

*** 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

1 Introduction

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

2 Model Architecture

Sigmoid Function
Probabilistic Interpretation
Model Training
Decision Boundary

3 Non-linear Data

Feature Mapping
Regularization

4 Multi-Class Classification

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

5 Wrap-Up

Summary
Self-Test Questions
Lecture Outlook

Section:
Introduction

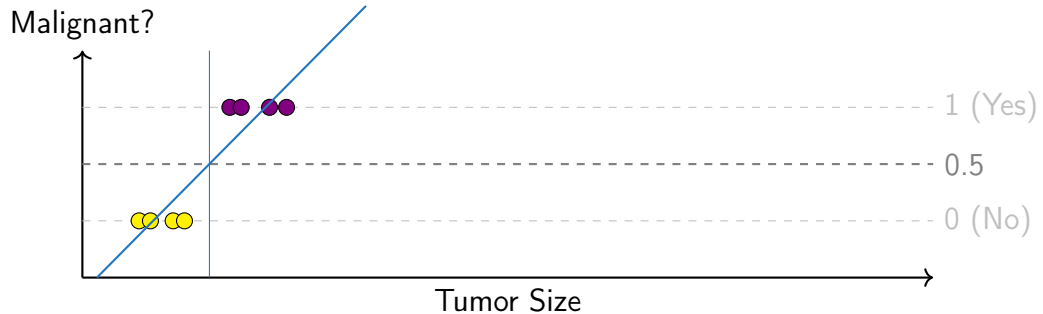


What is logistic Regression?

- Learning algorithm for **classification** (*despite the name...*)
- In its standard form it's applicable to **binary classification problems only**, but you can use techniques like:
 - **One-vs-One (OVO)**
 - **One-vs-Rest (OVR)**
- **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 (*probability of the instance belonging to the positive class*)

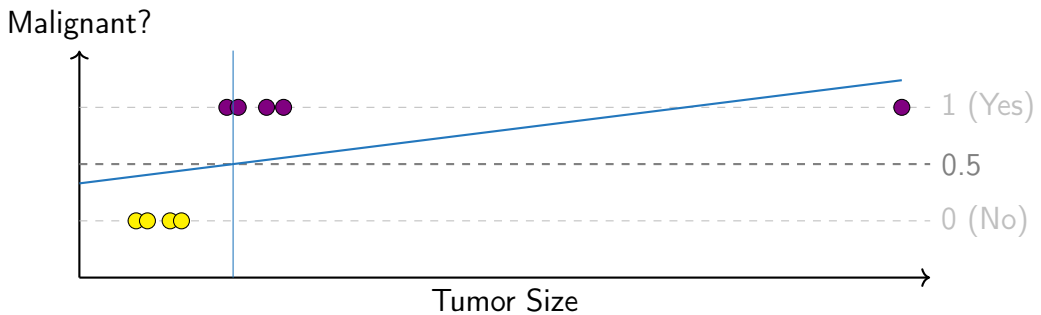


Why you should not use linear Regression...





Why you should not use linear Regression...



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

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

Section:
Model Architecture



Logistic Regression Model

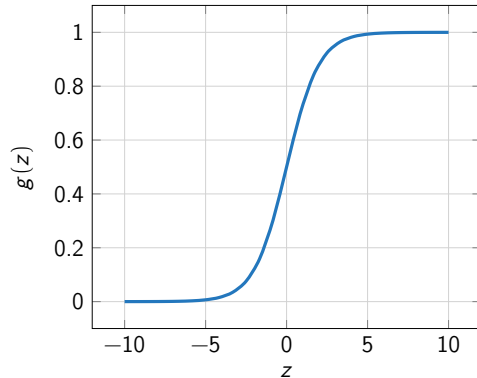
- Remember that we want: $0 \leq h_{\theta}(\mathbf{x}) \leq 1$
- Solution: Logistic / Sigmoid function:**

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

- We plug $\theta^T \mathbf{x}$ into the sigmoid function:

$$h_{\theta}(\mathbf{x}) = g(\theta^T \mathbf{x}) = \frac{1}{1 + e^{-(\theta^T \mathbf{x})}} \quad (2)$$

Logistic/Sigmoid Function



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



Where does the Sigmoid come from?

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{\sum_j p(\mathbf{x}, \mathcal{C}_j)} = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{\sum_j p(\mathbf{x}|\mathcal{C}_j)p(\mathcal{C}_j)}$$

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

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

$$= \frac{1}{1 + \exp\{-z\}} = g(z)$$

→ **logistic sigmoid**

$$z = \log \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}$$

→ **log odds**

Interpretation of Hypothesis Output

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

$$\mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ tumorSize \end{bmatrix} \quad (3)$$

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

Training Setup

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

$$\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \dots, (\mathbf{x}^{(n)}, y^{(n)})\} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^n \quad (4)$$

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

$$\mathbf{x} = \begin{bmatrix} x_0 \\ \vdots \\ x_m \end{bmatrix} \in \mathbb{R}^{m+1} \quad \text{and} \quad x_0 = 1 \quad \text{and} \quad y \in \{0, 1\} \quad (5)$$

- How to choose the parameters θ ?

Logistic Regression Cost Function

- Gradient descent is performed in order to find the parameters θ
- To this end, a cost function is needed:

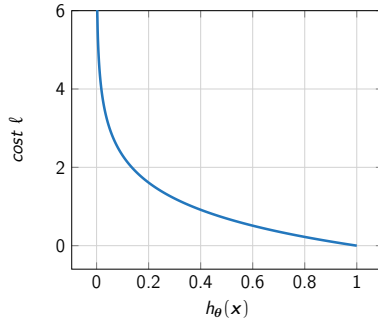
$$\mathcal{J}(\theta) = \frac{1}{n} \sum_{i=1}^n \ell(h_{\theta}(\mathbf{x}^{(i)}), y^{(i)}) \quad (6)$$

- The cost function $\ell(h_{\theta}(\mathbf{x}), y)$ is defined as follows:
(square loss would be **non-convex**...)

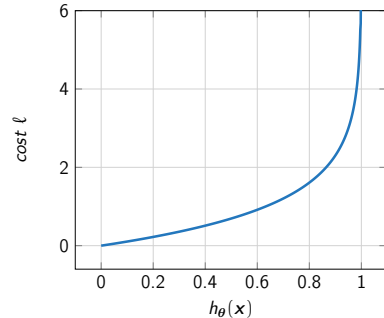
$$\ell(h_{\theta}(\mathbf{x}), y) = \begin{cases} -\log(h_{\theta}(\mathbf{x})) & \text{if } y = 1 \\ -\log(1 - h_{\theta}(\mathbf{x})) & \text{if } y = 0 \end{cases} \quad (7)$$

Logistic Regression Cost Function (Ctd.)

$y = 1$:



$y = 0$:





Logistic Regression Cost Function (Ctd.)

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

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

- If $y = 1$, we get: $-\log(h_{\theta}(\mathbf{x}))$
- If $y = 0$, we get: $-\log(1 - h_{\theta}(\mathbf{x}))$
- This gives the **(binary) cross entropy** cost function $\mathcal{J}(\theta)$:

$$\mathcal{J}(\theta) = \frac{1}{n} \sum_{i=1}^n [-y^{(i)} \log(h_{\theta}(\mathbf{x}^{(i)})) - (1 - y^{(i)}) \log(1 - h_{\theta}(\mathbf{x}^{(i)}))] \quad (9)$$



Derivation of (binary) Cross Entropy

- The likelihood function can be written in the form:

$$\mathcal{L}(\boldsymbol{\theta}) = \prod_{i=1}^n h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})^{y^{(i)}} \cdot (1 - h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}))^{1-y^{(i)}} \quad (10)$$

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

$$\mathcal{J}(\boldsymbol{\theta}) = -\frac{1}{n} \log \mathcal{L}(\boldsymbol{\theta}) \quad (11)$$



Derivation of (binary) Cross Entropy (Ctd.)

$$\mathcal{J}(\theta) = -\frac{1}{n} \log \left[\prod_{i=1}^n h_{\theta}(\mathbf{x}^{(i)})^{y^{(i)}} \cdot (1 - h_{\theta}(\mathbf{x}^{(i)}))^{1-y^{(i)}} \right]$$

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

$$\longrightarrow \log \prod = \sum \log$$

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

$$\longrightarrow \log(a \cdot b) = \log(a) + \log(b)$$

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

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

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**
- We need the gradient of $\mathcal{J}(\boldsymbol{\theta})$ which requires some math (cf. next slides)

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

- Due to the chain rule we also have to find the derivative of the sigmoid function. **Let's get started:**



Derivative of the Sigmoid Function

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

$$\frac{d}{dz}g(z) = \frac{0 \cdot (1 + e^{-z}) - (-e^{-z})}{(1 + e^{-z})^2}$$

→ quotient rule

$$= \frac{e^{-z}}{(1 + e^{-z})^2} = \overset{\text{a small trick...}}{\frac{1 + e^{-z} - 1}{(1 + e^{-z})^2}} = \frac{1 + e^{-z}}{(1 + e^{-z})^2} - \frac{1}{(1 + e^{-z})^2}$$

→ algebraic manipulation

$$= \frac{1}{1 + e^{-z}} \left[1 - \frac{1}{1 + e^{-z}} \right]$$

→ factorize term

$$= \boxed{g(z)(1 - g(z))}$$



Derivation of the Gradient based on a single Example (\mathbf{x}, y)

$$\begin{aligned}\frac{\partial}{\partial \theta_j} \mathcal{J}(\boldsymbol{\theta}) &= -\frac{\partial}{\partial \theta_j} y \log(g(\boldsymbol{\theta}^\top \mathbf{x})) - \frac{\partial}{\partial \theta_j} (1-y) \log(1-g(\boldsymbol{\theta}^\top \mathbf{x})) && \longrightarrow \text{derivative of sum terms} \\ &= \left[-\frac{y}{g(\boldsymbol{\theta}^\top \mathbf{x})} + \frac{1-y}{1-g(\boldsymbol{\theta}^\top \mathbf{x})} \right] \frac{\partial}{\partial \theta_j} g(\boldsymbol{\theta}^\top \mathbf{x}) && \longrightarrow \text{derivative of log function} \\ &= \left[-\frac{y}{g(\boldsymbol{\theta}^\top \mathbf{x})} + \frac{1-y}{1-g(\boldsymbol{\theta}^\top \mathbf{x})} \right] g(\boldsymbol{\theta}^\top \mathbf{x})(1-g(\boldsymbol{\theta}^\top \mathbf{x})) \frac{\partial}{\partial \theta_j} \boldsymbol{\theta}^\top \mathbf{x} && \longrightarrow \text{chain rule} \\ &= \left[\frac{g(\boldsymbol{\theta}^\top \mathbf{x}) - y}{g(\boldsymbol{\theta}^\top \mathbf{x})(1-g(\boldsymbol{\theta}^\top \mathbf{x}))} \right] g(\boldsymbol{\theta}^\top \mathbf{x})(1-g(\boldsymbol{\theta}^\top \mathbf{x})) x_j && \longrightarrow \text{algebraic manipulation} \\ &= (g(\boldsymbol{\theta}^\top \mathbf{x}) - y) x_j = \boxed{(h_{\boldsymbol{\theta}}(\mathbf{x}) - y) x_j} && \longrightarrow \text{cancel terms}\end{aligned}$$



Gradient Descent

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

$$\nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right) \mathbf{x}^{(i)} \quad (13)$$

Algorithm looks identical to linear regression, but $h_{\boldsymbol{\theta}}(\mathbf{x})$ is different!

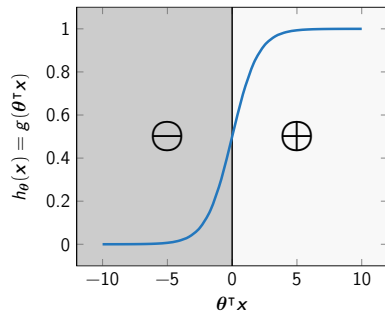
Decision Boundary

- We have to set a threshold
- Setting the threshold to 0.5 means:
 - Predict the positive class, if

$$h_{\theta}(\mathbf{x}) \geq 0.5 \Leftrightarrow \theta^T \mathbf{x} \geq 0$$

- Predict the negative class, if

$$h_{\theta}(\mathbf{x}) < 0.5 \Leftrightarrow \theta^T \mathbf{x} < 0$$



Decision Boundary (Ctd.)

- Suppose we have the following hypothesis:

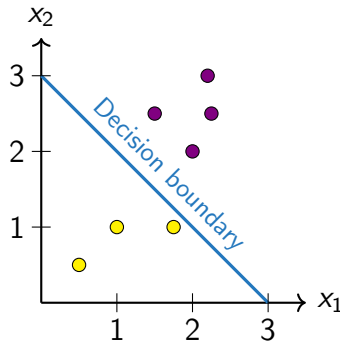
$$h_{\theta}(\mathbf{x}) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

- Using gradient descent we obtained the following coefficients:

$$\theta_0 = -3 \quad \theta_1 = 1 \quad \theta_2 = 1$$

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

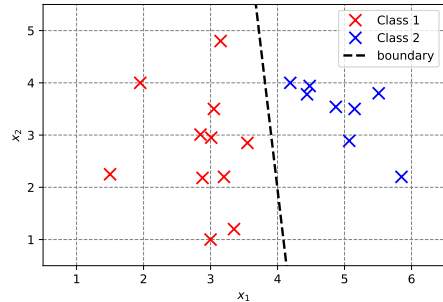
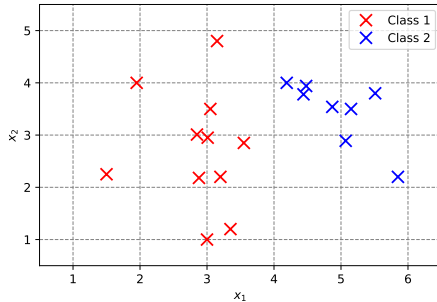
Decision Boundary (Ctd.)



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

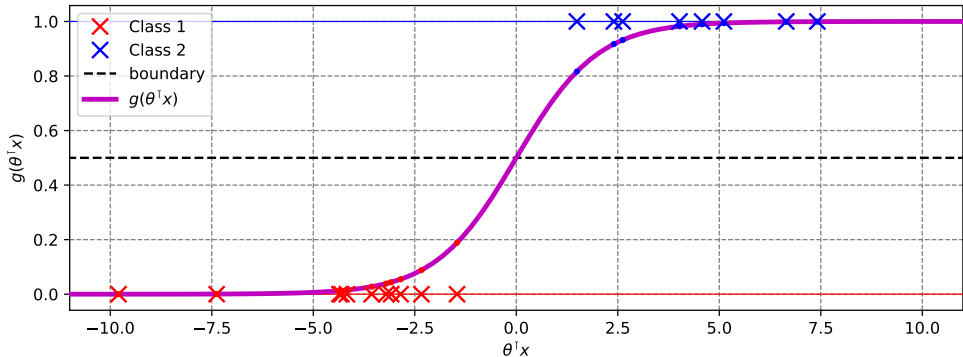
Logistic regression is not a maximum-margin classifier (although the cost function can be adjusted to get that \Rightarrow Hinge loss)

Example: Decision Boundary



Where is the sigmoid function?

Example: Logistic Function



Section:
Non-linear Data



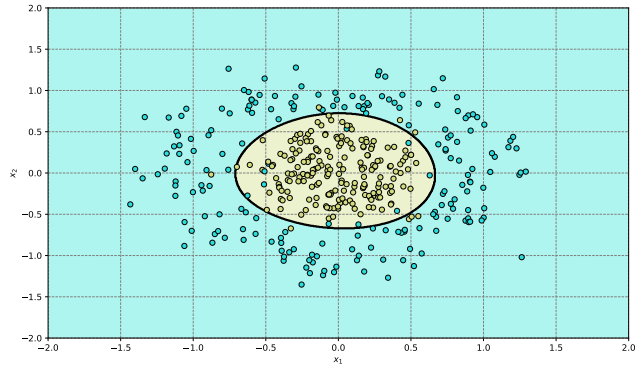
Non-Linear Decision Boundaries

- **Feature mapping** can be used to obtain non-linear decision boundaries
- **Example:**
 - Imagine a circular data set
 - Using the features...

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

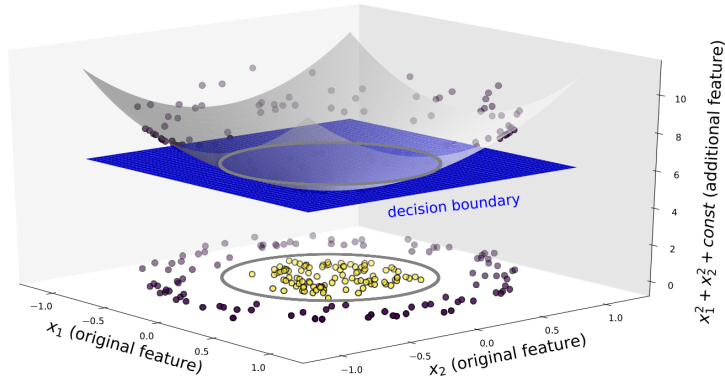
- ...the algorithm could e. g. choose: $\theta = [-1, 0, 0, 1, 1]^T$
- So we would get: $x_1^2 + x_2^2 = 1 \Rightarrow$ **equation of a unit circle**

Example: Non-Linear Decision Boundary



It is still linear!

Basis function classification



Logistic Regression with Regularization

- We should apply regularization for non-linear decision boundaries:

$$\frac{1}{n} \sum_{i=1}^n \left[-y^{(i)} \log(h_{\theta}(\mathbf{x}^{(i)})) - (1 - y^{(i)}) \log(1 - h_{\theta}(\mathbf{x}^{(i)})) \right] + \frac{\lambda}{2m} \sum_{j=1}^m \theta_j^2 \quad (14)$$

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

Section:
Multi-Class Classification



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
(→ **Multinomial Logistic Regression** / **Softmax Regression**)
 - ② Transform the problem into several binary problems.
Two common techniques are:
 - **One-vs-Rest (OvR)** → One-against-All
 - **One-vs-One (OvO)** → Pairwise classification
- Let's have a closer look into the second approach!

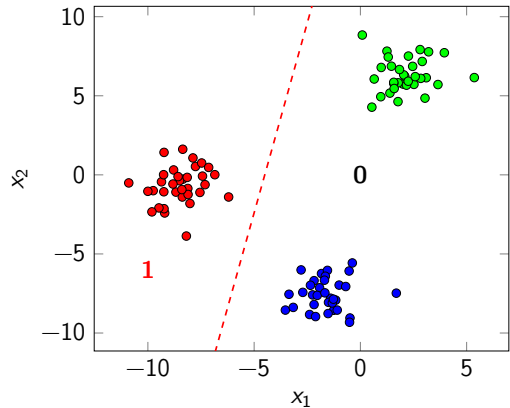


Transforming the Problem into several binary Problems

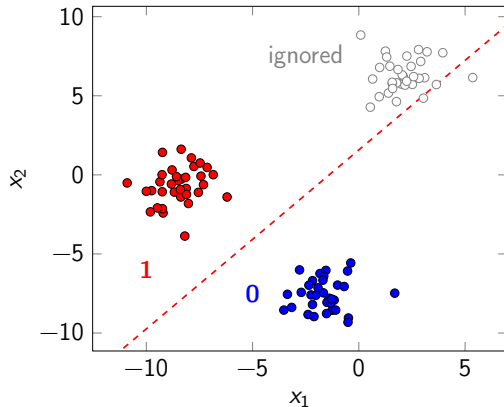
- Instead of adjusting the algorithm we can also **transform the multi-class problem into several binary problems**
- Two common techniques are:
 - **One-vs-Rest (OvR)** → One-against-All
 - **One-vs-One (OvO)** → Pairwise classification
- **General idea:**
 - Several classifiers are trained individually
 - 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 $|\mathcal{C}|$ 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 $\binom{|C|}{2}$ 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

- **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 State the formula for the logistic function.
- 3 Why do we use cross entropy instead of the squared error?
- 4 Does logistic regression find the best-separating 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?