

Linear Classification

Volker Tresp
Summer 2019

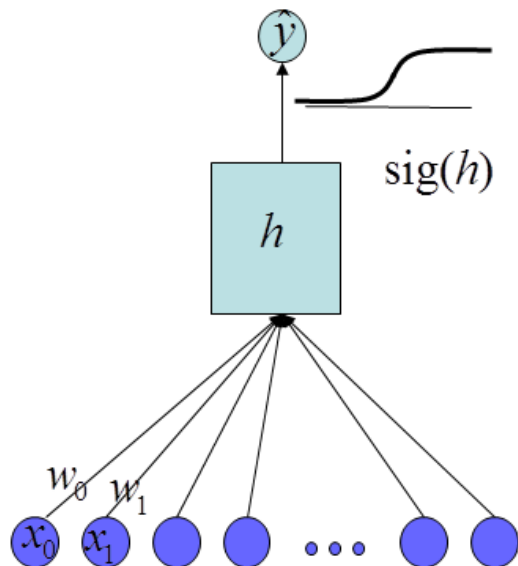
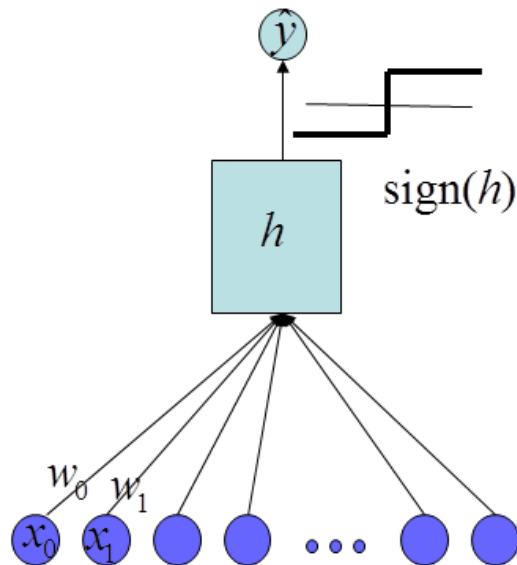
Classification

- 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

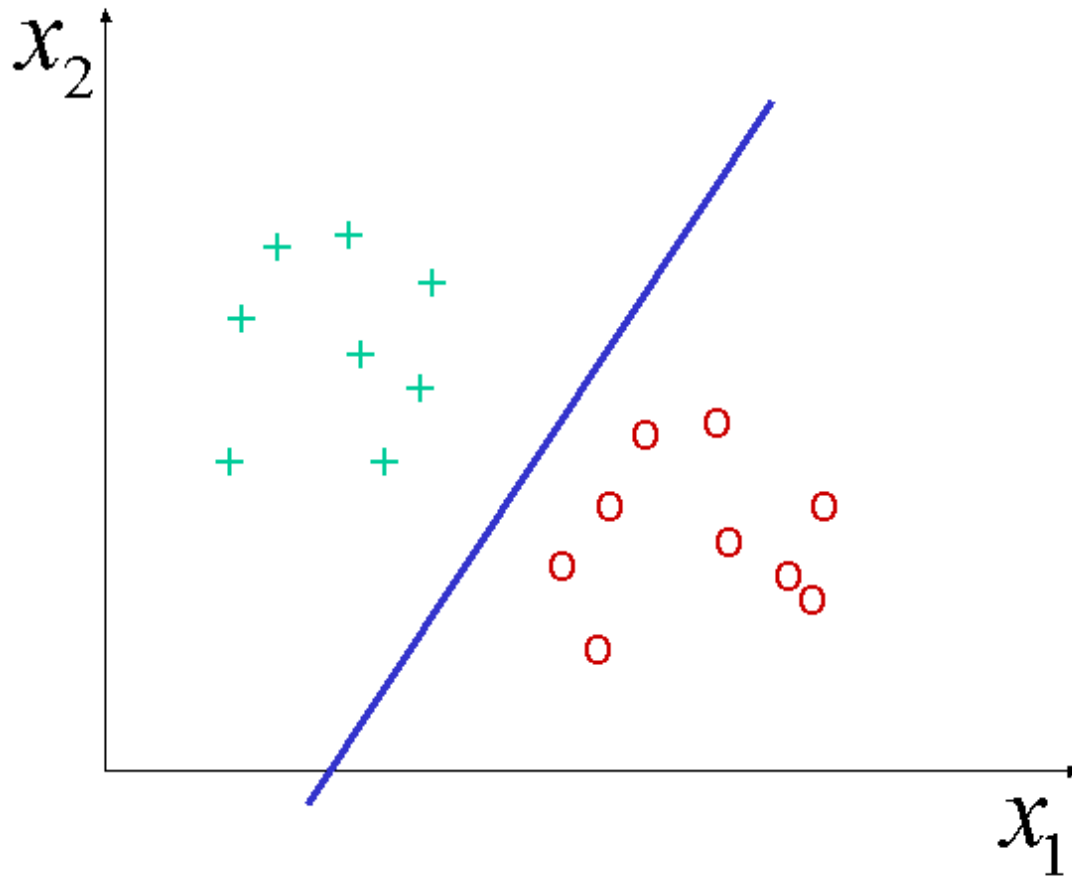
$$\text{Perceptron : } \hat{y} = \text{sign}(h(\mathbf{x}))$$

$$\text{Sigmoid function: } \hat{y} = \text{sig}(h(\mathbf{x}))$$

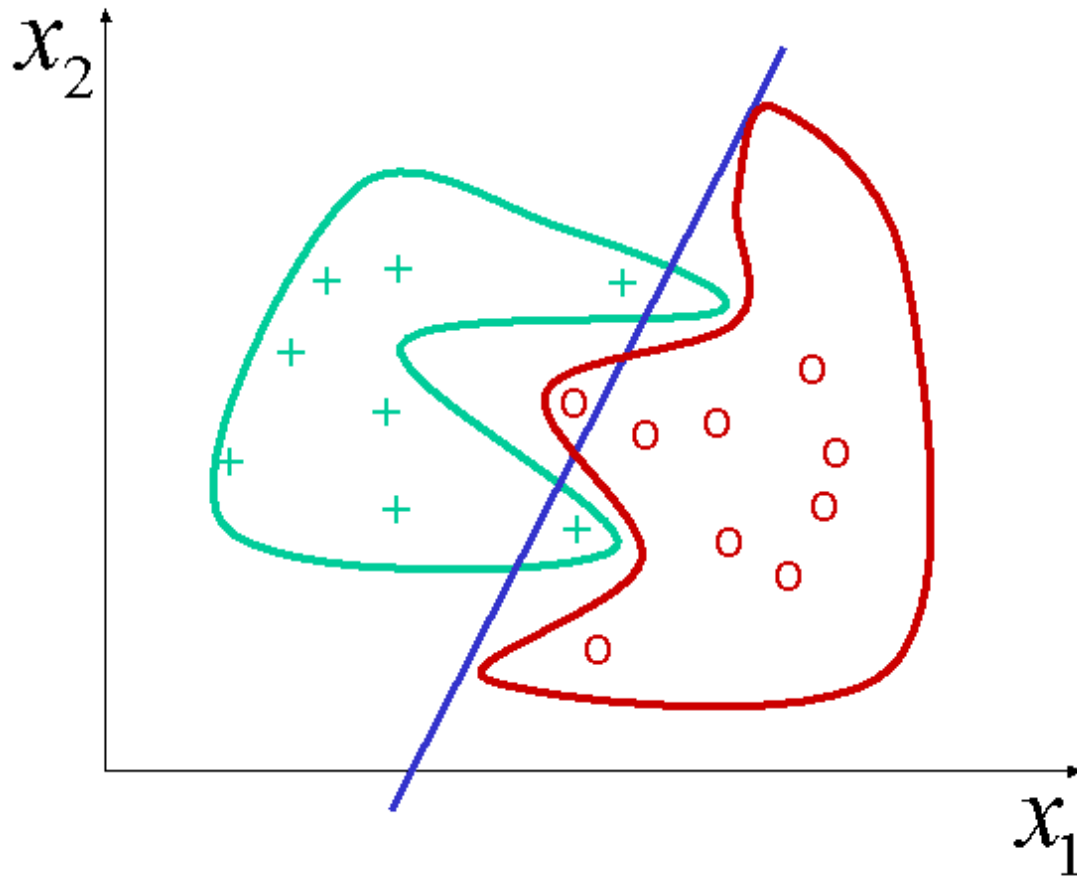
Binary Classification Problems

- 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

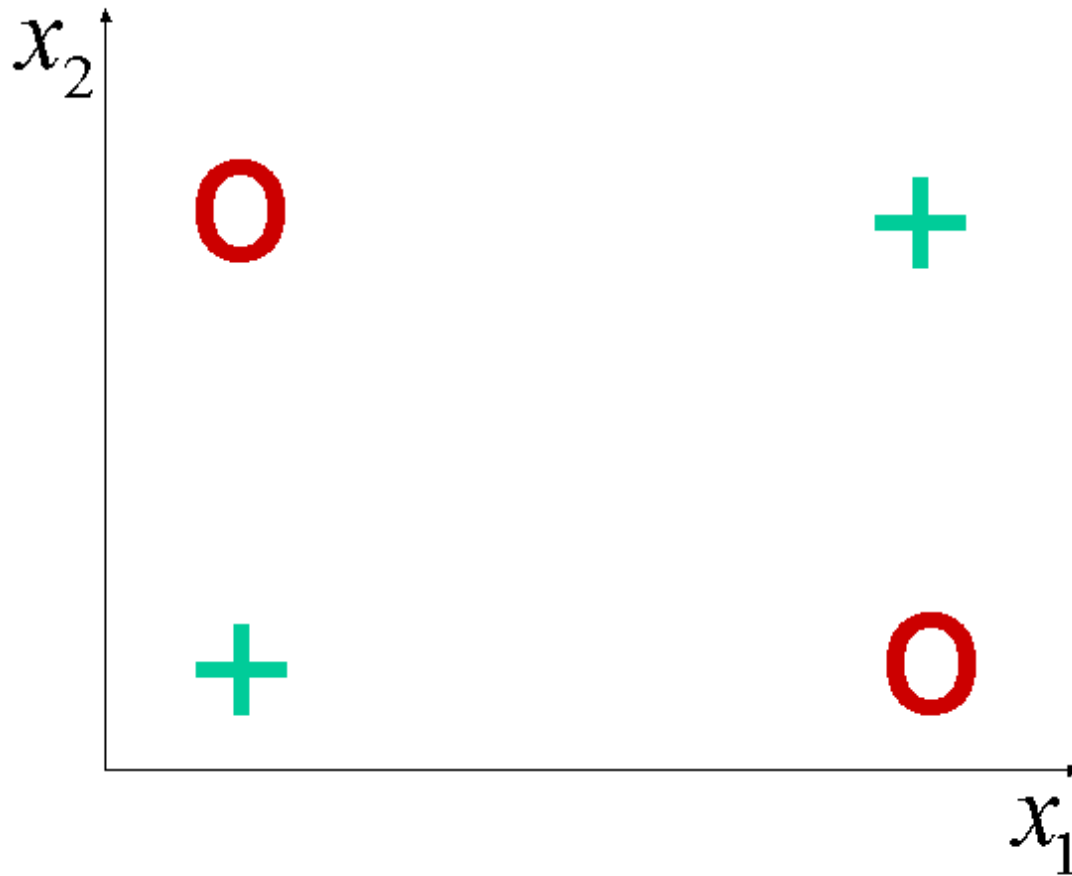
Two Linearly Separable Classes



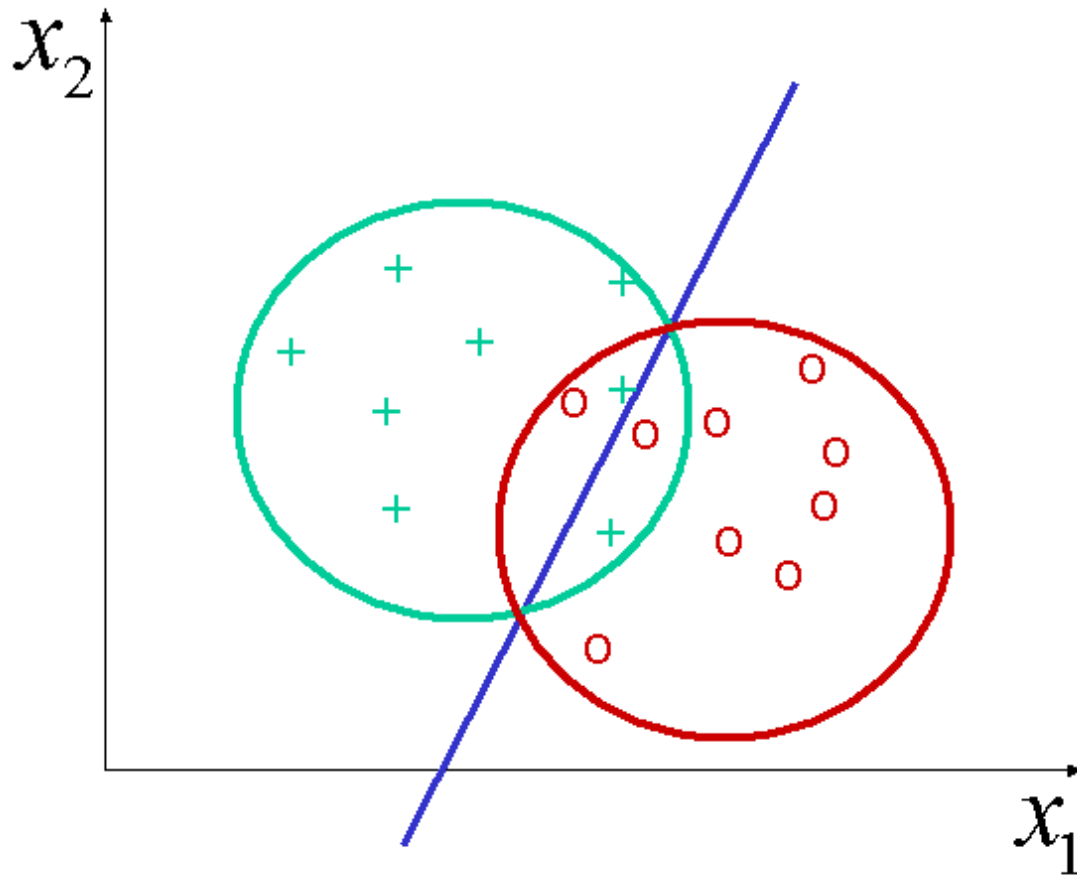
Two Classes that Cannot be Separated Linearly



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

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

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 $\mathbf{x}_i = (x_{i,1}, \dots, x_{i,M})^T$, 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

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

and

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

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

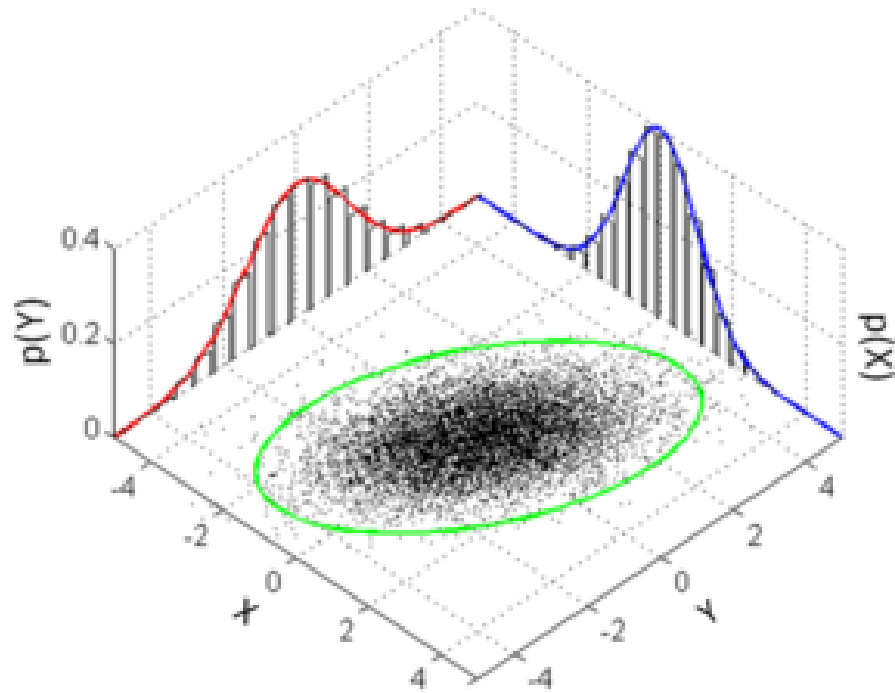
- To model $P(\mathbf{x}_i|y_i)$ one can choose an application specific distribution
- A popular choice is a Gaussian distribution (normal discriminant analysis)

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

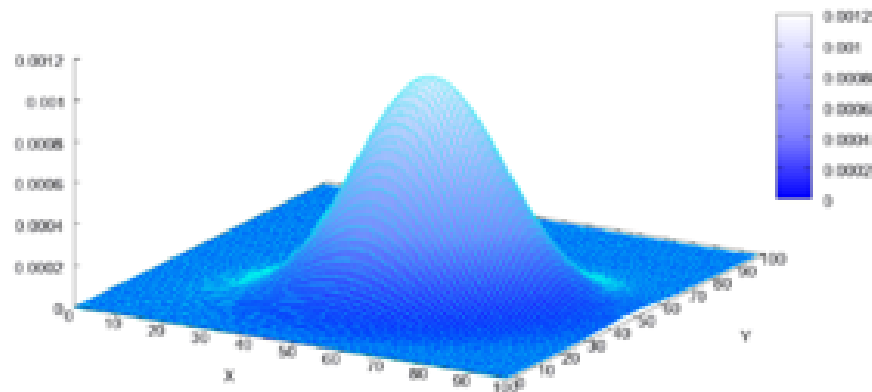
with

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

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



Multivariate Normal Distribution



Maximum-likelihood Estimators for Modes and Covariances

- One obtains a maximum likelihood estimators for the modes

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

- One obtains as unbiased estimators for the covariance matrix

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

Expanding the Quadratic Terms in the Exponent

- Note that

$$\begin{aligned} & -\frac{1}{2} \left(\mathbf{x}_i - \vec{\mu}^{(l)} \right)^T \Sigma^{-1} \left(\mathbf{x}_i - \vec{\mu}^{(l)} \right) \\ &= -\frac{1}{2} \mathbf{x}_i^T \Sigma^{-1} \mathbf{x}_i + \vec{\mu}^{(l)T} \Sigma^{-1} \mathbf{x}_i - \frac{1}{2} \vec{\mu}^{(l)T} \Sigma^{-1} \vec{\mu}^{(l)} \end{aligned}$$

- ... and thus, as we need next, for the difference of two quadratic terms (the first terms cancel)...

$$\begin{aligned} & -\frac{1}{2} \left(\mathbf{x}_i - \vec{\mu}^{(0)} \right)^T \Sigma^{-1} \left(\mathbf{x}_i - \vec{\mu}^{(0)} \right) + \frac{1}{2} \left(\mathbf{x}_i - \vec{\mu}^{(1)} \right)^T \Sigma^{-1} \left(\mathbf{x}_i - \vec{\mu}^{(1)} \right) \\ &= \left(\vec{\mu}^{(0)} - \vec{\mu}^{(1)} \right)^T \Sigma^{-1} \mathbf{x}_i - \frac{1}{2} \vec{\mu}^{(0)T} \Sigma^{-1} \vec{\mu}^{(0)} + \frac{1}{2} \vec{\mu}^{(1)T} \Sigma^{-1} \vec{\mu}^{(1)} \end{aligned}$$

A Posteriori Distribution

- It follows that

$$\begin{aligned} P(y_i = 1 | \mathbf{x}_i) &= \frac{P(\mathbf{x}_i | y_i = 1)P(y_i = 1)}{P(\mathbf{x}_i | y_i = 1)P(y_i = 1) + P(\mathbf{x}_i | y_i = 0)P(y_i = 0)} \\ &= \frac{1}{1 + \frac{P(\mathbf{x}_i | y_i = 0)P(y_i = 0)}{P(\mathbf{x}_i | y_i = 1)P(y_i = 1)}} \\ &= \frac{1}{1 + \frac{\kappa_0}{\kappa_1} \exp \left((\vec{\mu}^{(0)} - \vec{\mu}^{(1)})^T \Sigma^{-1} \mathbf{x}_i - \frac{1}{2} \vec{\mu}^{(0)T} \Sigma^{-1} \vec{\mu}^{(0)} + \frac{1}{2} \vec{\mu}^{(1)T} \Sigma^{-1} \vec{\mu}^{(1)} \right)} \\ &= \text{sig} \left(w_0 + \mathbf{x}_i^T \mathbf{w} \right) = \text{sig} \left(w_0 + \sum_j^M x_{i,j} w_j \right) \end{aligned}$$

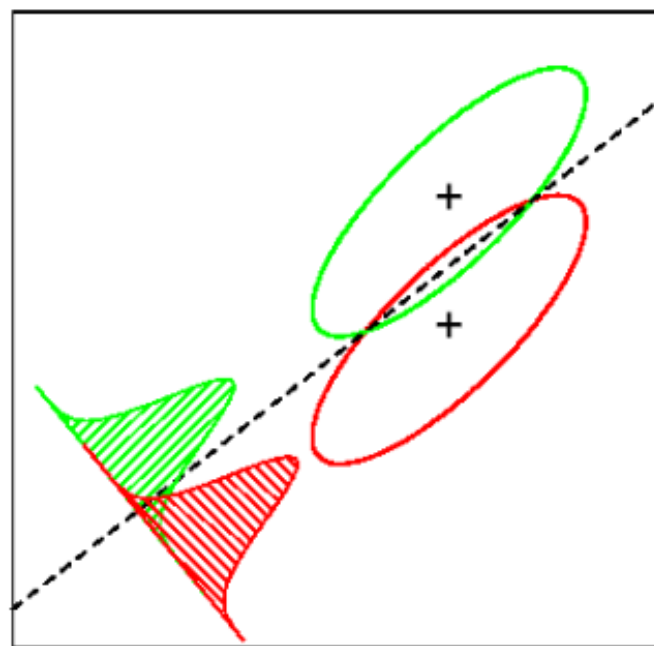
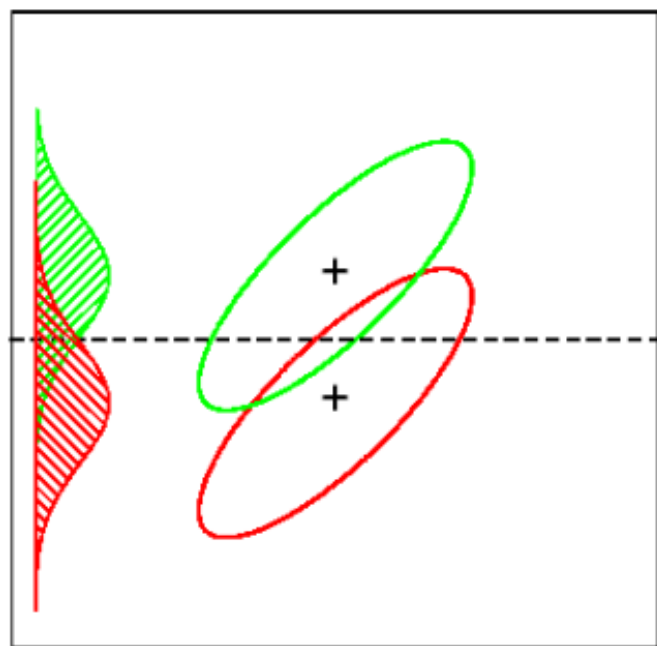
A Posteriori Distribution (cont'd)

- We get (\mathbf{w} is without w_0)

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

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

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



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; still, one can simply apply Bayes rule to obtain posterior probabilities

- The generalization to multiple classes is straightforward: simply estimate a different Gaussian for each class (with shared covariances or not) and apply Bayes rule
- Generative-Discriminative pair: Normal Discriminant Analysis (as a generative model) and logistic regression as a discriminant model

Special Case: Naive Bayes

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

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

- 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 (all Gaussian have identical variance):

$$P(\mathbf{x}_i | y_i = l) = \prod_{j=1}^M \mathcal{N}(x_{i,j}; \mu_j^{(l)}, \sigma^2)$$

Logistic Regression from Naive Bayes

- We have parameters, for the latter case,

$$w_j = \frac{1}{\sigma^2} \left(\mu_j^{(1)} - \mu_j^{(0)} \right)$$

$$w_0 = \log \kappa_1 / \kappa_0 + \frac{1}{2\sigma^2} \sum_j \left(\mu_j^{(0)} \right)^2 - \left(\mu_j^{(1)} \right)^2$$

- Note that w_j is completely independent of other inputs; adding or removing other inputs does not change w_j ;
- In contrast w_0 depends on all dimensions
- The smaller σ^2 , the sharper the transition

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

Special Case: Bernoulli Naive Bayes (cont'd)

- Then

$$P(\text{SPAM}|doc) = \frac{\kappa_s \prod_j \gamma_{j,s}^{word_j} (1 - \gamma_{j,s})^{1-word_j}}{\kappa_s \prod_j \gamma_{j,s}^{word_j} (1 - \gamma_{j,s})^{1-word_j} + \kappa_n \prod_j \gamma_{j,n}^{word_j} (1 - \gamma_{j,n})^{1-word_j}}$$

- 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)

Special Case: Bernoulli Naive Bayes (cont'd)

- Note, that we can also write

$$P(\text{SPAM}|\text{doc}) = \text{sig}(w_0 + \sum_j w_j \text{word}_j)$$

with

$$w_0 = \log \kappa_s / \kappa_n + \sum_j \log(1 - \gamma_{j,s}) - \log(1 - \gamma_{j,n})$$

$$w_j = [\log \gamma_{j,s} - \log \gamma_{j,n}] - [\log(1 - \gamma_{j,s}) - \log(1 - \gamma_{j,n})]$$

- Thus the Bernoulli naive Bayes classifier and logistic regression form a Generative-Discriminative pair, as well

II. Logistic Regression

- The Generative-Discriminative pairs, with logistic regression as the discriminant version, to directly model discriminatively

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

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

- One now optimizes the likelihood of the conditional model

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

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

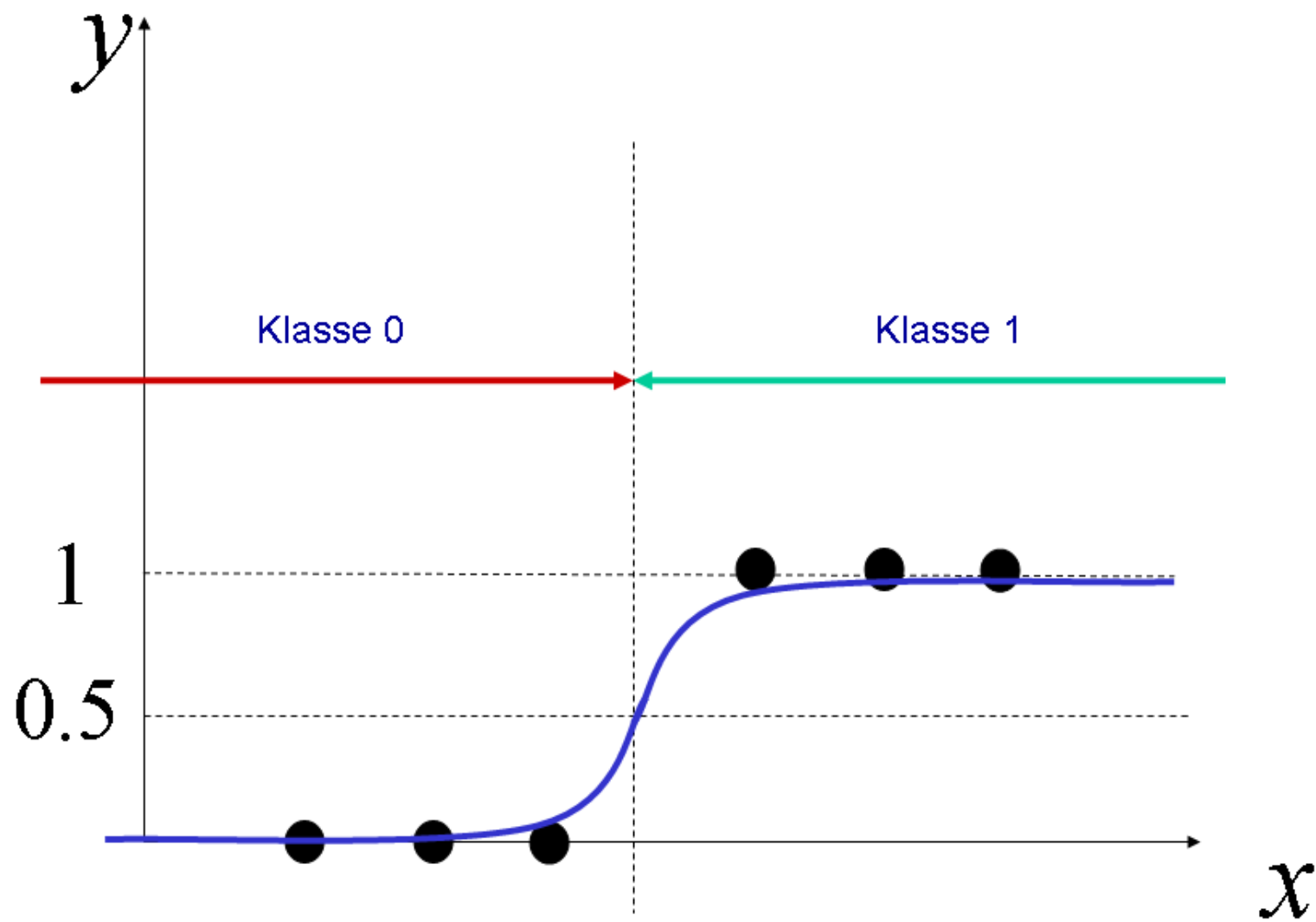
- The derivatives of the log-likelihood with respect to the parameters

$$\begin{aligned}\frac{\partial l}{\partial \mathbf{w}} &= \sum_{i=1}^N y_i \frac{\mathbf{x}_i \exp(-\mathbf{x}_i^T \mathbf{w})}{1 + \exp(-\mathbf{x}_i^T \mathbf{w})} - (1 - y_i) \frac{\mathbf{x}_i \exp(\mathbf{x}_i^T \mathbf{w})}{1 + \exp(\mathbf{x}_i^T \mathbf{w})} \\ &= \sum_{i=1}^N y_i \mathbf{x}_i (1 - \text{sig}(\mathbf{x}_i^T \mathbf{w})) - (1 - y_i) \mathbf{x}_i \text{sig}(\mathbf{x}_i^T \mathbf{w}) \\ &= \sum_{i=1}^N (y_i - \text{sig}(\mathbf{x}_i^T \mathbf{w})) \mathbf{x}_i\end{aligned}$$

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

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

- Typically one uses a Newton-Raphson optimization procedure



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 $\theta = \text{sig}(\eta)$. Now we get valid probabilities for any $\eta \in \mathbb{R}$!
- η is called the natural parameter and $\text{sig}(\cdot)$ 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)

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

- *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

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

or systems of basis functions or kernel systems

Multiple Classes and Softmax

- Consider a multinomial distribution with $P(y = c) = \theta_c$, with $\theta_c \geq 0$ and $\sum_{c=1}^C \theta_c = 1$. c is the class index and C is the number of classes
- We reparameterize (exponential family of distributions)

$$\theta_c = \frac{\exp(\eta_c)}{\sum_{c'=1}^C \exp(\eta_{c'})}$$

- The η_c are unconstrained; **softmax** notation: $\theta_c = \text{softmax}_c(\vec{\eta})$
- In GLM, we set $\eta_c = \mathbf{x}^T \mathbf{w}_c$ and

$$P(y = c | \mathbf{x}) = \frac{\exp(\mathbf{x}^T \mathbf{w}_c)}{\sum_{c'=1}^C \exp(\mathbf{x}^T \mathbf{w}_{c'})}$$

- The negative log-likelihood (cross entropy) becomes

$$-l = - \sum_{i=1}^N \left(\sum_{c=1}^C y_{i,c} \mathbf{x}_i^T \mathbf{w}_c - \log \sum_{c=1}^C \exp(\mathbf{x}_i^T \mathbf{w}_c) \right)$$

Multiple Classes and Softmax (cont'd)

- The gradient becomes

$$-\frac{\partial l}{\partial w_{j,c}} = -\sum_i \left(y_{i,c} x_{i,j} - \frac{x_{i,j} \exp(\mathbf{x}_i^T \mathbf{w}_c)}{\sum_{c=1}^C \exp(\mathbf{x}_i^T \mathbf{w}_c)} \right)$$

and SGD becomes

$$w_{j,c} \leftarrow w_{j,c} + \eta x_{i,j} (y_{i,c} - \text{softmax}_c(\mathbf{x}_i^T \mathbf{w}_c))$$

- Compare: for the Bernoulli model with C binary classes, we got

$$w_{j,c} \leftarrow w_{j,c} + \eta x_{i,j} \left(y_{i,c} - \text{sig} \left(\mathbf{x}_i^T \mathbf{w}_c \right) \right)$$

III. Classification via Regression

- Linear Regression:

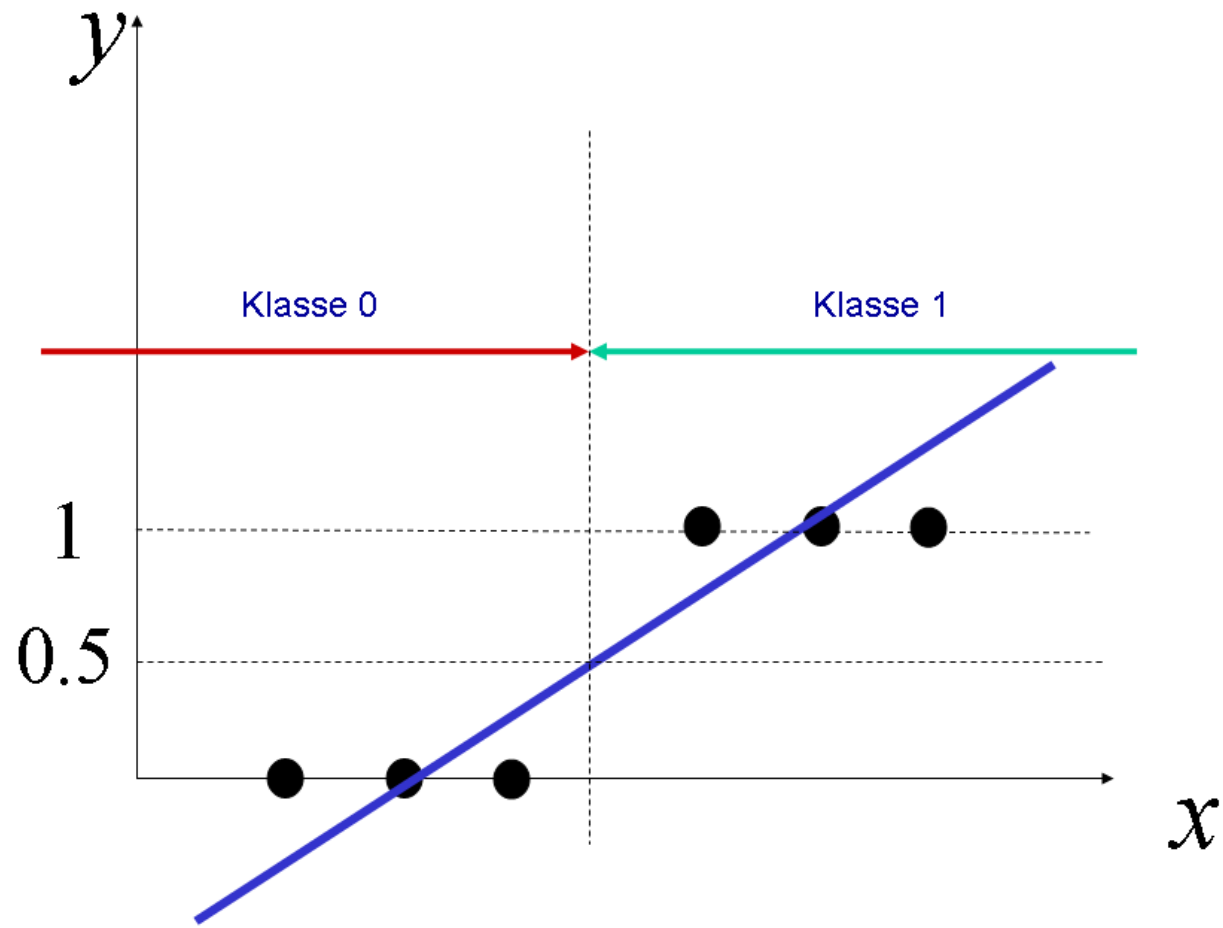
$$\begin{aligned} f(\mathbf{x}_i, \mathbf{w}) &= w_0 + \sum_{j=1}^{M-1} w_j x_{i,j} \\ &= \mathbf{x}_i^T \mathbf{w} \end{aligned}$$

- 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

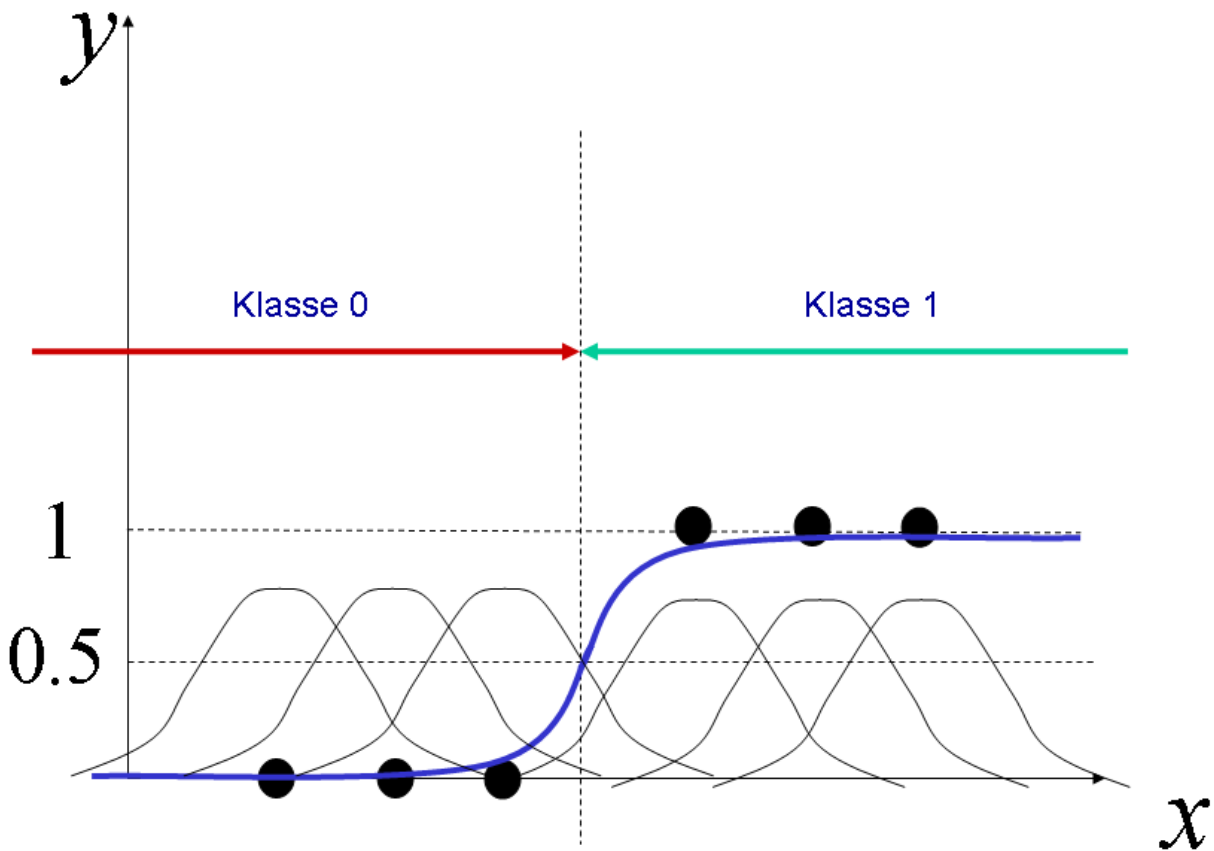
Bias

- Asymptotically, a LS-solution converges to the posterior class probabilities, although a linear function 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. In essence it is necessary that the linear model can model the expected probability $P(c = 1|\mathbf{x})$

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!

Logistic Regression in Medical Statistics

Logistic Regression in Medical Statistics

- Logistic regression has become one of the the most important tools in medicine to analyse the outcome of treatments
- In **epidemiology**, the output $y = 1$ 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, smoking, or an environmental factor) and $x_1 = 0$ might mean that the patient was not exposed; the other inputs are often typical confounders (age, sex, ...)
- In **treatment evaluation**, $x_1 = 1$ means that the patient received the treatment, and $x_1 = 0$ means that the patient did not receive the treatment; the output represents the outcome after treatment; e.g., $y = 1$ can mean that the patient is cured by the treatment
- Logistic regression then permits the prediction of the outcome for any patient
- Of course, of great interest is if w_1 is significantly nonzero, which relates to the individual's response, but also the population response (i.e., the exposure is harmful in the population; the treatment is successful in the population)

Log-Odds

- The **odds** for a patient with properties \mathbf{x}_i is defined as

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

- For logistic regression, the log odds is

$$\begin{aligned} \text{LogOdds} &= \log \frac{P(y_i = 1|\mathbf{x}_i)}{P(y_i = 0|\mathbf{x}_i)} = \log \frac{1}{1 + \exp(-\mathbf{x}_i^T \mathbf{w})} \frac{1 + \exp(-\mathbf{x}_i^T \mathbf{w})}{\exp(-\mathbf{x}_i^T \mathbf{w})} \\ &= \log \frac{1}{\exp(-\mathbf{x}_i^T \mathbf{w})} = \mathbf{x}_i^T \mathbf{w} \end{aligned}$$

- Thus the **log odds** of the outcome is $h = \mathbf{x}_i^T \mathbf{w}$, which is the net input or the **logit**

Log Odds Ratio

- The **odds ratio** is defined as

$$OR = \frac{Odds(\mathbf{x}_{x_1=1})}{Odds(\mathbf{x}_{x_1=0})}$$

- The **log odds ratio** evaluates the effect of the treatment

$$\log(OR) = \log Odds(\mathbf{x}_{x_1=1}) - \log Odds(\mathbf{x}_{x_1=0})$$

- In logistic regression, we get the log odds ratio is identical to w_1 , since

$$(w_0 + w_1 + \sum_{j=2}^N x_{i,j}) - (w_0 + 0 + \sum_{j=2}^N x_{i,j}) = w_1$$

- If w_1 is significantly nonzero, then the exposure/treatment has an effect; the odds ratio is commonly used in case-control studies!

Appendix: Information Theory

Entropy

- Here we consider discrete variables; corresponding results also exist for continuous variables
- In information theory, the **entropy** is defined as

$$H(P) = - \sum_x P(X = x) \log P(X = x)$$

- It represents the (minimum) number of bits required to encode data points generated from $P(X)$; this is a nonnegative quantity since $\log P(X = x) \leq 0$

Cross Entropy

- One is interested in the relationship between the true distribution P (which generated the data) and the approximating distribution Q (e.g., a machine learning model)
- The **cross entropy** between a true distribution P and an approximative distribution Q is defined as

$$H(P, Q) = - \sum_x P(X = x) \log Q(X = x)$$

- It represents the minimum number of required bits if the encoding assumes $Q(X)$ whereas the true distribution is $P(X)$

KL-Divergence

- The cross-entropy cannot be smaller than the entropy: $H(P, Q) \geq H(P)$. The difference between the cross entropy and the entropy is a distance measure for distributions and is called the **relative entropy** or **KL divergence** (Kullback-Leibler divergence)

$$D(P\|Q) = H(P, Q) - H(P)$$

- We get

$$D(P\|Q) = \sum_x P(X = x) \log \frac{P(X = x)}{Q(X = x)}$$

Cross-entropy Cost Function and Log-Likelihood

- Consider that you fit a **model** Q to the data
- We apply the **cross entropy** and assume that $P(X)$ is approximated by the empirical distribution of the data $\{\mathbf{x}_i\}_i$ and $Q(X)$ represents the model and we get

$$H(P, Q) \approx - \sum_i \log Q(\mathbf{x}_i) = -\log\text{Likelihood}(Q)$$

- Thus, with this approximation, the **cross entropy is identical to the negative log-likelihood**
- Assuming that $P(X)$ is approximated by the empirical distribution, the negative log-likelihood is also identical to the KL divergence plus the entropy

$$-\log\text{Likelihood}(Q) \approx D(P\|Q) + H(P)$$

$H(P)$ is only data dependent and not model dependent; thus minimizers agree

Cross-entropy Cost Function for Prediction (Conditional Models)

- We consider the true $P(Y|X)$ and the model $Q(Y|X)$; The cross entropy of interest is

$$H(P(Y|X), Q(Y|X)) = - \int P(Y|X) \log Q(Y|X) dY$$

- We now take the expectation w.r.t. X , and get

$$\begin{aligned} E_X H(P(Y|X), Q(Y|X)) &= - \int \int P(Y|X) \log Q(Y|X) P(X) dY dX \\ &= - \int \int P(X, Y) \log Q(Y|X) dY dX \end{aligned}$$

- We approximate $P(X, Y)$ by the observed data $\{\mathbf{x}_i, y_i\}_i$ and get

$$E_X H(P(Y|X), Q(Y|X)) \approx - \sum_i \log Q(y_i | \mathbf{x}_i)$$

which again is the negative log likelihood for supervised learning

Relationship between Random Variables

- Whereas in the definition of entropy, cross entropy and KL divergence, we were interested in the relationships between different distributions defined on the same random variables, another goal is to quantify relationships between two different random variables X and Y when $P(X, Y)$ is given

- **Conditional Entropy:** is defined as

$$H(Y|X) = \sum_x \sum_y P(X = x, Y = y) P(Y = y|X = x)$$

- In information theory, the conditional entropy (or equivocation) quantifies the amount of information needed to describe the outcome of a random variable Y given that the value of another random variable X is known

Mutual Information

- **Mutual Information:** is defined as

$$I(X; Y) = \sum_y \sum_x P(X = x, Y = y) \log \left(\frac{P(X = x, Y = y)}{P(X = x) P(Y = y)} \right)$$

- It is symmetric and can be written as

$$I(X; Y) = H(Y) - H(Y|X) = H(X) - H(X|Y)$$

- Intuitively, mutual information measures the information that X and Y both share: It measures how much knowing one of these variables reduces uncertainty about the other; it is useful for calculating the channel capacity in communication theory
- Mutual information is one quantity used to determine if two random variables are dependent or independent; for example it might be used if an input X is predictive for an output Y , in particular if both quantities are discrete

Cross-entropy Cost Function for Binary Classification

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

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

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

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

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

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