Kurs „Künstliche Intelligenz und Robotik“ für Schulen

Version 12.11.2022

Marco Lardelli

https://ki-kit.ch

Publizierte Versionen:

28.4.22	Erste Version
4.5.22	Kleinere Korrekturen
9.5.22	Variable „Anzahl Streifen“ anstatt „Gewicht“, div. Verbesserungen
31.5.22	Zusätzliche Slides mit Schlussfolgerungen
9.9.22	Korrekturen
25.9.22	Div. Verbesserungen
2.11.22	Div. Verbesserungen
12.11.22	Zusätzliche Aktivität (Slide 26.)
26.1.23	Ergänzungen

DANKE!

Ideen, Feedback, Testing etc.:

Christian Ungerer
(Schule Burghalde Baden)

Rolf Beck
(www.pglu.ch)

Rudolf Tanner
(mechmine GmbH)

Finanzielle Unterstützung:

WISSENSCHAFT. BEWEGEN
GEBERT RÜF STIFTUNG

PATREON

ki-kit

 = Aktivität

(✔) = Optional

Der Kurs ist weitgehend als Dialog mit den SchülerInnen gestaltet. Die SchülerInnen sollen sich die wesentlichen Konzepte selber anhand von einfachen Beispielen erarbeiten. In kurzen Abständen sind zudem Aktivitäten zur Vertiefung geplant (Diskussion, Rechenbeispiele, Serious-Games etc.).
Gewisse Inhalte sind als Optional gekennzeichnet. Sie dienen der weiteren Vertiefung bzw. vermitteln mathematische Konzepte, die u.U. den SchülerInnen schon bekannt sind.

1. Was ist Intelligenz?

?

Welche Begriffe fallen dir zum Wort „Intelligenz“ ein?

1. Was ist Intelligenz?

Genie

Wissen

Talent

Emotionale I.

Können

Kreativität

Rationale I.

Motorische I.

Begabung

Bildung

Wortwolke mit SuS erarbeiten. Z.B. mittels answergarden.ch

1. Was ist Intelligenz?

?

Welche intelligenten „Systeme“ (d.h. Lebewesen **und** technische Geräte) kennst du?

1. Was ist Intelligenz?

Beispiele für intelligente „Systeme“:








Die dargestellten Systeme sind Beispiele. Sie unterscheiden sich in folgenden Eigenschaften:

- Auf enges Gebiet beschränkte Fähigkeiten (Taschenrechner, Staubsaugerroboter) bzw. Fähigkeiten auf vielen Gebieten (Bonobo, Mensch)
- Kann selbständig neue Fähigkeiten erwerben (z.B. Katze) oder nicht (z.B. selbstfahrendes Auto)
- Künstlich bzw. natürlich

Fotos: Pixabay

1. Was ist Intelligenz?

Was können diese Systeme bzw. was macht sie intelligent?

Diskutiere mit deinem Banknachbar.

Welche Fähigkeiten machen letztlich Intelligenz aus?

1. Was ist Intelligenz?

1. Probleme lösen
2. Lernen

Lernen =
Selbständig neue Fähigkeiten zur Problemlösung erwerben.

Historisch lag der Fokus früher bei 1. (so hat man z.B. frühe Computer als „Elektronengehirne“ bezeichnet und ihnen Intelligenz zugeschrieben) und hat sich dann immer mehr zu 2. verlagert.

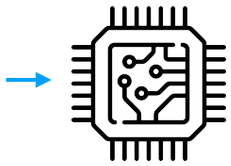
2. Was bedeutet Lernen?

Ein Computer soll lernen, Hunde von Katzen zu unterscheiden (nur anhand von deren **Grösse und der Anzahl Streifen im Fell**!)

Zusätzliche Angaben wären natürlich hilfreich. Aber wir wollen versuchen, das Ziel nur mit diesen beiden Angaben zu erreichen (sicher nicht perfekt möglich, da es z.B. auch gestreifte kleine Hunde gibt).
Wir zählen jede Art von Farbübergang als Streifen.

2. Was bedeutet Lernen?

Tierart	# Streifen	Grösse
Hund	0	0.4
Katze	5	0.15
Hund	1	0.3
Katze	12	0.17



Wir geben dem Computer für jedes Tier die Grösse und die Anzahl der Streifen und sagen ihm auch, ob es sich um einen Hund oder eine Katze handelt.

Wie könnte der Computer lernen, Hunde und Katzen zu unterscheiden?

Es sind natürlich viele Antworten möglich. Es ist möglich, dass einzelne SchülerInnen schon ein komplexeres Modell vorschlagen. Wir wollen zuerst die einfachsten Lösungen betrachten.

2. Was bedeutet Lernen?

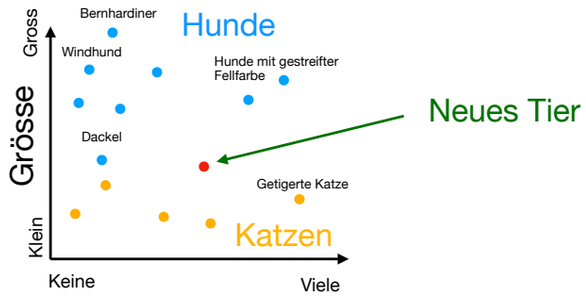
Eine erste mögliche Lösung:

# Streifen	Grösse	Tierart
0	0.4	→ Hund
5	0.15	→ Katze
1	0.3	→ Hund
12	0.17	→ Katze

Der Computer merkt sich einfach für jede Kombination von Grösse und Anzahl Streifen, ob es ein Hund oder eine Katze ist.

Warum ist diese Lösung schlecht?

2. Was bedeutet Lernen?

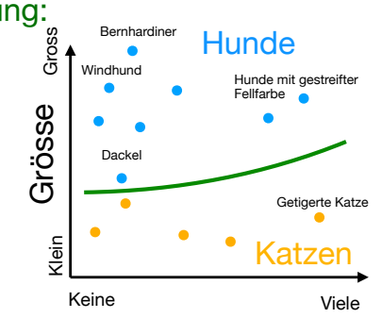


Problem: Wenn man dem Computer ein noch unbekanntes Tier übergibt, kann er keine Entscheidung treffen, da er es nicht kennt! Er hat gewissermassen nur auswendig gelernt, aber nichts verstanden!

Wie könnte eine bessere Lösung funktionieren?

2. Was bedeutet Lernen?

Eine bessere Lösung:

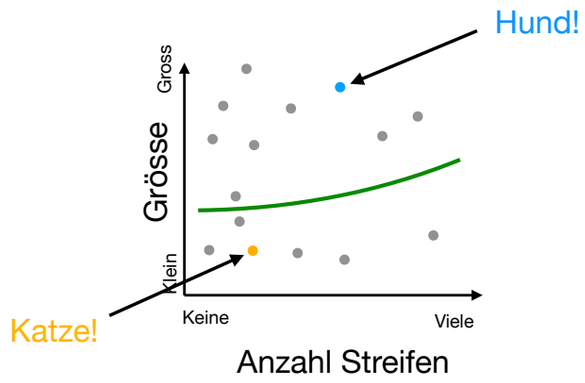


Der Computer könnte aus diesen Daten die grüne Kurve berechnen:

Alle Tiere oberhalb sind Hunde

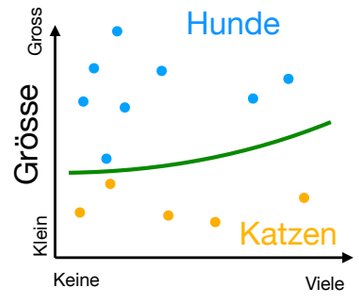
Alle Tiere unterhalb sind Katzen

2. Was bedeutet Lernen?



Nun kann der Computer auch für noch unbekannte Tiere sagen, ob sie Hunde oder Katzen sind. D.h. er hat tatsächlich etwas **gelernt**!

2. Was bedeutet Lernen?

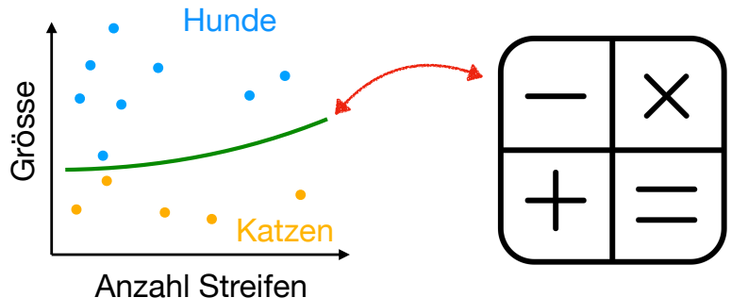


Wie kann ein Computer eine solche grüne Kurve finden?

D.h. wie kann ein Computer **lernen**?

Bemerkung:
In einem ähnlichen Beispiel mit drei Variablen müssten die Datenpunkte durch eine Fläche in einem dreidimensionalen Raum getrennt werden.
Mehr als drei Variablen sind möglich, lassen sich aber nicht mehr visualisieren. Aus diesem Grund betrachten wir nur 2 Variablen.

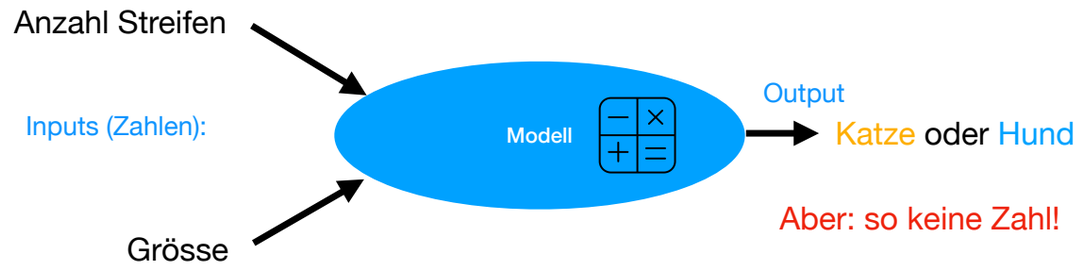
2. Was bedeutet Lernen?



Wir müssen dazu die Kurve irgendwie **mathematisch** beschreiben (nur dann kann ein Computer damit rechnen!).

Eine solche Beschreibung nennt man in der KI ein „**Modell**“.

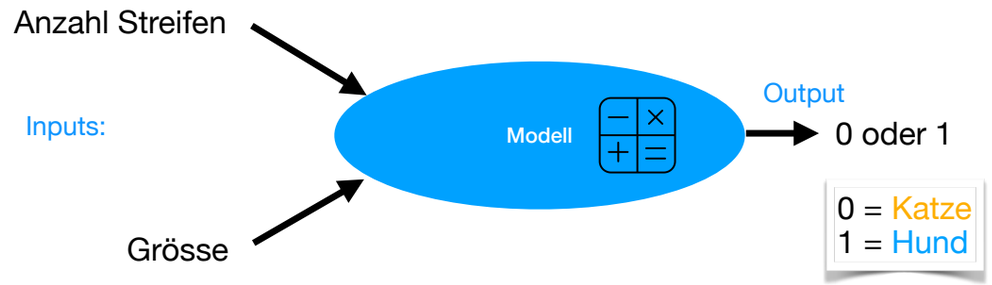
3. Wir erfinden ein KI-Modell



Das Modell soll eine mathematische Rechenvorschrift sein. Der Output muss also auch eine Zahl sein.
Wie können wir „Katze oder Hund“ mit einer einzigen Zahl beschreiben?

Natürlich kann ein Computer auch Zeichenketten-Outputs erzeugen (z.B. „HUND“). Das Problem damit ist, dass wir dann später nicht durch Hintereinanderschalten von einfachen Modellen („Neuronen“) komplexe Modelle bilden können. Der Input muss ja eine Zahl sein. Wir müssen also unbedingt einen numerischen Output verlangen.

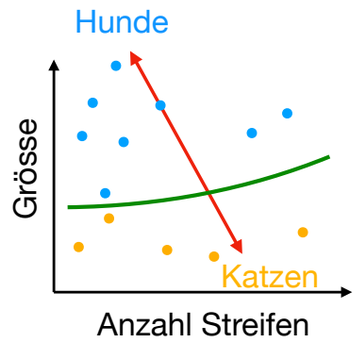
3. Wir erfinden ein KI-Modell



Wir verwenden für die beiden Kategorien zwei verschiedene Zahlen!

Es gäbe natürlich auch andere Lösungen. Aber diese ist sehr einfach.

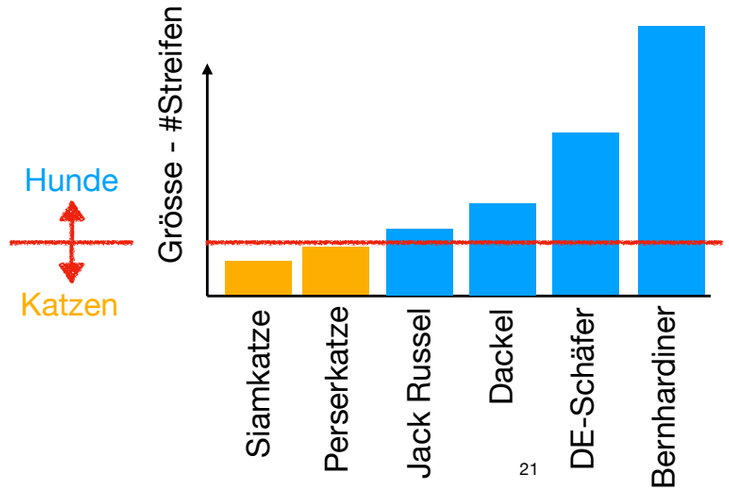
3. Wir erfinden ein KI-Modell



Wir wissen: Ein Tier ist umso eher ein Hund, je grösser und je weniger Streifen es hat.
Wie können wir dies **mathematisch** beschreiben?

3. Wir erfinden ein KI-Modell

Wir berechnen den Wert „Grösse minus die Anzahl der Streifen“:



Wenn (Grösse - #Streifen) grösser als ein bestimmter Schwellwert ist, ist es ein Hund (sonst eine Katze)

Schwellwert zur Unterscheidung von Hunden und Katzen

3. Wir erfinden ein KI-Modell

Bestimme für die folgenden Daten einen guten Schwellwert:

Anzahl Streifen	Grösse [cm]	Tierart
2	40	→ Hund
0	15	→ Katze
0	30	→ Hund
0	17	→ Katze
4	22	→ Katze
0	22	→ Hund

Es muss zuerst die Differenz gebildet werden und dann nach dieser sortiert werden.

3. Wir erfinden ein KI-Modell

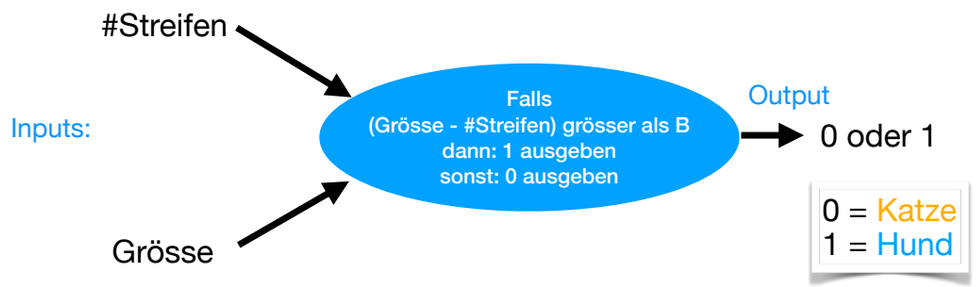
#Streifen	Grösse	Grösse-#Streifen	Tierart
0	15	15	→ Katze
0	17	17	→ Katze
4	22	18	→ Katze
0	22	22	→ Hund
0	30	30	→ Hund
2	40	38	→ Hund

Ein guter Schwellwert wäre z.B. der Wert 20

Ca. den Mittelwert der beiden trennenden Differenzen wählen.
Man sieht gut, wie die die Angabe der Anzahl Streifen die Trennung zwischen den beiden 22cm grossen Tieren ermöglicht.

3. Wir erfinden ein KI-Modell

Wir nennen den Schwellwert „B“. Unser Modell sieht dann so aus:



3. Wir erfinden ein KI-Modell

Falls (Grösse - #Streifen) = B ist, sagen wir das Tier ist eine Katze, aber es könnte eigentlich ebensogut ein Hund sein.

An dieser Grenze gilt daher:

Grösse = B + #Streifen

(Gleichung umformen).

Umformen: Auf beiden Seiten #Streifen addieren.

3. Wir erfinden ein KI-Modell

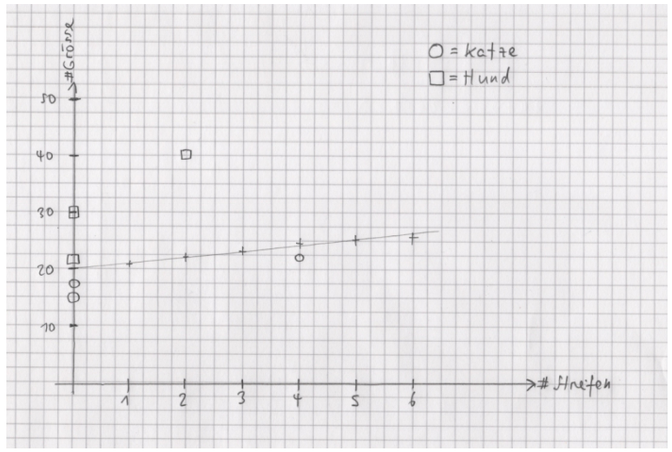
Übung:

- Erstelle ein geeignetes Koordinatensystem für Grösse und #Streifen
- Trage darin die Tiere aus unserer Tabelle ein. Zeichne einen kleinen Kreis für die Katzen bzw. ein kleines Rechteck für die Hunde
- Zeichne zusätzlich für die Streifenanzahl 1,2,3,4,5,6 die Grösse G = 20 + #Streifen ein und verbinde die Punkte miteinander.

#Streifen	Grösse	Tierart
0	15	→ Katze
0	17	→ Katze
4	22	→ Katze
0	22	→ Hund
0	30	→ Hund
2	40	→ Hund

3. Wir erfinden ein KI-Modell

Lösung:



Was sieht man nun?

1. Die Punkte für G = 20 + #Streifen liegen auf einer Geraden
2. Die Gerade trennt die Hunde von den Katzen!

3. Wir erfinden ein KI-Modell

Ein Computer vergleicht Zahlen, indem er sie voneinander subtrahiert und dann mit 0 vergleicht:

A > B

Ist das gleiche wie

A - B > 0

„>“ ist das mathematische Symbol für „grösser als“

Für unser Modell sieht das dann so aus:

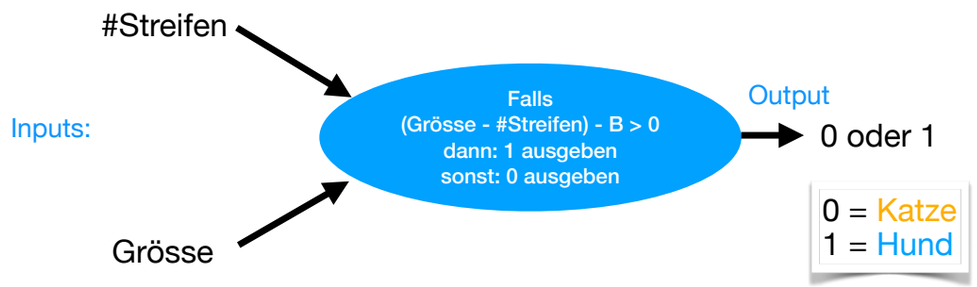
Grösse - #Streifen > B

Ist das gleiche wie

Grösse - #Streifen - B > 0

3. Wir erfinden ein KI-Modell

Unser Modell sieht nun so aus:



3. Wir erfinden ein KI-Modell

Tierart	#Streifen	Grösse [cm]
Hund	0	40
Katze	2	15
Hund	1	30
Katze	4	22

Stell dir vor, dass wir die Grösse in **Zentimetern** angeben.
Welches Problem könnte dann auftauchen? Schau dir die Zahlen genau an. Was fällt dir auf?

3. Wir erfinden ein KI-Modell

Grösse - Anzahl Streifen

Die Zahlen für die Grösse sind viel grösser als die Zahlen für die Anzahl Streifen!
→
Die Grösse wird in unserem Modell viel stärker berücksichtigt als die Anzahl Streifen. Das wird so nicht gut funktionieren!

Wie könnten wir das korrigieren?

3. Wir erfinden ein KI-Modell

Lösung: Wir multiplizieren beide Inputs je mit einer Zahl, damit sie etwa gleich gross werden.

Wir wählen z.B. eine kleine Zahl für W_1 und eine grosse für W_2.

$$W_1 * \text{Grösse} - W_2 * \#\text{Streifen}$$

z.B.

$$0.5 * \text{Grösse} - 3 * \#\text{Streifen}$$

Dann haben die zu subtrahierenden Werte wieder eine ähnliche Grösse.

Warum zwei Zahlen und nicht nur eine? Man könnte die beiden Eingangswerte auch auf eine ähnliche Grösse bringen, indem man nur einen Eingangswert mit einer Zahl W multipliziert.
Grund: Ein Modell kann viel mehr als nur zwei Eingänge haben! Es ist dann am einfachsten, für jeden Input ein eigenes W_i zu verwenden.

3. Wir erfinden ein KI-Modell

Wir modifizieren unser Modell nun noch ein wenig: Wir **addieren** immer und verwenden dafür eine **negative** Zahl für W_2:

$$W_1 * \text{Grösse} + W_2 * \#\text{Streifen}$$

Also:

$$0.5 * \text{Grösse} + (-3) * \#\text{Streifen}$$

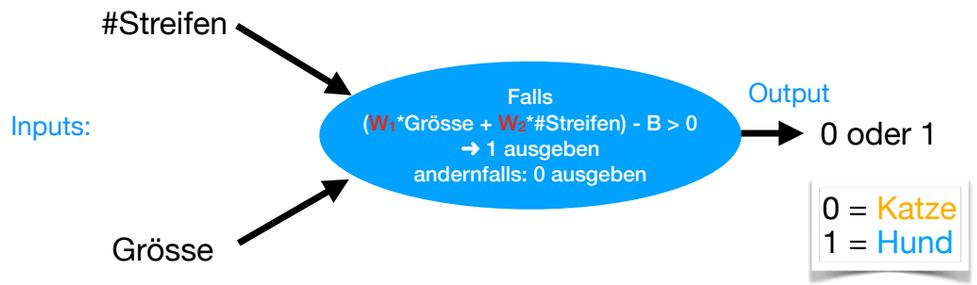
Das Resultat ändert sich dadurch nicht!

Man nennt einen Input mit einem negativen W_i einen sog. **hemmenden Eingang**. Einen Input mit einem positiven W_i nennt man einen **erregenden Eingang**.

Die Gewichte W_i sollen ganz allgemein der unterschiedlichen Bedeutung der Inputs Rechnung tragen (es geht also **nicht** nur darum, unterschiedliche Einheiten zu berücksichtigen).

3. Wir erfinden ein KI-Modell

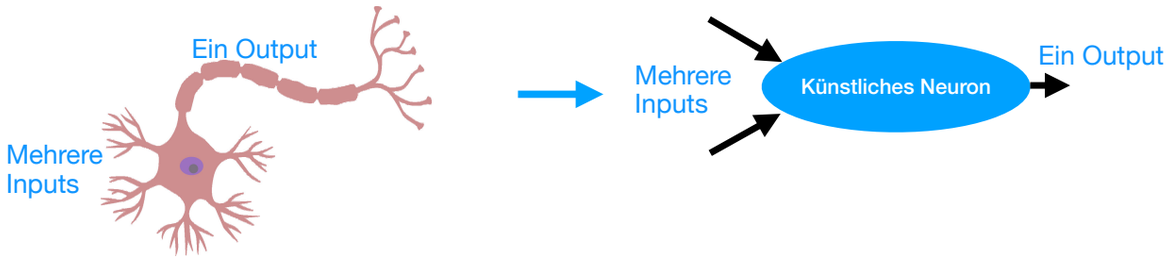
Nun sieht unser Modell so aus:



Das KI-Modell das wir soeben erfunden haben, nennt man in der KI das „**Perzeptron**“. Es ist ein Beispiel für ein **einfaches künstliches Neuron**.

Bemerkung:
Ein Perzeptron kann mehr als nur zwei Eingänge haben. Man könnte in unserem Beispiel z.B. noch das Gewicht des Tieres berücksichtigen.

3. Wir erfinden ein KI-Modell - Woher kommt der Name „Neuron“?

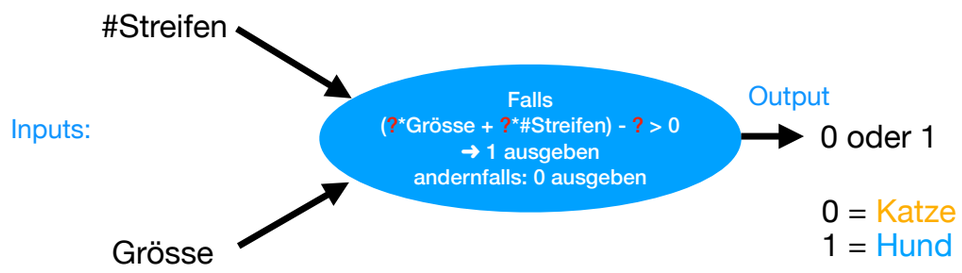


Das Vorbild für das k**ünstliche Neuron** ist das **biologische Neuron** (Nervenzelle, z.B. aus dem Gehirn).

Was bedeutet nun „lernen“ für unser Modell?

Eine genauere Betrachtung des biologischen Neurons ist sehr empfehlenswert und kann z.B. im Biologieunterricht stattfinden.

3. Wir erfinden ein KI-Modell - Lernen

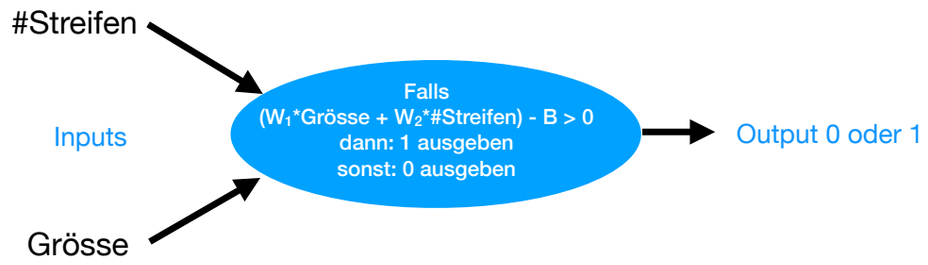


W_1, W_2 und B sind Zahlen, die wir **nicht kennen**! Wir müssen sie finden! Lernen bedeutet, gute Werte für diese Zahlen zu finden.

Man nennt diese gesuchten Zahlen auch die „**Parameter**“ des Modells.

Lernen (im Sinne der KI) bedeutet also, gute Parameter für ein bestimmtes KI-Modell zu finden.

4. Das „Perzeptron“: Rechenbeispiel



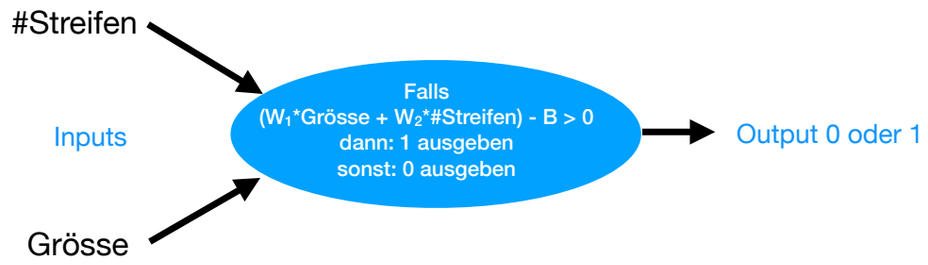
Die Parameter des Neurons seien $W_1 = 0.5$, $W_2 = -3$, $B = 10$

Berechne den Output für folgende Input-Paare:

1. Grösse = 40, Anzahl Streifen = 2
2. Grösse = 19, Anzahl Streifen = 5

1. Ist offensichtlich ein Hund, 2. vermutlich eine Katze.

4. Das „Perzeptron“: Rechenbeispiel - Lösung



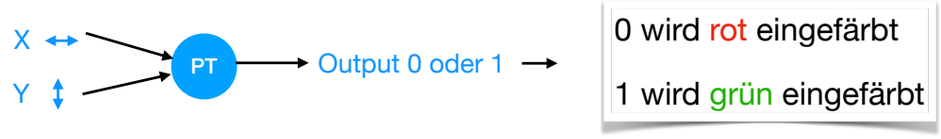
Die Parameter des Neurons seien $W_1 = 0.5$, $W_2 = -3$, $B = 10$

Lösung: Output für folgende Input-Paare:

1. Gr = 40, #Str = 2 → 0.5*40 + (-3)*2 - 10 = 4 (grösser als 0)→ 1 (Hund)
2. Gr = 19, #Str = 5 → 0.5*19 + (-3)*5 - 10 = -15.5 (kleiner als 0)→ 0 (Katze)

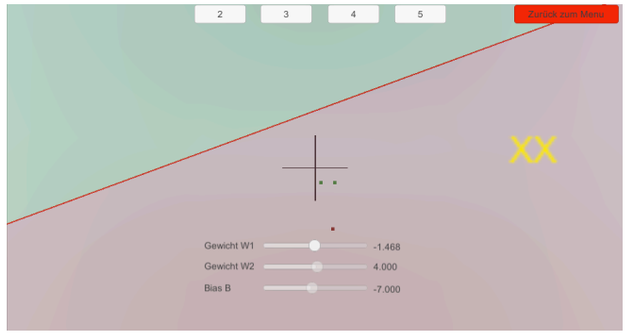
Das Perzeptron funktioniert mit diesen Parametern also recht gut!

4. Das „Perzeptron“: Bildliche Darstellung („Visualisierung“)



1. Für jeden Punkt („Pixel“) auf dem Bildschirm die Koordinaten x und y bestimmen → Inputs des Perzeptrons
2. Damit den Output des Perzeptrons berechnen
3. Gemäss Output und der obigen Regel den Punkt einfärben

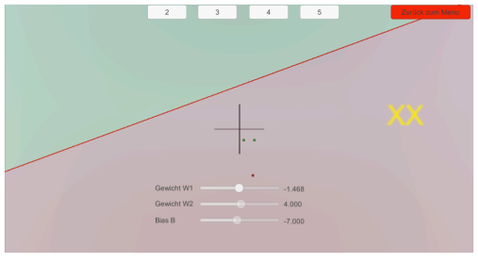
4. Das „Perzeptron“: Game



1. Zuerst oben die Anzahl der Datenpunkte („Tiere“) wählen → Es werden zufällige Daten erzeugt.

2. Versuche nun das Perzeptron durch Verändern der Parameter so einzustellen, dass es die Datenpunkte korrekt unterscheiden kann (d.h. rote Punkte im roten Bereich, grüne Punkte im grünen Bereich).

4. Das „Perzeptron“: Game - Schlussfolgerungen



Die „**Kurve**“ in unserem Perzeptron-Modell ist **tatsächlich nur eine Gerade**.

Das Perzeptron kann deshalb mit höchstens drei Datenpunkten sicher umgehen. Wenn es mehr sind, geht es nur manchmal.

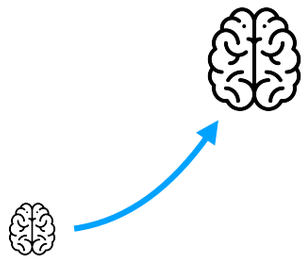
Das ist sehr wenig!

→

Wir benötigen ein besseres Modell!

Der mathematische Beweis, dass die Kurve bei einem Perzeptron mit zwei Eingängen durch eine Gerade gebildet wird, ist recht einfach und im Lehr- bzw. Lösungsbuch erklärt.

5. Ein verbessertes künstliches Neuron



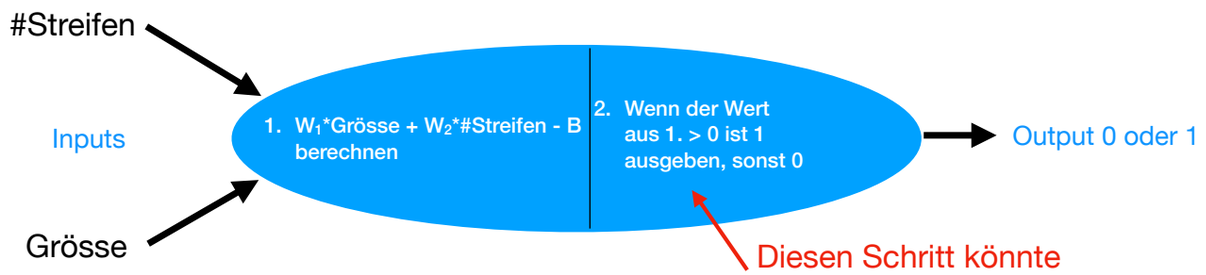
Zuerst versuchen wir das Perzeptron zu **verbessern**.

Dazu schauen wir uns den Aufbau von künstlichen Neuronen im Allgemeinen an.

Es gibt ganz viele verschiedene Arten.

5. Ein verbessertes künstliches Neuron

Zuerst zerlegen wir die Rechenschritte in unserem Perzeptron in **zwei Teile**:



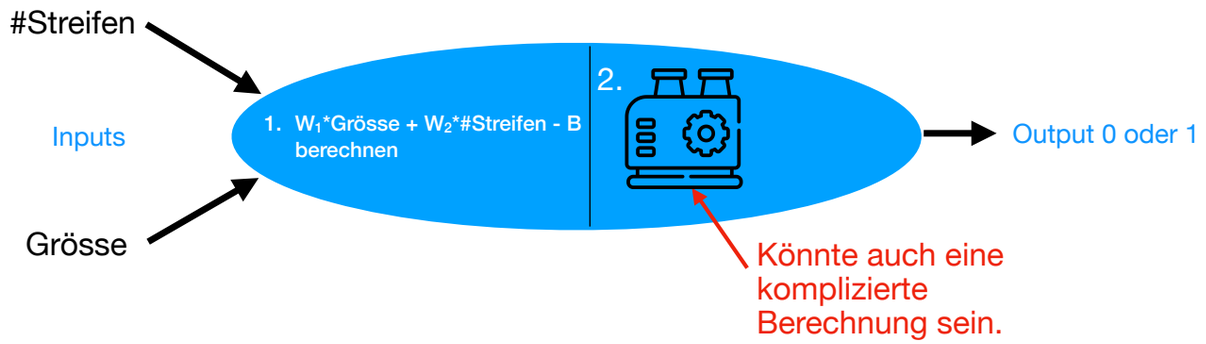
Dadurch ändert sich erstmal nichts!

Aber vielleicht können wir den 2. Rechenschritt durch einen besseren austauschen!

Diese Trennung in zwei Rechenschritte ist nötig, um andere Arten von künstlichen Neuronen zu verstehen.

5. Ein verbessertes künstliches Neuron

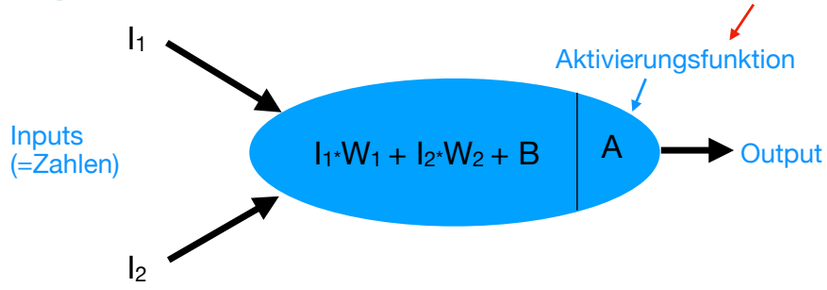
Wir stellen uns daher vor, dass wir im 2. Schritt beliebig komplizierte Berechnungen ausführen dürfen:



3. Ein verbessertes künstliches Neuron

Wir nennen die Berechnungsmaschine für Schritt 2 „Aktivierungsfunktion“

Aber was ist das?



Die Zahlen W_1 und W_2 heissen übrigens „Weights“ (Englisch f. „Gewichte“)

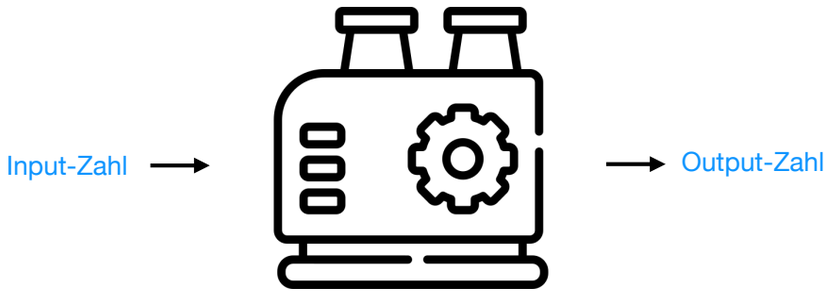
Die Zahl B heisst „Bias“

Alle diese Zahlen nennt man die „Parameter“

Es folgt eine einfache Erklärung des Funktionsbegriffs.

5. Aktivierungsfunktionen: Was ist eine Funktion? (✔)

Um verstehen zu können, was eine Aktivierungsfunktion ist,
müssen wir zuerst verstehen was eine Funktion ist.



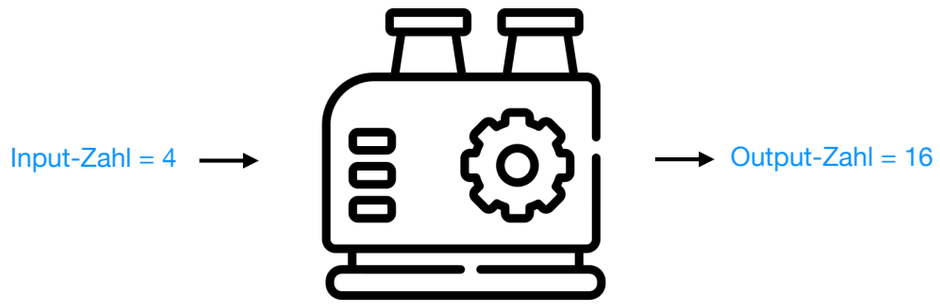
Als Funktion bezeichnet der Mathematiker eine kleine Rechenmaschine,
in welche man eine Zahl („Input“) hineingeben kann und dann daraus eine
andere Zahl (der „Output“) herauskommt.

Die folgenden Ausführungen sind nicht erforderlich, falls die SchülerInnen mit dem Funktionsbegriff schon einigermassen vertraut sind. Die Erklärungen sind minimal gehalten und orientieren sich an den einfachen Erfordernissen dieses Kurses (Verzicht auf mathematische Strenge und Fachsprache).

5. Aktivierungsfunktionen: Was ist eine Funktion? Beispiel(✔)

Beispiel:

Output-Zahl = Input-Zahl mit sich selber multipliziert



5. Aktivierungsfunktionen: Was ist eine Funktion? Übung

Beispiel für eine Funktion:

Output-Zahl = Input-Zahl mit sich selber multipliziert

Input-Zahl = 3 → Output-Zahl = ?

Input-Zahl = 1 → Output-Zahl = ?

Input-Zahl = -2 → Output-Zahl = ?

Achtung: negativ * negativ = positiv

5. Aktivierungsfunktionen: Was ist eine Funktion? Lösung

Beispiel für eine Funktion:

Output-Zahl = Input-Zahl mit sich selber multipliziert

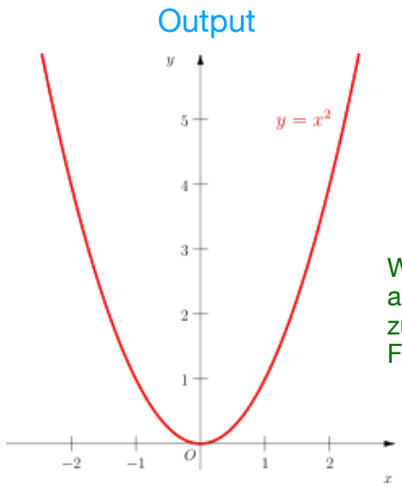
Input-Zahl = 3 → Output-Zahl = 9

Input-Zahl = 1 → Output-Zahl = 1

Input-Zahl = -2 → Output-Zahl = 4 (negativ mal negativ = positiv!)

5. Aktivierungsfunktionen: Was ist eine Funktion? Darstellung (✔)

Darstellung dieser Funktion:



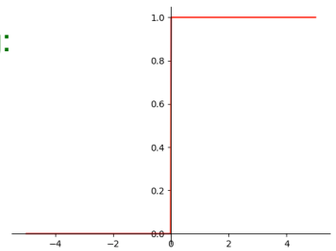
Wir zeichnen für jeden Punkt auf der x-Achse (Inputs) den zugehörigen Output der Funktion auf der y-Achse ein.

5. Aktivierungsfunktionen: Beispiele

Eine Aktivierungsfunktion ist eine spezielle Funktion, die in einem künstlichen Neuron verwendet wird.
Wir schauen uns zwei häufig verwendete Aktivierungsfunktionen an:

Step-Funktion: $\text{Output} = \begin{cases} 1, \text{ wenn der Input grösser als 0 ist} \\ 0, \text{ wenn der Input kleiner (oder gleich) 0 ist} \end{cases}$

Darstellung:



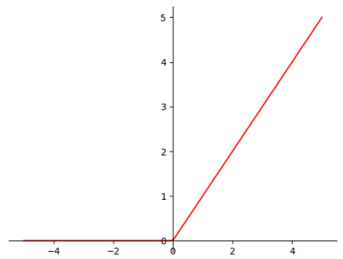
Diese Aktivierungsfunktion unterscheidet negative und positive Zahlen.

Die Step-Funktion wird z.B. bei Neuronen im Output-Layer verwendet (z.B. für ein Netz, das zwei Klassen unterscheiden soll).

5. Aktivierungsfunktionen: Beispiele

„ReLU“-Funktion: Output = { Gleich dem Input, wenn der Input grösser als 0 ist
0, wenn der Input kleiner (oder gleich) 0 ist

Darstellung:



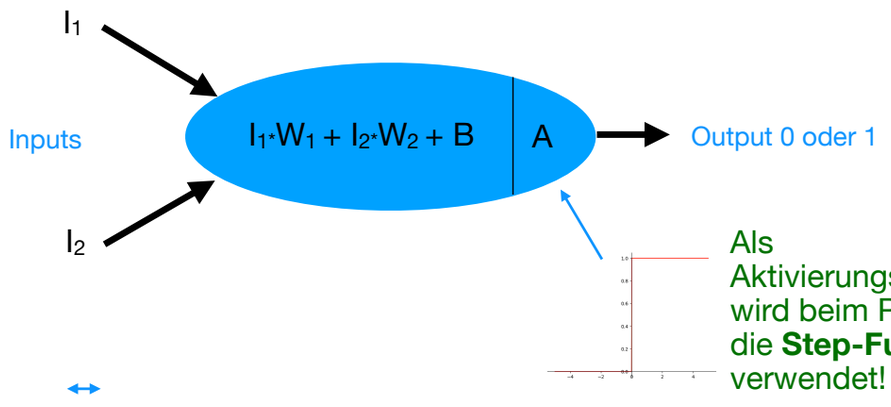
Diese Aktivierungsfunktion ersetzt negative Zahlen durch die 0.

Die ReLU-Aktivierungsfunktion wird in der Praxis sehr oft verwendet.

„ReLU“ bedeutet „Rectified Linear Unit“

5. Aktivierungsfunktionen: Zurück zum „Perzeptron“

Nun können wir verstehen, warum unser Perzeptron ebenfalls ein künstliches Neuron ist:

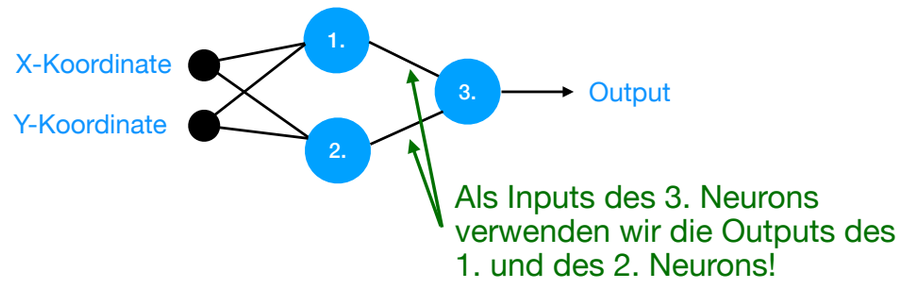


Als Aktivierungsfunktion wird beim Perzeptron die **Step-Funktion** verwendet!

6. Ein einfaches Neuronales Netz

Eine weitere Möglichkeit, ein besseres Modell zu erhalten:

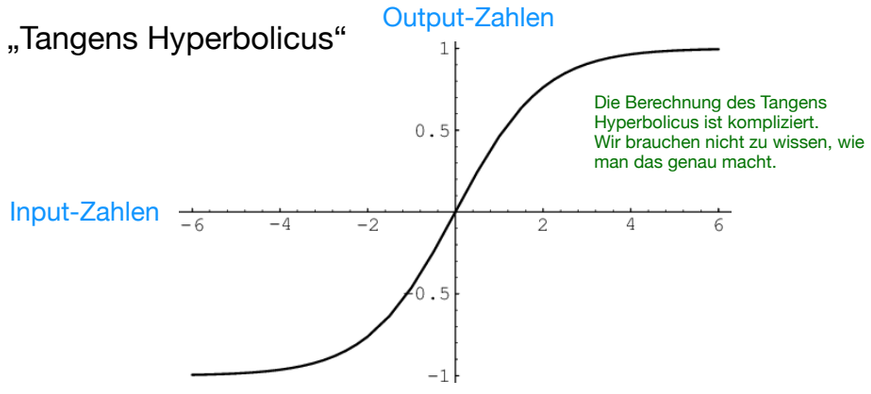
Wir schalten **mehrere** künstliche Neuronen hintereinander!



6. Ein einfaches neuronales Netz: Aktivierungsfunktion „Tanh“

Wir verwenden nun eine komplizierte Aktivierungsfunktion:

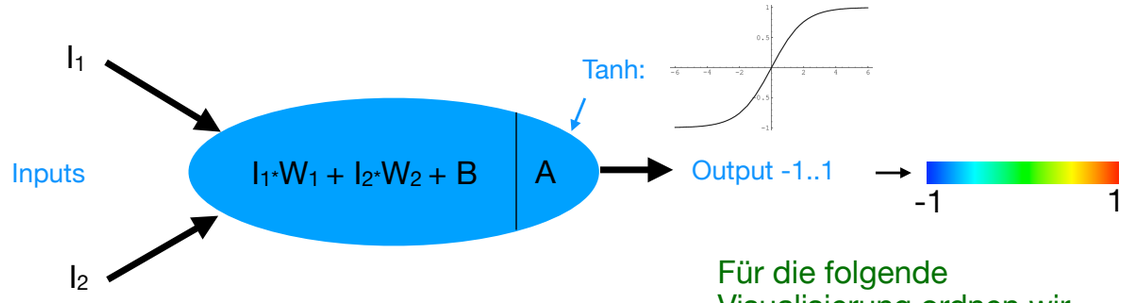
„Tangens Hyperbolicus“

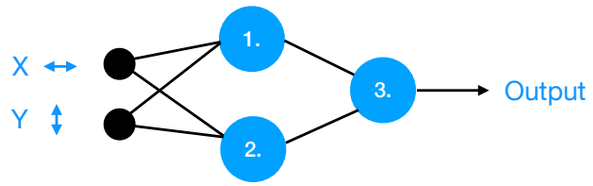


Die Berechnung des Tangens Hyperbolicus ist kompliziert. Wir brauchen nicht zu wissen, wie man das genau macht.

Wir sehen: Die Input-Zahlen können beliebig klein oder gross sein, die **Output-Zahlen** sind jedoch auf den **Bereich -1 bis 1 beschränkt**.

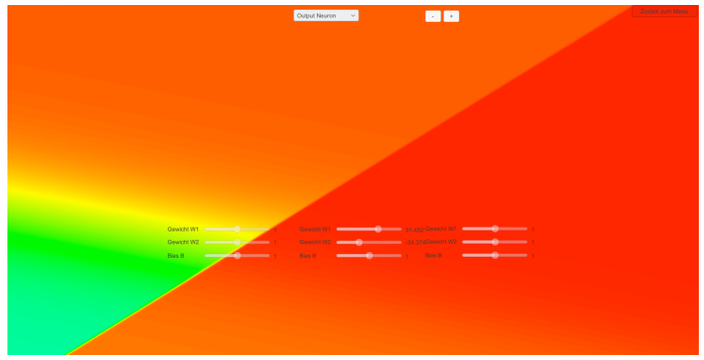
6. Ein einfaches neuronales Netz: Visualisierung





Für die folgende Visualisierung ordnen wir jedem Output-Wert zwischen -1 und 1 eine bestimmte Farbe zu.

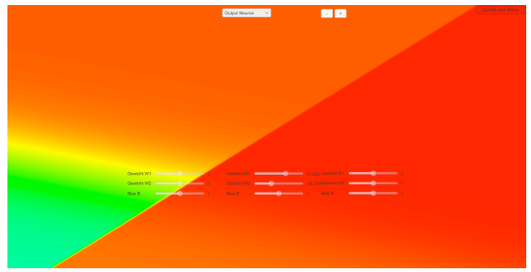
6. Ein einfaches neuronales Netz: Game



Spiele mit den Parametern der drei Neuronen und versuche ein möglichst interessantes Bild zu erzeugen.

Die SchülerInnen sollen erleben, dass es sehr schwierig ist, gezielt ein **bestimmtes** Bild durch Wählen von Parametern für die drei Neuronen zu erzeugen.
Interpretation im Kontext „Katze / Hund - Klassifikation“: Der Output-Wert des Output-Neurons entspricht nun der **Wahrscheinlichkeit**, dass das Tier ein Hund ist.

6. Ein einfaches neuronales Netz: Game - Schlussfolgerungen



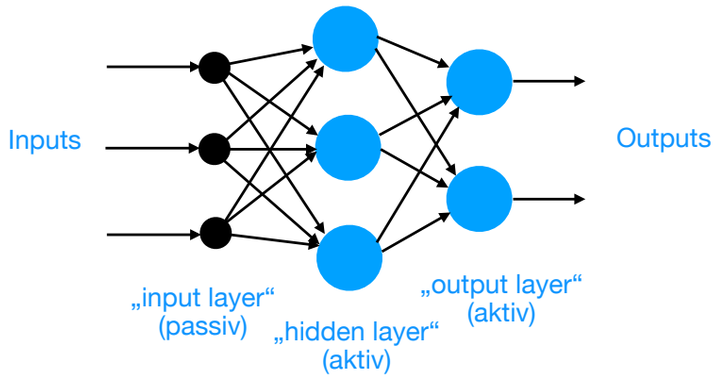
Das neuronale Netz kann, verglichen mit dem einzelnen Perzeptron, sehr viel mehr! Nun kann man mit richtigen Kurven arbeiten.

Es ist aber (zumindest für einen Menschen!) schwierig, **gezielt** ein bestimmtes Bild zu erzeugen.

Die Suche nach guten Parametern ist also ein schwieriges Problem!

Es wäre daher gut, wenn der Computer diese Arbeit für uns übernehmen könnte.

7. Neuronale Netze: Begriffe



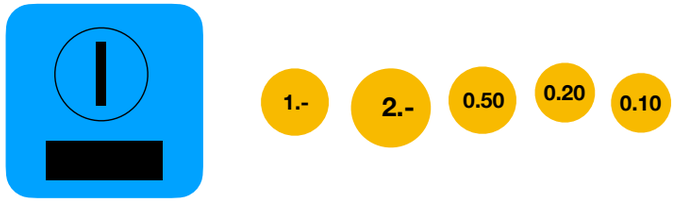
Grosse neuronale Netze können **viele tausend** Hidden-Layer haben
(und haben nicht selten **Millionen** von Neuronen)!

Das Input-Layer stellt nur die Eingangswerte bereit. Es besteht nicht aus Neuronen. Nur die blauen Punkte sind Neuronen und rechnen tatsächlich. Ein künstliches NN kann auch sehr viele Eingängen haben (z.B. Bildpunkte einer Videokamera).

8. „Der Fluch der vielen Dimensionen“: Gedankenexperiment

(✔)

Stell dir vor, du hast eine kleine Kiste, in welche du eine Münze einwerfen kannst:



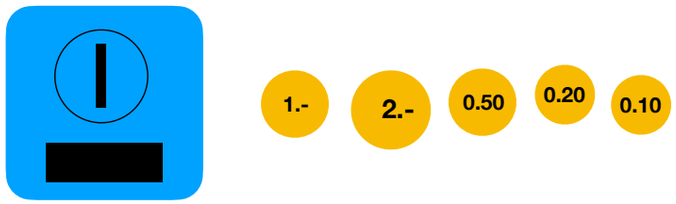
Je nach Münze, kommt ein mehr oder weniger wertvolles Geschenk heraus.

Wie oft musst du eine Münze einwerfen, wenn du wissen willst, bei welcher Münze das wertvollste Geschenk herauskommt?

Der folgende Teil ist optional. Es geht darum zu verstehen, weshalb überhaupt ein komplizierter Algorithmus zur Suche der Parameter erforderlich ist. Man könnte annehmen, dass eine systematische Suche durch alle möglichen Parameter-Kombinationen zum Erfolg (und sogar einem optimalen Ergebnis) führen kann. In den folgenden Folien wird gezeigt, dass die Anzahl der Parameter-Kombinationen sehr schnell mit der Anzahl der Parameter anwächst (exponentiell). Die Idee ist daher praktisch nicht durchführbar (KNN haben oft sehr viele Parameter!).

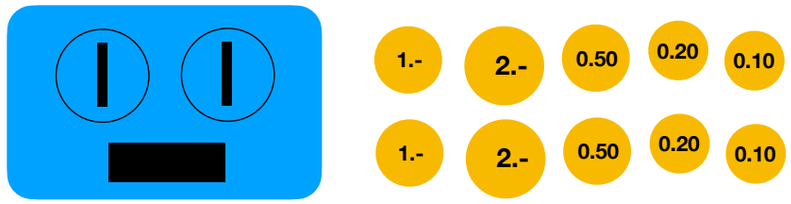
8. „Der Fluch der vielen Dimensionen“: Gedankenexperiment

(✔)



Das ist einfach: natürlich genau 5 mal!

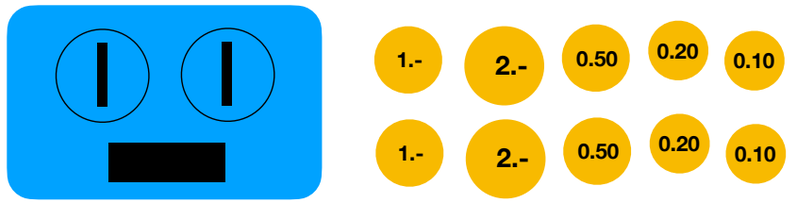
8. „Der Fluch der vielen Dimensionen“: Gedankenexperiment



Wenn die Kiste nun **zwei** Münzschlitze hat: Wie oft musst du zwei Münzen einwerfen, damit du sicher weisst, bei welcher Kombination das wertvollste Geschenk herauskommt?

8. „Der Fluch der vielen Dimensionen“: Gedankenexperiment

(✔)



Das ist schwieriger: Nun sind schon 25 Einwürfe nötig!

(Also viel mehr als die doppelte Anzahl, was nur 10 wären!)

Die Anzahl der erforderlichen Einwürfe wird mit jedem zusätzlichen Schlitz mit 5 multipliziert.

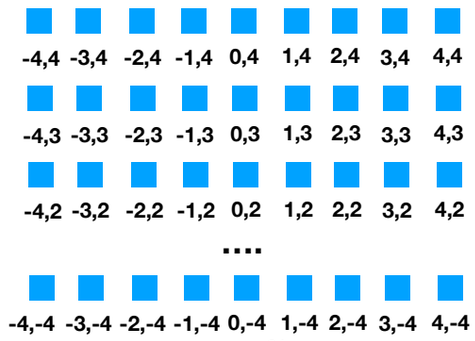
8. „Der Fluch der vielen Dimensionen“: Parameter-Suche ✔

Wir könnten einfach folgendermassen nach guten Kombinationen von Parametern suchen:
Wir probieren einfach sämtliche Möglichkeiten aus!

1 Parameter:

9 Tests

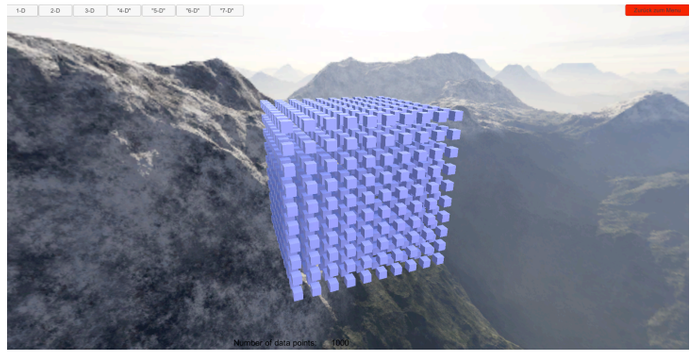
2 Parameter:



9*9 = 81 Tests !

Bemerkung:
Tatsächlich gibt es schon bei einem Parameter unendlich viele Möglichkeiten. Wir nehmen aber an, dass die Leistung des NN bei kleinen Änderungen der Parameter ebenfalls nur wenig ändert. Und wir betrachten nur Parameter innerhalb eines bestimmten Intervalls (im Beispiel -4 bis 4). Dann genügen einige wenige Werte.

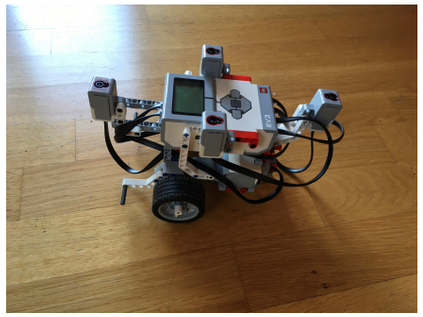
8. „Der Fluch der vielen Dimensionen“: Game



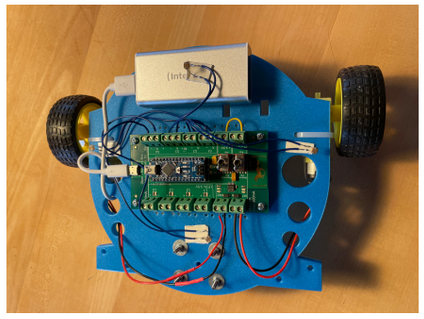
Mit jedem zusätzlichen Parameter verzehnfacht sich der Suchaufwand zum Finden der richtigen Parameter-Werte!

→ Der Suchaufwand wird sehr schnell **enorm** gross!

9. Motti - der lichtsuchende Roboter



System LEGO Mindstorms EV3



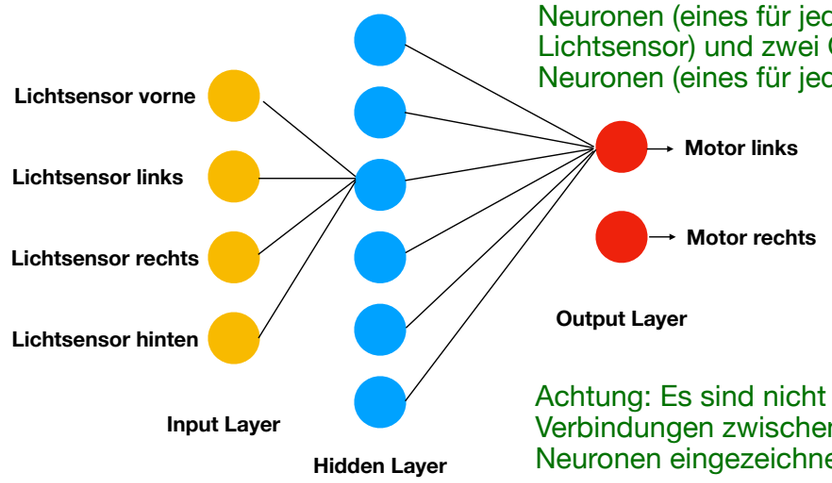
System PGLU (Arduino)

Testen des Roboters durch die SchülerInnen.

Der Roboter hat eine Steuerung, 4 Lichtsensoren und 2 Motoren.
Wo befinden sich diese Teile?

Der Roboter ist für diverse Systeme verfügbar. Siehe Download-Bereich auf ki-kit.ch (Bauanleitungen und Roboter-Code). Wir empfehlen das Roboterfahrzeug von PGLU (https://pglu.ch).
Es wird eine kräftige Taschenlampe benötigt. Demonstration im abgedunkelten Raum durchführen.
Es ist auch möglich, die Software für den Roboter konventionell zu programmieren. Wo liegen die Herausforderungen? Dies ist übrigens eine ideale Übung welche **vor** dem Kurs durchgeführt werden kann.

9. Motti - der lichtsuchende Roboter: neuronales Netz

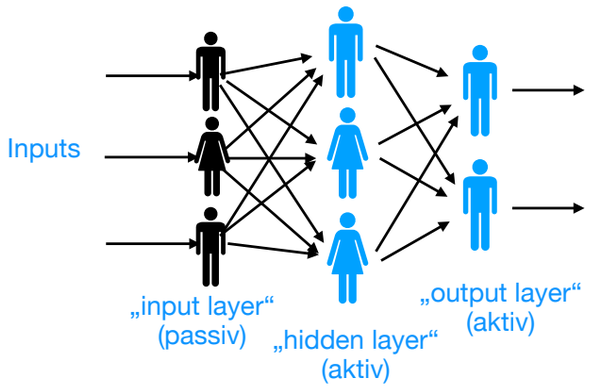


Das neuronale Netz hat vier Input-Neuronen (eines für jeden Lichtsensor) und zwei Output-Neuronen (eines für jeden Motor)

Achtung: Es sind nicht alle Verbindungen zwischen den Neuronen eingezeichnet!

Die beiden Outputs können negativ (= rückwärts fahren) oder positiv (= vorwärts fahren) sein.

9. Motti - der lichtsuchende Roboter: Game „Die Klasse als neuronales Netz“



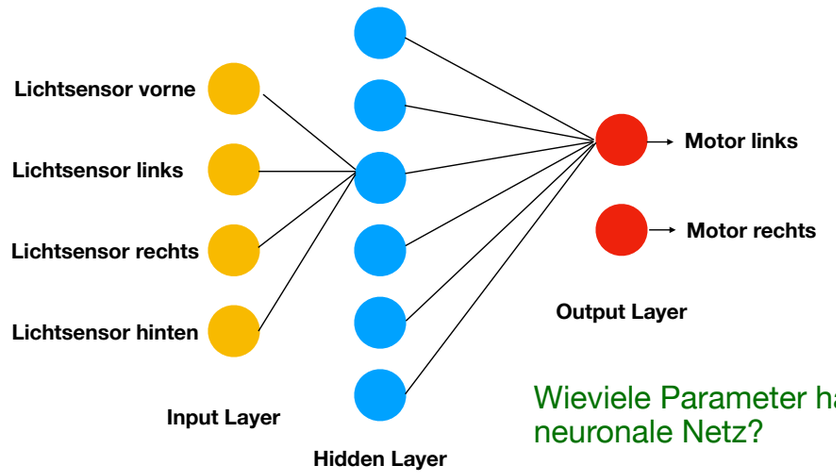
Berechnet den Output eines einfachen Motti-Netzes mit dem Taschenrechner in der Klasse!

Dabei berechnet jedes Kind den Output eines einzelnen Neurons.

(Zusätzliches Material! Siehe Download-Bereich ki-kit.ch)

Zeitbedarf ca. 1 Lektion. Siehe die zusätzlichen Unterlagen dazu auf https://ki-kit.ch.

9. Motti - der lichtsuchende Roboter: neuronales Netz

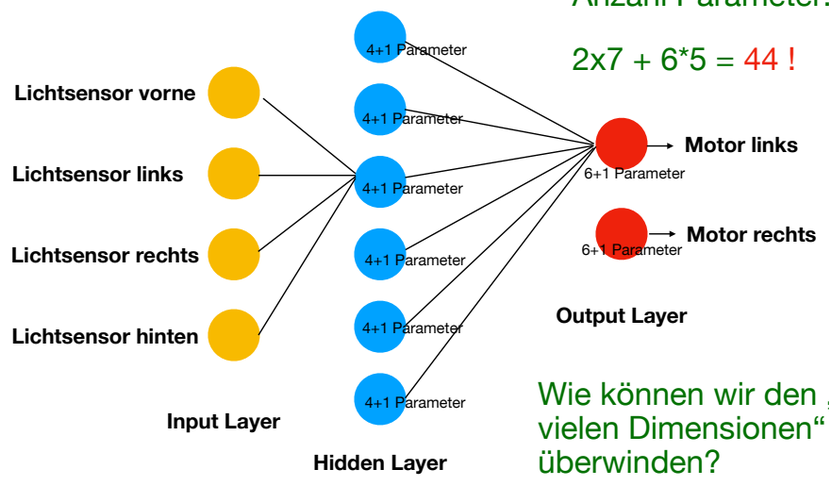


Wieviele Parameter hat dieses neuronale Netz?

Tip: Zuerst die Anzahl der Parameter in einem einzelnen Neuron für die verschiedenen Layer berechnen (Hidden und Output). Bias nicht vergessen!

Achtung: die Input-Neuronen sind keine richtigen Neuronen, sie haben daher keine Parameter.

9 Motti - der lichtsuchende Roboter: neuronales Netz - Lösung



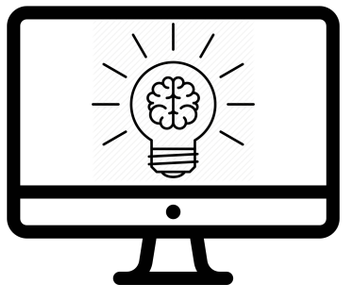
Anzahl Parameter:

2x7 + 6*5 = 44 !

Wie können wir den „Fluch der vielen Dimensionen“ überwinden?

Input-Layer: keine Neuronen, daher keine Parameter!

9. Motti - der lichtsuchende Roboter: neuronales Netz



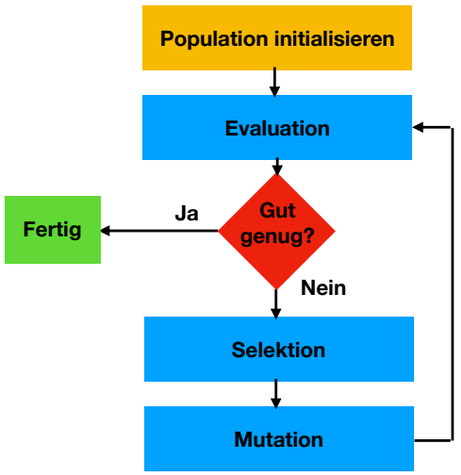
Wegen dem „Fluch der vielen Dimensionen“, muss der Computer bei der Suche der Parameter **schlau** vorgehen.

Wir benötigen also eine solche schlaue Rechenvorschrift.

Eine Rechenvorschrift für einen Computer nennt man allgemein **Algorithmus**.

Im Folgenden wird der genetische Algorithmus vorgestellt. In der heutigen Praxis wird dieser eher selten verwendet. Er ist aber für bestimmte Probleme durchaus leistungsfähig. Aktuell wird in den meisten Projekten „Backpropagation / gradient descent“ verwendet. Dieser Algorithmus kann aber, im Gegensatz zum viel einfacheren genetischen Algorithmus, nur mit höherer Mathematik (u.A. partielle Ableitungen) verstanden werden.
Es ist aber wichtig zu vermitteln, dass der vorgestellte Algorithmus nur einer von vielen ist.

10. Der genetische Algorithmus - Rechenvorschrift



Wir fangen mit z.B. 36 neuronalen Netzen an (wobei wir die Parameter jeweils zufällig wählen)

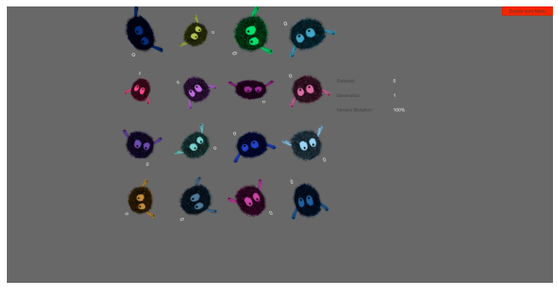
Wir testen diese in einem 3D Spiel (Computersimulation). Dabei messen wir, wie gut sie das „Licht“ einfangen können.

Wir behalten diejenigen 18 Netze, die sich in der Simulation am besten geschlagen haben

Aus diesen 18 Netzen erzeugen wir 18 ähnliche neue Netze → Nun haben wir wieder 36 Netze wie am Anfang!

In der Praxis wird der Algorithmus dann abgebrochen, wenn sich die Roboter nicht mehr wesentlich verbessern. Aus der finalen Population wird dann dasjenige neuronale Netz ausgewählt, das die höchste Anzahl von Generationen überlebt hat (sich also in vielen Situationen bewährt hat).

10. Der genetische Algorithmus - Game



1. Spiele mit den Parametern der Kreatur und überlege dir, mit welchen Parametern dir die Kreatur besonders gut gefällt.
2. Wähle im Spiel immer die in diesem Sinne besten Kreaturen aus (du musst genau die Hälfte der Kreaturen auswählen).

- Es wird nicht immer ein befriedigendes Resultat gefunden!
- Je grösser die Population gewählt wird, desto eher wird ein gutes Resultat gefunden.

Die im Spiel wählbaren Populationsgrössen sind (für die vorliegende Anzahl Parameter) bewusst klein gehalten. So werden die Grenzen des Algorithmus erlebbar.

10. Der genetische Algorithmus - Motti Trainer

Mit dieser Software (einer 3D Computersimulation) kannst du Motti trainieren, d.h. gute Werte für die Parameter des neuronalen Netzes finden.

Das Programm verwendet den genetischen Algorithmus.

Eine genaue Beschreibung der Roboter-Simulation ist im Lehrbuch enthalten.
Die Parameter des besten neuronalen Netzes können als Programmcode (Konstanten bzw. Arrays) für verschiedene Programmiersprachen heruntergeladen werden und auf den Roboter kopiert werden.

11. Schlussfolgerungen

- Der Roboter lernt nur während dem Training im virtuellen Raum, später nicht mehr! Für zusätzliche Fähigkeiten muss der Roboter neu trainiert werden.
- KI braucht (z.Z. noch) viele Beispiele (bzw. grosse Datenmengen) um zu lernen.
- Das Verhalten einer KI ist nicht so klar definiert, wie bei einem klassisch programmierten Algorithmus. Insbesondere machen KI's auch Fehler!
- Es ist praktisch unmöglich, aus den KNN-Parametern auf das Verhalten zu schliessen (Parameter = viele Zahlen, kann man nicht „lesen“ wie ein Computerprogramm).