

*** 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

② Model Architecture

③ Non-linear Data

④ Multi-Class Classification

⑤ Wrap-Up

Section: Introduction

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

What is logistic Regression?

- Logistic regression is a learning algorithm for **classification (!!!)**
- In its standard form it is 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 raw 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...

- Linear regression:

$$h_{\theta}(\mathbf{x}) = \theta^T \mathbf{x}$$

- We can turn linear regression into a classifier by putting a **threshold** at $h_{\theta}(\mathbf{x}) = 0.5$ (or any other value between 0 and 1)
 - 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** (see example below)
 - 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 $h_{\theta}(\mathbf{x}) \gg 1$

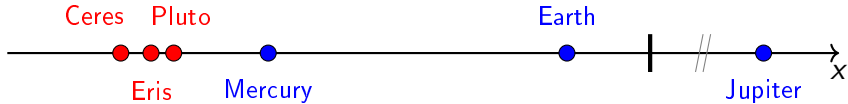
Why you should not use linear Regression... (Ctd.)

Consider the following dataset:

Row	Object	Radius ($\times 10^6$ m)	Label	Label encoded
1	Ceres	1.0	dwarf planet	0
2	Eris	2.3	dwarf planet	0
3	Pluto	2.4	dwarf planet	0
4	Mercury	4.9	planet	1
5	Earth	12.8	planet	1
6	Jupiter	143.0	planet	1

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

- Let us train a linear regression model for classification:

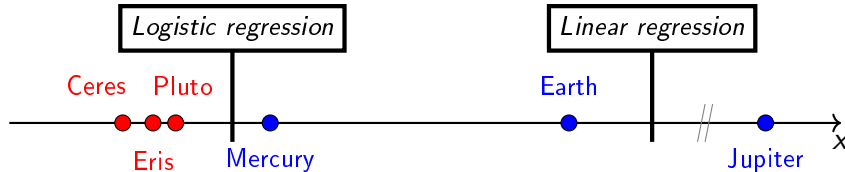


- Both, Mercury and Earth, are classified as dwarf planets due to Jupiter's massive radius!

Linear regression is sensitive to outliers! We need a better cost function!

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

- Logistic regression to the rescue:



- Logistic regression is less sensitive to outliers

(It is a valuable exercise to reproduce this result. See the exercise sheet!)

Section: Model Architecture

Sigmoid Function
Probabilistic Interpretation
Model Training
Decision Boundary



Logistic Regression Model

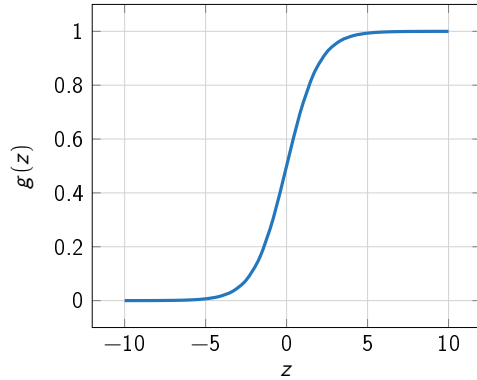
- Remember that we want: $0 \leq h_{\theta}(\mathbf{x}) \leq 1$
- Solution: **Logistic function / Sigmoid function:**

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

- We plug $\theta^T \mathbf{x}$ into the sigmoid function to obtain our new model 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?

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

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

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

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

→ **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

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

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

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

$$p(y = 0|x, \theta) = 1 - p(y = 1|x, \theta)$$

Training Setup

- We have a labeled training set:

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

- Each $y \in \{0, 1\}$ and each \mathbf{x} is a vector of features:

$$\mathbf{x} = \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_m \end{pmatrix} = \begin{pmatrix} 1 \\ x_1 \\ \vdots \\ x_m \end{pmatrix} \in \mathbb{R}^{m+1} \quad (5)$$

Logistic Regression Cost Function

- We require a suitable cost function:

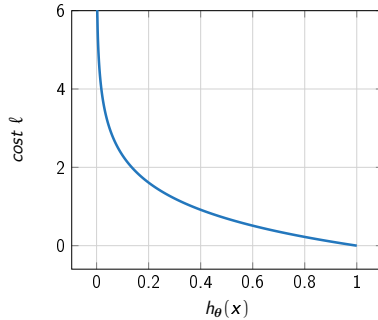
$$\mathcal{J}(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^n \ell(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}), y^{(i)}) \quad (6)$$

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

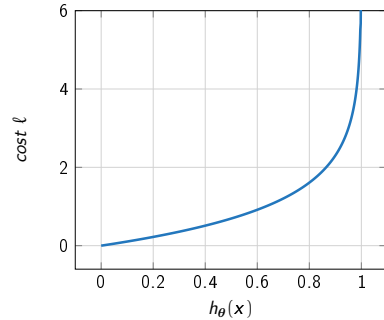
$$\ell(h_{\boldsymbol{\theta}}(\mathbf{x}), y) := \begin{cases} -\log(h_{\boldsymbol{\theta}}(\mathbf{x})) & \text{if } y = 1 \\ -\log(1 - h_{\boldsymbol{\theta}}(\mathbf{x})) & \text{if } y = 0 \end{cases} \quad (7)$$

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

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



Derivation of (binary) Cross Entropy using Maximum Likelihood

- The likelihood function for logistic regression can be written in the form:

$$p(y|\theta) := \prod_{i=1}^n h_{\theta}(x^{(i)})^{y^{(i)}} \cdot (1 - h_{\theta}(x^{(i)}))^{1-y^{(i)}} \quad (10)$$

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

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

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



Derivation of (binary) Cross Entropy (Ctd.)

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

Remember the rules:

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

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

*Where have we used
which rule?*

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

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

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

Optimization of (binary) Cross Entropy

- Unfortunately, there is **no closed-form solution** to logistic regression
(*due to the sigmoid non-linearity in the model function*)
- We have to resort to an iterative method like **gradient descent**

The partial derivative of $\mathcal{J}(\boldsymbol{\theta})$ (based on a single example) with respect to the j -th model parameter θ_j is given by:

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



Optimization of (binary) Cross Entropy (Ctd.)

- The **stochastic gradient** of the binary cross entropy cost function is

$$\nabla_{\theta} \mathcal{J}(\theta) = \begin{pmatrix} (g(\theta^T \mathbf{x}) - y) \cdot x_1 \\ \vdots \\ (g(\theta^T \mathbf{x}) - y) \cdot x_m \end{pmatrix} = (g(\theta^T \mathbf{x}) - y) \mathbf{x}. \quad (13)$$

- The **batch gradient** for logistic regression is given by the expression

$$\nabla_{\theta} \mathcal{J}(\theta) = \frac{1}{n} \mathbf{X}^T (\mathbf{g}(\mathbf{X}\theta) - \mathbf{y}) \quad (14)$$



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

- We give a proof of equation (12)
- In the derivation we will need the derivative of the sigmoid function:

$$\frac{d}{dz}g(z) = g(z) \cdot (1 - g(z)) \quad (15)$$

(You will be asked to proof this in the exercises!)

- Please find the derivation \Rightarrow [here](#)



Gradient Descent

- The goal is to minimize the cost function $\mathcal{J}(\boldsymbol{\theta})$:

$$\boldsymbol{\theta}^* = \arg \min_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta})$$

- Repeat until convergence

$$\{\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}_t)\}$$

- The gradient $\nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta})$ is given by equation (14)

The algorithm looks identical to linear regression, but the model function is different due to the sigmoid function!

Decision Boundary

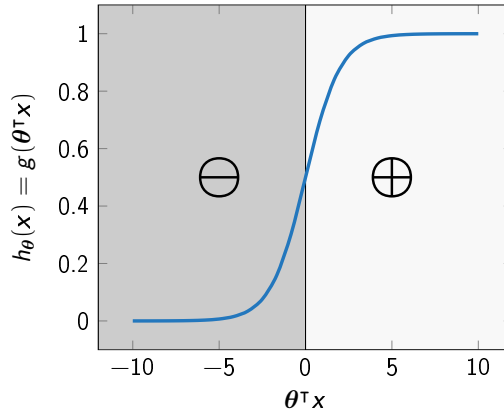
- Our trained model **outputs probabilities**
- To obtain a classifier, we have to **apply a threshold ρ** to the raw outputs
- Setting the threshold to $\rho := 0.5$ means:
 - Predict the positive class \oplus , if

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

- Predict the negative class \ominus , if

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

Decision Boundary (Ctd.)



Example: Decision Boundary

- Let us consider a simple example
- Suppose our model function takes the form:

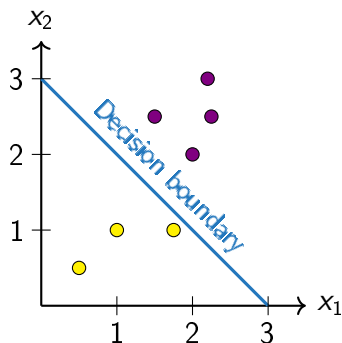
$$h_{\theta}(\mathbf{x}) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

- Assume we obtain the following model parameters using gradient descent:

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

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

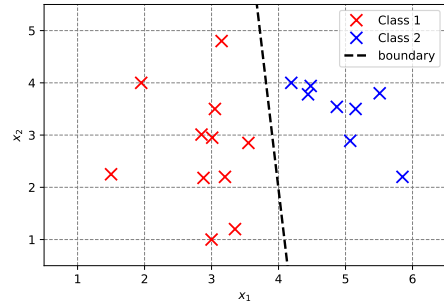
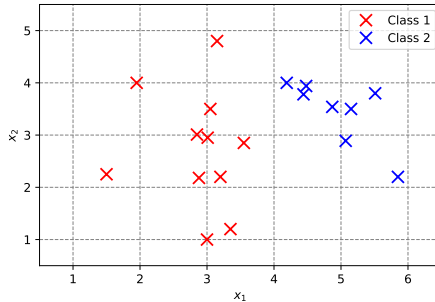
Example: Decision Boundary (Ctd.)



- 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

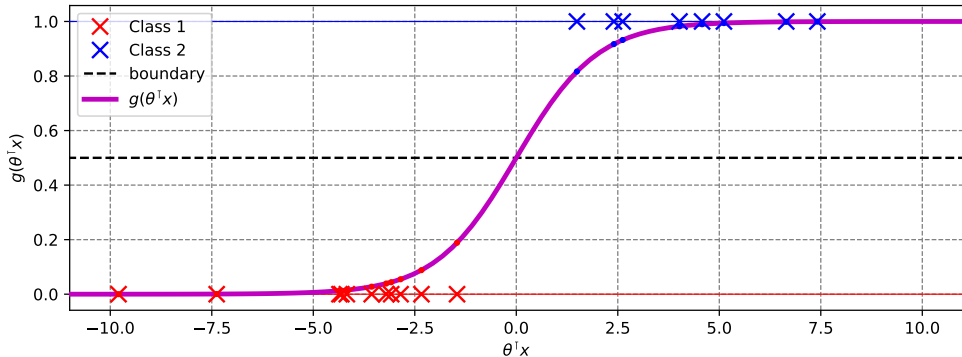
Logistic regression is not a maximum-margin classifier, but the cost function can be adjusted to get that \Rightarrow Hinge loss

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)

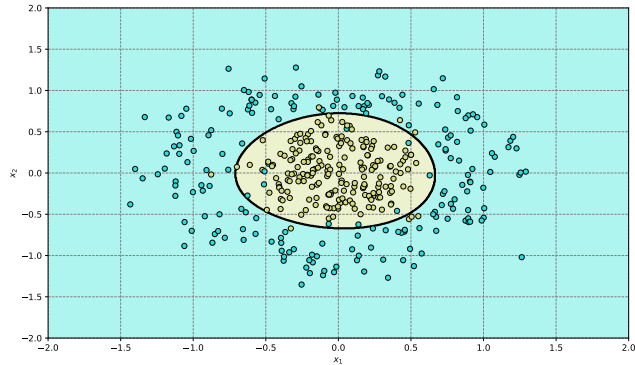
Example:

- Let a circular two-dimensional dataset be given (features x_1 and x_2)
- We choose the following model function:

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

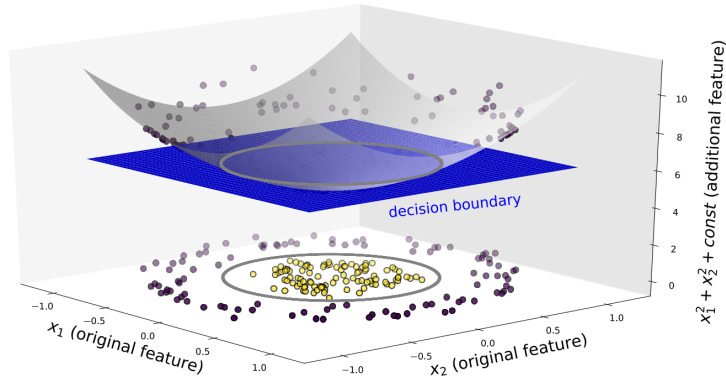
- The algorithm could choose the parameters $\theta^* := (-1, 0, 0, 1, 1)^T$
- So we would get: $x_1^2 + x_2^2 = 1$ (**equation of a unit circle**)

Example: Non-Linear Decision Boundary





It is still linear!

Basis function classification



Logistic Regression with Regularization

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

$$\tilde{\mathcal{J}}(\boldsymbol{\theta}) := \mathcal{J}(\boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}\|^2 \quad (16)$$

- The regularizer 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

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
(→ **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!

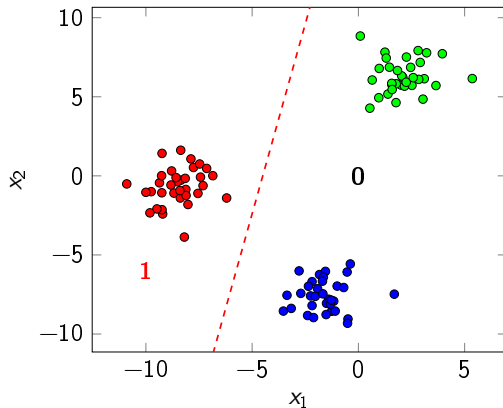


Transforming the Problem into several binary Problems

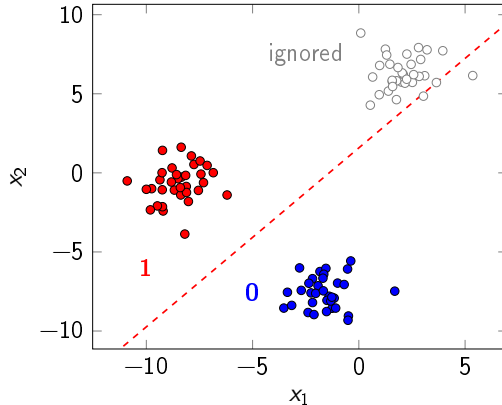
- Instead of adjusting the algorithm, we can also **transform the multi-class problem into several binary problems**
- Two common techniques are:
 - **One-vs-Rest (OvR)** \rightarrow One-against-All
 - **One-vs-One (OvO)** \rightarrow Pairwise classification
- **General idea:**
 - Several classifiers 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** ρ 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?