Digital Applications & Data Management

WS25/26

Dr. Jens Kohl

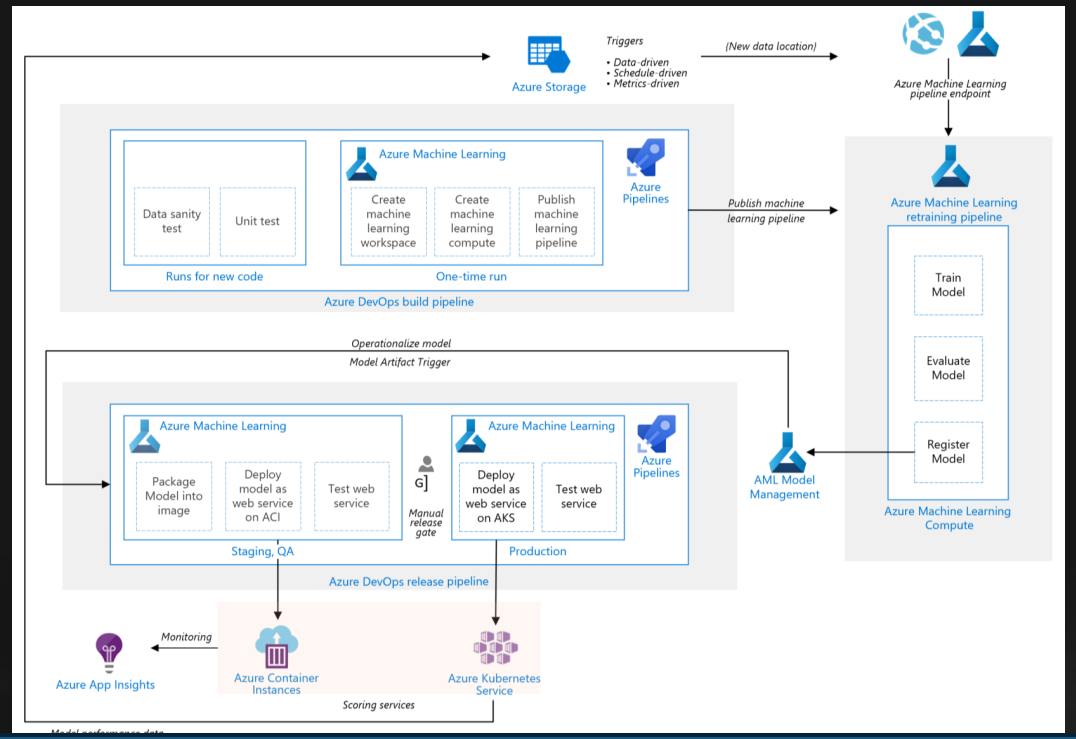
Roadmap Vorlesung

1. Einführung und Übersicht
2. Grundlagen Data Science
3. Vorgehen Data Science Use Case
4. Case Study Data Science
5. Grundlagen unüberwachtes Lernen
6. Grundlagen überwachtes Lernen (tabellarische Daten)
7. Case Study überwachtes Lernen (tabellarische Daten)
8. Grundlagen überwachtes Lernen (Bilddaten)
9. Case Study überwachtes Lernen und Transfer Learning (Bilddaten)
10. Grundlagen Generative AI
11. Generative AI mit Texten und Prompt Engineering
12. Agentic AI
13. Ausblick: Machine Learning in der Cloud und Reinforcement Learning

Vorlesung 13:
Ausblick – Machine Learning in der Cloud und Reinforcement Learning

Was machen wir heute?

Motivation



Beschleunigung des Trainings sehr großer Machine Learning Modelle durch Nutzen Cloud

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

Machine Learning in the cloud

Machine Learning in the cloud

Einleitung

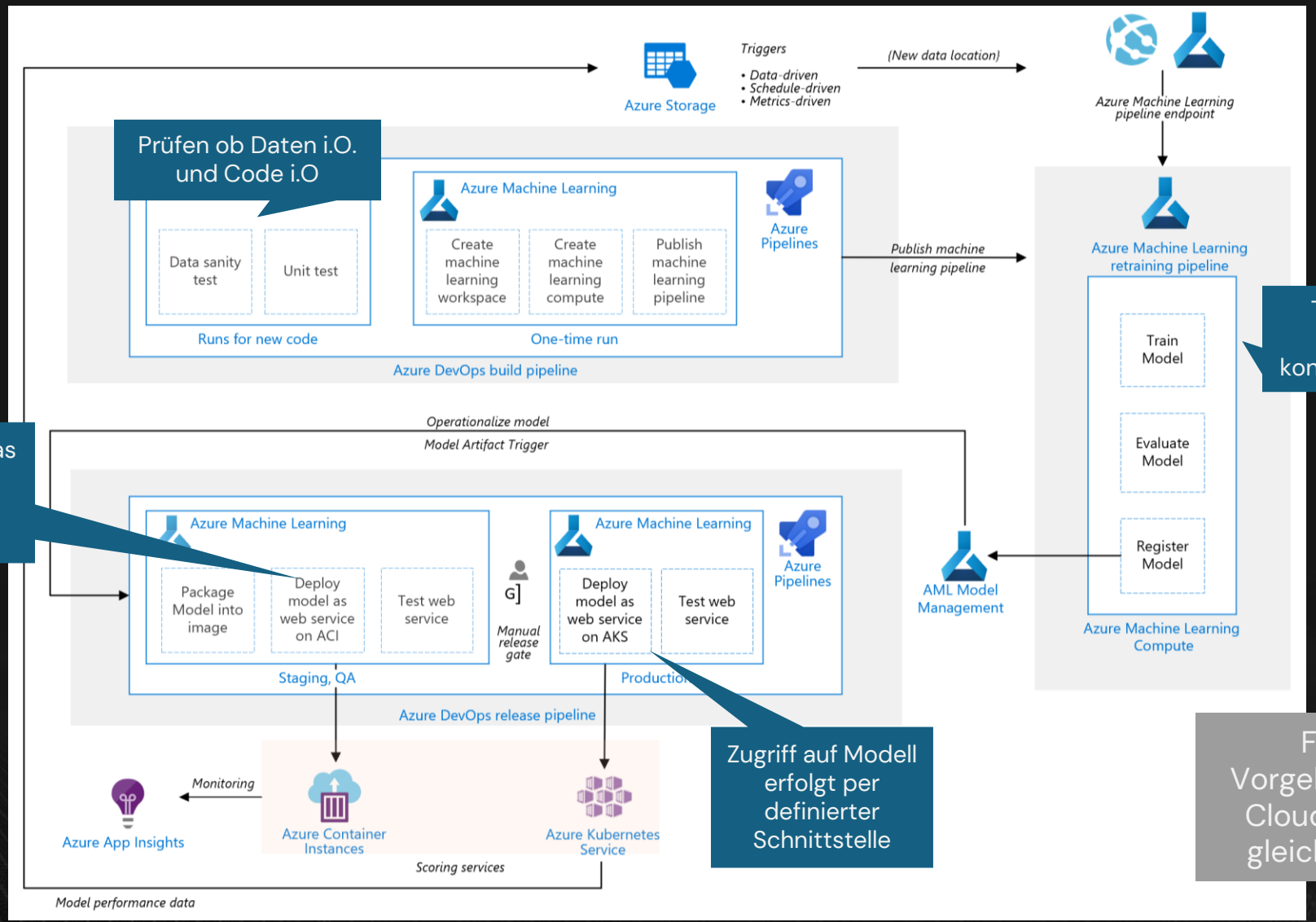
- Bis jetzt haben wir vorwiegend auf einem Rechner (lokal/ Cloud) unsere Use Cases umgesetzt.
- Wir haben gesehen, daß Machine Learning sehr ressourcen- und zeitintensiv ist:
  - Speicherbedarf: je mehr Daten, desto bessere Ergebnisse (2- oder 3-fache Datenmenge besser als jedes Parameter-Tunen[1])
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

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

➔ Auch Startups können ohne große Investitionen „gleiche“ Ressourcen wie finanzstarke, 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

Machine Learning in the cloud

Ausblick

- Starker Wettbewerbsdruck zwischen Amazon Web Services (erster Anbieter!), Microsoft Azure und Google Cloud für Europa und EMEA.
- Etablierte IT-Anbieter bauen Cloud-Geschäft aus: SAP, Oracle, ….
- Dieser steigende Wettbewerb führt zu:
  - Sinkenden Preisen für Nutzung der Cloud
  - Firmen: umfangreicher Support durch Cloud-Anbieter f. Einführung, Migration, Betrieb.
  - 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:

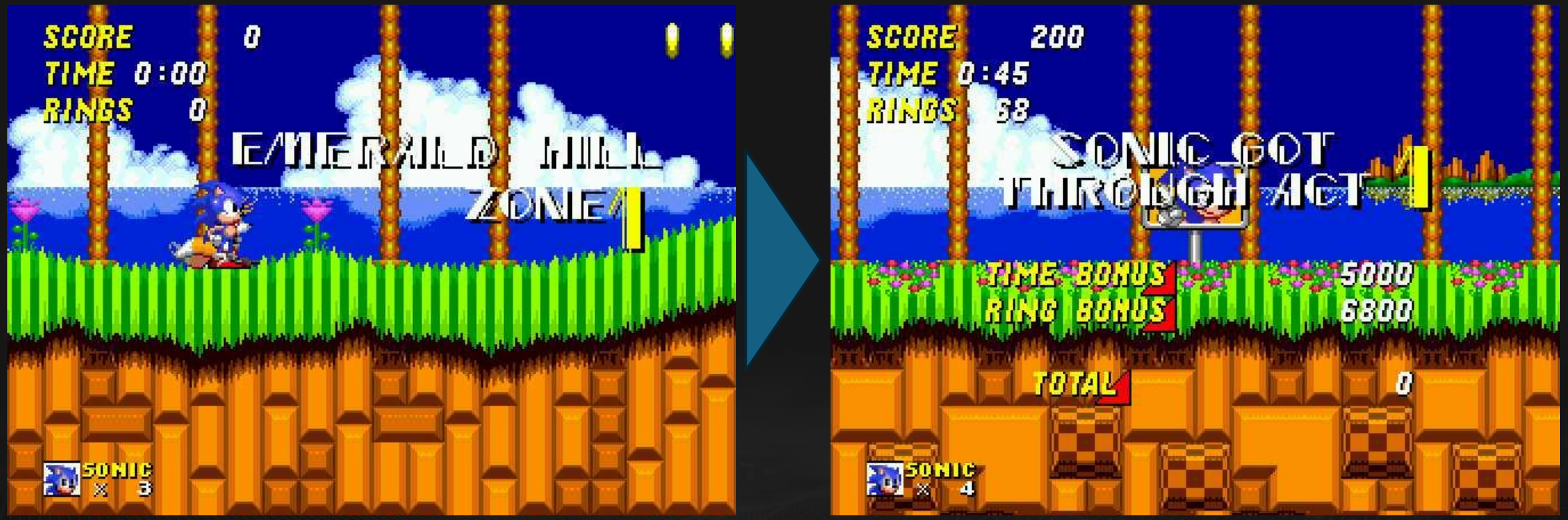
- Algorithmus agiert, um Belohnung zu maximieren und nimmt somit Einfluß auf seine Umgebung (und damit auf die nachfolgenden Daten).
- Agent weiß nicht (immer), wie die Umwelt auf seine Aktionen reagiert.
- Diese Belohnung erfolgt nicht zwingend per direktem Feedback, sondern auch ggf. später.
- Deshalb 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

Use Case Reinforcement Learning mit Spielen

Use Case Reinforcement Learning mit Spielen

„Kann eine AI lernen, ein Spiel eigenständig zu spielen?“



Use Case Reinforcement Learning mit Sonic 2

Wieso ist das schwierig?

Viele Pixel und Bildinformationen

Verschiedene Wege auf verschiedenen Ebenen

„Game physics“ und Speed

Hindernisse und Gegner, auch in der Höhe!

Verschiedene Aktionen

Sehr hoher Zustandsraum an möglichen Aktionen, und das Ergebnis einer Aktion sieht man oft nicht direkt

Use Case Reinforcement Learning mit Sonic 2

Wieso funktionieren die bisher in der Vorlesung kennengelernten Ansätze nicht?

Unsupervised Learning

Es gibt ein zu erreichendes Spielziel und Punkte

Supervised Learning

Verschiedene richtige Aktionen erschwert „Labeln“.
Beste Aktion ergibt sich oft nur langfristig.

Use Case Reinforcement Learning mit Sonic 2

Grundlegende Begriffe

Zustand: gesamtes Bild inkl. Sonics Position, Gegner, Ringe, Terrain, Punkte, …

Mögliche **Aktionen**:
links, rechts, Sprung, Dash, …

Belohnungsfunktion erlaubt Bewertung und Vergleich verschiedener Aktionen

Übergangsfunktion: Durch Auslösen einer Aktion erhalten wir einen neuen Zustand

Wähle in einem gegebenen Zustand die Aktion, die die höchste Belohnung gibt.
Um langfristiges Agieren zu ermöglichen, kann die Belohnung gewichtet/ **diskontiert** werden.

Use Case Reinforcement Learning mit Sonic 2

Herausforderung langfristiges statt kurzfristiges Handeln

Manchmal ist die korrekte Strategie, Aktionen zu ergreifen, die erst (deutlich) später einen Nutzen haben...

Reinforcement Learning

Fundamentale Problem des Reinforcement Learning: Exploitation vs. Exploration

Exploitation (Maximieren Belohnung)

- Agent wählt bekannte beste Aktion aus
- Kurzfristig beste Aktion muß aber nicht langfristig optimal sein.

Reine Exploitation: optimale Strategie nicht gefunden

Exploration (Ausprobieren)

- Initial sind Umgebung sowie Auswirkungen der Aktionen unbekannt
- Umgebung kann sich über Zeit ändern

- Agent muß neue Aktionen ausprobieren
- Das kann aber (kurzfristig) Belohnung verringern

Reine Exploration: Belohnung wird nicht maximiert

Balance notwendig!

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

Reinforcement Learning

Übersicht Verfahren

Value-based Learning:
Idee: lernt, wie „gut" Aktion in Zustand ist
Vorteile:
- Einfaches Konzept
- Funktioniert gut bei endlichen Aktionen (links, rechts, …)

Nachteile:
- Strategie wird nur indirekt gelernt
- Schwieriger bei kontinuierlichen Aktionen (z.b. Lenkwinkel Fahrzeug)

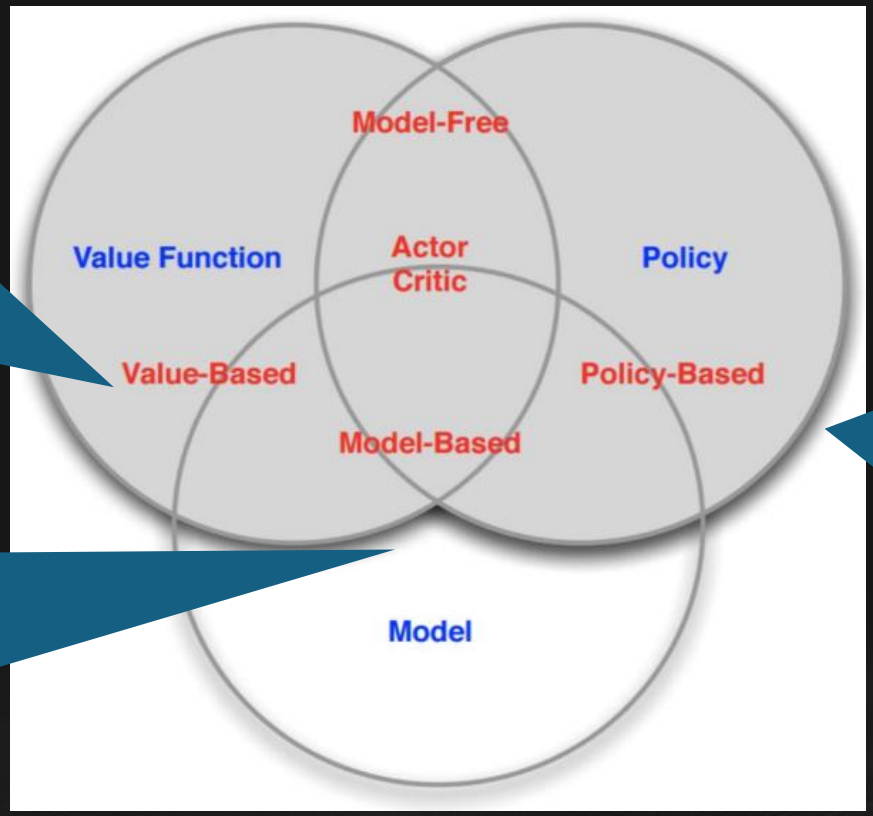
Bekannte Algorithmen:
Q-Learning[1], SARSA, Deep Q-Network (DQN)

Typische Use Cases: Atari-Spiele[2], Brettspiele mit diskreten Aktionen.

Model-based Learning:
Idee: lernt Modell der Umwelt
Vorteile:
- Kann vorausschauend planen
- Oft dateneffizient

Nachteile:
- Lernen des Modell oft sehr komplex. Fehler im Modell führen zu schlechten Entscheidungen

Bekannte Algorithmen:
AlphaZero, MuZero, …
Typische Use Cases:
Schach oder Go (AlphaZero/MuZero), Planungssysteme



Policy-based
Idee: lernt Strategie direkt.
Vorteile:
- Gut für kontinuierliche Aktionen (d.h. großer Wertebereich wie z.B. Lenkwinkel)
- Optimiert direkt langfristige Belohnung

Nachteile:
- Hohe Varianz der Gradienten, Training kann instabil sein

Bekannte Algorithmen:
REINFORCE[3], PPO[4]

Typische Use Cases: Autonome Fahrzeuge, Drohnen, Robotik.

Quellen: Bild entnommen aus Introduction to RL with David Silver, Link
1 Watkins, C. & Dayan, P.: "*Q-learning", 1992.*
2 Mnih et al.: „Human-level control through deep reinforcement learning", 2015.
3 Williams RJ: "Simple statistical gradient-following algorithms for connectionist reinforcement learning", 1992
4 Schulman, J. et al.: "Proximal policy optimization algorithms", 2017
5 Silver, D. et al.: "*Mastering the game of Go without human knowledge* (AlphaZero)", 2017
6 Schrittwieser, J et al.: "MuZero", 2020
für weitere Algorithmen siehe Sutton, R & Barto, A.: "Reinforcement learning: an introduction", 1998.

Use Case Reinforcement Learning mit Sonic 2

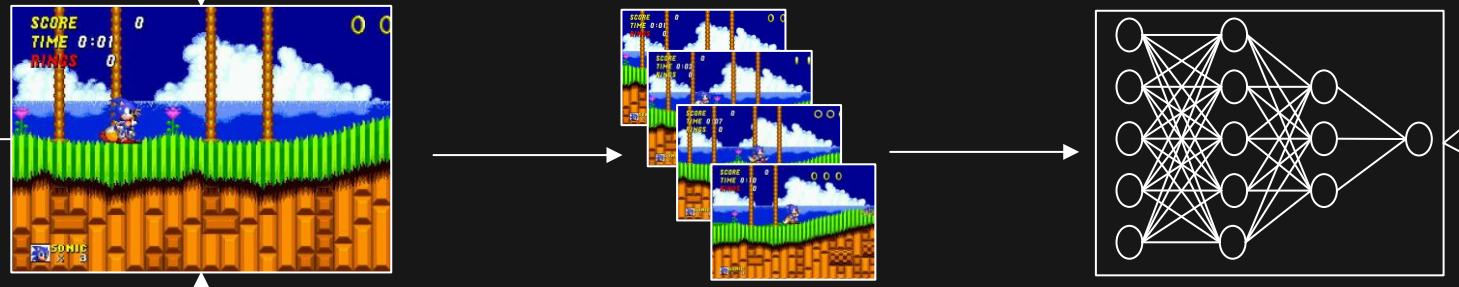
Schematischer Ablauf CNN mit Principal Policy Optimization (PPO)

Phase 1: Trainieren durch Spielen und Sammeln von Erfahrungen (für X Anzahl Schritte)

Aktion ausführen → neuer Zustand + Belohnung

Nutzen mehrerer Bilder um Bewegungen zu erkennen

CNN extrahiert relevante Merkmale



Schätzt langfristigen Wert des aktuellen Zustands

Berechnet Wahrscheinlichkeit je Aktion, wählt Aktion

Rückmeldung ausgewählte Aktion

Rollout-Buffer speichert pro Schritt: Zustand, gewählte Aktion, Belohnung, geschätzter Wert, Aktionswahrscheinlichkeiten, Level-Ende erreicht/ Tot?/aktiv

Phase 2: Lernen und Verbessern

PPO

- Liest gesammelte Daten aus Roll-out buffer
- Berechnet wie viel langfristige Belohnung tatsächlich erreicht wurde
- Prüft, ob eine Aktion besser oder schlechter war als erwartet
- Update Actor: gute Aktionen werden wahrscheinlicher, schlechte unwahrscheinlicher. **Änderungen werden begrenzt, damit Lernen stabil bleibt (ganz wichtig!)**
- Update Critic: lernt langfristigen Wert eines Zustands genauer vorherzusagen

Use Case Reinforcement Learning mit Sonic 2

Herleitung Belohnungsfunktion

- Ziel ist das Erreichen des Ende des Levels.
- Das Erzielen von Punkten ist ein nachgelagertes Ziel.
- Für das Erreichen des Ende des Levels muß er nach rechts laufen.
- Einen Gegnerkontakt überlebt Sonic nur wenn er Ringe hat, ansonsten ist er tot. Somit ist das Sammeln von Ringen gut.
- Wenn Sonic 3 Leben verliert, endet das Spiel.
- Um so schneller Sonic das Ende des Levels erreicht, umso mehr Punkte erhält er.

Wir versuchen die guten Ereignisse zu belohnen und die schlechten Ereignisse zu bestrafen.
Die Herausforderung ist, die Belohnungsfaktoren exakt zu definieren und untereinander zu gewichten.

Use Case Reinforcement Learning mit Sonic 2

Herausforderung Gewichtung der Belohnung untereinander

Höheres Gewichten Leben/ Ringe als Fortschritt

Problem: Agent lernt, daß Verlust von Ringe/ Leben durch Gegnerkontakt oder Hindernisse zu größerer negativer Belohnung als Stehenbleiben führt.

Bewegung nach rechts wird zu stark gewichtet

Problem: Agent lernt nicht, daß er für das Passieren vertikaler Objekte Geschwindigkeit benötigt, die er durch kurzes Zurücklaufen erhalten kann.

Use Case Reinforcement Learning mit Sonic 2

Herausforderung Gewichtung der Belohnung untereinander

Agent lernt, daß schnelles nach rechts laufen Belohnung gibt.
Problem:

- Agent hat Probleme, den Looping zu passieren.
- Deshalb läuft Agent nach kurzem Überlegen nach links und dann erneut nach rechts, um so Belohnung zu maximieren („Reward Hacking“).

Lösung: wir fügen eine neue Belohnung hinzu, falls der Agent unbekannte Gebiete erreicht.

Use Case Reinforcement Learning mit Sonic 2

Definieren (finale) Belohnungsfunktion

Pro Schritt erhält der Agent:

Reward r_t = $\omega_1(Progress)$

$+ \omega_2(Speed\ bonus_t)$

$+ \omega_3(FoundNewArea)$

$+\omega_4(Rings)$

$-\omega_5(LostLive\ _t)$

$+ \omega_6(LevelComplete_t)$

Geschwindigkeit in Schritt t höher/ niedriger als zuvor?

wurde im aktuellen Schritt t ein Leben verloren?

Der Agent erhält somit

- **Belohnung**: Fortschritt, Exploration, Ringe, Level-Ende gefunden
- **Bestrafung:** Leben verlieren

Die Gewichte ω bestimmen das Verhalten des Agenten.

Use Case Reinforcement Learning mit Sonic 2

Was lernt der Agent nun?

Maximiere die langfristige Gesamtbelohnung

$$\text{Total Reward} = \sum_{t=0}^{T} r_t$$

- Der Agent trifft in jedem Schritt eine Aktion
- Jede Aktion erzeugt eine unmittelbare Belohnung r_t

Der Agent lernt nun, wie er eine möglichst hohe Summe für Total Reward erhält.
WICHTIG: der Agent optimiert nicht einzelne Schritte, sondern die Gesamtsumme

Use Case Reinforcement Learning mit Sonic 2

How it all works out in the end

Source code verfügbar unter: https://github.com/JensKohl/Sonic2_ReinforcementLearning

Reinforcement learning ist langwieriger Prozeß und erfordert viel Rechenpower, Geduld und Feintuning

Reinforcement Learning

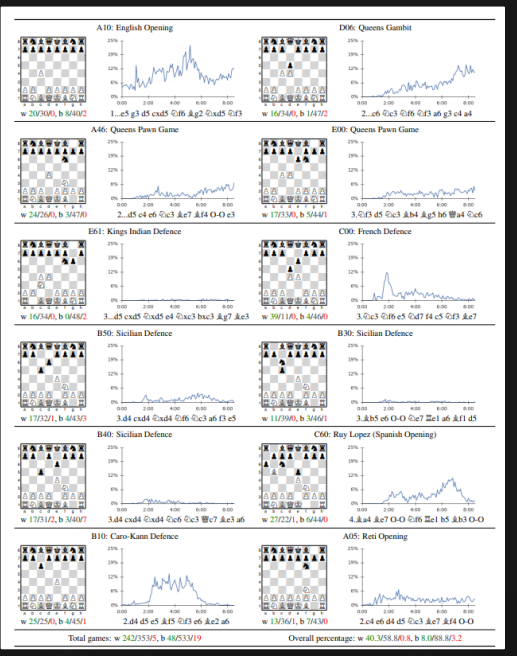
Weitere Fallbeispiele

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.

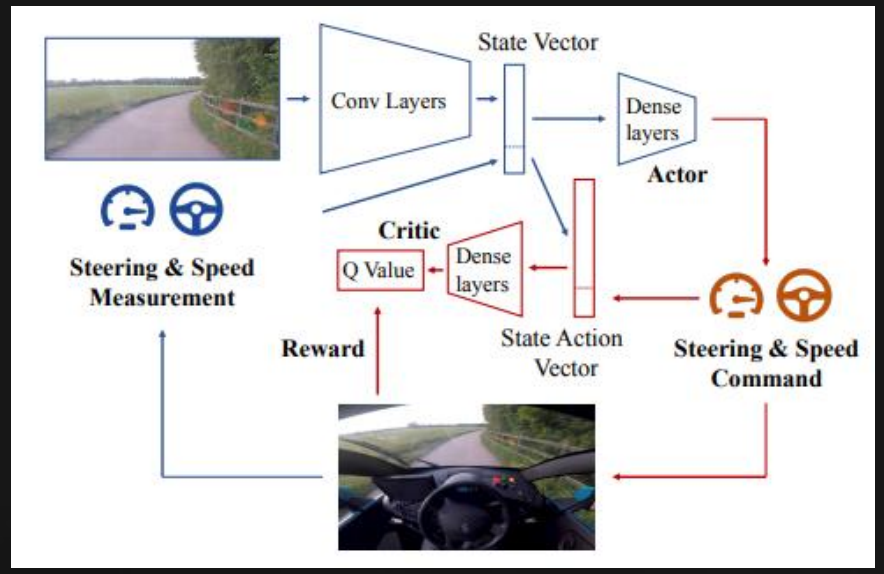
→ Go galt aufgrund seines gigantischen Zustandsraums als nicht lernbar für Computer



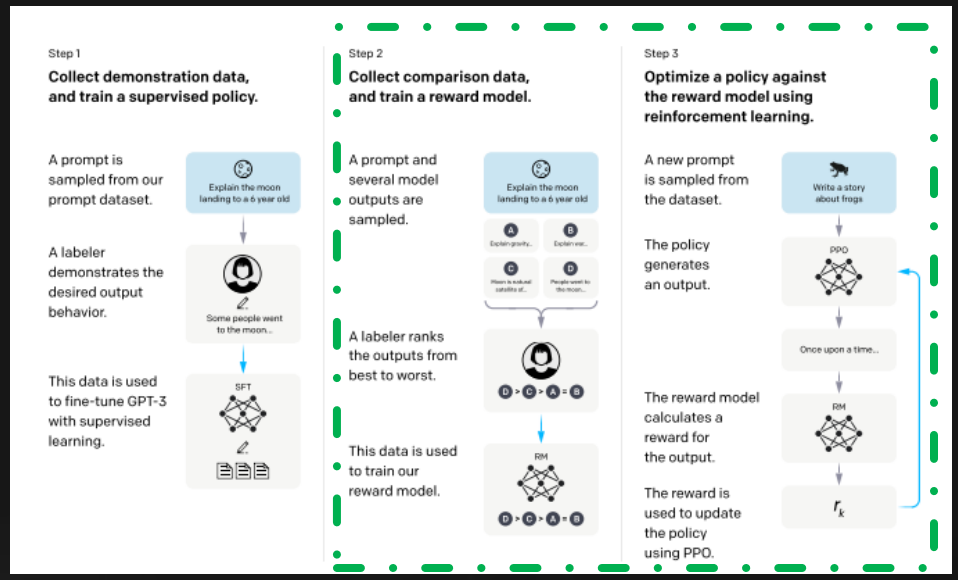
Silver et al: “Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm”, 2017.
→ Weiterentwicklung Go-Ansatz, Modell lernt komplett eigenständig.

Reinforcement Learning

Weitere Fallbeispiele (Teil 2)



Kendall et al.: „Learning to Drive in a Day", 2019
→ Modell lernte Spurfolgen in ~37 Minuten
Lu et al.: "Imitation Is Not Enough: Robustifying Imitation with Reinforcement Learning for Challenging Driving Scenarios", 2023.
→ RL für Verbesserung Sicherheit Autonomes Fahren



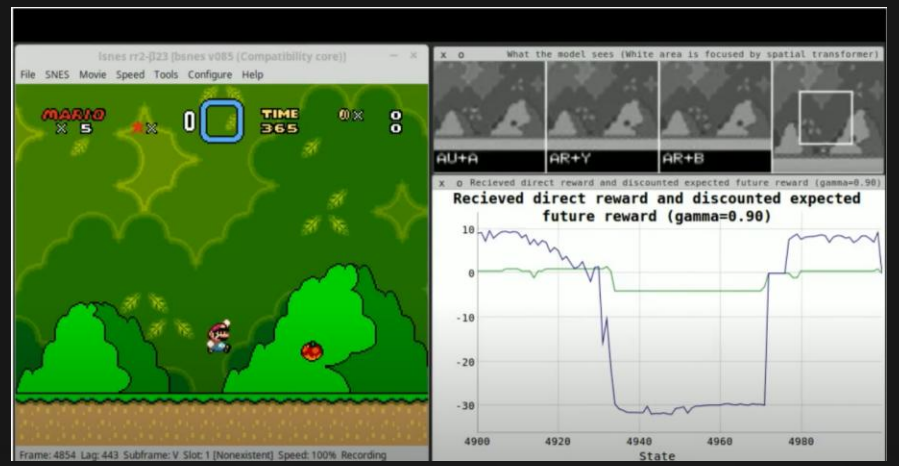
Christiano et al.: „Deep Reinforcement Learning from Human Preferences", 2017.
Ouyang et al.: „Training language models to follow instructions with human feedback", 2022.
→ RL ermöglicht Erhöhen der Genauigkeit des Modells

Detaillierung auf Folgefolien

Reinforcement Learning

„Live“ demos

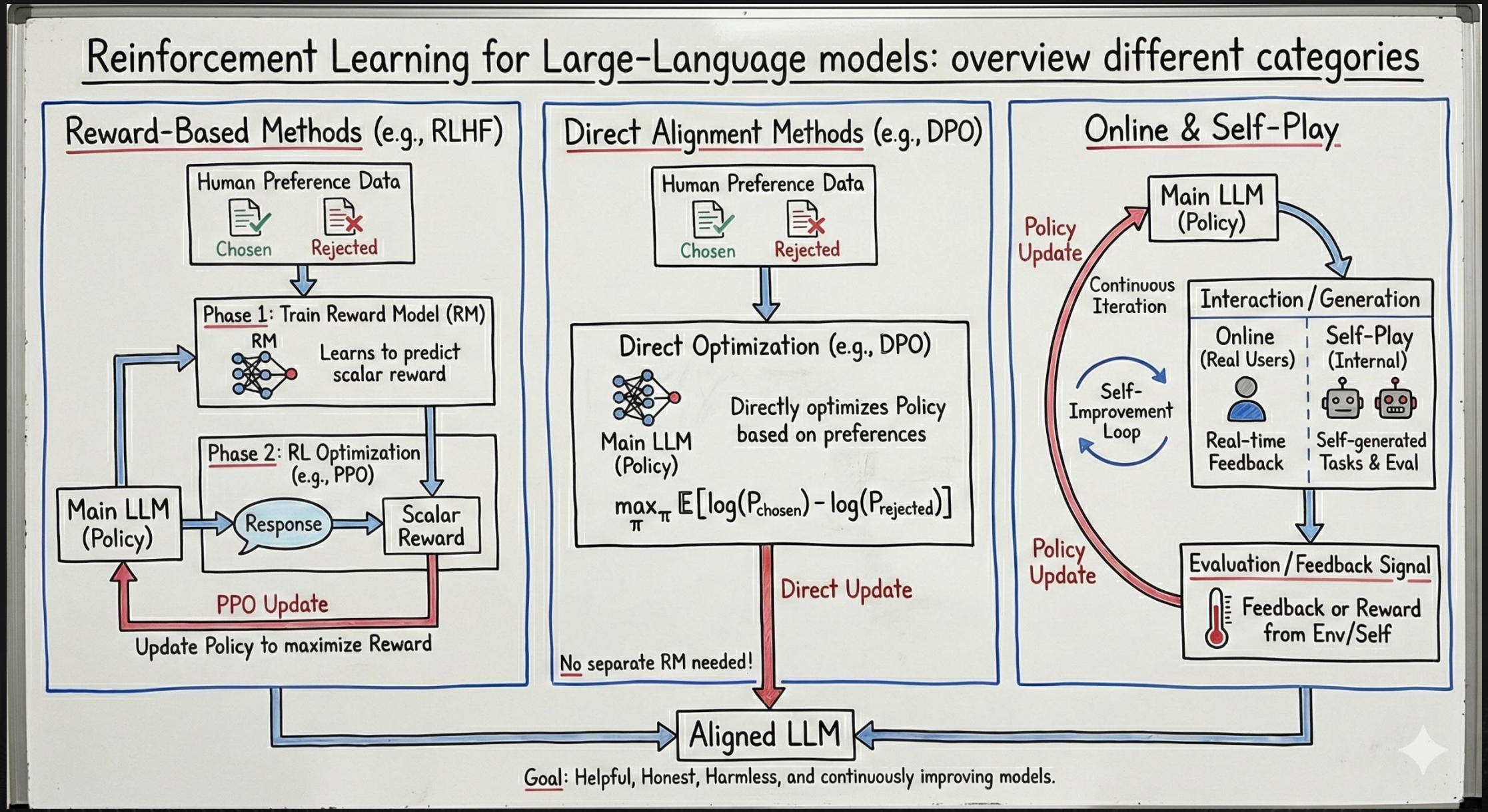
Link zur Demo:
https://www.youtube.com/watch?v=V1eYniJORnk



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

Link zur Demo:
https://www.youtube.com/watch?v=WEOzide5XFc
Link zu detaillierten Erklärungen:
https://www.youtube.com/watch?v=jtlrWblOyP4

Reinforcement Learning für Large Language Modelle/ GenAI



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.

Detaillierung Verfahren im Backup

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

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):

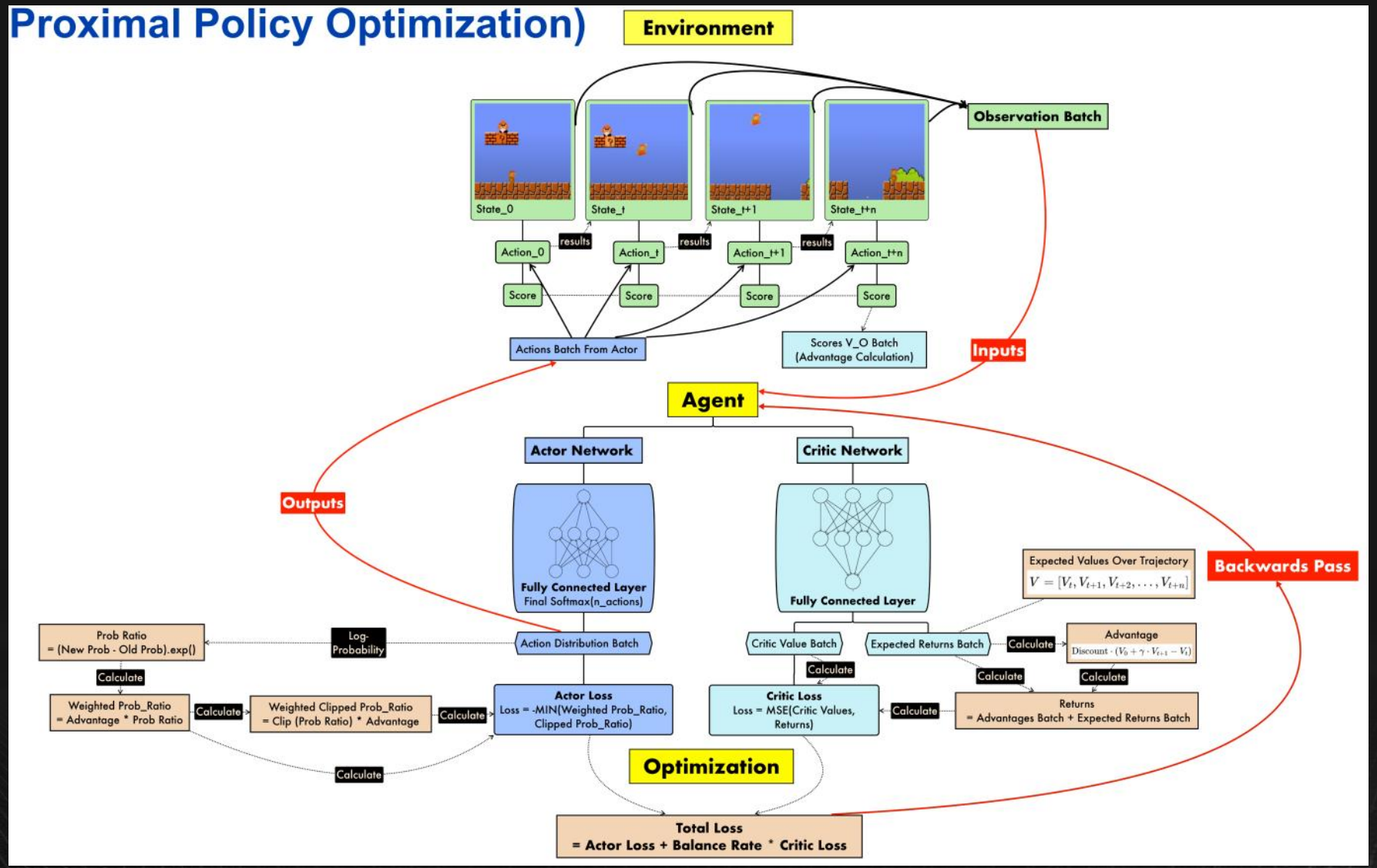
- Kostenfrei: Udacity: Reinforcement Learning, Link
- Coursera: Reinforcement Learning, Link
- Udacity: Deep Reinforcement Learning, Link

Und was jetzt?

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

Backup

Detaillierung PPO



Quelle: Link

Use Case Reinforcement Learning mit Sonic 2

Definition Reward-Funktion als Markov Decision Process (Auszug)

E: Episode mit vielen Schritten t. Episode endet mit Tod Sonic, Ablauf Zeit oder Ende Level

Bonus falls Agent nach rechts geht

Bonus, falls Agent ein vorher unbekanntes Gebiet erreicht

Malus, falls Anzahl Leben geringer als im vorigen Schritt

$$\max_{\pi} E\left[\sum_{t=0}^{T} \omega_1(Progress) + \omega_2(Speed\ bonus_t) + \omega_3(Neues\ Gebiet?) + \omega_4(Ringe_t) - \omega_5(Leben_t) + \omega_6(Level\_complete_t)\right]$$

$Policy\ \pi$: Regeln/ Strategie, welche Aktion in welchem Zustand zum Zeitpunkt t ausgeführt wird

Bonus, falls Agent schneller ist als im vorigen Schritt

Malus, falls Anzahl Ringe geringer als im vorigen Schritt

Bonus. falls Agent das Ende des Levels erreicht

Vereinfacht: Agent erhält

- Belohnung für Aktionen, bei denen er schnell nach rechts läuft, unbekannte Gebiete oder das Ziel findet
- Bestrafung für Aktionen,bei denen er Ringe oder Leben verliert

Ein Reinforcement Learning Agent lernt Menge an Aktionen auszuführen, die eine maximale Belohnung ergeben. Die Herausforderung ist, die Belohnungsfaktoren exakt zu definieren und untereinander zu gewichten.

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)

Verwendet von Deep Seek R1, Open AI o1, xAI

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.