Datenanalyse

Deskriptive Statistik

Statistik wird üblicherweise in drei (nicht disjunkte) Bereiche eingeteilt:

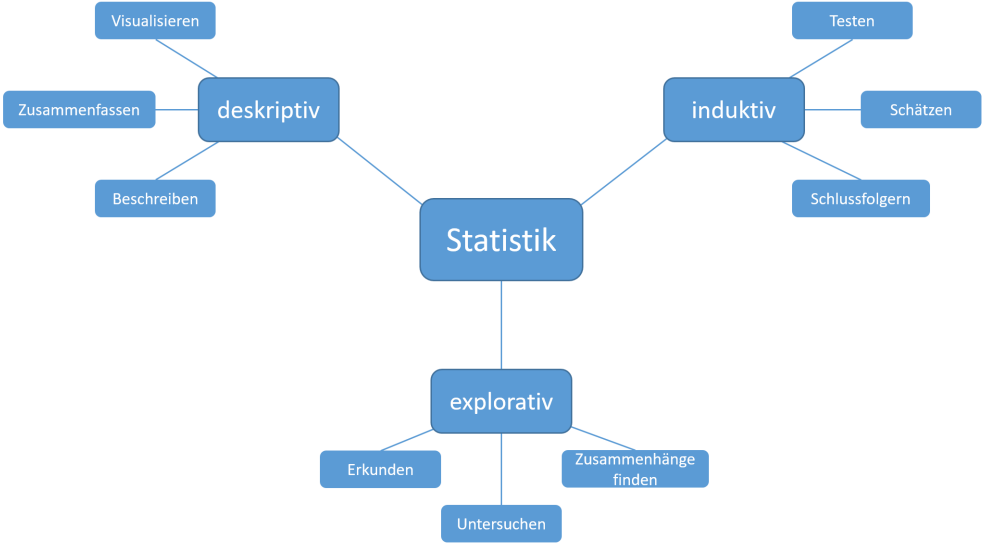
1. Deskriptive (beschreibende) Statistik:
   - Zusammenführung und Aufbereitung der Daten in Tabellen (sofern sinnvoll)
   - Visuelle Aufbereitung der Verteilung(en) der Daten durch Diagramme
   - Beschreibung der Daten durch Kennzahlen, wie Lagemaße, Streuungsmaße und Zusammenhangsmaße
   - Ziel: Beschreibung der Verteilungen von Merkmalen
   - Es sollen immer adäquate Methoden/Darstellungen verwendet werden und nicht-betrachtete Zusammenhänge/Informationen stets aufgeführt werden (Bsp.: Bei einem Streudiagramm werden mehrfache gleiche Punkte nicht dargestellt)

Induktive Statistik

2. Induktive (schließende) Statistik:
   - Induktion allgemein: Schließen vom Speziellen aufs Allgemeine
   - Ableiten von Eigenschaften der Grundgesamtheit aus einer Stichprobe
   - Grundlage bildet die Wahrscheinlichkeitstheorie
   - Eins der Hauptgebiete ist die Schätztheorie, z.B. Schätzen von Parametern (etwa Maximum-Likelihood-Schätzer) oder Schätzen von Konfidenzintervallen
   - Durchführung von Hypothesentests (Überprüfen einer Hypothese anhand von Beobachtungen)

Explorative Statistik

3. Explorative (erkundende) Statistik:
   - Aufdecken möglicher Zusammenhänge innerhalb von gegebenen Daten
   - Meist sind in den zugrunde liegenden Daten wenige Zusammenhänge bekannt
   - Formulierung und statistische Analyse/Bewertung von Hypothesen
   - Verbindet deskriptive und induktive Statistik
   - Ziel ist hier neue Erkenntnisse aus den Daten zu gewinnen
   - Starke Überschneidung mit Gebieten wie Data-Mining, KDD,...
   - Achtung: p-Hacking ist der Missbrauch von Data-Mining zur Verzerrung von (Forschungs-)Daten um irgendein statistisch signifikantes Forschungsergebnis zu publizieren


Visualisieren
Zusammenfassen
Beschreiben
deskriptiv
Testen
Schätzen
Schlussfolgern
induktiv
Statistik
explorativ
Erkunden
Untersuchen
Zusammenhänge finden

Wissensentdeckung in Datenbanken

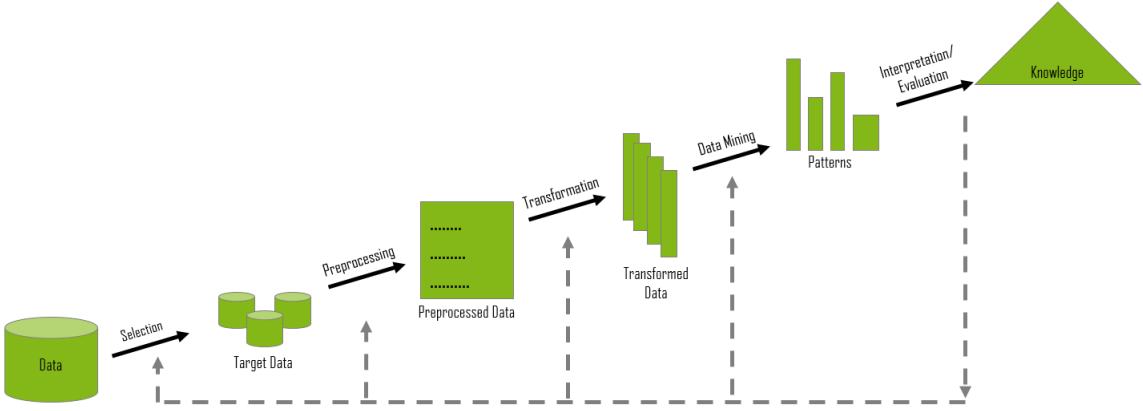
Definition: Wissensentdeckung in Datenbanken

Wissensentdeckung in Datenbanken (engl. *Knowledge Discovery in Databases*, kurz KDD) ist der nicht-triviale Prozess der Identifikation gültiger, bisher unbekannter, potentiell nützlicher und letztendlich verständlicher Muster in Daten.[1]

Anmerkungen:

- *Daten* sind hier eine Menge von Fakten
- *nicht-trivial* bedeutet hier, dass komplexere Vorgänge wie Suche oder Folgerungen (Inferenz) in Abgrenzung zu einfachen Datenbankabfragen angewendet werden
- das erlernte Wissen soll *gültig* im Sinne der Statistik sein
- *Muster* wird in dieser Definition allgemein und eher im Sinne von Wissen aufgefasst

KDD Vorgehensmodell nach Fayyad et. al.



Phasen des KDD-Vorgehensmodells

Die einzelnen Phasen bestehen aus folgenden Inhalten:

0. Zielsetzung:
   - Verständnis für Anwendungsgebiet aufbauen
   - Bestehendes Wissen zusammentragen
   - Ziel aus Sicht des Kunden entwickeln
1. Auswahl: Für das Ziel relevante Daten auswählen
2. Vorverarbeitung:
   - Entfernung von Rauschen (sofern sinnvoll) oder Informationssuche für Modellierung von Rauschen
   - Untersuchung von Ausreißern
   - Strategie für fehlende Werte festlegen
   - Betrachtung von zeitlichen Informationen und bekannten Veränderungen
3. Transformation:
   - Auswahl von geeigneten Merkmalen und gegebenenfalls Generierung neuer Merkmale
   - Gegebenenfalls Reduktion der Dimension
   - Auswahl geeigneter Datentypen

4. Data Mining:
   - Bestimmung der geeigneten Data Mining Methode(n) unter Berücksichtigung des formulierten Ziels aus Schritt 0 (Klassifikation, Regression, Clustern,...)
   - Explorative Analyse (Regressionsanalysen, Visualisierungen, ...)
   - Auswahl geeigneter Algorithmen
   - Ausführen der Algorithmen zur Mustererkennung
5. Evaluation:
   - Interpretation der bisherigen Ergebnisse (gegebenenfalls Wiederholung der Schritte 0 bis 4)
   - Visualisierung der Ergebnisse
   - Ergebniserstellung

CRISP-DM

Abbildung: Vorgehensmodell nach CRISP-DM

Maschinelles Lernen

Es gibt viele verschiedene Definitionen und Beschreibungen des Gebiets des maschinellen Lernens. Zwei bekannte sind etwa:

Maschinelles Lernen

- „[Machine learning is] the field of study that gives computers the ability to learn without explicity programmed“ -Arthur Samuel


Daten
Regeln
Klassische Programmierung
Antworten
Daten
Antworten
Maschinelles Lernen
Regeln

Arten des maschinellen Lernens

Die meisten Aufgaben des maschinellen Lernens lassen sich in die folgenden drei Kategorien einordnen:

1. Überwachtes Lernen (engl. supervised learning):
   - Eingabewerte x und Ausgabewerte y sind für historische Beispiele bekannt
   - Ziel: Funktion f zu erlernen mit $f(x) = y$
   - Für neue Eingabewerte kann somit der Ausgabewert geschätzt bzw. prognostiziert werden
2. Unüberwachtes Lernen (engl. unsupervised learning):
   - Hier existieren nur Eingabewerte x, **keine** Ausgabewerte
   - Ziel: (unbekannte) Strukturen innerhalb der Daten entdecken
3. Verstärkendes Lernen (engl. reinforcement learning):
   - Ein (Software-) Agent kann aus mehreren Aktionen auswählen
   - Der Agent befindet sich in einer Umgebung mit der er interagieren kann
   - Jede gewählte Aktion überführt den Agenten in einen neuen Status innerhalb der Umgebung
   - Der Agent erhält für jede Aktion eine Auszahlung bzw. Belohnung
   - Ziel des Agenten ist die erhaltene (Gesamt-) Auszahlung zu maximieren

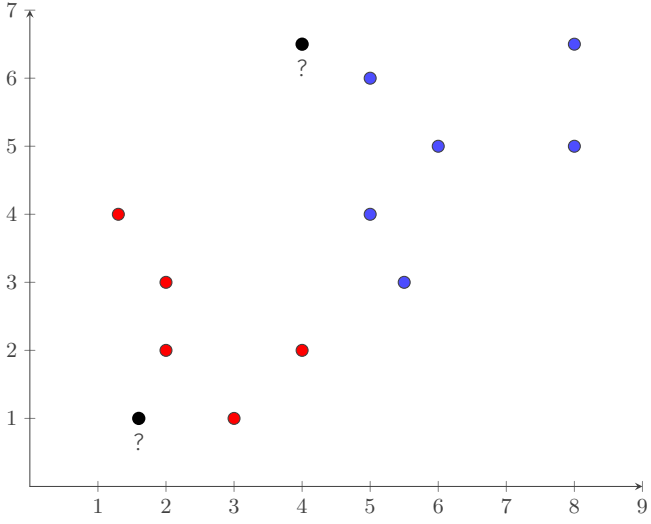
Aufgabenarten des maschinellen Lernens

Die meisten Probleme, die mit maschinellem Lernen gelöst werden sollen, lassen sich in folgende Aufgaben einteilen:

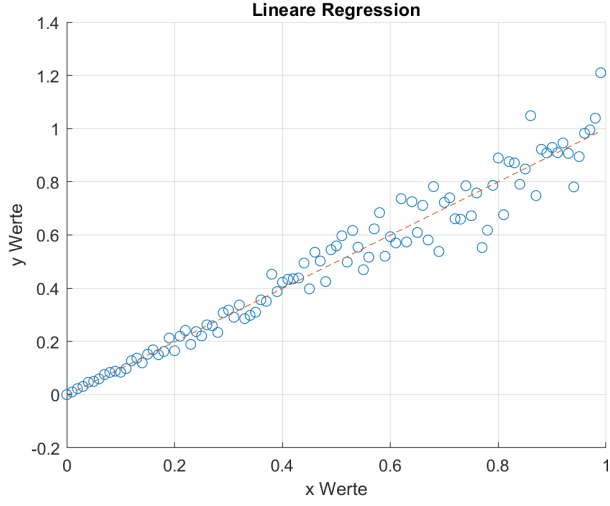
Aufgaben und Ziele für maschinelles Lernen

- Klassifikation: Gegeben seien n verschiedene Klassen. Jedes Objekte gehört einer dieser Klassen an. Für Objekte soll anhand ihrer Eigenschaften (features) die entsprechende Klasse prognostiziert werden. Die Zielgröße der Prognose ist hier eine nominale bzw. diskrete Größe. Insbesondere ist die wahre Klasse der Trainingsdaten bekannt.
- Regression: Soll ein stetiger Wert prognostiziert werden, spricht man von einer Regressionsaufgabe. Der wahre Wert der Zielgröße ist für die Trainingsdaten bekannt.
- Clusteranalyse: Verschiedene Objekte sollen zu Gruppen zusammengefasst werden, sodass die Objekte einer Gruppe möglichst ähnlich zueinander sind, und die Unterschiede zu den übrigen Gruppen möglichst groß sind. Hier existieren keine wahren Werte.
- Dimensionsreduktion: Hochdimensionale Daten sollen in ihrer Dimension reduziert werden, sodass die Struktur möglichst erhalten bleibt.

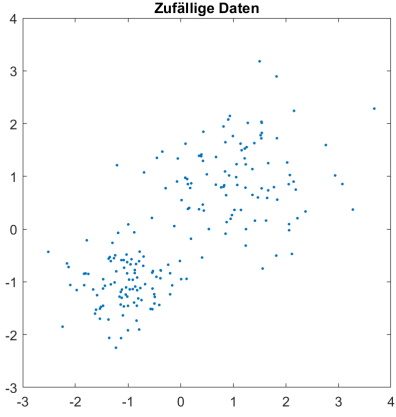
Klassifikation

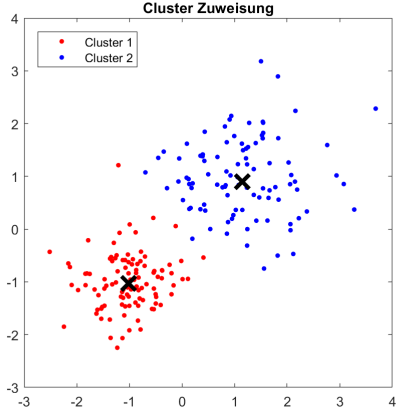


Regression


Lineare Regression
y Werte
x Werte
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
0.2
0.4
0.6
0.8
1

Clusteranalyse





Die Lineare Regression

Als ein erstes, einfaches und bekanntes Verfahren, soll hier die lineare Regression behandelt werden.

- Dazu seien folgende Beispielwerte gegeben:

Eingabe x	Ausgabe y
0	0
1	0
2	1.5
3	2

- Wir wollen den Zusammenhang zwischen x und y durch eine lineare Funktion modellieren, d.h.

$$h_\theta(x) = \theta_0 + \theta_1 \cdot x = \hat{y}$$

- h_θ ist die sog. Hypothese (mit Parameter $\theta = (\theta_0, \theta_1)$
- $\hat{y}$ ist der geschätzte Wert von y (auf Grundlage von h_θ)
- Eine mögliche Hypothese ist etwa $h_\theta(x) = 1 - x$

- Die Hypothese $h_\theta(x) = 1 - x$ führt zu folgenden Prognosen für y:

Eingabe x	Prognose $\hat{y}$	Wahre Ausgabe y	Differenz $\lvert y - \hat{y} \rvert$
0	1	0	1
1	0	0	0
2	-1	1.5	2.5
3	-2	2	4

- Offensichtlich liefert diese Hypothese keine annehmbaren Prognosen, da die Fehler groß sind
- Gesucht sind also Werte für θ_0 und θ_1 die „am besten“ zu den Daten passen
- Zunächst muss festgelegt werden, wie der Abstand zwischen der Hypothese und den wahren Ausgaben gemessen werden soll
- Dafür verwenden wir eine sogenannte Kostenfunktion L
- Für die lineare Regression ist der mittlere quadratische Fehler aus theoretischen Gründen gut geeignet:

$$L(\theta_0, \theta_1) = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2$$

- Lösen des Optimierungsproblems

$$\min_{\theta_0,\theta_1} L(\theta_0, \theta_1)$$

liefert also die „besten“ Parameter, in dem Sinne, dass hier ein minimaler Fehler bezüglich der gewählten Kostenfunktion und der gewählten Hypothese (hier linear) erzeugt wird

- Für den mittleren quadratischen Fehler als Kostenfunktion, kann im Falle der linearen Regression eine analytische Lösung angegeben werden[10]

$$\theta_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x - \bar{x})^2}$$

$$\theta_0 = \bar{y} - \theta_1 \cdot \bar{x}$$

- Alternativ kann das obige Minimierungsproblem mit Methoden der Optimierung (häufig Gradientenabstieg) gelößt werden

[10] Unter bestimmten formalen Annahmen, wie etwa standardnormalverteilten Fehlertermen

- Doch Vorsicht: Es sollten **nie** alle Daten zum bestimmen der Parameter verwendet werden
- Andernfalls kann die Genauigkeit der Hypothese nicht evaluiert werden
- Und die so gefundenen Parameter sind meistens zu stark an die Daten angepasst
- Das Ermitteln der Parameter wird *Training* genannt
- Um die Hypothese beurteilen zu können, werden die Daten in ein Trainings- und Testdatensatz aufgeteilt

Folgende Notation wird von nun an verwendet:

- N ist die Anzahl aller Datenpunkte bzw. Beispiele, K ist die Anzahl der Attribute eines jeden Beispiels
- $x_{i,k}$ ist der Wert des i-ten Beispiels/Datenpunktes vom k-ten Attribut
- X_i ist der Vektor $(x_{i,1}, \ldots, x_{i,N})$
- X ist die Matrix

$$\begin{pmatrix} 1 & x_{1,1} & x_{1,2} & \cdots & x_{1,K} \\ 1 & x_{2,1} & x_{2,2} & \cdots & x_{2,K} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N,1} & x_{N,2} & \cdots & x_{N,K} \end{pmatrix}$$

- y_i ist der Zielwert des i-ten Beispiels
- Y ist der Vektor $(y_1, \ldots, y_N)$

Residuenplots

Eine weitere, sehr hilfreiche Möglichkeit, das gelernte Modell zu evaluieren ist ein Plot der Residuen.

- Ein Residuum ist die Differenz von wahrem und prognostiziertem Wert, d.h.

$$r = y - \hat{y}$$

- Anschließend wird die Verteilung der Residuen dargestellt und der Verlauf der Residuen gezeichnet
- Im Idealfall liegen die meisten Residuen nahe der 0 und es gibt nur wenige Ausreißer, optisch erinnert die Verteilung der Residuen einer Gaußglocke
- Oftmals lassen sich systematische Fehler feststellen, wie
  - viel mehr positive als negative Residuen,
  - periodischer Wechsel zwischen positiven und negativen Residuen,
  - Gleichverteilung der Residuen,
  - ...

Trainings- und Testmenge

Um die Performance eines Klassifikators oder eine Regression zu evaluieren, wird die Datenmenge in eine Trainings- und in eine Testmenge aufgeteilt

- Seien $(X_1, y_1), (X_2, y_2), \ldots, (X_{n+m}, y_{n+m})$ alle Datenpunkte/Instanzen
- Die Instanzen $(X_1, y_1), \ldots, (X_n, y_n)$ werden für das Trainieren, also das Spezifizieren des Klassifikators verwendet
- Die übrigen m Instanzen $(X_{n+1}, y_{n+1}), \ldots, (X_{n+m}, y_{m+n})$ werden **nicht** für das Trainieren verwendet sondern bilden die Testmenge
- Um die Performance des Modells zu evaluieren, wird das Modell auf die Testmenge angewendet, also Instanzen die das Modell bisher nicht gesehen hat
- Da die Lösungen y_i bekannt sind, kann so das Modell geprüft werden
- Die Aufteilung in Trainings- und Testmenge sollte immer zufällig durchgeführt werden (außer bei zeitabhängigen Instanzen!)
- Üblicherweise werden 60% der Instanzen für das Trainieren und 40% für das Testen verwendet
- Bei sehr großen Datenmengen können auch 90% für das Trainieren und die übrigen 10% für das Testen verwendet werden

Validierungsdatensatz

- Der Trainingsfehler $L_{Training}$ist der Fehler nach der Kostenfunktion bezüglich der Trainingsmenge
- Der Testfehler L_{Test} ist der Fehler nach der Kostenfunktion bezüglich der Testmenge
- Sollen Hyperparameter, also Freiheitsgrade des Modells, oder verschiedene Hypothesen verglichen werden, so darf das **nicht** mittels des Testfehlers geschehen
- Statt die Daten nur in Trainings- und Testdaten aufzuteilen, werden die Daten in Trainings-, Validierungs- und Testdaten aufgeteilt
- Eine übliche Aufteilung ist 60% Trainings-, 20% Validierungs- und 20% Testdaten
- Der Validierungsfehler L_{Val} ist der Fehler nach der Kostenfunktion bezüglich der Validierungsmenge
- Der Vergleich verschiedener Hyperparameter oder Hypothesen erfolgt dann über den Validierungsfehler

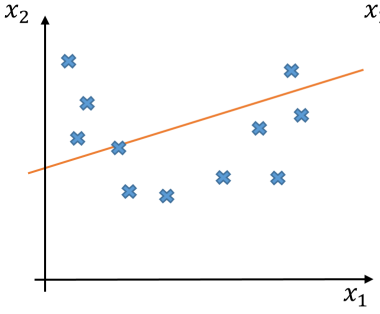
Über- und Unteranpassung

Ein auf maschinellem Lernen basierendes Modell muss gut evaluiert werden. Zwei häufige Probleme sind Über- und Unteranpassung.

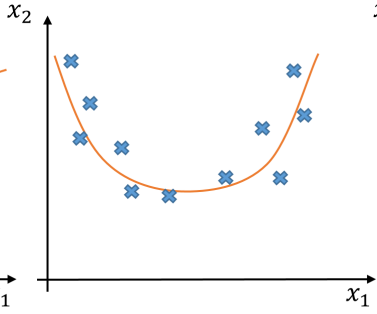
Über- und Unteranpassung

- Überanpassung (engl. overfitting): Ein Modell ist *überangepasst*, wenn es zu stark mit den Daten zusammenhängt und somit auch das Rauschen der Daten mitberücksichtigt. Das Modell enthält zu viele Freiheitsgrade, die nicht durch die Daten begründet werden können.
- Unteranpassung (engl. underfitting): Ein Modell ist *unterangepasst*, wenn es den Trend der zugrundeliegenden Daten nicht passend abbilden kann und die Daten nicht ausreichend beachtet. Dies kann etwa an zu wenigen Daten, oder einem nicht passenden Modell liegen (etwa ein lineares Modell für einen nicht-linearen Verlauf)

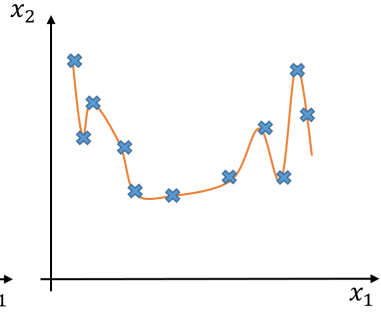
Über- und Unteranpassung



Unteranpassung



Vielversprechender Kandidat



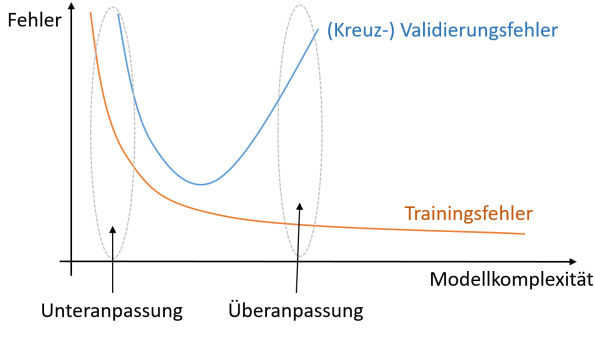
Überanpassung

Einfache Kreuzvalidierung

Um ungünstige Aaufteilungen in Trainings- und Testmenge zu vermeiden, kann dieses mehrfach wiederholt werden:

- Sei T die Menge der zur Verfügung stehenden Datenpunkte, mit $|T| = N$
- T wird in $k \leq N$ (möglichst) gleich große disjunkte Teilmengen $T_1, T_2, \ldots, T_k$ aufgeteilt
- Für $i = 1$ bis k:
  1. Wähle T_i als Testmenge (auch Kreuzvalidierungsmenge)
  2. Wähle $T \setminus T_i$ als Trainingsmenge
  3. Trainiere das gewählte Modell mit der Trainingsmenge aus Schritt 2 und der Kreuzvalidierungsmenge aus Schritt 1
  4. Bestimme den Trainings- und den Kreuzvalidierungsfehler
- Der Gesamte Fehler ist das arithmetische Mittel der einzelnen Fehler

Über- und Unteranpassung graphisch identifizieren



kNN-Algorithmus

Hier wird nur die binäre Klassifikation betrachtet, d.h. es gibt zwei mögliche Klassen: 0 und 1

Kommen wir nun zum ersten Algorithmus zur Klassifikation.

kNN

Der kNN-Algorithmus (k-nearest-neighbors) ist ein Klassifikations-Algorithmus, der für einen neuen Punkt x mit Hilfe von $k \in \mathbb{N}$ Trainingsbeispielen, die am nächsten an x liegen, die Klasse von x vorhersagt.

Diese Beschreibung lässt einige Fragen offen:

- Wie wird bestimmt, welche Punkte „am nächsten“ liegen?
- Wie wird auf Grundlage der k Punkte bestimmt, welche Klasse x zugeordnet wird?
- Wie wird k gewählt?

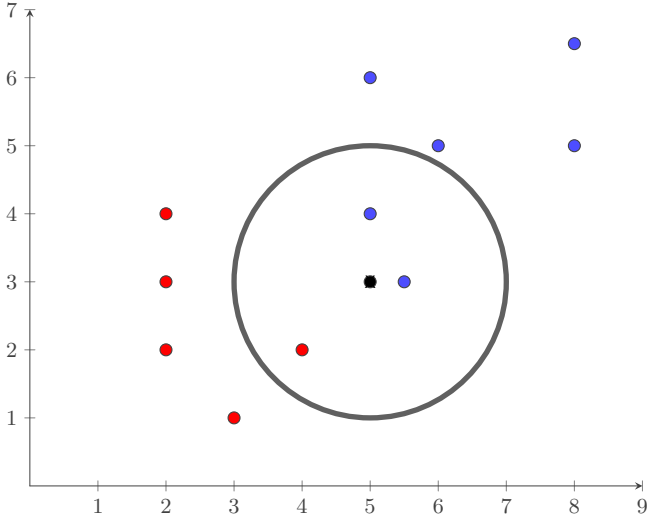


Abbildung: Nachbarschaft bzgl. des euklidischen Abstandes des Punktes $(5, 3)$ für $k = 3$

Vorgehen für kNN

Parameters k, Norm
Data: Traingingsdaten $(x_i, y_i), i = 1, \ldots, m$
Testdaten $(x_i, y_i), i = m+1, \ldots, m+n$
Result: Vorhergesagte Klassen $\hat{y}_i$ für $i = m+1, \ldots, m+n$
for Jeden Datenpunkt x_i der Trainingsdaten **do**
Bestimme die k Punkte aus der Trainingsmenge, die bzgl. der gewählten Norm am nächsten an x_i liegen
Die Prognostizierte Klasse $\hat{y}_i$ ist die Klasse, die am häufigsten unter diesen k Punkten vorkommt
end
return $[\hat{y}_{m+1}, \ldots, \hat{y}_{m+n}]$

- Der Algorithmus wird also maßgeblich durch die konkreten Wahl der Norm und k bestimmt
- Statt einfach die Mehrheit zu nehmen, kann auch die Entfernung der Nachbarpunkte mit in die Wertung einbezogen werden
- k wird häufig ungerade gewählt, um einen Gleichstand zu vermeiden

Für das Distanzmaß existiert eine Vielzahl von Möglichkeiten, einige bekannte sind etwa:

- **Euklidische Norm:** $\text{dist}(x, y) = \sqrt{\sum_i (x_i - y_i)^2}$
- **Manhatten Metrik:** $\text{dist}(x, y) = \sum_i |x_i - y_i|$
- **Maximumsnorm:** $\text{dist}(x, y) = \max_i |x_i - y_i|$

Als Kostenfunktion kann etwa die Anzahl der falsch klassifizierten Punkte herangezogen werden.

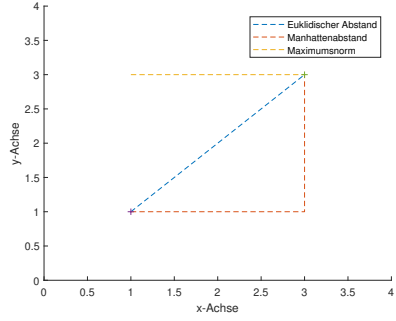
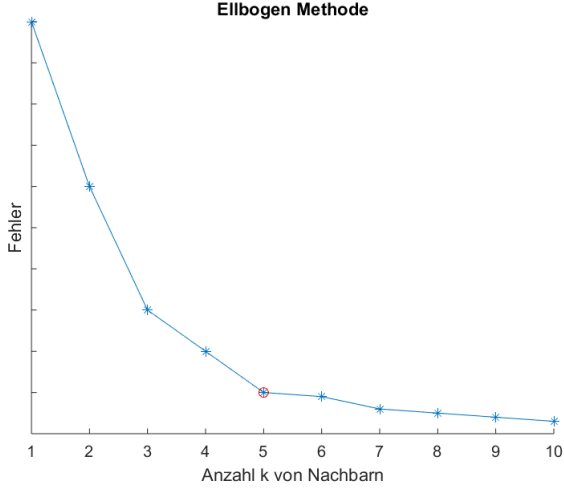


Abbildung: Visualisierung der verschiedenen Abstandsmaße

Wahl von k – Die Ellbogen Methode



- Für verschiedene Werte von k wird die Performance des Klassifikators evaluiert, indem der Validierungsfehler ermittelt wird
- In einem Plot werden die Werte von k gegen die dazugehörigen Validierungsfehler aufgetragen
- Visuell wird nun der „Knick“ in dem Verlauf ermittelt
- Da der Graph häufig wie ein geknickter Arm und der Knick-Punkt wie ein Ellbogen, wird die Methode als Ellbogen Methode bezeichnet
- In einigen Fällen kann ein eindeutiger Knick-Punkt ermittelt werden

Die logistische Regression

Bevor der eigentliche Algorithmus behandelt wird, betrachten wir zunächst die sogenannte Sigmoid Funktion:

$$\text{sigmoid}(x) = \frac{1}{1 + \exp(-x)}$$

Unter Sigmoid Funktionen fallen mehrere Funktionen. In der Regel wird mit *der* Sigmoid Funktion üblicherweise die obenstehende Funktion bezeichnet, die auch als *logistische Funktion* bekannt ist

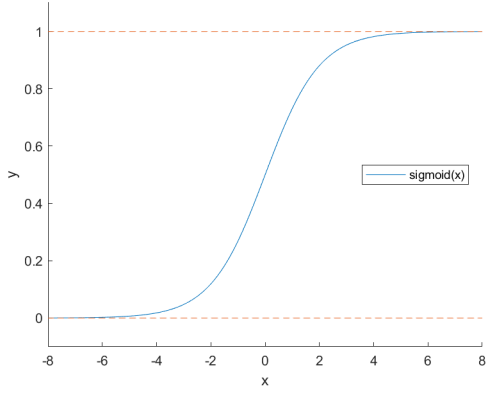


Abbildung: **Verlauf der Sigmoid-Funktion**

Wir betrachten zunächst eine Klassifikation mit zwei Klassen, repräsentiert durch 0 und 1.

- Zunächst sollen die Klassen durch eine lineare Funktion getrennt werden, d.h. $h_\theta(x) = \theta^T x$
- Der Klassifikator ist

$$S_h(x) := \text{sigmoid}(h_\theta(x)) = \frac{1}{1 + \exp(-h_\theta(x))}$$

- Ist $S_h(x) \geq 0.5$, so ist die prognostizierte Klasse 1 andernfalls 0
- Wann gilt $S_h(x) \geq 0.5$?

$$\begin{aligned} & S_h(x) \geq 0.5 \\ \Leftrightarrow \quad & \frac{1}{1 + \exp(-h_\theta(x))} \geq 0.5 \\ \Leftrightarrow \quad & 0 \leq h_\theta(x) \end{aligned}$$

- Das heißt, alles oberhalb der Geraden h_θ wird der Klasse 1 zugeordnet, alles unterhalb der Geraden der Klasse 0
- h_θ kann auch andere geeignete Formen annehmen, etwa Polynome vom Grad n

Entscheidungsgrenze

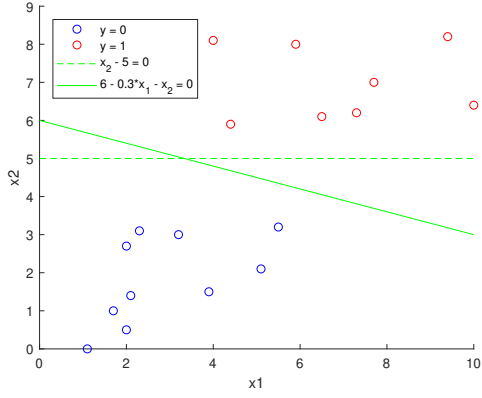


Abbildung: Beispiel für zwei Entscheidungsgrenzen bei logistischer Regression

Wie finden wir eine gute Hypothese bzw. eine gute Entscheidungsgrenze?

- Es soll die Klasse y prognostiziert werden, die bei der Beobachtung x und den gegebenen Werten von θ am wahrscheinlichsten sind
- Die Kostenfunktion für das Trainieren ist

$$L(\theta) = -\frac{1}{m}\sum_{i=1}^{m} y_i \cdot \log(h_\theta(x_i)) + (1 - y_i) \cdot \log(1 - h_\theta(x_i))$$

- Diese Funktion soll minimiert werden
- Dadurch wird der Maximum Likelihood Schätzer für θ approximiert

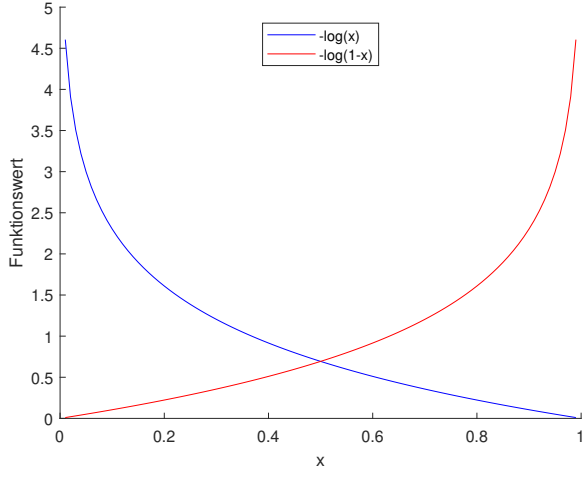


Abbildung: Darstellung der Kostenterme der Kostenfunktion für logistische Regression

Kenngrößen zur Bewertung eines Klassifikationsalgorithmus

Gegeben seien Prognosen für Klassenzugehörigkeiten aus der Testmenge. Wie wird Qualität der Prognosen bestimmt?

- Die möglichen Klassen seien 1 und 0
- Für jede Prognose in der Testmenge ist der wahre Wert bekannt
- Der wahre Wert wird mit y bezeichnet, der prognostizierte Wert mit $\hat{y}$
- Dafür werden vier Fälle unterschieden:

		Wahre Klasse	
		$y = 1$	$y = 0$
Prognose	$\hat{y} = 1$	true positive	false positive (Fehler 1. Art)
	$\hat{y} = 0$	false negative (Fehler 2. Art)	true negative

- Die Angabe *positive* und *negative* bezieht sich auf die Prognose ob die Klasse 1 (positive) oder 0 (negative) ist
- Die Angabe *true* gibt an, dass die Prognose richtig war, entsprechend gibt *false* an, dass die Prognose falsch war
- *true positive* (kurz tp) heißt, dass das Objekt zur Klasse 1 zugeordnet wurde und dass diese Zuordnung stimmt
- *true negative* (kurz tn) heißt, dass das Objekt zur Klasse 0 zugeordnet wurde und dass diese Zuordnung stimmt
- *false positive* (kurz fp) heißt, dass das Objekt zur Klasse 1 zugeordnet wurde und dass diese Zuordnung falsch ist
- *false negative* (kurz fn) heißt, dass das Objekt zur Klasse 0 zugeordnet wurde und dass diese Zuordnung falsch ist

Precision, Recall und F1-Score

Nach der Auswertung der Testmenge stehen folgende Werte zu Verfügung:

	$y = 1$	$y = 0$	$\sum$
$\hat{y} = 1$	tp	fp	tp+fp
$\hat{y} = 0$	fn	tn	fn+tn
$\sum$	tp+fn	fp+tn	tp+fp+fn+tn

Daraus lassen sich die folgenden Kennwerte berechnen:

$$\text{precision} = \frac{tp}{tp + fp}$$

$$\text{recall} = \frac{tp}{tp + fn}$$

$$\text{F1} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

precision:

- Anteil der korrekt positiv klassifizierten Instanzen an den insgesamt positiv klassifizierten Instanzen
- Ein Wert von 1 bedeutet, dass alle positiv klassifizierten Instanzen auch wirklich zur Klasse 1 gehören
- Ein Wert von 0 bedeutet, dass keine Instanz, die als positiv klassifiziert wurde, auch wirklich zur Klasse 1 gehört

recall:

- Anteil der korrekt positiv klassifizierten Instanzen an den wirklich positiven Instanzen
- Ein Wert von 1 heißt, dass alle Instanzen der Klasse 1 auch als solche erkannt wurden
- Ein Wert von 0 heißt, dass keine Instanz der Klasse 1 als solche erkannt wurde

F1-Score:

- Harmonisches Mittel von precision und recall
- Nimmt Werte zwischen 1 (perfekter precision und perfekter recall) und 0 (einer der beiden Werte ist 0) an

- precision, recall und F1-Score berücksichtigen nicht die true negative Werte
- Die bekannte *accuracy* berücksichtigt diese Werte:

$$\text{accuracy} = \frac{tp + tn}{tp + fp + fn + tn}$$

- Aber jede der vier Größen ist anfällig für schiefe Verteilungen
- Ein Maß, das die Größen berücksichtigt, ist der Korrelationskoeffzient von Matthew (engl. Matthews correlation coefficient):

$$MCC = \frac{tp \cdot tn - fp \cdot fn}{\sqrt{(tp + fn)(fp + tn)(tp + fp)(fn + tn)}}$$

- Der Wertebereich vom MCC ist $[-1, 1]$, wobei 1 der beste und -1 der schlechteste Wert ist
- Ein Wert von 0 bedeutet, dass der Klassifikator nicht besser ist, als eine zufällige Prognose
- Neben den hier vorgestellten Kennwerten existiert noch eine Vielzahl anderer Kennwerte, die eingesetzt werden können

Um ein Gefühl für die fünf Kennzahlen zu erhalten, betrachten wir ein hypothetisches Beispiel:

- Es seien 200 Bauteile untersucht worden, von denen 170 funktionsfähig und 30 defekt waren
- Es sei

$$y = \begin{cases} 1 & \text{, falls Bauteil funktionsfähig} \\ 0 & \text{, falls Bauteil defekt} \end{cases}$$

- Der erste Klassifikator prognostiziert immer Klasse 1, dann gilt

	$y = 1$	$y = 0$	$\sum$
$\hat{y} = 1$	170	30	200
$\hat{y} = 0$	0	0	0
$\sum$	170	30	200

Die Kennwerte sind dann

$$\text{precision} = \frac{170}{200} = 0.85$$
$$\text{accuracy} = \frac{170}{200} = 0.85$$
$$\text{recall} = \frac{170}{170} = 1$$
$$\text{F1} = 2 \cdot \frac{0.85 \cdot 1}{0.85 + 1} \approx 0.92$$

ABER: Dieser Klassifikator ist nicht besonders hilfreich, da er die entscheidende Information (Bauteil defekt) nicht liefert!
Insbesondere dass der MCC Wert nicht definiert ist, ist ein deutlicher Hinweis darauf, dass der Klassifikator problematisch ist!

Betrachten wir nun einen zweiten Klassifikator, der auch einige Bauteile zur Klasse 0 zuordnet:

	$y = 1$	$y = 0$	$\sum$
$\hat{y} = 1$	160	20	180
$\hat{y} = 0$	10	10	20
$\sum$	170	30	200

Es gilt:

$$\text{precision} = \frac{160}{180} \approx 0.89$$

$$\text{accuracy} = \frac{170}{200} \approx 0.94$$

$$\text{recall} = \frac{160}{170} \approx 0.94$$

$$\text{F1} = 2 \cdot \frac{0.89 \cdot 0.94}{0.89 + 0.94} \approx 0.91$$

$$MCC = \frac{160 \cdot 10 - 10 \cdot 20}{\sqrt{170 \cdot 30 \cdot 180 \cdot 20}} \approx 0.33$$

k means

Als letzter Algorithmus soll hier der k means Algorithmus vorgestellt werden.

- k means erstellt k Cluster
- Dafür wird ein Abstandmaß benutzt
- Das Vorgehen des k means Algorithmus:
  1. Parameter: Wähle k und das Abstandmaß
  2. Wähle zufällig k Zentrumspunkte $c_1, \ldots, c_k$, die den gleichen Wertebereich und die gleiche Dimension wie die Datenpunkte besitzen
  3. Bestimme für alle Datenpunkte x_i ihren Abstand zu den Punkten $c_1, \ldots, c_k$
  4. Der Punkt x_i wird dem Zentrumspunkt c_j zugeordnet (und damit vorläufig dem Cluster j), der ihm am nächsten ist
  5. c_j wird als der Mittelwert aller Punkte x_i aus dem Cluster j gesetzt
  6. Wiederhole Schritte 3-5 solange, bis eine maximale Iterationszahl erreicht wurde, oder die c_j sich nicht mehr ändern
  7. Ausgabe: Zuordnung der x_i zu den Clustern 1 bis k

- Der Parameter k kann mit der Ellbogen Methode gewählt werden, als Kostenmaß kann hier die Summe aller Abstände der x_i zu ihrem c_j gewählt werden
- Durch die zufällige Initialisierung der c_j ist die Ausgabe im Allgemeinen **nicht eindeutig**, daher sollte k means mehrmals wiederholt werden. Am Ende wird die Cluster Zuweisung gewählt, die die kleinsten Kosten aufweist
- Wie bei allen unüberwachten Algorithmen gibt es hier keine Trainings- oder Testmenge

Quellenverzeichnis I

[1] Usama Fayyad, Gregory Piatetsky-Shapiro und Padhraic Smyth. “From Data Mining to Knowledge Discovery in Databases”. en-US. In: 1 17.3 (März 1996), S. 37–37. ISSN: 2371-9621. DOI: 10.1609/aimag.v17i3.1230. URL: https://www.aaai.org/ojs/index.php/aimagazine/article/view/1230 (besucht am 29. 10. 2018).