1 Preface

1.1 Standardimporte

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

2 Supervised Learning

2.1 Lineare Regression

Optimale Anpassung einer Geraden an eine gegebene Menge an Punkten, d.h. für eine Funktion

$$h_\Theta(x) = \Theta_0 + \Theta_1 x_1 + \cdots + \Theta_n x_n$$

soll der Parametervektor Θ gefunden werden (mit Θ_0 als Konstante), der die Summe der quadrierten Abweichungen der Funktionswerte $h_\Theta(x)$ von den tatsächlichen Werten y minimiert (Methode der kleinsten Quadrate):

$$\min_\Theta L(D, f) = \min \sum_{i=1}^{m} (f(x^{(i)}) - y^{(i)})^2$$
$$\min_\Theta L(D, \Theta) = \min \|X_D\Theta - y_D\|^2$$

wobei X_D eine Matrix mit den Eingabedaten (zzgl. führende 1-Spalte) und y_D der Vektor der tatsächlichen Werte ist:

$$X_D = \begin{pmatrix} 1 & x_1^{(1)} & \dots & x_n^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(m)} & \dots & x_n^{(m)} \end{pmatrix}, \qquad y_D = \begin{pmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{pmatrix}$$

Lokales Minimum = Globales Minimum, da die Kostenfunktion konvex ist. Lösung numerisch oder per *Gradient Descent* $\rightarrow$ ist bei großen Trainingsdatensätzen und/oder vielen Attributen die praktikabelste Methode (s. Skript S. 13: $\nabla_\Theta L(D, \Theta) = 0 \Leftrightarrow (X_D^T X_D)^{-1} X_D^T y_D = \Theta$, wobei inverse von $X_D^T X_D$ sehr rechenaufwändig ist).

Evaluation mittels **Bestimmtheitsmaß** (=normalisierte Variante des quadratischen Fehlers):

$$R^2(D, f) = 1 - \frac{\sum_{i=1}^{m} (f(x^{(i)}) - y^{(i)})^2}{\sum_{i=1}^{m} (y^{(i)} - \bar{y})^2}$$

mit $\bar{y} = \frac{1}{m} \sum_{i=1}^{m} y^{(i)}$, wobei in der Praxis der Durchschnitt mehrerer R^2 berechnet wird (*Kreuzvalidierung*).

- $R^2(D, f)$ ist maximal 1 $\rightarrow$ f modelliert D perfekt
- $R^2(D, f) = 0 \rightarrow$ naives Modell, f sagt stets den Mittelwert $\bar{y}$ voraus
- $R^2(D, f) < 0 \rightarrow$ Modell schlechter als naives Modell
- $R^2(D^{\text{train}}, f)$ sollte relativ nahe an 1 liegen
- $R^2(D^{\text{test}}, f)$ ist üblicherweise kleiner als $R^2(D^{\text{train}}, f)$
- Je näher $R^2(D^{\text{test}}, f)$ an $R^2(D^{\text{train}}, f)$, desto besser ist das Modell generalisiert

2.2 Logistische Regression

Test

2.3 Support Vector Machines

Test

2.4 K-Nearest Neighbours

Test

2.5 Bayes-Klassifikator

Test

2.6 Entscheidungsbäume

Test

3 Unsupervised Learning

3.1 K-Means Clustering

Test

3.2 Hierarchisches Clustering

Test

3.3 Assoziationsregeln

Test

3.4 Anomalieerkennung

Test

3.5 Hauptkomponentenanalyse/Principal Component Analysis (PCA)

Test

4 Reinforcement Learning

4.1 Markov-Entscheidungsprozesse

Test

4.2 Passives Reinforcement-Learning

Test

4.3 Aktives Reinforcement-Learning

Test

5 Deep Learning

5.1 Künstliche Neuronale Netze

Test

5.2 Convolutional Neutral Networks

Test

5.3 Recurrent Neutral Networks

Test

5.4 Recurrent Neutral Networks

Test