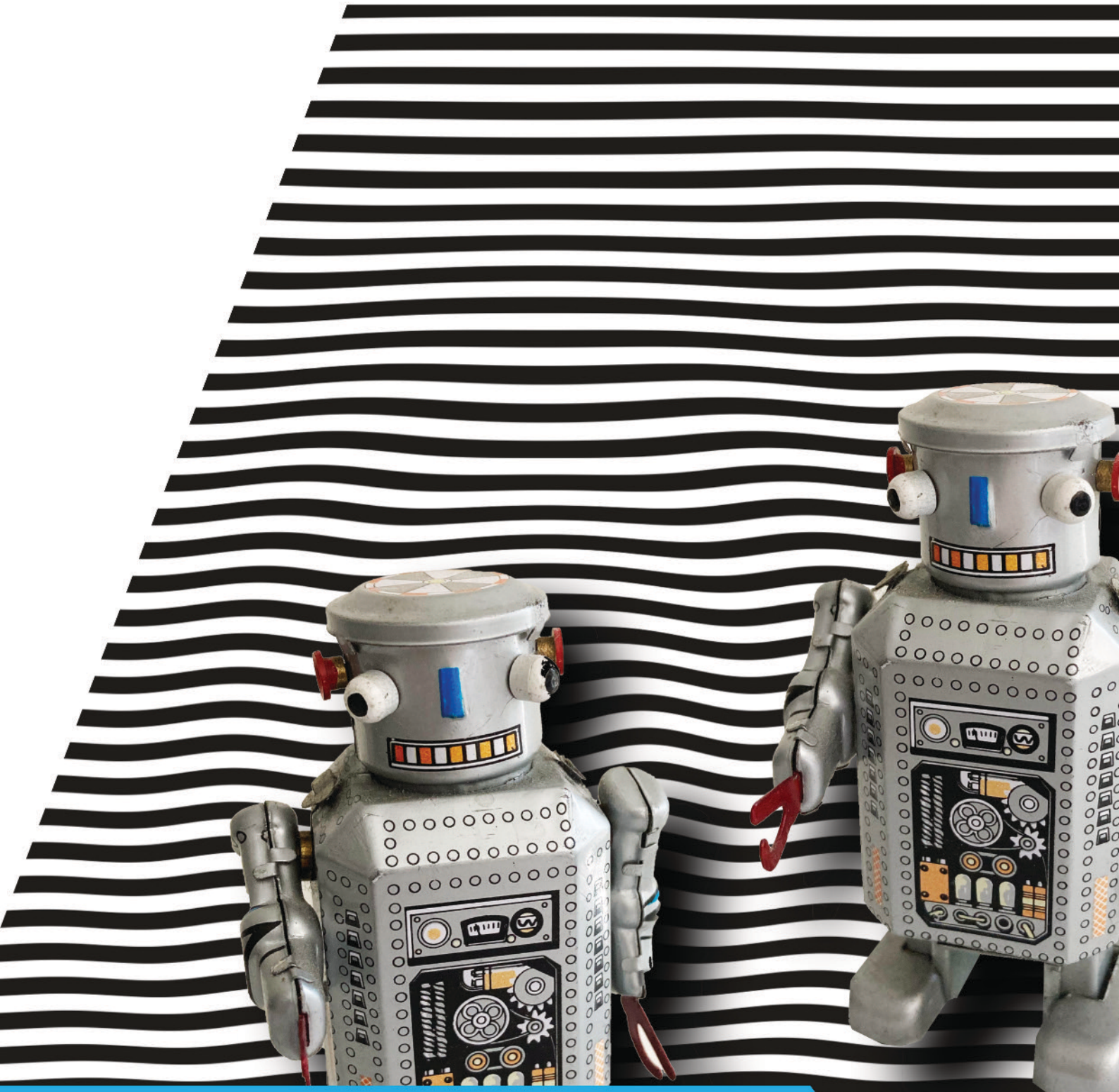


Ein Lehr- und Bastelbuch für Jugendliche
ab 13 Jahren

Künstliche Intelligenz und Robotik

Inkl. Lernspielen!

Version 1.0.5

Text und technische Illustrationen: Marco Lardelli
Ganzseitige Illustrationen: Dieter Kubli
Lektorat: Anne-Catherine Eigner

Inhalt

Einführung

Ein Wort an die Eltern

Schon seit vielen hundert Jahren versucht der Mensch das Geheimnis der menschlichen Intelligenz zu lüften. Dabei wurde auch immer wieder versucht, denkende Maschinen zu konstruieren. Neben dem Wunsch des Menschen, sich selbst zu verstehen, ging es dabei auch darum, Maschinen zu konstruieren, die dem Menschen mühsame oder schwierige Arbeiten abnehmen können. Solche Roboter sind heute aus unseren Fabriken kaum mehr wegzudenken. Dabei wurden in den letzten Jahren enorme Fortschritte erzielt: Computer und Roboter erreichen heute dank künstlicher Intelligenz kognitive Fähigkeiten, die es in vielen Disziplinen mit denjenigen des Menschen aufnehmen können. Dabei sind die Erfolge keineswegs nur auf die klassischen Computer-Domänen wie z. B. Rechnen beschränkt, sondern zeigen sich immer mehr auch in Fähigkeiten, die man bis vor kurzem noch ausschliesslich dem Menschen zugestanden hat (z. B. Verständnis von Sprache etc.).

Dieses Buch unternimmt (zusammen mit der dazugehörigen umfangreichen Software) den Versuch, einige der hinter diesen erstaunlichen Erfolgen stehenden zentralen Ideen und Konzepte Jugendlichen **ab ca. 13 Jahren** zugänglich zu machen. Dabei werden zuerst die mathematischen und technischen Grundlagen ausführlich erklärt. Erst dann, mit dem entsprechenden Verständnis als Rüstzeug, werden Anwendungen, Ethik, Sicherheit und Philosophie der künstlichen Intelligenz diskutiert.

Ein Wort an die Jugendlichen

Wenn du anfängst, dieses Buch zu lesen, wirst du schon sehr bald auf erste Übungen stossen. Bitte versuche unbedingt, die Übungen zu lösen, denn sie sind für das Verständnis äusserst wichtig. Künstliche Intelligenz kann man nicht erlernen, ohne selber aktiv zu werden, d. h. ohne viel zu üben und selber nachzudenken wirst du die Konzepte nicht verstehen können. Die Lösungen zu den Übungen enthalten zudem auch wichtige Erklärungen.
Natürlich wird es dir nicht in jedem Fall gelingen, die Übungen ganz richtig zu lösen oder du hast anfangs sogar keine Ahnung, wie du die Aufgabe anpacken sollst. Mach dann einfach folgendes:

1. Lies die Theorie noch einmal aufmerksam durch und versuche, sie besser zu verstehen.
2. Denke noch einmal über die Fragen nach und versuche die Übung erneut zu lösen.
3. Falls es dir immer noch nicht gelingt, nimm dir unbedingt die Zeit, die Lösung im Lösungsbuch genau zu studieren und fahre erst dann mit dem Kurs fort, wenn du sicher bist, dass du sie verstanden hast.

Das Buch enthält Anleitungen zum Bau eines Roboters, den du mittels KI selber trainieren kannst. Der Bau des Roboters ist zum Verständnis dieses Buches nicht zwingend notwendig, er ist aber sehr hilfreich und empfohlen (insbesondere wenn du dich neben KI auch für Robotik interessierst). Mit dem Roboter kann man natürlich auch viel Spass haben.

Die Software zu diesem Buch

An vielen Stellen in diesem Buch werden wichtige Zusammenhänge mittels Computerspielen erklärt. Auch für das Trainieren deines Roboters wird eine Software benötigt. Du kannst diese sogenannten “Serious Games” in einem normalen Web-Browser (z. B. Firefox, Safari oder Chrome) “spielen”. Du findest die Spiele unter der folgenden Web-URL:

https://ki-kit.ch/games

Bitte beachte, dass im Browser “WebGL” aktiviert sein muss. Du kannst unter folgender URL testen, ob dein Browser dafür richtig konfiguriert ist:

https://get.webgl.org (du solltest einen sich drehenden Würfel sehen)

Für die verschiedenen Roboter (LEGO Mindstorms, micro:bit, Arduino) wird ebenfalls eine kleine Software benötigt. Du kannst diese unter der folgenden URL herunterladen:

https://ki-kit.ch (Download-Bereich)

Bevor du dich in den Bau und die Programmierung von Robotern stürzt, solltest du nachschauen, ob es eine aktualisierte Version der Bauanleitung gibt. Du findest die aktuellste Version immer unter der folgenden URL:

https://ki-kit.ch (Download-Bereich)

Einen KI-Roboter selber bauen

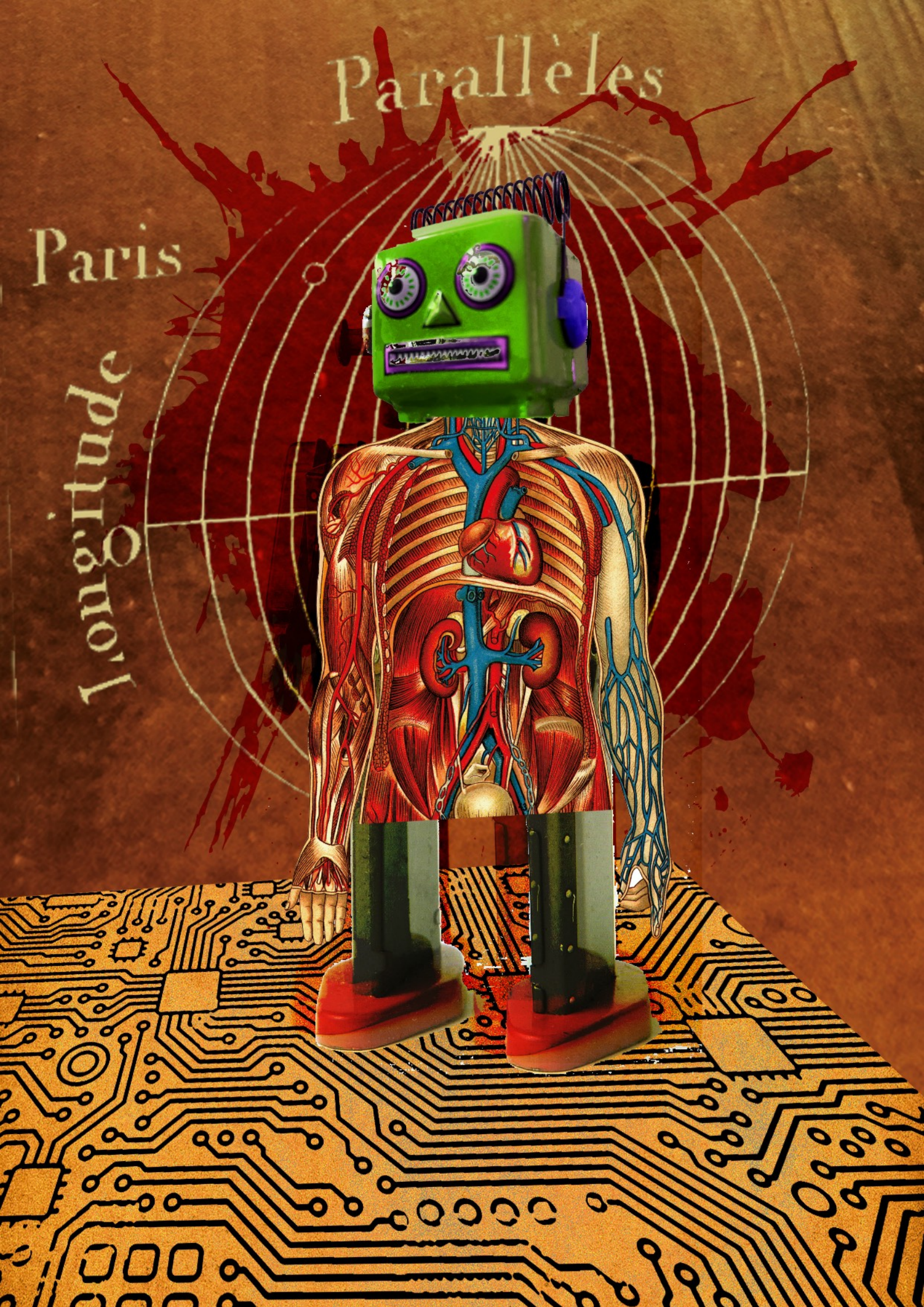
Mit Hilfe der Konzepte, Anleitungen und Software in diesem Buch, kannst du einen Roboter mit künstlicher Intelligenz selber bauen und trainieren. Den Roboter kann man auf unterschiedliche Weise bauen, die Funktion ist aber bei allen Varianten gleich. Es stehen folgende Systeme zur Auswahl:

- Roboter auf Basis “LEGO Mindstorms EV3”: Diese Variante ist am einfachsten zu bauen, aber auch die teuerste (ca. 400 EUR). Dieser Roboter wird mit der visuellen Programmiersprache Microsoft MakeCode (bzw. JavaScript) programmiert.

- Roboter auf Basis “micro:bit” (ein Einplatinen-Microcomputer für Lernzwecke): Günstiger als der LEGO-Roboter (ca. 100 EUR), aber auch anspruchsvoller zu bauen. Programmiersprache: ebenfalls Microsoft MakeCode bzw. JavaScript.

- Roboter auf Basis “Arduino”: Kann sehr günstig realisiert werden (ca. 30-50 EUR), ist aber deutlich schwieriger zu bauen als der micro:bit-Roboter oder gar der LEGO-Roboter. Die Programmiersprache ist dann Arduino C.
  Geeignet für die Älteren unter den Lesern (oder wenn deine technisch versierten Eltern dir manchmal etwas helfen können).

Einen Vergleich der verschiedenen Roboter (inkl. der benötigten Bauteile) findest du am Anfang von Anhang A (d. h. im Buch mit den Bauanleitungen).
Ein Roboter auf Basis von „LEGO Mindstorms Robot Inventor“ befindet sich in Planung.


Parallèles
Paris
Longitude

1. Was ist Intelligenz?

1.1 Definitionen

Der Begriff „Intelligenz“ macht es uns nicht einfach. Zwar meint jeder zu wissen, wovon die Rede ist, aber eine exakte Definition zu geben ist schwierig. So haben im Laufe der Zeit viele Wissenschaftler versucht, den Begriff zu beschreiben. Diese Definitionen unterscheiden sich z. T. erheblich.

Hier einige Definitionen von Wissenschaftlern (und aus Lexika):

A. „Intelligenz ist, was ein Intelligenztest misst“ (E. Boring, 1923)

B. „Die Fähigkeit, abstrakt zu denken” (L. Terman, 1921)

C. „Die Fähigkeit, schwierige Probleme zu lösen“ (M. Minsky, 1985)

D. „Intelligenz misst die Fähigkeit eines Agenten, Ziele in einem breiten Spektrum von Umgebungen zu erreichen“ (S. Legg & M. Hutter, 2007)

E. „…in einem breiten Spektrum von Aufgaben die Sache gut machen, ist eine empirische Definition von Intelligenz” (H. Masum, 2002)

F. „Die Fähigkeit, Kenntnisse zu erwerben und anzuwenden” (Compact Oxford English Dictionary, 2006)

ÜBUNG 1.1-1:

Welche der folgenden natürlichen und technischen Systeme erfüllen welche der obigen Definitionen des Begriffes „Intelligenz“ (du musst nicht unbedingt alles ausfüllen)? (Nur ankreuzen wo erfüllt, ein Kreuz in Klammern „(X)“ bedeutet „teilweise/bedingt erfüllt“):

	A	B	C	D	E	F
Smartphone						
Mensch						
Taschenrechner						
Schachcomputer						
Lernfähiger Roboter-Staubsauger						
Hund						

ÜBUNG 1.1-2:

Worin unterscheiden sich tendenziell die älteren (d. h. 20. Jahrhundert) Definitionen von den jüngeren (21. Jahrhundert)? Nimm dazu die obige Tabelle zur Hilfe.

Was könnte der Grund (oder die Gründe) sein für diese Unterschiede?

1.2 Arten von Intelligenz, verwandte Begriffe

ÜBUNG 1.2-1:

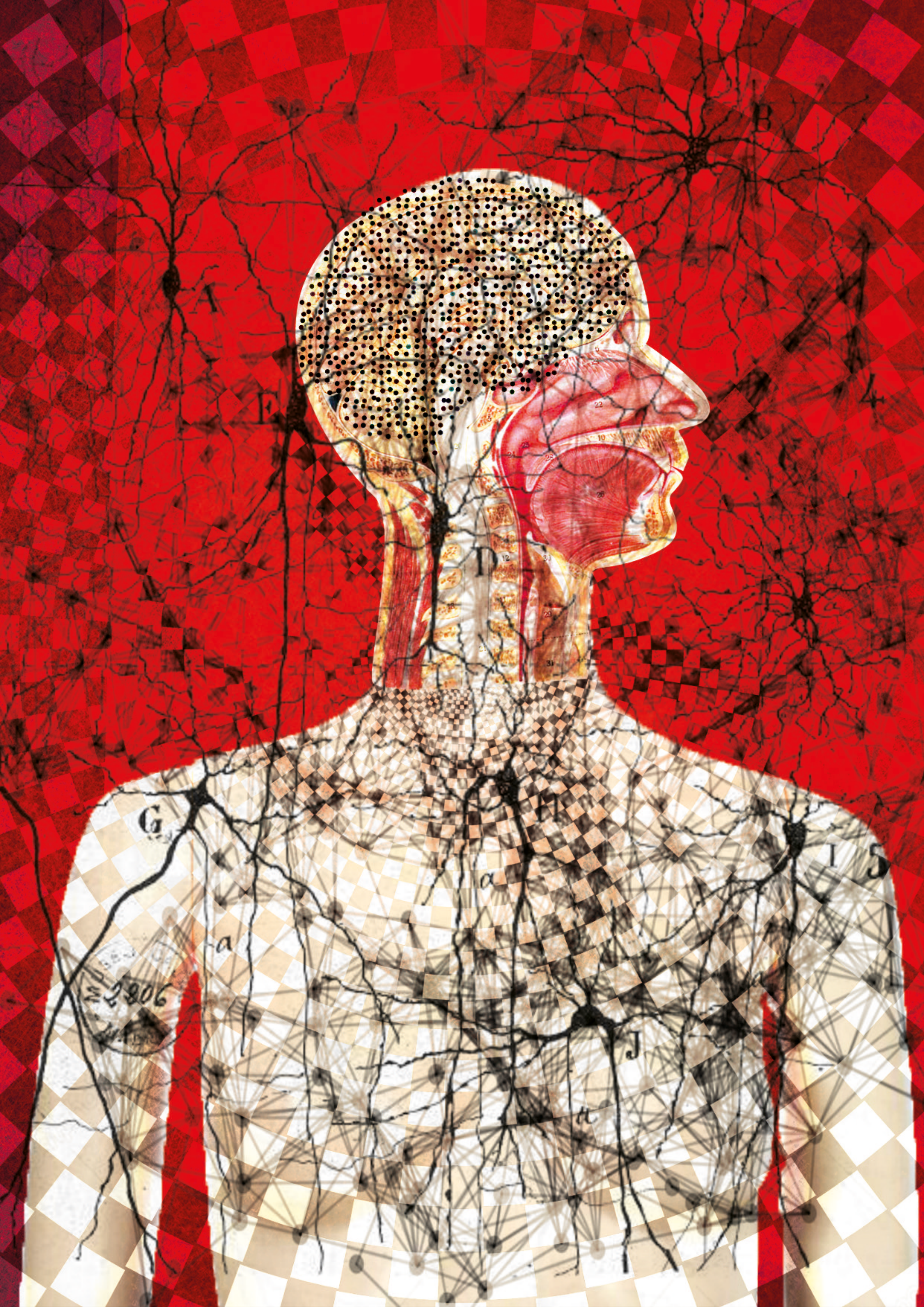
Man unterscheidet beim Menschen verschiedene Arten von Intelligenz. Was könnten die folgenden Begriffe bedeuten?

Rationale Intelligenz	
Emotionale Intelligenz	
Soziale Intelligenz	
Motorische Intelligenz	

ÜBUNG 1.2-2:

Folgende Begriffe haben etwas mit Intelligenz zu tun. Was sind die Unterschiede und Verbindungen zum Begriff der Intelligenz?

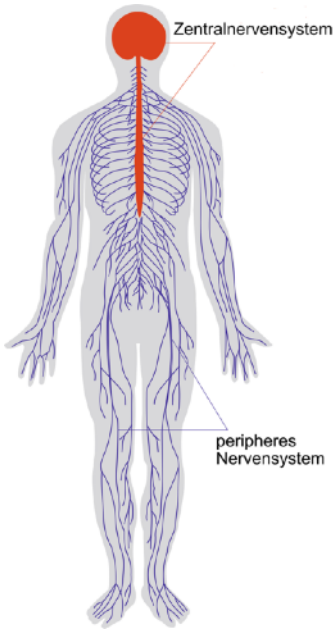
Genie	
Talent	
Kreativität	
Wissen	
Bildung	
Begabung	



2. Intelligenz in der Natur: Nervensystem und Gehirn

2.1 Das Nervensystem und seine Unterteilungen

Man unterteilt das gesamte Nervensystem in das **Zentralnervensystem** (bestehend aus Gehirn und Rückenmark) und das **periphere Nervensystem** (alles was nicht zu Gehirn und Rückenmark gehört). Das periphere Nervensystem verbindet das Zentralnervensystem (ZNS) mit der Umwelt: Es leitet Informationen von den Sinnesorganen ins ZNS weiter und Informationen über die Steuerung von Körperfunktionen und Bewegungen aus dem ZNS in den Körper.



Zusätzlich unterteilt man das Nervensystem in ein **willkürliches** und ein **vegetatives** Nervensystem ein. Das willkürliche Nervensystem steuert dem Willen unterworfene Vorgänge (z. B. Bewegungen der Extremitäten), während das vegetative Nervensystem der willentlichen Kontrolle entzogen ist. Es steuert Funktionen der inneren Organe, wie z. B. die Bewegung des Darmes bei der Verdauung.

ÜBUNG 2.1-1:

Werden die folgenden Körperfunktionen durch das willkürliche oder das vegetative Nervensystem gesteuert?

	Willkürlich	Vegetativ
Atmung		
Herzschlag		
Greifen mit der Hand		
Gehen		
Regelung des Blutdruckes		

ÜBUNG 2.1-2:

Das sog. sensorische System umfasst die menschlichen Sinne. Welche Sinne kennst du?

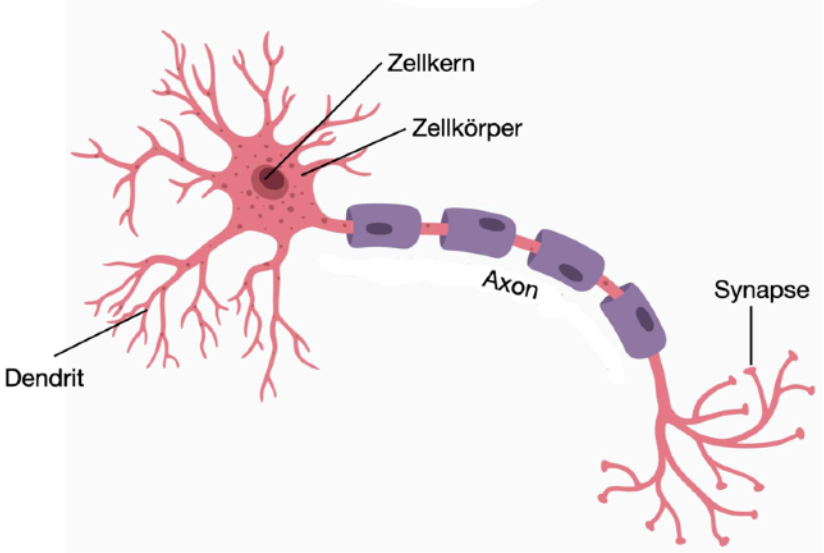
Das **Nervensystem** des Menschen besteht aus ca. 90 Milliarden einzelnen **Nervenzellen** (sog. **Neuronen**), die miteinander kommunizieren (d. h. Informationen austauschen).

ÜBUNG 2.1-3:

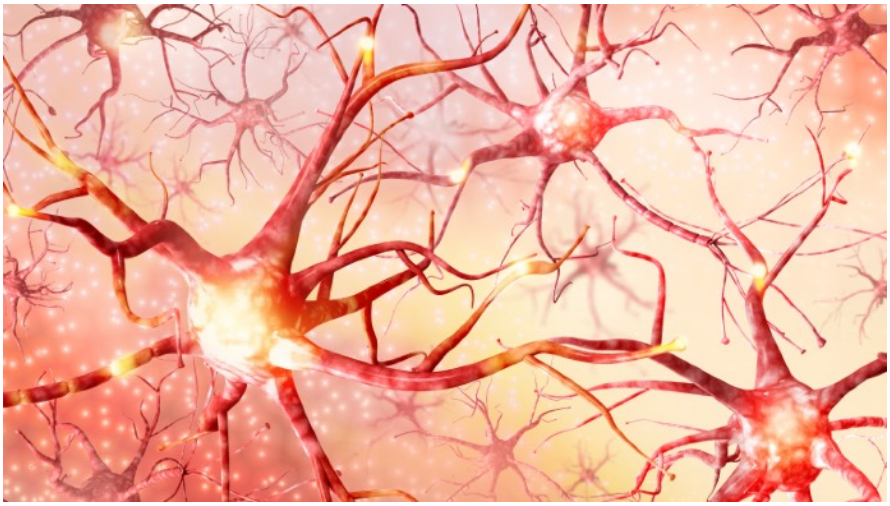
Schätze die Anzahl von Nervenzellen (Neuronen) im gesamten Nervensystem verschiedener Tiere:

A) Nematode (primitiver Wurm)		B) Fruchtfliege	
C) Meerschweinchen		D) Katze	
E) Bonobo (Menschenaffe)			

Neuronen können unterschiedliche Formen haben und auch sehr unterschiedlich gross sein. Der grundlegende Aufbau ist aber bei allen gleich:



Sie bestehen aus einem **Zellkörper**, welcher 0.01 bis 0.1 mm gross sein kann. Von diesem gehen zahlreiche Verbindungen zu bis zu 10’000 (!) anderen Nervenzellen. Jede Nervenzelle empfängt Signale über ihre **Dendriten**. Im Zellkörper werden dann die erregenden und die hemmenden Signale miteinander verrechnet (summiert). Wird dabei ein bestimmter Schwellwert überschritten, „feuert“ das Neuron, d. h. es gibt ein Signal über sein (einziges) **Axon** an andere Nervenzellen weiter. An den Enden von Dendriten und dem Axon befinden sich sog. **Synapsen**, welche die Verbindung zu anderen Nervenzellen herstellen.



Die Gesamtheit aller Verbindungen im Nervensystem eines Lebewesens nennt man **Konnektom**. Aufgrund der Vielzahl von Nervenzellen und der vielen Verbindungen pro Neuron, ist das Konnektom vieler Lebewesen enorm komplex. Bisher ist es Forschern gelungen, das Konnektom primitiver Lebewesen (z. B. eines Fadenwurms mit 300 Neuronen und 5000 Synapsen) im Computer zu rekonstruieren. Das menschliche Gehirn besitzt ca. 100 Billionen Synapsen (100.000.000.000.000), weshalb unser Konnektom bisher nicht rekonstruiert werden konnte.


in China bei einem Pilotp hnhof von
Zhengzhou die inzei Sonn lie t ein
Ge serkennun rfolgt zt. rhalb von
Sekunden werde iner
brecherkarte i f ein mobilen,
Tablet-ähnlichen C lächtige angezeigt. Auf
se Weise wurden bereits sieben Krimin
festgenommen.[13][14]
Die chinesische Firma Watrix hat im November 2018
eine KI vorgestellt, die
50 Metern

3. Künstliche Intelligenz (KI)

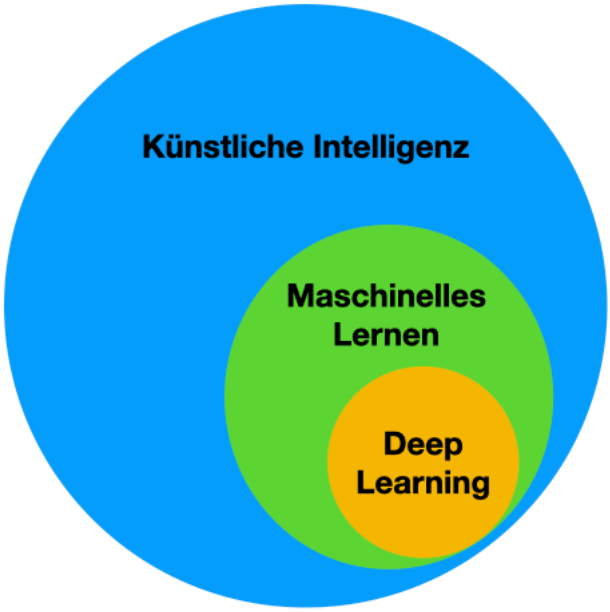
3.1 Begriffe

Der Begriff „**Künstliche Intelligenz**“ (engl. „**Artificial Intelligence**“) bezeichnet alle technischen Systeme, welche über Intelligenz verfügen. Wir haben gesehen, dass schon über den Begriff der Intelligenz keine Einigkeit besteht. Der Begriff der KI ist also ebenfalls nicht klar abgrenzbar.

Der Begriff „**Maschinelles Lernen**“ (engl. „**Machine Learning**“) bezeichnet intelligente technische Systeme, die aus Daten lernen können.

„**Deep Learning**“ (es ist keine deutsche Übersetzung verfügbar) bezeichnet eine spezielle Methode, lernfähige Systeme zu bauen, die seit ca. 2013 grosse Erfolge feiert. Dabei werden sogenannte tiefe neuronale Netze verwendet. Wir werden später lernen, was es damit auf sich hat.

Die Begriffe künstliche Intelligenz, maschinelles Lernen und „Deep Learning“ stehen zueinander in folgender Beziehung:



D. h. „Deep Learning“ ist auch maschinelles Lernen und dieses wiederum gehört zur künstlichen Intelligenz. Die blaue Fläche steht also für intelligente Systeme, die nicht lernen können. Die grüne Fläche steht für lernfähige intelligente Systeme, welche nicht mit tiefen neuronalen Netzen realisiert sind.

ÜBUNG 3.1:

Versuche die folgenden Systeme der korrekten Zone (Farbe) zuzuordnen:

A) Schachcomputer (1985)

B) Automatische Erkennung von Personen auf Facebook (2020)

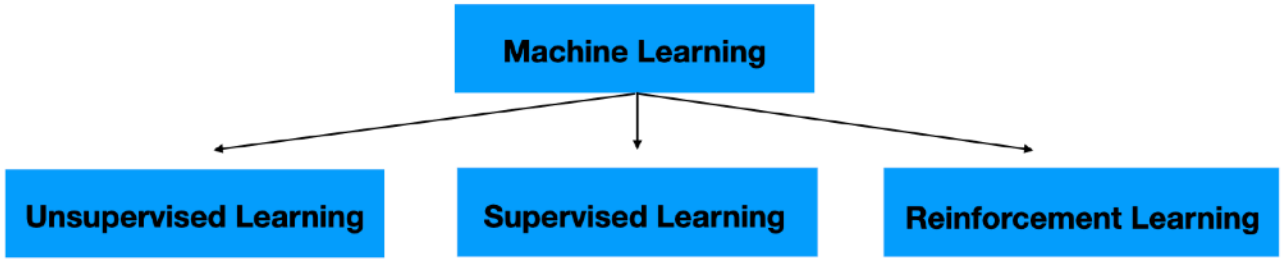
C) Diktiersoftware (Sprache zu Text konvertieren, stellt sich automatisch auf die Stimme des Anwenders ein) (2000)

D) Diktiersoftware (Sprache zu Text konvertieren, stellt sich automatisch auf die Stimme des Anwenders ein) (2020)

E) Smartphone-App, welche einem „Selfie“ einen Schnauz hinzufügt (2019)

3.2 Arten von „Machine Learning“

Beim Machine Learning (wir verwenden im Folgenden meistens den englischen Begriff, da dieser auch im Deutschen häufiger verwendet wird) werden mehrere Arten unterschieden:



A. **Unüberwachtes Lernen** (engl. **„Unsupervised Learning“**)

   Der Computer findet von selbst (d. h. ohne irgendwelche Hilfe) interessante Strukturen in den Daten.

B. **Überwachtes Lernen** (engl. **„Supervised Learning“**)

   Dem Computer werden zuerst verschiedene Eingaben (engl. „**Inputs**“) mit den dazugehörigen richtigen Ausgaben („**Outputs**“) gegeben. Daraus lernt der Computer die Beziehungen zwischen Eingaben und Ausgaben. Der Computer ist dann in der Lage, zu einer noch unbekannten Eingabe die korrekte Ausgabe zu berechnen.

C. **Bestärkendes Lernen** (engl. **„Reinforcement Learning“**)

   Um zu lernen, erhält der Computer, je nach korrektem oder falschem Verhalten, einen positiven (Belohnung) oder negativen (Bestrafung) sog. „**Reward“** (engl. für Belohnung).

ÜBUNG 3.2:

Bestimme für die folgenden Anwendungen die „Machine Learning“ Art:

A. Eine Software soll lernen, auf Fotos Hunde und Katzen zu unterscheiden. Dazu lernt die Software mit Hunderten von Bildern von Hunden und Katzen die entsprechend bezeichnet sind (also z.B. katze_123.jpg, hund_321.jpg).

B. Eine Firma lässt ihre Kundendaten (eine Tabelle mit Spalten, wie z.B. Jahresumsatz, Geschlecht, Geburtsjahr, Datum des ersten Kaufes etc.) durch eine Software in Gruppen von ähnlichen Kunden einteilen.

C. Eine Software soll lernen, an der Börse zu handeln. Als Feedback zum Lernen soll die Software den Gewinn der getätigten Transaktionen (d. h. Käufe und Verkäufe) verwenden.

D. Ein Roboter soll laufen lernen. Als Feedback zum Lernen soll der Roboter lediglich die Distanz, die er in 30 Sekunden zurückgelegt hat, verwenden.

E. Eine Software soll lernen, den Kunden eines Online-Shops möglichst Produkte vorzuschlagen, die ihnen gefallen könnten.

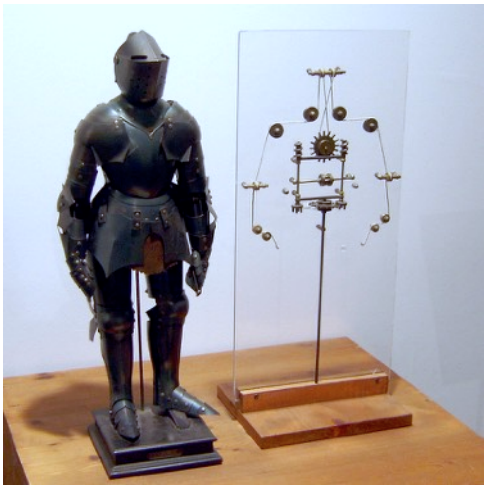

ASIMOV
MONDAY
May
11
TUESDAY
May
12
401
Joseph Weizenbaum
WEDNESDAY
May
13
WALTER PITTS
week 20
THURSDAY
May
14
0945
KONRAD ZUSE
SUNDAY
May
10
AS LEM

4. Die Geschichte der künstlichen Intelligenz

4.1 Mythen, Automaten und Philosophen

Schon früh träumten die Menschen von selbst erschaffenen Maschinenwesen. So kommt z. B. in der jüdischen Folklore seit ca. dem 12. Jahrhundert der **Golem** vor. Eine von Menschen aus Lehm geformte Kreatur, die zum Leben erweckt werden konnte, indem man ihr ein mit dem Namen Gottes beschriftetes Papier in den Mund steckte.

Das Universalgenie **Leonardo da Vinci** (1452-1519) erfand einen mechanischen Ritter:



Der Schweizer Alchemist **Paracelsus** (1493-1541) beschrieb eine Methode zur Erschaffung eines künstlichen Menschen, des sog. **Homunculus**, wobei allerlei unappetitliche Zutaten zum Einsatz kommen sollten.

Im 17. Jahrhundert dachten die Mathematiker und Philosophen **Leibnitz, Hobbes** und **Descartes** darüber nach, wie sich das rationale Denken systematisch beschreiben liesse.

Wolfgang von Kempelen konstruierte im späten 18. Jahrhundert einen mechanischen Schachautomaten. Der Automat war jedoch eine Illusion (man könnte auch sagen ein Betrug), denn im Inneren versteckte sich ein menschlicher Schachmeister, der den Automaten bediente.

4.2 Die Anfänge der modernen KI

1936 bewies der Mathematiker **Alan Turing**, dass eine einfache Maschine in der Lage ist, beliebig komplexe Algorithmen (d. h. Rechenvorschriften) auszuführen. Damit hat er ein wichtiges theoretisches Fundament für die künstliche Intelligenz gelegt.

1950 publizierte Turing eine Schrift, in der er über die Möglichkeit denkender Maschinen spekulierte. Er schlug den sog. **Turing-Test** vor, mit welchem man entscheiden kann, ob ein Computer wie ein Mensch denken kann.

Als eigentliche Geburtsstunde der KI gilt die Dartmouth-Konferenz von 1956, welche von den KI-Pionieren Marvin Minsky, John McCarthy, Claude Shannon und Nathan Rochester organisiert wurde. Während dieser Konferenz einigten sich die Teilnehmer auf den Begriff „Artificial Intelligence“ für das neue Fachgebiet.

Ende 1960 entwickelte **Joseph Weizenbaum** den Chatbot **ELIZA**, der einen Psychotherapeuten simulierte. Trotz des simplen Aufbaus des Programms, war die Illusion erstaunlich gut und viele Menschen glaubten tatsächlich, ELIZA sei intelligent.

Anfang der 70er Jahre wurden die ersten **Expertensysteme** entwickelt. So z.B. MYCIN, das Ärzte bei Therapieentscheidungen bei Blutinfektionen unterstützen sollte.

Es folgte der sog. **erste KI-Winter** (1974 bis 1980), in welchem kaum wesentliche Fortschritte erzielt wurden.

Ab 1980 wurden grosse Summen in die Entwicklung von Expertensystemen gesteckt. 1981 genehmigte die japanische Regierung unter dem Namen „Fifth generation computer project“ 850 Millionen US-Dollars für die Entwicklung von KI
Es stellte sich jedoch heraus, dass es meistens sehr schwierig war, Expertenwissen in solche Systeme zu transferieren. Das Projekt scheiterte und es folgte der **zweite KI-Winter** (1987-1993).

Im Jahre 1997 gewann der IBM-Computer „Deep Blue“ gegen den Schach-Weltmeister Garry Kasparov.

4.3. Das 21. Jahrhundert

Ab ca. 2011 erfolgte die stürmische Entwicklung des „Deep learning“. Dank der massiv gestiegenen Rechenleistung und verfügbaren Datenmengen sowie einigen neuen Ideen, konnten ältere Konzepte (wie z. B. das künstliche Neuron) wiederbelebt und zu grossem Erfolg gebracht werden. Es folgte ein Boom, den die KI bisher noch nicht erlebt hatte. Seitdem werden jedes Jahr riesige Summen in die Entwicklung künstlicher Intelligenz investiert. Wichtige Beiträge kommen u. a. von den „Deep learning“ Pionieren **Geoffrey Hinton**, **Yann LeCun**, **Yoshua Bengio**, **Andrew Ng** und **Jürgen Schmidhuber**.
2016 schlug die Software **AlphaGo** den südkoreanischen Go-Meister Lee Sedol. Selbst für viele Experten galt das Spiel Go noch vor kurzem als viel zu komplex für künstliche Intelligenz.
Schon 2017 stellte die Google-Firma DeepMind die Software AlphaZero vor. Diese lernte innert weniger Stunden die Spiele Schach, Go und Shogi mit tief übermenschlicher Leistungsfähigkeit.

ÜBUNG 4.3-1:

Finde heraus, was es mit dem **Turing-Test** auf sich hat (z. B. indem du mit einer Suchmaschine eine Beschreibung suchst).

A. Wie funktioniert der Test?

B. Welche Fähigkeiten werden geprüft?

C. Ein CAPTCHA ist ein kleines Rätsel, dass Menschen auf Webseiten lösen müssen, um sich als Mensch zu identifizieren (z. B. eine Zahl lesen und eintippen). Das sog. Akronym bedeutet: "**C**ompletely **A**utomated **P**ublic **T**uring test to tell **C**omputers and **H**umans **A**part".
   Das CAPTCHA wird auch ein **inverser Turing-Test** genannt. Warum?

ÜBUNG 4.3-2:

Suche mit einer Suchmaschine eine Online-Version von ELIZA (Stichworte z. B. „eliza" und „therapist") und führe einen ernsthaften Dialog mit „ihr".
Du musst natürlich Englisch schreiben!

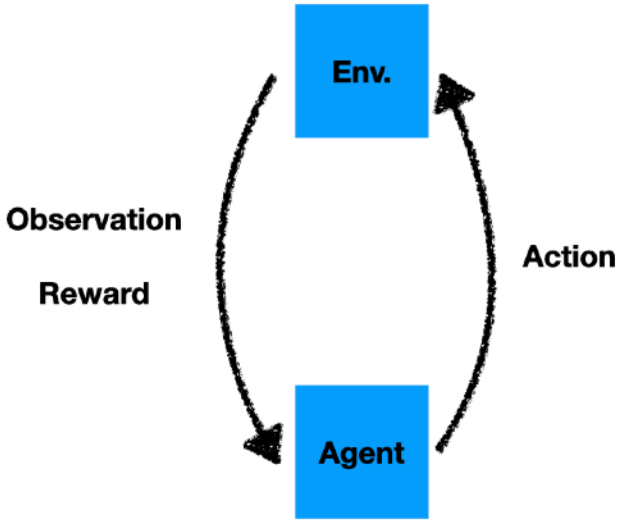
A. Woran merkst du, dass ELIZA nicht wirklich intelligent ist?
B. Was waren Weizenbaums Tricks, um das Programm intelligent erscheinen lassen?


Pôle Septentrional
Hauteur du Pôle
Cerc. Pol
Colure
Cancer
Tropiq

5. Reinforcement Learning

5.1 Problemstellung und Begriffe im RL

Im „Reinforcement Learning“ betrachtet man einen sogenannten Agenten (engl. „**Agent**“), welcher sich in einer Umgebung (engl. „**Environment**“) aufhält und dabei verschiedene Aktionen (engl. „**Actions**“) ausführen kann. Dieser Agent kann z. B. ein Roboter sein, der sich in einem Fabrikraum bewegt und Arbeiten ausführt oder eine Software, die versucht, an der Börse Gewinn zu machen.
Um entscheiden zu können, welche Aktion in einem bestimmten Zustand der Umgebung richtig ist, kann der Agent die Umgebung beobachten. Dem Agenten wird zudem nach jeder Aktion eine Belohnung (engl. „**Reward**“), die auch negativ sein kann (d. h. tatsächlich eine Bestrafung ist) mitgeteilt.



Die Interaktion des Agenten mit der Umgebung erfolgt im RL in sog. **diskreten** Zeitschritten. Dies bedeutet, dass der Agent eine Beobachtung erhält, zusammen mit dem Reward, welche er für die letzte Aktion verdient hat. Er wählt dann eine Aktion und bekommt erneut eine Beobachtung und einen Reward. Das geht dann immer so weiter. Es ergibt sich also eine Reihe von Beobachtungen, Rewards und Aktionen:

$(b_0, r_0) \rightarrow a_0 \rightarrow (b_1, r_1) \rightarrow a_1 \rightarrow (b_2, r_2) \rightarrow a_2 \rightarrow (b_2, r_2) \rightarrow a_3 \rightarrow \ldots.$

Die Aufgabe des Agenten ist es, die Summe der Rewards zu maximieren, d. h. so gross wie möglich werden zu lassen. Diese Summe nennt man auch den **Return** (englisch):

$R = r_0 + r_1 + r_2 + r_3 + r_4 + r_5 + \ldots.$

Man könnte auch sagen, der Agent soll bei seiner Arbeit insgesamt möglichst viel Spass haben.

Oft ist die Anzahl solcher Zeitschritte beschränkt. Man nennt dann die Reihe der Beobachtungen, Rewards und Aktionen eine **Episode**.

Beachte, dass sich die Umgebung durch jede Aktion des Agenten verändert. Der Roboter könnte z. B. einen Gegenstand verschieben, dadurch steht er nun an einem neuen Ort und die Umgebung ist verändert. Manchmal ist es möglich, dass der Agent die gesamte Umgebung beobachten kann („**full observability**“). So könnte z. B. der Börsen-Agent Zugriff auf sämtliche Börsendaten bekommen. Oft kann jedoch die Umgebung nur teilweise beobachtet werden („**partial observability**“). Der Fabrik-Roboter z. B. kann vielleicht nur nach vorne sehen und kann nicht beobachten, was hinter ihm geschieht.

Die Aktionen können ähnlich sein wie bei einer Playstation: Man kann nur zwischen verschiedenen Knöpfen wählen („**discrete action space**“) oder auch so, dass man beliebig viele Zwischenstufen wählen kann, wie z. B. bei einem Steuerrad in einem Auto („**continuous action space**“).

Gesucht wird beim „Reinforcement Learning“ eine Verhaltensweise des Agenten, welche den Return maximiert. Man nennt diese Verhaltensweise auch **Policy** (englisch). Eine Verhaltensweise, welche den grösstmöglichen Return bringt, nennt man auch **optimale Policy**. Eine Policy ist eigentlich eine Zuordnung[1] von Beobachtungen zu Aktionen: Für jede Beobachtung (Input) liefert die Policy eine auszuführende Aktion (Output).

ÜBUNG 5.1:

Beantworte folgende Fragen:

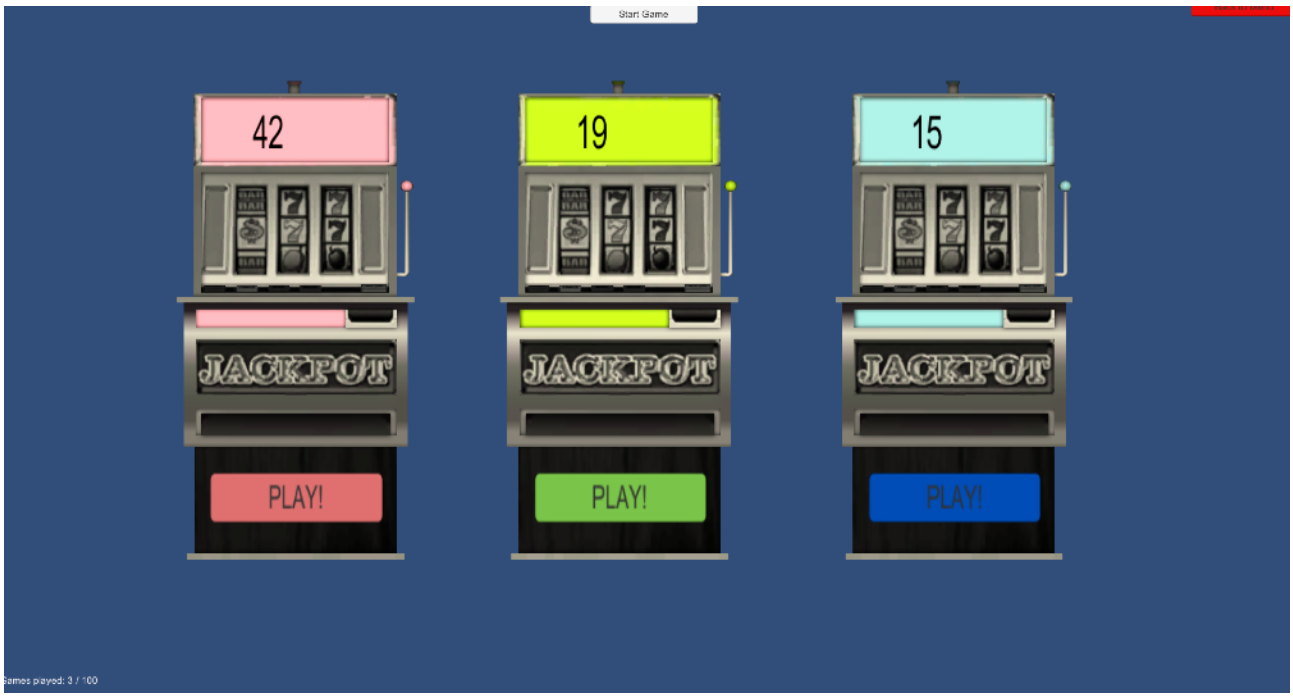
A. Was ist das Ziel des RL Agents?
B. Was wird im Reinforcement Learning gesucht?
C. Was bedeutet ein negativer Reward?

[1] Der Mathematiker nennt dies eine **Funktion**

5.2 Beispiel „multi-armed bandit“

Nun wollen wir ganz praktisch mit „Reinforcement Learning“ spielen.

Starte das Spiel „Exploration / Exploitation Dilemma“ auf deinem Computer. Wenn du auf den „Start Game“-Knopf drückst, beginnt das Spiel mit den einarmigen Banditen. Du darfst nun insgesamt 100 Mal auf einen der drei Knöpfe unten („Play!“) drücken. Jedesmal gewinnst du eine bestimmte Summe Geld. Der gewonnene Betrag wird dir oben auf dem gewählten Automaten angezeigt. Versuche dabei möglichst viel Geld zu verdienen!



ÜBUNG 5.2:

Beantworte folgende Fragen zu diesem Spiel (wir betrachten es nun als RL Problem!):

A. Wer ist der Agent?
B. Was entspricht den Rewards?
C. Was entspricht dem Return?
D. Was ist die Umgebung?
E. Was beobachtet der Agent?
F. Wie viele Zeitschritte dauert eine Episode?

5.3 Der „exploitation / exploration trade-off“

ÜBUNG 5.3-1:

Versuche herauszufinden, welche verschiedenen Strategien bei diesem Spiel von verschiedenen Spielern (z. B. in deiner Familie) angewendet wurden. Welche waren am erfolgreichsten und warum?

Wenn man einen Agenten mittels Reinforcement Learning trainiert, tritt oft folgendes Problem auf: Der Agent findet früh eine Verhaltensweise (Policy), die einen geringen positiven Reward bringt. Der Agent führt nun weiter immer die gleichen Aktionen aus, obschon andere Verhaltensweisen noch viel grössere Rewards bringen könnten. D. h. der Agent bleibt beim Lernvorgang sozusagen bei einer nicht optimalen Policy stecken.

Man nennt dieses Problem auch den „exploitation / exploration trade-off“ (Deutsch: Abwägung zwischen Ausnutzung und Erforschung). Du kennst dieses Problem: In welches Restaurant sollen wir heute gehen? Zum Italiener, den du kennst (und der sicher recht gut ist) oder doch lieber zum neuen Vietnamesen, bei dem du noch nie warst (der aber vielleicht besser ist als der Italiener!). Es heisst „wer wagt, gewinnt“, aber oft ist es besser, sich auf Bewährtes zu verlassen.

Es gibt viele verschiedene Strategien, wie man dieses Problem lösen kann. Wir schauen uns zwei ganz einfache an:

A. **Epsilon-first**
   Der Agent wählt am Anfang für eine beschränkte (im Vergleich zur gesamten Episode kurze) Zeit alle Aktionen zufällig. Dann berechnet er, welche Aktion am meisten Rewards gebracht hat. Während dem Rest der Episode wählt er dann immer diese Aktion.
   Im Beispiel der Restaurants würde man alle Restaurants zuerst testen und dann immer nur noch in das Beste gehen.

B. **Epsilon-greedy**
   Der Agent wählt bei jeder Aktion mit einer kleinen Wahrscheinlichkeit Epsilon (z.B. 1%) eine Aktion, die vermutlich nicht optimal ist (d. h. bisher nicht die höchsten Rewards geliefert hat). Dies nennt man „**Exploration**“ (Erforschung). Mit der Wahrscheinlichkeit 1-Epsilon (d. h. im Beispiel dann 99%) wählt er die bisher optimale Aktion. Dies nennt

man „**Exploitation**“ (Ausnutzung).
Im Beispiel der Restaurants würde man meistens in das bisher Beste gehen, aber immer wieder einmal ein anderes testen (auch solche, die bisher nicht so toll waren, vielleicht hat ja der Koch gewechselt).

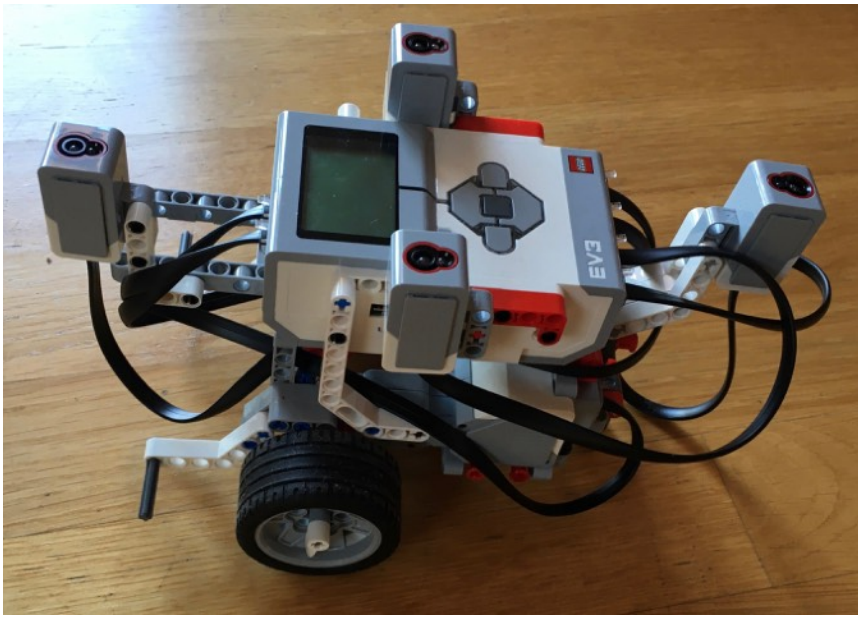
ÜBUNG 5.3-2:

Welche der beiden Strategien ist in unserem Spiel erfolgreicher? Warum?

5.4 „Motti“, der lichtsuchende Roboter

In diesem Kurs werden wir versuchen, einem einfachen Roboter ein wenig Intelligenz einzuhauchen.

Wir präsentieren „Motti“, den Auto-Roboter, der es gerne hell hat (hier in der LEGO Mindstorms Variante):



„Motti" verfügt über…

- …vier **Lichtsensoren**
  (vorne, hinten, links, rechts), damit kann der Roboter „sehen“

- …zwei **Motoren** mit Getriebe, welche zwei grosse Räder antreiben
  (ein drittes bewegliches Rad bzw. eine Kugel hinten ist ohne Antrieb)

- …eine **Stromversorgung**
  (AA Batterien oder ein Akku, welche elektrische Spannung liefern)

- …und einen **Steuercomputer**
  (z. B. LEGO Mindstorms „brick“ oder das BBC micro:bit).

Im Anhang findest du Bauanleitungen für den Roboter (mehrere Versionen mit unterschiedlichen Technologien). Falls du Zeit und Lust hast, einen solchen Roboter zu bauen, dann wäre nun der richtige Moment dazu.

ÜBUNG 5.4-1:

Teste deinen Roboter im (möglichst dunklen) Zimmer mit einer starken Taschenlampe. Wie verhält sich der Roboter?

Auf „Mottis“ Steuercomputer läuft eine Software (d. h. ein Computerprogramm), welche auf einem neuronalen Netz basiert (wir werden später lernen, was das bedeutet), also eine sehr einfache KI.

ÜBUNG 5.4-2:

Wäre es möglich, das Verhalten des Roboters mit einem herkömmlichen, von einem Menschen programmierten Computerprogramm zu erreichen? Welche Regeln könnte man dem Roboter einprogrammieren? Was müsste man dabei beachten, bzw. worin liegt die Schwierigkeit?

ÜBUNG 5.4-3:

Wenn wir „Motti“ später mittels „Reinforcement Learning“ trainieren wollen, müssen wir ihn aus dieser Perspektive betrachten. Beantworte folgende Fragen:

A. Wer ist der Agent?
B. Was entspricht den Rewards?
C. Was entspricht dem Return?
D. Was ist die Umgebung?
E. Was beobachtet der Agent?
F. Wie lange dauert eine Episode?

ÜBUNG 5.4-4 (SCHWER):

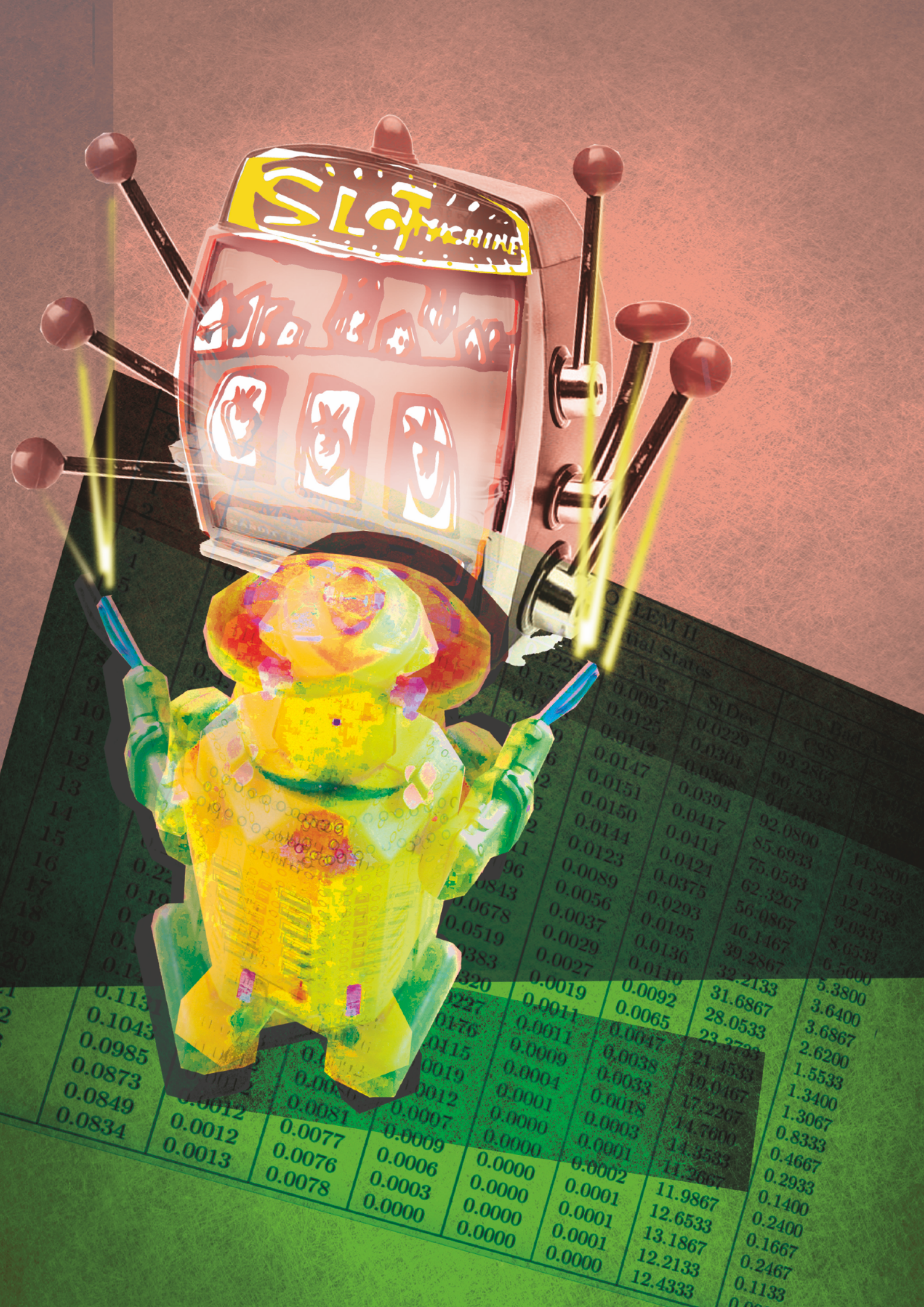
Ein Beispiel eines von Menschen programmierten Computerprogramms für „Motti“ befindet sich im Kurs-Kit. Lade dieses (der Dateiname lautet enthält das Wort „remote control“) auf den Computer von „Motti“ und teste das Verhalten des Roboters.

Schau dir das Programm im MakeCode Editor an und versuche herauszufinden, wie es funktioniert.
Wie könnte man das Programm verbessern?

5.5 Warum RL schwierig ist: Das ‚credit assignment problem‘

Warum ist ‚Reinforcement Learning’ ein schwieriges Problem? Bei „Motti“ sind die Rewards zeitlich eng an seine Aktionen gekoppelt. Unter solchen Umständen ist es für den Agenten relativ einfach, aus den Rewards das richtige Verhalten zu lernen. Aber stell dir vor, der Reward kommt erst stark verzögert. Z. B. bei einem Computerspiel könnte es erst dann einen Reward geben, wenn das Spiel gewonnen ist. Der Agent muss dann u. U. Tausende von Aktionen durchführen, bis er einen Reward erhält. Welche davon waren nun gut und welche nicht? Man nennt dieses sehr schwierige Problem im „Reinforcement Learning“ das „**credit assignment problem“** (etwa „Verdienst-Zuordnungsproblem“).

Wir werden in diesem Kurs allerdings eine Trainingsmethode einsetzen, bei welcher das „credit assignment problem“ tatsächlich kein Problem darstellt.


SLOT MACHINE

6. Künstliche neuronale Netze

6.1 Künstliche Neuronen

Künstliche Neuronen sind natürlichen Neuronen nachempfunden.
Es gibt mehrere Eingänge (Inputs, entsprechend den Dendriten des natürlichen Neurons) und einen einzigen Ausgang (Output, entsprechend dem Axon). In die Eingänge werden Zahlen gefüttert und der Ausgang ist ebenfalls eine Zahl.

Um verstehen zu können, wie ein künstliches Neuron funktioniert, müssen wir zuerst wissen was eine **Funktion** ist. Wir kümmern uns nicht um die exakte/vollständige mathematische Definition. Für unsere Zwecke soll eine Funktion eine kleine mathematische Maschine sein, in welche man eine Zahl hineingeben kann (Input) und auf der auf der anderen Seite wieder eine Zahl herauskommt (Output). Ein einfaches Beispiel wäre die folgendermassen definierte Funktion:

1. Wir bezeichnen den Input als x. Der Mathematiker nennt diese Zahl das **Argument**.
2. Der Output soll x^2 sein (also x mal x). Der Mathematiker nennt diese Zahl den **Funktionswert**.

Diese Funktion ordnet also z. B. dem Argument 3 den Funktionswert 9 zu. Wenn wir diese Funktion mit dem Buchstaben F bezeichnen, dann schreibt man auch 9 = F(3) .

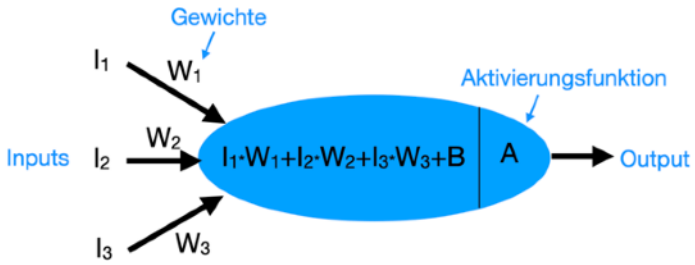
Nun zurück zum künstlichen Neuron:

Man berechnet den Output eines künstlichen Neurons aus dem Input in den folgenden drei Schritten:

1. Jede Input-Zahl wird mit einer zum Input gehörigen Zahl (dem sog. **Gewicht**, engl. „**weight**“) multipliziert. Dann werden alle Resultate addiert.

2. Zum Resultat von 1. wird eine weitere Zahl, der sog. **Bias**, hinzuaddiert

3. Das Resultat von 2. wird durch eine sog. **Aktivierungsfunktion** (engl. „**activation function**“) geschickt (wir sehen gleich was das ist)



Also, für ein Neuron mit 2 Eingängen:

$$\text{Output} = A(I_1 * W_1 + I_2 * W_2 + B)$$

wobei:

I_1, I_2 : Input 1 resp. 2
W_1, W_2 : Zu Input 1 resp. Input 2 gehörende Gewichte
B: Der Bias
A: Die Aktivierungsfunktion

Eine Aktivierungsfunktion ist eine Funktion, die sich für künstliche Neuronen eignet. Eine beliebte Aktivierungsfunktion in der künstlichen Intelligenz ist z. B. die sog. **ReLU**-Funktion (ReLU ist eine Abkürzung für den englischen Begriff „rectified linear unit“):

Die ReLU-Funktion ist wie folgt definiert:

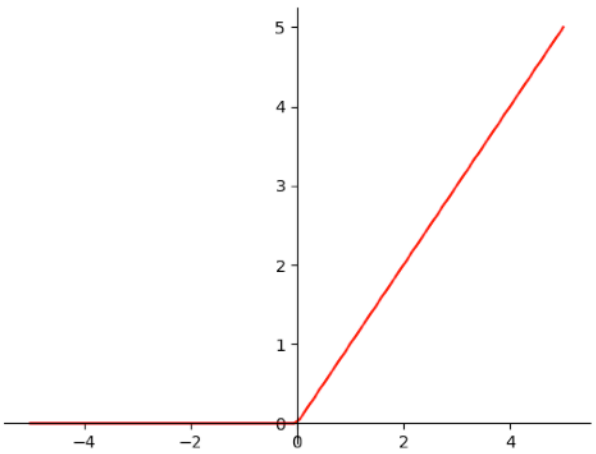
Die Output-Zahl ist…

1. …0 für alle Input-Zahlen kleiner oder gleich 0
   und
2. …gleich der Input-Zahl für alle Zahlen grösser als 0

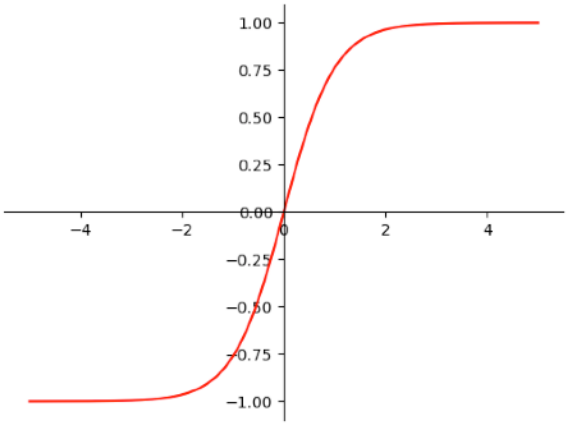
ÜBUNG 6.1-1:

A. Wie lautet der Output-Wert der ReLU-Funktion für den Input -12?
B. Wie lautet der Output-Wert der ReLU-Funktion für den Input 34.5?

Man kann Funktionen leicht visualisieren, indem man in einem x/y-Koordinatensystem für jeden Wert von x, auf der y-Achse den Output-Wert der Funktion für diesen x-Wert mit einem Punkt einzeichnet. Das sieht bei der ReLU-Funktion dann so aus:



Eine weitere beliebte Aktivierungsfunktion ist der sog. Tangens Hyperbolicus (seine Berechnung ist fast so kompliziert wie sein Name. Wir müssen diese aber nicht verstehen):

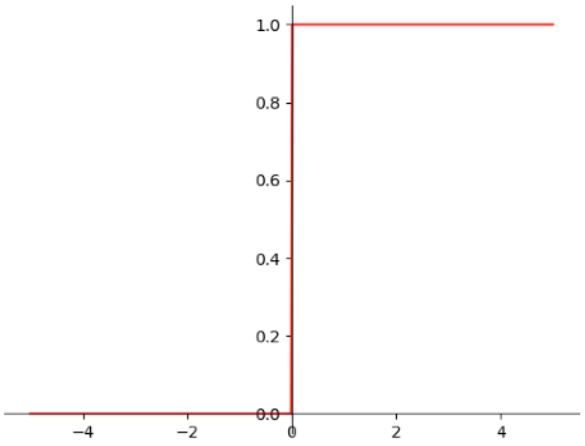


Wie man sieht, ist der Output[2] auf den Bereich[3] -1 bis 1 beschränkt, was oft sehr praktisch ist.

Ebenfalls oft verwendet wird die Step-Funktion (aus dem Englischen „step“ für Stufe/ Schritt):

Step-Funktion: Die Output-Zahl ist…

1. …0 für alle Input-Zahlen kleiner oder gleich 0
   und
2. …1 für alle Zahlen grösser als 0



[2] Der Mathematiker nennt die Menge der möglichen Outputs einer Funktion **Wertebereich**

[3] Der Mathematiker nennt einen solchen Bereich ein **Intervall**

ÜBUNG 6.1-2:

A. Wie lautet der Output-Wert der Step-Funktion für den Input -12?
B. Wie lautet der Output-Wert der Step-Funktion für den Input 34.5?

Die Gewichte W und den Bias B eines Neurons nennt man übrigens zusammen die **Parameter** des Neurons.

ÜBUNG 6.1-3:

Berechne für ein Neuron mit 2 Eingängen und den Parametern W_1=2, W_2=0.5, B=4 den Output für den Input I_1=3, I_2=2 mit dem Taschenrechner

A. Mit der ReLU-Funktion als Aktivierungsfunktion für das Neuron
B. Mit der Step-Funktion als Aktivierungsfunktion für das Neuron

Tipp: Berechne zuerst die Schritte 1. und 2. für das Neuron (sie sind für beide Aufgaben gleich) und löse dann A. und B. (Schritt 3).

ÜBUNG 6.1-4 (SCHWER):

Zeige, dass bei einem Neuron mit zwei Eingängen und der Step-Funktion als Aktivierungsfunktion, die Gebiete in denen der Output des Neurons 0 bzw. 1 ist, durch eine Gerade getrennt werden.

Beachte: Wenn das Neuron zwei Eingänge hat, können wir die Menge der möglichen Inputs mit einer Fläche darstellen, welche ein Koordinatensystem hat: die x- (horizontal/links-rechts) und die y-Achse (vertikal/unten-oben). Jeder Punkt der Fläche entspricht dann einem möglichen Input des Neurons.

6.2 Eigenschaften künstlicher Neuronen

Nun, da wir die grundlegende Mathematik künstlicher Neuronen kennen, können wir uns überlegen, was die verschiedenen Elemente bedeuten.

Beachte folgendes:

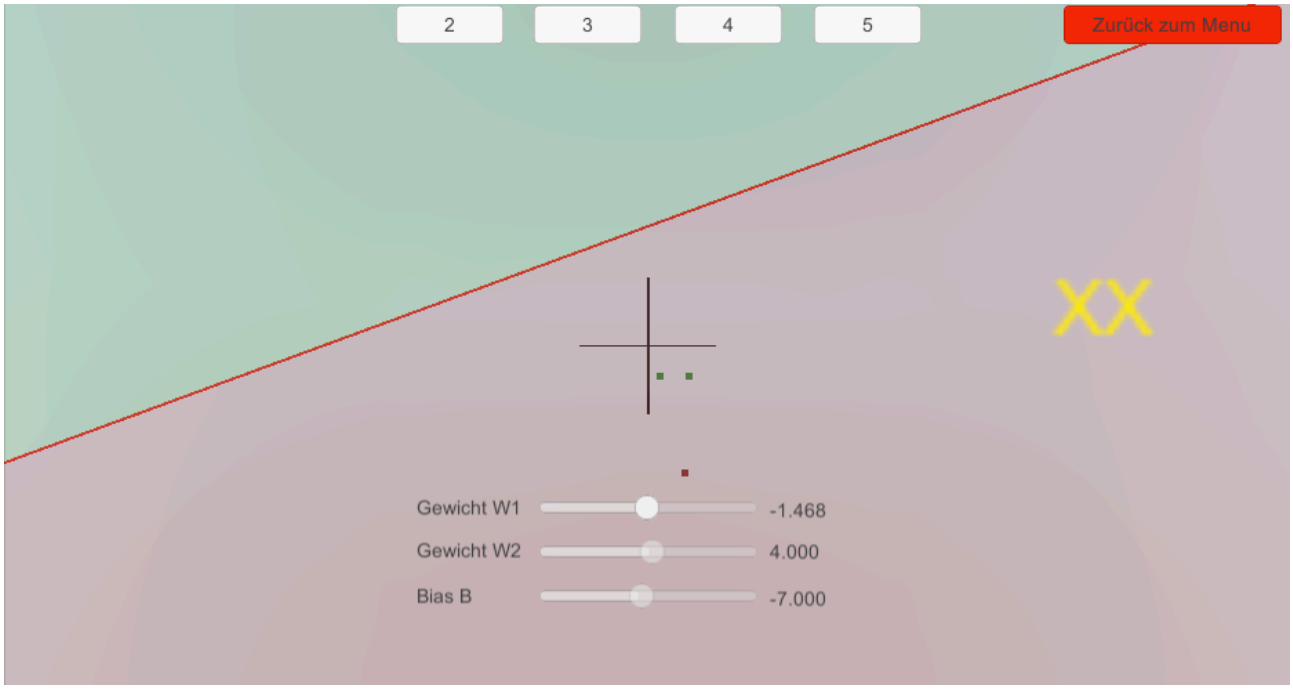
- Die Inputs können sowohl positiv wie auch negativ sein.
- Die Gewichte können sowohl positiv wie auch negativ sein. Damit können hemmende Eingänge (neg. Gewicht) und erregende Eingänge (pos. Gewicht) realisiert werden. Es hängt dann aber vom Vorzeichen des Inputs ab, ob der Eingang wirklich hemmend oder erregend wirkt. Der Betrag des Gewichtes gibt an, wie stark der entsprechende Input bei der Berechnung des Outputs berücksichtig werden soll.

Wir verwenden für die folgenden Überlegungen einfachheitshalber die Step-Funktion als Aktivierungsfunktion. Wir sehen, dass der Bias in diesem Fall wie ein Schwellwert funktioniert: Nehmen wir an, der Bias sei -10. Dann muss die Summe der Produkte Gewicht * Input grösser als 10 sein, damit das Neuron „feuert“ d. h. 1 ausgibt. Das künstliche Neuron funktioniert in diesem Beispiel also recht ähnlich wie ein natürliches.

Spiel „Perzeptron“:

Ein künstliches Neuron mit der Step-Funktion als Aktivierungsfunktion nennt man auch „**Perzeptron-Neuron**“. Es ist eines der frühesten erforschten künstlichen Neuronen.

Starte das Spiel „Perzeptron“ auf deinem Computer:

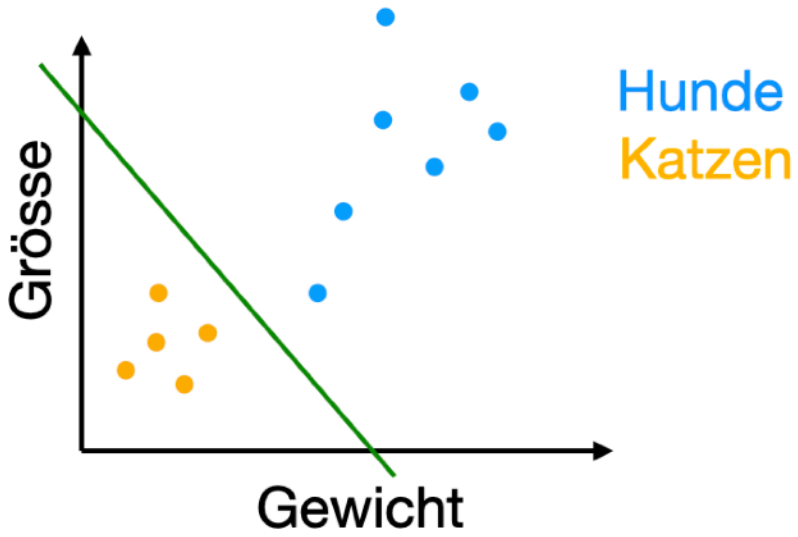


Das Programm visualisiert den Output eines Neurons mit zwei Eingängen (entsprechend den x- und y-Koordinaten auf dem Bildschirm) und der Step-Funktion als Aktivierungsfunktion (also ein Perzeptron-Neuron mit zwei Eingängen). Für jeden Punkt auf dem Bildschirm wird der Output des Perzeptron-Neurons berechnet.

Mit den Schiebereglern kannst du die Parameter des Perzeptron-Neurons (die beiden Gewichte und den Bias) einstellen. Du siehst, dass der Output des Neurons die Input-Fläche durch eine Gerade zweiteilt (in eine rote Fläche mit Output 0 und eine grüne mit Output 1).

ÜBUNG 6.2:

Wir versuchen nun, einen einfachen **Klassifikator** (engl. „**classifier**“) mit einem einzigen Neuron zu realisieren. Ein Klassifikator ist ein Computerprogramm, das Objekte in verschiedene Kategorien einteilen kann. Ein Beispiel wäre ein Programm, das anhand von Gewicht und Grösse, Hunde von Katzen unterscheiden kann:



Da die meisten Hunde grösser und schwerer sind als Katzen, gelingt es in diesem Beispiel, die beiden Klassen durch eine Gerade zu trennen. Alle Tiere oberhalb der Geraden werden als Hunde klassifiziert, diejenigen unterhalb der Geraden als Katzen.

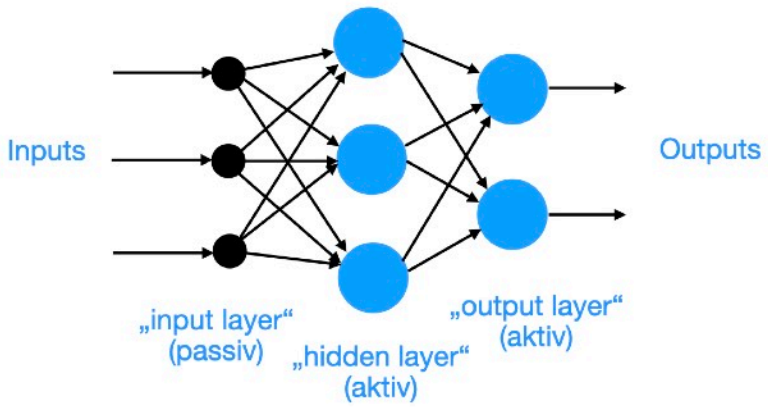
Versuche nun, selber einen solchen Klassifikator zu bauen:
Drücke auf einen der Knöpfe oben und es werden je nach Knopf 2-5 Datenpunkte auf dem Input-Raum gezeigt, wobei deren Position zufällig gewählt wird. Dabei sind einige Datenpunkte grün und andere rot (dies sind die beiden Klassen). Deine Aufgabe ist es, das Perzeptron-Neuron (bzw. seine Parameter) so einzustellen, dass es diese Datenpunkte korrekt klassifiziert, d. h. dem richtigen Bereich zuordnet. Dies bedeutet, dass die grünen Datenpunkte im grünen Bereich liegen sollten und die roten Datenpunkte im roten Bereich.
Versuche die Aufgabe für verschiedene Anzahl Datenpunkte zu lösen und beantworte dann die folgenden Fragen:

A. Kann dieses Problem für 2 Datenpunkte immer gelöst werden?
B. Kann dieses Problem für 3 Datenpunkte immer gelöst werden?
C. Kann dieses Problem für 4 Datenpunkte immer gelöst werden?
D. Kann dieses Problem für 5 Datenpunkte immer gelöst werden?

Beachte, dass wir den seltenen Fall, dass mehrere Datenpunkte zufälligerweise exakt zusammenfallen, nicht berücksichtigen wollen.

6.3 Künstliche neuronale Netze (KNN)

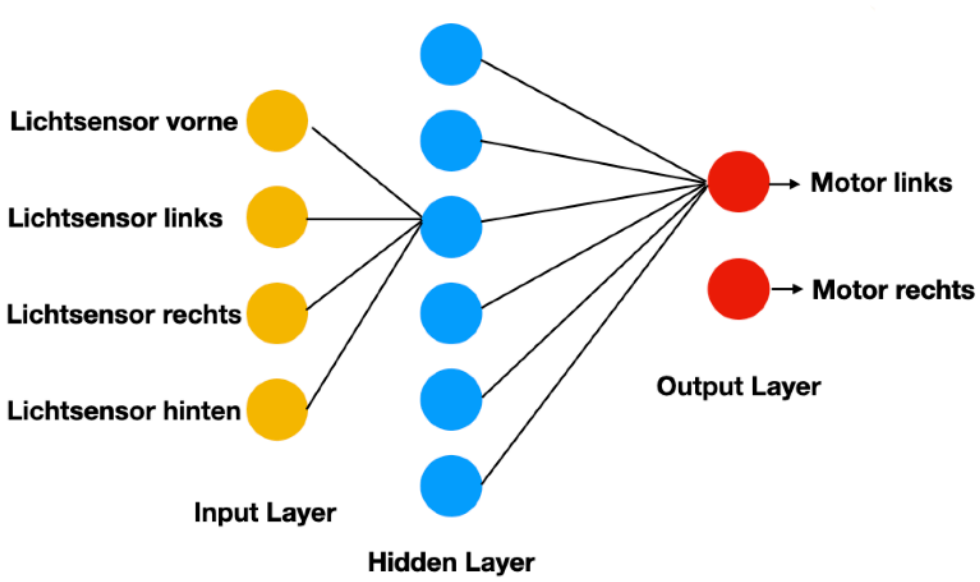
Ganz ähnlich wie im Nervensystem von Tieren kann man aus künstlichen Neuronen Netzwerke bilden. Dabei wird der Output von gewissen Neuronen mit dem Input von anderen verbunden. Man bildet dabei meistens sog. Schichten (engl. „layers“) von Neuronen:



Je nach Lage im neuronalen Netz haben die Schichten verschiedene Namen:

Die Zahlen des Inputs bezeichnet man als „**input layer**“. Es sind eigentlich keine richtigen Neuronen, weil sie gar nicht rechnen (sie haben keine Parameter und keine Aktivierungsfunktion). Sie verteilen lediglich die Input-Zahlen auf die folgenden Neuronen. Man nennt das „input layer“ daher auch **passiv**. Die nächste Schicht von Neuronen nennt man „**hidden layer**“, da sie gewissermassen zwischen input- und output-layer versteckt sind. Komplexe neuronale Netze (z. B. wie von Facebook verwendet) können Tausende von „hidden layers“ aufweisen. Die Neuronen schliesslich, welche den Output berechnen, nennt man „**output layer**“. „Hidden layer“ und „output layer“ sind **aktiv**, d. h. es sind richtige Neuronen mit allem Drum und Dran.

Wie sieht das bei „Motti“ aus? Im folgenden Bild sehen wir das neuronale Netz von „Motti“ (es sind nicht alle Verbindungen zwischen den Neuronen eingezeichnet, das wäre zu unübersichtlich):



Das Netzwerk hat vier Neuronen im „input layer“, welche die Werte der vier Lichtsensoren aufnehmen. Im einzigen „hidden layer“ gibt es 5-12 Neuronen (du kannst dann später mit verschiedenen Werten experimentieren). Das „output layer“ besteht aus zwei Neuronen, welche die Motorkraft (und über das Vorzeichen +/- auch die Drehrichtung) für die beiden Motoren ausgeben.

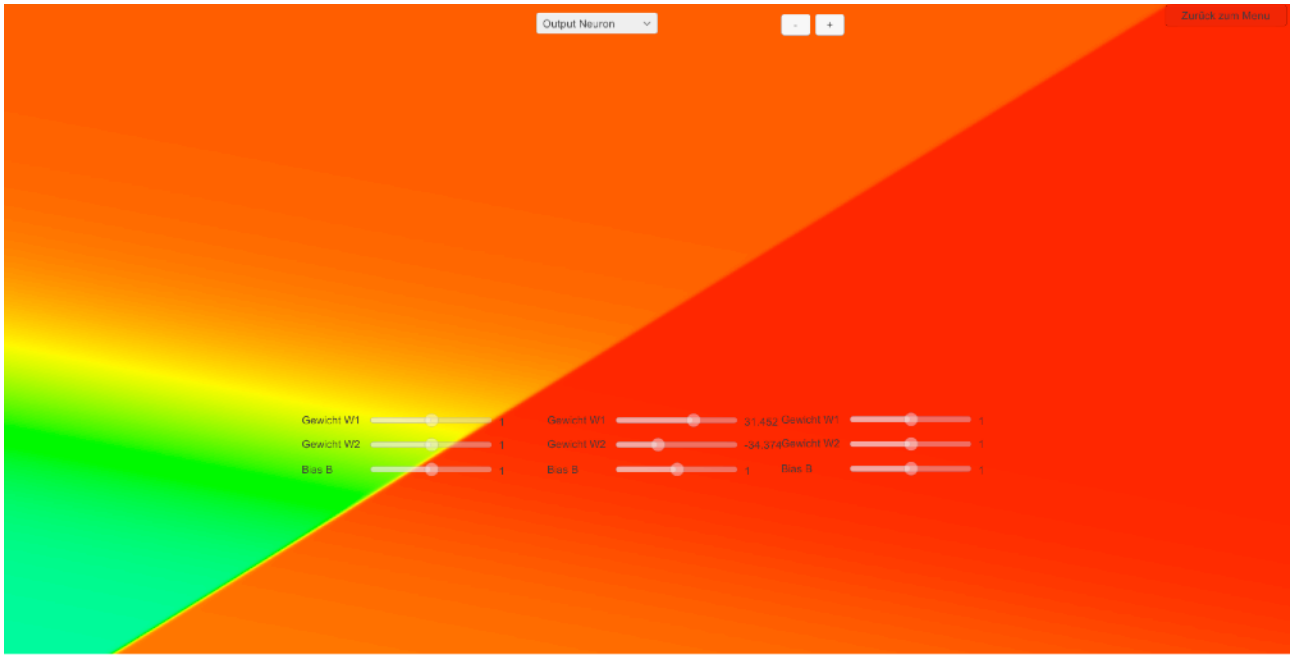
Du siehst auch, dass das „Gehirn“ von „Motti“ eine Policy bildet: Es ist offensichtlich eine Funktion, welche aus den Inputs der vier Lichtsensoren (der Beobachtung) eine Aktion (Drehrichtung und Kraft der Motoren) als Output berechnet.

ÜBUNG 6.3-1:

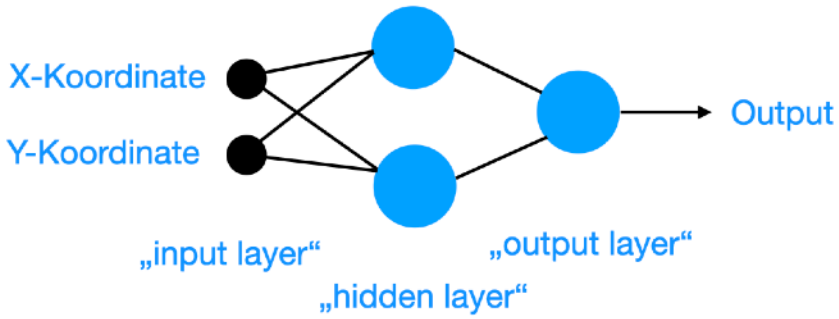
Welche Aktivierungsfunktion eignet sich besonders für das „output layer“ von „Motti“? Warum?

ÜBUNG 6.3-2:

Starte das Computerspiel „Mit einem neuronalen Netz spielen“:



Das Spiel visualisiert ein einfaches neuronales Netz aus drei Neuronen. Der Input-Raum besteht wieder aus der Bildschirmfläche mit dem Koordinatensystem mit der x- (links/rechts) und der y-Achse (oben/unten). Das „hidden layer“ besteht aus zwei Neuronen, das „output layer“ aus einem Neuron. Im folgenden Bild sehen wir die **Architektur** des neuronalen Netzes dieses Spiels:



Den Output des einen Output-Neurons benutzen wir, um für den zu den Koordinaten x,y gehörigen Pixel des Bildschirmes eine Farbe zu berechnen. Da wir nicht unendlich viele Farben unterscheiden können, wählen wir für alle Neuronen eine Aktivierungsfunktion mit beschränktem Output-Bereich. Ihr erinnert euch: Der Output der Tangens Hyperbolicus Funktion ist auf den Bereich -1 bis 1 beschränkt. Das ist für uns ideal.
Die Farben können wir nun z. B. wie folgt den Zahlen zuordnen:

-1 0 1

So können wir also den Output des KNN für eine grosse Menge von Inputs in einem Bild sehen!

Nun endlich zur Aufgabe:

Versuche durch das Verändern der Parameter des KNN (d.h. aller drei Neuronen) möglichst interessante Bilder für das Output-Neuron zu erzeugen.

Tipp: Visualisiere dazu zuerst die beiden „hidden layer“ Neuronen je einzeln und verändere nur die zugehörigen Parameter bis du ein spannendes Bild siehst. Schalte erst dann auf das Output-Neuron und verändere dessen Parameter.

Wir sehen, dass ein neuronales Netz den Input-Bereich auf viel raffiniertere Weise aufteilen kann, als ein einzelnes Neuron. Es ist nun aber, aufgrund der Vielzahl der Parameter sehr schwierig geworden, die Parameter von Hand so zu wählen, dass ein ganz bestimmtes Bild entsteht (um z .B. wie im letzten Spiel einen Klassifikator zu bauen). Wir müssen also herausfinden, wie der Computer diese Parameter für uns finden kann. Mit dem Computer diese Parameter zu finden (d. h. den Computer lernen zu lassen), ist die Kunst des „**deep learning**“. Ihr werdet bald auch etwas von dieser Kunst verstehen!

ÜBUNG 6.3-3: (FÜR SCHULKLASSEN / GRUPPEN)

Die Schüler als Neuronen, die Klasse als neuronales Netz

Wir berechnen in der Klasse für einige einfache Inputs den Output des in „Motti“ gespeicherten neuronalen Netzes.

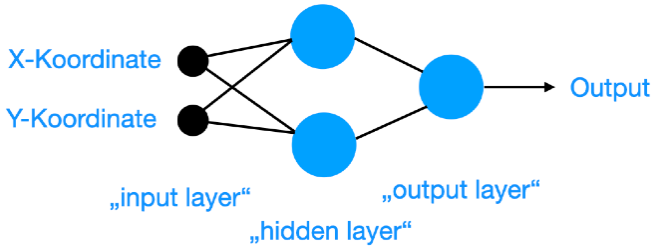
Zu diesem Zweck bekommt jeder Schüler ein Blatt mit den Parametern für sein Neuron (aus dem „hidden layer“ oder dem „output layer“, für das passive „input layer“ gibt es nichts zu tun). Zuerst müssen natürlich die Schüler des „hidden layer“ rechnen, dann diejenigen des „output layers“.

A. Wir nehmen an die Input-Neuronen seien in der Konfiguration 1,0,0,0 (d. h. es fällt nur Licht auf den vorderen Sensor, die übrigen Sensoren sind im Dunkeln). Wie lautet der Output von „Mottis" KNN, seinem „Gehirn“? Was bedeutet dieses Resultat für „Motti“? Tut er das Richtige?

B. Wir nehmen an, die Input-Neuronen seien in der Konfiguration 0,0,0,1 (d.h. es fällt nur Licht auf den hinteren Sensor, die übrigen Sensoren sind im Dunkeln). Wie lautet der Output von „Mottis“ KNN? Was bedeutet dieses Resultat für „Motti"?

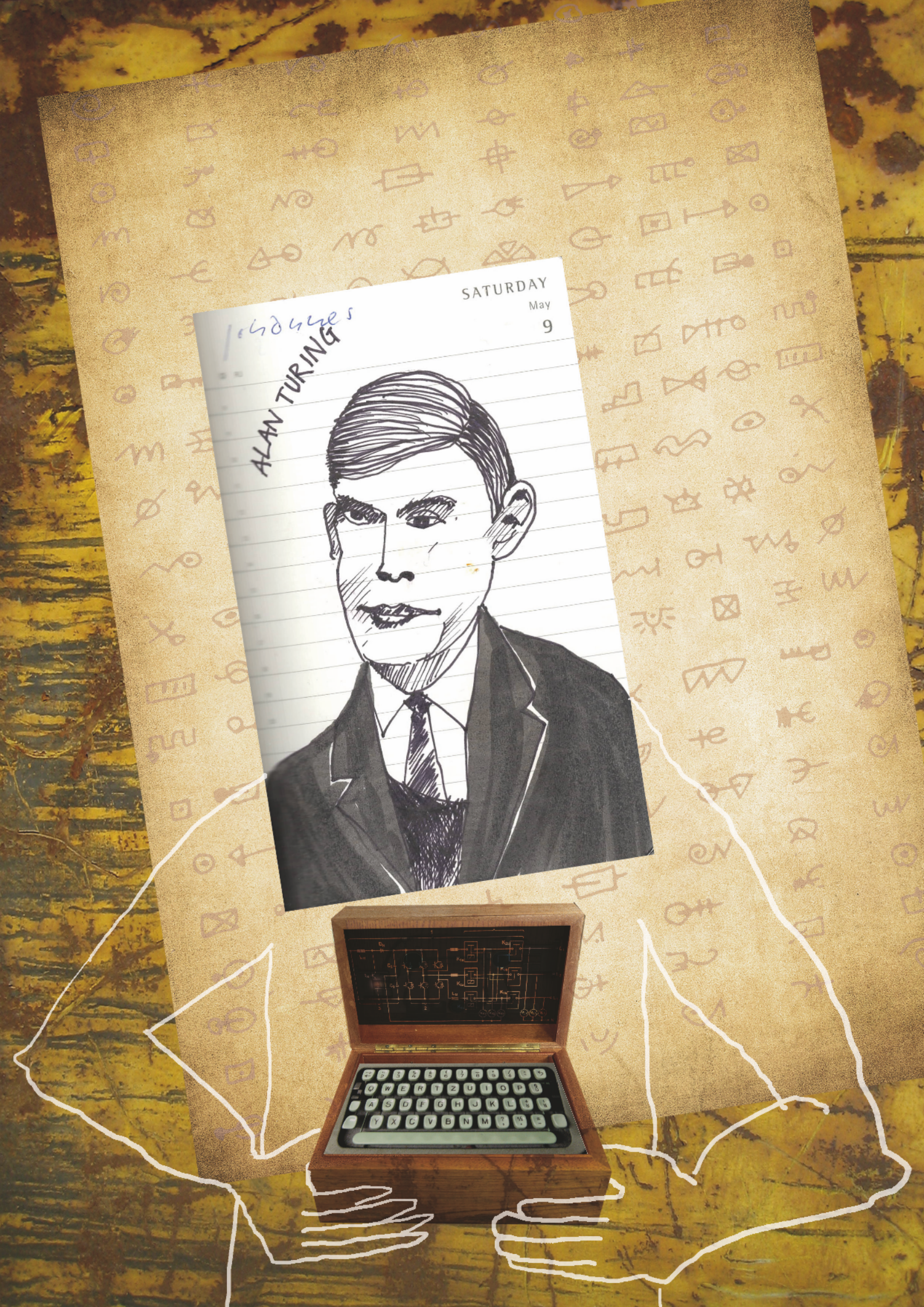
ÜBUNG 6.3-4 (SCHWER!):

Betrachte das KNN aus dem letzten Spiel:



Nun nehmen wir an, die Neuronen hätten keine Aktivierungsfunktion (d. h. wir verwenden den Output von Schritt 2 und lassen Schritt 3 weg). Zeige, dass sich dann das ganze neuronale Netz durch ein einzelnes Neuron ersetzen lässt.

Tipp: Berechne den Output des neuronalen Netzwerks für gegebene Inputs I1 und I2, indem du geeignete Variablen für die unbekannten Parameter der Neuronen definierst.


SATURDAY
May
9
ALAN TURING

7. Algorithmen der künstlichen Intelligenz

7.1 Das Problem des grossen Suchraumes

Wie wir in unserem Beispiel mit drei Neuronen gesehen haben, ist es nicht ganz einfach, ein KNN durch Wählen der Parameter dazu zu bringen, etwas Sinnvolles zu tun.

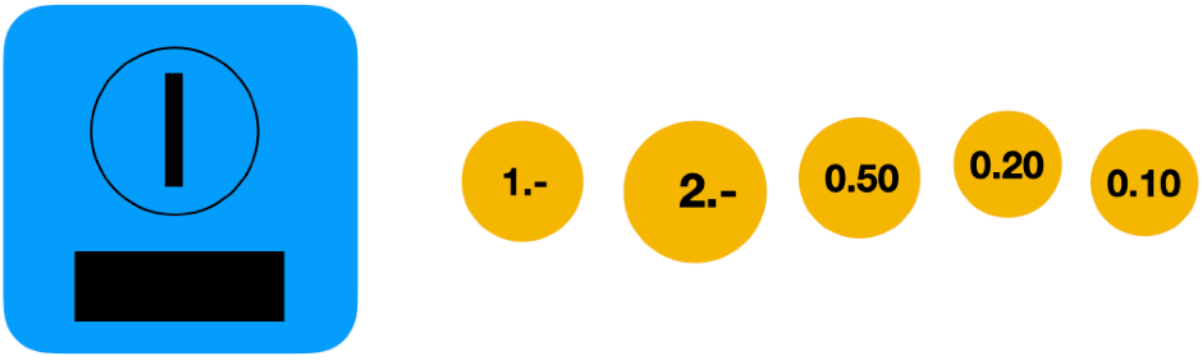
ÜBUNG 7.1-1:

Wie viele Parameter hat „Mottis“ KNN insgesamt, wenn man von 10 Neuronen im „hidden layer“ ausgeht?

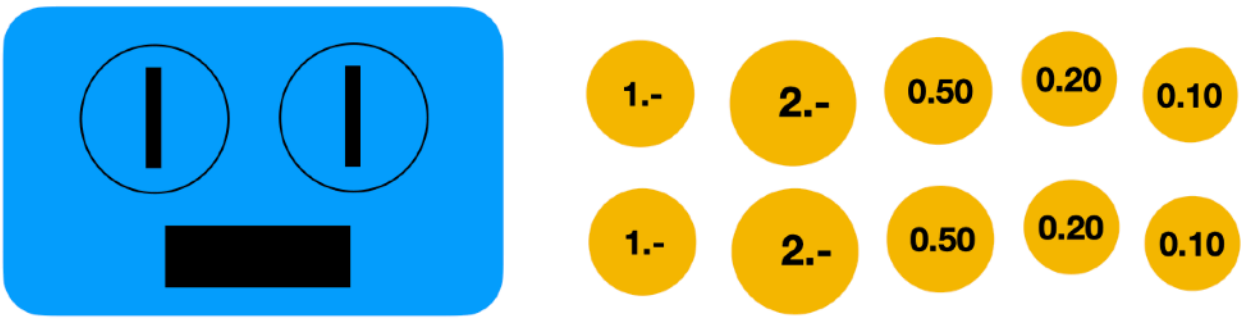
Das sind ziemlich viele Parameter! Nun wollen wir herausfinden, warum es so schwierig ist, gute Werte für diese Parameter zu finden.

Stell dir folgende Aufgabe vor: Du findest einen Münzautomaten, von dem du weisst, dass er Geld ausspuckt, wenn man eine Münze einwirft. Der ausgespuckte Betrag hängt von der eingeworfenen Münze ab. Du weisst aber nicht, welche Münze den grössten Ertrag liefert.

Wenn es fünf verschiedene Münzen gibt, musst du fünfmal eine Münze einwerfen, damit du sicher weisst, welche Münze den grössten Ertrag liefert:



Nun findest du einen weiteren Automaten, welcher zwei Einwurfschlitze hat. Der Automat spuckt je nach Kombination der beiden Münzen wiederum Geld aus. Es ist aber ein Unterschied, ob du links 1.- und rechts 2.- einwirfst oder umgekehrt:



ÜBUNG 7.1-2:

Wie viele Münzen musst du nun einwerfen, damit du sicher bist, bei welcher Kombination du den höchsten Betrag erhältst?

Du siehst, es sind plötzlich viel mehr Einwürfe erforderlich!

Stell dir nun vor, du suchst eine einzelne Zahl, welche den Output einer gegebenen Funktion maximiert (d.h. möglichst gross werden lässt). Wir können versuchen, der Funktion einfach verschiedene Zahlen zu übergeben und schauen, was herauskommt. Das erste Problem, das wir haben: es gibt unendliche viele Zahlen, die als Input in Frage kommen! Nehmen wir aber an, dass es genügt, wenn man…

1. …nur die Zahlen innerhalb eines bestimmten Intervalls, z.B. -10 bis 10, testet
   (z. B. weil man vermutet oder weiss, dass Zahlen die kleiner als -10 oder grösser als 10 sind sowieso einen schlechten Output erzielen)

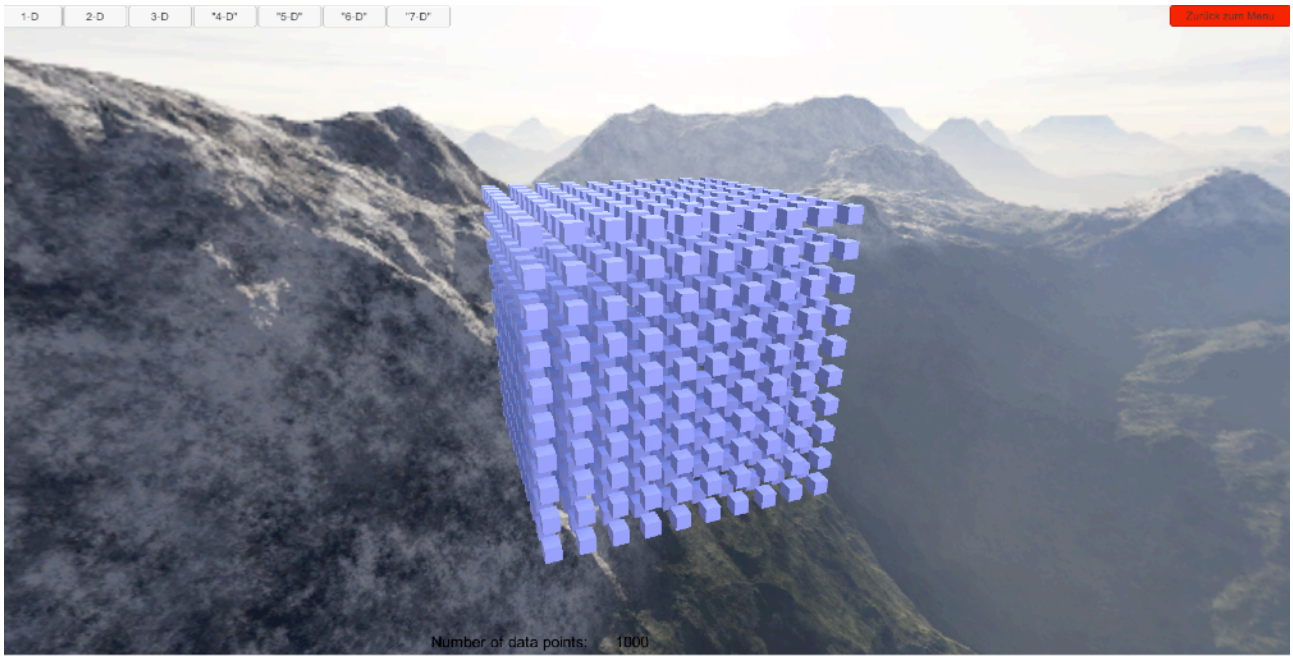
2. …nur ca. 10 Beispiele von Zahlen in diesem Intervall testet
   (z. B. weil die Funktionswerte für Zahlen zwischen zwei benachbarten Beispielzahlen sehr ähnlich sind).

ÜBUNG 7.1-3:

Wieviele Zahlen bzw. Zahlenkombinationen muss man in diesem Fall testen wenn:

A. Die Funktion eine Zahl als Input hat
B. Die Funktion zwei Zahlen als Input hat
C. Die Funktion drei Zahlen als Input hat

Wie geht das weiter? Lassen wir uns das vom Computer visualisieren!
Starte dazu das Computerprogramm „Suchraum“:



ÜBUNG 7.1-4:

Klicke auf die Anzahl der Dimensionen oben, wobei du bei „1-D“ startest und bei „7-D“ aufhörst. Die Anzahl der nötigen Tests nimmt schnell zu (du musst mit dem Mausrad mehrmals zoomen, damit du alles siehst!). Gehe am Schluss zurück zu „1-D“, damit du den Grössenunterschied klar siehst.

Die Suche nach geeigneten Parametern in einem KNN ist nun leider genau ein solches Problem: Die Funktion, die wir maximieren wollen, ist die Zuordnung von allen KNN-Parametern zum Return, welche der Roboter damit erzielen kann (also letztlich seiner Intelligenz). Wenn wir also naiv auf die besprochene Weise alle möglichen Parameter mit dem Roboter durchtesten wollen, dann dauert unsere Suche fast ewig!

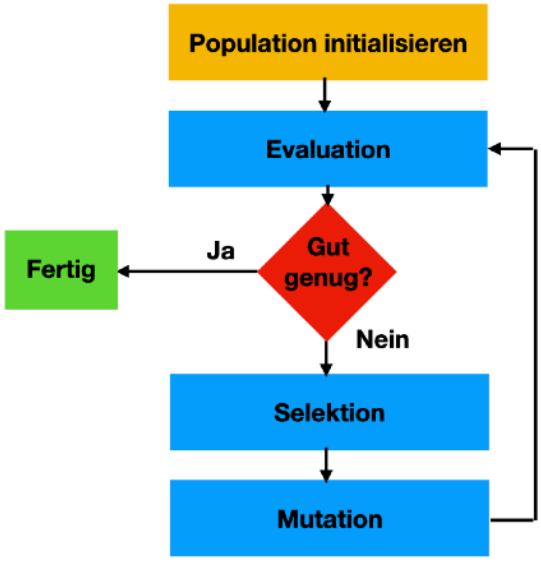
Man nennt dieses Problem im maschinellen Lernen auch „**the curse of dimensionality**“ (etwa „der Fluch der vielen Dimensionen“).

Wir brauchen also eine Idee, wie wir die Parameter viel effizienter berechnen können. Eine aus einer Idee resultierende Rechenvorschrift für einen Computer nennt man in der Informatik einen **Algorithmus**.

7.2 Der genetische Algorithmus

Im „Reinforcement Learning“ werden viele verschiedene Algorithmen zum Finden dieser Parameter verwendet. Einige davon sind sehr kompliziert und sogar für Experten schwer zu verstehen. Es gibt zum Glück aber auch einige, die sind einfach. Wir wollen uns im Folgenden einen solchen genauer anschauen und später damit „Motti“ trainieren. Er heisst **genetischer Algorithmus** und ist eigentlich der Natur abgeschaut.

Er funktioniert so:



1. In einem ersten Schritt erzeugt man eine grössere Anzahl von zufällig gewählten Parameter-Kombinationen (die sog. **Population**). Man nennt diesen Schritt das **Initialisieren** (d.h. „in den Anfangszustand versetzen“) der Population. In unserem Beispiel ist jede dieser Parameter-Kombinationen ein eigenes, vollständiges KNN, d. h. eigentlich eine Art „künstliches Gehirn“.
2. Nun testet man jede einzelne dieser Parameter-Kombinationen durch Simulation. Das bedeutet z. B. bei einem Roboter, dass man ihn in einer Art Computerspiel seine Aufgabe erledigen lässt und dabei misst, wie gut er sich dabei schlägt (d. h. wie gross der Return ist). Man nennt diesen Vorgang **Evaluation**.
3. Nun entscheidet man, ob in der Population schon eine Parameter-Kombination (d. h. ein „Gehirn“) vorhanden ist, das für unsere Zwecke gut genug ist, d. h. einen befriedigenden Return erzeugt hat. Wenn ja, sind wir fertig!
4. Nun wählen wir unter allen Kandidaten (d. h. eigentlich Parameter-Kombinationen) diejenigen 50% aus, die sich in der Simulation am besten geschlagen haben. Wir nennen diese die **Elite**. Dazu sortieren wir alle Kandidaten nach dem in der Simulation erzielten Return und nehmen die bessere Hälfte. Die schlechtere Hälfte löschen wir. Diesen Vorgang nennt man **Selektion**.
5. Nun haben wir nur noch die Hälfte der Population übrig. Damit wir wieder eine gleich grosse Population wie am Anfang haben, erzeugen wir neue Parameter-Kombinationen. Wir erzeugen diese aus den erfolgreicheren 50%, indem wir deren

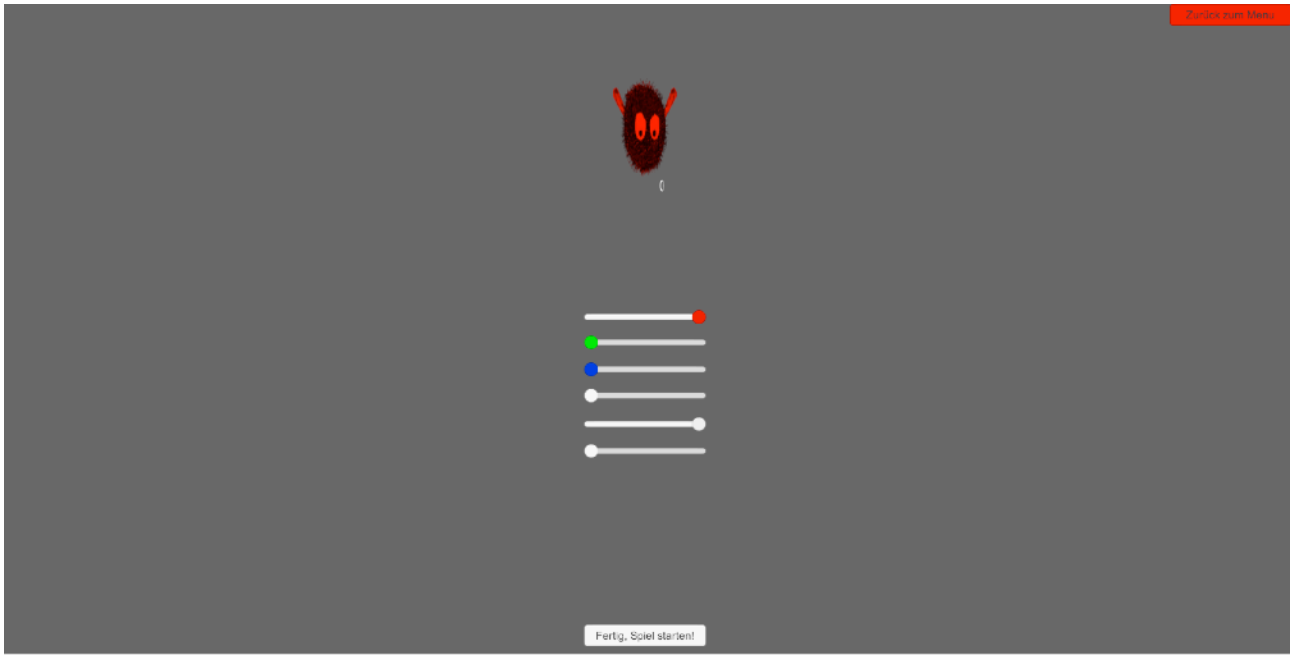
Parameter leicht abändern (am Anfang stärker, später weniger stark). Diesen Vorgang nennt man **Mutation**.

6. Nun haben wir wieder eine gleich grosse Population wie am Anfang und wir gehen zurück zu Schritt 1 und machen dort weiter.

Wir wollen diesen Algorithmus gleich austesten! Allerdings tun wir dies an einem Beispiel, das nichts mit KNN zu tun hat. Und du wirst selbst wichtige Aufgaben im Algorithmus übernehmen.

Starte das Computerspiel „genetischer Algorithmus“:

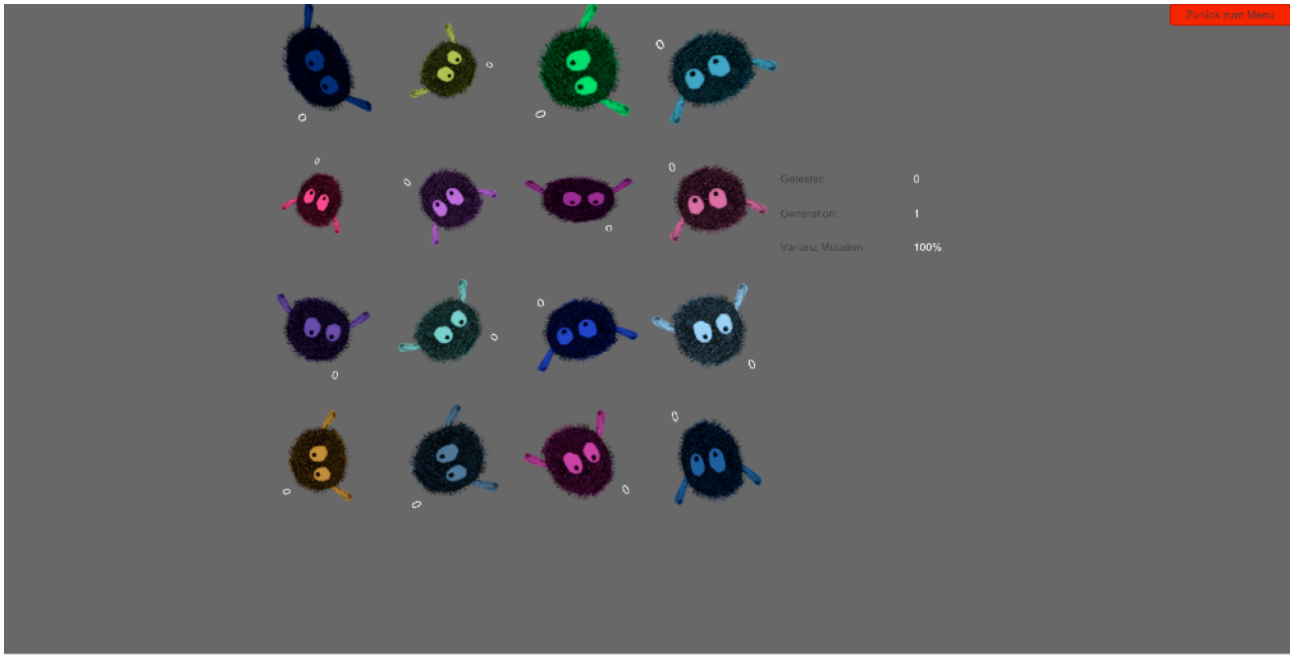
ÜBUNG 7.2-1:



Zuerst müssen wir uns selber ein Ziel setzen. Du siehst eine kleine Kreatur, die man über 6 Parameter in Farbe und Form konfigurieren kann. Erschaffe dir nun die Variante der Kreatur, die dir am besten gefällt. Diese soll der genetische Algorithmus im Folgenden für dich finden. Merke dir nun gut das Aussehen deiner Kreatur und klicke auf „Fertig. Spiel starten“.

Der Computer erzeugt nun eine zufällig initialisierte Population von 16 Kreaturen. Vermutlich ist noch keine dabei, die deiner Wunschkreatur wirklich ähnlich sieht. Wähle nun durch Klicken diejenige Hälfte der Kreaturen aus, welche deiner Wunschkreatur insgesamt am nächsten kommt. Sobald du die 8. Kreatur angeklickt hast, erzeugt der Computer sofort ein neues Bild. Darauf sind die von dir gewählten Kreaturen zu sehen, aber auch 8 neue, die der Computer aus diesen durch Mutation erzeugt hat. Sie sind den von dir gewählten Kreaturen ähnlich. Insgesamt sind die Kreaturen dem Wunschbild nun etwas ähnlicher geworden!
Wiederhole diesen Vorgang einige Male. Mit etwas Glück landest du ziemlich schnell bei einer Kreatur, die deinen Wünschen sehr nahe kommt.

Eine Frage noch: Welche Schritte im genetischen Algorithmus hast du übernommen?



ÜBUNG 7.2-2:

A. Wo in der Biologie haben die KI-Forscher diesen Algorithmus abgeschaut (Stichwort „Charles Darwin“)?
B. Welcher Mechanismus kommt in der Natur noch dazu, der in unserem Algorithmus fehlt?

7.3 Der „exploitation / exploration trade-off“ beim genetischen Algorithmus

Auch beim genetischen Algorithmus gibt es den „exploitation / exploration trade off“.

ÜBUNG 7.3 (SCHWER):

Versuche für die folgenden Massnahmen herauszufinden, ob sie den „exploitation / exploration trade off“ eher in Richtung mehr „exploitation“ oder in Richtung mehr „exploration“ verschieben:

A. Erhöhen der Mutationsrate (= Stärke der Abänderungen im Mutationsschritt)
B. Elite als nur 30% der Population wählen (statt 50%)

7.4 Die ideale Grösse der Generation. Hyperparameter

Die Grösse der Population zu wählen, ist nicht ganz einfach. Wählt man eine sehr grosse Population, ist es wahrscheinlicher, dass man ein gutes Resultat für die Parameter findet. Allerdings steigt auch der Rechenaufwand für jede Generation an (wir müssen jede Parameter-Kombination in der Simulation testen).

Wie sollen wir die Grösse der Population nun wählen? Was ist ein guter Kompromiss?

Eine Faustregel besagt, dass die Grösse der Population ca. 10 Mal der Anzahl der Parameter entsprechen sollte. D. h. je mehr Parameter es gibt, desto grösser muss die Population gewählt werden.

ÜBUNG 7.4:

A. Ist die Generationsgrösse im Kreaturen-Spiel zu gross, ideal oder zu klein?
B. Wie gross sollten wir die Population für „Mottis“ KNN ca. wählen?

Es ist in der künstlichen Intelligenz sehr häufig so, dass die Algorithmen selber Parameter haben (wie eben z. B. die Grösse der Generation). Man nennt solche Parameter **Hyperparameter**.

Dies kompliziert leider das Problem: Man muss nun neben den Parametern des neuronalen Netzes auch noch die optimalen Parameter für den Algorithmus schätzen oder suchen. Auch deshalb erfordert die Entwicklung einer KI oft sehr viel Rechenleistung. Nicht selten werden zum Trainieren Hunderte von Computern verwendet.

8. Grundlagen der Robotik

Das Wort „**Robot**“ wurde 1920 vom tschechischen Künstler und Schriftsteller Josef Čapek geprägt („robota“ bedeutet auf Tschechisch ‚Zwangsarbeit‘) während der russisch-amerikanische Science-Fiction-Autor **Isaac Asimov** im Jahre 1942 erstmals das Wort „Robotik“ erwähnte.
Roboter ermöglichen die Interaktion von Computern mit der realen Welt und nehmen dem Menschen schwere oder monotone Arbeiten ab.

Ein Roboter besteht stets aus den folgenden Komponenten:

1. **Sensoren**
   Diese wandeln physikalische Grössen (z.B. Luftdruck, Lichtstärke, Leitfähigkeit etc.) in elektrische Signale um. Auch Kameras und Mikrofone sind Sensoren.

2. **Aktoren**
   Sie wandeln elektrische Signale in physikalische Grössen um (also z. B. in Bewegung). Beispiele für Aktoren sind Räder oder Greifarme.

3. **Steuerung**
   Sie empfängt die Signale der Sensoren und berechnet, wie die Aktoren anzusteuern sind. Heute fast immer ein digitaler Computer.

ÜBUNG 8.-1:

Identifiziere die obigen Komponenten bei unserem Roboter „Motti“:

- Welche Sensoren hat er?
- Welche Aktoren hat er?
- Aus was besteht seine Steuerung?

ÜBUNG 8.-2:

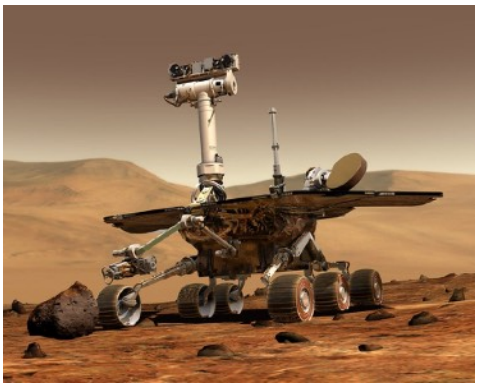
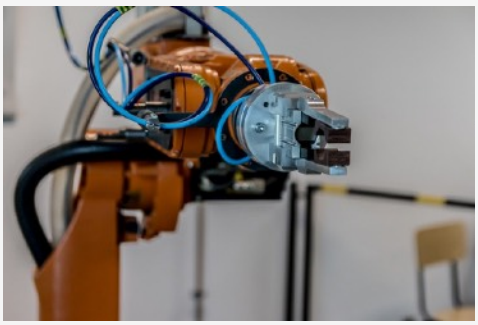
Wenn man den Menschen als eine Art „biologischen Roboter“ betrachtet, was wären dann die menschlichen Entsprechungen der obigen Komponenten:

A. Was entspricht den Sensoren?
B. Was entspricht den Aktoren?
C. Was entspricht der Steuerung?

ÜBUNG 8.-3:

Welche Arten von Robotern kennst du?

Verbinde die Fotos mit den richtigen Bezeichnungen.

Drohne		
Mars-Rover		
Humanoider Roboter		
Industrie-Roboter		

9. Wir trainieren ‚Motti' in einer virtuellen 3D-Umgebung

Nun haben wir endlich das ganze nötige Wissen, um selbst eine künstliche Intelligenz erschaffen zu können!

Um eine gute Parameter-Kombination zu finden, möchten wir unseren genetischen Algorithmus einsetzen. Wir könnten die Kombinationen mit dem physischen „Motti" evaluieren (d. h. testen). Das würde aber sehr sehr lange dauern:

- Wir haben nur einen einzigen Roboter zur Verfügung.
- Wir müssen die Parameter des KNN vor jedem Test auf den Roboter kopieren.
- Wir müssten den Return (d. h. die von den Sensoren gelieferte totale Lichtmenge während einer Episode von z. B. einer Minute) aus „Motti" nach jeder Episode auslesen.

Dieses Problem löst man in der Regel so, dass man eine Art 3D-Computerspiel für den Roboter programmiert. Das hat viele Vorteile:

- Man kann viele virtuelle Roboter gleichzeitig testen.
- Man kann, wenn man einen schnellen Computer hat, sogar im Zeitraffer testen.
- Der Roboter kann nicht beschädigt werden und benötigt keinen Strom. Es ist also viel billiger.
- Das ist evtl. sicherer (falls es sich um einen gefährlichen Roboter handelt).

Zum Glück haben wir für „Motti" eine solche Software:

ÜBUNG 9.-1:

Starte das Spiel. „Trainer für zweirädrigen Autoroboter" und klick einfach einmal auf „Start":

Du siehst, dass vier Roboter auf eine Fläche heruntergelassen werden, die nur teilweise von Lampen beleuchtet ist. Es ist wichtig, dass sich die Roboter flüssig bewegen, nur dann wird die Physik des Roboters korrekt berechnet.
Rechts unten hat es eine Anzeige der „frames per second“ (FPS). Einige von euch werden diese Angabe von Computerspielen kennen: Sie zeigt uns an, wie oft der Bildschirm pro Sekunde neu aufgebaut werden kann. Je schneller der Computer, desto höher ist diese Zahl. Bei uns sollte der Wert ca. 15 betragen. Ist die Zahl höher, können wir zusätzliche Roboter simulieren. Ist der Wert hingegen oft kleiner als ca. 15, sollten wir weniger Roboter gleichzeitig simulieren.

ÜBUNG 9.-2:

Unter dem Menü „Experiment“ kannst du die Anzahl der gleichzeitig simulierten Roboter einstellen:

- Wähle im Pulldown-Menu „Konfiguration Gitter“ eine passende Gitter-Grösse, so dass die FPS-Anzeige nach einigen Sekunden ungefähr 15 anzeigt. Je nach Leistungsfähigkeit des Computers kann man also 4 bis 36 Roboter gleichzeitig evaluieren.

Leider können wir nicht alle Roboter einer Generation gleichzeitig evaluieren. Aufgrund der Anzahl Parameter von „Mottis“ KNN müssen wir die Anzahl Roboter in einer Generation deutlich grösser als die max. möglichen 36 Roboter wählen (du erinnerst dich sicher an die Faustregel dazu). Wir teilen daher jede Generation in mehrere Gitter auf, welche nacheinander evaluiert werden.

- Du kannst nun die Anzahl Gitter pro Generation wählen und die Software berechnet für dich die Anzahl Roboter pro Generation.

- Als letztes wählen wir die Dauer der Episode, während der wir evaluieren. Ca. 4 Sekunden sind für den Anfang ein guter Wert (wählt man eine längere Zeit, kommt es vermehrt zu Kollisionen zwischen Robotern, was unerwünscht ist. Eine kürzere Zeit reicht kaum, um das Verhalten des Roboters gründlich beurteilen zu können).

Der Computer zeigt nun an, wie lange wir für das Evaluieren einer Generation brauchen.

- Als nächstes müssen wir unter „Training“ entscheiden, über wie viele Generationen wir trainieren wollen. Ca. 100 Generationen reichen meistens. Danach verbessert sich der Roboter meistens nur noch sehr wenig und es lohnt sich nicht, fortzufahren.

- Die restlichen Hyperparameter unter „Training“ lassen wir so, wie sie sind.

- Unter „Neuronales Netz“ wählen wir die Anzahl der Neuronen im „hidden layer“. Wähle eine Zahl zwischen 6 und 12 (je mehr Neuronen du hast, desto mehr Parameter müssen gefunden werden und desto grösser sollte deine Generation sein. Das Trainieren dauert dann entsprechend länger. Dafür wird, mit etwas Glück, „Motti“ intelligenter). Die anderen Hyperparameter ändern wir nicht.

- Jetzt klicken wir auf „Start“ und warten bis ca. 20 Generationen evaluiert wurden.

Nun sehen wir, dass die Roboter langsam lernen, das Licht zu suchen! Unter „Statistik“ können wir den Lernprozess genau verfolgen. Der grüne Punkt steht für den durchschnittlichen Return pro Roboter.

Es ist aber so, dass wir den Roboter in jeder Generation mit einem anderen Problem konfrontieren (sonst würde er nur in einer bestimmten Situation gut funktionieren und in allen anderen nicht!). Wir verändern z. B. von Generation zu Generation die Position und die Helligkeit der Lampe und die Stellung des Roboters.
Dies bedeutet aber auch, dass die Situationen für den Roboter unterschiedlich schwierig zu lösen sind. Deshalb ist der Return von Generation 20 nicht unbedingt höher als derjenige der 5. Generation.

Was wir daher machen, ist folgendes: Alle 10 Generationen testen wir die Roboter auf spezielle Weise. Normalerweise evaluieren wir alle Roboter einer Generation mit der gleichen Lichtsituation, denn sonst könnten wir die Roboter gar nicht vergleichen und so eine Elite bestimmen. Im speziellen Test geht es uns aber nur darum herauszufinden, wie gut die Roboter im Mittel sind, wir machen keine Selektion und Mutation. Wir setzen daher beim speziellen Test jeden Roboter einer anderen Lichtsituation aus, aber machen dies bei jedem Testlauf immer wieder gleich. Nun können wir die Generationen, die wir einem Spezialtest unterzogen haben, direkt vergleichen!

- Lass den genetischen Algorithmus ca. 100 Generationen lang laufen (das kann durchaus 1-4 Stunden dauern).

Schau dir nun die Statistik an und schalte per Pulldown-Menü auf „Tests“. Nun sollte klar erkennbar sein, dass die Returns immer grösser werden. Ist dies nicht der Fall, hat etwas nicht funktioniert und der Roboter lernt nicht oder zu langsam.

ÜBUNG 9.-3:

Jetzt transferieren wir das erzeugte neuronale Netz auf unseren „Motti“. Die Trainings-Software speichert für jede Generation eine Datei auf dem Computer ab, welche alle gefundenen KNN-Parameter (also die Gewichte und Biases für alle Neuronen) enthält. Wir kopieren uns die Datei der letzten Generation, indem wir auf den Download-Button klicken (vorher unbedingt die richtige Computersprache wählen. TypeScript ist korrekt für LEGO und micro:bit).

Nun müssen wir ein Programm für das micro:bit (bzw. unseren LEGO Brick) erzeugen, welches diese Parameter enthält. Die Parameter sind in einer Computersprache namens **JavaScript**[4] geschrieben.
Wir machen dazu folgendes: Wir besuchen die Programmierseite:

[4] Genau genommen eine Erweiterung davon, das sog. *TypeScript*

https://makecode.microbit.org (micro:bit)

bzw.

https://makecode.mindstorms.com (LEGO)

Dann laden wir das „Motti“-Programm aus dem Kit hoch und schalten den Editor in den JavaScript-Modus (Schalter oben rechts). Nun entfernen wir den ganzen Programmblock welcher mit „Automatisch erzeugt“ markiert ist (das ist das Standard-KNN, mit welchem die „Motti“-Software geliefert wird) und ersetzen ihn durch den Code in unserer kopierten Datei. Jetzt können wir das neue Programm auf unseren Computer herunterladen.

Nun nehmen wir das micro:bit aus „Motti“ heraus, verbinden es mittels USB-Kabel mit dem Computer und kopieren unser neues Programm darauf. Bei Mindstorms verbinden wir ebenfalls den Brick mittels beiliegendem USB-Kabel mit dem Computer und kopieren die Datei drauf. Wir setzen das micro:bit nun wieder (auf die richtige Orientierung achten!) in den Roboter ein.

Jetzt können wir unser selber trainiertes künstliches Gehirn mit „Motti“ testen!

Bravo, gut gemacht!

Du wirst sehen, dass „Motti“, je nach installiertem KNN, einen etwas anderen Charakter hat.

ÜBUNG 9.-4 (SCHWER):

Schau dir das „Motti“-Programm für das micro:bit bzw. Mindstorms genau an.

A. Wo befinden sich die KNN-Parameter?
B. Wo befindet sich der Code, welcher das KNN realisiert?
C. Wo befindet sich die Aktivierungsfunktion für das „hidden layer“? Es ist das sog. „Leaky ReLU“ programmiert:
   Dieses ist wie folgt definiert: Die Output-Zahl ist…

   1. … 0.1 mal die Input-Zahl für alle Input-Zahlen kleiner oder gleich 0
      und
   2. … gleich der Input-Zahl für alle Zahlen grösser als 0

D. Wo befindet sich die Aktivierungsfunktion für das „output layer“?

---

10. Lokale Optima und Reward-Funktion

Wie du sicher beobachtet hast, ist „Motti" mit dem selbst trainierten KNN ziemlich quirlig. Er dreht oft einfach an Ort und Stelle. Das ist nicht schlimm, aber nicht ganz das, was wir uns vorgestellt haben. Aber ist es nicht unsere Schuld? Schliesslich haben wir ‚Motti' nicht verboten, sich ständig zu drehen.
Dies passiert im ‚Reinforcement Learning' häufig: Man definiert eine Reward-Funktion (d. h. in welchen Zuständen der Umgebung soll dem Agenten welcher Reward gegeben werden) aber der Agent….

… findet einfach keine gute Lösung.

(oder)

… findet eine Lösung mit unerwünschten Nebeneffekten, an welche wir nicht gedacht haben.

Wir werden nun die Reward-Funktion für „Motti" so erweitern, dass er sich nicht mehr ständig dreht. Und zwar werden wir ihn für Drehungen bestrafen. D. h. nach jedem Frame (d. h. Bildschirmaufbau) messen wir, wie stark sich „Motti" gedreht hat und ziehen vom Reward aus dem Lichteinfall einen dazu proportionalen Wert ab (blau):

Reward = KonstanteA * (Summe der Lichtstärken) - KonstanteB * Drehung

ÜBUNG 10.:

Die Konstante B (Stärke der Bestrafung für Drehung) können wir in der Software einstellen. Stell ‚Drehung bestrafen' auf einen Wert von ca. 4000 und trainiere so (alle anderen Einstellungen wie vorher) ein neues KNN.

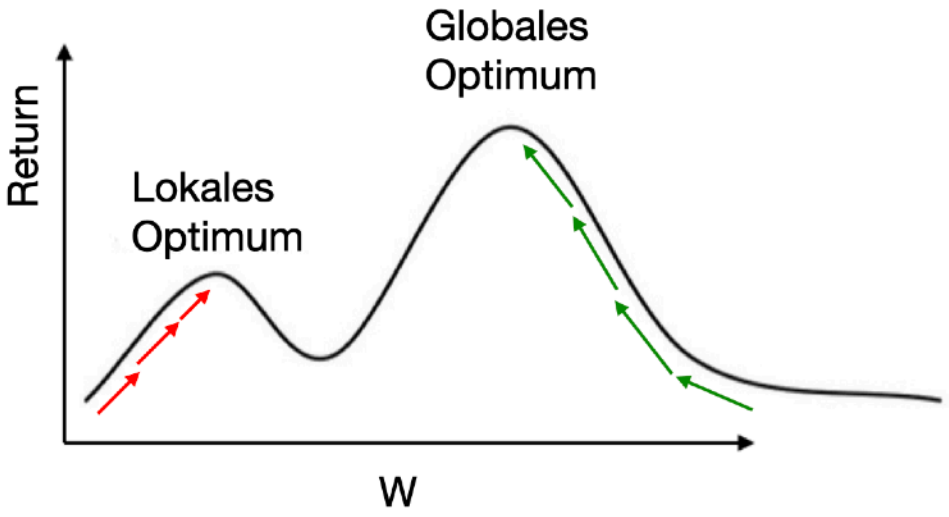
In der Statistik siehst du drei Punkte pro Generation / Test:

- Grün für die Belohnung aus der Lichtstärke
- Rot für die Bestrafung aus der Drehung
- Orange für den Return (d. h. beide Terme zusammen)

Wenn du das neue KNN auf ‚Mottis' Computer installiert hast, wirst du feststellen, dass der Roboter nun viel ruhiger ist und auch besser das Licht findet.

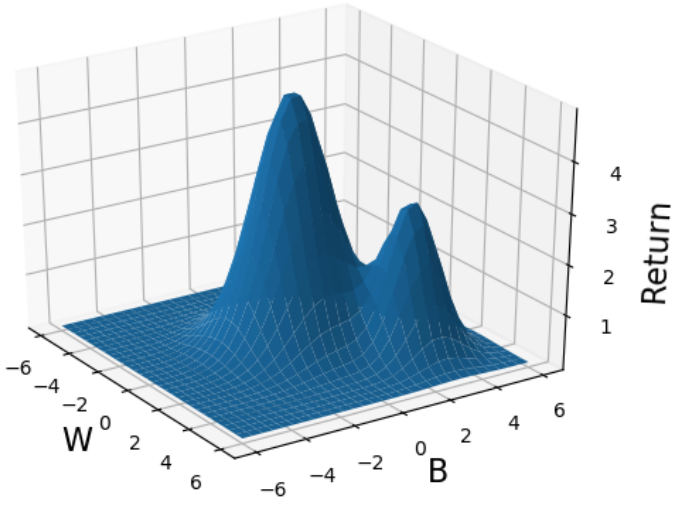
Warum zieht ‚Motti' ohne diese Bestrafung die Lösung mit viel Drehung vor? Weil der Roboter bei starker Drehung mehr oder weniger an Ort und Stelle bleibt. Und das ist leider schon eine ganz ordentliche Lösung für das Problem, da das Licht ja nicht allzu weit weg vom Roboter ist. Man sagt, die schlechtere Lösung ist ein sog. schlechtes **lokales Optimum**.

Man kann das auch mathematisch betrachten:



Die x-Achse (d. h. horizontal) stellt hier einen Parameter dar (ein Gewicht W oder ein Bias B), die y-Achse (vertikal) den Return. Der Agent sollte eigentlich ein gutes lokales Optimum oder sogar das globale Optimum finden, bleibt aber in einem schlechten lokalen Optimum stecken. Im Diagramm passiert das, wenn wir links anfangen zu suchen (d. h. bei kleinen Werten für W) und uns nach rechts bewegen (rot). Wenn wir von rechts anfangen zu suchen (d.h. bei grossen Werten von W), finden wir hingegen das globale Optimum (grün).
Beachte, dass wir diese Grafik beim richtigen Trainieren natürlich nicht kennen, wir stellen uns für diese Überlegung einfach vor, dass sie uns bekannt sei.

Natürlich müssen wir eigentlich alle Parameter berücksichtigen und nicht nur einen. Wir können dies aber nicht mehr in einem Bild darstellen. Für 2 Parameter (z. B. ein einzelnes Neuron mit einem einzigen Eingang, d. h. einem Gewicht W und einem Bias B) geht es gerade noch:



Richtige KNN haben aber, wie wir nun wissen, sehr viel mehr Parameter, so dass man das unmöglich visualisieren kann.

Tatsächlich findet man in der Praxis das globale Optimum eigentlich praktisch nie. Wie wir früher gesehen haben, ist dafür die Suche im hochdimensionalen Raum der Parameter viel zu schwierig. Aber wie wir selber erlebt haben, kann man ganz ordentliche lokale Optima finden, was für praktische Zwecke völlig ausreicht.


1. e2–e4 c7–c6 2.
–d5 3. Sb1–c3 d5xe4 4.
Sc3xe4 Sb8–d7 5.
Sg8–f6 6. Lf1–d3 e7–e6 7.
Sg1–f3 h7–h6?!
6! Dd8–e7 9. 0–0 f7xe6 10.
Ld3–g
7–b5 12. a2–a4
Lc8
Lf4–g3 Kd8–c8 15.
a4x
6 17. Lg6–f5 e6
18. Te1

11. KI in unserem Alltag

11.1 Suchmaschinen

In Suchmaschinen (z. B. Google) wird KI für folgende Aufgaben eingesetzt:

- Berechnen des sogenannten „site-rankings“ (wie wichtig eine Seite in Bezug auf ein bestimmtes Stichwort ist, damit die Suchresultate entsprechend sortiert werden können).
- Das Verstehen von Suchanfragen.
- Das Verhindern von Betrug. Gelegentlich versuchen Website-Betreiber durch verschiedene Tricks ihr Site-Ranking zu manipulieren. KI wird eingesetzt um solche Versuche aufzudecken.

11.2 Soziale Medien

Soziale Medien benutzen KI für eine Vielzahl von Aufgaben:

- Personen vorschlagen, die du vielleicht kennst.
- Allgemein Vorschläge berechnen für interessante Gruppen, Videos, Posts etc.
- Personen in Gruppenbildern automatisch markieren (sog. „facial recognition“).
- Die emotionale Stimmung von Beiträgen identifizieren („sentiment analysis“).
- Einblendung von Werbung, die möglichst zu einem Kauf führt („recommender systems“).

Dass soziale Medien vor allem Inhalte zeigen, die den persönlichen Interessen entsprechen, gilt als problematisch: Es lässt uns in einer sog. **Filterblase** leben (engl. „**filter bubble**“), in welcher wir in unseren bisherigen Ansichten immer weiter bestärkt werden. Dies kann bei manchen Menschen zu Extremismus führen.

11.3 Marketing

KI wird intensiv im Online-Marketing eingesetzt. Dabei werden möglichst viele Daten über dich gesammelt (z. B. welche Webseiten du besucht hast und welche Produkte du in Online-Shops angeschaut hast). Dann versucht die KI, dir möglichst solche Werbung zu zeigen, die dich zu einem Kauf animiert. Dabei werden oft enorme Datenmengen analysiert („**big data analytics**“).

11.4 In der industriellen Produktion (Robotik)

KI wird in der Industrie zur Steuerung von Robotern und anderen Produktionsmaschinen eingesetzt. Ein wichtiges Gebiet dabei ist das **maschinelle Sehen** („**machine vision**“), z. B. zur Erkennung von Defekten in Produkten oder für die Vollständigkeitsprüfung vor der Verpackung.

11.5 Sicherheitstechnik

In der Sicherheitstechnik wird KI zur Gesichtserkennung eingesetzt („facial recognition“). Damit werden Zutrittskontrollen oder eine Überwachung des öffentlichen Raumes realisiert. In einigen Ländern wird KI auch für Zensuraufgaben (z. B. in sozialen Medien oder Nachrichten-Apps) eingesetzt. Kreditkartenfirmen setzen KI zur Betrugserkennung ein. So können z. B. gestohlene Karten anhand von Käufen identifiziert werden.

11.6 Selbstfahrende Autos

Fast alle grossen Autofirmen arbeiten intensiv an der Entwicklung selbstfahrender Autos („self-driving cars“). Die dabei verwendeten KI Systeme verwenden oft LIDAR-Sensoren (das funktioniert ähnlich wie RADAR, aber mit Licht anstatt Radiowellen), um die Umgebung zu scannen. Zum Training wird oft „Reinforcement Learning“ verwendet.

11.7 KI und Sprache

KI kann auch zur Verarbeitung und Synthese von Text und gesprochener Sprache verwendet werden, z. B.:

- Automatische Übersetzungen von einer Sprache in eine andere („machine translation“).
- Verstehen von gesprochenem Text („**speech to text“**).
- Vorlesen von Texten („**text to speech“**).
- **Chatbots**, die einen geschriebenen Dialog mit dem Computer erlauben (z. B. Support in Online-Shops).
- Assistenten (Siri, Alexa, Cortana etc.), die einen gesprochenen Dialog mit dem Computer ermöglichen.

11.8 Bild-, Video- und Textsynthese

KI kann auch zur Synthese von Videos und Bildern verwendet werden. Dabei ist es z. B. möglich, in einem Video Gesicht und Stimme eines Protagonisten gegen diejenigen einer anderen Person auszutauschen (**„deep fakes“**). Es können auch völlig neue Bilder erzeugt werden (z. B. Porträts von Personen, die gar nicht existieren). Personen oder Objekte können so aus Bildern entfernt werden, dass dies nur sehr schwer zu erkennen ist.

Auch Texte können durch KI generiert werden. So werden z.B. im Internet Tausende von Artikeln automatisch erzeugt, um z. B. Suchmaschinen eine grosse Bedeutung eines Produktes zu vorzutäuschen.

11.9 In der Militärtechnik

KI wird heute schon in der Kriegsführung eingesetzt, z. B. zur automatischen Identifizierung von Zielen aus Bilddaten von Drohnen.

Viele Länder arbeiten auch schon an der Entwicklung von sog. **autonomen** intelligenten Waffensystemen, also an Waffen, die selbstständig Entscheidungen treffen. So könnten in Zukunft Drohnen ohne Kontrolle durch Menschen Ziele (z. B. feindliche Panzer) identifizieren und direkt angreifen.

Die Entwicklung solcher Waffen gilt als grosse Gefahr für die Menschheit. Es gibt Bestrebungen, autonome KI-Waffen per Völkerrecht zu verbieten, wie das z.B. bei chemischen und biologischen Waffen schon der Fall ist. Der Besitz von KI-Waffen verspricht jedoch derart grosse Vorteile auf dem Schlachtfeld, dass einige Staaten darauf nicht verzichten wollen. Dies setzt natürlich die übrigen Staaten unter Druck, diese Waffen ebenfalls zu erwerben oder zu entwickeln.

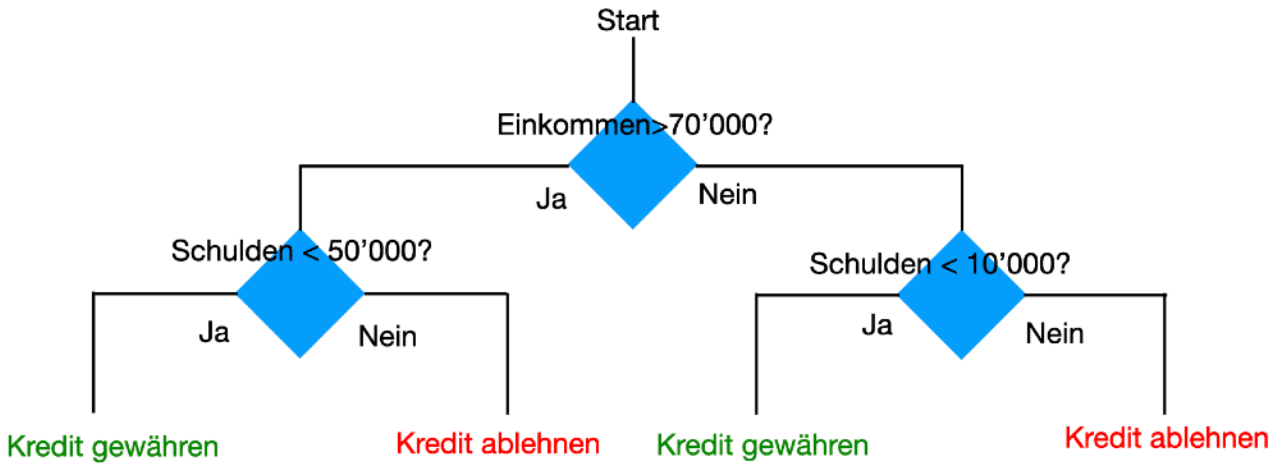

E.SUTER Basel
№39358
Rapid Aplanat №2

12. KI-Ethik und KI-Sicherheit

12.1 Begründbarkeit von Entscheidungen

Wir haben im Spiel „Spielen mit einem neuronalen Netz“ gesehen, dass KNN ihre Entscheidungen eigentlich mit Hilfe der Geometrie treffen. Diese kann bei grossen KNN sehr komplex werden (es gibt neuronale Netze mit Milliarden von Parametern). Es ist also u. U. sehr sehr schwierig herauszufinden, wie genau ein KNN zu einer Entscheidung gekommen ist. Das ist für viele Anwendungsfälle ein grosses Problem und Forscher auf der ganzen Welt versuchen, dafür eine Lösung zu finden.

Ein Lösungsansatz ist, dass man zusätzlich zum KNN noch ein anderes „machine learning“ Modell trainiert, welches Menschen besser analysieren können. Ein beliebtes solches Modell sind **Entscheidungsbäume** (engl. „**decision trees“**). Zu verstehen, wie ein Entscheidungsbaum funktioniert, ist recht einfach:



Er besteht aus einer Hierarchie von zu treffenden Entscheidungen, welche schliesslich zu den verschiedenen Outputs führen. Es gibt Algorithmen, die solche Entscheidungsbäume aus Daten automatisch erzeugen (d. h. lernen). Die Idee dabei ist, dass man das KNN benutzt, um die Entscheidung zu treffen und den Entscheidungsbaum, um die Entscheidung zu begründen. Das grosse Problem dabei ist, dass ein Entscheidungsbaum, der einfach genug ist, um in nützlicher Zeit von einem Menschen verstanden zu werden, kaum die ganze Komplexität eines grossen KNN abbilden kann. Die Begründung ist also stets simpler als die tatsächlichen „Überlegungen“ des KNN. Wirklich schwierig wird es bei KNNs, die über sog. „**superhuman“** (übermenschliche) Leistung verfügen. Dies ist heute schon möglich, wenn auch nur in sehr eng begrenzten Anwendungsgebieten. Dann ist das etwa so, wie wenn du deinem Hund erklären willst, wie du deine Mathe-Aufgaben gelöst hast. Er wird es unmöglich verstehen können, egal wie sehr du dich um eine einfache Darstellung bemühst.

Wir müssen uns zudem bewusst sein, dass das Problem grundsätzlich auch auf den Menschen zutrifft. Auch wenn wir die Motive für unsere Handlungen zu kennen glauben, werden diese doch oft vom sogenannten **Unterbewusstsein** gesteuert, zu dem wir kaum Zugang haben.

12.2 KI-Ethik

Wie wir gesehen haben, kann eine KI durchaus schlecht funktionieren und in bestimmten Fällen Fehler machen. Deshalb ist eine zentrale Forderung der Ethiker an KI-Systeme, dass diese **transparent** sind. Dies bedeutet z. B., dass man bei der Bank nachfragen kann, warum ein Bankkredit abgelehnt wurde, auch wenn diese Entscheidung von einer KI gefällt wurde. Die Forderung nach Transparenz bedingt also Begründbarkeit und das ist, wie wir gesehen haben, technisch nicht ganz einfach zu realisieren.

Eine weitere Forderung ist diejenige nach Fairness. So sollen KI-Systeme z. B. keine Menschen diskriminieren. Dies kann aber ungewollt einfach passieren. Nehmen wir z. B. an, dass eine Bank eine „deep learning"-Software zur Berechnung der Kreditwürdigkeit entwickelt hat. Die Software wird dazu mit Daten aus verschiedenen Regionen gefüttert inkl. Angaben, ob die Personen ihre Raten bei früheren Krediten stets bezahlt habe („supervised learning"). Stell dir vor, dass in bestimmten Regionen mehr arme Personen leben und gleichzeitig mehr Personen mit dunkler Hautfarbe. Der Lernalgorithmus eines „machine learning" Algorithmus könnte daraus den falschen Schluss ziehen, dass Menschen mit dunkler Hautfarbe nicht kreditwürdig sind und ihnen keine Kredite gewähren. Dies passiert leicht, wenn wir das System mit den falschen Daten füttern (z. B. mit der Hautfarbe anstatt der Region).

Es ist also wichtig, dass KI-Systeme nach dem Trainieren automatisch auf solche Diskriminierungen getestet werden. Das ist grundsätzlich möglich. Füttert man z. B. unser trainiertes Kredit-System mit einem Input, der einem Menschen mit bestimmten Eigenschaften und dunkler Hautfarbe entspricht, dann sollte das System sich gleich entscheiden, wie bei einem Menschen mit gleichen Eigenschaften, aber heller Hautfarbe. Um solche Tests durchführen zu können, muss aber die Variable „Hautfarbe" in den Daten vorhanden sein, was aus Gründen des Datenschutzes aber oft nicht erwünscht oder gar verboten ist.

Beachte, dass wir sehr vorsichtig sein müssen, auch wenn wir die Variable ‚dunkle Hautfarbe' gar nicht für das Training der KI verwenden, da das System vielleicht aus anderen, harmlos erscheinenden Variablen die Hautfarbe berechnen kann (z. B. aus dem Geburtsort) und dann intern verwendet. Dies kann sehr leicht geschehen, ohne dass man das bemerkt.

Diskriminierendes Verhalten ist auch möglich in Bezug auf das Geschlecht, das Alter und die Nationalität.

Eine weitere Forderung ist, dass KI-Systeme keine Schäden anrichten sollen. Dies führt uns zum nächsten wichtigen Thema, der KI-Sicherheit.

12.3 KI-Sicherheit

Wie wir gesehen haben, macht auch die beste KI manchmal Fehler. Es ist wichtig zu wissen, wie häufig diese Fehler vorkommen und welcher Art sie sind. Man macht das in der Regel so, dass man das KI-System an einem zweiten Datensatz testet, den man während des Trainings noch nie verwendet hat. Man teilt dazu die ganzen Daten vor dem Training in ein „**training set**“ und ein „**test set**“ auf (z. B. im Verhältnis 80% zu 20%). Wenn man mit dem Trainieren an den „training set“-Daten fertig ist, testet man das System, indem man es mit den Daten aus dem „test set“ füttert und sein Verhalten beobachtet.

Nehmen wir an, wir wollen ein System bauen, das anhand eines Röntgenbildes erkennen kann, ob jemand an Tuberkulose (eine Lungenkrankheit) erkrankt ist oder nicht (also einen Klassifikator). Das System wird zweierlei Arten von Fehlern machen:

- Bei einigen Menschen, die tatsächlich Tuberkulose haben, die Krankheit nicht erkennen (sog. „**false negatives**“)
- Bei einigen Menschen Tuberkulose diagnostizieren, die tatsächlich gesund sind (sog. „**false positives**“)

Der Schaden aus diesen beiden Fällen ist oft unterschiedlich gross. In unserem Beispiel ist z. B. ein „false negative“ viel schlimmer als ein „false positive“. Im ersten Fall kann der Patient schwer erkranken oder sterben, im zweiten Fall wird er bloss unnötig behandelt.

Es ist leider schwierig, beide Arten von Fehlern zusammen klein zu halten. So ist es z. B. sehr simpel ein System zu bauen, welches überhaupt keine „false positives“ erzeugt. Dies ist nämlich dann der Fall, wenn das System Patienten nie als krank diagnostiziert. Natürlich gibt es dann viele „false negatives“. Umgekehrt könnte man die „false negatives“ einfach auf null reduzieren, indem das System alle Patienten als krank klassifiziert. Dann gibt es natürlich viele „false positives“.
Oft ist es deshalb nötig, eine günstige Balance zwischen beiden Anforderungen zu finden.

Man kann dem unterschiedlichen Schaden von „**false negatives**“ und „**false positives**“ Rechnung tragen, indem man diese Fehler beim Trainieren unterschiedlich stark bestraft (im RL z. B. über den Reward).

ÜBUNG 12.3:

Eine KI soll aus Blutproben eine Krankheit diagnostizieren, die bei 1% der Menschen vorkommt. Dabei irrt sie sich in nur (?) 10% der Fälle.

Nehmen wir an, wir testen 10’000 Menschen.

A. Wie viele der getesteten Menschen sind tatsächlich krank, wieviele gesund?
B. Wie viele „true positives“ gibt es ca.?
C. Wie viele „false negatives“ gibt es ca.?
D. Wie viele „false positives“ gibt es ca.?
E. Ist die KI brauchbar?

13. Die Zukunft der KI

13.1 Das Moore'sche Gesetz

Gordon Moore publizierte 1975 seine Beobachtung, dass sich die Anzahl der Transistoren in integrierten Schaltungen (wie z. B. Mikroprozessoren) alle zwei Jahre verdoppelt. Dies bedeutet, dass die Leistung von Computern sehr schnell zunimmt (sog. exponentielles Wachstum). Wir sind dem exponentiellen Wachstum schon einmal begegnet: in der Zunahme des Suchaufwandes, wenn wir KNN-Parameter hinzufügen.

Die exponentielle Zunahme von Rechenleistung, Speichervolumen und Geschwindigkeit des Datentransfers hat auch Auswirkungen auf die Entwicklung der KI:

- Sie erlaubt uns, immer grössere KNN zu trainieren.
- Wir erzeugen immer grössere Datenmengen (z. B. aus sozialen Medien), die es uns ermöglichen, neue und komplexere KNN zu trainieren.

13.2 Soziale Auswirkungen und Überwachungsstaat

Experten rechnen damit, dass schon in 10 bis 30 Jahren viele, wenn nicht die meisten, menschlichen Tätigkeiten von Computern und Robotern übernommen werden können. Damit verliert die menschliche Arbeitskraft ihren wirtschaftlichen Wert. Es ist eine grosse ungelöste Frage, wie wir damit umgehen sollen. Es gibt z. B. Vorschläge, den Besitz von Robotern zu besteuern und mit den Einnahmen jedem Bürger ein sog. „bedingungsloses Grundeinkommen" zu ermöglichen. Um dieses zu erhalten, wäre also keine Arbeitsleistung erforderlich. Es ist jedoch heute so, dass sich viele Menschen über den Erfolg in ihrer Arbeit definieren. Wir können uns heute noch kaum vorstellen, wie es in Zukunft sein wird, wenn es keine Arbeitswelt mehr gibt.
Es droht vielleicht eine Spaltung der Menschheit in drei Gruppen:

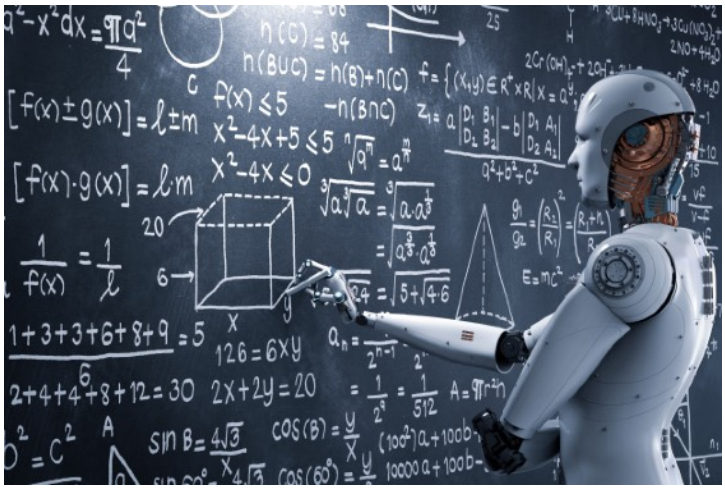
- Diejenigen, die die Roboter besitzen
- Diejenigen, die Roboter bauen und kontrollieren können
- Der Rest der Menschheit

Mit „deep learning" können schon heute Überwachungsaufgaben sehr effektiv automatisiert werden. So werden in einigen Ländern schon intensiv Kameras mit Gesichtserkennung in der Öffentlichkeit eingesetzt. Dies mag einige Vorteile bringen (z.B. Verhinderung von Kriminalität), birgt aber die grosse Gefahr, dass mit der Zeit ein Überwachungsstaat installiert wird, in welchem die Menschen auf Schritt und Tritt kontrolliert werden. Dieser könnte irgendwann so umfassend werden, dass eine perfekte Diktatur realisiert werden kann, gegen welche jede Rebellion scheitern muss.

13.3 ‚Artificial General Intelligence' (AGI)

Als ‚**Artificial General Intelligence**' bezeichnet man eine künstliche Intelligenz, die jede geistige Tätigkeit eines Menschen erledigen kann. Man nennt solche KI auch **starke KI** (im Gegensatz zur **schwachen KI**, die nur Teilaufgaben mit hoher Leistung erledigen kann). Heute gibt es erst schwache KI, allerdings in einigen Gebieten schon mit übermenschlicher (‚superhuman') Leistung. So können Computer z. B. schon viele Spiele (Brettspiele und Computerspiele) mit tief übermenschlicher Leistung spielen. Z. B. spielen Computer heute das extrem komplexe asiatische Brettspiel **Go** sehr viel besser als jeder Mensch. Eine Leistung, die noch vor wenigen Jahren für die meisten Experten als unmöglich galt.

Einige Experten halten die Entwicklung einer solchen starken KI für gefährlich. Sie wäre wohl durch Menschen kaum zu kontrollieren. Das wäre wie wenn Meerschweinchen auf Menschen aufpassen müssten, damit diese nichts Dummes anstellen können.



Was, wenn die Menschen einer solchen KI falsche Ziele setzen? Das berühmte Beispiel ist jenes vom Besitzer einer Büroklammern-Fabrik, der einer starken KI den Auftrag gibt, möglichst viele Büroklammern zu produzieren. Die KI beginnt dann sofort mit der Arbeit und lässt sich dabei durch nichts stoppen, bis die ganze Erde in einen Berg von Büroklammern verwandelt ist.

13.4 Kommt es zu einer Singularität?

Was würde passieren, wenn wir eine erste starke KI hätten? Wir könnten sie natürlich dazu einsetzen, eine noch bessere KI zu entwickeln. Nehmen wir an, wir brauchen 4 Jahre um damit eine KI zu entwickeln, die doppelt so intelligent wie ein Mensch ist (eine sog. **Superintelligenz**). Wenn wir diese KI wiederum dazu verwenden, die KI zu verbessern, brauchen wir vermutlich nur noch 2 Jahre um eine weitere Verdoppelung zu erreichen. Die nächste Verdoppelung mittels der so verbesserten KI dauert dann nur noch ein Jahr. Usw., usw.!

Manche Experten glauben, dass dieser Prozess zu einer sogenannten **Singularität** führen wird. Dies bedeutet, dass an einem bestimmten Punkt der Zukunft die Leistungsfähigkeit und Intelligenz von Computern explosionsartig zunimmt. Dies könnte so schnell gehen, dass sich in der Nähe dieses Zeitpunkts die Intelligenz einer KI innert Sekunden verdoppelt. Eine Zukunft jenseits dieser Singularität wäre schlicht nicht vorstellbar. Die menschliche Geschichte würde wahrscheinlich mit ihr enden.

Der Futurist **Ray Kurzweil** vermutet, dass die Singularität um das Jahr 2045 herum eintreten wird. Es gibt aber auch namhafte KI-Forscher, die sagen, das sei alles Unfug.

14. Künstliche Intelligenz und Philosophie

Wie wir schon bei der Geschichte des Intelligenzbegriffes gesehen haben, empfinden viele Menschen (darunter auch Philosophen) die Entwicklung intelligenter Maschinen als grosse Provokation. Der Mensch droht von seinem Thron zu stürzen, auf welchem er seit Tausenden von Jahren unangefochten sitzt. Es stellt sich damit die Frage, was denn genau den Menschen ausmacht. Was uns vielleicht auch in Zukunft speziell macht, wenn eines Tages Maschinen unsere ganze Arbeit verrichten.

Schauen wir uns doch einige Fähigkeiten an, die oft nur dem Menschen (oder höheren Tieren) zugeschrieben werden:

1. **Die Fähigkeit komplexe Probleme zu lösen und effektiv zu lernen**
   Man kann davon ausgehen, dass Computer und Roboter diese Fähigkeit erlangen werden. Teilweise ist dies, wie wir gesehen haben, sogar schon der Fall. Auf immer mehr Gebieten sogar mit „übermenschlicher“ Leistung.

2. „**Objektive Selbstaufmerksamkeit**“ (engl. „**self awareness**“).
   Dies bedeutet z.B., dass sich ein Wesen im Spiegel selbst erkennen kann. Es kann die eigenen Stimmungen analysieren und entsprechend das Verhalten ändern (z.b. „Heute bin ich etwas traurig, ich sollte im Park spazieren gehen“).
   Es ist sehr wahrscheinlich, dass in Zukunft KI-Systeme über ein Modell von sich selbst verfügen und damit diese Eigenschaft erlangen. Sie lässt sich durch Tests objektiv ermitteln und setzt vermutlich kein Bewusstsein (siehe unten) voraus.

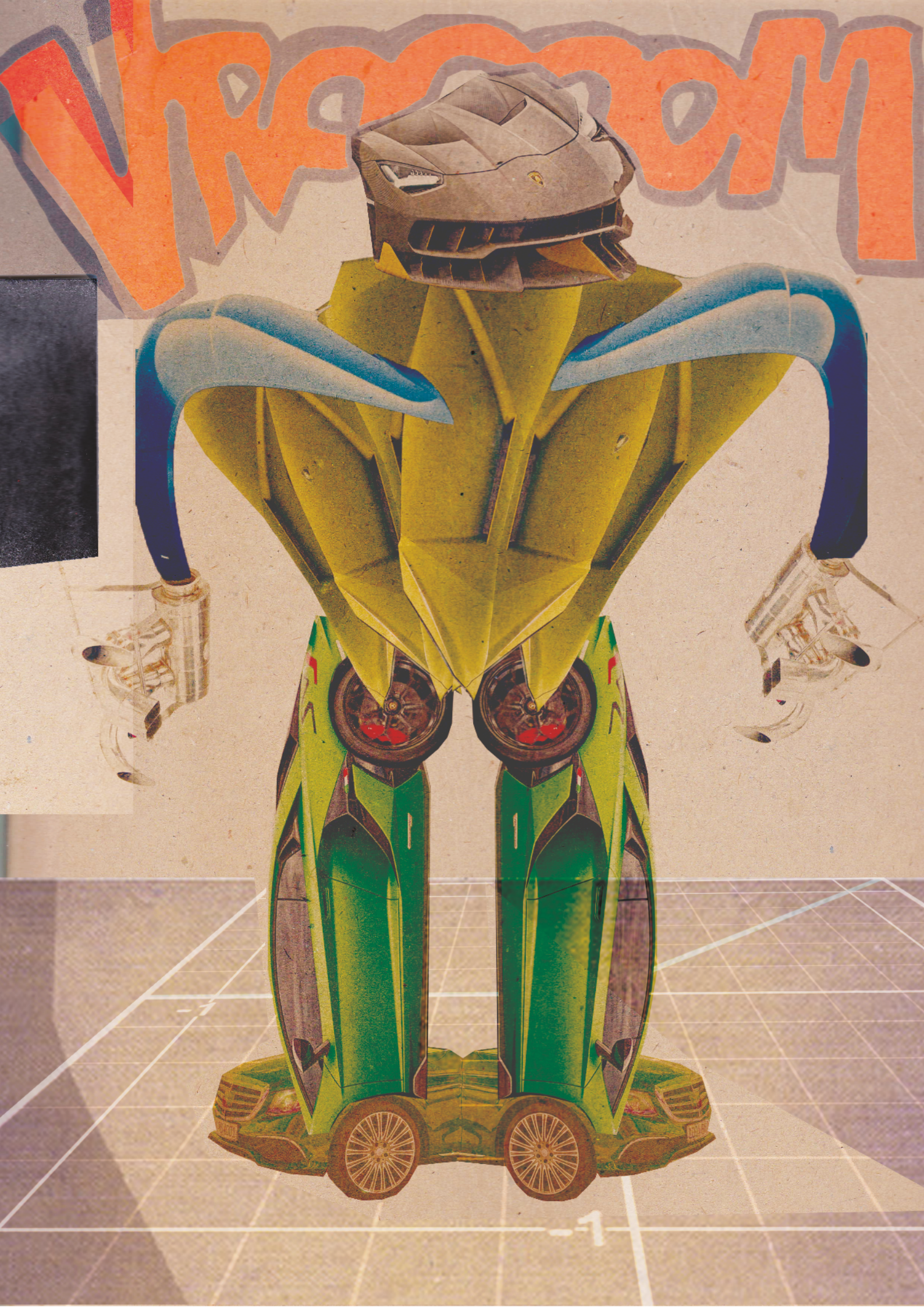
3. **Bewusstsein**
   Bewusstsein beinhaltet die Fähigkeit, Gefühle zu empfinden, Schmerz zu fühlen und Farben wahrzunehmen (der Philosoph nennt dies „**Qualia**“). Es ist unklar, ob Computer Bewusstsein erlangen können. Es gibt gute Argumente dafür: z. B. funktionieren biologische Gehirne letztlich auch elektrisch. Ein Beweis ist aber vielleicht unmöglich. Genau genommen sind wir nur bei uns selbst sicher, dass wir ein Bewusstsein haben. Von anderen Menschen können wir das grundsätzlich nicht wissen, auch wenn sie das von sich behaupten und es natürlich offensichtlich scheint. Es ist leicht, simple Maschinen zu bauen, die so reagieren, als würden sie Schmerz empfinden oder Gefühle haben (wir erinnern uns an Weizenbaums ELIZA). Aus dem Verhalten kann man also nicht auf die Existenz von Bewusstsein schliessen.
   Die Frage ist insofern wichtig, als wir uns vielleicht eines Tages entscheiden müssen, ob wir Robotern gewisse Rechte zugestehen wollen.

4. **Menschenähnliches Verhalten**
   Nach gegenwärtigem Stand des Wissens müsste man dazu die Reward-Funktion eines Menschen aus dem Verhalten rekonstruieren (in einem Prozess der „Inverse Reinforcement Learning“ heisst). Dies wäre mit Sicherheit ein extrem schwieriges Unterfangen und es ist unklar, ob das je machbar sein wird.
   Andererseits scheint es für gewisse eng begrenzte Tätigkeiten zu funktionieren: So können Computer heute schon ganz erstaunlich gute Texte (z. B. Geschichten) verfassen.

ÜBUNG 14. (SCHWER):

Suche im Internet nach Informationen zum Gedankenexperiment „das chinesische Zimmer“ des Philosophen John Searle.

- Wie funktioniert das Gedankenexperiment?
- Was meinst du: Welche der obigen vier Fähigkeiten könnte das Zimmer haben, welche nicht? Begründe deine Antwort.


VROOM

Bildnachweis

Ganzseitige Illustrationen: Dieter Kubli, https://www.kubligrafik.ch

S. 11: Shutterstock
S. 13: Nematode: Wikipedia, Public Domain
S. 13: Fruchtfliege: Pixabay
S. 13: Meerschweinchen: Pixabay
S. 13: Katze: Pixabay
S. 13: Bonobo: Pixabay
S. 14: Shutterstock
S. 20: Wikipedia, Public Domain
S. 27: Pixabay
S. 54: Pixabay
S. 64: Shutterstock
S. 65: Wikipedia, Public Domain
S. 71: Shutterstock
S. 72: Sutterstock

Künstliche Intelligenz und Robotik

Eine faszinierende Reise durch die Welt der künstlichen Intelligenz und Robotik. Der Inhalt des Buches wird mit eigens für diesen Kurs entwickelter Software (Serious Games) ergänzt. Viele praktische Übungen mit detaillierten Lösungen erleichtern das Verständnis. Alle wichtigen Konzepte und Algorithmen werden Schritt für Schritt genau erklärt und danach am Computer visualisiert und spielerisch erkundet.
Der Bau und das Trainieren eines Roboters (es sind Bauanleitungen und Software für die **Systeme LEGO Mindstorms EV3, micro:bit und Arduino** enthalten) macht viel Spass und macht die gelernten Konzepte konkret erlebbar.

Ein grosses intellektuelles Abenteuer für Jugendliche und ein fundierter Einstieg in ein wichtiges Zukunftsthema.

- Was ist Intelligenz?
- Wie funktioniert unser Gehirn? Was sind Neuronen?
- Was ist künstliche Intelligenz? Wer hat sie erfunden?
- Wie funktionieren künstliche neuronale Netze?
- Wie trainiert man einen Roboter in einer virtuellen 3D Umgebung?
- Was sind Algorithmen und wie funktionieren sie?
- Künstliche Intelligenz in unserem Alltag
- K.I.-Ethik und Sicherheit
- Die Zukunft der K.I.
- K.I. und Philosophie

Ein Lehr- und Bastelbuch für Jugendliche ab 13 Jahren