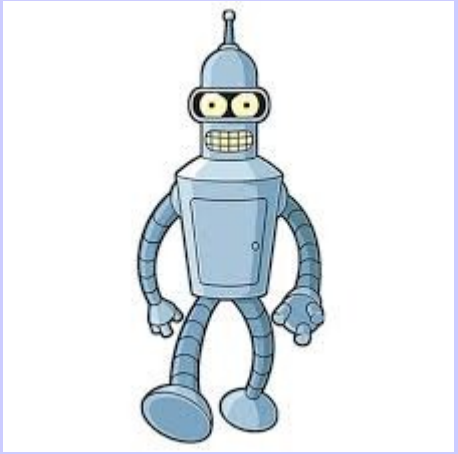
hochschule mannheim

Prof. Dr. Jörn Fischer
j.fischer@hs-mannheim.de

Willkommen zur Vorlesung

Künstliche Intelligenz für Autonome Systeme



Vorstellung

Zu meiner Person...

Überblick

Inhalt

- Organisatorisches
- Praktische Einführung Arduino und Raspberry PI
- Künstliche Intelligenz
  - Planung mit A*
  - Monte Carlo Tree Search
  - Genetische Algorithmen
  - Reinforcement Learning
  - Neuronale Netze

Organisatorisches

Fragen…

- Fragen können jederzeit an uns gerichtet werden...
- Rückkopplung ist ausdrücklich erwünscht !!!!!

Einführung

Ziel der Vorlesung

- ...ist, dass ihr maximal dabei lernt
- Dass ihr die Algorithmen versteht
- Dass ihr sie selbst programmieren könnt
- Dass Ihr sie auf den Robotern umsetzen könnt

Einführung

Studienleistung und Prüfung

- Ein ausgewählter Algorithmus (Lernen oder Planung) soll von 1-3 Personen auf einen Roboter der Wahl implementiert werden, so dass hierdurch eine autonome Funktion erkennbar wird.
- Eine wissenschaftliche Ausarbeitung auf 4-8 Seiten
- Die Prüfung findet in mündlicher Form statt, wobei das Projekt vorgestellt werden soll
- Außerdem sollen Fragen zum Umfeld des Projekts beantwortet werden

Organisatorisches

Fragen zu organisatorischen Dingen?

Organisatorisches

Vorschläge für mögliche Projekte

- Kartierung eines Labyrinths und auf dem kürzestem Weg zurückfahren
- Fahrzeugroboter mit Raspberry PI und Kamera lernt aufgrund von Bumper-Signalen Hindernissen auszuweichen
- Vierbeiner lernt laufen bzw. Hindernissen auszuweichen
- Vierbeiner lernt Hindernisse zu übersteigen
- Vierbeiner lernt auf Radgetriebene Platform zu steigen und fährt mit dieser davon
- Pioneer Roboter kartiert Umgebung
- Personenverfolgung mit beweglichem Kamerakopf
- Legoroboter lernt Linie zu folgen oder suchen sich!
- Roboterarm lernt Dinge zu greifen

---- weitere Ideen von eurer Seite sind gern gesehen ----

Organisatorisches

Literatur

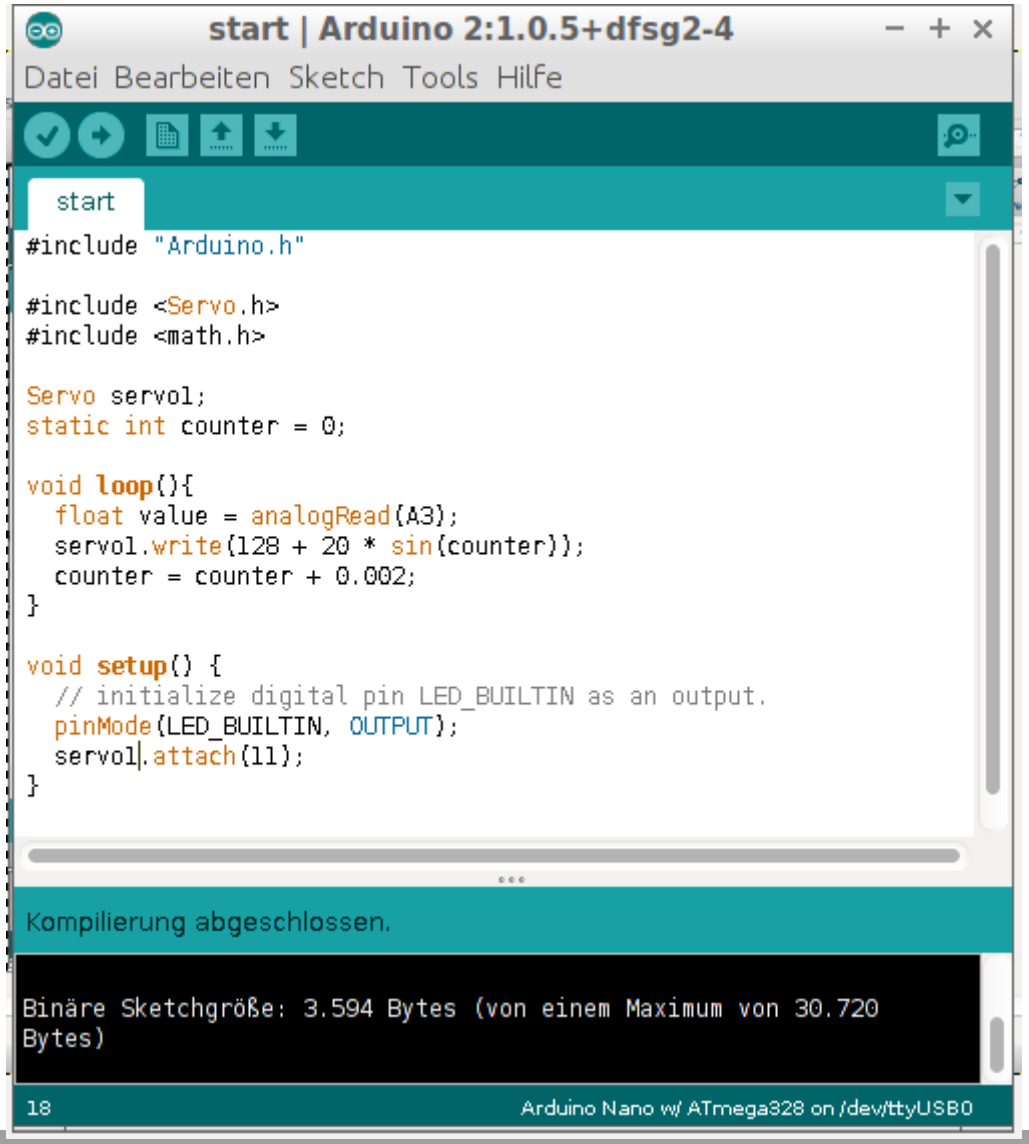
- I. Goodfellow, Y.Bengio, A. Courville : Deep Learning, MIT Press, ISBN: 978-0262035613, 2016
- Russel, Stuard; Norvig, Peter: Künstliche Intelligenz. Prentice Hall, New Jersey, 1995
- Mitchell, Tom: Machine Learning. McGraw-Hill, 1997
- Zell, Andreas: Simulation Neuronaler Netze. Oldenbourg Verlag, München, 1997
- Sutton, Richard; Barto, Andrew G.: Reinforcement Learning. MIT Press, 1998

Arduino und Raspberry Pi

Arduino und Raspberry Pi

Arduino

- Der Arduino ist ein Micro-controllerboard mit 8 Bit Befehlssatz und mehreren Digitalen und Analogen IO's
- Programmiert wird der Arduino in der Programmiersprache C
- Starten Sie die Programmierumgebung Arduino_IDE und tippen Sie das rechts stehende Programm ein.
- Dann schließen Sie den Vierbeinroboter an und drücken den Haken zum compilieren
- Hochgeladen wird das Programm mit der Pfeiltaste nach rechts



Arduino und Raspberry Pi

Aufgabe:

- Der Vierbeinroboter läuft, wenn die vorderen Beinen sich Sinusförmig und die hinteren Beinen sich Cosinusförmig bewegen.
- Vorher sollte aber für jedes Bein der Offset bestimmt werden, um den der Sinus/Cosinus schwingen soll!
- Ändern Sie das Programm dahingehend ab, dass sich der Roboter vorwärts bewegt!

Arduino und Raspberry Pi

Raspberry Pi

- Der Raspberry Pi ist ein Board mit einem ARM Prozessor
- Es ist ein vollständiger PC in Scheckkartengröße mit HDMI, USB, SD-Card und Kamera-Interface.
- Schließen Sie Mouse, Tastatur, Monitor und Netzteil an den Raspberry Pi an und Booten Sie das Linux System.
- Öffnen Sie den Filebrowser und erzeugen Sie ein neues Textfile mit dem Namen main.cpp. Editieren Sie dieses File gemäß Handout und Anleitung
- Öffnen Sie ein Konsolenfenster und compilieren bzw. starten Sie den Code

Arduino und Raspberry Pi

Aufgabe

- Programmieren Sie eine Fußgängerampel, die bei Drücken eines Tasters von Rot auf Grün umschaltet.

Künstliche Intelligenz

Künstliche Intelligenz

Inhalt

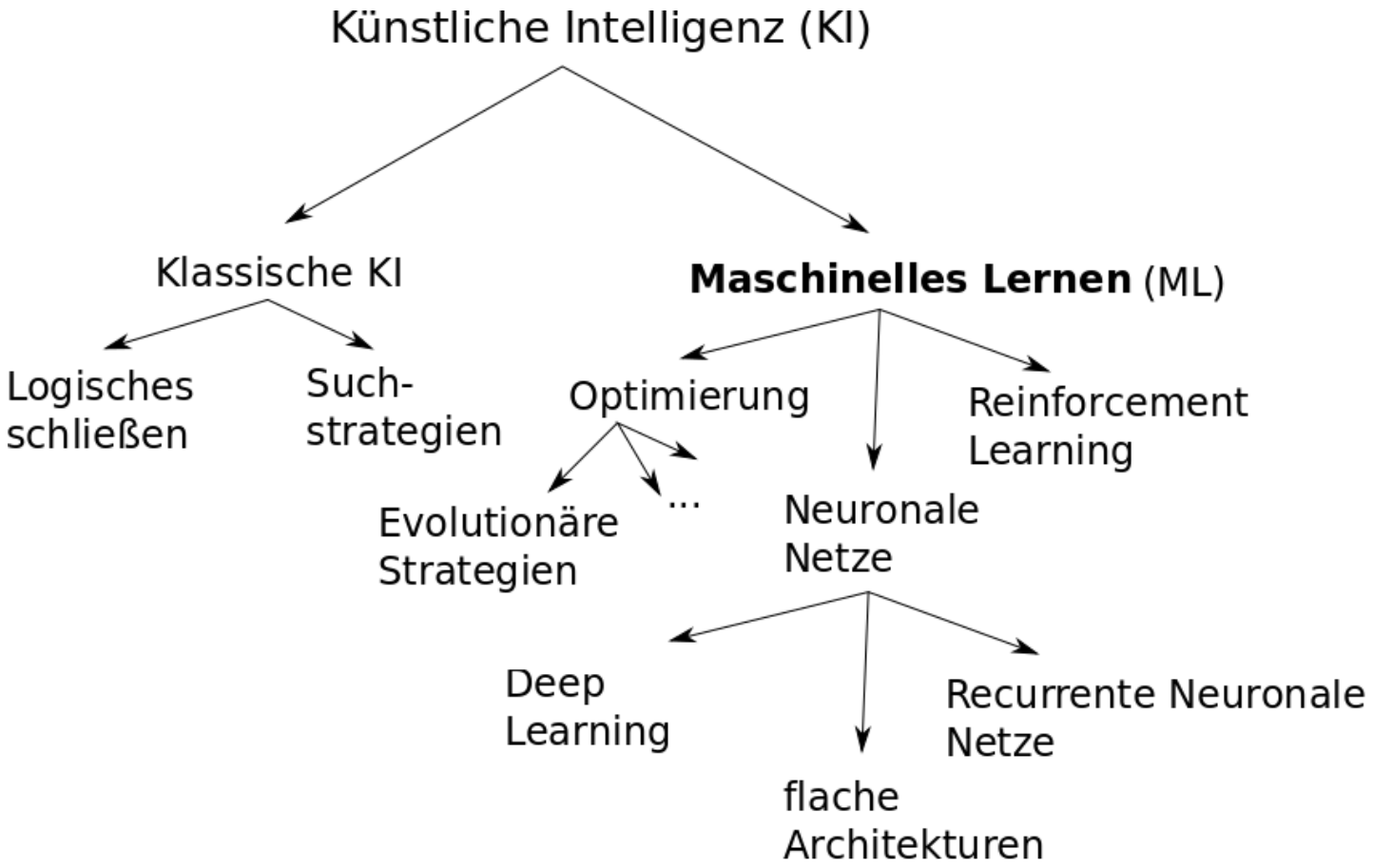
- Einführung
- Planung mit A*
- Monte Carlo Tree Search
- Genetische Algorithmen
- Reinforcement Learning
- Neuronale Netze

Einführung

Künstliche Intelligenz

Einführung

KI, ML, Deep Learning… wie hängt das zusammen?



Künstliche Intelligenz

Planung mit A*

Künstliche Intelligenz

Planung mit A*

- Beim A* Algorithmus befinden sich die Knoten K_i eines Graphen in einem der Zustände: (1) K_i unbekannt, (2) K_i in Warteschlange, (3) K_i abgearbeitet
- Zunächst legt man den Startknoten K_0 in die Warteschlange
- Jeder Knoten K_i hat f-Wert $f = dist[K_0,K_i] + estimatedDist[K_i, K_{goal}]$
- Der Knoten mit dem kleinsten f-Wert wird in der Warteschlange als nächstes abgearbeitet
- Ist es der Zielknoten, so ist der kürzeste Weg gefunden
- Ist die Warteschlange leer, so wurde kein Weg zum Ziel gefunden
- Ist der Knoten kein Zielknoten, so ist er abgearbeitet und seine Nachbarn werden betrachtet:
  - Ist der Nachbarknoten abgearbeitet, so passiert nichts
  - Ist der Nachbarknoten bereits in der Warteschlange, so wird geprüft, ob der Weg dorthin kürzer ist und ggf. der f-Wert aktualisiert
  - Ist er noch nicht in der Warteschlange, so wird er anhand seines f-Wertes eingereiht

Monte Carlo Tree Search

Künstliche Intelligenz

Monte Carlo Tree Search

- Beim Monte Carlo Tree Search werden ausgehend vom aktuellen Zustand (root node) Folgezustände (child nodes) **selektiert**, bis man bei einem nicht expandierten Knoten ankommt.
- Man **expandiert** diesen zufällig.
- Man **simuliert** das Spiel oder das Szenario bis zum Ende und ...
- ...aktualisiert alle traversierten Knoten (**Backpropagation**)

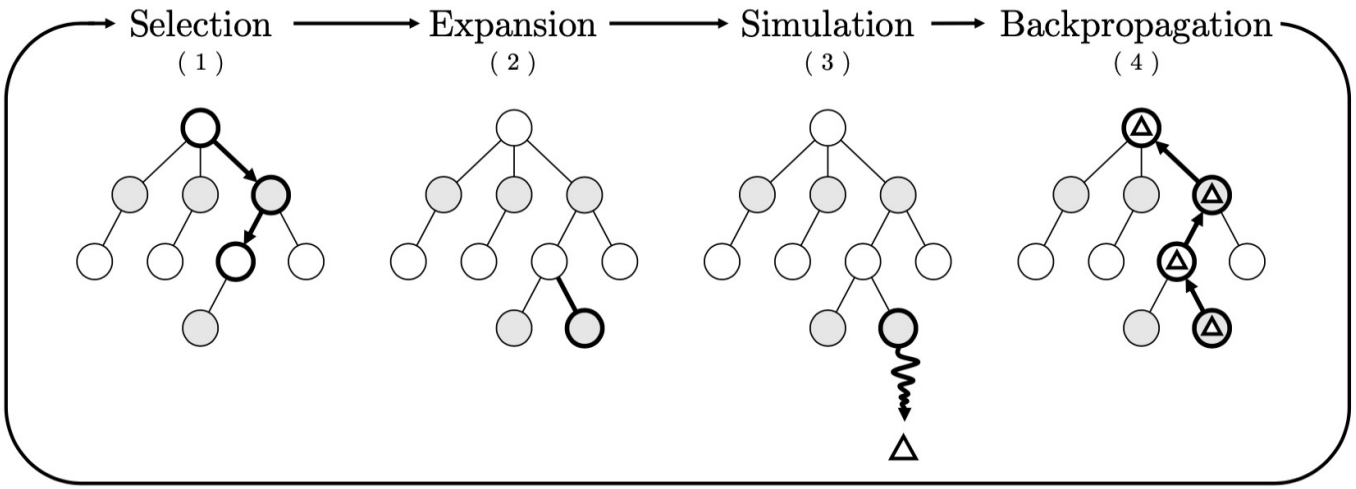


Abbildung aus der Bachelorarbeit von Moritz Vogt

Künstliche Intelligenz

Monte Carlo Tree Search

- Um bei der **Selektion** ein Gleichgewicht zwischen Exploration von unbekannten bzw. wenig besuchten Zuständen und Gewinnmaximierung zu bekommen nutzt man häufig die UCT Strategie:

$$UCT = \frac{w_i}{v_i} + C\sqrt{\frac{\ln(v_r)}{v_i}}$$

wobei w_i die Gewinnzahl von Knoten i, v_i die Besuchszahl von Knoten i, v_r die Besuchszahl des Wurzelknotens, die mit der Gesamtzahl der Simulationen korreliert, und c eine Konstante ist, die die Gewinnmaximierung begünstigt, wenn sie niedrig ist, und die Exploration bevorzugt, wenn sie hoch ist.

- Am Ende, nachdem man die Schritte Selektion, Expansion, Simulation und Backpropagation x mal ausgeführt hat, wählt man den Erfolgversprechendsten Knoten aus.

Evolutionäre Algorithmen

Künstliche Intelligenz

Genetische Algorithmen

- Genetische Algorithmen dienen zur Optimierung von Parametern
- Ähnlich wie bei biologischer Vererbung werden Teile eines (binären) Gens durch Mutation, Rekombination und Selektion weiter vererbt und optimiert
- **Mutation:** In der Gensequenz werden mit gegebener Wahrscheinlichkeit zufällig Bits „gekippt“
- **Rekombination (Crossover):** Aus den Genen zweier effizienter Individuen wird durch Wählen eines oder mehrerer Crossover Punkte ein neues Gen (Individuum) produziert. Dabei wird der erste Teil des einen Gens (bis zum Crossoverpunkt) und der zweite Teil des anderen Gens zu einem neuen Gen zusammengefügt.
- **Selektion:** Bei der Selektion werden die Individuen bewertet und nur die effizientesten werden in die darauffolgende Generation übernommen

Künstliche Intelligenz

Genetische Algorithmen

p sei die Anzahl der Hypothesen(=Individuen) in der Population P
r sei der Anteil, der in jedem Schritt durch Crossover ersetzt wird
m sei Mutationsrate
Initialisiere die Population (erzeuge Zufallshypothesen)
Errechne die Fitness für alle Hypothesen
while maxFitness<fitnessThreshold
   Selektion: wähle (1-r)*p Individuen aus P mit der Wahrscheinlichkeit:

$$Pr(h_i) = \frac{fitness(h_i)}{\sum_{j=1}^{p} fitness(h_j)}$$

und füge sie der neuen Generation P_s hinzu

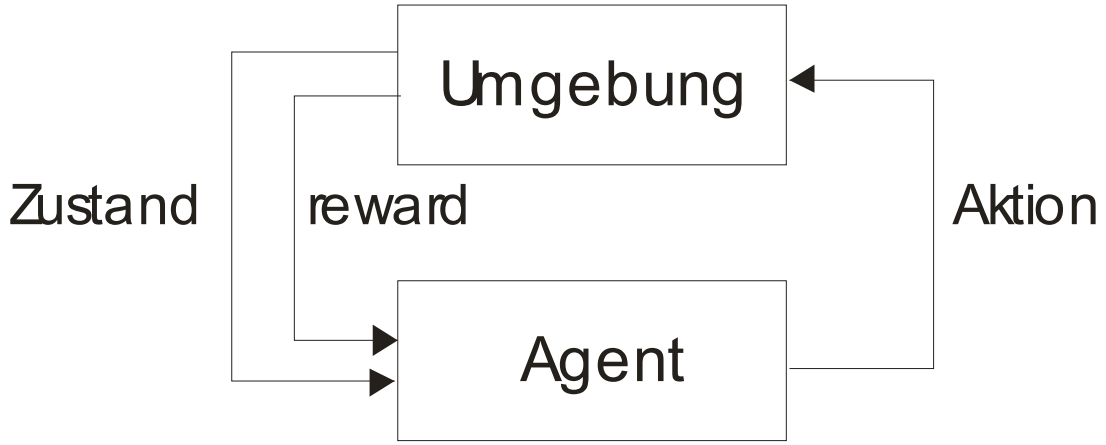
   Crossover: wähle r*p/2 Hypothesenpaare von P mit der oben stehenden Wahrcheinlichkeit $Pr(h_i)$. Erzeuge unter Anwendung des Crossover Operators 2 Nachkommen für jedes Paar und füge sie P_s hinzu.
   Mutation: Wähle m Kandidaten aus Ps mit gleichverteilter Wahrscheinlichkeit und kippe jeweils ein zufällig gewähltes Bit
   update: P ← P_s
   errechne die Fitness für alle Hypothesen
endwhile

Reinforcement Learning

Künstliche Intelligenz

Reinforcement Learning

Reinforcement Learning (bzw. bestärkendes Lernen) ist ein Lernen, bei dem ein Agent lediglich aufgrund von Belohnung (positivem „reward“) und Bestrafung (negativem „reward“) lernt seinen Nutzen zu optimieren



Zustand s kann sein:
- Sensorwerte der Umwelt
- eigene errechnete Position
- Innere Zustände wie Speicherinhalte

Aktion a kann sein:
- Manipulator bewegen
- Abwarten
- Speicher beschreiben

Reward r ist:
- Belohnung, die positiv oder negativ sein kann

Künstliche Intelligenz

Reinforcement Learning
Q-Learning(Watkins, 1989)

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_t + \gamma \, max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

α = [0..1] ist die Lernrate $\gamma = [0..1]$ ist Discountfaktor.

Der Algorithmus:

```
Initialisiere Q(s,a) zufällig mit kleinen Zahlen
repeat (für jede Episode)
     Initialisiere st
     Repeat (für jeden Schritt in der Episode)
          Wähle eine Aktion a aus (z.B. mit Epsilon Greedy)
          Führe Aktion a aus und erhalte r und st+1
          Q(st, at) = Q(st, at) + α[rt + γ max_a Q(st+1, a) − Q(st, at)]
          st = st+1
     until st is terminal
```

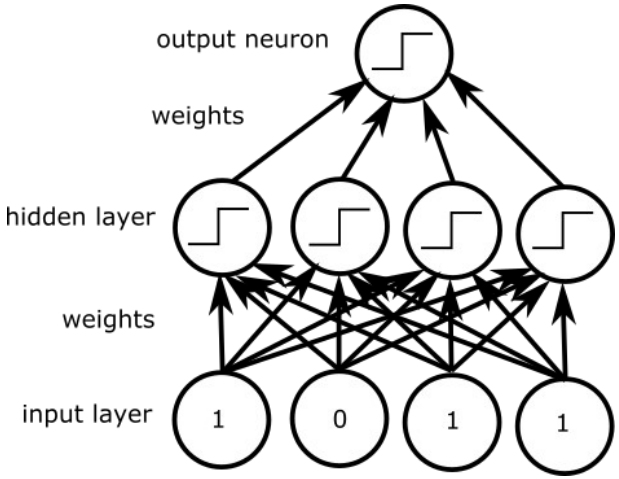
Künstliche Intelligenz

Neuronale Netze

Künstliche Intelligenz

Neuronale Netze

- Ein Neuronales Netz bildet Eingänge auf Ausgänge mittels nichtlinearer Neuronen ab



- Jedes Neuron hat mehrere Eingänge, die mit Gewichtung an andere Neuronen angeschlossen sind.
- Die Gewichte sind dabei die Veränderlichen, die mittels Gradientenabstieg (meist Backprop) angepasst werden.

Künstliche Intelligenz

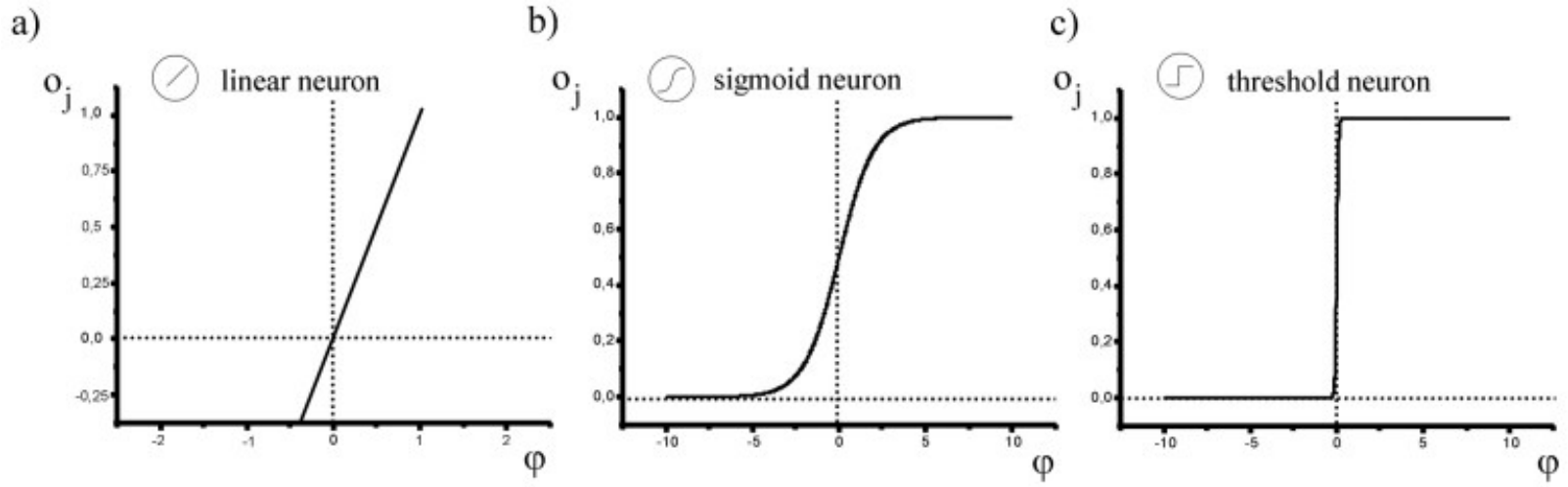
Neuronale Netze

- Die **Aktivität** eines Neurons j ist gleich die Summe der gewichteten Eingänge:

$$\varphi_j = \sum_i W_{ji} o_i$$

- Der **Output** des Neurons ist die über eine Transferfunktion abgebildete Aktivität:

$$o_j = F(\varphi_j)$$



Künstliche Intelligenz

Neuronale Netze

- Trainiert werden die Netze mittels Fehlerpropagation (Backpropagation)
- Man findet eine leicht zu erlernende Python Bibliothek KERAS, die auch auf dem Raspberry Pi läuft
- Trainieren auf dem Pi ist speziell und sollte vermieden werden!