

Contents

1	Machine Learning Grundlagen	1
2	Data Quality Assessment	1
3	Machine Learning Fundamentals	2
4	Supervised Learning Basics	2
5	Linear Regression	2
6	Gradient Descent	3
7	Logistische Regression (eigentlich Klassifikation)	3

1 Machine Learning Grundlagen

Disciplines in Machine Learning

1. **Supervised Learning**
- The algorithm is given **labeled training data**

- The algorithm learns to predict the label of yet unseen examples
2. **Unsupervised Learning**
- The algorithm is given **unlabeled data**

- The algorithm detects and exploits the inherent structure of the data
3. **Semi-Supervised Learning**
- A mixture of supervised and unsupervised machine learning techniques

- Usually there is only very limited labeled data available
4. **Reinforcement Learning**
- No data available but the algorithm is guided by a **reward function**

- It searches the ideal behavior that maximizes the agent's reward

Identifying target groups for marketing campaigns using clustering techniques	passt zu	Unsupervised Machine Learning
Calculating product recommendations with collaborative filtering techniques	passt zu	Supervised Machine Learning
Market basket analysis using association rules	passt zu	Unsupervised Machine Learning
Medical image analysis for detection of skin diseases based on human expert markings	passt zu	Supervised Machine Learning
Search query analysis for e-commerce by semantical clustering	passt zu	Unsupervised Machine Learning
Prediction of selling prices for the real estate market	passt zu	Supervised Machine Learning
Dimensionality reduction for data visualization	passt zu	Unsupervised Machine Learning
Learning to play Jass by self-play	passt zu	Reinforcement Machine Learning
Detecting animals on high-resolution photographs	passt zu	Supervised Machine Learning
Identifying most-valuable customers on e-commerce platforms using transactions and tracking data	passt zu	Unsupervised Machine Learning

2 Data Quality Assessment

1. Data Cleaning¹

- (a) Dublizierte Daten erkennen und entfernen
- (b) Daten mit nullen können ersetzt werden.

¹Auch wenn die Datenqualität selbständig verbessert werden kann sollten: alle Änderungen dokumentiert werden, data-repository mit versionierung verwendet werden, den Herausgeber der Daten auf fehler in den Daten hinweisen

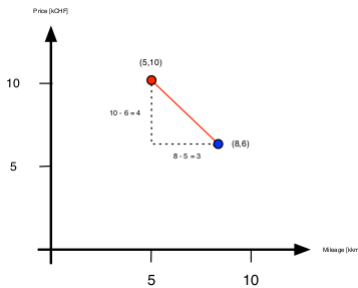
- (c) Daten Machine Learning freundlicher gestalten (z.B. für Farben eigene Zeilen erstellen, damit die Euklid-Distanz gerechnet werden kann.

2. Analyse mit Hilfe von

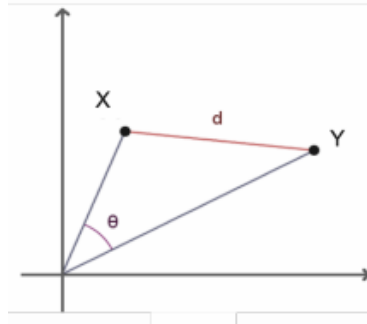
- (a) 5 Nummer Zusammenfassung (median Q2, Quartile Q1 und Q3 sowie min und max)
- (b) Boxplots um das Datenset auf Ausreisser zu prüfen.
- (c) Varianz und Standardabweichung berechnen

3 Machine Learning Fundamentals

Euklid Distanz



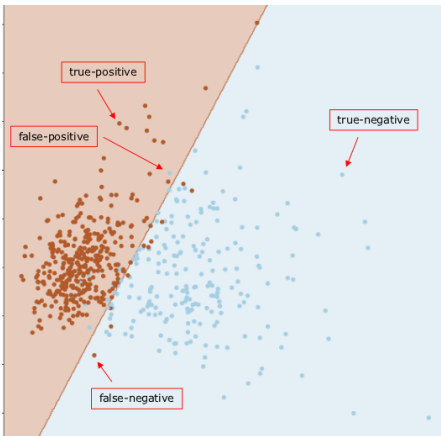
Kosinus Ähnlichkeit



Formel Kosinus Similarity

$$\begin{aligned} \text{sim}(X, Y) &= \frac{\langle X, Y \rangle}{\|X\| \|Y\|} \\ &= \frac{\sum_{i=1}^n x_i y_i}{\sqrt{\sum_{i=1}^n x_i^2} \sqrt{\sum_{i=1}^n y_i^2}} \\ \text{dist}(X, Y) &= 1 - \text{sim}(X, Y) \end{aligned}$$

4 Supervised Learning Basics



$$\text{Accuracy} = \frac{TP+TN}{\text{Total}}$$

$$\text{Errorrate} = \frac{FP+FN}{\text{Total}}$$

$$\text{Sensitivity} = \frac{TP}{\text{ActualYes}} = \frac{TP}{TP+FN}$$

$$\text{Specificity} = \frac{TN}{\text{ActualNo}} = \frac{TN}{TN+FP}$$

$$\text{Precision} = \frac{TP}{\text{PredictedYes}} = \frac{TP}{TP+FP}$$

5 Linear Regression

Das Modell hat generell die folgende Form: $y = h_{\theta}(x) = \theta_0 + \theta_1 x$.

Mit \bar{x} und \bar{y} als Mittelwerte der Datenreihe, können somit die Werte θ_1 und θ_0 berechnet werden.²

$$\theta_1 = \frac{\sum_{i=1}^n (y^{(i)} - \bar{y})(x^{(i)} - \bar{x})}{\sum_{i=1}^n (x^{(i)} - \bar{x})^2} = \frac{S_{xy}}{S_{xx}} \quad \theta_0 = \bar{y} - \theta_1 \bar{x}$$

6 Gradient Descent

Mit der Kostenfunktion

$$J(\theta) = \frac{1}{2n} \sum_{i=1}^n (h(\theta, x^{(i)}) - y^{(i)})^2$$

wo $h(\theta, x^{(i)})$ die Vorhersage von y ist und als

$$h(\theta, x^{(i)}) = (x^{(i)})^T \theta = \theta_0 + \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} + \dots + \theta_m x_m^{(i)}$$

ausgeschrieben wird, kann θ optimiert werden. Diese wird mit $\theta = (X^T X)^{-1} X^T y$ umgesetzt. In python wird das mit X als

$n \times m$ Matrix, bei der die erste Spalte mit Einsen aufgefüllt wurde und mit y als Zielwert $\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \dots \\ \theta_m \end{bmatrix}$ definiert wird.

```
theta = np.linalg.inv(X.T.dot(X)).dot(X.T).dot(y)
```

Korrelation liegt immer zwischen $-1 \leq r \leq 1$. $r = 0$ bedeutet keine Korrelation und $|r| = 1$ vollständige Korrelation.

7 Logistische Regression (eigentlich Klassifikation)

Logistische Regression zielt darauf ab eine binäre Zuordnung vorzunehmen (z.B. Brustkrebs oder nicht; Spam-Mail oder nicht etc.). Dabei können die unabhängigen Variablen numerisch (12mm) oder kategorisch (mag Skifahren) sein.

Die Logistische Funktion nennt sich auch "Sigmoid Funktion" und lautet wie folgt:

$$\sigma(z) = \frac{1}{1 + e^{-z}} = \frac{e^z}{e^z + 1}, z \in \mathbb{R}$$

Die Logistische Funktion ist auch die neue Hypothese:

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

Die Ableitungen der Logistischen Funktion sehen folgendermassen aus.

$$\sigma'(x) = \sigma(z)(1 - \sigma(z))$$

²Bei $var^{(i)}$ ist i ein Index für den Datenpunkt und kein Exponent.

$$\sigma''(z) = \sigma(z)(1 - \sigma(z))(1 - 2\sigma(z))$$

Wenn $z > 0$ wird die Aussage als wahr (Wahrscheinlichkeit über 0.5) und sonst als falsch interpretiert.

z ist mit $z = \theta^T x = \theta_0 + \theta_1 x_1 + \theta_2 x_2$ gegeben.

Mit der Logistischen Regression muss auch eine neue Kostenfunktion gewählt werden, welche folgendermassen aussieht:

$$Cost(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{wenn } y = 1 \\ -\log(1 - h_\theta(x)) & \text{wenn } y = 0 \end{cases}$$

Damit ergibt sich die Formel für Gradient Descent wie folgt:

$$\theta_{k+1} = \theta_k - \alpha \frac{1}{n} X^T (\sigma(X\theta_k) - y)$$

Codebeispiel zum trainieren von θ

```
def sigmoid(z):
    return 1/(1+np.exp(-z))

def cost_function(X, y, theta):
    y_hat = sigmoid(np.dot(X, theta))
    J_i = -y*np.log(y_hat)-(1-y)*np.log(1-y_hat)
    J = J_i.sum()/len(y)
    return J

def update_theta(X,y, theta, alpha):
    theta -= alpha * np.dot(X.T, sigmoid(np.dot(X, theta))-y)/len(y)
    return theta

def train(X, y, theta, alpha, kmax):
    cost_history=[]
    for i in range(kmax):
        theta=update_theta(X,y, theta, alpha)
        cost = cost_function(X,y, theta)
        cost_history.append(cost)
```

```
return theta, cost_history
```

Danach wird zuerst ein θ definiert, wo die vorgegebenen startwerte angegeben werden. Daraus trainiert der Algorithmus danach mit α als Schrittgrösse und **kmax** als Anzahl Zyklen die optimalen Werte für θ .