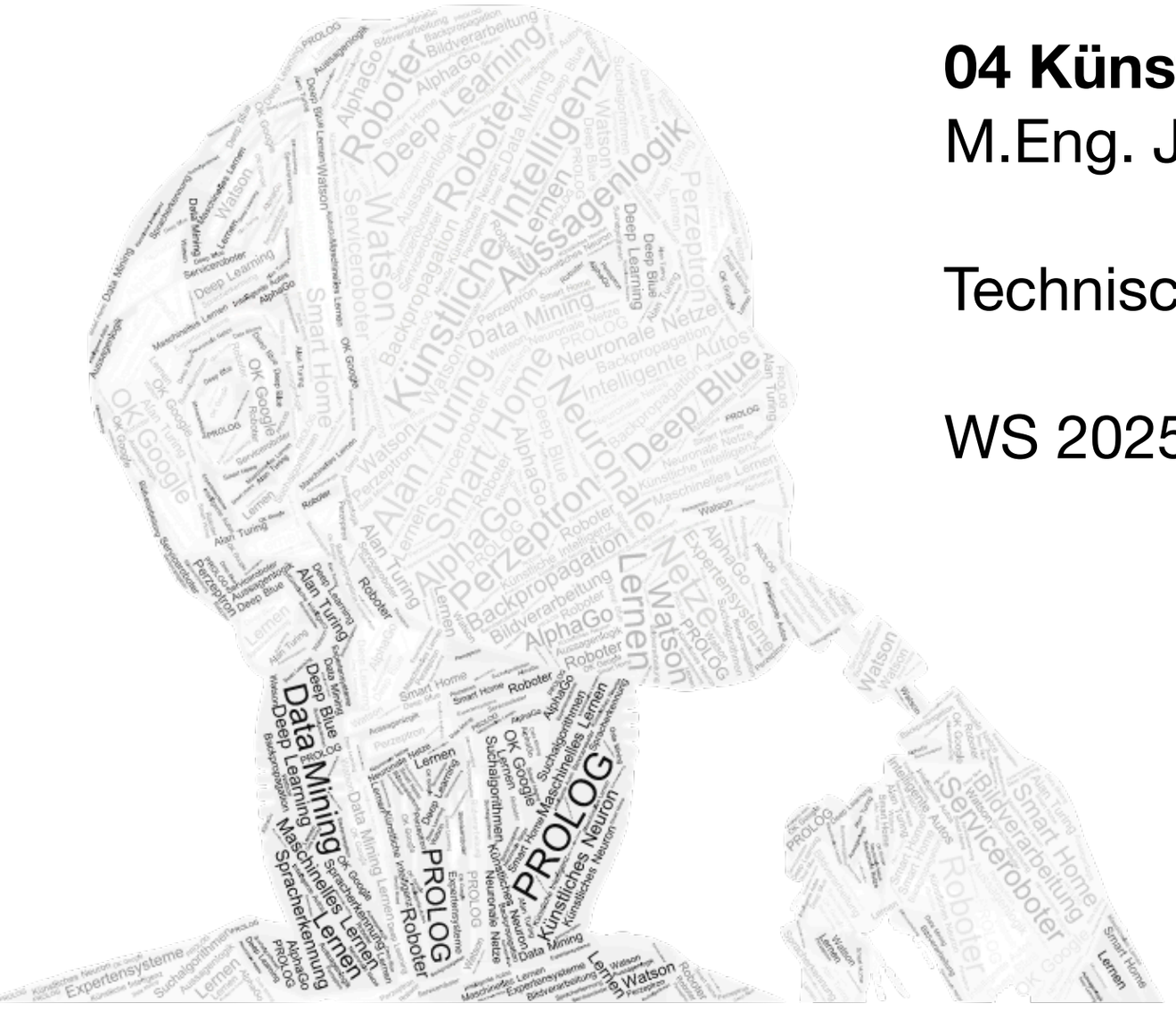


Künstliche Intelligenz

04 Künstliche Neuronale Netze

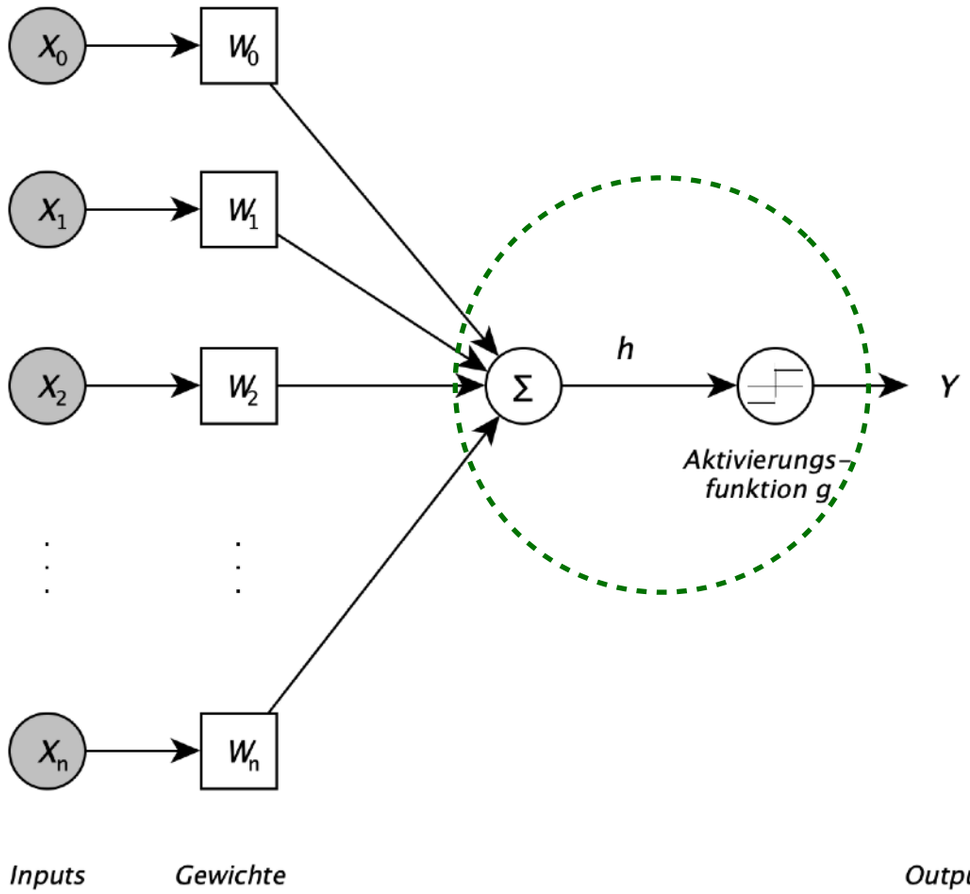
M.Eng. Janine Breßler

Technische Hochschule Wildau

WS 2025/2026

Künstliche Neuronale Netze

Das Künstliche Neuron



$$h(x) = \sum_{i=0}^{n} x_i w_i$$

y = g(h)

Künstliche Neuronale Netze

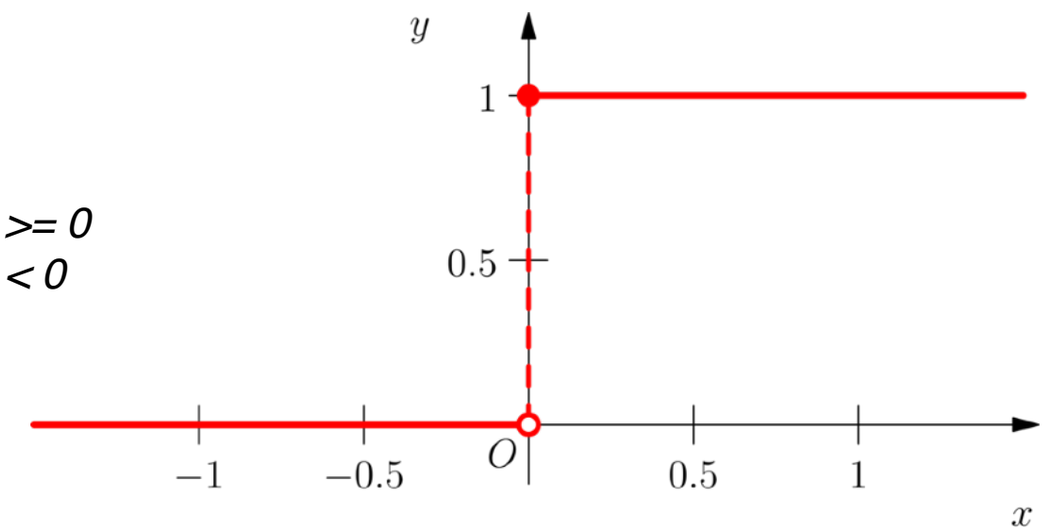
Das Perzeptron

1957 in den USA entwickelt, ursprünglich als Hardware-Modell

Verwendet als Aktivierungsfunktion **immer** die Heaviside-Funktion
➡ Klassifikation, da nur **endlich** viele Werte möglich (0 und 1)

Bei Regression **Spektrum** von Zahlen möglich, z.B. von 0 bis 1

$$H(x) = \begin{cases} 1 \text{ für } x >= 0 \\ 0 \text{ für } x < 0 \end{cases}$$



Heaviside-Funktion

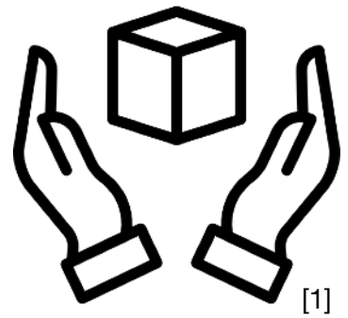
Das Perzeptron

Lernalgorithmus

1. Initialisiere Gewichte, z.B. alle $w_i = 0$ oder **kleinen zufälligen** Wert (Random)
2. Es gibt ein Trainingsset von Datenvektoren $\vec{(x_1)} bis \vec{(x_m)}$,
   $\vec{(x_i)} = (x_{i1} \ldots x_{in})$ $n = Anzahl\ der\ Inputs$,
   $jeweils\ mit\ korrektem\ Ergebnis\ (Label)\ d_i$
3. Für alle i = 1, …, m wird folgendes ausgeführt:
   - x_i als Input für das Perzeptron, dieses berechnet den Output y_i
   - Gewichte werden aktualisiert: $w_j \leftarrow w_j + \alpha * x_{i,j} * (d_j - y_j)$
   - α = Lernrate/Lernfaktor (0, 1]
4. Bei keinem zufriedenstellenden Ergebnis → wieder zu Punkt 2

Das Perzeptron am Beispiel*

Beispiel: Verkauf von einem bestimmten Produkt



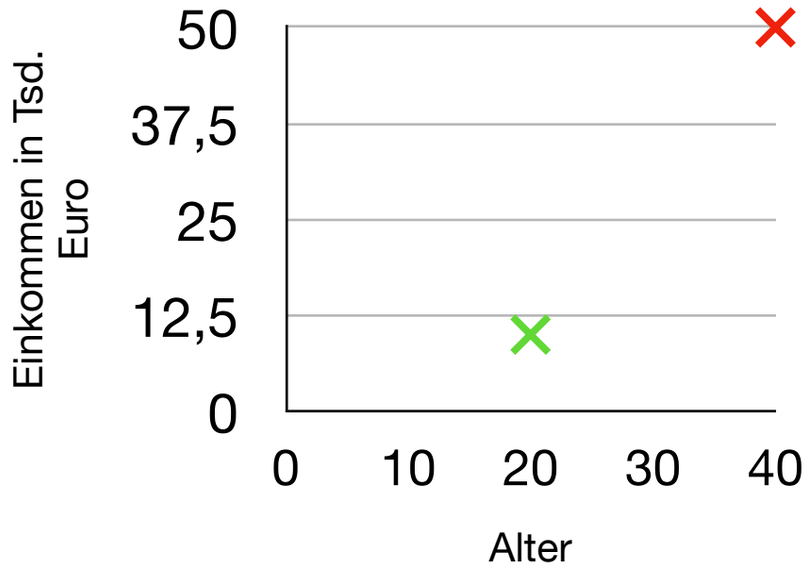
[1]

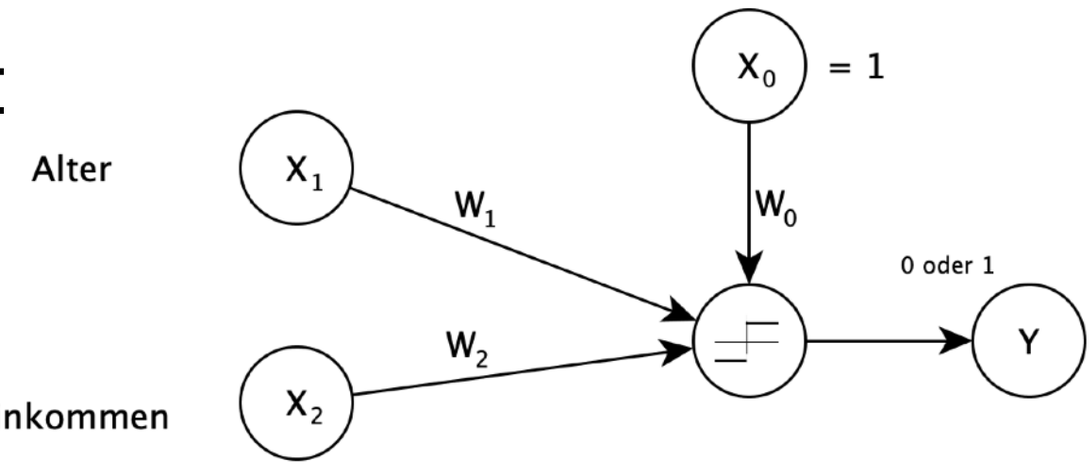
Perzeptron erhält zwei Eingänge, Alter und Einkommen

Zusätzlicher dritter Eingang **Bias** mit konstantem Wert wird meist als Schwellwert eingesetzt

Zwei Ergebnisse möglich

- 1 = Kunde kauft Produkt
- 0 = Kunde kauft nicht Produkt

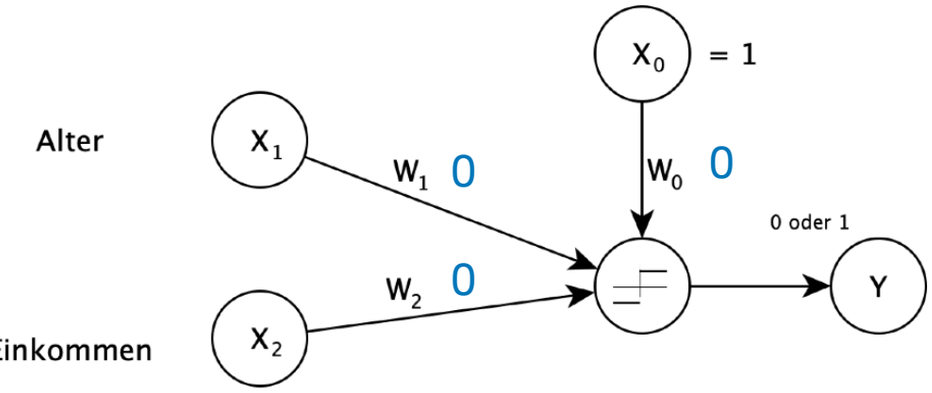




*Beispiel von Weitz / HAW Hamburg, Department Medientechnik

[1] Icon made by Pixel perfect from www.flaticon.com

Das Perzeptron am Beispiel



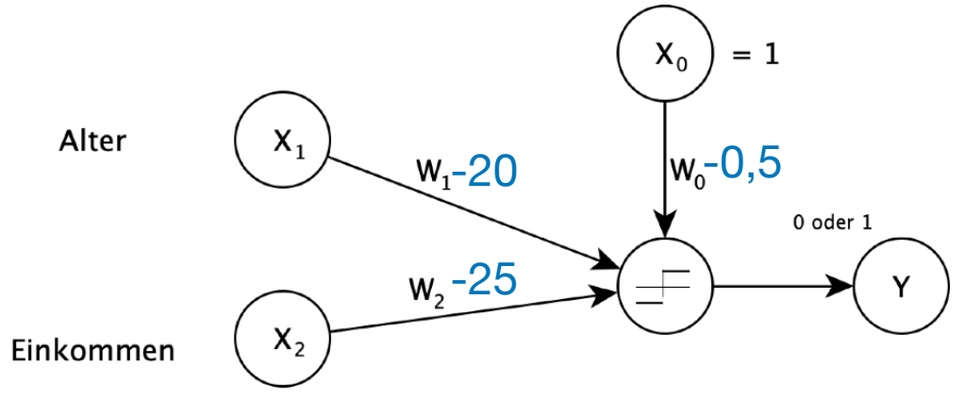
Trainingsvorbereitung:

- alle Gewichte auf 0: $w_0 = w_1 = w_2 = 0$ ← üblicherweise random
- Lernfaktor: $\alpha = 0{,}5$
- Trainingsset: (20, 10) → 1
  (40, 50) → 0 ← d (Label)

Trainingsdurchführung:

1. Für den ersten Trainingsdatensatz (20, 10)
   H(0 * 1 + 0 * 20 + 0 * 10) = H(0) = 1
   Vergleich berechneter und erwarteter Wert d: 1 == 1, also keine Korrektur des Gewichts
2. Für den zweiten Trainingsdatensatz (40, 50)
   H(0 * 1 + 0 * 40 + 0 * 50) = H(0) = 1
   Vergleich berechneter und erwarteter Wert d: 0 != 1, also Korrektur des Gewichts:
   w0 ← w0 + 0,5 * 1 * (0 - 1) = 0 - 0,5 = - 0,5
   w1 ← w1 + 0,5 * 40 * (0 - 1) = 0 - 20 = - 20
   w2 ← w2 + 0,5 * 50 * (0 - 1) = 0 - 25 = - 25

Wiederholung der Schritte 1 und 2, bis jeweils das gewünschte Ergebnis (Label) bei Eingabe der Trainingsdaten berechnet wird!

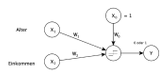


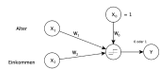
Das Perzeptron am weiteren Beispiel

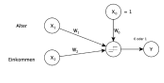
Es entsteht eine Gerade (Hyperebene), die aus dem Ursprung verschoben ist

Diese dient als Klassifikator, wenn die zwei Mengen **linear separierbar** sind:

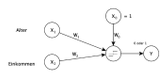
Vorhersage mit Testset:

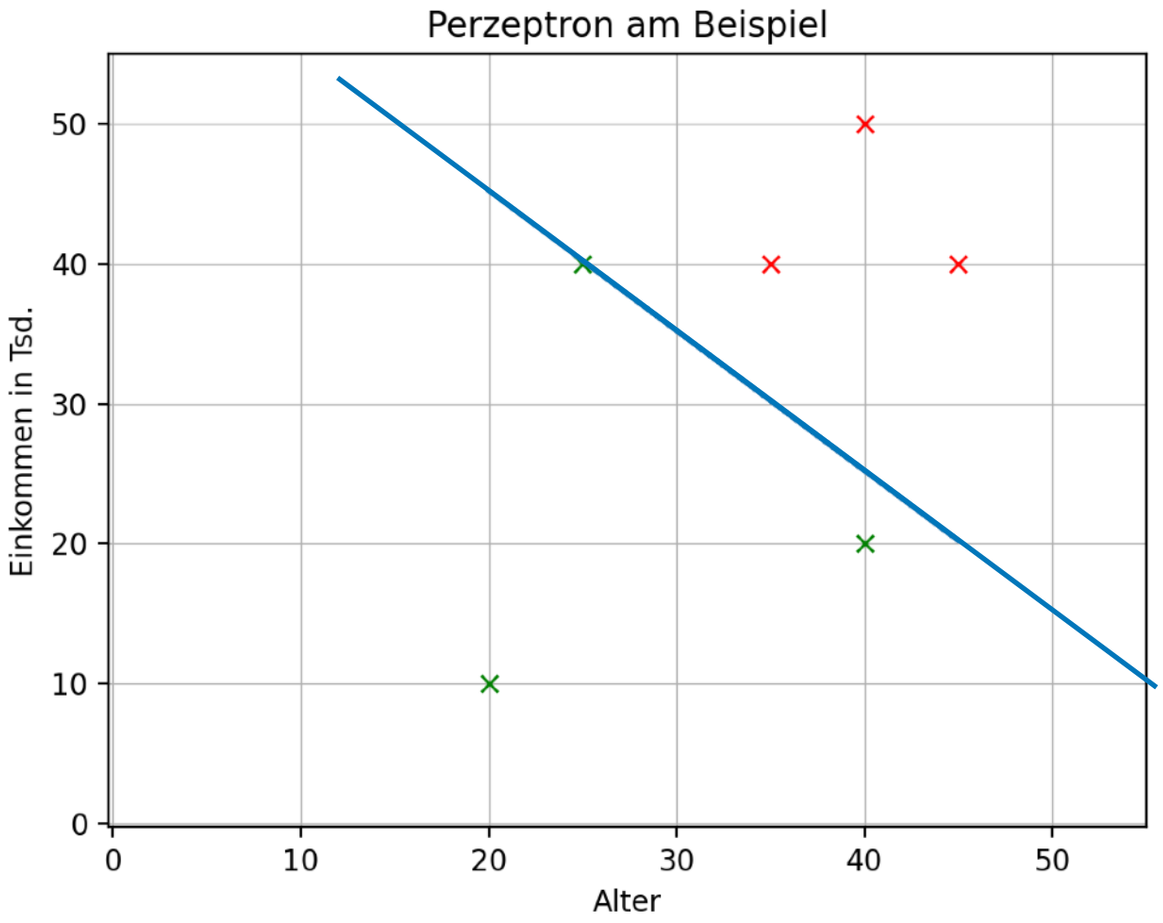
(25, 35)  1

(40, 45)  0

(45, 15)  1

(30, 48)  0

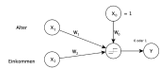
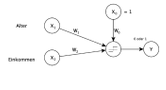
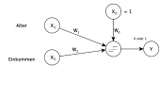
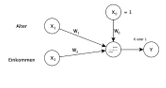
(25, 41)  0

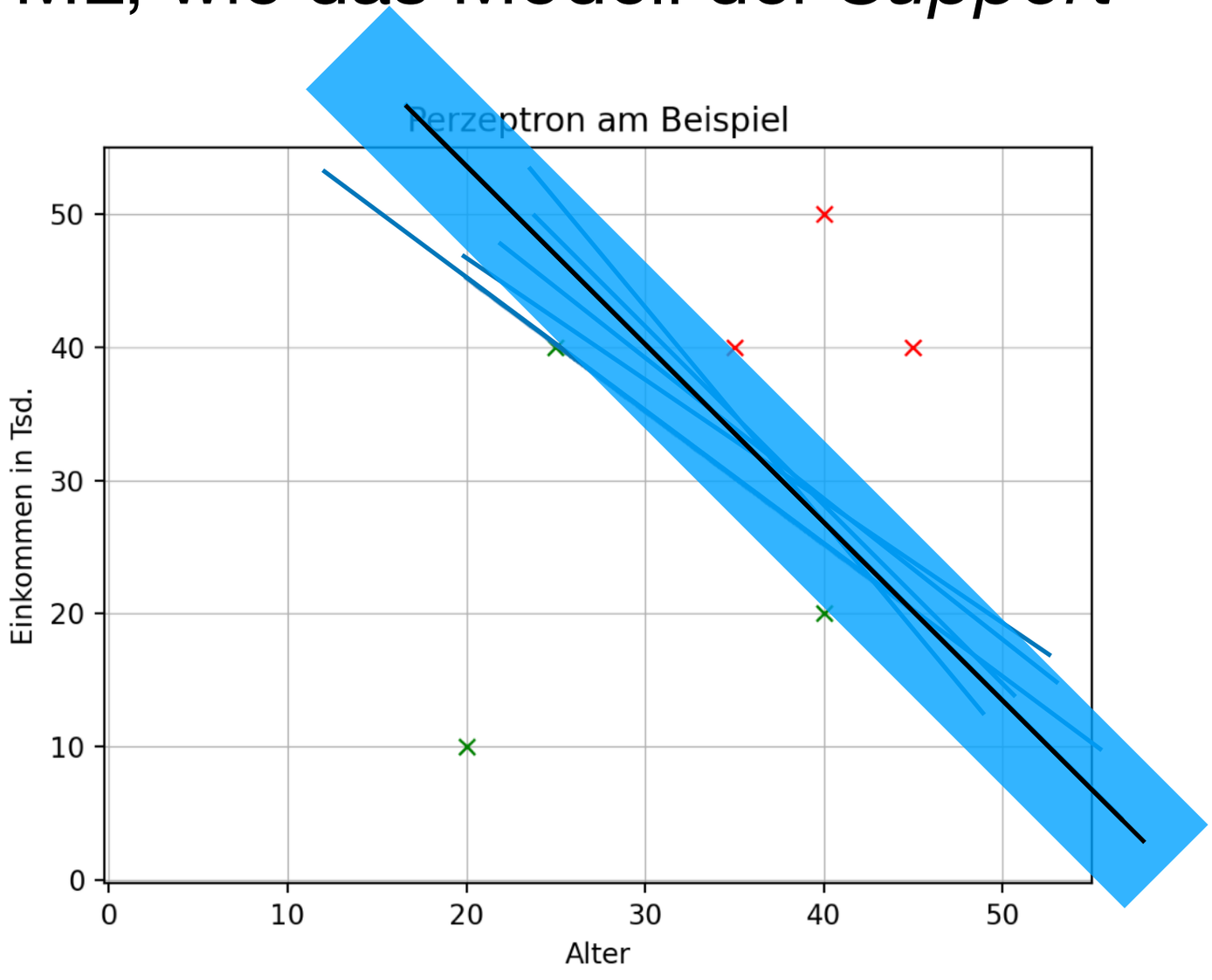


Das Perzeptron am Beispiel

Beim Perzeptron ist die gewählte Hyperebene nicht immer optimal.

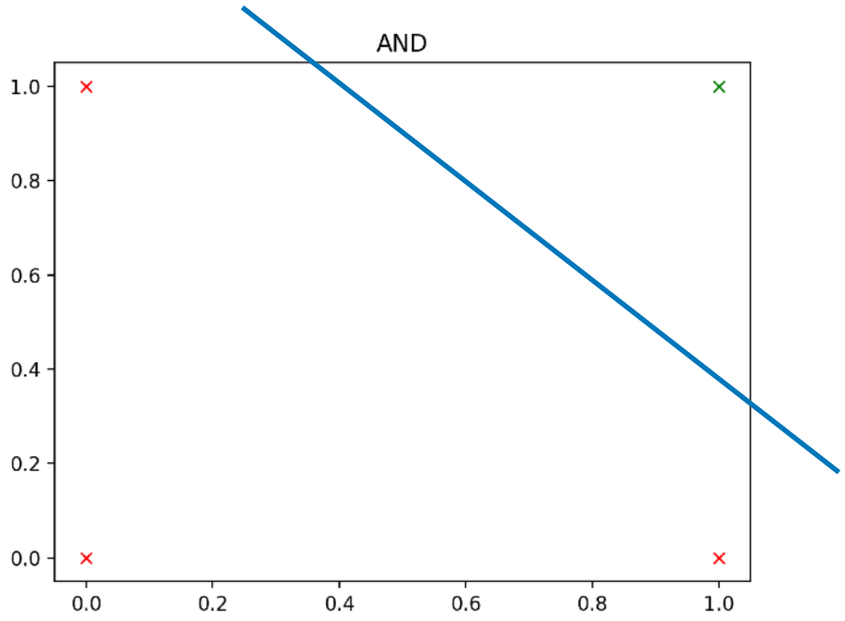
Andere Ansätze des ML, wie das Modell der *Support Vector Machine*, bestimmen diese besser

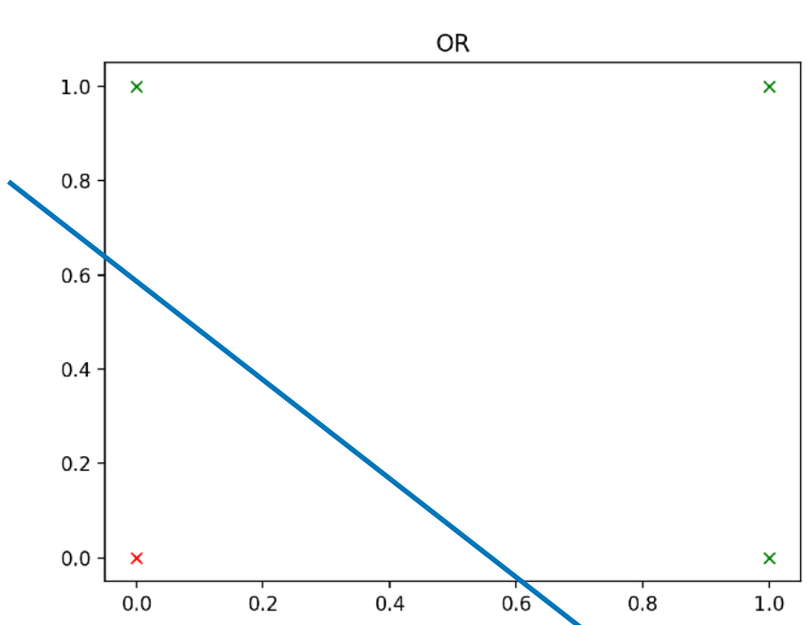
(25, 35)  1
(40, 45)  0
(45, 15)  1
(30, 48)  0
(25, 41)  1

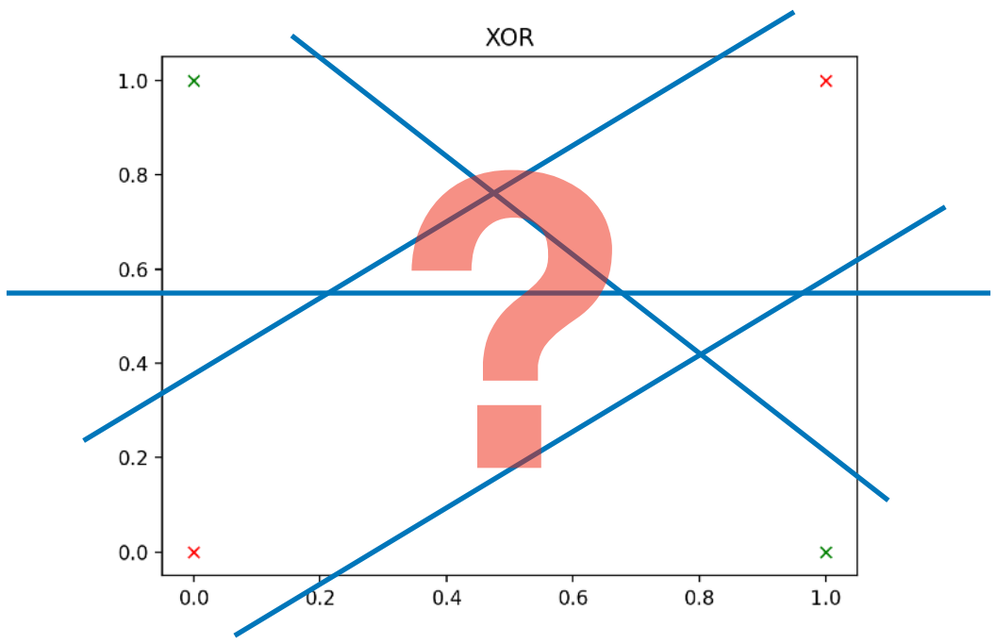


Das Perzeptron am Beispiel

Die Grenzen des Perzeptrons liegen darin, wenn zwei Mengen nicht mehr durch eine Gerade trennbar sind:







Komplexere Strukturen sind nötig, in denen Neuronen neben- und hintereinander geschaltet sind.

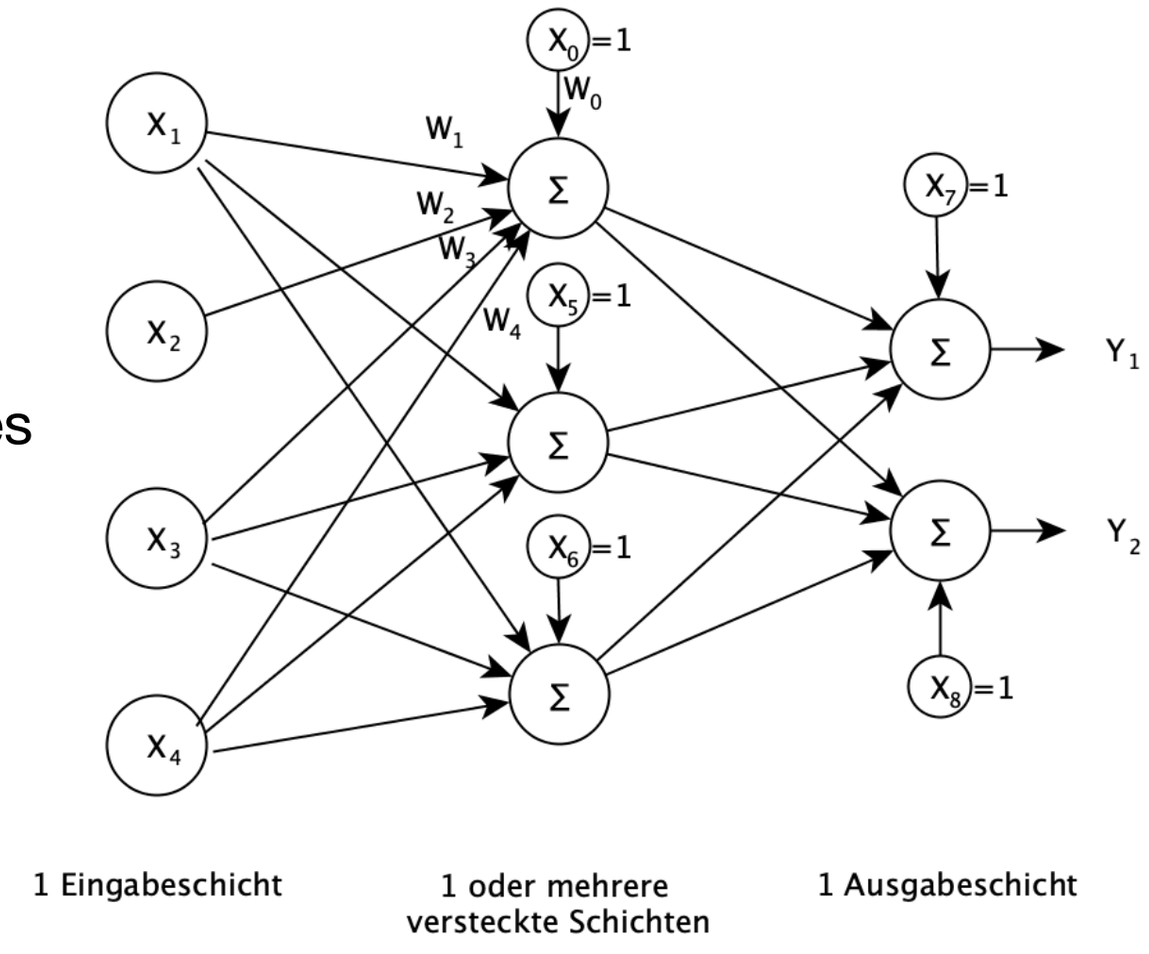
Mehrlagiges Netzwerk

Ein mehrlagiges Netzwerk aus vielen Neuronen.

Idee: Output von Neuronen wird Input von anderen Neuronen

Es entstehen verdeckte bzw. versteckte Schichten (Blackbox)

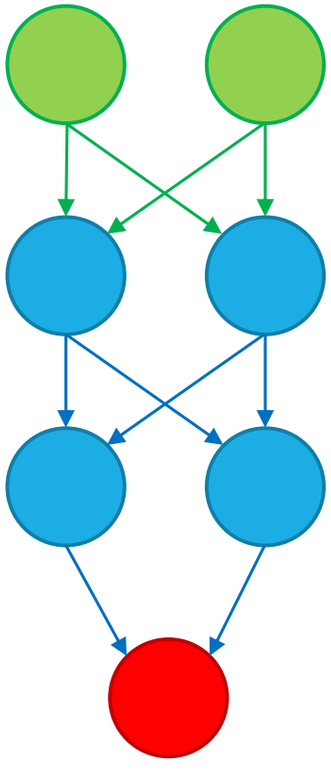
Beispiel eines Feed-Forward-Netzes
mit 5 Neuronen



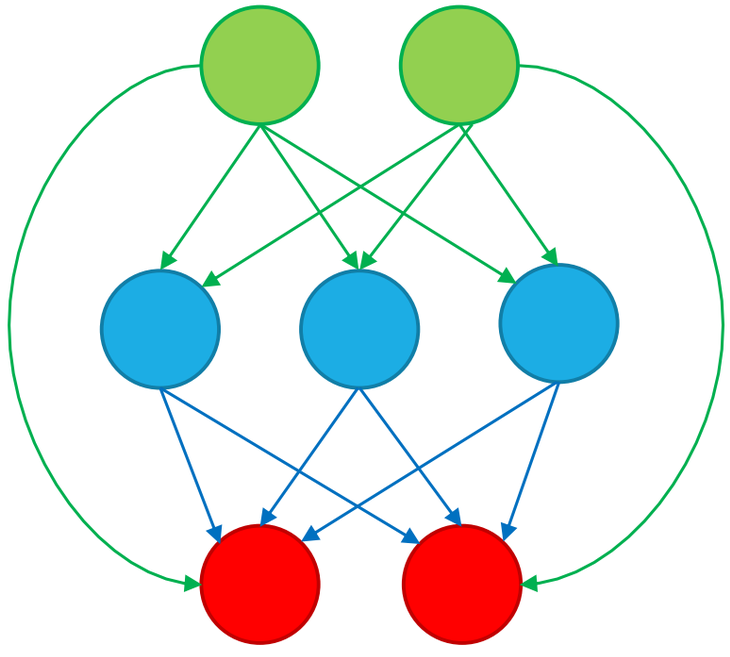
Künstliche Neuronale Netze

Architektur:

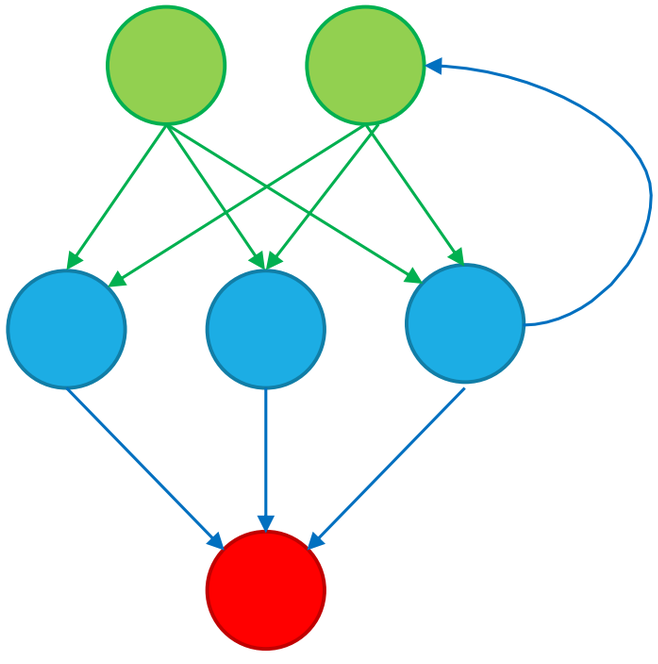
Feed-Forward-Netz



Feed-Forward-Netz
mit shortcut connection



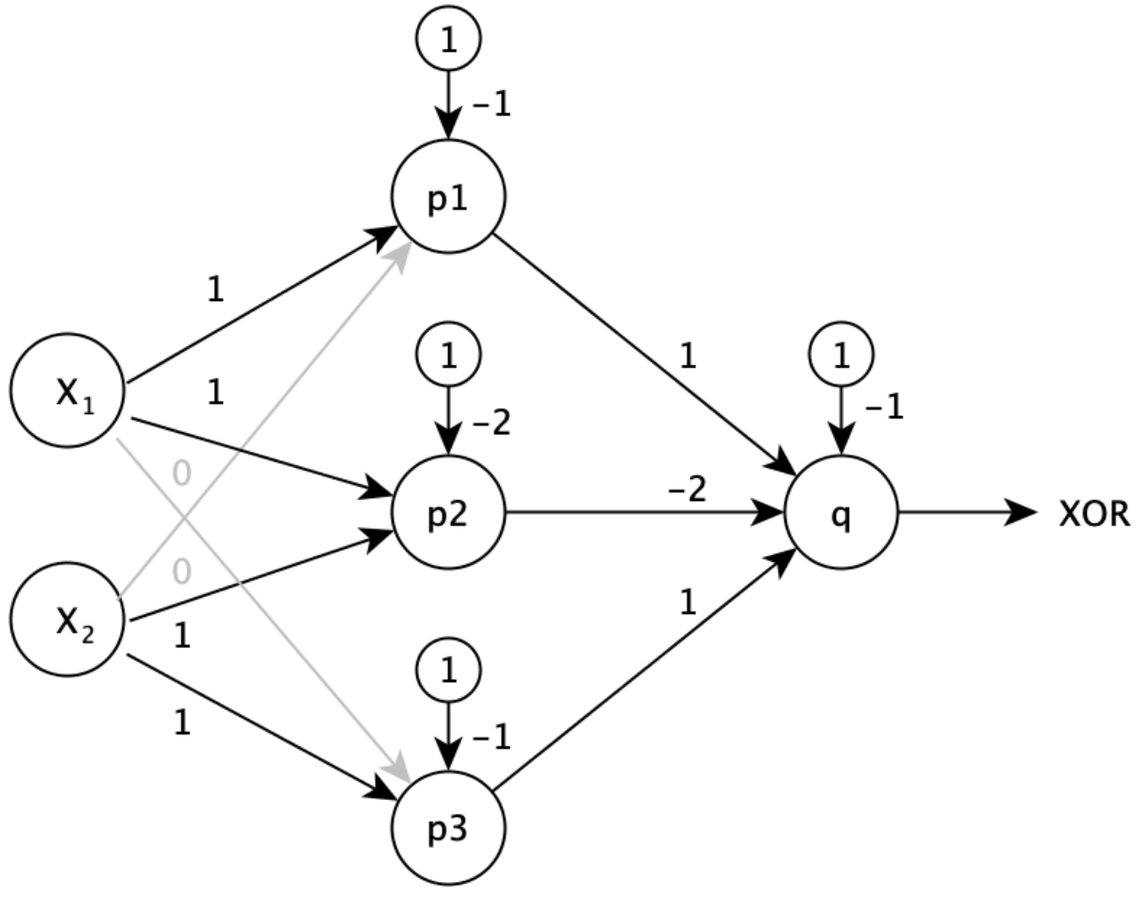
Rekurrentes neuronales-Netz



Mehrlagiges Perzeptron

Aufbau mehrlagiges Perzeptron zur Berechnung von XOR:

- Eingangsschicht mit den zwei Eingangswerten X_1 und X_2
- Eine verdeckte Schicht à 3 Neuronen p1, p2, p3
- Ein Ausgangsneuron q
- Aktivierungsfunktion: Heaviside



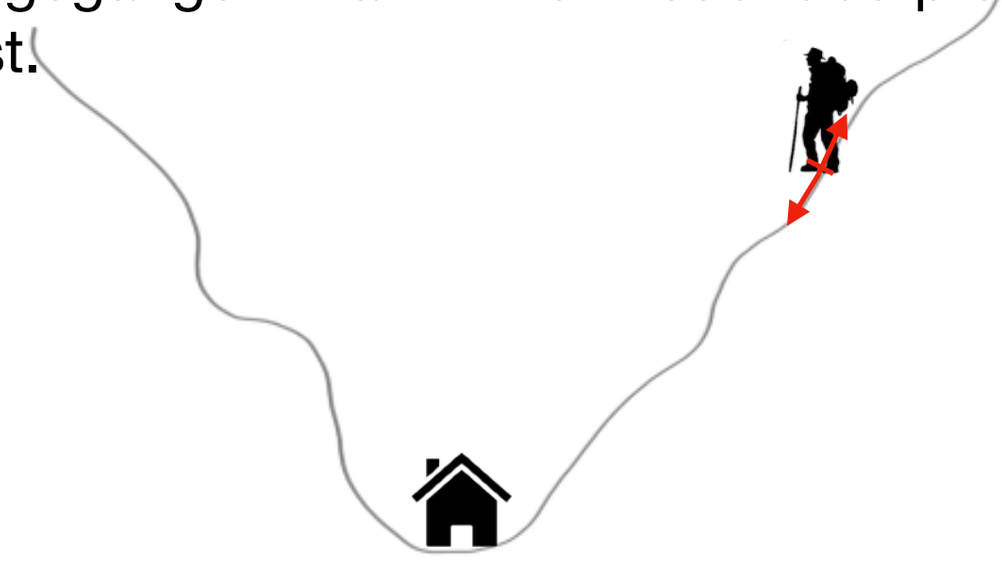
Künstliche Neuronale Netze

Lernverfahren Backpropagation

Gradientenverfahren:
Verfahren, um den minimalen Fehler in einer Fehlerfläche zu finden.
Auch als globales Minimum bezeichnet

Assoziation:
Gebirge (2-dimensional)
Bergwanderer/in
Gesucht: Tiefste Punkt des Gebirges (Home sweet Home)
Strategie: ?

Überprüfen aller möglichen Laufrichtungen, in die Richtung mit dem größten Abstieg werden ein paar Schritte gegangen. Dann wird wieder überprüft, so lange, bis Minimum gefunden ist.

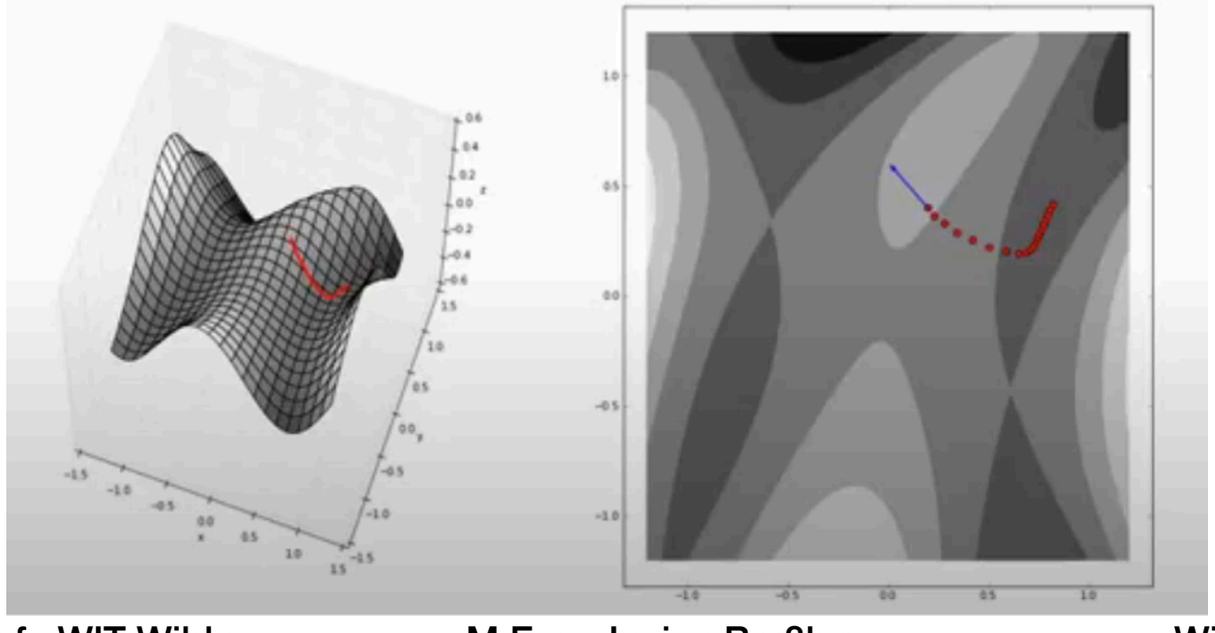


Künstliche Neuronale Netze

Lernverfahren Backpropagation

Gradientenverfahren:
Verfahren, um den minimalen Fehler in einer Fehlerfläche zu finden.
Auch als globales Minimum bezeichnet

Eigenschaft:
- Gradient zeigt in Richtung des steilsten Anstiegs
- Umgekehrte Richtung zeigt in Richtung des steilsten Abstiegs
- Gradient ist länger desto steiler der Anstieg
- Schrittgröße wird in Abhängigkeit der Größe des Gradienten gewählt
  - sehr steil, große Schritte
  - weniger steil, weniger große Schritte

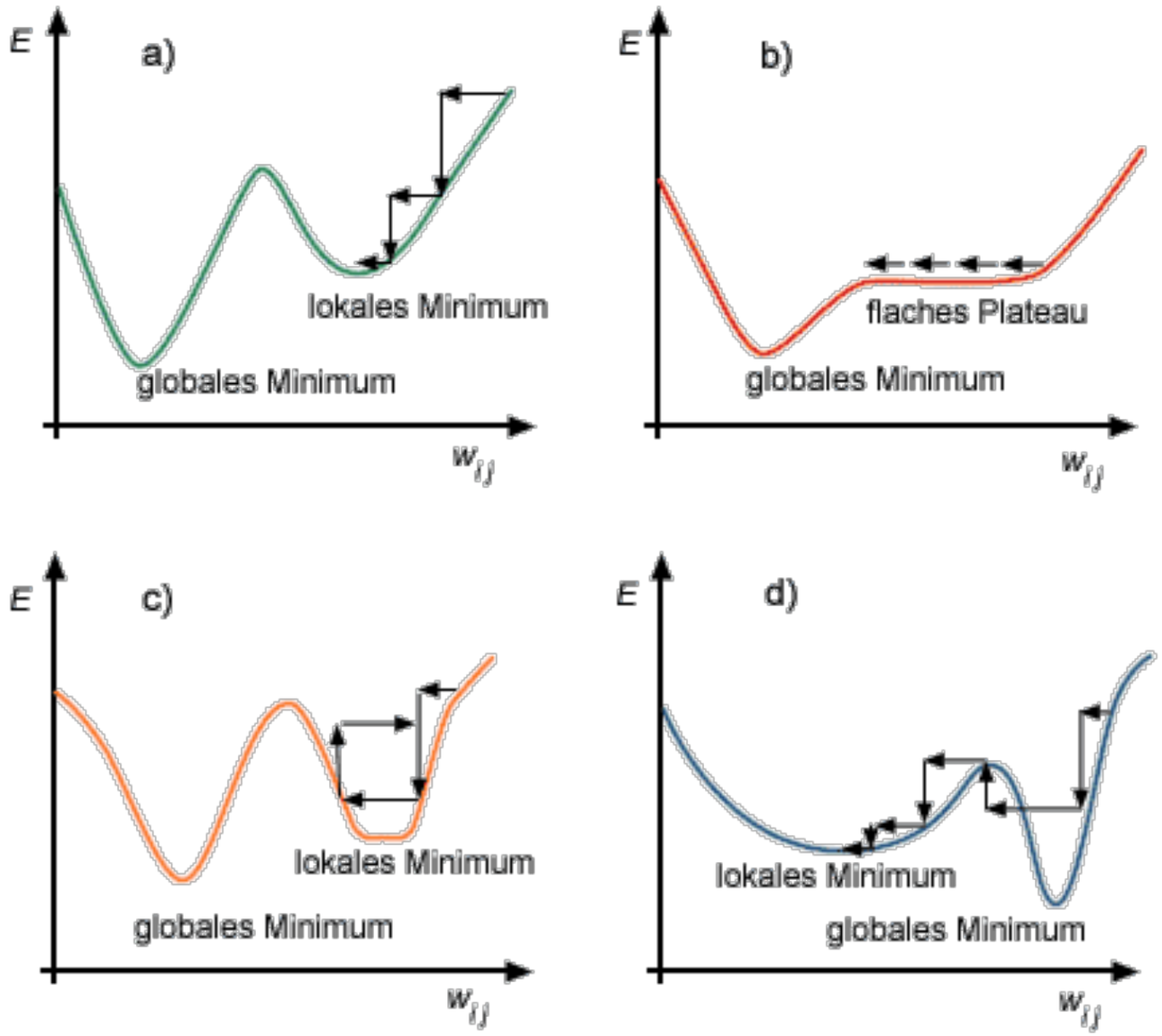


Quelle: Weitz / HAW Hamburg
Department Medientechnik

Künstliche Neuronale Netze

Lernverfahren Backpropagation

Problem: keine Information über die Fehlerfläche insgesamt hat, sondern nur aus der Kenntnis der lokalen Umgebung

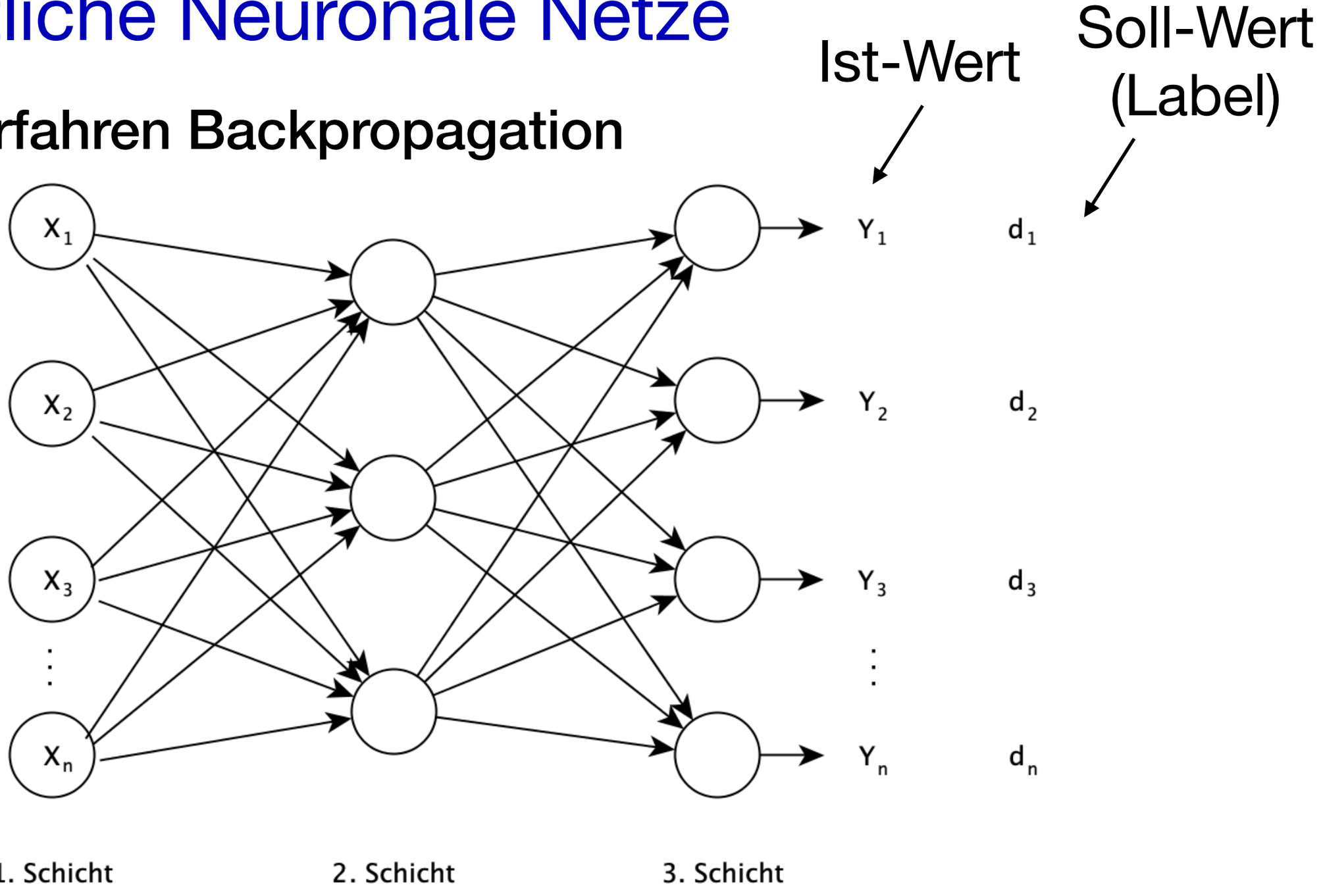


a) lokale Minima der Fehlerfläche
b) Plateaus
c) Oszillation durch große Gradienten
d) Verlassen von Minima zu lokalen Minima

Quelle: http://www.chemgapedia.de/

Künstliche Neuronale Netze

Lernverfahren Backpropagation

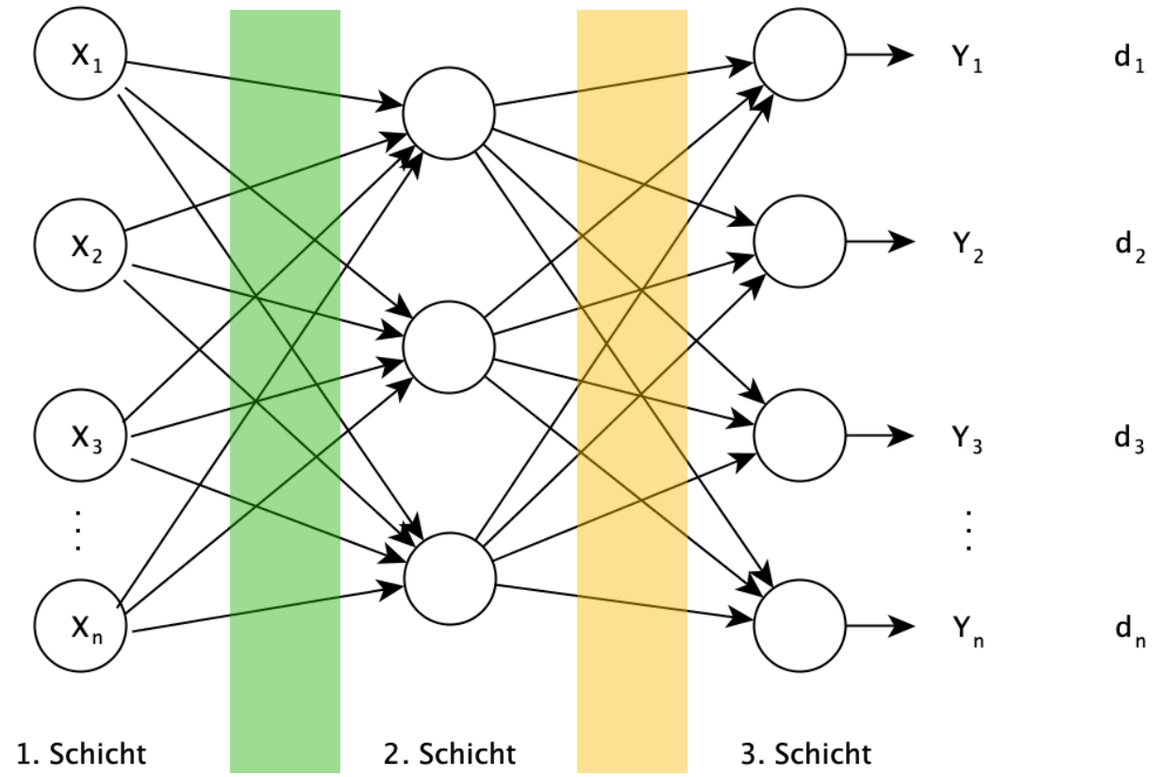


$$\sum (y_i - d_i)^2 = E$$

Größe von E ist abhängig von den Gewichten im Netzwerk!

Künstliche Neuronale Netze

Lernverfahren Backpropagation



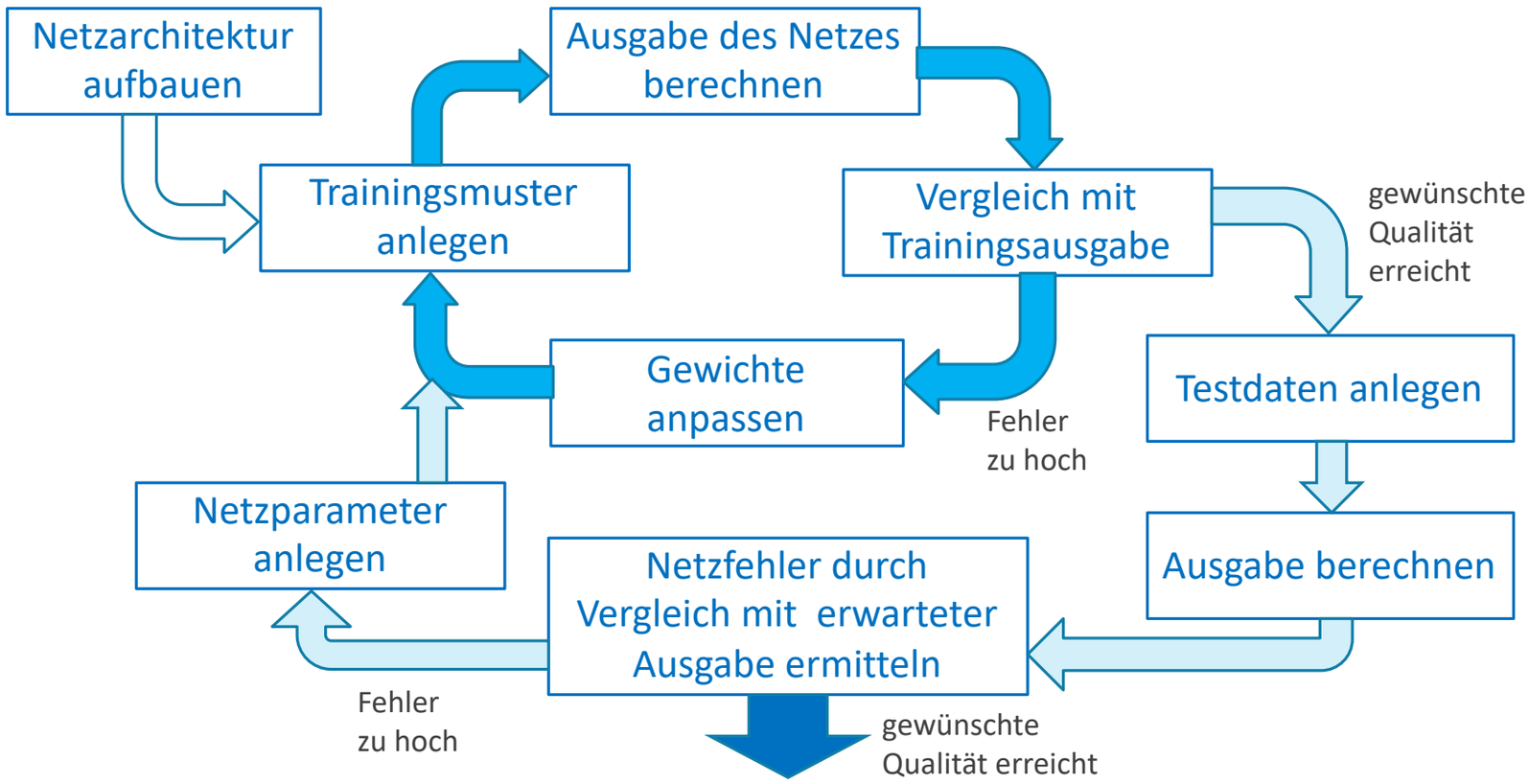
Gradienten werden für die Gewichte der letzten Schicht berechnet. Mit Hilfe von Gradient werden die Gewichte korrigiert.

Anschließend werden die Gradienten für die vorletzte Schicht berechnet und damit die Gewichte angepasst. Dabei kann auf die bereits berechneten Gradienten zurückgegriffen werden.

Gradient wird stückweise von hinten nach vorne berechnet —> **Fehlerrückführung**

Künstliche Neuronale Netze

Trainingsdurchführung:



- Ein kompletter Durchlauf aller Trainings-Daten wird als *Epoche* bezeichnet.
- Anzahl der Trainings-Epochen ist ein wichtiger Hyperparameter für das Training von neuronalen Netzen.

Künstliche Neuronale Netze

Metriken zur Bewertung des trainierten Künstlichen Neuronales Netzes

Beispiel: Spamfilter

	Positive	Negative
Predicted Positive *Spam*	**True Positive (TP):** Eine Spam-E-Mail wurde vom KNN **richtig** als Spam klassifiziert	**False Positive (FP):** Eine normale E-Mail wurde **fälschlicher Weise** als Spam klassifiziert
Predicted Negative *Kein Spam*	**False Negative (FN):** Eine Spam-E-Mail wurde als **fälschlicher Weise** als kein Spam klassifiziert	**True Negative (TN):** Eine normale E-Mail wurde vom KNN **richtig** als kein Spam klassifiziert

Künstliche Neuronale Netze

Metriken zur Bewertung des trainierten Künstlichen Neuronales Netzes

Accuracy:

$$Accuracy = \frac{True\ Positive + True\ Negative}{True\ Positive + True\ Negative + False\ Positive + False\ Negative}$$

Bei einem ausgewogenen Datenset (Verteilung der Daten auf die Klassen) kann die Genauigkeit / Accuracy ein grober Indikator für die Modellqualität sein. Es sollten jedoch weitere Messwerte betrachtet werden.

Künstliche Neuronale Netze

Metriken zur Bewertung des trainierten Künstlichen Neuronales Netzes

Precision:
Maß für korrekt identifizierte positive Klassifizierung gegenüber allen positiven Klassifizierungen. Wichtig, wenn False-Positive Klassifizierungen „teuer" sind.

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

Recall:
Maß für korrekt identifizierte positive Klassifizierung gegenüber allen positiven Daten im Datenset. Wichtig, wenn False-Negative Klassifizierungen teuer sind.

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

Künstliche Neuronale Netze

Metriken zur Bewertung des trainierten Künstlichen Neuronales Netzes

F1 - Score:
Harmonisches Mittel aus Precision und Recall. Bei unausgewogenen Datensets (Verteilung der Daten auf die Klassen) ist der F1-Score bessere Metrik zur Bewertung des Modells gegenüber der Accuracy.

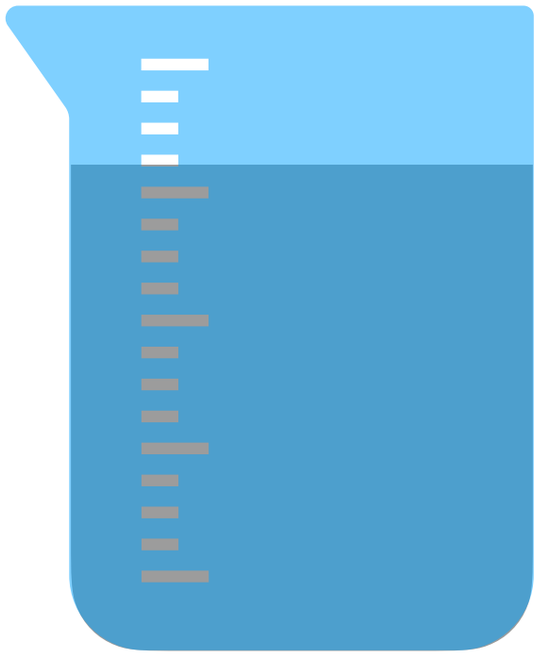
$$F1 = \frac{2 * Precision * Recall}{Precision + Recall}$$

Künstliche Neuronale Netze

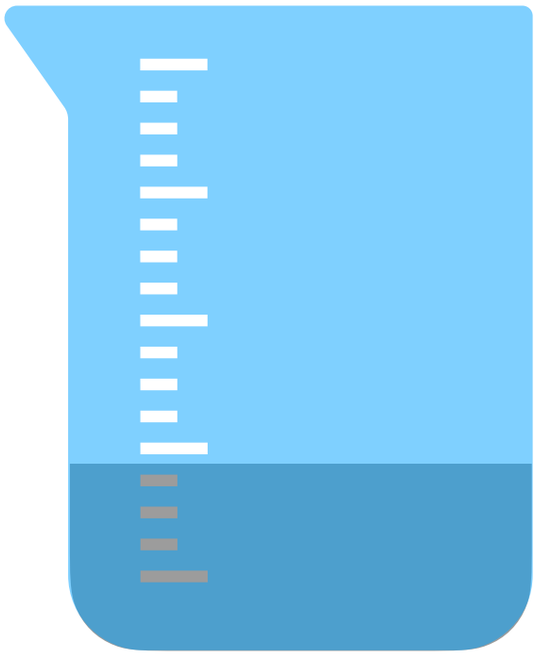
Faustregel:
Gelabelte Daten werden in 80% Trainings- und 20% Testdaten aufgeteilt!

Güte der Testdaten ist essentiell für Güte des Neuronalen Netzes!

80% Trainingsdaten



20% Testdaten



Künstliche Neuronale Netze

Overfitting und Underfitting

Bei der Generalisierung durch ein Künstliches Neuronales Netz können zwei Probleme auftreten

- **Overfitting:** das Modell ist überangepasst
  - reagiert gut auf die Trainingsdaten und schlecht auf die Testdaten
  - zu stark auf Beispiele abgestimmt
  - irrelevante Unterschiede oder statistisches Rauschen werden mit einbezogen
- **Underfitting:** das Modell ist zu wenig angepasst
  - reagiert bereits schlecht auf Trainingsdaten
  - zu wenig Ausdruckskraft, um relevante Unterschiede berücksichtigen zu können

Künstliche Neuronale Netze

Overfitting Beispiel

Trainingsdaten

- Einzelne Logos der Flugzeuge werden mit einbezogen
- Geometrie von Flugzeugen wird nicht ausreichend berücksichtigt

Testdaten


- Abhilfe:
  - Regularisierung: z.B. Reduzierung der Parameter (Dropout)
  - bessere Beispieldaten, hier z.B. Flugzeuge ohne Logos

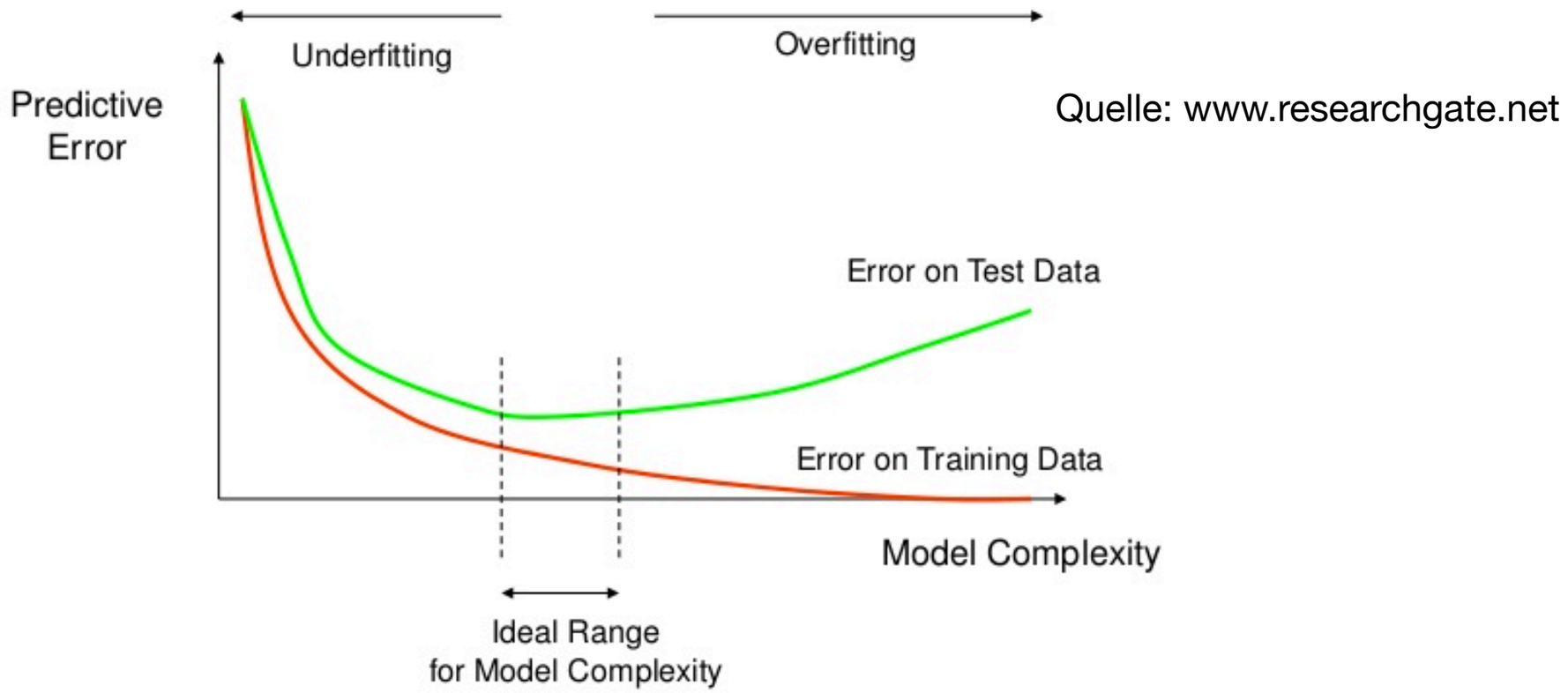
Künstliche Neuronale Netze

Underfitting Beispiel

- Zu wenig Merkmale/Features verwendet:
  - *Größe*
  - *Anzahl der Beine*

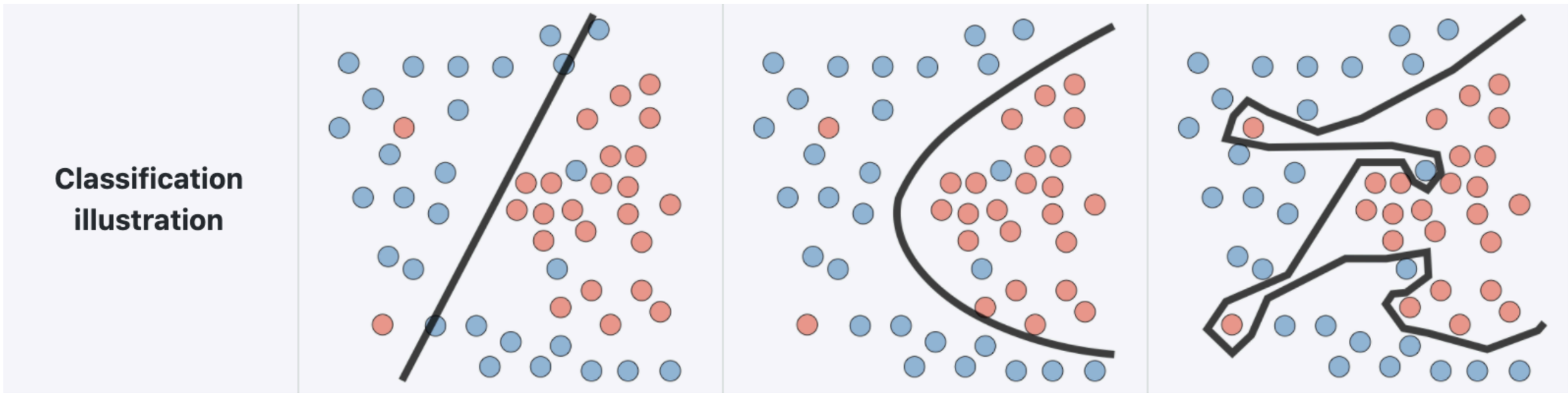
- Abhilfe:
  - ⚙ Weitere(s) Merkmal(e): z.B. *Vorhandensein von Hörnern*

Künstliche Neuronale Netze

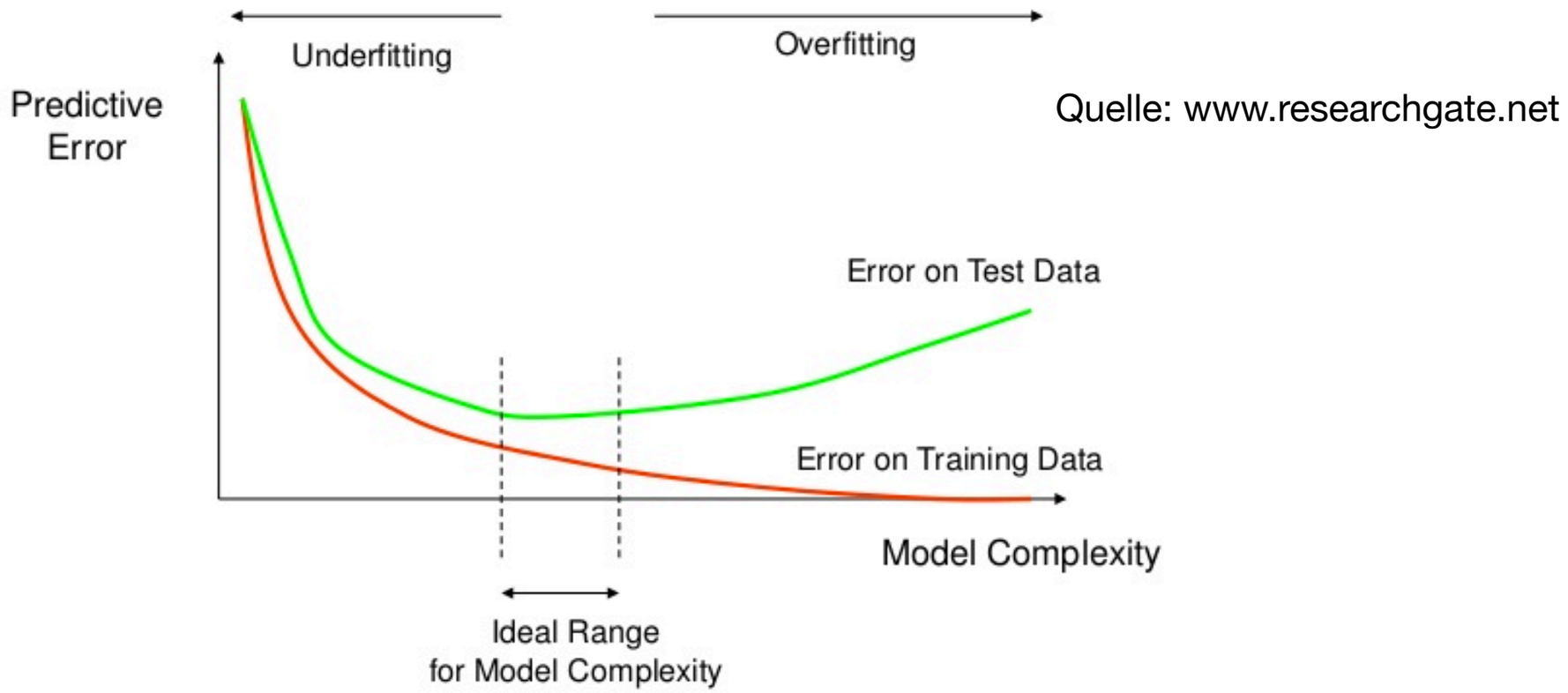


Quelle: www.researchgate.net

Quelle: www.kaggle.com

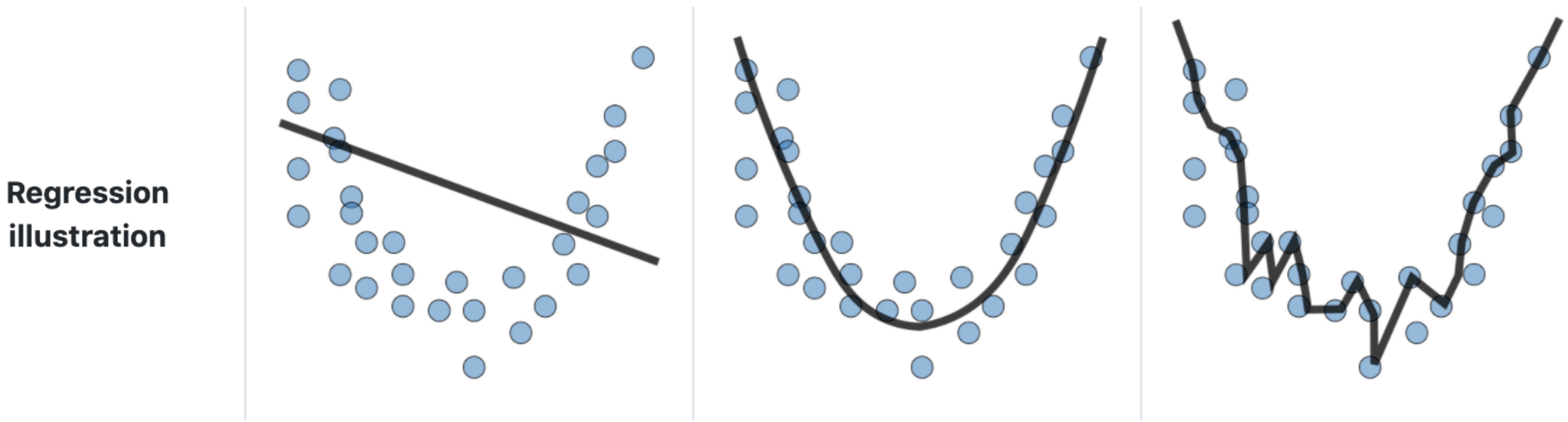


Künstliche Neuronale Netze

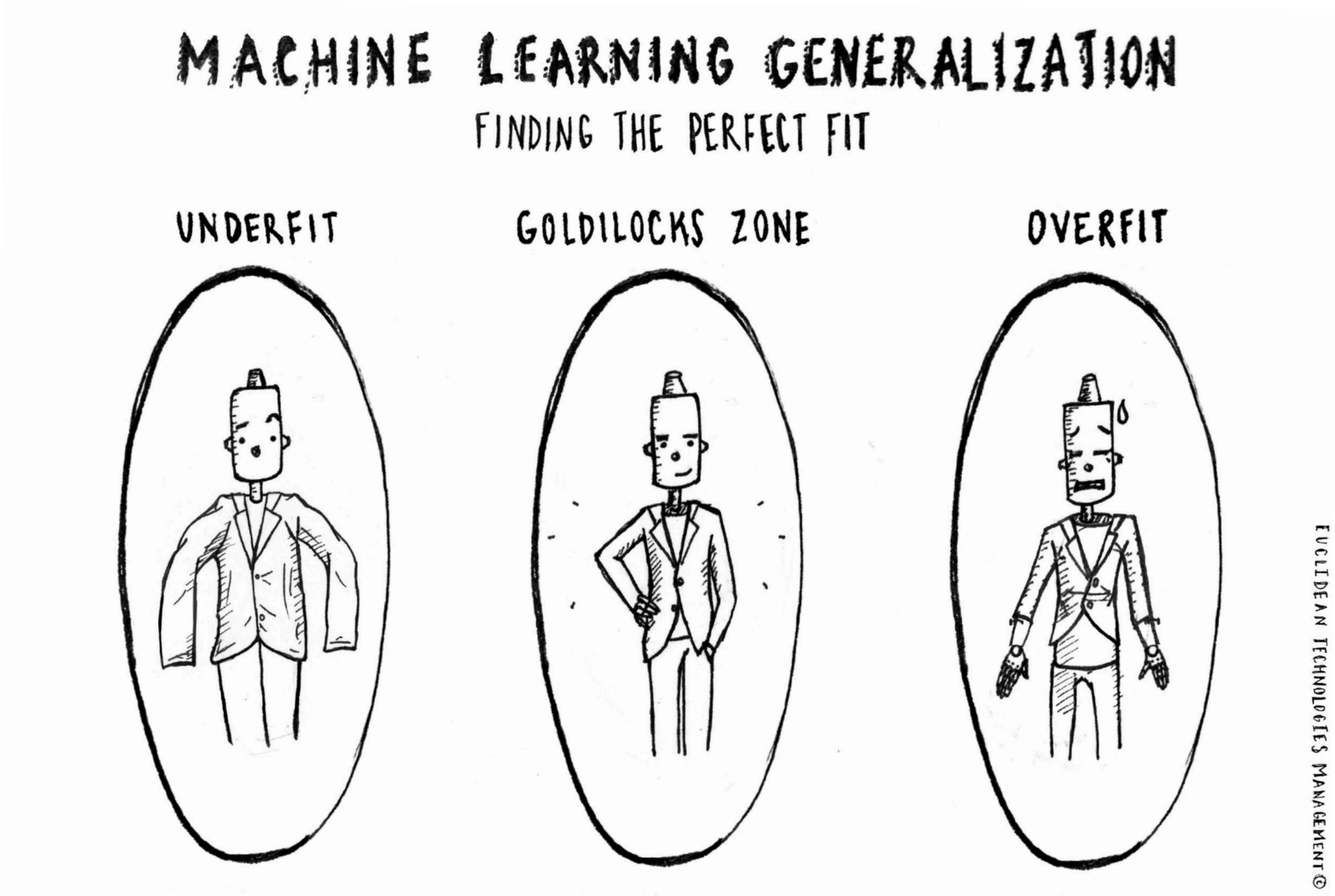


Quelle: www.researchgate.net

Quelle: www.kaggle.com



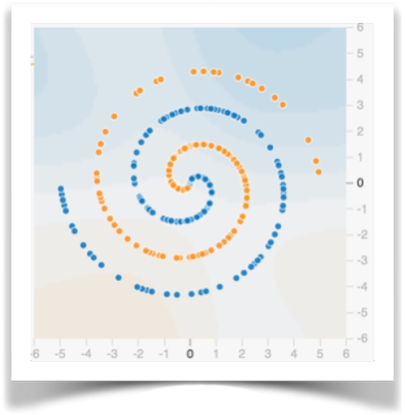
Künstliche Neuronale Netze



Künstliche Neuronale Netze

Übersicht Stellschrauben Künstliche Neuronale Netze

- Trainingsdaten
- Trainingsdurchläufe (Epochen)
- Architektur des Künstlichen Neuronalen Netzes
  - Anzahl versteckter Schichten und jeweiliger Neuronen
  - Bias
  - Aktivierungsfunktion
  - Regularisierung (z.B. Dropout, L1 und L2)
- Lernrate
- Lernverfahren



Übungsaufgaben

1. Erklären Sie mit eigenen Worten, warum es mit einem einzelnen Perzeptron nicht möglich ist, eine XOR-Funktion erfolgreich anzutrainieren.

2. Führen Sie auf dem Papier das Beispieltraining von Folie 7 zu Ende, indem Sie solange die einzelnen Gewichte neu berechnen, bis das Perzeptron den Ergebniswert der Trainingsdaten richtig berechnet.
   Wie viele Durchläufe sind nötig, damit die Trainingsdaten von dem Perzeptron richtig bestimmt werden?