

KI & Machine Learning

1

Einführung Machine Learning

- **Künstliche Intelligenz** (KI, engl. AI=artificial intelligence) bezeichnet die Fähigkeit von Maschinen, Aufgaben zu lösen, die normalerweise menschliche Intelligenz benötigen
- **Machine Learning** (ML) ist ein Teilgebiet der Künstlichen Intelligenz
- ML bezeichnet eine Klasse von Algorithmen, die „lernen“, d.h. die sich selber verbessern, ohne das alles explizit programmiert wird
- Der Algorithmus lernt anhand der Bewertung seiner Entscheidungen bzw. Handlungen. Dabei versucht er, eine sogenannte **Verlustfunktion** zu minimieren

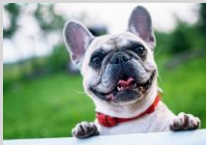
Beispiel: Klassifikation von Bildern, ob ein Hund auf dem Bild ist

Lernphase

Algorithmus bekommt einen Datensatz von Bildern, bei denen angegeben ist, ob ein Hund auf dem Bild ist. Anhand dieser Daten optimiert er seine internen Parameter, um möglichst viele der Bilder richtig zu klassifizieren.



ja



ja



nein



ja



ja

Anwendungsphase

Der trainierte Algorithmus bekommt ein Bild als Input. Er gibt aus, ob ein Hund auf dem Bild ist oder nicht, ggf. noch mit einer Prozentangabe seiner Sicherheit.



ja (90%)

Nein (10%)

Welche Beispiele kennt ihr?

- ChatGPT / Assistenz-Bots
(Language Models)
- Versicherungen
- Industrie 4.0 Bsp.: autonomes
Fahren in Hallen
- Autonomes Fahren
- Staubsaugerroboter
- Rasenroboter
- Bilderstellung (per Prompt)
- Recommender Systems
- IoT (Internet of things)
- Spracherkennung und –
verarbeitung
- Katastrophenwarnungen
- Wetterprognosen

Innerhalb von ML gibt es im Wesentlichen drei Klassen von Algorithmen

Überwachtes Lernen (Supervised Learning)

Der Algorithmus lernt anhand eines Datensatzes, der Input und Lösung beinhaltet

z.B.

- Regression/Prognosen
- Klassifikation
- Ähnlichkeitsbeurteilung

Unüberwachtes Lernen (Unsupervised Learning)

Der Algorithmus erkennt Muster in den Daten und benutzt diese für seine Aufgabe

z.B.

- Clustering/Gruppierung
- Dimensionsreduktion
- Ausreißererkennung



Bestärkendes Lernen (Reinforcement Learning)

Der Algorithmus (Agent) interagiert in einer Umgebung und bekommt Feedback zu seinen Aktionen

z.B.

- Robotik
- Automation/Steuerung

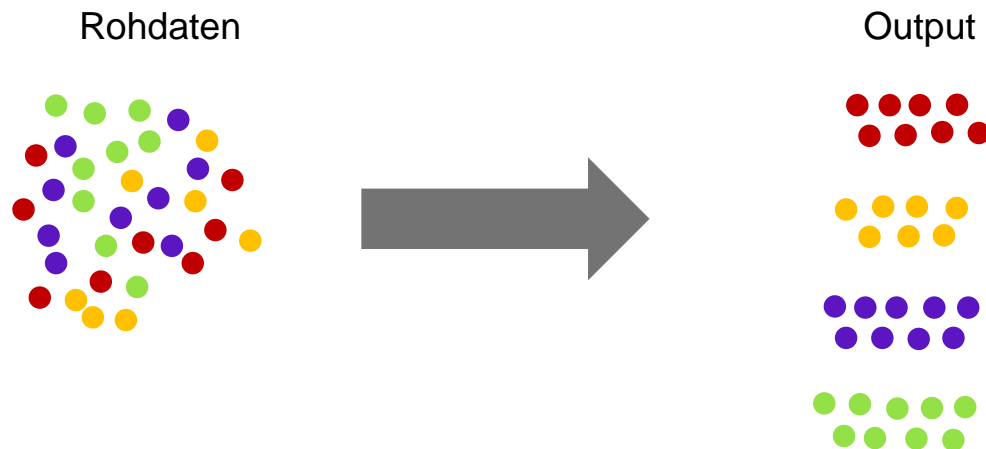
Algorithmen der Klasse überwachtes Lernen werden am häufigsten verwendet. Dafür wird ein Datensatz inklusive Lösung benötigt, der häufig nur aufwändig manuell erzeugt werden kann.

Input	Ergebnis
	3
	8
	9
	3
	0
	0
	1

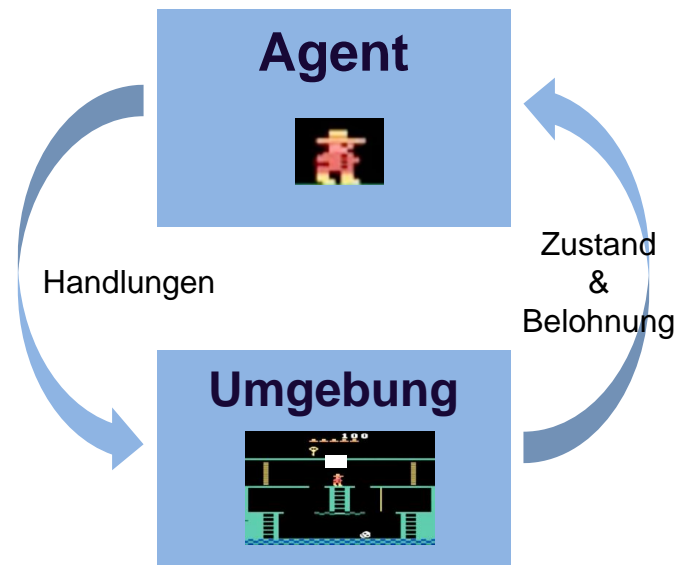


Ziffer	Wahrscheinlichkeit	y_{ij}
0	0,094	0
1	0,003	0
2	0,009	0
3	0,010	0
4	0,010	0
5	0,013	0
6	0,014	0
7	0,009	0
8	0,174	0
9	0,664	1

Unüberwachte Algorithmen haben den Vorteil, dass sie direkt mit den gesammelten Daten arbeiten können. Allerdings umfassen sie nur einige Anwendungsfelder, z.B. Clustering, Dimensionsreduktion oder Feature Engineering.



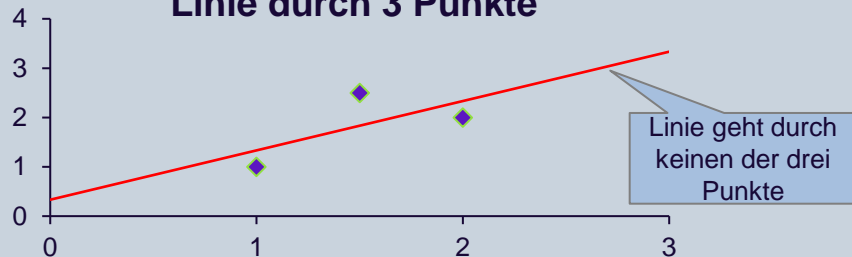
- **Reinforcement Learning** erscheint als die Algorithmenklasse, die dem menschlichen Lernen am nächsten kommt
- Es gibt vielversprechende Resultate in den letzten Jahren, z.B. Go, Videospiele (Atari, Dota), Protein Folding
- Wichtig ist die Möglichkeit, schnell Szenarien durchzuspielen (virtuelle Simulation). So wird z.B. eine Robotersteuerung zuerst in einer Simulation trainiert und dann in der echten Welt mit wenigen Trainings-Durchläufen optimiert



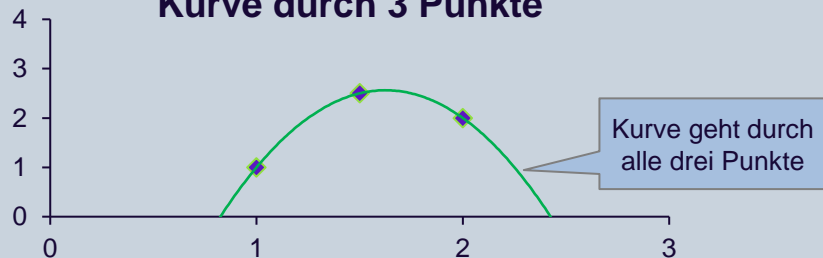
Ein großes Problem im maschinellen Lernen ist das sogenannte **Overfitting**. Damit bezeichnet man das zu starke Optimieren der Parameter an die vorhandenen Daten. Damit wird die Vorhersagekraft geschwächt bzw. ist überhaupt nicht mehr vorhanden. Meistens tritt Overfitting auf, wenn man **zu viele Parameter** und **zu wenig Daten** hat.

Welche Funktion (Linie oder Parabel) beschreibt den Zusammenhang besser?

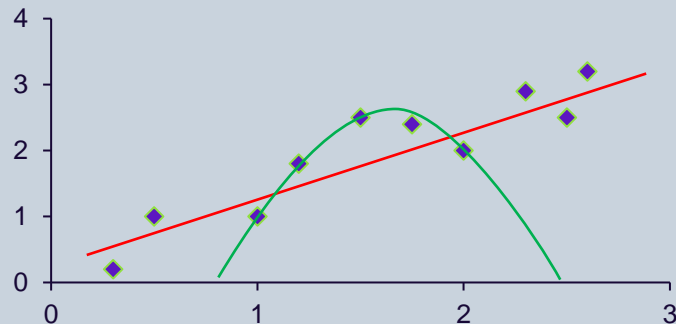
Linie durch 3 Punkte



Kurve durch 3 Punkte



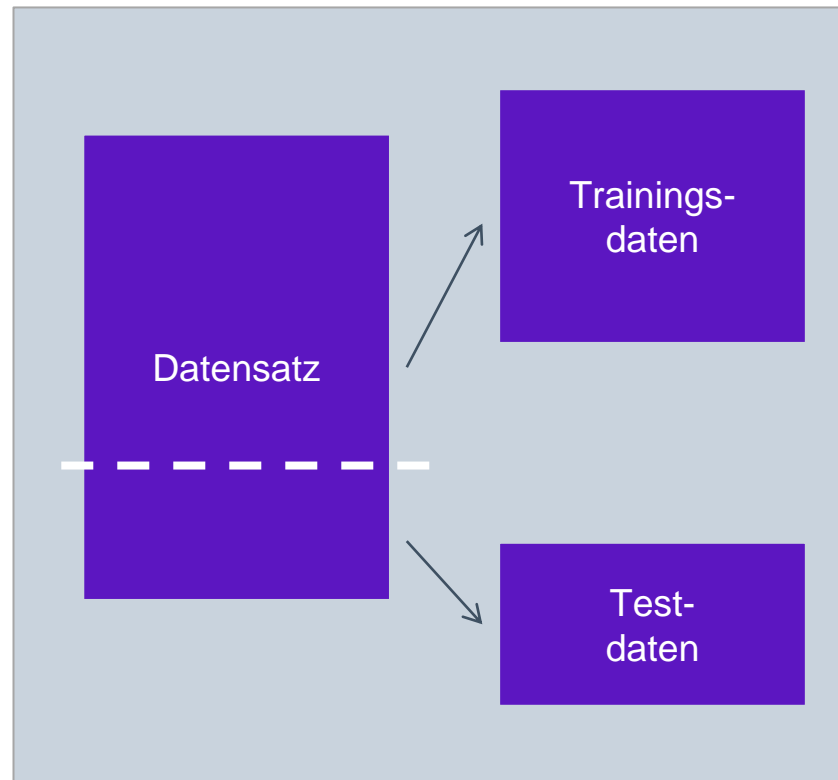
Mehr Datenpunkte



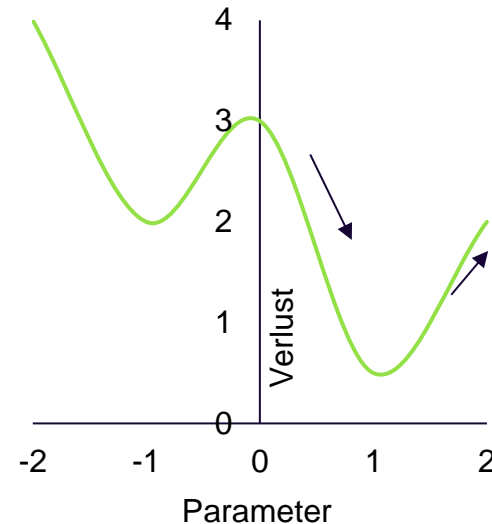
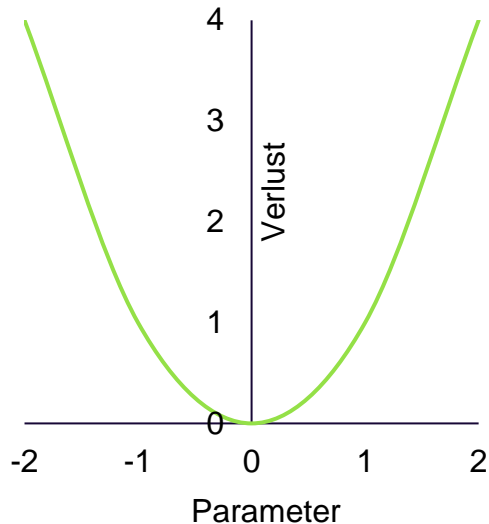
Wenden wir einen Algorithmus aus der Klasse des **überwachten Lernens** an, so trennt man den zur Verfügung stehenden Datensatz in zwei Teile, um Overfitting zu vermeiden:

- Der **Trainingsdatensatz** dient dazu, die Parameter des Modells zu optimieren
- Der **Testdatensatz** ist ausschließlich für die Bestimmung der Güte vorgesehen

Es gibt verschiedene Regeln zur Wahl des Trainings-Test-Verhältnis, welche von der Anzahl Parameter abhängt. Als Daumenregel kann man 80%/20% verwenden.

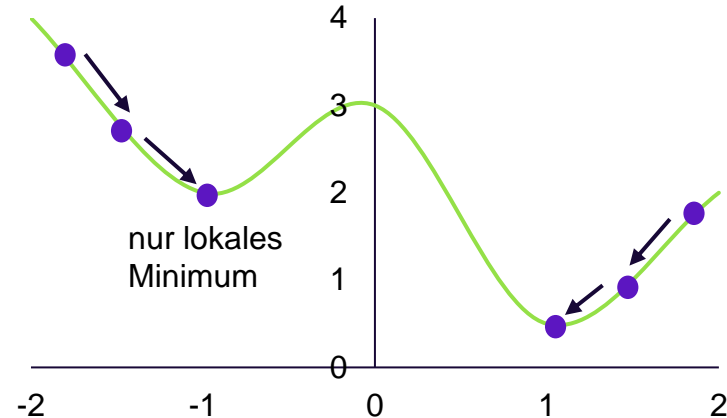
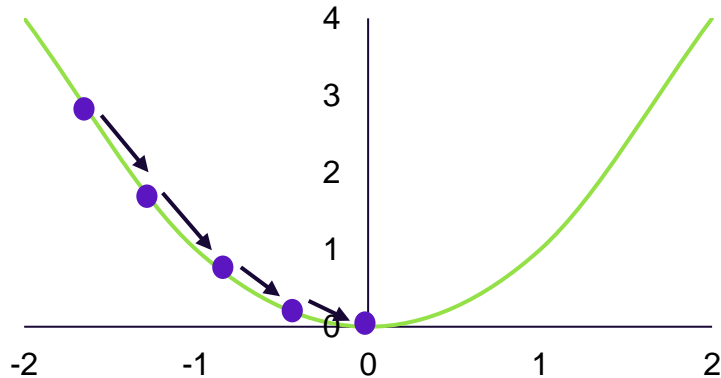


Der ML-Algorithmus versucht, die Gütefunktion zu maximieren bzw. die Verlustfunktion (loss function) zu minimieren

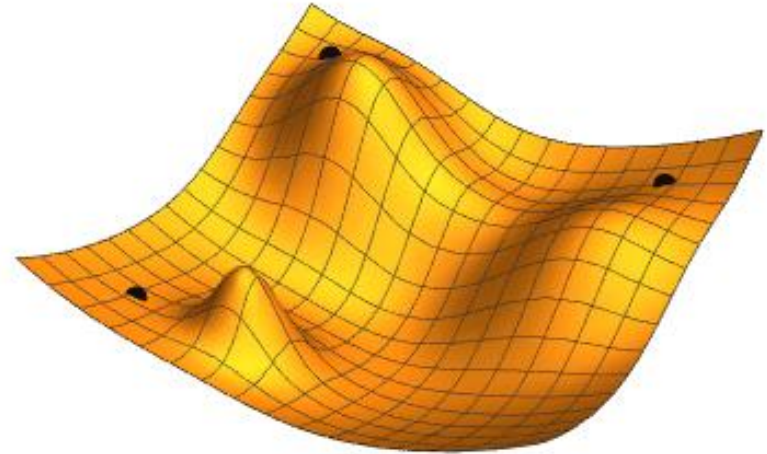


Nur wenige Algorithmen erlauben das Berechnen einer expliziten Lösung, z.B. lineare Regression. Andernfalls geht man iterativ vor:

- Zuerst wird ein Startpunkt gewählt
- Die Parameter werden etwas geändert (in Richtung des Abstiegs)
- Das wird so lange wiederholt, bis sich die Verlustfunktion nicht mehr verringert



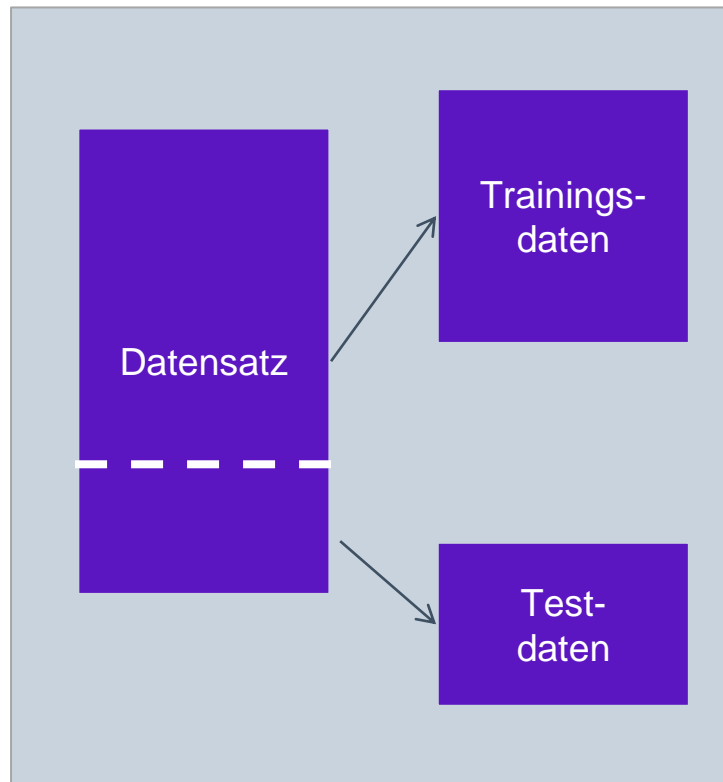
- Normalerweise hat man mehr als einen Parameter, künstliche neuronale Netze haben Millionen/Milliarden, d.h. man hat eine hochdimensionale Fläche
- Gradient Descent berechnet die Richtung des steilsten Abstiegs
- Mathematisch wird die mehrdimensionale Ableitung (Gradient) gebildet
- Das Resultat hängt vom Startpunkt ab
- Die Berechnung ist rechenintensiv, daher wird eine stochastische Approximation verwendet (stochastic gradient descent)



Gradient Descent bei zwei Parametern

Das Python-Package scikit-learn (sklearn) bietet für die Trennung eine einfache Funktion

```
import sklearn.model_selection as ms  
  
df_train, df_test =  
ms.train_test_split(df, test_size=0.2)
```



Hyperparameter bestimmen die grundlegenden Parameter eines Algorithmus, z.B. bei einem Clustering-Algorithmus die Anzahl Cluster oder bei einem neuronalen Netz dessen Aussehen (Anzahl Ebenen/Knoten/... Häufig passiert das, indem verschiedene Werte durchgespielt werden.

Es ist wichtig, den Testdatensatz erst ganz am Ende einzusetzen, um eine realistische Güte bestimmen zu können.

Daher wird der gesamte Datensatz in drei Datensätze aufgeteilt. Der Validierungsdatensatz ist dafür da, die Güte für die aktuelle Wahl der Hyperparameter zu bestimmen.

