Reinforcement Learning

Grundlagen Grundbegriffe Grundübungen

Alfatraining Kurs 2026-06 Dozent: Karsten Flügge

Einführung

Begrüßung,
Vorstellungsrunde,
Einführung in die Unterrichtsform:
alfaview & digitale Lernumgebung

Allgemeine Einführung in die Thematik

Einführung in Reinforcement-Learning

Einführung Reinforcement Learning

Kleine Historie Machine & Reinforcement Learning

Grundlegende Konzepte

Einordnung

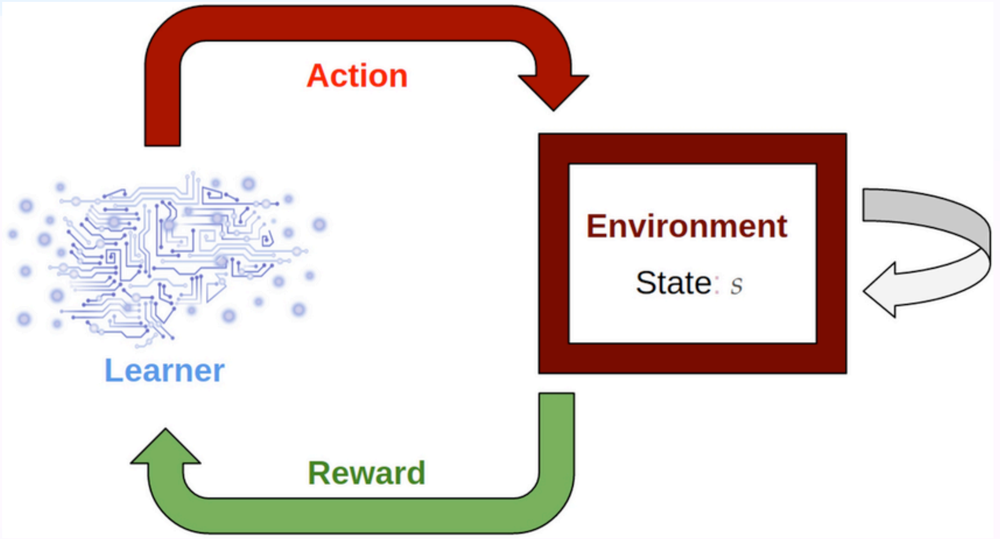
Reinforcement Learning als eine Art von Machine Learning

Unterschiede zu anderen Lernmethoden

SOTA State of the Art

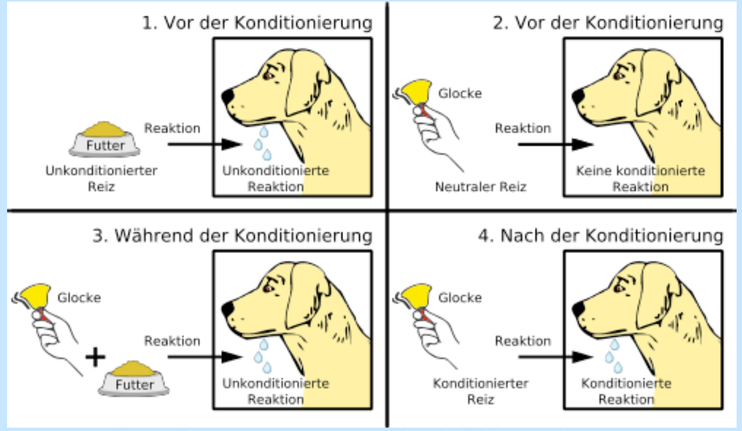
Reinforcement Learning

- **Reinforcement Learning (RL)** ≈ (selbst)Verstärkendes Lernen:
- Teilgebiet des maschinellen Lernens
- ein **Agent** lernt, in einer **Umgebung** durch **Interaktion** optimale **Entscheidungen** zu treffen.
- Ziel: **Maximierung von Belohnungen** durch Auswahl von **Aktionen** basierend auf **Beobachtungen.**



💡 Aktionen ändern oft den (internen) Zustand (State s) der Umgebung!

Kleine Geschichte von RL



Kleine Geschichte des Reinforcement Learning

1. **Anfänge in der Psychologie**: Thorndike formulierte Anfang des 20. Jahrhunderts das „Law of Effect“, das **Verhalten durch Belohnung verstärkt** wird – eine Grundlage des RL.
2. **Begriff „Reinforcement“**: In den 1920ern geprägt, besonders durch **Pawlow** und die klassische **Konditionierung** – später übernommen in die KI-Forschung .
3. **Frühe Ideen in der KI**: Alan **Turing** beschrieb 1948 ein „pleasure-pain system“, Claude Shannon baute 1952 eine Maus („Theseus“), die aus Fehlern lernte .
4. **1954 – Minsky & SNARCs**: Marvin *Minsky* konstruierte „Stochastic Neural-Analog Reinforcement Calculators“ – frühe ***analoge*** RL-Maschinen.
5. **1957 Entwicklung von Dynamic Programming**: Begriff der *Markov* **Decision Processes** (MDPs) und Richard Bellman-Gleichungen
6. **Lernende Automaten (1960er)**: Russische und US-Forscher entwickelten mathematische Modelle zur Lösung von **Bandit**-Problemen
7. **Reinforcement Learning vs. (Un)Supervised Learning**: In den 1980ern erklärten Sutton und Barto RL als **eigenständiges Paradigma**: keinen Lehrer
8. **Lernen mit Zwischenbelohnungen:**

   Arthur Samuel nutzte 1959 in seinem Checkers-Programm bereits „temporal difference“ Konzepte – später durch **Sutton** formalisiert .
9. **Lernen ohne Modell Q-Learning (1989)**: Methode, mit der ein **Agent** lernen kann, wie gut eine bestimmte Handlung in einer bestimmten Situation ist – ganz ohne zu wissen, wie die Umwelt funktioniert.
10. **Boom durch Deep Learning (ab 2013)**: Durch die Kombination mit **Deep Neural Networks** (z. B. Deep Q-Networks von DeepMind) wurde RL massentauglich für komplexe Probleme wie Spiele, Robotik, Steuerung, …

Kleine Geschichte von ML

1960 trucks (nachts) vs cars, ELIZA

1980 AI Winter (ging nicht recht weiter mit Theorie)

2012 AlexNet (8 Schichten tiefes Netz für Bilderkennung)

Was brachte den Durchbruch:
- mehr Rechenpower mit GPUs
- Theoretische / Praktische Lösungen von vorigen Problemen:
  - Exploding / Vanishing Gradient
  - Over / Underfitting
- Deep Learning (viele Schichten)
  - dank dropout, skip/residuals, norms (Regularisierungstechniken)

heute: Netze (fast)unendlich **tief**, GPT: 128

Skalierung: Mehr hilft mehr

Kleine Geschichte Spiel-Agenten

Erfolgsgeschichte

Schach: Der Schach-Computer Deep Blue, der Garry Kasparov besiegt hat, war ein Beispiel für einen deliberativen Spiel-Agenten, der durch umfassende Berechnungen Züge *analysiert*.
Jeopardy: When IBM was great (komplexe Architektur)
DeepMind: 2013 verschiedenste **Atari** Spiele mit dem **selben Netz** geschlagen (Transfer Learning von Meta-Strategien!)
Go: Alpha**Go**, basierend auf Deep Learning und Reinforcement Learning, besiegte **2016** den weltbesten Go-Spieler.
Moderne Alpha**Zero** (ein Netz ohne Daten: Selbstlernend!)
Videospiele: NPCs (Non-Player **Characters**) in Computerspielen sind oft KI-gesteuerte Agenten, die auf bestimmte Umgebungen oder Spieleraktionen reagieren.
Minecraft/Pokemon? Roblox Claude / Grok / ... live on Twitch

SOTA State Of The Art

SOTA AI allgemein:

LLM Large Language Models 2022 Openai
***Multimodal* Text, Ton & Bild & Video**
Bild Erzeugung, Erkennung und Beschreibung
Video *Erzeugung* (https://openai.com/index/sora/ …) SeeDance
3D Modell Erzeugung (Gaussian Splatter, schlechte Topographie?)
Robotic ≈ Physical AI (Cosmos, Physical Intelligence (π), World Labs, Figure AI, Optimus)
Welt Modelle (Integration aller obigen 'Modalitäten')

Kaggle Wettbewerbe
Arena Elo
Kleine Netze auf eigenem Rechner?

SOTA RL spezifisch …

SOTA RL

SOTA RL:

Dreamer https://www.nature.com/articles/s41586-025-08744-2
diamonds in **Minecraft** from scratch without human data or curricula.
3 nets:
world model predicts the outcomes of potential actions, w(a)=s
critic judges the value of each outcome v(a)
actor chooses actions π(s)=a or π(a l s)

AlphaZero Schach + Go spielt gegen sich selbst (Regeln via reward)

DeepSeek RL:
GRPO group relative policy optimization >
ausgefeilteres Actor-Critic-Modell namens Proximal-Policy-Optimierung (**PPO**) statt DQN

Foundation Model? GPT auf eigenem Rechner! LLMs?
Keine öffentlichen Gewichte vorhanden, aber in 10 Tagen selbst trainierbar!

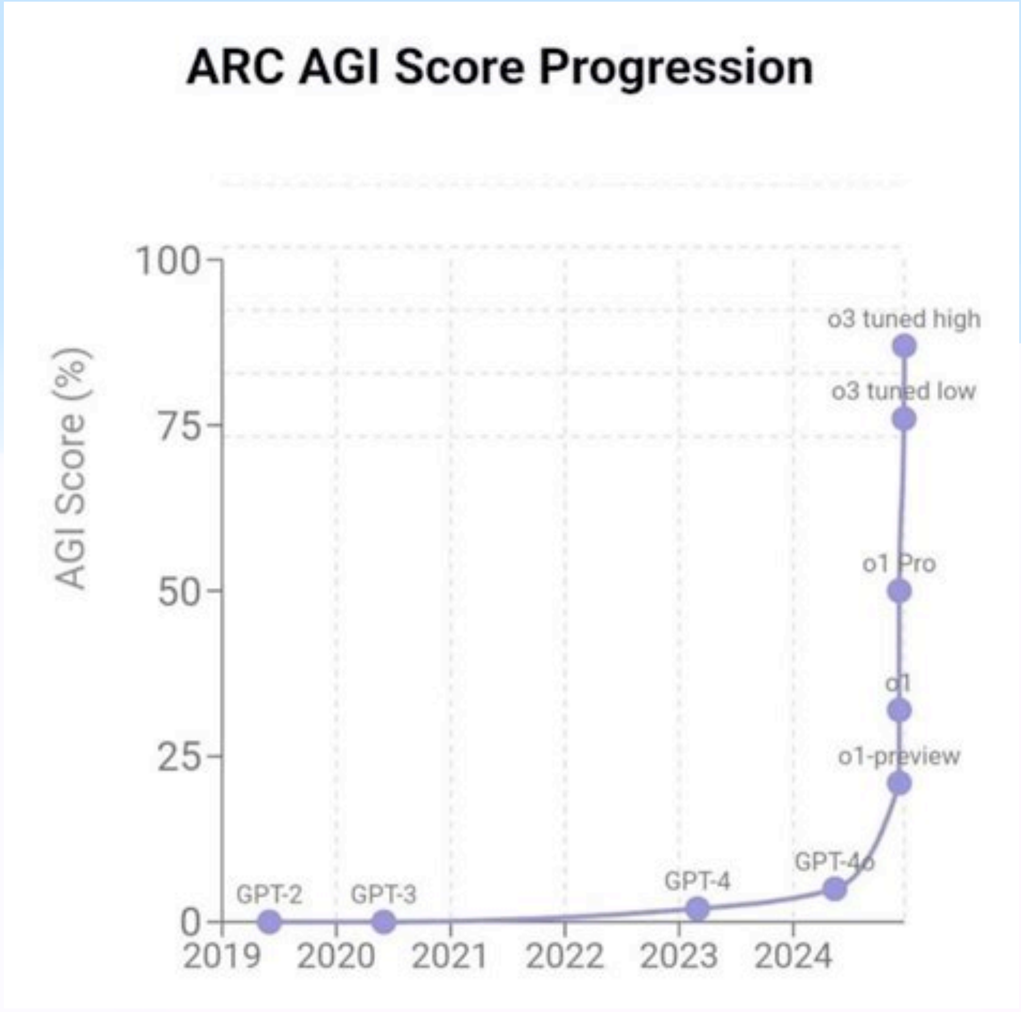
SOTA State Of The Art

Wie gut ist mein Netzwerk?

<u>Baseline</u> (Richtline)
- **random**

Minimales Grundmodell, oder SoTA
- **Linear model**
- **per Bibliothek**

ARC Automated Reasoning Challenge
https://arcprize.org/

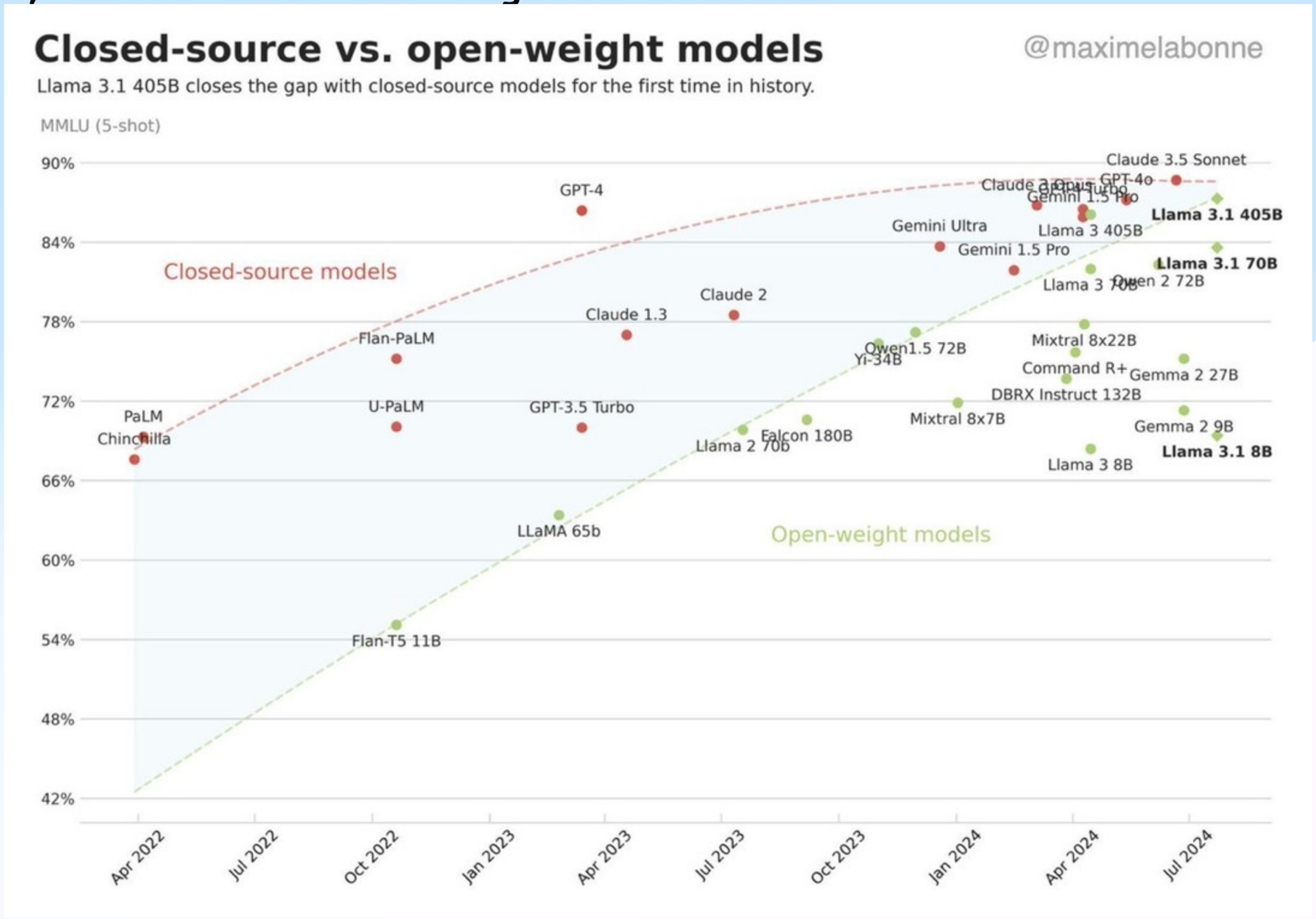


SOTA State Of The Art

Objektive Tests: MMLU (Massive Multitask Language Understanding) **benchmark**

`LLM Large Language Models, Small LLMs auf eigenem Rechner`

`` `ollama run llama3.1` ``



SOTA State Of The Art

🥈 International Mathematical Olympiad 🥈 Gold!

https://deepmind.google/discover/blog/ai-solves-imo-problems-at-silver-medal-level/

IQ Test?

Verbal IQ of the ChatGPT was **155**

https://www.scientificamerican.com/article/i-gave-chatgpt-an-iq-test-heres-what-i-discovered/

"Was schwer für Menschen ist, ist einfach für AI und umgekehrt."

Neue IQ Tests werden gesucht! Moving target.
https://arcprize.org/

EQ Tests? "ChatGPT Outshines Humans in Emotional Tests"

https://neurosciencenews.com/chatgpt-emotion-awareness-23231/

3D taktiles, physikalisches Verständnis

Sutton

Standardwerk des Reinforcement Learning

Sutton (english)

Vokabular Vereinheitlicht

(ZU) Mathematisch, aber erste Kapitel sind verständlich

Sutton

Die Vorstellung, dass wir **durch Interaktion mit unserer Umgebung lernen**, ist wahrscheinlich die erste, die uns in den Sinn kommt, wenn wir über die Natur des Lernens nachdenken. Wenn ein Säugling spielt, mit den Armen wedelt oder sich umsieht, hat er **keinen expliziten Lehrer**, aber eine direkte sensomotorische **Verbindung zur Umwelt**. Durch das Ausüben dieser Verbindung entsteht eine Fülle an Informationen über **Ursache und Wirkung**, über die Konsequenzen von Handlungen und darüber, was zu tun ist, um Ziele zu erreichen. Solche Interaktionen sind zweifellos eine bedeutende Quelle unseres Wissens über die Umwelt und uns selbst – sei es beim Autofahren lernen oder beim Führen eines Gesprächs. Wir nehmen stets wahr, **wie die Umwelt auf unser Verhalten reagiert** und versuchen, durch unser Handeln Einfluss zu nehmen. Lernen durch Interaktion ist eine grundlegende Idee, die nahezu allen Theorien des Lernens und der Intelligenz zugrunde liegt.

Sutton

Wir untersuchen einen **rechnergestützten Ansatz zum Lernen durch Interaktion**. Anstatt direkt darüber zu theoretisieren, wie Menschen oder Tiere lernen, untersuchen wir **idealisierte Lernsituationen** und bewerten die Effektivität verschiedener Lernmethoden. Wir nehmen dabei die Perspektive eines KI-Forschers oder Ingenieurs ein. Wir entwerfen Maschinen, die effektiv Lernprobleme von wissenschaftlichem oder wirtschaftlichem Interesse lösen, und **bewerten die Entwürfe durch mathematische Analysen oder rechnergestützte Experimente.** Der untersuchte Ansatz, das Verstärkungslernen (Reinforcement Learning), ist wesentlich stärker auf **zielgerichtet**es Lernen aus Interaktionen fokussiert als andere maschinelle Lernansätze.

Themen des Kurses

REINFORCEMENT LEARNING

Einführung in Reinforcement Learning (ca. 1 Tag)
Definition und grundlegende Konzepte
Unterschiede zu anderen Lernmethoden
Anwendungsbereiche und Beispiele

Markov Decision Processes (MDPs) (ca. 2 Tage)
Definition und Eigenschaften von MDPs
Value-Funktionen und Policy
Bellman-Gleichungen
Dynamic Programming Ansatz

Q-Learning (ca. 2 Tage)
Definition und Algorithmus
Exploration vs. Exploitation
Konvergenz- und Optimierungseigenschaften
Anwendungen in Spielen, Robotik und anderen Bereichen

Deep Reinforcement Learning (ca. 3 Tage)
Deep Q-Learning
Deep Deterministic Policy Gradients (DDPG)
Actor-Critic-Methoden
Policy Gradient-Methoden

Fortgeschrittene Themen (ca. 4 Tage)
Model-Based Reinforcement Learning
Multi-Agent Reinforcement Learning
Inverse Reinforcement Learning
Meta Reinforcement Learning

Praktische Anwendungen (ca. 3 Tage)
Implementierung von Reinforcement Learning Algorithmen
Anwendung auf ausgewählte Probleme und Fallstudien
Evaluation und Tuning der Algorithmen

Zusammenfassung und Ausblick (ca. 2 Tage)
Zusammenfassung der wichtigsten Konzepte und Ergebnisse
Herausforderungen und zukünftige Entwicklungen in Reinforcement Learning

Projektarbeit (ca. 3-5 Tage)
Zur Vertiefung der gelernten Inhalte
Präsentation der Projektergebnisse

Themen des Kurses

Woche 1

Montag
Einführung in Reinforcement Learning
Grundlegende Konzepte
Unterschiede zu anderen Lernmethoden
Anwendungsbereiche und Beispiele

Dienstag
Markov Decision Processes (MDPs)
Definition und Eigenschaften von MDPs
Value-Funktionen
Policy

Mittwoch
Bellman-Gleichungen
Kapitel 7.3 im Buch!
Dynamic Programming Ansatz
Praktische Übungen

Donnerstag
Q-Learning: Definition, Algorithmus
Exploration vs. Exploitation
Kapitel 3.2 im Buch

Freitag
Konvergenz- und Optimierungseigenschaften
Anwendungen in Spielen, Robotik etc.
Vertiefung durch praktische Übungen

Themen des Kurses

Woche 2

Montag
Deep Reinforcement Learning
Deep Q-Learning (TD)

Dienstag
Policy Gradient-Methoden (SGD)
REINFORCE

Mittwoch
Deep Deterministic Policy Gradients (DDPG)
Actor-Critic-Methoden
11.1 Actor–Critic Methods . . . . . 257 in Sutton

Donnerstag
Model-Based Reinforcement Learning

Freitag
Multi-Agent Reinforcement Learning
Praktische Anwendungen und Übungen

Themen des Kurses

Woche 3

Montag
Inverse Reinforcement Learning

Dienstag
Meta Reinforcement Learning

Mittwoch
Implementierung von RL-Algorithmen

Donnerstag
Anwendung auf ausgewählte Probleme und Fallstudien

Freitag
Evaluation und Tuning aktueller Algorithmen

Themen des Kurses

Woche 4

Montag
Zusammenfassung der wichtigsten Konzepte und Ergebnisse

Dienstag
Herausforderungen und zukünftige Entwicklungen

Mittwoch
Start der Projektarbeit, Themenvergabe

Donnerstag
Erstellung der Projektarbeit

Freitag
Abschluss der Projektarbeit, Präsentation und Bewertung

Sutton

1.1 **Verstärkungslernen**

Verstärkungslernen bedeutet zu lernen, was zu tun ist – also wie **Situationen in Handlungen** überführt werden –, um ein numerisches Belohnungssignal zu maximieren. Der Lernende wird *nicht* darüber informiert, welche Handlungen er ausführen soll, sondern muss durch Ausprobieren **selbst herausfinden**, welche Handlungen am meisten **Belohnung** bringen. In den interessantesten und herausforderndsten Fällen wirken sich Handlungen nicht nur auf die unmittelbare Belohnung aus, sondern auch auf die nächste Situation und darüber hinaus auf alle **zukünftigen Belohnungen**. Diese beiden Merkmale – **Lernen durch Versuch und Irrtum und verzögerte Belohnung – sind die wichtigsten Unterscheidungsmerkmale des Verstärkungslernens.**

Verstärkungslernen ist wie viele andere Themen, die auf „-en/-ing“ enden (z.B. maschinelles Lernen oder Bergsteigen), gleichzeitig ein **Problem**, eine Klasse von **Lösungsverfahren** und ein **Forschungsgebiet**. Es ist praktisch, für alle drei Bereiche denselben Namen zu verwenden, aber gleichzeitig ist es wichtig, sie gedanklich zu unterscheiden. Besonders wichtig im Verstärkungslernen ist die Unterscheidung zwischen Problemen und Lösungsverfahren – das Versäumnis, diese Unterscheidung zu machen, ist Quelle vieler Missverständnisse.

(selbst) Verstärkendes Lernen

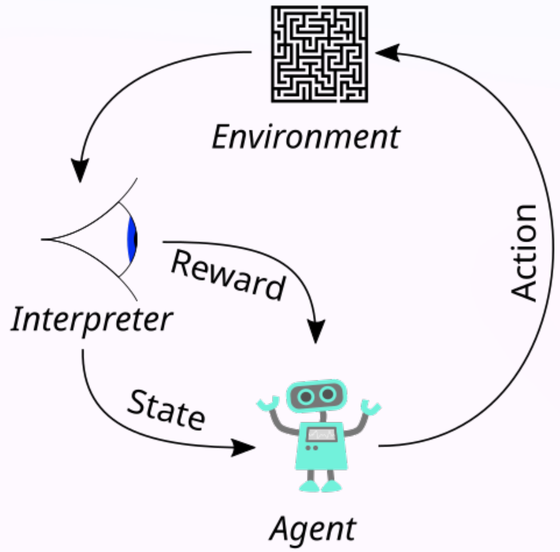
Übersicht und Definition

- **Reinforcement Learning (RL)** ≈ (selbst)Verstärkendes Lernen "Bestärkendes Lernen"
- Teilgebiet des maschinellen Lernens, bei dem ein **Agent** lernt, in einer **Umgebung** durch **Interaktion** optimale **Entscheidungen** zu treffen.
- Ziel: **Maximierung einer Belohnung** durch Auswahl von **Aktionen** basierend auf **Beobachtungen**.
- Grundlage: **Trial-and-Error**-Prinzip + Rückmeldung durch **Reward**-(Belohnung)-Signale.

- Allgemeines Framework welches sich auf verschiedenste Situationen anwenden lassen soll

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

https://de.wikipedia.org/wiki/Bestärkendes_Lernen



Anwendungsbereiche und Beispiele

Anwendungsbereiche und Beispiele

Spiele: z. B. Alpha**Go**, Schach, Atari-Spiele, ...

Robotik: Laufverhalten, Greifen, Navigation, ...

Industrie: Prozessoptimierung, Energieeinsparung, ...

Healthcare: Therapieplanung, ...

Autonomes Fahren: Entscheidungsfindung im Verkehr, ...

- **Steuerung**stechnik
- **Simulation** von Menschenmengen
- **Wirtschaft,** z.B. für autonome Handelssysteme.

Grundlegende Konzepte

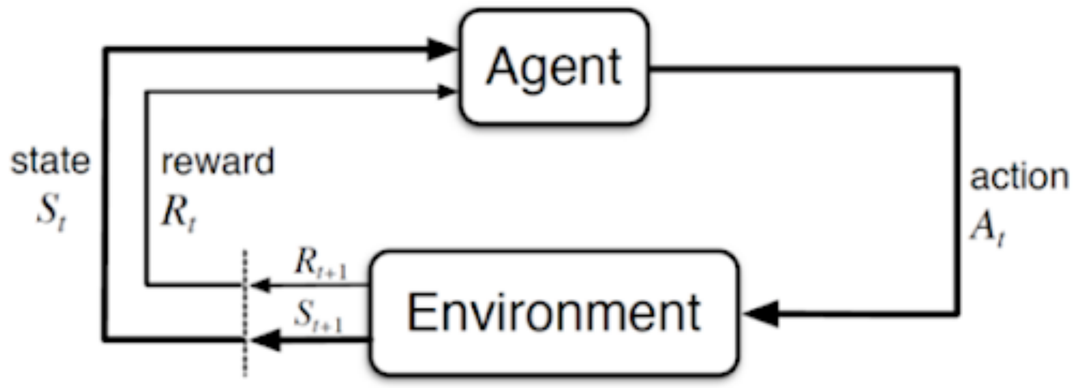
Historisch wurde RL auf sehr viele Gebiete angewendet deswegen sind die Definitionen etwas allgemein es hilft aber immer sich ein konkretes Beispiel vorzustellen zum Beispiel **Schach**.

- RL: ein **Agent** lernt, in einer **Umgebung** durch Inter**aktion** optimale **Belohnung** zu erhalten.

Grundlegende Konzepte

- **Agent**: trifft Entscheidungen
- **Umgebung (environment)**: liefert Rückmeldungen (Zustände, Belohnungen)
- **Zustand (state) s**: Beschreibung der aktuellen **S**ituation / Position
- **Aktion (action) a**: Handlung des Agenten s->**s'** Züge
- **Belohnung (reward) r**: Feedback der Umgebung (real, 'abstrakt' oder von uns selbst, Hauptsache quantisierbar, also als Zahl)
- **Policy (π)**: Strategie, die angibt, welche Aktion in welchem Zustand gewählt wird
  - deterministisch π(**s**)=**a** (immer die selbe Aktion pro Situation)
  - oder als stochastisch **π(a | s)**

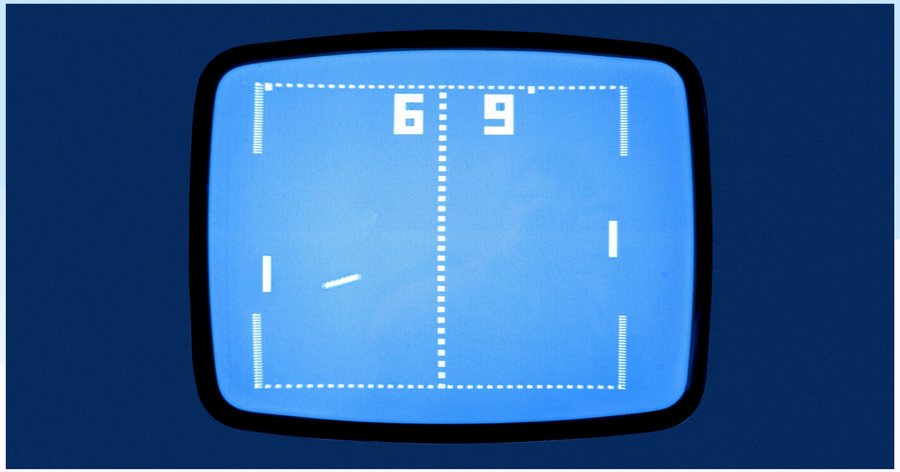
π weil p schon vergeben ist für probability



Grundlegende Konzepte am Beispiel Pong

Grundlegende Konzepte am Beispiel Pong

- **Umgebung (Environment)**: liefert Rückmeldungen (Zustände, Belohnungen) **Pong**
- **Zustand (State)** s: Beschreibung der aktuellen Situation **Ball & Schläger Positionen**
- **Agent:** **(Mensch / Computer)**: trifft Entscheidungen:
- **Aktion a**: Handlung des Agenten: **Schläger hoch a=1 / runter a=-1**
- **Belohnung (Reward) r**: Feedback der Umgebung
  - **Sieg / Verlust? 1 / -1 oft genug!!!**
  - **Verzögerte Belohnung:** erst nach vielen 'Zügen' 100 mal hoch/runter



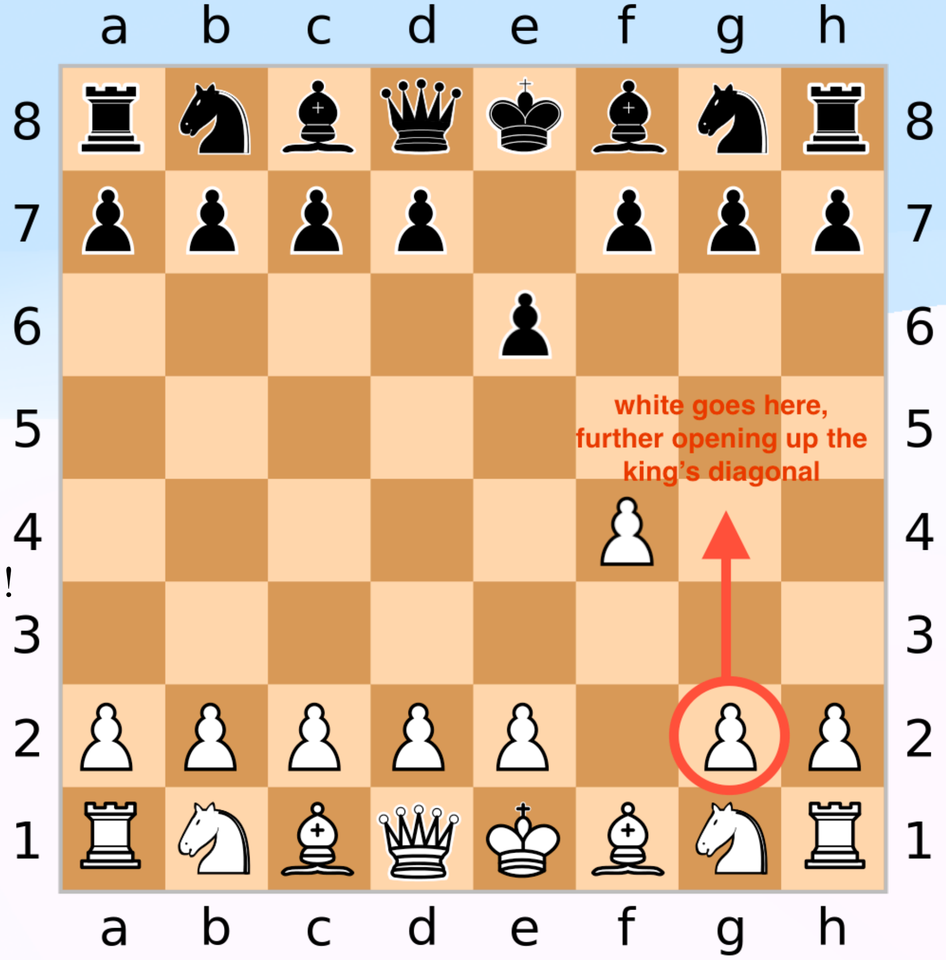
- **Policy (π)**: Strategie, die angibt, welche Aktion in welchem Zustand gewählt wird
  - gut: einfach Ball folgen (hard coded: langweilig) **(kann / muss vom Netz gelernt werden!)**
  - besser: position des Balles **antizipieren** (hard coded: langweilig) **(kann vom Netz gelernt werden!)**

Grundlegende Konzepte am Beispiel Schach

Grundlegende Konzepte am Beispiel Schach

- **Umgebung (Environment)**: liefert Rückmeldungen (Zustände, Belohnungen) **Schach Programm**
- **Zustand (State) s**: Beschreibung der aktuellen Situation **aktuelles Schachfeld**
- **Agent: (Mensch / Computer)**: trifft Entscheidungen:
- **Aktion (Action) a**: Handlung des Agenten: **nächster Zug**
- **Belohnung (Reward) r**: Feedback der Umgebung
  - **Sieg: +1 -1** (sparse reward ist *schwer* zu trainieren)

- **Policy (π)**: Strategie, die angibt, welche Aktion in welchem Zustand gewählt wird
  - bei Schach **sehr komplex**. *Versuche Dame zu klauen*
  - Versuche den König zu jagen
  - all die Komplexität wollen wir in einem **neuronalen Netzwerk** 'verstecken'!

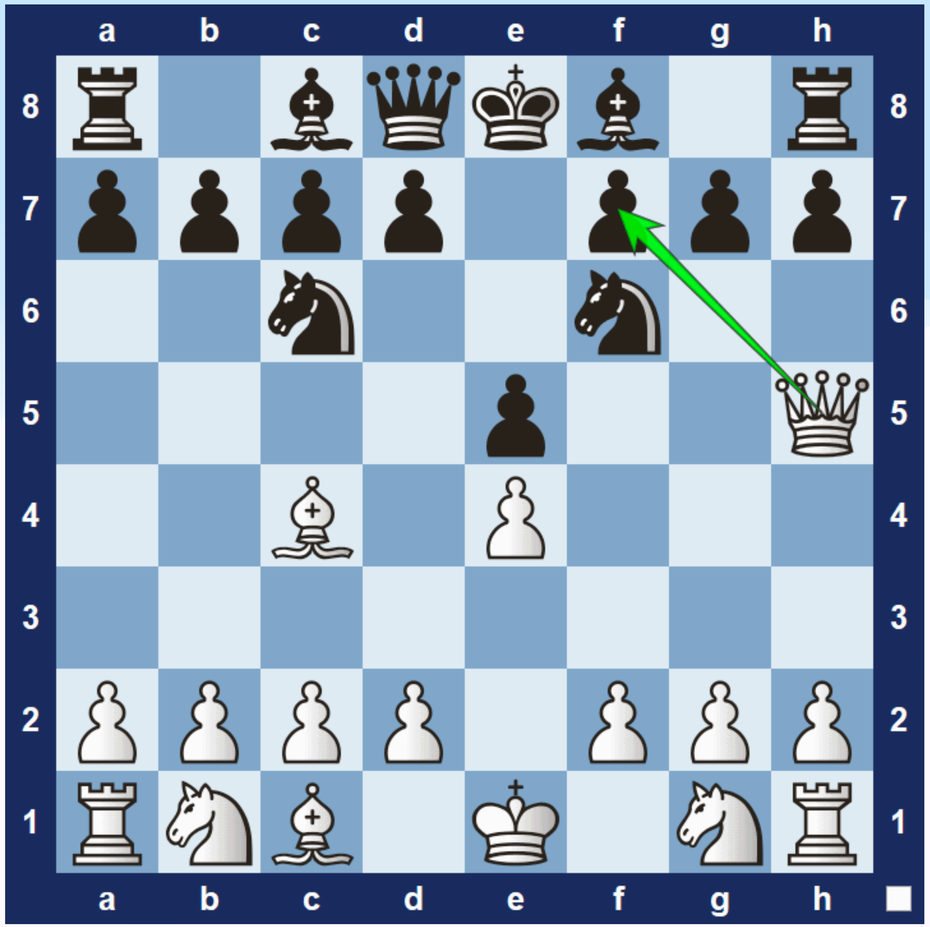


Grundlegende Konzepte am Beispiel Schach

Grundlegende Konzepte am Beispiel Schach

- **Umgebung (Environment)**: liefert Rückmeldungen (Zustände, Belohnungen) **Schach Programm**
- **Zustand (State) s**: Beschreibung der aktuellen Situation **aktuelles Schachfeld** (meist nicht: Gegner ELO, Stimmung …)
- **Agent: (Mensch / Computer)**: trifft Entscheidungen:
- **Aktion (Action) a**: Handlung des Agenten: **nächster Zug**
- **Belohnung (Reward) r**: Feedback der Umgebung
  - legaler Zug? -10000
  - Intrinsische Belohnung: Figur geschlagen? **Wert** einer Figur? Bauer **1 Dame 10** … ?
  - Matt? **+10000**
  - Extrinsische Belohnung: Sieg / Verlust? **1 / -1 oft genug!!!**
  - Eigene Bewertung: "stabile Verteidigung" … …
  - Fremd Bewertung: **Lehrer, "Schul Eröffnungen", Erfahrungsnetz …**
  - **…**
  - **Wir wollen Belohnung der Umgebung verstehen (Regeln, Taktiken)**
  - **WIR HABEN KREATIVE WAHL fürs mixen eigener und fremder Belohnungen!**

- **Policy (π)**: Strategie, die angibt, welche Aktion in welchem Zustand gewählt wird
  - bei Schach **sehr komplex**. Versuche Dame zu klauen
  - all die Komplexität wollen wir in einem neuronalen Netzwerk 'verstecken'!



Grundlegende Konzepte am eigenen Beispiel

Grundlegende Konzepte am eigenen Beispiel

- **Umgebung (Environment)**:
- **Zustand (State)** s:
- **Agent:** **(Mensch / Computer)**: **Tier?**
- **Aktion a**:
- **Belohnung (Reward) r**:
- **Policy (π)**:

Grundlegende Konzepte am eigenen Beispiel

Grundlegende Konzepte am eigenen Beispiel

- **Umgebung (Environment)**: Cyber Abwehr (Internet) Honeypots, IDS Intrusion Detection System
- **Zustand (State)** s: Clean, Exploited, Pakete, Response 404, 200 ...
- **Agent: (Mensch / Computer)**: **Tier? Angreifer / Verteidiger**
- **Aktion a**: Pakete Senden (Vektor von bytes)
- **Belohnung (Reward) r**: +1000 Pentagon gehackt, +10 für '500'
- **Policy (π)**: Unerkannt bleiben, ...

Grundlegende Konzepte am eigenen Beispiel

Grundlegende Konzepte am eigenen Beispiel

- **Umgebung (Environment)**: Softwareentwicklung (Runtime)
- **Zustand (State)** s: Fehler?
- **Agent:** **(Mensch / Computer)**: **Tier? C**
- **Aktion a**: Programm als Zeichenfolge (Vektor von Ascii)
- **Belohnung (Reward) r**: +1 für passed test
- **Policy (π)**: Korrekte Programme Schreiben

"Reward hacking, cheating" Agent heimst Belohnungen ein, die nicht das 'Ziel' waren.

Hier Agent spuckt: print() aus

Grundlegende Konzepte am eigenen Beispiel

Grundlegende Konzepte am eigenen Beispiel

- **Umgebung (Environment)**: Mario Cart
- **Zustand (State)** s: Position aller Autos, Sterne zum Sammeln, Straße?
- **Agent:** **(Mensch / Computer)**: **Tier?**
- **Aktion a**: rechts/links/speed/drift/token (Gegner stören)
- **Belohnung (Reward) r**: Ranking im Ziel
- **Policy (π)**: antizipiere die Straße, Rammen, Stern Sammeln

Grundlegende Konzepte am eigenen Beispiel

Grundlegende Konzepte am eigenen Beispiel

- **Umgebung (Environment)**: Aufklärungsdrohne Luftraum
- **Zustand (State)** s: Wetterdaten, GPS Position, Batterie, Radar
- **Agent: (Mensch / Computer)**: **Tier?** Pilot/Software
- **Aktion a**: Höhe ändern, Winkel ändern
- **Belohnung (Reward) r**: +1 für Neue Positionen, +100 für Ziel erreicht, -1 pro Batterieverbrauch
- **Policy (π)**: Beste Route, Unerkanntes Aufklären

Grundlegende Konzepte am eigenen Beispiel

Grundlegende Konzepte am eigenen Beispiel

- **Umgebung (Environment)**: Aktien Stock Trading
- **Zustand (State)** s: Preis, Volume, MI50
- **Agent:** **(Mensch / Computer)**: **Tier? Algo**
- **Aktion a**: halten, kaufen, verkaufen 0,1,2
- **Belohnung (Reward) r**: Echter Gewinn
- **Policy (π)**: beim Steigen kaufen, sonst halten, bei runter verkaufen

Grundlegende Konzepte am eigenen Beispiel

Grundlegende Konzepte am eigenen Beispiel

- **Umgebung (Environment)**: Auto Crash Test
- **Zustand (State)** s: Spannungen im Gurt,
- **Agent:** **(Mensch / Computer)**: **Tier?** **Spannermotor** Dummy;)
- **Aktion a**: Spannen (1,-1) (momentan mechanisch), ideal: variabel je nach Kraft!
- **Belohnung (Reward) r**: +1 bei unter kritischem Niveau
- **Policy (π)**: Kurtkraft antizipieren und konstant halten, blaue Flecken vermeiden

Grundlegende Konzepte am eigenen Beispiel

Grundlegende Konzepte am eigenen Beispiel

- **Umgebung (Environment)**: 2d Irrgaten
- **Zustand (State)** s: Position x,y
- **Agent: (Mensch / Computer)**: **Tier?**
- **Aktion a**: rauf, runter, rechts, links (1,2,3,4) oder (+1,0) (0,-1) ...
- **Belohnung (Reward) r**: 10000/t
- **Policy (π)**: Neue Wege finden

Grundlegende Konzepte am eigenen Beispiel

Grundlegende Konzepte am eigenen Beispiel

- **Umgebung (Environment)**: Fahren auf Straße
- **Zustand (State)** s: Ampel Rot, Frontkamera, Speed, Lidar
- **Agent:** **(Mensch / Computer)**: **Tier?** (Software im) Auto
- **Aktion a**: Lenkrad Grad, Gaspedal
- **Belohnung (Reward) r**: **+1** wenn keine Kollision -1000 Katze tot
- **Policy (π)**: Unfallfrei fahren, bester Weg

Grundlegende Konzepte am eigenen Beispiel

Grundlegende Konzepte am eigenen Beispiel

- **Umgebung (Environment)**: Kochen
- **Zustand (State)** s: Wassertemperatur, Konsistenz?
- **Agent:** **(Mensch / Computer)**: **Tier?**
- **Aktion a**: Nudeln rein 0,1 Platte aus?
- **Belohnung (Reward) r**: 47.11 für gare Nudeln (vom Mensch bewertet)
- **Policy (π)**: Wann Nudeln rein? Wenn Wasser kocht.

```
if temp>=100:
  action = rein # 1
```

Grundlegende Konzepte am eigenen Beispiel

Grundlegende Konzepte am eigenen Beispiel

- **Umgebung (Environment)**: Gerätesteuerung Solar **Energiesparen + Mensch**
- **Zustand (State)** s: Energieverbrauch: Watt
- **Agent:** **(Mensch / Computer)**:
- **Aktion a**: Heizung / **Geräte an / ausschalten (schlaue Waschmaschine)**
- **Belohnung (Reward) r**: -1 pro Watt/h
- **Policy (π)**: A**uto Programm (waschen bis sauber!)**

Grundlegende Konzepte am eigenen Beispiel

Grundlegende Konzepte am eigenen Beispiel

- **Umgebung (Environment)**: Film/Musikvorschlag (Spotify/Netflix/…)
- **Zustand (State)** s: Aktueller Song/Film
- **Agent: (Mensch / Computer)**: Jukebox
- **Aktion a**: Ausspielen von Vorschlägen (ids) Kategorien (tag-ids)
- **Belohnung (Reward) r**: Thumbs up/down **±100**, Kuckdauer + 1/min, Komplett=+100
- **Policy (π)**: Kucke nach ähnlichen Nutzern …

Grundlegende Konzepte am eigenen Beispiel

Grundlegende Konzepte am eigenen Beispiel

- **Umgebung (Environment)**: Luftabwehr gegen Drohnen (Ukraine) Sky sentinel
- **Zustand (State)** s: **Bild** / Lidar / **Radar**? Schiessend?
- **Agent:** **(Mensch / Computer)**: **Geschützturm**
- **Aktion a**: Schiessen (an/aus) Drehen (Grad) Höhe (Grad) Alarm?
- **Belohnung (Reward) r**: Treffer=1000 Selbergetroffen=-1000 Jeder Schuss=-1
- **Policy (π)**: Schiessen **genau dann,** wenn Objekt anvisiert ist. Hoch wenn Objekt oben ist.

Grundlegende Konzepte am eigenen Beispiel

Grundlegende Konzepte am eigenen Beispiel

- **Umgebung (Environment)**: Produktionskette Membranfilter(Poren) Produktionlinie
- **Zustand (State)** s: Processparameter Größe von Poren, Temperatur, Geschwindigkeit von Maschinenteilen, Druck, Prognose von anderm Model!
- **Agent:** **(Mensch / Computer)**: Computer **+** Engineer
- **Aktion a**: Temperatur erhöhen / senken, Geschwindigkeit erhöhen / senken, Warnung
- **Belohnung (Reward) r**: Ziel: Qualitätsmerkmale Permeabilität +x%
- Höhe Qualitötsmerkmale wie höhe Permeabilität (positive), niedrige oder blockierte Fluss (negativ)
- **Policy (π)**: bie größen Pören: Temperatur erhöhen Bei kleinen Pören: Polymerviskosität anpassen

Grundlegende Konzepte am eigenen Beispiel

Grundlegende Konzepte am eigenen Beispiel

- **Umgebung (Environment)**: Der Rasenmähroboter™
- **Zustand (State)** s: Fahren**d** Halten**d** (Rasenhöhe?) Entfernungsmessung zu Hindernissen (Kamerabild 2025!)
- **Agent: (Mensch / Computer)**: Robi
- **Aktion a**: Fahren halten drehen (winkel, vor/zurück ±1) **piepen bei Festfahren / bei Anheben**
- **Belohnung (Reward) r**: Bei Sensormessung: Niedrieger Rasen = +10 Optimale Route +1 Niedrige Zeit +1 via Lange Route **-1**/s
- -1 bei Treffen auf Hinderniss / Festfahren
- **Policy (π)**: Kurze Route, nicht immer die selbe Strecke fahren. Bei Hindernis: a = (0 , -1) zurückfahren dann drehen (30, 10) oder piepsen

Gibts schon!

Grundlegende Konzepte am eigenen Beispiel

Grundlegende Konzepte am eigenen Beispiel

- **Umgebung (Environment)**: Werbung Kunden Advertiser
- **Zustand (State)** s: Zeit der eingeblendeten Werbungen? <<
- **Agent:** **(Mensch / Computer)**: **Multi-Agent**
- **Aktion a**: bidding Biete für Einblendung $xyz
- **Belohnung (Reward) r**: Kunde kauft +$1000 -$0.20
- **Policy (π)**: Bewerbe kaufkräftige Kunden

Grundlegende Konzepte am eigenen Beispiel

Grundlegende Konzepte am eigenen Beispiel

- **Umgebung (Environment)**: *Deep Seek* **R1** bei LLM
- Ursprünglich ChatGPT: Internet durchlesen (Finetuning RLHF)

  Reinforcement Learning from Human Feedback. "kein echtes RL" : Vorgegebener Datensatz

  Echtes Reinforcement Learning seit 2025 : Agent interagiert autonom mit Umgebung

  (Mathematische) Aufgaben + **Lösungen**
- **Zustand (State)** s: gibts nicht?
- **Agent: (Mensch / Computer)**: Sprachmodel
- **Aktion a**: Text erzeugen (Tokens auswerfen) <think>**…**</think><result>…</result>
- **Belohnung (Reward) r**: Lösungen passt, Text länge, im Zweiten Schritt: <think> Ausgabe auch menschlich?
- **Policy (π)**: Gewichte im LLM => Softmax für Token

Grundlegende Konzepte am eigenen Beispiel

Grundlegende Konzepte am eigenen Beispiel

- **Umgebung (Environment)**: Chat System / Genereller Content
- **Zustand (State)** s: Aktueller Chatverlauf
- **Agent:** **(Mensch / Computer)**: Mensch greift ein bei zu viel Daumen spam
- **Aktion a**: Daum Hoch/Runter
- **Belohnung (Reward) r**: +1 / -1
- **Policy (π)**: 'Besten' Content liefern, Formel beliebig anpassbar

Grundlegende Konzepte am eigenen Beispiel

Grundlegende Konzepte am eigenen Beispiel

- **- Umgebung (Environment):** Ein Raum mit **Schmutzstellen** und Möbeln
- **- Zustand (State) s:** Position des Roboters und Position des Schmutzes (**Kamerabild, Sensorinformationen**)
- **- Agent (Mensch / Computer):** Computer (**Reinigungsroboter** mit Steuerungsprogramm)
- **- Aktion (a):**
- -- Vorwärts fahren
- -- Rückwärts fahren
- -- Nach links fahren
- -- Nach rechts fahren
- **-- Reinigen (vorprogrammiertes Programm)**
- Version 2030: Batterie aufladen
- **- Belohnung (Reward) r:**
- **-- +10 für das Reinigen eines Schmutzflecks**
- **--** -5 für das Anstoßen an Möbel oder Wände
- **- Policy (π): Wie kann der Roboter den Raum effizient reinigen, ohne irgendwo anzustoßen?**

Grundlegende Konzepte am eigenen Beispiel

Grundlegende Konzepte am eigenen Beispiel

Umgebung (Environment): Robotik, **Roboterhund** auf Teststrecke/Terrain, Sensoren, Kameras

Zustand (**State**) **s**: Stehend?, Umgefallen, Kaputt? Sensoren: Gyroskop, Winkelsensoren, **Druck**, Beschleunigung, Geschwindigkeit, Kamerabild

Position im Terrain, GPS, Akustik (Pfeifsensor zum Kommen)

Agent: (Mensch / Computer): Roboter, Computer, **Mensch greift ein**

Aktion **a**: einen Schritt machen, stoppen bei Hindernis (high level), drehen, …

Belohnung (Reward) **r:** +10 Belohnen bei erfolgreichem Schritt, +100 belohnen bei Stoppen bei Hindernis , -1000 bei Umfallen

Policy (π): Wenn Hindernis kommt, stoppe bei einem Hindernis, …

Grundlegende Konzepte am eigenen Beispiel

Grundlegende Konzepte am eigenen Beispiel

- **Umgebung (Environment)**: Straße, selbstfahrende Autos: Abstand+Spur halten
- **Zustand (State)** s: Abstandssensor, Kamerabild IR?, magnetische? Führungs- Signale von der Straße/Autos?
- **Agent:** **(Mensch / Computer)**: Computer
- **Aktion a**: links/rechts lenken + bremsen/beschleunigen + Warnsignal
- **Belohnung (Reward) r**: negativ fürs Abweichen / zu nahe kommen
- **Policy (π)**: Mittelwert / Abstände maximieren zum Fahrstreifen,

Grundlegende Konzepte am eigenen Beispiel

Grundlegende Konzepte am eigenen Beispiel

- **Setting / Ziel: Verkehrsampel, guter Verkehrsfluss**
  - **Umgebung (Environment):**Ampel, Kamera, Bodensensoren, Timer, Fußgängerknopf
  - **Zustand (State) s: Rot, Gelb, Grün,** länge der Warteschlange, Zeit seit dem letzten grün/rot, länge der einzelnen Phasen, Tageszeit
    - Bilddaten, Bodensensoren: **an/aus**
  - **Agent: (Mensch / Computer):** Computer, Mensch (in Verkehrssteuerzentrale)
  - **Aktion a:** wechsel von **Rot auf Grün** oder umgekert (mit Gelb)
  - **Belohnung (Reward) r:**
    pro Auto / Fußgänger +1
    pro Wartezeit: -1 (pro Minute?) ***Priorisierung***
    kein Stau +1, lange Wartezeiten -1, -1000 bei zwei mal grün/rot (besser hardcoded;)
  - **Policy (π):** wenn lange Schlange länger grün, **erst umschalten wenn Autos warten**/kommen.
    automatisch gelernt

Grundlegende Konzepte am eigenen Beispiel

Grundlegende Konzepte am eigenen Beispiel

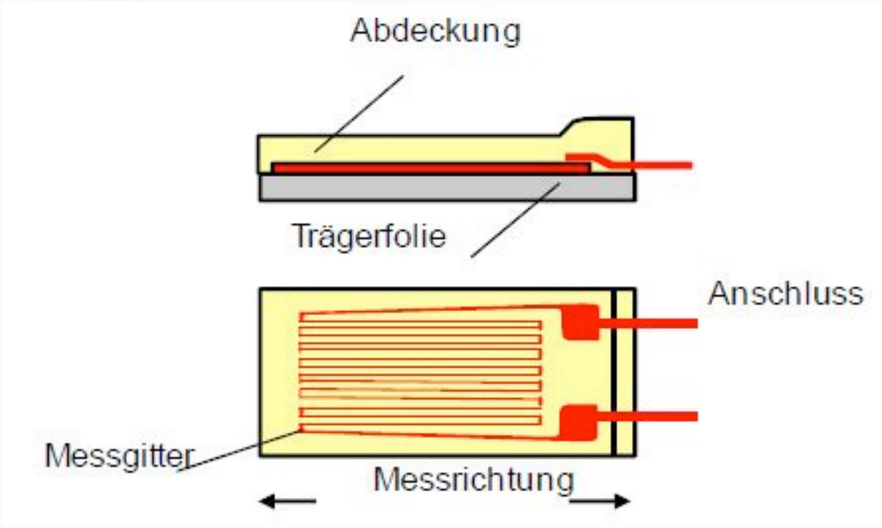
- **Umgebung (Environment)**: **Hemdenbügelroboter**
- **Zustand (State)** s: Kamerabilder: Falten, Formen
- **Agent:** **(Mensch / Computer)**: Computer mit **Armen / Schienen**
- **Aktion a**: Bügeleisen über Schiene / Arme bewegen / **drehen**
- **Belohnung (Reward) r**: abnehmende / minimale Falten
- **Policy (π)**: komplex, möglichst nur einmal abfahren, **nicht verweilen** => brandgefahr!

Grundlegende Konzepte am eigenen Beispiel

Aufgabe:

Grundlegende Konzepte am eigenen Beispiel

- **Umgebung (Environment)**: Spritzguß**gießerei** Ziel: möglichst hoher output
- **Zustand (State)** s: Temperatur, Abstände (Blatt Papier!), Schmierung (indirekt), DMS Messstreifen Verformung
- **Agent:** **(Mensch / Computer)**: Mensch + Maschine
- **Aktion a**: Regelung der Motoren
- **Belohnung (Reward) r**: Abschluss Schliess**zeit**, **Grenze**inhaltung (Hohlräume, Klunker), Kraftbalance…
- **Policy (π)**: PI-Regler …



Grundlegende Konzepte am eigenen Beispiel

- Grundlegende Konzept – **Aktienkurs**
- Umgebung (Environment): Börse Frankfurt + WIR (Mediator / Proxy)
- Zustand (State) s: Aktien**kurs/Wert** einer best. Aktie
- Agent: (Mensch / Computer): Programm ruft Aktienkurs ab **und entscheidet:**
- Aktion a: **Kauf/Verkauf**/Halten/Warnung
- Belohnung (Reward) **r**: Aktienverkauf erzeugt **Gewinn** oder Mensch gibt Belohnung ein = positiv,
- Aktienverkauf erzeugt Verlust oder Mensch greift ein = negativ
- Ziel: 100% positive autonome Steuerung **Gewinne**
- Policy (**π**): ab xy % Verkaufen, ab xy % Kaufen, ab % Minus/Plus erfolgt eine Verlust-/Verkaufswarnung
  - kann kompliziert werden

Grundlegende Konzepte am eigenen Beispiel

Aufgabe:

Grundlegende RL Konzepte am eigenen Beispiel

- **Umgebung (Environment)**: Wohnung
- **Zustand (State)** s: Temperatur, Bewegungsmelder, Sensoren am Fenster, **Zeit,** Videokamera Bilder
- **Agent:** **(Mensch / Computer)**: Heizung
- **Aktion a**: Termostat regeln
- **Belohnung (Reward) r**: Nähe zum Sollwert, negativ für große Abweichungen
- Ziele: angenehme Temperatur, Energie & kosten sparen
- **Policy (π)**: wenns warm ist runter regeln , wenns kalt ist hoch regeln

Grundlegende Konzepte am eigenen Beispiel

Aufgabe:

Grundlegende RL Konzepte am eigenen Beispiel

- **Umgebung (Environment)**: CNC Maschine
- **Agent:** **(Mensch / Computer)**: **Regler** (manchmal: Bilderkenner)
- **Zustand (State)** s: Sensoren: Leistung, Temperatur, Härte(indirekt) ...
- **Aktion a**: Werkzeug auswählen, Schnitt-Geschwindigkeiten, Drehrichtung
- **Belohnung (Reward) r**: Knicken = -10 ... Kaputt : -100 … & implizites *Ziel : schnell sparsamster Weg verschleissarm sicher*
- **Policy (π)**: niemanden Umbringen. nicht ins Futter fahren. härter => kleinere Schnittwerte

Grundlegende Konzepte am eigenen Beispiel

Aufgabe:

Grundlegende RL Konzepte am eigenen Beispiel

- **Umgebung (Environment)**: Mannschaft im Computer Spiel(feld)
- **Zustand (State)** s: Positionen
- **Agent:** **(Mensch / Computer)**: C.
- **Aktion a**: Schuss, Flanke, Dribbling
- **Belohnung (Reward) r**: TOOOOOOR! +100 Faul +10 Passen +1 Abseits -10 Rote Karte -50 & implizites *Ziel*
- **Policy (π)**: Formation ändern / beibehalten. Ball flach halten.

Grundlegende Konzepte am eigenen Beispiel

Aufgabe: **2 möglichst unterschiedliche Beispiele (ohne GPT)**

Grundlegende RL Konzepte am eigenen Beispiel

- **Umgebung (Environment)**: Smart Hause, Licht Strom Heizung
- **Zustand (State)** s: *Kombination*: Sensoren: Zeit, Fenster Offen, Licht, Strompreis, jemand da?
- **Agent:** **(Mensch / Computer)**: PC
- **Aktion a**: Geräte / Jalousie / Fenster / Heizung **hoch runter** fahren.
- **Belohnung (Reward) r**: über Nutzer: "keine Eingriffe" => plus, "Eingriff" => negativ & implizites *Ziel*
- **Policy (π)**: z.B. Geräte an, Bewohner weg => Geräte aus! … Draussen heiß => Heizung aus! Tabelle / Algorithmen

Grundlegende Konzepte am eigenen Beispiel

Aufgabe:

Grundlegende Konzepte am eigenen Beispiel

- **Umgebung (Environment)**: Fahrradfahren auf Straße
- **Zustand (State)** s: Geschwindigkeit, Position, Winkel, Wetter …
- **Agent:** **(Mensch / Computer)**: wir
- **Aktion a**: lenken, treten, bremsen
- **Belohnung (Reward) r**: Zeit zum Ziel
- **Policy (π)**: schwer in Worte zu fassen! Bremsen bei Bergabfahren… (high level vs low level)

Grundlegende Konzepte am eigenen Beispiel

Aufgabe:

Grundlegende Konzepte am eigenen Beispiel

- **Umgebung (Environment)**: Autonomes Fahren auf Straße (simuliert)
- **Zustand (State)** s: Abstände, Geschwindigkeit, Position, Winkel, Sensoren: Motoren, Passanten, Ampel vision, …
- **Agent:** **(Mensch / Computer)**: eingebaut
- **Aktion a**: lenken, bremsen (hart, weich, Notfall), Beschleunigung
- **Belohnung (Reward) r**: Kollision -100 '*unnötig* Bremsen' -10 …
- **Policy (π)**: schwer in Worte zu fassen! weiter wenn keine Fußgänger...

Grundlegende Konzepte am eigenen Beispiel

Aufgabe:

Grundlegende Konzepte am eigenen Beispiel

- **Umgebung (Environment)**: Börse
- **Zustand (State)** s: Preis der Aktie
- **Agent:** **(Mensch / Computer)**: Bot
- **Aktion a**: buy/sell
- **Belohnung (Reward) r**: Gewinn
- **Policy (π)**: Tief kaufen, hoch verkaufen

Grundlegende Konzepte am eigenen Beispiel

Aufgabe:

Grundlegende Konzepte am eigenen Beispiel

- **Umgebung (Environment)**: Bank
- **Zustand (State)** s: Informationen über Kunden
- **Agent:** **(Mensch / Computer)**: Angestellter
- **Aktion a**: gebe Kredit / verweigere Kredit , setze Zinssatz
- **Belohnung (Reward) r**: Rückzahlung + Zins-Gewinn
- **Policy (π)**: Kreditvergabe bei Glaubwürdigkeit

Grundlegende Konzepte am eigenen Beispiel

Aufgabe:

Grundlegende Konzepte am eigenen Beispiel

- **Umgebung (Environment)**: Fertigungslinie (Förderbänder + Werkstücke)
- **Zustand (State)** s: Positionen, Robotorarm Winkel …, Visueller Scanner
- **Agent:** **(Mensch / Computer)**: Bot
- **Aktion a**: Greifen, …
- **Belohnung (Reward) r**: +1 bei korrekter Sortierung
- **Policy (π)**: manueller Algorithmus basierend auf den Sensoren.

Grundlegende Konzepte am eigenen Beispiel

Aufgabe:

Grundlegende Konzepte am eigenen Beispiel

- **Umgebung (Environment)**: Kaffemaschine
- **Zustand (State)** s: Temperatur, Füllstände, …?
- **Agent:** **(Mensch / Computer)**: trifft Entscheidungen:
- **Aktion a**: Mische Milch mit Kaffebohnen …
- **Belohnung (Reward) r**: Skala 1-10 über Sprachmodelle, Bilderkennung, Herzsensor!
- **Policy (π)**:

Grundlegende Konzepte am eigenen Beispiel

Aufgabe:

Grundlegende Konzepte am eigenen Beispiel

- **Umgebung (Environment)**: single cell foundation model, question: inhibitory
- **Agent:** **(Mensch / Computer)**:
- **Zustand (State)** s: gene expression
- **Belohnung (Reward) r**: text comparison
- **Aktion a**: GPRO Action Ai = (ri - μ) / s
- **Policy (π)**: GPRO ^^

Grundlegende Konzepte am eigenen Beispiel

Aufgabe:

Grundlegende Konzepte am eigenen Beispiel

- **Umgebung (Environment)**: Energy management Eigenheim, PV, Batterie, Verbraucher. **Ziel**
- **Agent:** **(Mensch / Computer)**:
- **Zustand (State)** s: Ladezustand, PV Leistung, $/kwh, Netzlast
- **Belohnung (Reward) r**: Gewinn bei Verkauf, +10 bei Versorgung Haus
- **Aktion a**: Laden / entladen / Verbraucher ein/ausschalten, PV "strings" ein/ausklinken (Schatten)
- **Policy (π)**: sollte vom Netz gelernt werden: möglichst bei Sonne verbrauchen …

off-policy RL ≈ supervised learning: sammle erst alle Daten, optimiere dann (aber: was für Daten!? fast immer die selben)

on-policy RL ≈ optimiere im Betrieb, policy beinflusst Daten! (generiert viel mehr Datenpunkte!)

Grundlegende Konzepte am eigenen Beispiel

Aufgabe:

Grundlegende Konzepte am eigenen Beispiel

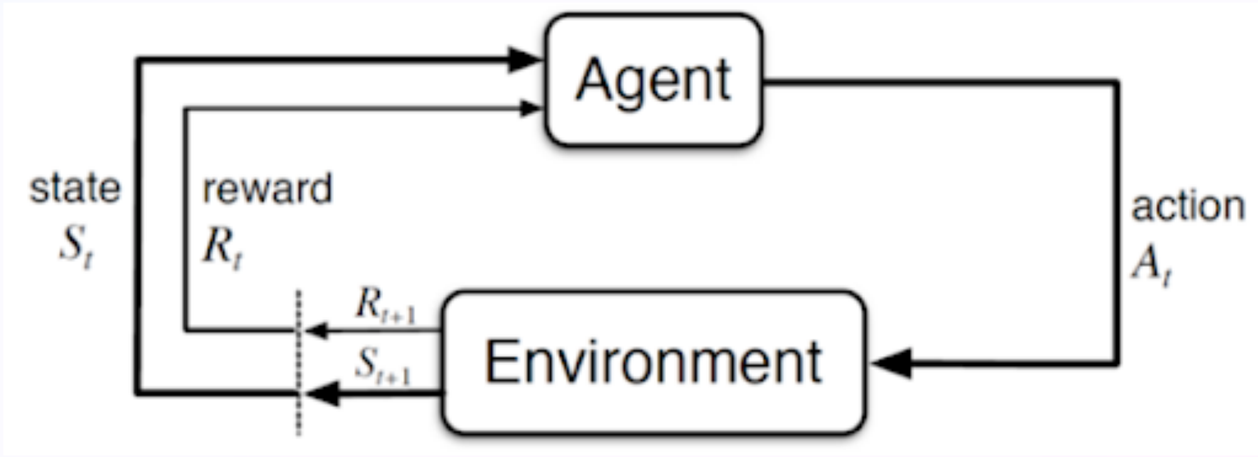
- **Umgebung (Environment)**: Roboter mit Schaufel
- **Zustand (State)** s: Lochvolumen, Gewicht auf Schaufel
- **Agent:** **(Mensch / Computer)**: trifft Entscheidungen:
- **Aktion a**: Schaufel heben, …
- **Belohnung (Reward) r**: Volumen
- **Policy (π)**: selbsterlernt: Schaufel nicht zu voll…

reward-hacking!

Vertiefung Grundlegender Konzepte

Vertiefung Grundlegender Konzepte

- **Agent**: trifft Entscheidungen
- **Umgebung (environment)**: liefert Rückmeldungen (Zustände, Belohnungen)
- **Zustand (state) s**: Beschreibung der aktuellen Situation / Position **Zahlen, Vektor, Matrizen, *Tensoren***
- **Aktion (action) a**: Handlung des Agenten s->**s'** Züge **Zahl oder Vektor**
- **Belohnung (reward) r**: Feedback der Umgebung (real, 'abstrakt' oder von uns selbst, Hauptsache quantisierbar, also als **Zahl**)
- **Policy (π)**: Strategie, die angibt, welche Aktion in welchem Zustand gewählt wird
  - deterministisch **π(s)**=a oder als stochastisch **π(a | s)**
  - Funktion def **policy**(state): return (random) action
  - Netz **model_policy**(state): return action probability
    - so wie Klassifizierung **softmax**



Agentensysteme, Spiel-Agenten

Ein **Agent** ist ein System, das in einer Umgebung **autonom Entscheidungen** trifft und durch seine **Aktionen** *implizite Ziele* verfolgt. In der Regel reagiert ein Agent auf **Umweltreize** und hat die Fähigkeit, **selbstständig** zu **lernen** und sich an neue Situationen anzupassen.

Aktion

Übergang von einem Zustand zum nächsten.

- Auswahl anhand von **Policy (π)**: *Strategie*, die angibt, welche **Aktion** in welchem Zustand gewählt wird

Aktionen beeinflussen den inneren State der Umgebung, gegebenenfalls bekommen wir eine Belohnung

Environment

Environment

- Kann man manchmal selbst definieren
- Oder über API Schnittstelle / Bibliothek

E(s,**a**)=(r,**s'**) Environment gibt nächsten Zustand

s' = nächster Zustand / Situation / **S**pielstand

Reward

Belohnung

Extrinsischer Reward = Das was die **Umgebung** uns gibt (oft nur +1)
  standard "von Aussen" (*exter* (“outer, outward”) + *secus* (“side”))

Intrinsischer Reward = Das was wir uns **selbst** als Belohnung definieren.

Intrinsischer Reward = "custom reward **shaping**" :
  wir formen den gegebenen extrinsischer Reward zu unseren zwecken.

z.B. Schach :
+1 bei Gewinn => **+1000**
Schlagen der anderen Dame = **+9**

Immediate Reward: **Sofortige** Belohnung
Delayed Reward: **Verzögert** Erst am Ende des Spiels (z.B. +1 Sieg)

Extrinsischer Reward : äußere Belohnung

Policy

- Auswahl der Aktion anhand von **Policy (π)**:

- Strategie, die angibt, welche **Aktion** in welchem Zustand gewählt wird

π(s)=a Policy wählt Aktion deterministisch

π(a|s)=… Policy wählt Aktion probabilistisch

E(s,**a**)=(r,s') Environment gibt nächsten Zustand und reward

Policy: Funktion welche in state eine action wählt
Tabelle map[s]=a oder Funktion
def policy(s): return … oder Netz

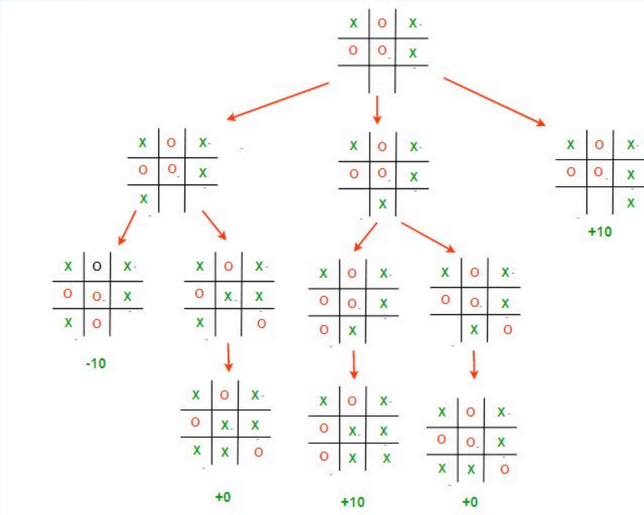
Training durch Belohnungen

Intuitiver Ansatz:
Merke bei jedem Spiel *Verlauf* alle **Positionen**
Zähle wie oft eine Position zu **Siege**n führte
Wähle neue Positionen teils zufällig nach Historie

Entscheidungsbaum -
Decision tree

Probleme:
- Keine guten Züge für **unbekannte Situationen** erzeugen
- **zu viele** mögliche **Felder * Züge**

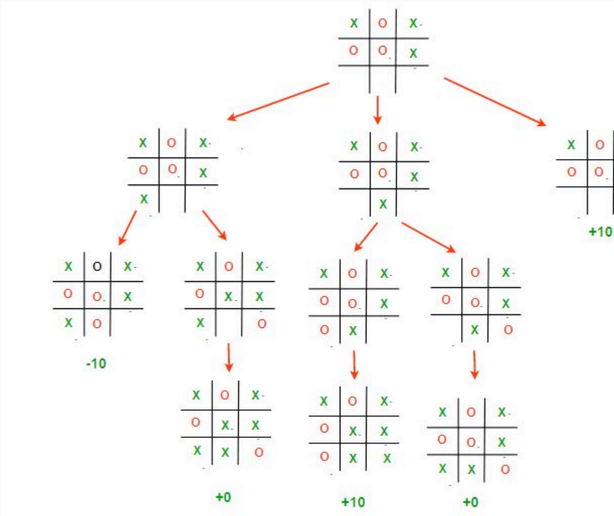
Generalization!



Training durch Belohnungen

Decision Trees:
Speicher explodiert (z.B. Schach 10^30) bei sparse rewards
Generalisierung: Decision Trees liefern nur was bei bekannten Zügen
Vorteil: gegebenenfalls präziser
Nur für Diskrete Probleme! (Kontinuierliche Probleme? Winkel über Töpfe)

Neuronale Netze:
Für Kontinuierliche Probleme gemacht
Speicher begrenzt
Generalisierung: können auch auf unbekannte Situationen reagieren, π(s) Vorhersagen auch wenn s neu ist



Belohnungen - credit assignment problem

- Wie können wir aus der *verzögerten* Belohnung der Umgebung lernen (**zurückverfolgen**) welche Aktionen **a** / Stellungen **s**/ Strategien π gut sind?

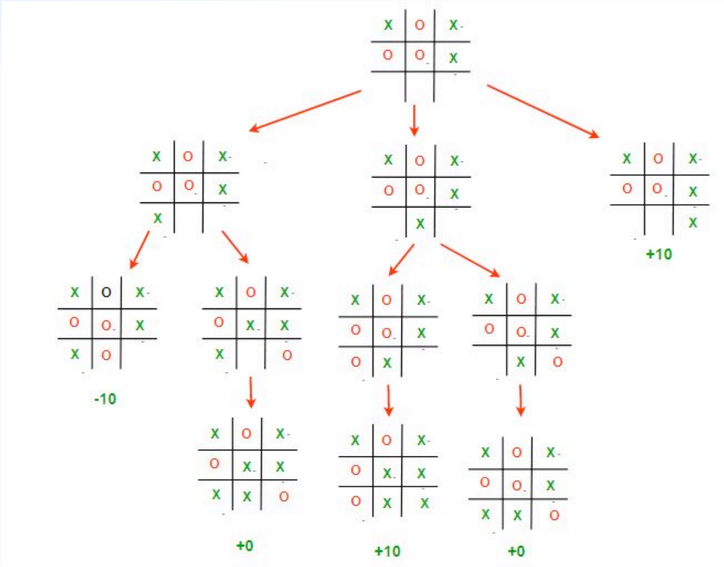
Was heisst eigentlich gut?

Erwartete **Belohnung zu maximieren!**

Theoretisch 'optimal' $E[\pi]=max$

Das wird uns in den nächsten Tagen beschäftigen …

Welche **Aktion** hat zum **Erfolg beigetragen**(credit)?



Beispiele für konkrete Policies

Temperatur Steuerung:

- Wenn die Temperatur im Raum um 5 Grad unter dem Zielwert liegt, heize mit mittlerer Leistung
- **Wenn** Temperatur ≥ Solltemperatur: **Heizung ausschalten**.

Roboter-Navigation:

- **Zustand:** Position und Orientierung des Roboters.
- **Policy:** Wenn das Hindernis innerhalb von 1 Meter ist, drehe um 90 Grad nach rechts. Wenn kein Hindernis, bewege dich geradeaus.

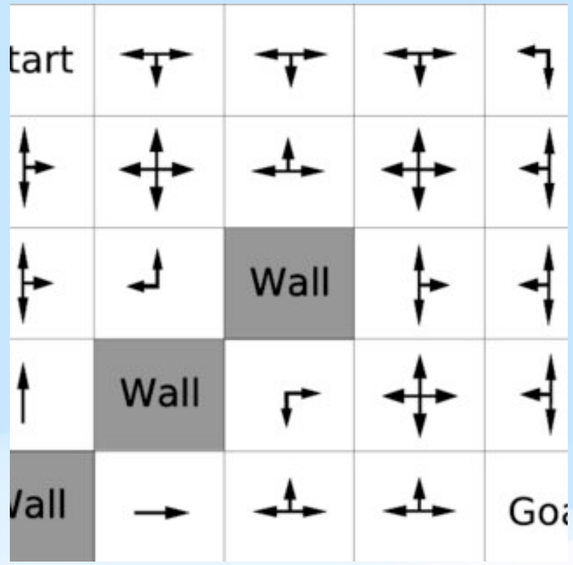
2. **Finanzportfolio-Management:**
   - **Zustand:** Marktdaten (z.B. Aktienpreise, Volatilität).
   - **Policy:** Wenn Aktienpreis steigt um 5%, kaufe mehr Anteile. Wenn der Preis um 10% fällt, verkaufe.
3. **Spielsteuerung (z.B. Schach):**
   - **Zustand:** Aktuelle Brettposition.
   - **Policy:** Wenn König bedroht, ziehe eine Figur zur Verteidigung. Wenn der Gegner eine Schwachstelle hat, setze eine aggressive Figur ein.
4. **Versicherung**
   - Zustand: Kunden mit Verträgen
   - Policy: Wenn Kunde Unfall erzeugt erhöhe Prämie
   - Belohnungsfunktion: Maximiere Gewinne der Versicherung
5. Eigen Ernährung:
   - Zustand: Gesundheit
   - Policy: Esse süsses
6. Sicherheit
   - Zustand: Friede?
   - Policy: Tit for tat

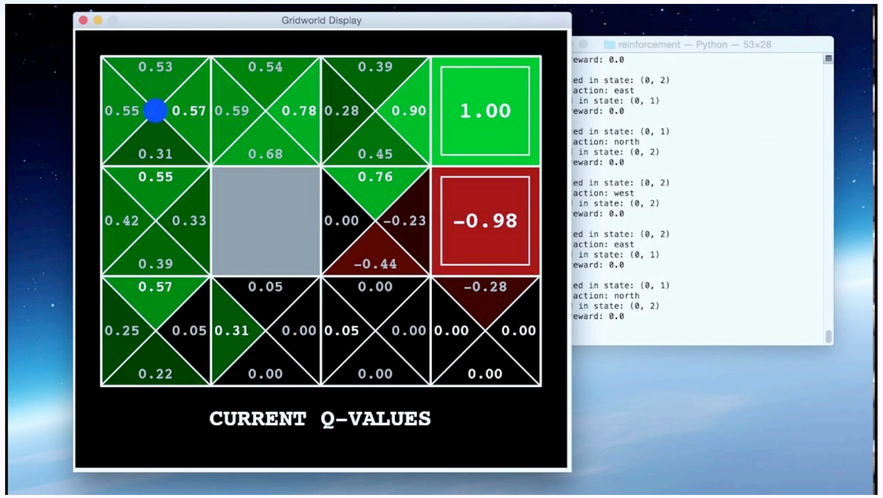
`hat erst mal nichts mit Deep Learning zu tun, aber …`

Grundlegende Konzepte am Beispiel **Grid-World**

📦 Grid-World

- **Umgebung**: Gitter mit Hindernissen und Ziel
- **Zustand**: Position des Agenten
- **Aktion**: Oben, Unten, Links, Rechts
- **Belohnung**: Ziel erreichen = +1, Wand treffen = -1
- **Policy**: Wähle kürzesten Weg zum Ziel anhand von:
- **Value**: Wie gut ist eine bestimmte Position?
- **Quality**: Wie gut ist eine bestimmte Richtung an Position?





z.B. Rasenmäher-Roboter

Klasse: Pfadfindungs Probleme

Kapitel 3.2 im Buch

Grundlegende Konzepte am Beispiel **Einarmiger Bandit**

🎰 Bandit-Problem

- **Umgebung:** Spielhalle mit mehreren Spielautomaten, Typ "**Einarmiger Bandit**"
  - **Zustand**: keiner bzw. konstant
  - **Aktionen**: Auswahl eines Automaten/Hebel
  - **Belohnung**: Sofort, aber stochastisch (jeder Automat mit seiner eigenen W-Keit)
  - **Ziel**: Den Automaten mit der höchsten Auszahlung finden (mehrere Spiele notwendig!)
  - Policy: **sehen wir** (idealerweise: am Ende immer den Automaten mit **höchten Gewinn wählen**)

- wie geht man das geschickt an? Morgen: systematisch / mathematisch / algorithmisch.

k **Hebel** (engl. *arms*), jeder mit unbekannter Belohnungsverteilung

Notation

Notation in Sutton:

s state
a action
S set of all nonterminal states
S**+** set of all states, including the **terminal state**
A(s) set of actions possible in state s
R set of possible **r**ewards
t discrete **t**ime step
T final time step of an episode
S_t state at t
A_t action at t
R_t reward at t, dependent, like S_t, on A_{t-1} and S_{t-1}
G_t **gain** / goal / return (cumulative discounted reward) following t
$G_t(n)$ n-step return (Section 7.1)
$G_t\lambda$ λ-return (Section 7.2)
π policy, decision-making rule
π(s) action taken in state s under deterministic policy π
π(a | s) probability of taking action a in state s under stochastic policy π
p(s′,r | s,a) probability of transitioning to state s′, with reward r, from s,a
v_p / vπ(s) value of state s under policy π (expected return)
$v_*(s)$ value of state s under an *optimal* policy
qπ(s,a) / $q_p(a)$ value of taking action a in state s under policy π
$q_*(s,a)$ value of taking action a in state s under the optimal policy
$V_t(s)$ estimate (a random variable) of vπ(s) or $v_*(s)$
$Q_t(s,a)$ estimate (a random variable) of qπ(s,a) or $q_*(s,a)$
v(s,w) approximate value of state s given a vector of weights w
q(s,a,w) approximate value of "quality" state–action pair s,a given weights w

Notation

Notation in *Sutton*:

s **s**tate
a **a**ction
S set of all nonterminal states
S**+** set of all states, including the **Terminal state** (z.B. Schachmatt / game over)
A(s) set of actions possible in state s
R set of possible **r**ewards (vs Return)
t discrete **t**ime step
T final time step of an episode
s_t state at t
a_t action at t
R_t reward at t, dependent, like St, on At−1 and St−1
π policy, decision-making rule
π(s) action taken in state s under deterministic policy π
π(a | s) probability of taking action a in state s under stochastic policy π

Grundlegende Konzepte am Beispiel ...

Aufgabe: Finde 2 Beispiele um die Konzepte von RL zu veranschaulichen:

- **Umgebung / Problem XYZ**
- **Zustand (state s)**:
- **Aktionen a**:
- **Belohnung (Reward) r:**
- **Policy (π) Strategie:**

Gegebenenfalls:

- Value Function v (value of state) oder
- Qualität q (Quality of action)

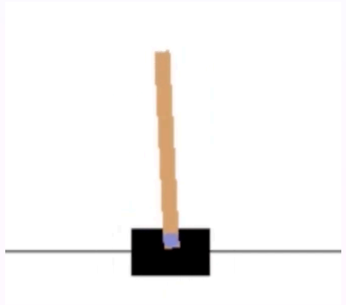
Pause

Pause

Steuerung dynamischer Systeme

Anwendung von RL (reinforcement learning) auf RL (real life)

Dynamisches System: Ein System, das seinen Zustand über die Zeit basierend auf internen Regeln oder physikalischen Gesetzen verändert. Beispiele sind mechanische, elektrische, biologische oder chemische Systeme.
Steuerung: Der Prozess der Beeinflussung eines dynamischen Systems durch gezielte Eingaben (Kontrolleingaben), um ein bestimmtes Verhalten zu erreichen.
Regelung: Ein Spezialfall der Steuerung, bei dem ein Sollwert (z.B. eine bestimmte Position oder Temperatur) automatisch erreicht wird.



Auch im Computer: resource management, task scheduling, or control loops within operating systems.

Deep Reinforcement Learning

Damit spart man es sich mühsam eine **optimale Strategie (Policy)** selbst zu Programmieren, das Netz lernt die *passenden* Aktionen automatisch! Z.B. Positionen mit höchstem Wert anstreben, Aktion mit höchster Qualität wählen "**greedy**".

Was gibt das **Netz** konkret aus?

Aktionen codiert als **Klassen:**
z.B. Joystick Bewegungen (links:**0** rechts:1 oben:2 …)
Tic-Tac-Toe **9 Positionen** als 9 Klassen
illegale Positionen (belegt oder not-in-reach) mit loss Funktionen bestrafen!

Aktionen codiert als **Werte** (Regression):
z.B. Kaufpreis, Beschleunigung(Geschwindigkeit)…

Wie wird es trainiert?

einfaches Lernen

- Ursprünglich sehr mathematisches Feld aber
- in der Anwendung einer **einfachen Grundidee** zusammen mit **Deep Learning** recht **einfach.**

- Wir können alle Architekturentechniken und Tricks aus dem Machine Learning Kurs direkt auf unseren lernenden Agenten übertragen
- AI Blackbox in neuem Umfeld
- Ideal: "Computer, mach mal" der Computer (**Agent)** braucht dann nur passende Schnittstelle zu einer **Umgebung**
- um durch **Interaktion** möglichst optimale **Entscheidungen** zu treffen

Nobelpreisträger Jeffrey Hinton: "ich verstehe immer zuerst die **Intuition** und später dann vielleicht die Mathematik!"

Mathematik soll uns ein **nützliches Tool** sein um die Konzepte und die Intuition zu vereinheitlichen **kompakt** zu schreiben.

Unterschiede zu anderen Lernverfahren

Supervised Learning: Lernen mit **gelabelten** Beispielen

Unsupervised Learning: **Mustererkennung** ohne Labels (automatische Clusterung)

Reinforcement Learning:

- nicht immer direkter **Lehrer / Datensatz** vorhanden
- Lernen durch **eigenes Handeln**
- **Belohnung** statt Fehler**label** (**indirekt**, schwieriger)
- oft **unendliche** neue 'eigene' Daten durch **Exploration**

💡 alle bisher gelernten Lernverfahren, **Architekturen** und **Techniken** lassen sich aber an bestimmten Stellen in DRL verwenden.

Reinforcement Learning

💡 Idee als Pseudocode

solange nicht abgeschlossen:

beobachte Zustand **s**

wähle Aktion **a** gemäß Policy **π(s)**

führe Aktion a in Environment aus

beobachte neue Belohnung **r** und neuen Zustand **s'**

aktualisiere Policy π basierend auf (s, a, r, s')

Reinforcement Learning

Idee Implementierung Pseudo Code

```
environment = load(…)
agent = init(environment)
while not is_game_over():
  state = environment.get_state()
  action = agent.choose_action(state) # via policy
  reward, next_state = environment.step(action)  <<
  agent.update_policy(state, action, reward, next_state)
```

Pause

No free lunch theorem

"Es gibt keine Architektur die bei beliebigen Problemen zu den besten Ergebnissen führt."

Sehr genau auf die Axiome schauen unter welchen das Theorem gültig ist.

In der Praxis gibt es Vorwissen und Ähnlichkeiten: Text, Bild, …

Für solche Probleme kann man sagen dass unsere Reinforcement Learning Architekturen universell besser sind als Vorgänger Architekturen. (Bei begrenztem compute)

Deep Reinforcement Learning

Deep Reinforcement Learning erweitert klassische Reinforcement Learning (RL) Ansätze durch die Verwendung von **tiefen** neuronalen Netzen, um komplexe Zustandsräume besser zu verarbeiten und zu generalisieren. Bekannte Algorithmen sind **Deep Q-Learning (DQN)**, **Policy-Gradient-Methoden**, und **Actor-Critic-Modelle**.

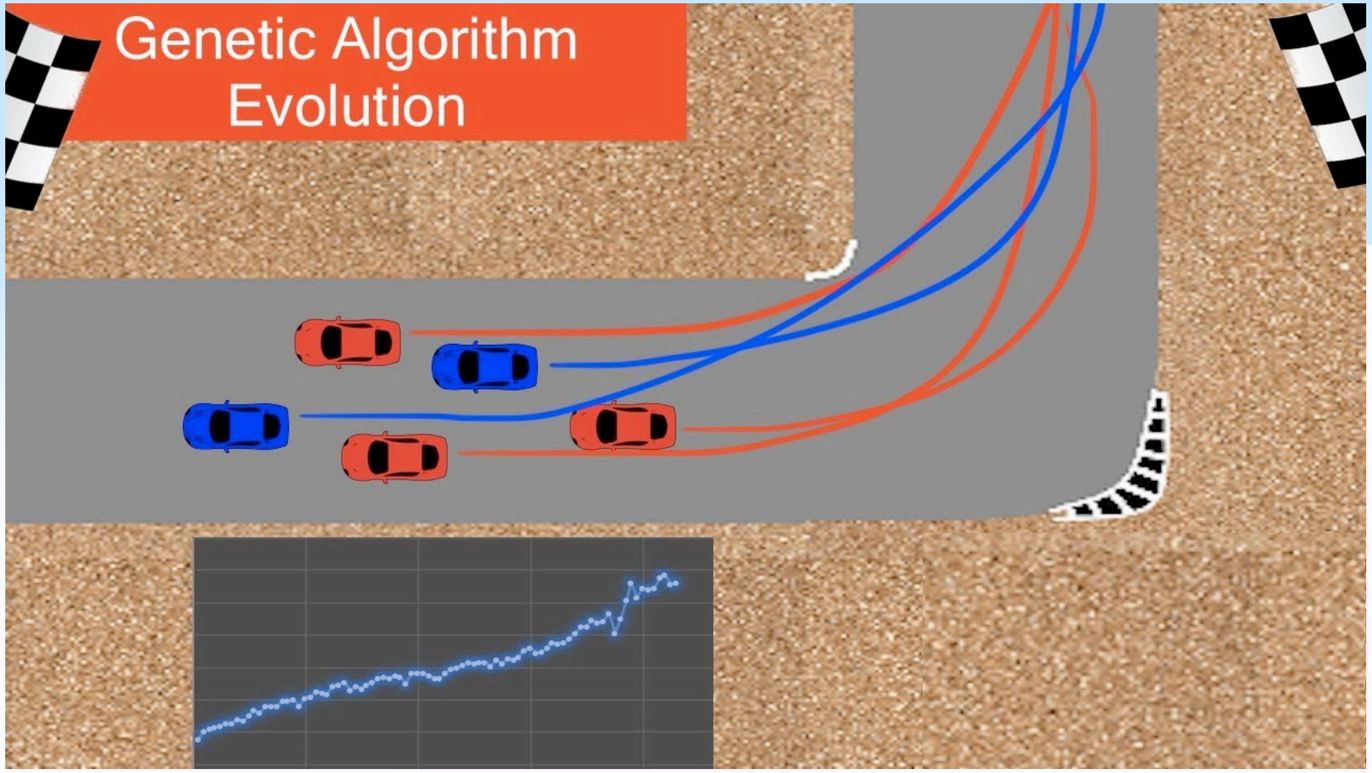
💡 Der **Zustandsraum** wird bei Spielen schnell quasi **unendlich**, so dass wir auf unser Netz für **Schätzungen** angewiesen sind.

genetische Algorithmen

"***genetische Algorithmen***." zufällig eine erste Generation von 100 Policies erzeugen, ausprobieren und die **schlechte**sten 80 **Policies eliminieren**. **Gradient**enfrei**!** Methode unabhängig von RL!

Die 'überlebenden' Policies können dann automatisch leicht verändert werden und das Spiel beginnt von neuem.

So könnten etwa selbst fahrende Autos im spiel an verschiedenen Positionen zur Kurve einschwenken...



`hat erst mal nichts mit Deep Learning zu tun, aber …`

Github

privates repository mit code für jeden Tag:

http://github.com//pannous/RL.git

Github account anlegen oder in chat

(freiwillig, ansonsten kann ich zip zur Verfügung stellen)

Buch

TEIL II
18 Reinforcement Learning Seite 727

https://github.com/ageron/handson-ml3

https://pytorch.org/tutorials

Buch

Alexander Zai Brandon Brown

Einstieg in Deep Reinforcement Learning

KI-Agenten mit Python und PyTorch programmieren

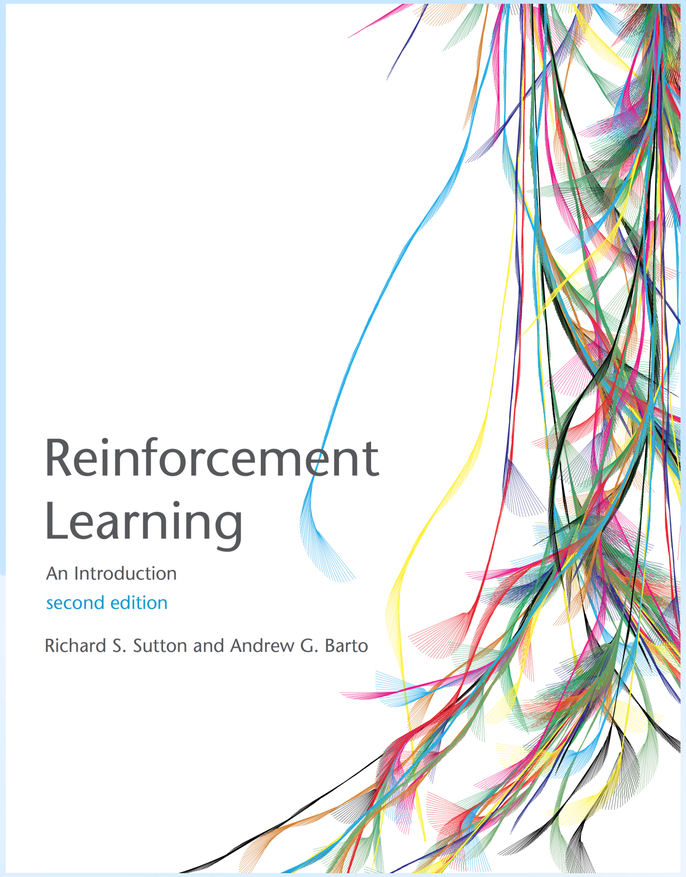
Link zum Download sollte bis spätestens 12:00 Uhr per Email vorliegen.

Lesen Tag 1 a:
Teil I: Grundlagen - **Kapitel 1**
Einführung in das Deep Reinforcement Learning

bis Seite **25 oder 42**, maximal bis Seite **57**

Lesen Tag 2:
Teil I: Grundlagen - **Kapitel 2**
Modellierung von Reinforcement-Learning-Problemen: Markov Decision Processes

Buch



Sutton: bis Seite 53

Episodic and Continuing Tasks

Wenn die Folge der Zustände $\mathbf{s}_t$ irgendwann in einem so genannten **terminalen Zustand** endet dann spricht man von **episodischen Problemen.** Das ist unser Standardszenario, bei welchen wir am Ende eine **finale Belohnung** erhalten. Letzter Zustand **T**=S_t(final)

Ansonsten spricht man von **kontinuierlichen Aufgaben** (werden wir nur am Rande betrachten)
Auch: Nicht endlichen Problemen.

Episode = Menge aller Zustände { s_t , a_t } **"ein Spiel"**

Vorausblick: off policy learning: speichere Episoden und lerne neue Policy später!

Model

Ein **Modell** ahmt das Verhalten der Umgebung nach oder erlaubt allgemeiner gesagt Rückschlüsse darauf, wie sich die Umgebung verhalten wird. Zum Beispiel könnte das Modell bei gegebenem Zustand und gegebener Aktion den resultierenden nächsten *Zustand* und die nächste *Belohnung* **vorhersagen.**

Dank Deep Learning bilden sich Modelle **automatisch** aus d.h. die Muster welche unsere neuronalen Netze 'erkennen' spiegeln irgendwann idealerweise die Muster der realen Umgebung wieder!

Modellbasierte Methoden verwenden **Planung**, im Gegensatz zu einfacheren modellfreien Methoden, die explizit durch Versuch und Irrtum lernen.

In Chapter 9 we explore reinforcement learning systems that simultaneously learn by trial and error, learn a model of the environment, and use the model for planning.

Zusammenfassung

Reinforcement Learning (Verstärkungslernen) ist eine Methode des maschinellen Lernens, bei der ein Agent in einer **Umgebung** agiert, um durch Interaktionen zu lernen. Das Ziel ist es, eine **optimale Strategie (Policy)** zu finden, die zu maximalen kumulierten Belohnungen führt. Diese Art des Lernens unterscheidet sich von *Supervised* Learning (überwachtem Lernen), da der Agent nicht explizit mit den richtigen Aktionen trainiert wird, sondern durch Versuch und Irrtum lernt, **basierend auf den Rückmeldungen (Belohnungen) der Umgebung.**

- 💡 **Belohnung nicht nur bei Sieg sondern auch für neue Situationen**
- 💡 **Belohnung für verschiedenste Fortschritts-Metriken**

Zusammenfassung

Wann **RL** statt hardcoden?

- Sobald man 1000e if/elses hat zum ausspucken von numerischen Regelwerten bei zu komplexen Problemen
- Wenn Zustandsraum zu groß / **chaotisch** wird
- Bei kontinuierlichen Zustandsraumen mit fein
- Mustererkennung (high dim / Bilder z.B. MNIST…)

Reinforcement Learning (Verstärkungslernen) ist **gut**, wenn wir das **Ziel** kennen aber den **Weg** nicht!

Zusammenfassung

Wann **RL** statt supervised?

- wenn keine supervised Daten vorliegen
- wenn supervised Daten zu begrenzt sind
- wenn man komplett neuen Situationen ausgesetzt ist

Model basiert.

pragmatisch:
Mit Daten vortrainieren, und dann RL weiter machen

Vorbereitungen

Passender Rechner? Alfatraining laptop mit GPU?

IDE Einrichten (**PyCharm** oder VsCode oder **CoLab**, Notebooks, Spyder …)

Freiwillig: **Github** account anlegen (und Name pasten für Kooperation)

Empfohlen: **ChatGPT** oder Anthropic Claude oder DeepSeek, Gemini account

ChatGPT am besten nur zum erklären, nicht für kompletten Code!

gegebenenfalls schon mal

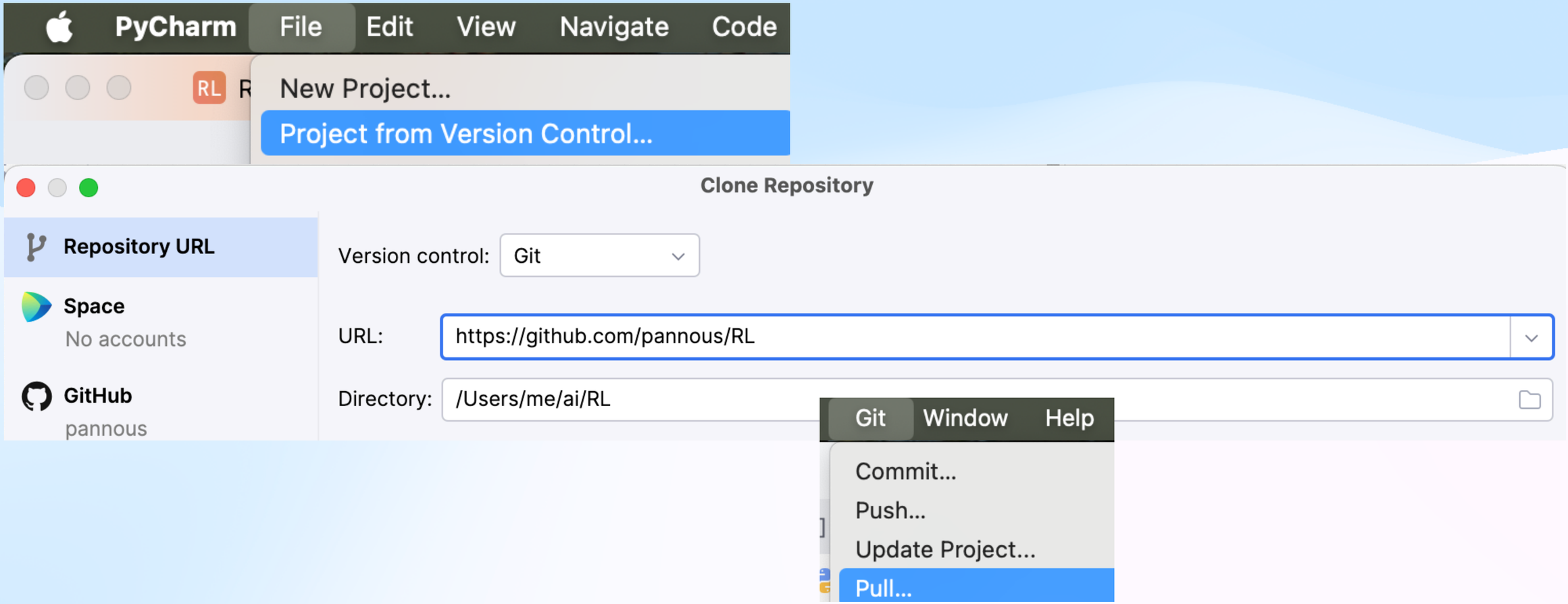
`pip install **torch**`
`pip install tensorflow`

(oder per IDE)

Aufgaben

Gegebenenfalls git installieren, gelegentlich mit Git => Pull aktualisieren

Starte code/mnist_trivial.py
Starte code/pole/pole_random.py



Aufgaben

Gegebenenfalls git installieren, gelegentlich mit Git => Pull aktualisieren

Bringe mnist_trivial.py zum Laufen,
Vergleiche mit mnist_better.py und eigenen tensorflow (vom ML Kurs)

Starte code/pole/pole_random.py und passe policy an
Starte code/car/car_custom_policy.py und passe policy an

Eine Zeile, nur um ein Gefühl für eigene Policy zu bekommen
 z.B. return 1 // immer links

Vokabular

Grundlegende Konzepte

- **Agent**: trifft Entscheidungen
- **Aktion (Action) a**: Handlung des Agenten s->**s'** Züge
- **Umgebung (Environment)**: liefert Rückmeldungen (Zustände, Belohnungen)
- **Zustand (State) s**: Beschreibung der aktuellen Situation / Position
- **Belohnung (Reward) r**: Feedback der Umgebung (real, 'abstrakt' oder von uns selbst, Hauptsache quantisierbar, also als Zahl)
  custom reward / reward shaping / intrinsic / extrinsic / sofort / zwischen / final / terminal / sparseπ
- **Policy (π)**: Strategie, die angibt, welche Aktion in welchem Zustand gewählt wird π(s)=a oder π(a | s) = n%
- Spiel = Episode ≈
- Trajektorie = Pfad (theoretische Zugabfolge)
- Terminaler Zustand ("Game Over")
- Modell
- reward hacking (cheating)
- `reward` **`shaping`**
- Zug, Runde, Step ≈ eine gewählte Aktion + neuer Zustand

⚠️ Interne Hilfskonzepte / klassische Theorie ⚠️ :

- **Gewinn** *G (goal / gain)* nach Folge von Aktionen (bei policy π) ∑Belohnungen pro Episode / ab Zustand
- **Value Function v**: Erwartete zukünftige Belohnung/Gewinn ab einem **Zustand** ≈
- **Qualität (Quality) q**: Erwartete zukünftige Belohnung/Gewinn für eine **Aktion** in einem Zustand.