

HUMBOLDT-UNIVERSITÄT ZU BERLIN

Institut für Informatik
Lehrstuhl Wissensmanagement



Reinforcement-Lernen

Tobias Scheffer
Steffen Bickel

Überblick

- Grundbegriffe: Agent, Umgebung/Prozess, Zustand, Aktion, Reinforcement, Steuerung.
- Welche Zielfunktion wird beim Lernen optimiert?
- Mathematisches Modell der Prozesse: Markov-Prozesse.
- Langfristiger Nutzen von Zuständen: Bellmann-Gleichung.

Im Mittelpunkt steht...

- Ein **Agent**, der ein **Ziel** verfolgen möchte (z.B. durch **Bewertungsfunktion** definiert),
- Der **Zustände beobachten** kann und
- Der **Aktionen** ausführen kann, die den Zustand beeinflussen.

... Im Gegensatz zum Klassifikationslernen

- Dort kann eine Zielfunktion, die approximiert werden soll, schon durch **Beispiele** beobachtet werden.
- Ein Klassifikator muss „nur“ die Beispiele imitieren.
- Unser Reinforcement-Agent sieht keine Beispiele für das Steuerungsverhalten, das er realisieren soll.
- Er hat nur **Performance-Feedback**.

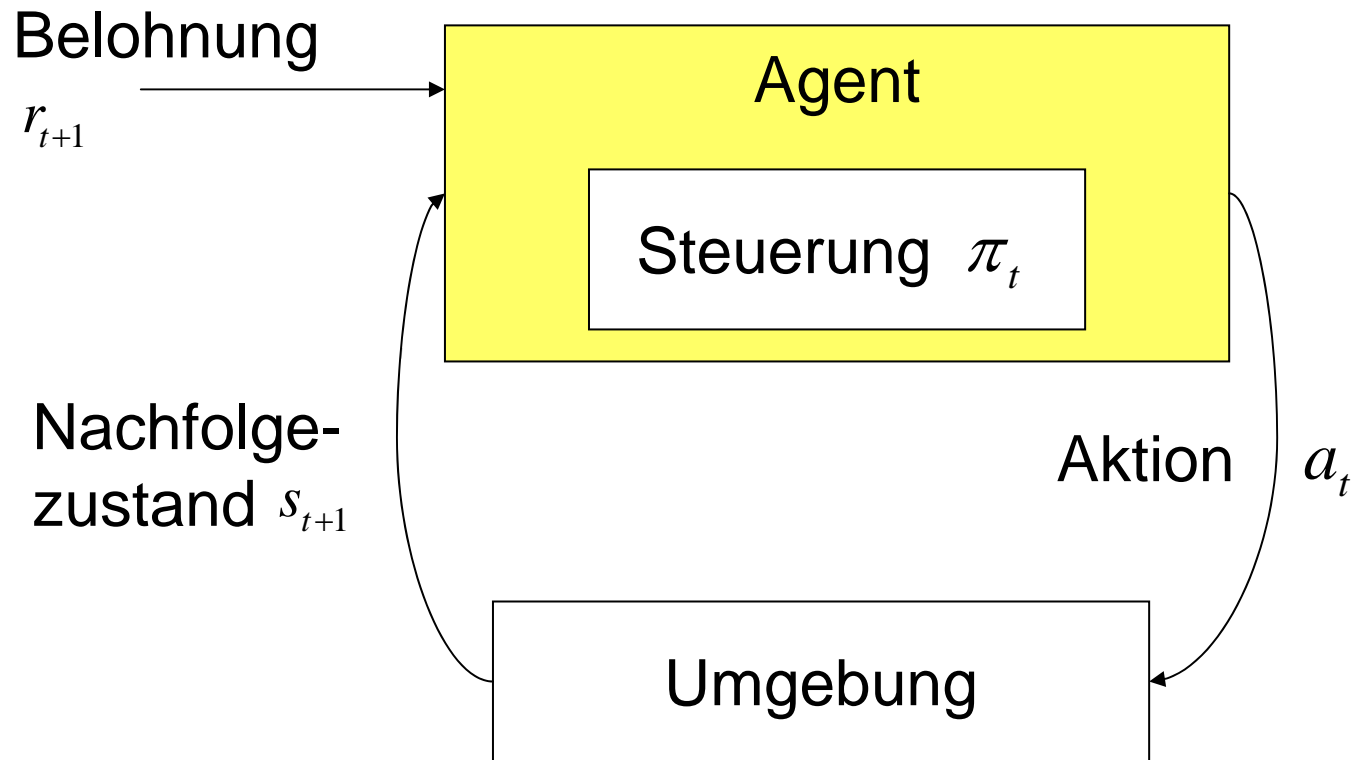
Grundelemente

- Ein **Agent** beobachtet eine **Umgebung** (oder einen **Prozess**).
- In jedem **Zeitschritt** t nimmt er einen **Zustand** s_t wahr.
- Für diesen Zustand wählt er eine **Aktion** a_t aus.
- Im nächsten Zeitschritt bekommt er eine **Belohnung** r_{t+1} (**Reinforcement**).

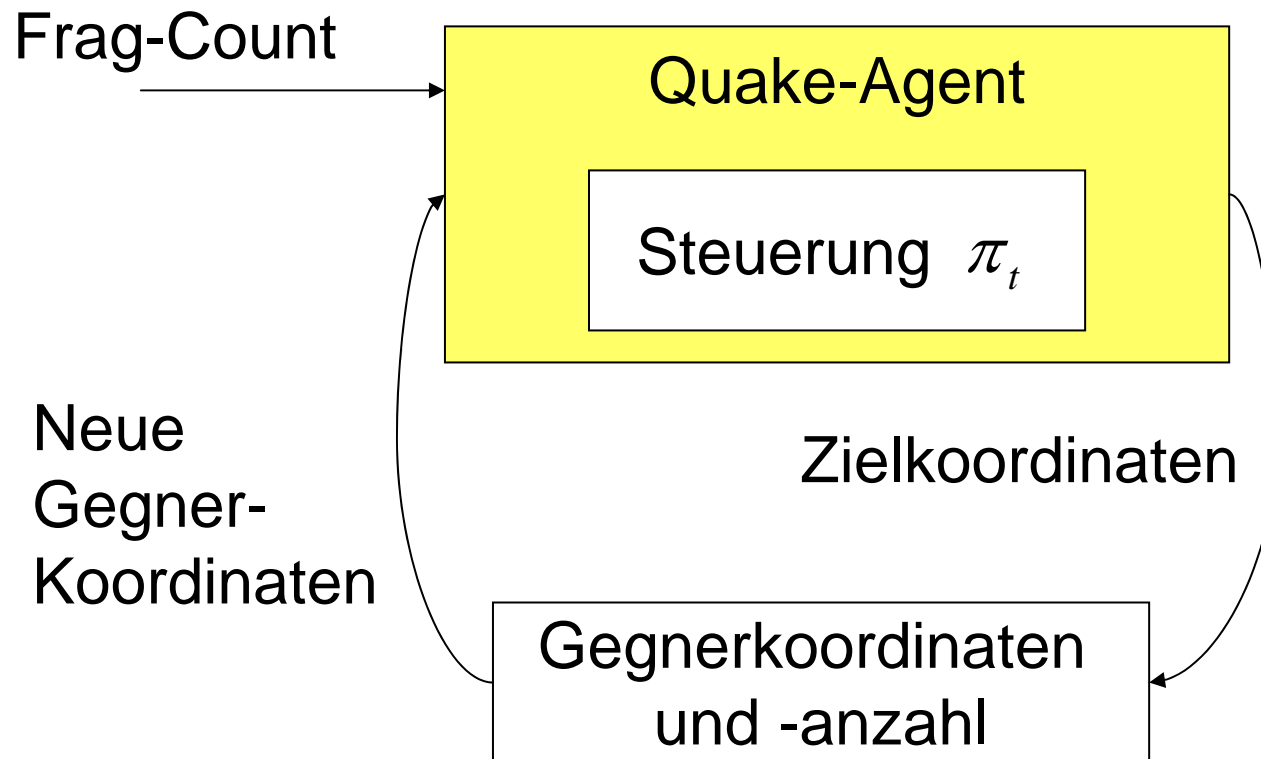
Grundelemente

- In jedem Zeitschritt kann der Agent seine **Steuerung** π_t verändern.
- Eine **Steuerung (Policy)** kann zufallsabhängig sein.
- $\pi_t(s, a)$ ist die Wahrscheinlichkeit dafür, dass der Agent im Zustand s die Aktion a wählt.

Reinforcement-Agent



Reinforcement-Agent



Ziel des Lernprozesses

- Agent bekommt Belohnungen r_1, r_2, r_3, \dots .
- Was genau soll der Agent maximieren?
- Episodische Prozesse:

$$R = \frac{1}{T} \sum_{t=1}^T r_t$$

- Kontinuierliche Prozesse:

$$R = r_1 + \gamma r_2 + \gamma^2 r_3 + \dots = \sum_{t=1}^{\infty} \gamma^t r_t$$

Umgebungen

- Wie können wir Umgebungen/Prozesse mathematisch beschreiben?
- Von jedem Zustand s und Aktion a gibt es Nachfolgezustände, in die der Prozess mit einer bestimmten Wahrscheinlichkeit übergehen kann.
- Auch die Belohnung kann vom Zufall abhängen.

$$\Pr[s_{t+1} = s', r_{t+1} = r' | s_t, a_t]$$

Markov-Prozesse

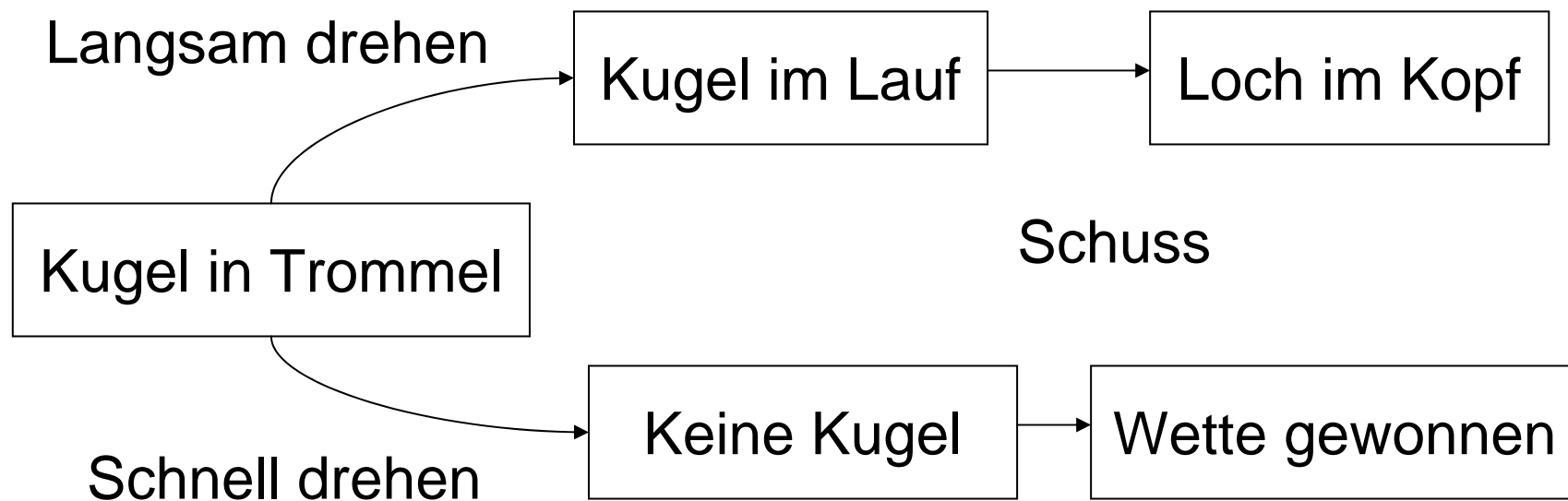
- Wenn die Wahrscheinlichkeit für einen Nachfolgezustand s_{t+1} und eine Belohnung $r_{t+1}(s, a)$
- Nur von Zustand s_t und Aktion a_t abhängt,
- Nicht aber davon, wie wir in den Zustand a_t gelangt sind,

$$\begin{aligned}\Pr[s_{t+1} = s', r_{t+1} = r' | s_t, a_t] \\ = \Pr[s_{t+1} = s', r_{t+1} = r' | s_t, a_t, \dots, s_1, a_1]\end{aligned}$$

- Dann heißt der Prozess Markov-Prozess

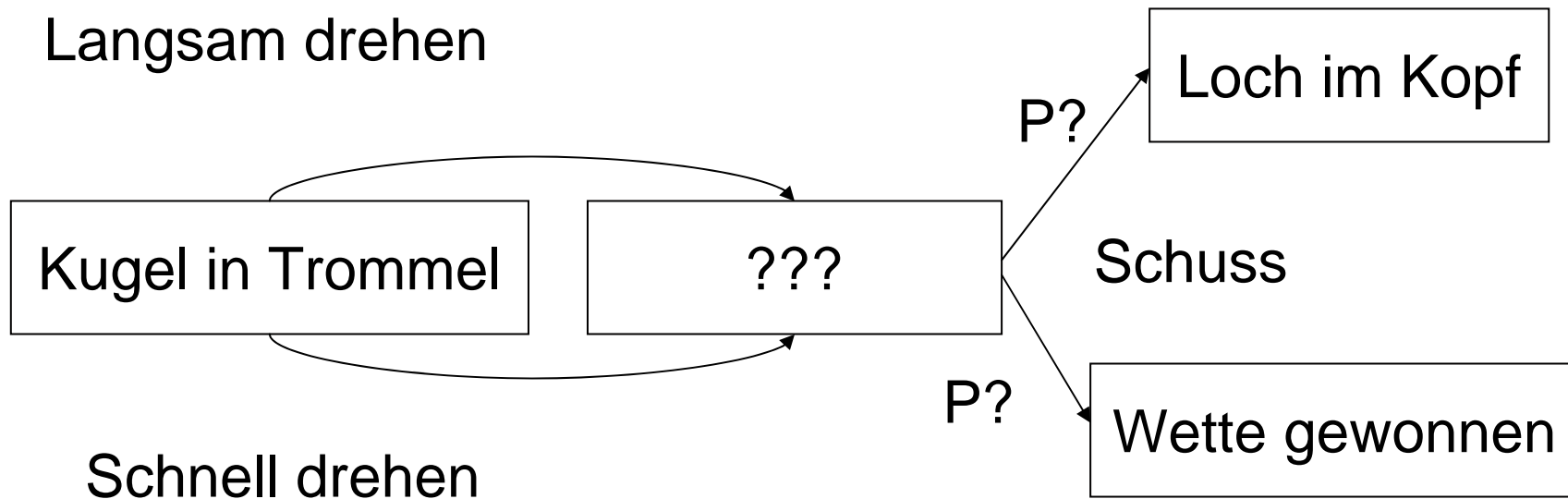
Markov-Prozesse

- Markov-Prozess: Wir wissen immer, in welchem Zustand wir sind.



Nicht-Markov-Prozesse

- Nicht-Markov-Prozess: Nicht alle Zustände können identifiziert werden.



Markov-Entscheidungsprozesse

- Ein Reinforcement-Lernproblem mit Markov-Eigenschaft heißt **Markov-Entscheidungsprozeß (Markov Decision Process, MDP)**
- Mit endlich vielen Zuständen: **Endlicher Markov-Entscheidungsprozess.**
- Ohne Markov-Eigenschaft: **Partiell beobachtbarer Markov-Entscheidungsprozess (Partially Observable Markov Decision Process, POMDP)**

Wie gut ist Zustand s (langfristig)?

- Wie gut ist es im Mittel, bei einer Policy π im Zustand s zu sein?

$$\begin{aligned} V^\pi(s) &= E[R_t | s_t = s] = E\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\right] \\ &= E[r_{t+1} + \gamma R_{t+1} | s_t = s] \\ &= \sum_{s'} \Pr[s_{t+1} = s' | s_t = s, a_t = \pi(s)] \\ &\quad \left[E[r_{t+1} | s_t = s, a_t = \pi(s), s_{t+1} = s] + \gamma E[R_{t+1} | s_{t+1} = s'] \right] \\ &= \sum_{s'} \Pr[s_{t+1} = s' | s_t = s, a_t = \pi(s)] \\ &\quad \left[E[r_{t+1} | s_t = s, a_t = \pi(s), s_{t+1} = s'] + \gamma V^\pi(s') \right] \end{aligned}$$

Bellmann-Gleichung

- Value-Funktion $V^\pi(s)$: langfristiger Nutzen eines Zustandes s .
- Wird durch rekursive Bellmann-Gleichung beschrieben.

$$V^\pi(s) = \sum_{s'} \Pr[s_{t+1} = s' | s_t = s, a_t = \pi(s)] \left[E[r_{t+1} | s_t = s, a_t = \pi(s), s_{t+1} = s] + \gamma V^\pi(s') \right]$$

- Wenn die Lösung für ein Problem bekannt ist, dann lässt sich schnell eine optimale Steuerung finden.

Bellmann-Gleichung

- Problem: Es müssen gleichzeitig
- $V^\pi(s)$ geschätzt und
- π zum Maximieren von $V^\pi(s)$ festgelegt werden.
- Beides hängt aber voneinander ab.
- $\Pr[s_{t+1} = s', r_{t+1} = r' | s_t, a_t]$ und $r(s,a)$ müssen bekannt sein (wir brauchen ein Modell des Prozesses).

Value-Iteration-Algorithmus

- Beginne mit $V(s) = \max_a r(s, a)$ für alle s .
- Solange V sich noch ändert

- ◆ Für alle Zustände s ,

$$V(s) \leftarrow \max_a \left\{ r(s, a) + \sum_{s'} \Pr[s' | s, a] \gamma V(s') \right\}$$

- Optimale Policy:

$$\pi(s) = \arg \max_a \left\{ r(s, a) + \sum_{s'} \Pr[s' | s, a] \gamma V(s') \right\}$$

Beispiel

start			$r=-100$
			$r=100$

Q-Learning

- Der Value-Iteration-Algorithmus kann nur angewandt werden, wenn ein Prozessmodell existiert.
- ($\Pr[s_{t+1} = s', r_{t+1} = r' | s_t, a_t]$ und $r(s,a)$ müssen bekannt sein).
- Q-Learning funktioniert „durch ausprobieren“.
- Q-Funktion: Erwartete langfristige Belohnung, wenn wir im Zustand s Aktion a ausführen.
$$Q(s, a) = r(s, a) + \gamma \sum_{s'} \Pr[s' | s, a] \max_{a'} Q(s', a')$$
- Kann wie folgt geschätzt werden.

Q-Learning-Algorithmus

- Initialisiere Tabelle $Q(s,a)$ auf 0.
- Solange sich die Q-Tabelle noch ändert
 - ◆ Beobachte Zustand s
 - ◆ Wähle Aktion a aus (halb-zufällig)
 - ◆ Beobachte $r(s,a)$ (kurzfristige Belohnung) und Nachfolgezustand s' .
 - ◆ Aktualisiere $Q(s,a) \leftarrow r(s,a) + \gamma \max_{a'} Q(s',a')$
 - ◆ $s \leftarrow s'$
- Optimale policy: $\pi^*(s) = \arg \max_a Q(s,a)$

Beispiele

- Backgammon: TD-Gammon spielt auf Weltmeisterniveau.
- Prozesssteuerung: Steuerung biochemischer Prozesse
- Robotik: Steuerung mobiler Roboter.

Zusammenfassung

- Reinforcement-Agent beobachtet Zustände und reagiert mit Aktionen.
- Ziel ist eine Steuerung (Policy) π ,
- die die Reinforcement-Funktion R für einen Markov-Prozess maximiert.
- Dabei beschreibt die Bellmann-Gleichung den langfristigen Nutzen $V^\pi(s)$ eines Zustandes bei gegebener Policy π
- Prozessmodell bekannt: Value-Iteration-Algorithmus.
- Unbekanntes Prozessmodell: Q-Learning.