



Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES



inovex

AI Labor - Sommersemester 2020

Reinforcement Learning Einführung

Frederik Martin, Sebastian Blank

Karlsruhe, 29. Mai 2020



Frederik Martin
Software Entwickler
seit 2014

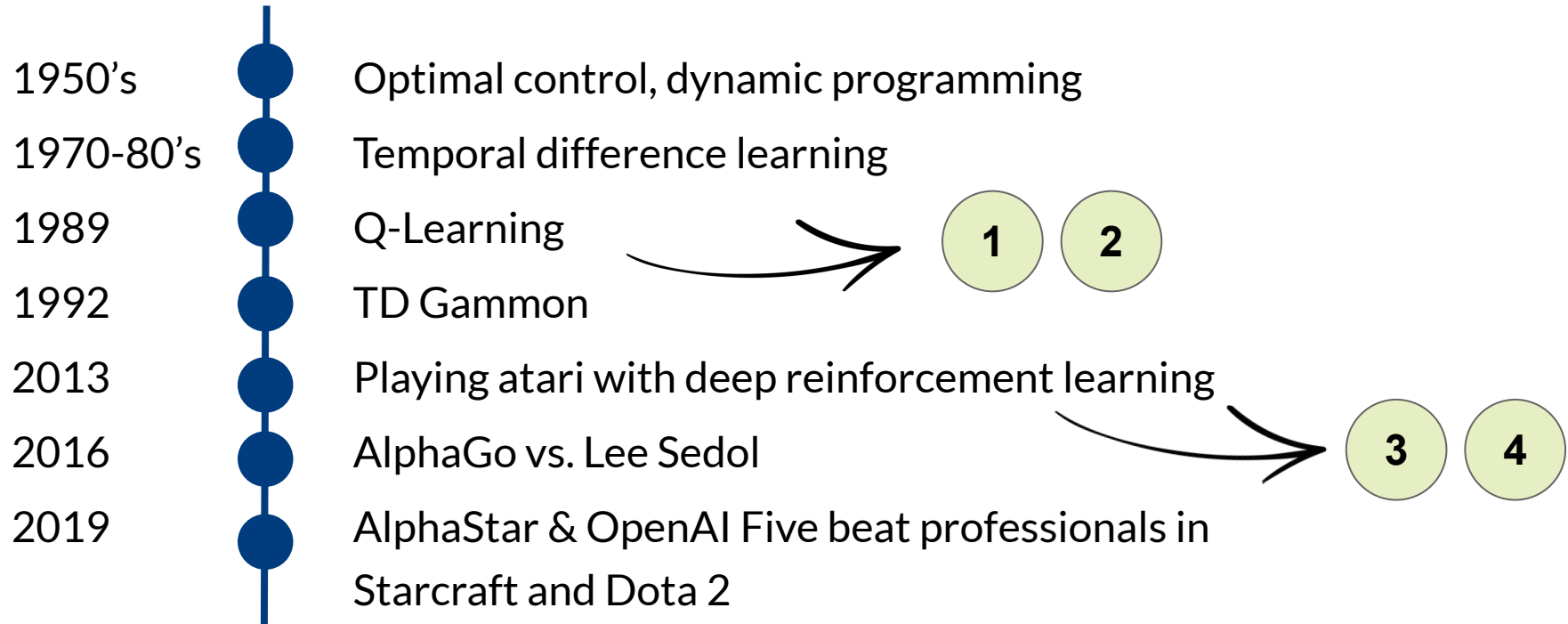


Sebastian Blank
Data Scientist
seit 2017

Reinforcement Learning



Meilensteine im Reinforcement Learning



Reinforcement Learning

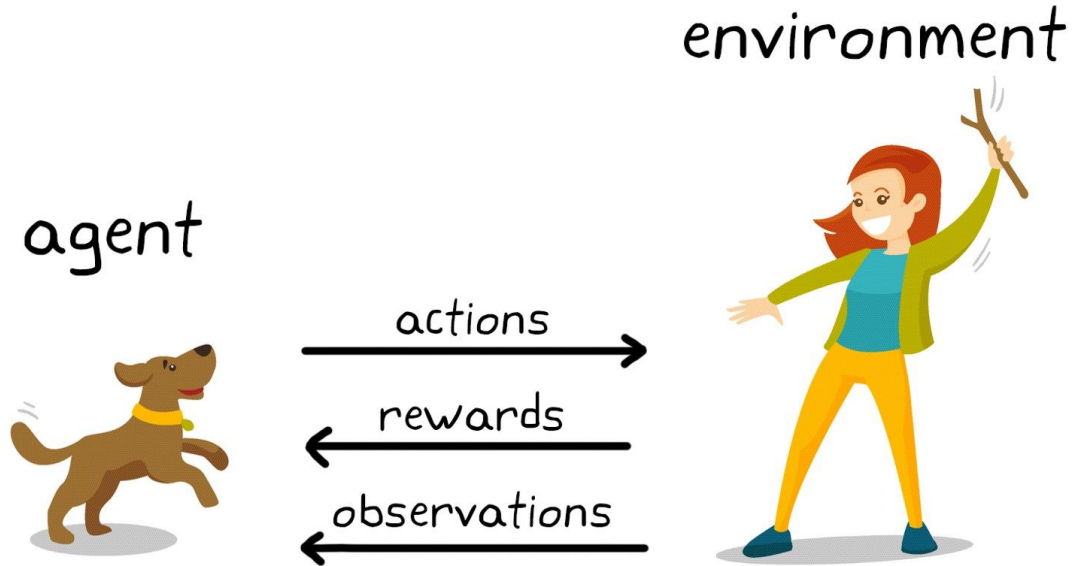
› **Theorie**

- Problemstellung & Lösungsansatz
- Monte Carlo Methoden
- Temporal-Difference Methoden
- Q-Learning

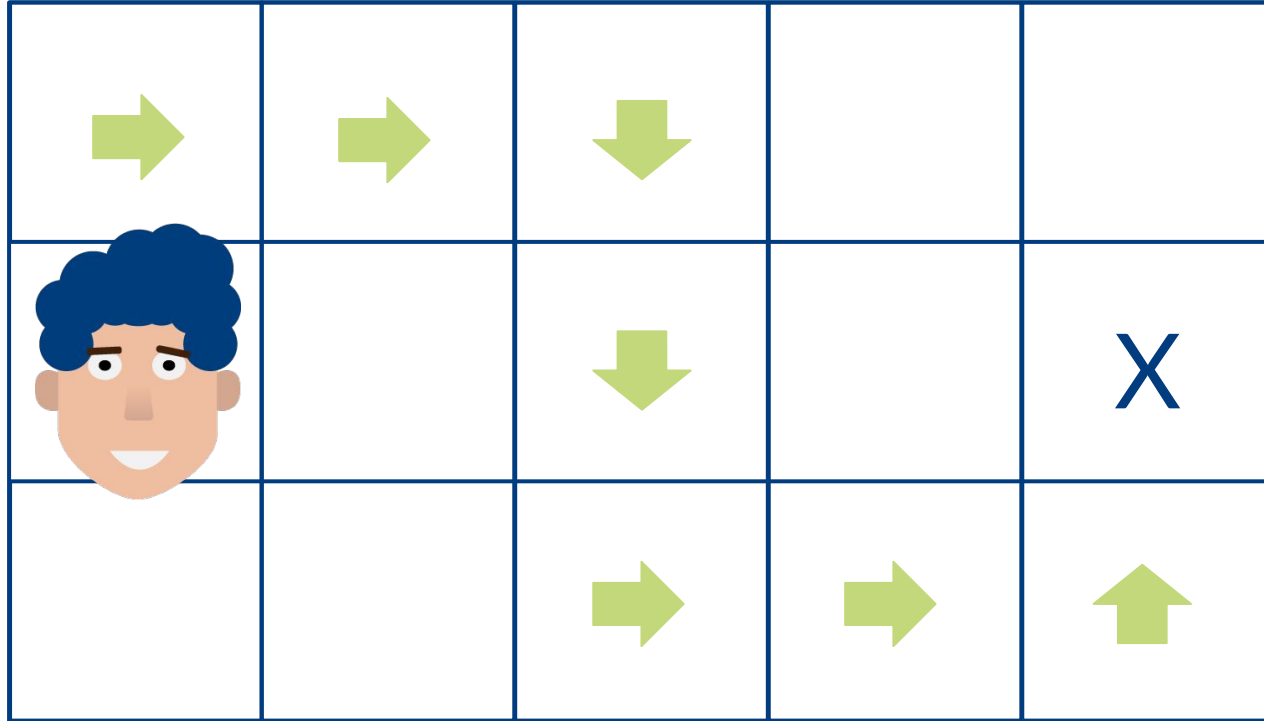
› **Praxis**

- CliffWalking mit Q-Learning (Aufgabe 1)
- CartPole Gym mit Q-Learning (Aufgabe 2)

Reinforcement Learning

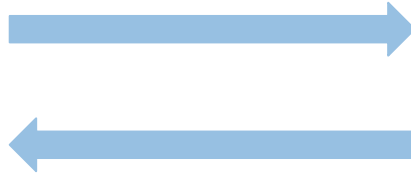


Setting

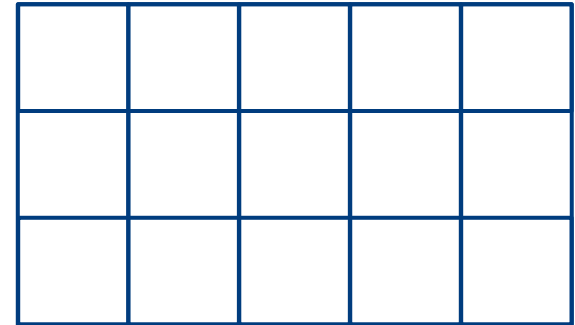


Grundbegriffe

Agent

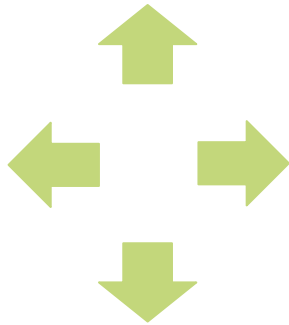


Environment



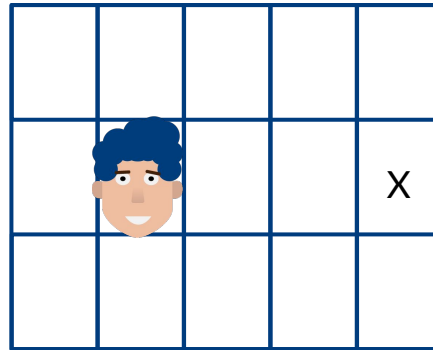
Grundbegriffe

Actions



$$a_t \in \mathcal{A}$$

States



$$s_t \in \mathcal{S}$$

Rewards

-1	-1	-1	-1	-1
-1	-1	-1	-1	0
-1	-1	-1	-1	-1

$$r_t \in \mathcal{R}$$

Return

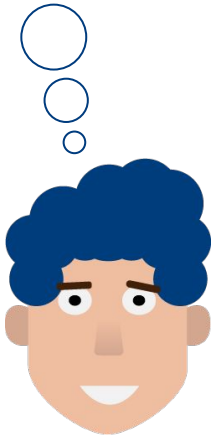
$$\underline{G_t} = \underline{R_{t+1}} + \underline{\gamma} \underline{R_{t+2}} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Return

Rewards

Discount factor

Interaktion zwischen Agent und Umwelt



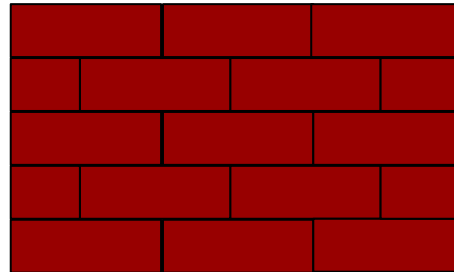
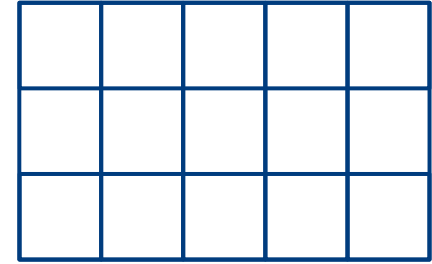
State_t



Action_t



$\text{State}_{t+1}, \text{Reward}_t$



$$\text{State}_{t+1} = P(\text{State}_t, \text{Action}_t)$$

$$\text{Reward}_t = R(\text{State}_t)$$

Transition probabilities $P(s,a)$



Transition probabilities $P(s,a)$

		tatsächlicher Zustandsübergang				
			←	→	↑	↓
Action des Agenten	←	1	80%	0%	10%	10%
	→	2	0%	80%	10%	10%
	↑	3	10%	10%	80%	0%
	↓	4	10%	10%	0%	80%



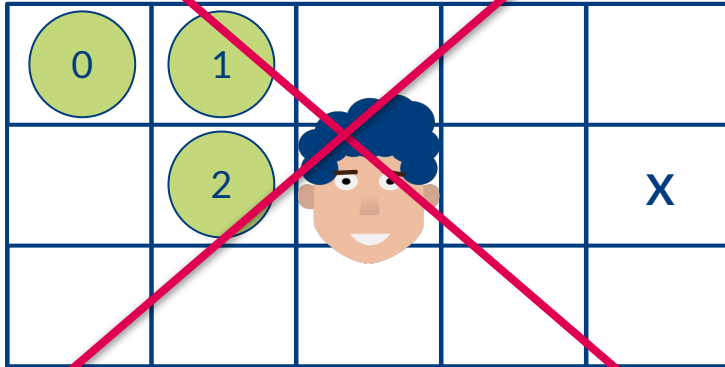
Markov Decision Process (MDP)

Formale Beschreibung der Interaktion im RL

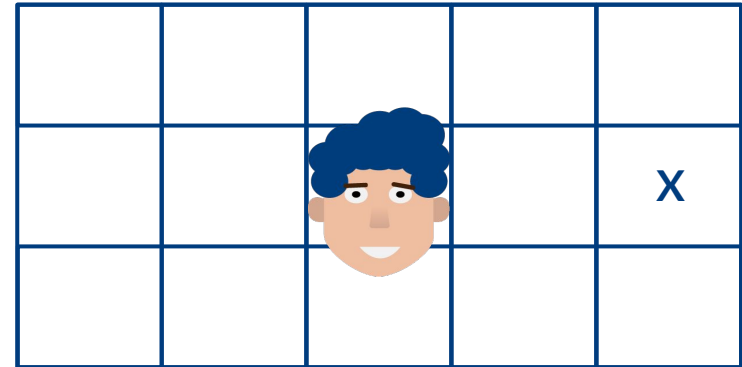
- S Set von States
- A Set von Aktionen
- $R(s,a)$ Reward Funktion
- $P(s,a)$ Transition Probabilities
- γ Discount Factor

Markov Decision Process (MDP)

Wissen über vorherige Aktionen



Unwissenheit über Vergangenheit



Der Übergang in den nächsten Zustand s' hängt nur von aktuellem Zustand s und Aktion a ab.

Policy π



Deterministische Policy

↓	↓	↓	↓	↓
↓	↓	↓	↓	x
→	→	→	→	↑

Stochastische Policy


→	→	→	→	↓
↑ →	↑ →	↑ →	↑ →	x
↑ →	↑ →	↑ →	↑ →	↑

Policies definieren das Verhalten eines Agenten gegeben einem State.

State-value function

Können wir mit diesen Infos den Wert des Zustands bestimmen?

State s

				X

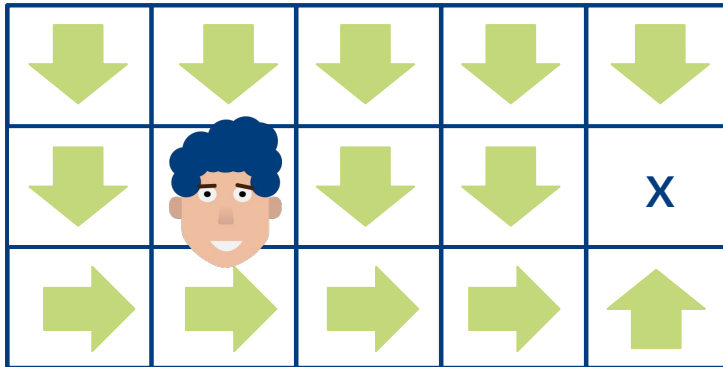
Rewards

-1	-1	-1	-1	-1
-1	-1	-1	-1	0
-1	-1	-1	-1	-1

State-value function

-1	-1	-1	-1	-1
-1	-1	-1	-1	0
-1	-1	-1	-1	-1

State s und Policy π



State-value function $v_{\pi}(s)$

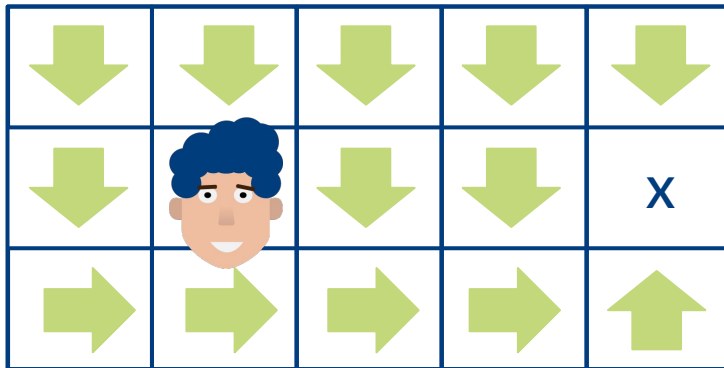
-7	-6	-5	-4	-1
-6	-5	-4	-3	X
-5	-4	-3	-2	-1

$$v_{\pi}(s) = \mathbb{E}_{\pi}[G_t | S_t = s] = \mathbb{E}[R_{t+1} + G_{t+1}(S_{t+1}) | S_t = s]$$

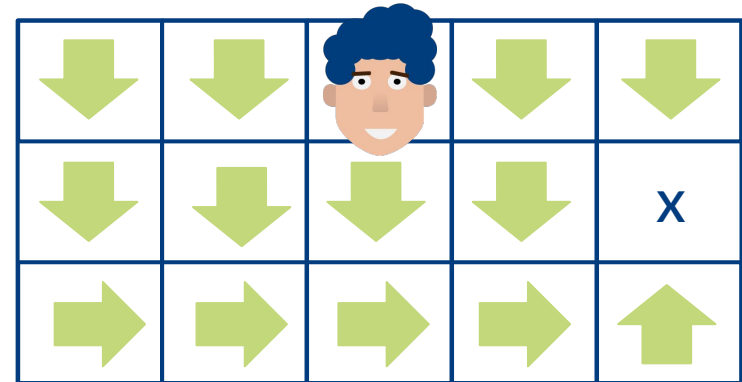
State-value function

Welcher Zustand ist besser?

State 1

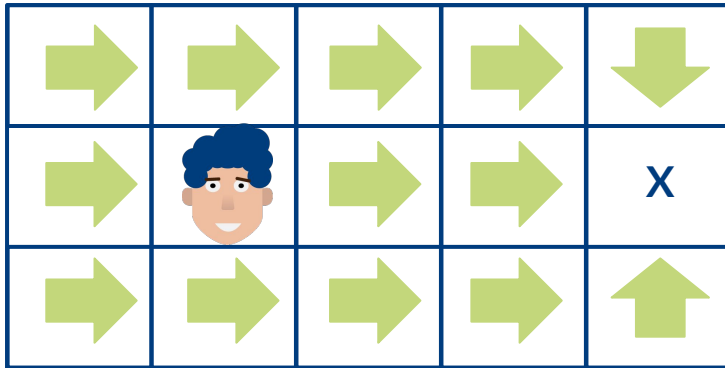


State 2

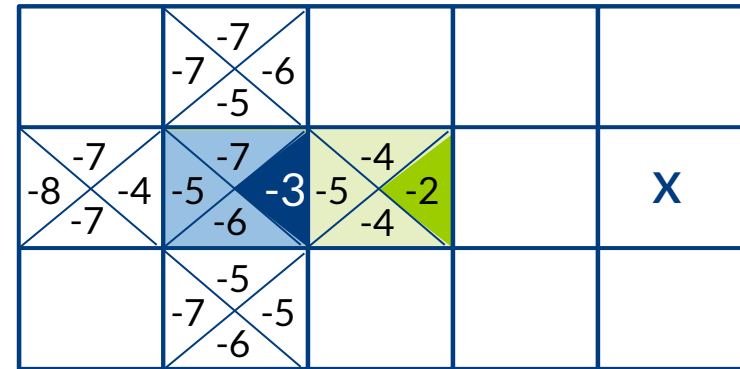


Action-value function $q_{\pi}(s,a)$

State s und Policy π



Action-value function $q_{\pi}(s,a)$



$$\begin{aligned}
 q_{\pi}(s, a) &= \mathbb{E}_{\pi}[G_t | S_t = s, A_t = a] \\
 &= \mathbb{E}[R_{t+1} + G_{t+1}(S_{t+1}, A_{t+1}) | S_t = s, A_t = a]
 \end{aligned}$$

Action-value function $q_{\pi}(s,a)$

Action-value function $q_{\pi}(s,a)$

	<div>-7 -7 -6 -5</div>			
<div>-8 -7 -4 -7</div>	<div>-7 -5 -6 -3</div>	<div>-4 -5 -4 -2</div>		X
	<div>-5 -7 -5 -6</div>			

Q-Table

				
s_{12}	-6	-7	-7	-5
s_{21}	-4	-8	-7	-7
s_{22}	-3	-5	-7	-6
s_{23}	-2	-5	-4	-4
s_{32}	-5	-7	-5	-6

Optimal Policies π^*



-5 →	-4 →	-3 →	-2 →	-1 ↓
-4 →	-3 →	-2 →	-1 →	X
-5 →	-4 →	-3 →	-2 →	-1 ↑

-5 →	-4 →	-3 →	-2 →	-1 ↓
-6 ↓	-5 ↓	-4 ↓	-1 →	X
-5 →	-4 →	-3 →	-2 →	-1 ↑

Optimale Policy ist besser
alle andere Policies:

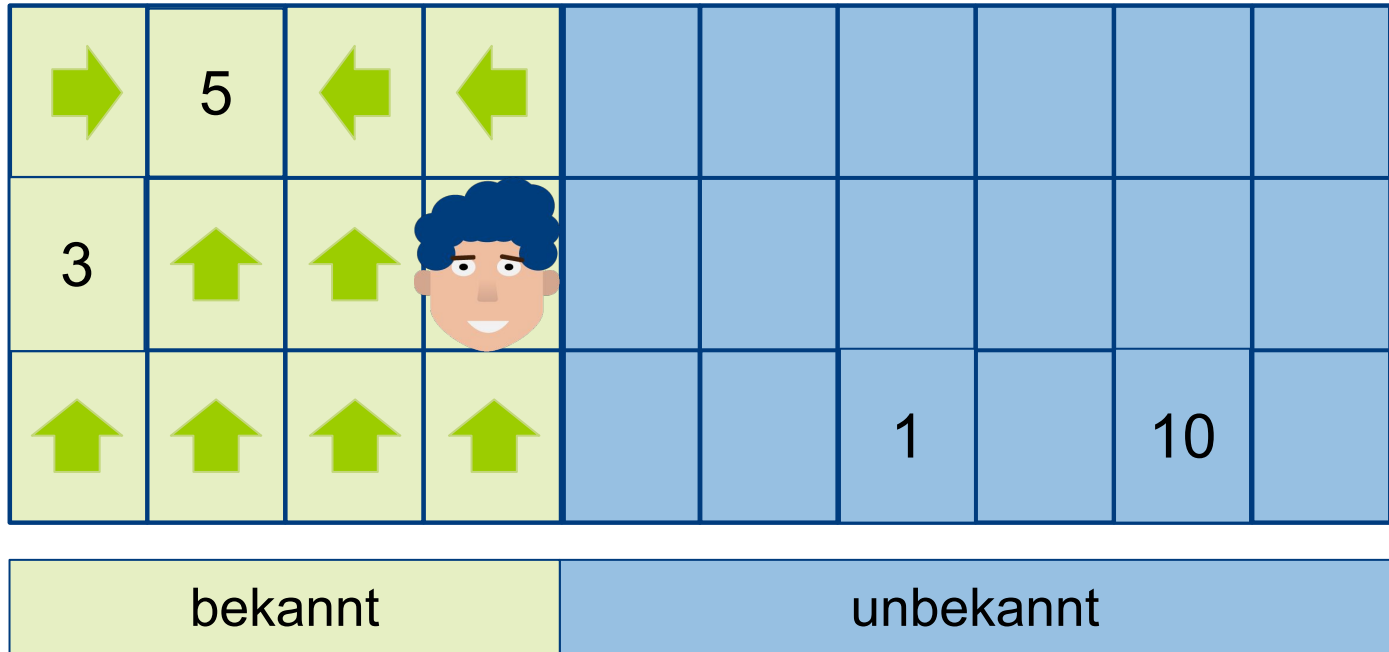
$$\pi^* \geq \pi, \forall \pi$$

Was bedeutet besser?

$$\pi \geq \pi', \text{ if } v_\pi(s) \geq v_{\pi'}(s), \forall s$$

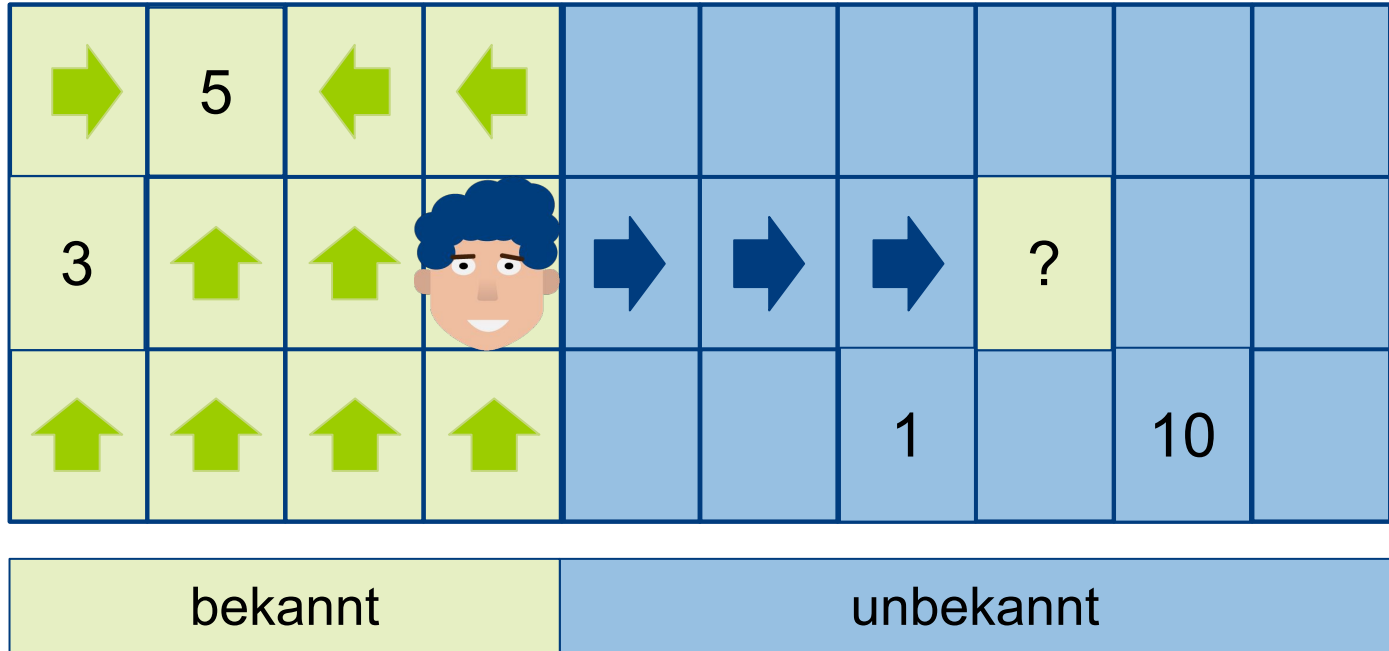
Exploitation

Maximierung des Rewards gg. bekannter Information

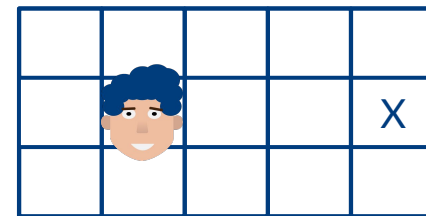


Exploration

Erschließung neuer, unbekannter Bereiche



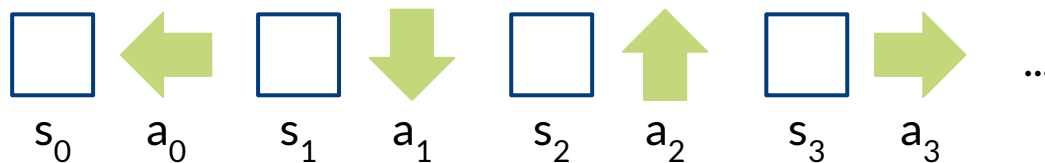
Monte Carlo Methods



Random Policy π

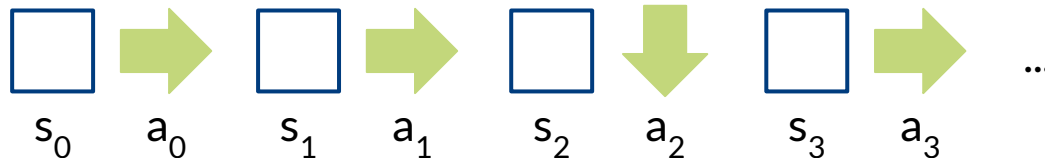
Episode 1

Score:
-4



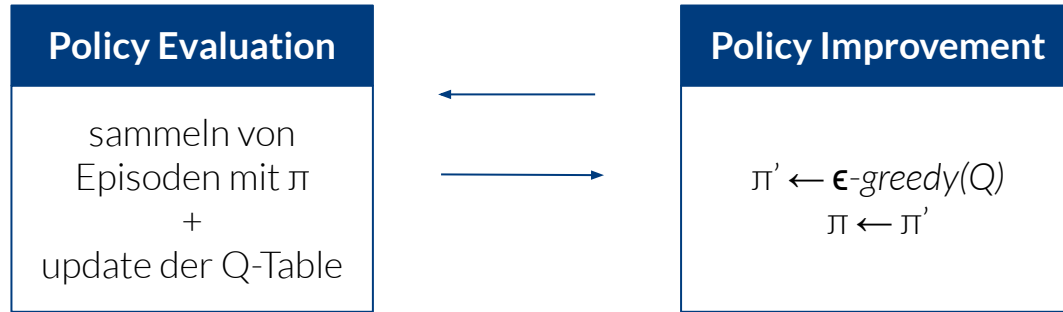
Episode 2

Score:
-7



...

Monte Carlo Prediction



$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(\underbrace{G_t}_{\text{alternative Schätzung}} - \underbrace{Q(S_t, A_t)}_{\text{aktuelle Schätzung}})$$

Control Problem: Estimate the optimal policy

Temporal-Difference Methods

Monte Carlo Control

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left(\underbrace{G_t}_{\text{alternative Schätzung}} - \underbrace{Q(S_t, A_t)}_{\text{aktuelle Schätzung}} \right)$$

alternative
Schätzung

aktuelle
Schätzung

Temporal-Difference Control

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left(\underbrace{R_{t+1} + \gamma Q(S_{t+1}, A_{t+1})}_{\text{alternative Schätzung}} - \underbrace{Q(S_t, A_t)}_{\text{aktuelle Schätzung}} \right)$$

alternative
Schätzung

aktuelle
Schätzung

Q-Learning

Off-Policy TD-Control

TD Update

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \underbrace{\gamma Q(S_{t+1}, A_{t+1})}_{\text{alternative Schätzung}} - Q(S_t, A_t))$$

alternative
Schätzung

Q-Learning
Update

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \underbrace{\gamma \max_a Q(S_{t+1}, a)}_{\text{alternative Schätzung}} - Q(S_t, A_t))$$

alternative
Schätzung

Q-Learning

Off-Policy TD-Control

Initialize $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\text{terminal-state}, \cdot) = 0$
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'
 $Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma \max_a Q(S', a) - Q(S, A)]$
 $S \leftarrow S'$;
 until S is terminal

Aufgaben

OpenAI Gym

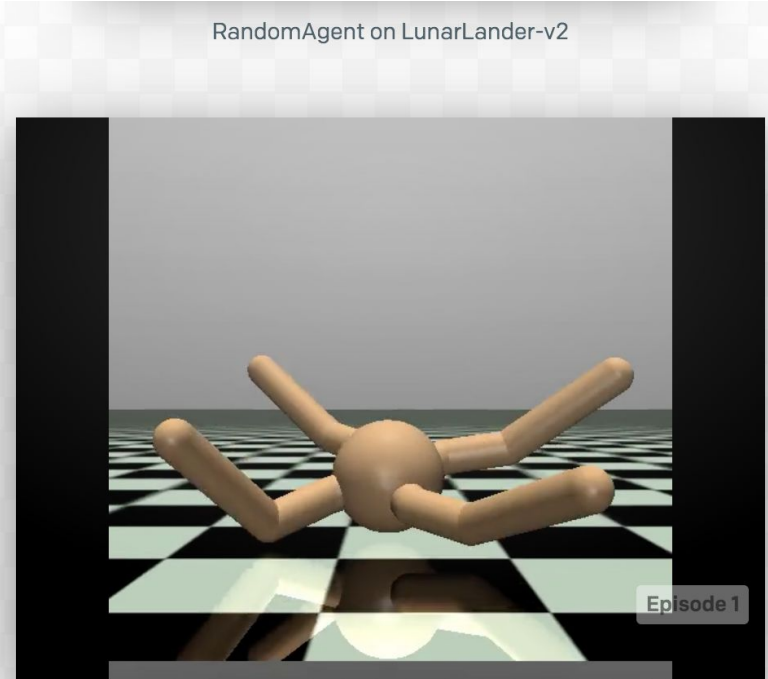


Gym

Gym is a toolkit for developing and comparing reinforcement learning algorithms. It supports teaching agents everything from walking to playing games like Pong or Pinball.

[View documentation >](#)

[View on GitHub >](#)



RandomAgent on Ant-v2

OpenAI Gym

```
import gym

env = gym.make('CartPole-v0')
for i_episode in range(20):
    observation = env.reset()
    for t in range(100):
        env.render()
        action = env.action_space.sample()
        observation, reward, done, info = env.step(action)
        if done:
            print("Episode finished after {} timesteps".format(t+1))
            break
    env.close()
```

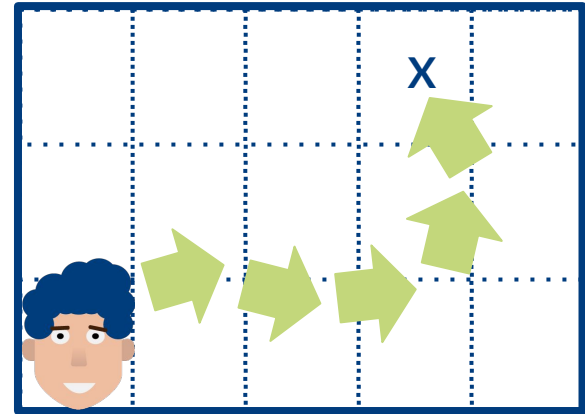
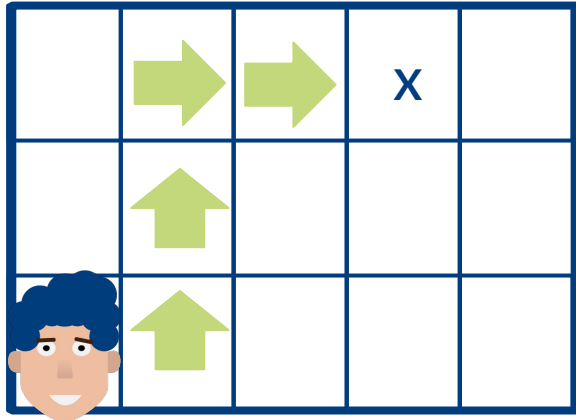


Aufgabe 1: CliffWalking mit Q-Learning

- › **Jupyter Lab Notebook**

Zustands- und Aktionsräume

Wie unterscheiden sich diese beiden Environments?



Aufgabe 2: CartPole Gym mit Q-Learning

- › **Jupyter Lab Notebook**

colab

Google Colab

- › In GCP gehosteter Jupyter Notebook Service
- › Kollaborative Arbeit (ähnlich GDoc) möglich
- › Frei nutzbar (inklusive GPU/TPU!)
 - › → Ressourcen sind limitiert
 - › → GPU/TPU Runtime sparsam nutzen!
- › Notebooks und Daten liegen im Google Drive



#TFDevSummit

Making the most of Colab



Literatur

- › Kostenlose "Standard"-Lektüre für den Einstieg in RL:
Reinforcement Learning: An Introduction (Sutton and Barto), siehe <http://incompleteideas.net/book/RLbook2018.pdf>
- › Ausführlich und gut erklärter Einstieg in RL (Video-Lektionen):
UCL Course on RL (David Silver, Google DeepMind), siehe <https://www.davidsilver.uk/teaching/>
- › *Algorithms in Reinforcement Learning* von Csaba Szepesvári, siehe <https://sites.ualberta.ca/~szepesva/papers/RLAlgsInMDPs.pdf>
- › Blog mit Videos zum Einstieg in RL und Q-Learning, DQN und vieles mehr:
Reinforcement Learning – Introducing Goal Oriented Intelligence, siehe <https://deeplizard.com/learn/video/nyjbcRQ-uQ8>

Feedback



<https://forms.gle/zDjawTGbbs1Z8ryXA>

Vielen Dank

Frederik Martin

fmartin@inovex.de

Sebastian Blank

sblank@inovex.de

