Knowledge Discovery in Databases
SS 2016

Chapter 6: Classification

Lecture: Prof. Dr. Thomas Seidl

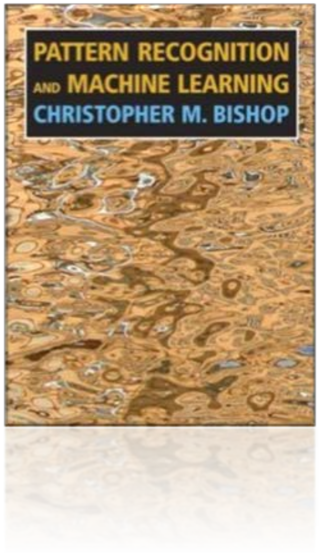
Tutorials: Julian Busch, Evgeniy Faerman,
Florian Richter, Klaus Schmid

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

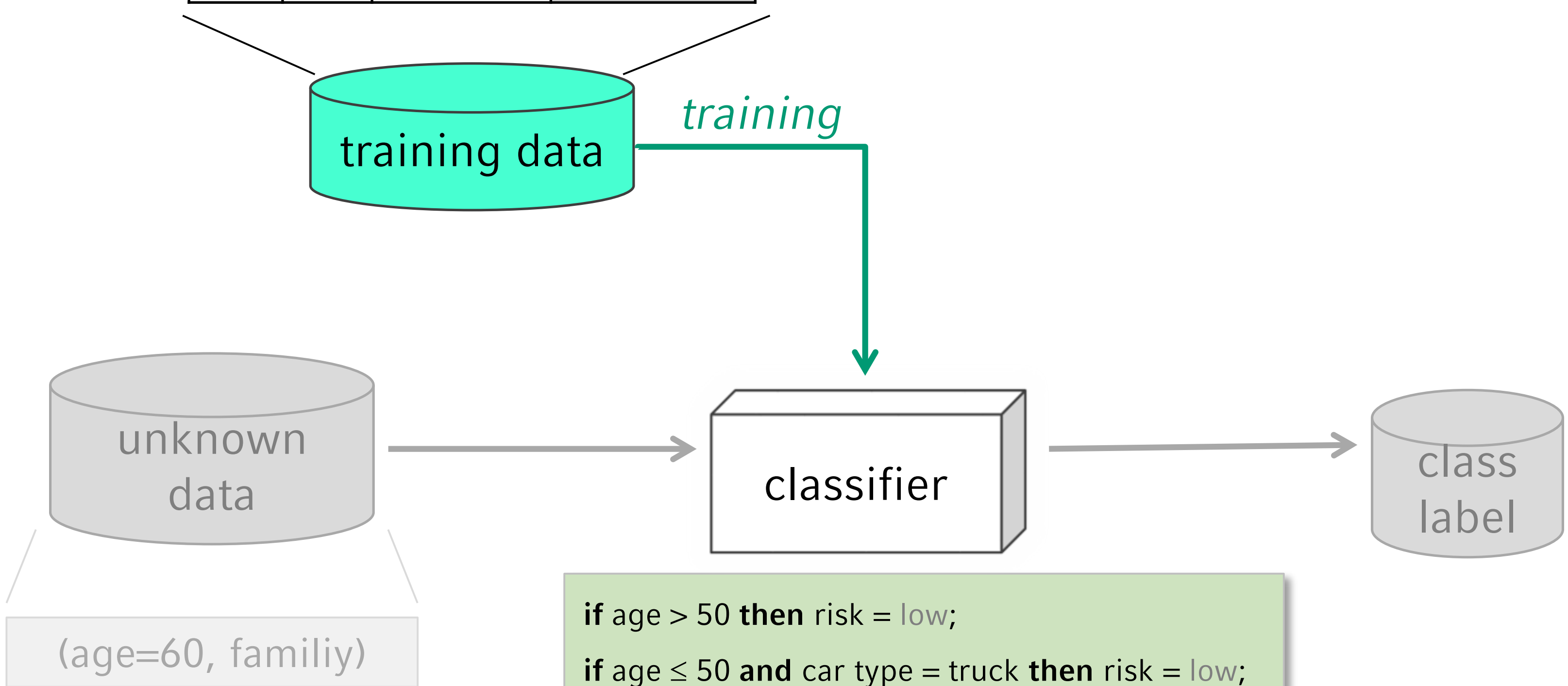
if age > 50 **then** risk = low;

if age $\leq$ 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



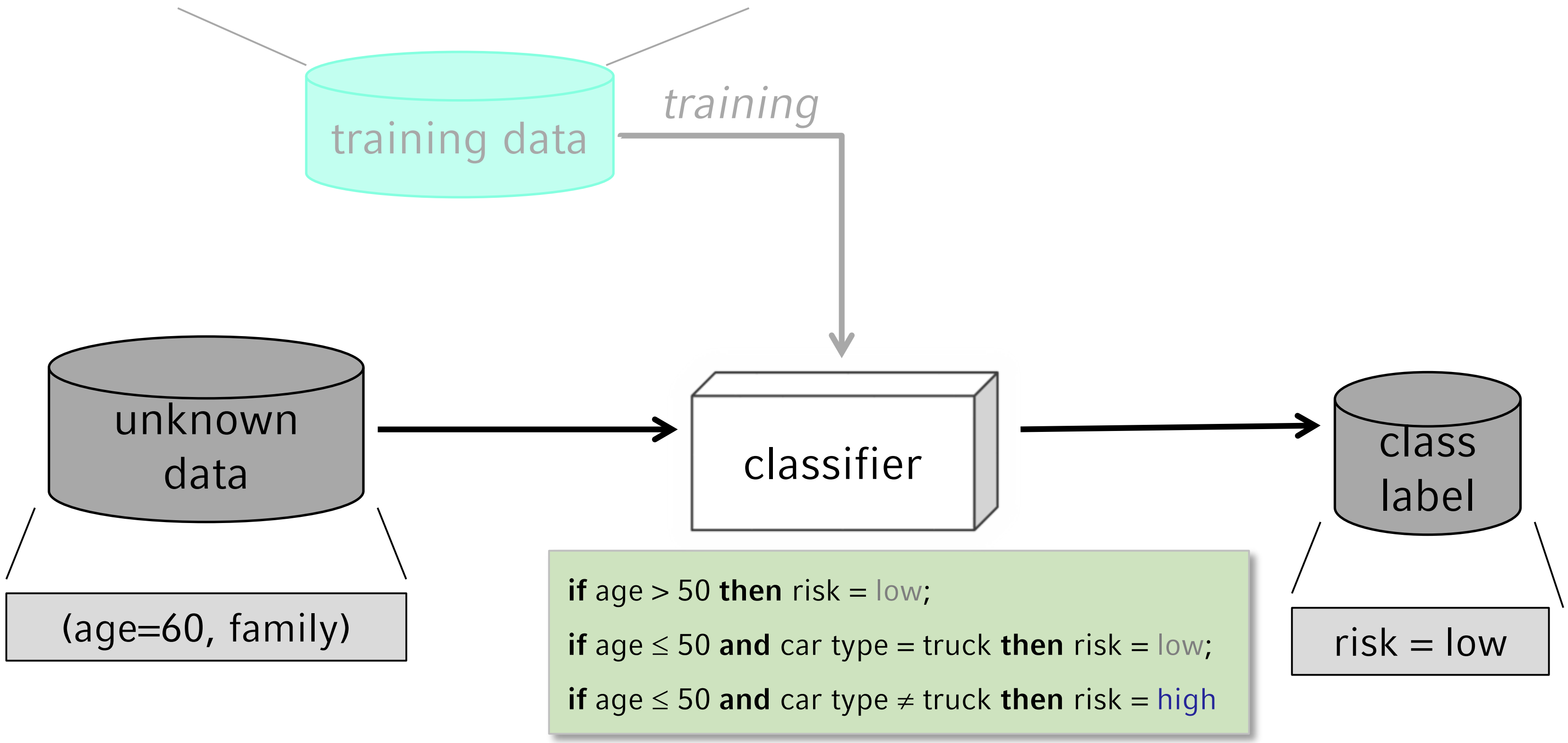
if age > 50 **then** risk = low;

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

if age ≤ 50 **and** car type ≠ 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



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, $i = 1, \dots, d$ (not necessarily vector space)
      - Some other space, e.g. metric space
    - $Y = \{y_1, \dots, y_k\}$: set of k distinct class labels y_j, $j = 1, \dots, k$
    - $O \subseteq D$: set of training objects, $o = (o_1, \dots, o_d)$, with known class labels $y \in Y$
  - Classification: application of classifier K on objects from $D - O$
- 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

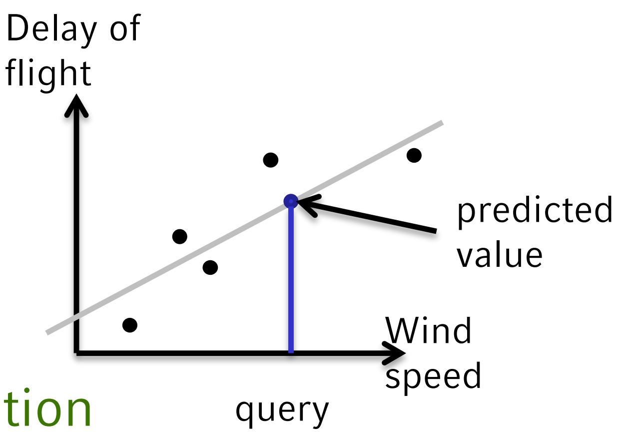
Supervised vs. Unsupervised Learning

- 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

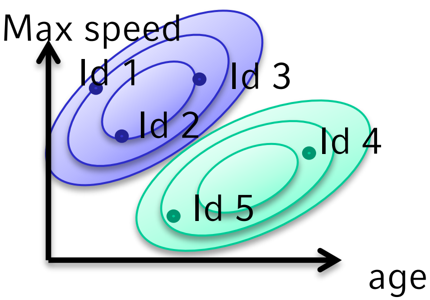


- Numerical prediction is *different* from classification
  - Classification refers to predict categorical class label
  - Numerical prediction models continuous-valued functions
- Numerical prediction is *similar* to classification
  - First, construct a model
  - Second, use model to predict unknown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

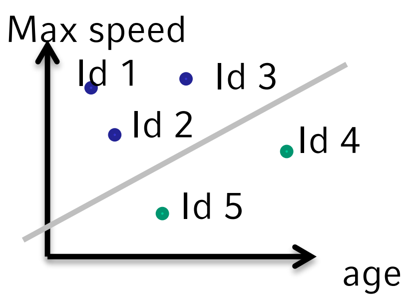
Goals of this lecture

1. Introduction of different classification models

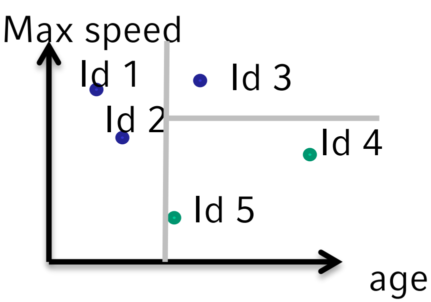
ID	ag e	car type	Max speed	risk
1	23	family	180	high
2	17	sportive	240	high
3	43	sportive	246	high
4	68	family	173	low
5	32	truck	110	low



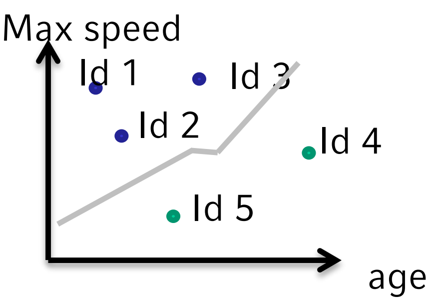
Bayes classifier



Linear discriminant function & SVM



Decision trees



k-nearest neighbor

2. Learning techniques for these models

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

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
  - due to 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 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* is a special case of cross validation (m=n)
  - For each of the objects o in the data set O:
    - Use set $O \backslash \{o\}$ 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 $C(o)$ 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:

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

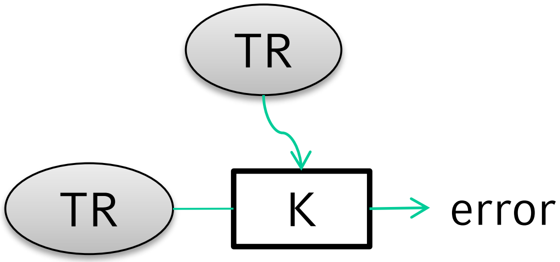
  - *Classification Error* of *K:*

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

Quality Measures: Accuracy and Error

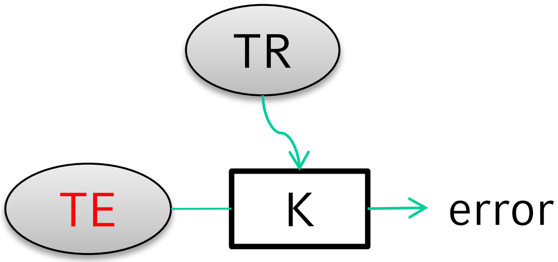
- Let K be a classifier
- Let $TR \subseteq O$ be the training set – used to build the classifier
- Let $TE \subseteq O$ be the test set – used to test the classifier
  - *resubstitution error* of K:

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



  - (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 ... (columns); correct class label ... (rows)

	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

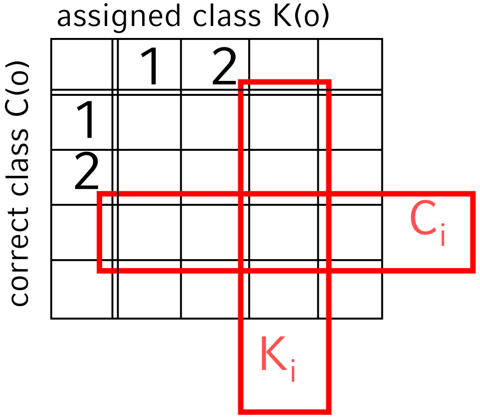
correctly classified objects

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

Quality Measures: 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

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

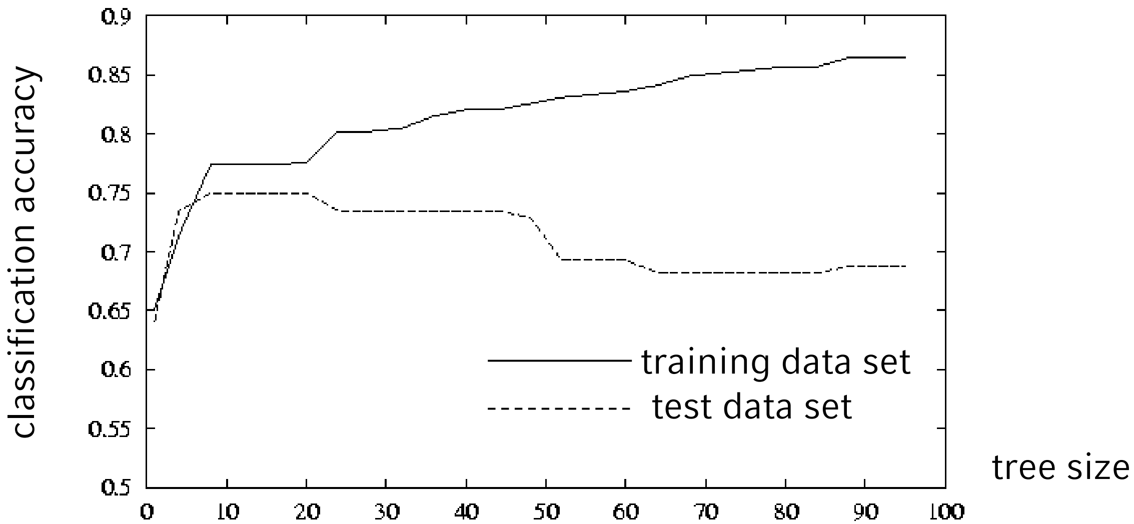


- *Precision:* fraction of test objects assigned to class *i*, which have been identified correctly
- Let $K_i = \{o \in TE \mid K(o) = i\}$, then

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

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"

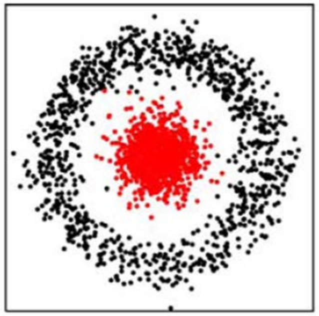
Overfitting (2)

- *Overfitting*
  - occurs when the classifier is too optimized to the (noisy) training data
  - 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

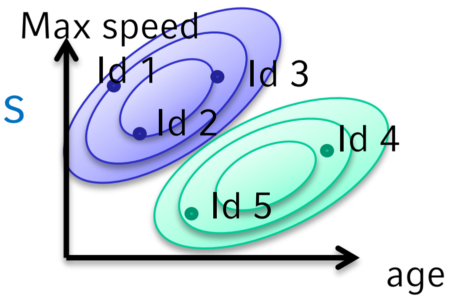
- *Underfitting*
  - Occurs when the classifiers model is too simple, e.g. trying to separate classes linearly that can only be separated by a quadratic surface
  - happens seldomly



- *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: $p(c_j|o) = \frac{p(o|c_j) \cdot p(c_j)}{p(o)}$

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

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

- Final decision rule for the *Bayes classifier*

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

- 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.

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

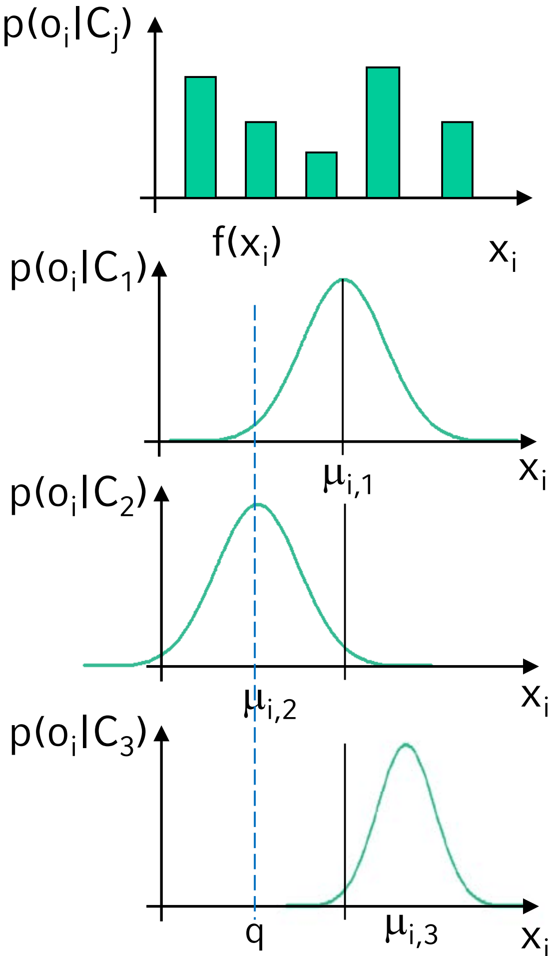
- Decision rule for the *naïve Bayes classifier*

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

Naïve Bayes Classifier (2)

- Independency assumption: $p(o_1, \dots, o_d | c_j) = \prod_{i=1}^{d} p(o_i | c_j)$

- If i-th attribute is categorical:
  $p(o_i|C)$ can be estimated as the relative frequency of samples having value x_i as i-th attribute in class C in the training set



- If i-th attribute is continuous:
  $p(o_i|C)$ 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

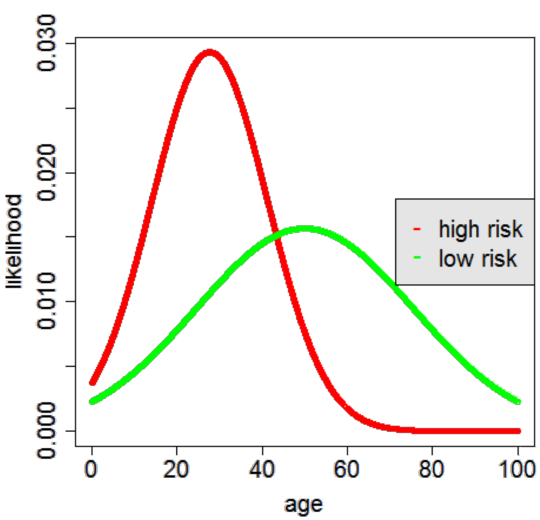
- Model setup:
  - Age $\sim N(\mu, \sigma)$ (normal distribution)
  - Car type ~ relative frequencies
  - Max speed $\sim N(\mu, \sigma)$ (normal distribution)

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

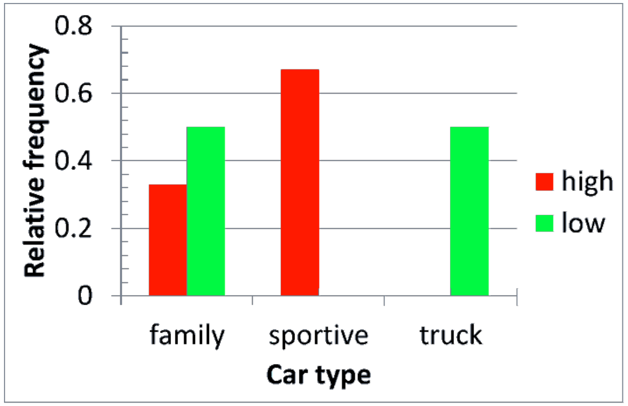
Age:

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

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



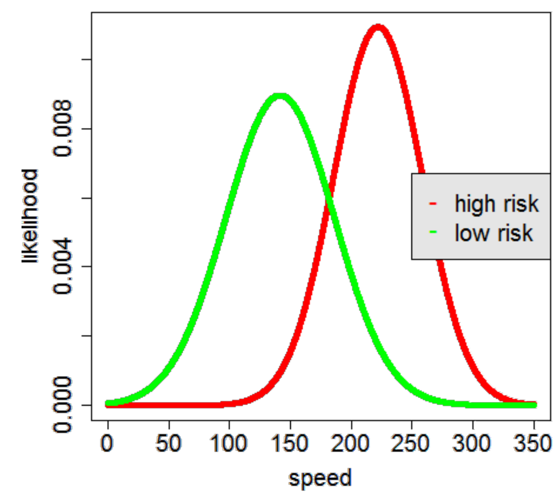
Car type:



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

$$p(high|q) = \frac{p(q|high) \cdot p(high)}{p(q)}$$
$$= \frac{p(age = 60|high) \cdot p(car\ type = family|high) \cdot p(\text{max}\ speed = 190|high) \cdot p(high)}{p(q)}$$
$$= \frac{N(27.67, 13.61|60) \cdot \frac{1}{3} \cdot N(222{,}36.49|190) \cdot \frac{3}{5}}{p(q)} = 15.32\%$$

$$p(low|q) = \frac{p(q|low) \cdot p(low)}{p(q)}$$
$$= \frac{p(age = 60|low) \cdot p(car\ type = family|low) \cdot p(\text{max}\ speed = 190|low) \cdot p(low)}{p(q)}$$
$$= \frac{N(50, 25.45|60) \cdot \frac{1}{2} \cdot N(141.5{,}44.54|190) \cdot \frac{2}{5}}{p(q)} = 84{,}68\%$$

← 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

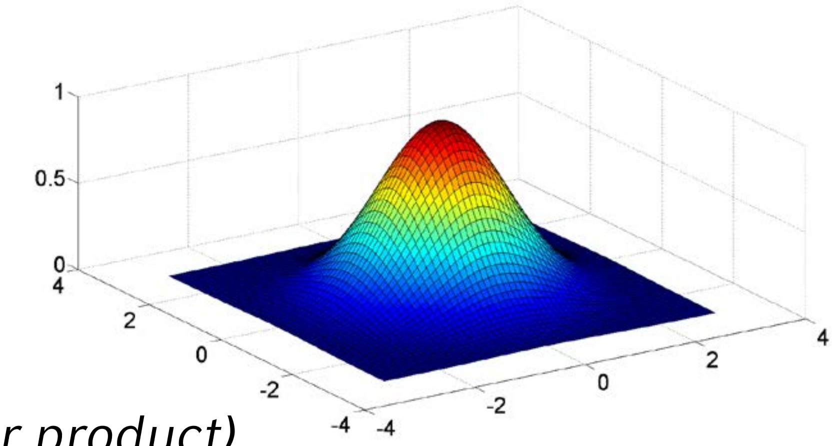
μ_j mean vector of class C_j

N_j is number of objects of class C_j

Σ_j is the $d \times d$ covariance matrix

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

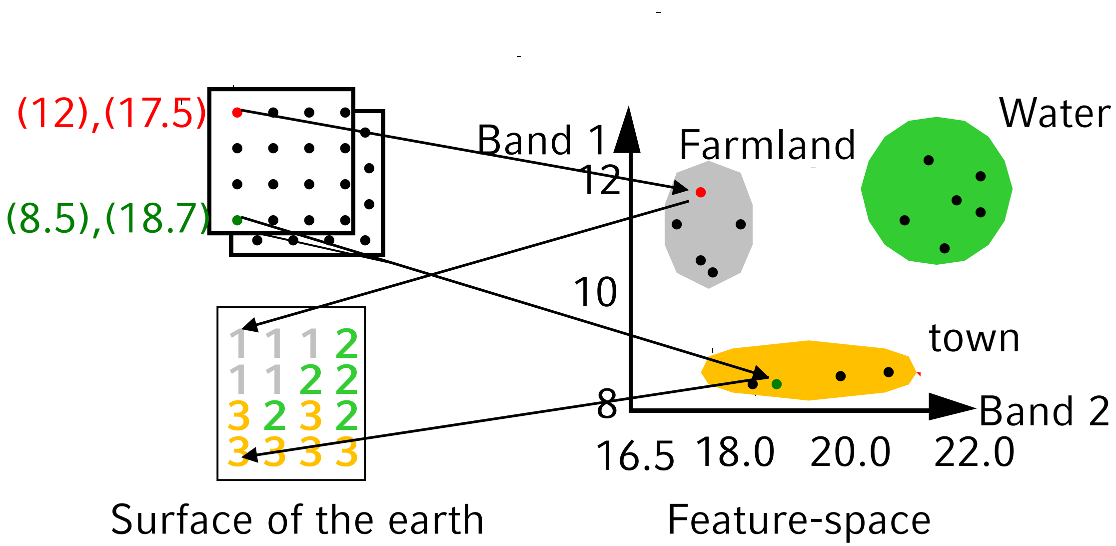
(outer product)

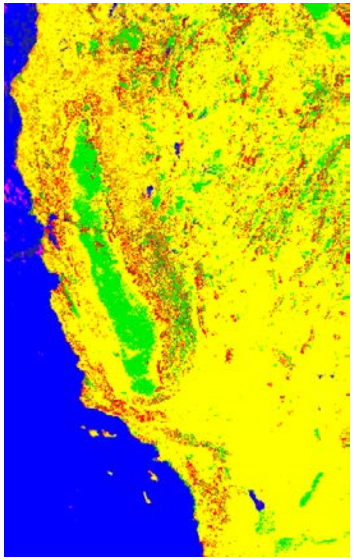


$|\Sigma_j|$ is the determinant of Σ_j and Σ_j^{-1} the inverse of Σ_j

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: $(o_1, \ldots, o_d)$
- Basic assumption: different surface properties of the earth („landuse“) follow a characteristic reflection and emission pattern

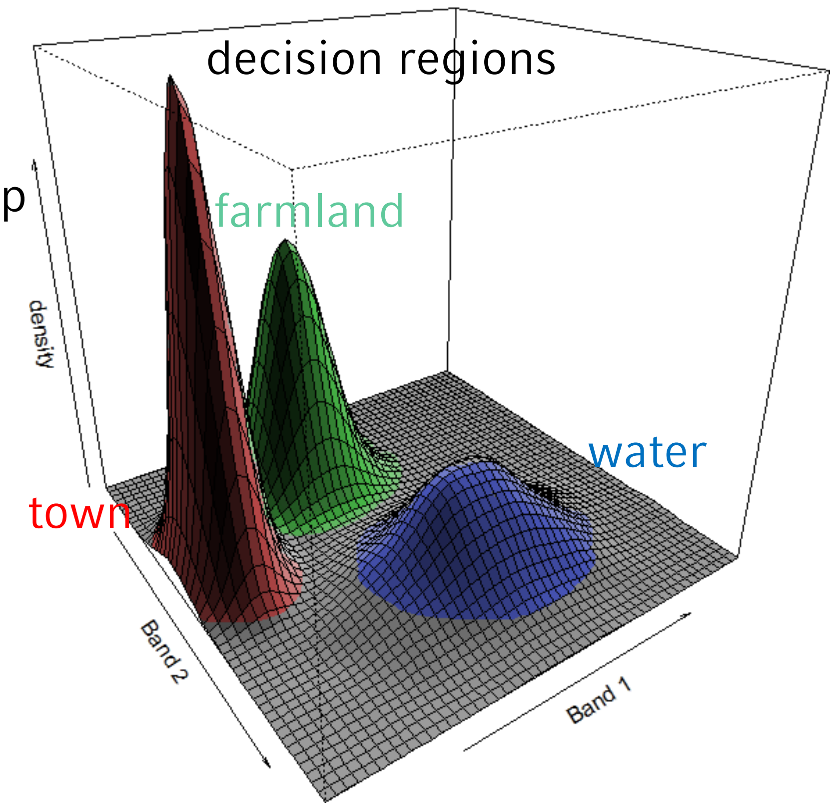




Example: Interpretation of Raster Images

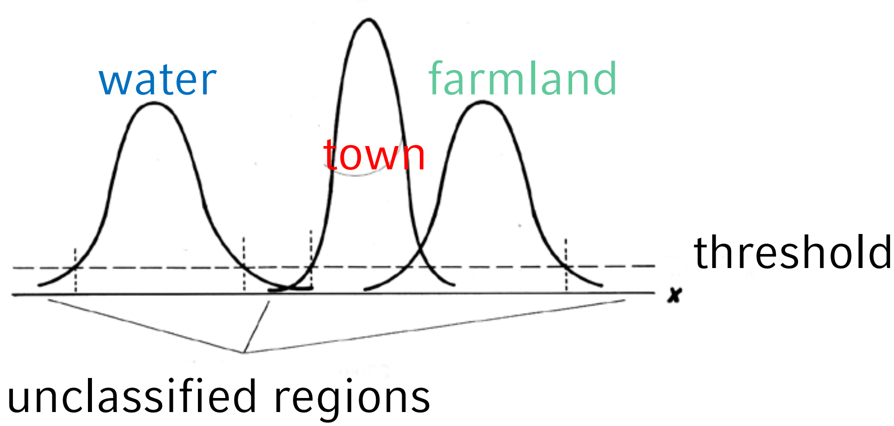
- Application of the Bayes classifier
  - Estimation of the p(*o* | *c*) without assumption of conditional independence
  - Assumption of d-dimensional normal (= Gaussian) distributions for the value vectors of a class

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



Bayesian Classifiers – Discussion

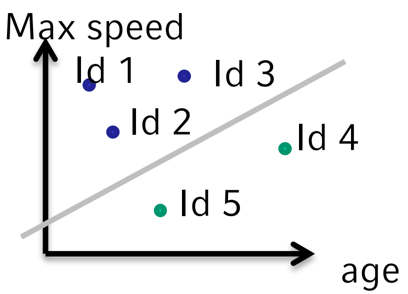
- Pro
  - High classification accuracy for many applications if density function defined properly
  - Incremental computation
    → many models can be adopted to new training objects by updating densities
    - For Gaussian: store *count*, *sum*, *squared sum* to derive *mean*, *variance*
    - For histogram: 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

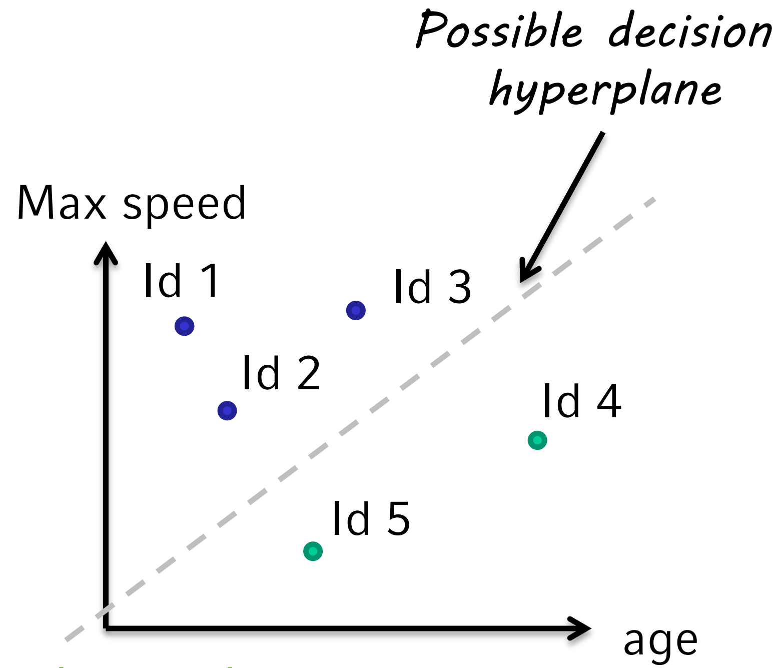
Chapter 6: Classification



Linear discriminant function classifier

- Example

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



- 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 are in the other half space
- Questions:
  - How to formalize the classifier?
  - How to find optimal parameters of the model?

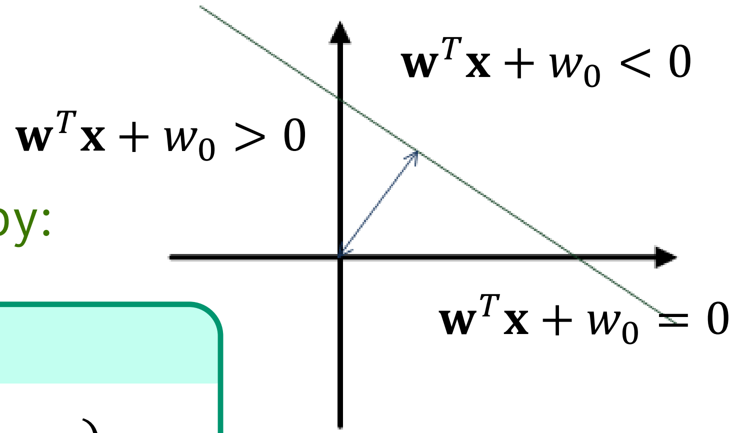
Basic notions

- Recall some general algebraic notions for a vector space V:
  - $\langle \mathbf{x}, \mathbf{y} \rangle$ denotes an inner product of two vectors $\mathbf{x}, \mathbf{y} \in V$:
    e.g., the scalar product: $\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^T \mathbf{y} = \sum_{i=1}^{d} (\mathrm{x}_i \cdot \mathrm{y}_i)$
  - $H(\mathbf{w}, w_0)$ denotes a hyperplane with normal vector $\mathbf{w}$ and constant term w_0:
    $\mathbf{x} \in H(\mathbf{w}, w_0) \Leftrightarrow \langle \mathbf{w}, \mathbf{x} \rangle + w_0 = 0$
  - The normal vector $\mathbf{w}$ may be normalized to $\boldsymbol{w}'$:
    $$\mathbf{w}' = \frac{1}{\sqrt{\langle \mathbf{w}, \mathbf{w} \rangle}} \cdot \mathbf{w} \Rightarrow \langle \mathbf{w}', \mathbf{w}' \rangle = 1$$
  - Distance of a vector x to the hyperplane $H(\mathbf{w}', w_0)$:
    $dist\big(\mathbf{x}, H(\mathbf{w}', w_0)\big) = |\langle \mathbf{w}', \mathbf{x} \rangle + w_0|$

Formalization

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

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



- Classification rule (linear classifier) given by:

Classification rule

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

Optimal parameter estimation

- How to estimate optimal parameters $\mathbf{w}, w_0$?
  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 → 0-1 loss

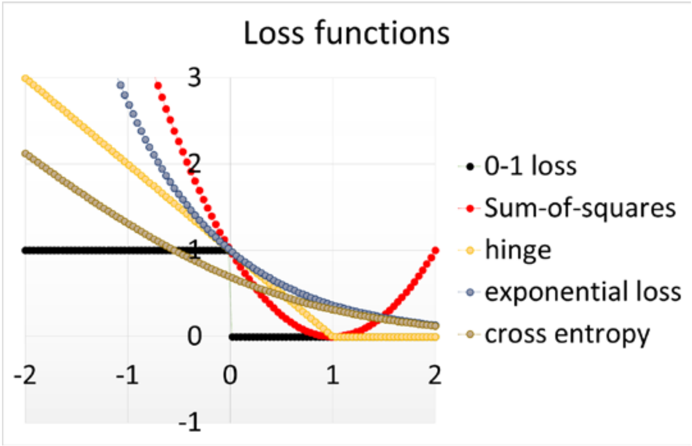
0-1 loss objective function for linear classifier

$$L(\mathbf{w}, w_0) = \min_{\mathbf{w}, w_0} \sum_{n=1}^{N} I(y_i \neq K_{H(\mathbf{w}, w_0)}(\mathbf{x}_i))$$

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

Loss functions

- 0-1 loss
  - Minimize the overall number of training errors, but…
    - NP-hard to optimize in general (non-smooth, non-convex)
    - Small changes of $\mathbf{w}, w_0$ can lead to large changes of the loss
- Alternative convex loss functions
  - Sum-of-squares loss: $(\mathbf{w}^T\mathbf{x}_i + w_0 - y_i)^2$
  - Hinge loss: $(1 - y_i(w_0 + \mathbf{w}^T\mathbf{x}_i)_+ = \max\{0,\ 1 - y_i(w_0 + \mathbf{w}^T\mathbf{x}_i\}$ (SVM)
  - Exponential loss: $e^{-y_i(w_0 + \mathbf{w}^T\mathbf{x}_i)}$ (AdaBoost)
  - Cross-entropy error: $-y_i \ln g(\mathbf{x}_i) + (1 - y_i)\ln(1 - g(\mathbf{x}_i))$ (Logistic regression)
    where $g(\mathbf{x}) = \frac{1}{1+e^{-(w_0+\mathbf{w}^T\mathbf{x})}}$
  - … and many more
- Optimizing different loss function leads to several classification algorithms
- Next, we derive the optimal parameters for the sum-of-squares loss



Optimal parameters for SSE loss

- Loss/Objective function: sum-of-squares error to real class values

Objective function

$$SSE(\mathbf{w}, w_0) = \sum_{i=1..N} \{(\mathbf{w}^T \mathbf{x}_i + w_0) - y_i\}^2$$

- Minimize the error function for getting optimal parameters
  - Use standard optimization technique:
    1. Calculate first derivative
    2. Set derivative to zero and compute the global minimum (SSE is a convex function)

Optimal parameters for SSE loss (cont'd)

- Transform the problem for simpler computations
  - $w^T o + w_0 = \sum_{i=1}^{d} w_i \cdot o_i + w_0 = \sum_{i=0}^{d} w_i \cdot o_i$, with $o_0 = 1$
  - For $\mathbf{w}$ let $\widetilde{\mathbf{w}} = (w_0, \dots, w_d)^T$
- Combine the values to matrices $\tilde{O} = \begin{pmatrix} 1 & o_{1,1} & \cdots & o_{1,d} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & o_{N,1} & \cdots & o_{N,d} \end{pmatrix}$, $Y = \begin{pmatrix} y_1 \\ \cdots \\ y_N \end{pmatrix}$
- Then the sum-of-squares error is equal to:

$$\sum_i a_{ii}^2 = \mathrm{tr}(A^T A)$$

$$SSE(\widetilde{\mathbf{w}}) = \frac{1}{2}\mathrm{tr}\left(\left(\tilde{O}\widetilde{\mathbf{w}} - Y\right)^T\left(\tilde{O}\widetilde{\mathbf{w}} - Y\right)\right)$$

Optimal parameters for SSE loss (cont'd)

- Take the derivative:

$$\frac{\partial}{\partial \tilde{\mathbf{w}}} SSE(\tilde{\mathbf{w}}) = \tilde{O}^T(\tilde{O}\tilde{\mathbf{w}} - Y)$$

- Solve $\frac{\partial}{\partial \widetilde{\mathbf{w}}} SSE(\widehat{\mathbf{w}}) = 0$:

Left-inverse of $\tilde{O}$ ("Moore-Penrose-Inverse")

$$\tilde{O}^T(\tilde{O}\widehat{\mathbf{w}} - Y) = 0 \Leftrightarrow \tilde{O}\widehat{\mathbf{w}} = Y \Leftrightarrow \widehat{\mathbf{w}} = \underbrace{\left(\tilde{O}^T\tilde{O}\right)^{-1}\tilde{O}^T}Y$$

- Set $\widehat{\mathbf{w}} = \left(\tilde{O}^T\tilde{O}\right)^{-1}\tilde{O}^T Y$

- Classify new point $\mathbf{x}$ with $\mathbf{x}_0 = 1$:

Classification rule

$$K_{H(\widehat{\mathbf{w}}, w_0)}(\mathbf{x}) = \mathrm{sign}(\widehat{\mathbf{w}}^T\mathbf{x})$$

Example SSE

- Data (consider only age and max. speed):

$$\tilde{O} = \begin{pmatrix} 1 & 23 & 180 \\ 1 & 17 & 240 \\ 1 & 43 & 246 \\ 1 & 68 & 173 \\ 1 & 32 & 110 \end{pmatrix}, Y = \begin{pmatrix} 1 \\ 1 \\ 1 \\ -1 \\ -1 \end{pmatrix}$$

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

encode classes as {high = 1, low = –1}

$$\Rightarrow \left(\tilde{O}^T\tilde{O}\right)^{-1}\tilde{O}^T = \begin{pmatrix} 0.7647 & -0.0678 & -0.9333 & -0.4408 & 1.6773 \\ -0.0089 & -0.0107 & 0.0059 & 0.0192 & -0.0055 \\ -0.0012 & 0.0034 & 0.0048 & -0.0003 & -0.0067 \end{pmatrix}$$

$$\Rightarrow \hat{\mathbf{w}} = \left(\tilde{O}^T\tilde{O}\right)^{-1}\tilde{O}^T Y = \begin{pmatrix} w_0 \\ w_{age} \\ w_{maxspeed} \end{pmatrix} = \begin{pmatrix} -1.4730 \\ -0.0274 \\ 0.0141 \end{pmatrix}$$

Example SSE (cont'd)

- Model parameter:

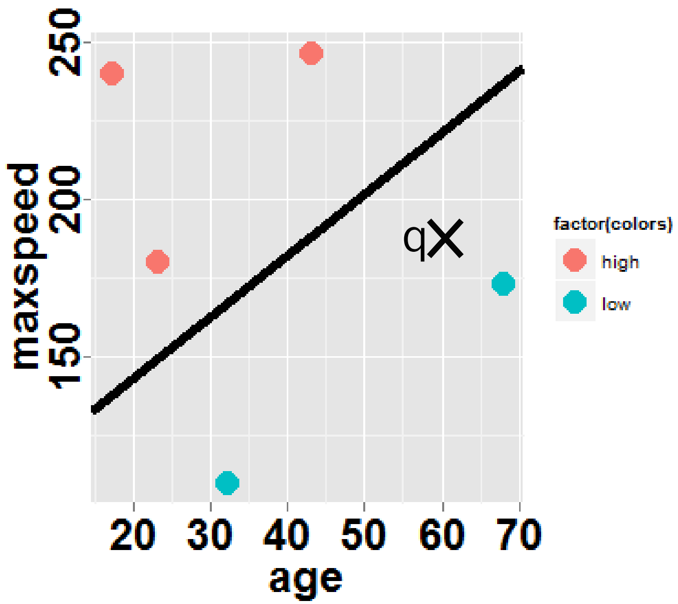
$$\hat{\mathbf{w}} = \left(\tilde{O}^T\tilde{O}\right)^{-1}\tilde{O}^T Y = \begin{pmatrix} w_0 \\ w_{age} \\ w_{maxspeed} \end{pmatrix} = \begin{pmatrix} -1.4730 \\ -0.0274 \\ 0.0141 \end{pmatrix}$$

$$\Rightarrow K_{H(\mathbf{w},w_0)}(\mathbf{x}) = \mathrm{sign}\left(\begin{pmatrix} -0.0274 \\ 0.0141 \end{pmatrix}^T \mathbf{x} - 1.4730\right)$$

Query: q = (age=60, max speed = 190)

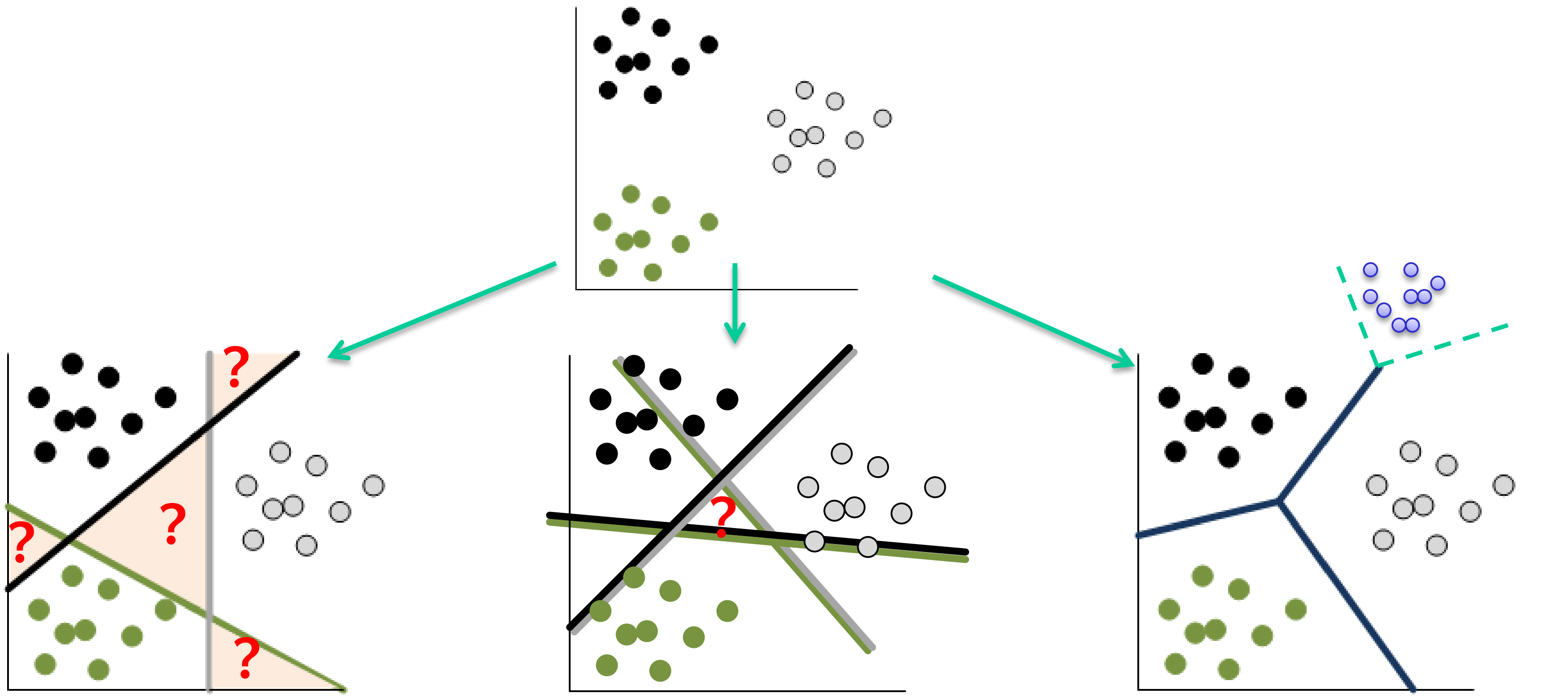
$\Rightarrow \mathrm{sign}(\hat{\mathbf{w}}^T q) = \mathrm{sign}(-0.4397) = -1$

$\Rightarrow$ Class = low



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 (cont'd)

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

$$\mathrm{y_j} = \arg\max_{j=1,\dots,k} H_{\mathbf{w}_j,w_{j,0}}(\mathbf{x})$$

- Advantage
  - No ambiguous regions except for points on decision hyperplanes

- The optimal parameter estimation is also extendable to k classes $Y = (y_1, \dots, y_k)$

Discussion (SSE)

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

- Contra
  - Sensitive to outliers → not stable classifier
    - How to define and efficiently determine the maximum stable hyperplane?
  - Only good results for linearly separable data
  - Expensive computation of selected hyperplanes

- Approach to solve the problems
  - Support Vector Machines (SVMs) [Vapnik 1979, 1995]