Data Science 1

Vorlesung 4 - Maschinelles Lernen

Prof. Dr. Christian Bockermann

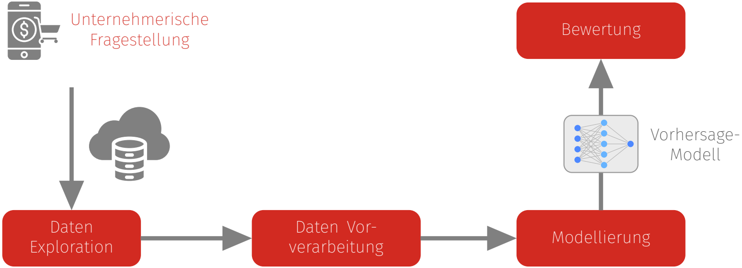
Hochschule Bochum

Wintersemester 2025/2026

Vorwort - Data Analysis Lifecycle

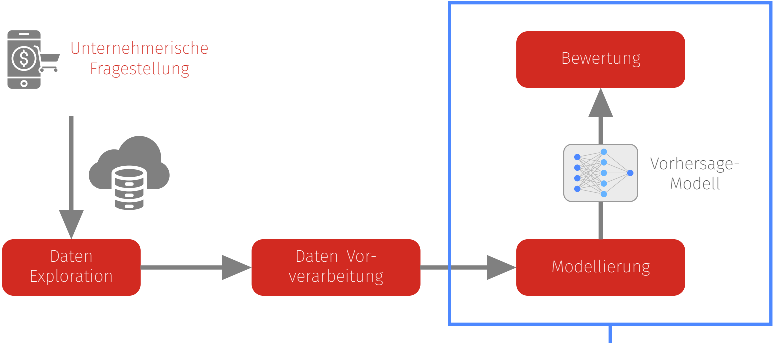
Hochschule Bochum
Bochum University
of Applied Sciences

Wir erinnern uns: Vorgehen bei der Datenanalyse (CRISP-DM)



Hochschule Bochum
Bochum University
of Applied Sciences

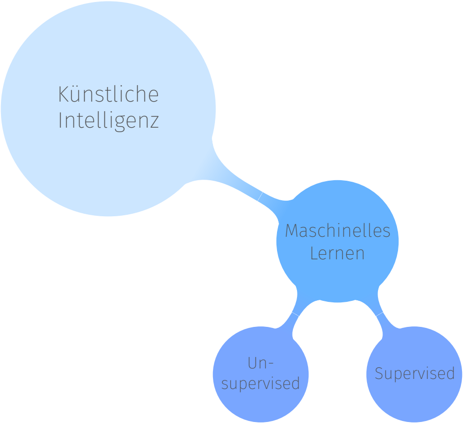
Wir erinnern uns: Vorgehen bei der Datenanalyse (CRISP-DM)



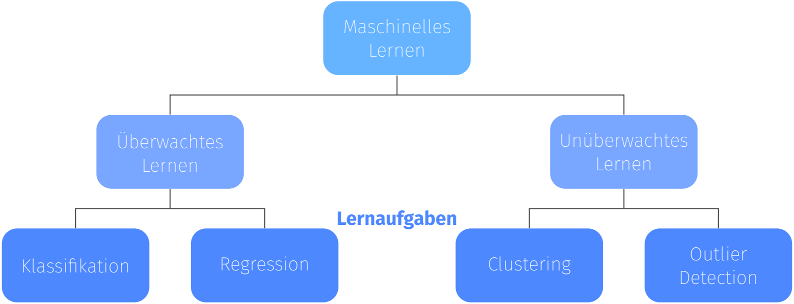
Modellierung + Validierung
mit Python / SciKit-Leran (später)

Inhalt

Maschinelles Lernen ist Teilgebiet der künstlichen Intelligenz

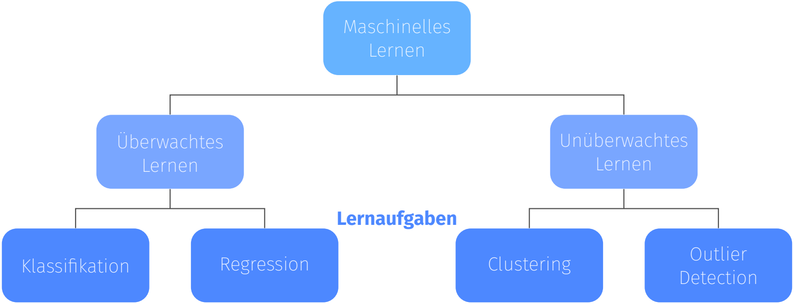


Maschinelles Lernen ist Teilgebiet der künstlichen Intelligenz



Hochschule Bochum
Bochum University
of Applied Sciences

Maschinelles Lernen ist Teilgebiet der künstlichen Intelligenz



Algorithmen

- Decision Trees
- Naive Bayes
- SVM

- Linear Regression
- Logistic Regression
- kNN

- kMeans
- DBScan
- Birch

- Isolation Forests
- Local Outlier Factor

Kategorien des Lernens

Überwachtes Lernen

supervised learning

- Trainingsdaten enthalten Zielvariable (z.B. Spam=Ja/Nein)
- Zielvariable oft als *Klasse*, *Label* oder *Class* bezeichnet
- Mit Trainingsdaten, unbekannte Daten vorhersagen

Unüberwachtes Lernen

unsupervised learning

- Trainingsdaten enthalten keine Zielinformation
- Unbekannte Muster/Gruppen in Daten finden

Lernaufgaben definieren Ein- und Ausgabe, sowie das Ziel der Modellierung, z.B.

> "Entscheide für einen Text **x** ob er zur Klasse *Spam* oder zur Klasse *KeinSpam* gehört."

Lernaufgaben definieren Ein- und Ausgabe, sowie das Ziel der Modellierung, z.B.

> "Entscheide für einen Text $\mathbf{x}$ ob er zur Klasse *Spam* oder zur Klasse *KeinSpam* gehört."

Eingabedaten werden typischerweise in einen Merkmalsraum $\mathcal{X}$ der Dimension d abgebildet

$$\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^d$$

Die Ausgabemenge $\mathcal{Y}$ kann eine Menge von Klassen oder eine reelle Zahl sein, z.B.

$$\mathcal{Y} = \{\text{Spam}, \text{KeinSpam}\}$$

Das Ziel besteht darin, eine Funktion (Modell) $f : \mathcal{X} \to \mathcal{Y}$ zu lernen, mit

$$f(\mathbf{x}) = \begin{cases} +1, & \text{falls } \mathbf{x} \text{ Spam Nachricht} \\ -1, & \text{sonst} \end{cases}$$

Das Ziel besteht darin, eine Funktion (Modell) $\boldsymbol{f} : \mathcal{X} \to \mathcal{Y}$ zu lernen, mit

$$f(\mathbf{x}) = \begin{cases} +1, & \text{falls } \mathbf{x} \text{ Spam Nachricht} \\ -1, & \text{sonst} \end{cases}$$

Bei der binären Klassifikation wird häufig $\mathcal{Y} = \{-1, +1\}$ gewählt.

Das Ziel besteht darin, eine Funktion (Modell) $f : \mathcal{X} \to \mathcal{Y}$ zu lernen, mit

$$f(\mathbf{x}) = \begin{cases} +1, & \text{falls } \mathbf{x} \text{ Spam Nachricht} \\ -1, & \text{sonst} \end{cases}$$

Bei der binären Klassifikation wird häufig $\mathcal{Y} = \{-1, +1\}$ gewählt.

Für die Regression gilt $\mathcal{Y} = \mathbb{R}$.

Lern-Algorithmen erwarten Daten häufig in Form einer Tabelle:

	d Merkmale				
ID	a_1	a_2	$\cdots$	a_d	y
1	0	0	$\cdots$	1	**-1**
2	0	1	$\cdots$	1	**+1**
3	1	0	$\cdots$	1	**-1**

$$\text{Beispiel } \mathbf{x_2} = (x_{a_1}, x_{a_2}, \ldots, x_{a_4}, y)$$
$$= (0, 1, \ldots, 1, \mathbf{+1})$$

- Beispiele werden auch *examples* oder *instances* genannt
- Merkmale (engl. *features*) werden auch *attributes* oder *Variablen* (Statistik) bezeichnet

ML - Modell Training

a_1	a_2	$\cdots$	a_d	y
0	0	$\cdots$	1	**-1**
0	1	$\cdots$	1	**+1**
1	0	$\cdots$	1	**-1**

Trainingsdaten $\mathbf{X}, \mathbf{y}$

Algorithmus/
Optimierung

$$f : \mathcal{X} \rightarrow \mathcal{Y}$$

Modell

Hochschule Bochum
Bochum University
of Applied Sciences

a_1	a_2	$\cdots$	a_d	y
0	0	$\cdots$	1	**-1**
0	1	$\cdots$	1	**+1**
1	0	$\cdots$	1	**-1**

Trainingsdaten $\mathbf{X}, \mathbf{y}$

Algorithmus/
Optimierung

$$f : \mathcal{X} \rightarrow \mathcal{Y}$$

Modell

a_1	a_2	$\cdots$	a_d	y
1	1	$\cdots$	0	**?**

Neue Daten $\mathbf{x}'$,
y unbekannt

ML - Modell Training

a_1	a_2	$\cdots$	a_d	y
0	0	$\cdots$	1	**-1**
0	1	$\cdots$	1	**+1**
1	0	$\cdots$	1	**-1**

Trainingsdaten $\mathbf{X}, \mathbf{y}$

Algorithmus/
Optimierung

$f : \mathcal{X} \rightarrow \mathcal{Y}$

Modell

a_1	a_2	$\cdots$	a_d	y
1	1	$\cdots$	0	**?**

Neue Daten $\mathbf{x}'$,
y unbekannt

Vorhersage

$$\hat{y} = f(\mathbf{x}')$$

Definition von Lernaufgaben

Im Folgenden werfen wir einen genaueren Blick auf die Definition von Lernaufgaben, die wir in den nächsten Vorlesungen behandeln:

Überwachtes Lernen

- Klassifikation
- Regression

Unüberwachtes Lernen

- Clustering
- (Outlier-Detection)
- Frequent Itemsets / Frequent Patterns

Klassifikation ordnet Beispielen diskrete Klassen zu

- Vorgegebene Klassen $\mathcal{Y} = \{C_1, \dots, C_k\}$
- Gegeben Menge $\mathbf{X} \times \mathbf{y} \subset \mathcal{X} \times \mathcal{Y}$ bei der jedem Beispiel x_i die zugehörige Klasse zugeordnet ist: (x_i, y_i)
- Qualitätsfunktion $q : (\mathcal{X} \times \mathcal{Y}) \times (\mathcal{X} \to \mathcal{Y}) \to \mathbb{R}$

Ziel:

- Finde Modell

$$f : \mathcal{X} \to \mathcal{Y},$$

das die Qualitätsfunktion <u>optimiert</u>.

Klassifikation ordnet Beispielen diskrete Klassen zu

- Vorgegebene Klassen $\mathcal{Y} = \{C_1, \ldots, C_k\}$
- Gegeben Menge $\mathbf{X} \times \mathbf{y} \subset \mathcal{X} \times \mathcal{Y}$ bei der jedem Beispiel x_i die zugehörige Klasse zugeordnet ist: (x_i, y_i)
- Qualitätsfunktion $q : (\mathcal{X} \times \mathcal{Y}) \times (\mathcal{X} \to \mathcal{Y}) \to \mathbb{R}$

Ziel:

- Finde Modell

$$f : \mathcal{X} \to \mathcal{Y},$$

das die Qualitätsfunktion optimiert.

Lernen als Optimierungsproblem!

Beispiel: Klassifikation von Schwertlilien

- Klassen: $\mathcal{Y} = \{\text{setosa}, \text{versicolor}, \text{virginica}\}$
- Menge $\mathbf{X} \times \mathbf{y}$ mit 150 Beispiele mit Spalte "`species`"
- Qualitätsfunktion

$$q(\mathbf{X}\times\mathbf{y}, f) = \sum_{(x,y)\in\mathbf{X}\times\mathbf{y}} err(y, \underbrace{f(x)}_{=\hat{y}}), \quad err(y, \hat{y}) = \begin{cases} 0, & \text{falls } y = \hat{y} \\ 1, & \text{sonst.} \end{cases}$$

Beispiel: Klassifikation von Schwertlilien

- Klassen: $\mathcal{Y} = \{\text{setosa}, \text{versicolor}, \text{virginica}\}$
- Menge $\mathbf{X} \times \mathbf{y}$ mit 150 Beispiele mit Spalte "`species`"
- Qualitätsfunktion

$$q(\mathbf{X}\times\mathbf{y}, f) = \sum_{(x,y)\in\mathbf{X}\times\mathbf{y}} err(y, \underbrace{f(x)}_{=\hat{y}}), \;\; err(y, \hat{y}) = \begin{cases} 0, & \text{falls } y = \hat{y} \\ 1, & \text{sonst.} \end{cases}$$

Funktion q zählt die Anzahl der Vorhersagefehler des Modells f auf der Menge X

Beispiel: Klassifikation von Schwertlilien

- Klassen: $\mathcal{Y} = \{\text{setosa}, \text{versicolor}, \text{virginica}\}$
- Menge $\mathbf{X} \times \mathbf{y}$ mit 150 Beispiele mit Spalte "`species`"
- Qualitätsfunktion

$$q(\mathbf{X}\times\mathbf{y}, f) = \sum_{(x,y)\in\mathbf{X}\times\mathbf{y}} err(y, \underbrace{f(x)}_{=\hat{y}}), \quad err(y, \hat{y}) = \begin{cases} 0, & \text{falls } y = \hat{y} \\ 1, & \text{sonst.} \end{cases}$$

Funktion q zählt die Anzahl der Vorhersagefehler des Modells f auf der Menge X

Ziel: Finde f^* mit minimalem $q(X, f)$

Beispiel: Klassifikation von Schwertlilien

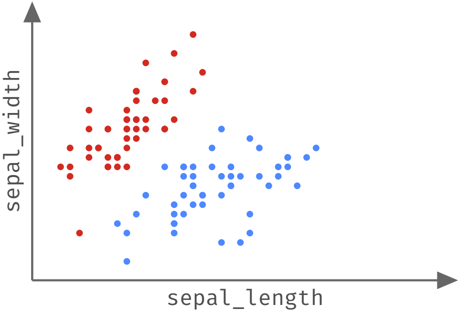
- Klassen: $\mathcal{Y} = \{\text{setosa}, \text{versicolor}, \text{virginica}\}$
- Menge $\mathbf{X} \times \mathbf{y}$ mit 150 Beispiele mit Spalte “`species`”
- Qualitätsfunktion

$$q(\mathbf{X}\times\mathbf{y}, f) = \sum_{(x,y)\in\mathbf{X}\times\mathbf{y}} err(y, \underbrace{f(x)}_{=\hat{y}}), \quad err(y,\hat{y}) = \begin{cases} 0, & \text{falls } y = \hat{y} \\ 1, & \text{sonst.} \end{cases}$$

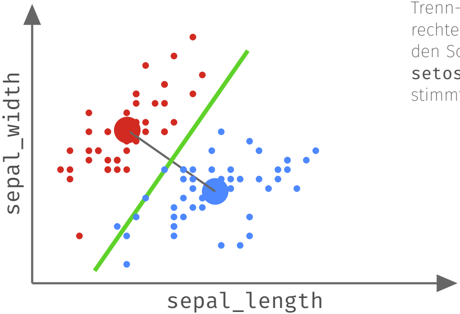
Funktion q zählt die Anzahl der Vorhersagefehler des Modells f auf der Menge X

Ziel: Finde f^* mit minimalem $q(X, f)$ $\longrightarrow$ Optimierungsproblem

Beispiel: Klassifikation von Schwertlilien



Beispiel: Klassifikation von Schwertlilien

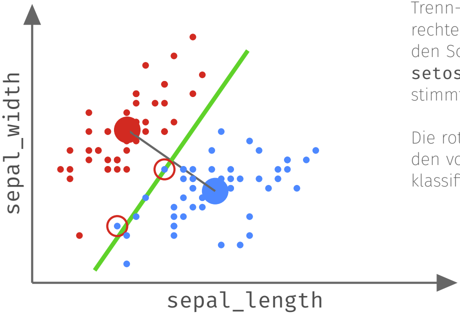


In diesem Fall wurde eine Trenn-Ebene als Mittelsenkrechte auf der Strecke zwischen den Schwerpunkten der Klasse `setosa` und `versicolor` bestimmt.

Einfacher Algorithmus:

Trenn-Ebene über die Klassenschwerpunkte der Attribute `sepal_length` und `sepal_width`

Beispiel: Klassifikation von Schwertlilien



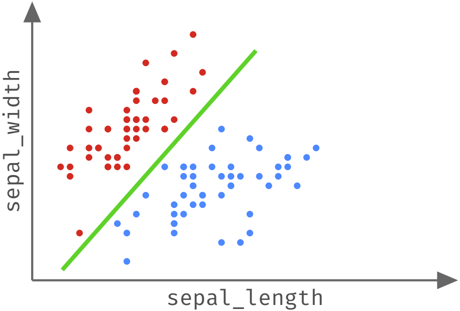
In diesem Fall wurde eine Trenn-Ebene als Mittelsenkrechte auf der Strecke zwischen den Schwerpunkten der Klasse `setosa` und `versicolor` bestimmt.

Die rot umkreisten Punkte werden von der Trenn-Ebene falsch klassifiziert.

Einfacher Algorithmus:

Trenn-Ebene über die Klassenschwerpunkte der Attribute `sepal_length` und `sepal_width`

Beispiel: **Klassifikation von Schwertlilien**



Die Daten sind *linear separierbar* – eine andere Ebene schafft dies ohne Fehler.
Die Optimierung der Qualitätsfunktion sucht nach der besten Ebene.

Regression liefert reellwertige Vorhersagen

- Für Regression gilt $\mathcal{Y} = \mathbb{R}$
- Menge $\mathbf{X} \times \mathbf{y}$, d.h. jedem Beispiel x_i ist ein $y_i \in \mathbb{R}$ zugeordnet
- Qualitätsfunktion $q : (\mathcal{X} \times \mathcal{Y}) \times (\mathcal{X} \to \mathcal{Y}) \to \mathbb{R}$

Ziel:

- Finde Modell

$$f : \mathcal{X} \to \mathcal{Y},$$

das die Qualitätsfunktion optimiert.

Regression liefert reellwertige Vorhersagen

- Für Regression gilt $\mathcal{Y} = \mathbb{R}$
- Menge $\mathbf{X} \times \mathbf{y}$, d.h. jedem Beispiel $\boldsymbol{x}_i$ ist ein $\boldsymbol{y}_i \in \mathbb{R}$ zugeordnet
- Qualitätsfunktion $\boldsymbol{q} : (\mathcal{X} \times \mathcal{Y}) \times (\mathcal{X} \to \mathcal{Y}) \to \mathbb{R}$

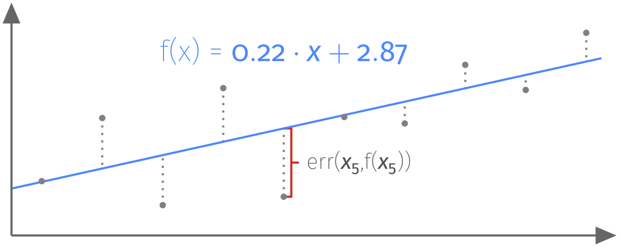
Ziel:

- Finde Modell
$$f : \mathcal{X} \to \mathcal{Y},$$
das die Qualitätsfunktion optimiert.

Auch hier wieder: Lernen als Optimierungsproblem!

Hochschule Bochum
Bochum University
of Applied Sciences

Beispiel: Regression

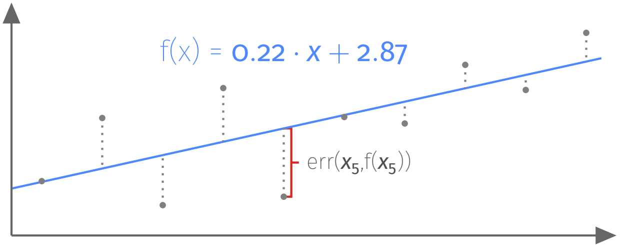


Qualitätsfunktion:

Summe der Abstände von $f(x)$ zu den “richtigen” Werten

$$q(X, f) = \sum_{(x,y)\in X} (y - f(x))^2 = RSS(X, f)$$

Beispiel: Regression



Qualitätsfunktion:

Summe der Abstände von $f(x)$ zu den “richtigen” Werten

$$q(X,f) = \sum_{(x,y)\in X} (y - f(x))^2 = \boxed{RSS(X,f)}$$

Residual Sum of Squares

Clustering sucht Aufteilung von Daten in ähnliche Gruppen

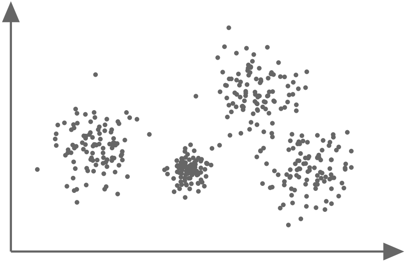
- Datenmenge $\mathbf{X}$ von Beispielen (keine Klassen gegeben!)
- Parameter $\boldsymbol{k}$ zu findender Gruppen
- Abstandsmaß $\boldsymbol{d} : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$
- Qualitätsfunktion $\boldsymbol{q}$

Ziel:

- Abstand *innerhalb* der Gruppen soll <u>minimiert</u>, Abstand *zwischen* den Gruppen soll <u>maximiert</u> werden

Beispiel: Clustering

Sei $\mathbf{C} = C_1, \ldots, C_k$ eine Aufteilung der Daten X (ein *Clustering*)

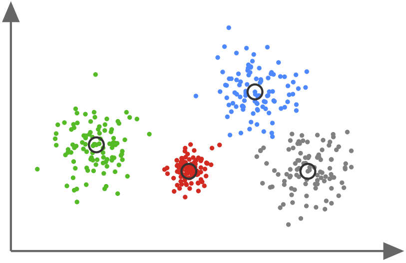


Qualitätsfunktion: (Innere Abstände)

$$q_{inner}(\mathbf{C}) = \sum_{i=1}^{k} \sum_{x \in C_i} d(x, \bar{\mathbf{c}}_i) \quad \text{, mit } \bar{\mathbf{c}}_i \text{ Zentrum von } C_i$$

Beispiel: Clustering

Sei $\mathbf{C} = C_1, \ldots, C_k$ eine Aufteilung der Daten X (ein *Clustering*)



Clustering auf Datenpunkten mit $k = 4$. Die schwarzen Kreise markieren jeweils das Zentrum $\bar{\mathbf{c}}_i$ des jeweiligen Cluster C_i.

Qualitätsfunktion: (Innere Abstände)

$$q_{inner}(\mathbf{C}) = \sum_{i=1}^{k} \sum_{x \in C_i} d(x, \bar{\mathbf{c}}_i) \quad \text{, mit } \bar{\mathbf{c}}_i \text{ Zentrum von } C_i$$

Hochschule Bochum
Bochum University
of Applied Sciences

Beispiel: Clustering

- Clustering unter mehreren Qualitätsaspekten:

$$q_{inner}(\mathbf{C}) = \sum_{i=1}^{k} \sum_{x \in C_i} d(x, \bar{\mathbf{c}}_i) \quad \rightarrow \text{Minimieren}$$

$$q_{outer}(\mathbf{C}) = \sum_{i=1}^{k} \sum_{x \in C_j, j \neq i} d(x, \bar{\mathbf{c}}_j) \quad \rightarrow \text{Maximieren}$$

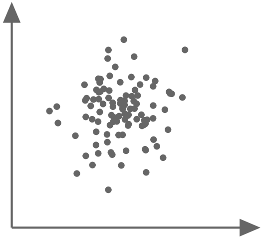
Outlier-Detection sucht nach *isolierten* Punkten

- Gegen ist Datensatz $\mathbf{X}$ (keine Label)
- i.d.R. noch Abstandsmaß $\boldsymbol{d} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$

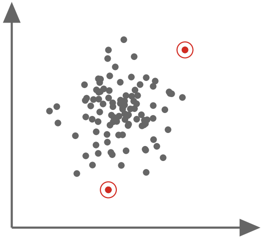
Ziel:

- Finde Punkte, die *weit weg* von allen anderen Punkten liegen

Beispiel: Outlier-Detection

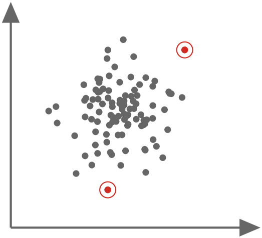


Beispiel: Outlier-Detection



Die rot markierten Punkte sind Ausreißer die im Abstand von $d = 0.35$ keine Nachbarpunkte haben.

Beispiel: Outlier-Detection



Die rot markierten Punkte sind Ausreißer die im Abstand von $d = 0.35$ keine Nachbarpunkte haben.

Unterschiedliche Ansätze für Ausreißer-Erkennung:

- Vorgabe des minimalen Abstandes zu Nachbarn (siehe oben)
- Dichte-basiert - Verteilung der Abstände
- Über Mini-Clustering (ganz viele kleine Cluster finden)

Frequent Itemset Mining sucht häufige Muster

- Gegeben ist Menge $\mathbf{S}$ von Symbolen (z.B. Artikel)
- Eingabe ist Menge $\mathbf{X}$ von Transaktionen (Einkäufe) über $\mathbf{S}$

$$\mathbf{X} = \{\, x \mid x \subseteq \mathbf{S} \,\}$$

Ziel:

- Fragestellung: Welche Symbole tauchen häufig zusammen auf?
- Finde die Muster $p \in \mathcal{P}(\mathbf{S})$ die in $\mathbf{X}$ am häufigsten vorkommen

Beispiel: Frequent Itemsets auf Einkäufen

ID	Artikel
1	{ A, B, F }
2	{ B, D, E, F }
3	{ C, E }
4	{ B, E, F }
5	{ A, B, E }

Hochschule Bochum
Bochum University
of Applied Sciences

Beispiel: Frequent Itemsets auf Einkäufen

ID	Artikel
1	{ A, B, F }
2	{ B, D, E, F }
3	{ C, E }
4	{ B, E, F }
5	{ A, B, E }

- Artikel B = Muster { B } taucht in 4/5 der Einkäufe auf

Beispiel: Frequent Itemsets auf Einkäufen

ID	Artikel
1	{ A, B, F }
2	{ B, D, E, F }
3	{ C, E }
4	{ B, E, F }
5	{ A, B, E }

- Artikel B = Muster { B } taucht in 4/5 der Einkäufe auf
- Muster { B, F } taucht in 3/5 aller Einkäufe auf

Beispiel: Frequent Itemsets auf Einkäufen

ID	Artikel
1	{ A, B, F }
2	{ B, D, E, F }
3	{ C, E }
4	{ B, E, F }
5	{ A, B, E }

- Artikel B = Muster { B } taucht in 4/5 der Einkäufe auf
- Muster { B, F } taucht in 3/5 aller Einkäufe auf

Welche Artikel werden häufig zusammen gekauft?

Überwachtes Lernen

Charakterisierung des Überwachten Lernens

- Lernen auf Daten $\mathbf{X}$ mit zugeordnetem Label $\mathbf{y}$ (="Wahrheit")
- Label oft manuell vergeben oder Messwerte (Regression)
- Validierung von Modell $\boldsymbol{f}$ durch Vergleich mit $\mathbf{y}$ möglich

Charakterisierung des Überwachten Lernens

- Lernen auf Daten **X** mit zugeordnetem Label **y** (="Wahrheit")
- Label oft manuell vergeben oder Messwerte (Regression)
- Validierung von Modell ***f*** durch Vergleich mit **y** möglich

Beispiel: MNIST-Datensatz - Ziffernerkennung

Für Trainingsdaten: Manuelle Zuordnung der Ziffernbilder zum richtigen Label (2, 9, 6,...)

Lernen auf Daten

a1	a2	a3
4	1	2
5	1	3
3	8	7

$\mathbf{X}$

y
1
-1
1

$\mathbf{y}$

ŷ
-1
-1
1

$f(\mathbf{X})$

- Lernalgorithmus sucht bestes Modell f^* für Daten $\mathbf{X}, \mathbf{y}$
- Ziel des Trainings: Fehler auf $\mathbf{X}, \mathbf{y}$ minimieren:

$$f^* = \arg\min_{f} \sum_{y \in \mathbf{y}} err(y, f(y)) \quad \text{(Trainingsfehler)}$$

Wie lernt ein Algorithmus?

- Algorithmus hat Klasse von Lösungen (z.B. Trenn-Ebenen)
- Lösungsraum wird parametrisiert und die beste Lösung gesucht (Optimierungsproblem)

Beispiel: Lineare Modelle

- Alle Ebenen in $\mathcal{X}$ darstellbar als

$$\vec{x} \cdot \vec{n} = d \quad \text{(Hessesche Normalform)}$$

- Suche $\vec{n} = (n_1, \ldots, n_k)$ und d, das möglichst viele Beispiele aus $\mathbf{X}$ richtig klassifiziert (den Trainingsfehler minimiert)

Zentrale Frage: Wie gut ist das gelernte Modell f^* ?

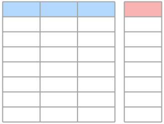
- Trainingsfehler gibt nur Auskunft über $\boldsymbol{f}^*$ auf *bekannten Daten* $\mathbf{X} \times \mathbf{y}$

Zentrale Frage: Wie gut ist das gelernte Modell f^* ?

- Trainingsfehler gibt nur Auskunft über $\boldsymbol{f}^*$ auf *bekannten Daten* $\mathbf{X} \times \mathbf{y}$

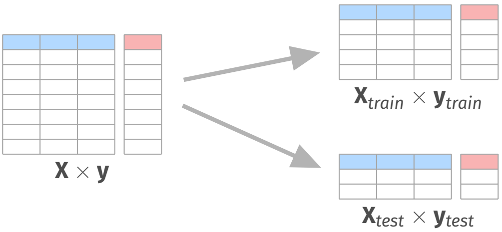
Wie gut ist f^* auf unbekannten Daten?

Ansatz: Aufteilung in Trainings- und Test-Daten



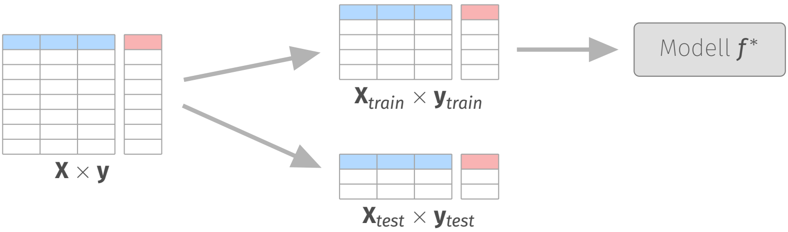
$\mathbf{X} \times \mathbf{y}$

Ansatz: Aufteilung in Trainings- und Test-Daten



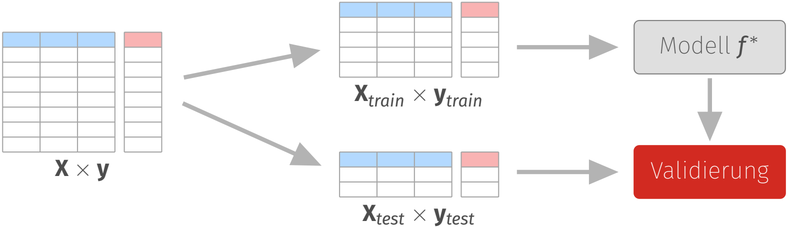
- Nutze *unabhängige* Test-Daten um f^* zu validieren!
- Oft 80% Trainingsdaten, 20% zum Testen (auch 70/30)

Ansatz: Aufteilung in Trainings- und Test-Daten



- Nutze *unabhängige* Test-Daten um f^* zu validieren!
- Oft 80% Trainingsdaten, 20% zum Testen (auch 70/30)

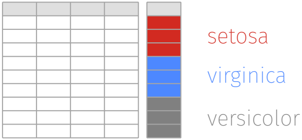
Ansatz: Aufteilung in Trainings- und Test-Daten



- Nutze *unabhängige* Test-Daten um f^* zu validieren!
- Oft 80% Trainingsdaten, 20% zum Testen (auch 70/30)

Aufteilung in Train/Test Daten

- Ok, nehmen wir 80:20 - was müssen wir beachten?
- Denken Sie an den Iris Datensatz (Übungsblatt 2, Aufgabe 2)!



Was passiert bei folgender Aufteilung von 60:40?

```
n = iris.shape[0]          # n Beispiele
splitAt = int(0.6 * n)     # 60% zum Training

X_train = iris[0:splitAt]
X_test = iris[splitAt:]
```

Wie ähnlich sollten sich $\mathbf{X}_{train} \times \mathbf{y}_{train}$ und *Xytest* sein?

Klassenverhältnis im Iris Datensatz:

- Gleichverteilt: setosa / virginica / versicolor jeweils 1/3
- Bei *linearem Splitting* im Verhältnis 60:40 ergibt sich:

$\mathbf{X}_{train} \times \mathbf{y}_{train}$	$\mathbf{X}_{test} \times \mathbf{y}_{test}$
50 × setosa 40 × virginica	10 × virginica 50 × versicolor

Wie ähnlich sollten sich $\mathbf{X}_{train} \times \mathbf{y}_{train}$ und *Xytest* sein?

Klassenverhältnis im Iris Datensatz:

- Gleichverteilt: setosa / virginica / versicolor jeweils 1/3
- Bei *linearem Splitting* im Verhältnis 60:40 ergibt sich:

$\mathbf{X}_{train} \times \mathbf{y}_{train}$	$\mathbf{X}_{test} \times \mathbf{y}_{test}$
50 × setosa 40 × virginica	10 × virginica 50 × versicolor

Klasse versicolor in Trainingsdaten <u>nicht enthalten</u>!

Klasse setosa in Testdaten <u>nicht enthalten</u>!

Split gemäß der Klassenverteilung: Stratified Sampling

- Stratified Sampling erhält die Klassenverhältnisse
- Beispiel für 60:40 Split:

$n = 150$

setosa	$50 = \frac{1}{3} \cdot n$
virginica	$50 = \frac{1}{3} \cdot n$
versicolor	$50 = \frac{1}{3} \cdot n$

$\mathbf{X} \times \mathbf{y}$

setosa	$\frac{1}{3} \cdot \frac{3}{5} \cdot n = 30$
virginica	$\frac{1}{3} \cdot \frac{3}{5} \cdot n = 30$
versicolor	$\frac{1}{3} \cdot \frac{3}{5} \cdot n = 30$

$\mathbf{X}_{train} \times \mathbf{y}_{train}$

setosa	$\frac{1}{3} \cdot \frac{2}{5} \cdot n = 20$
virginica	$\frac{1}{3} \cdot \frac{2}{5} \cdot n = 20$
versicolor	$\frac{1}{3} \cdot \frac{2}{5} \cdot n = 20$

$\mathbf{X}_{test} \times \mathbf{y}_{test}$

Split gemäß der Klassenverteilung: Stratified Sampling

- Stratified Sampling erhält die Klassenverhältnisse
- Beispiel für 60:40 Split:

$n = 150$

setosa	$50 = \frac{1}{3} \cdot n$
virginica	$50 = \frac{1}{3} \cdot n$
versicolor	$50 = \frac{1}{3} \cdot n$

$\mathbf{X} \times \mathbf{y}$

setosa	$\frac{1}{3} \cdot \frac{3}{5} \cdot n = 30$
virginica	$\frac{1}{3} \cdot \frac{3}{5} \cdot n = 30$
versicolor	$\frac{1}{3} \cdot \frac{3}{5} \cdot n = 30$

$\mathbf{X}_{train} \times \mathbf{y}_{train}$

setosa	$\frac{1}{3} \cdot \frac{2}{5} \cdot n = 20$
virginica	$\frac{1}{3} \cdot \frac{2}{5} \cdot n = 20$
versicolor	$\frac{1}{3} \cdot \frac{2}{5} \cdot n = 20$

$\mathbf{X}_{test} \times \mathbf{y}_{test}$

Aber: Was ist mit den Verteilungen der anderen Attribute? Zum Beispiel `sepal_length`?

Weiteres Problem:

- Daten mit Label oft nur begrenzt verfügbar
- Wir wollen möglichst viele Daten für ein gutes Modell nutzen

Idee: Leave-One-Out

1. Wähle ein Beispiel $x_i \in \mathbf{X}$
2. Trainiere das Modell f auf $(n-1)$ Beispielen $\mathbf{X} \setminus \{x_i\}$
3. Wir validieren f auf dem einen ausgewählten Beispiel x_i
4. Wiederhole das für alle n Beispiel und berechne den Durchschnittsfehler

Weitere Validierungsmethode ist Cross Validation

Vorgehen: Datenmenge sei $\mathbf{T}$

- Teile Daten $\mathbf{T}$ zufällig in k Teilmengen $\mathbf{T}_1, \ldots, \mathbf{T}_k$ auf
- Nutze $\mathbf{T}_i$ als Testmenge und $\mathbf{T} \setminus \mathbf{T}_i$ als Trainingsdaten
- Wiederhole dies für $i = \{1, \ldots, k\}$ und berechne Durchschnittsfehler über alle $\mathbf{T}_i$

$\mathbf{T}_1$	$\mathbf{T}_2$	$\mathbf{T}_3$	$\mathbf{T}_4$	$\mathbf{T}_5$

Varianten:

- Stratified Cross-Validation erhält Klassenverteilung in den $\mathbf{T}_i$
- Spezialfall $k = n = |T|$ ergibt Leave-one-out Validierung