

# Maschinelles Lernen I - Grundverfahren

## V08 Grundlagen des Reinforcement Learning



Rec

Sommersemester 2018/2019

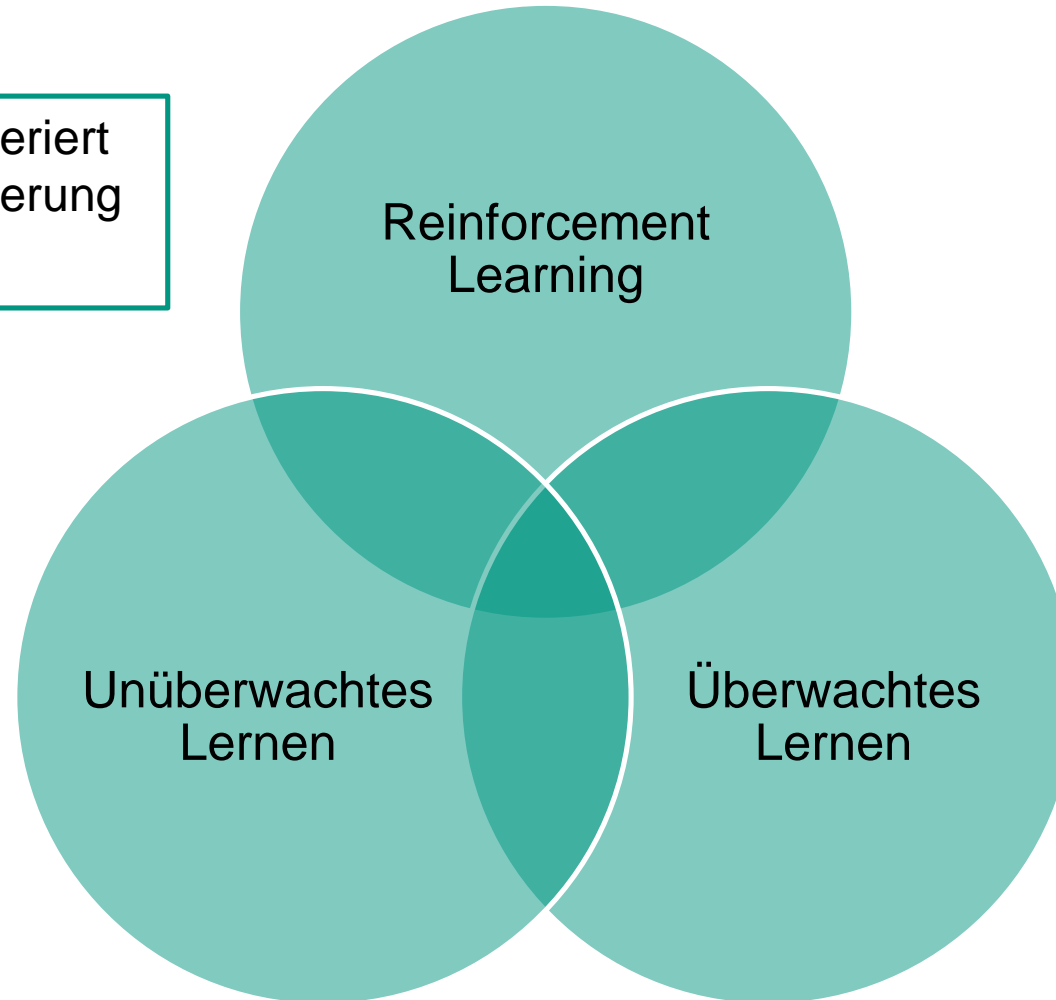
Prof. Dr. J.M. Zöllner, Karam Daaboul & Karl Kurzer

INSTITUT FÜR ANGEWANDTE INFORMATIK UND FORMALE BESCHREIBUNGSVERFAHREN  
INSTITUT FÜR ANTHROPOMATIK UND ROBOTIK



# Einordnung von Reinforcement Learning

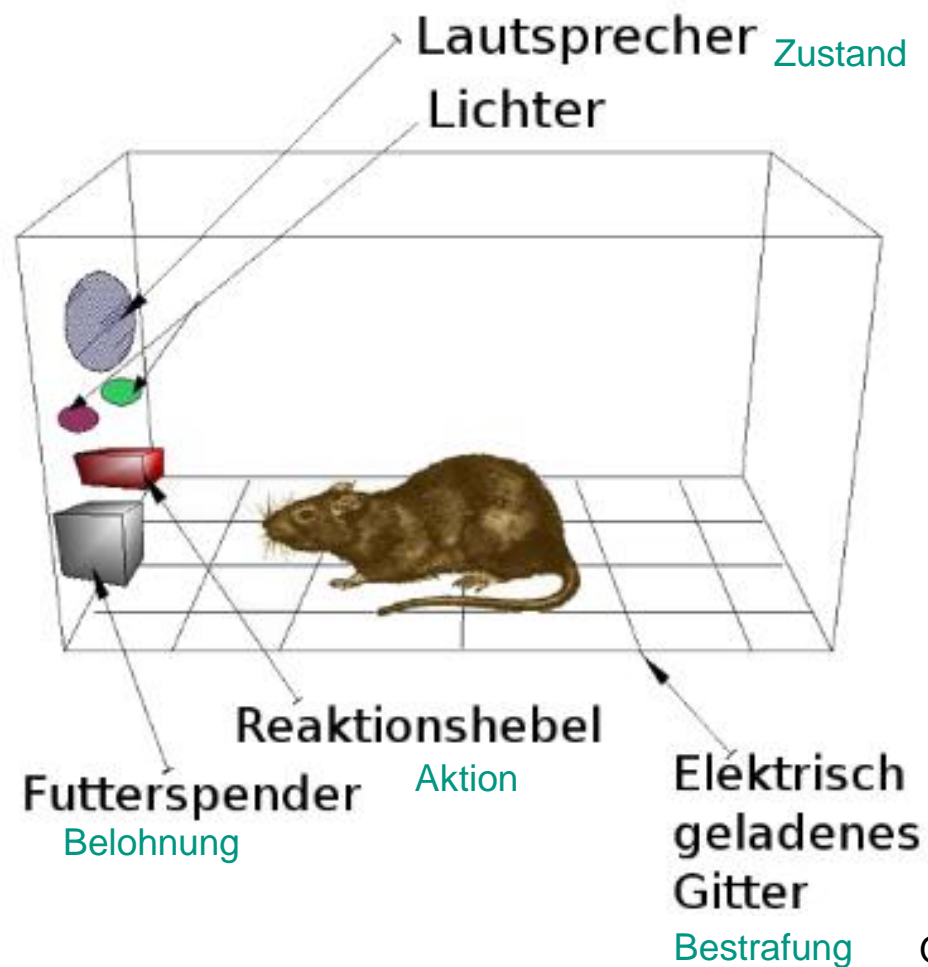
Daten werden generiert  
Belohnungsmaximierung  
 $\max_{\tau} R(\tau)$



Daten sind nicht annotiert  
Strukturapproximation  
 $x \approx x'$

Daten sind annotiert  
Funktionsapproximation  
 $x \rightarrow y$

# Lernen durch Belohnung/Bestrafung



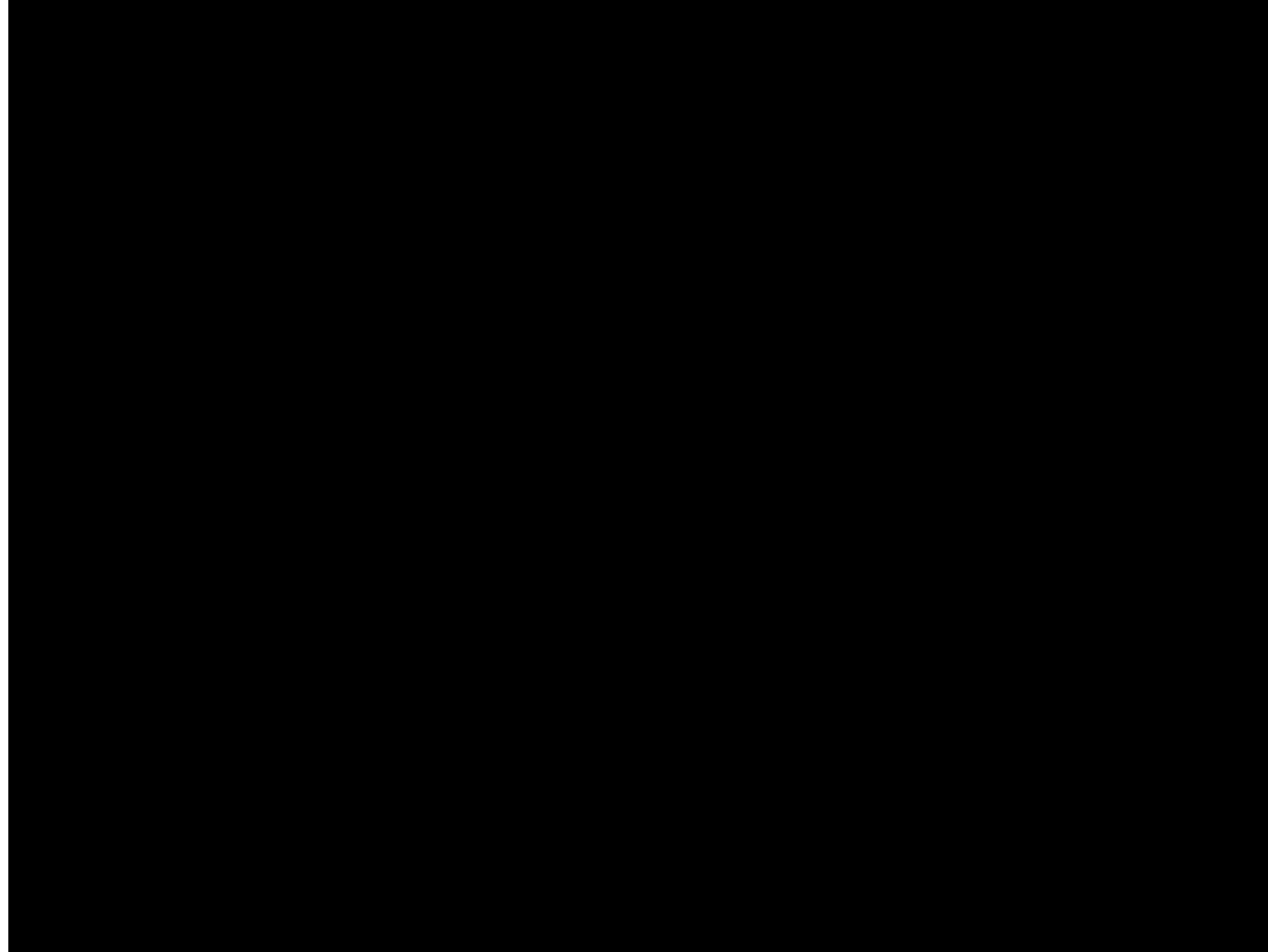
Quelle: Wikipedia

# Bedeutende Fortschritte

- Grandmaster level in StarCraft II using multi-agent reinforcement learning  
[Vinyals, 2019]
- Solving the Rubik's Cube Without Human Knowledge  
[McAleer, Agostinelli, Shmakov 2018]
- Mastering the game of Go without human knowledge  
[Silver, Schrittwieser, Simonyan 2017]
- Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm  
[Silver, Hubert, Schrittwieser 2017]
- Mastering the game of Go with deep neural networks and tree search  
[Silver, Huang 2016]
- Human-level control through deep reinforcement learning  
[Mnih, Kavukcuoglu, Silver 2015]

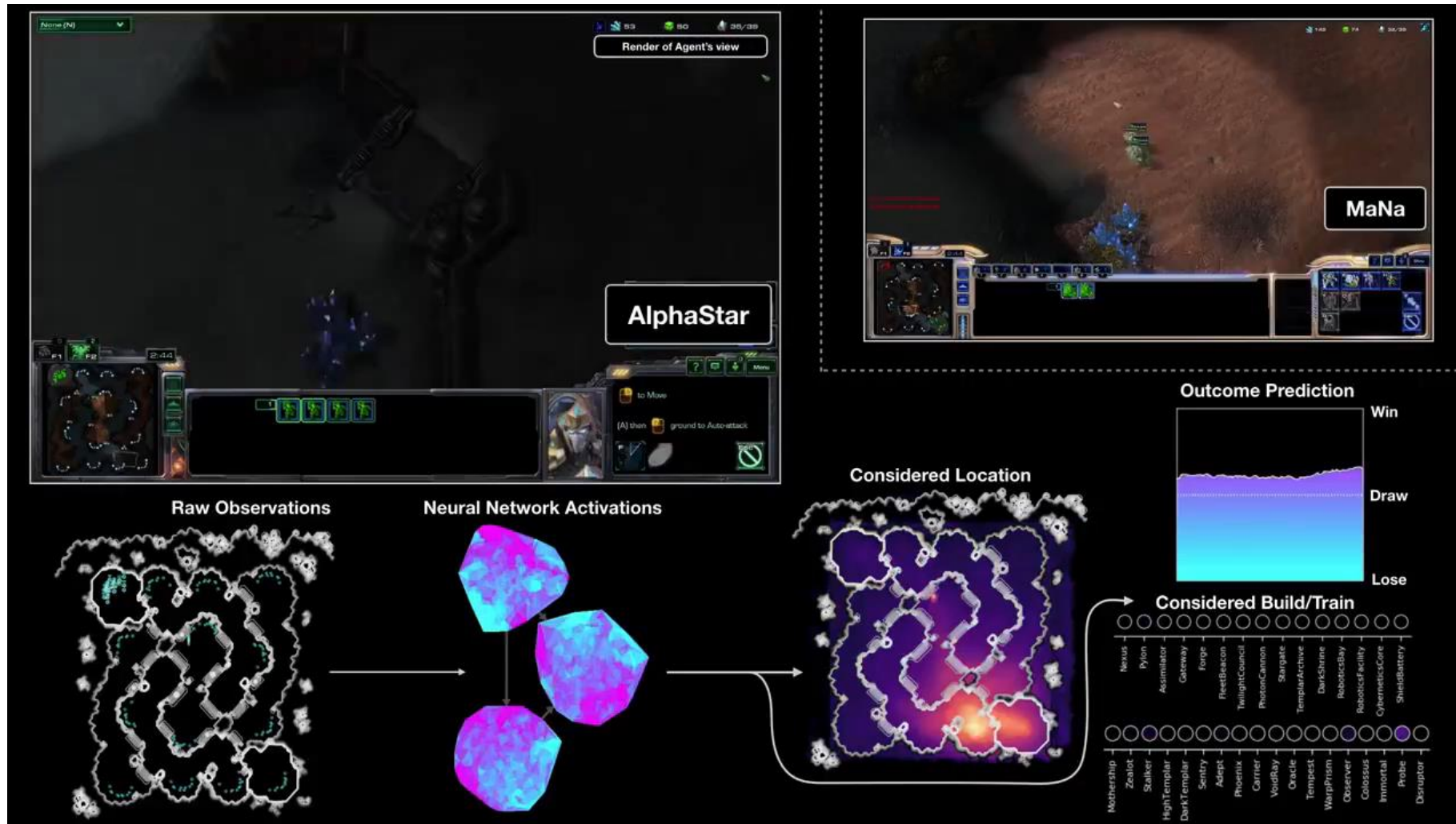
# Von Atari zu AlphaStar







# AlphaStar



# Reinforcement Learning (RL)

- Bestandteile des RL Problems
  - Markov'scher Entscheidungsprozess
- Grundlegende Lösungsverfahren
  - Dynamische Programmierung (DP)
  - Monte Carlo Learning
  - Temporal Difference Learning
- Reinforcement Learning Algorithmen
  - SARSA
  - Q-Learning
- DQN



# Markov'scher Entscheidungsprozess (MDP)

## ■ Formalisierung der sequentiellen Entscheidungsfindung

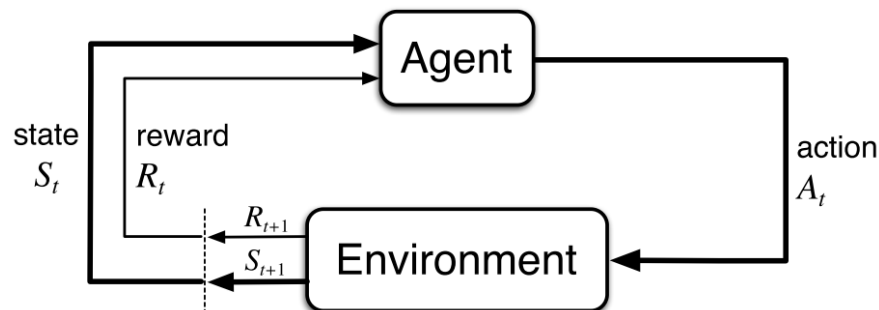
- $S$  ist ein endlicher Satz von Zuständen
- $A$  ist ein endlicher Satz von Aktionen
- $\mathcal{P}$  ist die Transitionswahrscheinlichkeitsmatrix

$$\mathcal{P}_{ss'}^a = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$$

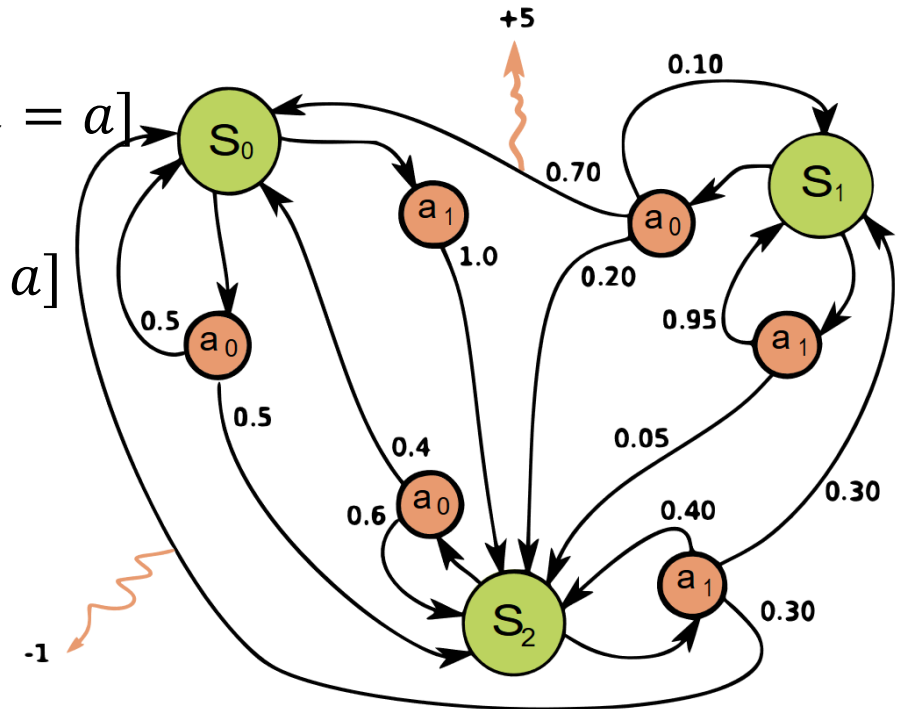
- $\mathcal{R}$  ist die Belohnungsfunktion

$$\mathcal{R}_s^a = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$$

- $\gamma$  ist der Diskontierungsfaktor  $\in [0,1]$



Quelle: Sutton2017



Quelle: Wikipedia

# Zustände und Beobachtungen

- Ein Zustand  $s$  ist eine vollständige Beschreibung des Zustands der Welt.
  - Es gibt keine Informationen, die dem Zustand verborgen bleiben.
  - Welt vollständig beobachtbar
  
- Eine Beobachtung  $o$  ist eine unvollständige Beschreibung des Zustands der Welt.
  - Es gibt Informationen, die der Beobachtung verborgen bleiben.
  - Welt unvollständig beobachtbar

Zustand



Beobachtung



# Markov Eigenschaft

## Definition

Ein Zustand erfüllt die Markov Eigenschaft wenn und nur wenn:

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

- Die Zukunft ist unabhängig von der Vergangenheit, wenn man die Gegenwart betrachtet

$$H_{1:t} \rightarrow S_t \rightarrow H_{t+1:\infty}$$

- Sobald der Zustand bekannt ist, kann die Historie verworfen werden

# Aktionen

- Unterschiedliche Umgebungen ermöglichen unterschiedliche Arten von Aktionen.
- Die Menge aller gültigen Aktionen in einer bestimmten Umgebung wird als Aktionsraum bezeichnet.
- Aktionsräume können diskret oder kontinuierlich sein

# Diskontierung

- Die meisten Markov Entscheidungsprozesse werden diskontiert! Warum?
  - Mathematisch praktisch
  - Vermeidet unendliche Belohnungen in zyklischen Markovprozessen
  - Impliziert Unsicherheit über die Zukunft
  - Verhalten von Tier und Mensch zeigt Präferenzen für sofortige Belohnungen

# Modelle

- Ein Modell sagt voraus, was die Umgebung als nächstes tun wird
- $\mathcal{P}$  prognostiziert den nächsten Zustand:

$$\mathcal{P}_{ss'}^a = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$$

- $\mathcal{R}$  prognostiziert die nächste (unmittelbare) Belohnung:

$$\mathcal{R}_s^a = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$$

# Belohnung (Reward)

- Eine **Belohnung**  $R_t$  ist ein skalares Feedbacksignal
- Beschreibt die Güte der Aktion zum Zeitpunkt  $t$
- Der Agent versucht die kummulierte Belohnung zu maximieren
- Basis für Reinforcement Learning ist die **Belohnungshypothese**

## Definition

Ziele können als Maximierung der kummulierten Belohnung beschrieben werden.



# Strategie (Policy)

- Eine Strategie  $\pi$  beschreibt das Verhalten des Agenten
- Psychologisch gesehen: Reiz-Reaktions-Modell
  - Abhängig vom aktuellen Zustand

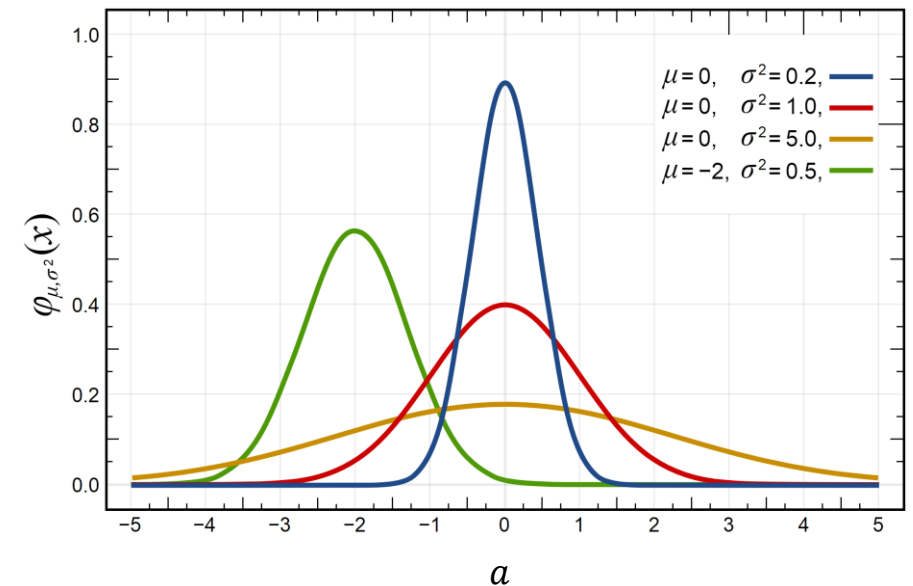
■ Deterministische Strategie:  $a = \pi(s)$

■ Stochastische Strategie:  $a \sim \pi(\cdot | s)$

■  $\pi(a|s) = \mathbb{P}[A_t = a | S_t = s]$

■ z.B. Gaussverteilung:  $a \sim N(\mu(s); \sigma(s)^2)$

Stochastische Strategie



# Zustandswertfunktion (State Value Function)

## V-Funktion $\rightarrow V(s)$

- Die Bewertungsfunktion ist eine Vorhersage der zukünftigen Belohnung
- Bewertet die Güte eines Zustands

$$V_{\pi}(s) = \sum_{a \in A} \pi(a|s) \left( \mathcal{R}_s^a + \gamma \sum_{s' \in S} \mathcal{P}_{ss'}^a V_{\pi}(s') \right)$$

### Definition

Die Zustandswertfunktion eines MDP ist die erwartete kummulierte Belohnung ausgehend vom Zustand  $s$  wenn im folgenden die Strategie  $\pi$  verfolgt wird.

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

# Aktionswertfunktion (Action Value Function)

## Q-Funktion $\rightarrow Q(s, a)$

- Die Bewertungsfunktion ist eine Vorhersage der zukünftigen Belohnung
- Bewertet die Güte einer Aktion in einem Zustand

$$Q_{\pi}(s, a) = \mathcal{R}_s^a + \gamma \sum_{s' \in S} \mathcal{P}_{ss'}^a \sum_{a' \in A} \pi(a'|s') Q_{\pi}(s', a')$$

### Definition

Die Aktionswertfunktion eines MDP ist die erwartete kummulierte Belohnung ausgehend vom Zustand  $s$  und der Aktion  $a$  wenn im folgenden die Strategie  $\pi$  verfolgt wird.

$$Q_{\pi}(s, a) = \mathbb{E}_{\pi}[R_{t+1} + \gamma Q_{\pi}(S_{t+1}, A_{t+1}) | S_t = s, A_t = a]$$

# Optimale Wertfunktionen

- Die optimale Zustandswertfunktion ist der Maximalwert über alle Strategien

$$V^*(s) = \max_{\pi} V_{\pi}(s)$$

- Die optimale Aktionswertfunktion ist der Maximalwert über alle Strategien

$$Q^*(s, a) = \max_{\pi} Q_{\pi}(s, a)$$

- Die optimale Wertfunktion gibt den bestmöglichen Wert eines MDP an
- Ein MDP ist gelöst, wenn die optimale Wertfunktion bekannt ist

# Bellman Optimalitätsgleichung

- Bellman Optimalitätsgleichung für die Zustandswertfunktion

$$V^*(s) = \max_a \left( \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a V^*(s') \right)$$

- Bellman Optimalitätsgleichung für die Aktionswertfunktion

$$Q^*(s, a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \max_{a'} Q^*(s', a')$$

# Reinforcement Learning (RL)

- Bestandteile des RL Problems

  - Markov'scher Entscheidungsprozess

- Grundlegende Lösungsverfahren

  - Dynamische Programmierung (DP)

  - Monte Carlo Methoden

  - Temporal Difference Learning

- Reinforcement Learning Algorithmen






















  - SARSA

  - Q-Learning

- DQN

# Beispiel – Grid World

## Initiale Strategie

V: 0.00 	V: 0.00 	V: 0.00 	V: 0.00 	V: 0.00 
V: 0.00 	V: 0.00 			V: 0.00 R: -0.2 
V: 0.00 	V: 0.00 R: -0.2 		V: 0.00 R: 1.0 	V: 0.00 
V: 0.00 	V: 0.00 		V: 0.00 	V: 0.00 
V: 0.00 	V: 0.00 	V: 0.00 	V: 0.00 	V: 0.00 

- Zustandsraum  $S \in [0, 24]$
- Aktionsraum  $a(s) \forall S \in [hoch, runter, links, rechts]$
- Transitionswahrscheinlichkeitsmatrix  $\mathcal{P}_{ss}^a$  ist deterministisch
- $\mathcal{R}$  die Belohnungsfunktion  $\mathcal{R}_s^a$  ist deterministisch
- Belohnung bei Ausführung der Aktion in Zustand
- Diskontierungsfaktor  $\gamma = 0,9$



# Dynamische Programmierung (DP) – Policy Iteration

$\hat{\pi}^*$  Schätzung von  $\pi^*$

Initialisiere  $V(s)$  und  $\hat{\pi}^*(s) \in A(s) \forall s \in S$  zufällig

## 1. Strategieevaluation

Wiederhole bis  $\Delta \leq \epsilon$

$\Delta \leftarrow 0$

Für alle  $s \in S$

$v \leftarrow V(s)$

$V_{\hat{\pi}^*}(s) \leftarrow \mathcal{R}_s^{\hat{\pi}^*(s)} + \gamma \sum_{s' \in S} \mathcal{P}_{ss'}^{\hat{\pi}^*(s)} V_{\hat{\pi}^*}(s')$

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

## 2. Strategieverbesserung

$\hat{\pi}^* \leftarrow \hat{\pi}^{*'}$

Für alle  $s \in S$








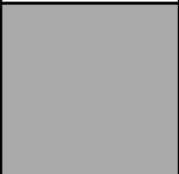
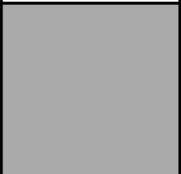
















$\hat{\pi}^{*'}(s) \leftarrow \arg \max_a \mathcal{R}_s^a + \gamma \sum_{s' \in S} \mathcal{P}_{ss'}^a V_{\hat{\pi}^*}(s')$

Wenn  $\hat{\pi}^* = \hat{\pi}^{*'}$ , dann terminiere













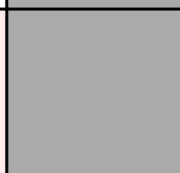




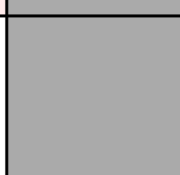







Sonst evaluiere  $\hat{\pi}^{*'} \rightarrow$  **Evaluierung Strategie**

# Policy Iteration

Initiale Strategie

V: 0.00 	V: 0.00 	V: 0.00 	V: 0.00 	V: 0.00 
V: 0.00 	V: 0.00 			V: 0.00 R: -0.2 
V: 0.00 	V: 0.00 R: -0.2 		V: 0.00 R: 1.0 	V: 0.00 
V: 0.00 	V: 0.00 		V: 0.00 	V: 0.00 
V: 0.00 	V: 0.00 	V: 0.00 	V: 0.00 	V: 0.00 






















Strategieevaluation (Schritt 1)

V: 0.00 	V: 0.00 	V: 0.00 	V: 0.00 	V: 0.00 
V: 0.00 	V: 0.00 			V: -0.20 R: -0.2 
V: 0.00 	V: -0.20 R: -0.2 		V: 1.00 R: 1.0 	V: 0.00 
V: 0.00 	V: 0.00 		V: 0.00 	V: 0.00 
V: 0.00 	V: 0.00 	V: 0.00 	V: 0.00 	V: 0.00 






















$$V_{\hat{\pi}^*}(s) \leftarrow \mathcal{R}_s^{\hat{\pi}^*(s)} + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^{\hat{\pi}^*(s)} V_{\hat{\pi}^*}(s')$$

# Policy Iteration

## Strategieevaluation (Schritt 2)

V: 0.00 	V: 0.00 	V: 0.00 	V: 0.00 	V: -0.09 
V: 0.00 	V: -0.05 			V: -0.26 R: -0.2 
V: -0.06 	V: -0.25 R: -0.2 		V: 1.00 R: 1.0 	V: 0.24 
V: 0.00 	V: -0.05 		V: 0.23 	V: 0.00 
V: 0.00 	V: 0.00 	V: 0.00 	V: 0.00 	V: 0.00 

## Strategieevaluation (Schritt 3)

V: 0.00 	V: -0.01 	V: 0.00 	V: -0.03 	V: -0.12 
V: -0.03 	V: -0.07 			V: -0.23 R: -0.2 
V: -0.07 	V: -0.29 R: -0.2 		V: 1.00 R: 1.0 	V: 0.22 
V: -0.03 	V: -0.07 		V: 0.28 	V: 0.14 
V: 0.00 	V: -0.01 	V: 0.00 	V: 0.07 	V: 0.00 

$$V_{\hat{\pi}^*}(s) \leftarrow \mathcal{R}_s^{\hat{\pi}^*(s)} + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^{\hat{\pi}^*(s)} V_{\hat{\pi}^*}(s')$$

# Policy Iteration

## Strategieverbesserung (Schritt 4)

V: 0.00 →	V: -0.01 ↔	V: 0.00 ↓	V: -0.03 ←	V: -0.12 ←
V: -0.03 ↑	V: -0.07 ↑			V: -0.23 R: -0.2 ↓
V: -0.07 ↕	V: -0.29 R: -0.2 ↕		V: 1.00 R: 1.0 ↕	V: 0.22 ←
V: -0.03 ↓	V: -0.07 ↓		V: 0.28 ↑	V: 0.14 ←
V: 0.00 →	V: -0.01 ↔	V: 0.00 →	V: 0.07 ↑	V: 0.00 ↑

## Strategieevaluation (Schritt 5)

V: -0.01 →	V: 0.00 ↔	V: 0.00 ↓	V: 0.00 ←	V: -0.02 ←
V: 0.00 ↑	V: -0.01 ↑			V: -0.00 R: -0.2 ↓
V: -0.03 ↕	V: -0.26 R: -0.2 ↕		V: 1.00 R: 1.0 ↕	V: 0.90 ←
V: 0.00 ↓	V: -0.01 ↓		V: 0.90 ↑	V: 0.25 ←
V: -0.01 →	V: 0.00 ↔	V: 0.06 →	V: 0.25 ↑	V: 0.13 ↑

$$V_{\hat{\pi}^*}(s) \leftarrow \mathcal{R}_s^{\hat{\pi}^*(s)} + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^{\hat{\pi}^*(s)} V_{\hat{\pi}^*}(s')$$

# Policy Iteration

Strategieevaluation (Schritt 6)

V: 0.00 →	V: -0.01 ↔	V: 0.00 ↓	V: 0.00 ←	V: 0.00 ←
V: -0.01 ↑	V: 0.00 ↑			V: 0.61 R: -0.2 ↓
V: 0.00 ↕	V: -0.21 R: -0.2 ↕		V: 0.99 R: 1.0 ↕	V: 0.90 ←
V: -0.01 ↓	V: 0.00 ↓		V: 0.90 ↑	V: 0.81 ←
V: 0.00 →	V: 0.02 ↔	V: 0.22 →	V: 0.81 ↑	V: 0.22 ↑

Strategieevaluation (Schritt 7)

V: -0.00 →	V: 0.00 ↔	V: 0.00 ↓	V: 0.00 ←	V: 0.00 ←
V: 0.00 ↑	V: -0.00 ↑			V: 0.61 R: -0.2 ↓
V: -0.01 ↕	V: -0.20 R: -0.2 ↕		V: 1.00 R: 1.0 ↕	V: 0.89 ←
V: 0.00 ↓	V: 0.02 ↓		V: 0.89 ↑	V: 0.81 ←
V: 0.02 →	V: 0.10 ↔	V: 0.73 →	V: 0.81 ↑	V: 0.73 ↑

$$V_{\hat{\pi}^*}(s) \leftarrow \mathcal{R}_s^{\hat{\pi}^*(s)} + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^{\hat{\pi}^*(s)} V_{\hat{\pi}^*}(s')$$

# Policy Iteration

Strategieevaluation (Schritt 19)

V: 0.06 →	V: 0.09 ↔	V: 0.13 ↘	V: 0.25 ↔	V: 0.55 ↓
V: 0.36 ↓	V: 0.30 ↓			V: 0.61 R: -0.2 ↓
V: 0.48 ↓	V: 0.33 R: -0.2 ↓		V: 1.00 R: 1.0 ↔	V: 0.90 ←
V: 0.53 ↘	V: 0.59 ↓		V: 0.90 ↑	V: 0.81 ↖
V: 0.59 →	V: 0.66 →	V: 0.73 →	V: 0.81 ↑	V: 0.73 ↖

Strategieverbesserung (Schritt 20)

V: 0.06 ↓	V: 0.09 ↓	V: 0.13 →	V: 0.25 →	V: 0.55 ↓
V: 0.36 ↓	V: 0.30 ←			V: 0.61 R: -0.2 ↓
V: 0.48 ↓	V: 0.33 R: -0.2 ↓		V: 1.00 R: 1.0 ↔	V: 0.90 ←
V: 0.53 ↘	V: 0.59 ↓		V: 0.90 ↑	V: 0.81 ↖
V: 0.59 →	V: 0.66 →	V: 0.73 →	V: 0.81 ↑	V: 0.73 ↖

$$V_{\hat{\pi}^*}(s) \leftarrow \mathcal{R}_s^{\hat{\pi}^*(s)} + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^{\hat{\pi}^*(s)} V_{\hat{\pi}^*}(s')$$

# Dynamische Programmierung (DP) – Value Iteration

$\hat{\pi}^*$  Schätzung von  $\pi^*$

Initialisiere  $V(s)$  und  $\hat{\pi}^*(s) \in A(s) \forall s \in S$  zufällig

## 1. Strategieevaluation

Wiederhole bis  $\Delta \leq \epsilon$

$\Delta \leftarrow 0$

Für alle  $s \in S$

$v \leftarrow V(s)$

$V_{\hat{\pi}^*}(s) \leftarrow \max_a (\mathcal{R}_s^a + \gamma \sum_{s' \in S} \mathcal{P}_{ss'}^a V^*(s'))$

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

## 2. Strategiebestimmung






















Für alle  $s \in S$

$\hat{\pi}^*(s) \leftarrow \arg \max_a \mathcal{R}_s^a + \gamma \sum_{s' \in S} \mathcal{P}_{ss'}^a V_{\hat{\pi}^*}(s')$
























# Value Iteration

Evaluation/Verbesserung (Schritt 1/2)

V: 0.00 	V: 0.00 	V: 0.00 	V: 0.00 	V: 0.00 
V: 0.00 	V: 0.00 			V: -0.20 R: -0.2 
V: 0.00 	V: -0.20 R: -0.2 		V: 1.00 R: 1.0 	V: 0.00 
V: 0.00 	V: 0.00 		V: 0.00 	V: 0.00 
V: 0.00 	V: 0.00 	V: 0.00 	V: 0.00 	V: 0.00 





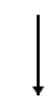


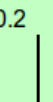












Evaluation/Verbesserung (Schritt 3/4)

V: 0.00 	V: 0.00 	V: 0.00 	V: 0.00 	V: 0.00 
V: 0.00 	V: 0.00 			V: -0.20 R: -0.2 
V: 0.00 	V: -0.20 R: -0.2 		V: 1.00 R: 1.0 	V: 0.90 
V: 0.00 	V: 0.00 		V: 0.90 	V: 0.00 
V: 0.00 	V: 0.00 	V: 0.00 	V: 0.00 	V: 0.00 








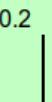













$$V_{\hat{\pi}^*}(s) \leftarrow \max_a \left( \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a V^*(s') \right)$$

# Value Iteration

Evaluation/Verbesserung (Schritt 5/6)

V: 0.00 	V: 0.00 	V: 0.00 	V: 0.00 	V: 0.00 
V: 0.00 	V: 0.00 			V: 0.61 R: -0.2 
V: 0.00 	V: -0.20 R: -0.2 		V: 1.00 R: 1.0 	V: 0.90 
V: 0.00 	V: 0.00 		V: 0.90 	V: 0.81 
V: 0.00 	V: 0.00 		V: 0.81 	V: 0.00 

Evaluation/Verbesserung (Schritt 7/8)

V: 0.00 	V: 0.00 	V: 0.00 	V: 0.00 	V: 0.55 
V: 0.00 	V: 0.00 			V: 0.61 R: -0.2 
V: 0.00 	V: -0.20 R: -0.2 		V: 1.00 R: 1.0 	V: 0.90 
V: 0.00 	V: 0.00 		V: 0.90 	V: 0.81 
V: 0.00 	V: 0.00 	V: 0.73 	V: 0.81 	V: 0.73 

$$V_{\hat{\pi}^*}(s) \leftarrow \max_a \left( \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a V^*(s') \right)$$

# Value Iteration

Evaluation/Verbesserung (Schritt 9/10)

V: 0.00 ↙	V: 0.00 ↔	V: 0.00 →	V: 0.49 →	V: 0.55 ↓
V: 0.00 ↕	V: 0.00 ↔			V: 0.61 R: -0.2 ↓
V: 0.00 ↕	V: -0.20 R: -0.2 ↕		V: 1.00 R: 1.0 ↕	V: 0.90 ←
V: 0.00 ↕	V: 0.00 ↓		V: 0.90 ↑	V: 0.81 ↖
V: 0.00 →	V: 0.66 →	V: 0.73 →	V: 0.81 ↑	V: 0.73 ↖

Evaluation/Verbesserung (Schritt 11/12)

V: 0.00 ↙	V: 0.00 →	V: 0.44 →	V: 0.49 →	V: 0.55 ↓
V: 0.00 ↕	V: 0.00 ↔			V: 0.61 R: -0.2 ↓
V: 0.00 ↕	V: -0.20 R: -0.2 ↓		V: 1.00 R: 1.0 ↕	V: 0.90 ←
V: 0.00 ↙	V: 0.59 ↓		V: 0.90 ↑	V: 0.81 ↖
V: 0.59 →	V: 0.66 →	V: 0.73 →	V: 0.81 ↑	V: 0.73 ↖

$$V_{\hat{\pi}^*}(s) \leftarrow \max_a \left( \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a V^*(s') \right)$$

# Value Iteration

Evaluation/Verbesserung (Schritt 13/14)

V: 0.00 →	V: 0.40 →	V: 0.44 →	V: 0.49 →	V: 0.55 ↓
V: 0.00 ↕	V: 0.00 ↑			V: 0.61 R: -0.2 ↓
V: 0.00 ↓	V: 0.33 R: -0.2 ↓		V: 1.00 R: 1.0 ↔	V: 0.90 ←
V: 0.53 ↙	V: 0.59 ↓		V: 0.90 ↑	V: 0.81 ↖
V: 0.59 →	V: 0.66 →	V: 0.73 →	V: 0.81 ↑	V: 0.73 ↖

Evaluation/Verbesserung (Schritt 15/16)

V: 0.36 →	V: 0.40 →	V: 0.44 →	V: 0.49 →	V: 0.55 ↓
V: 0.00 ↓	V: 0.36 ↑			V: 0.61 R: -0.2 ↓
V: 0.48 ↓	V: 0.33 R: -0.2 ↓		V: 1.00 R: 1.0 ↔	V: 0.90 ←
V: 0.53 ↙	V: 0.59 ↓		V: 0.90 ↑	V: 0.81 ↖
V: 0.59 →	V: 0.66 →	V: 0.73 →	V: 0.81 ↑	V: 0.73 ↖

$$V_{\hat{\pi}^*}(s) \leftarrow \max_a \left( \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a V^*(s') \right)$$

# Value Iteration

Evaluation/Verbesserung (Schritt 17/18)

V: 0.36 ↓	V: 0.40 →	V: 0.44 →	V: 0.49 →	V: 0.55 ↓
V: 0.43 ↓	V: 0.36 ←			V: 0.61 R: -0.2 ↓
V: 0.48 ↓	V: 0.33 R: -0.2 ↓		V: 1.00 R: 1.0 ↕	V: 0.90 ←
V: 0.53 ↙	V: 0.59 ↓		V: 0.90 ↑	V: 0.81 ↖
V: 0.59 →	V: 0.66 →	V: 0.73 →	V: 0.81 ↑	V: 0.73 ↖

$$V_{\hat{\pi}^*}(s) \leftarrow \max_a (\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a V^*(s'))$$

Policy Iteration (Schritt 20)

V: 0.06 ↓	V: 0.09 ↓	V: 0.13 →	V: 0.25 →	V: 0.55 ↓
V: 0.36 ↓	V: 0.30 ←			V: 0.61 R: -0.2 ↓
V: 0.48 ↓	V: 0.33 R: -0.2 ↓		V: 1.00 R: 1.0 ↕	V: 0.90 ←
V: 0.53 ↙	V: 0.59 ↓		V: 0.90 ↑	V: 0.81 ↖
V: 0.59 →	V: 0.66 →	V: 0.73 →	V: 0.81 ↑	V: 0.73 ↖

# Monte Carlo Methoden (MC)

- lernen direkt aus Erfahrungsepisoden
- sind modellfrei: benötigt kein Wissen über MDP-Übergänge / Belohnungen
- lernen aus kompletten Episoden: kein Bootstrapping
- verwenden die einfachste Idee: Wert = durchschnittliche Belohnung
- Problem: kann nur auf episodische MDPs angewendet werden
  - Alle Episoden müssen terminieren

# Monte Carlo Methoden (MC) – Exploring Starts

$\hat{\pi}^*$  Schätzung von  $\pi^*$

Initialisiere  $\hat{V}^*(s) \forall s \in S$  zufällig

Wiederhole:

Generiere Episode mit  $\hat{\pi}^*$  von zufälligem Zustand mit zufälliger Aktion

Für alle Tupel  $s, a$  der Episode (von  $s_T$  zu  $s_0$ ):

$G \leftarrow$  Entlohnung welche auf  $s_t, a_t$  folgt

Füge  $G$  zu *Entlohnungen*( $s, a$ ) hinzu

$\hat{Q}_{\pi^*}(s, a) \leftarrow \text{Durchschnitt}(\text{Entlohnungen}(s, a))$

Bestimmung Aktionswertfunktion

Für jedes  $s$  der Episode:

$\hat{\pi}^*(s) \leftarrow \arg \max_a \hat{Q}_{\pi^*}(s, a)$

Verbesserung Strategie



# Temporal-Difference Learning (TD)

- lernt direkt aus Erfahrungsepisoden
- ist modellfrei: benötigt kein Wissen über MDP-Übergänge / Belohnungen
- lernt aus unvollständigen Episoden, durch Bootstrapping
- aktualisiert eine Schätzung mit einer Schätzung

# Temporal-Difference Prediction (TD)

$\hat{Q}_{\pi^*}$  Schätzung von  $Q_{\pi^*}$

$\hat{\pi}^*$  Schätzung von  $\pi^*$

$\alpha$  Lernrate

Wiederhole:

Generiere Episode mit  $\hat{\pi}^*$

Für alle Tupel  $s, a, R$  der Episode:

$$\hat{Q}_{\pi^*}(s, a) \leftarrow \hat{Q}_{\pi^*}(s, a) + \alpha [R + \gamma \hat{Q}_{\pi^*}(s', \hat{\pi}^*(s)) - \hat{Q}_{\pi^*}(s, a)] \quad \text{Bestimmung Aktionswertfunktion}$$

Für jedes  $s$  der Episode:

$$\hat{\pi}^*(s) \leftarrow \arg \max_a \hat{Q}_{\pi^*}(s, a) \quad \text{Verbesserung Strategie}$$

# Bootstrapping und Sampling

## Bootstrapping (shallow backups)

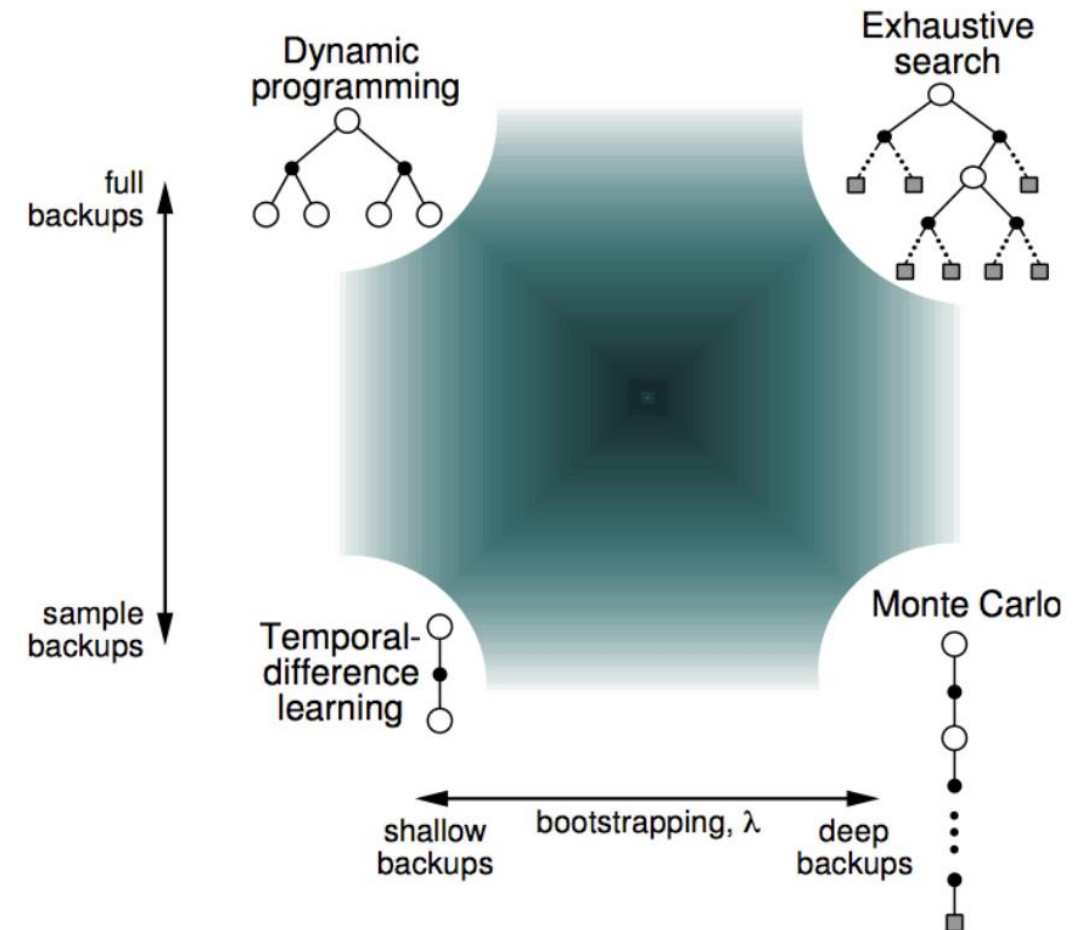
Aktualisierung beruht auf Schätzung

- Dynamische Programmierung
- Temporal Difference Learning

## Sampling (sample backups)

Aktualisierung beruht auf Erwartungswert

- Monte Carlo Methoden
- Temporal Difference Learning



Quelle: Silver2015

# Reinforcement Learning (RL)

- Bestandteile des RL Problems
  - Markov'scher Entscheidungsprozess

- Grundlegende Lösungsverfahren
  - Dynamische Programmierung (DP)
  - Monte Carlo Methoden
  - Temporal Difference Learning

- Reinforcement Learning Algorithmen
  - SARSA
  - Q-Learning

- DQN

# SARSA – (State, Action, Reward, State, Action)

Initialisiere  $Q(s, a) \forall s \in S, a \in A(s)$  zufällig

Wiederhole (für jede Episode):

Initialisiere  $s$

Wähle  $a$  von  $s$  mit Strategie abgeleitet von  $Q$  (z.B.  $\epsilon$ -greedy)

Wiederhole (für jeden Schritt der Episode):

Nimm Aktion  $a$ , beobachte  $R, s'$

Wähle  $a'$  von  $s'$  mit Strategie abgeleitet von  $Q$  (z.B.  $\epsilon$ -greedy)

$$Q(s, a) \leftarrow Q(s, a) + \alpha[R + \gamma Q(s', a') - Q(s, a)]$$

$\epsilon$ -greedy

$$a = \begin{cases} \arg \max_a Q(s, a), & x \sim U[0,1] < \epsilon \\ a \in A(s), & x \sim U[0,1] \geq \epsilon \end{cases}$$

# Q-Learning

Initialisiere  $Q(s, a) \forall s \in S, a \in A(s)$  zufällig

Wiederhole (für jede Episode):

Initialisiere  $s$

Wiederhole (für jeden Schritt der Episode):

Wähle  $a$  von  $s$  mit Strategie abgeleitet von  $Q$  (z.B.  $\epsilon$ -greedy)

Nimm Aktion  $a$ , beobachte  $R, s'$

$$Q(s, a) \leftarrow Q(s, a) + \alpha [R + \gamma \max_{a'} Q(s', a') - Q(s, a)]$$

$\epsilon$ -greedy

$$a = \begin{cases} \arg \max_a Q(s, a), & x \sim U[0,1] < \epsilon \\ a \in A(s), & x \sim U[0,1] \geq \epsilon \end{cases}$$

# On-Policy Learning / Off-Policy Learning

- On-Policy Methoden evaluieren oder verbessern die Strategie, welche verwendet wird um Entscheidungen zu treffen
- “Learning on the job”
- Off-Policy Methoden evaluieren oder verbessern eine Strategie, welche sich von der Strategie unterscheidet um Entscheidungen zu generieren
- “Look over someone’s shoulder”

# Exploration vs Exploitation

- Reinforcement Learning ist Trial-and-Error-Lernen
- Aktionen müssen unendlich oft ausprobiert werden
- Änderung der Suchstrategie von global (Exploration) zu lokal (Exploitation) während des Lernprozesses
- Beispiel:  $\epsilon$ -greedy/decreasing
  - Folge Strategie  $\pi(s)$  mit Wahrscheinlichkeit  $1 - \epsilon$ , sonst zufällige Aktion
  - Epsilon konvergiert im Laufe des Trainings gegen 0,  $\epsilon = \frac{1}{n+1}$

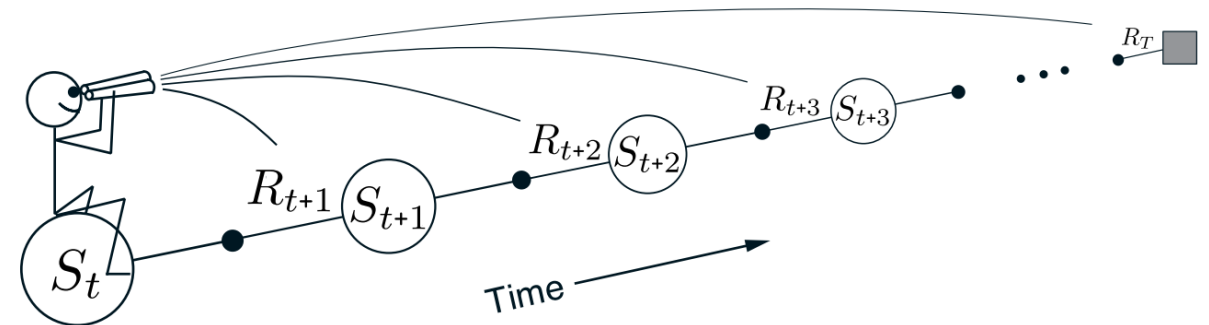


# Lernen von Aktionssequenzen

- Belohnung erst nach einer Sequenz von Aktionen bekannt
    - Schach: Züge bauen aufeinander auf (nicht nur der einzelne Zug relevant)
  
  - Belohnung erst am Ziel
    - Schach: erst am Spielende ist klar wer gewonnen hat
- bei langen Aktionssequenzen kann erst am Ende der Sequenz gelernt werden
- frühere Aktionen können für den schlechten Ausgang verantwortlich sein

# Die Vorwärtssicht TD( $\lambda$ )

- Aktualisiert die Bewertungsfunktion in Richtung der  $\lambda$ -Entlohnung
- Vorwärtssicht schaut in die Zukunft um die Entlohnung zu berechnen
- Ähnlich zu Monte Carlo: kann nur für terminierte Episoden berechnet werden

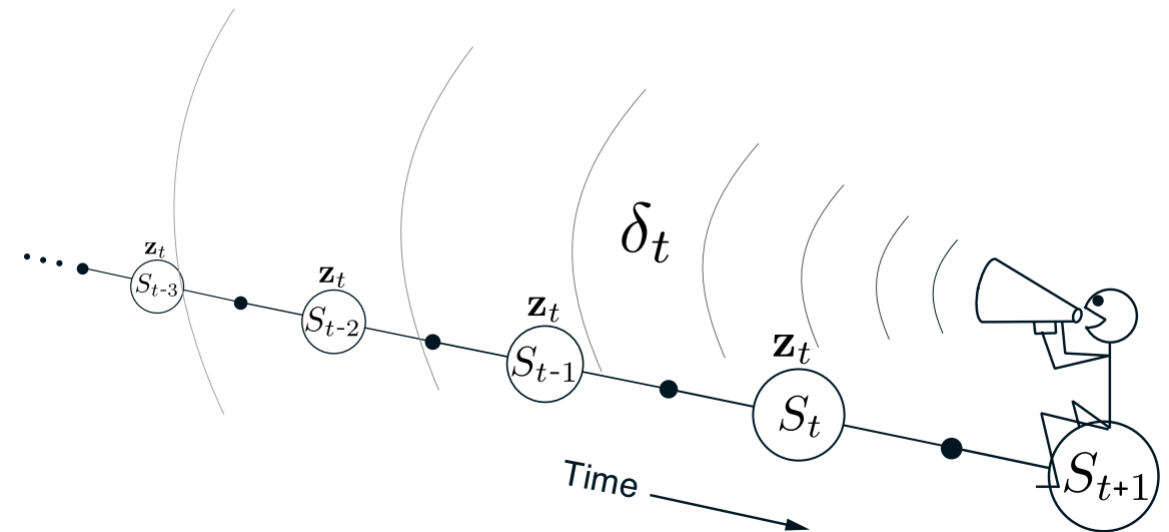


Quelle: Sutton2017

# Die Rückwärtssicht TD( $\lambda$ ) und Eligibility Traces

- Erstelle eine Verantwortlichkeitsspur für jeden Zustand
- $\delta_t$ : TD Fehler, rückwärts gerichtet auf gerade besuchten Zustand
- $E_t(s)$ : Verantwortlichkeitsspur

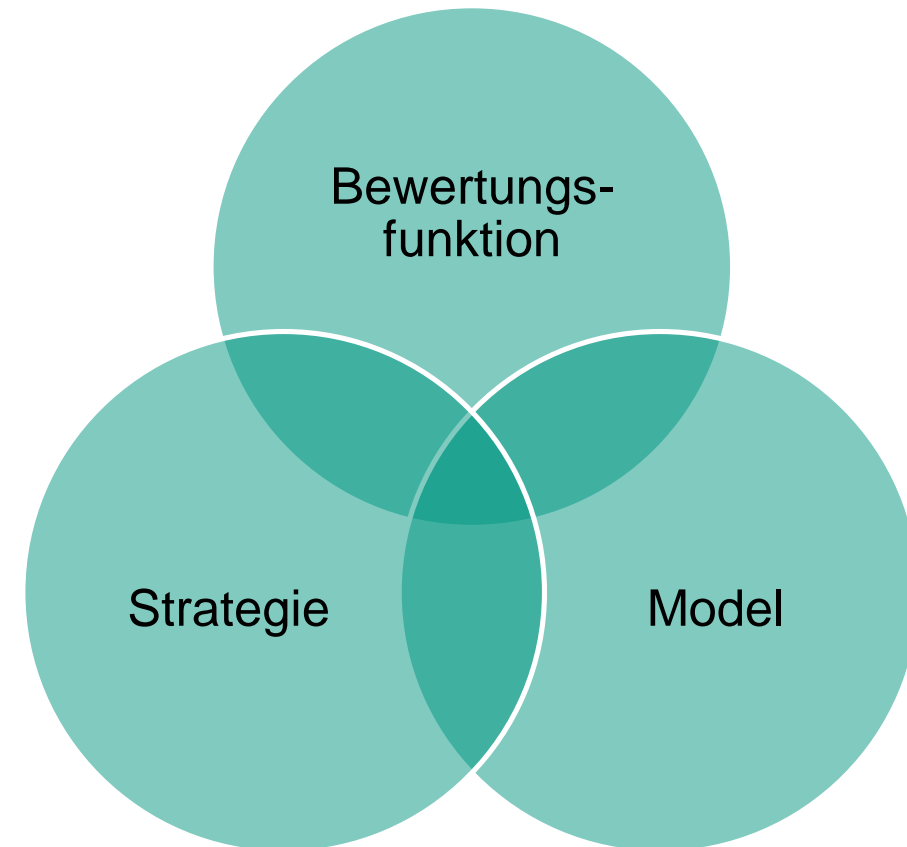
$$V(s) \leftarrow V(s) + \alpha \delta_t E_t(s)$$



Quelle: Sutton2017

# Taxonomie des Reinforcement Learning

- **Bewertungsbasiert**
  - Keine Strategie
  - Bewertungsfunktion
- **Strategiebasiert**
  - Strategie
  - Keine Bewertungsfunktion
- **Actor Critic**
  - Strategie
  - Bewertungsfunktion



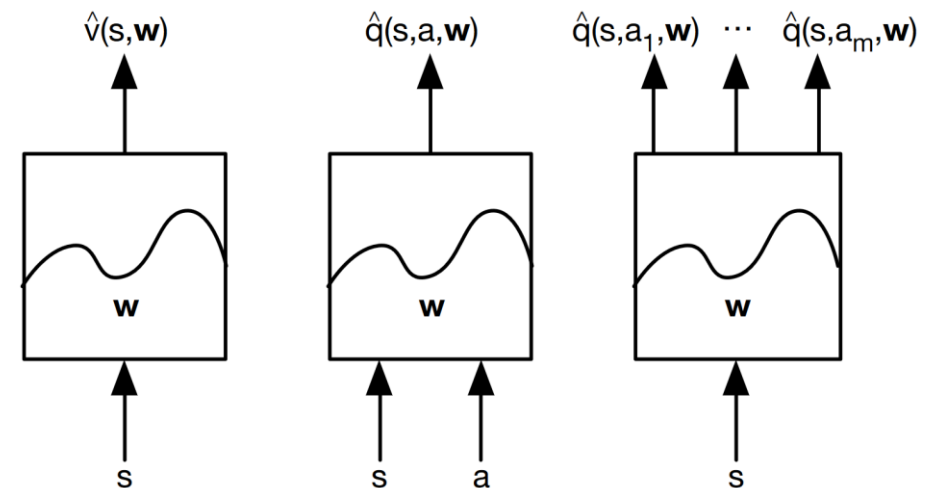
# RL mit Funktionsapproximation – Deep RL

## ■ Warum?

- Reinforcement Learning wird häufig auf komplexen Problemen angewandt
  - Backgammon:  $10^{20}$  Zustände
  - Go:  $10^{170}$  Zustände
  - StarCraft II: quasi kontinuierlich
- Unmöglich alle Werte in Tabelle zu speichern

## ■ Wie?

- Linearkombination von Merkmalen
- Neurale Netze



Quelle: Silver2015

# Reinforcement Learning (RL)

- Bestandteile des RL Problems
    - Markov'scher Entscheidungsprozess
  - Grundlegende Lösungsverfahren
    - Dynamische Programmierung (DP)
    - Monte Carlo Methoden
    - Temporal Difference Learning
  - Reinforcement Learning Algorithmen
    - SARSA
    - Q-Learning
- DQN

# Q-Learning

Initialisiere  $Q(s, a) \forall s \in S, a \in A(s)$  zufällig

Wiederhole (für jede Episode):

Initialisiere  $s$

Wiederhole (für jeden Schritt der Episode):

Wähle  $a$  von  $s$  mit Strategie abgeleitet von  $Q$  (z.B.  $\epsilon$ -greedy)

Nimm Aktion  $a$ , beobachte  $r, s'$

$$T = r + \gamma \max_{a'} Q(s', a')$$

$$Q(s, a) \leftarrow Q(s, a) + \alpha [T - Q(s, a)]$$

Temporal Difference Error

$\epsilon$ -greedy

$$a = \begin{cases} \arg \max_a Q(s, a), & x \sim U[0,1] < \epsilon \\ a \in A(s), & x \sim U[0,1] \geq \epsilon \end{cases}$$

# Q-Learning mit Funktionsapproximation

Initialisiere  $Q_\theta(s, a) \forall s \in S, a \in A(s)$  zufällig

Wiederhole (für jede Episode):

Initialisiere  $s$

Wiederhole (für jeden Schritt der Episode):

Wähle  $a$  von  $s$  mit Strategie abgeleitet von  $Q$  (z.B.  $\epsilon$ -greedy)

Nimm Aktion  $a$ , beobachte  $r, s'$

$$T = r + \gamma \max_{a'} Q_\theta(s', a')$$

$$\theta \leftarrow \theta - \alpha \nabla_\theta \mathbb{E}_{s' \sim \mathcal{P}_{ss'}^a} [(T - Q_\theta(s, a))^2]$$

nicht stationär

Korrelation

$\epsilon$ -greedy

$$a = \begin{cases} \arg \max_a Q(s, a), & x \sim U[0,1] < \epsilon \\ a \in A(s), & x \sim U[0,1] \geq \epsilon \end{cases}$$



# Target Network

- Das Target (Sollwert) wird nicht mit den aktuellen Aktionswerten des Netzes berechnet, sondern mit einer niederfrequent aktualisierten Kopie des Netzes
  - Periodische Aktualisierung des Target Network

$$T = R + \gamma \max_{a'} Q_{\theta'}(s', a')$$
$$\theta \leftarrow \theta - \alpha \nabla_{\theta} \mathbb{E}_{s' \sim \mathcal{P}_{ss'}^a} [(T - Q_{\theta}(s, a))^2]$$

- Stabileres Training
  - Bricht die Korrelation von Aktionswert (Q) und Target
  - Resultiert in einer statischeren Verteilung der Labels

# Experience Replay

- Ein endlicher Erfahrungsspeicher aus welchem zufällige Erfahrungen  $(s_t, a_t, r_t, s_{t+1})$  gezogen werden um zu lernen
  - Erfahrungen werden zunächst im Erfahrungsspeicher gespeichert
  - Um zu lernen werden Minibatches aus zufälligen Erfahrungen aus dem Erfahrungsspeicher erstellt
- Stabileres Training
  - Bricht die Korrelation von Erfahrungen welche durch Trajektorien entstehen
  - Resultiert in einer statischeren Verteilung der Daten
  - Führt zu besserer Dateneffizienz

# Deep Q-Learning

Initialisiere Erfahrungspuffer  $D$  mit Kapazität  $N$

Initialisiere  $Q_\theta$  mit zufälligen Gewichten  $\theta$

Initialisiere  $Q_{\theta'}$  mit  $\theta' = \theta$

Wiederhole (für jede Episode):

Initialisiere  $s$

Wiederhole (für jeden Schritt der Episode):

Wähle  $a$  von  $s$  mit Strategie abgeleitet von  $Q$

Nimm Aktion  $a$ , beobachte  $r, s'$

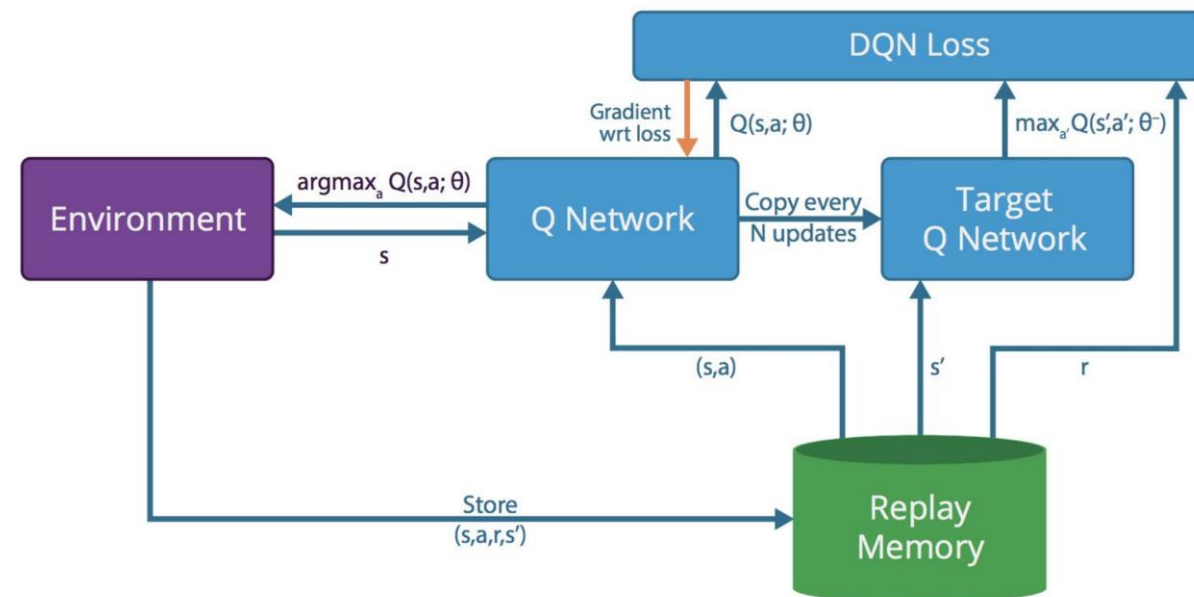
Speicher Erfahrung  $(s, a, r, s')$  in  $D$

Ziehe eine zufällige Stichprobe von Erfahrungen aus  $D$

$$T = \begin{cases} r & \text{Falls die Episode im nächsten Schritt terminiert} \\ r + \gamma \max_{a'} Q_{\theta'}(s', a') & \text{ansonsten} \end{cases}$$

$$\theta \leftarrow \theta - \alpha \nabla_{\theta} \mathbb{E}_{s' \sim \mathcal{P}_{ss'}} \left[ (T - Q_{\theta}(s, a))^2 \right]$$

Alle  $C$  Schritte  $Q_{\theta'} = Q_{\theta}$



# Learning How to Drive in a Real World Simulation with Deep Q-Networks

# Charakteristika – Q-Learning

- Strategie wird nur implizit gelernt/verbessert
- Lernt deterministische Strategien
- Generell weniger stabil (siehe Erweiterungen)
- Dateneffizient (Off-Policy)
- Diskrete Aktionsräume

# Lernen und Planen

Zwei grundlegende Probleme bei der sequentiellen Entscheidungsfindung

## ■ Reinforcement Learning

- Die Umgebung ist zunächst unbekannt
- Der Agent interagiert mit der Umgebung
- Der Agent verbessert seine Strategie

## ■ Planen

- Ein Modell der Umgebung ist bekannt
- Der Agent führt mit seinem Modell Berechnungen durch (ohne externe Interaktion)
- Der Agent verbessert seine Strategie

# Literatur

- R. Sutton 2018 – “Reinforcement Learning: An Introduction”
- S. Levine 2018 – “Deep Reinforcement Learning” (Berkley Course on RL)
- P. Abbeel 2017 – “Deep RL Bootcamp” (Berkley Course on RL)
- D. Silver 2015 – “Reinforcement Learning” (UCL Course on RL)
- OpenAi 2018 – “SpinningUp”