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

Daniel Wehner, M.Sc.

SAP SE / DHBW Mannheim

Winter term 2021/2022

Find all slides on GitHub (DaWe1992/Applied_ML_Fundamentals)

Lecture Overview

Agenda for this Unit

Section:

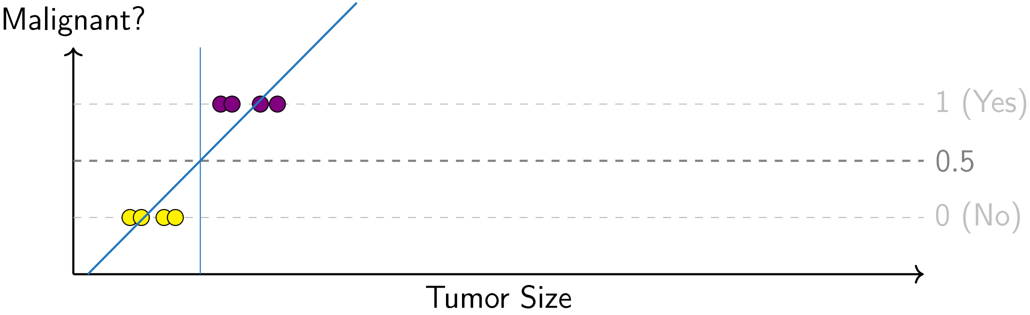
Introduction

Duale Hochschule
Baden-Württemberg

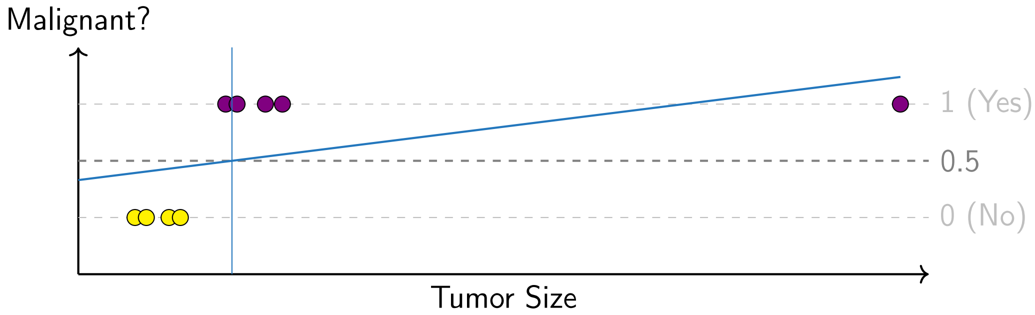
What is logistic Regression?

- 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: $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{\theta}^\intercal \boldsymbol{x}$
- 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**
  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

Logistic Regression Model

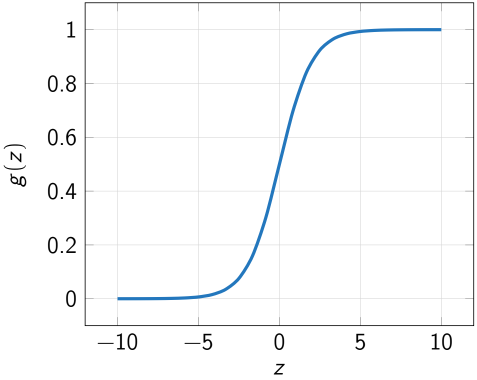
- Remember that we want: $0 \leqslant h_{\boldsymbol{\theta}}(\boldsymbol{x}) \leqslant 1$
- **Solution:** **Logistic / Sigmoid function:**

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

- We plug $\boldsymbol{\theta}^{\intercal}\boldsymbol{x}$ into the sigmoid function:

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

Logistic/Sigmoid Function



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

Where does the Sigmoid come from?

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

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

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

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

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

Interpretation of Hypothesis Output

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

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

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

Training Setup

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

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

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

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

- How to choose the parameters θ?

Logistic Regression Cost Function

- Gradient descent is performed in order to find the parameters $\boldsymbol{\theta}$
- To this end, a cost function is needed:

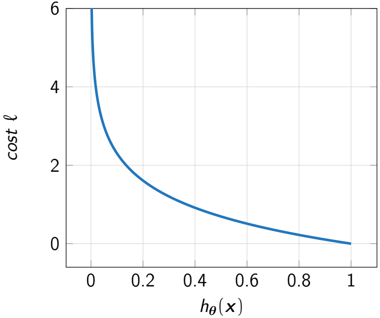
$$\mathcal{J}(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} \ell(h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}), y^{(i)}) \tag{6}$$

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

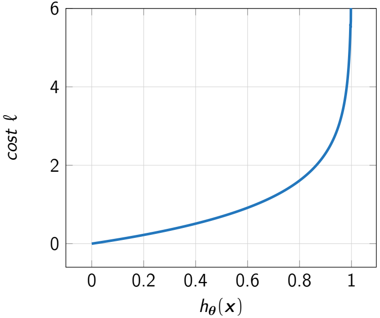
$$\ell(h_{\boldsymbol{\theta}}(\boldsymbol{x}), y) = \begin{cases} -\log(h_{\boldsymbol{\theta}}(\boldsymbol{x})) & \text{if } y = 1 \\ -\log(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x})) & \text{if } y = 0 \end{cases} \tag{7}$$

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 the **(binary) cross entropy** cost function $\mathcal{J}(\boldsymbol{\theta})$:

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

Derivation of (binary) Cross Entropy

- The likelihood function can be written in the form:

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

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

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

Derivation of (binary) Cross Entropy (Ctd.)

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

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

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

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

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)

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

- 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

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

$$\frac{\mathrm{d}}{\mathrm{d}z} g(z) = \frac{0 \cdot (1+e^{-z}) - (-e^{-z})}{(1+e^{-z})^2} \qquad \longrightarrow \text{quotient rule}$$

$$= \frac{e^{-z}}{(1+e^{-z})^2} = \frac{\overbrace{1+e^{-z}-1}^{\text{a small trick...}}}{(1+e^{-z})^2} = \frac{1+e^{-z}}{(1+e^{-z})^2} - \frac{1}{(1+e^{-z})^2} \qquad \longrightarrow \text{algebraic manipulation}$$

$$= \frac{1}{1+e^{-z}} \left[1 - \frac{1}{1+e^{-z}}\right] \qquad \longrightarrow \text{factorize term}$$

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

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

$$\frac{\partial}{\partial \theta_j} \mathcal{J}(\boldsymbol{\theta}) = -\frac{\partial}{\partial \theta_j} y \log(g(\boldsymbol{\theta}^\intercal \boldsymbol{x})) - \frac{\partial}{\partial \theta_j} (1-y) \log(1-g(\boldsymbol{\theta}^\intercal \boldsymbol{x})) \qquad \longrightarrow \text{derivative of sum terms}$$

$$= \left[-\frac{y}{g(\boldsymbol{\theta}^\intercal \boldsymbol{x})} + \frac{1-y}{1-g(\boldsymbol{\theta}^\intercal \boldsymbol{x})} \right] \frac{\partial}{\partial \theta_j} g(\boldsymbol{\theta}^\intercal \boldsymbol{x}) \qquad \longrightarrow \text{derivative of log function}$$

$$= \left[-\frac{y}{g(\boldsymbol{\theta}^\intercal \boldsymbol{x})} + \frac{1-y}{1-g(\boldsymbol{\theta}^\intercal \boldsymbol{x})} \right] g(\boldsymbol{\theta}^\intercal \boldsymbol{x})(1-g(\boldsymbol{\theta}^\intercal \boldsymbol{x})) \frac{\partial}{\partial \theta_j} \boldsymbol{\theta}^\intercal \boldsymbol{x} \qquad \longrightarrow \text{chain rule}$$

$$= \left[\frac{g(\boldsymbol{\theta}^\intercal \boldsymbol{x}) - y}{g(\boldsymbol{\theta}^\intercal \boldsymbol{x})(1-g(\boldsymbol{\theta}^\intercal \boldsymbol{x}))} \right] g(\boldsymbol{\theta}^\intercal \boldsymbol{x})(1-g(\boldsymbol{\theta}^\intercal \boldsymbol{x})) x_j \qquad \longrightarrow \text{algebraic manipulation}$$

$$= (g(\boldsymbol{\theta}^\intercal \boldsymbol{x}) - y) x_j = \boxed{(h_{\boldsymbol{\theta}}(\boldsymbol{x}) - y) x_j} \qquad \longrightarrow \text{cancel terms}$$

Gradient Descent

- The goal is to minimize $\mathcal{J}(\boldsymbol{\theta})$: $\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta})$
- `Repeat until convergence {`

$$\boldsymbol{\theta}^{(t+1)} \longleftarrow \boldsymbol{\theta}^{(t)} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}^{(t)}) \qquad \textit{// simultaneously update all } \theta_j$$

`}`

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

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

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

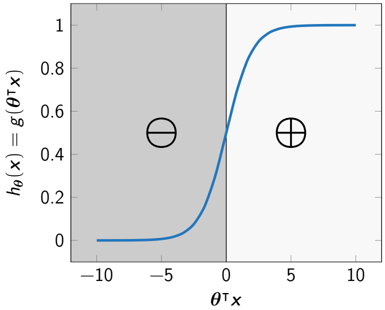
Decision Boundary

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

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) \geqslant 0.5 \Leftrightarrow \boldsymbol{\theta}^\intercal \boldsymbol{x} \geqslant 0$$

  - Predict the negative class, if

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) < 0.5 \Leftrightarrow \boldsymbol{\theta}^\intercal \boldsymbol{x} < 0$$



Decision Boundary (Ctd.)

- Suppose we have the following hypothesis:

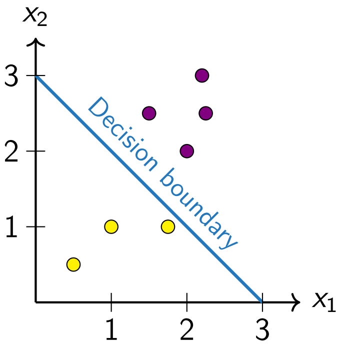
$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

- Using gradient descent we obtained the following coefficients:

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

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

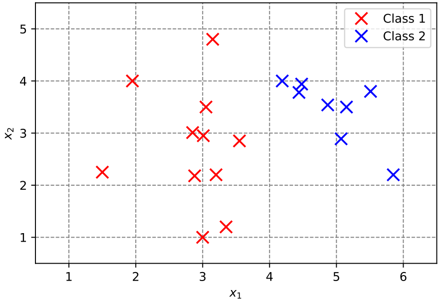
Decision Boundary (Ctd.)

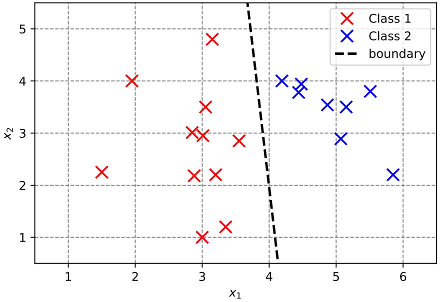


- Predict $y = 1$, if $-3 + x_1 + x_2 \geqslant 0$
- The decision boundary satisfies $-3 + x_1 + x_2 = 0$
- If $x_2 = 0$, then $x_1 = 3$ and vice versa

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

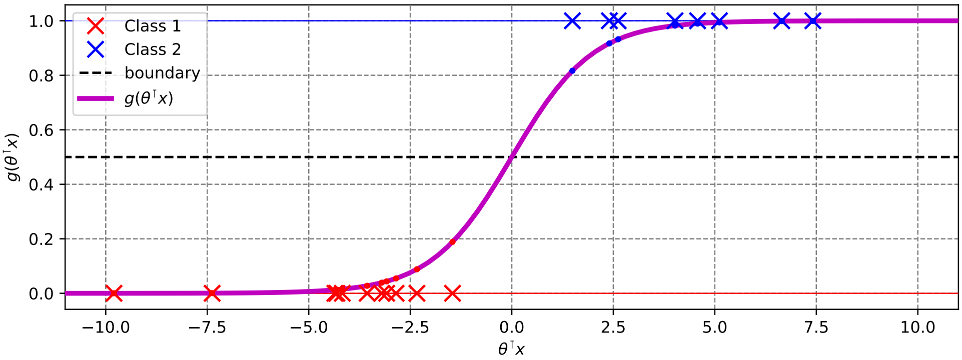
Example: Decision Boundary





Where is the sigmoid function?

Example: Logistic Function



Section:

Non-linear Data

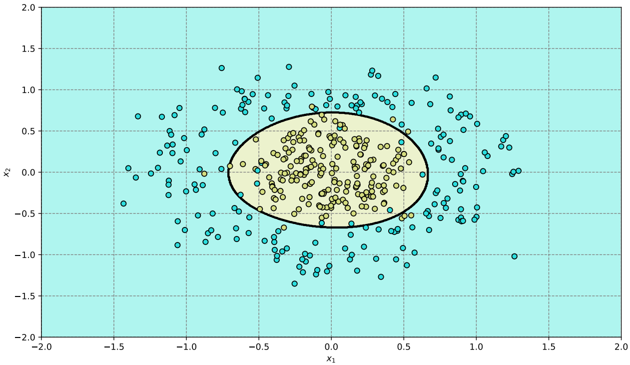
Non-Linear Decision Boundaries

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

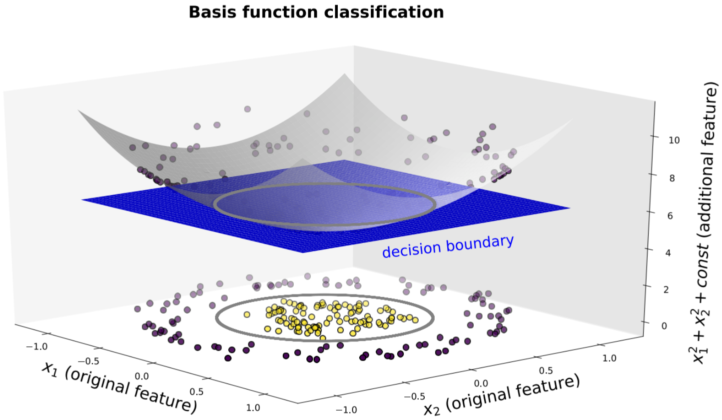
$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$$

  - ...the algorithm could e. g. choose: $\boldsymbol{\theta} = \left[-1, 0, 0, 1, 1\right]^\intercal$
  - 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!



Logistic Regression with Regularization

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

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

- The last term 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

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 examine these approaches a bit closer

Multinomial Logistic Regression Introduction

- The logistic regression model has to be changed in order to deal with multiple classes
- The sigmoid function is replaced by the Softmax function:

$$\boldsymbol{g} : \mathbb{R}^{\mathrm{K}} \to \mathbb{R}^{\mathrm{K}}, \qquad \boldsymbol{z} \mapsto \boldsymbol{g}(\boldsymbol{z}), \qquad g_k(\boldsymbol{z}) = \frac{e^{z_k}}{\sum_{n=1}^{\mathrm{K}} e^{z_n}} \tag{15}$$

- K is the number of possible outcomes / classes
- The softmax function returns a **probability distribution over the possible outcomes**, i. e. $\sum_{k=1}^{\mathrm{K}} g_k(\boldsymbol{z}) = 1$

Multinomial Logistic Regression Introduction (Ctd.)

- $\boldsymbol{z} = \begin{pmatrix} \boldsymbol{\theta}_1^\intercal \boldsymbol{x} & \boldsymbol{\theta}_2^\intercal \boldsymbol{x} & \dots & \boldsymbol{\theta}_\kappa^\intercal \boldsymbol{x} \end{pmatrix}^\intercal$ is the vector of **logits**
- This means we learn a separate set of parameters $\boldsymbol{\theta}_k \in \mathbb{R}^{m+1}$ for each possible class
- All parameter vectors $\boldsymbol{\theta}_k$ are stacked into a single matrix $\boldsymbol{\Theta}$:

$$\boldsymbol{\Theta} = \begin{pmatrix} | & | & \dots & | \\ \boldsymbol{\theta}_1 & \boldsymbol{\theta}_2 & \dots & \boldsymbol{\theta}_\kappa \\ | & | & \dots & | \end{pmatrix} \in \mathbb{R}^{(m+1) \times \kappa} \tag{16}$$

Generalized Cross Entropy Cost Function

- We have to generalize the cross entropy cost function as well:

$$\mathcal{J}(\boldsymbol{\Theta}) = -\sum_{k=1}^{K} y_k \log(g_k(\boldsymbol{z})) \qquad \text{with } \boldsymbol{z} = \begin{pmatrix} \boldsymbol{\theta}_1^\intercal \boldsymbol{x} \\ \vdots \\ \boldsymbol{\theta}_K^\intercal \boldsymbol{x} \end{pmatrix} \tag{17}$$

- This definition of the cross entropy function requires the label to be encoded as a **one-hot vector**, i. e. each label is now a vector:

$$\boldsymbol{y} = \begin{pmatrix} 0 & \dots & 0 & 1 & 0 & \dots & 0 \end{pmatrix}^\intercal \in \{0, 1\}^K \tag{18}$$

Derivative of the Cross Entropy Function

$$\mathcal{J}(\boldsymbol{\Theta}) = -\sum_{k=1}^{K} y_k \log(g_k(\boldsymbol{z})) \qquad \text{with } \boldsymbol{z} = \begin{pmatrix} \boldsymbol{\theta}_1^\intercal \boldsymbol{x} \\ \boldsymbol{\theta}_2^\intercal \boldsymbol{x} \\ \vdots \\ \boldsymbol{\theta}_K^\intercal \boldsymbol{x} \end{pmatrix}$$

$$\frac{\partial}{\partial \theta_{ij}} \mathcal{J}(\boldsymbol{\Theta}) = -\sum_{k=1}^{K} y_k \frac{\partial \log(g_k(\boldsymbol{z}))}{\partial g_k(\boldsymbol{z})} \cdot \frac{\partial g_k(\boldsymbol{z})}{\partial z_i} \cdot \frac{\partial z_i}{\partial \theta_{ij}} \qquad \longrightarrow \text{chain rule}$$

$$= -\sum_{k=1}^{K} y_k \frac{1}{g_k(\boldsymbol{z})} \cdot \frac{\partial g_k(\boldsymbol{z})}{\partial z_i} \cdot \frac{\partial z_i}{\partial \theta_{ij}} \qquad \longrightarrow \frac{d}{dx} \log(x) = \frac{1}{x}$$

Derivative of the Softmax Function

$$g_k(\boldsymbol{z}) = \frac{e^{z_k}}{\sum_{n=1}^{K} e^{z_n}} \qquad \frac{\partial}{\partial z_i} g_k(\boldsymbol{z}) = \begin{cases} g_i(\boldsymbol{z})(1 - g_i(\boldsymbol{z})) & \text{if } i = k \\ -g_k(\boldsymbol{z}) g_i(\boldsymbol{z}) & \text{if } i \neq k \end{cases}$$

Case ❶: $i = k$

$$\begin{aligned} \frac{\partial}{\partial z_i} g_k(\boldsymbol{z}) &= \frac{e^{z_k} \sum_{n=1}^{K} e^{z_n} - e^{z_k} e^{z_i}}{\left(\sum_{n=1}^{K} e^{z_n}\right)^2} \\ &= \frac{e^{z_k}}{\sum_{n=1}^{K} e^{z_n}} \left[1 - \frac{e^{z_i}}{\sum_{n=1}^{K} e^{z_n}}\right] \\ &= g_k(\boldsymbol{z})(1 - g_i(\boldsymbol{z})) \\ &= g_k(\boldsymbol{z})(1 - g_k(\boldsymbol{z})) \end{aligned}$$

Case ❷: $i \neq k$:

$$\begin{aligned} \frac{\partial}{\partial z_i} g_k(\boldsymbol{z}) &= \frac{0 \cdot \sum_{n=1}^{K} e^{z_n} - e^{z_k} e^{z_i}}{\left(\sum_{n=1}^{K} e^{z_n}\right)^2} \\ &= -\frac{e^{z_k}}{\sum_{n=1}^{K} e^{z_n}} \frac{e^{z_i}}{\sum_{n=1}^{K} e^{z_n}} \\ &= -g_k(\boldsymbol{z}) g_i(\boldsymbol{z}) \end{aligned}$$

Derivative of the Cross Entropy Function (Ctd.)

$$\frac{\partial}{\partial \theta_{ij}} \mathcal{J}(\boldsymbol{\Theta}) = -\sum_{k=1}^{K} \frac{y_k}{g_k(\boldsymbol{z})} \cdot \frac{\partial g_k(\boldsymbol{z})}{\partial z_i} \cdot \frac{\partial z_i}{\partial \theta_{ij}}$$

$\longrightarrow$ see slide 33

$$= \left[-\frac{y_k}{g_k(\boldsymbol{z})} g_k(\boldsymbol{z})(1 - g_k(\boldsymbol{z})) + \sum_{\substack{k=1 \\ k \neq i}}^{K} \frac{y_k}{g_k(\boldsymbol{z})} g_k(\boldsymbol{z}) g_i(\boldsymbol{z}) \right] \frac{\partial z_i}{\partial \theta_{ij}}$$

$\longrightarrow$ separate cases

$$= \left[-y_k + y_k g_k(\boldsymbol{z}) + \sum_{\substack{k=1 \\ k \neq i}}^{K} y_k g_i(\boldsymbol{z}) \right] \frac{\partial z_i}{\partial \theta_{ij}} = \left[-y_k + \sum_{k=1}^{K} y_k g_i(\boldsymbol{z}) \right] \frac{\partial z_i}{\partial \theta_{ij}}$$

$\longrightarrow$ cancel terms

$$= (-y_k + g_k(\boldsymbol{z}))x_j = \boxed{(g_k(\boldsymbol{z}) - y_k)x_j}$$

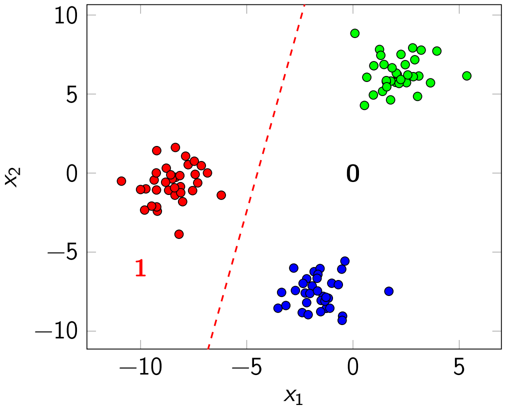
Important

Transforming the Problem into several binary Problems

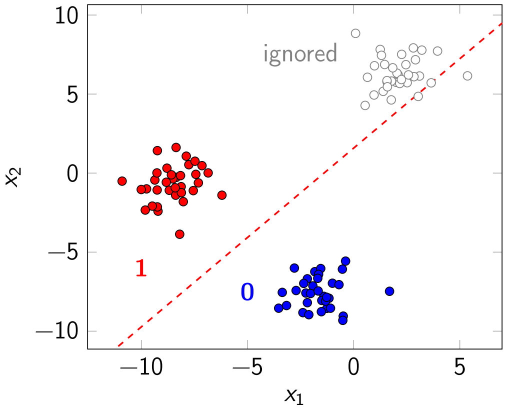
- Instead of adjusting the algorithm we can also **transform the multi-class problem into several binary problems**
- Two common techniques are:
  - One-vs-Rest (OvR) $\rightarrow$ One-against-All
  - One-vs-One (OvO) $\rightarrow$ Pairwise classification
- **General idea:**
  - Several classifiers are 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 $|\mathcal{C}|$ 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 $\mathbf{1}$, all the others to $\mathbf{0}$



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



- **Train one classifier for each pair of classes**
- We get $\binom{|\mathcal{C}|}{2}$ 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

DHBW
Duale Hochschule
Baden-Württemberg

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

Important

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...?

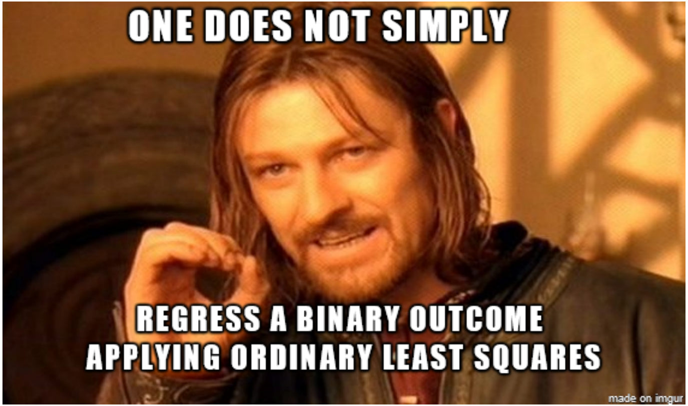
Unit I	Machine Learning Introduction
Unit II	Mathematical Foundations
Unit III	Bayesian Decision Theory
Unit IV	Probability Density Estimation
Unit V	Regression
Unit VI	**Classification I**
Unit VII	Evaluation
Unit VIII	Classification II
Unit IX	Clustering
Unit X	Dimensionality Reduction

Recommended Literature and further Reading I

[1] Pattern Recognition and Machine Learning
Christopher Bishop. Springer. 2006.
→ Link, cf. chapter 4.3.2

[2] Machine Learning: A Probabilistic Perspective
Kevin Murphy. MIT Press. 2012.
→ Link, cf. chapter 8

Meme of the Day



Thank you very much for the attention!

Topic: *** Applied Machine Learning Fundamentals *** Logistic Regression
Term: Winter term 2021/2022

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

Do you have any questions?