Digital Applications & Data Management

WS25/26

Modul 13

Dr. Jens Kohl

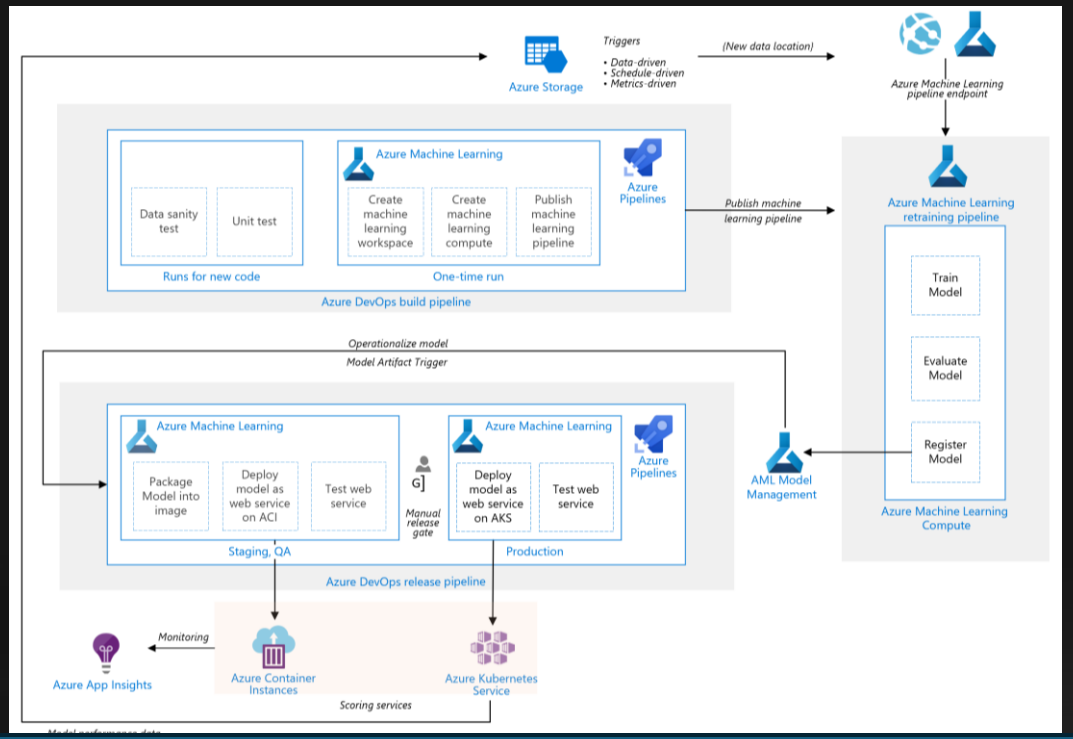
Roadmap Vorlesung

1. Grundlagen Künstlicher Intelligenz, Machine Learning und Daten
2. Grundlagen Data Science
3. Angewandte Data Science (Teil 1)
4. Angewandte Data Science (Teil 2)
5. Data Science Use Case
6. Grundlagen unüberwachtes Lernen
7. Grundlagen überwachtes Lernen (tabellarische Daten)
8. Machine Learning Use Case
9. Grundlagen überwachtes Lernen (Bilddaten)
10. Transfer Learning Bilddaten und Fallbeispiel
11. Grundlagen Generative AI
12. Prompt Engineering, Agenten
13. Ausblick, Wiederholung, Fragestunde
14. Fragestunde

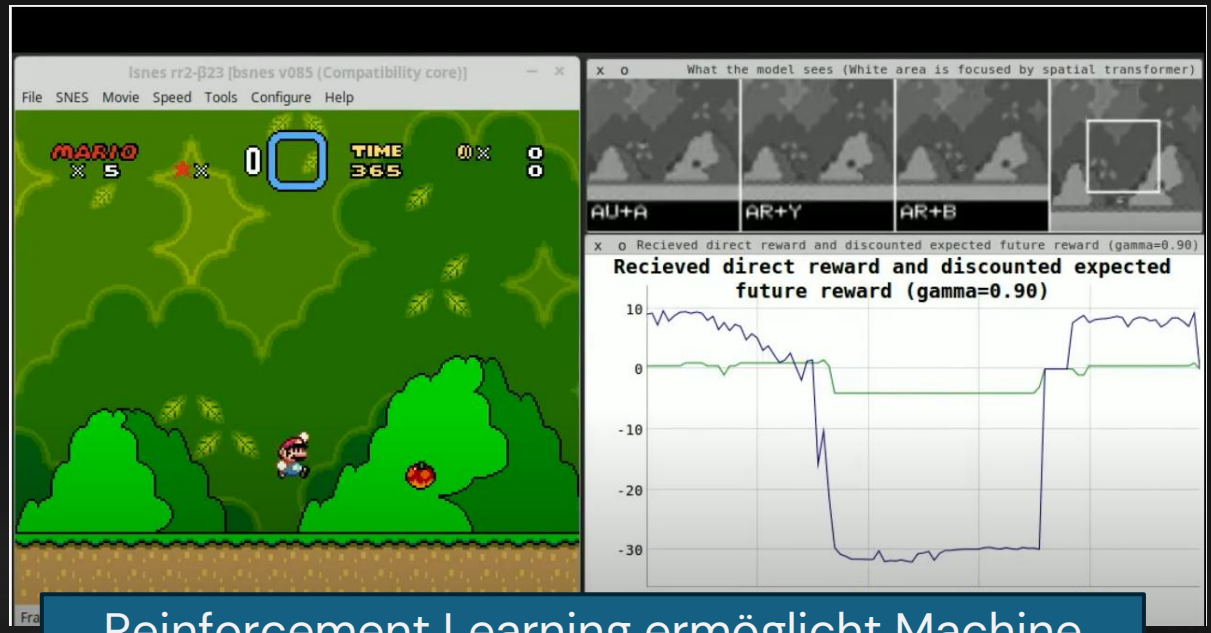
13. Ausblick

Was machen wir heute?

Motivation



Beschleunigung des Trainings sehr großer Machine Learning Modelle durch Nutzen Cloud



Reinforcement Learning ermöglicht Machine Learning Modelle die selbstständig lernen und sich kontinuierlich verbessern können.

Machine Learning in the cloud

Machine Learning in the cloud

Einleitung

- Bis jetzt haben wir vorwiegend auf einem Rechner (lokal/ Cloud) unsere Use Cases umgesetzt.
- Wir haben gesehen, daß Machine Learning sehr ressourcen- und zeitintensiv ist:
  - Speicherbedarf: je mehr Daten, desto bessere Ergebnisse (2- oder 3-fache Datenmenge besser als jedes Parameter-Tunen[1])
  - Rechenbedarf: je länger trainiert wird, desto (meist) besser sind die Ergebnisse

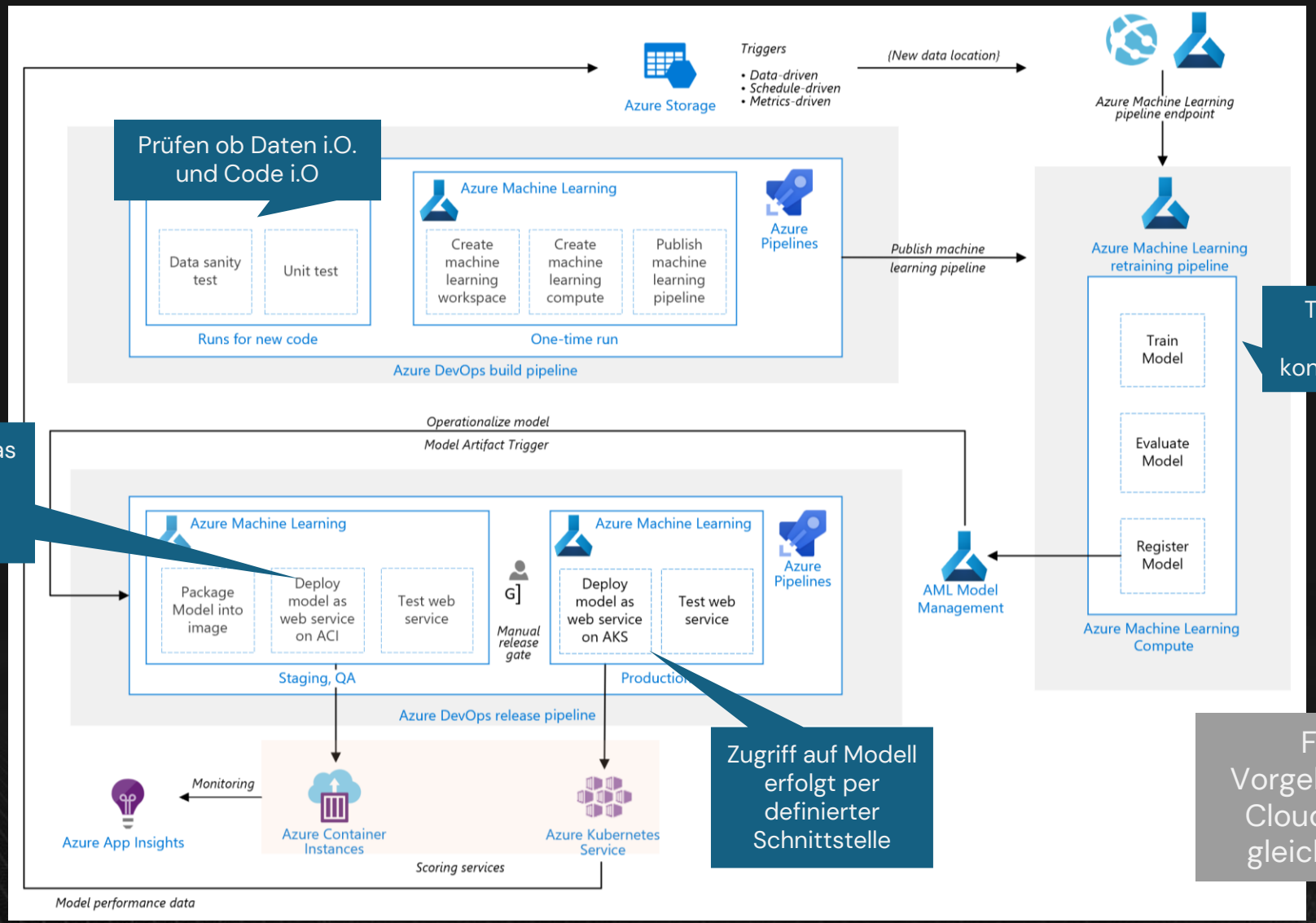
- Nach anfänglichem Training werden vor allem die Rechenbedarfe nicht mehr kontinuierlich in dem Ausmaß benötigt.

➔ Große wirtschaftliche Vorteile bei Nutzung von Cloud-Anbietern statt Aufbau eigener Strukturen (on-premise).

➔ Auch Startups können ohne große Investitionen „gleiche“ Ressourcen wie große Firmen nutzen.

Quellen: 1 Banko, Brill: Scaling to very very large corpora for natural language disambiguation, 2001.

Machine Learning in the cloud



Fallbeispiel Azure, Vorgehensweise bei Google Cloud und AWS prinzipiell gleich, nur andere Namen

Quelle: Link

Machine Learning in the cloud

Ausblick

- Starker Wettbewerbsdruck zwischen Amazon Web Services (erster Anbieter!), Microsoft Azure und Google Cloud
- Etablierte IT-Anbieter werden in das Cloud-Geschäft einsteigen: SAP, Oracle, ….
- Dieser steigende Wettbewerb führt zu:
  - Sinkenden Preisen
  - Firmen: Unterstützungsleistungen bei Einführung der Cloud
  - Privatanwender: freie Kontingente für Ausprobieren (200$ Azure, 300$ Google)
  - Allgemein: Entwicklung von automatisierten ML-Ansätzen (AutoML), Bereitstellen von Standardlösungen, Tutorials, …

In dieser Vorlesung haben Sie die Grundlagen gelernt; schauen Sie sich doch (bei Interesse ;-)) die Tutorials an!

Quellen: 1 Banko, Brill: Scaling to very very large corpora for natural language disambiguation, 2001.

Reinforcement Learning

Übersicht Machine Learning Verfahren

Unsupervised Learning	Supervised Learning	Reinforcement Learning
Lernen **ohne** vorher definierte **Zielwerte** oder Belohnung	Algorithmus lernt eine Funktion, die Eingabegrößen auf **vorher** definierte Outputs mappt.	Agent/ Algorithmus lernt **selbständig** mit Ziel, eine Belohnung zu **maximieren**.

Reinforcement Learning

Vereinfachte Darstellung

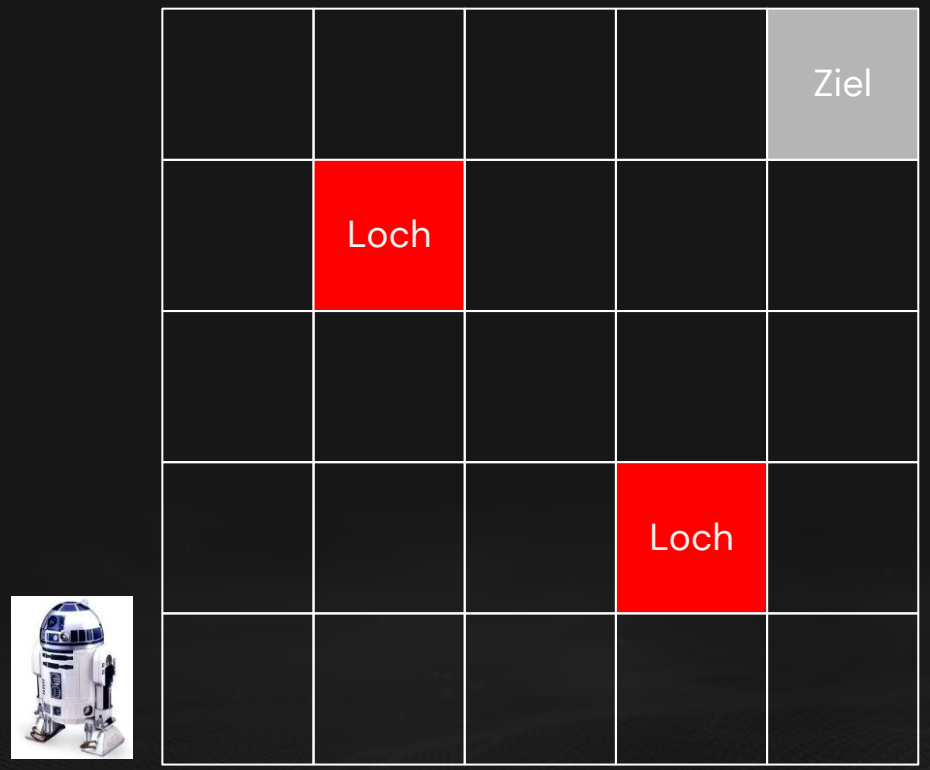
Unterschiede zu bisherigen Verfahren:

- Algorithmus agiert, um Belohnung zu maximieren und nimmt somit Einfluß auf seine Umgebung (und damit auf die nachfolgenden Daten).
- Agent weiß nicht (immer), wie die Umwelt auf seine Aktionen reagiert.
- Diese Belohnung erfolgt nicht zwingend per direktem Feedback, sondern auch ggf. später.
- Deshalb immer zeitabhängige, sequentielle Daten.
- Kann zur Laufzeit auch vorher nicht gelernte Sachen lernen (supervised Models sind statisch!).

Quellen: CS231n: Convolutional Neural Networks for Visual Recognition, Link
Introduction to RL with David Silver, Link

Reinforcement Learning

Fallbeispiel Wegfinden Roboter.



Ziel: Gegeben eine zufällige Startposition des Roboters, finde das Ziel mit der geringsten Anzahl von Schritten.

Reinforcement Learning

Formale Grundlage sind Markov Decision Processes (MDP)

Eigenschaften:

- S = Menge aller möglichen Zustände/ States s
- A = Menge aller Aktionen,
- T(s, a, s') = Übergangsfunktion von einem Zustand s in nächsten s' durch Aktion a. Diese wird in Prozent angegeben, da Unsicherheit (keine vollständige Sicht auf die Umwelt!!)
- R(s, a, s') = Belohnung für diesen o.a. Übergang.
  Da wir mehrere Schritte haben können, gilt:

$$G_t = R_{t+1} + \gamma R_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

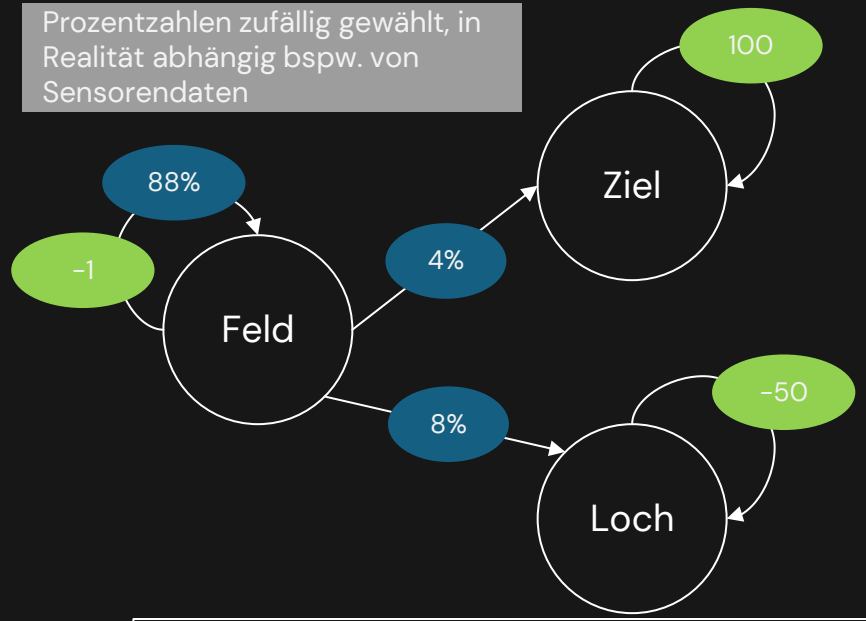
Aktuelle Belohnung

Anschließende Belohnung inkl. Gewichtungsfaktor Wichtigkeit zukünftiger Schritte γ (1:= sehr, 0:= gar nicht)

- V(s) = Wert eines Status. $V(s) = \mathbb{E}[G|S_t = s]$

Gemittelte/ erwartete Belohnung im Zustand s

Prozentzahlen zufällig gewählt, in Realität abhängig bspw. von Sensorendaten



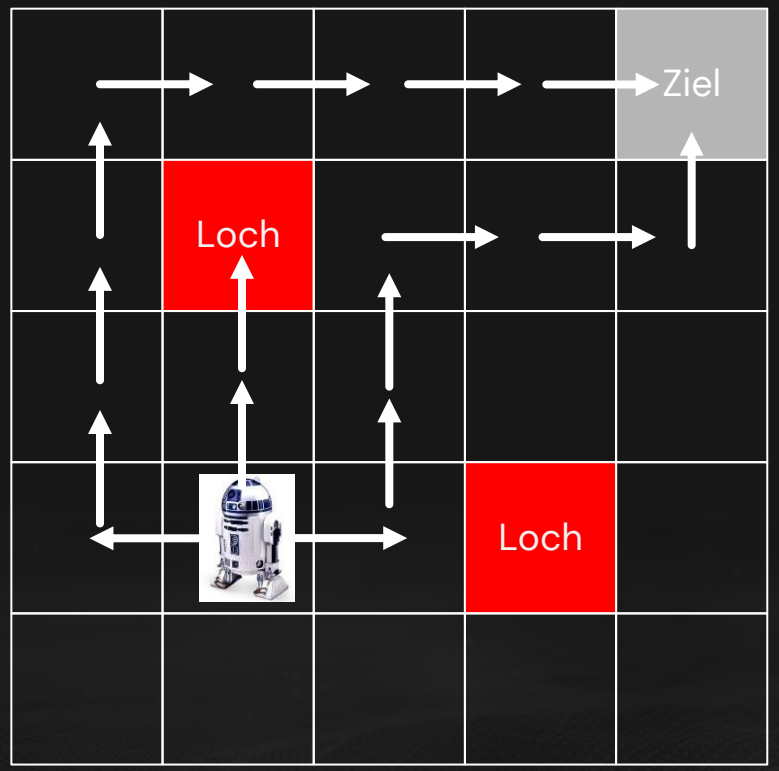
Fürs Fallbeispiel:
Anzahl Felder = 5x5 = 25
Zustände S:={Feld, Ziel, Loch}
Aktionen A: ={←, ↑,→,↓}
Übergangsfunktion T(Feld, *, Ziel) = 1/25 = 4%
Reward R(Feld, *, Ziel) = 100,
Reward R(Feld, *, Loch) = -50
Reward R(Feld, *, Feld) = -1 (Ziel: schneller Weg!)
V(Feld) = -1*0,88, V(Ziel) = 100*0,04 + -1*0,88

Vereinfachte Annahme bei Markov-Prozessen: nur aktuelle Zustand ist relevant für Zukunft, nicht die vorherigen!

Reinforcement Learning

Fallbeispiel Wegfinden Roboter.

Welcher der 3 Wege ist der Beste?



Ziel: Finde das Ziel mit der geringsten Anzahl von Schritten

Quellen: CS231n: Convolutional Neural Networks for Visual Recognition, Link
Introduction to RL with David Silver, Link

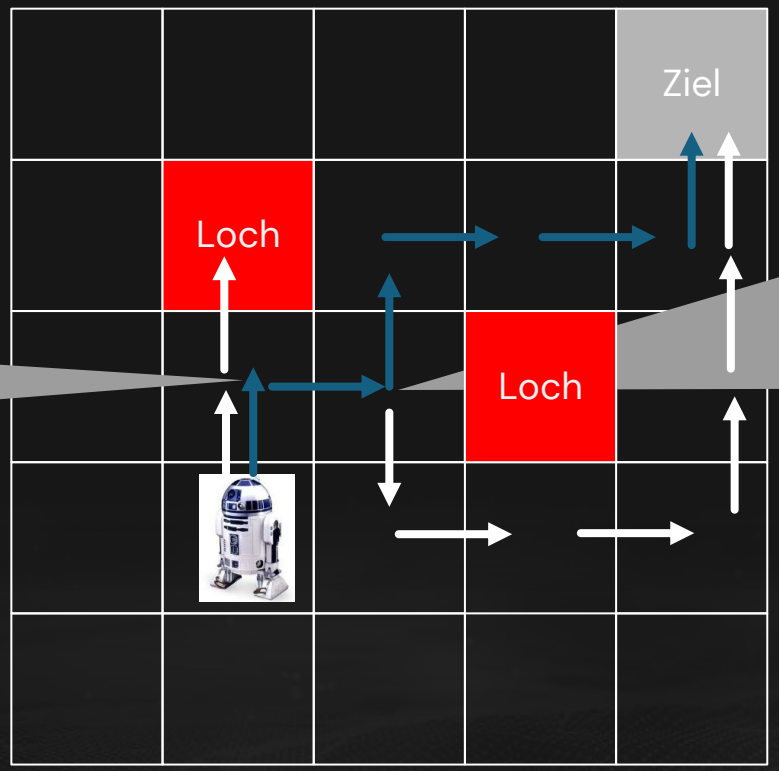
Reinforcement Learning

Fallbeispiel Wegfinden Roboter.

Wie würde ein Mensch das lösen?
Stets prüfen, was die beste Aktion ist-

Aber wie trifft und lernt eine Maschine solche Eigenschaften?

Entscheidung RL Agent:
Weg 1: führt ins Loch, R = -50
Weg 2: kein Loch



Entscheidung RL Agent:
Blaue Weg: in weniger Schritten zum Ziel
Weißer Weg: mehr Schritte zum Ziel.

Wir hatten definiert, daß Schritte die nicht direkt zum Ziel führen, eine Belohnung von -1 erhalten.

Dadurch erhält der blaue, kürzere Weg eine höhere Belohnung als der schwarze und wird deshalb gewählt.

Quellen: CS231n: Convolutional Neural Networks for Visual Recognition, Link
Introduction to RL with David Silver, Link

Reinforcement Learning

Fundamentale Problem: Exploration vs. Exploitation

Exploration:
- Initiale Umgebung & Auswirkungen auf diese unbekannt
- Umwelt kann sich über Zeit ändern
- Agent muß explorieren, um Umwelt kennenzulernen. Das heißt, er führt eine Aktion aus, erfaßt geänderte Umwelt und die Belohnung und speichert das ab.

Exploitation
- Ziel des Agenten ist Maximieren Belohnung.
- Agent führt also die Aktion bzw. Sequenz von Aktionen aus, die ihm am meisten Belohnung zurückgibt.

Dilemma: Wenn der Agent nur Exploitation wählt, lernt er nie, ob es nicht bessere Aktionen gibt.
Wenn der Agent nur exploriert, wird der Nutzen nicht maximiert und viele schlechte Aktionen ausgeführt.
➔ (langfristiger) Ausgleich notwendig.

Falls Sie immer in die gleichen Bars, Clubs, Restaurants gehen, lernen Sie nie besseres kennen.
Andererseits vermeiden Sie so Reinfälle.
Aber irgendwie müssen Sie die für Sie besten Läden ja auch mal kennengelernt haben....

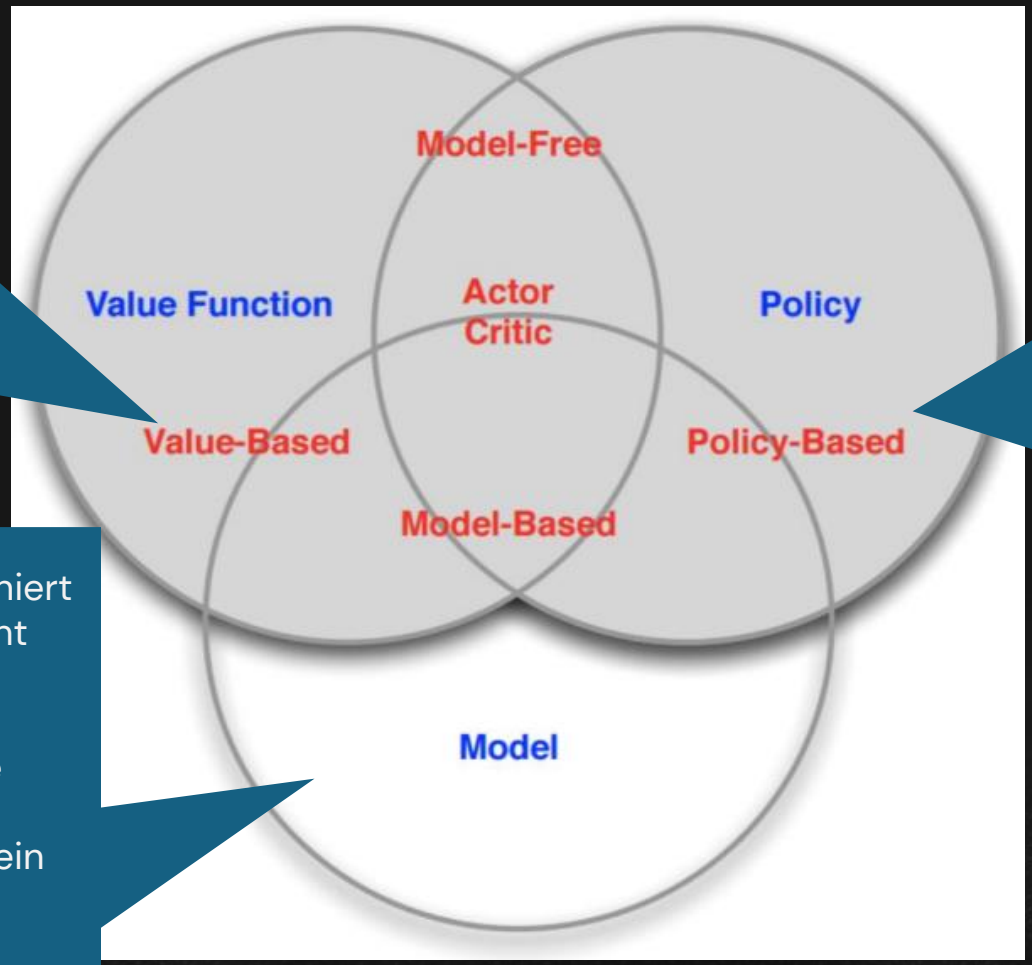
Lösung Ausgleich Exploration vs. Exploitation durch Epsilon-Greedy[1]-Verfahren:
- Initiale Definition eines Wertes ε, bspw. 0,1.
- Es wird vor jeder Aktion eine Zufallszahl p berechnet.
- Falls $p < \varepsilon$ dann beliebige Aktion ausführen, sonst beste Aktion (mit Wahrscheinlichkeit $1- \varepsilon$).
- In der Praxis starten wir mit einem höheren ε-Wert, der dann kontinuierlich verringert wird.

Quelle: Sutton, R & Barto, A.: "Reinforcement learning: an introduction", 1998.

Reinforcement Learning

Übersicht Entscheidungsverfahren

- Iterative Berechnung der Value-Funktion mit höchstem Wert, dann indirektes Ableiten der Entscheidung für Aktion (Policy) aus dieser Value-Fkt. durch Wahl des Status mit höchstem Wert
- Vorteil: geringer Speicherbedarf, keine Tabelle notwendig.
- Nachteil: Aktionen nicht direkt ableitbar.

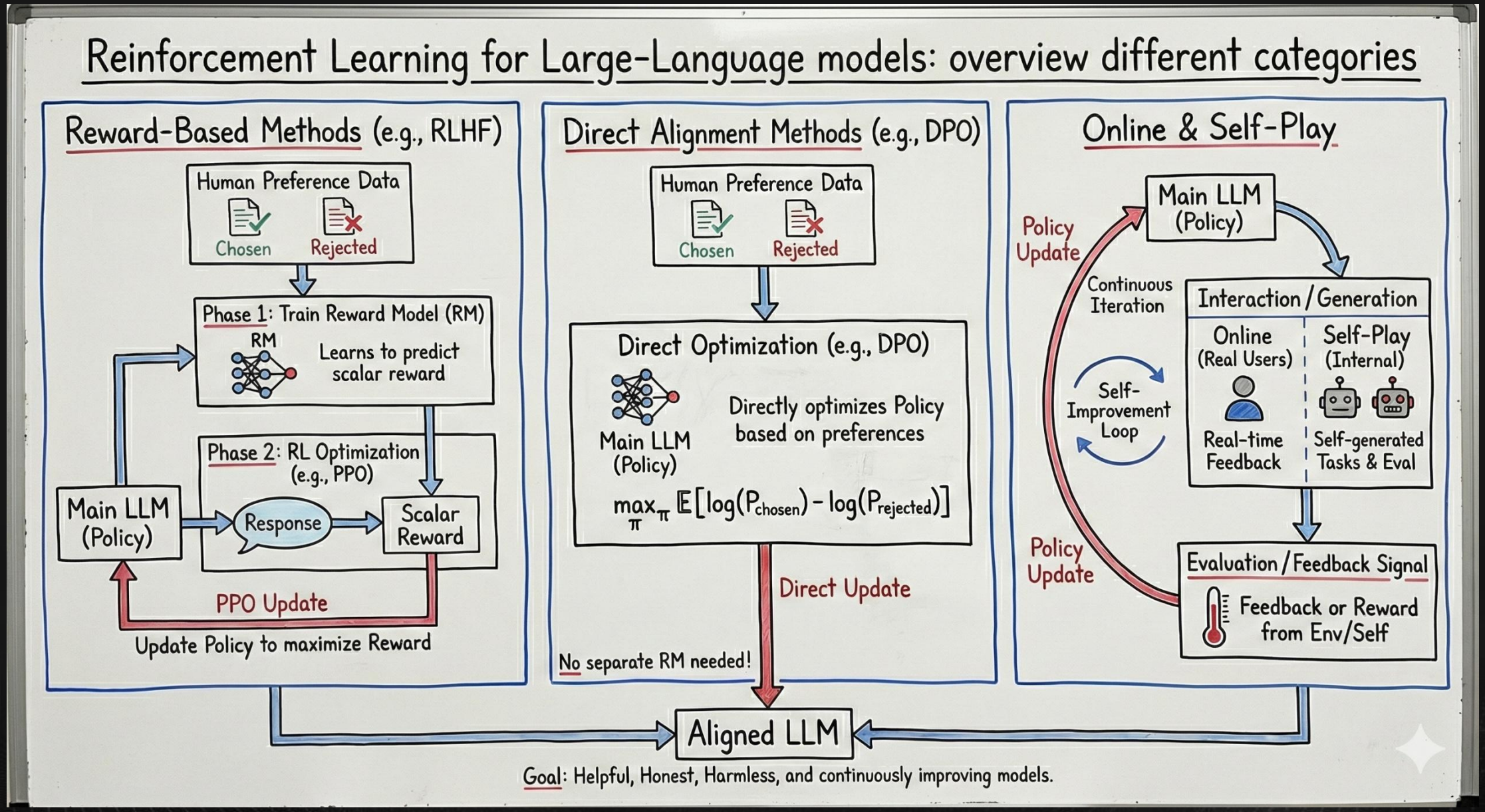


- Algorithmus lernt Tabelle mit Zusammenhang Status s und Aktion a.
- In jedem Zustand s wird aus der Tabelle die Aktion a gewählt, die die höchste Belohnung verspricht
- Vorteil: Aktionen direkt ableitbar
- Nachteil: sehr hoher Speicherbedarf für Tabelle bei komplexen Problemen

- Agent lernt Modell, wie Umwelt funktioniert basierend auf seinen Aktionen und plant darauf basierend seine Aktionen.
- Vorteil: kann auch mit sehr komplexen Umgebungen umgeben und lernt diese schneller.
- Nachteil: Modell kann schwer lernbar sein (und dann ist simples Policy-Based einfacher).

Detaillierung im folgenden

Quellen: Introduction to RL with David Silver, Link



Prompt für Google Gemini: "create an image of a whiteboard that summarizes how reinforcement learning is used for large-language models. show Reward-Based Methods, Direct Alignment Methods, and Online & Self-Play. Use diagrams, arrows, colors, and captions to explain the core idea. Name the header "Reinforcement Learning for Large-Language models: overview different categories

Reinforcement Learning für Large-Language Modelle

Übersicht Ansätze

Kategorie	Grundlegende Philosophie	Wichtige Algorithmen	Am besten geeignet für
Belohnungsbasiert *(Reward-Based)*	Trainiert separates **Belohnungsmodell (Reward Model)**, um Ausgaben zu bewerten, und nutzt dann RL, um diesen Score zu maximieren.	**PPO**, **GRPO**, **ReMax**	Komplexes logisches Denken (Mathematik/ Code), Vermeidung von "Reward Hacking", offene Textgenerierung.
Direkte Ausrichtung *(Direct Alignment)*	Überspringt explizites Belohnungsmodell. Optimiert "Policy" (das Modell) direkt auf Basis eines statischen Datensatzes, um Präferenzen zu erfüllen.	**DPO**, **IPO**, **KTO**	Allgemeine Chats, Befolgung von Anweisungen (Instruction Following), einfachere Pipelines, geringerer Speicherbedarf.
Iterativ / Self-Play	Modell generiert seine eigenen Daten, die gefiltert/bewertet und für das erneute Training genutzt werden (Schwungrad-Effekt).	**SPIN**, **STaR**, **Online DPO**	"System 2"-Denken (langsames, logisches Denken), Selbstkorrektur, Überwindung der Grenzen menschlicher Trainingsdaten.

Prompt für Google Gemini: “show an overview of reinforcement learning approaches for Large-language models”

Reinforcement Learning

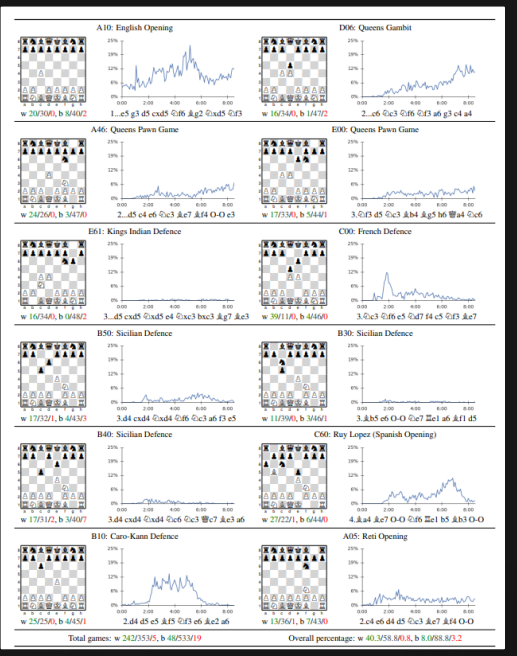
Fallbeispiele

Mnih et al.: “Playing Atari with Deep Reinforcement Learning”, 2013.

→ Erster Einsatz von Deep Learning Verfahren für Lernen des Modells (aus Sensorendaten).

Silver et al.: “Mastering the game of Go with Deep Neural Networks & Tree Search”, 2016.

→ Go galt aufgrund seines gigantischen Zustandsraums als nicht lernbar für Computer

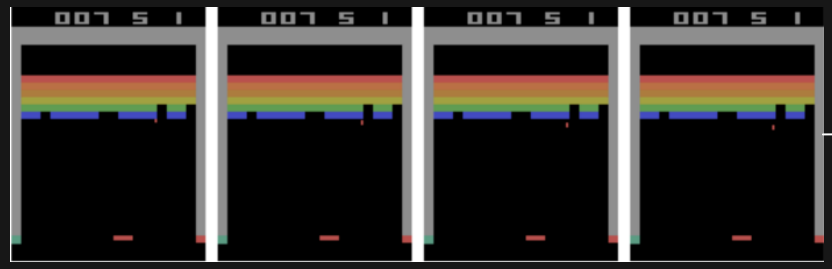


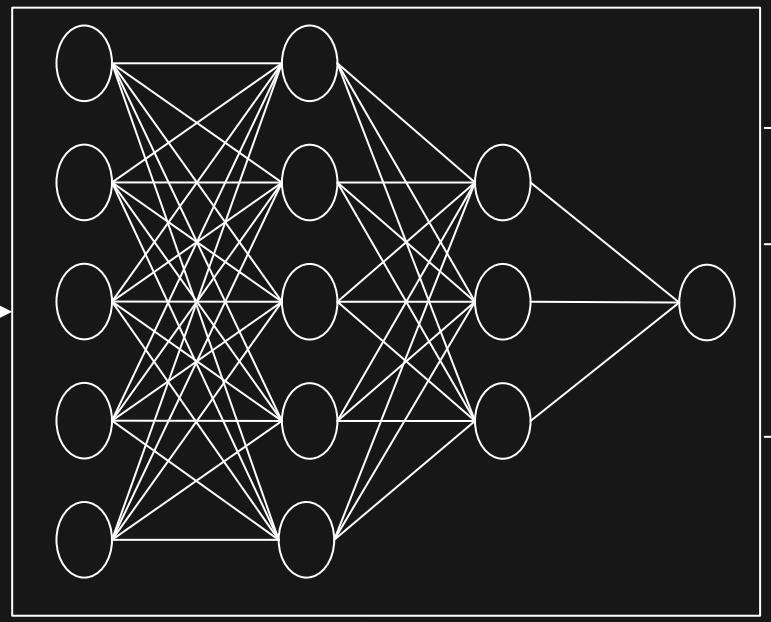
Silver et al: “Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm”, 2017.

→ Weiterentwicklung Go-Ansatz, Modell lernt komplett eigenständig.

Reinforcement Learning

Wie funktioniert model-based Reinforcement Learning?





- Q-value Action 1
- Q-value Action 2
- ...
- Q-value Action n

Wir nehmen 4 sequentielle Frames, um Zeitverlauf abzubilden

CNN reduziert Bilder auf ihre wesentlichen Inhalte (wo ist Ball? Wo Schläger? ...)

Exploitation: berechne alle möglichen State-Aktionen-Paare
Exploration: wähle das vorhandene State-Action Paar mit höchstem Wert

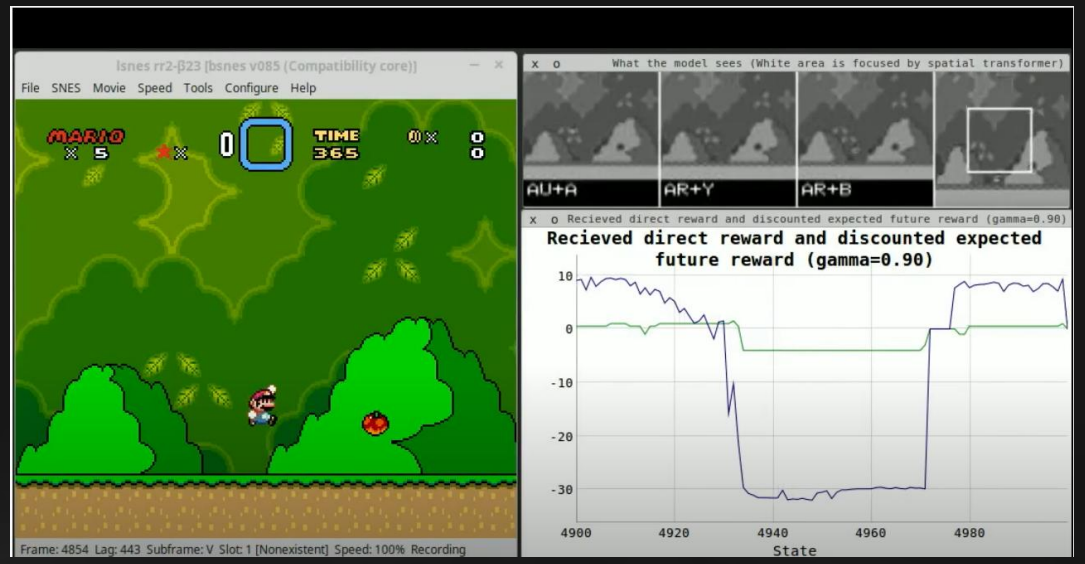
Ergebnisse Bellman Gleichung werden in Datenbank gespeichert als Kombination (Zustand, ergriffene Aktion, Belohnung, Nachfolgezustand)

Bellman-Gleichung $Q(s_t, a_t) \leftarrow (1 - \alpha) * \underbrace{Q(s_t, a_t)}_{alter\ Wert} + \alpha * (r_t + \gamma * \overbrace{\underbrace{max_a Q(s_{t+1}, a))}_{gesch\ddot{a}tzter\ zuk\ddot{u}nftiger\ optimaler\ Wert}}^{Gelernter\ Wert}$

Quelle: basierend auf Link

Reinforcement Learning

Live demos



https://www.youtube.com/watch?v=V1eYniJORnk

https://www.youtube.com/watch?v=L4KBBAwF_bE

Literatur und Quellen

Künstliche Intelligenz:

- Russel, Norvig: Artificial Intelligence – a modern approach
- Lapan: Deep Reinforcement Learning Hands-on.
- Silver: Introduction to Reinforcement Learning (Link)
- Barto, Sutton: Reinforcement Learning (Link)

Online-Kurse (bei Interesse):

- Kostenfrei: Udacity: Reinforcement Learning, Link
- Coursera: Reinforcement Learning, Link
- Udacity: Deep Reinforcement Learning, Link

Und was jetzt?

Falls Sie Ideen oder Anregungen von anderen Machine Learning Projekten möchten:
https://github.com/lukasmasuch/best-of-ml-python

Backup

Reward-based models

Reinforcement Learning from Human Feedback

Wird von quasi allen Modellen verwendet

Phase	Action	Examples	Analogy
1. Supervised Fine-Tuning (SFT): *The Pre-requisite*	Before RL, we teach the raw model to behave like a chatbot. Humans write high-quality prompt-response pairs, and the model mimics them.	**Math:** Human writes: *"Calculate 2x2."* → *"Answer is 4."* Model learns to answer, not just complete the text. **Creative:** Human writes: *"Write a haiku about rain."* → *"Soft drops fall from grey / washing the world clean today / life drinks from the sky."*	**Boot Camp:** New recruits are drilled on the basics. They don't know *why* they do things yet, they just copy exactly what the drill sergeant does.
2. Reward Modeling (RM): *The "Judge"*	The SFT model generates multiple responses. Humans rank them (A > B). We train a **Reward Model** to predict these rankings (assign a scalar score).	**Math:** • Response A: "2+2=5" (Score: 0.1) • Response B: "2+2=4" (Score: 0.9) RM learns: Correct math = High Score. **Creative:** • Response A: "Rain is wet." (Score: 0.2 - Boring) • B: "Tears of the sky..." (Score: 0.8 - Poetic) RM learns: Poetic language = High Score.	**Training a Food Critic:** Instead of teaching a chef how to cook, we teach a critic how to *taste*. The critic learns that "burnt toast" is bad and "soufflé" is good by looking at human reviews.
3. PPO Optimization: *The Training*	The model (Policy) generates a response. The **Reward Model** gives it a score. **PPO** updates the model to maximize this score, but uses a constraint (KL Penalty) to keep it from changing too much.	**Math:** Model tries: "The sum is 4." RM gives reward: +0.9. PPO Update: Increase probability of saying "sum" and "4". **Creative:** Model tries: "The sky cries." RM gives reward: +0.8. PPO Update: Increase probability of metaphorical words. *Constraint:* Don't start speaking gibberish just to get points.	**The Dog & The Treat:** You tell the dog to "Sit." If it sits, you give a treat (Reward). The dog learns "Sitting = Treat." *KL Penalty:* You ensure the dog is still a dog and doesn't turn into a treat-seeking robot that forgets how to walk.
4. Evaluation: *The Test*	We test the new PPO-trained model against the original SFT model to see if humans (or GPT-4) prefer the new answers.	**Math:** Does the model solve problems more reliably than before? **Creative:** Are the poems more touching and less generic than the SFT version?	**The Taste Test** Serving the new dish to customers. If they like it more than the old recipe, the training was a success.

Gemini Prompt: build a table to explain RLHF with PPO. First column should be the phase. the second column should describe the action. The third column should contain an example with calculation and beneath one example for creative writing. The fourth column should have an analogy"

Zelikman et al., „Star: Bootstrapping reasoning with reasoning", Advances in Neural Information Processing Systems, 2022.

Direct alignment options

Direct Preference Optimization

Industriestandard aktuell

Phase	Action	Examples	Analogy
1. Preference Data Collection: *The Dataset*	We collect pairs of responses for a single prompt: a **Winner (y_w)** and a **Loser (y_l)**. No explicit score (like 7/10) is needed, just "A is better than B."	**Math:** Prompt: *Estimate 12 $\times$ 13.* • Winner (y_w): "156" • Loser (y_l): "150" **Creative:** Prompt: *Describe a villain.* • Winner (y_w): "His eyes were cold shards of ice." • Loser (y_l): "He was a very bad man with blue eyes."	**The Eye Exam** The doctor asks: *"Better 1... or 2?"* They don't ask you to score the clarity on a scale of 1 to 100. They just want to know which one wins.
2. DPO Training: *The Update*	The model looks at the pair. It calculates the probability of generating the Winner vs. the Loser. The **Loss Function** forces the probability of the Winner to go **UP** and the Loser to go **DOWN**, relative to the original model.	**Math (Calculation):** • *Reference Model* probability for Winner: 50% • *Reference Model* probability for Loser: 40% • **Goal:** Push Winner > 50% and Loser < 40%. • *Result:* New Model gives Winner 80%, Loser 10%. **Creative:** • The model realizes "shards of ice" (Winner) is preferred over "bad man" (Loser). • It adjusts weights to make "shards" more likely next time.	**Sibling Rivalry** A parent tells a child: *"Look at your brother. See how he eats his vegetables? Do that. See how he throws food? Don't do that."* There is no scorecard. Just "Be more like the Winner, be less like the Loser."
3. Implicit Regularization: *The Safety Net*	DPO mathematically includes the "KL Penalty" inside its own formula. It ensures the model prefers the winner *without* drifting too far from the original reference model's style.	**Math:** Even though "156" is the winner, the model is prevented from shouting *"156 156 156!!!"* just to be sure. It must stay close to the reference grammar. **Creative:** It learns to write the villain description vividly, but still using standard English sentence structures defined by the base model.	**The Rubber Band** You are running toward the prize (the Winning answer), but you are tied to a pole (the Reference Model) by a rubber band. You can stretch toward the win, but the rubber band prevents you from running off the cliff into nonsense.

Online/ Self-play

STaR (Self-Taught Reasoner)

Wird von Deep Seek R1, Open AI o1, xAI verwendet

Phase	Action	Examples	Analogy
1. Generation: Exploration	Model is given a prompt but **no solution**. It generates multiple different "Chain of Thought" attempts (rationales) to solve it.	**Math:** Prompt: *Solve 3 + 4*2* • *Attempt A:* "Add 3+4=7, then *2. Answer: 14" • *Attempt B:* "Multiply 4*2=8, then +3. Answer: 11" **Creative:** Prompt: *Write sentence without 'e'.* • *Attempt A:* "The apple is red." • *Attempt B:* "A dark bird flies high."	**Brainstorming:** Throwing spaghetti at the wall to see what sticks, without knowing yet which recipe is correct.
2. Filtering: The "Judge"	We verify the **final output** against a Ground Truth (for math) or a Constraint Checker (for creative). We discard the failures and keep the successes.	**Math:** • Check: Is the answer 11? • *Result:* Attempt A (14) is **Rejected**. Attempt B (11) is **Accepted**. **Creative:** • Check: Does string contain 'e'? • *Result:* Attempt A ("appe") is **Rejected**. Attempt B (No 'e') is **Accepted**.	**The Golden Ticket:** A teacher grading a test. They only look at the final answer. If it's correct, they keep the paper; if it's wrong, they throw it in the trash.
3. Fine-Tuning: The Learning	The model is trained (fine-tuned) on the **Accepted** attempts. It treats its own generated reasoning as if it were a human-written textbook example.	**Math:** The model updates its weights to memorize: *"To solve $3+4 \times 2$, I should prioritize multiplication..."* **Creative:** The model updates its weights to understand: *"When asked to avoid 'e', I should select words like 'dark', 'bird', 'high'."*	**Study Notes:** A student realizing they got a question right by luck, so they write down the steps they took to ensure they remember the logic for next time.
4. Iteration: The Loop	The model—now slightly smarter—runs the process again on **harder** problems or problems it previously failed.	**Math:** Now attempting: *Solve (3+4) * (2+1)* (Requires brackets + order of operations). **Creative:** Now attempting: *Write a whole paragraph without the letter 'e'.*	**Leveling Up:** You beat the Level 1 boss, gained experience points (XP), and now you use your new skills to fight the Level 2 boss.

Gemini Prompt: „build a table to explain STaR (Self-Taught Reasoner). First column should be the phase. the second column should describe the action. The third column should contain an example with calculation and beneath one example for creative writing. The fourth column should have an analogy"

Zelikman et al., „Star: Bootstrapping reasoning with reasoning", Advances in Neural Information Processing Systems, 2022.