Arbeitsblatt: Perzeptron

Einführung in neuronale Netze mit Step-Funktion

Informatik

Einstieg: Was ist ein Perzeptron?

Das Perzeptron ist die einfachste Form eines künstlichen Neurons: Es berechnet eine gewichtete Summe der Eingaben, addiert einen Bias (Schwelle) und gibt per Step-Funktion 0 oder 1 aus. Damit kann es Klassen trennen, sofern eine **gerade Linie** genügt (linear separabel). Alles Komplexere (mehr Neuronen, mehr Schichten) baut auf genau diesem Grundbaustein auf.

1. Lernziel

Ich kann

- den Aufbau eines Perzeptrons erklären (Inputs, Gewichte, Bias, Aktivierung),
- einen Vorwärtsdurchlauf mit der Step-Funktion berechnen,
- die Perzeptron-Lernregel auf ein Trainingsbeispiel anwenden.

2. Überblick: Aufbau

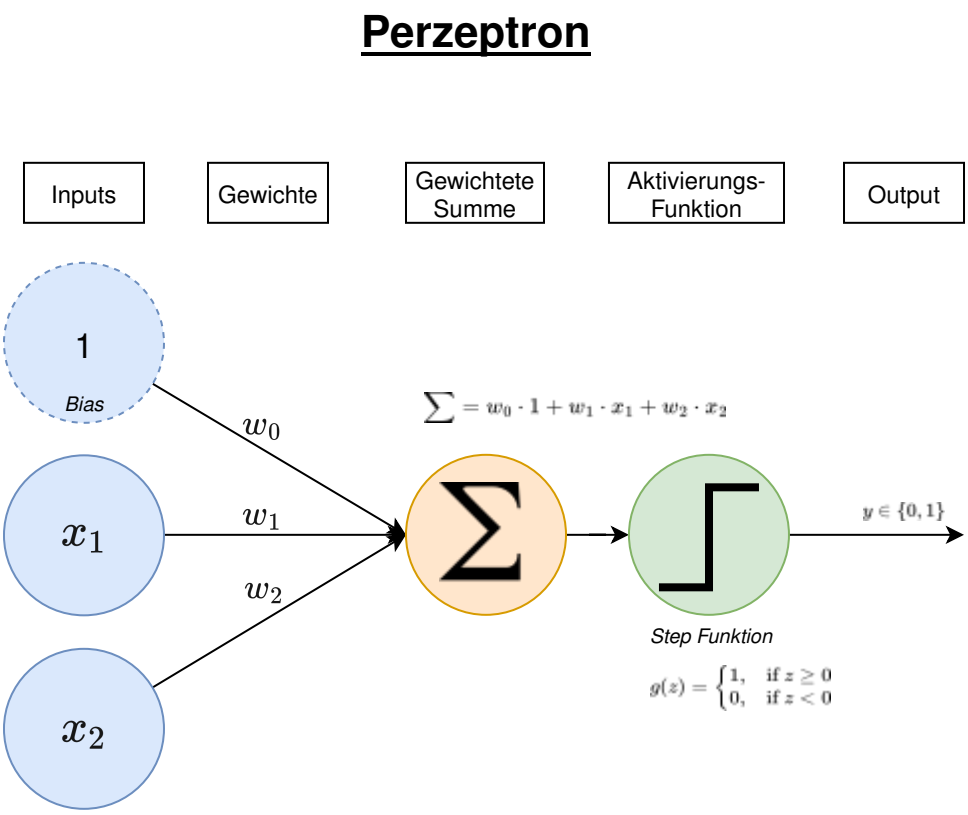

Abbildung 1: Perzeptron Übersicht

- Bias-Eingang: $x_0 = 1$ ist immer aktiv; w_0 wirkt damit als konstanter Offset.
- Eingänge: $x_1, x_2, \ldots$ mit Gewichten $w_1, w_2, \ldots$
- Gewichtete Summe in Sigma-Notation:

$$z = \sum_{i=0}^{n} w_i x_i = w_0 \cdot 1 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$$

- Step-Aktivierung und Ausgabe $\hat{y}$:

$$\hat{y} = g(z) = \begin{cases} 1, & z \geq 0 \\ 0, & z < 0 \end{cases}$$

2.1 Bias-Effekt

Der Bias-Input $x_0 = 1$ mit Gewicht w_0 ermöglicht es, einen Schwellenwert zu setzen, der die Trennlinie im Eingaberaum verschiebt. Dadurch kann das Perzeptron flexibler auf verschiedene Datenverteilungen reagieren. Dies funktioniert analog zur Linearen Gleichung einer Geraden im zweidimensionalen Raum: in der Gleichung $y = mx + b$ entspricht der Bias b dem y-Achsenabschnitt, der die Gerade nach oben oder unten verschiebt, ohne ihre Steigung m zu verändern. Da das Perzeptron eine gewichtete Summe berechnet, wirkt der Bias w_0 ähnlich wie der y-Achsenabschnitt in der Geradengleichung. Das Bias-Gewicht w_0 beeinflusst, wo die Trennlinie die Achsen schneidet, und somit, wie die Klassen im Eingaberaum getrennt werden können. Ein negativer Bias verschiebt die Trennlinie nach oben, während ein positiver Bias sie nach unten verschiebt. Im unteren Beispiel sieht man, wie sich die Trennlinie durch unterschiedliche Bias-Gewichte verändert. Im ersten Fall ($w_0 = -0.5$) schneidet die Linie die y-Achse höher als im zweiten Fall ($w_0 = 0.2$).

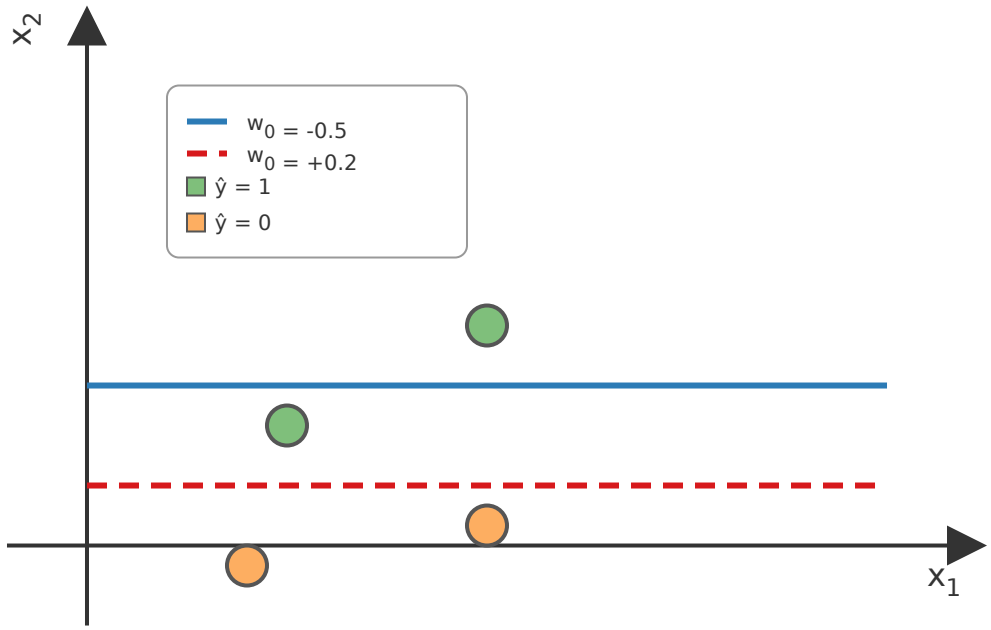

Abbildung 2: Bias verschiebt die Trennlinie

Beispiel 1: AND-Gatter nachbilden

Ein AND-Gatter gibt nur dann 1 aus, wenn **beide** Eingänge 1 sind - sonst ist die Ausgabe 0. Das Perzeptron soll genau dieses Verhalten nachbilden. Ein solches Gatter wird in der Digitaltechnik verwendet, um logische Operationen durchzuführen. Beispielsweise bei einem Garagentor, der nur dann öffnet, wenn sowohl der "Tür offen"-Sensor als auch der "Schlüssel im Schloss"-Sensor aktiviert sind (beide Eingänge sind 1). In diesem Fall sorgt das AND-Gatter dafür, dass die Garage nur dann geöffnet wird, wenn beide Bedingungen erfüllt sind.

Gegebenes Perzeptron (Step-Funktion) ist folgend konfiguriert:

- Bias-Gewicht $w_0 = -0.9$ (Bias-Eingang bleibt $x_0 = 1$)
- $w_1 = 0.6$, $w_2 = 0.6$

AND-Wahrheitstafel (Soll-Ausgabe y):

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

Berechne nun z $\hat{y}$ und prüfe, ob das Perzeptron die Soll-Ausgabe trifft:

x_1	x_2	z	$\hat{y}$	$y = \hat{y}$? (Ja/Nein)
0	0			
0	1			
1	0			
1	1			

Visualisierung von AND-Gatter

1) Färbe die Punkte nach der AND-Wahrheitstafel: $y = 1$ grün, $y = 0$ rot.

2) Zeichne eine Trennlinie, die alle grünen von allen roten Punkten separiert.

3) Prüfe, ob deine Linie zum berechneten $\hat{y}$ passt.

Abbildung 3: AND-Gatter Visualisierung

Beispiel 2: OR-Gatter nachbilden

Ein OR-Gatter gibt 1 aus, wenn **mindestens einer** der beiden Eingänge 1 ist; nur bei $x_1 = 0$ **und** $x_2 = 0$ ist die Ausgabe 0. Das Perzeptron soll dieses Verhalten nachbilden.

Gegebenes Perzeptron (Step-Funktion), passend für OR:

- Bias-Gewicht $w_0 = -0.2$ (Bias-Eingang bleibt $x_0 = 1$)
- $w_1 = 0.6$, $w_2 = 0.6$

OR-Wahrheitstafel (Soll-Ausgabe y):

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

Berechne z, $\hat{y}$ und prüfe, ob das Perzeptron die Soll-Ausgabe trifft:

x_1	x_2	z	$\hat{y}$	$y = \hat{y}$? (Ja/Nein)
0	0			
0	1			
1	0			
1	1			

Visualisierung von OR-Gatter

1) Färbe die Punkte nach der OR-Wahrheitstafel: $y = 1$ grün, $y = 0$ rot.

2) Zeichne eine Trennlinie, die alle grünen von allen roten Punkten separiert.

3) Prüfe, ob deine Linie zum berechneten $\hat{y}$ passt.

Abbildung 4: OR-Gatter Visualisierung

Beispiel 3: Perzeptron trainieren (NAND)

In Beispiel 1 und 2 waren die Gewichte bereits passend gewählt. Jetzt soll das Perzeptron **selbst lernen**: Wir trainieren auf ein NAND-Gatter (Ausgabe 0 nur bei $x_1 = 1$ und $x_2 = 1$, sonst 1).

Wahrheitstabelle / Trainingsdaten der NAND-Logik:

x_1	x_2	y
0	0	1
0	1	1
1	0	1
1	1	0

Verwende folgende Startwerte für das Perzeptron:

- $w_0 = 0.1$, $w_1 = 0.1$, $w_2 = 0.1$
- Lernrate $\eta = 0.2$

Perzeptron-Lernregel: Die Lernrate gibt an, wie stark die Gewichte bei einem Fehler angepasst werden. In diesem Fall bedeutet $\eta = 0.2$, dass die Gewichte um 20 % des Fehlers korrigiert werden. Korrigieren bedeutet hier, dass die Gewichte in Richtung des Fehlers angepasst werden, um die Vorhersage des Perzeptrons zu verbessern. Eine zu hohe Lernrate kann jedoch dazu führen, dass das Modell über das Optimum hinausschiesst und nicht konvergiert, während eine zu niedrige Lernrate den Lernprozess verlangsamen kann. Konvergieren bedeutet in diesem Kontext, dass das Perzeptron lernt, die Trainingsdaten korrekt zu klassifizieren, sodass keine weiteren Gewichtsanpassungen mehr notwendig sind.

Nutze das Flussdiagramm als Leitfaden, um zu verstehen, wie die Gewichte im Perzeptron angepasst werden. Nachfolgend ist die Perzeptron-Lernregel als vereinfachte Schritt-für-Schritt-Anleitung:

1. Führe einen Vorwärtsdurchlauf durch: Berechne z und $\hat{y}$.
2. Vergleiche die Vorhersage $\hat{y}$ mit der Soll-Ausgabe
3. Wenn $\hat{y} \neq y$, passe die Gewichte an nach der Regel:

$$w_i = w_i + \eta\,(y - \hat{y})\,x_i$$

4. Wiederhole für alle Trainingsbeispiele bis keine Fehler mehr auftreten.

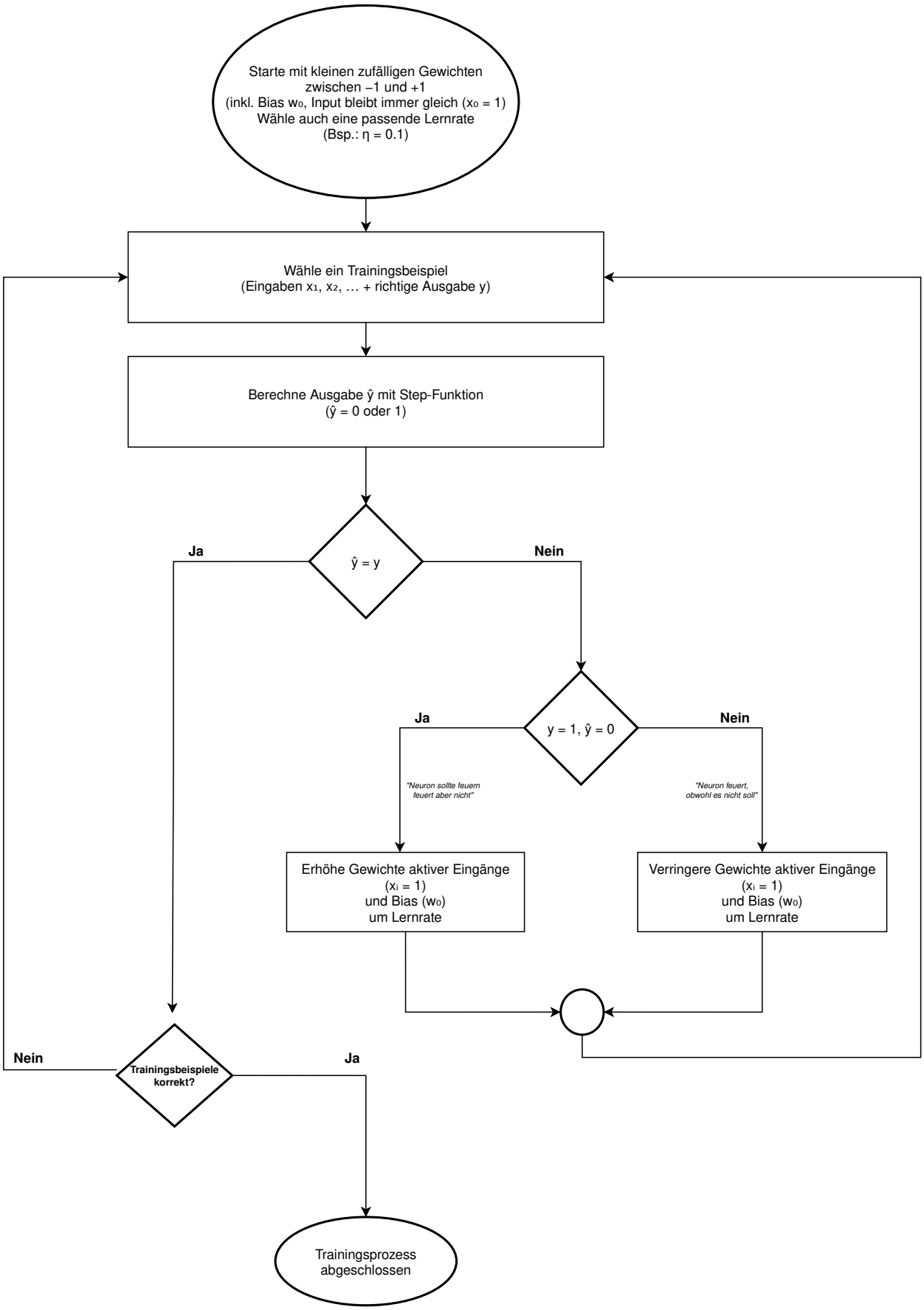

Abbildung 5: Perzeptron-Lernregel

Aufgabe 3a: Vorhersage und Fehler

Führe einen Durchlauf über die vier Beispiele durch.

x_1	x_2	z	$y = \hat{y}$? (Ja/Nein)
0	0		
0	1		
1	0		
1	1		

Aufgabe 3b: Gewichte aktualisieren

- Führe die Lernschritte nacheinander aus bis keine Fehler mehr auftreten.
  1. Nimm das **erste** falsch klassifizierte Beispiel aus 3a und aktualisiere alle aktiven Gewichte ($x_i = 1$) sowie den Bias w_0.
  2. Überprüfe die verbliebenen Beispiele mit den **neuen** Gewichten und wähle erneut das erste falsch klassifizierte Beispiel für einen **weiteren** Update-Schritt.
- In beiden Schritten gilt die Lernregel:

$$w_i = w_i + \eta\,(y - \hat{y})\,x_i$$

Trage nach jedem Schritt die Werte ein (Schritt 2 startet mit den neuen Gewichten aus Schritt 1):

Update 1

Gewicht	Alter Wert	$y - \hat{y}$	Neues w
w_0			
w_1			
w_2			

Überprüfe erneut alle Beispiele mit den neuen Gewichten und finde das erste falsche Beispiel:

x_1	x_2	z	$y = \hat{y}$? (Ja/Nein)
0	0		
0	1		
1	0		
1	1		

Update 2

Gewicht	Alter Wert	$y - \hat{y}$	Neues w
w_0			
w_1			
w_2			

Überprüfe erneut alle Beispiele mit den neuen Gewichten und finde das erste falsche Beispiel:

x_1	x_2	z	$y = \hat{y}$? (Ja/Nein)
0	0		
0	1		
1	0		
1	1		

Aufgabe 3c: Visualisierung

1) Färbe die Punkte nach der NAND-Wahrheitstafel: $y = 1$ grün, $y = 0$ rot.

2) Zeichne die Trennlinie nach dem **ersten** Update.

3) Zeichne die Trennlinie nach dem **zweiten** Update.

4) Beschreibe, wie sich die Linie verändert hat.

Abbildung 6: NAND-Gatter Visualisierung

Aufgabe 4: Visualisierung XOR-Gatter

Das XOR-Gatter gibt 1 aus, wenn **genau einer** der beiden Eingänge 1 ist; bei beiden Eingängen 0 oder beiden Eingängen 1 ist die Ausgabe 0. Zeichne die Punkte des XOR-Gatters in das Koordinatensystem ein und versuche, eine Trennlinie zu finden.

Wahrheitstafel des XOR-Gatters:

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

Abbildung 7: XOR-Gatter Visualisierung