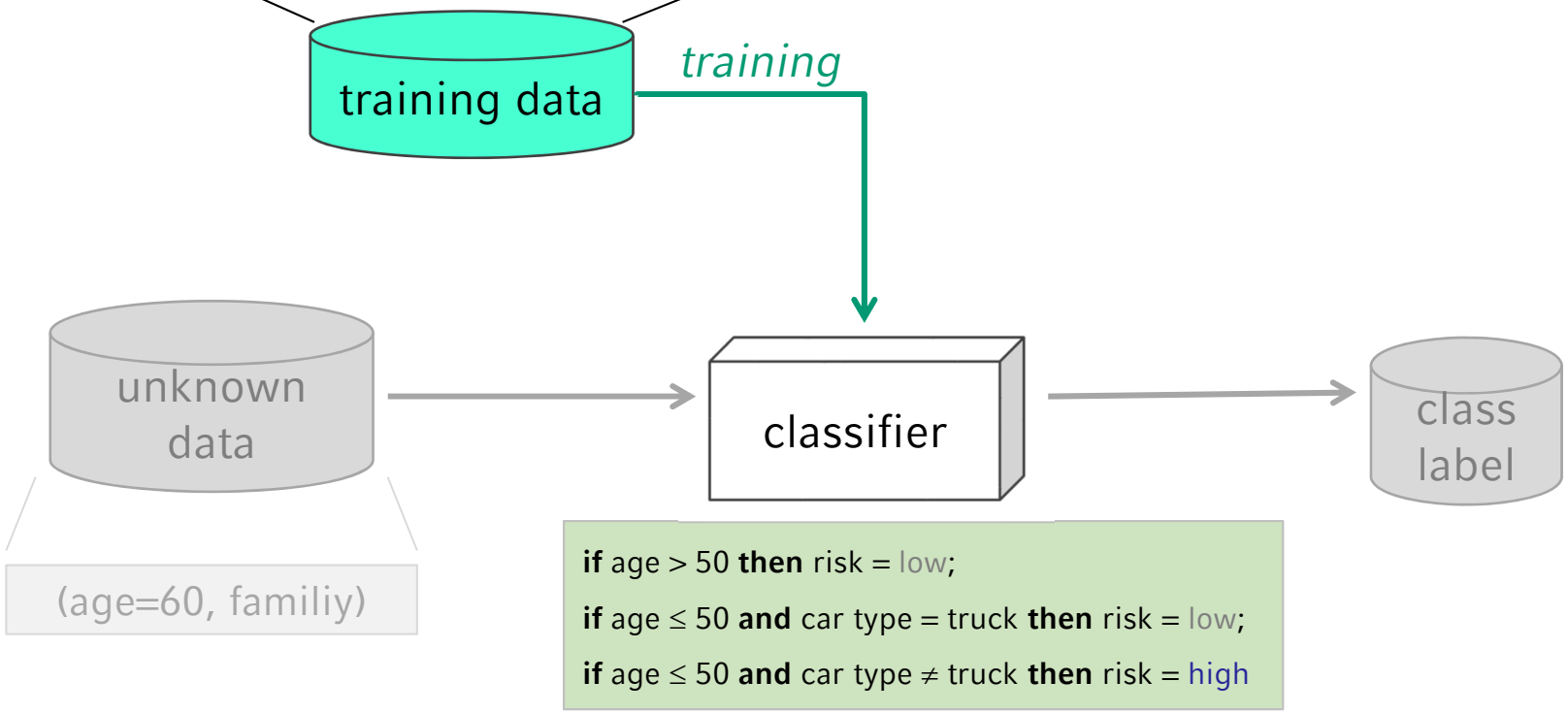
ID	age	car type	risk
1	23	family	high
2	17	sportive	high
3	43	sportive	high
4	68	family	low
5	32	truck	low

- Simple classifier

```
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.
```

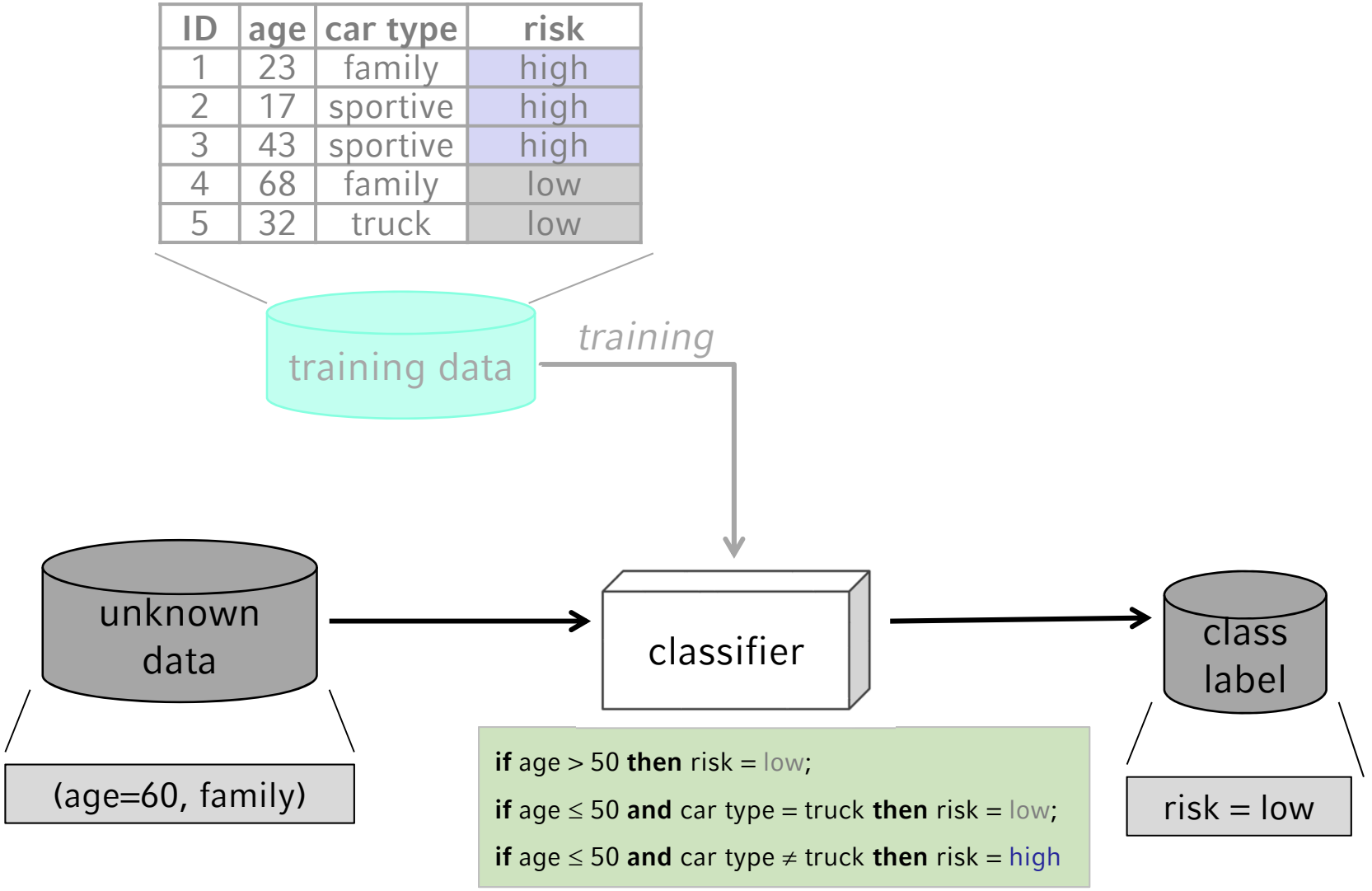
Classification: Training Phase (Model Construction)

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



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



Classification

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

- Formally,
  - a classifier K for a model $M(\theta)$ is a function $K_{M(\theta)}$: $D \rightarrow Y$, where
    - D: data space
      - *Often* d-dimensional space with attributes a_i, $i = 1, \ldots, d$ (not necessarily vector space)
      - Some other space, e.g. metric space
    - $Y = \{y_1, \ldots, y_k\}$: set of k distinct class labels y_j, $j = 1, \ldots, k$
    - $O \subseteq D$: set of training objects, $o = (o_1, \ldots, o_d)$, with known class labels $y \in Y$
  - Classification: application of classifier K on objects from D
- Model $M(\theta)$ is the "type" of the classifier, and θ are the model parameters
- Supervised learning: find/learn optimal parameters θ for the model $M(\theta)$ from the given training data

Supervised vs. Unsupervised Learning

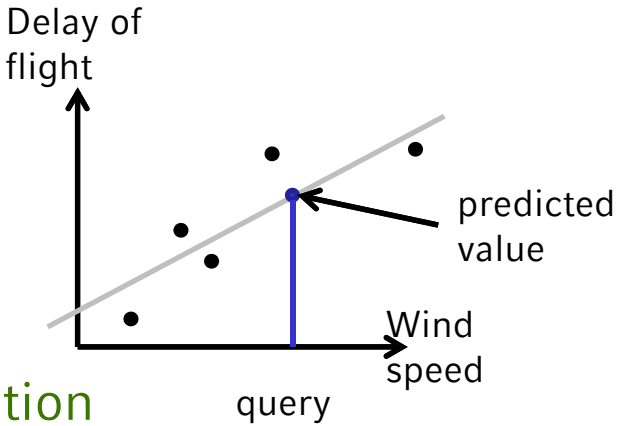
- Unsupervised learning (clustering)
  - The class labels of training data are unknown
  - Given a set of measurements, observations, etc. with the aim of establishing the existence of classes or clusters in the data
    - Classes (=clusters) are to be determined

- Supervised learning (classification)
  - Supervision: The training data (observations, measurements, etc.) are accompanied by labels indicating the class of the observations
    - Classes are known in advance (a priori)
  - New data is classified based on information extracted from the training set

[WK91] S. M. Weiss and C. A. Kulikowski. Computer Systems that Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems. Morgan Kaufman, 1991.

Numerical Prediction

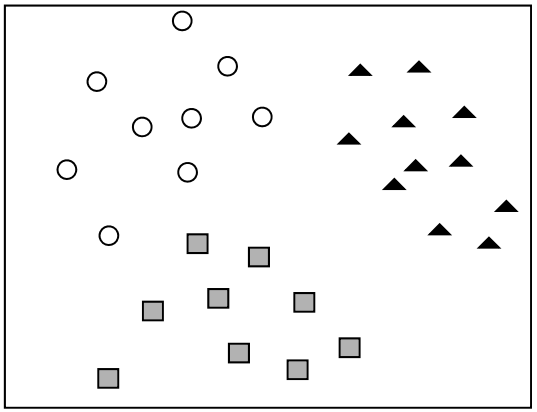
- Related problem to classification: numerical prediction
  - Determine the numerical value of an object
  - Method: e.g., regression analysis
  - Example: prediction of flight delays



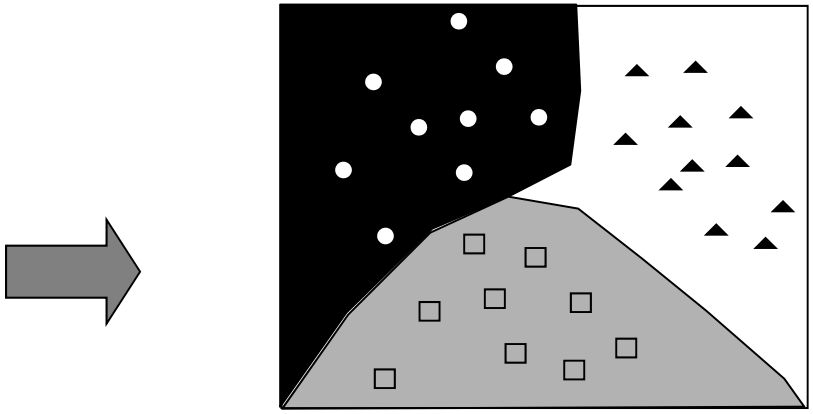
- Numerical prediction is *different* from classification
  - Classification refers to predict categorical class label
  - Numerical prediction models continuous-valued functions
- Numerical prediction is *similar* to classification
  - First, construct a model
  - Second, use model to predict unknown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

Überblick über Klassifikationsmethoden

Trainingsmenge mit 3 Klassen



3 Klassenbereiche (weiß, grau, schwarz)

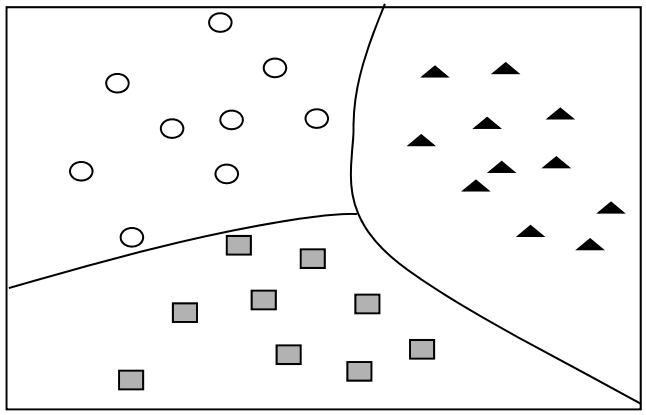


Klassifikatoren legen beim Training im Allgemeinen ***Klassengrenzen*** fest.

Aber: Es gibt viele Methoden, Klassengrenzen aus Trainingsdaten abzuleiten.

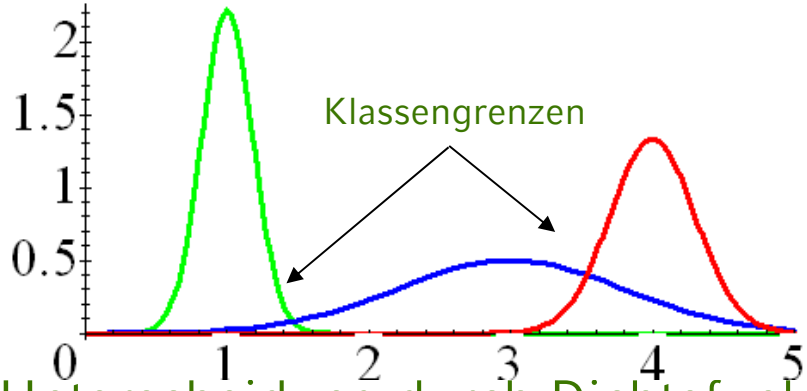
=> Unterschiedliche Klassifikatoren (Modeltypen)
(statistische Kl., Entscheidungsbäume, Support Vektor Maschinen,
kNN-Klassifikatoren, neuronale Netze, ...)

Motivation der Klassifikationsmethoden(1)

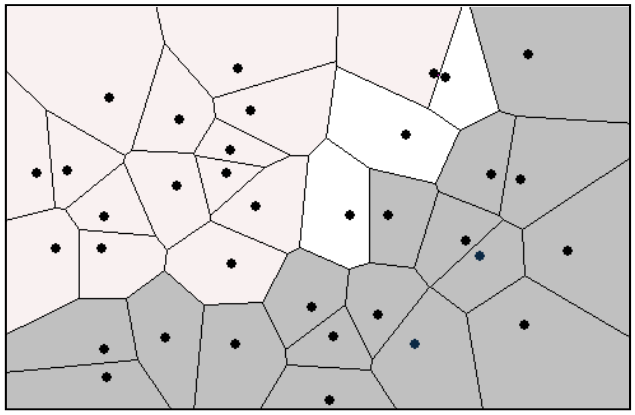


Bayes Klassifikatoren

1-dimensionale Projektion



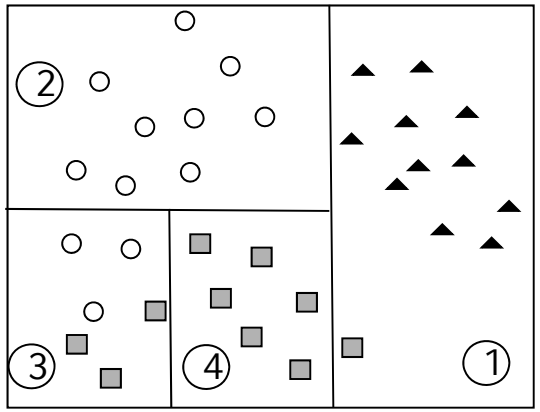
Unterscheidung durch Dichtefunktionen.



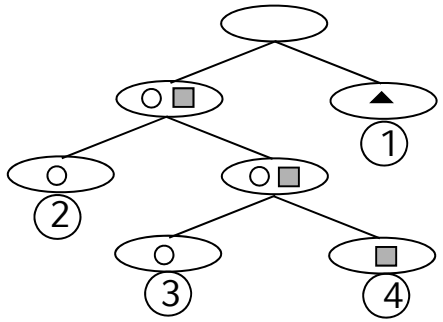
NN-Klassifikator

Unterscheidung durch Voronoi-Zellen
(1 nächster Nachbar Klassifikator)

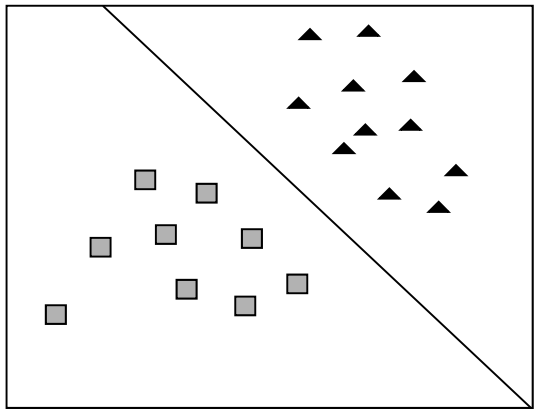
Motivation der Klassifikationsmethoden(2)



Entscheidungsbäume



Festlegen der Grenzen durch rekursive Unterteilung in Einzeldimension.



Support Vektor Maschinen

Grenzen über lineare Separation

Intuition und Grundannahmen

- Es gibt einen (natürlichen, technischen, sozial-dynamischen, ...) Prozess (im statistischen Sinne), der die beobachteten Daten O als Teilmenge der möglichen Daten D erzeugt bzw. für ein $x \in D$ eine Klassenentscheidung für eine Klasse $y_i \in Y$ trifft.
- Die beobachteten Daten sind Beispiele für die Wirkung des Prozesses.
- Es gibt eine ideale (unbekannte) Funktion, die einen Beispiel-Datensatz auf die zugehörige Klasse abbildet:

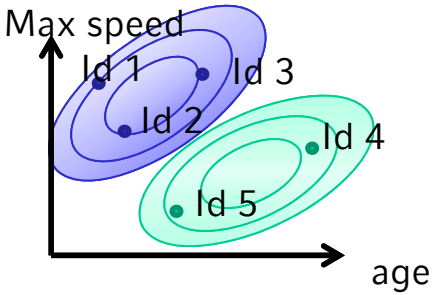
  $f: D \rightarrow Y$
- Aufgabe des Lernalgorithmus ist es, eine möglichst gute Approximation K von f zu finden: eine Hypothese, die die gegebenen Daten erklärt (und im Idealfall hilft, den Prozess zu verstehen, der die Daten erzeugt hat).

  (vgl. induktives Lernen, aber keine vollständige Induktion)

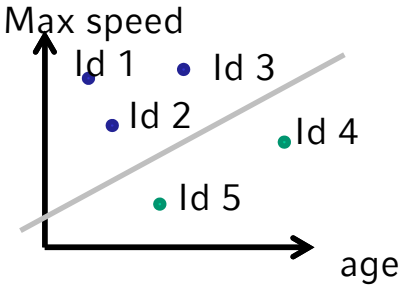
Goals of this lecture

1. Introduction of different classification models

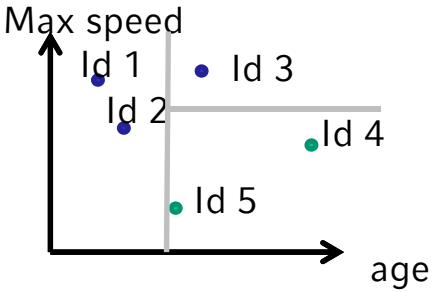
ID	age	car type	Max speed	risk
1	23	family	180	high
2	17	sportive	240	high
3	43	sportive	246	high
4	68	family	173	low
5	32	truck	110	low



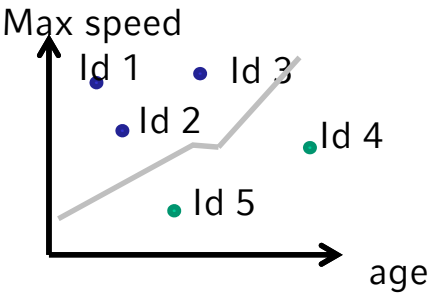
Bayes classifier



Linear discriminant function & SVM



Decision trees



k-nearest neighbor

2. Learning techniques for these models (i.e. optimizing their parameters)

Quality Measures for Classifiers

But before: How do we know that a classifier is "good"?

- Classification accuracy or classification error (complementary)

But also very important measures are:

- 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

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

WHY?

- We aim at a good accuracy for all objects from *D*, but the quality for objects from *D**O* is unknown in general.
- The classifier *K* is optimized for *O*
- Applying *K* on objects from *O* will typically provide better results than applying *K* to objects from *D**O*

- In other words: the results from the application of *K* to *O* is not a relaistic view for the performance of applaying *K* to *D*
- This is called ***Overfitting***

Evaluation of Classifiers – Notions

Solution:

- *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 as new and compare results 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

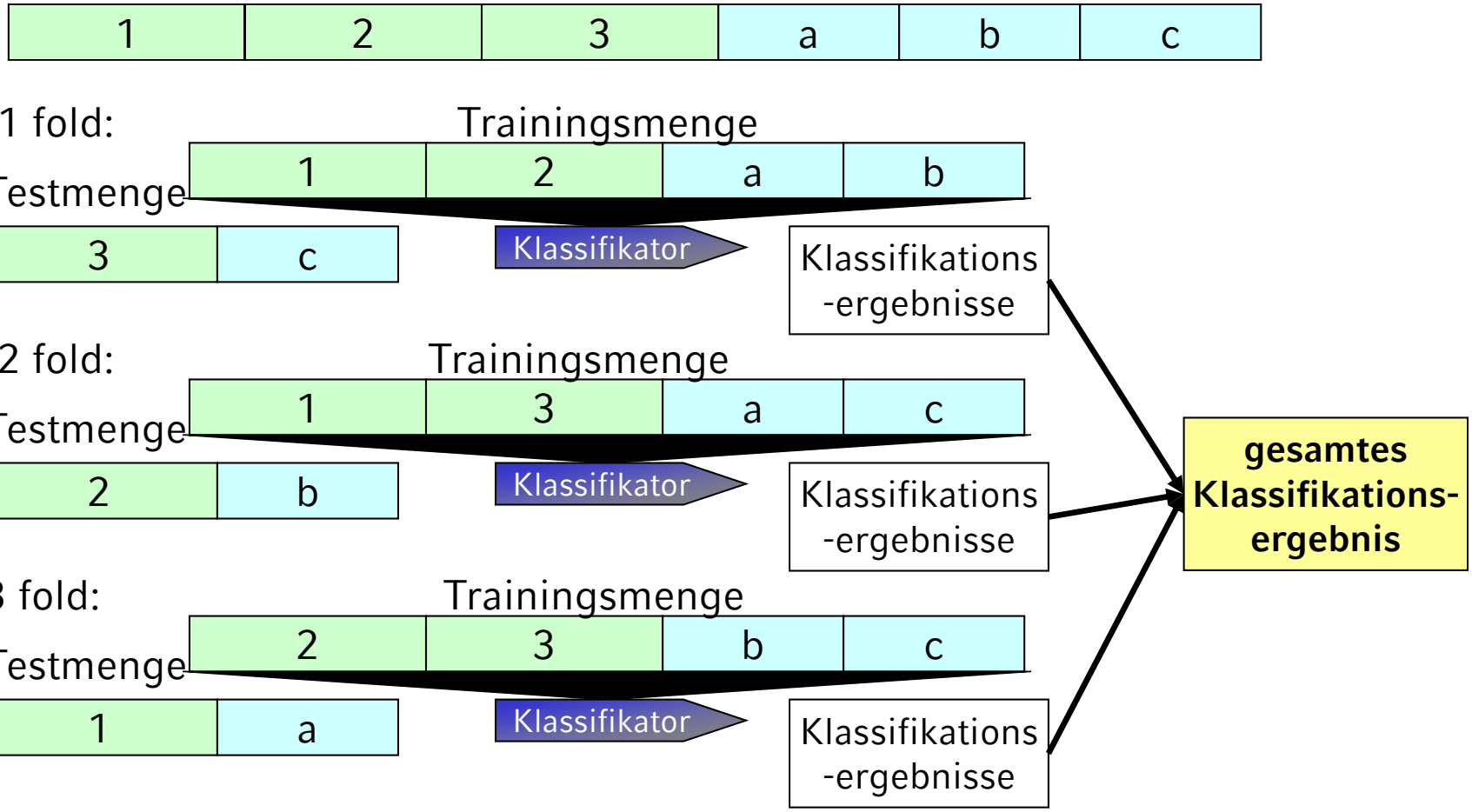
Evaluation of Classifiers – Cross Validation

- *m*-fold *Cross Validation*
  - Decompose data set evenly into *m* subsets of (nearly) equal size
  - Iteratively use $m-1$ partitions as training data and the remaining single partition as test data.
  - Combine the *m* classification accuracy values to an overall classification accuracy, and combine the *m* generated models to an overall model for the data.
- *Leave-one-out* (aka "Jackknife") is a 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

Example: Cross Validation

Ablauf 3-fache Überkreuzvalidierung (3-fold Cross Validation)

Sei n = 3 : Menge aller Daten mit Klasseninformation die zu Verfügung stehen



Evaluation of Classifiers – Cross Validation

Additional requirement: ***stratified*** folds

- Proportionality of classes should be preserved
- Minimum requirement: each class should be present in *O* (i.e. in each fold)
- Better: class distributions in training and test set should represent the class distribution in *D* (or at least in *O*)

- Leaf-one-out is by designed NOT stratified

- Standard: 10-fold, stratified cross validation, repeated 10 times

Bewertung von Klassifikatoren

Alternative: Bootstrap

bilde Trainingsmenge durch Ziehen mit Zurücklegen.

- jedes Sample hat die gleiche Größe wie die ursprüngliche Trainingsmenge
- ein Sample enthält durchschnittlich 63% der Ausgangsbeispiele (einige mehrfach, etwa 37% gar nicht):
  - ein einzelnes Beispiel in einem Datensatz mit *n* Beispielen hat bei jedem Ziehen die Chance *1/n* gezogen zu werden, wird also mit Wahrscheinlichkeit *1-1/n* **nicht** gezogen
  - nach *n*-mal Ziehen ist ein bestimmtes Element mit Wahrscheinlichkeit $\left(1-\frac{1}{n}\right)^n$ nicht gezogen worden
  - für große *n* ist $\left(1-\frac{1}{n}\right)^n \approx e^{-1} \approx 0.368$
- daher auch der Name "0.632 bootstrap" für diese Sampling-Methode

Achtung: Im Allgemeinen eher optimistische Fehlerschätzung

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:

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

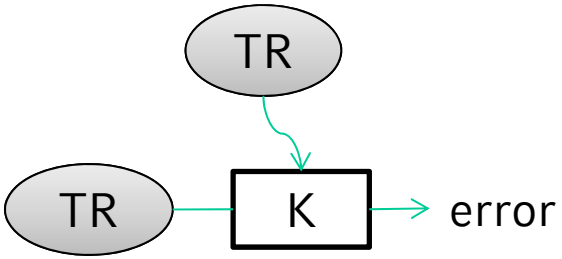
  - *Classification Error* of *K:*

$$F_T(K) = \frac{|\{o \in T, K(o) \neq C(o)\}|}{|T|}$$

Quality Measures: Accuracy and Error

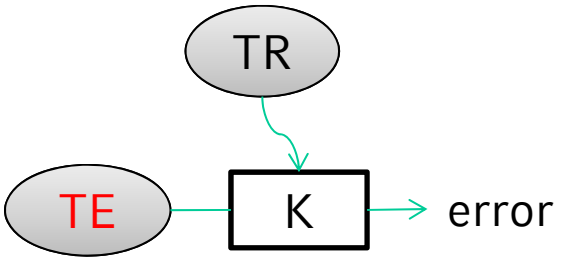
- Let K be a classifier
- Let $TR \subseteq O$ be the training set – used to build the classifier
- Let $TE \subseteq O$ be the test set – used to evaluate the classifier
  - *resubstitution error* of K:

$$F_{TR}(K) = \frac{|\{o \in TR, K(o) \neq C(o)\}|}{|TR|}$$



  - (true) *classification error* of K:

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



Quality Measures: Confusion Matrix

- Results on the test set: confusion matrix

classified as ...

correct class label ...	class1	class 2	class 3	class 4	other
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
other	3	1	9	15	13

correctly classified objects

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

Quality Measures: Class of Interest

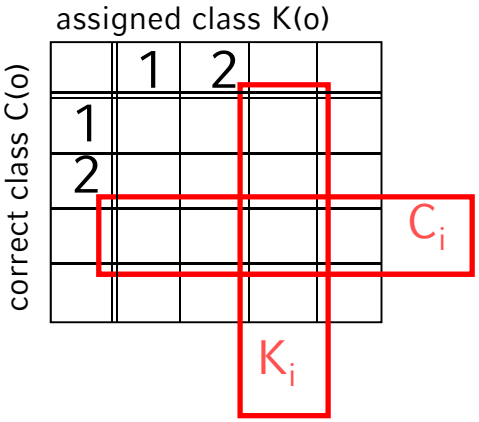
- Consider a classification problem with a so-called *class of interest*, i.e. where the main focus lies on classifying a particular class correctly.
- Example: A medical researcher wants to develop a test for early recognition of breast cancer
  - $Y = \{-1, 1\}$, where $y_i = 1$ corresponds to the women that have breast cancer
  - In this case, the 'data-objects' with $y_i = 1$ would be called *positive objects* or *positive tuples* while the others are called *negative objects*
- This leads to additional accuracy measures:
  - recall (true positive rate): $\frac{TP}{P}$
  - precision: $\frac{TP}{TP+FP}$
  - specificity (true negative rate): $\frac{TN}{N}$

P: positives
N: negatives
TP: true positives
FP: false positives

Quality Measures: Precision and Recall

- *Recall:* fraction of test objects of class *i*, which have been identified correctly
- Let $C_i = \{o \in TE \mid C(o) = i\}$, then

$$\text{Recall}_{TE}(K, i) = \frac{|\{o \in C_i | K(o) = C(o)\}|}{|C_i|}$$

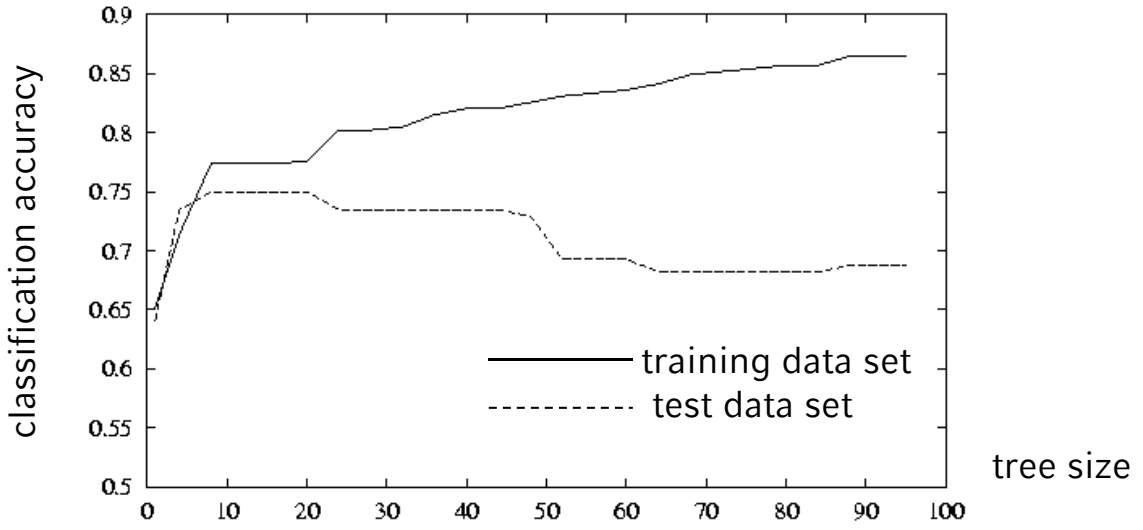


- *Precision:* fraction of test objects assigned to class *i*, which have been identified correctly
- Let $K_i = \{o \in TE \mid K(o) = i\}$, then

$$\text{Precision}_{TE}(K, i) = \frac{|\{o \in K_i | K(o) = C(o)\}|}{|K_i|}$$

Overfitting

- Characterization of overfitting:
  There are two classifiers K and K' for which the following holds:
  - on the training set, K has a smaller error rate than K'
  - on the overall test data set, K' has a smaller error rate than K
- Example: Decision Tree



generalization ← ***classifier*** → **specialization**

"overfitting"

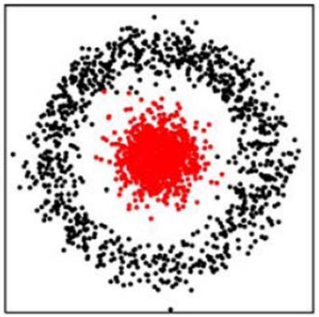
Overfitting (2)

- *Overfitting*
  - occurs when the classifier is too optimized to the (noisy) training data
    Worst case: memorization ("Auswendiglernen")
  - 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* and *erroneous* training data; in particular, remove 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

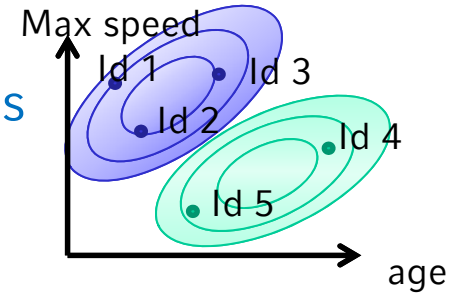
- *Underfitting*
  - Occurs when the classifiers model is too simple, e.g. trying to separate classes linearly that can only be separated by a quadratic surface
  - happens seldomly



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

Chapter 6: Classification

1) Introduction
   - Classification problem, evaluation of classifiers, prediction
2) Bayesian Classifiers
   - Bayes classifier, naive Bayes classifier, applications
3) Linear discriminant functions & SVM
   1) Linear discriminant functions
   2) Support Vector Machines
   3) Non-linear spaces and kernel methods
4) Decision Tree Classifiers
   - Basic notions, split strategies, overfitting, pruning of decision trees
5) Nearest Neighbor Classifier
   - Basic notions, choice of parameters, applications
6) Ensemble Classification



Bayes Classification

- Probability based classification
  - Based on likelihood of observed data, estimate explicit probabilities for classes
  - Classify objects depending on costs for possible decisions and the probabilities for the classes
- Incremental
  - Likelihood functions built up from classified data
  - Each training example can incrementally increase/decrease the probability that a hypothesis (class) is correct
  - Prior knowledge can be combined with observed data.
- Good classification results in many applications

Bayes' theorem

- Probability theory:
  - Conditional probability: $P(A|B) = \frac{P(A \wedge B)}{P(B)}$ ("probability of A given B")
  - Product rule: $P(A \wedge B) = P(A|B) \cdot P(B)$
- Bayes' theorem
  - $P(A \wedge B) = P(A|B) \cdot P(B)$
  - $P(B \wedge A) = P(B|A) \cdot P(A)$
  - Since

$$P(A \wedge B) = P(B \wedge A) \Rightarrow$$
$$P(A|B) \cdot P(B) = P(B|A) \cdot P(A) \Rightarrow$$

Bayes' theorem

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

Bayes Classifier

- Bayes rule: $p(c_j|o) = \frac{p(o|c_j) \cdot p(c_j)}{p(o)}$
- We are interested in maximizing this, i.e.

$$\underset{c_j \in C}{\text{argmax}}\{p(c_j|o)\} = \underset{c_j \in C}{\text{argmax}}\left\{\frac{p(o|c_j) \cdot p(c_j)}{p(o)}\right\} = \underset{c_j \in C}{\text{argmax}}\{p(o|c_j) \cdot p(c_j)\}$$

Value of $p(o)$ is constant and does not change the result.

- Final decision rule for the *Bayes classifier*

$$K(o) = c_{max} = \underset{c_j \in C}{\text{argmax}}\{p(o|c_j) \cdot p(c_j)\}$$

- Estimate the apriori probabilities $p(c_j)$ of classes c_j by using the observed frequency of the individual class labels c_j in the training set, i.e., $p(c_j) = \frac{N_{c_j}}{N}$
- How to estimate the values of $p(o|c_j)$?

Density estimation techniques

- Given a database DB, how to estimate conditional probability $p(o|c_j)$?
  - Parametric methods: e.g. single Gaussian distribution
    - Compute by maximum likelihood estimators (MLE), etc.
  - Non-parametric methods: Kernel methods
    - Parzen's window, Gaussian kernels, etc.
  - Mixture models: e.g. mixture of Gaussians (GMM = Gaussian Mixture Model)
    - Compute by e.g. EM algorithm
- Curse of dimensionality often lead to problems in high dimensional data
  - Density functions become too uninformative
  - Solution:
    - Dimensionality reduction
    - Usage of statistical independence of single attributes (extreme case: naïve Bayes)

Naïve Bayes Classifier (1)

- Assumptions of the naïve Bayes classifier
  - Objects are given as d-dim. vectors, $o = (o_1, \ldots, o_d)$
  - For any given class c_j the attribute values o_i are ***conditionally independent***, i.e.

$$p(o_1, \ldots, o_d | c_j) = \prod_{i=1}^{d} p(o_i | c_j) = p(o_1 | c_j) \cdot \ldots \cdot p(o_d | c_j)$$

- Decision rule for the *naïve Bayes classifier*

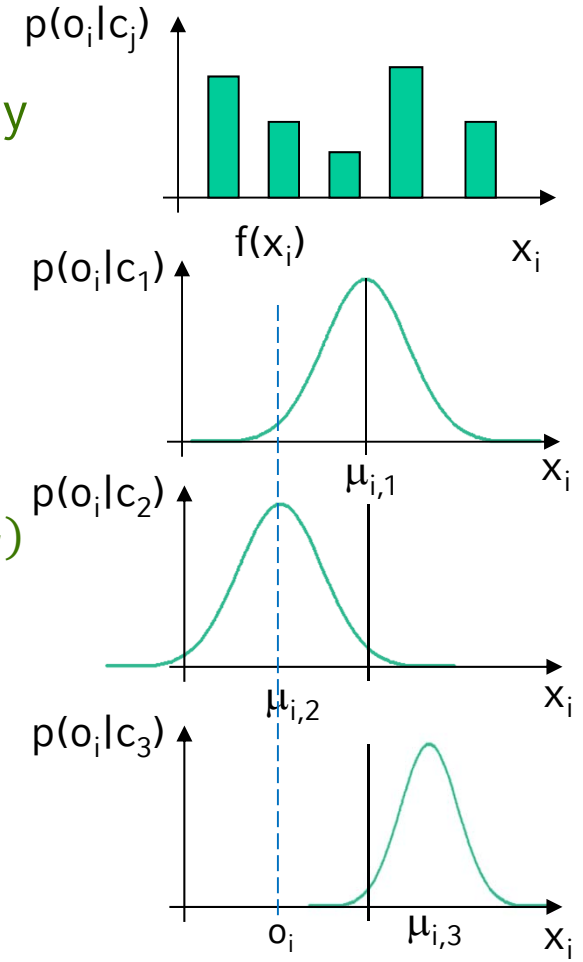
$$K_{naive}(o) = \underset{c_j \in C}{\operatorname{argmax}} \left\{ p(c_j) \cdot \prod_{i=1}^{d} p(o_i | c_j) \right\}$$

Naïve Bayes Classifier (2)

- Independency assumption: $p(o_1, \ldots, o_d | c_j) = \prod_{i=1}^{d} p(o_i | c_j)$
- If i-th attribute is categorical:
  $p(o_i|c_j)$ can be estimated as the relative frequency of samples having value x_i as i-th attribute in class c_j in the training set
- If i-th attribute is continuous:
  $p(o_i|c_j)$ can, for example, be estimated through:
  - Gaussian density function determined by $(\mu_{i,j}, \sigma_{i,j})$

$$\rightarrow p(o_i|c_j) = \frac{1}{\sqrt{2\pi}\sigma_{i,j}} e^{-\frac{1}{2}\left(\frac{o_i - \mu_{i,j}}{\sigma_{i,j}}\right)^2}$$

- Computationally easy in both cases



Example: Naïve Bayes Classifier

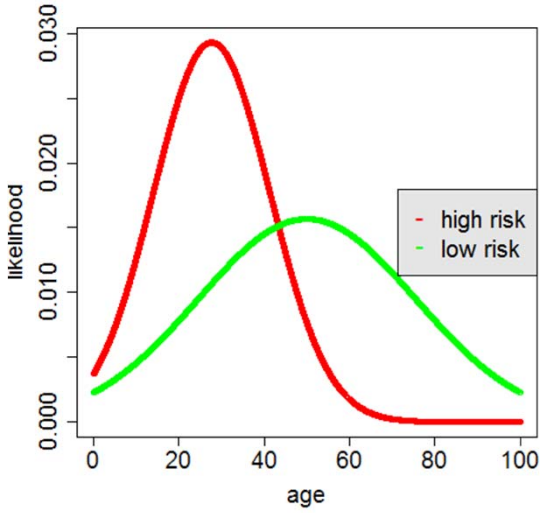
- Model setup:
  - Age $\sim N(\mu, \sigma)$ (normal distribution)
  - Car type $\sim$ relative frequencies
  - Max speed $\sim N(\mu, \sigma)$ (normal distribution)

ID	age	car type	Max speed	risk
1	23	family	180	high
2	17	sportive	240	high
3	43	sportive	246	high
4	68	family	173	low
5	32	truck	110	low

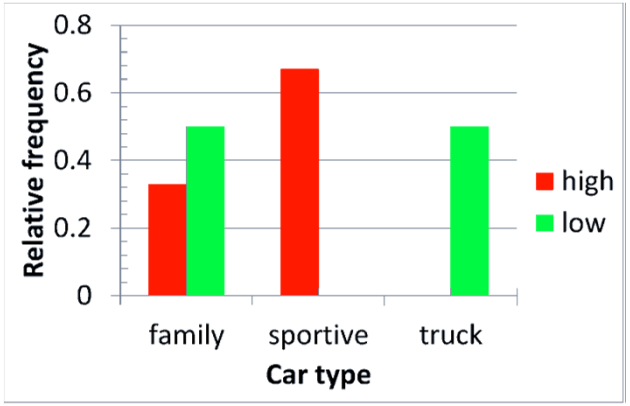
Age:

$\mu_{age}^{high} = 27.67, \sigma_{age}^{high} = 13.61$

$\mu_{age}^{low} = 50, \sigma_{age}^{low} = 25.45$



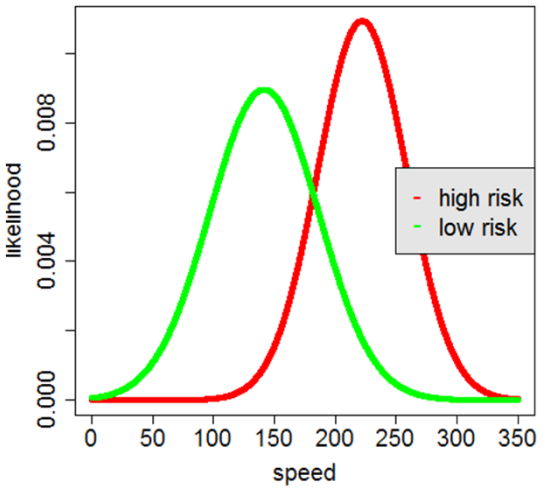
Car type:



Max speed:

$\mu_{speed}^{high} = 222, \sigma_{speed}^{high} = 36.49$

$\mu_{speed}^{low} = 141.5, \sigma_{speed}^{low} = 44.54$



Example: Naïve Bayes Classifier (2)

- Query: q = (age = 60, car type = family, max speed = 190)
- Calculate the probabilities for both classes:

With:
$1 = p(high|q) + p(low|q)$

$$p(high|q) = \frac{p(q|high) \cdot p(high)}{p(q)}$$

$$= \frac{p(age = 60|high) \cdot p(car\ type = family|high) \cdot p(\max speed = 190|high) \cdot p(high)}{p(q)}$$

$$= \frac{N(27.67, 13.61|60) \cdot \frac{1}{3} \cdot N(222,36.49|190) \cdot \frac{3}{5}}{p(q)} = 15.32\%$$

$$p(low|q) = \frac{p(q|low) \cdot p(low)}{p(q)}$$

$$= \frac{p(age = 60|low) \cdot p(car\ type = family|low) \cdot p(\max speed = 190|low) \cdot p(low)}{p(q)}$$

$$= \frac{N(50, 25.45|60) \cdot \frac{1}{2} \cdot N(141.5,44.54|190) \cdot \frac{2}{5}}{p(q)} = 84,68\%$$

← Classifier decision

Bayesian Classifier

- Assuming dimensions of $o = (o_1 \ldots o_d)$ are not independent
- Assume multivariate normal distribution (=Gaussian)

$$P(o \mid C_j) = \frac{1}{(2\pi)^{d/2} \mid \Sigma_j \mid^{1/2}} e^{-\frac{1}{2}(o-\mu_j)\Sigma_j^{-1}(o-\mu_j)^T}$$

with

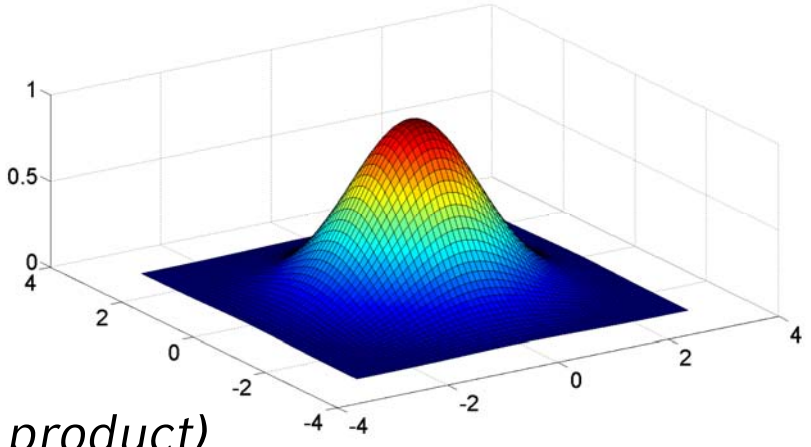
μ_j mean vector of class C_j

N_j is number of objects of class C_j

Σ_j is the $d \times d$ covariance matrix

$$\Sigma_j = \frac{1}{N_j - 1} \sum_{i=1}^{N_j} (o_i - \mu_j)^T \cdot (o_i - \mu_j)$$

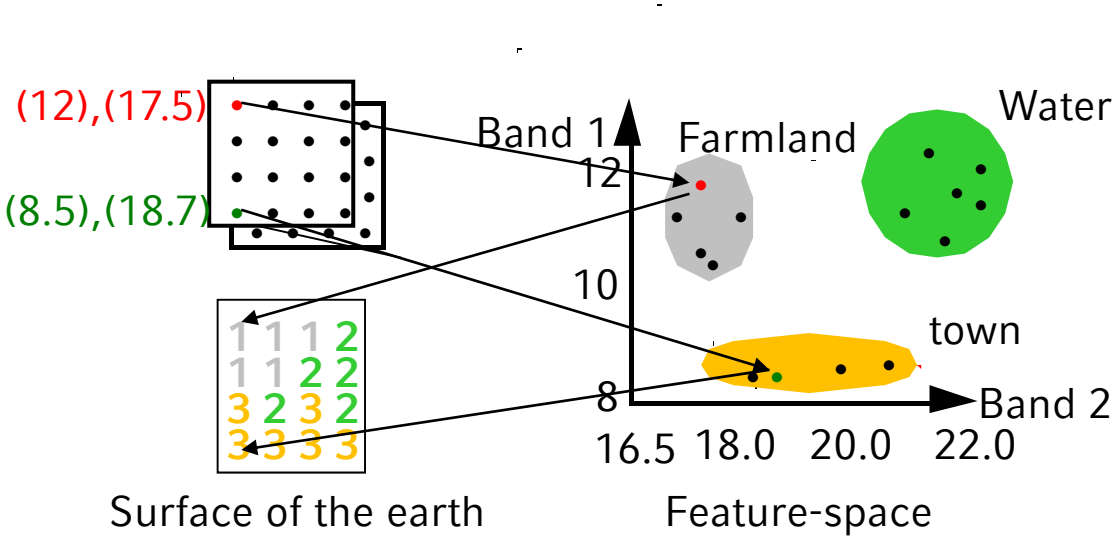
(outer product)

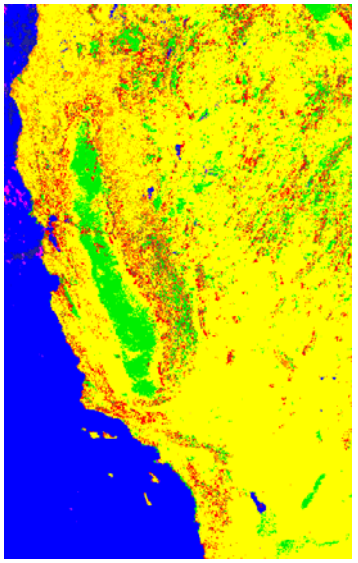


$|\Sigma_j|$ is the determinant of Σ_j and Σ_j^{-1} the inverse of Σ_j

Example: Interpretation of Raster Images

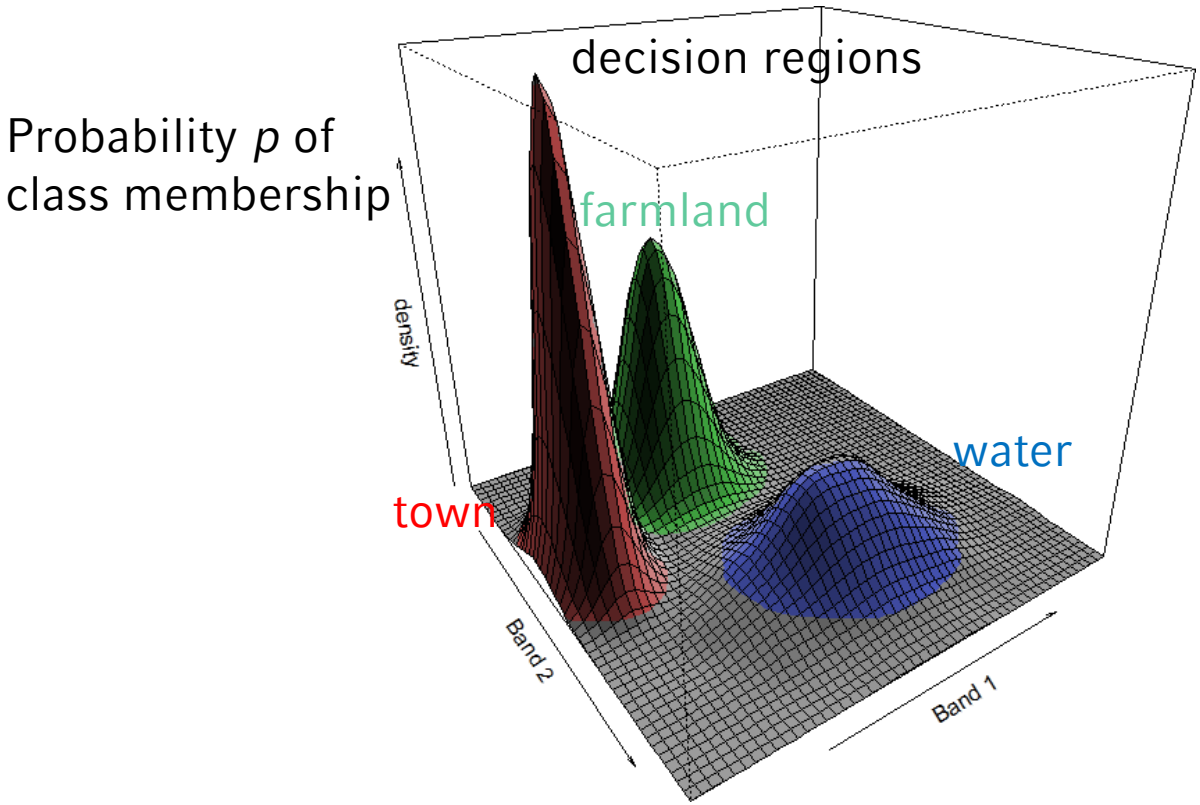
- Scenario: automated interpretation of raster images
  - Take an image from a certain region (in *d* different frequency bands, e.g., infrared, etc.)
  - Represent each pixel by *d* values: (o_1, …, o_d)
- Basic assumption: different surface properties of the earth („landuse“) follow a characteristic reflection and emission pattern





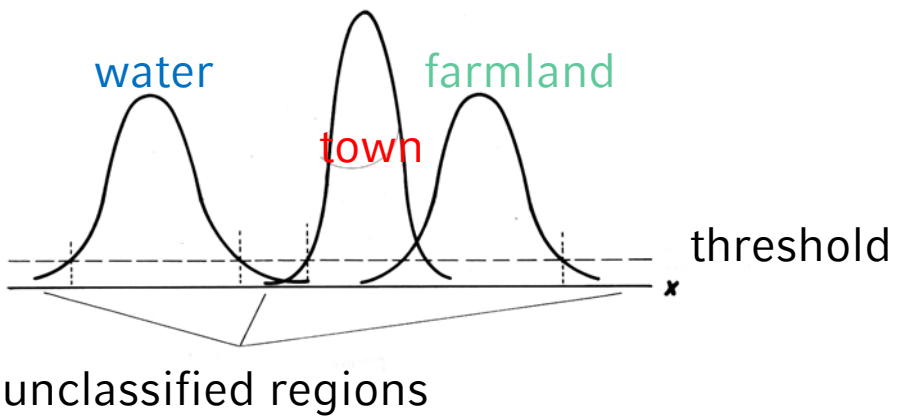
Example: Interpretation of Raster Images

- Application of the Bayes classifier
  - Estimation of the p(*o* | *c*) without assumption of conditional independence
  - Assumption of d-dimensional normal (= Gaussian) distributions for the value vectors of a class



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



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…

- … 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

- Nevertheless, naïve Bayes is very often (surprisingly) good in practice