*** Applied Machine Learning Fundamentals ***
Logistic Regression

Daniel Wehner, M.Sc.

SAP SE / DHBW Mannheim

Winter term 2023/2024

Find all slides on GitHub (DaWe1992/Applied_ML_Fundamentals)

Lecture Overview

Unit I	Machine Learning Introduction
Unit II	Mathematical Foundations
Unit III	Bayesian Decision Theory
Unit IV	Regression
Unit V	**Classification I**
Unit VI	Evaluation
Unit VII	Classification II
Unit VIII	Clustering
Unit IX	Dimensionality Reduction

Agenda for this Unit

Section:

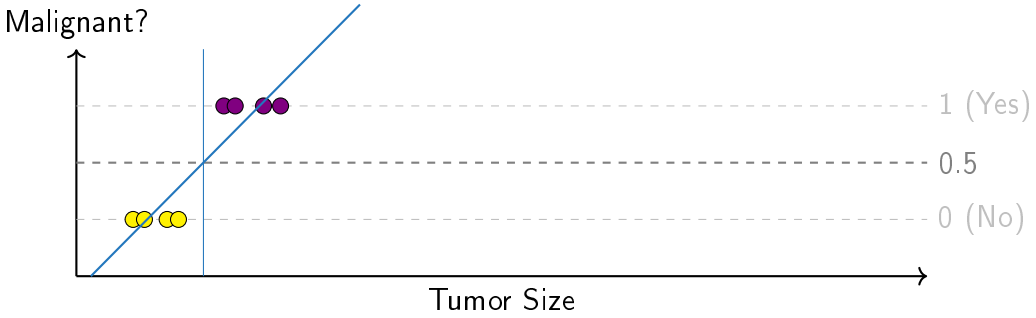
Introduction

What is logistic Regression?
Why you should not use linear Regression

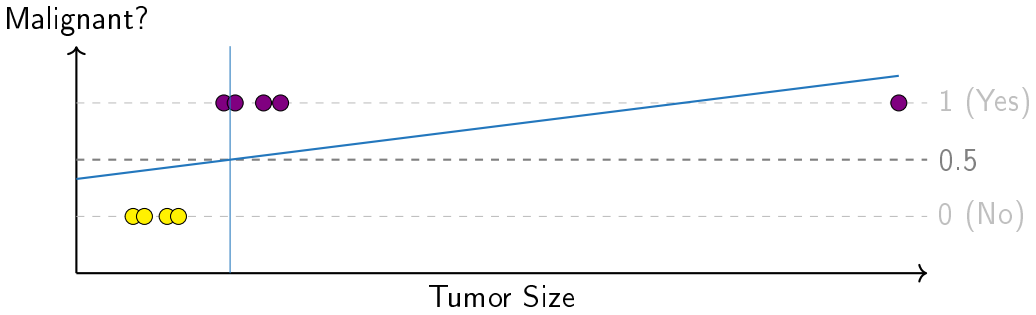
What is logistic Regression?

- Learning algorithm for **classification** *(despite the name...)*
- In its standard form it's applicable to **binary classification problems only**
- **Class labels:**
  - The 'positive class' $\oplus$ is encoded as **1**
  - The 'negative class' $\ominus$ as **0**
- **Probabilistic interpretation:** The output of the algorithm is between 0 and 1 and can be interpreted as the probability of the instance belonging to the positive class

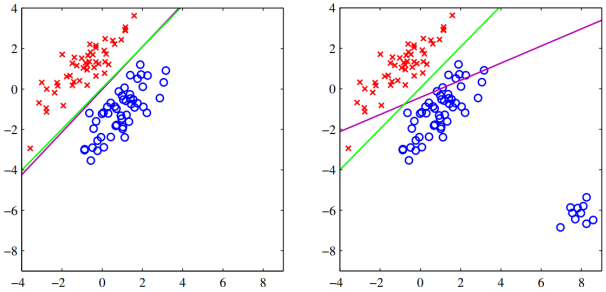
Why you should not use linear Regression...



Why you should not use linear Regression...



Why you should not use linear Regression... (Ctd.)



cf. Bishop.2006, page 186

Magenta line: Linear regression
Green line: Logistic regression

The decision boundary found by linear regression is **heavily affected by outliers!**

Why you should not use linear Regression... (Ctd.)

- Linear regression: $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x}$
- By putting a **threshold** at 0.5, we can turn linear regression into a classifier
  - If $h_{\boldsymbol{\theta}}(\boldsymbol{x}) \geqslant 0.5$, predict $y = 1$
  - If $h_{\boldsymbol{\theta}}(\boldsymbol{x}) < 0.5$, predict $y = 0$
- Problems:
  1. **Outliers heavily affect the decision boundary** (see figure above)
  2. Furthermore, we only want $0 \leqslant h_{\boldsymbol{\theta}}(\boldsymbol{x}) \leqslant 1$; linear regression can output values $h_{\boldsymbol{\theta}}(\boldsymbol{x}) \ll 0$ or $h_{\boldsymbol{\theta}}(\boldsymbol{x}) \gg 1$
- We need a better strategy!

Section:

Model Architecture

Sigmoid Function
Probabilistic Interpretation
Model Training
Decision Boundary

Logistic Regression Model

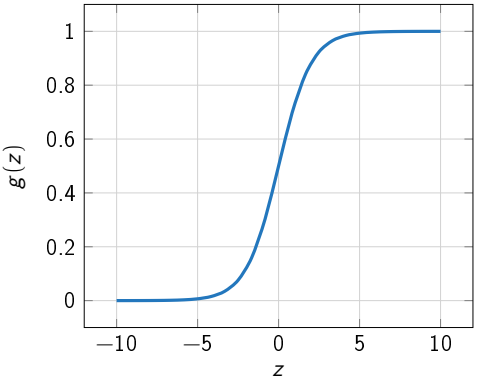
- Remember that we want: $0 \leqslant h_{\boldsymbol{\theta}}(\boldsymbol{x}) \leqslant 1$
- **Solution:** Logistic / Sigmoid function:

$$g(z) := \frac{1}{1 + e^{-z}} \tag{1}$$

- We plug $\boldsymbol{\theta}^{\intercal}\boldsymbol{x}$ into the sigmoid function to obtain our new model function:

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) := g(\boldsymbol{\theta}^{\intercal}\boldsymbol{x}) = \frac{1}{1 + e^{-(\boldsymbol{\theta}^{\intercal}\boldsymbol{x})}} \tag{2}$$

Logistic/Sigmoid Function



- $g(z)$ is symmetric around $z = 0$
- $0 \leqslant g(z) \leqslant 1$ holds true

Where does the Sigmoid come from?

$$p(\mathcal{C}_1|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\boldsymbol{x})} = \frac{p(\boldsymbol{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{\sum_k^K p(\boldsymbol{x}, \mathcal{C}_k)} = \frac{p(\boldsymbol{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{\sum_k^K p(\boldsymbol{x}|\mathcal{C}_k)p(\mathcal{C}_k)}$$

$$= \frac{p(\boldsymbol{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\boldsymbol{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\boldsymbol{x}|\mathcal{C}_2)p(\mathcal{C}_2)}$$

$$= \frac{1}{1 + p(\boldsymbol{x}|\mathcal{C}_2)p(\mathcal{C}_2)/(p(\boldsymbol{x}|\mathcal{C}_1)p(\mathcal{C}_1))}$$

$$= \frac{1}{1 + \exp\{-z\}} = g(z) \quad \longrightarrow \textbf{logistic sigmoid}$$

$$z := \log \frac{p(\boldsymbol{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\boldsymbol{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \quad \longrightarrow \textbf{log odds}$$

Interpretation of Hypothesis Output

- $h_{\theta}(x)$ is interpreted as the probability of instance x belonging to class $y = 1$
- **Example:**

$$x = \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} = \begin{pmatrix} 1 \\ tumorSize \end{pmatrix} \tag{3}$$

- If $h_{\theta}(x) = 0.7$, we have to tell the patient that there is a **70 % chance** of the tumor being malignant $\Rightarrow p(y = 1|x, \theta)$
- **Binary case:** $p(y = 0|x, \theta) = 1 - p(y = 1|x, \theta)$

Training Setup

- We have a labeled training set ($\Rightarrow$ **supervised learning**):

$$\mathcal{D} = \left\{ \left(\boldsymbol{x}^{(1)}, y^{(1)} \right), \left(\boldsymbol{x}^{(2)}, y^{(2)} \right), \ldots, \left(\boldsymbol{x}^{(n)}, y^{(n)} \right) \right\} = \left\{ \left(\boldsymbol{x}^{(i)}, y^{(i)} \right) \right\}_{i=1}^{n} \tag{4}$$

- Each $\boldsymbol{x}$ is a vector of features:

$$\boldsymbol{x} = \begin{pmatrix} x_0 \\ \vdots \\ x_m \end{pmatrix} \in \mathbb{R}^{m+1} \quad \text{with} \quad x_0 = 1 \quad \text{and} \quad y \in \{0, 1\} \tag{5}$$

- How to choose the parameters $\boldsymbol{\theta}$?

Logistic Regression Cost Function

- We require a suitable cost function:

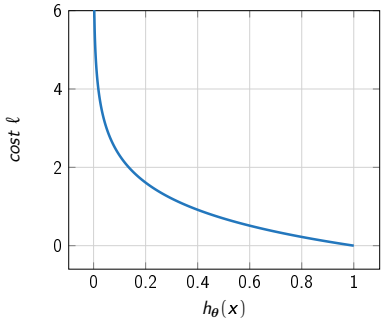
$$\mathcal{J}(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} \ell\big(h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}), y^{(i)}\big) \tag{6}$$

- In our case, the cost function $\ell(h_{\boldsymbol{\theta}}(\boldsymbol{x}), y)$ is defined as follows:
  *(square loss would be **non-convex** due to the sigmoid non-linearity...)*

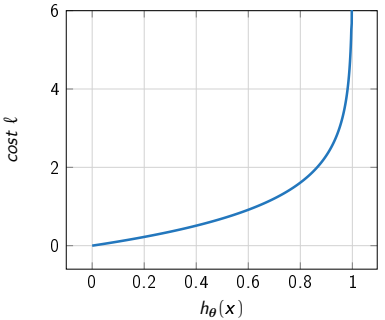
$$\ell(h_{\boldsymbol{\theta}}(\boldsymbol{x}), y) := \begin{cases} -\log(h_{\boldsymbol{\theta}}(\boldsymbol{x})) & \text{if } y = 1 \\ -\log(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x})) & \text{if } y = 0 \end{cases} \tag{7}$$

Logistic Regression Cost Function (Ctd.)

$y = 1$:



$y = 0$:



Logistic Regression Cost Function (Ctd.)

- $\ell(h_{\boldsymbol{\theta}}(\boldsymbol{x}), y)$ can be written in a more compact form:

$$\ell(h_{\boldsymbol{\theta}}(\boldsymbol{x}), y) = -y \log(h_{\boldsymbol{\theta}}(\boldsymbol{x})) - (1 - y) \log(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x})) \tag{8}$$

  - If $y = 1$, we get: $-\log(h_{\boldsymbol{\theta}}(\boldsymbol{x}))$ ✓
  - If $y = 0$, we get: $-\log(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}))$ ✓

- This gives rise to the **(binary) cross entropy** cost function $\mathcal{J}(\boldsymbol{\theta})$:

$$\mathcal{J}(\boldsymbol{\theta}) := \frac{1}{n} \sum_{i=1}^{n} \left[-y^{(i)} \log\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right)\right) - \left(1 - y^{(i)}\right) \log\left(1 - h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right)\right) \right] \tag{9}$$

Derivation of (binary) Cross Entropy using Maximum Likelihood

- The likelihood function can be written in the form:

$$p(\boldsymbol{y}|\boldsymbol{\theta}) = \prod_{i=1}^{n} h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right)^{y^{(i)}} \cdot \left(1 - h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right)\right)^{1-y^{(i)}} \tag{10}$$

- The cost function is then given by the **negative log-likelihood**:

$$\mathcal{J}(\boldsymbol{\theta}) = -\frac{1}{n} \log p(\boldsymbol{y}|\boldsymbol{\theta}) \tag{11}$$

We consider the **negative** log-likelihood because – in machine learning – we prefer minimizing functions over maximizing them. This is allowed as $\max f(x) = \min -f(x)$.

Derivation of (binary) Cross Entropy (Ctd.)

$$\log p(\boldsymbol{y}|\boldsymbol{\theta}) = \log\left(\prod_{i=1}^{n} h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right)^{y^{(i)}} \cdot \left(1 - h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right)\right)^{1-y^{(i)}}\right)$$

Remember the rules:

$$\log(ab) = \log a + \log b$$

$$\log(a^b) = b \log a$$

Where have we used which rule?

$$= \sum_{i=1}^{n} \log\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right)^{y^{(i)}} \cdot \left(1 - h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right)\right)^{1-y^{(i)}}\right)$$

$$= \sum_{i=1}^{n} \left[\log\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right)^{y^{(i)}}\right) + \log\left(1 - h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right)\right)^{1-y^{(i)}}\right)\right]$$

$$= \sum_{i=1}^{n} \left[y^{(i)} \cdot \log\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right)\right) + \left(1 - y^{(i)}\right) \cdot \log\left(1 - h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right)\right)\right]$$

Optimization of (binary) Cross Entropy

- Unfortunately, there is **no closed-form solution** to logistic regression *(due to the sigmoid function)*
- We have to resort to an iterative method like **gradient descent**
- The gradient of $\mathcal{J}(\boldsymbol{\theta})$ (based on a single example) is given by:

$$\frac{\partial}{\partial \theta_j} \mathcal{J}(\boldsymbol{\theta}) = (h_{\boldsymbol{\theta}}(\boldsymbol{x}) - y) \cdot x_j \tag{12}$$

- In the derivation we will need the derivative of the sigmoid function:

$$\frac{\mathrm{d}}{\mathrm{d}z} g(z) = g(z) \cdot (1 - g(z)) \tag{13}$$

$$\frac{\partial}{\partial \theta_j} \mathcal{J}(\boldsymbol{\theta}) = \frac{\partial}{\partial \theta_j} \Big(-y \cdot \log\big(h_{\boldsymbol{\theta}}(\boldsymbol{x})\big) - (1-y) \cdot \log\big(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x})\big) \Big)$$

$$= -y \cdot \frac{\partial}{\partial \theta_j} \Big(\log\big(h_{\boldsymbol{\theta}}(\boldsymbol{x})\big) \Big) - (1-y) \cdot \frac{\partial}{\partial \theta_j} \Big(\log\big(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x})\big) \Big)$$

[Use the derivative of log function: $(\log x)' = \frac{1}{x}$]

$$= \frac{-y}{h_{\boldsymbol{\theta}}(\boldsymbol{x})} \cdot \frac{\partial}{\partial \theta_j} \big(h_{\boldsymbol{\theta}}(\boldsymbol{x})\big) - \frac{1-y}{1 - h_{\boldsymbol{\theta}}(\boldsymbol{x})} \cdot \frac{\partial}{\partial \theta_j} \big(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x})\big)$$

[Factor out the derivative of the model function]

$$= \left(\frac{-y}{h_{\boldsymbol{\theta}}(\boldsymbol{x})} + \frac{1-y}{1 - h_{\boldsymbol{\theta}}(\boldsymbol{x})} \right) \cdot \frac{\partial}{\partial \theta_j} h_{\boldsymbol{\theta}}(\boldsymbol{x})$$

[Find the common denominator]

$$= \frac{-y \cdot \big(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x})\big) + (1-y) \cdot h_{\boldsymbol{\theta}}(\boldsymbol{x})}{h_{\boldsymbol{\theta}}(\boldsymbol{x}) \cdot \big(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x})\big)} \cdot \frac{\partial}{\partial \theta_j} h_{\boldsymbol{\theta}}(\boldsymbol{x})$$

[Expand the numerator]

$$= \frac{-y + y \cdot h_\theta(x) + h_\theta(x) - y \cdot h_\theta(x)}{h_\theta(x) \cdot \big(1 - h_\theta(x)\big)} \cdot \frac{\partial}{\partial \theta_j} h_\theta(x)$$

[Simplify the fraction]

$$= \frac{h_\theta(x) - y}{h_\theta(x) \cdot \big(1 - h_\theta(x)\big)} \cdot \frac{\partial}{\partial \theta_j} h_\theta(x)$$

[Use the definition of the model function: $h_\theta(x) = g(\theta^\intercal x)$]

$$= \frac{g(\theta^\intercal x) - y}{g(\theta^\intercal x) \cdot \big(1 - g(\theta^\intercal x)\big)} \cdot \frac{\partial}{\partial \theta_j} g(\theta^\intercal x)$$

[Use the derivative of the sigmoid function (see equation 13) and apply the chain rule]

$$= \frac{g(\theta^\intercal x) - y}{g(\theta^\intercal x) \cdot \big(1 - g(\theta^\intercal x)\big)} \cdot g(\theta^\intercal x) \cdot \big(1 - g(\theta^\intercal x)\big) \cdot \frac{\partial}{\partial \theta_j} \theta^\intercal x$$

[Cancel redundant terms]

$$= \big(g(\boldsymbol{\theta}^\top \boldsymbol{x}) - y\big) \cdot \frac{\partial}{\partial \theta_j} \boldsymbol{\theta}^\top \boldsymbol{x}$$

[Use the derivative of the scalar product]

$$= \big(g(\boldsymbol{\theta}^\top \boldsymbol{x}) - y\big) \cdot x_j$$

The **stochastic gradient** of the binary cross entropy cost function is therefore given by

$$\nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}) = \begin{pmatrix} \big(g(\boldsymbol{\theta}^\top \boldsymbol{x}) - y\big) \cdot x_1 \\ \vdots \\ \big(g(\boldsymbol{\theta}^\top \boldsymbol{x}) - y\big) \cdot x_m \end{pmatrix} = \big(g(\boldsymbol{\theta}^\top \boldsymbol{x}) - y\big) \cdot \boldsymbol{x}. \tag{14}$$

The **batch gradient** for logistic regression is given by the expression (without proof)

$$\nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}) = \frac{1}{n} \boldsymbol{X}^\top \big(\boldsymbol{g}(\boldsymbol{X}\boldsymbol{\theta}) - \boldsymbol{y}\big), \tag{15}$$

Gradient Descent

- The goal is to minimize $\mathcal{J}(\boldsymbol{\theta})$: $\boldsymbol{\theta}^{\star} = \arg\min_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta})$
- `Repeat until convergence {`
  $\boldsymbol{\theta}^{(t+1)} \longleftarrow \boldsymbol{\theta}^{(t)} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{J}\left(\boldsymbol{\theta}^{(t)}\right)$ *// simultaneously update all* θ_j
  `}`
- The gradient $\nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta})$ is given by equation 15

The algorithm looks identical to linear regression, but the model function $h_{\theta}(x)$ is different (due to the sigmoid function)!

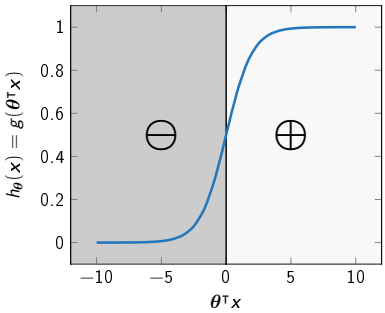
Decision Boundary

- Our trained model **outputs probabilities**
- To obtain a classifier we have to **apply a threshold** to the raw model outputs
- Setting the threshold to 0.5 means:
  - Predict the positive class, if

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) \geqslant 0.5 \iff \boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x} \geqslant 0$$

  - Predict the negative class, if

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) < 0.5 \iff \boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x} < 0$$



Example: Decision Boundary

- Let us consider an easy example
- Suppose we have the following hypothesis:

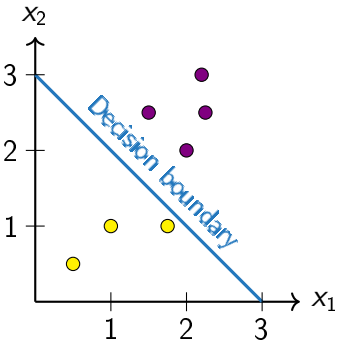
$$h_{\theta}(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

- Using gradient descent we obtained the following model coefficients:

$$\theta_0 = -3, \qquad \theta_1 = 1, \qquad \theta_2 = 1$$

- Predict $y = 1$, if $-3 + x_1 + x_2 \geqslant 0$

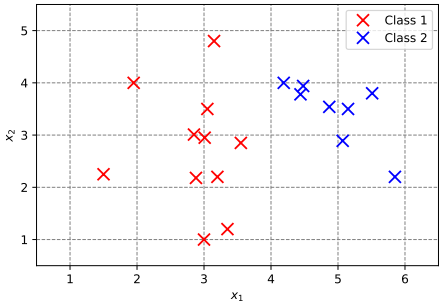
Example: Decision Boundary (Ctd.)

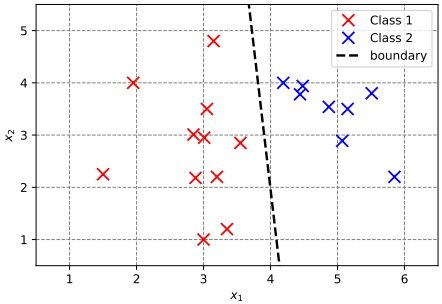


- Predict $y = 1$, if $-3 + x_1 + x_2 \geqslant 0$
- The decision boundary satisfies $-3 + x_1 + x_2 = 0$
- If $x_2 = 0$, then $x_1 = 3$ and vice versa

Logistic regression is not a maximum-margin classifier (although the cost function can be adjusted to get that $\Rightarrow$ Hinge loss)

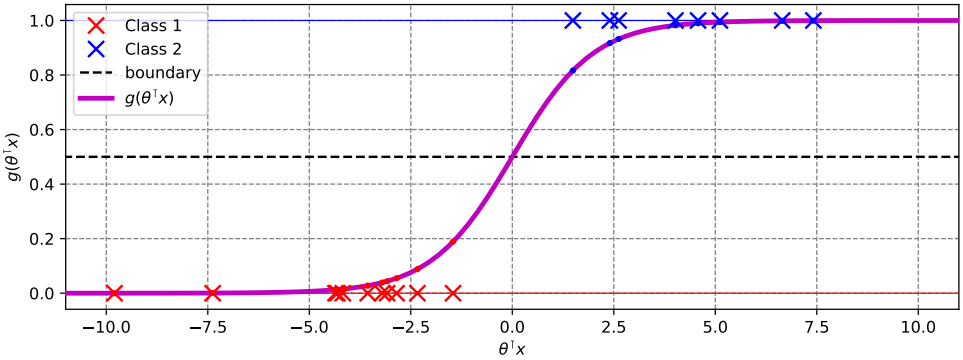
Another Example: Decision Boundary





Where is the sigmoid function?

Another Example: Logistic Function



Section:

Non-linear Data

Feature Mapping
Regularization

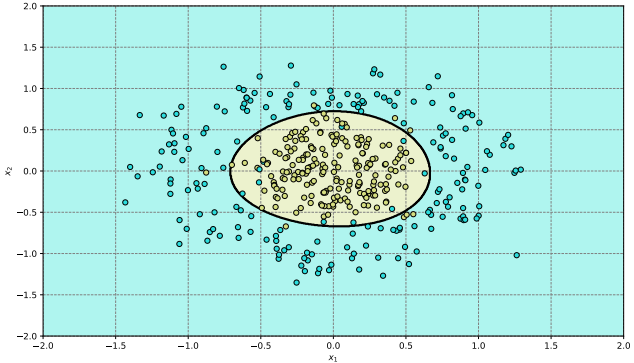
Non-Linear Decision Boundaries

- **Feature mapping** can be used to obtain non-linear decision boundaries/surfaces (same procedure already introduced for linear regression)
- **Example:**
  - Imagine a circular dataset
  - Using the features...

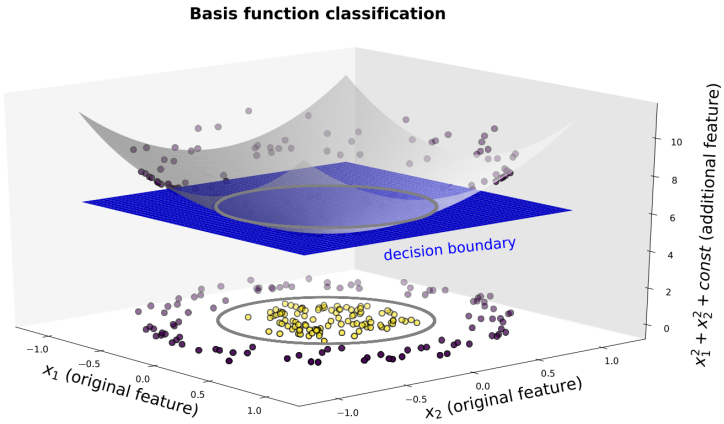
$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$$

  - ...the algorithm could e. g. choose: $\boldsymbol{\theta}^\star = \left[-1, 0, 0, 1, 1\right]^\mathsf{T}$
  - So we would get: $x_1^2 + x_2^2 = 1$ **(equation of a unit circle)**

Example: Non-Linear Decision Boundary



It is still linear!



Logistic Regression with Regularization

- We should apply **regularization** when using the feature mapping approach to avoid running into overfitting
- Add a regularizer to the cost function:

$$\widetilde{\mathcal{J}}(\boldsymbol{\theta}) := \frac{1}{n}\sum_{i=1}^{n}\left[-y^{(i)}\log(h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})) - (1-y^{(i)})\log(1-h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}))\right] + \frac{\lambda}{2m}\sum_{j=1}^{m}\theta_j^2 \qquad (16)$$

- The regularizer prevents the parameters θ_j from becoming too large
- $\lambda \geqslant 0$ controls the degree of regularization
- This leads to smoother decision boundaries

Section:

Multi-Class Classification

Techniques Overview
One-vs-Rest (OvR)
One-vs-One (OvO)

Multi-Class Classification

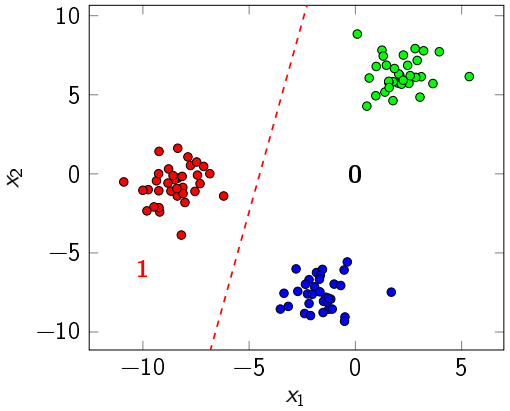
- In its basic form logistic regression can handle two classes only
- **What if there are more than two classes?**
- Two approaches:
  1. Change the algorithm so that it can deal with more classes
     ($\rightarrow$ **Multinomial Logistic Regression** / **Softmax Regression**)
  2. Transform the problem into several binary problems
     Two common techniques are:
     - **One-vs-Rest (OvR)** $\rightarrow$ One-against-All
     - **One-vs-One (OvO)** $\rightarrow$ Pairwise classification
- Let's have a closer look into the second approach!

Transforming the Problem into several binary Problems

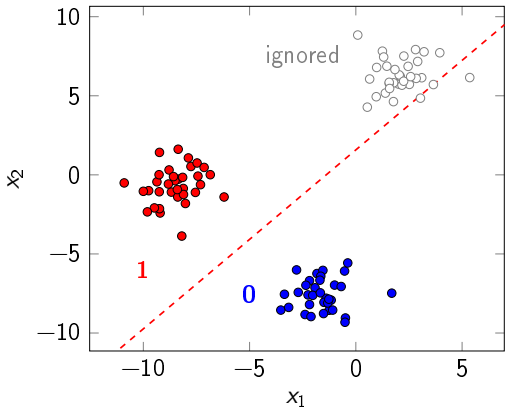
- Instead of adjusting the algorithm we can also **transform the multi-class problem into several binary problems**
- Two common techniques are:
  - **One-vs-Rest (OvR)** → One-against-All
  - **One-vs-One (OvO)** → Pairwise classification
- **General idea:**
  - Several classifiers are trained individually
  - During prediction the classifiers **vote for the correct class**
- Such techniques can be used **for all binary classifiers**

Multi-Class Classification: One-vs-Rest (OvR)

- **Train one classifier per class** (expert for that class)
- We get K classifiers
- The k-th classifier learns to distinguish the k-th class from all the others
- Set the labels of examples from class k to **1**, all the others to **0**



Multi-Class Classification: One-vs-One (OvO)



- **Train one classifier for each pair of classes**
- We get $\binom{K}{2}$ classifiers
- Ignore all other examples that do not belong to either of the two classes
- **Voting**: Count how often each class wins; the class with the highest score is predicted

Section:

Wrap-Up

Summary
Self-Test Questions
Lecture Outlook

Summary

- **Logistic regression is used for classification (!!!)**
- It is used for **binary classification problems** (generalizations exist)
- **Output**: Probability of instance belonging to positive class
- Apply a **threshold** to get the classification
- The algorithm minimizes the **cross entropy cost function**
- There is **no closed-form solution** (unlike for linear regression)
- **Basis functions** can be used for non-linear data
- **Multi-class classification**: One-vs-Rest, One-vs-One

Self-Test Questions

1. Why should you not use linear regression for classification?
2. How is the logistic function defined?
3. Why do we use cross entropy instead of the squared error?
4. Does logistic regression find the best-separating (i.e. maximum margin) hyper-plane?
5. What techniques do you know for multi-class classification problems?

What's next...?

Unit I	Machine Learning Introduction
Unit II	Mathematical Foundations
Unit III	Bayesian Decision Theory
Unit IV	Regression
Unit V	**Classification I**
Unit VI	Evaluation
Unit VII	Classification II
Unit VIII	Clustering
Unit IX	Dimensionality Reduction

Thank you very much for the attention!

Topic: *** Applied Machine Learning Fundamentals *** Logistic Regression
Term: Winter term 2023/2024

Contact:
Daniel Wehner, M.Sc.
SAP SE / DHBW Mannheim
daniel.wehner@sap.com

Do you have any questions?