Logistic Regression

a probabilistic and discriminative classification model

Contents

- Discriminative classification
- Logistic Regression
- Generalized Linear Models
- Training
- Multi-class Problems
- Discussion

Discriminative Classifiers

- **Discriminative classifiers:**
  - Idea: direct modelling of $p(C \mid \mathbf{x})$
  - Motivation: separating feature space into regions that represent individual classes
  - In general, this leads to simpler models and, therefore, requires fewer training samples
- Discriminant function: a function $g_i(\mathbf{x})$ that assigns $\mathbf{x}$ to a class L^i, if $g_i(\mathbf{x}) > g_j(\boldsymbol{x})$ for all $i \neq j$
- The discriminant function sub-divides the feature space into regions R_i which are assigned to the class L^i
- The boundaries of these regions are given by $g_i(\mathbf{x}) = g_j(\mathbf{x})$

Discriminative Methods: Overview

- Probabilistic discriminative classifiers: Discriminant function is based on $p(C^i \mid \mathbf{x})$
  - Logistic Regression: first designed for binary classification
  - Generalized Linear Models: extension for high-dimensional decision boundaries
- Non-probabilistic discriminative classifiers: the discrimant function cannot be interpreted as a probability e.g.
  - Decision trees
  - Random forests
  - Support vector machines
  - Artificial neural networks

Logistic Sigmoid Function

- Distinction of two classes L^1, L^2 (e.g. object and background)
- Start with Theorem of Bayes:

$$p(C{=}L^1|\mathbf{x}) = \frac{p(\mathbf{x}|C{=}L^1)\cdot p(C{=}L^1)}{p(\mathbf{x}|C{=}L^1)\cdot p(C{=}L^1) + p(\mathbf{x}|C{=}L^2)\cdot p(C{=}L^2)} =$$

$$= \frac{1}{1+\dfrac{p(\mathbf{x}|C{=}L^2)\cdot p(C{=}L^2)}{p(\mathbf{x}|C{=}L^1)\cdot p(C{=}L^1)}} = \frac{1}{1+e^{-a}} = \sigma(a)$$

with
$$a(\mathbf{x}) = \ln\frac{p(\mathbf{x}|C=L^1)\cdot p(C=L^1)}{p(\mathbf{x}|C=L^2)\cdot p(C=L^2)} = \ln\frac{p(C=L^1|\mathbf{x})}{p(C=L^2|\mathbf{x})}$$

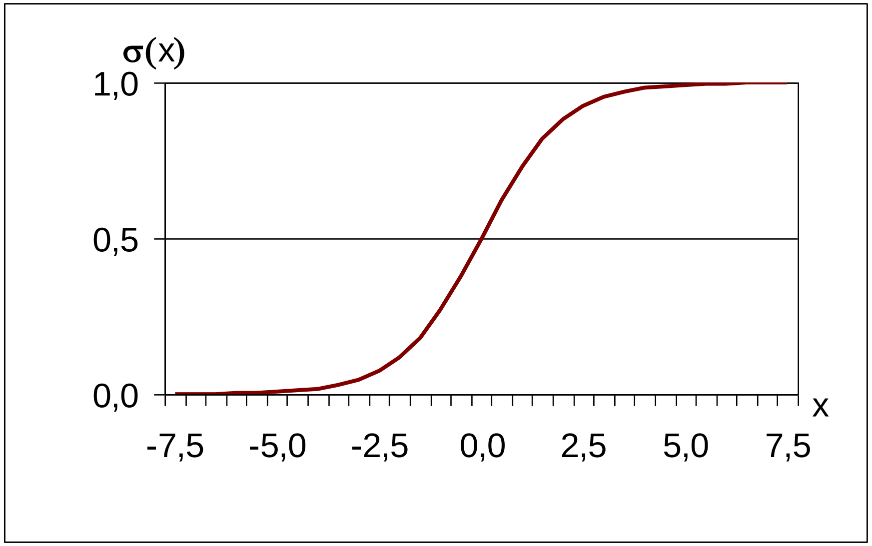
- Logistic sigmoid function

$$\sigma(a) = \frac{1}{1+e^{-a}}$$

Logistic Sigmoid Function

- Originally, this is a generative model, because it is based on the theorem of Bayes
- $a(\mathbf{x})$ is the negative logarithm of the ratio of the posterior probabilities
- From now on: consideration of $a(\mathbf{x})$ without Bayesian interpretation
- Simple models for $a(\mathbf{x})$: linear or quadratic functions
- logistic sigmoid function:

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



Logistic Regression

- (Unrealistic) assumption (but, to be able to have linear function for $a(\mathbf{x})$ later): The features of $\mathbf{x}$ are normally distributed with mean values $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$ and identical covariance matrices $\Sigma_1 = \Sigma_2 = \Sigma$

$$a(\mathbf{x}) = \ln \frac{p(\mathbf{x}|C=L^1) \cdot p(C=L^1)}{p(\mathbf{x}|C=L^2) \cdot p(C=L^2)} =$$

$$= -\tfrac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_1)^T \cdot \Sigma^{-1} \cdot (\mathbf{x} - \boldsymbol{\mu}_1) + \tfrac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_2)^T \cdot \Sigma^{-1} \cdot (\mathbf{x} - \boldsymbol{\mu}_2) + \ln p(C=L^1) - \ln p(C=L^2) =$$

$$= \underbrace{(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T \cdot \Sigma^{-1} \cdot \mathbf{x}}_{\mathbf{w}^T \cdot \mathbf{x}} \underbrace{- \tfrac{1}{2} \boldsymbol{\mu}_1^T \cdot \Sigma^{-1} \cdot \boldsymbol{\mu}_1 + \tfrac{1}{2} \boldsymbol{\mu}_2^T \cdot \Sigma^{-1} \cdot \boldsymbol{\mu}_2 + \ln p(C=L^1) - \ln p(C=L^2)}_{w_0} =$$

$$= \mathbf{w}^T \cdot \mathbf{x} + w_0$$

$$p(C=L^1|\mathbf{x}) = \frac{1}{1 + e^{-(\mathbf{w}^T \cdot \mathbf{x} + w_0)}} = \sigma(a(\mathbf{x})) = \sigma(\mathbf{w}^T \cdot \mathbf{x} + w_0)$$

- thus: → $a(\mathbf{x})$ is a linear function of the features!

Logistic Regression: Parameters

- In the binary case, we have $p(C{=}L^2|\mathbf{x}) = 1 - p(C{=}L^1|\mathbf{x})$, due to $1 - \sigma(a) = \sigma(-a)$,

$$p\left(C = L^1 \,|\, \mathbf{x}\right) = \frac{1}{1 + e^{-\left(\mathbf{w}^T \cdot \mathbf{x} + w_0\right)}} \quad \textit{and} \quad p\left(C = L^2 \,|\, \mathbf{x}\right) = \frac{1}{1 + e^{\left(\mathbf{w}^T \cdot \mathbf{x} + w_0\right)}}$$

- Class boundary in feature space: → $\frac{1}{1 + e^{-\left(\mathbf{w}^T \cdot \mathbf{x} + w_0\right)}} = \frac{1}{1 + e^{\left(\mathbf{w}^T \cdot \mathbf{x} + w_0\right)}}$

  → $-\left(\mathbf{w}^T \cdot \mathbf{x} + w_0\right) = \mathbf{w}^T \cdot \mathbf{x} + w_0$

  → $\mathbf{w}^T \cdot \mathbf{x} + w_0 = 0$ → The decision boundary between the classes is a hyperplane

- Parameters to be learned: $\mathbf{w}$, w_0

  → with D features: $D + 1$ parameters

  → The number of parameters grows linearly with D

Logistic Regression: Seperating Surface

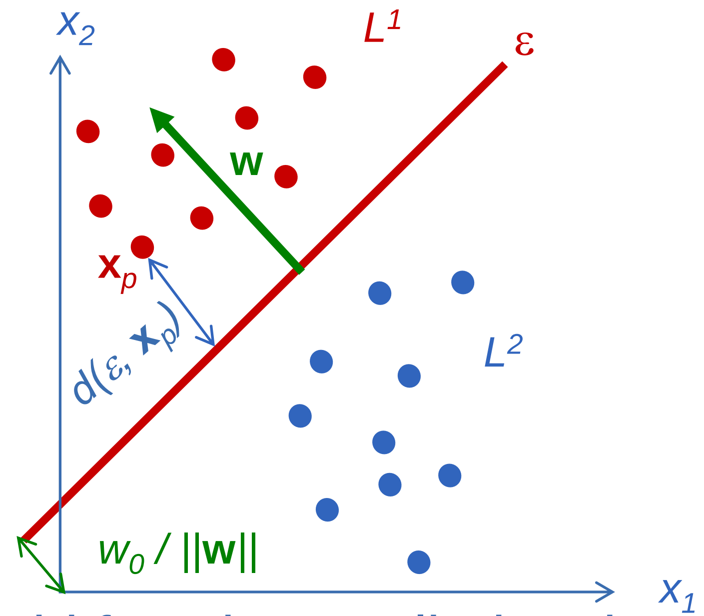
- Decision boundary in feature space: $\mathbf{w}^T \cdot \mathbf{x} + w_0 = 0$
  - Normal vector $\mathbf{w} = \mathbf{S}^{-1} \cdot (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$ depends on the vector between the class centers, direction is also influenced by **S**
  - Offset w_0:
    $w_0 = -\frac{1}{2} \boldsymbol{\mu}_1^T \cdot \mathbf{S}^{-1} \cdot \boldsymbol{\mu}_1 + \frac{1}{2} \boldsymbol{\mu}_2^T \cdot S^{-1} \cdot \boldsymbol{\mu}_2 + \ln p(C=L^1) - \ln p(C=L^2)$
  - Changes to the prior lead to a parallel shift of the decision boundary

Logistic Regression: Geometrical Interpretation

- Decision boundary in feature space: ε: $\mathbf{w}^T \cdot \mathbf{x} + w_0 = 0$
- For a point $\mathbf{x}_p$ that does not lie on the separating surface:

$$\mathbf{w}^T \cdot \mathbf{x}_p + w_0 = \| \mathbf{w} \| \cdot d(\varepsilon, \mathbf{x}_p)$$

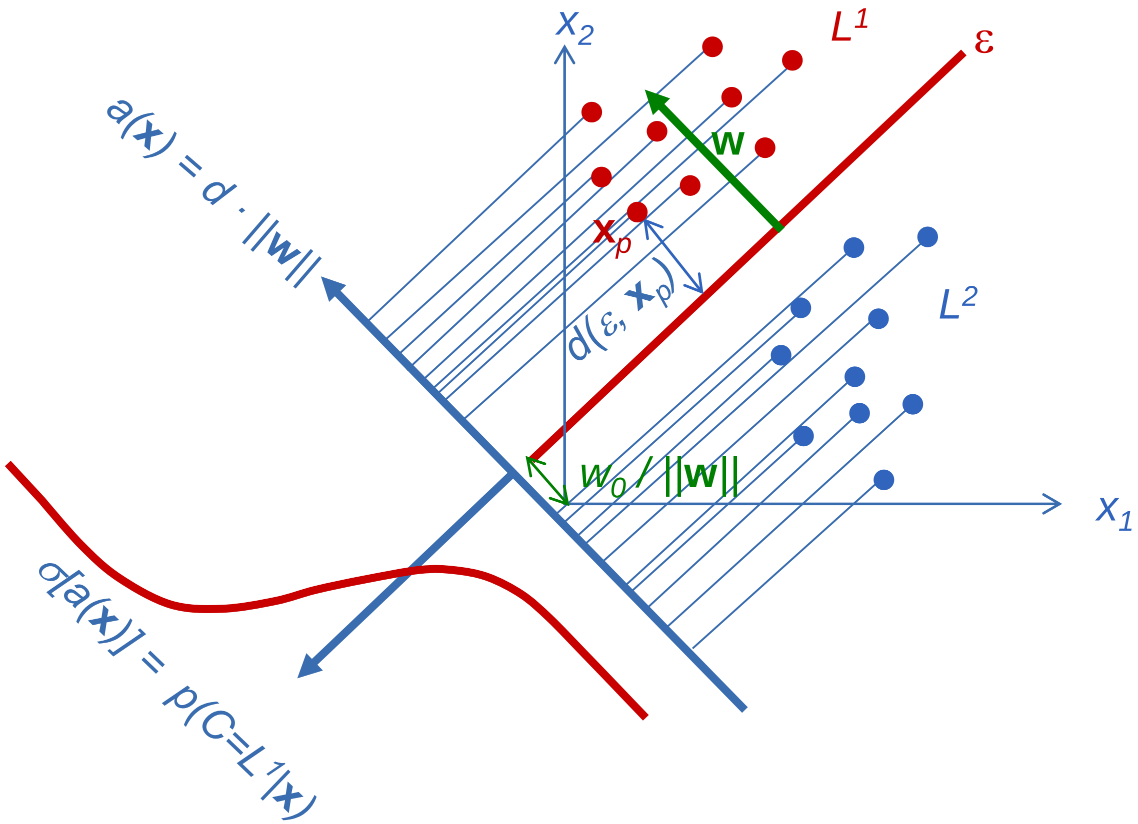
$$p\left(C = L^1 \mid \mathbf{x}\right) = \frac{1}{1 + e^{-\left(\mathbf{w}^T \cdot \mathbf{x} + w_0\right)}}$$



- Interpretation of probability:as a sigmoid function applied to the (scaled) distance from the separating surface that maps this distance into the interval [0,1]!

Logistic Regression: Geometrical Interpretation

- Interpretation of || **w** ||: The larger || **w** ||, the steeper the sigmoid function



Notion: “Logistic Regression“

- "Regression": search for an optimal linear separating surface in feature space
- "logistic": Basis is the logistic sigmoid function
- The principle that the sigmoid function is applied to a scaled distance to get a probability is often used in other contexts
- What happens with data that are not linearly separable?

Generative Model: Normal Distribution with different Covariance Matrices

- In general, the class boundary is not a hyperplane but a **hyperquadric**
- New assumption for features distribution: the covariance matrices are not identical, the quadratic term in the exponent does not disappear:

$$p(C=L^1|\mathbf{x}) = \frac{1}{1 + e^{-(\mathbf{x}^T \cdot \mathbf{W} \cdot \mathbf{x} + \mathbf{w}^T \cdot \mathbf{x} + w_0)}}$$

with $\mathbf{W} = \frac{1}{2} \cdot (\mathbf{S}_2^{-1} - \mathbf{S}_1^{-1})$

$\mathbf{w} = \mathbf{S}_1^{-1} \cdot \boldsymbol{\mu}_1 - \mathbf{S}_2^{-1} \cdot \boldsymbol{\mu}_2$

$w_0 = \frac{1}{2} \cdot \boldsymbol{\mu}_2^T \cdot \mathbf{S}_2^{-1} \cdot \boldsymbol{\mu}_2 - \frac{1}{2} \cdot \boldsymbol{\mu}_1^T \cdot \mathbf{S}_1^{-1} \cdot \boldsymbol{\mu}_1 +$
$+ \frac{1}{2} \cdot \ln \| \mathbf{S}_2 \| - \frac{1}{2} \cdot \ln \| \mathbf{S}_1 \| + \ln p(C=L^1) - \ln p(C=L^2)$

- With increasing complexity of the models for the probability densities: a quadratic form for normal distributions
- In order to be able to work with linear medels: Transformation of the feature space (Feature Space Mapping)

Feature Space Transformations and Generalized Linear Models

- **Feature Space Mapping $\Phi(\mathbf{x})$** $= [\mathit{\Phi}_1(\mathbf{x}), \mathit{\Phi}_2(\mathbf{x}), \ldots, \mathit{\Phi}_N(\mathbf{x})]^T$
  - $\mathit{\Phi}_i(\mathbf{x})$: (in principle) arbitrary functions: frequently, polynomials
  - N: Dimension of the transformed feature vector (usually greater than the dimension of $\mathbf{x}$)
  - Frequent choice: $\mathit{\Phi}_1(\mathbf{x}) = 1$
  - Example for 2D feature space, i.e. $\mathbf{x} = (x_1, x_2)^T$:

$$\Phi(\mathbf{x}) = (1, x_1, x_2, x_1 \cdot x_2, x_1^2, x_2^2)^T$$

- Instead of using a complex model for $a(\mathbf{x})$: Transition into a higher dimensional feature space in which $a(\Phi(\mathbf{x}))$ is linear
  $\Rightarrow$ **Generalized Linear Models**

Feature Space Transformations and Generalized Linear Models

- **Generalized Linear Models:**

$$p(C=L^1|\mathbf{x}) = \sigma[a(\mathbf{x})] = \frac{1}{1 + e^{-a(\mathbf{x})}}$$

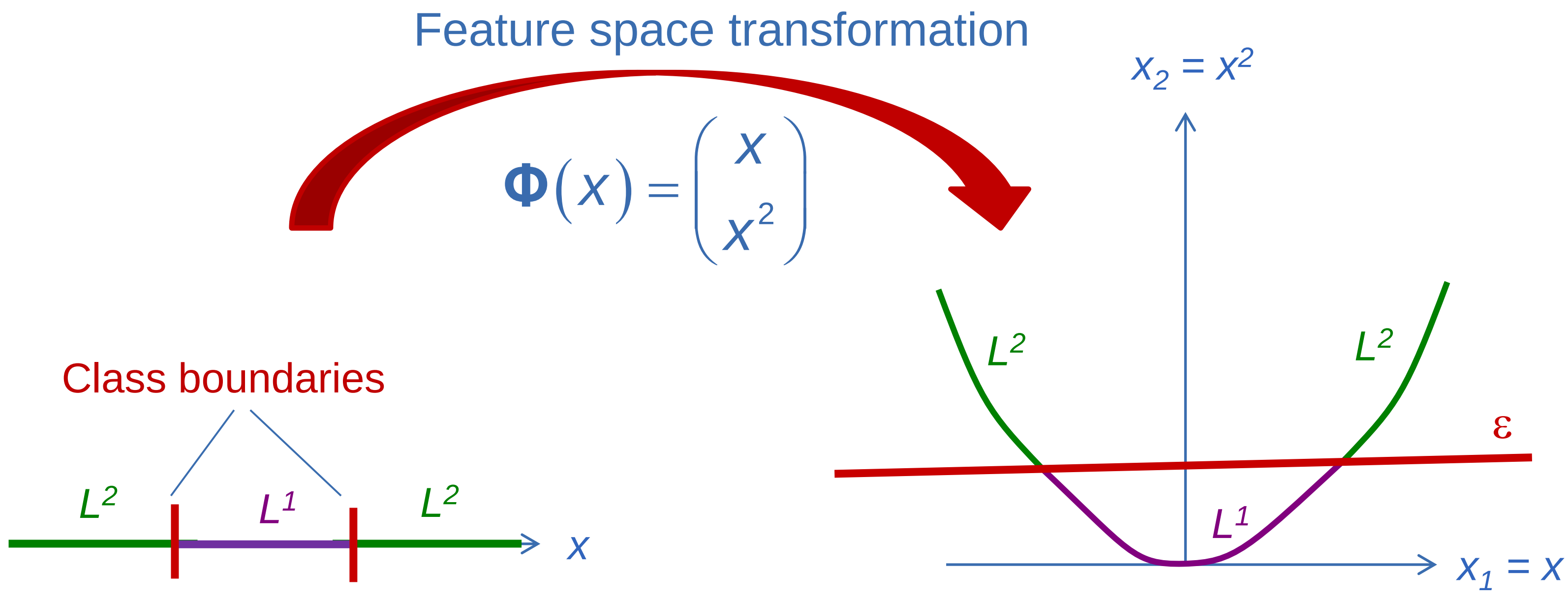
with $a(\mathbf{x}) = \mathbf{w}^T \cdot \mathbf{\Phi}(\mathbf{x})$

and $\mathbf{\Phi}(\mathbf{x}) = [\Phi_1(\mathbf{x}), \Phi_2(\mathbf{x}), \ldots, \Phi_N(\mathbf{x})]^T$

- Note: Due to $\Phi_1(\mathbf{x}) = 1$, w_0 becomes the first component of $\mathbf{w}$
- The example of $\mathbf{\Phi}(\mathbf{x}) = (1, x_1, x_2, x_1 \cdot x_2, x_1^2, x_2^2)^T$ leads to a quadratic form for $a(\mathbf{x})$ similar to the normal distribution!
- Assumptions about the distribution of the features are dropped in favour of a choice of a feature space mapping
- Choices: Quadratic expansion, Cubic expansion, Kernel logistic regression

Examples of Feature Space Mappings I

- Transition to a higher-dimensional feature vector $\mathbf{\Phi}(\mathbf{x})$
- Example:



1D feature space, 2 classes

Not linearly separable

After feature space transformation:

2D feature space (x, x^2)

Classes can be separated by a plane ε

Feature Space Mapping

- Using a feature space mapping, linear models can also be applied to problems where the classes are not linearly separable
- **Disadvantage:** Increase of the number N of parameters:
  - Polynomial expansion: with D features (incl. $\Phi_1(\mathbf{x}) = 1$), order G:

$$N = \binom{D+G-1}{G}$$

    - $G = 2 \rightarrow N = D \cdot (D+1) / 2$
    - $G = 3 \rightarrow N = D \cdot (D+1) \cdot (D+2) / 6$
  - Kernel Function: N is equal to the number of training points
  - Could be problematic for feature spaces with $D > 10$

Logistic Regression: Training

- **Given:**
  - Functional model of feature space mapping
  - N points $\mathbf{x}_i$ with known $t_i \in \{0,1\}$
  - t_i: **indicator variable** that shows if $\mathbf{x}_i$ belongs to L^1 ($t_i = 1$) or not ($t_i = 0$)
  - All the indicator variables t_i can be collected in a vector $\mathbf{t}$
- **Wanted :**
  - Parameter vector **w** of the generalized linear model

$$p\left(C = L^1 \mid \mathbf{x}\right) = \frac{1}{1 + e^{-\left[\mathbf{w}^T \cdot \mathbf{\Phi}(\mathbf{x})\right]}}$$

Logistic Regression: Maximum Likelihood Training

- Determine **w** in such that $p(\mathbf{t} \mid \mathbf{w}, \mathbf{x}_1, \ldots \mathbf{x}_N) \rightarrow \max$

with

$$y_n = p\left(C = L^1 \mid \mathbf{x}_n\right) = \frac{1}{1 + e^{-\left[\mathbf{w}^T \cdot \mathbf{\Phi}(\mathbf{x}_n)\right]}} \quad and \quad p\left(C = L^2 \mid \mathbf{x}_n\right) = 1 - y_n$$

- Result $$p\left(\mathbf{t} \mid \mathbf{w}, \mathbf{x}_1, \ldots, \mathbf{x}_N\right) = \prod_{n=1}^{N} y_n^{t_n} \cdot \left(1 - y_n\right)^{(1-t_n)}$$
  - for $t_n = 1$: y_n will contribute
  - for $t_n = 0$: $(1 - y_n)$ will contribute
- Instead of the maximization of $p(\mathbf{t} \mid \mathbf{w}, \mathbf{x}_1, \ldots \mathbf{x}_N)$:

  Minimization of the negative log-likelihood

$$E(\mathbf{w}) = -\ln p(\mathbf{t} \mid \mathbf{w}, \mathbf{x}_1, \ldots \mathbf{x}_N) \rightarrow \min$$

Logistic Regression: Maximum Likelihood Training

- Negative log-Likelihood E(**w**):

$$E(\mathbf{w}) = -\sum_{n=1}^{N} \left[t_n \cdot \ln(y_n) + (1 - t_n) \cdot \ln(1 - y_n) \right] \rightarrow \min$$

- As y_n depends on **w**, E(**w**) is a non-linear function of **w**
- Therefore, the minimum of E(**w**) can only be determined iteratively
- Initial values $\mathbf{w}^0$: e.g. random numbers
- E(**w**) is concave and has a single minimum
- Determination of the minimum: gradient $\nabla E(\mathbf{w}) = 0$
- Newton-Raphson method(find path to the minimum): using the initial values $\mathbf{w}^{\tau-1}$: $\mathbf{w}^{\tau} = \mathbf{w}^{\tau-1} - \mathbf{H}^{-1} \cdot \nabla E\left(\mathbf{w}^{\tau-1}\right)$

Logistic Regression: Maximum Likelihood Training

- Gradient $\nabla E(\mathbf{w})$: $$\nabla E(\mathbf{w}) = \sum_{n=1}^{N} (y_n - t_n) \cdot \boldsymbol{\Phi}(\mathbf{x}_n)$$
  - Interpretation: $(y_n - t_n)$ can be interpreted as classification error for the training point $\mathbf{x}_n$:

    → If $t_n = 1$ → $C = L^1$ → $y_n = p(C^1 \mid \mathbf{x}_n)$ should be close to 1

    → If $t_n = 0$ → $C = L^2$ → y_n should be close to 0

  - $\nabla E(\mathbf{w})$: sum of the feature vectors weighted by $(y_n - t_n)$
- **Hesse Matrix** $$\mathbf{H} = \nabla\nabla E(\mathbf{w}) = \sum_{n=1}^{N} y_n \cdot (1 - y_n) \cdot \boldsymbol{\Phi}(\mathbf{x}_n) \cdot \boldsymbol{\Phi}(\mathbf{x}_n)^T$$
- Hesse-Matrix is positive definite → inverse exists

Logistic Regression: Maximum Likelihood Training

- In order to avoid numerical problems:

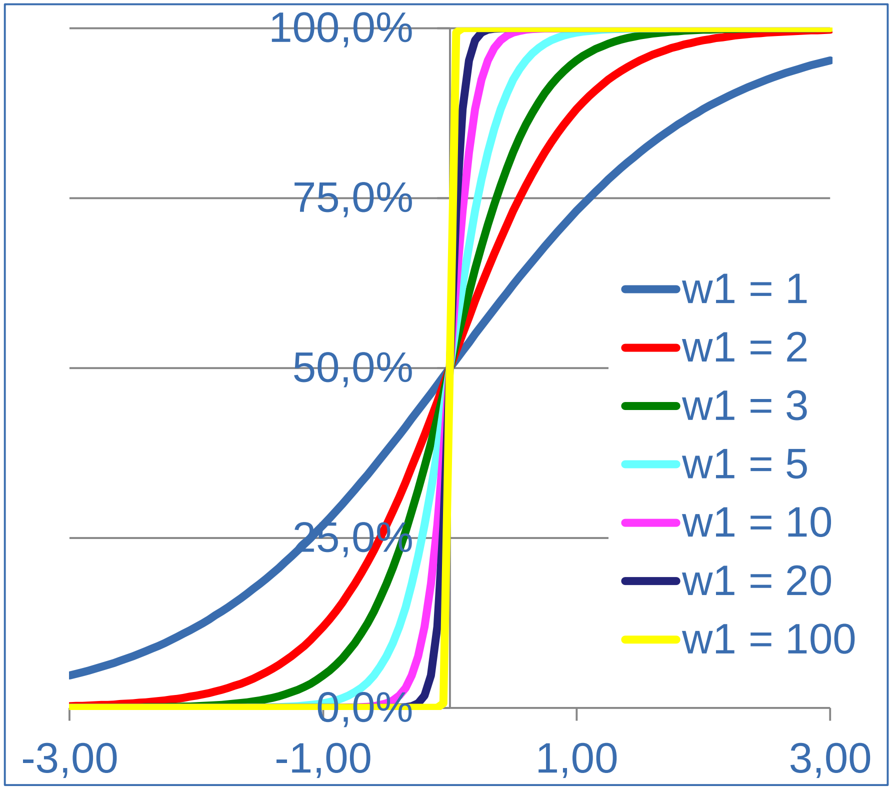
  → Scaling of the features :

  - Shift by mean value μ, scaling with standard deviation $1 / \sigma$
    → Features all have the same range of values
  - The same scaling has to be applied for training and classification!

- ML has the tendency to overfit the classifier to the training data: classifier memorizes the training samples and isn't generalizing to unseen data→ regularisation of parameters using prior for **w**
- MAP: Maximization of $p(\mathbf{w} \mid \mathbf{t}, \mathbf{x}_1, \ldots \mathbf{x}_N) \propto p(\mathbf{t} \mid \mathbf{w}, \mathbf{x}_1, \ldots \mathbf{x}_N) \cdot p(\mathbf{w})$
- $p(\mathbf{t} \mid \mathbf{w}, \mathbf{x}_1, \ldots \mathbf{x}_N)$ Corresponds to the Likelihood (as with ML)

Logistic Regression: Training with Regularization

- Prior $p(\mathbf{w})$:
  - Sigmoid slope depends on the size of the numerical values of the coefficients w_i in $\mathbf{w}$:
    - The larger $|w_i|$, the steeper the sigmoid function
    - The steeper the sigmoid function, the less smooth the transition
    - For $w_i \rightarrow \infty$ the sigmoid function becomes a step function



Logistic Regression: Training with Regularization

- To keep the numerical values of **w** small:
- Prior $p(\mathbf{w})$: Normal distribution with expectation value **0** and Covariance Matrix $\sigma^2 \cdot \mathbf{I}$
- Corresponds to regularization in adjustment theory
- Requires hyper-parameter σ which is either fixed by the user or determined via a procedure such as cross-validation
- Negative logarithm (excluding constant terms):

$$E(\mathbf{w}) = -\sum_{n=1}^{N}\left[t_n \cdot \ln(y_n) + (1-t_n)\cdot \ln(1-y_n)\right] + \frac{\mathbf{w}^T \cdot \mathbf{w}}{2\cdot\sigma^2} \rightarrow \min$$

- Leads to the numerical values of **w** that are as small as possible

Logistic Regression: Training with Regularization

- Gradient has to be extended compared to the ML method:

$$E(\mathbf{w}) = -\sum_{n=1}^{N}\left[t_n \cdot \ln(y_n) + (1-t_n)\cdot \ln(1-y_n)\right] + \frac{\mathbf{w}^T \cdot \mathbf{w}}{2\cdot\sigma^2} \rightarrow \min$$

- This is also true for the Hesse Matrix:

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N}(y_n - t_n)\cdot \mathbf{\Phi}(\mathbf{x}_n) + \frac{1}{\sigma^2}\cdot \mathbf{w}$$

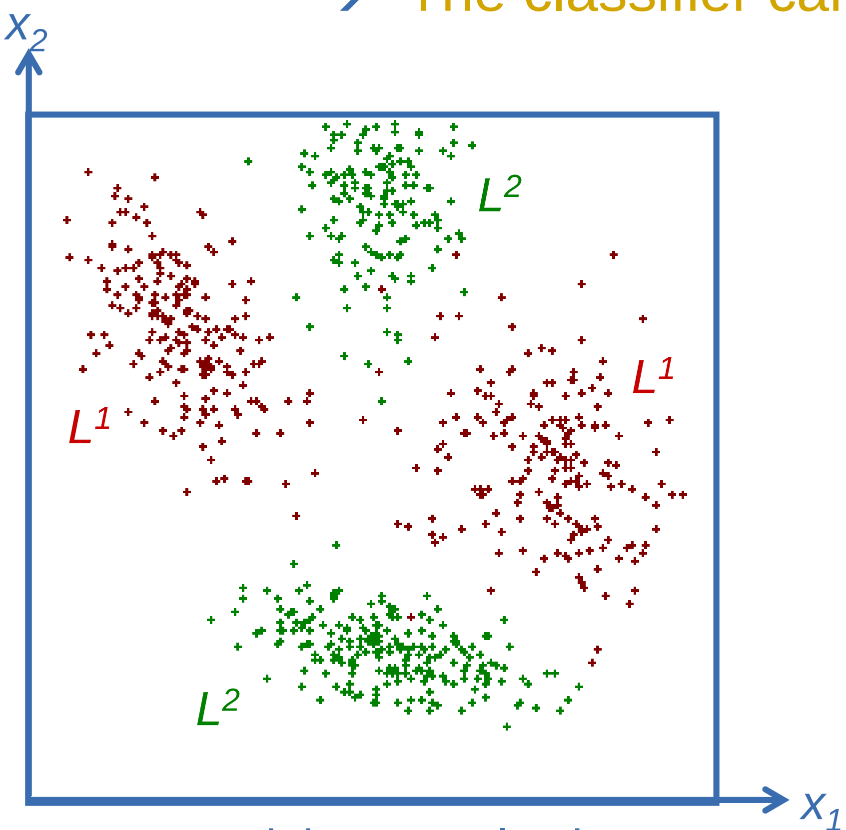
i.e. in the main diagonal, the weights of the direct observations for **w** are added (as in the case of regularization in adjustment)

$$\mathbf{H} = \nabla\nabla E(\mathbf{w}) = \sum_{n=1}^{N}\left[y_n \cdot (1-y_n)\cdot \mathbf{\Phi}(\mathbf{x}_n)\cdot \mathbf{\Phi}(\mathbf{x}_n)^T\right] + \frac{1}{\sigma^2}\cdot \mathbf{I}$$

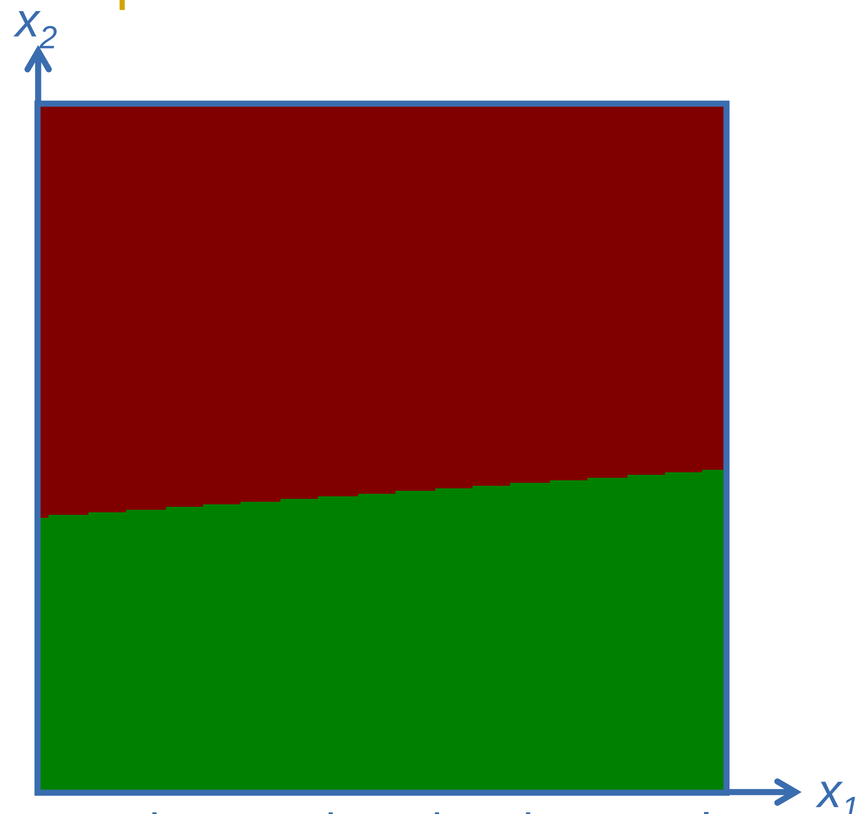
Logistic Regression: Example

Two classes, two features: non linearly separable case

→ The classifier cannot seperate the classes!



Training samples in feature space (800)



Regions assigned to the two classes in feature space

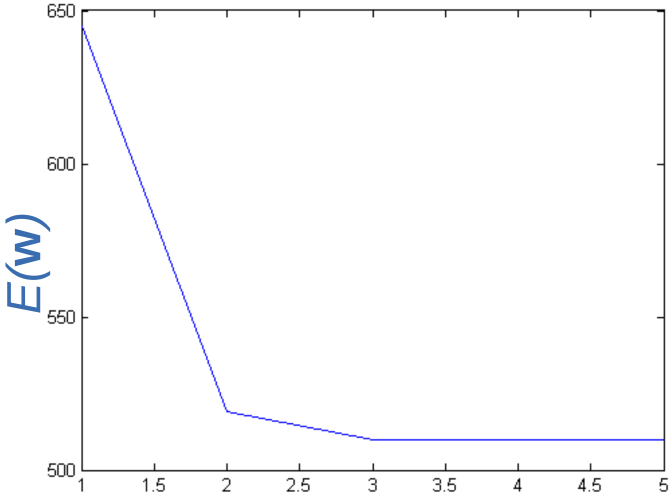
Logistic Regression: Example

Two classes, two features: non-linearly separable case

$p(C=L^1|x_1,x_2)$

$p(C=L^2|x_1,x_2)$

white ... high probability
black ... low probability



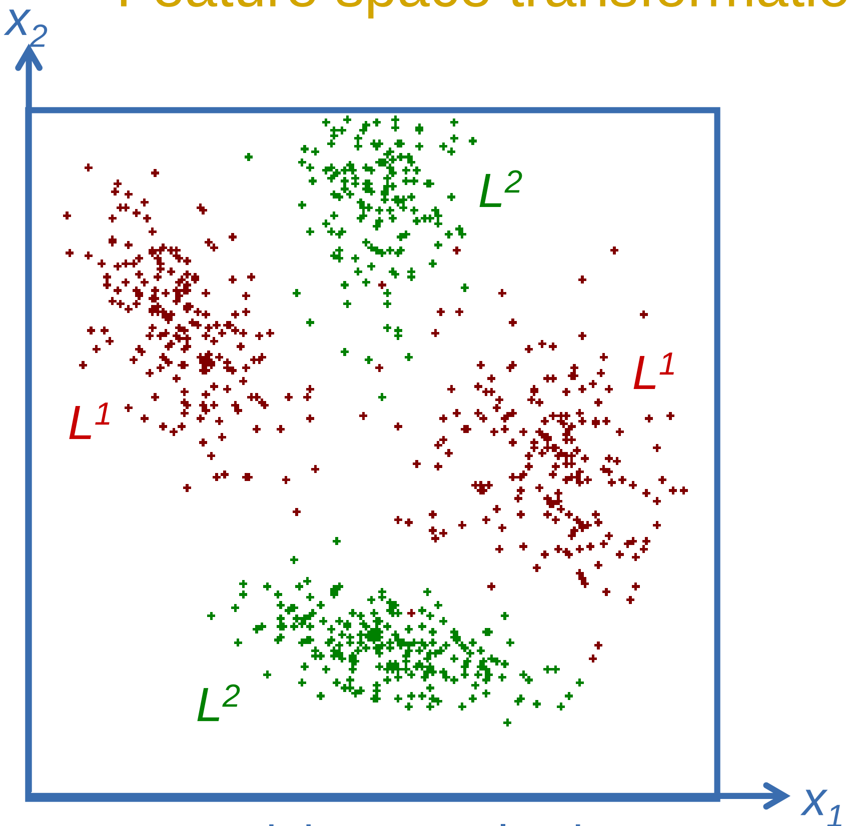
log-likelihood as a function of the iteration count in training

- Small differences in the posterior probabilities
- Relatively large value for $E(\mathbf{w})$

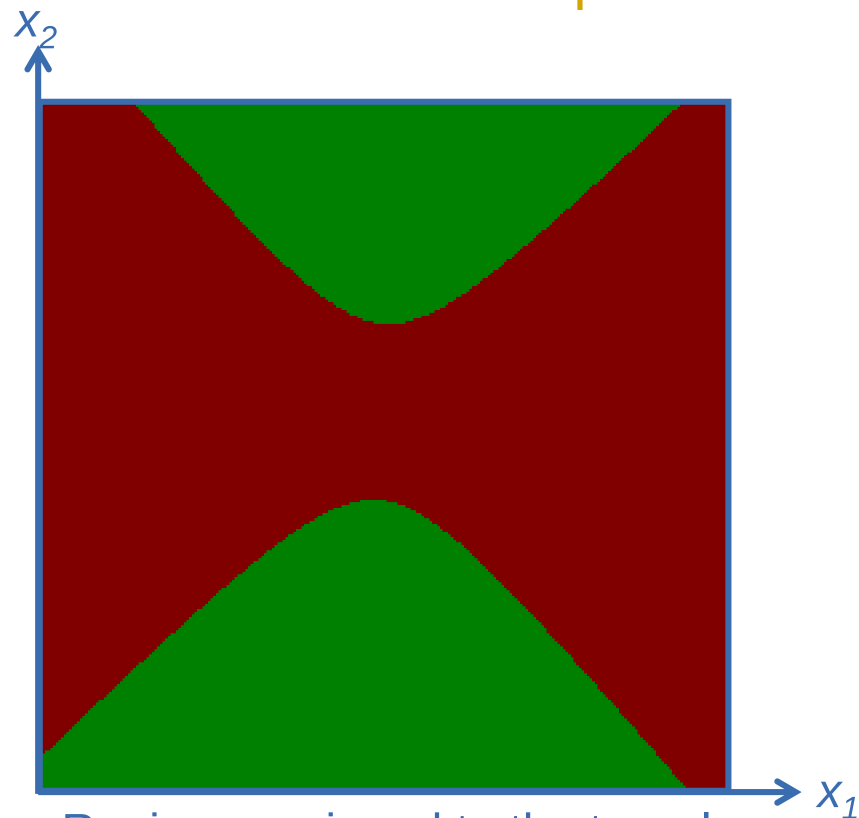
Logistic Regression: Example

Two classes, two features: non-linearly separable case

Feature space transformation: the classes can be separated



Training samples in feature space (800)



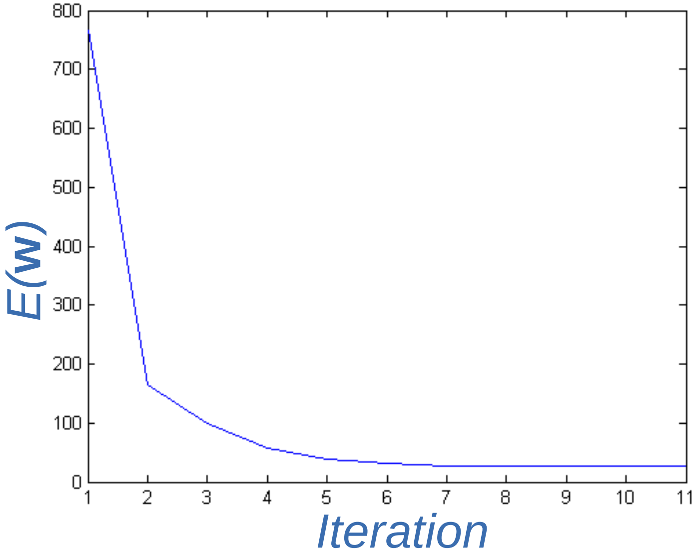
Regions assigned to the two classes in feature space

Logistic Regression: Example

Two classes, two features: non-linear separated case with characteristic spatial transformation

$p(C=L^1|x_1,x_2)$

$p(C=L^2|x_1,x_2)$



log-likelihood as a function of the iteration count in training

white ... high probability,
black ... low probability

- Significant differences in the posterior probabilities
- Low value for $E(\mathbf{w})$ is reached

Transition to Multi-class Problems

- The posterior probability $p(C=L^k \mid \mathbf{x})$ for each class L^k can be modelled using the **softmax function**:

$$p(C=L^k|\mathbf{x}) = \frac{\exp\left[a_k(\mathbf{x})\right]}{\sum_j \exp\left[a_j(\mathbf{x})\right]}$$

with $a_k(\mathbf{x}) = \ln\left[p(\mathbf{x} \mid C=L^k)\right] + \ln\left[p(C=L^k)\right]$

- Assumptions about $p(\mathbf{x} \mid C=L^k)$ and $p(C=L^k)$ lead to models for $a_k(\mathbf{x})$
- Again, feature space mapping can help to obtain linear models: $a_k(\mathbf{x}) = a_k(\Phi(\mathbf{x})) = \mathbf{w}_k^T \cdot \Phi(\mathbf{x})$
- In training, one parameter vector $\mathbf{w}_k$ per class has to be determined
- **Softmax function**: $p\left(C = L^k \mid \mathbf{x}_n\right) = \frac{\exp\left[\mathbf{w}_k^T \cdot \mathbf{\Phi}\left(\mathbf{x}_n\right)\right]}{\sum_{j=1}^{M} \exp\left[\mathbf{w}_j^T \cdot \mathbf{\Phi}\left(\mathbf{x}_n\right)\right]} = y_{nk}$

Multi-class Logistic Regression: Training

- Training: class label C_n is given for each training point $\mathbf{x}_n$
- Maximum Likelihood training is similar to the two-class case: the negative log-likelihood has to be minimized:

$$E\left(\mathbf{w}_1, \ldots \mathbf{w}_M\right) = -\sum_{n=1}^{N}\sum_{k=1}^{M} t_{nk} \cdot \ln\left(y_{nk}\right) \rightarrow \min$$

with the binary indicator variables

M… number of classes

$$t_{nk} = \begin{cases} 1 & if \quad C_n = L^k \\ 0 & otherwise \end{cases}$$

- Again, the the Newton-Raphson can be applies: Using the current values $\mathbf{w}^{\tau-1}$ from the previous iteration, the weights are updated according to

$$\mathbf{w}^{\tau} = \mathbf{w}^{\tau-1} - \mathbf{H}^{-1} \cdot \nabla E\left(\mathbf{w}^{\tau-1}\right)$$

Multi-class Logistic Regression: Maximum Likelihood Training

- The parameter vectors are not independent

  → One parameter vector must be declared to be constant, e.g. $\mathbf{w}_1^T = (0, \dots 0)^T$

- $\mathbf{w}_1$ is not changed in the optimization procedure
  → The parameter vector **w** to be determined if *M* classes are to be discerned becomes: $\mathbf{w} = (\mathbf{w}_2^T, \dots, \mathbf{w}_M^T)^T$

- Gradient of the negative log-likelihood
  (Derivative of *E* by the weight vector of the class *j*):

$$\nabla_{\mathbf{w}_j} E(\mathbf{w}_1, \dots \mathbf{w}_M) = \sum_{n=1}^{N} (y_{nj} - t_{nj}) \cdot \mathbf{\Phi}(\mathbf{x}_n)$$

- Total gradient vector :

$$\nabla E(\mathbf{w}_1, \dots \mathbf{w}_M) = \left[\nabla_{\mathbf{w}_2} E(\mathbf{w}_1, \dots \mathbf{w}_M)^T, \dots, \nabla_{\mathbf{w}_M} E(\mathbf{w}_1, \dots \mathbf{w}_M)^T \right]^T$$

Multi-class Logistic Regression: Maximum Likelihood Training

- Again, the gradient can be interpreted as the sum of the (transformed) feature vectors weighted by the "classification error" $(y_{nj} - t_{nj})$
- Hesse matrix H also consists of several components :

$$\mathbf{H} = \begin{pmatrix} \mathbf{H}_{22} & \mathbf{H}_{23} & \cdots & \mathbf{H}_{2M} \\ \mathbf{H}_{23}^T & \mathbf{H}_{33} & \cdots & \mathbf{H}_{3M} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{H}_{2M}^T & \mathbf{H}_{3M}^T & \cdots & \mathbf{H}_{MM} \end{pmatrix}$$

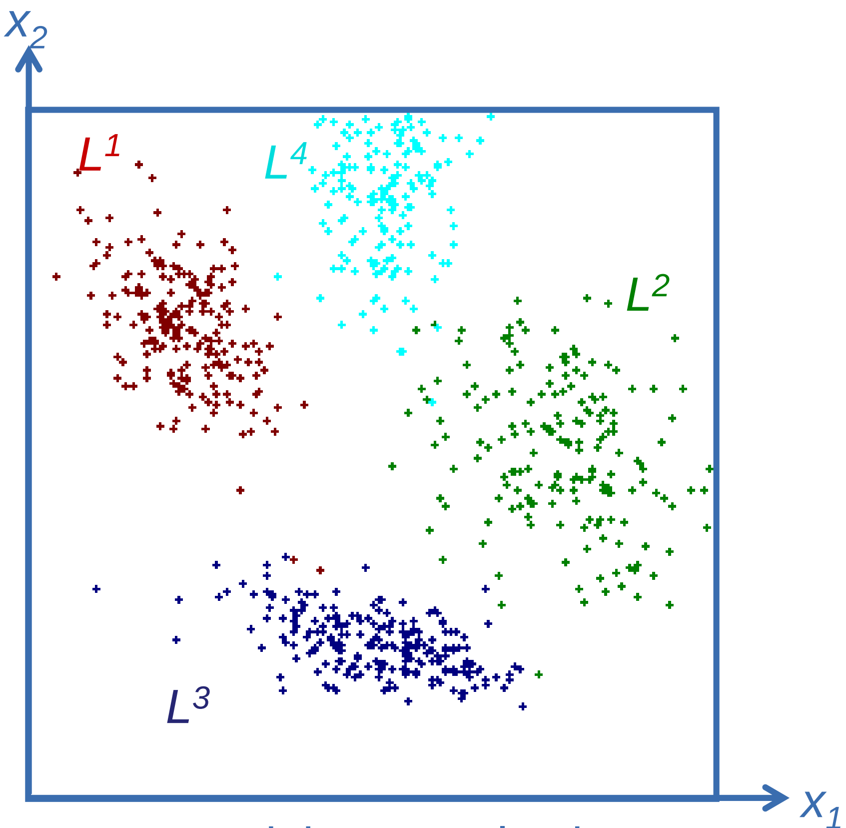
$$\mathbf{H}_{jk} = \nabla_{\mathbf{w}_j} \nabla_{\mathbf{w}_k} E(\mathbf{w}) = \sum_{n=1}^{N} y_{nk} \cdot \left(\mathbf{I}_{kj} - y_{nj}\right) \cdot \mathbf{\Phi}(\mathbf{x}_n) \cdot \mathbf{\Phi}(\mathbf{x}_n)^T$$

$\mathbf{I}_{kj}$ … Elements of a unit matrix

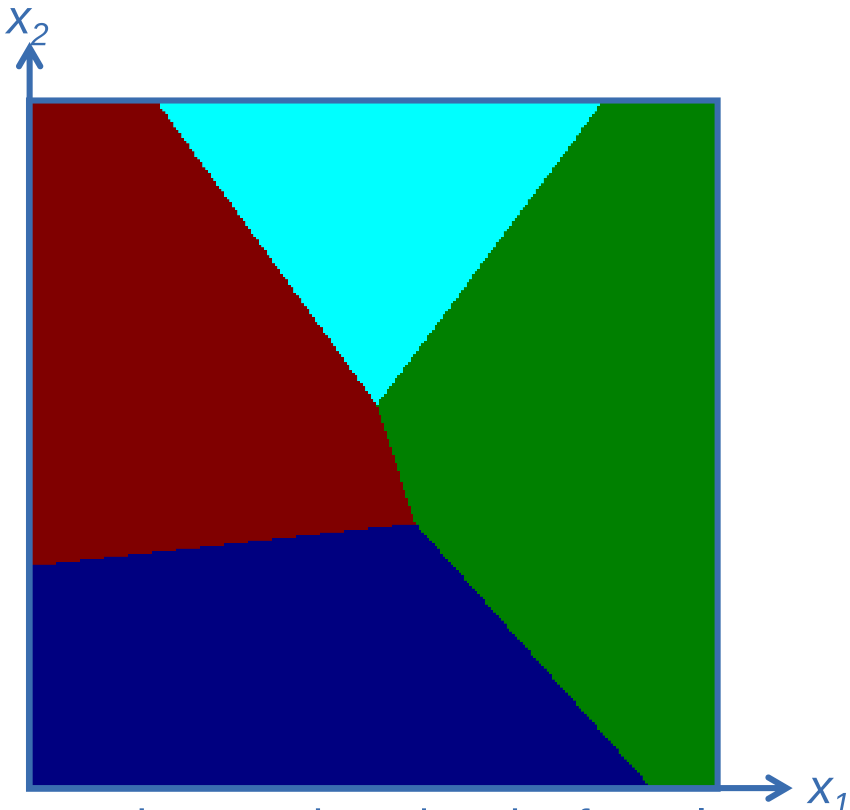
- **Regularisation**: As in the binary case (Gaussian prior with expectation **0** and Covariance $\sigma \cdot \mathbf{I}$)

Multi-class Case: Example (ML-Training)

Four classes, two features



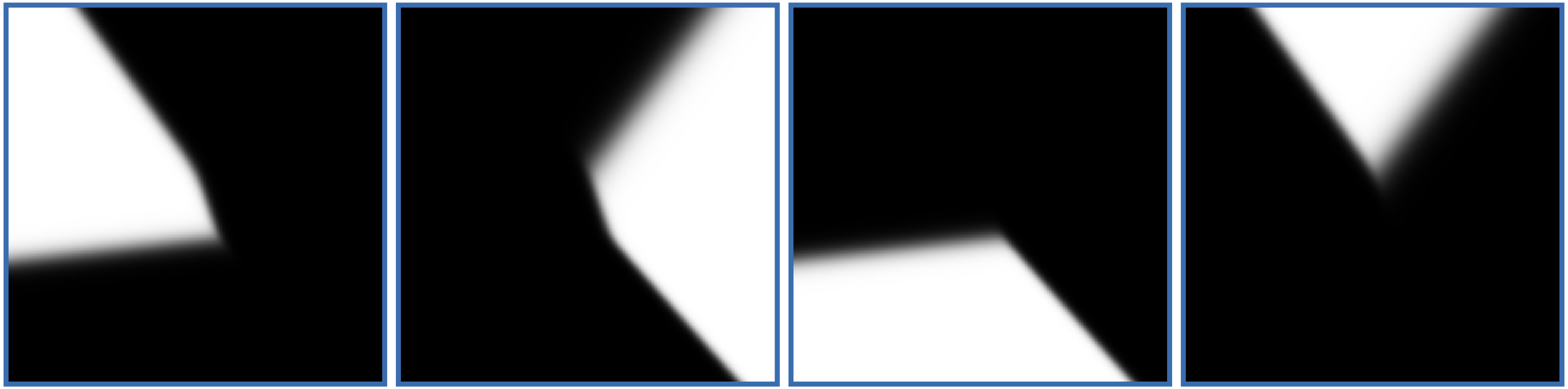
Training samples in feature space (800)



Regions assigned to the four classes in feature space

Multi-class Case: Example (ML-Training)

Four classes, two features: posterior probabilities



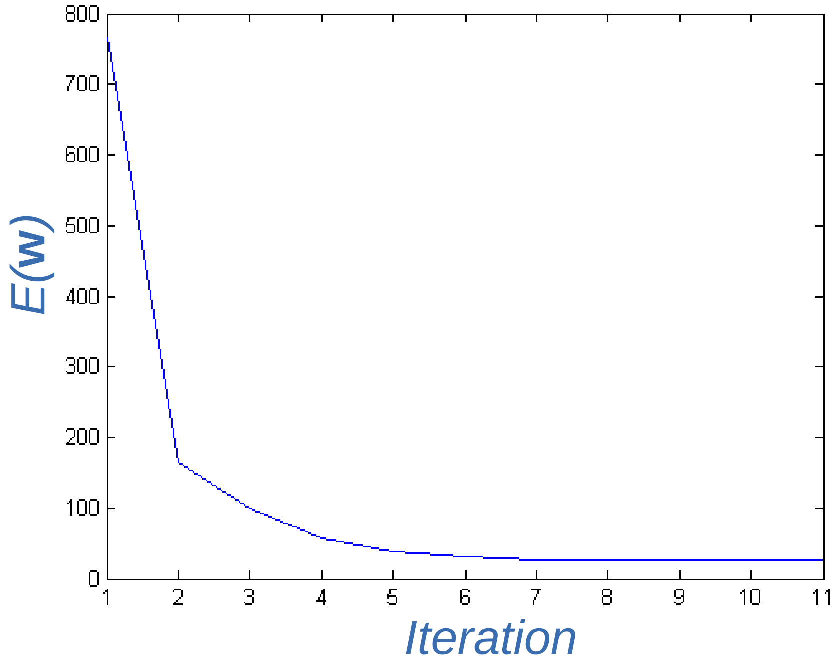
$p(C=L^1|x_1,x_2)$ $p(C=L^2|x_1,x_2)$ $p(C=L^3|x_1,x_2)$ $p(C=L^4|x_1,x_2)$

white... high probability, black ... low probability

In the areas where the feature distributions overlap, the boundaries are slightly blurred

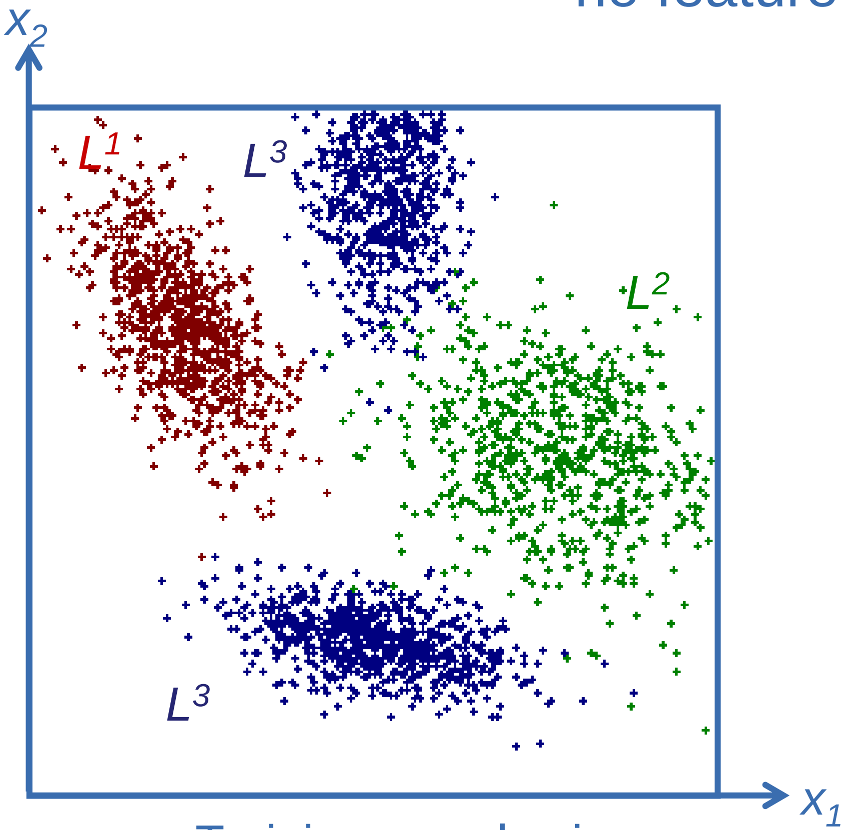
Multi-class Case: Example (ML-Training)

Four classes, two features:
Development of the log-likelihood during training

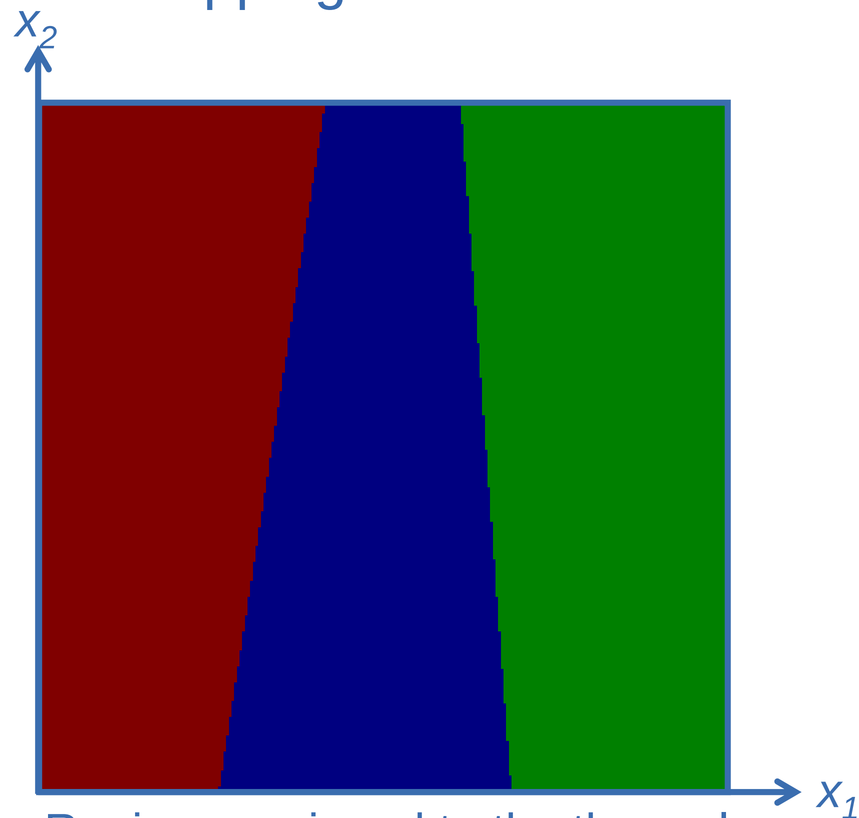


Multi-class Case: Example (ML-Training)

Three classes, two features, not linearly separable
no feature space mapping



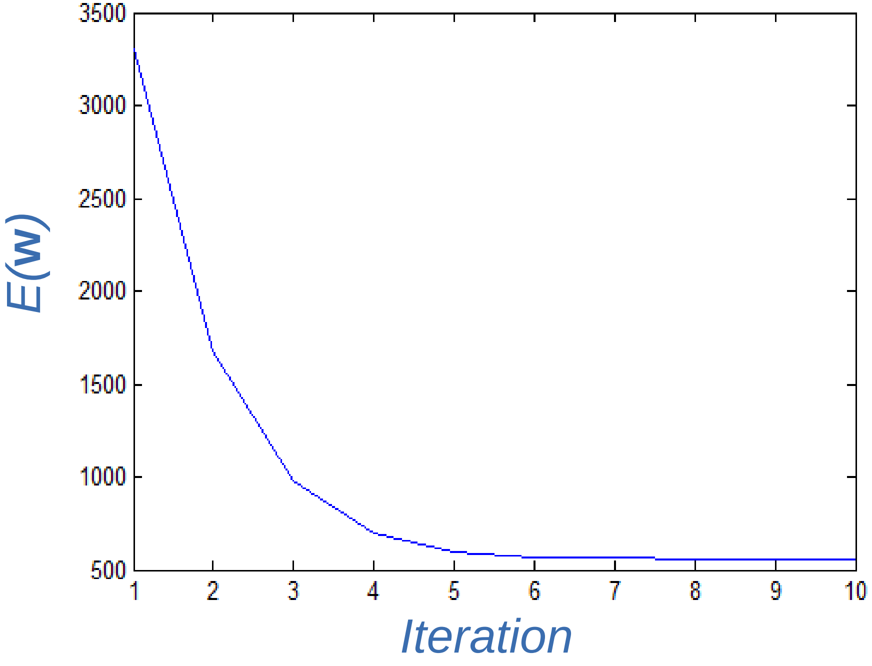
Training samples in feature space (800)



Regions assigned to the three classes in feature space

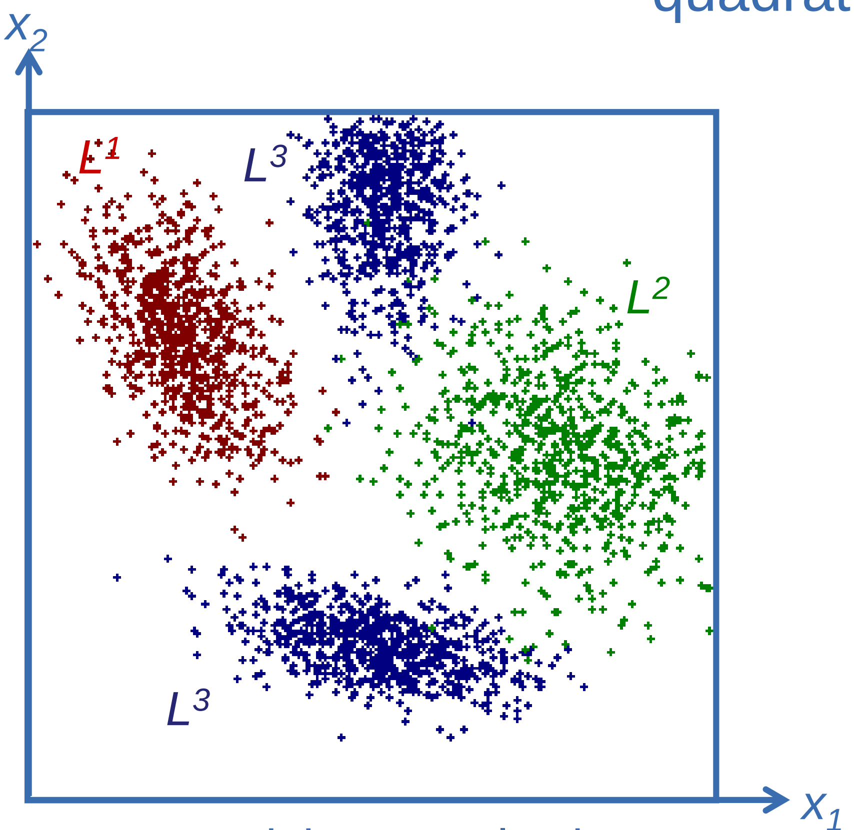
Multi-class Case: Example (ML-Training)

Three classes, two features, not linearly separable, no feature space mapping: development of log-likelihood during training

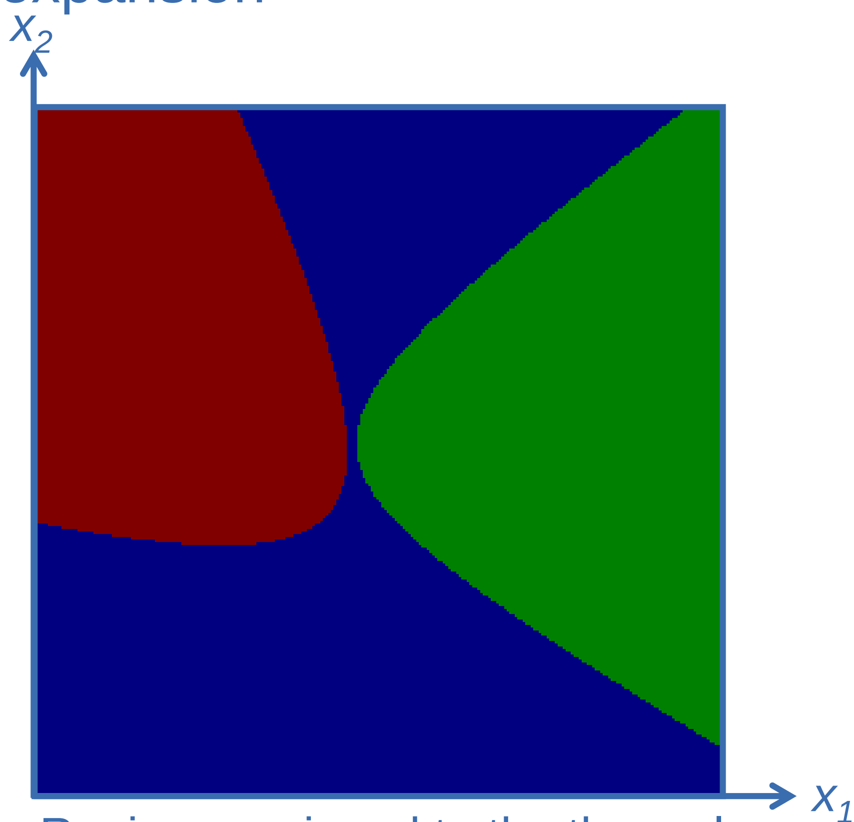


Multi-class Case: Example (ML-Training)

Three classes, two features, not linearly separable – quadratic expansion



Training samples in feature space (800)



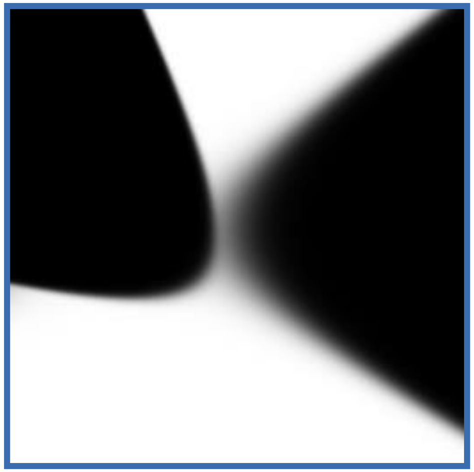
Regions assigned to the three classes in feature space

Multi-class Case: Example (ML-Training)

Three classes, two features, not linearly separable – quadratic expansion: posterior probabilities

$p(C=L^1|x_1,x_2)$

$p(C=L^2|x_1,x_2)$



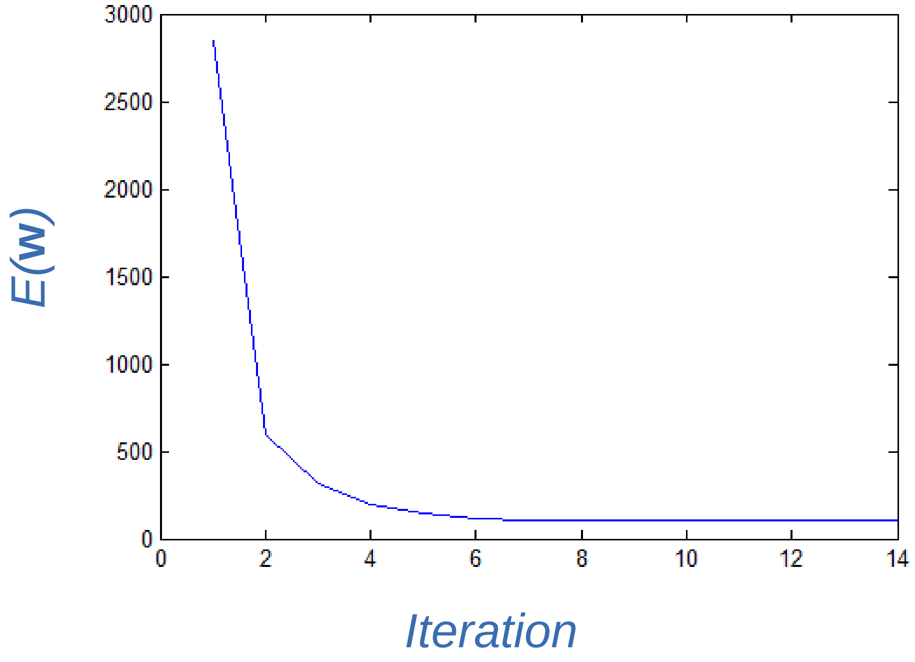
$p(C=L^3|x_1,x_2)$

white ... high probabilitiy, black ... low probability

- In the areas where the feature distributions overlap, the boundaries are slightly blurred
- However, in general there is a very clear distinction → Overfitting

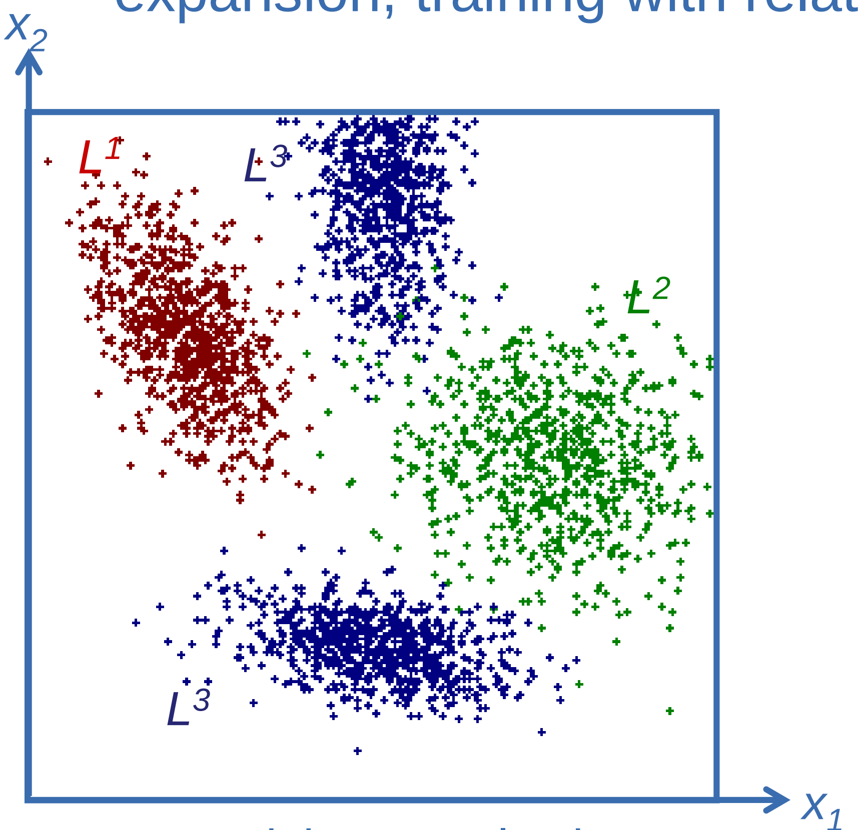
Multi-class Case: Example (ML-Training)

Three classes, two features, not linearly separable –
quadratic expansion: development of log-likelihood during training

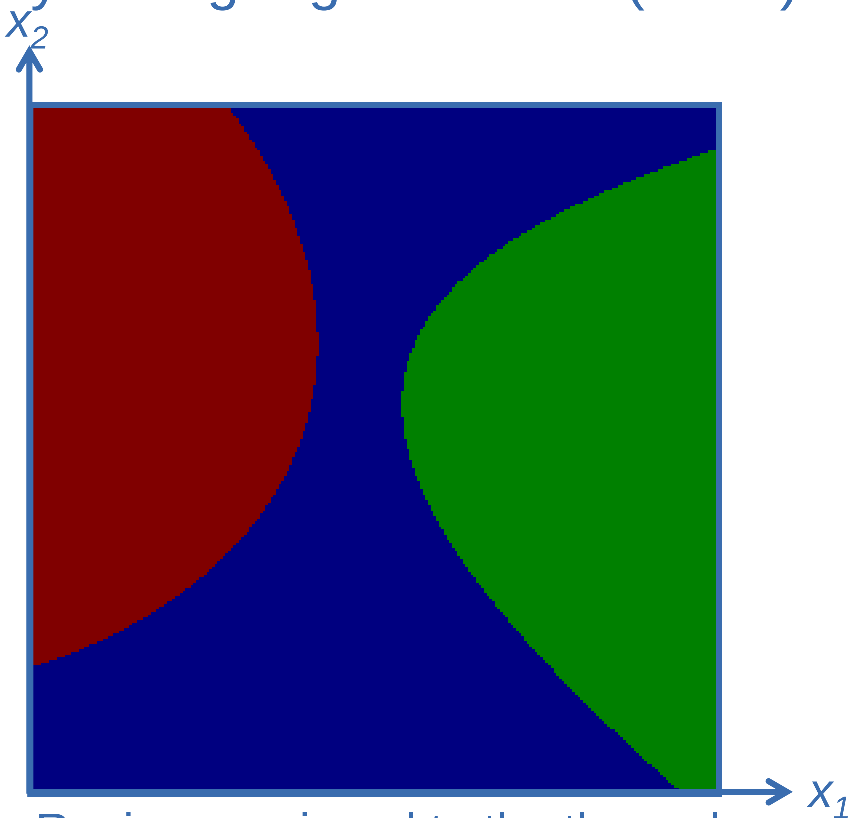


Multi-class Case: Example (ML-Training)

Three classes, two features, not linearly separable – quadratic expansion, training with relatively strong regularization (σ = 2)



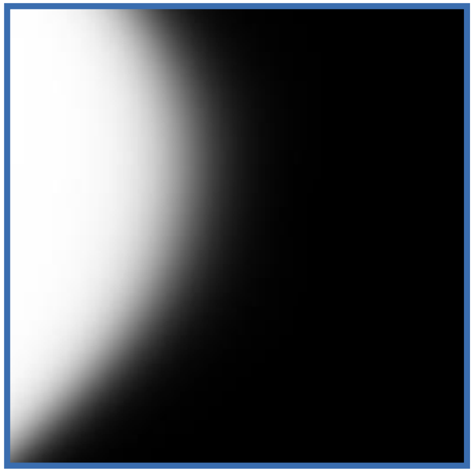
Training samples in feature space (800)



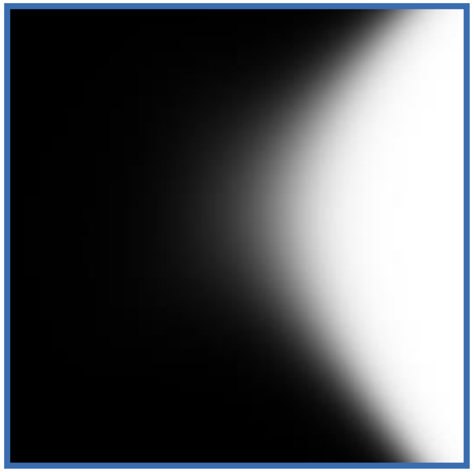
Regions assigned to the three classes in feature space

Multi-class Case: Example (ML-Training)

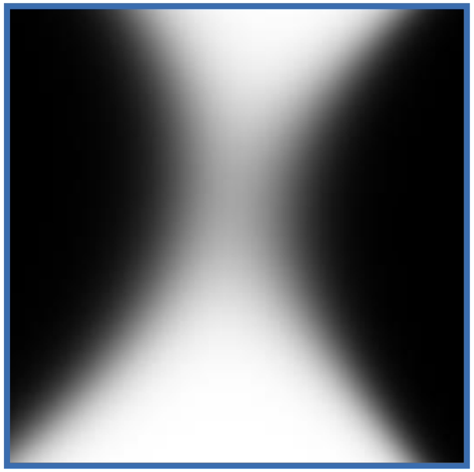
Three classes, two features, not linearly separable – quadratic expansion, training with regularization: posterior probabilities



p(C=L^1*|*x_1,x_2*)*



p(C=L^2*|*x_1,x_2*)*



p(C=L^3*|*x_1,x_2*)*

white ... high probability, black ... low probability

- Much smoother transitions, uncertainty of the classification is better represented
- Class boundaries may be regularized too strongly

Discussion

- Discriminative probabilistic methods directly model the posterior probability
  - No assumption about the distribution of data required
  - Basically, boundaries between classes are learned
  - Linear Models with / without feature space transformation
    - ➢ Fewer parameters to be determined
    - ➢ Fewer training data is required
  - Can be expanded to multi-class problems(model posterior probability using softmax function)
  - Efficient learning / classification
  - Probabilistic output simplifies further processing

Discussion

- Despite feature space transformation, the functional model cannot fit properly to the distribution of the data

  → Transition to non-probabilistic methods

- High-dimensional feature vectors can lead to a large number of parameters to be learned
- Numerical problems → scaling of the features in training and during the classification
- ML-Learning: Problem of overfitting → Regularisation
  - Requires prior for the parameter vector **w**
    → Hyper-parameter σ (cross validation)