

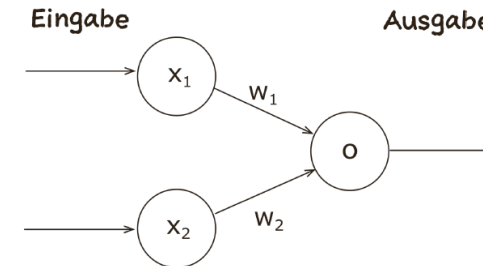
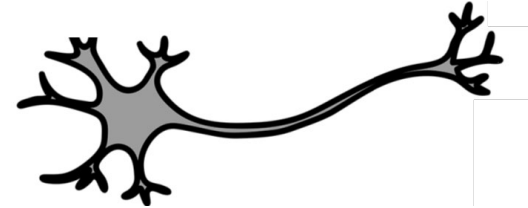
Künstliche Intelligenz kapieren und programmieren

Teil 5: Perzeptron

Michael Weigend
Universität Münster



mw@creative-informatics.de
www.creative-informatics.de
2024



Materialien bei GitHub:
<https://github.com/mweigend/ki-workshop>

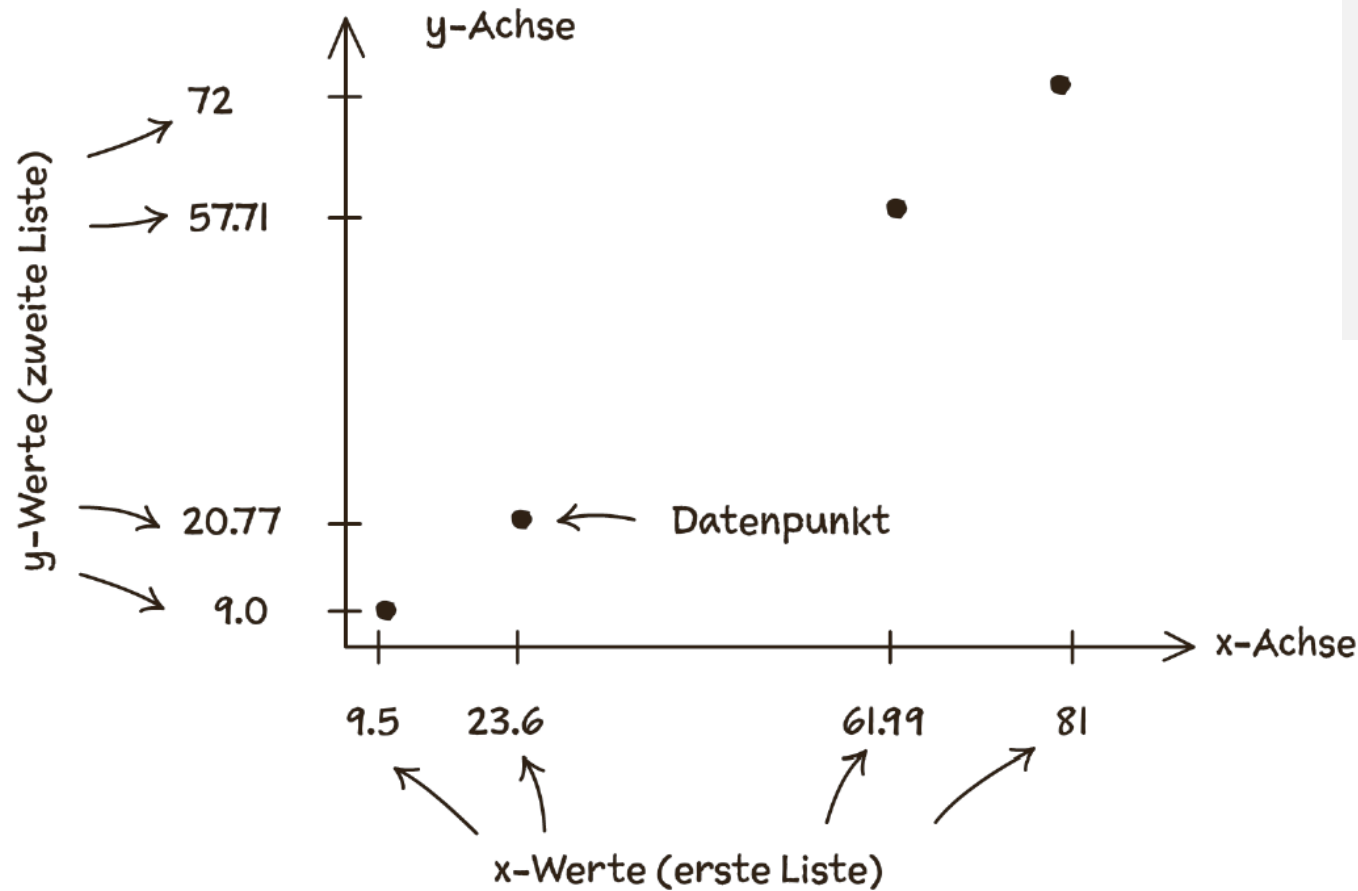
Tag 2

Zeit	Thema	Inhalte
9.00	Perzeptron	Neuron, Aktivierungsfunktion, Daten visualisieren mit Matplotlib, Rosenblatt-Perzeptron für logische Operationen
11.00	Aus Fehlern lernen	Error-Backpropagation, einfaches künstliches neuronales Netz (KNN) mit verborgenen Knoten
12.30	<i>Mittagspause</i>	
13.30	Ziffern erkennen	NumPy, KNN mit Array-Operationen, das Ziffern erkennen kann
15.00	Anwendung von KI	Verkehrsschilder erkennen, Gesichter erfassen, Experimente mit OpenCV, Schlussrunde
16.00	<i>Ende</i>	

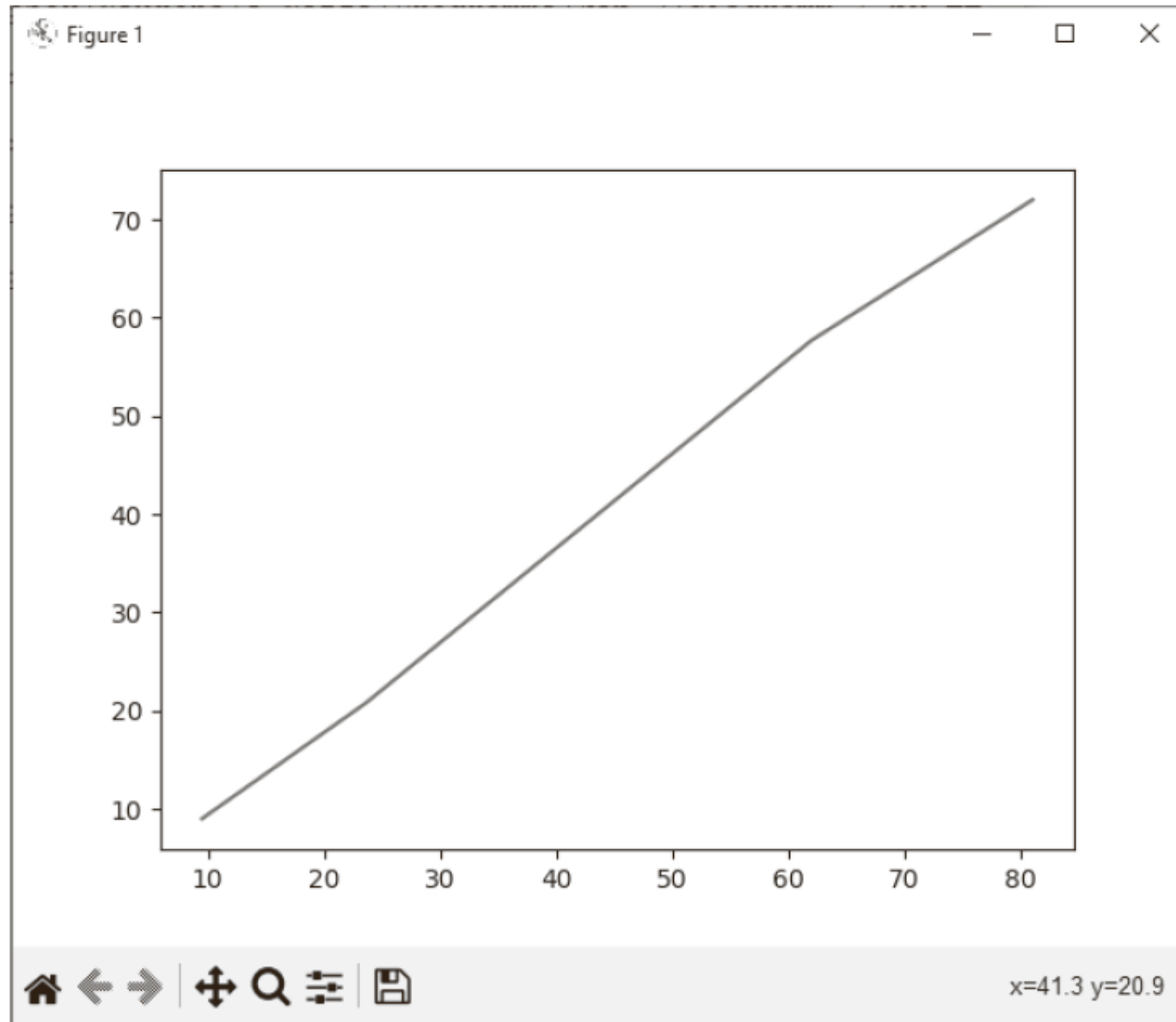
5.1 Trainingscamp: Daten visualisieren



Diagramm



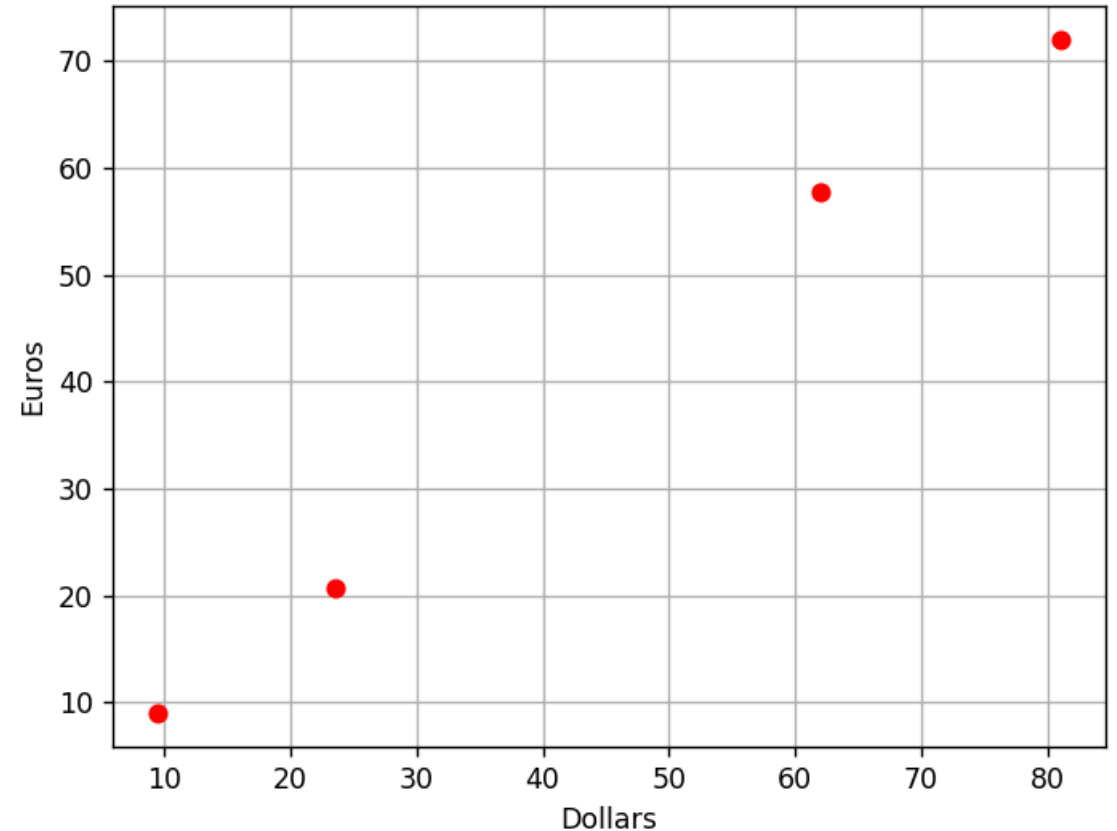
```
# diagramm_1.py
from matplotlib import pyplot
dollarWerte = [9.5, 23.6, 61.99, 81]
euroWerte = [9.0, 20.77, 57.71, 72]
pyplot.plot(dollarWerte, euroWerte)
pyplot.show()
```



Darstellung verfeinern

```
# diagramm_2.py
from matplotlib import pyplot
dollarWerte = [9.5, 23.6, 61.99, 81]
euroWerte = [9.0, 20.77, 57.71, 72]
pyplot.plot(dollarWerte, euroWerte, 'or')
pyplot.xlabel('Dollars')
pyplot.ylabel('Euros')
pyplot.grid()
pyplot.show()
```

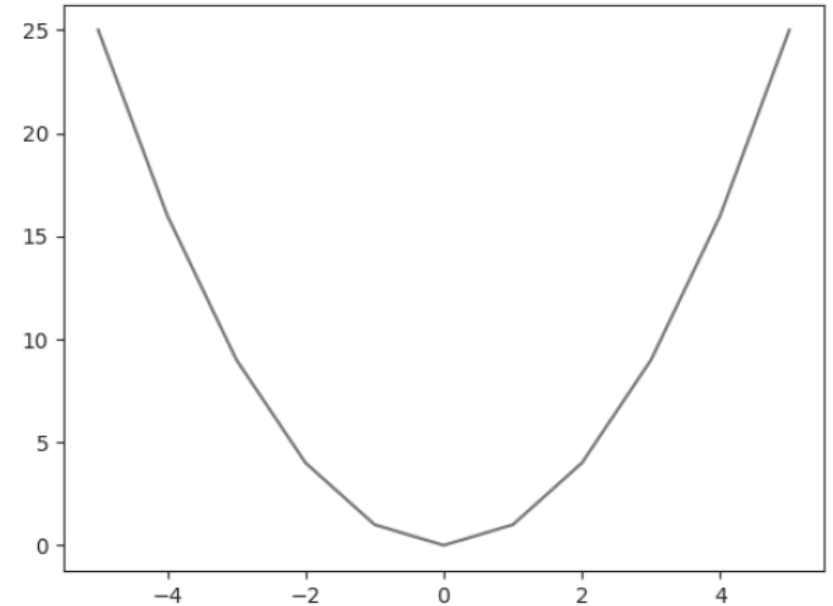
Rote dicke Punkte



Graph einer Funktion

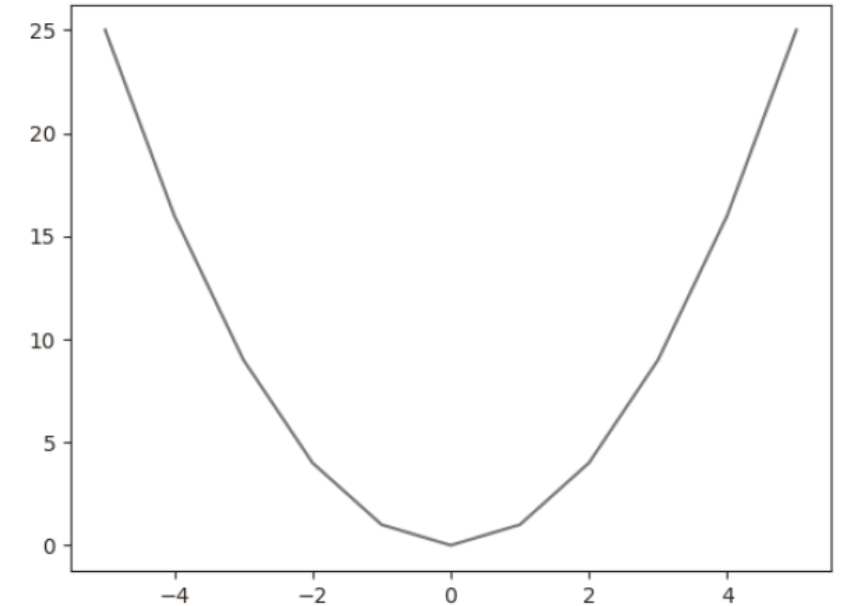
x	-5	-4	-3	-2	-1	0	1	2	3	4	5
y	25	16	9	4	1	0	1	4	9	16	25

```
diagramm_3.py
from matplotlib import pyplot
xWerte = [-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5]
yWerte = [25, 16, 9, 4, 1, 0, 1, 4, 9, 16, 25]
pyplot.plot(xWerte, yWerte)
pyplot.show()
```



Quadratische Funktion

x	-5	-4	-3	-2	-1	0	1	2	3	4	5
y	25	16	9	4	1	0	1	4	9	16	25



```
# diagramm_4.py
from matplotlib import pyplot
xWerte = range(-5, 6)
yWerte = [x**2 for x in xWerte]
pyplot.plot(xWerte, yWerte)
pyplot.show()
```

Zahlen zwischen -5 und 5

Übung 5.1 (5 min)

1) Erzeugen Sie eine Liste mit allen geraden Zahlen von 2 bis 20.

```
gerade = [          for x in range          ]
```

2) Gegeben ist eine Liste:

```
paare = [(1, 3), (12, 7), (4, 5)]
```

Erzeugen Sie eine Liste mit den Summen der Zahlen in den Zahlenpaaren.

```
summen = [      for      in paare]
```

3) Wir haben die Funktion $f(x) = x^3 - 20 \cdot x$. Schreiben Sie ein Programm, das den Graphen der Funktion für ganzzahlige x-Werte von -10 bis +10 ausgibt. Wandeln Sie das Starterprojekt `diagramm 4.py` ab.

```
# diagramm_4.py
from matplotlib import pyplot
xWerte = range(-5, 6)
yWerte = [x**2 for x in xWerte]
pyplot.plot(xWerte, yWerte)
pyplot.show()
```

Lösungen 5.2

1) Erzeugen Sie eine Liste mit allen geraden Zahlen von 2 bis 20.

```
gerade = [2*x for x in range(1, 11)]  
gerade = [2*x + 2 for x in range(10)]
```

2) Gegeben ist eine Liste:

```
paare = [(1, 3), (12, 7), (4, 5)]
```

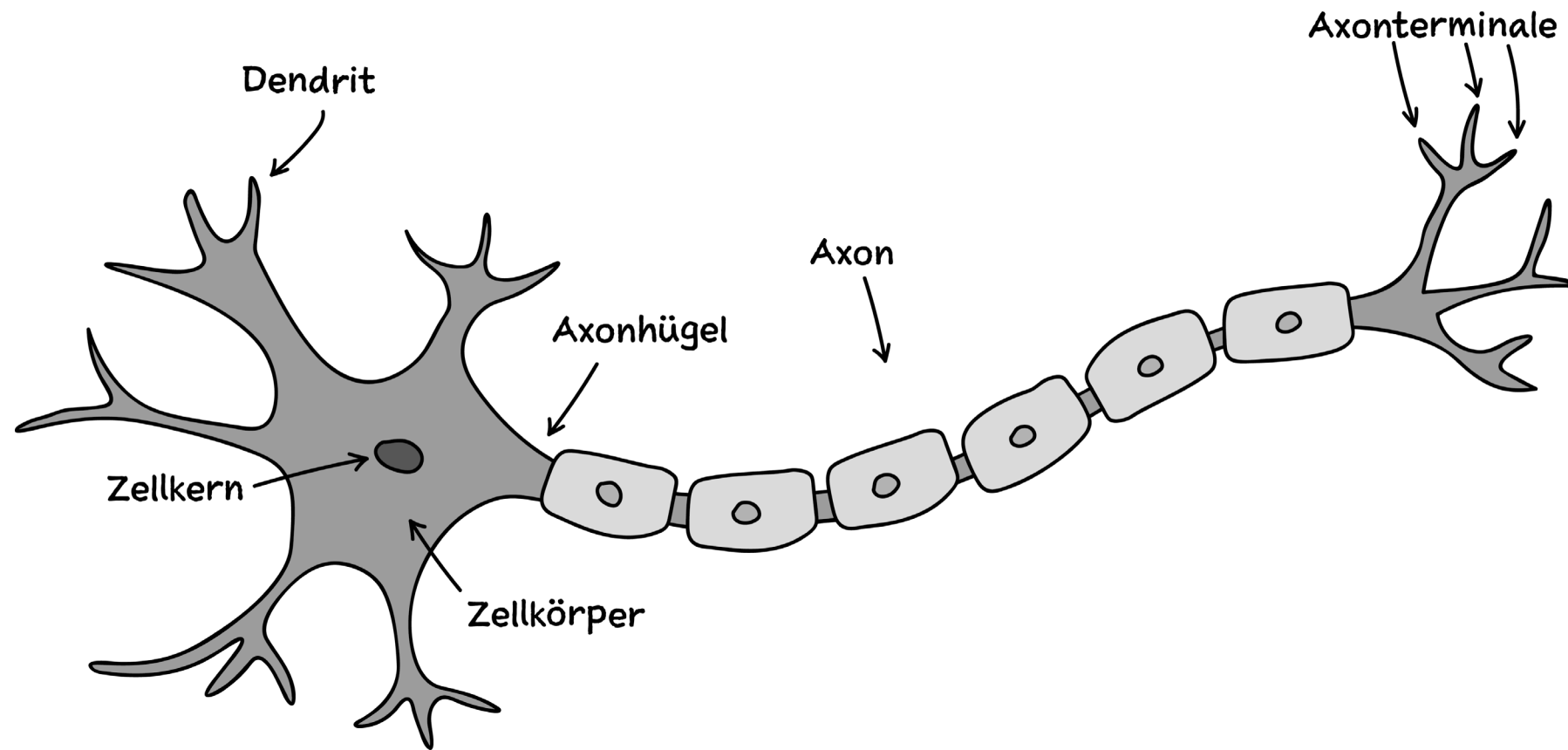
Erzeugen Sie eine Liste mit den Summen der Zahlen in den Zahlenpaaren.

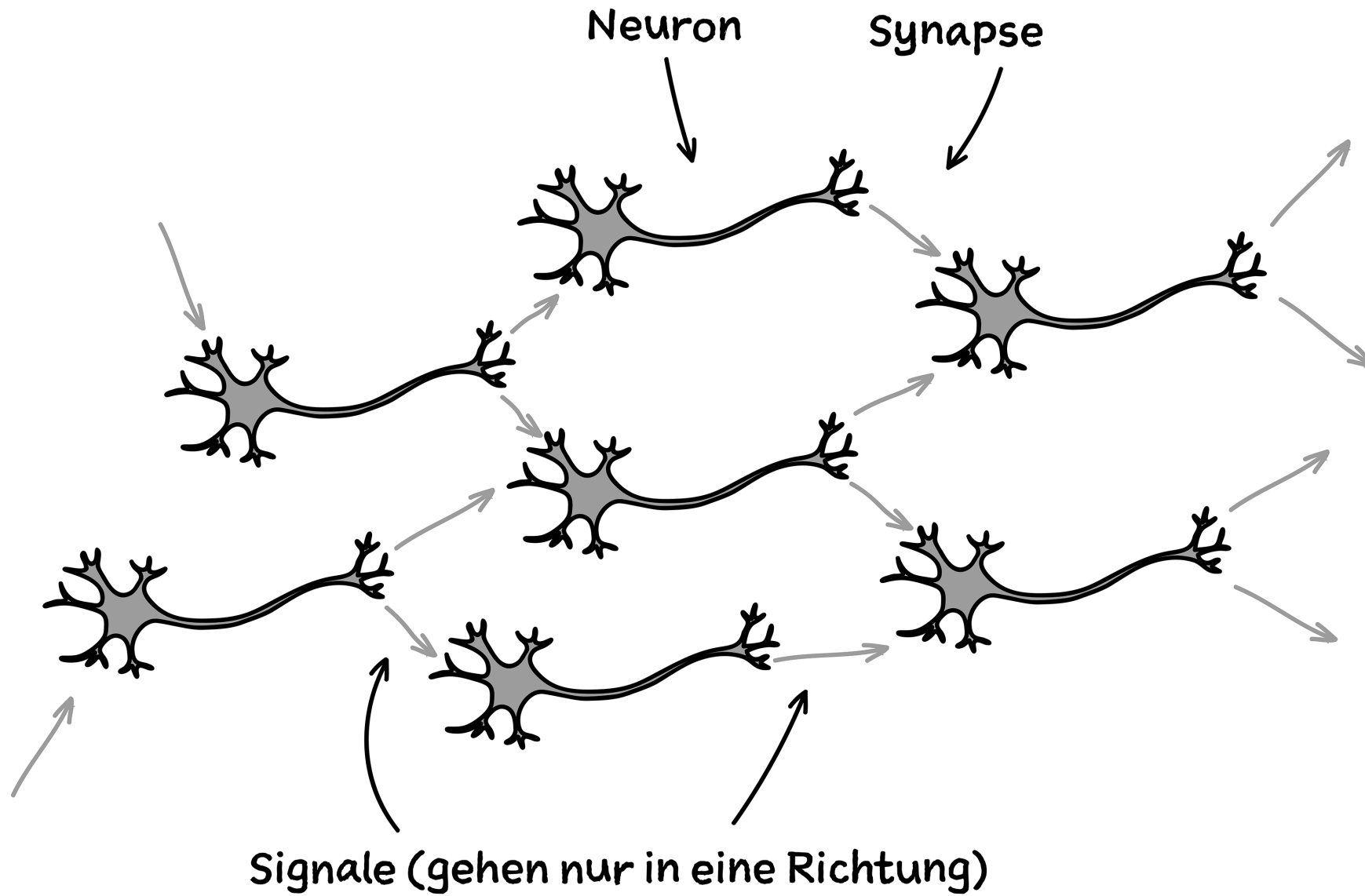
```
summen = [x + y for x, y in paare]  
summen = [sum(t) for t in paare]
```

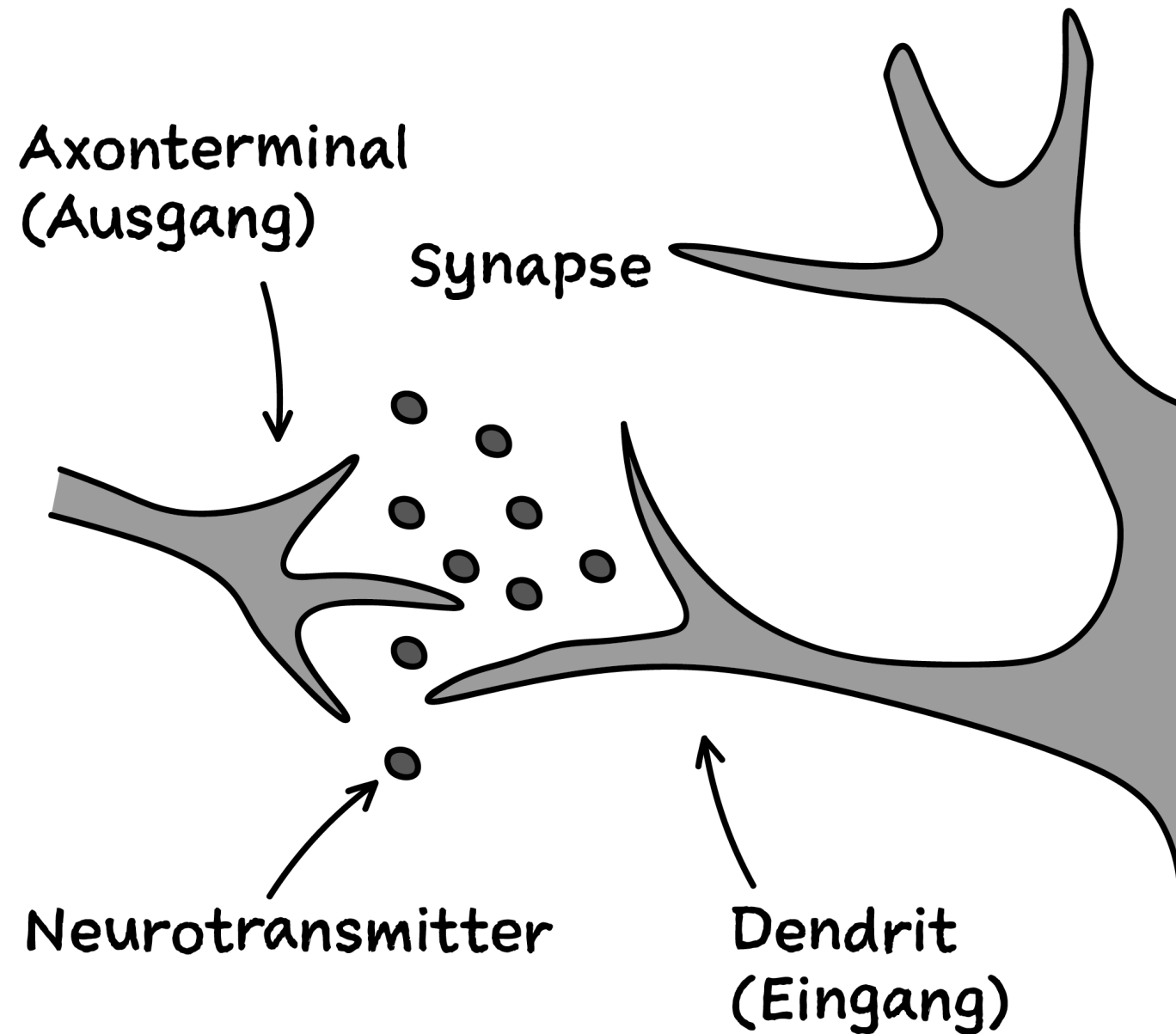
3) Wir haben die Funktion $f(x) = x^3 - 20 \cdot x$. Schreiben Sie ein Programm, das den Graphen der Funktion für ganzzahlige x-Werte von -10 bis +10 ausgibt. Wandeln Sie das Starterprojekt ab.

```
from matplotlib import pyplot  
xWerte = range(-10, 11)  
yWerte = [x**3 - 20*x for x in xWerte]  
pyplot.plot(xWerte, yWerte)  
pyplot.show()
```

Neuronale Netze

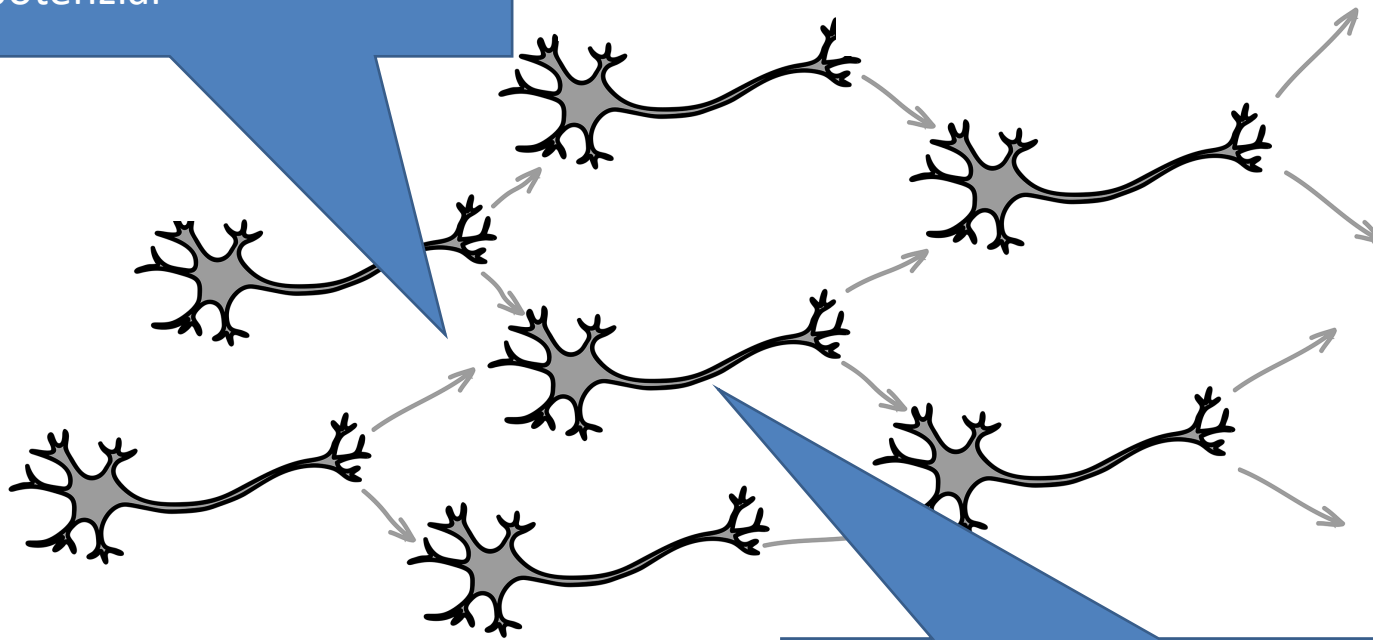






Feuern: Das Alles-oder-nichts-Prinzip

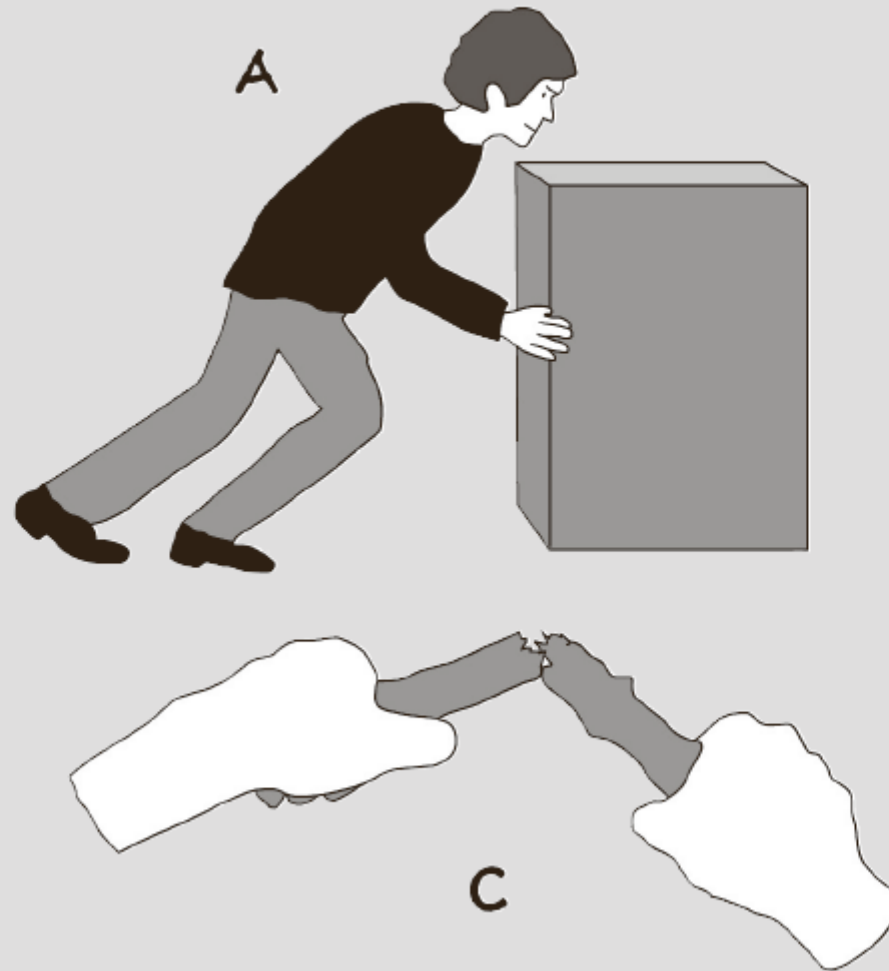
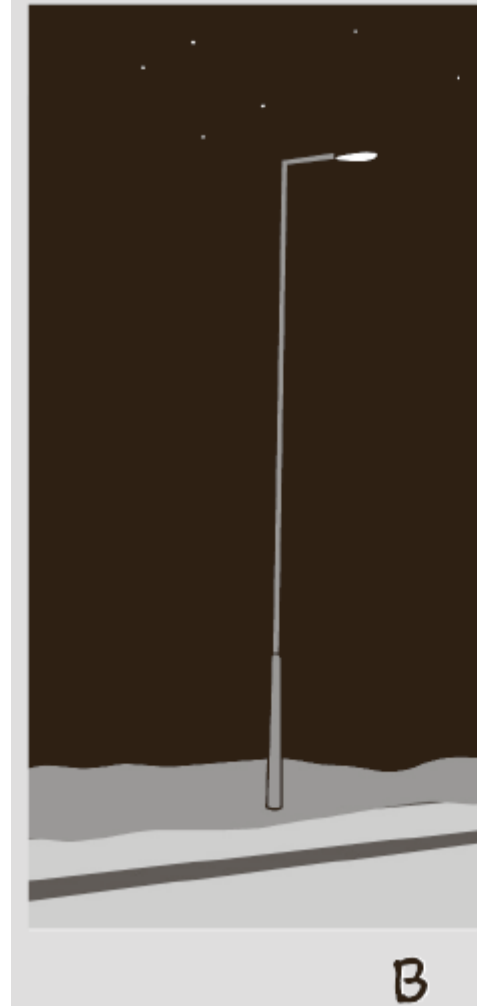
Eingang: Reize addieren sich, erhöhen das Erregungspotenzial



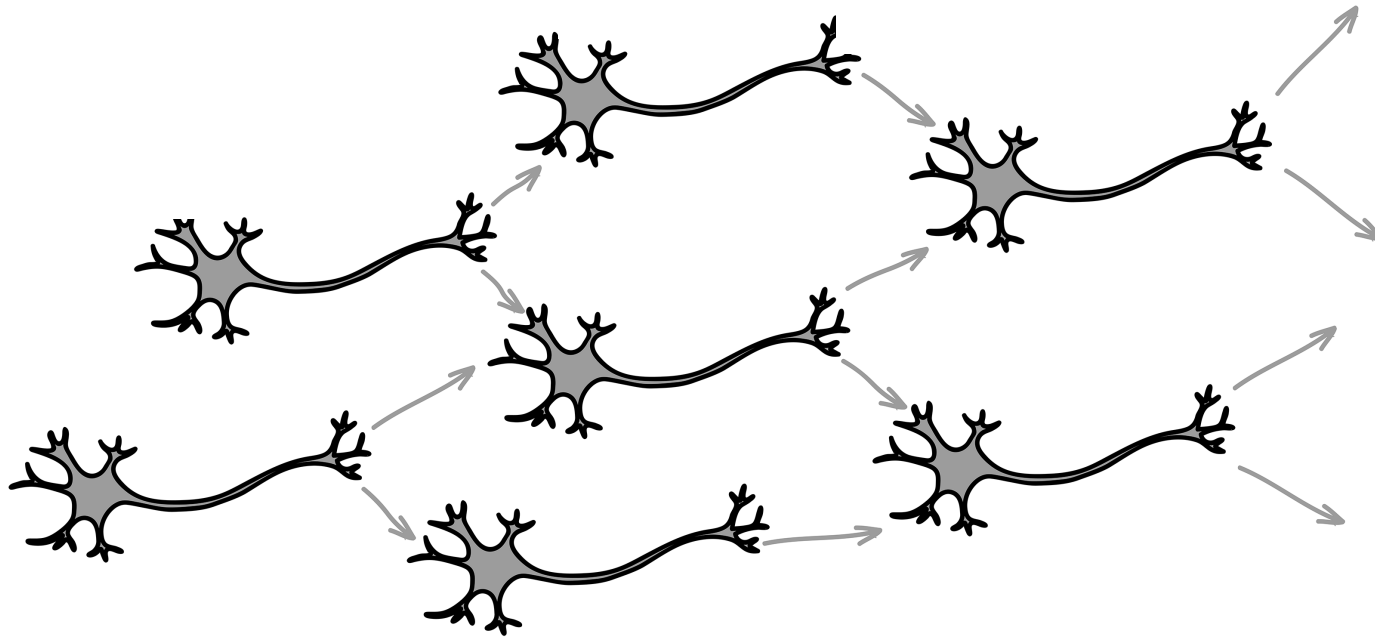
Schaltzeit: 2 ms

Ruhepotenzial: -70mV
Wenn -50 mV erreicht ist, feuert das Neuron

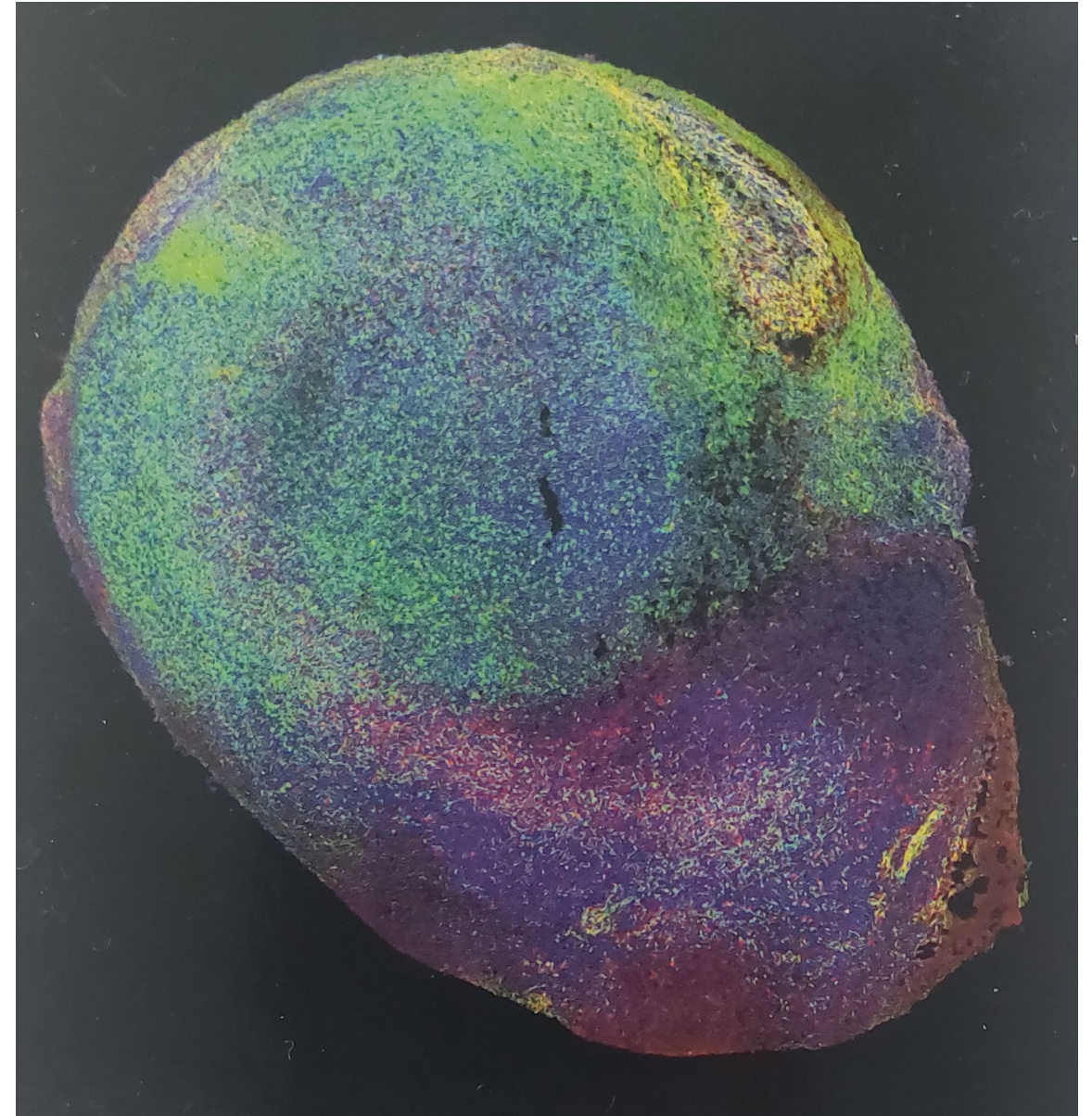
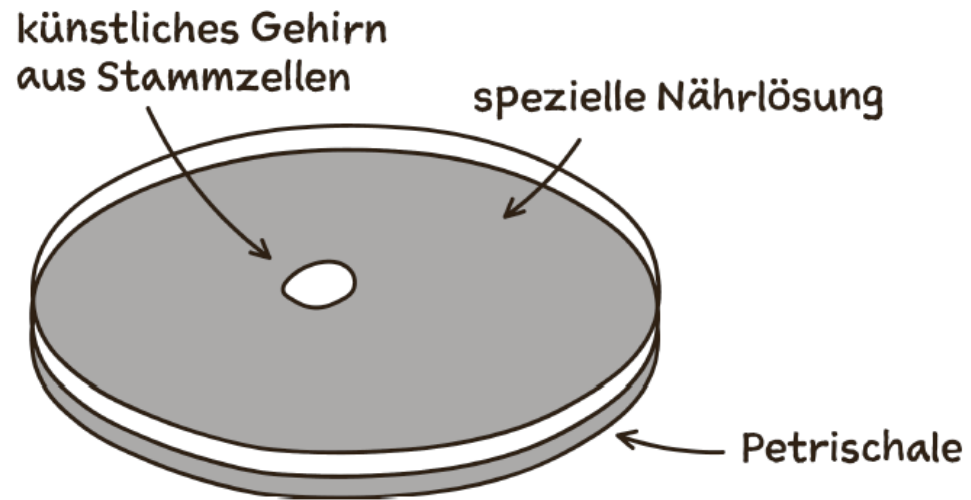
Inwiefern tritt hier das „Alles-oder-nichts-Prinzip“ in Erscheinung?



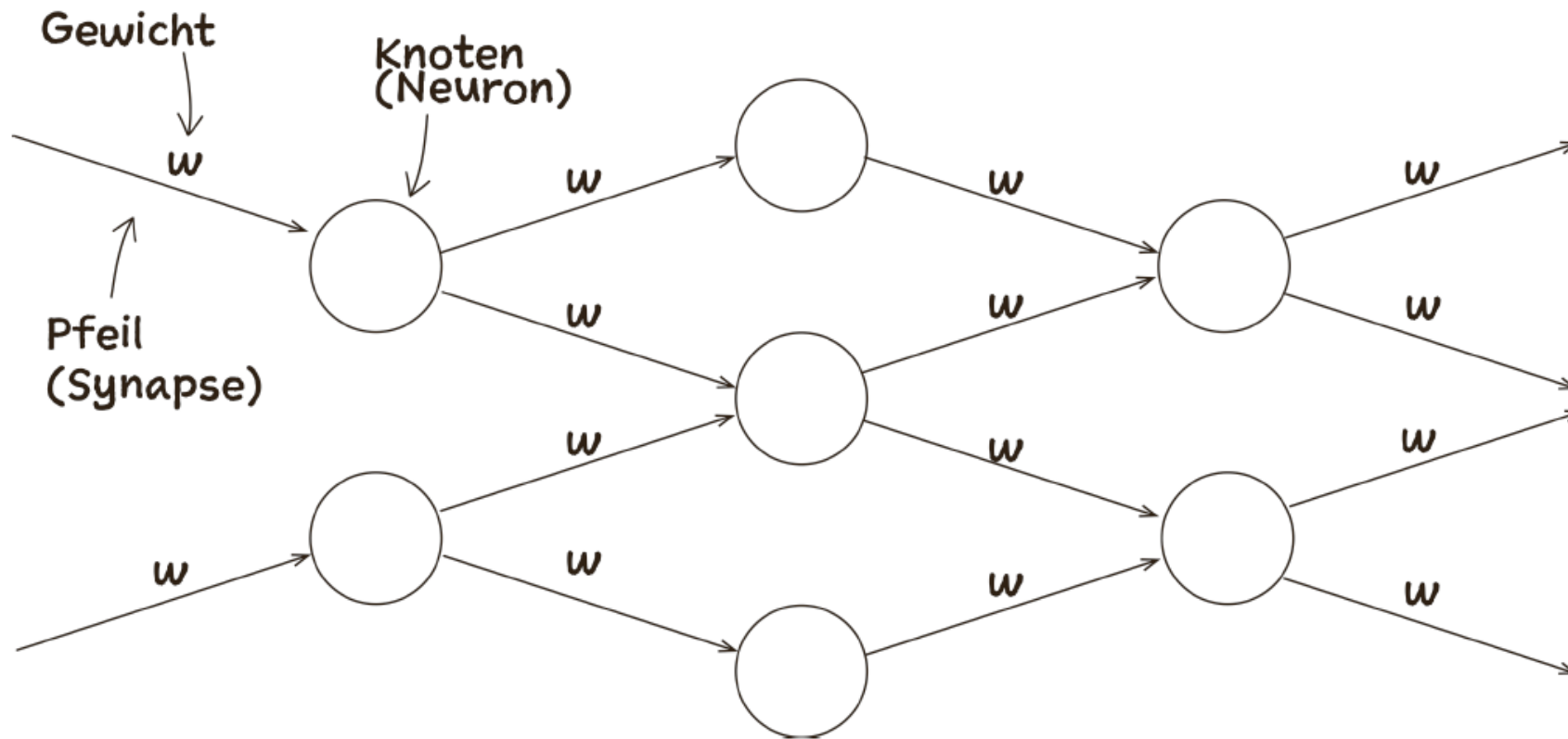
Was passiert beim Lernen?



Künstliches Gehirn



Künstliches neuronales Netz (KNN) = Computerprogramm

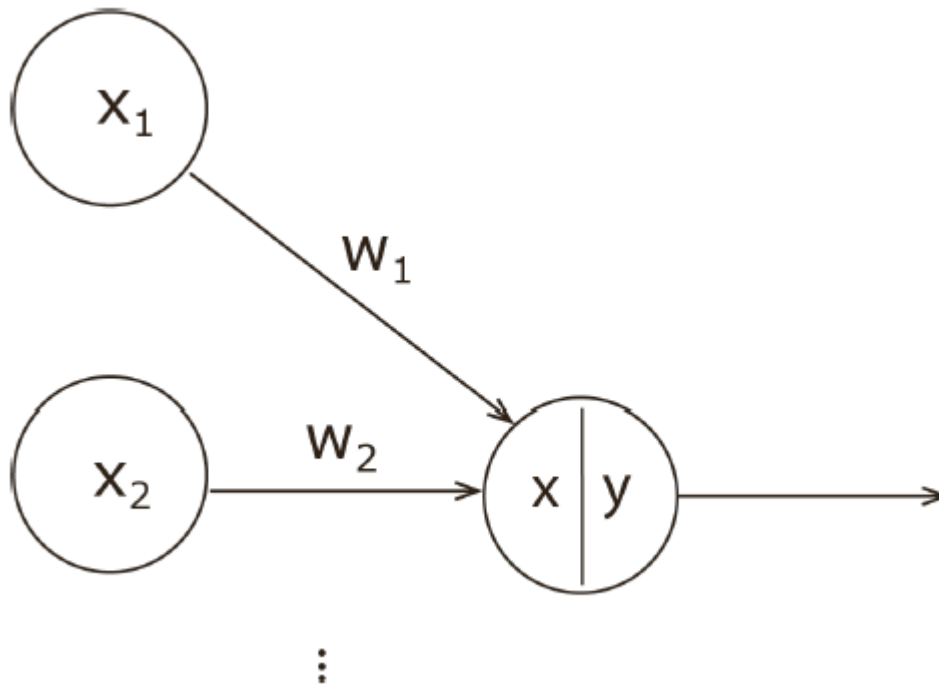


Gerichteter
Graph

Modell eines Neurons

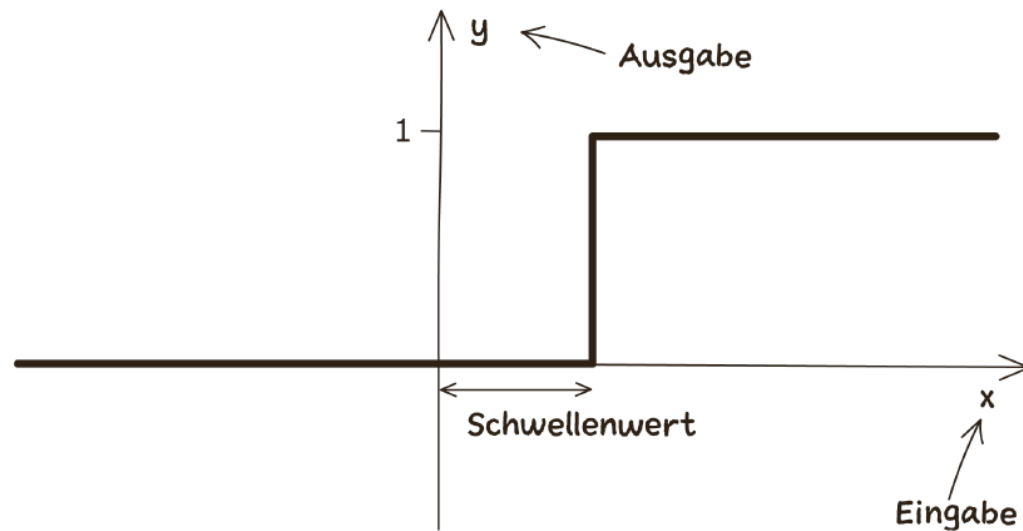
Eingabe:

$$x = x_1 \cdot w_1 + x_2 \cdot w_2 + \dots$$



Aktivierungsfunktion

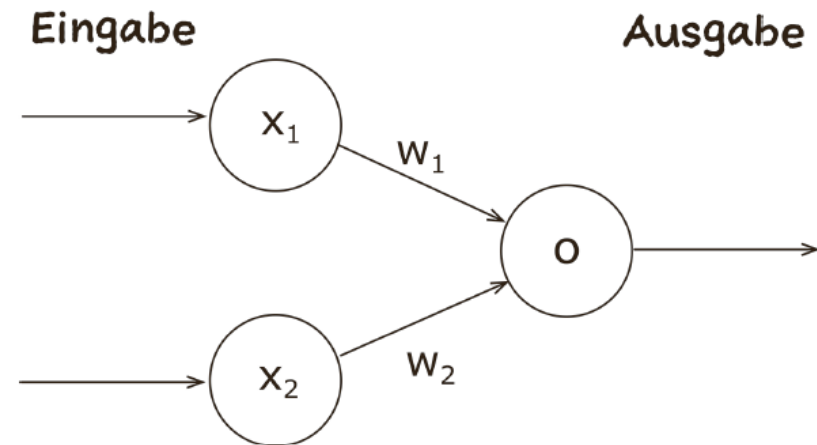
$$y = \begin{cases} 1, & \text{falls } x > b \\ 0, & \text{sonst} \end{cases}$$



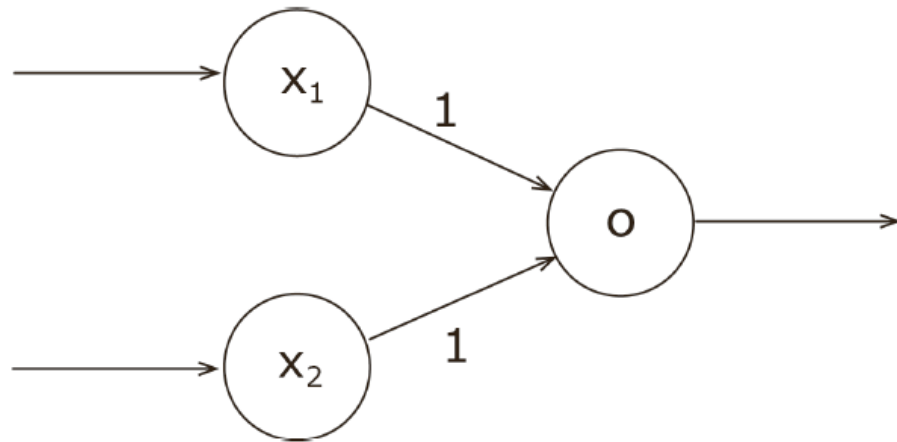
Das Rosenblatt-Perzeptron



Frank Rosenblatt, 1957



Ein Perzeptron, das ODER erkennt



$$A = \pi r^2$$

a	b	a ODER b
0	0	0
0	1	1
1	0	1
1	1	1

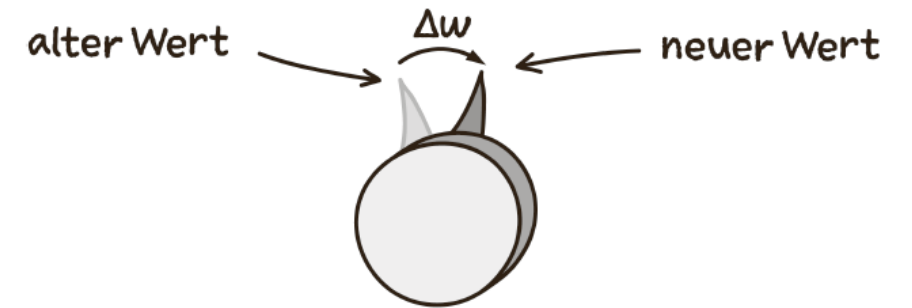
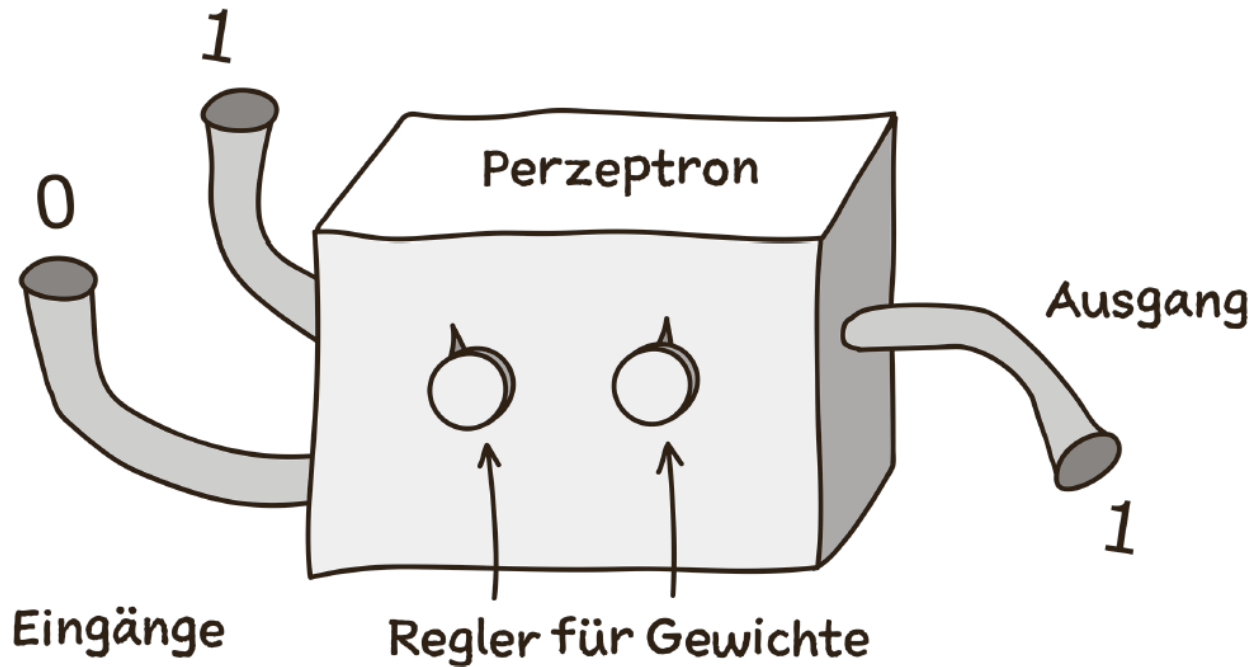
Aktivierungsfunktion:

$$o = \begin{cases} 1, & \text{falls } 1 \cdot x_1 + 1 \cdot x_2 > 0,5 \\ 0, & \text{sonst} \end{cases}$$

Vorgegebene Gewichte

Lernfähiges Perzeptron

Training = Ändern der Gewichte



Lernregel

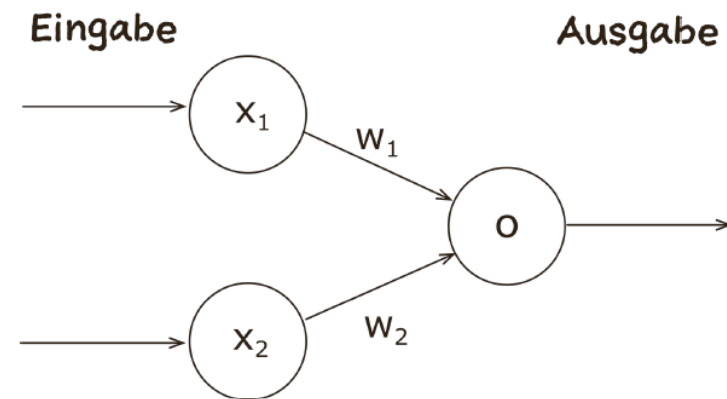
- Wenn die Ausgabe o der erwarteten Ausgabe t entspricht, werden die Gewichte nicht geändert. $\Delta w = 0$
- Wenn die Ausgabe den Wert 0 hat, aber der Wert 1 erwartet wurde, werden die Gewichte *vergrößert*. $\Delta w > 0$
- Wenn die Ausgabe den Wert 1 hat, aber der Wert 0 erwartet wurde, werden die Gewichte *verkleinert*. $\Delta w < 0$

Lernrate

$$\Delta w_1 = \alpha \cdot (t - o) \cdot x_1$$

$$\Delta w_2 = \alpha \cdot (t - o) \cdot x_2$$

Nur wenn der Eingangsknoten den Wert 1 hat, wird das Gewicht geändert



Projekt: Ein Rosenblatt-Perzeptron (1)

```
# perzeptron.py
DATEN = [(0, 0, 0), (0, 1, 1), (1, 0, 1), (1, 1, 1)]
LR = 0.1                # Lernrate
SW = 0.5                # Schwellwert
w1 = 0.5
w2 = 0.5
```

```
def vorhersehen(x1, x2):
    x = w1 * x1 + w2 * x2
    if x > SW:
        return 1
    else:
        return 0
```

a	b	a ODER b
0	0	0
0	1	1
1	0	1
1	1	1

Projekt: Ein Rosenblatt-Perzeptron (2)

```
def trainieren(x1, x2, t):  
    global w1, w2  
    o = vorhersehen(x1, x2)  
    w1 += LR * (t - o) * x1  
    w2 += LR * (t - o) * x2  
  
for epoche in range(10):  
    for x1, x2, t in DATEN:  
        trainieren(x1, x2, t)  
  
for x1, x2, t in DATEN:  
    o = vorhersehen(x1, x2)  
    print('Eingaben:', x1, x2,  
          'Ausgabe:', o, 'Erwartet:', t)
```

$$\Delta w_1 = \alpha \cdot (t - o) \cdot x_1$$

$$\Delta w_2 = \alpha \cdot (t - o) \cdot x_2$$

```
Eingaben: 0 0 Ausgabe: 0 Erwartet: 0  
Eingaben: 0 1 Ausgabe: 1 Erwartet: 1  
Eingaben: 1 0 Ausgabe: 1 Erwartet: 1  
Eingaben: 1 1 Ausgabe: 1 Erwartet: 1
```

Übung 5.2 Perzeptron

Aufgabe 1

Testen Sie das Starterprojekt `perzeptron.py`.

Aufgabe 2

Erweitern Sie das Programm: Sorgen Sie dafür, dass am Ende auch die Werte der Gewichte ausgegeben werden

Aufgabe 3

Experimentieren Sie mit dem Perzeptron.

- Setzen Sie den Schwellenwert SW auf einen kleineren Wert (z.B. 0,3) und prüfen Sie, ob das Perzeptron noch richtig funktioniert. Welche Auswirkung hat diese Änderung auf die Gewichte nach dem Training?
- Setzen Sie die Anfangswerte der Gewichte auf 0,1 und prüfen Sie ob das Perzeptron noch funktioniert.

Aufgabe 4

Ändern Sie die Trainingsdaten des Perzeptrons und sorgen Sie dafür, dass es nun die UND-Verknüpfung der Eingabedaten erkennt.

a	b	a UND b
0	0	0
0	1	0
1	0	0
1	1	1

Übung 5.2

Aufgabe 5

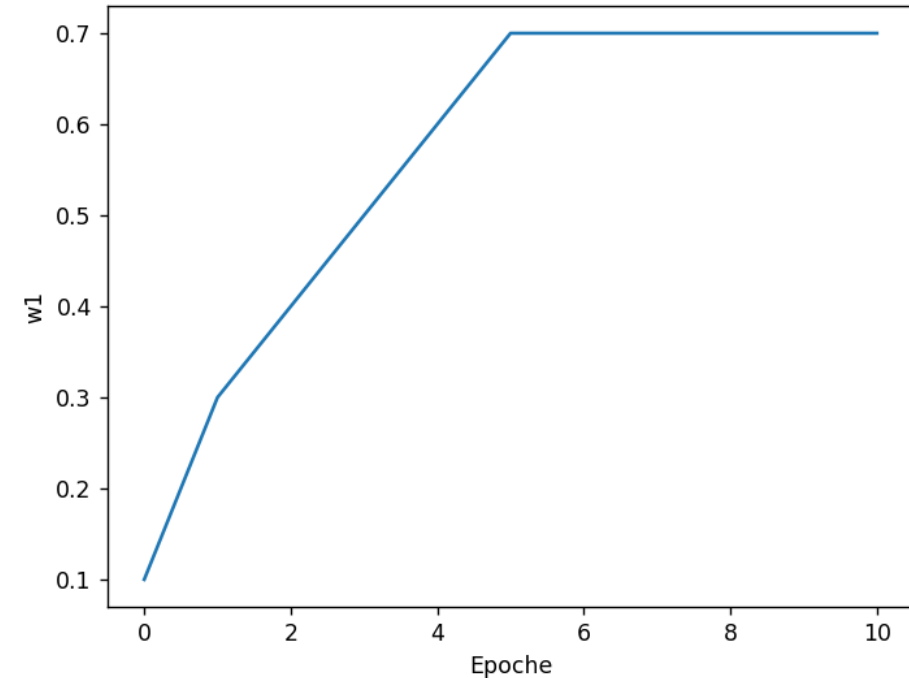
Sorgen Sie dafür, dass das Programm den Lernprozess visualisiert, z.B. indem es den Wert des Gewichts w_1 nach jeder Epoche in einer Liste speichert und am Ende ein Diagramm ausgibt.

Tipp: Mit der Listenmethode `append()` können Sie an eine Liste einen neuen Wert anhängen. Beispiel:

```
s = [1, 5]
s.append(8)
print(s)
```

Ausgabe:

```
[1, 5, 8]
```



Mehr Ideen? https://docs.google.com/document/d/140INsIEWA_AwUwtpVMCZqtH7_eOkU8rmvfrgWqE5M3E/edit?usp=sharing

Lösungen 5.2

Aufgabe 2

Erweitern Sie das Programm: Sorgen Sie dafür, dass am Ende auch die Werte der Gewichte ausgegeben werden

```
print('w1:', w1, 'w2:', w2)
```

Aufgabe 3

Experimentieren Sie mit dem Perzeptron.

- Setzen Sie den Schwellenwert SW auf einen kleineren Wert (z.B. 0,3) und prüfen Sie, ob das Perzeptron noch richtig funktioniert. Welche Auswirkung hat diese Änderung auf die Gewichte nach dem Training?
- Setzen Sie die Anfangswerte der Gewichte auf 0,1 und prüfen Sie ob das Perzeptron noch funktioniert.

Aufgabe 4

Ändern Sie die Trainingsdaten des Perzeptrons und sorgen Sie dafür, dass es nun die UND-Verknüpfung der Eingabedaten erkennt.

```
DATEN = [ (0, 0, 0), (0, 1, 0), (1, 0, 0), (1, 1, 1) ]
```

a	b	a UND b
0	0	0
0	1	0
1	0	0
1	1	1

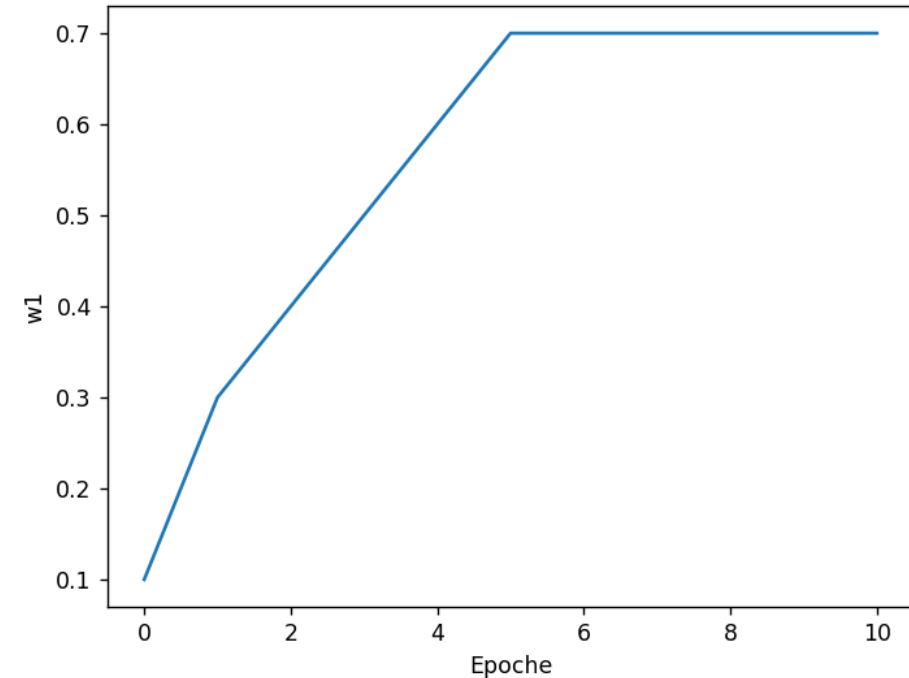
Lösungen 5.2

Aufgabe 5

```
w1Werte =[w1]
for epoche in range(10):
    for x1, x2, t in DATEN:
        trainieren(x1, x2, t)
    w1Werte.append(w1)

for x1, x2, t in DATEN:
    o = vorhersehen(x1, x2)
    print('Eingaben:', x1, x2,
          'Ausgabe:', o, 'Erwartet:', t)

pyplot.plot(range(11), w1Werte)
pyplot.xlabel('Epoche')
pyplot.ylabel('w1')
pyplot.show()
```



Rückblick

- Das Python-Modul **matplotlib** unterstützt die Visualisierung von Daten.
- Mit der Funktion **pyplot.plot()** erzeugt man Diagramme (Plots). Die Funktion benötigt zwei Zahlenfolgen gleicher Länge als Argumente.
- Mit einer List Comprehension kann man leicht eine Liste mit Funktionswerten erzeugen.
- Ein Gehirn besteht aus einem Netzwerk von Neuronen.
- Ein Neuron »feuert«, wenn es durch die empfangenen Signale ein genügend hohes Erregungspotenzial erreicht („Alles-oder-nichts-Prinzip“).
- Ein **Perzeptron** ist eine Struktur mit Eingabe- und Ausgabeknoten, die ähnlich wie Neuronen arbeiten (Rosenblatt 1957). Die Verbindungen zwischen Eingabeknoten und Ausgabeknoten sind mit Gewichten belegt.
- Die Aktivierungsfunktion berechnet aus den gewichteten Eingabewerten einen Ausgabewert nach dem Alles-oder-nichts-Prinzip.
- Ein Perzeptron kann durch Training lernen, UND- und ODER- verknüpfte Eingaben erkennen.
- Beim Training werden die Gewichte geändert.