Linear Classifiers

Volker Tresp

Winter 2024-2025

Classification

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

Overlapping Classes

- The beauty of Machine Learning is that a few model classes (neural networks, kernel approaches, ...) can be applied to almost any supervised learning task
- This hides a bit that the data settings can be quite different
- There are problems where class boundaries are well defined but maybe quite complex; an example is OCR; here Deep Neural Networks, manifold learning and kernel systems are quite effective; this concerns often our Cases I and II
- In other applications there is little structure in the data and classes overlap; this the situation encountered in many healthcare applications (biomedicine); this concerns often our Cases III and IV
- Often, the problem is not as much to separate classes, but to show that there is a signal at all; the question might be if there is a detectable positive effect of the new medication!

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

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

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

(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

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

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

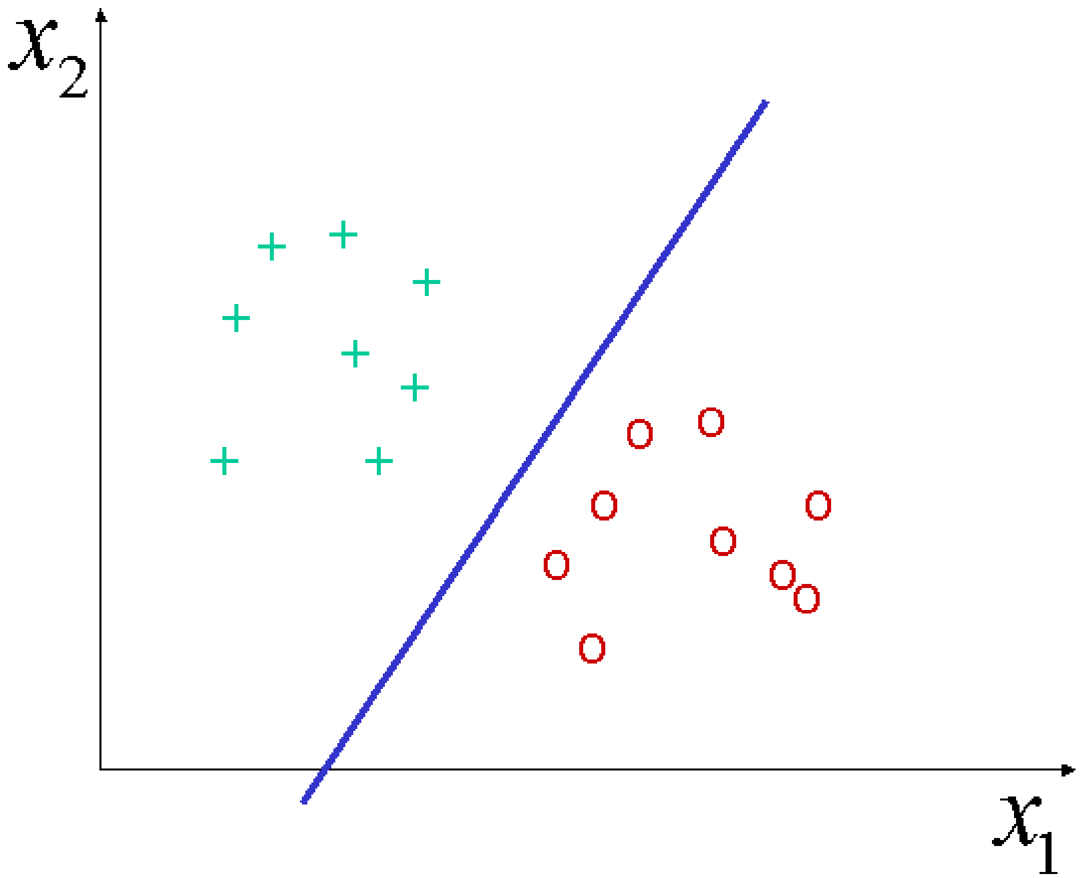
Binary Classification Problems

- 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 $y_i = -1$)

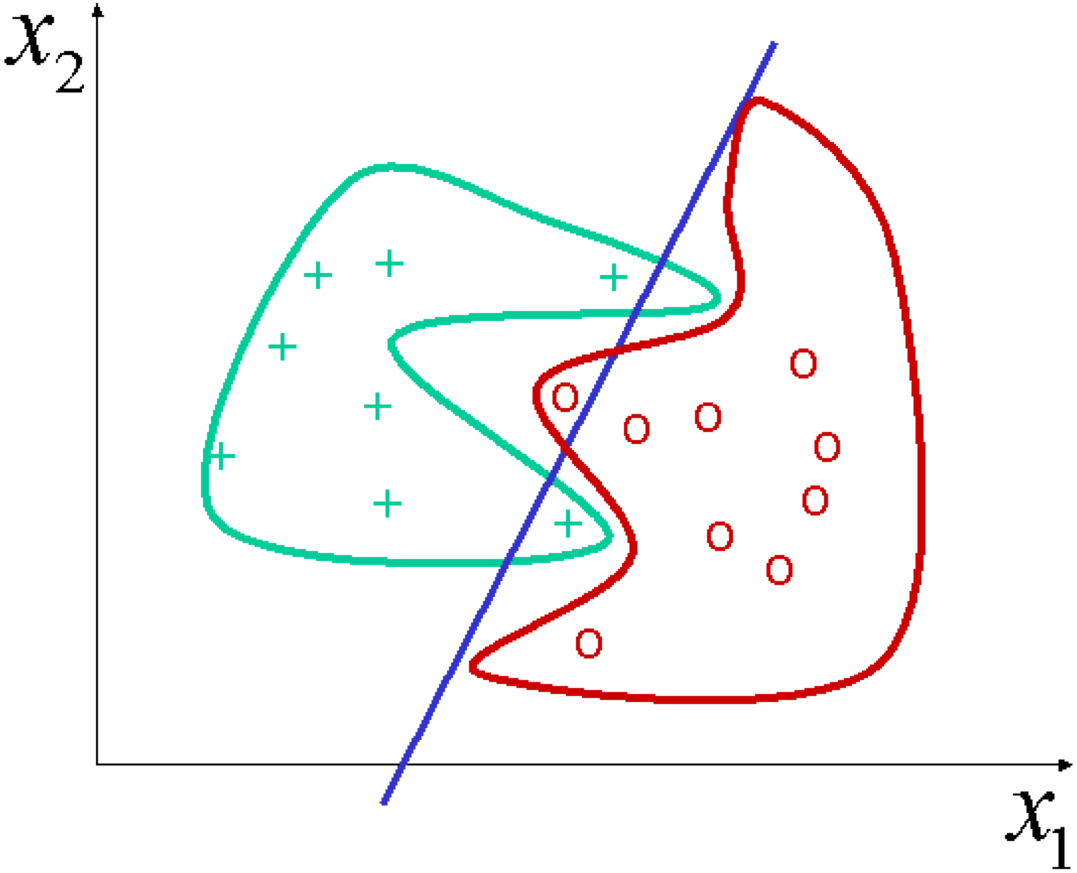
- We already know the *Perceptron*. Now we learn about additional approaches

  - I. Generative models for classification
  - II. Logistic regression
  - III. Classification via regression

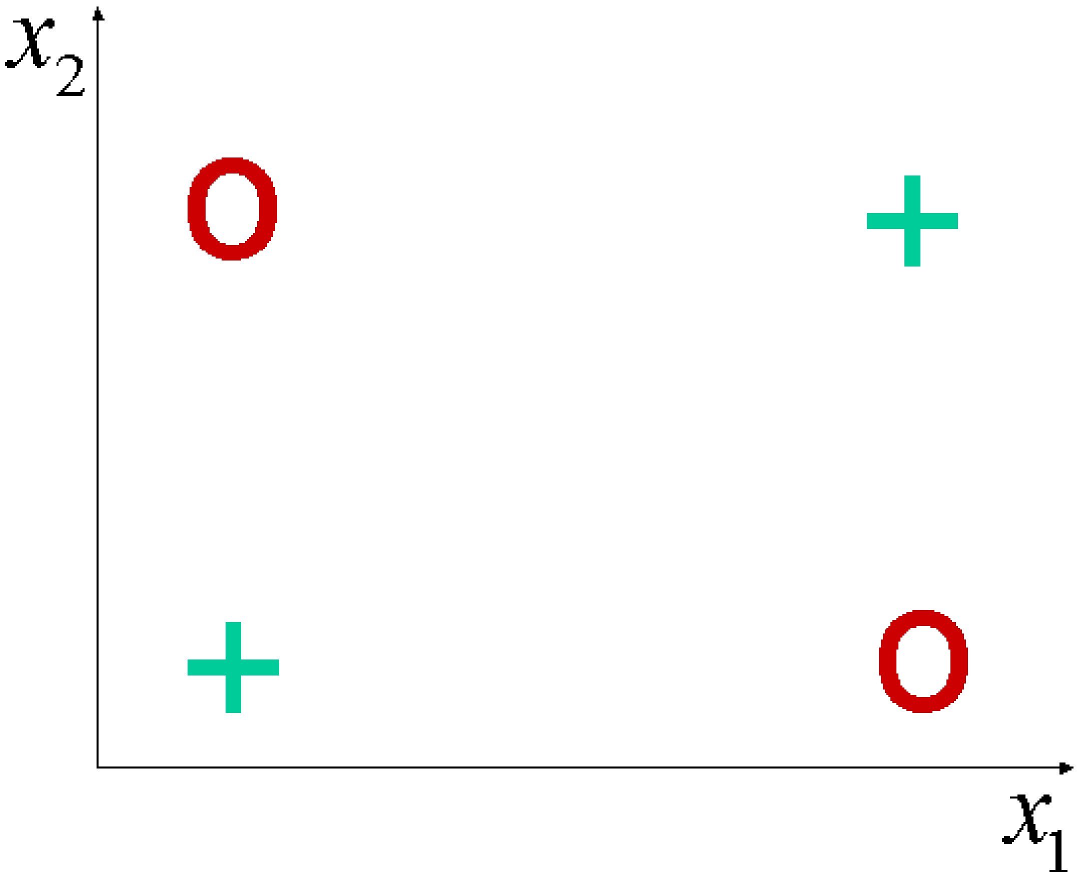
Two Linearly Separable Classes



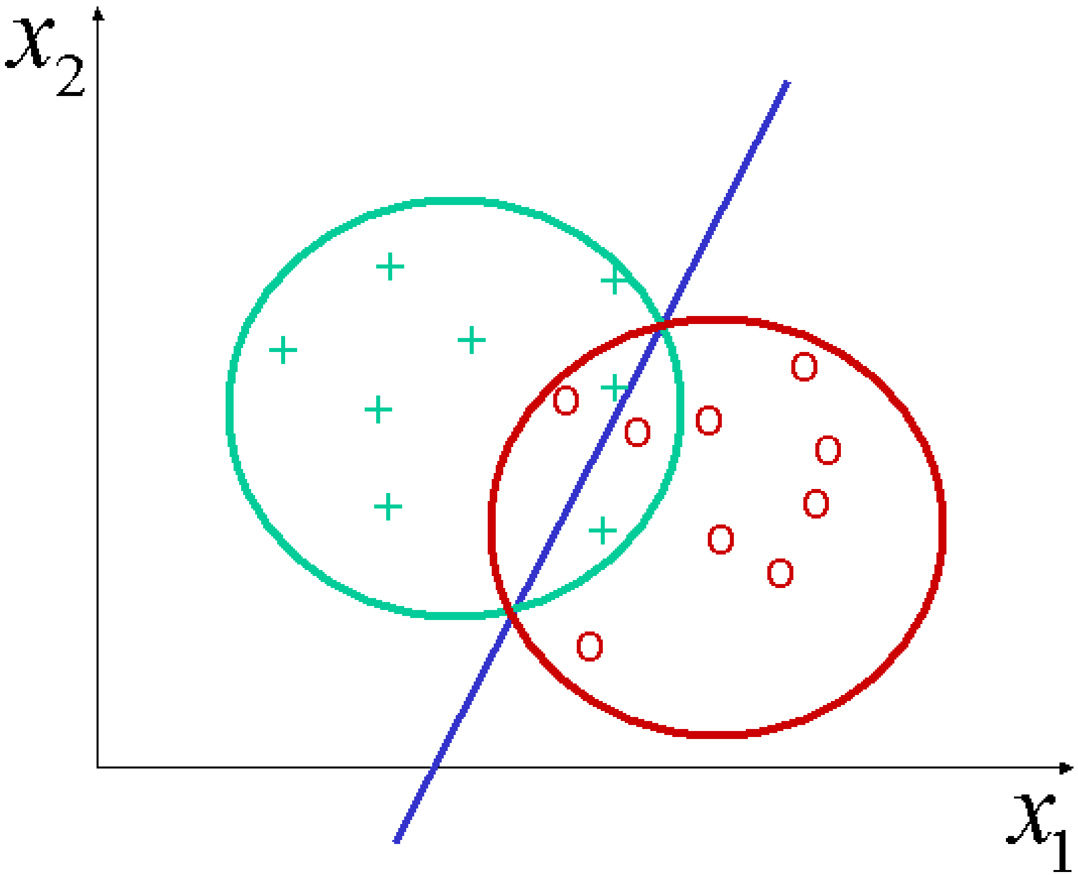
Two Classes that Cannot be Separated Linearly



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: data is generated from class-specific Gaussian distributions

- 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.

Generative Model for Classification (cont'd)

- Then we have a model how attributes are generated, given the classes $P(\tilde{\mathbf{x}}|y)$. This could stand for
  - Income, age, occupation ($\tilde{\mathbf{x}}$) given a customer is a good customer ($y = 1$)
  - Income, age, occupation ($\tilde{\mathbf{x}}$) given a customer is not a good customer ($y = 0$)
- Using Bayes formula, we then derive $P(y|\tilde{\mathbf{x}})$: 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?

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

$$P(y_i = 1) = \kappa = \text{sig}(q) \qquad P(y_i = 0) = 1 - \kappa_1 = 1 - \text{sig}(q)$$

with $0 \leq \kappa \leq 1$.

- In a next step, a data point $\tilde{\mathbf{x}}_i$ has been generated from $P(\tilde{\mathbf{x}}_i | y_i)$
- (Note, that $\tilde{\mathbf{x}}_i = (x_{i,1}, \ldots, x_{i,M})^T$, which means that $\tilde{\mathbf{x}}_i$ does not contain the bias $x_{i,0}$)
- We now have a complete model: $P(y_i)P(\tilde{\mathbf{x}}_i | y_i)$

Bayes' Theorem

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

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

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

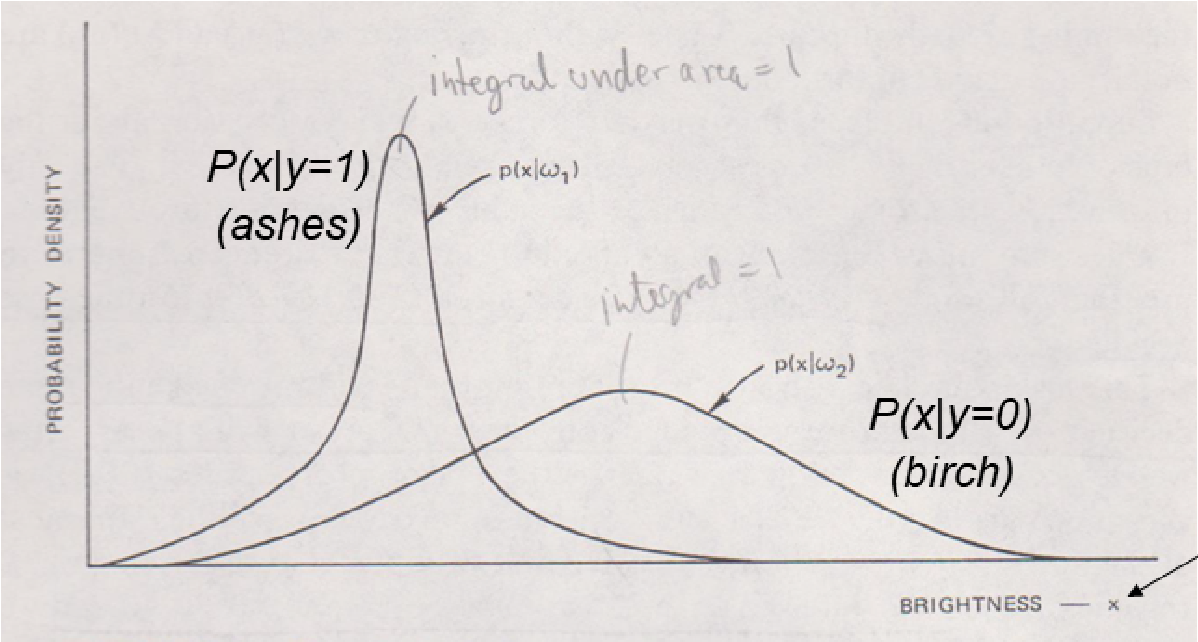
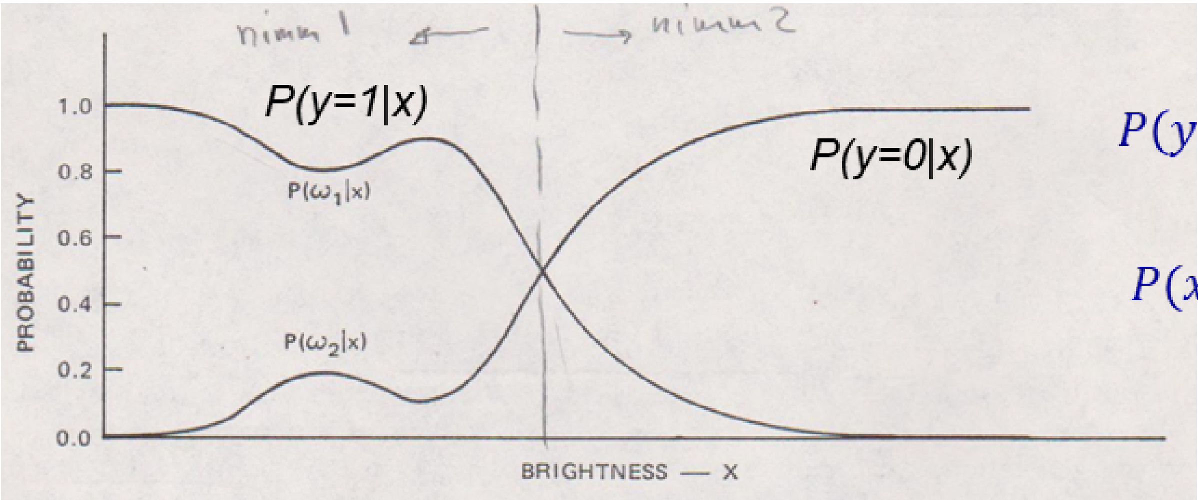


FIGURE 2.1. Hypothetical class-conditional probability density functions.

only one input



Distinguishing between ashes ("Eschen") and birches ("Birken")
(Duda & Hart, Pattern Classification and Scene Analysis, First Edition)

The magic of Bayes formula:

$$P(y=1|x) = \frac{P(x|y=1)P(y=1)}{P(x)}$$

$$P(x) = P(x|y=1)P(y=1) + P(x|y=0)P(y=0)$$

Birches versus Ashes

- The last figure also nicely exemplifies the problem of overlapping classes
- Given brightness level as input, one cannot separate the classes and this problem cannot be solved by a more powerful classifier!
- The only way to solve this issue is to use more features (inputs, sensors); for example one might measure spectral amplitudes at different frequencies, including infrared
- Another problem might be that the brightness detector is unreliable ("noisy labels")

Class-specific Distributions

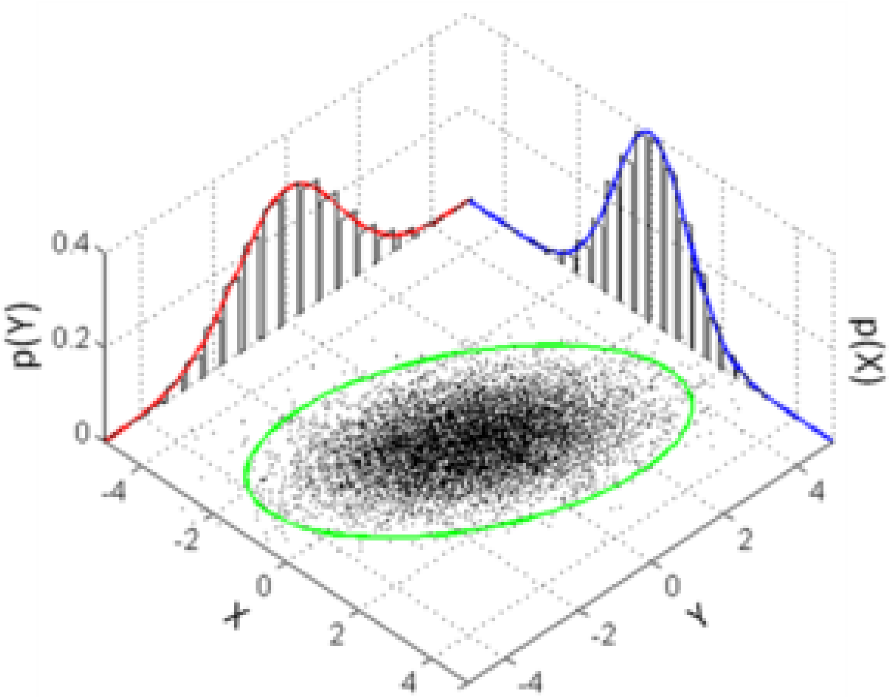
- To model $P(\tilde{\mathbf{x}}_i|y_i)$ one can chose an application specific distribution
- A popular choice is a Gaussian distribution (normal discriminant analysis)

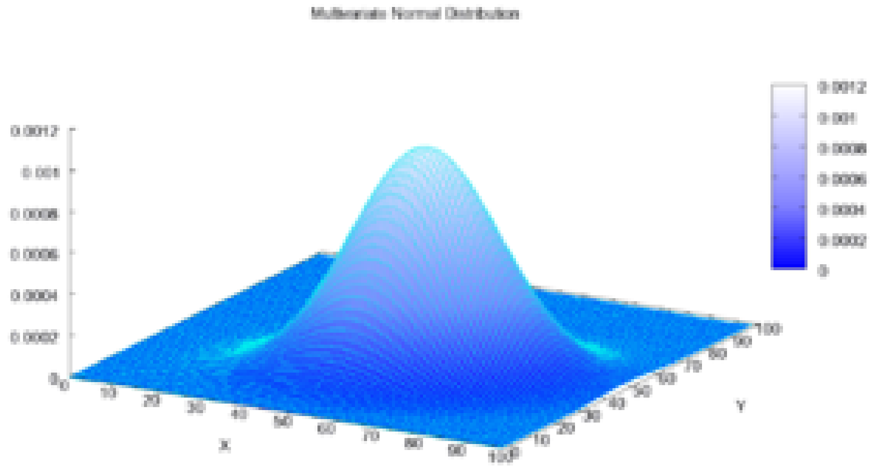
$$P(\tilde{\mathbf{x}}_i|y_i = l) = \mathcal{N}(\tilde{\mathbf{x}}_i; \vec{\mu}^{(l)}, \Sigma)$$

  with

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

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


p(Y)
p(X)
X
Y


Multivariate Normal Distribution
X
Y

Class-specific Distributions

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

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

$$= \mathcal{N}(\tilde{\mathbf{x}}_i; l\vec{\mu}^{(1)} + (1-l)\vec{\mu}^{(0)}, \Sigma)$$

$$= \mathcal{N}(\tilde{\mathbf{x}}_i; l(\vec{\mu}^{(1)} - \vec{\mu}^{(0)}) + \vec{\mu}^{(0)}, \Sigma)$$

Maximum-likelihood Estimators for Modes and Covariances

- One obtains a maximum likelihood estimators for the modes

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

- One obtains as unbiased estimators for the covariance matrix

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

Expanding the Quadratic Terms in the Exponent

- Note that

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

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

The Difference of the Quadratic

- Now we calculate the difference of the quadratic terms of the two Gaussians

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

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

$$+\frac{1}{2}\tilde{\mathbf{x}}_i^T\Sigma^{-1}\tilde{\mathbf{x}}_i+\frac{1}{2}\vec{\mu}^{(1)^T}\Sigma^{-1}\vec{\mu}^{(1)}-\vec{\mu}^{(1)^T}\Sigma^{-1}\tilde{\mathbf{x}}_i$$

- .... since two terms cancel,

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

A Posteriori Distribution

- It follows that

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

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

$$= \mathrm{sig}[\log(P(\tilde{\mathbf{x}}_i|y_i = 1)) + \log(P(y_i = 1)) - \log(P(\tilde{\mathbf{x}}_i|y_i = 0) - \log P(y_i = 0))]$$

$$= \mathrm{sig}\left(w_0 + \tilde{\mathbf{x}}_i^T \tilde{\mathbf{w}}\right) = \mathrm{sig}\left(w_0 + \sum_{j=1}^{M} x_{i,j} w_j\right)$$

Weights

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

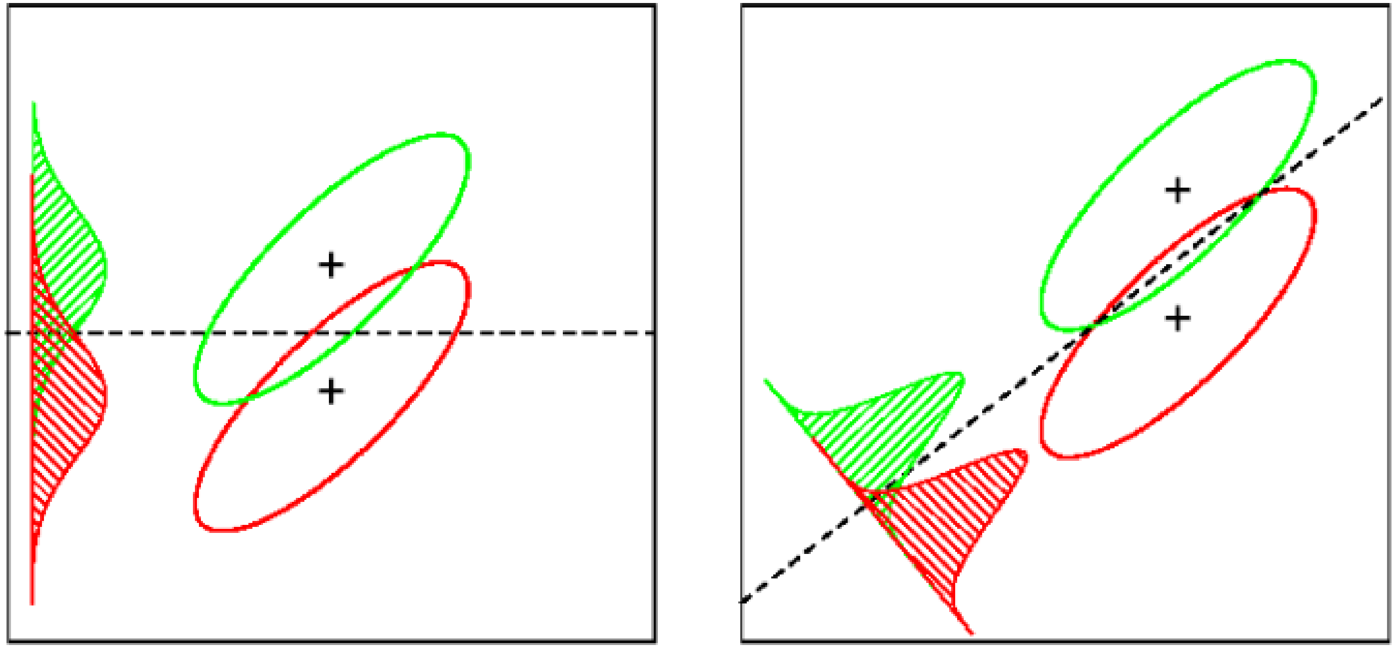
$$\tilde{\mathbf{w}} = \Sigma^{-1} \left(\vec{\mu}^{(1)} - \vec{\mu}^{(0)} \right)$$

- Note that $\tilde{\mathbf{w}}$ is independent of κ and is thus independent of the class proportions in the training data! This is important, e.g., for case-control studies
- Recall: $\mathrm{sig}(\mathit{arg}) = 1/(1 + \exp(-\mathit{arg}))$

Bias Term

- We get,

$$w_0 = q + \frac{1}{2}\,\vec{\mu}^{(0)^T}\Sigma^{-1}\vec{\mu}^{(0)} - \frac{1}{2}\,\vec{\mu}^{(1)^T}\Sigma^{-1}\vec{\mu}^{(1)}$$



Comments

- This specific generative model leads to linear class boundaries
- But we do not only get class boundaries, we get probabilities
- Although we have used Bayes formula, the analysis was frequentist. A Bayesian analysis with a prior distribution on the parameters is also possible

Comments (cont'd)

- 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*: (1) Gaussian Analysis (as a generative model) and (2) logistic regression as a discriminant model

- Generalization to basis functions is straight forward: $\mathbf{x}$ is replaced by $\vec{\phi}(\mathbf{x})$

- With an explicit $P(\tilde{\mathbf{x}}_i|y_i = l) = \mathcal{N}(\tilde{\mathbf{x}}_i; \vec{\mu}^{(l)}, \Sigma)$, we can apply Bayes formula for a posteriori class estimation

- This is not easy, or even impossible, e.g., for GANs, which are able to generate samples but where the likelihood is not easily evaluated (likelihood free methods)

Special Case: Naive Bayes

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

$$P(\tilde{\mathrm{x}}_i | y_i = l) = \prod_{j=1}^{M} \mathcal{N}(x_{i,j}; l(\mu_j^{(1)} - \mu_j^{(0)}) + \mu_j^{(0)}, \sigma_j^2)$$

- 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):

$$P(\tilde{\mathrm{x}}_i | y_i = l) = \prod_{j=1}^{M} \mathcal{N}(x_{i,j}; l(\mu_j^{(1)} - \mu_j^{(0)}) + \mu_j^{(0)}, \sigma^2)$$

Logistic Regression from Naive Bayes

- We have parameters, for the latter case,

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

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

- Note that w_j is completely independent of inputs other than x_j; 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 ($l = 1$) and no-SPAM ($l = 0$) and the features are keywords in the texts
- Instead of a Gaussian distribution, a Bernoulli distribution is employed

Special Case: Bernoulli Naive Bayes

- $P(\mathit{word}_j = 1|l = 1) = \gamma_{j,s} = \mathrm{sig}(a_{j,s})$ is the probability of observing word $\mathit{word}_j \in \{0, 1\}$ in the document for SPAM documents (Bernoulli distribution)
- $P(\mathit{word}_j = 1|l = 0) = \gamma_{j,n} = \mathrm{sig}(a_{j,n})$ is the probability of observing word $word_j$ in the document for non-SPAM documents

Special Case: Bernoulli Naive Bayes (cont'd)

- Then, the posterior is

$$P(\mathsf{SPAM} = 1|\mathit{doc}) =$$

$$\frac{\kappa \prod_j \gamma_{j,s}^{\mathit{word}_j} (1-\gamma_{j,s})^{1-\mathit{word}_j}}{\kappa \prod_j \gamma_{j,s}^{\mathit{word}_j} (1-\gamma_{j,s})^{1-\mathit{word}_j} + (1-\kappa) \prod_j \gamma_{j,n}^{\mathit{word}_j} (1-\gamma_{j,n})^{1-\mathit{word}_j}}$$

- 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, using the equations on the logistic function,

$$P(\mathrm{SPAM} = 1 | doc) = \mathrm{sig}(w_0 + \sum_j w_j \textit{word}_j)$$

with re-parametrization and logit identity,

$$w_j = a_{j,s} - a_{j,n}$$

$$w_0 = q - \sum_j \log(1 + \exp a_{j,s}) + \sum_j \log(1 - \exp a_{j,n})$$

- *Generative-Discriminative pair:* (1) Bernoulli naive Bayes classifier and (2) logistic regression

II. Logistic Regression

- In I. (Generative models for classification) we first defined a generative model for $P(\mathbf{x}, y)$; from this model we then derived $P(y|\mathbf{x}) = P(y)P(\mathbf{x}|y)$ which models $\mathbf{x}$ given y (generative modelling)

- Here, we model the reverse $P(y|\mathbf{x})$ (standard supervised learning)

- With logistic regression as the discriminant version, we model discriminatively

$$\hat{y}_i = P(y = 1|\mathbf{x}_i) = \mathrm{sig}\left(\mathbf{x}_i^T \mathbf{w}\right)$$

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

- One now optimizes the likelihood of the conditional model

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

Cross Entropy is the Negative Log-Likelihood Function

- Log-likelihood function

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

- With the rules about the logistic function

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

- Cross-entropy cost function (negative log-likelihood)

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

SGD

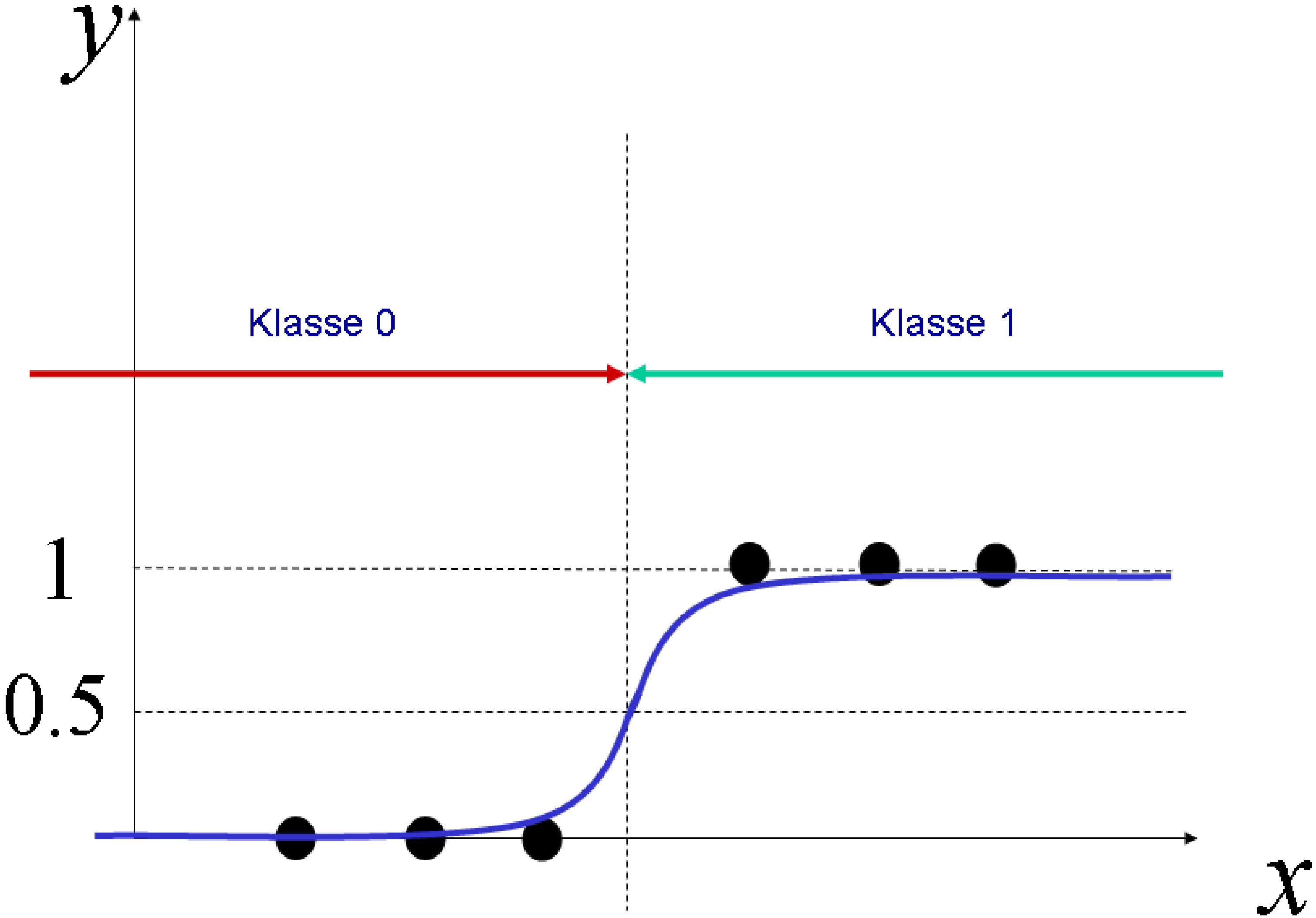
- The gradient of the cross-entropy cost function is

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

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

- SGD becomes for step t

$$w_j \leftarrow w_j + \eta(y_t - \widehat{y}_t)x_{t,j}$$


y
Klasse 0
Klasse 1
1
0.5
x

Logistic Regression as a Generalized Linear Models (GLM)

- Consider a Bernoulli distribution with $P(y = 1) = \kappa$ and $P(y = 0) = 1 - \kappa$, with $0 \leq \kappa \leq 1$
- In the theory of the exponential family of distributions, one sets

  $$\kappa = \mathrm{sig}(\eta)$$

  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

Logistic Regression as a Generalized Linear Models (GLM) (cont'd)

- 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, as we have done before,

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

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: $\theta_c = \text{softmax}_c(\vec{\eta}_c)$

Multiple Classes and Softmax: GLM

- In GLM, we set $\eta_c = \mathbf{x}^T \mathbf{w}_c$ and

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

Multiple Classes and Softmax (cont'd)

- The negative log-likelihood (softmax 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

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

  and SGD becomes for iteration t

$$w_{j,c} \leftarrow w_{j,c} + \eta x_{t,j}(y_{t,c} - \widehat{y}_{t,c})$$

III. Classification via Regression

- Linear Regression:

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

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

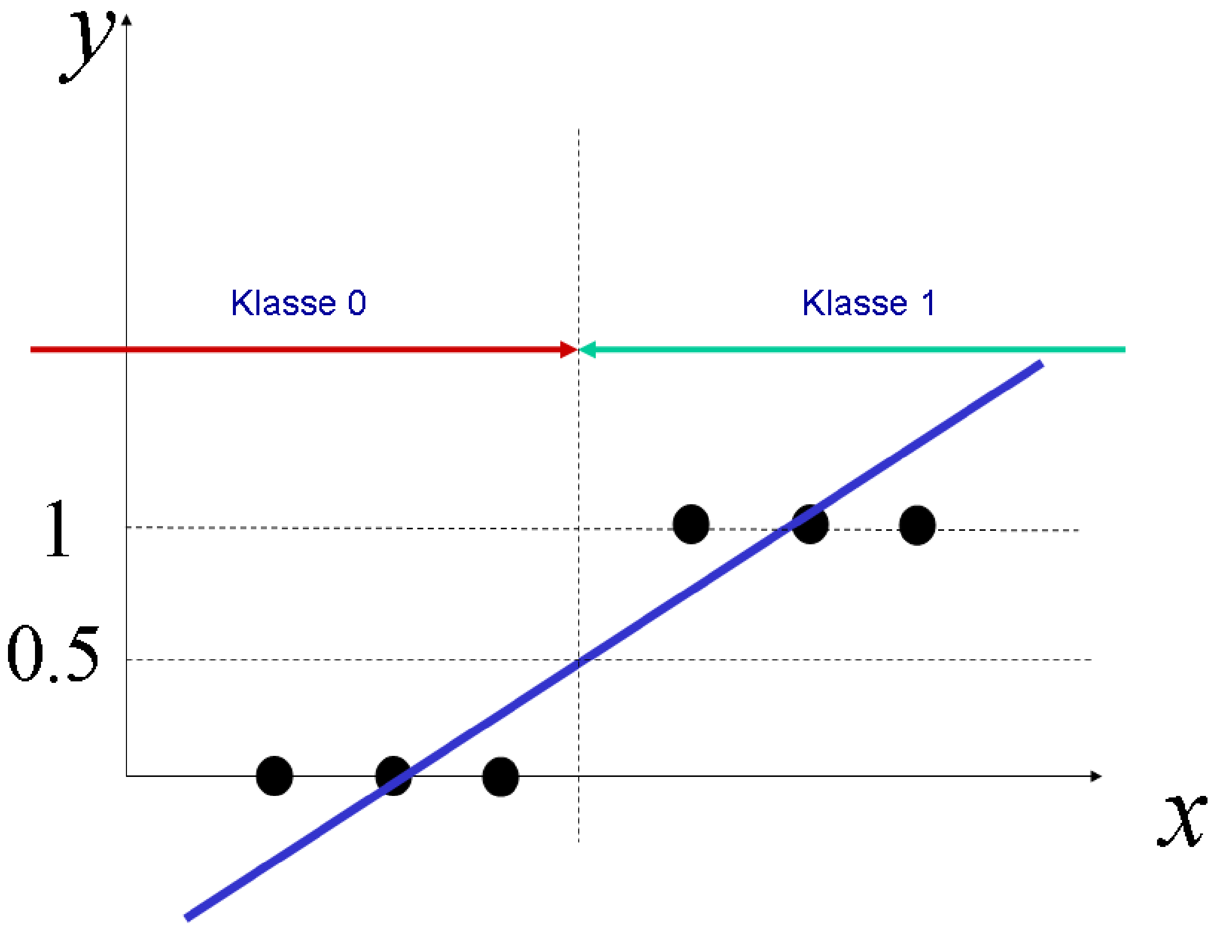
- We define as target $y_i = 1$ if the pattern $\mathbf{x}_i$ belongs to class 1 and $y_i = 0$ (or $y_i = -1$) if pattern $\mathbf{x}_i$ belongs to class 0
- We calculate weights $\mathbf{w}_{LS} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$ as LS solution, exactly as in linear regression
- 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

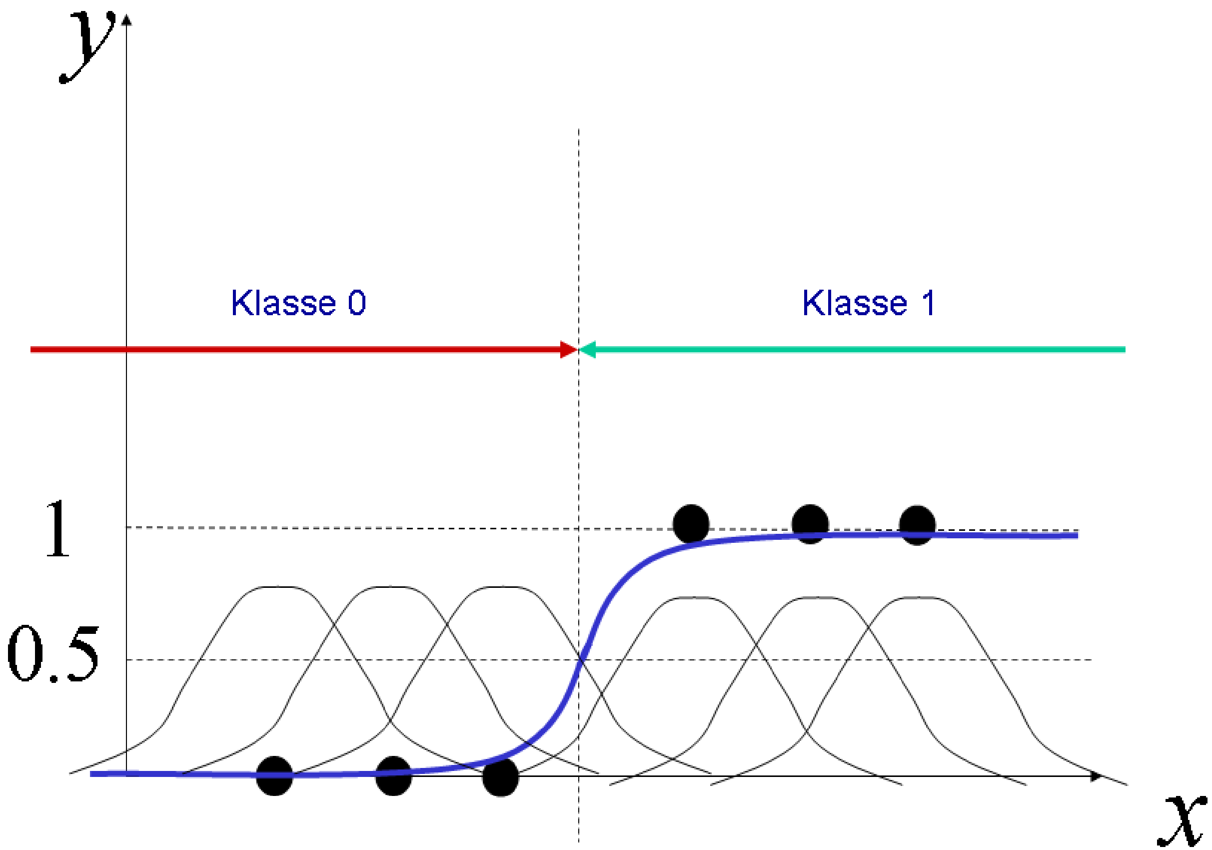
- Asymptotically, a LS-solution converges to the posterior class probabilities, although a linear functions is typically not able to represent $P(c = 1|\mathbf{x})$. The resulting class boundary can still be sensible

- 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. 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



Causal Effect

- Assume that all relevant inputs are considered in the model (no other confounders) and that we use "Classification via Regression"

- The causal effect is independent of the individual, and can be estimated as

$$P(Y = 1|x_{i,1} = 1, x_{i,2}, \ldots, x_{i,M}) - P(Y = 1|x_{i,1} = 0, x_{i,2}, \ldots, x_{i,M}) = w_1$$

- $x_1 = 1$ means that the individual has received the treatment, and $x_1 = 0$ means that the individual has not received the treatment,

- $Y = 1$ means that the patient is healthy after the treatment

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!

- Classification via regression is commonly used in treatment effect prediction in medicine if the influence of the treatment is small, on average

Appendix: Useful Identities for Logistic Regression

Useful Identities for Logistic Regression

- Logistic function (sigmoid function): $\kappa = \mathrm{sig}(\eta) = 1/(1 + \exp(-\eta))$
- For the logarithm, we get

$$\log(\kappa) = -\log(1 + \exp(-\eta)) = \eta - \log(1 + \exp(\eta))$$

- Another useful identity

$$\frac{\exp a}{\exp a + \exp b} = \mathrm{sig}(a - b)$$

- The inverse is the logit function:

$$\eta = \mathrm{logit}(\kappa) = \mathrm{sig}^{-1}(\kappa) = \log \frac{\kappa}{1 - \kappa} = \log(\kappa) - \log(1 - \kappa)$$

Modelling Probabilities with Logistic Regression

- Let $P(Y = 1|\eta) = \kappa = \mathsf{sig}(\eta)$; then $P(Y = 0|\eta) = (1 - \kappa) = 1 - \mathsf{sig}(\eta)$
- We can write concisely ($l \in \{0, 1\}$)

$$P(Y = l|\eta) = \kappa^l (1 - \kappa)^{1-l}$$

and the log-probability

$$\log P(Y = l) = l \log \kappa + (1 - l) \log(1 - \kappa)$$

- Another way of writing this is

$$P(Y = l|\eta) = \frac{\exp(l\eta)}{1 + \exp(\eta)} = \exp[l\eta - \log(1 + \exp(\eta))]$$

and the log-probability becomes

$$\log P(Y = l|\eta) = l\eta - \log(1 + \exp(\eta))$$

Exercise: Bayes Inference

- Assume we know $P(Y)$ and the likelihood $P(X|Y)$, what is the posterior $P(Y|X)$?
- Prior (as before): $P(Y = 1) = \mathrm{sig}(q)$

$$P(Y = l) = \exp[lq - \log(1 + \exp(q))]$$

- Likelihood: $P(X = 1|Y = 1) = \mathrm{sig}(\eta_1)$ and $P(X = 1|Y = 0) = \mathrm{sig}(\eta_0)$
- Can be written as

$$P(X = i|Y = 1) = \exp(i\eta_1 - \log(1 + \exp(\eta_1)))$$

$$P(X = i|Y = 0) = \exp(i\eta_0 - \log(1 + \exp(\eta_0)))$$

Bayes Inference (cont'd)

- Posterior:

$$P(Y = 1|X = i)$$

$$= \frac{P(Y = 1)P(X = i|Y = 1)}{P(Y = 1)P(X = i|Y = 1) + P(Y = 0)P(X = i|Y = 0)}$$

$$= \frac{\mathrm{sig}(q)\mathrm{sig}(\eta_1)^i(1 - \mathrm{sig}(\eta_1))^{1-i}}{\mathrm{sig}(q)\mathrm{sig}(\eta_1)^i(1 - \mathrm{sig}(\eta_1))^{1-i} + (1 - \mathrm{sig}(q))\mathrm{sig}(\eta_0)^i(1 - \mathrm{sig}(\eta_0))^{1-i}}$$

- This can also be written as

$$= \frac{\exp(a)}{\exp(a) + \exp(b)}$$

where

$$a = q - \log(1 + \exp(q)) + i\eta_1 - \log(1 + \exp(\eta_1))$$

$$b = -\log(1 + \exp(q)) + i\eta_0 - \log(1 + \exp(\eta_0))$$

Bayes Inference (cont'd)

- Thus,

$$P(Y = 1|X = i) = \text{sig}[i(\eta_1 - \eta_0) + q - \log(1 + \exp(\eta_1)) + \log(1 + \exp(\eta_0))]$$

$$= \text{sig}(w_0 + w_1 i)$$

where

$$w_1 = \eta_1 - \eta_0$$

$$w_0 = q - \log(1 + \exp(\eta_1)) + \log(1 + \exp(\eta_0))$$