Linear Classification

Volker Tresp
Summer 2016

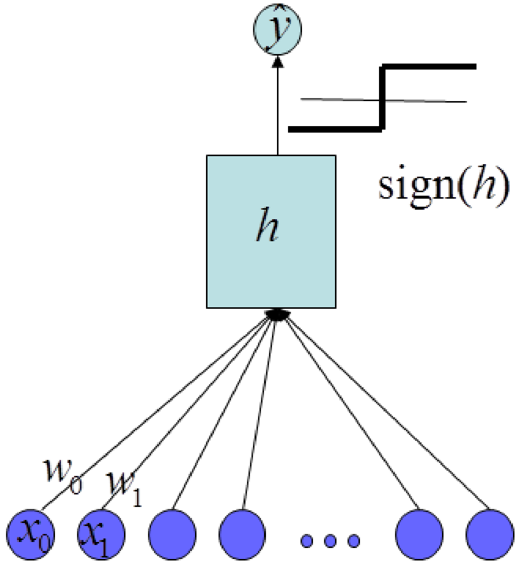
Classification

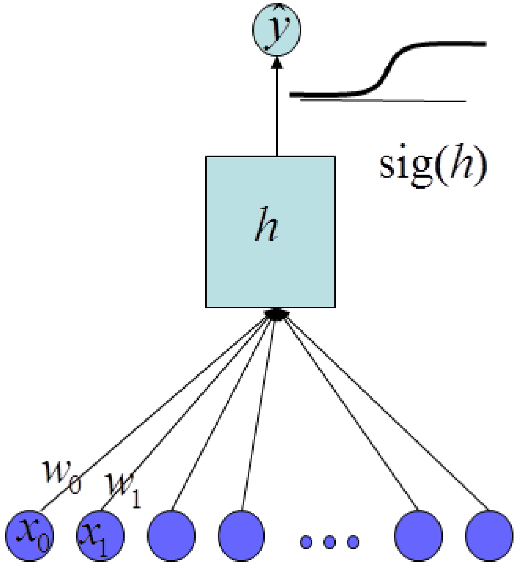
- Classification is the central task of pattern recognition
- Sensors supply information about an object: to which class do the object belong (dog, cat, ...)?

Linear Classifiers

- Linear classifiers separate classes by a linear hyperplane
- In high dimensions a linear classifier often can separate the classes
- Linear classifiers cannot solve the *exclusive-or* problem
- In combination with basis functions, kernels or a neural network, linear classifiers can form nonlinear class boundaries

Hard and Soft (sigmoid) Transfer Functions





- First, the activation function of the neurons in the hidden layer are calculated as the weighted sum of the inputs x_i as

$$h(\mathbf{x}) = \sum_{j=0}^{M-1} w_j x_j$$

(note: $x_0 = 1$ is a constant input, so that w_0 corresponds to the bias)

- The sigmoid neuron has a soft (sigmoid) transfer function

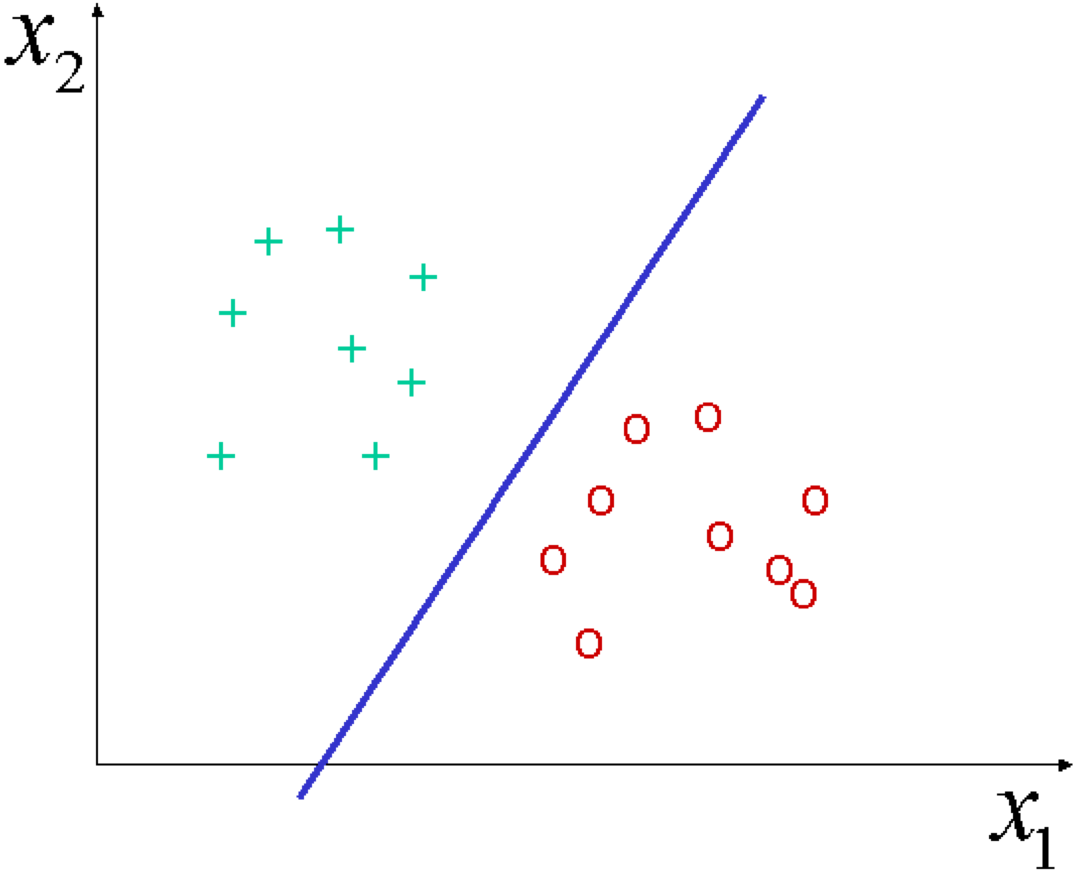
$$\textbf{Perceptron}: \quad \hat{y} = \text{sign}(h(\mathbf{x}))$$

$$\textbf{Sigmoidal neuron:} \quad \hat{y} = \text{sig}(h(\mathbf{x}))$$

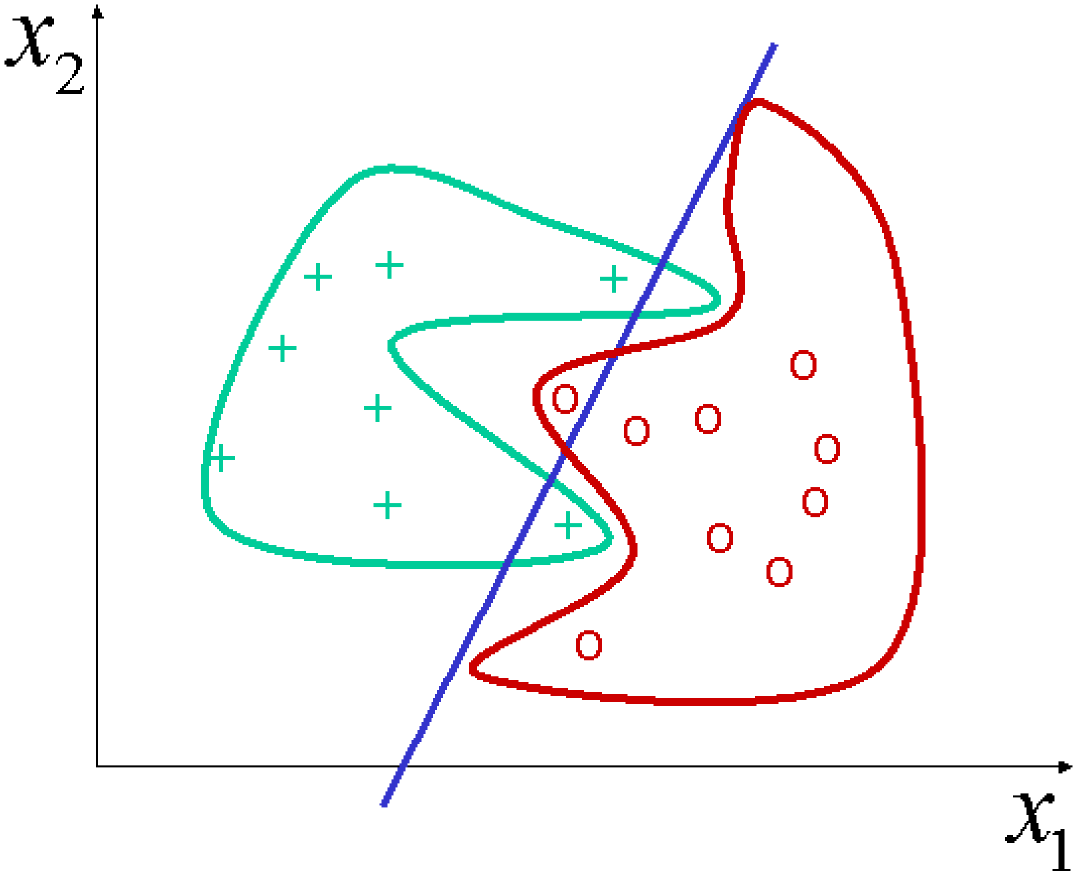
Binary Classification Problems

- We will focus first on binary classification where the task is to assign binary class labels $y_i = 1$ and $y_i = 0$ (or $y_i = 1$ and $y_i = -1$)
- We already know the *Perceptron*. Now we learn about additional approaches
  - I. Generative models for classification
  - II. Logistic regression
  - III. Classification via regression

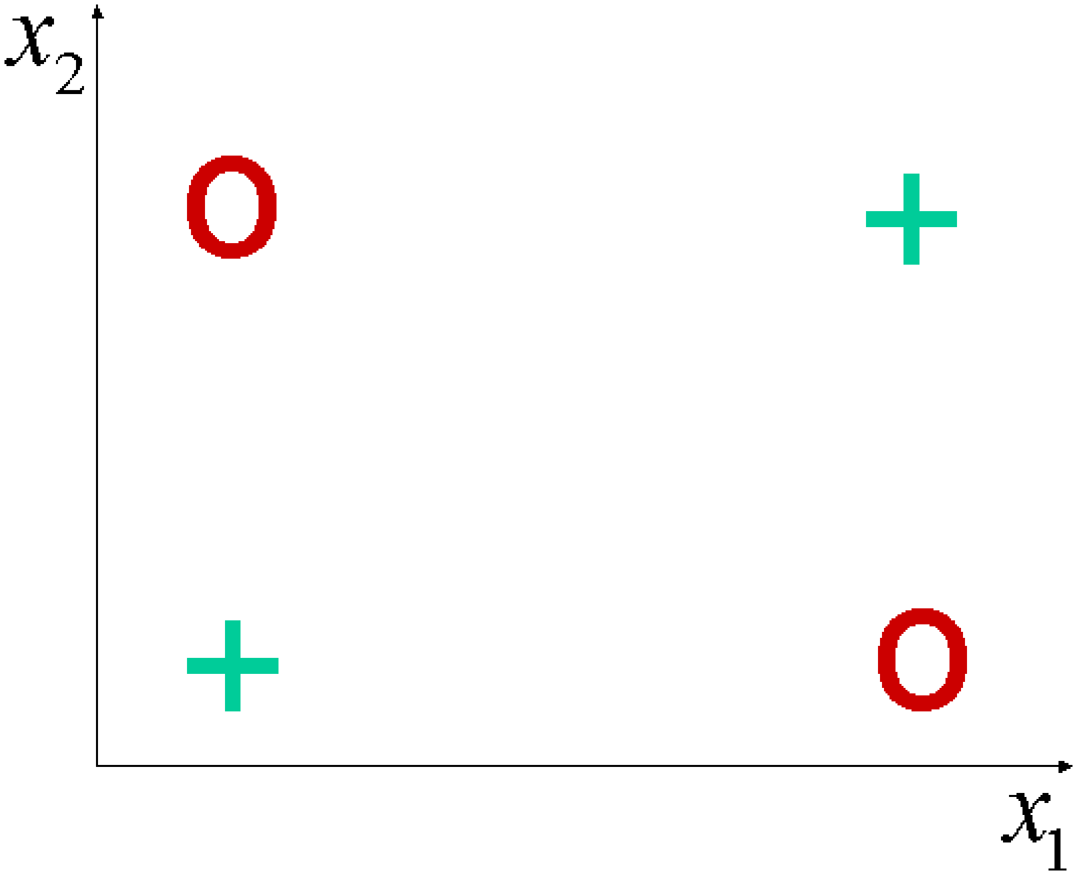
Two Linearly Separable Classes



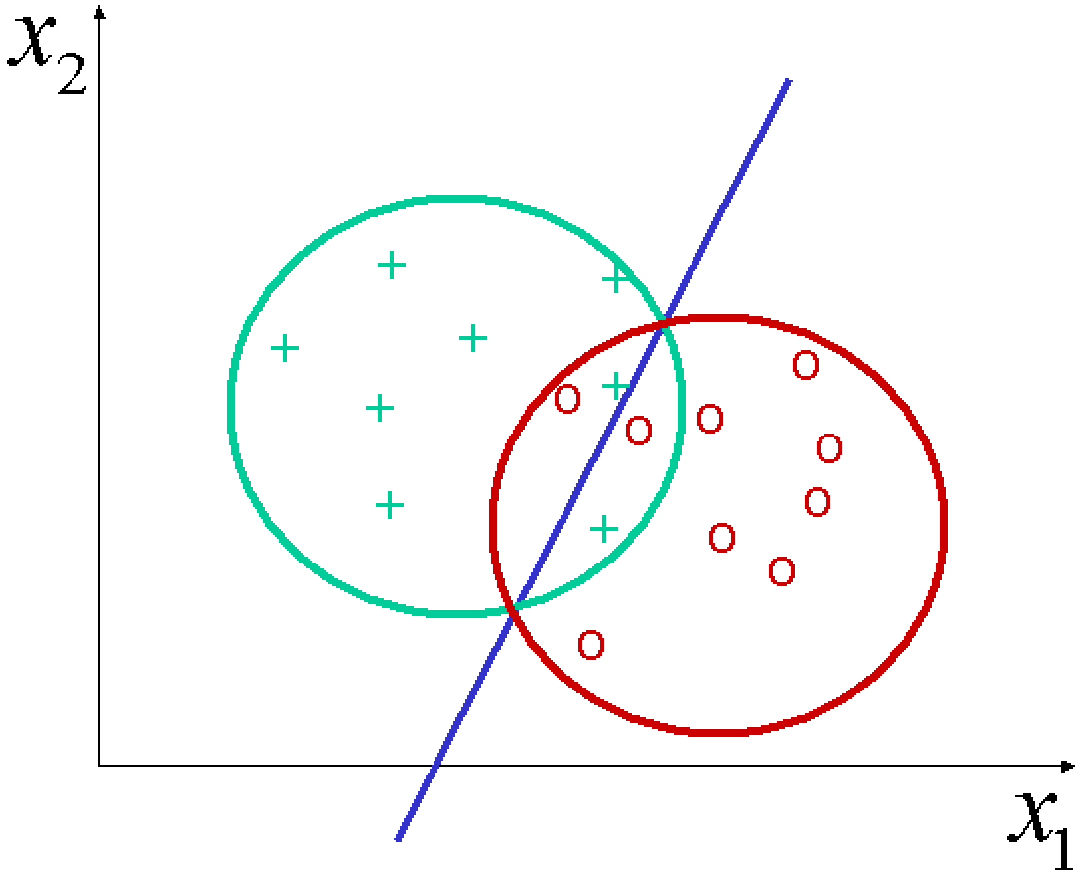
Two Classes that Cannot be Separated Linearly



The Classical Example not two Classes that cannot be Separated Linearly: XOR



Separability is not a Goal in Itself. With Overlapping Classes the Goal is the Best Possible Hyperplane



I. Generative Model for Classification

- In a generative model one assumes a probabilistic data generating process (likelihood model). Often generative models are complex and contain unobserved (latent, hidden) variables

- Here we consider a simple example: how data are generated in a binary classification problem

- First we have a model how classes are generated $P(y)$. $y = 1$ could stand for a good customer and $y = 0$ could stand for a bad customer.

- Then we have a model how attributes are generated, given the classes $P(\mathbf{x}|y)$. This could stand for

  - Income, age, occupation $(\mathbf{x})$ given a customer is a good customer $(y = 1)$
  - Income, age, occupation $(\mathbf{x})$ given a customer is not a good customer $(y = 0)$

- Using Bayes formula, we then derive $P(y|\mathbf{x})$: the probability that a given customer is a good customer $y = 1$ or bad customer $y = 0$, given that we know the customer's income, age and occupation

How is Data Generated?

- We assume that the observed classes y_i are generated with probability

  $$P(y_i = 1) = \kappa_1 \qquad P(y_i = 0) = \kappa_0 = 1 - \kappa_1$$

  with $0 \leq \kappa_1 \leq 1$.

- In a next step, a data point $\mathbf{x}_i$ has been generated from $P(\mathbf{x}_i | y_i)$

- (Note, that $\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,M})^T$, which means that $\mathbf{x}_i$ does not contain the bias $x_{i,0}$)

- We now have a complete model: $P(y_i)P(\mathbf{x}_i|y_i)$

Bayes' Theorem

- To classify a data point $\mathbf{x}_i$, i.e. to determine the y_i, we apply Bayes theorem and get

$$P(y_i|\mathbf{x}_i) = \frac{P(\mathbf{x}_i|y_i)P(y_i)}{P(\mathbf{x}_i)}$$

$$P(\mathbf{x}_i) = P(\mathbf{x}_i|y_i = 1)P(y_i = 1) + P(\mathbf{x}_i|y_i = 0)P(y_i = 0)$$

ML Estimate for Class Probabilities

- Maximum-likelihood estimator for the prior class probabilities are

  $$\widehat{P}(y_i = 1) = \widehat{\kappa}_1 = N_1/N$$

  and

  $$\widehat{P}(y_i = 0) = \widehat{\kappa}_0 = N_0/N = 1 - \widehat{\kappa}_1$$

  where N_1 and N_0 is the number of training data points for class 1, respectively class 0

- Thus the class probabilities simply reflect the class mix

Class-specific Distributions

- To model $P(\mathbf{x}_i|y_i)$ one can chose an application specific distribution
- A popular choice is a Gaussian distribution

$$P(\mathbf{x}_i|y_i = l) = \mathcal{N}(\mathbf{x}_i; \vec{\mu}^{(l)}, \Sigma)$$

with

$$\mathcal{N}\left(\mathbf{x}_i; \vec{\mu}^{(l)}, \Sigma\right) = \frac{1}{(2\pi)^{M/2}\sqrt{|\Sigma|}} \exp\left(-\frac{1}{2}\left(\mathbf{x}_i - \vec{\mu}^{(l)}\right)^T \Sigma^{-1}\left(\mathbf{x}_i - \vec{\mu}^{(l)}\right)\right)$$

- Note, that both Gaussian distributions have different modes (centers) but the same covariance matrices. This has been shown to often work well

Maximum-likelihood Estimators for Modes and Covariances

- One obtains a maximum likelihood estimators for the modes

$$\widehat{\vec{\mu}}^{(l)} = \frac{1}{N_l} \sum_{i:y_i=l} \mathbf{x}_i$$

- One obtains as unbiased estimators for the covariance matrix

$$\widehat{\Sigma} = \frac{1}{N-M} \sum_{l=0}^{1} \sum_{i:y_i=l} (\mathbf{x}_i - \widehat{\vec{\mu}}^{(l)})(\mathbf{x}_i - \widehat{\vec{\mu}}^{(l)})^T$$

Expanding the Quadratic Terms in the Exponent

- Note that

$$\left(\mathbf{x}_i - \vec{\mu}^{(l)}\right)^T \Sigma^{-1} \left(\mathbf{x}_i - \vec{\mu}^{(l)}\right)$$

$$= \mathbf{x}_i^T \Sigma^{-1} \mathbf{x}_i - 2\, \vec{\mu}^{(l)^T} \Sigma^{-1} \mathbf{x}_i + \vec{\mu}^{(l)^T} \Sigma^{-1} \vec{\mu}^{(l)}$$

- ... and ...

$$\left(\mathbf{x}_i - \vec{\mu}^{(0)}\right)^T \Sigma^{-1} \left(\mathbf{x}_i - \vec{\mu}^{(0)}\right) - \left(\mathbf{x}_i - \vec{\mu}^{(1)}\right)^T \Sigma^{-1} \left(\mathbf{x}_i - \vec{\mu}^{(1)}\right)$$

$$= 2\left(\vec{\mu}^{(0)} - \vec{\mu}^{(l)}\right)^T \Sigma^{-1} \mathbf{x}_i + \vec{\mu}^{(0)^T} \Sigma^{-1} \vec{\mu}^{(0)} - \vec{\mu}^{(1)^T} \Sigma^{-1} \vec{\mu}^{(1)}$$

A Posteriori Distribution

- It follows that

$$P(y_i = 1|\mathbf{x}_i) = \frac{P(\mathbf{x}_i|y_i = 1)P(y_i = 1)}{P(\mathbf{x}_i|y_i = 1)P(y_i = 1) + P(\mathbf{x}_i|y_i = 0)P(y_i = 0)}$$

$$= \frac{1}{1 + \frac{P(\mathbf{x}_i|y_i=0)P(y_i=0)}{P(\mathbf{x}_i|y_i=1)P(y_i=1)}}$$

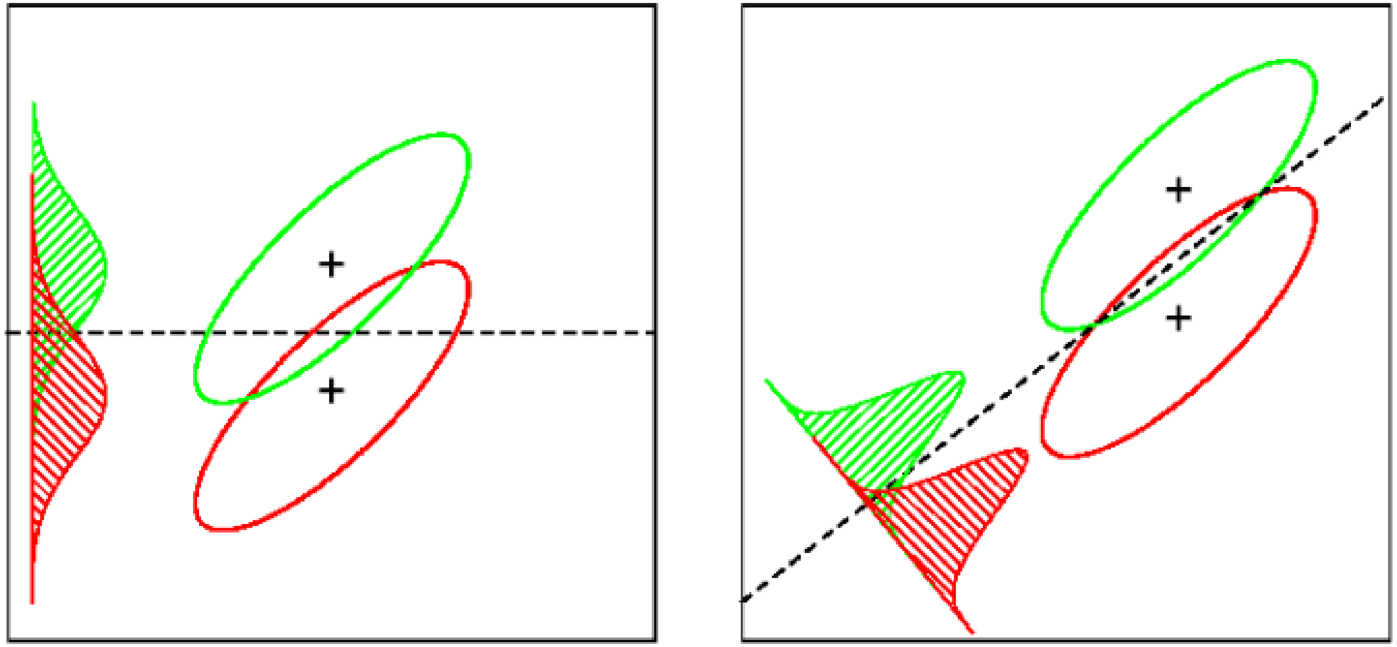
$$= \frac{1}{1 + \frac{\kappa_0}{\kappa_1} \exp\left((\vec{\mu}^{(0)} - \vec{\mu}^{(1)})^T \Sigma^{-1} \mathbf{x}_i + \frac{1}{2}\, \vec{\mu}^{(0)}{}^T \Sigma^{-1} \vec{\mu}^{(0)} - \frac{1}{2}\, \vec{\mu}^{(1)}{}^T \Sigma^{-1} \vec{\mu}^{(1)} \right)}$$

$$= \text{sig}\left(w_0 + \mathbf{x}_i^T \mathbf{w} \right) = \text{sig}\left(w_0 + \sum_j^M x_{i,j} w_j \right)$$

$$\mathbf{w} = \Sigma^{-1} \left(\vec{\mu}^{(1)} - \vec{\mu}^{(0)} \right)$$

$$w_0 = \log \kappa_1/\kappa_0 - \frac{1}{2}\vec{\mu}^{(0)^T}\Sigma^{-1}\vec{\mu}^{(0)} + \frac{1}{2}\vec{\mu}^{(1)^T}\Sigma^{-1}\vec{\mu}^{(1)}$$

- Recall: $\text{sig}(\textit{arg}) = 1/(1 + \exp(-\textit{arg}))$



Comments

- This specific generative model leads to linear class boundaries
- But we do not only get class boundaries, we get probabilities
- (Comment: The solution is analogue to Fisher's linear discriminant analysis (LDA), where one projects the data into a space in which data from the same class have small variance and where the distance between class modes are maximized. In other words, one gets the same results from an optimization criterion without assuming Gaussian distributions)
- Although we have used Bayes formula, the analysis was frequentist. A Bayesian analysis with a prior distribution on the parameters is also possible
- If the two class-specific Gaussians have different covariance matrices $(\Sigma^{(0)}, \Sigma^{(1)})$ the approach is still feasible but one would need to estimate two covariance matrices and the decision boundaries are not linear anymore

Special Case: Naive Bayes

- With diagonal covariances matrices, one obtains a *Naive-Bayes* classifier

$$P(\mathbf{x}_i|y_i = l) = \prod_{j=1}^{M} \mathcal{N}(\mathbf{x}_{i,j}; \vec{\mu}_j^{(l)}, \sigma_j^2)$$

- The naive Bayes classifier has considerable fewer parameters but completely ignores class-specific correlations between features; this is sometimes considered to be naive

- Even more naive:

$$P(\mathbf{x}_i|y_i = l) = \prod_{j=1}^{M} \mathcal{N}(\mathbf{x}_{i,j}; \vec{\mu}_j^{(l)}, 1)$$

  and then

$$\mathbf{w} = \vec{\mu}^{(1)} - \vec{\mu}^{(0)}$$

$$w_0 = \log \kappa_1/\kappa_0 - \frac{1}{2}\vec{\mu}^{(0)^T}\vec{\mu}^{(0)} + \frac{1}{2}\vec{\mu}^{(1)^T}\vec{\mu}^{(1)}$$

Special Case: Bernoulli Naive Bayes

- Naive Bayes classifiers are popular in text analysis with often more than 10000 features (key words). For example, the classes might be SPAM and no-SPAM and the features are keywords in the texts

- Instead of a Gaussian distribution, a Bernoulli distribution is employed

- $P(\mathit{word}_j = 1|\text{SPAM}) = \gamma_{j,s}$ is the probability of observing word $word_j$ in the document for SPAM documents

- $P(\mathit{word}_j = 0|\text{SPAM}) = 1 - \gamma_{j,s}$ is the probability of not observing word $word_j$ in the document for SPAM documents

- $P(\mathit{word}_j = 1|\text{no-SPAM}) = \gamma_{j,n}$ is the probability of observing word $word_j$ in the document for non-SPAM documents

- $P(\mathit{word}_j = 0|\text{no-SPAM}) = 1-\gamma_{j,n}$ is the probability of not observing word $word_j$ in the document for non-SPAM documents

- Then

$$P(\text{SPAM}|doc) \propto \kappa_s \prod_j \gamma_{j,s}^{\textsf{\textit{word}}_j} \, (1-\gamma_{j,s})^{1-\textsf{\textit{word}}_j}$$

$$P(\text{no-SPAM}|doc) \propto \kappa_n \prod_j \gamma_{j,n}^{\textsf{\textit{word}}_j} \, (1-\gamma_{j,n})^{1-\textsf{\textit{word}}_j}$$

- Simple ML estimates are $\gamma_{j,s} = N_{j,s}/N_s$ and $\gamma_{j,n} = N_{j,n}/N_n$

  (N_s is the number of SPAM documents in the training set, $N_{j,s}$ is the number of SPAM documents in the training set where *word*$_j$ is present)

  (N_n is the number of no-SPAM documents in the training set, $N_{j.n}$ is the number of no-SPAM documents in the training set where *word*$_j$ is present)

II. Logistic Regression

- The generative model motivates

$$P(y_i = 1|\mathbf{x}_i) = \mathsf{sig}\left(\mathbf{x}_i^T \mathbf{w}\right)$$

(now we include the bias $\mathbf{x}_i^T = (x_{i,0} = i, 1, x_{i,1}, \ldots, x_{i,M-1})^T$). sig() as defined before (logistic funktion).

- One now optimizes the likelihood of the conditional model

$$L(\mathbf{w}) = \prod_{i=1}^{N} \mathsf{sig}\left(\mathbf{x}_i^T \mathbf{w}\right)^{y_i} \left(1 - \mathsf{sig}\left(\mathbf{x}_i^T \mathbf{w}\right)\right)^{1-y_i}$$

Log-Likelihood

- Log-likelihood function

$$l = \sum_{i=1}^{N} y_i \log \left(\text{sig}\left(\mathbf{x}_i^T \mathbf{w}\right)\right) + (1 - y_i) \log \left(1 - \text{sig}\left(\mathbf{x}_i^T \mathbf{w}\right)\right)$$

$$l = \sum_{i=1}^{N} y_i \log \left(\frac{1}{1 + \exp(-\mathbf{x}_i^T \mathbf{w})}\right) + (1 - y_i) \log \left(\frac{1}{1 + \exp(\mathbf{x}_i^T \mathbf{w})}\right)$$

$$= - \sum_{i=1}^{N} y_i \log(1 + \exp(-\mathbf{x}_i^T \mathbf{w})) + (1 - y_i) \log(1 + \exp(\mathbf{x}_i^T \mathbf{w}))$$

Adaption

- The derivatives of the log-likelihood with respect to the parameters

$$\frac{\partial l}{\partial \mathbf{w}} = \sum_{i=1}^{N} y_i \frac{\mathbf{x}_i \, \mathsf{exp}(-\mathbf{x}_i^T \mathbf{w})}{1 + \mathsf{exp}(-\mathbf{x}_i^T \mathbf{w})} - (1 - y_i) \frac{\mathbf{x}_i \, \mathsf{exp}(\mathbf{x}_i^T \mathbf{w})}{1 + \mathsf{exp}(\mathbf{x}_i^T \mathbf{w})}$$

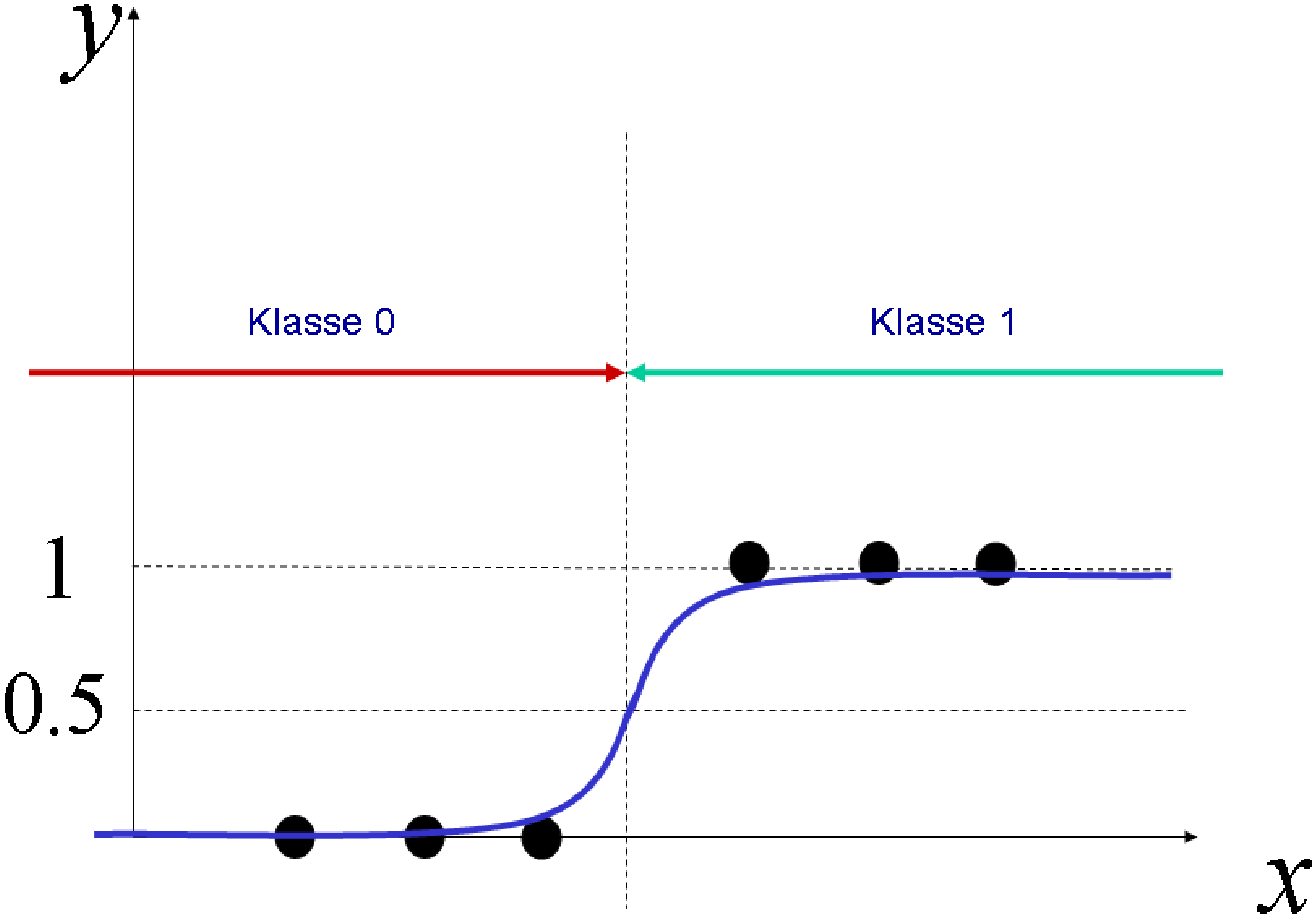
$$= \sum_{i=1}^{N} y_i \mathbf{x}_i (1 - \mathsf{sig}(\mathbf{x}_i^T \mathbf{w})) - (1 - y_i) \mathbf{x}_i \mathsf{sig}(\mathbf{x}_i^T \mathbf{w})$$

$$= \sum_{i=1}^{N} (y_i - \mathsf{sig}(\mathbf{x}_i^T \mathbf{w})) \mathbf{x}_i$$

- A gradient-based optimization of the parameters to maximize the log-likelihood

$$\mathbf{w} \longleftarrow \mathbf{w} + \eta \frac{\partial l}{\partial \mathbf{w}}$$

- Typically one uses a Newton-Raphson optimization procedure


y
Klasse 0
Klasse 1
1
0.5
x

Cross-entropy Cost Function

- In information theory, the cross entropy between a true distribution P and an approximative distribution Q is defined as $H(P,Q) = -\sum_x P(X=x) \log Q(X=x)$. (in information theory, it represents the number of bits if the encoding uses $Q(x)$ whereas the true distribution is $P(x)$. It is greater of equal the entropy of $P(x)$)

- We apply the cross entropy to the conditional distributions $P(Y|X), Q(Y|X)$ and we assume that $P(Y|X)$ is represented by the data and $Q(Y|X)$ represents the model we get $H(P,Q) = -\sum_i \log Q(x_i)$. Thus the cross entropy is identical to the negative log-likelihood

- The **log-likelihood cost function** is identical to the **cross entropy cost function** and is written for $y_i \in \{0,1\}$

$$\mathrm{cost} = -\sum_{i=1}^{N} y_i \log(\mathbf{sig}(\mathbf{x}_i^T \mathbf{w})) - (1-y_i)\log(1-\mathbf{sig}(\mathbf{x}_i^T \mathbf{w})))$$

$$= \sum_{i=1}^{N} y_i \log(1+\exp(-\mathbf{x}_i^T \mathbf{w})) + (1-y_i)\log(1+\exp(\mathbf{x}_i^T \mathbf{w}))$$

$$= \sum_{i=1}^{N} \log\left(1 + \exp\left((1 - 2y_i)\mathbf{x}_i^T \mathbf{w}\right)\right)$$

- ... and for $y_i \in \{-1, 1\}$

$$\text{cost} = \sum_{i=1}^{N} \log\left(1 + \exp\left(-y_i \mathbf{x}_i^T \mathbf{w}\right)\right)$$

Logistic Regression in Medical Statistics

- Logistic regression has become one of the the most important tools in medicine to analyse the outcome of treatments

- $y = 1$ means that the patient has the disease. $x_1 = 1$ might represent the fact that the patient was exposed (e.g., by a genetic variant) and $x_1 = 0$ might mean that the patient was not exposed. The other inputs are often typical confounders (age, sex, ...)

- Logistic regression then permits the prediction of the outcome for any patient

- Of course, of great interest is if w_1 is significantly positive (i.e., the exposure was harmful)

Log-Odds

- The logarithm of the odds is defined as

$$\log(Odds(\mathbf{x}_i)) = \log \frac{P(y_i = 1|\mathbf{x}_i)}{P(y_i = 0|\mathbf{x}_i)}$$

- For logistic regression,

$$\log(Odds(\mathbf{x}_i)) = \log \frac{P(y_i = 1|\mathbf{x}_i)}{P(y_i = 0|\mathbf{x}_i)} = \log \frac{1}{1 + \exp(-\mathbf{x}_i^T \mathbf{w})} \frac{1 + \exp(-\mathbf{x}_i^T \mathbf{w})}{\exp(-\mathbf{x}_i^T \mathbf{w})}$$

$$= \log \frac{1}{\exp(-\mathbf{x}_i^T \mathbf{w})} = \mathbf{x}_i^T \mathbf{w}$$

Log Odds Ratio

- The log odds ratio evaluates the effect of the treatment (valid for any patient)

$$w_1 = \log(Odds(x_1 = 1)) - \log(Odds(x_1 = 0)) = \log \frac{Odds(x_1 = 1)}{Odds(x_1 = 0)}$$

- This is the logarithm of the so called odds ratio (OR). If $w_1 \approx 0$, then the exposure does not have an effect: the odds ratio is commonly used in case-control studies!

- What we have calculated is the log-odds for a patient with properties $\mathbf{x}_i$ to get the disease

- We get a nice interpretation of the parameters in logistic regression: w_0 is the log odds of getting the disease, when the patient does not get the treatment (and all other inputs are zero). This is the only term that changes, when the class mix is changed

- w_1 is the log odds ratio associated with the treatment. This is independent of the class mix or of other input factors (thus we cannot model interactions)

- $w_2, \ldots, w_{M-1}$ models the log odds ratio associated with confounders (age, sex, ...)

Logistic Regression as a Generalized Linear Models (GLM)

- Consider a Bernoulli distribution with $P(y = 1) = \theta$ and $P(y = 0) = 1 - \theta$, with $0 \leq \theta \leq 1$
- In the theory of the exponential family of distributions, one sets $\theta = \mathrm{sig}(\eta)$. Now we get valid probabilities for any $\eta \in \mathbb{R}$!
- η is called the natural parameter and $\mathrm{sig}(\cdot)$ the inverse parameter mapping for the Bernoulli distribution
- This is convenient if we make η a linear function of the inputs and one obtains a Generalized Linear Model (GLM)

$$P(y_i = 1|\mathbf{x}_i, \mathbf{w}) = \mathrm{sig}(\mathbf{x}_i^T \mathbf{w})$$

- *Thus logistic regression is the GLM for the Bernoulli likelihood model*

Application to Neural Networks and other Systems

- Logistic regression essentially defines a new cost function
- It can be applied as well to neural networks

$$P(y_i = 1|\mathbf{x}_i, \mathbf{w}) = \mathsf{sig}(\mathrm{NN}(\mathbf{x}_i))$$

or systems of basis functions or kernel systems

III. Classification via Regression

- Linear Regression:

$$f(\mathbf{x}_i, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j x_{i,j}$$

$$= \mathbf{x}_i^T \mathbf{w}$$

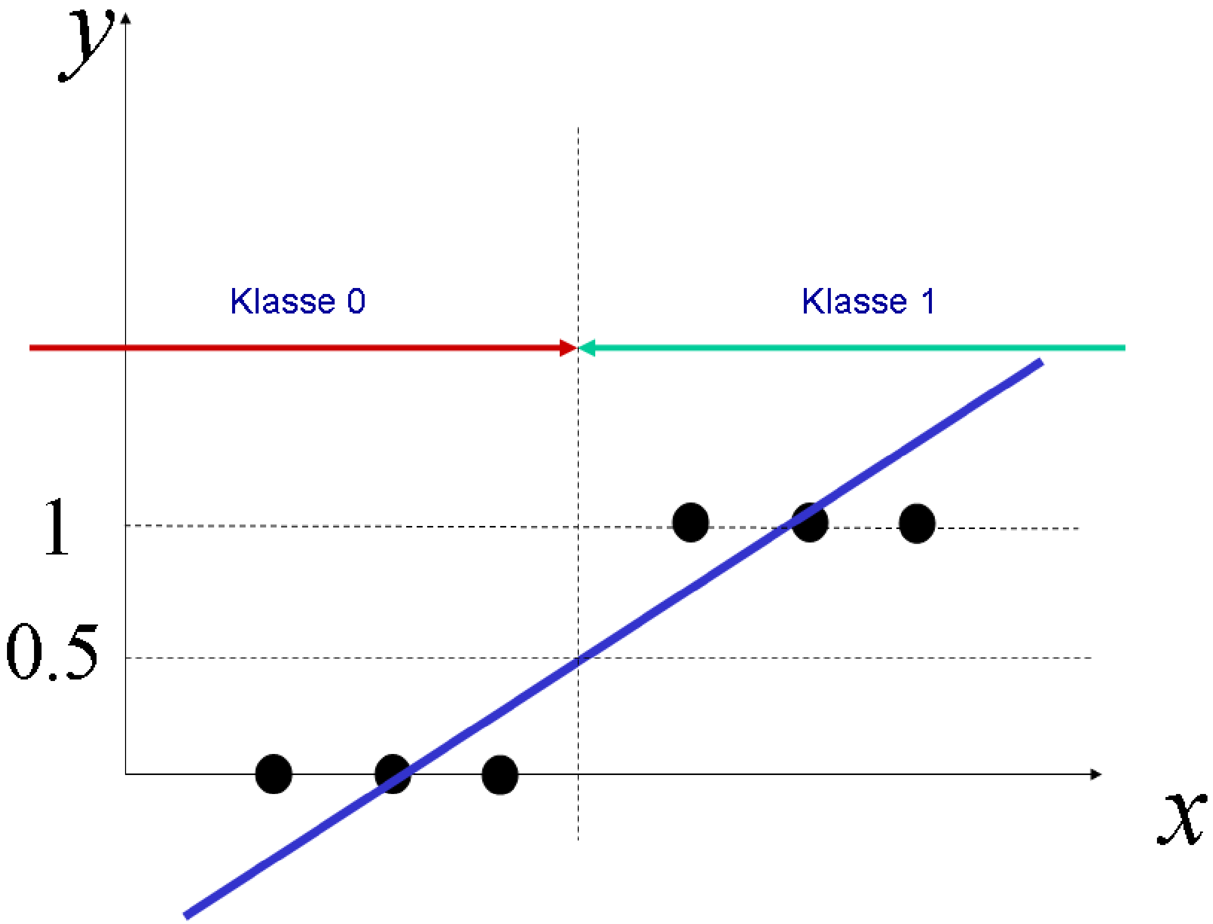
- We define as target $y_i = 1$ if the pattern $\mathbf{x}_i$ belongs to class 1 and $y_i = 0$ (or $y_i = -1$) if pattern $\mathbf{x}_i$ belongs to class 0
- We calculate weights $\mathbf{w}_{LS} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$ as LS solution, exactly as in linear regression
- For a new pattern $\mathbf{x}$ we calculate $f(\mathbf{x}) = \mathbf{x}^T\mathbf{w}_{LS}$ and assign the pattern to class 1 if $f(\mathbf{x}) > 1/2$ (or $f(\mathbf{x}) > 0$) ; otherwise we assign the pattern to class 0

Bias

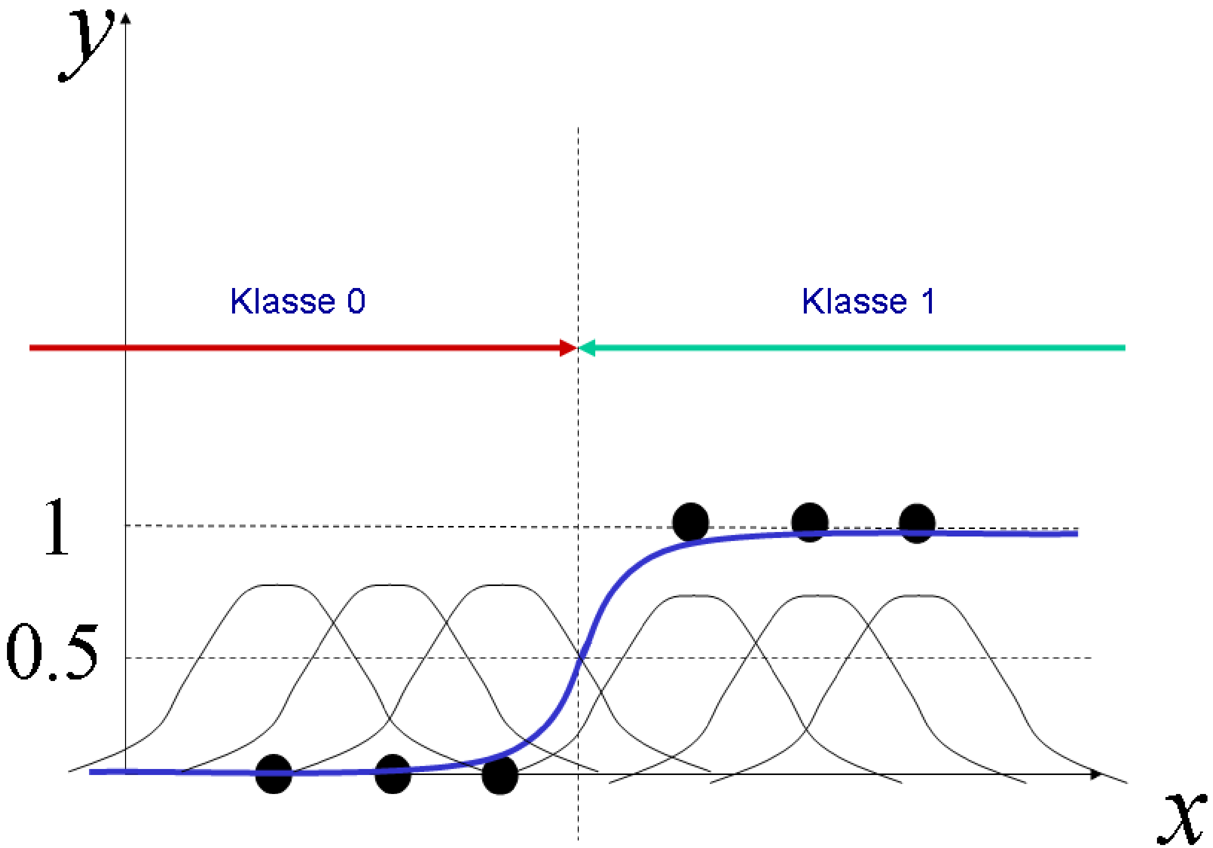
- Asymptotically, a LS-solution converges to the posterior class probabilities, although a linear functions is typically not able to represent $P(c = 1|\mathbf{x})$. The resulting class boundary can still be sensible

- One can expect good class boundaries in high dimensions and/or in combination with basis functions, kernels and neural networks; in high dimensions sometimes consistency can be achieved

Classification via Regression with Linear Functions



Classification via Regression with Radial Basis Functions



Performance

- Although the approach might seem simplistic, the performance can be excellent (in particular in high dimensions and/or in combination with basis functions, kernels and neural networks). The calculation of the optimal parameters can be very fast!