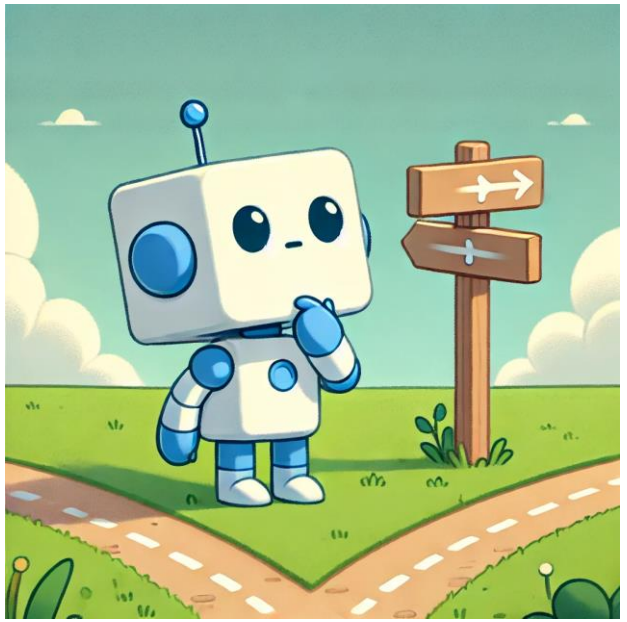
Reinforcement Learning

Eine Einführung

Neele Kemper
neele.kemper@uni-a.de

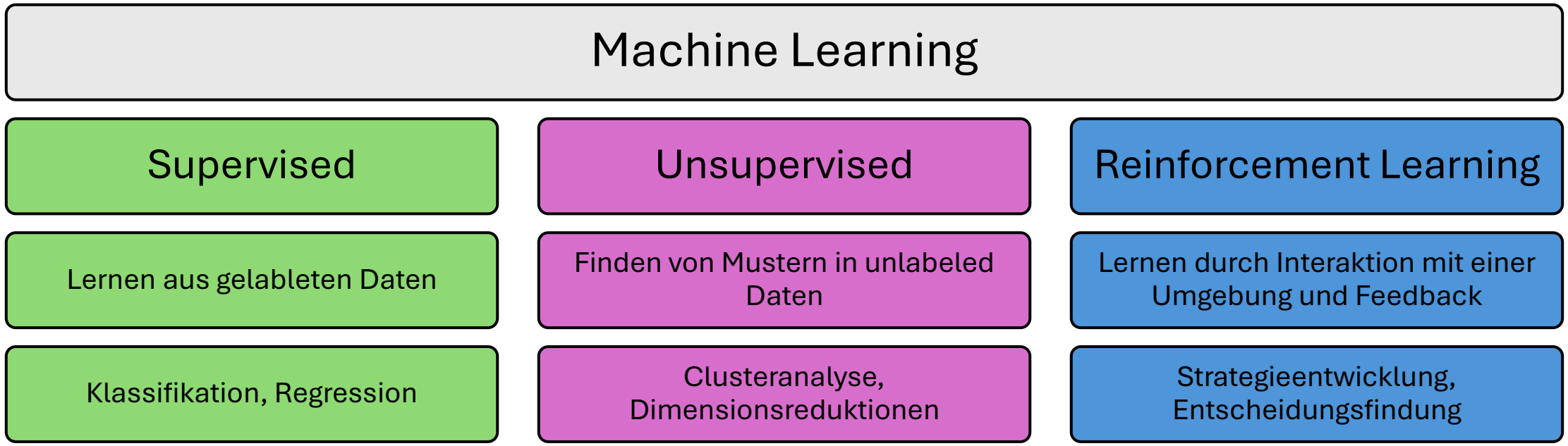
Was ist Reinforcement Learning?

- Reinforcement Learning (RL) ist die Wissenschaft der **Entscheidungsfindung**
- RL ist eine Schnittstelle verschiedener Wissenschaften
  - Informatik, Neurowissenschaften, Psychologie, Mathematik usw.
- **Perspektive der Informatik**
  - Teilgebiet von Machine Learning
  - Fokus auf Lernen durch Interaktion und Belohnung



Das auf dieser Folie verwendeten Bild wurden mithilfe der KI-Anwendung DALL-E generiert

Machine-Learning-Paradigmen



Charakteristika von Reinforcement Learning

- **Kein direkter Supervisor,** sondern ein Belohnungssignal (Reward)
- **Verzögertes Feedback** statt sofortiger Rückmeldung
- **Zeit spielt eine Rolle**: Die Aktionen wirken sequentiell und beeinflussen zukünftige Zustände
- **Agent beeinflusst die Daten**: Durch sein Handeln verändert er die Umgebung und somit auch die eigenen Beobachtungen

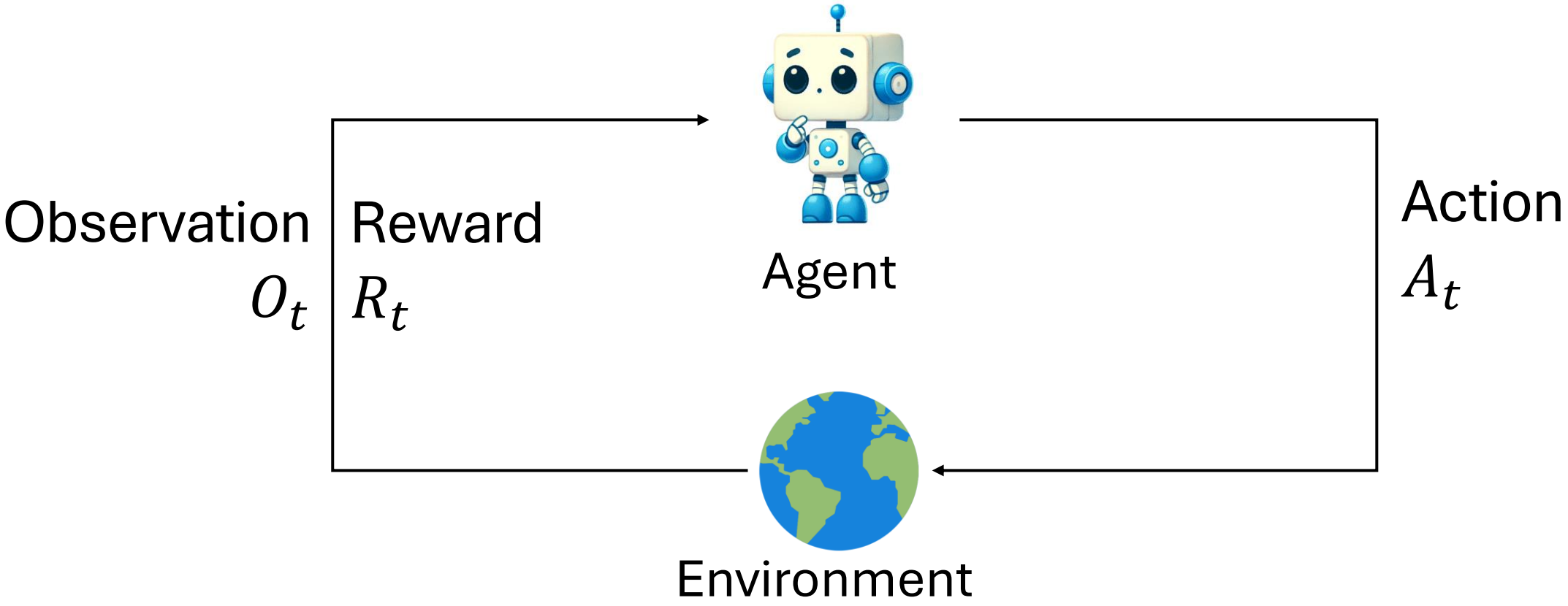
Beispiele für RL-Anwendungen

- **Steuerung von Drohnen**
  - Autonome Navigation, Routenplanung
- **Robotik**
  - Präzise Bewegungsabläufe, flexible Anpassung an wechselnde Umgebungen
- **Spiele-KI**
  - Strategische Entscheidungsfindung (z. B. Atari, Go, Schach)
- **Large Language Models**
  - Kontextsensitives Text- und Dialogverhalten durch RL-Feintuning
- **Anlageportfolios**
  - Dynamische Ressourcenallokation, Risikomanagement

Sequentielle Entscheidungsfindung

- **Ziel:** Aktionen wählen, um langfristigen Gesamtertrag zu maximieren
- **Langfristige Konsequenzen:** Aktionen haben Auswirkungen auf zukünftige Belohnungen
- **Verzögerte Belohnung:** Manchmal lohnt sich Verzicht auf Sofort-Belohnung zugunsten eines größeren, späteren Nutzens

Agent–Umgebungs-Interaktion



$$H_t = A_1, O_1, R_1, \ldots, A_t, O_t, R_t$$

Markov-Zustände (Informationszustand)

- **Formell:**

$$\mathbb{P}[S_{t+1}|S_t] = \mathbb{P}[S_{t+1}|S_1, \dots, S_t]$$

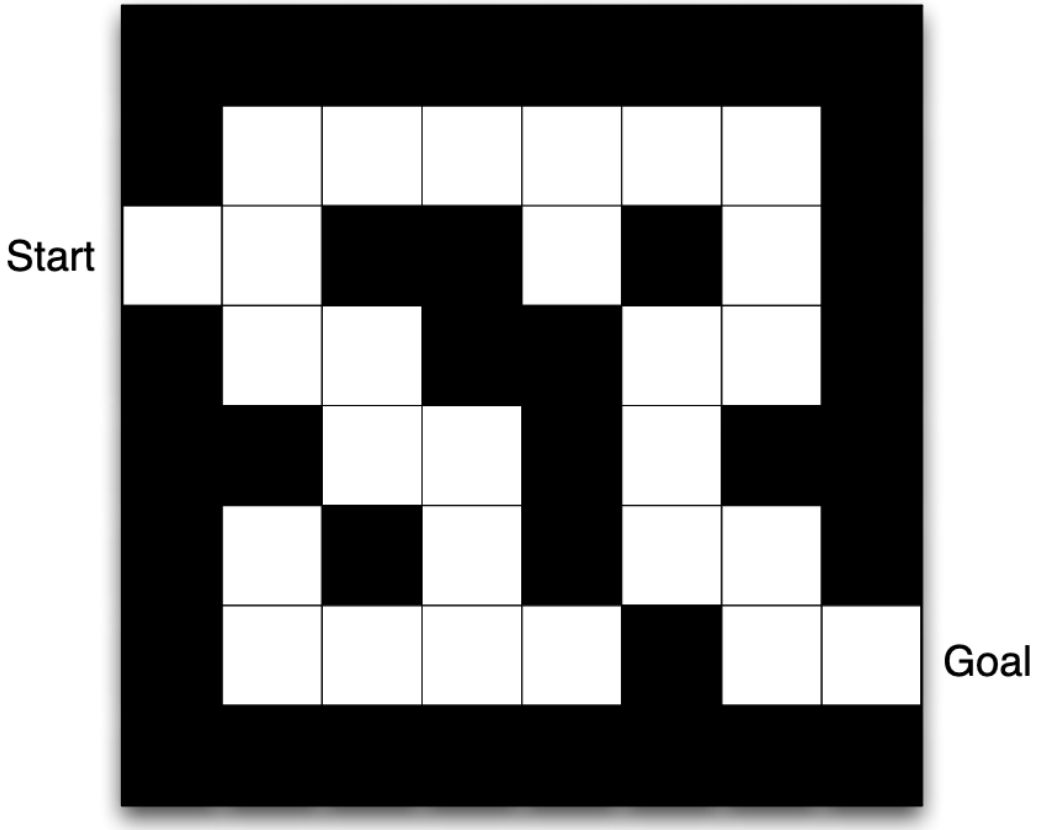
- **Markov-Eigenschaft:**
  - „Die Zukunft ist, gegeben die Gegenwart, unabhängig von der Vergangenheit.“
- **Praktische Folge:**
  - Ist der Zustand (State) S_t bekannt, kann die Historie vernachlässigt werden

Komponenten eines RL-Agenten

- **Policy (π)**
  - Verhaltensfunktion (Abbildung von Zuständen auf Aktionen)
  - Deterministisch: $\pi(s) = a$
  - Stochastisch: $\pi(s) = \mathbb{P}[\mathrm{A} = \mathrm{a}|\mathrm{S} = \mathrm{s}]$
- **Value-Funktion**
  - Schätzt, wie „gut“ ein Zustand oder eine Aktion ist
  - Vorhersage des zukünftigen Rewards

$$v_\pi(s) = \mathbb{E}_\pi[R_t + \gamma R_{t+1} + \gamma^2 R_{t+2}, \ldots | S_t = s])$$

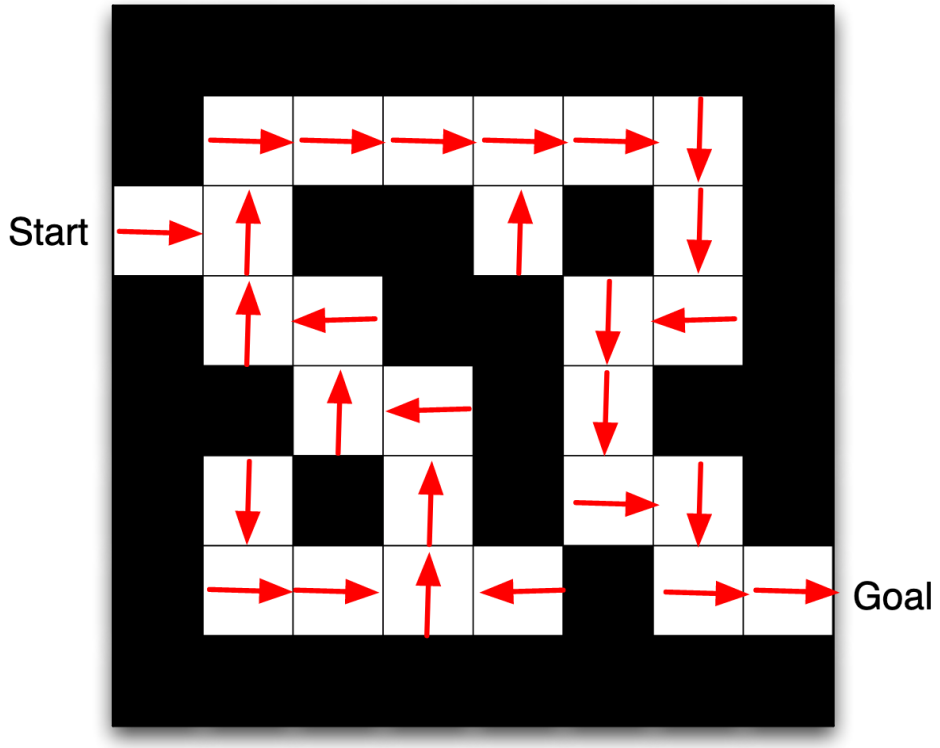
Beispiel: Policy und Value Funktion



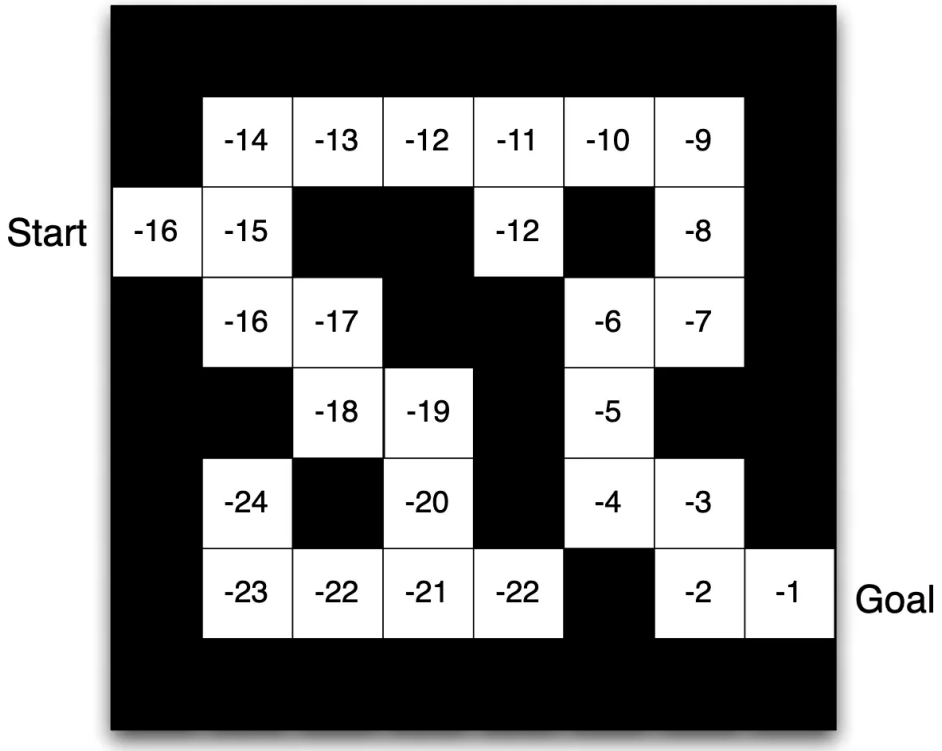
- **Reward**: -1 pro Zeitschritt
- **Aktionen**: {N, E, S, W}
- **Zustand**: Position des Agenten im Labyrinth

Beispiel: Policy und Value Funktion

Policy



Value Function

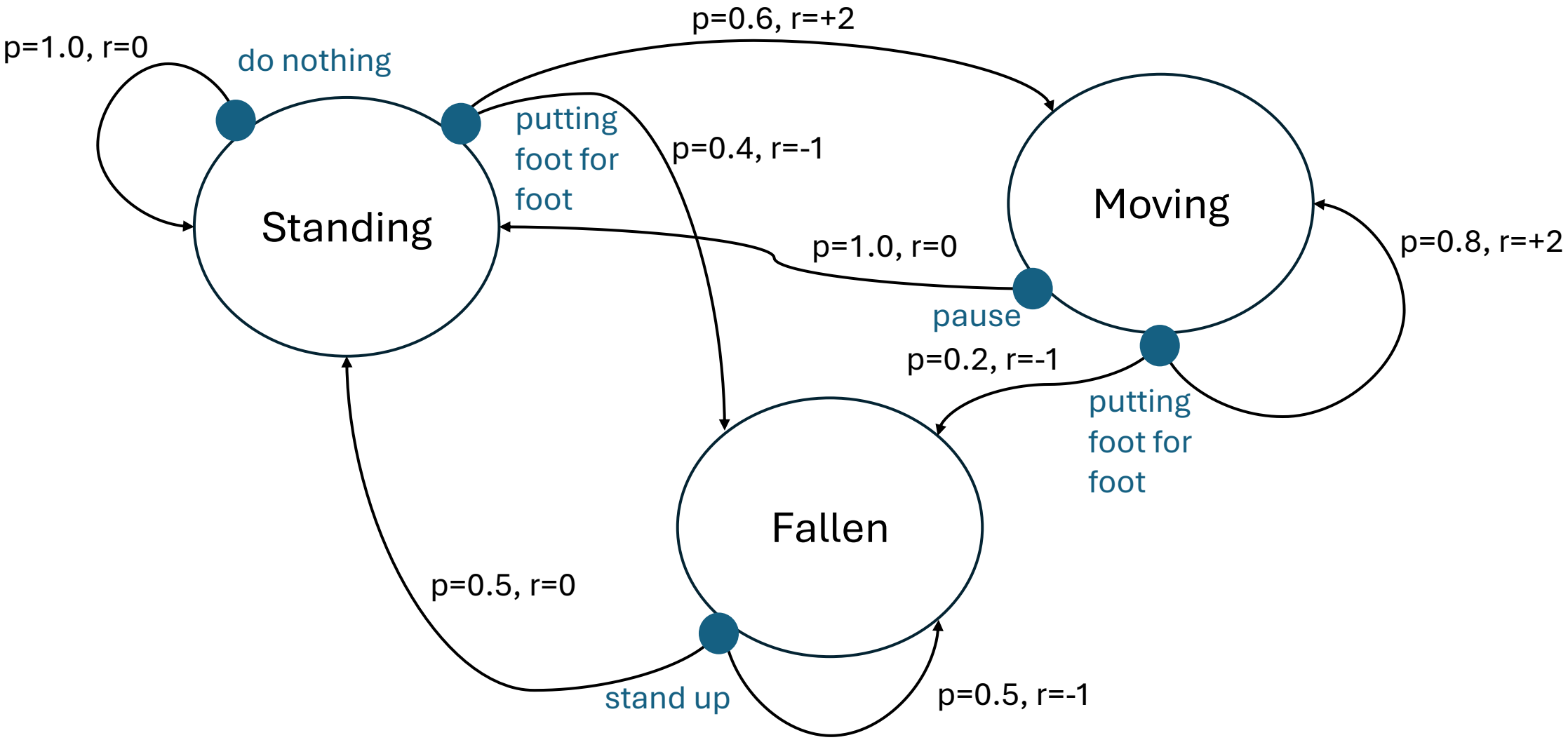


Exploration vs. Exploitation

- RL basiert auf **Trial-and-Error**
- **Exploration**: Neue Zustände und Aktionen erproben, um Wissen über die Umgebung zu sammeln
- **Exploitation**: Das bereits Erlernte nutzen, um Belohnung zu maximieren
- In der Praxis ist ein **Balanceakt** zwischen beiden Strategien erforderlich
- **ϵ-Greedy**: Mit einer kleinen Wahrscheinlichkeit ϵ wählt der Agent eine zufällige Aktion (Exploration), während er sonst (mit Wahrscheinlichkeit (1- ϵ) die aktuell beste bekannte Aktion (Exploitation) ausführt.

Markov Decision Process

Markov Decision Process



Wichtige MDP-Eigenschaften

- **Markov-Eigenschaft**: Alle nötigen Infos im aktuellen Zustand
- **Übergangswahrscheinlichkeit und Reward-Funktion:**

$$p(s_{t+1}, r|s, a) = \mathbb{P}[S_t = s_{t+1}, R_t = r|S_{t-1} = s, A_{t-1} = a]$$

Ertrag (Return)

- **Definition des Ertrag G_t:**

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + \ldots + R_T$$

- **Diskontierung:**

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{T} \gamma^k R_{t+k+1}$$

- **Bellman's Konsistenz:**

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} \ldots \\ &= R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4}) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$

Bellman-Gleichung für State-Value-Funktion

- $v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s]$

  $= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1}|S_t]$

  $= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a)\left[r + \gamma \mathbb{E}_\pi[G_t|S_{t+1} = s']\right]$

  $= \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')]$

Value-Funktion

Summe über alle Aktionen

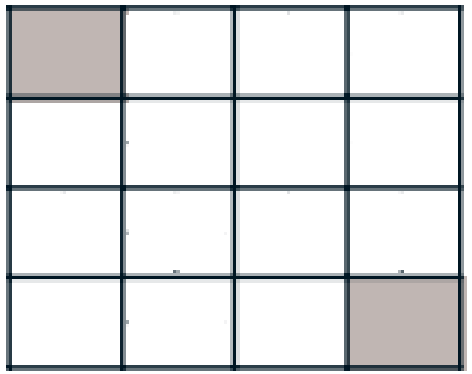
Wahrscheinlichkeit, dass eine Aktion a aus dem Zustand s nach der Policy π ausgeführt wird.

Summe über alle nächsten Zustände s' und allen Belohnungen r

Wahrscheinlichkeit, im Zustand s' zu landen und den Belohnung r zu erhalten, wenn man im Zustand s beginnt und die Aktion a wählt.

Multipliziert mit der Summe von Belohnung r plus des erwarteten Werts (Value) des nächsten Zustands multipliziert mit Diskontierungsfaktor γ

Beispiel: Berechnung der Zustandswerte (State-Value)



- **Szenario**: Gridworld mit zufälliger Policy
- **Aktionen**: {N, E, S, W}
- **Zustand**: terminal (grau); nicht terminale (weiß)
- **Belohnung**: -1 pro Zeitschritt
- **Übergangswahrscheinlichkeiten**: Gleichverteilt (0.25)
- **Diskontfaktor** : $\gamma = 1$
- $v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')]$
- $v(s) = \sum_{s',r} p(s', r|s, a)[r + \gamma v(s')]$

Beispiel: Berechnung der Zustandswerte

v_k for the Random Policy

$k = 0$

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

$k = 1$

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0

$k = 2$

0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0

$k = 3$

0.0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0.0

$k = 10$

0.0	-6.1	-8.4	-9.0
-6.1	-7.7	-8.4	-8.4
-8.4	-8.4	-7.7	-6.1
-9.0	-8.4	-6.1	0.0

$k = \infty$

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0

$$v(s) = \sum_{s',r} p(s', r|s, a)[r + \gamma v(s')]$$

k=0

- $v(s_1) = 0.25 * (-1 + 0.0) + 0.25 * (-1 + 0.0) + 0.25 * (-1 + 0.0) + 0.25 * (-1 + 0.0) = -1$

k=1

- $v(s_6) = 0.25 * (-1 + (-1.0)) + 0.25 * (-1 + (-1.0)) + 0.25 * (-1 + (-1.0)) + 0.25 * (-1 + (-1.0)) = -2$

k=2

- $v(s_{10}) = 0.25 * \big(-1 + (-2.0)\big) + 0.25 * \big(-1 + (-2.0)\big) + 0.25 * \big(-1 + (-1.7)\big) + 0.25 * \big(-1 + (-1.7)\big) = -2.9$

State-Value und Action-Value Funktion

State-Value Funktion für Policy π

- $v_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s]$
- Für jeden Zustand s
- Ergibt sich der erwartete Return
- Wenn der Agent in Zustand s beginnt
- Und dann die Policy π verwendet um seine Aktionen in jedem Zeitschritt zu wählen

Action-Value Funktion für Policy π

- $q_\pi(s, a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a]$
- Für jeden Zustand s und Aktion a
- Ergibt sich der erwartete Return
- Wenn der Agent in Zustand s beginnt und Aktion a wählt
- Und dann die Policy π verwendet um seine Aktionen in jedem Zeitschritt zu wählen

Bellman Optimalitätsgleichung

- **Optimaler State-Value**

$$v_*(s) = \max_\pi v_\pi(s)$$

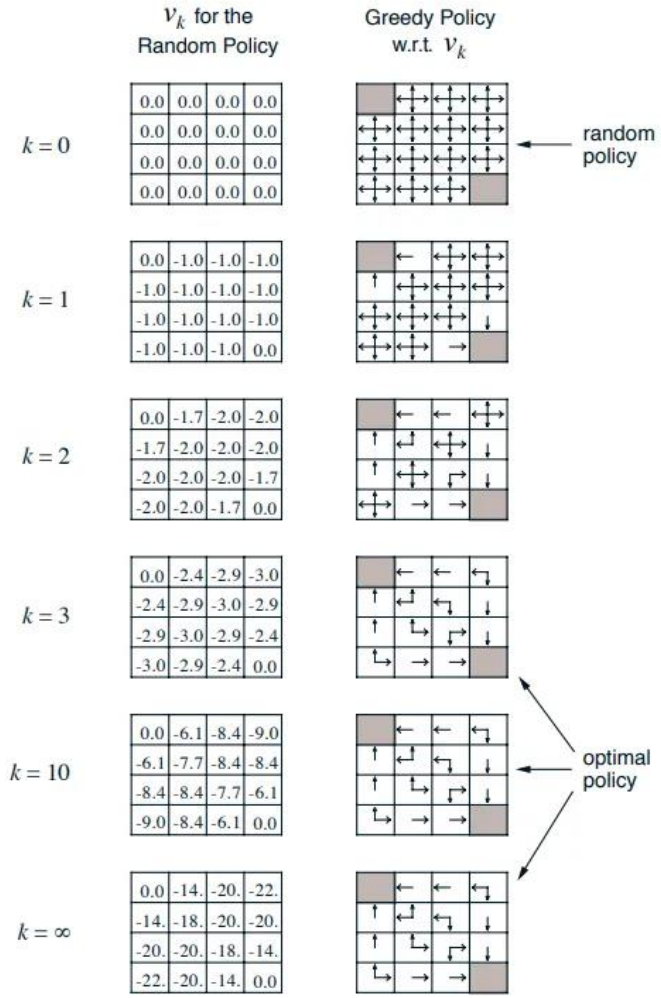
- **Optimaler Action-Value**

$$q_*(s,a) = \max_\pi q_\pi(s,a)$$

- **Bellman-Optimalitätsgleichung**

$$\begin{aligned} v_*(s) &= \max_a q_{\pi*}(s,a) \\ &= \max_a \textstyle\sum_{s',r} p(s',r|s,a)[r + \gamma v_*(s')] \end{aligned}$$

Beispiel: Bellman-Optimalitätsgleichung



Temporal Difference Leraning

Temporal Difference (TD) Leraning

- **Stichprobenbasiert**: Lernt direkt aus gesammelten Erfahrungen (Episoden).
- **Modellfrei**: Es ist kein Wissen über MDP-Übergänge oder Belohnungsfunktionen erforderlich.
- **Bootstrapping**: Lernt aus unvollständigen Episoden, indem es bereits vorhandene Schätzwerte nutzt.
- **TD(0)-Lernregel**: Aktualisierungen erfolgen nach jedem einzelnen Schritt.
- **„TD passt eine Schätzung mithilfe einer Schätzung an“**: Die aktuelle Wertschätzung wird mithilfe der geschätzten Werte des nächsten Zustands aktualisiert.

TD(0)-Aktualisierung

$$V(s_t) \leftarrow V(s_t) + \alpha[R + \gamma V(s_{t+1}) - V(s_t)]$$

- Neuer Wert des Zustands s_t
- Vorherige Schätzung des Werts des Zustands s_t
- Lernrate
- Reward
- Abgezinster Wert bei nächstem Zustand
- TD-Ziel: $R + \gamma V(s_{t+1})$
- Temporaldifferenz: $R + \gamma V(s_{t+1}) - V(s_t)$

SARSA

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Choose A from S using policy derived from Q (e.g., ϵ-greedy)
  Repeat (for each step of episode):
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ϵ-greedy)
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma Q(S',A') - Q(S,A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

Q-Learning

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
    Initialize S
    Repeat (for each step of episode):
        Choose A from S using policy derived from Q (e.g., ϵ-greedy)
        Take action A, observe R, S'
        $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
        $S \leftarrow S'$
    until S is terminal

Praktisches Tutorial

RL Tutorial

https://github.com/NeeleKemper/rl-einfuehrung/tree/main

Google Colab

https://colab.research.google.com/

Ausblick

- Deep Reinforcement Learning
  - Kombination von Neural Networks mit RL zur Lösung hochkomplexer Probleme
  - Ermöglicht das Lernen aus hochdimensionalen Daten
- Multi-Agent Reinforcement Learning
  - RL in Umgebungen mit mehreren Agenten, die miteinander kooperieren oder konkurrieren
  - Herausforderungen bei Kommunikation, Koordination und Skalierbarkeit
- Multi-Objective Reinforcement Learning
  - Optimierung mehrerer, häufig widersprüchlicher Ziele
  - Erfordert Policies zum Balancieren von Trade-Offs zwischen unterschiedlichen Kriterien
- Sicherheit und Erklärbarkeit
  - Absicherung gegen unerwünschtes Verhalten (Safety)
  - Erklärbarkeit von Entscheidungen für mehr Akzeptanz und Transparenz
- …

Literatur

Sutton, Richard S. – Reinforcement Learning: An Introduction. A Bradford Book (2018).

https://github.com/tonberry22/Reinforcement-Learning/blob/master/Reinforcement%20Learning%20-%20An%20Introduction%20%28Sutton%20and%20Barton%20March%202018%29.pdf

RL Course by David Silver (2015)

https://www.youtube.com/watch?v=2pWv7GOvuf0&list=PLzuuYNsE1EZAXYR4FJ75jcJseBmo4KQ9-

Sonstige Empfehlungen

AlphaGo - The Movie

https://www.youtube.com/watch?v=WXuK6gekU1Y&t=3360s