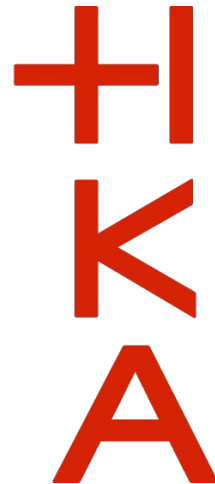




KI Labor - Sommersemester 2022

**R**einforcement **L**earning



Robin Baumann, Sven Müller, Maximilian Blanck,  
Pascal Fecht, **Matthias Richter**, **Tim Bossenmaier**

Karlsruhe, 20. Mai 2022

# Schedule



Datum	Thema	Inhalt	Präsenz
18.03.22	Allg.	Organisation, Teamfindung, Vorstellung CV	Ja
25.03.22	CV	Q&A Sessions	Nein
01.04.22	CV	Sprintwechsel, Vorstellung Assignment	Ja
08.04.22	CV	Q&A Sessions	Nein
05.04.22	Ostern		
22.04.22	CV / NLP	Abgabe CV, Vorstellung NLP	Ja
29.04.22	NLP	Q&A Sessions	Nein
06.05.22	NLP	Sprintwechsel, Vorstellung Assignment	Ja
13.05.22	NLP	Q&A Sessions	Nein
20.05.22	NLP / RL	Abgabe NLP, Vorstellung RL	Ja
27.05.22	RL	Sprintwechsel, Vorstellung Assignment	Nein
03.06.22	Ausfall		
10.06.22	Pfingsten (H-KA zu)		
17.06.22	RL	Q&A Sessions (Brückentag)	Nein
24.06.22	RL	Abgabe RL, Abschluss KI Labor	Ja



Matthias Richter  
Machine Learning Engineer  
seit 2019



Tim Bossenmaier  
Softwareentwickler Datenplattformen  
seit 2021

# Agenda

## › **Theorie**

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

## › **Übungsaufgaben**

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

# Reinforcement Learning



# “Robots that learn a little like humans do”

Law of effect (nach Thorndike, 1898):

responses that produce a satisfying effect in a particular situation become more likely to occur again in that situation, and responses that produce a discomforting effect become less likely to occur again in that situation.

**Menschen lernen Verhalten durch  
Belohnung und Strafe**

(mathematische)  
Psychologie

Kontroll-  
Theorie

Reinforcement  
Learning

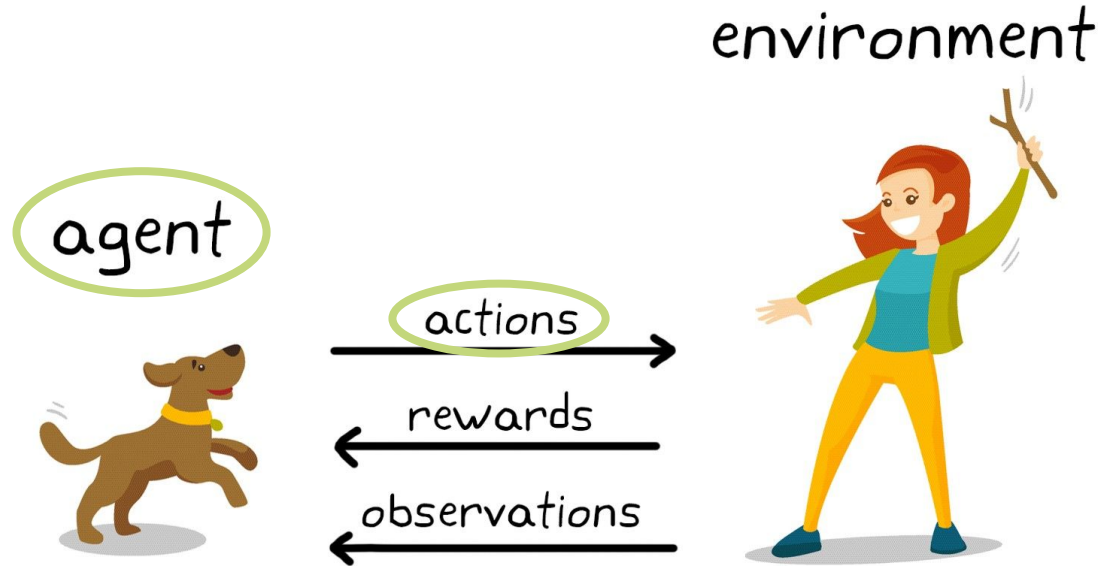
Künstliche  
Intelligenz

Neuro-  
wissenschaften

Operations  
Research



# Reinforcement Learning



# Vergleich mit (un)überwachtem Lernen

(Un)Supervised Learning

Lernen mit Datensätzen

Ziel: Loss minimieren

Interaktion mit Umwelt nicht  
Teil des Systems

Getrennte Trainings- &  
Durchführungsphase

Reinforcement Learning

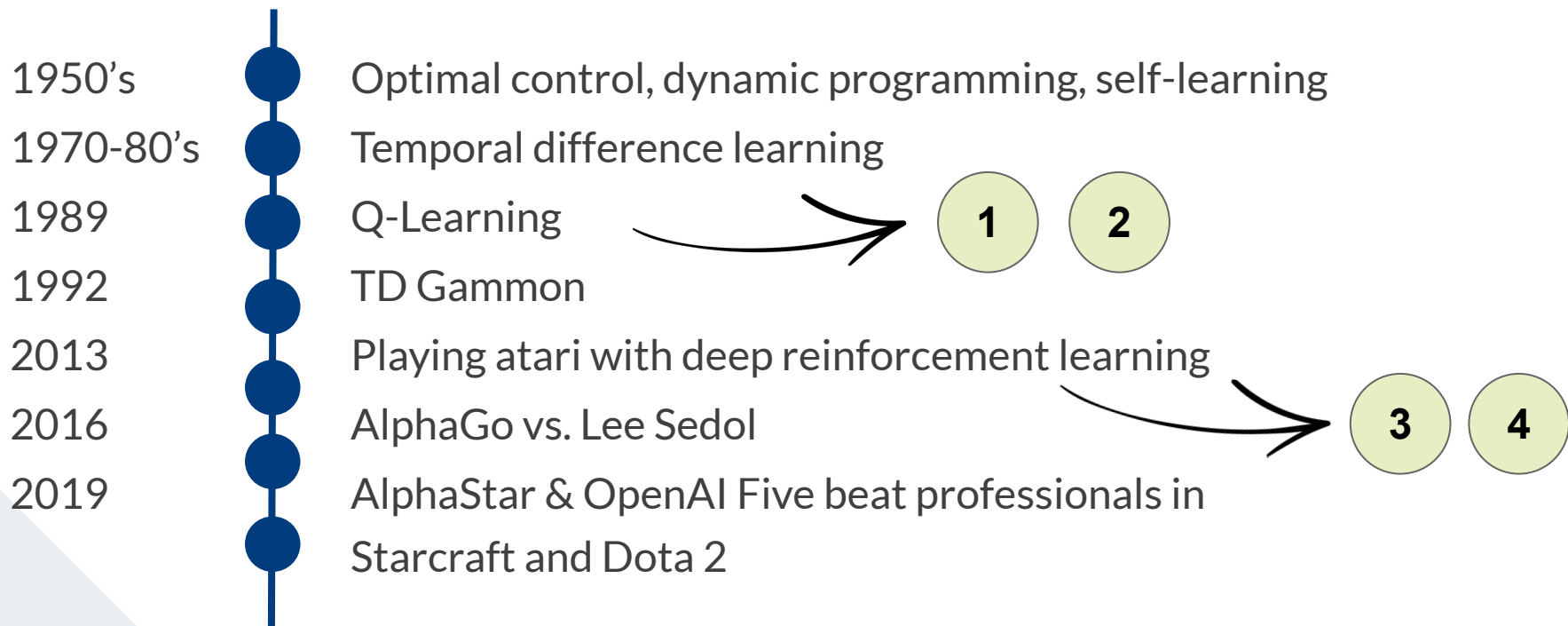
Ziel: Reward maximieren

Lernen durch Ausprobieren

Interaktion mit Umwelt ist  
zentraler Teil des Systems

Kontinuierliches Lernen /  
Exploration vs. Exploitation

# Meilensteine im Reinforcement Learning



# Beispiel: Tic-Tac-Toe

- 9 Felder
- je 3 mögliche Belegungen

⇒ # Zustände  $\leq 3^9 = 19.683$

# Spiele  $\leq 9! = 362.880$

Lösbar, aber aufwendige  
Programmierung



# MENACE [Michie1963]

## Matchbox Educable Naughts And Crosses Engine



- Eine Schachtel pro Spielzustand für MENACE
- Perlen in Schachteln für mögliche Spielzüge
- Spielzug bestimmen = Bohne aus Schachtel ziehen

[Michie 1963]: <https://people.csail.mit.edu/brooks/idocs/matchbox.pdf>

Fotos: James Bridle, <http://jamesbridle.com/works/menace>

Moderne Implementierung: <https://www.youtube.com/watch?v=R9c-neaxeU>

# MENACE [Michie1963]

Nach dem Spiel: Lernen

## Gewonnen

je 2 Perlen gleicher Farbe  
in Schachtel zurücklegen

## Unentschieden

Perlen zurücklegen

## Verloren

Perlen entfernen



[Michie1963]: <https://people.csail.mit.edu/brooks/idocs/matchbox.pdf>

Fotos: James Bridle, <http://jamesbridle.com/works/menace>

Moderne Implementierung: <https://www.youtube.com/watch?v=R9c-neaxeU>

# MENACE ist Reinforcement Learning

MENACE

Reinforcement Learning

Regeln & Gegner

Umwelt

Schachtel

(Spiel-)Zustand  $s_t \in \mathcal{S}$

Perlen/Spielzüge

Aktionen  $a_t \in \mathcal{A}$

Perlen zurück/weg legen

Reward  $r : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$

Spielrunde

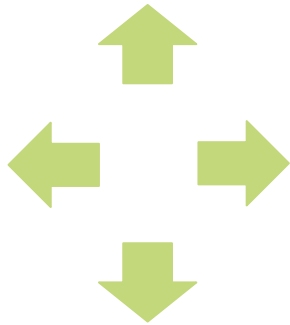
Zeit  $t = 0, 1, 2, \dots$



# Gridworld

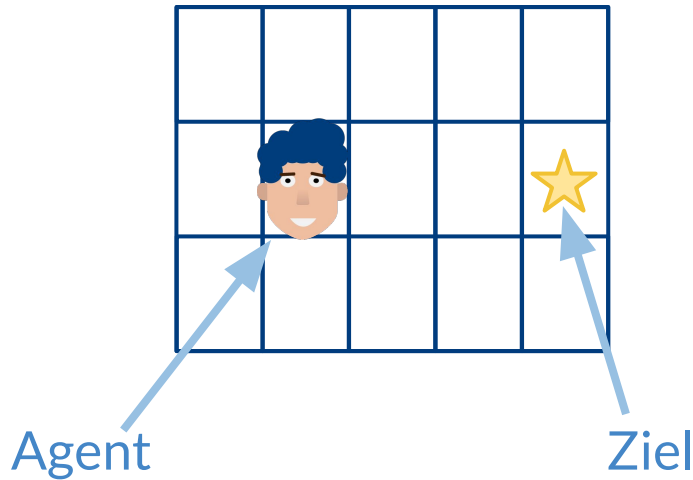
Actions

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



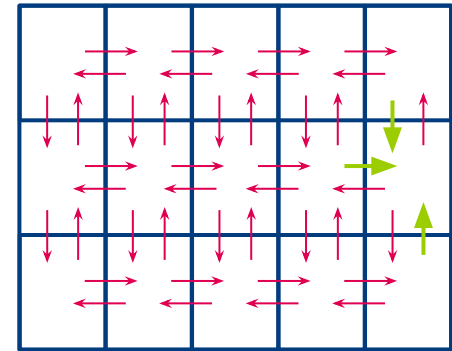
States

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



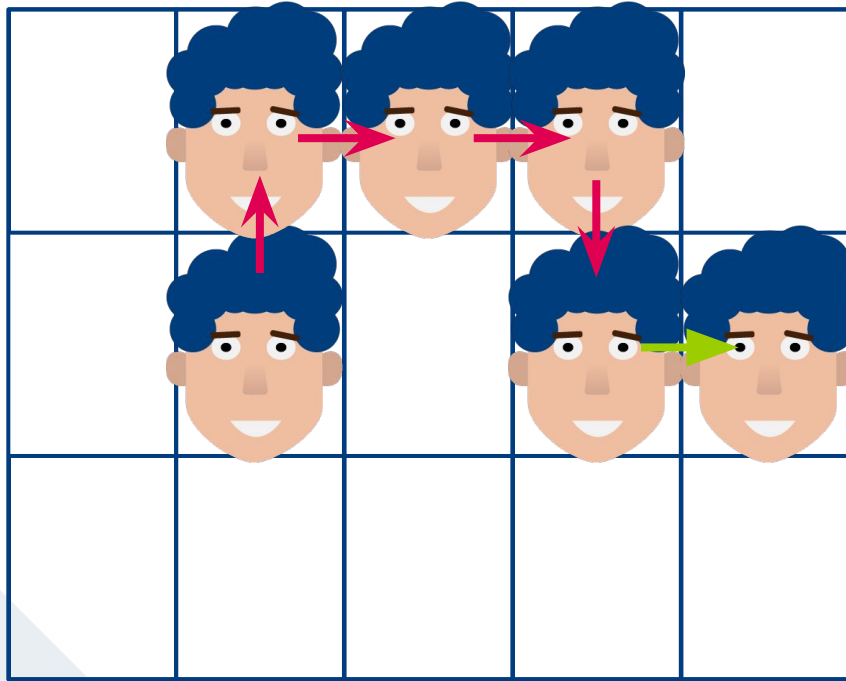
Rewards

$$R_t = r(s_t, a_t)$$





# Buchhaltung



Aktion



Reward gesamt

$$R_0 = -1$$

$$R_0 + R_1 = -2$$

$$R_0 + R_1 + R_2 = -3$$

$$R_0 + \dots + R_3 = -4$$

$$R_0 + \dots + R_4 = -3$$

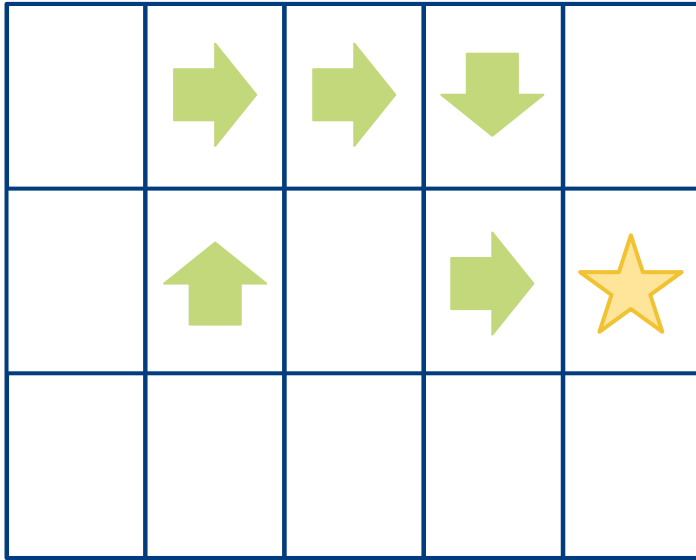
Ziel: maximiere gesammelten Reward

# Formalisierung

$$\begin{aligned} \underline{G_t} &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \\ \text{Return ab Zeitpunkt } t &= \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \quad 0 \leq \underline{\gamma} \leq 1 \\ &\quad \underbrace{\hspace{10em}} \\ &= r(s_{t+k+1}, a_{t+k+1}) \end{aligned}$$

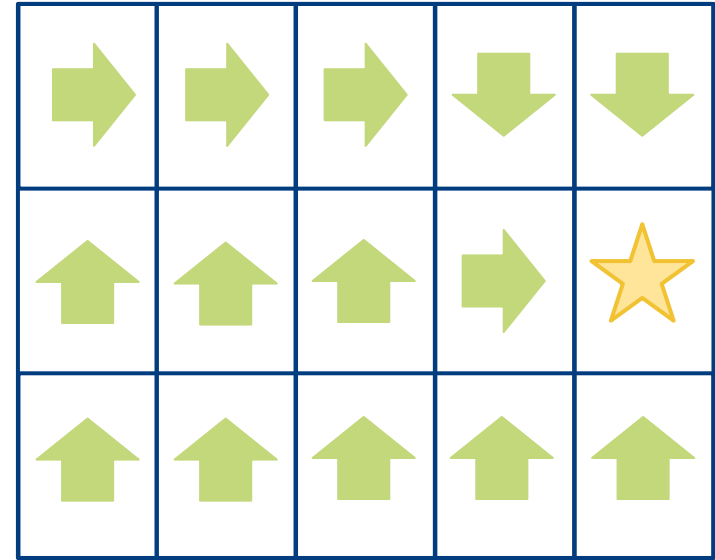
Discount Factor

# Wie die nächste Aktion auswählen?



Plan

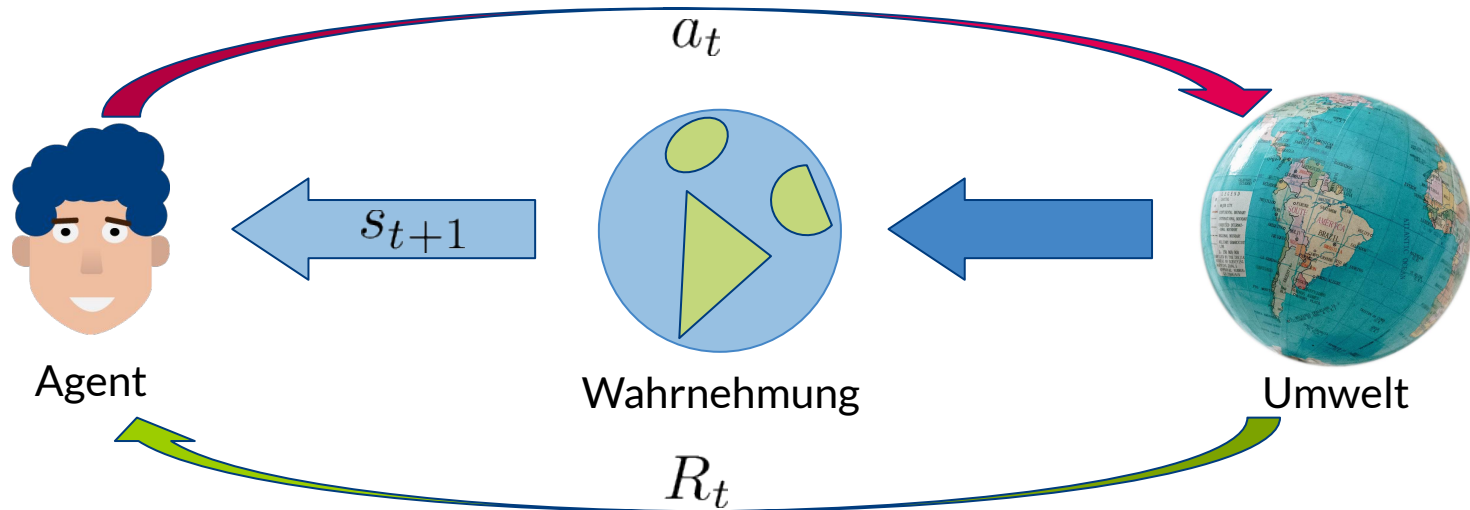
$a_1, a_2, \dots$



Policy

$a = \pi(s)$

# Modell für Zustandsübergänge



Modell für Statusübergang:  $P(s_{t+1} | a_t, s_t, \dots, a_0, s_0)$

... und Reward:  $P(s_{t+1}, R_t | a_t, s_t, \dots, R_0, a_0, s_0)$

# Aktionen ändern die Umwelt

## Transition probabilities $P(s|a)$

Beispiel: Geradeaus gehen



$P(s|a)$  eher groß



$P(s|a)$  eher klein

# Transition probabilities $P(s|a)$

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



# Markov-Annahme

Ein stochastischer Prozess ist *Markov'sch*, wenn der aktuelle Zustand nur vom vorherigen Zustand abhängt:

$$P(x_t | x_{t-1}, \dots, x_0) = P(x_t | x_{t-1})$$

Zustandsübergangs-und-Reward-Modell:

$$P(s_{t+1}, R_t | a_t, s_t, \dots, R_0, a_0, s_0) = P(s_{t+1}, R_t | a_t, s_t)$$

# Markov Decision Process (MDP)

Formale Beschreibung der Interaktion im RL

States  $s \in \mathcal{S}$

Actions  $a \in \mathcal{A}$

Time  $t$

Model  $P(s_{t+1}, R_t | a_t, s_t)$

Reward  $r(s, a)$

## Markov-Annahme:

Zustandsübergang und Reward  
hängen nur von vorherigen  
Zustand und Aktion ab





# Policy $\pi$

Eine Policy definiert das Agenten-Verhalten für **alle** Zustände  $s$

Deterministisch:  $a = \pi(s)$

↓	↓	↓	↓	↓
↓	↓	↓	↓	x
→	→	→	→	↑

Stochastisch:  $a \sim P_{\pi}(a|s)$

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

# Wie findet ein Agent eine gute Policy?

Reminder: Agent will Return  $G_t$  maximieren

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


Kurzsichtiger Agent:

$$\begin{aligned} \gamma &:= 0 \rightsquigarrow G_t = R_{t+1} = r(s_{t+1}, a_{t+1}) \\ &\Rightarrow \pi(s_t) = \arg \max_a r(s_{t+1}, a) \end{aligned}$$

# Kurzsichtiger Agent

... wählt eine Aktion, die den nächsten Reward maximiert

State  $s$

				X

Rewards

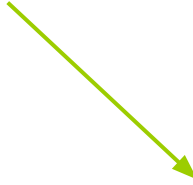
-1	-1	-1	-1	-1
-1	-1	-1	-1	0
-1	-1	-1	-1	-1

# Bessere Strategie

Wähle Policy  $\pi$ , die den *erwarteten* Return maximiert:

$$\pi^{\star} = \arg \max_{\pi} \mathbb{E}_{\pi} [G_t]$$

mit  $G_t = \sum_{k=0}^{\infty} \gamma^k r(s_{t+k+1}, \pi(s_{t+k}))$

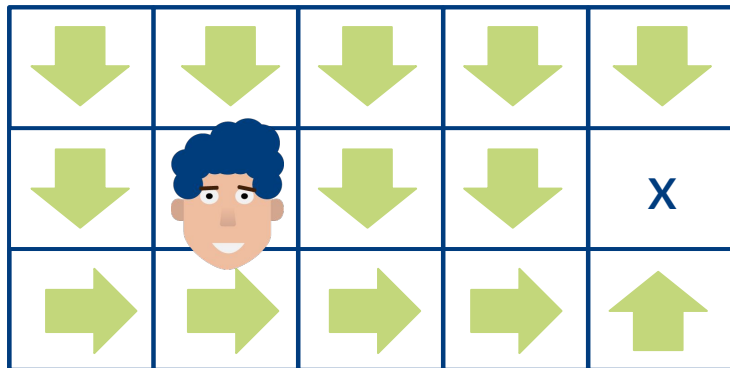

$$\mathbb{E}[x] = \sum_{x \in \mathcal{X}} P(x) \cdot x$$

# State-value function

Welcher State verspricht größten Return?

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

State  $s$  und Policy  $\pi$



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

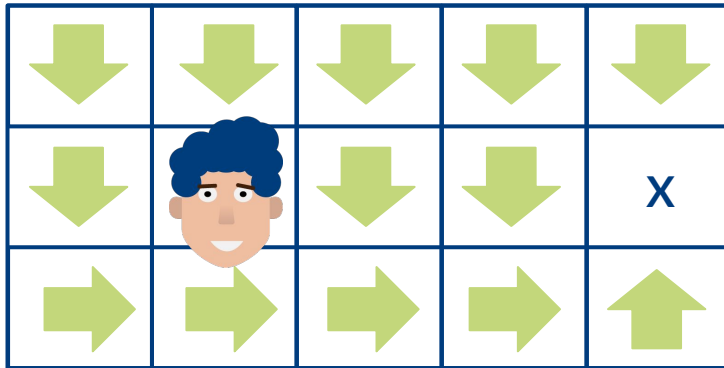
-7	-6	-5	-4	-1
-6	-5	-4	-3	X
-5	-4	-3	-2	-1

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

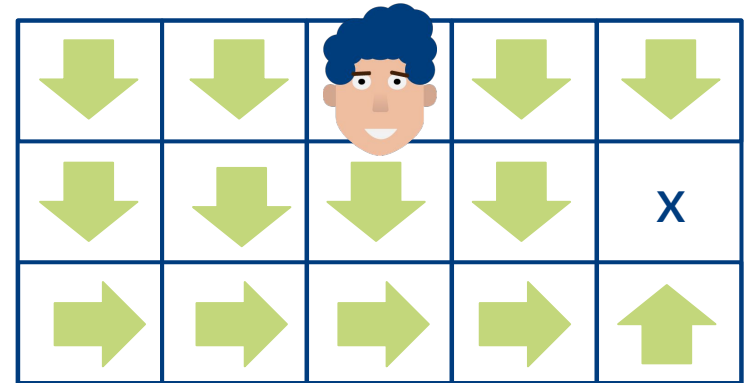
# State-value function

Welcher Zustand ist besser?

State 1

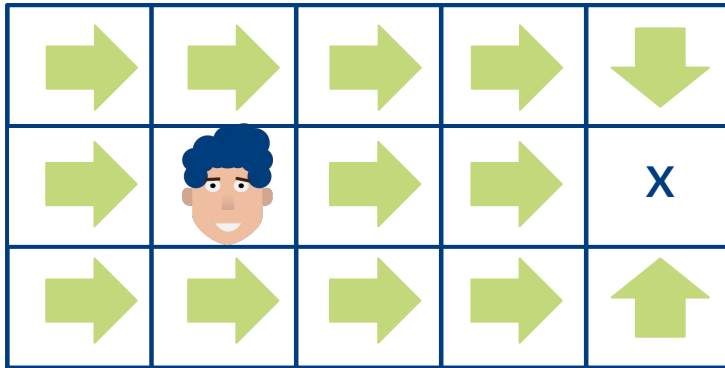


State 2

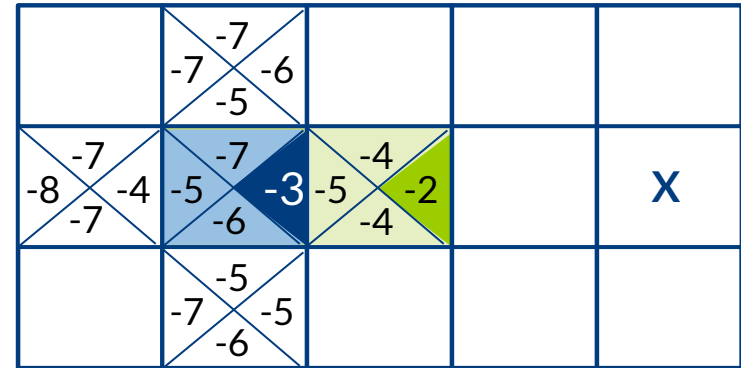


# Action-value function $q_{\pi}(s,a)$

State  $s$  und Policy  $\pi$



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 $\pi^*$



-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:

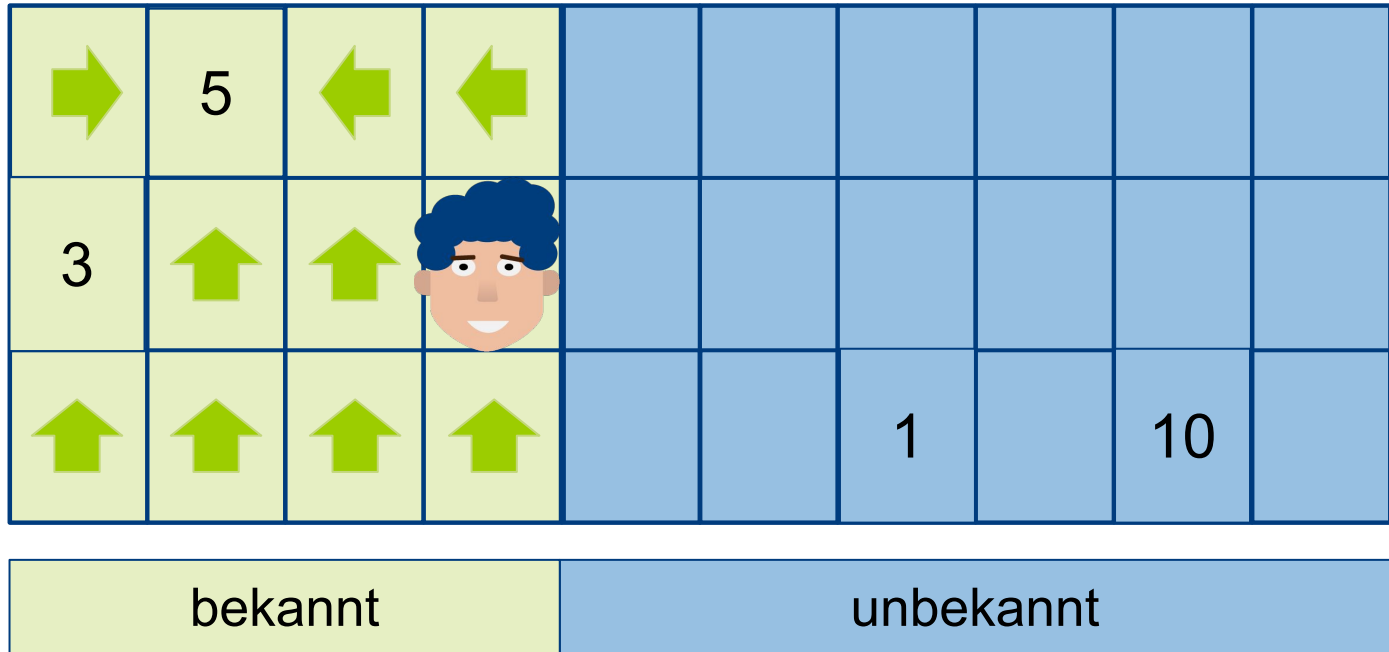
$$\pi^* \geq \pi, \forall \pi$$

Was bedeutet besser?

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

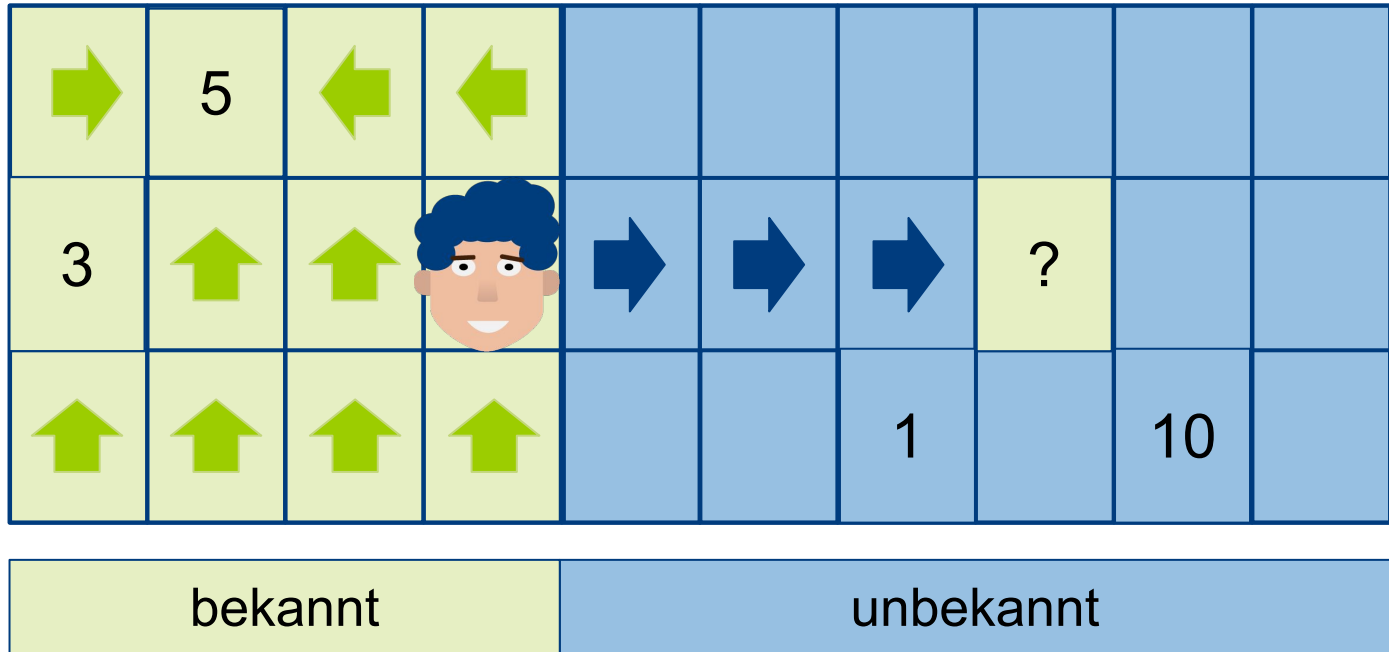
# Exploitation

Maximierung des Rewards gg. bekannter Information

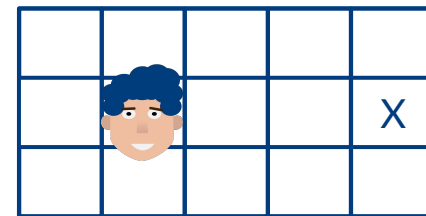


# Exploration

Erschließung neuer, unbekannter Bereiche



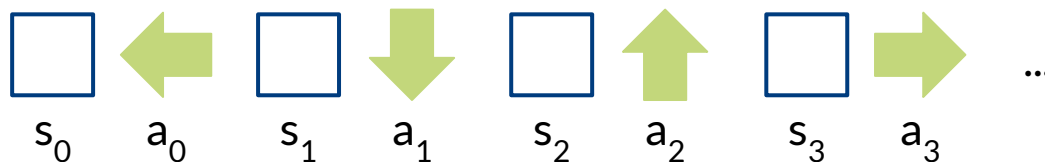
# Monte Carlo Methods



Random Policy  $\pi$

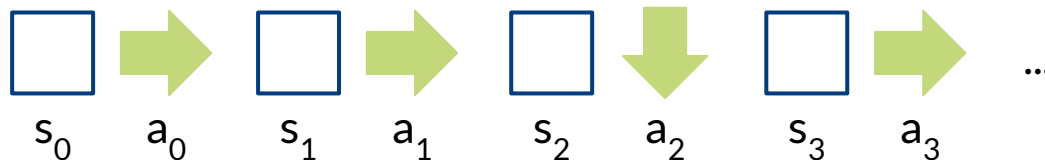
## Episode 1

Score:  
-4



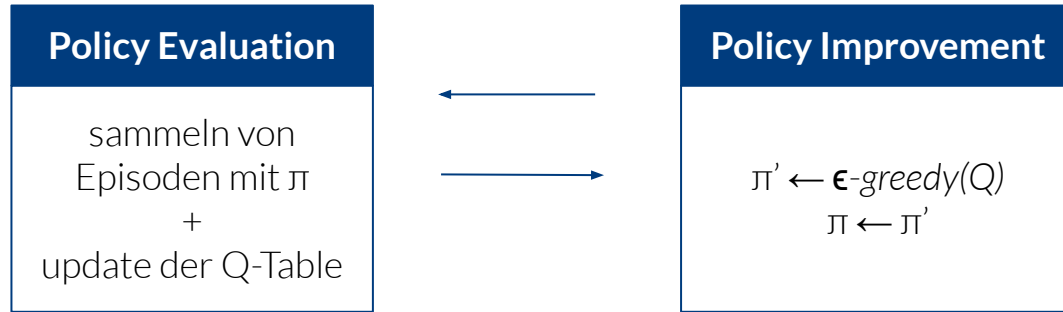
## Episode 2

Score:  
-7



...

# Monte Carlo Prediction



$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(G_t - Q(S_t, A_t))$$

alternative Schätzung      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.,  $\epsilon$ -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

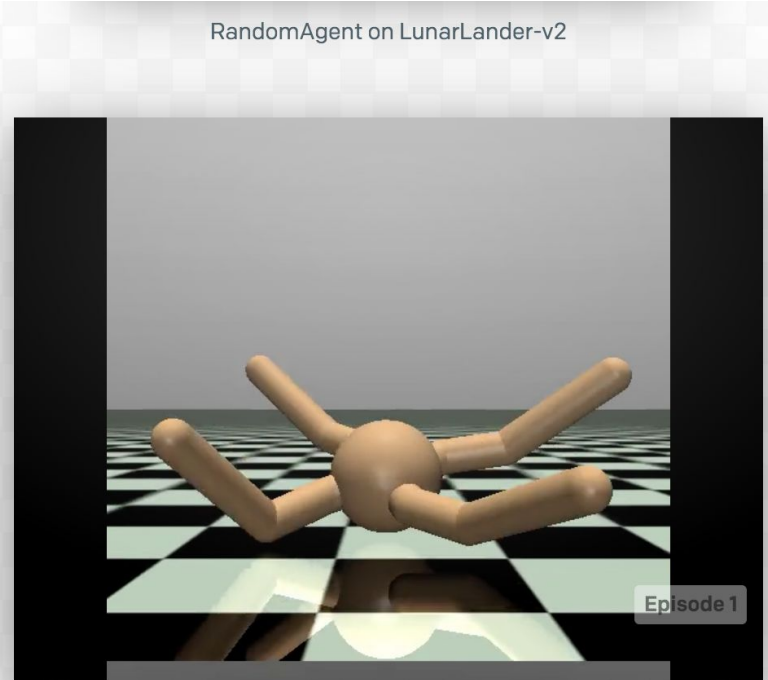


## Gym

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()
```

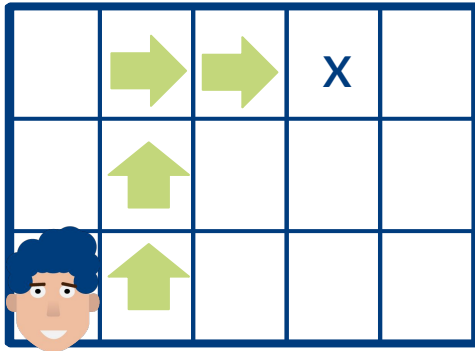


# Aufgabe 1: Einstieg in RL mit MENACE

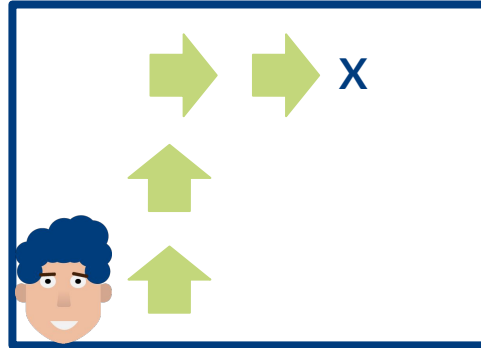
- › **Jupyter Lab Notebook**

# Zustands- und Aktionsräume

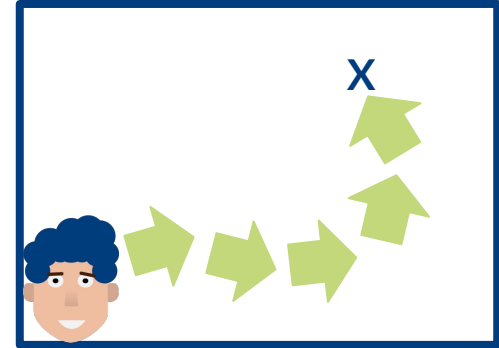
Wie unterscheiden sich diese Environments?



Zustände & Aktionen  
diskret



Zustände kontinuierlich &  
Aktionen diskret



Zustände & Aktionen  
kontinuierlich

# Aufgabe 2: CartPole Gym mit Q-Learning

- › **Jupyter Lab Notebook**

# 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/4UeEVTWmHGpRGYkm9>



# Vielen Dank

Matthias Richter

[matthias.richter@inovex.de](mailto:matthias.richter@inovex.de)

Tim Bossenmaier

[tim.bossenmaier@inovex.de](mailto:tim.bossenmaier@inovex.de)

