Geo Data Science

Logistic Regression

Prof. Dr. Martin Kada

Chair Methods of Geoinformation Science (GIS)
Institute of Geodesy and Geoinformation Science

Technische Universität Berlin

Copyright Notice

Content of this Lecture

Technische Universität Berlin

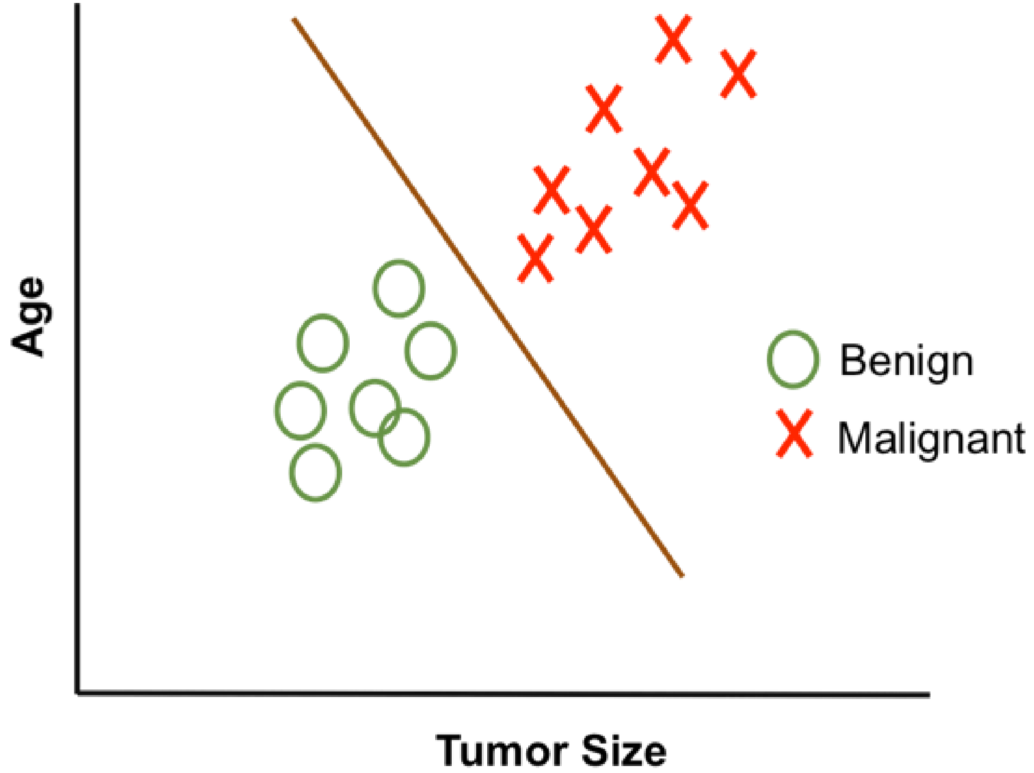
- Linear Regression
- Polynomial Regression
- Regularized Linear Models

regression

- Logistic Regression
- Multinomial Regression

classification: binary, multi-class

Content of this Lecture



- Learn a model by finding a (straight) line that separates the (two) classes
- Predict the class by determining in which region your unseen input dataset lies

Logistic Regression

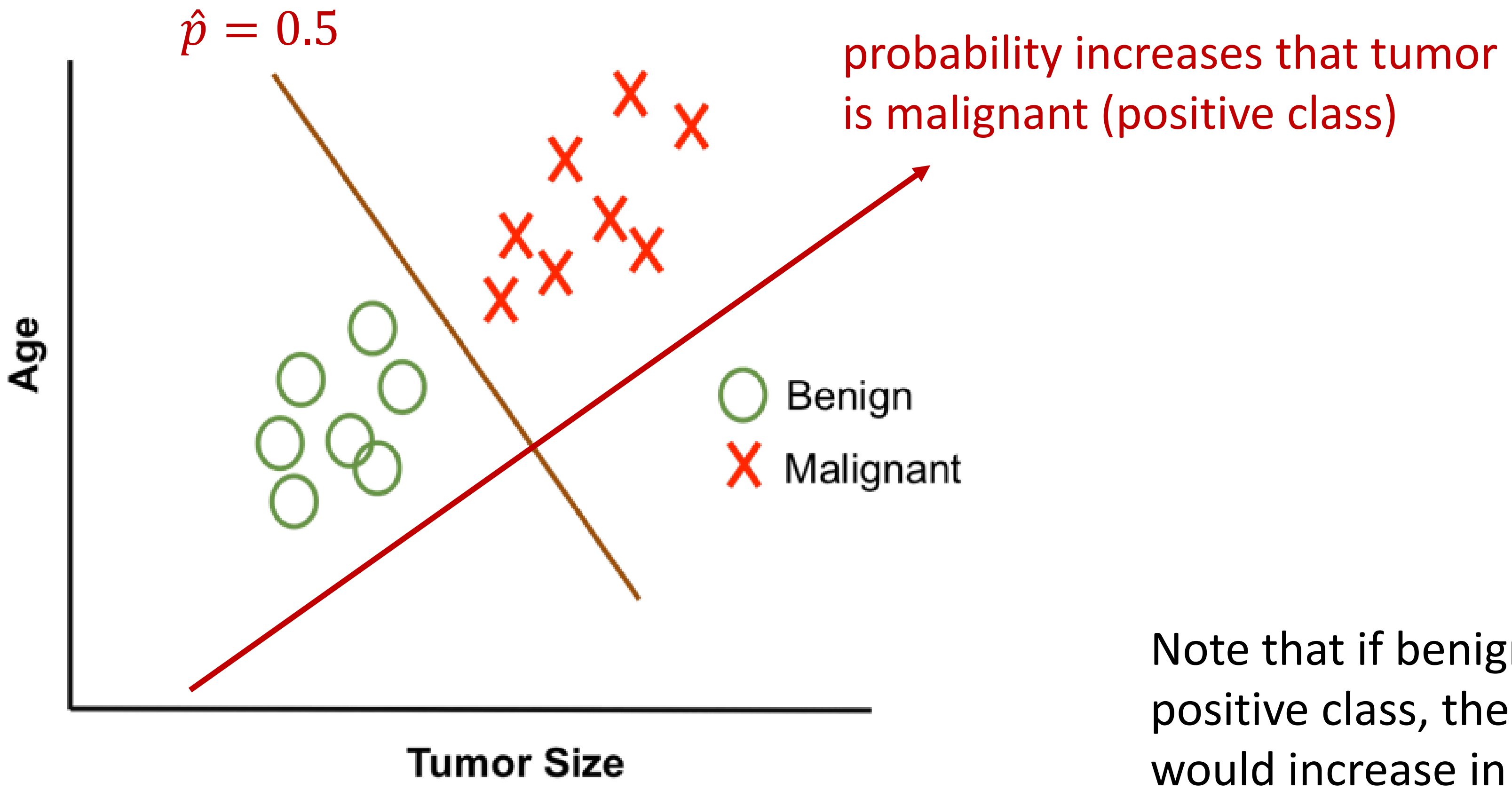
- Logistic Regression (also called Logit Regression) estimates the probability $\hat{p}$ that a data instance belongs to a particular class
  - Estimated probability $\hat{p}$ is greater or equal than 50%
    $\longrightarrow$ instance belongs to this class (positive class, labeled "1")
  - Estimated probability $\hat{p}$ is lesser than 50%
    $\longrightarrow$ instance does not belong to this class (negative class, labeled "0")

$\longrightarrow$ binary classifier

$$\hat{y} = \begin{cases} 1 & \text{if} \quad \hat{p} \geq 0.5 \\ 0 & \text{if} \quad \hat{p} < 0.5 \end{cases}$$

predicted class

Logistic Regression



Note that if benign would be the positive class, then the probabilities would increase in the opposite direction

Estimating Probabilities

- Compute the weighted sum of the input features (plus a bias term) and output the logistic of this result

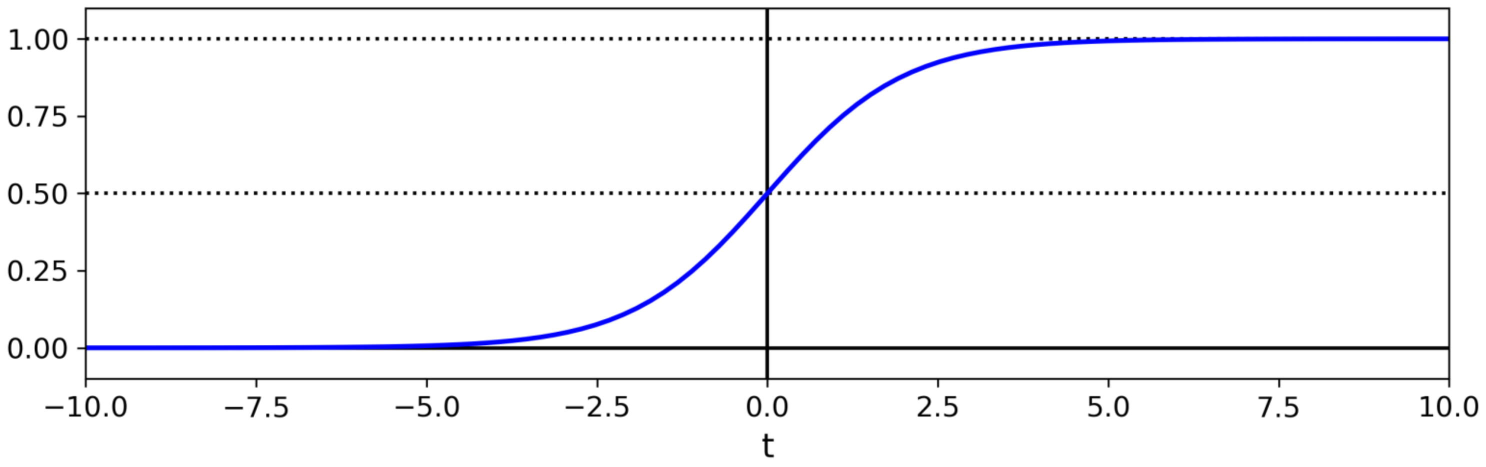
$$\hat{p} = h_\theta(\mathbf{x}) = \underset{\substack{\uparrow \\ \text{logistic} \\ \text{function}}}{\sigma}(\theta^T \cdot \mathbf{x})$$

Estimating Probabilities

Technische Universität Berlin

- Logistic function (also called logic) is a sigmoid function that outputs a number between 0 and 1:

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



Logistic Regression

$$\begin{bmatrix}\theta_0 & \theta_1 & \theta_2 & \dots & \theta_n\end{bmatrix} \begin{bmatrix}1\\ x_1\\ x_2\\ \dots\\ x_n\end{bmatrix} = s \xrightarrow{\sigma(s)} \hat{p} \longrightarrow \hat{y}$$

sometimes called
a score

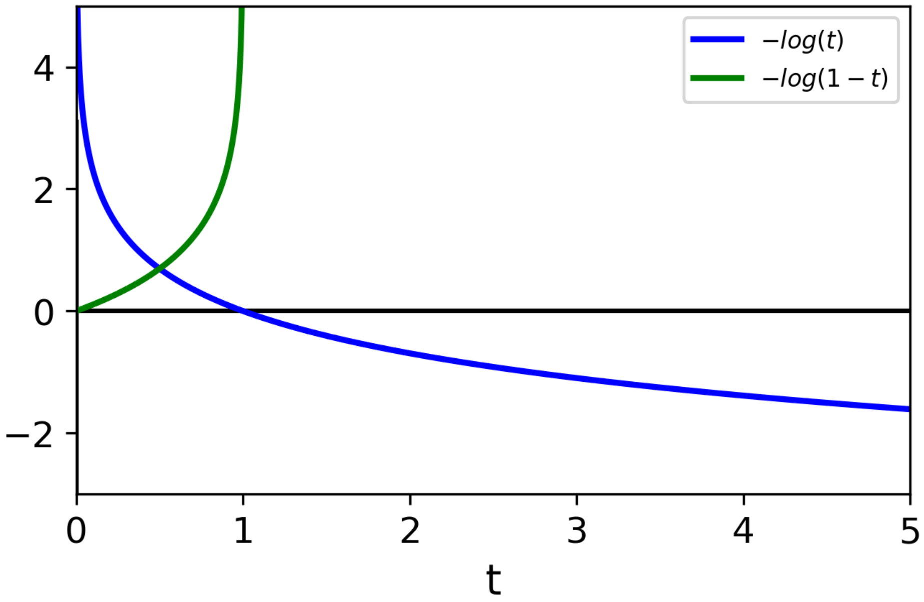
$$\hat{y} = \begin{cases}1 & \text{if } \hat{p} \geq 0.5\\ 0 & \text{if } \hat{p} < 0.5\end{cases}$$

Training and Cost Function

- Learn parameter vector θ so that the model estimates
  - High probabilities for positive instances ($y = 1$)
  - Low probabilities for negative instances ($y = 0$)

- Cost function for a single training instance:

$$c(\theta) = \begin{cases} -\log(\hat{p}) & \text{if } y = 1 \\ -\log(1 - \hat{p}) & \text{if } y = 0 \end{cases}$$



  - By using $-\log(t)$ the cost grows very large if the model estimates a probability close to 0 (or 1) for a positive (or negative) instance
  - $-\log(t)$ is close to 0 for correct estimates

Training and Cost Function

- Cost function over the whole training set (with m data items):

$$J(\theta) = -\underbrace{\frac{1}{m}\sum_{i=1}^{m}}_{\text{average}}\Big[\underbrace{y^{(i)}\log\big(\hat{p}^{(i)}\big)}_{\substack{\text{positive class}\\(y^{(i)}=1)}} + \underbrace{\big(1-y^{(i)}\big)\log\big(1-\hat{p}^{(i)}\big)}_{\substack{\text{negative class}\\(y^{(i)}=0)}}\Big]$$

  - There is no closed-form equation to compute the value of θ that minimizes $J(\theta)$
  - Note that superscript i (e.g. $y^{(i)}$) refers to the i^{th} data item

Training and Cost Function

- Cost function $J(\theta)$ is convex $\longrightarrow$ Gradient Descent finds global optimum

- Partial derivatives of cost function w.r.t. the j^{th} model parameter θ_j:

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left(\sigma\left(\theta^T \cdot \mathbf{x}^{(i)}\right) - y^{(i)} \right) x_j^{(i)}$$

- Compute the gradient vector that contains all the partial derivatives and use it in the Batch Gradient Descent algorithm

Iris Flower Data Set

- 50 samples from each of three species of Iris
  - Four features: length and width of sepals and petals (in centimeters)

Iris versicolor

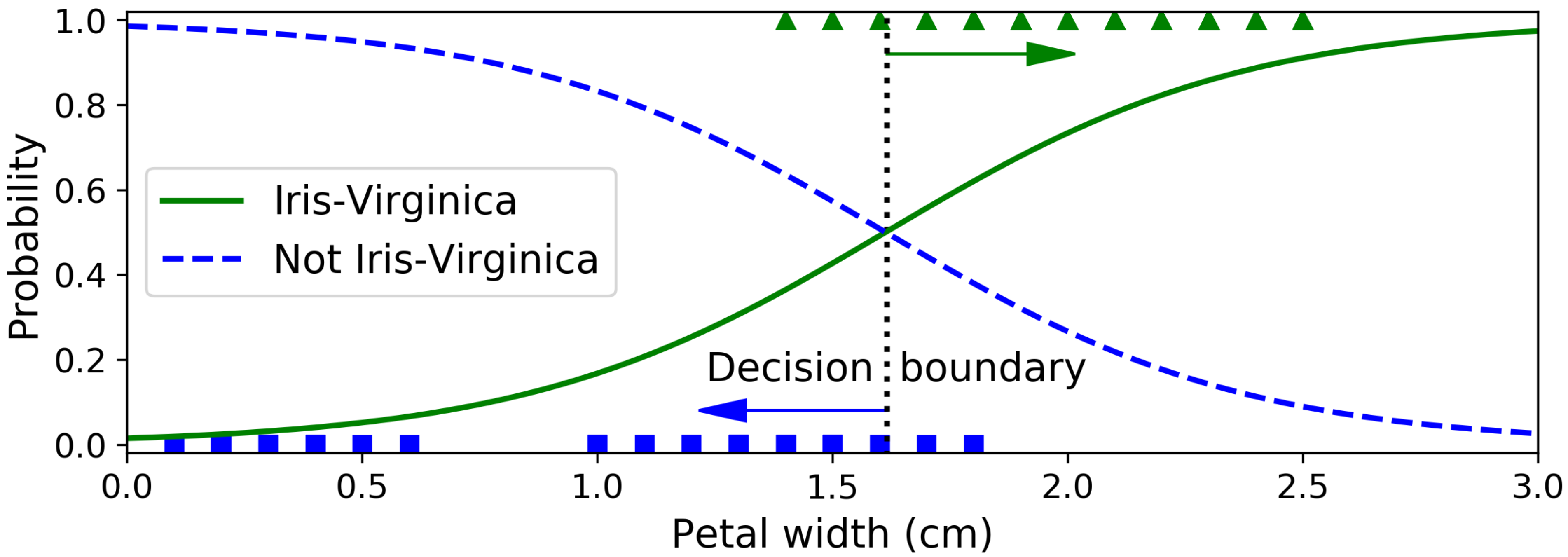
Iris virginica

Iris setosa

  - Ronald Fisher developed 1936 a linear discriminant model to distinguish the species from each other

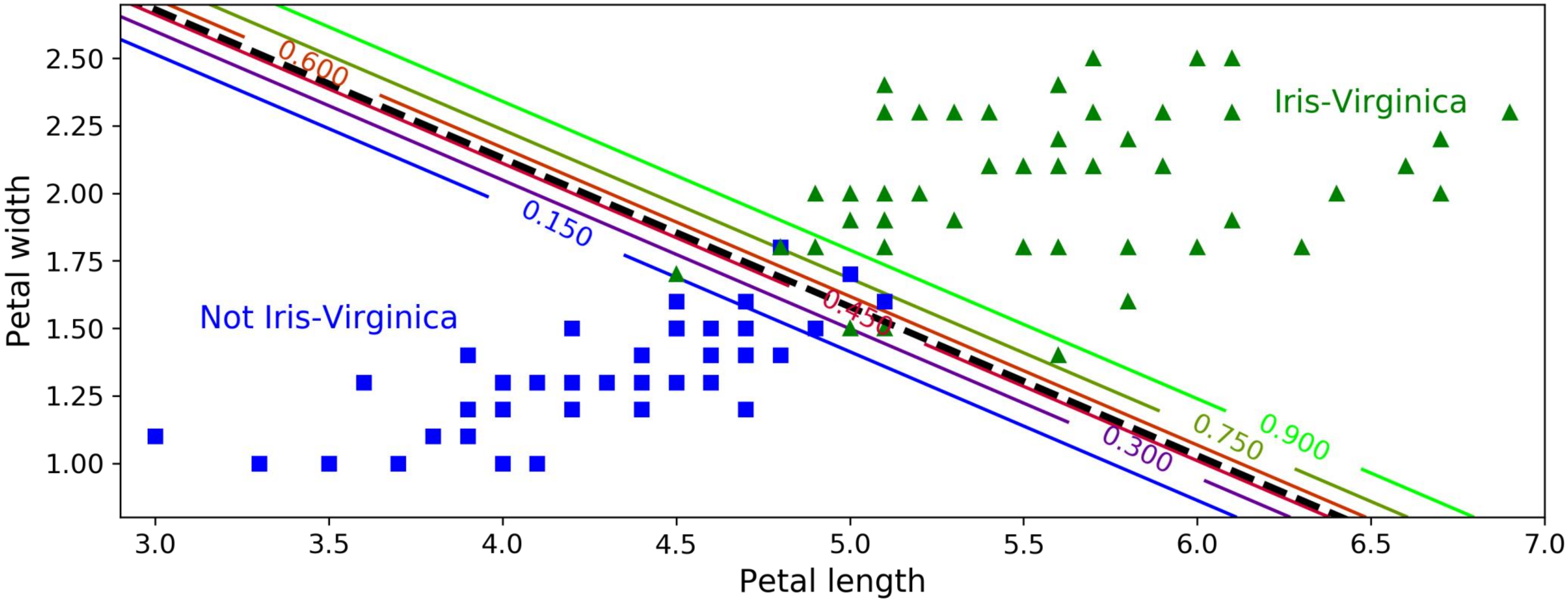
Decision Boundaries

- Decision boundary of model with 1 input feature (petal width)



Linear Decision Boundary

- Decision boundary of model with 2 input features (petal width & length)



Multinomial Logistic Regression

- Multinomial Logistic Regression (also called Softmax Regression)
  - Generalization of Logistic Regression to multiple classes

1. Compute a score $s_k(\mathbf{x})$ for each class k from given feature instance $\mathbf{x}$

$$s_k(\boldsymbol{x}) = \left(\theta^{(k)}\right)^T \cdot \mathbf{x}$$

2. Apply the **softmax function** (also called normalized exponential) to each score to estimate the probability of each class

$$\hat{p}_k = \sigma\big(s(\mathbf{x})\big)_k = \frac{e^{(s_k(\mathbf{x}))}}{\sum_{j=1}^{K} e^{(s_j(\mathbf{x}))}}$$

exponential score

normalization

Multinomial Logistic Regression

- Predicts the class with the highest estimated probability (highest score)

$$\hat{y} = \underset{k}{\operatorname{argmax}}\, \sigma\big(s(\mathbf{x})\big)_k = \underset{k}{\operatorname{argmax}}\, s_k(\mathbf{x}) = \underset{k}{\operatorname{argmax}} \left(\left(\theta^{(k)}\right)^T \cdot \mathbf{x}\right)$$

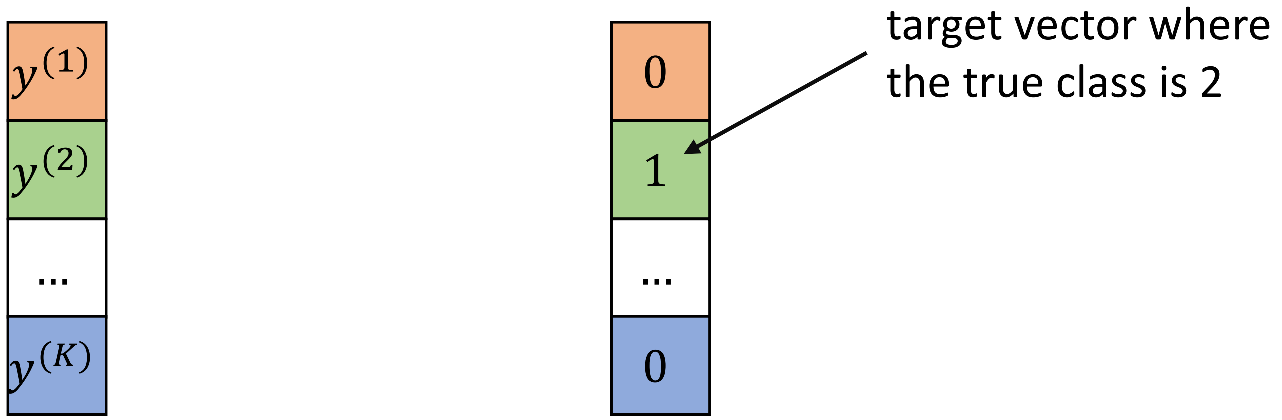
the argmax operator returns the value of the variable k (index) that maximizes the function

Multinomial Logistic Regression

$$\begin{bmatrix} \theta_0^{(1)} & \theta_1^{(1)} & \theta_2^{(1)} & \dots & \theta_n^{(1)} \\ \theta_0^{(2)} & \theta_1^{(2)} & \theta_2^{(2)} & \dots & \theta_n^{(2)} \\ \dots & \dots & \dots & \dots & \dots \\ \theta_0^{(K)} & \theta_1^{(K)} & \theta_2^{(K)} & \dots & \theta_n^{(K)} \end{bmatrix} \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix} = \begin{bmatrix} s^{(1)} \\ s^{(2)} \\ \dots \\ s^{(K)} \end{bmatrix} \xrightarrow{\sigma(s(\mathbf{x}))_k} \begin{bmatrix} \hat{p}^{(1)} \\ \hat{p}^{(2)} \\ \dots \\ \hat{p}^{(K)} \end{bmatrix} \xrightarrow[\underset{k}{\operatorname{argmax}}]{} \boxed{\hat{y}}$$

Multinomial Logistic Regression

- A training instance is typically given in one-hot-encoding, where all values are 0 with the exception of the true class, which is given as 1



Multinomial Logistic Regression

Technische Universität Berlin

- Careful as the meaning of the superscript changes from here on!
- In previous slides, superscript is used for the class, e.g. $\hat{p}^{(2)}$ for the probability of the second class
- In the following slides, the superscript is used for the training instance, e.g. $\hat{p}^{(3)}$ for the data instance 3 (of the training data)
- And the subscript is now used for the class, e.g. $\hat{p}_2^{(3)}$ for the probability of the second class for the data instance 3

Multinomial Logistic Regression

- Objective of training:
  - A model that estimates a high probability for the target (correct) class and low probabilities for the other classes

- **Cross entropy** cost function:

$y_k^{(i)}$ is equal to 1 if the target class for the ith instance is k; otherwise 0

$$J(\Theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log\left(\hat{p}_k^{(i)}\right)$$

  - Measures how well a set of estimated class probabilities match the target class
    - Penalizes the model when it estimates a low probability for the target class
  - Is equivalent to the Logistic Regression's cost function (log loss) for $K = 2$

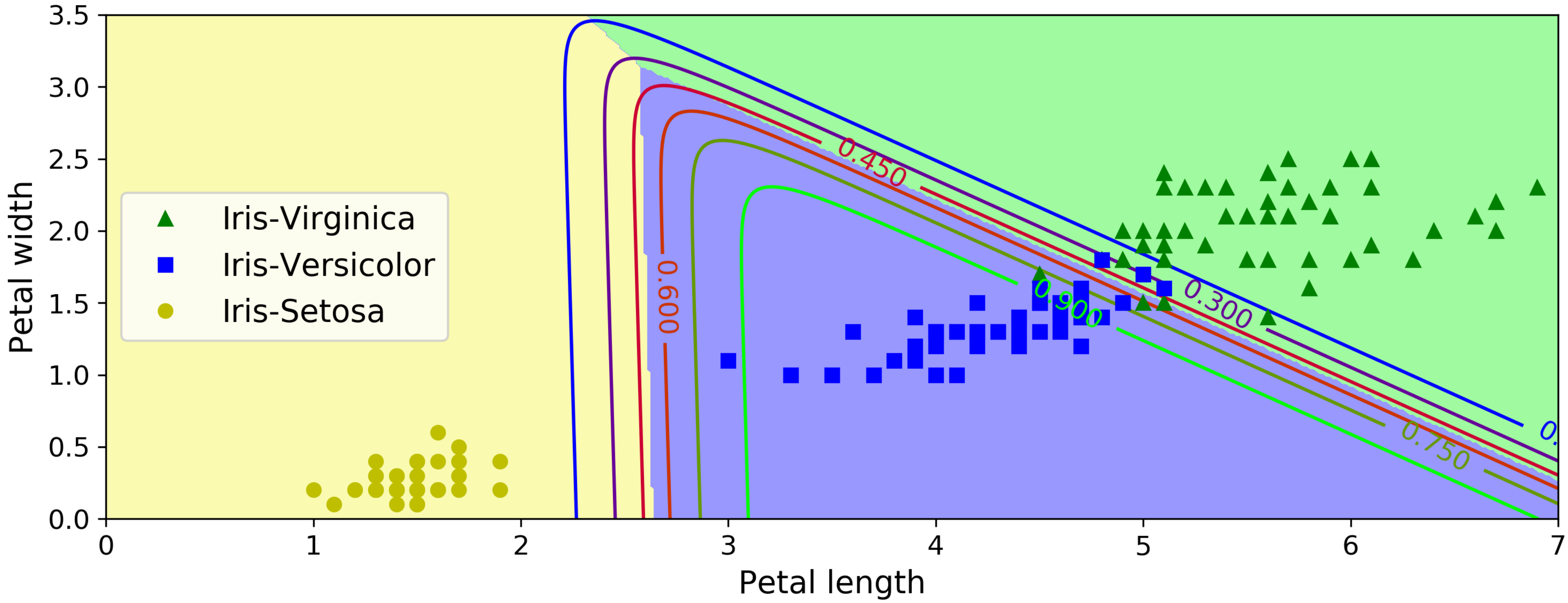
Multinomial Logistic Regression

- Cross entropy gradient vector for class k

$$\nabla_{\theta^{(k)}} J(\Theta) = \frac{1}{m} \sum_{i=1}^{m} \left(\hat{p}_k^{(i)} - y_k^{(i)} \right) \mathbf{x}^{(i)}$$

- Training:
  - Compute the gradient vector for every class, then use Gradient Descent to find the parameter matrix Θ that minimizes the cost function

Softmax Regression Decision Boundaries



Technische Universität Berlin

Thank you for your attention!