Edu4AI

Künstliche Intelligenz und maschinelles Lernen zur Förderung der Fähigkeiten des 21. Jahrhunderts in der Sekundarbildung

Einführung und Überblick über künstliche neuronale Netze

Von Thomas Jörg, thomas@iludis.de

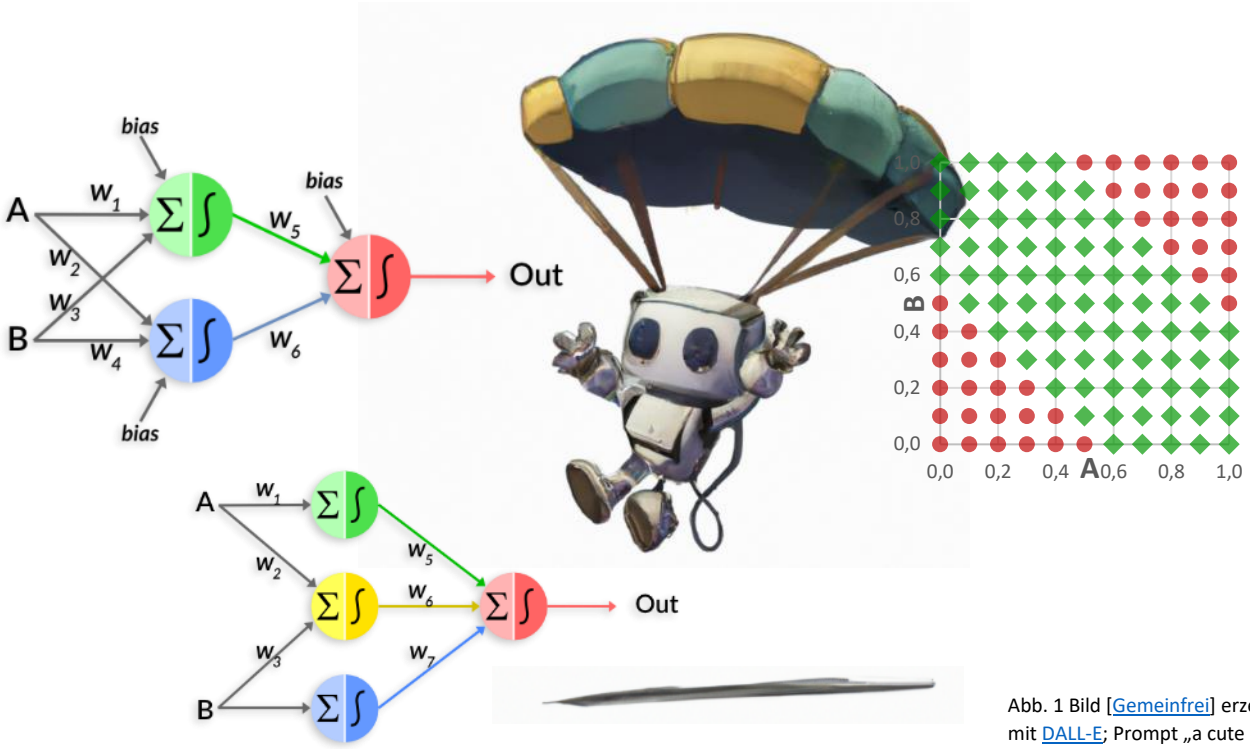


Abb. 1 Bild [Gemeinfrei] erzeugt mit DALL-E; Prompt „a cute little robot during parachute jump with white background, digital art“ von Jörg [CC BY-SA 4.0 International]

In diesem Bild sehen Sie auf der linken Seite zwei mögliche Perceptron-Netze, die das XOR-Problem verarbeiten können. Auf der rechten Seite ist ein Wahrheitsdiagramm dargestellt, das von einem Perceptron-Algorithmus zur Klassifizierung verwendet wird. In der Mitte sehen Sie einen niedlichen kleinen Roboter, der einen Fallschirmsprung macht, wie er von DALL-E 2 interpretiert wird.

Eine Sammlung von Lektionen zur Einführung in die grundlegenden Konzepte der Künstliche Intelligenz / Neuronale Netze im Schulunterricht.

Zielgruppe: Schüler zwischen 14 und 18 Jahren.

Inhaltsübersicht

Abb. 2 Bild [Gemeinfrei] erzeugt mit DALL-E; Prompt „a grinning cat mad scientist doing thumbs up“ von Jörg [CC BY-SA 4.0 International]

Hinweis:
Alle (!) Bilder und Grafiken in diesem Dokument wurden entweder selbst erstellt oder durch die Künstlichen Intelligenzen „DALL-E2“ oder „Stable Diffusion“

Ich danke Ihnen!

Um diese Lektion zu planen und zu gestalten, waren Anregungen von vielen verschiedenen Autoren notwendig. Jeder dieser Autoren hat seine eigenen Ideen entwickelt, um dieses komplexe und faszinierende Gebiet ein wenig verständlicher zu machen. Deshalb soll deren Arbeit hier gewürdigt werden:

Youtube-Kanäle

Autor	Titel / Kanal	Link
Josh Starmer	Statquest	https://www.youtube.com/c/joshstarmer
Luis Serrano	Luis-Serrano-Akademie	https://www.youtube.com/c/LuisSerrano
Brandon Rohrer	Brandon Rohrer	https://www.youtube.com/c/BrandonRohrer
Grant Sanderson	3Blau1braun	https://www.youtube.com/c/3blue1brown
Patrick Loeber	Python-Ingenieur	https://www.youtube.com/c/PythonEngineer

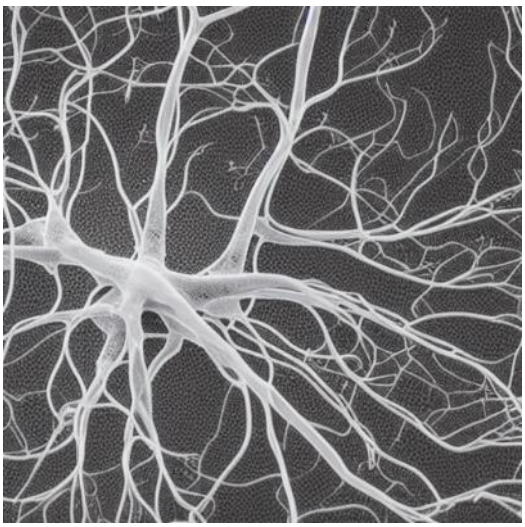
Bücher

Autor	Titel	Herausgeber
Andrew W. Trask (2019)	Deep Learning erforschen	Manning-Veröffentlichungen
Luis Serrano (2021)	Maschinelles Lernen begreifen	Manning-Veröffentlichungen
Michael Taylor (2017)	Die Mathematik der neuronalen Netze	Blaue Windmühle Medien
Michael Taylor (2017)	Deep Learning: Eine visuelle Einführung für Anfänger	Blaue Windmühle Medien

Simulationen

Visualisierungen, Demonstrationen, Simulationen	Thema
https://www.cs.ryerson.ca/~aharley/vis/conv/flat.html	CNN-Demo
https://poloclub.github.io/cnn-explainer/	CNN-Explainer
https://lecture-demo.ira.uka.de/convolution-demo/	Faltungs-Demo
https://playground.tensorflow.org/	FAMOUS NeuralNetwork-Demo
https://lecture-demo.ira.uka.de/neural-network-demo/	Perceptron-Demo
https://www.mladdict.com/neural-network-simulator	NeuralNet-Simulator
https://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html	CNN-Demo
https://anhcoi123.github.io/neural-network-demo/	NeuralNet-Simulator
http://alexlenail.me/NN-SVG/LeNet.html	Zeichnen von NNetzen in verschiedenen Stilen
https://editor.aifiddle.io/	NN interaktiv im Browser aufbauen
https://iludis.de/XOR_Perceptron/index.html	NN-Simulator für XOR
https://iludis.de/XOR_Perceptron2/index.html	NN-Simulator für XOR, Variante 2
https://iludis.de/PerceptronArea/index.html	Einfacher Perceptron-Simulator
https://iludis.de/Perceptron/index.html	Perceptron mit Gradient dezent

Lektion 1: Einführung in neuronale Netze: Neuron und Rosenblatt-Perzeptron



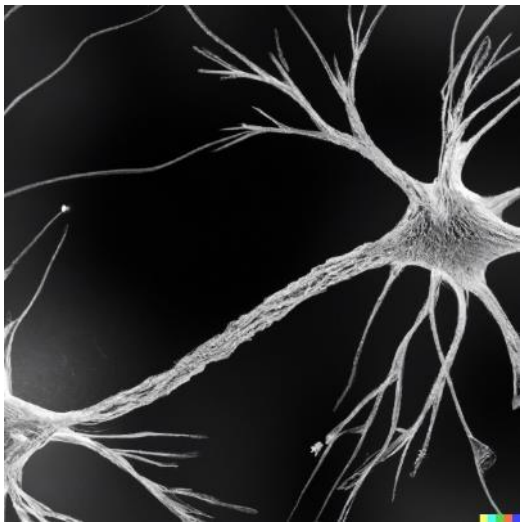


Abb. 3 Bild [Gemeinfrei] erzeugt mit DALL-E; Prompt „neuron with dendrites, electron microscope photo, realictic“ von Jörg [CC BY-SA 4.0 International]

Was Schüler lernen sollten

1958 hatte der Psychologe Frank Rosenblatt, der auch Informatiker war, eine geniale Idee: Könnte man das Prinzip des Lernens aus der Biologie kopieren und maschinell nachbilden? Das Prinzip des künstlichen Neurons war geboren.

Die Schüler lernen den Aufbau natürlicher Neuronen und deren Verbindungen durch einen Einblick in die Biologie. Das Lernen erfolgt durch die Anpassung der Neuron-Neuron-Verbindungen. Durch die Verbindung vieler Neuronen untereinander entsteht emergentes Verhalten: Im Zusammenspiel wird das Ganze mehr als die Summe aller Teile. Künstliche Neuronen werden von ihren natürlichen Vorbildern abgeleitet.

Dabei werden Wahrnehmungen als einfache binäre Klassifikatoren erzeugt. Durch die Verbindung vieler Neuronen (mehrschichtige Perzeptronen) können immer komplexere Muster gelernt und erkannt werden. Das Lernen erfolgt auf der Grundlage von Anpassungen der Gewichtungsfaktoren.

Mögliche Aktivitäten der Schüler

Die Schüler hören das Prinzip des biologischen Neurons in einem kurzen Vortrag des Lehrers. Der Schwerpunkt liegt auf dem Prinzip, wie das Neuron funktioniert - und weniger auf der technischen Terminologie. Dieses Funktionsprinzip wird abstrahiert, um das Perzeptron in einem fragend-entwickelnden Abschnitt der Lektion Lehrer und Schüler vorzustellen:

Die Lehrkraft erarbeitet mit den Schülerinnen und Schülern das Perzeptron-Prinzip als Abstraktion des natürlichen Neurons. Jede Funktionseinheit aus der Biologie spiegelt sich in dem Modell wider: Signalaufnahme, Signalverarbeitung und die Entscheidung des Neurons, ob es feuert oder nicht.

Teil I: Das biologische Neuron in Lebewesen

Das Gehirn - das Organ, mit dem wir denken - besteht aus vielen "denkenden Zellen", den Neuronen.
Diese Neuronen haben eine typische, gemeinsame Struktur:

a) In einem vorderen Teil der Neuronenzelle (dem Zellkörper mit Kern und Dendriten) werden die Signale von anderen Neuronen aufgezeichnet und addiert.

b) Wenn diese zusammengefassten Signale von anderen Nervenzellen stark genug sind, d.h. wenn ein bestimmter Schwellenwert der Erregung überschritten wird, feuert das Neuron. ***Ein einzelnes Neuron trifft bereits eine Entscheidung: Wenn Signale eintreffen, feuert das Neuron oder nicht (eventuell mit unterschiedlicher Intensität). Das sind die Reaktionsmöglichkeiten, die ein Neuron hat.***

c) Ist dies der Fall, wird ein Signal über eine Signalleitung (das Axon) geleitet,

d) die dann über die Axonenden (Synapsen genannt) an weitere Neuronen weitergeleitet werden.

e) Auf diese Weise hat ein Signal eine bestimmte Richtung:
Vom Eingang wird es unter der Schwellenbedingung an weitere, nachgelagerte Neuronenebenen weitergeleitet.

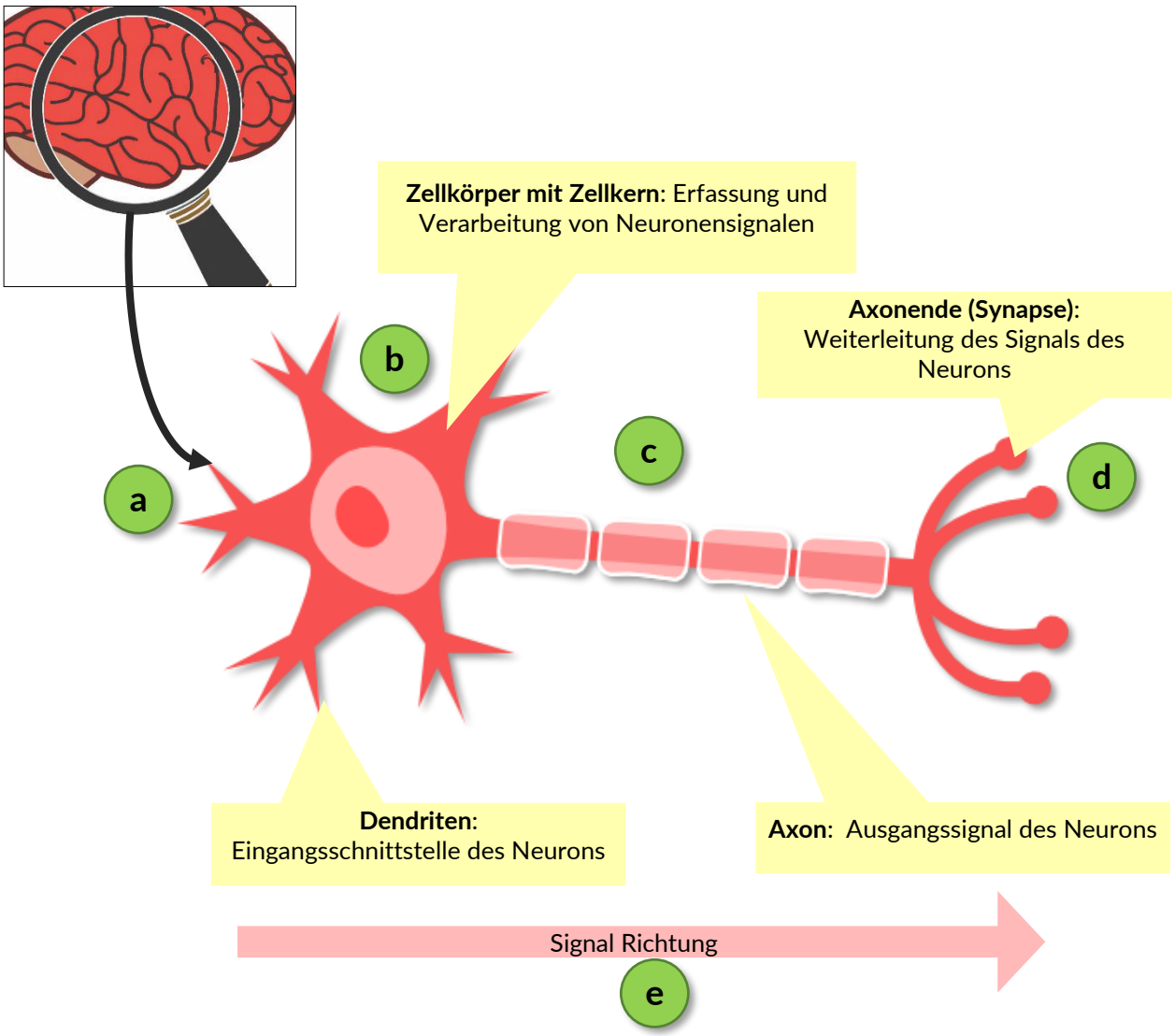


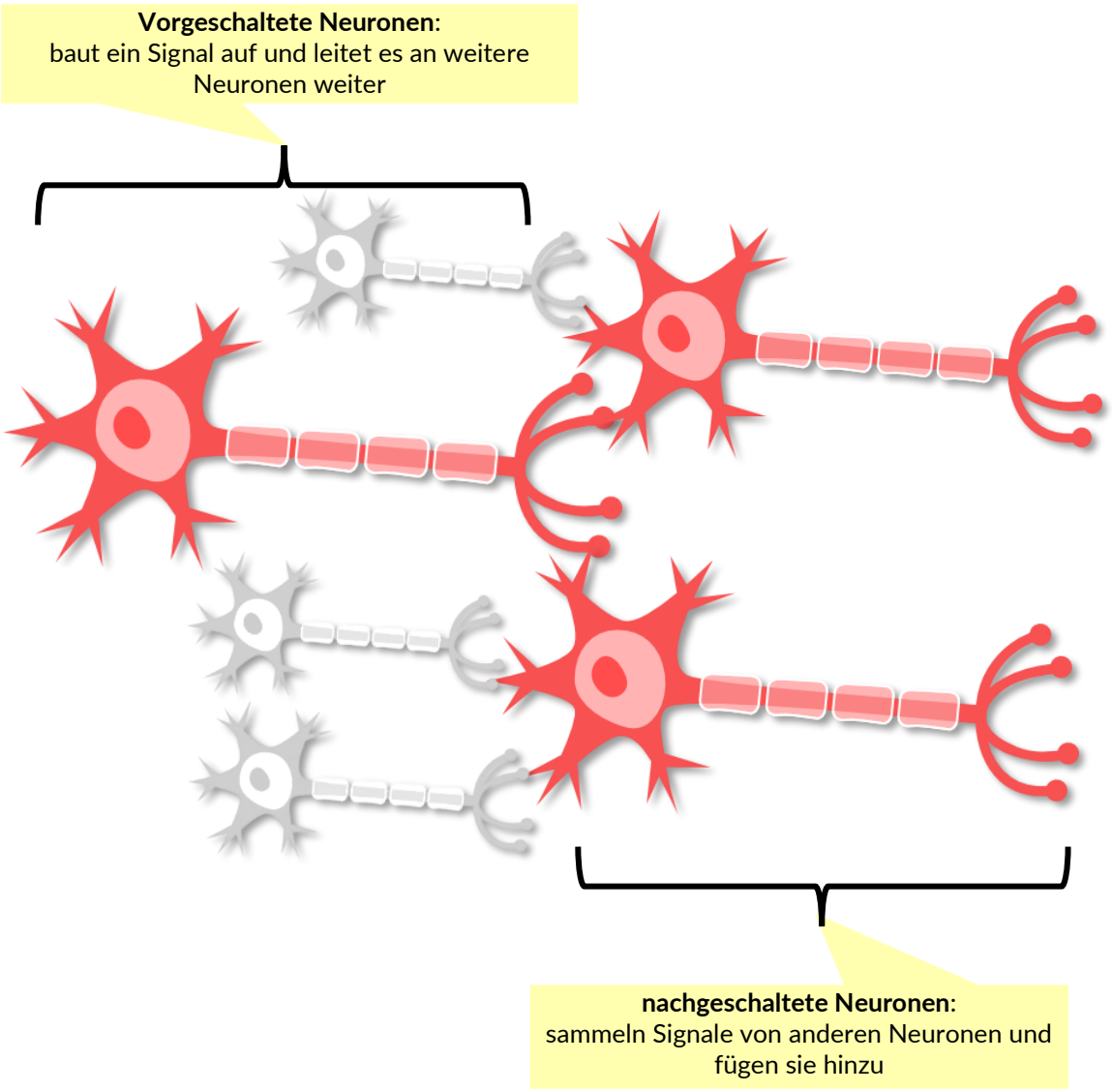
Abb. 4 Bild [Gemeinfrei] erzeugt mit DALL-E; Prompt „a brain in which a part is anlarged with the magnifying glass“ von Jörg [CC BY-SA 4.0 International]

Teil II: Verbindung von biologischen Neuronen zu neuronalen Netzen

Die Entscheidung eines einzelnen Neurons wird zu einem Teil eines ganzen Entscheidungs-Netzwerks:

- Viele Neuronen sind so miteinander verbunden, dass die Ausgabe eines Neurons die Eingabe des nächsten Neurons wird.
- Und es geht sogar noch weiter: Viele vorgelagerte Neuronen können die Eingänge eines nachgelagerten Neurons bilden.
- So sammelt ein nachgeschaltetes Neuron die Entscheidungen vieler Vorgänger, gewichtet sie und bildet daraus seine eigene Entscheidung.

Dies wird als emergentes Verhalten bezeichnet: Ein Netzwerk von miteinander verbundenen Neuronen ist also plötzlich zu intelligentem Verhalten fähig.



Die wahre Kraft der Entscheidungen liegt in den Verbindungen zwischen den Neuronen:

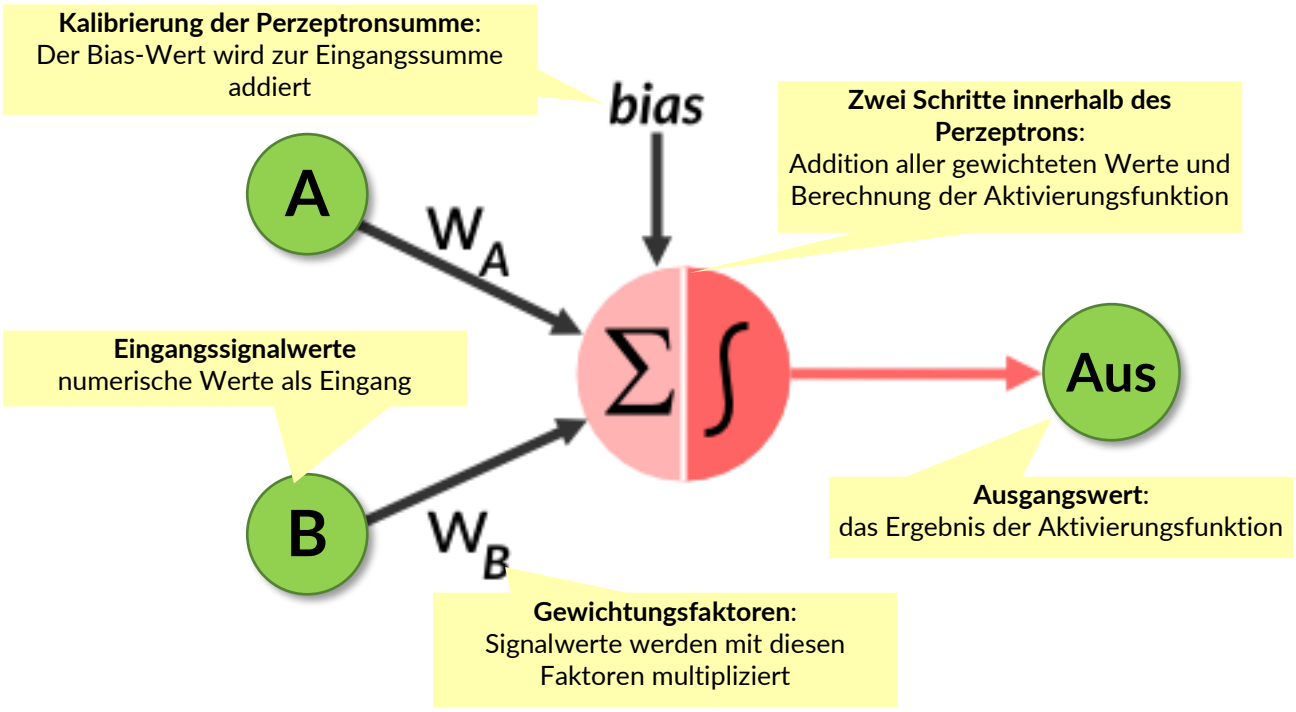
kann die synaptische Verbindung zwischen den Axon-Enden des Vorgängers und den Dendriten des Nachfolgers variiert werden:

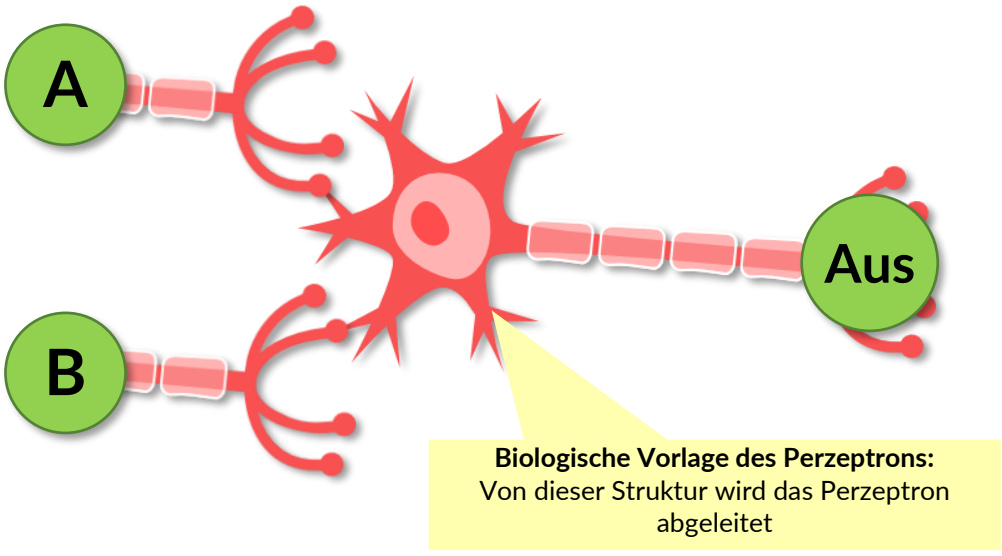
- Neuronen-Neuronen-Verbindungen können mehr oder weniger durchlässig sein. Man spricht von unterschiedlichen Gewichten der Verbindungen.

- Lernen entsteht dabei durch die Anpassung der Gewichte der Verbindungen

- Ein neuronales Netz kann auf diese Weise trainiert werden, indem
  die Gewichte angepasst werden, bis das Netz in der Lage ist, die gegebene Aufgabe zu erfüllen.

Teil III: Das künstliche Neuron einer Maschine





Das Perzeptron ist die Abstraktion des natürlichen Neurons:

- Signale von A und B gehen in das Perzeptron ein. Diese Signale werden durch numerische Werte, meist als Fließkommazahlen, dargestellt.
- Diese numerischen Signalwerte von A und B werden mit den Gewichtungsfaktoren wA und wB multipliziert. Sie wirken wie eine Signalverstärkung oder eine Signalabschwächung, je nach Größe des Gewichtungswertes.

Im Perzeptron werden zwei Berechnungen durchgeführt:

- Im ersten Schritt werden die gewichteten Zahlenwerte summiert.
- Manchmal wird der Signalsumme ein Bias-Wert hinzugefügt; er dient als eine Art Kalibrierung des Neurons und kann je nach Zweck frei variiert werden.
- Im zweiten Schritt wird diese Summe als Input einer sogenannten Aktivierungsfunktion weiterverwendet. Diese Aktivierungsfunktion kann jede beliebige Form annehmen, der Fantasie sind keine Grenzen gesetzt. Um die Berechnungen einfach zu halten, werden häufig einfache Funktionen verwendet, wie z. B. die so genannte Schwellenwertfunktion.
- Der Ausgang der Schwellenwertfunktion ist auch der Ausgang des gesamten Neurons.

Lektion 2: Der Perzeptron-Algorithmus

Was Schüler lernen sollten

Entmystifizierung: Künstliche Neuronen funktionieren in einfachen Schritten, die leicht zu wiederholen sind. Dies wird an verschiedenen Beispielen ausgearbeitet und nachberechnet. Es handelt sich um eine Drill-and-Practice-Lektion, die den Schülern ein Bild davon vermitteln soll, wie neuronale Netze funktionieren.

Mögliche Aktivitäten der Schüler

Die Schüler können selbst aktiv werden, denn Perzeptronen funktionieren in drei Rechenschritten:

1. Eingehende Signalwerte werden gewichtet, indem sie mit entsprechenden Faktoren multipliziert werden.
2. Anschließend wird alles addiert: gewichtete Eingangssignale und ein Korrekturfaktor namens "Bias".
3. Im letzten Schritt wird eine mathematisch einfache Aktivierungsfunktion verwendet, um zu berechnen, ob das Perzeptron ein Ausgangssignal liefern wird.

All dies wird von den Schülern anhand einfacher und anschaulicher Beispiele geübt.

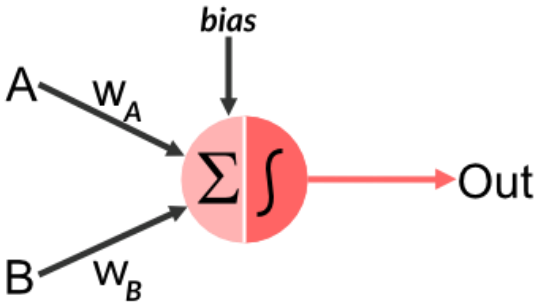
Teil I: Die Berechnungen des Perzeptrons erfolgen in folgenden Schritten:

Ganz am Anfang:

Legen Sie einen Schwellenwert für die Aktivierungsfunktion fest (*der in der Regel Null ist*)

Dann wiederholen Sie dies in einer Schleife, um Entscheidungen zu treffen:

1. Alle Eingabewerte mit ihren Gewichten multiplizieren
2. Summieren aller gewichteten Werte zu einer Summe
3. Aktivieren Sie den Ausgang, wenn die Summe größer ist als den Schwellenwert 0



Zum Beispiel setzen wir

die Gewichte $w_A = +0.8$ und $w_B = -0.3$ *(was negativ ist!)*

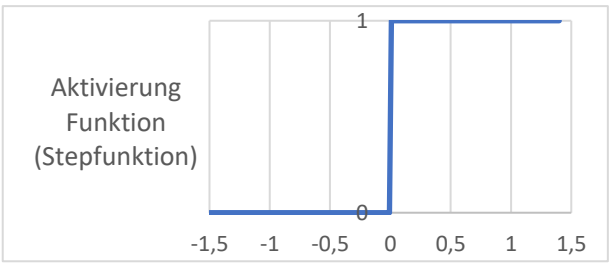
In unseren ersten Berechnungen vernachlässigen wir den Bias: $bias = 0$

Eingabe	...multipliziert mit den Gewichten w_A und w_B	... und zur Summe addiert	Aktivierung, Vergleich mit Schwellenwert 0	Ausgabe ∫
$A = 1$	$1 \cdot 0.8 = 0.8$	$= 0.8 - 0.3$ $= 0.5$	$Output = \begin{cases} 0, & \Sigma \leq 0^{**} \\ 1, & \Sigma > 0 \end{cases}$	1 da > 0
$B = 1$	$1 \cdot -0.3 = -0.3$			

** In einfacher Sprache bedeutet dies: Wenn die Summe kleiner oder gleich 0 ist, ist die Ausgabe 0. Wenn die Summe größer als 0 ist, ist die Ausgabe 1.

Wir können für das gesamte Perzeptron schreiben:

$$Output = \begin{cases} 0, & if \quad A \cdot w_A + B \cdot w_B + bias \leq 0 \\ 1, & if \quad A \cdot w_A + B \cdot w_B + bias > 0 \end{cases}$$



Aufgabe 1: Berechnen Sie die gewichteten Signale, die Summe und den Ausgang:

Eingabe	...multipliziert mit den Gewichten w_A und w_B	... und addiert zur Summe	Aktivierung	Ausgabe $\int$
$A = 1$	_____ $\cdot 0.8 =$ _____	$=$ _____	$Output = \begin{cases} 0, & \Sigma \leq 0^{**} \\ 1, & \Sigma > 0 \end{cases}$	_____
$B = 0$	_____ $\cdot$ -0.3 $=$ _____			

Eingabe	...multipliziert mit den Gewichten w_A und w_B	... und addiert zur Summe	Aktivierung	Ausgabe $\int$
$A = 0$	_____ $\cdot 0.8 =$ _____	$=$ _____	$Output = \begin{cases} 0, & \Sigma \leq 0^{**} \\ 1, & \Sigma > 0 \end{cases}$	_____
$B = 1$	_____ $\cdot$ -0.3 $=$ _____			

Aufgabe 2: Berechnen Sie die Ergebnisse für bias = +0.2

A	$\cdot w_A$		B	$\cdot w_B$		Bias		Summe	Ausgabe
0			0			0.2			
0			1						
1			0						
1			1						

Lösung für alle vier möglichen Eingabefälle in einer Tabelle:

A	$\cdot w_A$		B	$\cdot w_B$		Bias		Summe	Ausgabe
0	*0*		0	*0*		0		0	0
0	*0*		1	*-0.3*				-0.3	0
1	*0.8*		0	*0*				0.8	1
1	*0.8*		1	*-0.3*				0.5	1

Lösung für Bias = +0,2

A	$\cdot w_A$		B	$\cdot w_B$		Bias		Summe	Ausgabe
0	*0*		0	*0*		0.2		0.2	1
0	*0*		1	*-0.3*				-0.1	0
1	*0.8*		0	*0*				1.0	1
1	*0.8*		1	*-0.3*				0.7	1

Teil 2: Ein Perceptron trifft Entscheidungen

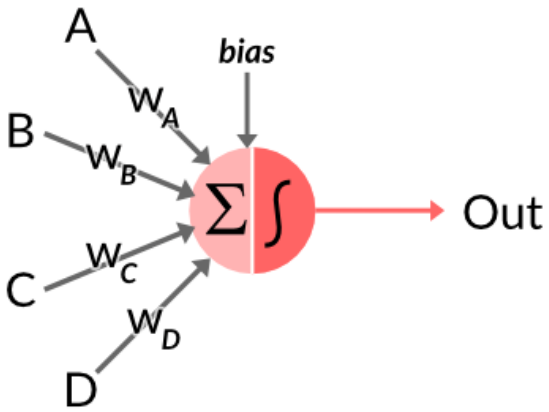
Abb. 5 Bild [Gemeinfrei] erzeugt mit DALL-E; Prompt „a pizza with onions, salami and pinapple on a dark blue table, professional food phootography“ von Jörg [CC BY-SA 4.0 International]

Ihr Perceptron muss eine wichtige Entscheidung treffen:
Soll ich heute eine Pizza essen gehen?

Für diese Entscheidung gibt es vier Kriterien:

A) Ich bin hungrig genug — Gewichtsfaktor $w_A = +0.6$
B) Es regnet ☹ — Gewichtungsfaktor $w_A = -0.3$
C) Ich habe Lust auf eine Pizza! — Gewichtsfaktor $w_A = +0.4$
D) Ich habe einen Wunsch für Hamburger — Gewichtsfaktor $w_A = -0.5$

Kriterien mit einem positiven Gewichtungswert wie A oder C haben eine fördernde, solche mit einem negativen Gewichtungswert wie B oder D eine hemmende Wirkung.



Aufgabe 3:
Berechnen Sie beide Tabellen für die Verzerrung +/- 0.1?

Aufgabe 4:
Wie würden Sie die Auswirkung des Bias-Wertes interpretieren?

Was ändert sich, wenn der Wert kleiner wird?

A	· w_A		B	· w_B		C	· w_C		D	· w_D		Bias		Sum	Out
0			0			1			1						
1			0			1			1			+0.1			
0			1			1			0						

A	· w_A		B	· w_B		C	· w_C		D	· w_D		Bias		Sum	Out
0			0			1			1						
1			0			1			1			-0.1			
0			1			1			0						

Lösung
Ein hoher Bias-Wert würde einer aktiven Person entsprechen, die nicht viel äußeren Einfluss braucht, um sich zu begeistern. Ein niedriger oder sogar negativer Wert würde eine Person im Sessel sitzen lassen, selbst wenn einige aktivierende Kriterien erfüllt sind.

Lösung für Verzerrung = +0.1 | Out = 0, 1, 1

Lösung für Verzerrung = -0.1 | Out = 0, 1, 0

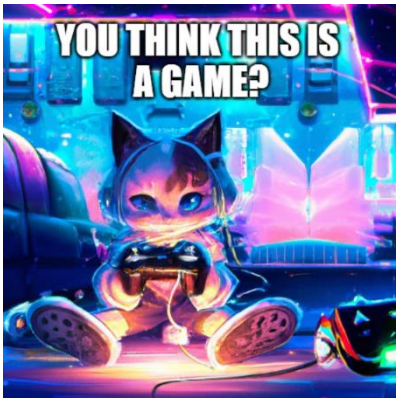


Abb. 6 Bild [Gemeinfrei] erzeugt mit Stable Diffusion; Prompt „a cat playing video game“ von Jörg [CC BY-SA 4.0 International]

Lektion 3: Jetzt wird es ernst: Logik-Funktionen

Was Schüler lernen sollten

Boolesche Logikfunktionen gehören zum Grundrepertoire der theoretischen Informatik, sie ziehen sich wie ein roter Faden durch alle ihre Teilgebiete. Alle anderen Prinzipien beruhen auf ihnen - von der Digitaltechnik über die Kontrollstrukturen von Programmiersprachen bis zur Automatentheorie. Die Boolesche Algebra ist Teil des genetischen Materials der Informatik.

Daher werden auch künstliche Neuronen in dieses übergreifende Konzept eingeordnet: Können einfache Neuronen boolesche Logikfunktionen nachbilden? In diesem Zusammenhang entpuppt sich das einfache Perzeptron als binärer Klassifikator, der neben der binären Entweder-Oder-Teilung weitere, erstaunliche Fähigkeiten besitzt.

Mögliche Aktivitäten der Schüler

Die logischen Funktionen der UND-, EINSCHLIESSUNG- und ODER-Verknüpfung werden wiederholt. Während die Schüler mit der booleschen "Wahrheitstabelle" vertraut sein sollten, wird das "Wahrheitsdiagramm" als ergänzendes Werkzeug eingeführt. Die Schüler können die Gewichtungsfaktoren für die klassifizierenden Wahrheitstabellen des Perzeptrons berechnen.

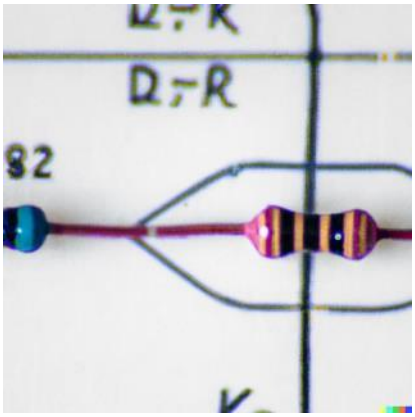
Teil I: Überarbeitung der Booleschen Logikfunktionen

Alle 16 logischen Operationen eines binären Operators können hier nicht wiederholt werden. Wir beschränken uns auf eine Auswahl der relevanten und wichtigen Vertreter, die logischen:

ODER UND Implikation XOR NICHT

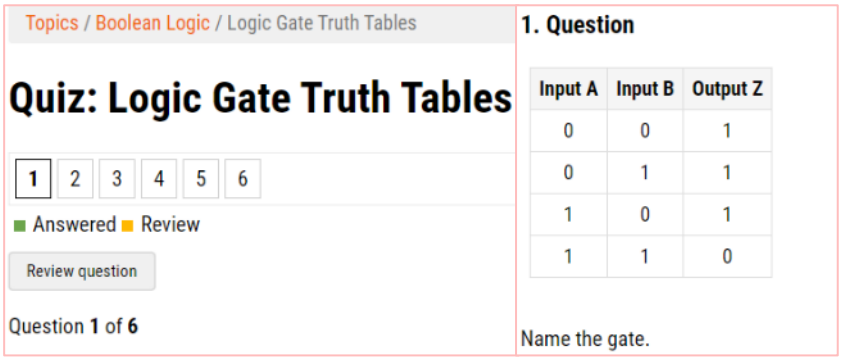
Eine gute Übersicht finden Sie hier:

https://en.wikipedia.org/wiki/Truth_table



Aufgabe 5: Eine kurze Übung:

https://www.computerscience.gcse.guru/quiz/logic-gate-truth-tables



Teil II: Mnemonics für logische Funktionen und Wahrheitstabellen

Abb. 7 Bild [Gemeinfrei] erzeugt mit Stable Diffusion; Prompt „a vanilla ice and chocolate on a dark blue table, professional food phootography“ von Jörg [CC BY-SA 4.0 International]

Mnemonik für OR

- Nach dem Mittagessen gibt es einen Nachtisch: Schokolade oder Eiscreme.
Natürlich kann man beides haben!

A steht für Schokolade, B für Eiscreme,

- keine Schokolade bedeutet "0",
- kein Eis bedeutet „0“.
- Alle Kombinationen sind möglich 😉 .

A	B	Aus
0	*0*	0
0	1	1
1	*0*	1
1	1	1

Mnemonik für AND

Um ein Auto zu fahren, müssen Sie mindestens 18 Jahre alt sein und einen Führerschein besitzen. Beide Bedingungen müssen erfüllt sein! Dies ist die einzige Kombination, die erlaubt ist. *(sogar in diesen drei Ländern wie Ägypten, Honduras und Indien)* 😉

A	B	Aus
0	*0*	0
0	1	0
1	*0*	0
1	1	1

Eselsbrücke für IMPLICATION

Wenn es regnet, ist die Straße nass. Es ist eine 'wenn ... dann'-Verknüpfung. Und sie ist nicht ganz intuitiv, deshalb erklären wir sie im Detail:

- Wenn es nicht regnet (A=0), ist die Straße nicht nass (B=0). Dies *kann* wahr sein.
- Wenn es nicht regnet (A=0), ist die Straße nass (B=1). Dies *kann* wahr sein.
- Wenn es regnet (A=1), ist die Straße nicht nass (B=0). Dies ist falsch.
- Wenn es regnet (A=1), ist die Straße nass (B=1). Das ist wahr.

A	B	Aus
0	*0*	1
0	1	1
1	*0*	0
1	1	1

Abb. 8 Bild [Gemeinfrei] erzeugt mit Stable Diffusion; Prompt „a toy train and plane on wood table, professional phootography“ von Jörg [CC BY-SA 4.0 International]

Mnemonik für XOR

Entweder man fährt mit dem Zug oder man fliegt mit dem Flugzeug.
Man kann nicht beides gleichzeitig machen.

A	B	Aus
0	*0*	0
0	1	1
1	*0*	1
1	1	0

Aufgabe: Beantworten Sie die folgenden Fragen zu den logischen Funktionen:

- Finden Sie andere Mnemonics für diese logischen Funktionen?
- Wenn ja: Können Sie sie erklären?
- Welche logische Funktion ist das? *"Wenn ein Feuer ausbricht, muss es Sauerstoff geben.* "

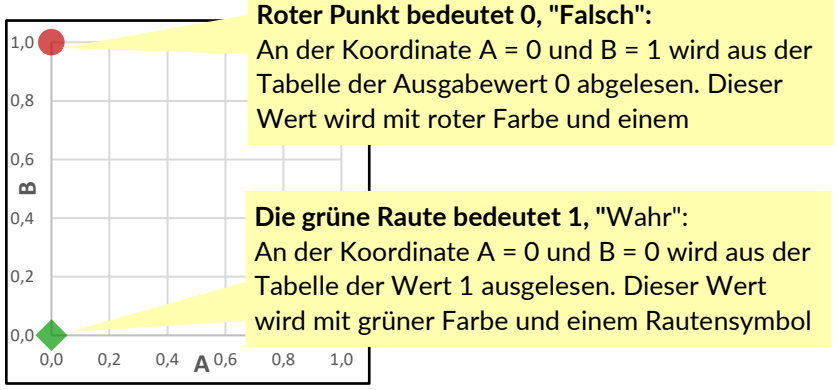
Teil III: Was ist ein "Wahrheitsdiagramm"?

Eine kurze Erklärung des häufig verwendeten Wahrheitsdiagramms, das eine Ergänzung zur Wahrheitstabelle darstellt:

Angesichts dieser Wahrheitstabelle:

A	B	Aus
0	*0*	1
0	1	0

... die auch als Diagramm dargestellt werden kann, mit A als x-Achse und B als y-Achse.



Roter Punkt bedeutet 0, "Falsch":
An der Koordinate A = 0 und B = 1 wird aus der Tabelle der Ausgabewert 0 abgelesen. Dieser Wert wird mit roter Farbe und einem

Die grüne Raute bedeutet 1, "Wahr":
An der Koordinate A = 0 und B = 0 wird aus der Tabelle der Wert 1 ausgelesen. Dieser Wert wird mit grüner Farbe und einem Rautensymbol

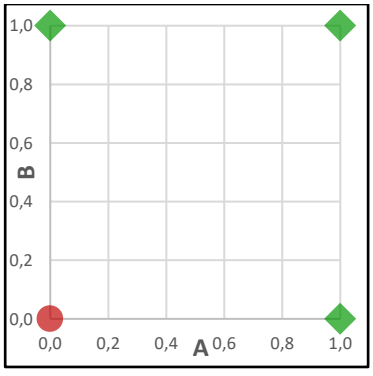
Für die Darstellung einfacher Wahrheitstabellen benötigen Sie keine grafische Darstellung, da die Zwischenbereiche zwischen den diskreten Werten ungenutzt bleiben. Die Darstellung wird jedoch in den späteren Kapiteln sinnvoll, wenn der Wertebereich erweitert wird. Deshalb wird diese Darstellung bereits an dieser Stelle eingeführt.

Logisches ODER

Wahrheitstabelle

A	B	Aus
0	*0*	0
0	1	1
1	*0*	1
1	1	1

Wahrheitsdiagramm



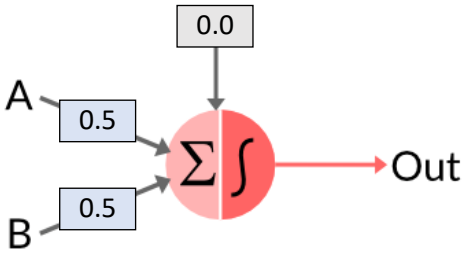
Gewichte

w_A
0.5

w_B
0.5

Bias
0.0

Perceptron



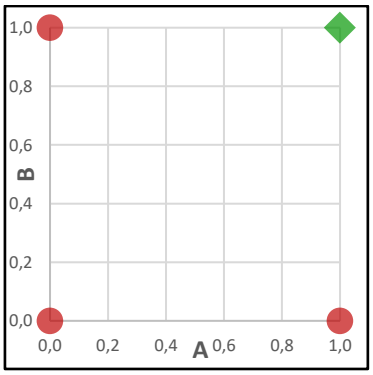
A	· w_A		B	· w_B		Bias		Summe	Ausgabe
0	*0*		0	*0*		0.0		0	0
0	*0*		1	*0.5*				0.5	1
1	*0.5*		0	*0*				0.5	1
1	*0.5*		1	*0.5*				1.0	1

Logisches UND

Wahrheitstabelle

A	B	Aus
0	*0*	0
0	1	0
1	*0*	0
1	1	1

Wahrheitsdiagramm



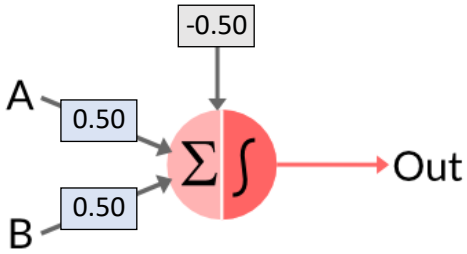
Gewichte

w_A
0.50

w_B
0.50

Bias
-0.50

Perceptron



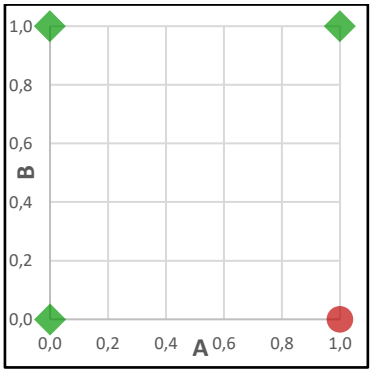
A	· w_A		B	· w_B		Bias		Summe	Ausgabe
0	*0*		0	*0*		-0.50		-0.50	0
0	*0*		1	*0.5*				0	0
1	*0.5*		0	*0*				0	0
1	*0.5*		1	*0.5*				+0.50	1

Logische Implikation

Wahrheitstabelle

A	B	Aus
0	*0*	1
0	1	1
1	*0*	0
1	1	1

Wahrheits-Diagramm



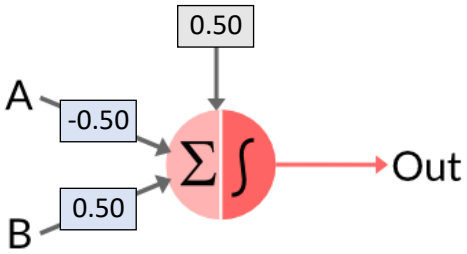
Gewichte

w_A
-0.50

w_B
0.50

Bias
0.50

Perceptron



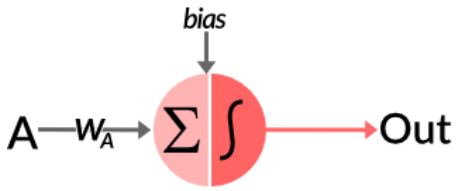
A	· w_A		B	· w_B		Bias		Summe	Ausgabe
0	*0*		0	*0*		0.50		0.5	1
0	*0*		1	*0.5*				1.0	1
1	*-0.5*		0	*0*				0	0
1	*-0.5*		1	*0.5*				0.5	1

Die Tabelle der logischen Implikationen kann für Klassenarbeiten verwendet werden. Sie ist nicht sehr wichtig, kann aber verwendet werden, um das Verständnis der Schüler für die Konzepte zu testen.

Arbeitsblatt: Logische Funktionen

Aufgabe 6: Logisches NOT

Stellen Sie sich ein Neuron vor, das den Wert einer Eingabe invertiert: Es verwandelt eine Eingabe-1 in eine Ausgabe-0 und umgekehrt. Wie sollten der Gewichtungsfaktor und der Bias eingestellt werden, damit dies funktioniert?



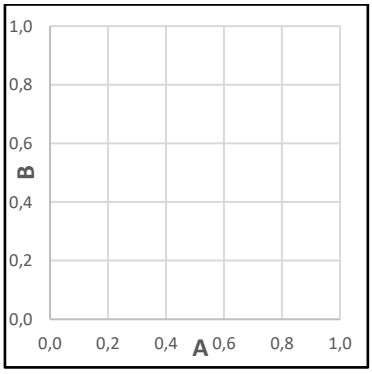
A	· w_A		Bias		Summe	Ausgabe
0	*0*		0.5		0.5	1
1	*-1*				-0.5	0

Aufgabe 7: Logische Hemmungen

Wahrheitstabelle

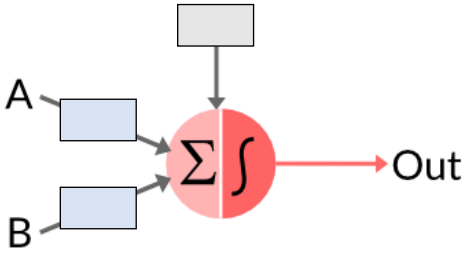
A	B	Aus
0	*0*	0
0	1	1
1	*0*	0
1	1	0

Wahrheits-Diagramm



Gewichte

Perceptron



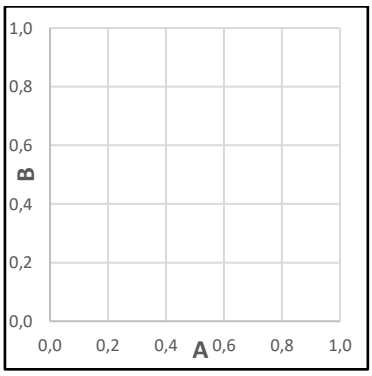
A	· w_A		B	· w_B		Bias		Summe	Ausgabe
0			0						0
0			1						1
1			0						0
1			1						0

Aufgabe 8: Logisches NAND, Invertierung der AND-Funktion

Wahrheitstabelle

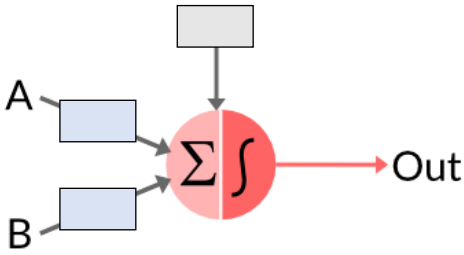
A	B	Aus
0	*0*	1
0	1	1
1	*0*	1
1	1	0

Wahrheitsdiagramm



Gewichte

Perceptron



A	· w_A		B	· w_B		Bias		Summe	Ausgabe
0			0						1
0			1						1
1			0						1
1			1						0

Lösung für NOT:

w_A = -0.5

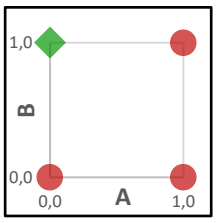
Bias = 0.5

Lösung für Inhibition:

w_A = -0.5

w_B = 0.5

Bias = 0

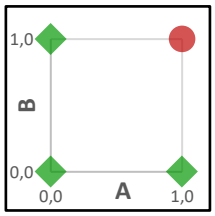


Lösung für NAND:

w_A = -0.5

w_B = -0.5

Bias = 1

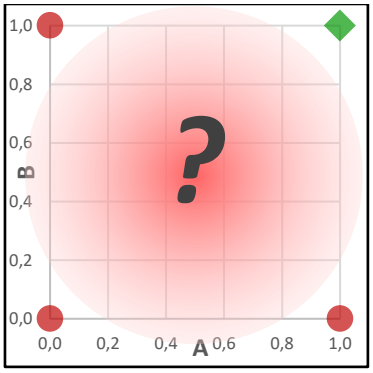


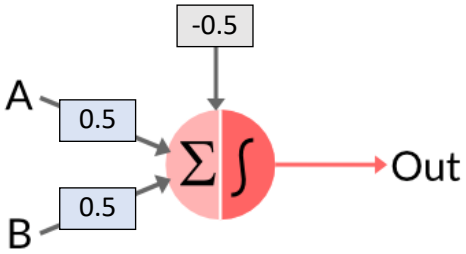
Lektion 4: Perceptron als binärer Klassifikator

Im letzten Kapitel wurde gelehrt, dass ein Perzeptron verschiedene logische Operationen durch Anpassung der Gewichte abbilden kann. Aber das Perceptron kann mehr als nur binäre Divisionen berechnen, es ist viel leistungsfähiger.

Abb. 9 Bild [Gemeinfrei] erzeugt mit DALL-E; Prompt „an old man with beard, hat and light ball“ von Jörg [CC BY-SA 4.0 International]

Teil I: Das logische UND neu betrachtet. Was ist im Bereich zwischen den vier Punkten?





In der Perceptron-Gleichung wird nichts über die akzeptierten Werte für A und B gesagt:

$$Output = \begin{cases} 0, & if \quad \mathrm{A} \cdot \mathrm{w}_A + \mathrm{B} \cdot \mathrm{w}_B + bias \leq 0 \\ 1, & if \quad \mathrm{A} \cdot \mathrm{w}_A + \mathrm{B} \cdot \mathrm{w}_B + bias > 0 \end{cases}$$

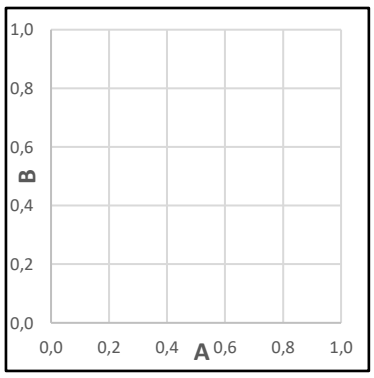
Was passiert, wenn wir verschiedene Werte in unsere Gleichung einsetzen? Zum Beispiel:

Aufgabe 9: "LOGISCHES UND"-Perzeptron

Berechnen Sie die fehlenden Werte und zeichnen Sie das Wahrheitsdiagramm:

A	· 0.5		B	· 0.5		Bias		Summe	Ausgabe
0.5			0.5			-0.5			
0.8			0.6						
0.2			0.6						
1			0.1						

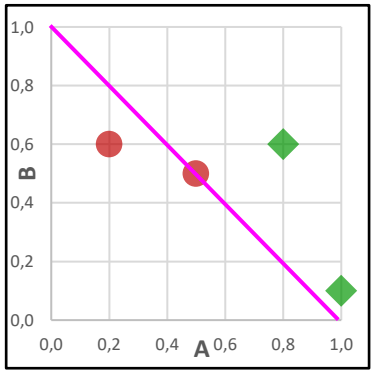
Können Sie sich vorstellen, wie die Grenzlinie



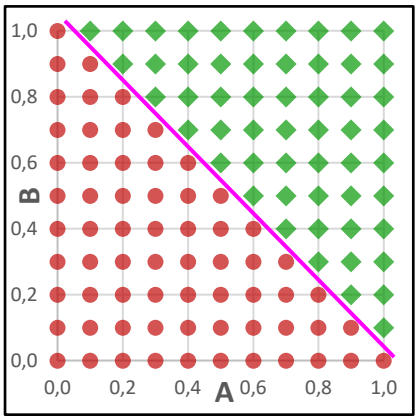
verlaufen könnte?

Lösung für Aufgabe 9:

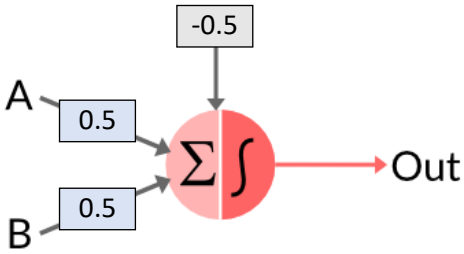
A	· 0.5		B	· 0.5		Bias		Summe	Ausgabe
0.5	*0.25*		0.5	*0.25*		-0.5		0	0
0.8	*0.4*		0.6	*0.3*				0.2	1
0.2	*0.1*		0.6	*0.3*				-0.1	0
1	*0.5*		0.1	*0.05*				0.05	1

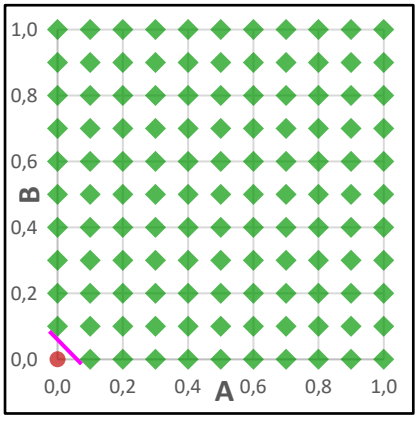


Teil II: Ein systematischer Ansatz mit vielen Berechnungen

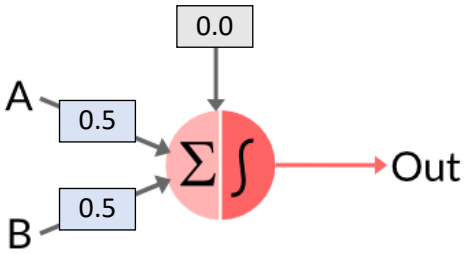


Wenn wir eine Wertewolke für das AND-Perceptron berechnen, erhalten wir dieses Diagramm, in dem die Grenzlinie leicht zu erkennen ist:

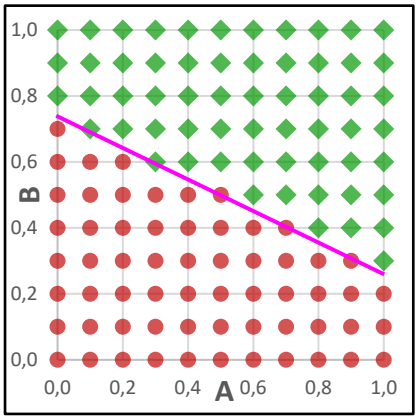




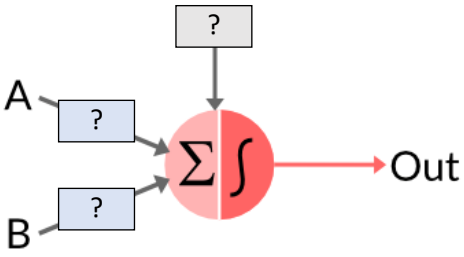
Wenn wir diese vielen Werte für das OR-Perceptron berechnen, erhalten wir dieses Diagramm, in dem die Grenzlinie ebenfalls klar zu sein scheint



Aufgabe 10: eine andere lineare Gleichung



Können Sie **abschätzen** (nicht berechnen), wie die Gewichtswerte des Perzeptrons angepasst werden müssen, um dieses Diagramm zu erhalten?



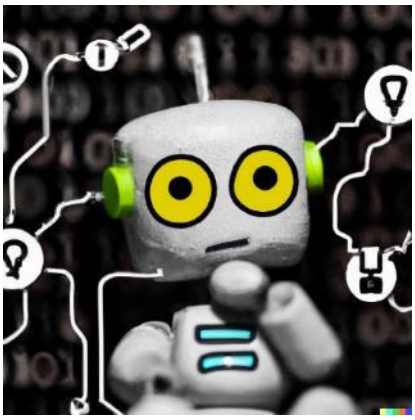
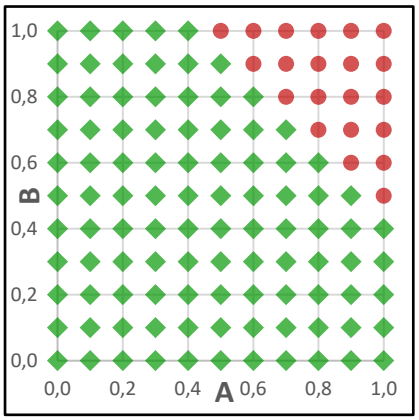
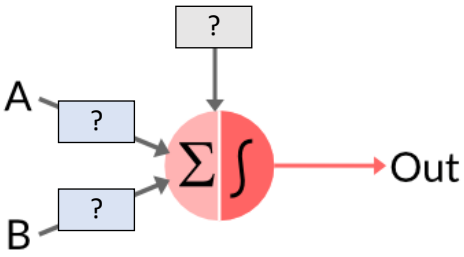


Abb. 10 Bild [Gemeinfrei] erzeugt mit DALL-E; Prompt „a photo of a cute robot who is looking at a complicated electronical circuit“ von Jörg [CC BY-SA 4.0 International]

Aufgabe 11: eine optimale lineare Gleichung für NAND



Können Sie berechnen, wie die Gewichtswerte des Perzeptrons angepasst werden müssen, um dieses Diagramm zu erhalten?



Lösung Aufgabe 10:

Die Gewichtungswerte sind $w_A = 0.3$, $w_B = 0.7$, Verzerrung = -0.5

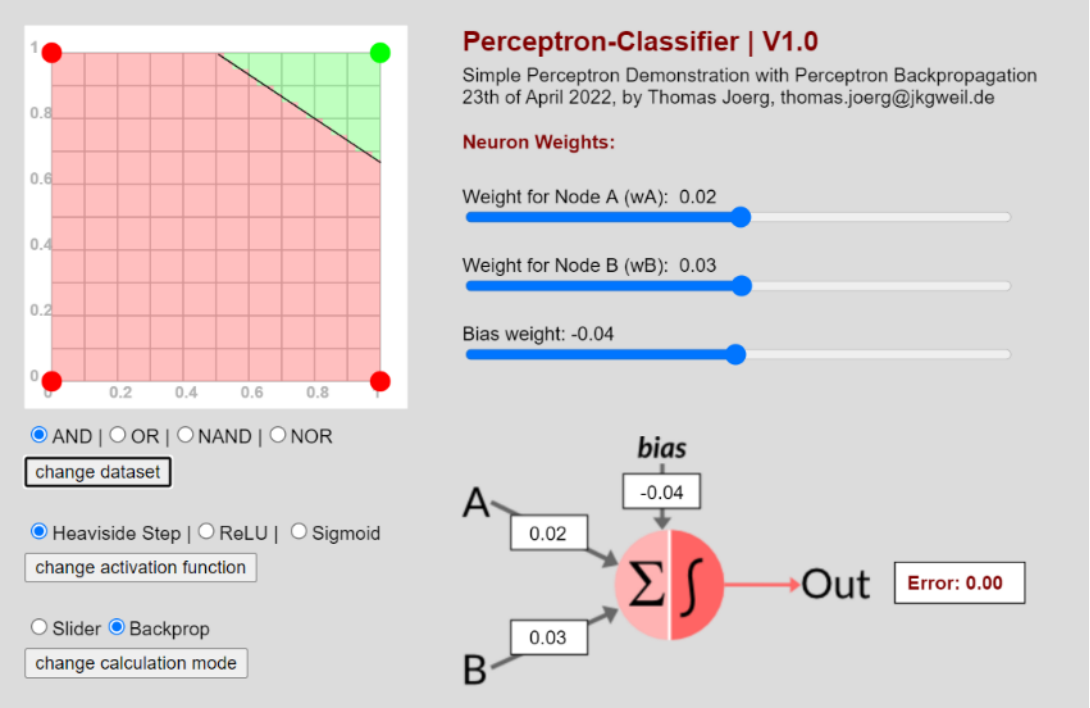
Lösung Aufgabe 11:

Die Gewichtungswerte sind $w_A = -0.5$, $w_B = -0.5$, bias = +0.75

Teil III: Spielen mit einer Perceptron-Simulation

Um ein Gefühl für die Funktionsweise eines Perzeptrons zu entwickeln, ist es eine gute Idee, eine Simulation als Unterstützung zu verwenden. In der folgenden Simulation können die Schüler einige Aufgaben spielerisch und experimentell lösen.

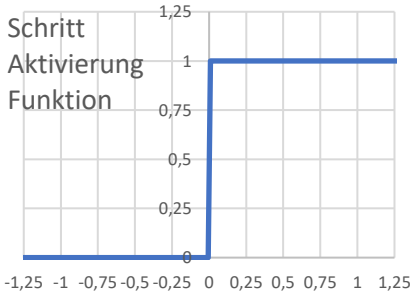
https://iludis.de/PerceptronArea/index.html

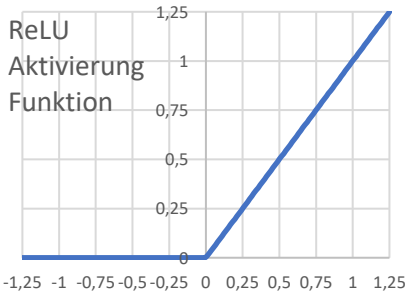


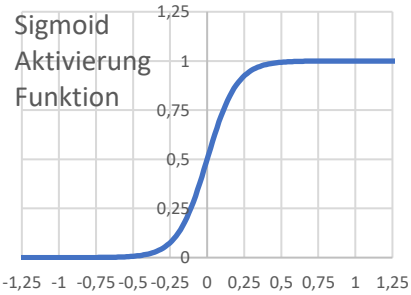
Aufgabe 12:

a) Stellen Sie den Berechnungsmodus auf "Schieberegler". Auf diese Weise können Sie das Perzeptron mit Hilfe der drei Schieberegler selbst einstellen.
Verändern Sie die Schieberegler so, dass der Fehler gleich Null ist. Was ändert sich in w_A, in w_B und im Bias?

b) Schalten Sie die Aktivierungsfunktion auf ReLU und später auf Sigmoid um. Warum ändert sich die Darstellung?
Unten sehen Sie die drei verschiedenen Aktivierungsfunktionen. Wie verhalten sie sich?







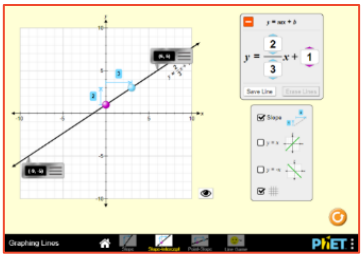
c) Versetzen Sie nun das Perzeptron in den Lernmodus mit Backpropagation. Erzielt das Perzeptron immer einen Lernerfolg? Wie können Sie es beeinflussen?

Teil IV: Mathematische Betrachtung der Perceptron-Klassifikation oder: Es muss einen einfacheren Weg geben!

In einem Mathematikkurs in der Mittelstufe kennt jeder Schüler die Formel für eine lineare Gleichung mit m als Wert für die Steigung und b als Konstante für den Achsenabschnitt:

$$y = m \cdot x + b$$

Lineare Gleichungen

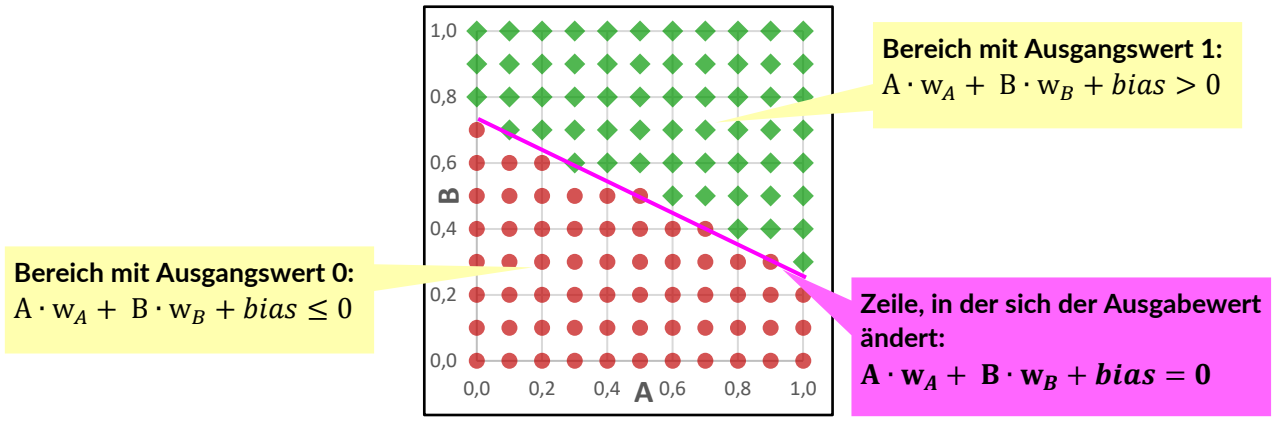


https://phet.colorado.edu/sims/html/graphing-lines/latest/graphing-lines_en.html

Wenn es notwendig erscheint, die Grundlagen der linearen Gleichungen zu wiederholen oder zu vertiefen, können Sie mit dieser Simulation beginnen. Die Grundlagen von Y-Achsenschnitt und Steigung werden dabei interaktiv erarbeitet und geübt. Ein interessantes Rätsel ("Linienspiel") rundet die Simulation ab.

Können beide Gleichungen - die lineare und des Perzeptrons - ineinander umgewandelt werden?

Die beiden Bereiche, in denen die unterschiedlichen Werte für den Output gelten, müssen irgendwo eine Grenzlinie haben. Diese Grenzlinie muss sich mit einer linearen Gleichung beschreiben lassen - und die Information dafür ist in der Perceptron-Gleichung enthalten.



Unsere Anfangsgleichung für die Grenzlinie: $A \cdot w_A + B \cdot w_B + bias = 0$

Zur besseren Verständlichkeit benennen wir beide Achsen um: Die Eingangsvariable A wird durch Umbenennung zur Variable x. Auf die gleiche Weise wird die Eingangsvariable B durch Umbenennung zur Variablen y.

$$x \cdot w_A + y \cdot w_B + bias = 0$$

Das Umstellen der Gleichung ergibt:

$$y \cdot w_B = -x \cdot w_A - bias$$

$$y = -\frac{w_A}{w_B} \cdot x \quad -\frac{bias}{w_B}$$

Diese Gleichung kann interpretiert werden:

Steigung: $-\frac{w_A}{w_B}$

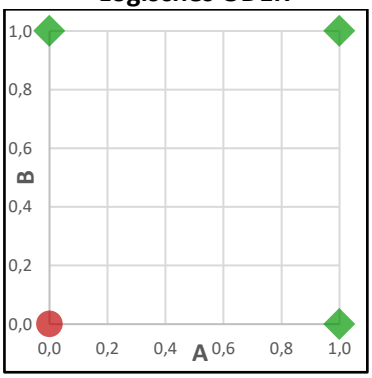
Y-Achsenabschnitt: $-\frac{bias}{w_B}$

Wenn w_A und w_B das gleiche Vorzeichen haben, ist die **Steigung** negativ.
Wenn w_A und w_B entgegengesetzte Vorzeichen haben, ist die **Steigung** positiv.

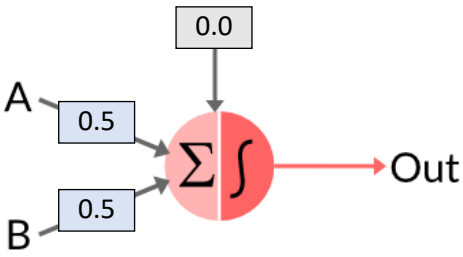
Wenn Bias und w_B die gleichen Vorzeichen haben, ist der **Y-Achsenabschnitt** negativ.
Wenn Bias und w_B entgegengesetzte Vorzeichen haben, ist der **Y-Achsenabschnitt** positiv.

Aufgabe 13: Berechne die linearen Gleichungen für die Logik-Funktionen und zeichne ihre Linie ein:

Logisches ODER



Perceptron

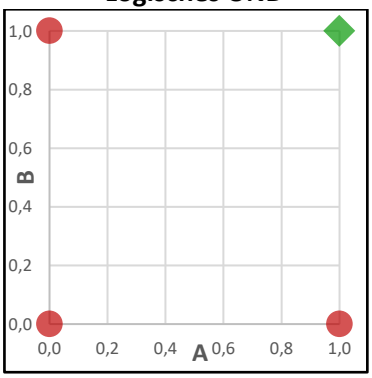


Gleichung:

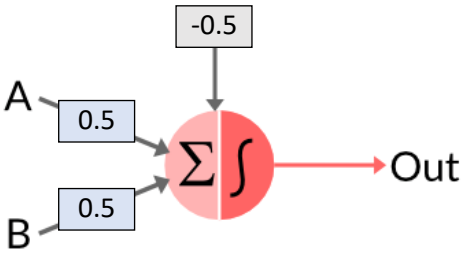
Start:

$$0.5 \cdot A + 0.5 \cdot \mathrm{B} + 0 = 0$$

Logisches UND



Perceptron

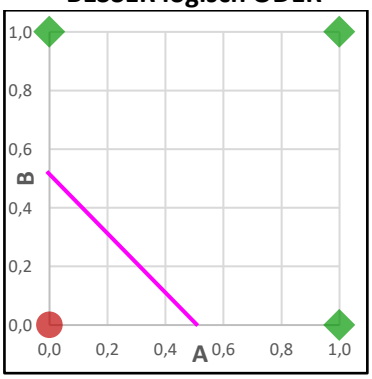


Gleichung:

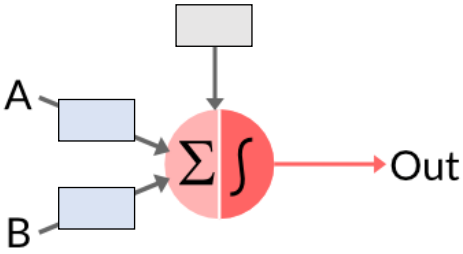
Start:

$$0.5 \cdot A + 0.5 \cdot \mathrm{B} - 0.50 = 0$$

BESSER logisch ODER



Perceptron



Gleichung:

a) Versuchen Sie, die gezeichnete Linie in eine lineare Gleichung zu übersetzen. Der Schnittpunkt der Geraden mit der y-Achse liegt bei 0.5.

b) Warum ist die gerade Linie besser?

c) Versuchen Sie dann, die lineare Gleichung in die Perceptron-Gleichung umzuwandeln.

Lösung:

Logisches Und: y = -x logisches ODER: y = -x + 1besseres logisches ODER: y = -x + 0.5, $A \cdot 0.5 + \mathrm{B} \cdot 0.5 - 0.25 = 0$

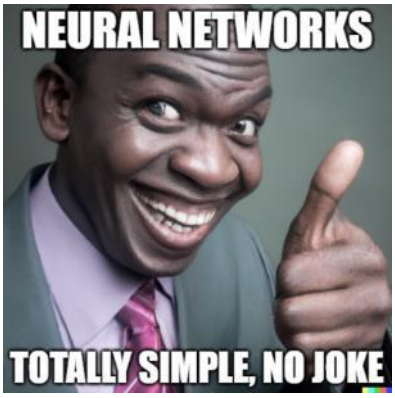


Abb. 11 Bild [Gemeinfrei] erzeugt mit DALL-E; Prompt „black man laughing, giving thumbs up “ von Jörg [CC BY-SA 4.0 International]

Lektion 5: Einführung in neuronale Netze, XOR-Funktion

Was Schüler lernen sollten

Das Eingabeproblem der nicht linear trennbaren XOR-Funktion führt zur Kombination von Neuronen zu einem neuen Gebilde: dem mehrschichtigen neuronalen Netz und damit letztlich zum tiefen neuronalen Netz. An diesem Punkt sollte der Lehrer vorsichtig vorgehen, damit jeder Schritt sorgfältig nachvollzogen werden kann.

Mögliche Aktivitäten der Schüler

Die Schüler sollten zunächst versuchen, mit Hilfe des Wahrheitsdiagramms eine Perzeptron-Gewichtung zu finden, die der XOR-Verknüpfung zugeordnet ist. Sie werden feststellen, dass dies nicht funktionieren kann: Sie brauchen zwei Diskriminatorlinien, um diese Logik vollständig zu beschreiben.

Anhand einer grafischen Ausarbeitung der XOR-Verknüpfung finden die Schüler heraus, dass die Lösung durch eine Überlagerung von UND- und ODER-Perzeptrons erreicht wird. Ein drittes Perzeptron wird benötigt, um die Ausgänge der beiden Perzeptronen zu kombinieren.

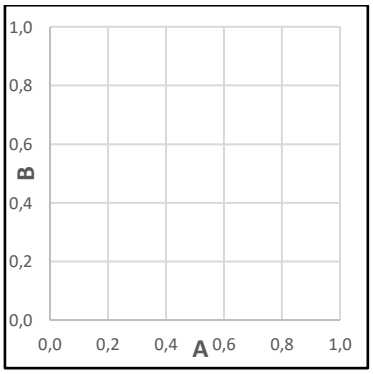
An dieser Stelle erarbeiten sich die Schüler das Funktionsprinzip des mehrschichtigen neuronalen Netzes: Einzelne Neuronen übernehmen Teilbereiche einer Gesamtaufgabe, nachgeschaltete Neuronen fügen die einzelnen Teilbereiche zu einem Ganzen zusammen.

Aufgabe 14: Zeichnen Sie das Wahrheitsdiagramm und versuchen Sie, die Perzeptrongewichte zu finden

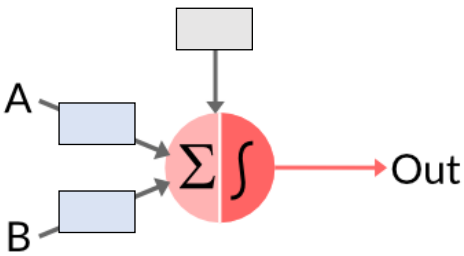
Wahrheitstabelle

A	B	Aus
0	*0*	0
0	1	1
1	*0*	1
1	1	0

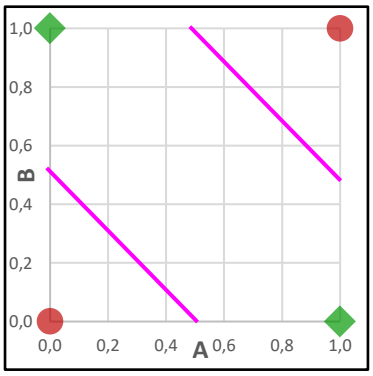
Wahrheits-Diagramm



Perceptron



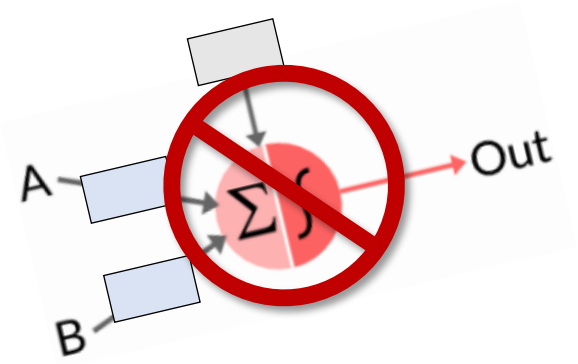
Lösung



Es ist die logische XOR-Funktion, und das Wahrheitsdiagramm sieht wie folgt aus:

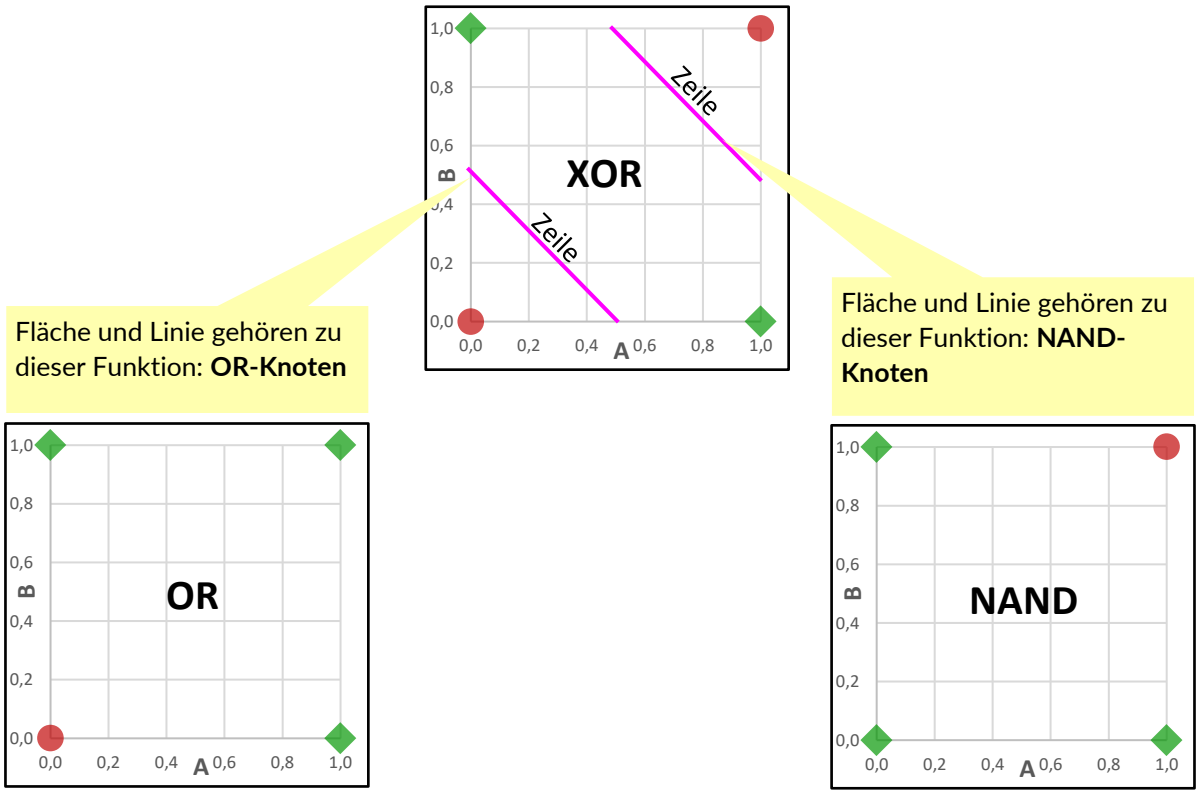
Die Suche nach den Gewichten für das Perzeptron ist vergeblich
Perzeptron zu suchen, weil die Gewichte aller bisher
bekannten logischen Funktionen auf eine
einer einzigen linearen Gleichung entsprechen (AND, OR, NAND, IMP).

Aber für die Trennung der XOR-Elemente
braucht man zwei gerade Linien; mit einer geraden
Linie kann man sie nicht trennen.



Teil II: Analyse des Problems

Die XOR-Funktion kann als eine Kombination oder Überlagerung einer ODER- und einer NAND-Funktion betrachtet werden:



Wenn das der Fall ist, muss man sich als Nächstes die Wahrheitstabellen der beiden logischen Funktionen ansehen:

A	B	OutA
0	*0*	0
0	1	1
1	*0*	1
1	1	1

OR

NAND

A	B	OutB
0	*0*	1
0	1	1
1	*0*	1
1	1	0

Kombination

Ein dritter Knoten muss eingeführt werden, um die Kombination rechnerisch durchzuführen

A	B	OutA	OutB	Aus
0	*0*	0	1	0
0	1	1	1	1
1	*0*	1	1	1
1	1	1	0	0

XOR

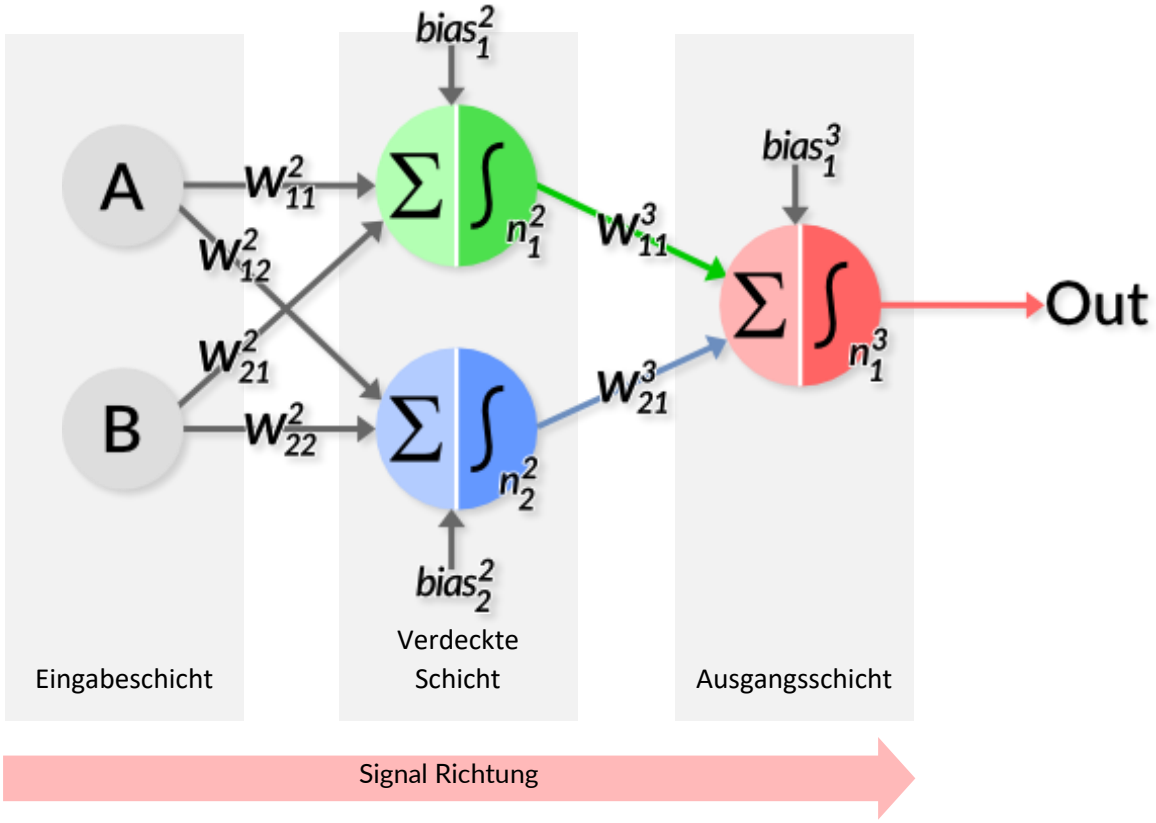
Aufgabe

Welche Art von Knoten muss hinzugefügt werden, um die gewünschte Kombination zu erreichen?

Lösung: Es ist der AND-Knoten

Teil III: Ein sehr einfaches tiefes neuronales Netz

Der Trick besteht nun darin, mehrere Neuronen zu einem Netzwerk zu verbinden.



Allgemeine Struktur:

Die Struktur ist in drei verschiedene Arten von Schichten unterteilt. Das oben dargestellte Netz ist in diese Schichten unterteilt:

- Eine Eingabeschicht mit zwei Eingabe-Neuronen,
- Eine versteckte Schicht mit zwei Neuronen
- Eine Ausgabeschicht mit einem einzigen Neuron.

Eingabeschicht

Nach der Nomenklatur zählt man die Eingänge (hier A und B) als unabhängige Neuronen, so genannte Eingangsneuronen. Das erscheint zunächst etwas seltsam, denn diese Neuronen führen keine Berechnungen durch. Sie dienen lediglich als Signalquelle.

Verdeckte Schicht

Die verborgene Schicht enthält Neuronen, die von außen nicht zu sehen sind - wenn das Netzwerk ein real existierendes Ding wäre. Im Prinzip werden alle Schichten zwischen der Eingabe- und der Ausgabeschicht als versteckte Schichten bezeichnet.

Signalfluss

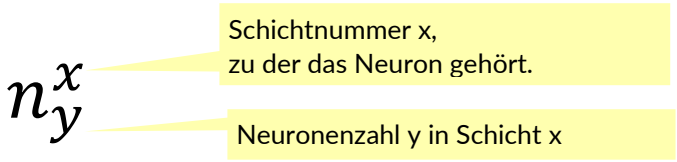
Der Signalfluss verläuft durch alle drei Schichten:

- Die erste Schicht besteht aus Eingangsneuronen, die die Eingangssignale liefern.
- In der 2. Schicht befinden sich zwei Neuronen, die von Eingangsneuronen gespeist werden und die Signale verarbeiten.
- In der 3. Schicht befindet sich das einzelne Ausgangsneuron, das das Berechnungsergebnis ausgibt.

Nomenklatur der Neuronen

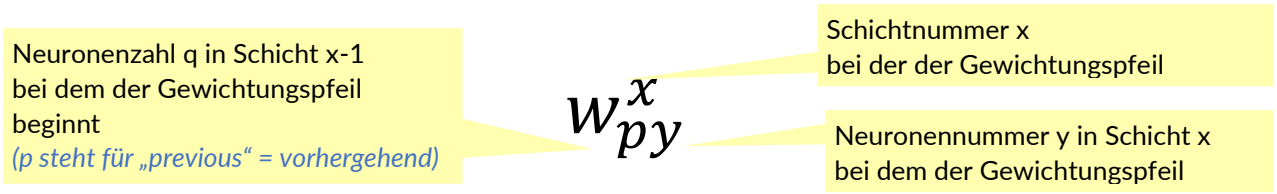
Die Neuronen werden mit zwei Bezeichnungen versehen:

- Ein hochgestelltes x, das die Schicht angibt, zu der das Neuron gehört**.**
- Ein tiefgestelltes Symbol y, dies ist eine fortlaufende Nummerierung von oben nach unten für jedes Neuron der Schicht.



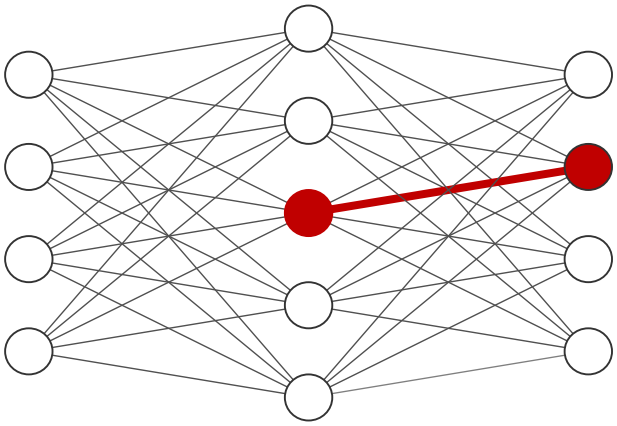
Nomenklatur der Gewichte

Auch die Gewichte der Verbindungspfeile erhalten eine eigene Nomenklatur:



- Das hochgestellte Symbol "x" ist die Ebenenbezeichnung und bedeutet "Verbindung zur Ebene x",
- Die beiden tiefgestellten Buchstaben bezeichnen die Neuronen py,
    - das erste Symbol p steht für die Nummer des Ursprungsneurons, das sich in der **vorherigen** Schicht x-1 befindet,
    - das zweite Symbol y für das Zielneuron in Schicht n.

Beispiel:



Eingabeschicht $\in \mathbb{R}^4$ Verdeckte Schicht $\in \mathbb{R}^5$ Ausgangsschicht $\in \mathbb{R}4$

Startneuron:

$$n_3^2$$

Gewichtsverbindung:

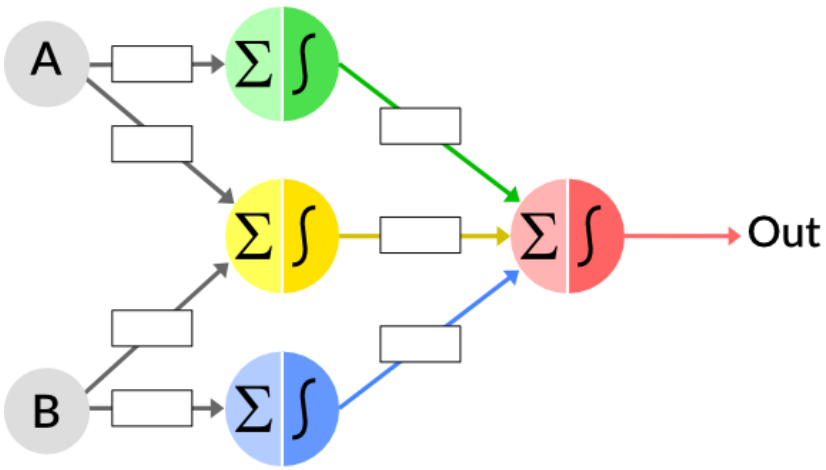
$$w_{32}^3$$

Endneuron:

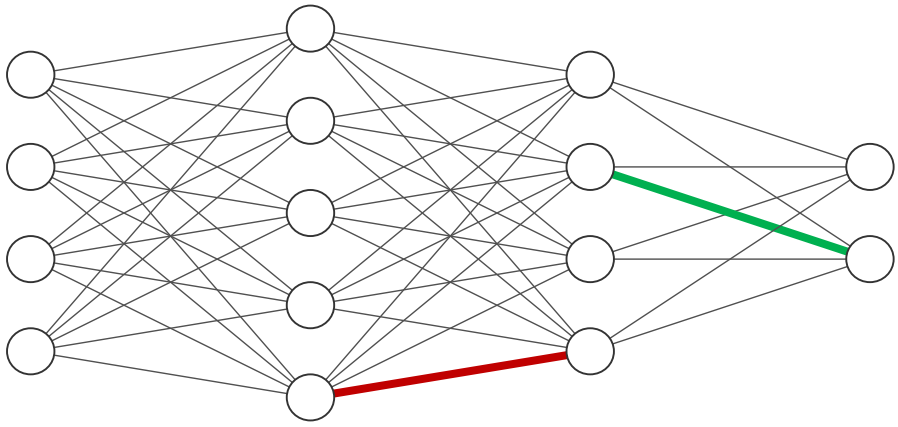
$$n_2^3$$

Aufgabe 15: Terminologie der Gewichtungsfaktoren

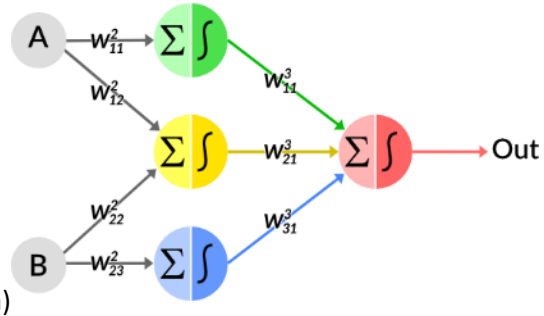
a) Können Sie erklären, wie die Nomenklatur der Vorurteile aufgebaut ist?
b) Versuchen Sie, die richtige Bezeichnung für das gelbe Neuron in der Mitte zu finden.
c) Versuchen Sie, die richtigen Bezeichnungen für die fehlenden Gewichte zu finden:



d) Beschreiben Sie das Netz: wie viele Schichten, wie viele verborgene Schichten, wie viele Knoten und wie viele Verbindungen gibt es in diesem Netz?
e) die zweifarbigen Gewichte nach der korrekten Nomenklatur zu formulieren



Lösung:

a) 	n_3^2 4 Schichten, 2 versteckte Schichten, 15 Knotenpunkte 48 Verbindungen (Gewichte)	Rote Verbindung: w_{54}^3 Grüne Verbindung: w_{22}^4

Teil IV: Das Spiel "Neuronales Netz", unplugged-Aktivität

Diese Aktivität gehört ursprünglich zu "TechGirlz":
https://docs.google.com/document/d/1_uGzFd-iHBgCui1NMSwtcQJbrICKpiXZGp55tZv3ILk/edit

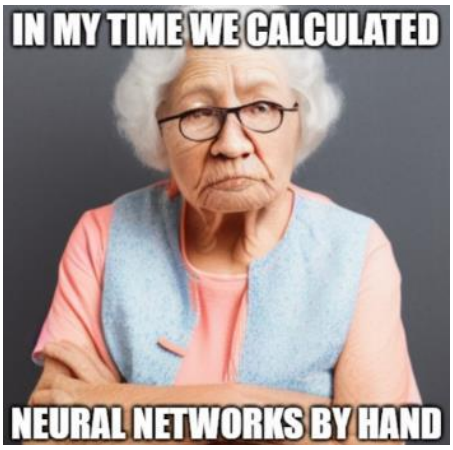


Abb. 12 Bild [Gemeinfrei] erzeugt mit Stable Diffusion; Prompt „old woman with white hair and glasses, hands crossed, looks angry “ von Jörg [CC BY-SA 4.0 International]

Lektion 6: Berechnung des Vorwärtsdurchgangs eines Netzes

Was Schüler lernen sollten

Nachdem nun die mehrschichtigen neuronalen Netze im Prinzip eingeführt sind, können die ersten Berechnungen erfolgen. Diese einfachen Netze, die das XOR-Problem lösen, lassen sich noch gut von Hand mit einfachen Gewichten berechnen.

Durch die Berechnungen bekommen die Schüler ein Gefühl dafür, wie ein Netzwerk funktioniert: Die Eingabe wird von links nach rechts weitergegeben, indem jedes Neuron seinen Anteil hinzufügt. Die Ausgabe des einen Neurons wird zur Eingabe des nächsten Neurons. Dabei werden die Eingaben und Ausgaben vermischt. Dies zu erkennen ist das Ziel dieser Lektion.

Mögliche Aktivitäten der Schüler

Es gibt zwei Standardlösungen für die Architektur eines neuronalen XOR-Netzes. Beide Netze werden von den Schülern zum einen mit einfachen Zahlenwerten berechnet und somit verstanden.

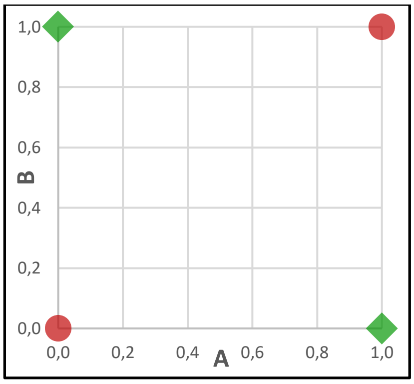
Durch die Logik der Zahlenwerte wird die Aufgabenverteilung der einzelnen Neuronen erkennbar: Die Schülerinnen und Schüler erkennen aktivierende, hemmende, kombinierende Neuronen, Neuronen, die Funktionskomponenten liefern und solche, die diese einzelnen Funktionskomponenten miteinander verrechnen.

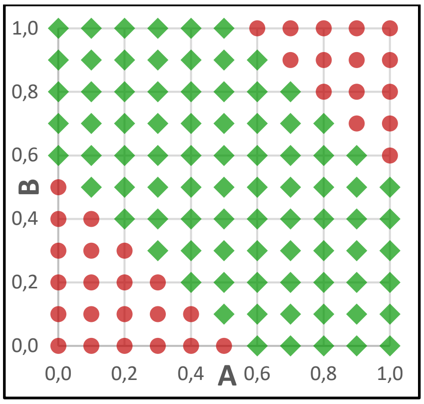
Diese Funktionsprinzipien sind die gleichen wie bei großen Netzen, wie sie beispielsweise für die Bilderkennung oder die Sprachsynthese (NLP) verwendet werden. Die Lehrkraft sollte speziell auf diese Funktionsprinzipien eingehen.

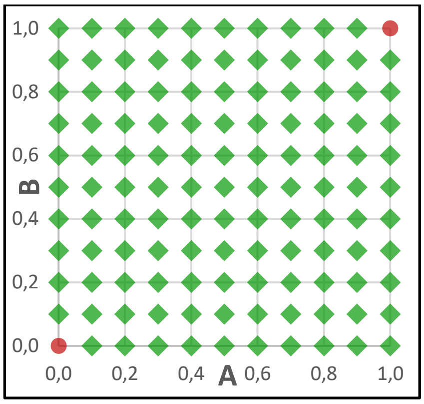
Zwei Simulationen, die das Klassifizierungsverhalten der beiden XOR-Netze abbilden, ermöglichen es den Schülern, ein Gefühl für das Verhalten und die Leistung der Netze zu entwickeln, indem sie experimentieren und herumspielen.

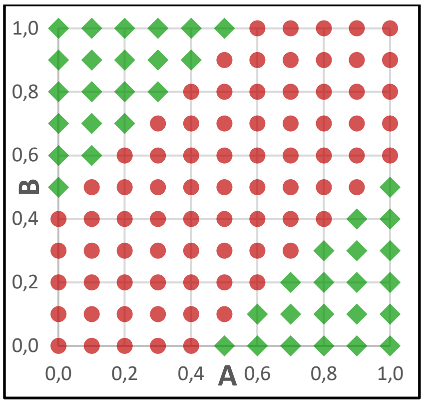
Teil I: Das Problem verstehen

Mögliche Arten von Lösungen für das XOR-Problem

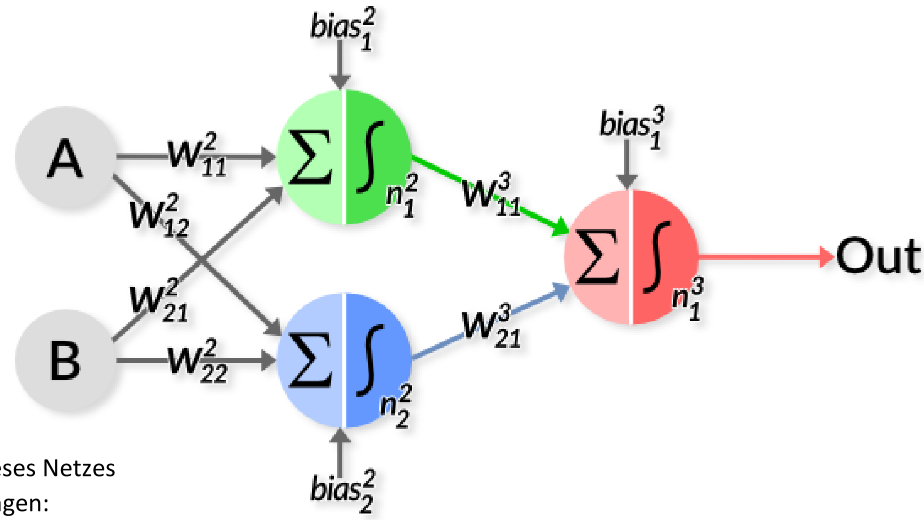








Teil II: Berechnung des einfachsten neuronalen Netzes für XOR von Hand



Aufgabe 16:

Berechnen wir nun das Ausgangssignal dieses Netzes von Anfang an, ausgehend von den Eingängen:

Die Gewichte sind:

Grünes Neuron n_1^2			Blaues Neuron n_1^2			Rotes Neuron n_1^3		
w_{11}^2	w_{21}^2	$bias_1^2$	w_{12}^2	w_{22}^2	$bias_2^2$	w_{11}^3	w_{31}^3	$bias_1^3$
1	1	0	-1	-1	2	1	1	-1

Neuron 1, n_1^2:

A	$\cdot\, w_{11}^2$		B	$\cdot\, w_{21}^2$		$bias_1^2$		Sum_1^2	Out_1^2
0			0						
0			1			0			
1			0						
1			1						

Neuron 2, n_2^2:

A	$\cdot w_{12}^2$		B	$\cdot w_{22}^2$		$bias_2^2$		Sum_2^2	$\boldsymbol{Out_2^2}$
0			0			1			
0			1						
1			0						
1			1						

Neuron 3, n_1^3:

$\boldsymbol{Out_1^2}$	$\cdot w_{11}^3$		$\boldsymbol{Out_2^2}$	$\cdot w_{21}^3$		$bias_1^3$		Sum_1^3	$\boldsymbol{Out_1^3}$
						-1			

Aufgabe 17)

Welche Arten von logischen Funktionen haben wir in allen drei Neuronen?

Lösung 16)

A	$\cdot w_{11}^2$		B	$\cdot w_{21}^2$		$bias_1^2$		Sum_1^2	$\boldsymbol{Out_1^2}$
0	*0*		0	*0*		0		0	0
0	*0*		1	*1*				1	1
1	*1*		0	*0*				1	1
1	*1*		1	*1*				1	1
A	$\cdot w_{12}^2$		**B**	$\cdot w_{22}^2$		$bias_2^2$		Sum_2^2	$\boldsymbol{Out_2^2}$
0	*0*		0	*0*		2		2	1
0	*0*		1	*-1*				1	1
1	*-1*		0	*0*				1	1
1	*-1*		1	*-1*				0	0
$\boldsymbol{Out_1^2}$	$\cdot w_{11}^3$		$\boldsymbol{Out_2^2}$	$\cdot w_{21}^3$		$bias_1^3$		Sum_1^3	$\boldsymbol{Out_1^3}$
0	*0*		1	*1*		-1		0	0
1	*1*		1	*1*				1	1
1	*1*		1	*1*				1	1
1	*1*		0	*0*				0	0

Lösung 17)

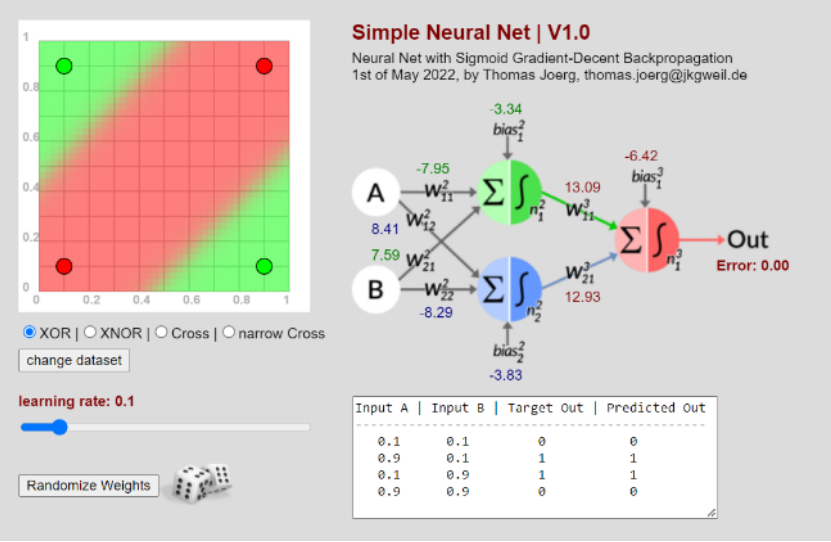
- Neuron 1 der beiden versteckten Neuronen stellt eine ODER-Schaltung dar und sorgt dafür, dass in den Fällen (1,0), (0,1) und (1,1) zuerst das Ausgangsneuron aktiviert wird. Es hat eine aktivierende Wirkung.

- Neuron 2 der beiden versteckten Neuronen stellt eine NAND-Schaltung dar und sorgt dafür, dass im Fall (1,1) das Ausgangsneuron wieder ausgeschaltet wird. Es hat eine hemmende Wirkung.

- Neuron 3, das Ausgangsneuron, stellt eine UND-Schaltung dar und kombiniert die beiden Ausgänge der versteckten Neuronen.

Teil II: Spielereien mit einer Perceptron-Simulation

https://iludis.de/XOR_Perceptron/index.html



Aufgaben:

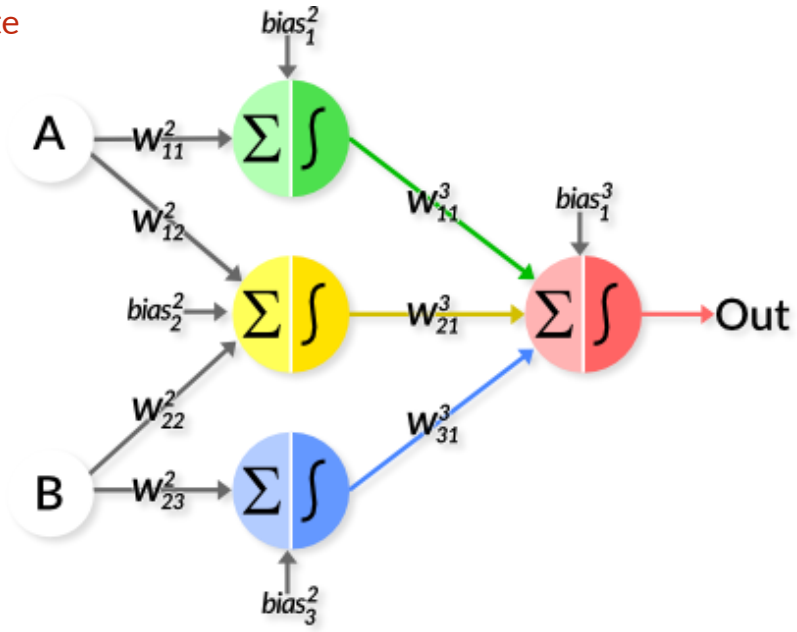
Diese Simulation zeigt nicht nur den Vorwärtsdurchlauf des neuronalen Netzes, sondern auch den Lernprozess. Darauf wird später noch eingegangen.

a) Wechseln Sie zwischen den Datensätzen XOR, XNOR, Cross und narrow Cross. Wie verändern sich die Gewichte?

b) Kommt die Berechnung immer zu dem Ergebnis? Was passiert, wenn Sie die Gewichte über die Schaltfläche erneut randomisieren?

c) Wenn man das Netz reinitialisiert hat, kann es den Datensatz lernen, aber manchmal mit anderen Gewichten. Können Sie interpretieren, welche logischen Funktionen die einzelnen Neuronen darstellen?

Teil III: Berechnung der zweiten Variante Neuronales Netz für XOR von Hand



Aufgabe 18)

Berechnen wir das Ausgangssignal dieses Netzes von Anfang an, ausgehend von den Eingängen:

Die Gewichte sind:

Grünes Neuron n_1^2		Gelbes Neuron n_2^2			Blaues Neuron n_3^2		Rotes Neuron n_1^3			
w_{11}^2	$bias_1^2$	w_{12}^2	w_{22}^2	$bias_2^2$	w_{23}^2	$bias_3^2$	w_{11}^3	w_{21}^3	w_{31}^3	$bias_1^3$
1	0	1	1	0	1	0	-1	2	-1	0

Neuron 1, n_1^2:

A	$\cdot w_{11}^2$		$bias_1^2$		Sum_1^2	Out_1^2
0			1			
1						

Neuron 2, n_2^2:

A	$\cdot w_{12}^2$		B	$\cdot w_{22}^2$		$bias_2^2$		Sum_2^2	Out_2^2
0			0			2			
0			1						
1			0						
1			1						

Neuron 3, n_3^2:

B	$\cdot w_{23}^2$		$bias_3^2$		Sum_3^2	Out_3^2
0			1			
1						

Neuron 3, n_1^3:

Out_1^2	$\cdot w_{11}^3$		Out_2^2	$\cdot w_{21}^3$		Out_3^2	$\cdot w_{31}^3$		$bias_1^3$		Sum_1^3	Out_1^3
									0			

Aufgabe 19)

Welche Arten von logischen Funktionen haben wir in allen vier Neuronen?

Lösung 18:

A	$\cdot w_{11}^2$		$bias_1^2$		Sum_1^2	$\mathbf{Out_1^2}$
0	0		0		0	0
1	1				1	1

A	$\cdot w_{12}^2$		B	$\cdot w_{22}^2$		$bias_2^2$		Sum_2^2	$\mathbf{Out_2^2}$
0	0		0	0		0		0	0
0	0		1	1				1	1
1	1		0	0				1	1
1	1		1	1				2	1

B	$\cdot w_{23}^2$		$bias_3^2$		Sum_3^2	$\mathbf{Out_3^2}$
0	0		0		0	0
1	1				1	1

$\mathbf{Out_1^2}$	$\cdot w_{11}^3$		$\mathbf{Out_2^2}$	$\cdot w_{21}^3$		$\mathbf{Out_3^2}$	$\cdot w_{31}^3$		$bias_1^3$		Sum_1^3	$\mathbf{Out_1^3}$
0	0		0	0		0	0		0		0	0
0	0		1	-1		1	2				1	1
1	-1		0	0		1	2				1	1
1	-1		1	-1		1	2				0	0

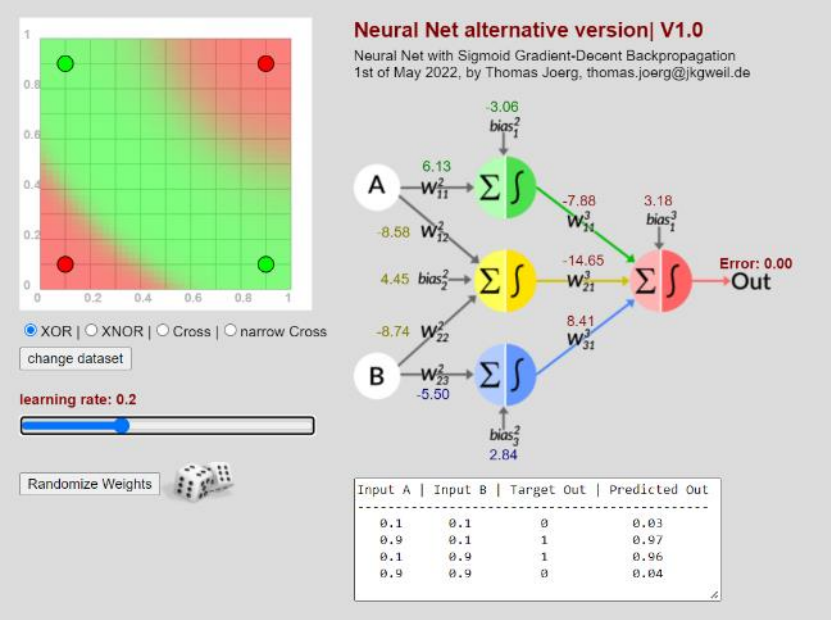
Lösung 19

Neuron 1 und Neuron 3 leiten das Eingangssignal unverändert weiter. Bevor sie jedoch in das Ausgangsneuron in Schicht 3 gelangen, werden diese Signale durch negative Gewichtung invertiert. Dadurch haben sie eine hemmende Wirkung. Das mittlere Neuron 2 wirkt hier gleichzeitig verstärkend und aktivierend: Zunächst wird als ODER-Verknüpfung das Ausgangssignal mit der Gewichtung "2" verstärkt. Das Ausgangsneuron in Schicht 3 wird dadurch aktiviert.

Wirkt Neuron 1 zusammen mit Neuron 2 oder Neuron 3 mit Neuron 2, überwiegt die aktivierende Wirkung von Neuron 2 und das Ausgangsneuron feuert, wirken Neuron 1 und Neuron 2 dagegen gleichzeitig hemmend zusammen, überwiegt ihre Wirkung die von Neuron 3. Infolgedessen wird das Ausgangsneuron stummgeschaltet.

Teil VI: Spielen mit einer Perceptron-Simulation

https://iludis.de/XOR_Perceptron2/index.html



Aufgaben:

Diese Simulation zeigt nicht nur den Vorwärtsdurchlauf des neuronalen Netzes, sondern auch den Lernprozess. Darauf wird später noch eingegangen.

a) Wechseln Sie zwischen den Datensätzen XOR, XNOR, Cross und narrow Cross. Wie verändern sich die Gewichte?

b) Kommt die Berechnung immer zu dem Ergebnis? Was passiert, wenn Sie die Gewichte über die Schaltfläche erneut randomisieren?

c) Wenn man das Netz reinitialisiert hat, kann es den Datensatz lernen, aber manchmal mit anderen Gewichten. Können Sie interpretieren, welche logischen Funktionen die einzelnen Neuronen darstellen?

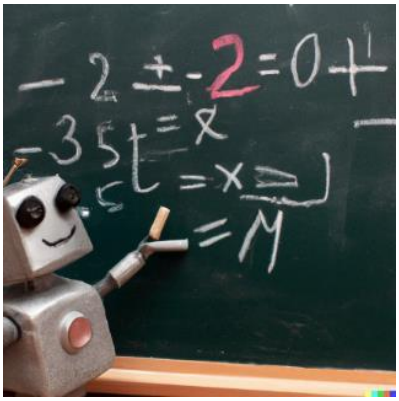


Abb. 13 Bild [Gemeinfrei] erzeugt mit DALL-E; Prompt „a photo of a cute and smiling robot who is calculating on a blackboard“ von Jörg [CC BY-SA 4.0 International]

Lektion 7a: Einführung in die Backpropagation für Lehrer

Mathematische Herleitung für Lehrer, ***bitte nicht für Schüler!***

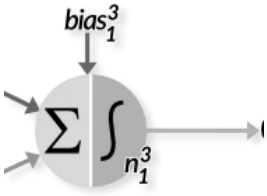
Wer das qualitative Prinzip der Backpropagation unterrichtet, sollte einen gewissen Wissensvorsprung gegenüber den Schülern haben. Deshalb ist es notwendig, sich mit den mathematischen Grundlagen der Backpropagation zu beschäftigen. Die Gleichungen, die auch in Programmiersprachen oder Excel-Sheets übernommen werden können, sollen hier wiederholt und zusammengefasst werden. Die Algorithmen sind implementiert und getestet; die damit programmierten neuronalen Netze konvergieren ;-)

alle Ableitungen sind für die sigmoidale Aktivierungsfunktion berechnet!

Erinnern Sie sich an die Begriff Definitionen in Lektion 5:

n_y^x	Der Begriff "n" steht für "Neuron" in Schicht x an Stelle y von oben nach unten
w_{py}^x	Der Begriff "w" steht für das "Gewicht" eines Neurons in der Schicht "x", das die Neuronen zwischen Ort "p" und Ort "y" verbindet.
$bias_y^x$	Der Begriff "Bias" steht für sich selbst und wird mit dem Neuron n_y^x verbunden

Im Inneren des Neurons n_y^x finden wir die folgenden zwei Komponenten:



sum_y^x	Der Begriff "Sum" steht für "Summe aller Eingaben des Neurons", die durch Addition aller gewichteten Eingaben innerhalb des Neurons berechnet wird
out_y^x	Der Begriff "out" steht für die "Ausgabe des Neurons", die durch die Aktivierungsfunktion innerhalb des Neurons berechnet wird n_y^x

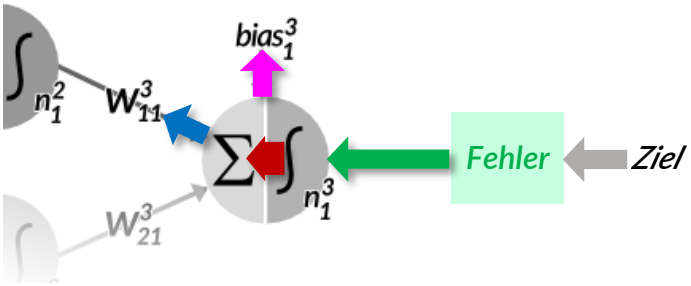
Außerhalb des Neurons n_y^x finden wir die folgenden zwei Komponenten:

$Error_{total}$	Sie wird als Differenz zwischen der Ausgabe des neuronalen Netzes und dem wahren Wert (dem Zielwert) berechnet. Er ist ein Maß dafür, wie richtig oder falsch ein NN voraussagen kann.
$target_y$	Der Zielwert ist der tatsächliche, reale Wert eines Datensatzes, mit dem der von einem neuronalen Netz vorhergesagte Wert verglichen wird.

Alle Komponenten werden berechnet:

$sum_y^x = w_{1y}^x \cdot input_1 + w_{2y}^x \cdot input_2 + \cdots + bias_y^x$	Die Summe der Neuronen n_y^x wird durch Addition aller Eingaben multipliziert mit ihren Gewichten berechnet.
$out_y^x = ActivationFunction(sum_y^x)$	Die Aktivierungsfunktion nimmt als Variable die bereits berechnete Summe sum_y^xund erzeugt als Funktionswert die Ausgabe out_y^x.
$error_{total} = \frac{1}{2} \sum_{\substack{all\ output \\ neurons\ y}} (target_y - out_y^x)^2$	Handelt es sich um ein Ausgangsneuron, trägt es zum Fehlerwert bei. Alle Fehler aller Ausgangsneuronen werden addiert. Dies ergibt den Gesamtfehler.

Berechnen Sie den Gradienten für die beiden Komponenten w_{11}^3 und $bias_1^3$in der Ausgabeschicht



Für das Ausgangsneuron finden wir (allgemein formuliert):	Für unser Beispiel eines neuronalen Netzes finden wir für n_1^3:
$\frac{\partial error_{total}}{\partial w_{py}^x} = \frac{\partial error_{total}}{\partial out_y^x} \cdot \frac{\partial out_y^x}{\partial sum_y^x} \cdot \frac{\partial sum_y^x}{\partial w_{py}^x}$	$\frac{\partial error}{\partial w_{11}^3} = \frac{\partial error}{\partial out_1^3} \cdot \frac{\partial out_1^3}{\partial sum_1^3} \cdot \frac{\partial sum_1^3}{\partial w_{11}^3}$
$\frac{\partial error_{total}}{\partial bias_y^x} = \frac{\partial error_{total}}{\partial out_y^x} \cdot \frac{\partial out_y^x}{\partial sum_y^x} \cdot \frac{\partial sum_y^x}{\partial bias_y^x}$	$\frac{\partial error}{\partial bias_1^3} = \frac{\partial error}{\partial out_1^3} \cdot \frac{\partial out_1^3}{\partial sum_1^3} \cdot \frac{\partial sum_1^3}{\partial bias_1^3}$

A) Berechnen Sie die Ableitung des ersten Terms *für die Beziehung zwischen Ausgang und Fehler*:

$$\frac{\partial error}{\partial out_1^3} = \frac{\partial \frac{1}{2}\sum \left(target_1 - out_1^3\right)^2}{\partial out_1^3} = \frac{\partial \frac{1}{2}\left(target_1 - out_1^3\right)^2}{\partial out_1^3} = -\left(target_1 - out_1^3\right) = \left(out_1^3 - target_1\right)$$

B) Berechnen Sie die Ableitung des zweiten Terms *für die Beziehung zwischen der Summe und der sigmoiden Aktivierungsfunktion*:

$$\frac{\partial out_1^3}{\partial sum_1^3} = \frac{\partial\left[\frac{1}{1+e^{-sum_1^3}}\right]}{\partial sum_1^3} = out_1^3 \cdot \left(1 - out_1^3\right)$$

C) Berechnen Sie die Ableitung des dritten Terms *für die Beziehung zwischen Gewicht und Summe*:

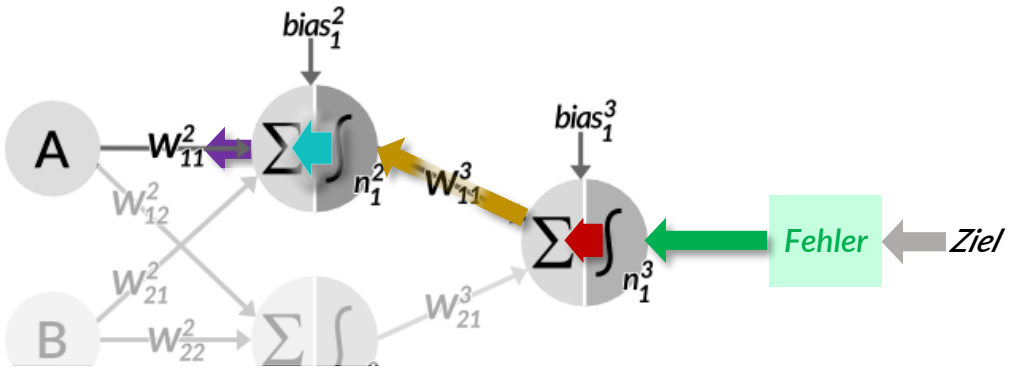
$$\frac{\partial sum_1^3}{\partial w_{11}^3} = \frac{\partial\{w_{11}^3 \cdot out_1^2 + w_{21}^3 \cdot out_2^2 + bias_1^3\}}{\partial w_{11}^3} = out_1^2 \qquad since\ out_1^2\ is\ the\ neuron_1^3\ input$$

D) Alles zusammengenommen:

$$\frac{\partial error}{\partial w_{11}^3} = \left(out_1^3 - target_1\right) \cdot out_1^3 \cdot \left(1 - out_1^3\right) \cdot out_1^2 \qquad since\ out_1^2\ is\ the\ neuron_1^3\ input$$

$$\frac{\partial error}{\partial bias_1^3} = \left(out_1^3 - target_1\right) \cdot out_1^3 \cdot \left(1 - out_1^3\right) \cdot 1 \qquad since\ \frac{\partial sum_1^3}{\partial bias_1^3}\ is\ 1$$

Berechnen Sie den Gradienten für die beiden Komponenten w_{11}^2 und $bias_1^2$in der verborgenen Schicht



Für ein verstecktes Neuron finden wir (allgemein formuliert):

$$\frac{\partial error_{total}}{\partial w_{pz}^{hidden}} = \left[\frac{\partial error_{total}}{\partial out_y^{outlayer}} \cdot \frac{\partial out_y^{outlayer}}{\partial sum_y^{outlayer}}\right] \cdot \frac{\partial sum_y^{outlayer}}{\partial out_z^{hidden}} \cdot \frac{\partial out_z^{hidden}}{\partial sum_z^{hidden}} \cdot \frac{\partial sum_z^{hidden}}{\partial w_{pz}^{hidden}}$$

Für unser Beispiel eines neuronalen Netzes finden wir (beispielhaft):

$$\frac{\partial error}{\partial w_{11}^2} = \left[\frac{\partial error}{\partial out_1^3} \cdot \frac{\partial out_1^3}{\partial sum_1^3}\right] \cdot \frac{\partial sum_1^3}{\partial out_1^2} \cdot \frac{\partial out_1^2}{\partial sum_1^2} \cdot \frac{\partial sum_1^2}{\partial w_{11}^2}$$

A) Berechnen Sie den ersten Term, der aus der Ausgabeschicht bekannt ist:

$$\left[\frac{\partial error}{\partial out_1^3} \cdot \frac{\partial out_1^3}{\partial sum_1^3}\right] = \left[\left(out_1^3 - target_1\right) \cdot out_1^3 \cdot \left(1 - out_1^3\right)\right]$$ this can be reviewed on the previous page

B) Berechnen Sie die Gradientenverbindung zwischen dem verborgenen und dem Ausgangsknoten:

$$\frac{\partial sum_1^3}{\partial out_1^2} = \frac{\partial\{w_{11}^3 \cdot out_1^2 + w_{21}^3 \cdot out_2^2 + bias_1^3\}}{\partial out_1^2} = w_{11}^3$$

C) Die Berechnung der Ableitung des dritten Terms ist B) auf der vorherigen Seite sehr ähnlich:

$$\frac{\partial out_1^2}{\partial sum_1^2} = \frac{\partial\left[\frac{1}{1+e^{-sum_1^2}}\right]}{\partial sum_1^2} = out_1^2 \cdot \left(1 - out_1^2\right)$$

D) Berechnen Sie die Ableitung des vierten Terms:

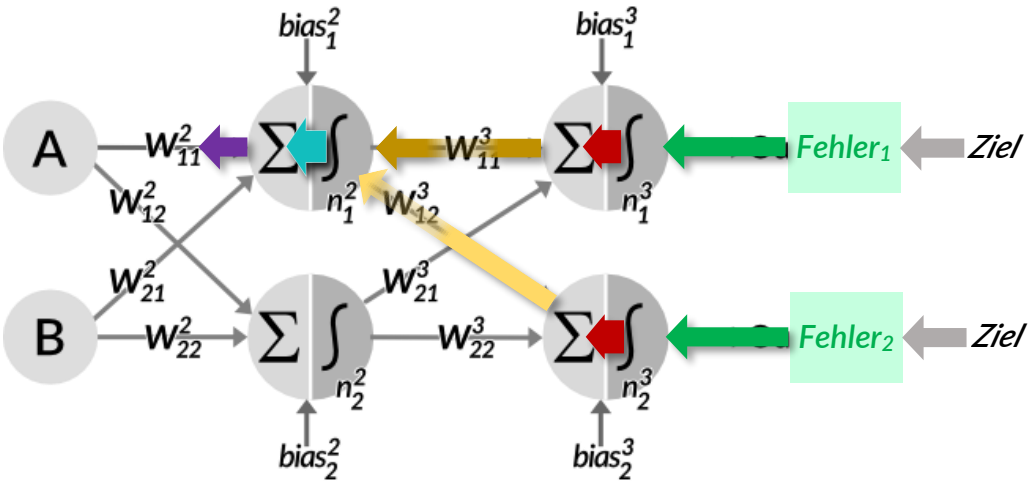
$$\frac{\partial sum_1^2}{\partial w_{11}^2} = \frac{\partial\{w_{11}^2 \cdot A + w_{21}^2 \cdot B + bias_1^2\}}{\partial w_{11}^2} = A$$

E) Das Ganze zusammenfügen:

$$\frac{\partial error}{\partial w_{11}^2} = \left[\left(out_1^3 - target_1\right) \cdot out_1^3 \cdot \left(1 - out_1^3\right)\right] \cdot w_{11}^3 \cdot out_1^2 \cdot \left(1 - out_1^2\right) \cdot A$$

$$\frac{\partial error}{\partial bias_1^2} = \left[\left(out_1^3 - target_1\right) \cdot out_1^3 \cdot \left(1 - out_1^3\right)\right] \cdot w_{11}^3 \cdot out_1^2 \cdot \left(1 - out_1^2\right) \cdot 1 \quad since \frac{\partial sum_1^2}{\partial bias_1^2} \; is \; 1$$

Berechnen Sie den Gradienten für die beiden Komponenten w_{11}^2 und $bias_1^2$ in der verborgenen Schicht



Für ein verstecktes Neuron finden wir (allgemein formuliert):

$$\frac{\partial \text{error}_{total}}{\partial w_{pz}^{\text{hidden}}} = \sum_{\substack{\textit{All output} \\ \textit{neurons y}}} \left[\frac{\partial \text{error}_{\text{total}}}{\partial \text{out}_y^{\text{outlayer}}} \cdot \frac{\partial \text{out}_y^{\text{outlayer}}}{\partial \text{sum}_y^{\text{outlayer}}} \right] \cdot \frac{\partial \text{sum}_y^{\text{outlayer}}}{\partial \text{out}_z^{\text{hidden}}} \cdot \frac{\partial \text{out}_z^{\text{hidden}}}{\partial \text{sum}_z^{\text{hidden}}} \cdot \frac{\partial \text{sum}_z^{\text{hidden}}}{\partial w_{pz}^{\text{hidden}}}$$

Für unser Beispiel eines neuronalen Netzes finden wir (beispielhaft):

$$\frac{\partial \text{error}}{\partial w_{11}^2} = \left[\frac{\partial \text{error}_1}{\partial \text{out}_1^3} \cdot \frac{\partial \text{out}_1^3}{\partial \text{sum}_1^3} + \frac{\partial \text{error}_2}{\partial \text{out}_2^3} \cdot \frac{\partial \text{out}_2^3}{\partial \text{sum}_2^3} \right] \cdot \frac{\partial \text{sum}_1^3}{\partial \text{out}_1^2} \cdot \frac{\partial \text{out}_1^2}{\partial \text{sum}_1^2} \cdot \frac{\partial \text{sum}_1^2}{\partial w_{11}^2}$$

Zusammengefasst (aus Gründen der Übersichtlichkeit wird der Begriff "Ziel" mit einem einzigen "t" abgekürzt):

$$\frac{\partial \text{error}}{\partial w_{11}^2} = \left[(\text{out}_1^3 - t_1) \cdot \text{out}_1^3 \cdot (1 - \text{out}_1^3) + (\text{out}_2^3 - t_2) \cdot \text{out}_2^3 \cdot (1 - \text{out}_2^3) \right] \cdot w_{11}^3 \cdot \text{out}_1^2 \cdot (1 - \text{out}_1^2) \cdot A$$

$$\frac{\partial \text{error}}{\partial \text{bias}_1^2} = \left[(\text{out}_1^3 - t_1) \cdot \text{out}_1^3 \cdot (1 - \text{out}_1^3) + (\text{out}_2^3 - t_2) \cdot \text{out}_2^3 \cdot (1 - \text{out}_2^3) \right] \cdot w_{11}^3 \cdot \text{out}_1^2 \cdot (1 - \text{out}_1^2) \cdot 1$$

Nützliche Abkürzungen:

Von hier an wird die Formel sehr verwirrend. Deshalb wird eine praktische Abkürzung eingeführt: Das sogenannte "Knoten-Delta". Nehmen wir unser Netz mit Schicht 3 als Ausgabeschicht und Schicht 2 als verborgene Schicht an:

Für ein Ausgangsneuron in Schicht 3 ist sie definiert als:

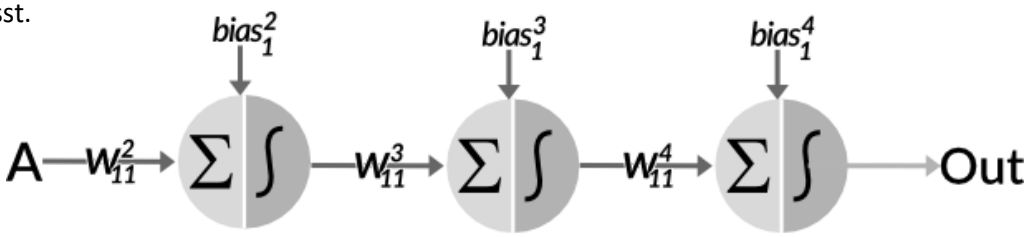
$$\text{node}\Delta_1^3 = (\text{out}_1^3 - \text{target}_1) \cdot \text{out}_1^3 \cdot (1 - \text{out}_1^3)$$

Für ein verstecktes Neuron in Schicht 2 ist sie definiert als:

$$\text{node}\Delta_1^2 = \sum_y [\text{node}\Delta_y^3] \cdot w_{11}^3 \cdot \text{out}_1^2 \cdot (1 - \text{out}_1^2)$$

die Gemeinsamkeiten und Muster erkennen: allgemeiner Fall

der schritt von der mathematischen Darstellung zur Implementierung ist oft schwer zu erkennen. deshalb soll hier eine Art verallgemeinerte konkrete Darstellung abgeleitet werden, die sich leichter z.B. auf JavaScript oder Python übertragen lässt.



System von Gleichungen für partielle Ableitungen

$$\frac{\partial \text{error}}{\partial w_{11}^4} = \frac{\partial \text{error}}{\partial \text{out}_1^4} \cdot \frac{\partial \text{out}_1^4}{\partial \text{sum}_1^4} \cdot \frac{\partial sum_1^4}{\partial w_{11}^4}$$

$$\frac{\partial \text{error}}{\partial w_{11}^3} = \frac{\partial \text{error}}{\partial \text{out}_1^4} \cdot \frac{\partial \text{out}_1^4}{\partial \text{sum}_1^4} \cdot \frac{\partial \text{sum}_1^4}{\partial \text{out}_1^3} \cdot \frac{\partial \text{out}_1^3}{\partial \text{sum}_1^3} \cdot \frac{\partial \text{sum}_1^3}{\partial w_{11}^3}$$

$$\frac{\partial \text{error}}{\partial w_{11}^2} = \frac{\partial \text{error}}{\partial \text{out}_1^4} \cdot \frac{\partial \text{out}_1^4}{\partial \text{sum}_1^4} \cdot \frac{\partial \text{sum}_1^4}{\partial \text{out}_1^3} \cdot \frac{\partial \text{out}_1^3}{\partial \text{sum}_1^3} \cdot \frac{\partial \text{sum}_1^3}{\partial \text{out}_1^2} \cdot \frac{\partial \text{out}_1^2}{\partial \text{sum}_1^2} \cdot \frac{\partial \text{sum}_1^2}{\partial w_{11}^2}$$

Konkret formuliert für die sigmoid-Funktion: Gewichte

$$\frac{\partial \text{error}}{\partial w_{11}^4} = (o_1^4 - t_1) \cdot o_1^4 \cdot (1 - o_1^4) \cdot o_1^3$$

$$\frac{\partial \text{error}}{\partial w_{11}^3} = (o_1^4 - t_1) \cdot o_1^4 \cdot (1 - o_1^4) \cdot w_{11}^4 \cdot o_1^3 \cdot (1 - o_1^3) \cdot o_1^2$$

$$\frac{\partial \text{error}}{\partial w_{11}^2} = (o_1^4 - t_1) \cdot o_1^4 \cdot (1 - o_1^4) \cdot w_{11}^4 \cdot o_1^3 \cdot (1 - o_1^3) \cdot w_{11}^3 \cdot o_1^3 \cdot (1 - o_1^3) \cdot A$$

Konkret formuliert für die sigmoid-Funktion: Verzerrungen

$$\frac{\partial \text{error}}{\partial \text{bias}_1^4} = (o_1^4 - t_1) \cdot o_1^4 \cdot (1 - o_1^4)$$

$$\frac{\partial \text{error}}{\partial \text{bias}_1^3} = (o_1^4 - t_1) \cdot o_1^4 \cdot (1 - o_1^4) \cdot w_{11}^4 \cdot o_1^3 \cdot (1 - o_1^3)$$

$$\frac{\partial \text{error}}{\partial \text{bias}_1^2} = (o_1^4 - t_1) \cdot o_1^4 \cdot (1 - o_1^4) \cdot w_{11}^4 \cdot o_1^3 \cdot (1 - o_1^3) \cdot w_{11}^3 \cdot o_1^3 \cdot (1 - o_1^3)$$

Korrekturterm für die Gewichte und Verzerrungen:

$$w_{11}^{4,\text{after corr}} = w_{11}^{4,\text{before corr}} - \text{learnrate} \cdot \frac{\partial \text{error}}{\partial w_{11}^4} \qquad bias_1^{4,\text{after corr}} = \text{bias}_1^{4,\text{before corr}} - \text{learnrate} \cdot \frac{\partial \text{error}}{\partial \text{bias}_1^4}$$

$$w_{11}^{3,\text{after corr}} = w_{11}^{3,\text{before corr}} - \text{learnrate} \cdot \frac{\partial \text{error}}{\partial w_{11}^3} \qquad bias_1^{3,\text{after corr}} = \text{bias}_1^{3,\text{before corr}} - \text{learnrate} \cdot \frac{\partial \text{error}}{\partial \text{bias}_1^3}$$

$$w_{11}^{2,\text{after corr}} = w_{11}^{2,\text{before corr}} - \text{learnrate} \cdot \frac{\partial \text{error}}{\partial w_{11}^2} \qquad bias_1^{2,\text{after corr}} = \text{bias}_1^{2,\text{before corr}} - \text{learnrate} \cdot \frac{\partial \text{error}}{\partial \text{bias}_1^2}$$

Lektion 7b: Einführung in die Backpropagation für Schüler

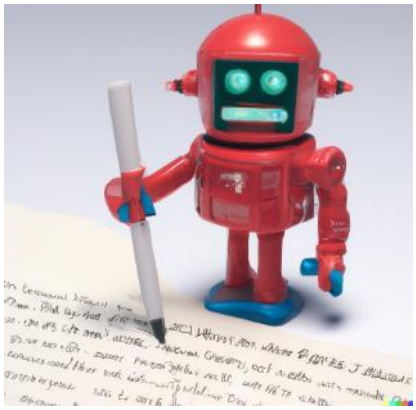


Abb. 14 Bild [Gemeinfrei] erzeugt mit DALL-E; Prompt „a small red robot writing with a white pen “ von Jörg [CC BY-SA 4.0 International]

Was Schüler lernen sollten

Das Grundprinzip ist einfach: Der gemessene Fehler eines neuronalen Netzes wird nach dem Verursacherprinzip an die nachgelagerten Neuronen zurückgemeldet. Neuronen mit großen Gewichtungsfaktoren erhalten somit einen größeren Korrekturauftrag. Während die Grundidee leicht zu verstehen ist, ist die mathematische Formulierung kompliziert. Deshalb beschränkt man sich zum einen auf die Vermittlung des Wirkungsprinzips und auf die Berechnung einfacher Fälle. Zu den einfachen Fällen gehört der Perceptron-Algorithmus, den die Schüler von Hand ausführen.

Außerdem wird der Mechanismus der Backpropagation erklärt und durchgespielt, die sogenannte Kettenregel. Mit Hilfe dieser Regel, die aus der Differentialrechnung stammt, wird der Fehler in neuronalen Netzen aufgeteilt und an die weiter hinten liegenden Neuronen weitergegeben. Anhand eines einfachen Beispiels aus dem Alltag wird das Funktionsprinzip veranschaulicht.

Mögliche Aktivitäten der Schüler

Backpropagation wird in vier aufeinander aufbauenden Teilen gelehrt:

Im ersten Teil verwendet die Lehrkraft Illustrationen, um die Grundidee zu vermitteln. Die Schüler sollten hier verstehen, dass der Fehler eines einzelnen Neurons zurückverfolgt wird, um die Gewichtungsfaktoren zu verbessern. Die Schüler sollten die Illustrationen in ihre Hefte kopieren.

Bei mehreren verknüpften Neuronen, d. h. einem neuronalen Netz, muss ein Fehler aufgeteilt werden. Hier setzt sich der Gedanke fort, dass die Gewichtungsfaktoren umso mehr verändert werden, je größer ihr Einfluss ist. Dies kann im Lehrer-Schüler-Gespräch auf fragend-entwickelnde Weise herausgearbeitet werden.

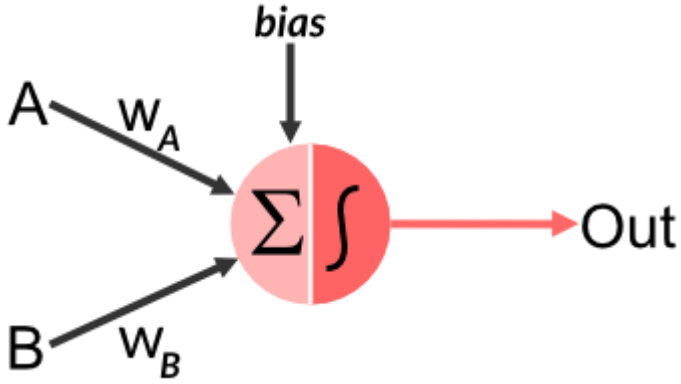
Anhand des einfachen Beispiels von Körper- und Schuhgröße wird die Kettenregel vermittelt. Diese wird mit Hilfe eines Arbeitsblattes von Hand ausgearbeitet. Anschließend erfolgt die Übertragung der Kettenregel auf ein Perceptron. Dies geschieht jedoch nur qualitativ, höhere Mathematik wird nicht verwendet.

Dass eine Backpropagation mit einer Lernrate tatsächlich funktioniert, wird mit einer Logikfunktion selbständig erarbeitet. Ein Perzeptron mit zunächst falsch gewählten Gewichtungsfaktoren wird schrittweise verbessert (Losgröße 1), bis nach einer Epoche die korrigierten Gewichtungswerte erreicht sind.

In einem letzten Schritt werden die erlernten Prinzipien auf neuronale Netze ausgedehnt, was zu einem mehrdimensionalen Problem führt, das nicht mehr analytisch, sondern nur noch iterativ gelöst werden kann. Die Terminologie der lokalen und globalen Minima wird eingeführt und das allgemeine Verfahren zur Optimierung von Lernprozessen mit Hilfe des Fallschirmmodells modelliert.

Teil I: Das Grundprinzip des Vorwärts- und Rückwärtspasses

Berechnen Sie den Output: Vorwärtspass



Im Vorwärtsdurchlauf sind die Gewichte und Bias-Werte fest und unveränderlich. Die Eingangssignale werden durch das Netzwerk propagiert und erzeugen das Ergebnissignal an den Ausgangsneuronen.

Dies geschieht sowohl während der Lernphase als auch später, wenn das Netz das Lernen abgeschlossen hat, während der Inferenzphase, in der das Netz regelmäßig arbeitet.

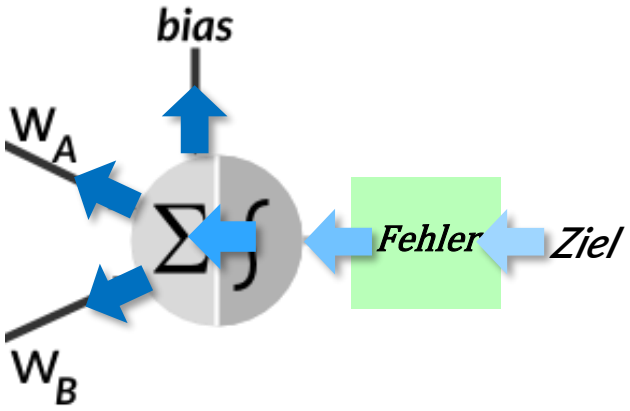
Berechnen Sie den Gesamtfehler als Summe der Fehler aller Ausgangsneuronen

$$error_{total} = \sqrt{\sum_{\substack{all\ output \\ neurons\ i}} (target_i - output_i)}$$

Nur während der Lernphase wird der Fehler berechnet: Die Ausgangssignale des Netzes werden mit den tatsächlichen Sollwerten verglichen. Der Fehler ergibt sich aus der Differenz zwischen den beiden Signalen. Er wird quadriert, um vorzeichenunabhängig zu werden.

Alle Fehlersignale aller Ausgangsneuronen werden addiert: So wird der Gesamtfehler des Netzes berechnet.

Berechnen Sie die Korrekturen: Backpropagation



Nun zur Fehlerrückübertragung in das neuronale Netz, wo der Fehler verteilt wird:

Je größer der Eingabewert eines Neurons ist, desto mehr wird der Fehler dort verteilt. Dabei werden die Gewichte entsprechend dem zugewiesenen Fehleranteil backpropagiert. Dieses Prinzip wird von der Ausgabe bis zurück zur Eingabeschicht fortgesetzt.

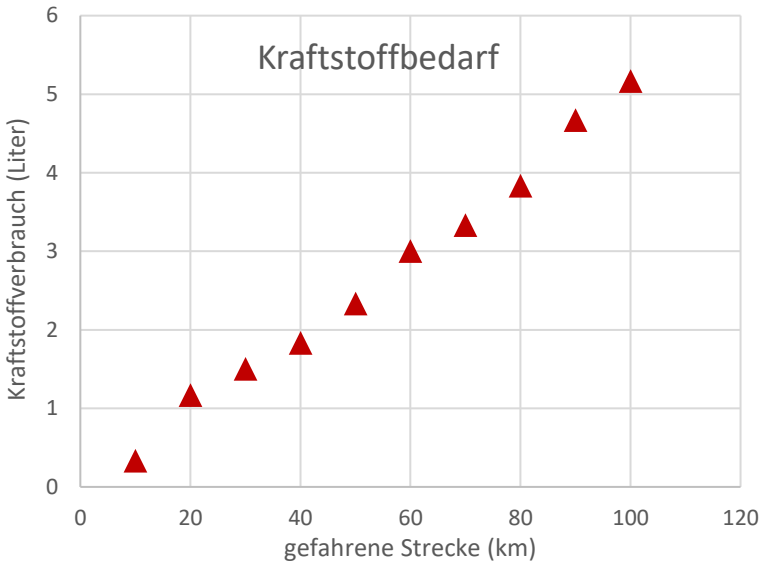
Während das Vorhersagesignal des Netzes im Vorwärtsdurchlauf von links nach rechts durch alle Neuronen läuft, läuft das Fehlersignal im Backpropagation-Durchlauf von rechts nach links zurück und alle Gewichte werden leicht in die rechte Richtung verändert.

Abb. 15 Bild [Gemeinfrei] erzeugt mit DALL-E; Prompt „a petrol station with a robot in the style of children's books“ von Jörg [CC BY-SA 4.0 International]

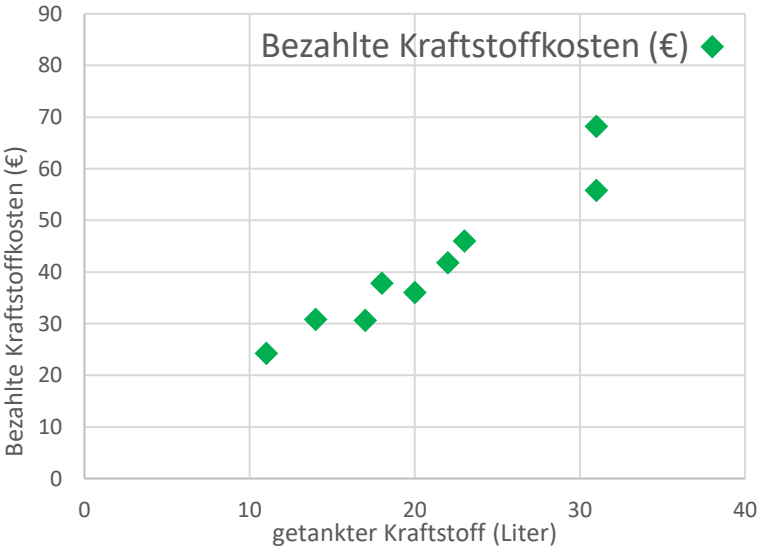
Teil II: Ein Modell für die Kettenregel

Sie wollen eine längere Strecke (250 km) mit dem Auto fahren und abschätzen, wie viel das Benzin kosten wird. Zu diesem Zweck erhalten Sie zwei Datensätze in Diagrammform. Aus diesen Diagrammen können Sie in zwei Schritten berechnen, wie teuer es voraussichtlich sein wird.

a) Im ersten Diagramm ist der Kraftstoffverbrauch in Abhängigkeit von der gefahrenen Strecke dargestellt. In einem ersten Schritt können Sie mit einem Lineal eine gerade Linie durch die Datenpunkte ziehen. Wie können Sie den Zusammenhang mathematisch formulieren?



b) Im zweiten Diagramm sind die Kosten der letzten Tankstellenbesuche dargestellt. Auch hier können Sie den Zusammenhang mit einer Geraden formulieren.



c) Wie können wir nun die zu erwartenden Kosten für die 250-Kilometer-Fahrt abschätzen?
mit Hilfe der beiden Beziehungen "Entfernung zu Kraftstoffverbrauch" und und "getankte Kraftstoffmenge zu gezahlten Euro"?

d) Nehmen wir an, dass der Tankstellenpreis für den Liter Benzin korrekt ist. Nun stellen Sie aber fest, dass Sie für die 250 km lange Strecke tatsächlich 75 Euro bezahlen mussten. Wie groß ist der Fehler und wie können Sie ihn korrigieren?

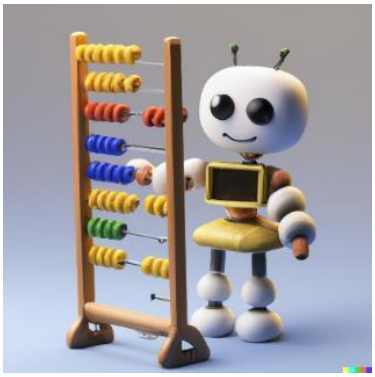
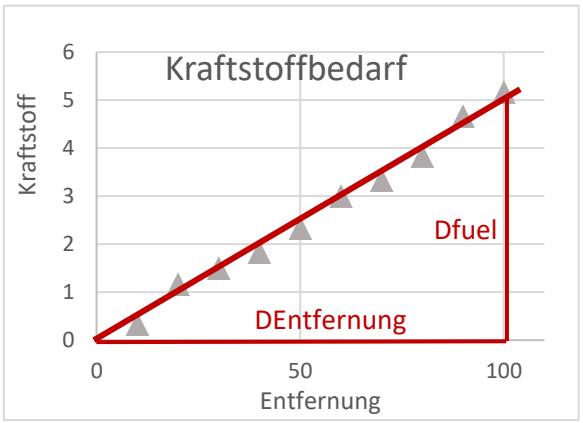


Abb. 16 Bild [Gemeinfrei] erzeugt mit DALL-E; Prompt „cute robot stands in front of the abacus “ von Jörg [CC BY-SA 4.0 International]

Lösung

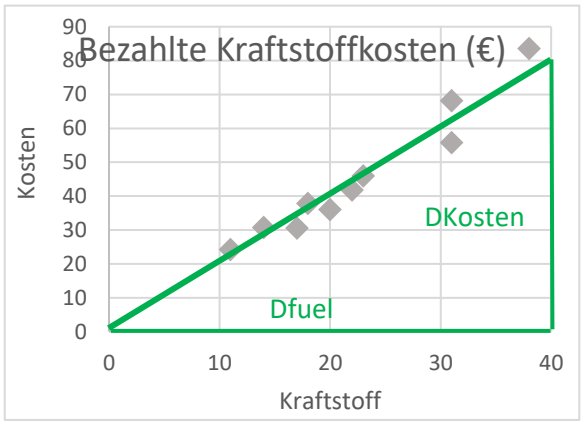
a) Die Datenpunkte können linear interpoliert werden.
Die Steigung kann durch Division von Δdistance durch Δfuel bestimmt werden:

$$\text{fuel} = \frac{\Delta\text{fuel}}{\Delta\text{distance}} \cdot \text{distance} = \frac{5}{100} \cdot \text{distance}$$



b) Auch hier können die Datenpunkte linear interpoliert werden.
Die Steigung kann durch Division von Δdistance durch Δfuel ermittelt werden:

$$\text{costs} = \frac{\Delta\text{costs}}{\Delta\text{fuel}} \cdot \text{fuel} = \frac{80}{40} \cdot \text{fuel}$$



c) Um die Kosten in Abhängigkeit von der Entfernung zu berechnen, muss man die beiden Gleichungen miteinander verknüpfen: **Das ist die Kettenregel!**

$$\text{costs} = \frac{\Delta\text{costs}}{\Delta\text{fuel}} \cdot \text{fuel} = \frac{\Delta\text{costs}}{\Delta\text{fuel}} \cdot \frac{\Delta\text{fuel}}{\Delta\text{distance}} \cdot \text{distance} = \frac{80}{40} \cdot \frac{5}{100} \cdot \text{distance}$$

Dazu multiplizieren Sie die beiden Steigungen miteinander. Für 250 km benötigen Sie 25 Euro.

d) Sie müssen den Fehler berechnen, der wie folgt lautet:

Ziel - Ausgang = 75 Euro - 25 Euro = 50 Euro ("Ziel" ist der tatsächliche Wert, "Ausgang" ist der berechnete Begriff)

Der erste Term, $\frac{\Delta\text{costs}}{\Delta\text{fuel}}$ ist fest. Daher müssen wir den zweiten Term ändern $\frac{\Delta\text{fuel}}{\Delta\text{distance}}$:

$$\text{costs} = \frac{80}{40} \cdot \frac{\Delta\text{fuel}_{corrected}}{\Delta\text{distance}} \cdot \text{distance} = \frac{80}{40} \cdot \frac{15}{100} \cdot \text{distance}$$

Sie muss größer werden!
Wie man das automatisch macht, wird im nächsten Kapitel erklärt.

Hinweis für die Lehrkraft:
Dieses Ergebnis lässt sich direkt auf ein Perzeptron übertragen: Der Fehler wird durch Anpassung der Steigung - also des Gewichtungsfaktors - über die Vermittlung der Aktivierungsfunktion rückgekoppelt. Dies kann im Unterricht genauer ausgearbeitet werden.

Teil III: Perceptron-Backpropagation

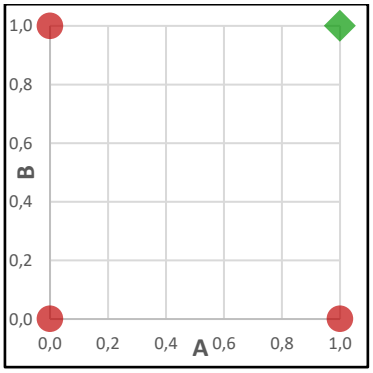
https://sefiks.com/2020/01/04/a-step-by-step-perceptron-example/

Nehmen wir an, wir wollen ein UND-Gatter-Perzeptron optimieren:

Wahrheitstabelle, Ziel Ausgabe

A	B	Aus
0	*0*	0
0	1	0
1	*0*	0
1	1	1

Wahrheitsdiagramm



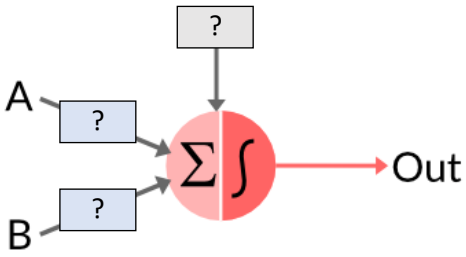
Zufällige Start-Gewichte

w_A
0.8

w_B
0.8

Bias
-0.50

Untrainiert Perceptron



Wir werden die Gewichte zufällig festlegen. Sagen wir, dass

- $w_A = 0.8$, $w_B = 0.8$
- bias = -0.5 (was korrekt ist und nicht geändert werden sollte, um die Aufgabe einfach zu halten)
- Lernrate = 0.2

Die Optimierung dauert einige Runden, Epochen genannt. Eine Epoche bedeutet, dass alle Stichproben einmal berechnet werden. Die zweite Epoche beginnt, wenn alle Stichproben abgeschlossen sind und zum zweiten Mal berechnet werden, usw.

Berechnen wir nun die erste Epoche

1.1) Als Beispiel wird der Vorwärtsdurchlauf der ersten Stichprobe [0, 0] angegeben. Hier ist der Fehler gleich Null.

2.1) Berechnen Sie nun den Vorwärtspass für die zweite Stichprobe [0,1]. Beenden Sie die Berechnung, wenn Sie den ersten Fehler finden. *(Da wir vier Stichproben haben ([0,0], [0,1], [1,0], [1,1]) und wir jeden einzelnen Fehler sofort korrigieren, ist die sogenannte "batch-size" 1).*

A	$\cdot w_A^{current}$ 0.8	B	$\cdot w_B^{current}$ 0.8	bias	sum	out (Schritt)	target	Fehler = Ziel - aus
0	*0*	0	*0*	-0.5	-0.5	0	0	0
0		1					0	
1		0					0	
1		1					1	

Lösung:

A	$\cdot w_A^{current}$	B	$\cdot w_B^{current}$	bias	sum	out	target	Fehler
0	*0*	0	*0*	-0.5	-0.5	0	0	0
0	***0***	1	***0.8***		***0.3***	***1***	0	***-1***

Fehler (für die erste Probe A = 0, B = 0) = tatsächlich - Vorhersage = 0 - 0 = 0 (kein Stopp erforderlich)

Fehler (für zweite Probe A = 0, B = 1) = tatsächlich - Vorhersage = 0 - 1 = -1 (STOP)

2.2)Backpropagation-Schritt:

Wir fügen zu den Gewichten den folgenden Korrekturterm hinzu: Fehler mal Lernrate. Wir setzen die Lernrate auf einen Wert von 0.2 (dies ist ein sogenannter Hyperparameter, da er vor den Berechnungen festgelegt und definiert werden muss):

$$w_A^{new,corrected} = w_A^{old} + learningrate \cdot error = 0.8 + 0.2 \cdot (-1) = 0.8 - 0.2 = 0.6$$

$$w_B^{new,corrected} = w_B^{old} + learningrate \cdot error = 0.8 + 0.2 \cdot (-1) = 0.8 - 0.2 = 0.6$$

3.1) Vorwärtspass für die dritte Probe [1,0]:

A	$\cdot w_A^{current}$ 0.6		B	$\cdot w_B^{current}$ 0.6		bias		sum	out (step)		target	Fehler = Ziel - Aus
0	*0*		0	*0*		-0.5		-0.5	0		0	0
0			1								0	
1	***0.6***		0	***0***				***0.1***	***1***		0	***-1***
1			1								1	

3.2) Backpropagation-Schritt:

Wir fügen zu den Gewichten den folgenden Korrekturterm hinzu: Fehler mal Lernrate. Wir setzen die Lernrate auf einen Wert von 0.3 (dies ist ein sogenannter Hyperparameter, da er vor den Berechnungen festgelegt und definiert werden muss):

$$w_A^{new,corrected} = w_A^{old} + learningrate \cdot error = 0.6 + 0.2 \cdot (-1) = 0.6 - 0.2 = 0.4$$

$$w_B^{new,corrected} = w_B^{old} + learningrate \cdot error = 0.6 + 0.2 \cdot (-1) = 0.6 - 0.2 = 0.4$$

4.1) Vorwärtspass für die vierte Probe [1,1]:

A	$\cdot w_A^{current}$ 0.4		B	$\cdot w_B^{current}$ 0.4		bias		sum	out (step)		target	Fehler = Ziel - Aus
0	*0*		0	*0*		-0.5		-0.5	0		0	0
0			1								0	
1			0								0	
1	***0.4***		1	***0.4***				***0.3***	***1***		1	***0***

4.2)**Backpropagation-Schritt:** Nicht notwendig, da der Fehler gleich Null ist!

Schüler Aufgabe: Berechnen Sie die zweite Epoche

Lösung:

A	$\cdot w_A^{current}$		B	$\cdot w_B^{current}$		bias		sum	out		target	Fehler
0	*0*		0	*0*		-0.5		-0.5	0		0	0
0	***0***		1	***0.4***				***-0.1***	***0***		0	***0***
1	***0.4***		0	***0***				***-0.1***	***0***		0	***0***
1	***0.4***		1	***0.4***				***0.3***	1		1	***0***

Der Fehler ist in allen Stichproben gleich Null, das neuronale Netz ist perfekt trainiert und macht in allen Fällen korrekte Vorhersagen.

Abb. 17 Bild [Gemeinfrei] erzeugt mit DALL-E; Prompt „a cute little robot during parachute jump with white background, digital art “ von Jörg [CC BY-SA 4.0 International]

Teil IV: Das Modell des Fallschirmspringers

Was die Schüler lernen sollten:

Die Optimierung bzw. der Lernprozess eines neuronalen Netzes gleicht zu Beginn einem Glücksspiel: Da man nicht weiß oder nicht abschätzen kann, wie man die Gewichtungsfaktoren setzen soll, bleibt nur eine Möglichkeit:

muss man die Anfangswerte der Gewichtungsfaktoren schätzen. Das haben wir bei den Simulationen gesehen: Bei jedem Neustart wurden die Startwerte zufällig zurückgesetzt.

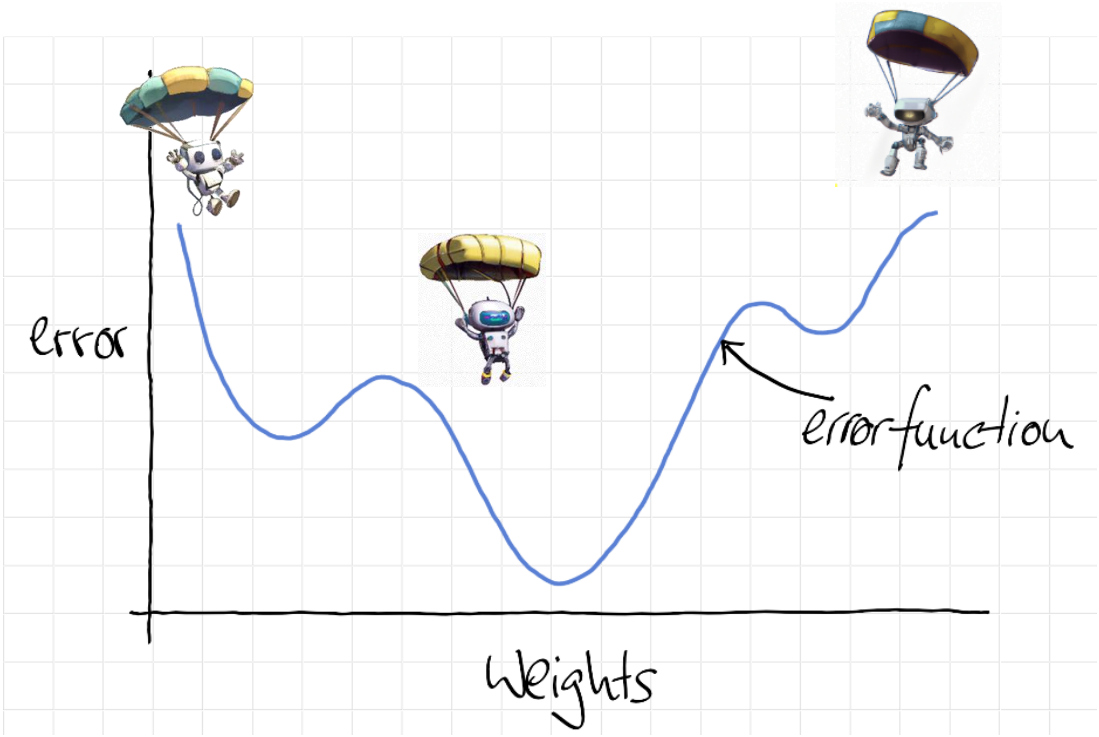
Daraus ergeben sich verschiedene Nachteile, nämlich dass man nie sicher sein kann, durch den Lernprozess die bestmöglichen Gewichtungswerte zu erhalten. Und weil man nicht weiß, ob es nicht vielleicht eine bessere Variante gibt, muss man diesen Prozess öfter wiederholen:

1. Sie werfen die Würfel,
2. den Optimierungsprozess durchführen
3. und werten Sie das Ergebnis aus
4. und wieder und wieder.

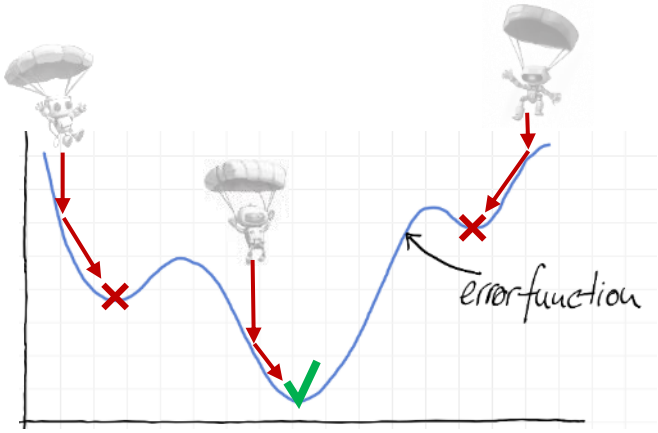
Aufgabe 1: Wohin gehen die Fallschirmspringer?

Da die Optimierung entlang einer (leider unbekannten) mathematischen Funktion verläuft, kann man ihre Form mit einem unbekannten Terrain vergleichen. Die gewürfelten Ausgangswerte ähneln einem Schwarm zufällig abgeworfener Fallschirmspringer.

Wenn sie auf der unbekannten mathematischen Funktion gelandet sind, ist ihre weitere Aufgabe einfach: Sie laufen immer bergab und hoffen, durch ihre Bewegung ins Tal den optimalen tiefsten Punkt des gesamten Geländes zu finden. Dies ist die bildhafte Vorstellung des mathematischen Gradientenabstiegs.



Lösung



Viele der Fallschirmspringer erreichen leider nur höhere Plateaus - so genannte lokale Minima. Und nur sehr wenigen Fallschirmspringern - wenn überhaupt - ist es vergönnt, das globale Minimum zu erreichen. Und oft muss man sich mit einem möglichst guten lokalen Minimum zufriedengeben.

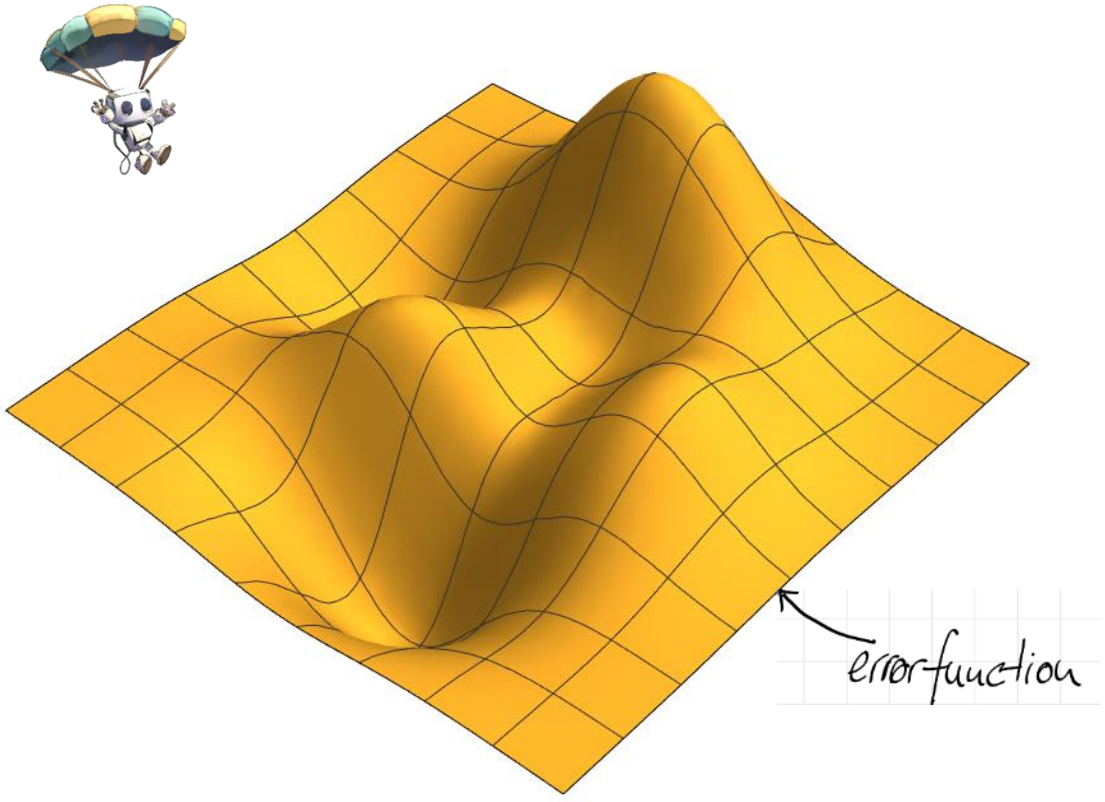
Die mit einem roten Kreuz markierten Stellen sind die lokalen Minima, d.h. die Gewichte haben ihre optimalen Werte nicht erreicht.

Die mit dem grünen Haken markierte Stelle in der Mitte ist der beste Wert der Fehlerfunktion und bezeichnet den optimalen Satz von Werten für die Gewichte eines neuronalen Netzes.

Aufgabe 1: Eine kompliziertere Fehlerfunktion

Das obige Modell kann gedanklich erweitert werden, und man kann sich vorstellen, dass die Struktur der Fehlerfunktionen umso komplizierter wird, je mehr Gewichtungsfaktoren eine Rolle spielen. Was man sich bildlich vorstellen kann, ist die abgebildete Funktion.

Markieren Sie hier zwei bis drei Fallschirmspringer (also zufällige Startwerte) und zeichnen Sie deren Gradientenabstieg ein. Wo sind lokale, wo globale Minima?



Lösung:

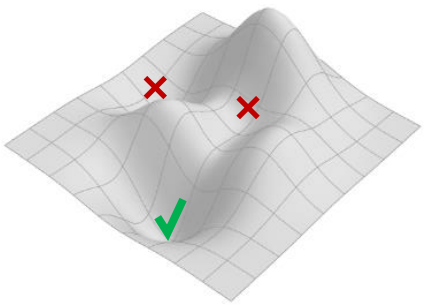


Abb. 18 Bild [Gemeinfrei] erzeugt mit DALL-E; Prompt „a cute little robot starrit at a Van Gogh drawing, digital art “ von Jörg [CC BY-SA 4.0 International]

Lektion 8: Einführung in die Bildklassifizierung

Was Schüler lernen sollten

Was kann man nun mit all dieser Theorie anfangen? Eine konkrete Anwendung ist die Unterscheidung von Mustern - also von strukturierten Eingaben. Dabei ist es zunächst unerheblich, ob es sich bei den Eingaben um Sensordaten, Töne oder Bilder handelt.

Für Schüler ist es jedoch intuitiver, wenn die Eingabedaten in Form von Bildern vorliegen. In diesem Fall werden didaktisch reduzierte Bilder ausgewählt, die aus wenigen Pixeln bestehen. Und diese Pixel können sogar nur die beiden Werte 0 oder 1 annehmen.

Es ist wichtig zu wissen, dass jedes Pixel des Bildes ein entsprechendes Eingangsneuron erhält. Bilder, die aus 3x3 Pixeln bestehen, benötigen also 9 Eingabeneuronen. Je mehr Pixel ein Bild hat, desto mehr Eingangsneuronen werden benötigt.

Diese Lektion unterscheidet sich von CNN (convolutional neural networks), die normalerweise zur Bilderkennung eingesetzt werden. Hier ist ein zusätzlicher theoretischer Überbau notwendig, für den hier die Grundlagen gelegt werden.

Mögliche Aktivitäten der Schüler

Die Schüler müssen in einem ersten Schritt in die Theorie der Pixelbilder eingeführt werden. Dieser Inhalt ist jedoch nicht in diesem Dokument zu finden. Darauf aufbauend wird die sehr leistungsfähige neuronale Netzsimulation von Tran Vuong Quoc Anh vorgestellt, mit der sich die Schüler zunächst vertraut machen sollten.

Mit Hilfe dieser Simulation können die bereits bekannten logischen Funktionen neu trainiert werden: AND, OR, IMPLICATION und XOR werden wiederholt und vertieft. Die Möglichkeiten der Simulation sind ausgeschöpft: Zum einen die grafische Darstellung und zum anderen die tabellarische Darstellung ("Tabelleneingabe") erlauben unterschiedliche Sichten auf die Vorgänge.

Die eigentliche Stärke der Simulation zeigt sich in der Eingabemöglichkeit für eigene Datensätze. Diese können in tabellarischer Form eingetippt werden. Außerdem können die Schüler die Netztopologie frei einstellen, d.h. sie können wählen, wie viele versteckte Schichten mit wie vielen Neuronen ihr neuronales Netz haben soll.

Wichtige Erkenntnisse aus Ihren eigenen Projekten sollten sein:

- Jedes Bildpixel benötigt ein eigenes Eingangsneuron.
- Jedes unterscheidbare Ergebnis benötigt ein eigenes Ausgangsneuron.

Wenn Sie zum Beispiel ein Bild mit 4x4 Pixeln haben und 5 verschiedene Bilder unterscheiden wollen, brauchen Sie 16 Eingangsneuronen und 5 Ausgangsneuronen.

Fertige Arbeitslösungen können dann gespeichert und der Klasse präsentiert werden. Ebenso sind Protokolle der Vorgehensweise möglich, die von den Schülern über ihre Projekte erstellt werden.

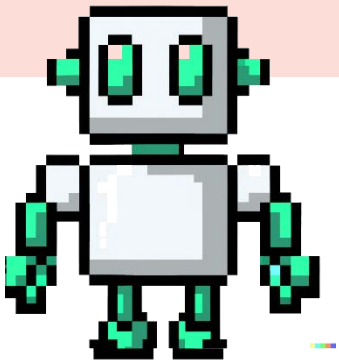


Abb. 19 Bild [Gemeinfrei] erzeugt mit DALL-E; Prompt „a cute little robot pixel art “ von Jörg [CC BY-SA 4.0 International]

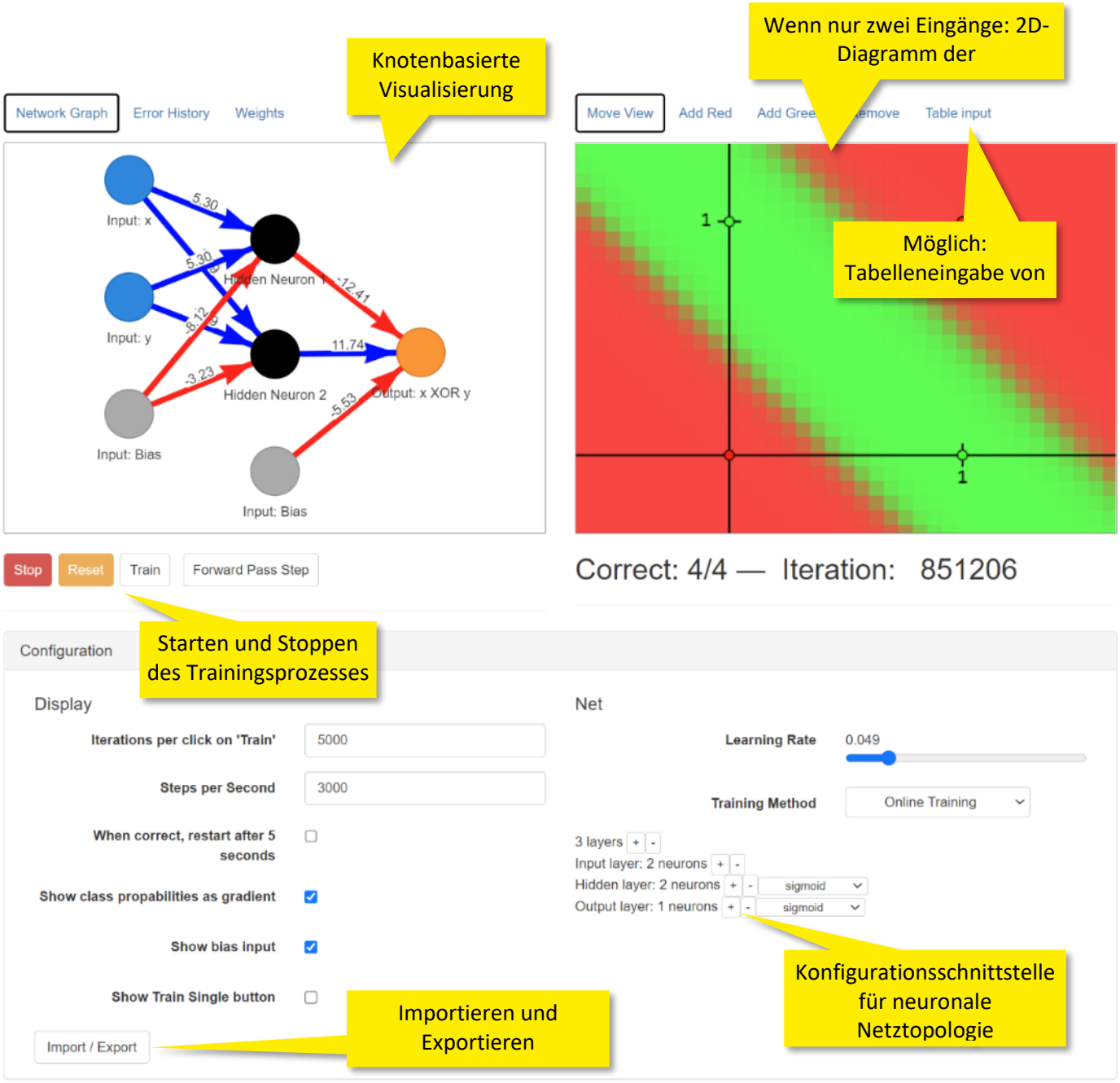
Aufgabe I: Sich mit der Simulation vertraut machen

Webbasierte Version:
https://anhcoi123.github.io/neural-network-demo/

Github-Repository für die Offline-Nutzung:
https://github.com/anhcoi123/neural-network-demo/

Die hier verwendete Simulation ist webbasiert. Die obere Hälfte enthält die Visualisierungselemente, die untere Hälfte die Steuerelemente zur Einstellung der Hyperparameter wie Lernrate, Iterationen, Stapelgröße oder Trainingsmethode.

Auf eine vollständige Einführung muss hier verzichtet werden, die Schüler sollen durch einfache Aufgabenstellungen spielerisch an das Planspiel herangeführt werden.



Einfache Einführungsaufgaben

a) Laden Sie die Voreinstellung für "Binary Classifier for XOR" mit der Schaltfläche 'Load Preset' in der oberen rechten Ecke. Starten Sie das Training mit "Animate". Wie verhält sich das Netz während des Trainings, wenn Sie die Aktivierungsfunktionen der einzelnen Neuronenschichten ändern?

b) Fügen Sie nun weitere Punkte hinzu, indem Sie auf "Grün hinzufügen" oder "Rot hinzufügen" klicken. Versuchen Sie, Muster zu erstellen, zum Beispiel in Form eines Kreuzes oder eines Donuts. Wo liegen die Grenzen dieser einfachen Gittertopologie? Wie viele Neuronen müssen Sie hinzufügen, um selbst komplizierte Muster zu erkennen?

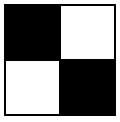
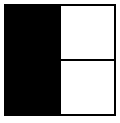
c) Laden Sie nun das Preset "Bit Position Auto Encoder". Warum kann hier kein Diagramm mehr angezeigt werden, sondern nur noch eine tabellarische Ansicht? Wie erkennen Sie den Trainingsfortschritt?

Aufgabe II: Muster für 4 Eingangsneuronen

- mindestens drei verschiedene Pixelmuster mit 2x2 Pixeln entwerfen.
- Jeder Pixel kann entweder schwarz oder weiß sein.
- Dann erstellen Sie ein neuronales Netz, das Sie auf diese Pixelmuster trainieren.

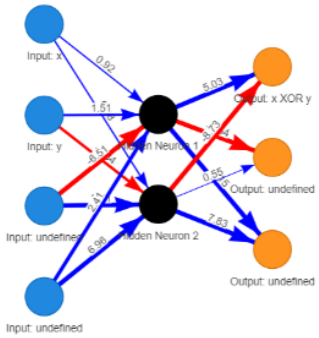
Verfahren:

- zeichnen Sie die verschiedenen Muster,
- die Muster in Bitfolgen umwandeln und für jedes Element dieser Bitfolge ein Eingangsneuron erstellen.
- für jede mögliche Ausgabe wird ebenfalls ein eigenes Neuron erstellt.
- Wenn die Netztopologie festgelegt ist, können Sie die Werte eingeben und sie dann trainieren.
- halten Sie Ihre Beobachtungen schriftlich fest.

Muster	Pixel-Muster	Serialisiert Reihenfolge	Verschlüsselt Buchstabe			Vollständige Reihenfolge
		a,b,c,d	L	X	I	a,b,c,d,L,X,I
	0 1 0 0	0,1,0,0	1	0	0	0,1,0,0,1,0,0
	0 1 1 0	0,1,1,0	0	1	0	0,1,1,0,0,1,0
	0 1 0 1	0,1,0,1	0	0	1	0,1,0,1,0,0,1

Drei verschiedene Optionen für Pixelmuster. Da es vier Pixel gibt, sind theoretisch $2^4 = 16$ verschiedene Muster möglich. Die Schülerinnen und Schüler sollten hier eine kleine Auswahl treffen. In einem Zwischenschritt wird das Pixelmuster serialisiert, d.h. in ein 1D-Array umgewandelt. Die jeweiligen Muster erhalten eine binäre Darstellung nach dem One Hot Encoding: Bei der One-Hot-Codierung werden für kategoriale Daten neue Binärspalten erzeugt, die das Vorhandensein jedes möglichen Wertes aus den Originaldaten kennzeichnen.

Mögliche Lösung:



Inputs				Expec Outpu			Actual Output		
a	b	c	d	L	X	I	L	X	I
0	1	0	0	1	0	0	0.977	0.014	0.022
0	1	1	0	0	1	0	0.019	0.976	0.02
0	1	0	1	0	0	1	0.016	0.014	0.976

Dies ist eine mögliche Lösung für die Tabelleneingabe. In den Spalten "L", "X" und "I" des Abschnitts "Expected Output" finden Sie die ein-hot-codierte Darstellung der drei Kategorien. Im tatsächlichen Output finden Sie den trainierten Klassifikationsstatus des Netzes. Bevor Sie mit dem Training beginnen, werden diese Werte zufällig festgelegt.

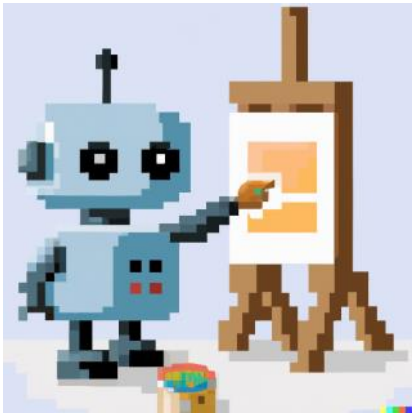


Abb. 20 Bild [Gemeinfrei] erzeugt mit DALL-E; Prompt „a cute little robot painting a bitmap with 9 pixels, digital art “ von Jörg [CC BY-SA 4.0 International]

Aufgabe III: Muster für 9 Eingangsneuronen

Wiederholen Sie den Vorgang aus der vorherigen Aufgabe, diesmal mit 3x3 Pixel-Bitmaps.

a) Wie viele verschiedene Muster sind möglich?

b) Nehmen Sie 3 oder 4 mögliche Muster und bereiten Sie sie für die Eingabe in die Tabelle vor.

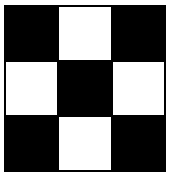
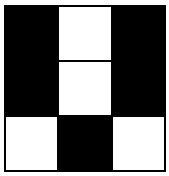
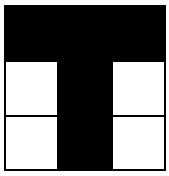
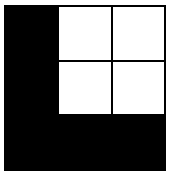
c) Exportieren Sie die Daten in eine csv-Datei, ändern Sie die Muster, laden Sie sie und trainieren Sie Ihr Netzwerk.

d) Lassen Sie sich die Gewichte anzeigen (neben "Netzwerkdiagramm" und "Fehlerhistorie"). Können Sie erklären, was Sie hier sehen?

Mögliche Lösung:

a) $2^9 = 512$ Patterns, von denen einige nützlich sind.

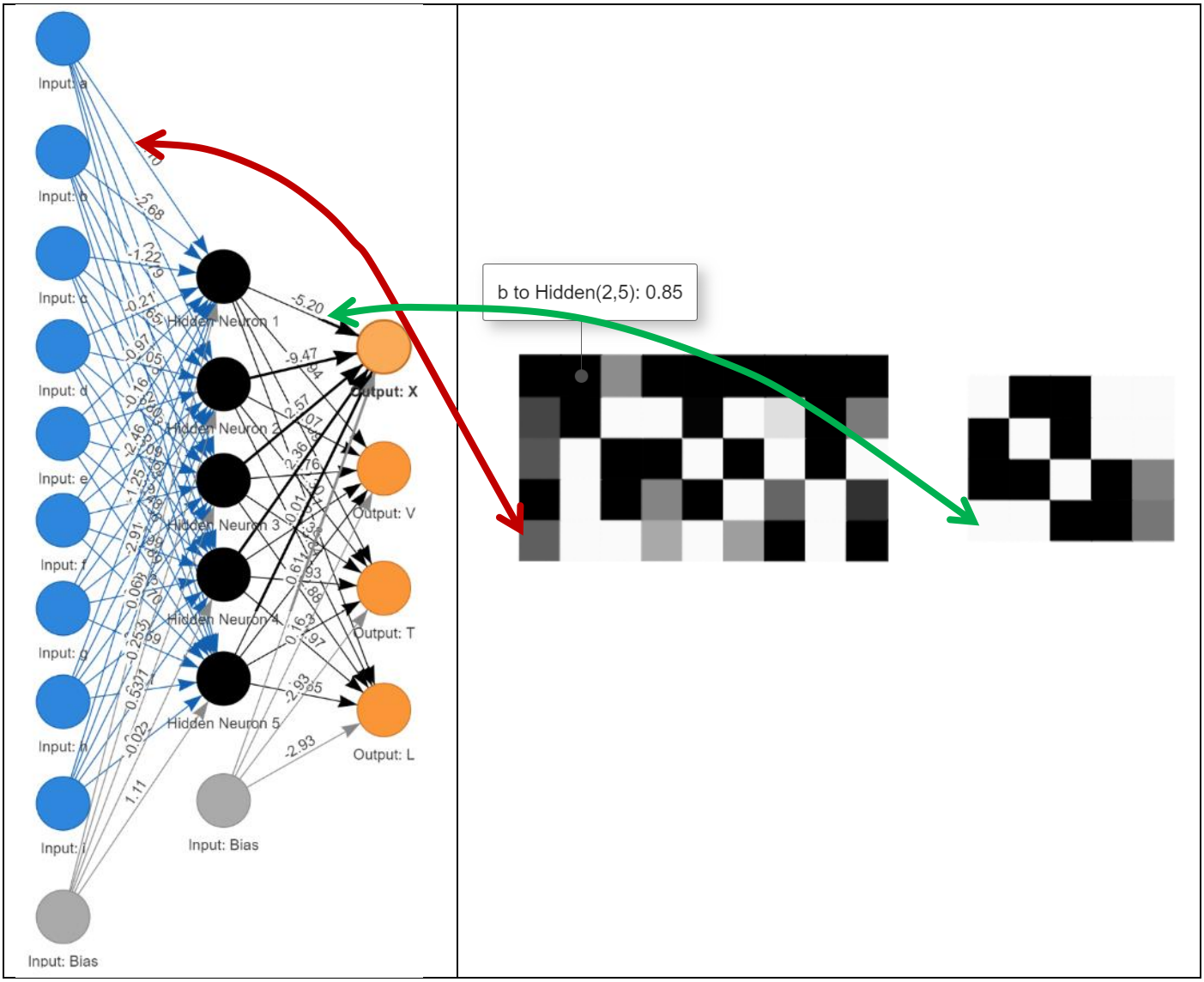
b) Nehmen wir zum Beispiel die Buchstaben X, V, T und L:

Brief	Pixelmuster	Serialisierte Sequenz	Codierte Buchstaben				Vollständige Lernsequenz
		a,b,c,d,e,f,g,h,i	**X**	**V**	**T**	**L**	**a,b,c,d,e,f,g,h,i,X,V,T,L**
	0 1 0 1 0 1 0 1 0	0,1,0,1,0,1,0,1,0	1	0	0	0	0,1,0,1,0,1,0,1,0,1,0,0,0
	0 1 0 0 1 0 1 0 1	0,1,0,0,1,0,1,0,1	0	1	0	0	0,1,0,0,1,0,1,0,1,0,1,0,0
	0 0 0 1 0 1 1 0 1	0,0,0,1,0,1,1,0,1	0	0	1	0	0,0,0,1,0,1,1,0,1,0,0,1,0
	0 1 1 0 1 1 0 0 0	0,1,1,0,1,1,0,0,0	0	0	0	1	0,1,1,0,1,1,0,0,0,0,0,0,1

c)

```
a,b,c,d,e,f,g,h,i,X,V,T,L
0,1,0,1,0,1,0,1,0,1,0,0,0
0,1,0,0,1,0,1,0,1,0,1,0,0
0,0,0,1,0,1,1,0,1,0,0,1,0
0,1,1,1,1,1,0,0,0,0,0,0,1
```

d)



Jeder graue Wert der Gewichtskarte entspricht einem numerischen Gewichtsfaktorwert im neuronalen Netz. Dies wird anhand von zwei Beispielen veranschaulicht: ein Gewichtsfaktor verbindet die Eingabe- und die verborgene Schicht (roter Pfeil) und ein anderer Gewichtsfaktor die verborgene und die Ausgabeschicht (grüner Pfeil).

Einige Aufgaben für Klassenarbeiten:

a) ***(2P)*** Skizzieren und erläutern Sie mit Beschriftungen die Grundstruktur eines Perzeptrons. Erläutern Sie die einzelnen Funktionseinheiten und wie sie miteinander verbunden sind.

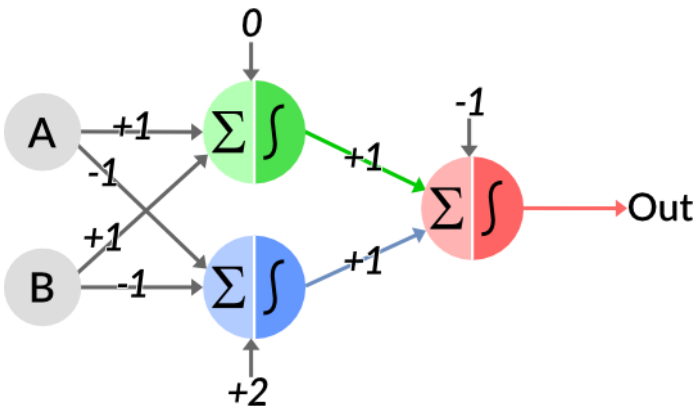
b) ***(4P)*** Die logische Tabelle der sogenannten "Implikation" ist rechts dargestellt:

- Erstellen Sie ein Logikdiagramm, in dem Sie die Tabelle visualisieren und die Diskriminatorlinie einzeichnen, wie wir es im Unterricht gelernt haben.

- Berechnen Sie ein Perceptron, dass diese Logikfunktion erkennt.

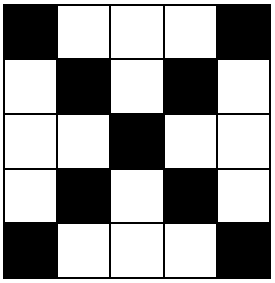
Bit 1 (A)	Bit 2 (B)	Ergebnisbit (Y)
0	0	1
0	1	1
1	0	0
1	1	1

c) ***(3P)*** Berechnung eines neuronalen Netzes:
Gegeben sei das neuronale Netz auf der rechten Seite. Es arbeitet mit der Step-Funktion als Aktivierung. Der Eingang A ist 1 und der Eingang B ist ebenfalls 1. Berechnen Sie den Ausgang des Netzes für diesen Eingang.



d) ***(2P)*** Damit ein Perzeptron lernen kann, verwendet man einen speziell entwickelten Algorithmus. Benennen und beschreiben Sie diesen Lernalgorithmus in Stichworten.

e) ***(2P)*** Stellen Sie die folgende 1-Bit-Bitmap als Array dar:



f) ***(3P)*** Konstruieren und zeichnen Sie ein neuronales Netz für dieses Pixelbild. Dieses Netz sollte in der Lage sein, zwischen 6 verschiedene Pixelbilder zu unterscheiden. Wie viele Eingänge und wie viele Ausgänge braucht das Netz?

g) ***(2P)*** Wie könnte eine mögliche Netzwerktopologie zwischen Eingang und Ausgang aussehen? Verwenden Sie für Ihre Beschreibung die Fachsprache.

Abb. 21 Bild [Gemeinfrei] erzeugt mit DALL-E; Prompt „a cute robot standing next to a big rocket under a black sky with stars, digital art“ von Jörg [CC BY-SA 4.0 International]

Einige Gedanken zum Schluss

Dieser Kurs behandelt die Grundlagen neuronaler Netze, um jüngeren Schülern ein grundlegendes Verständnis dieser Technologie zu vermitteln.

Es beginnt mit der Idee, die Natur mit Hilfe der Informatik zu imitieren, baut die Computerdarstellung von Neuronen mit Hilfe einfacher mathematischer Modelle und Beziehungen auf und führt über einfache Aufgaben zum Aufbau von Netzwerken.

Der Kurs endet mit einer einfachen neuronalen Bilderkennung. Ein bewusster Schnitt wird vor der Einführung von CNN (Convolutional Neural Network) gemacht. Dieser Inhalt ist komplex, weil er Theorien aus verschiedenen Bereichen kombiniert: Zum Beispiel gehören Faltungsfilter nicht mehr in die KI.

Weitere Lehrgänge können jedoch auf dem hier Erreichten aufbauen: Seien es Autoencoder, CNNs oder vielleicht sogar Transformer-Modelle. Denkbar ist auch eine Vertiefung in Richtung eingebetteter Systeme, d.h. kleine neuronale Netze auf Mikrocontrollern.

Die Entwicklung im Bereich der künstlichen Intelligenz schreitet schnell voran und die Entwicklung einer leistungsfähigen Didaktik bleibt eine wichtige Herausforderung für die Informatikausbildung.

Es bleibt spannend!