

Universität Potsdam

Institut für Informatik
Lehrstuhl Maschinelles Lernen



Reinforcement Learning

Uwe Dick

Inhalt

- Problemstellungen
- Beispiele
- Markov Decision Processes
- Planen – vollständige MDPs
- Lernen – unbekannte MDPs
- Monte-Carlo
- Temporal Difference

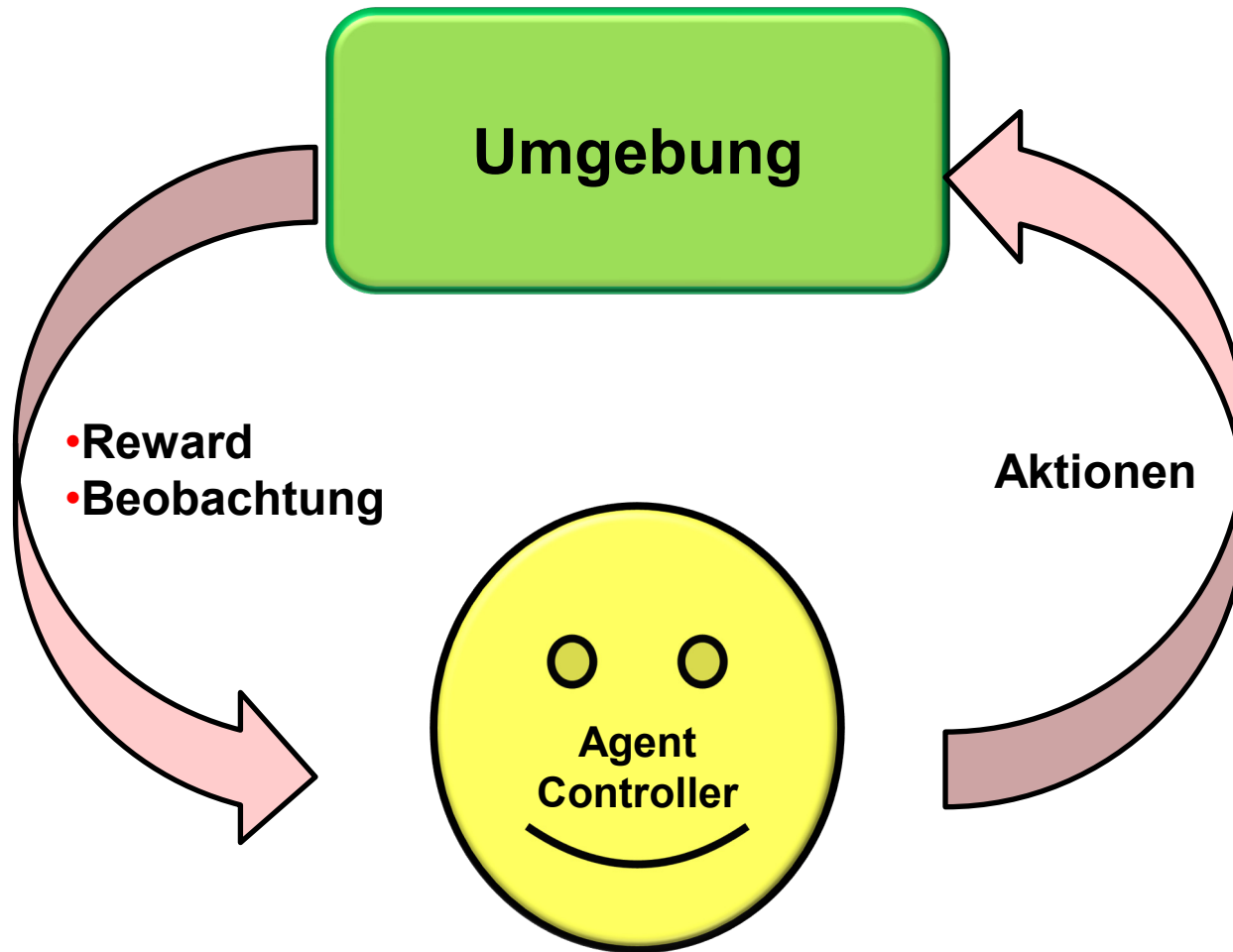
Problemstellungen des maschinellen Lernens

- Überwachtes Lernen:
Lernen einer Entscheidungsfunktion aus Beispielen der richtigen Entscheidung.
- Unüberwachtes Lernen:
Lernen von zB. Partitionierungen von Daten (Clustern) ohne Beispiele für die richtige Partitionierung.
- Reinforcement Learning:
Lernen von sequenziellen Entscheidungen. Die Güte einer Entscheidung wird durch die Güte der Entscheidungssequenz bestimmt.
→ Temporal Credit Assignment Problem.

Beispiele

- Schach: Welcher Zug hatte wieviel Einfluss auf das Spielergebnis?
- Robofußball: Ziel ist es, ein Tor zu schießen. Aber welche Bewegungen haben das ermöglicht?
- Helikopterflug: Welche Bewegungen müssen ausgeführt werden, um bei unvorhersehbaren äußeren Bedingungen nicht abzustürzen.

Lernen aus Interaktionen



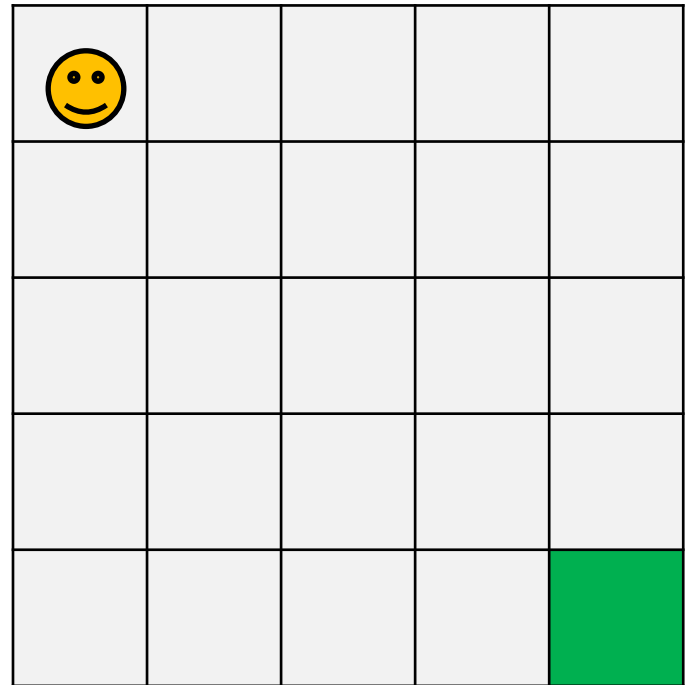
Wann Reinforcement Learning?

- Verzögerte Bewertung von Aktionen.
(Temporal credit assignment problem)
- Kontrollprobleme
- Agenten – Das volle KI-Problem

Was ist Reinforcement Learning?

- RL-Methoden sind „Sampling based methods to solve optimal control problems “ (Richard Sutton)
- Suche nach einer optimalen Policy: Funktion von Zustand zu Aktion.
- Optimalität des Lernens: Policy mit höchstem erwarteten Reward.
- Aber auch: schnelles Lernen ohne zuviele Fehler zu machen.

Beispiel: Gridworld



Markov Decision Processes

- Markov-Entscheidungsprozess (S, A, R, P)
- S : endliche Zustandsmenge
- A : endliche Aktionsmenge
- P : Übergangswahrscheinlichkeiten

$$P(s'|s, a) \quad s, s' \in S, a \in A$$

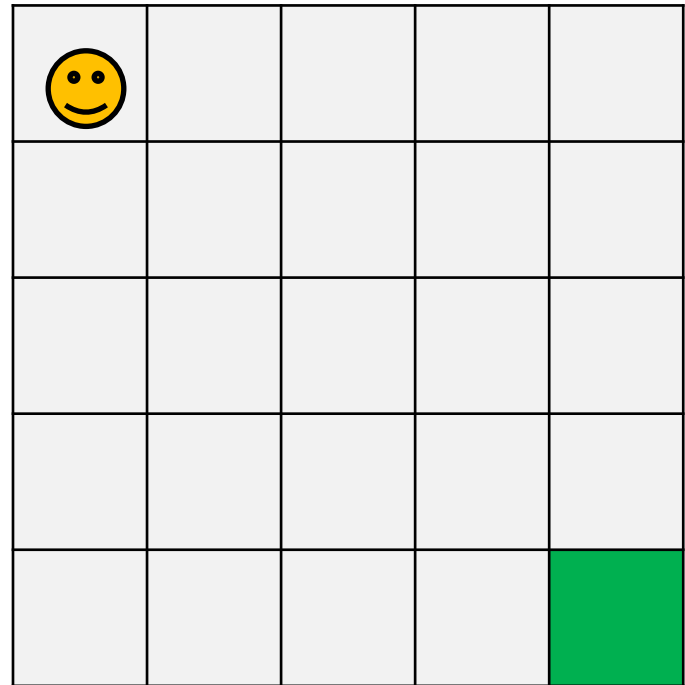
- R : Erwarteter Reward. Beschreibt den sofort erzielten Gewinn.

$$R : (S \times A) \rightarrow \mathbb{R}$$

- Discount factor $0 \leq \gamma < 1$.

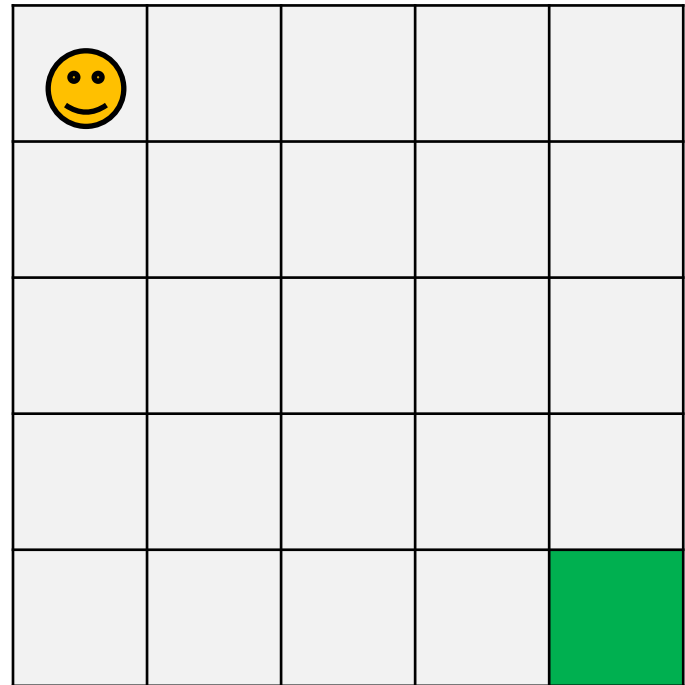
Beispiel: Gridworld

- Zustandsraum \mathcal{S}
 - ◆ Startzustand $s_s \in \mathcal{S}$
 - ◆ Zielzustand $s_z \in \mathcal{S}$



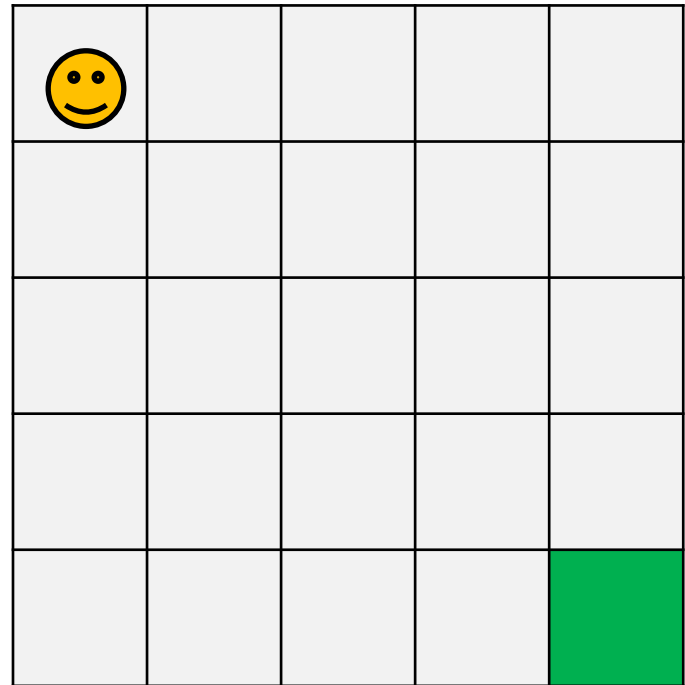
Beispiel: Gridworld

- Zustandsraum S
- Aktionsmenge A
 - ◆ $A=(\text{links, rechts, oben, unten})$



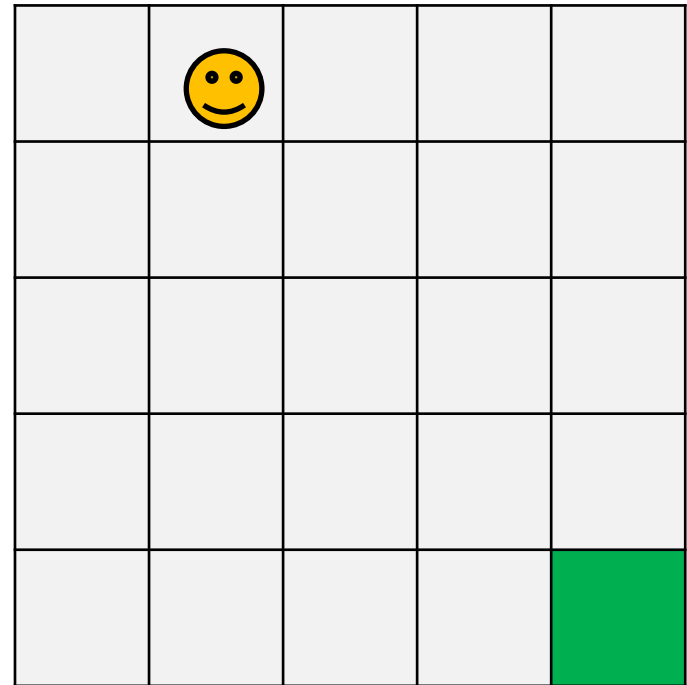
Beispiel: Gridworld

- Zustandsraum S
- Aktionsmenge A
- Übergangswahrscheinlichkeit P
 - ◆ $P((1,2)|(1,1), \text{rechts}) = 1$



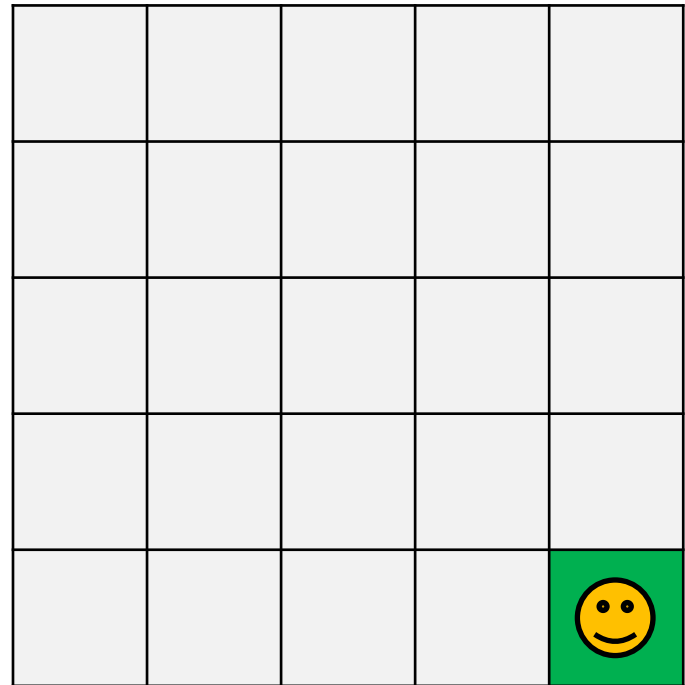
Beispiel: Gridworld

- Zustandsraum S
- Aktionsmenge A
- Übergangswahrscheinlichkeit P
- Erwarteter Reward R
 - ◆ $R((1,1),\text{rechts}) = 0$



Beispiel: Gridworld

- Zustandsraum S
- Aktionsmenge A
- Übergangswahrscheinlichkeit P
- Erwarteter Reward R
 - ◆ $R((4,5),\text{unten}) = 1$



Markov Decision Processes

- Markov-Entscheidungsprozess (S, A, R, P)
- S : endliche Zustandsmenge
- A : endliche Aktionsmenge
- P : Übergangswahrscheinlichkeiten

$$P(s'|s, a) \quad s, s' \in S, a \in A$$

- R : Erwarteter Reward. Beschreibt den sofort erzielten Gewinn.

$$R : (S \times A) \rightarrow \mathbb{R}$$

- Discount factor $0 \leq \gamma < 1$.

MDP


- Eine deterministische stationäre Policy bildet Zustände auf Aktionen ab.

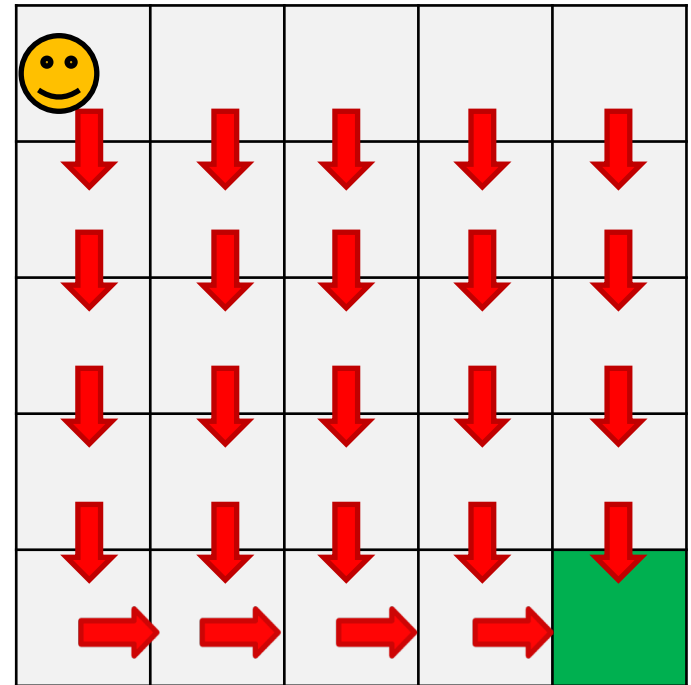
$$\pi : S \rightarrow A$$

- Stochastische Policy: Funktion von Zuständen auf eine Verteilung von Aktionen.
- Ziel: Finde Policy π , die den erwarteten kumulativen (discounted) Gewinn maximieren.

$$E_{\pi, P} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t)) \right]$$


Beispiel: Gridworld

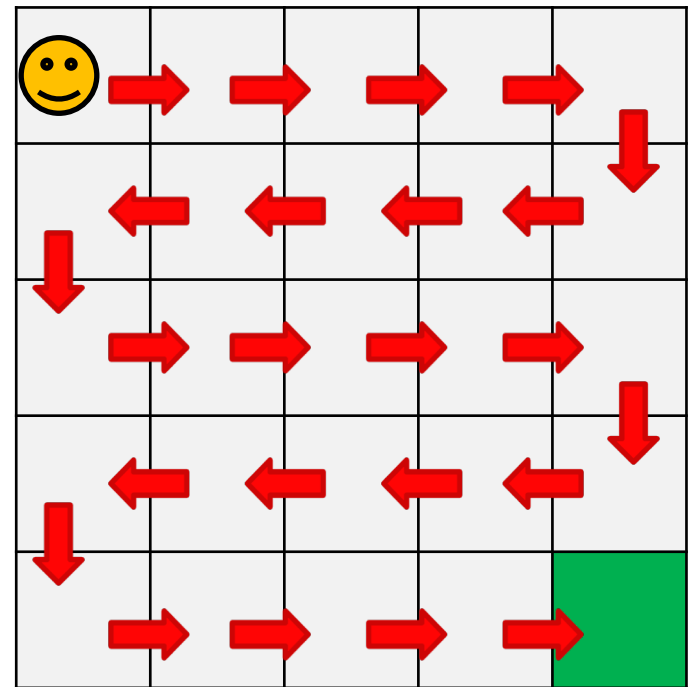
- Zustandsraum S
- Aktionsmenge A
- Übergangswahrscheinlichkeit P
- Erwarteter Reward R
- Discountfaktor $\gamma = 0,9$
- Policy π
 - ◆ Gute Policy π_2 
- Erwarteter discounted Reward



$$E_{\pi, P} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t)) \mid s_0 = s_s \right] = 0,9^7$$

Beispiel: Gridworld

- Zustandsraum S
- Aktionsmenge A
- Übergangswahrscheinlichkeit P
- Erwarteter Reward R
- Discountfaktor $\gamma = 0,9$
- Policy π
 - ◆ Schlechte Policy π_1 
- Erwarteter discounted Reward



$$E_{\pi, P} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t)) \mid s_0 = s_s \right] = 0,9^{23}$$

Markov-Eigenschaft

- Markov-Eigenschaft:

$$\begin{aligned}P(s_{t+1} | s_t, a_t, s_{t-1}, a_{t-1}, \dots, s_0, a_0) &= P(s_{t+1} | s_t, a_t) \\R(s_t, a_t, s_{t-1}, a_{t-1}, \dots, s_0, a_0) &= R(s_t, a_t)\end{aligned}$$

- Aus Sequenz von Beobachtungen und Aktionen wird Zustand.
- Markov-Eigenschaft in Realität selten genau erfüllt.

Value Functions – Bewertungsfunktionen

- Value function $V^\pi(s)$ für einen Zustand s und Policy π beschreibt den erwarteten kumulativen Gewinn der von diesem Zustand aus erreicht wird.


$$V^\pi(s_t) = E_{\pi, P} \left[\sum_{k=0}^{\infty} \gamma^k R(s_{t+k}, \pi(s_{t+k})) \right]$$

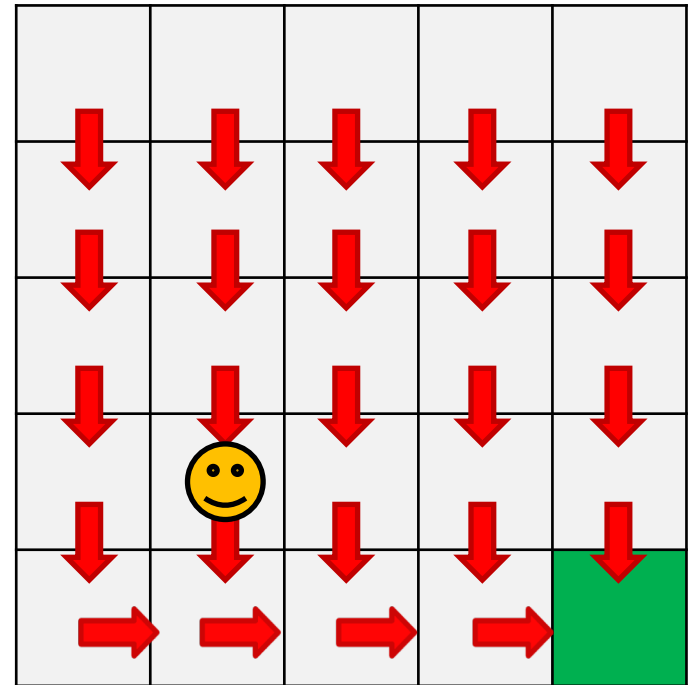
- Es existiert immer eine optimale deterministische stationäre Policy π^* , die die Value Function maximiert.

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

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

Beispiel: Gridworld

- Zustandsraum S
- Aktionsmenge A
- Übergangswahrscheinlichkeit P
- Erwarteter Reward R
- Discountfaktor $\gamma = 0,9$
- Policy π
 - ◆ Gute Policy π_2 
- Erwarteter discounted Reward



$$V^{\pi_1}(s_t) = E_{\pi, P} \left[\sum_{k=0}^{\infty} \gamma^k R(s_{t+k}, \pi(s_{t+k})) \right] = 0,9^3$$

Value Functions

- Bewertungsfunktion für Zustand-Aktion-Paar:

$$Q^\pi(s_t, a_t) = R(s_t, a_t) + E_{\pi, P} \left[\sum_{k=1}^{\infty} \gamma^k R(s_{t+k}, \pi(s_{t+k})) \right]$$

- Optimale Bewertungsfunktion

$$Q^*(s_t, a) = R(s_t, a) + \gamma E_P[V^*(s_{t+1})]$$

$$\text{und } V^*(s_t) = \max_a Q^*(s_t, a)$$

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

$$\pi^*(s) = \arg \max_a Q^*(s, a)$$

- Annahme: tabellarische Speicherung der Bewertungen aller Zustände

Bellman-Gleichungen

- Für Bewertungsfunktionen gelten die Bellman-Gleichungen (durch Markov-Eigenschaft)

$$\begin{aligned} V^\pi(s_t) &= E_{\pi, P} \left[\sum_{k=0}^{\infty} \gamma^k R(s_{t+k}, \pi(s_{t+k})) \right] \\ &= E_{\pi, P} \left[R(s_t, \pi(s_t)) + \gamma \sum_{k=0}^{\infty} \gamma^k R(s_{t+k+1}, \pi(s_{t+k+1})) \right] \\ &= E_{\pi, P} \left[R(s_t, \pi(s_t)) + \gamma V^\pi(s_{t+1}) \right] \\ &= R(s_t, \pi(s_t)) + \gamma \sum_{s_{t+1} \in S} P(s_{t+1} | s_t, \pi(s_t)) V^\pi(s_{t+1}) \end{aligned}$$

Bellman-Gleichungen

- Zustand-Aktion-Bewertungsfunktion:

$$Q^\pi(s_t, a_t) = R(s_t, a_t) + \gamma E_{\pi, P} \left[Q^\pi(s_{t+1}, \pi(s_{t+1})) \right]$$

- Die Bellman-Gleichungen bilden ein lineares Gleichungssystem

$$\begin{aligned} V^\pi &= R + \gamma P^\pi V^\pi \\ \rightarrow V^\pi &= (I - \gamma P^\pi)^{-1} R \end{aligned}$$

Bellman-Operatoren

- In (linearer) Operatorschreibweise:

$$V^\pi = T^\pi V^\pi$$

- Mit linearem Operator T^π :

$$(T^\pi V)(s) = R(s, \pi(s)) + \gamma \sum_{s' \in S} P(s'|s, \pi(s))V(s')$$

- V^π ist ein Fixpunkt des Bellman-Operators T^π .

Bellman-Optimalitätsgleichungen

- Bellman-Gleichungen für das Kontrollproblem.
- Rekursive Beziehungen der optimalen Value Functions.

$$V^*(s_t) = \max_a E_{\pi, T} \left[R(s_t, a) + \gamma V^*(s_{t+1}) \right]$$

$$Q^*(s_t, a_t) = R(s_t, a_t) + \gamma E_{\pi, T} \left[\max_a Q^*(s_{t+1}, a) \right]$$

Modellwissen

- Es ergeben sich unterschiedliche Problemstellungen je nach Wissen über den MDP.
- MDP vollständig bekannt.
→ Planen.
- MDP nicht oder teilweise bekannt. Es kann Erfahrung gesammelt werden durch Interaktion mit der Umgebung.
→ Reinforcement Learning.

Arten von Reinforcement Learning

- Reinforcement Learning-Methoden können eingeteilt werden bezüglich der Verwendung der Interaktionsbeispiele.
- Indirekte Methoden:
 - ◆ Model learning
- Direkte Methoden:
 - ◆ Direct Policy Search
 - ◆ Value function estimation



MDP vollständig bekannt – Dynamische Programmierung

- 2 Schritte zum Berechnen der optimalen Policy:
 - ◆ Policy Evaluation: V^π berechnen für festes π_k
 - ◆ Policy Improvement: Neues π_{k+1} bestimmen
- Policy Iteration.

- Bellman-Gleichungen bilden ein lineares Gleichungssystem.
- Zustandsmengen sind allerdings in der Realität in der Regel zu groß um Standardlösungsverfahren für LGS zu verwenden.

Policy Iteration


- Iteratives Verfahren: V^π iterativ durch Folge von Approximationen V_k berechnen

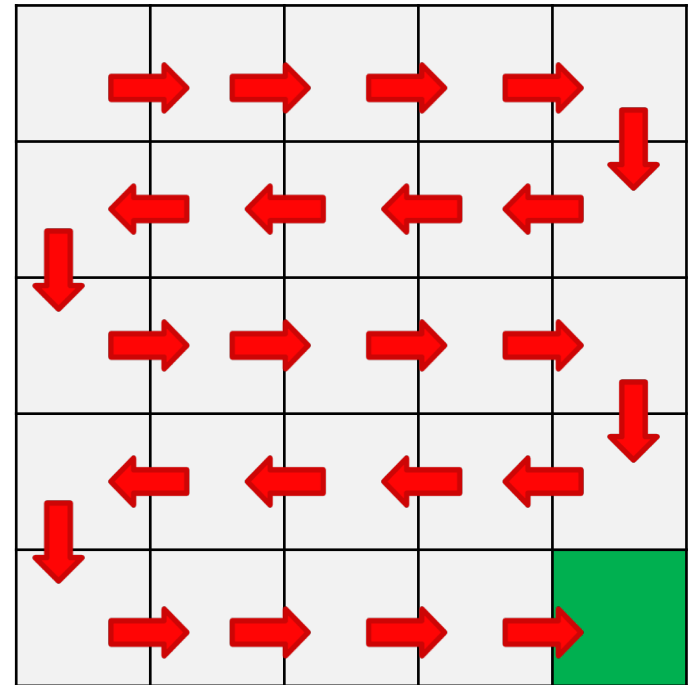
$$\begin{aligned} V_{k+1}(s_t) &= E_{\pi, T} \left[R(s_t, \pi(s_t)) + \gamma V_k(s_{t+1}) \right] \\ &= R(s_t, \pi(a_t)) + \gamma \sum_{s_{t+1} \in \mathcal{S}} T(s_{t+1} | s_t, \pi(s_t)) V_k(s_{t+1}) \end{aligned}$$

- Greedy Policy Improvement:


$$\pi_{k+1}(s) = \arg \max_a Q^{\pi_k}(s, a)$$

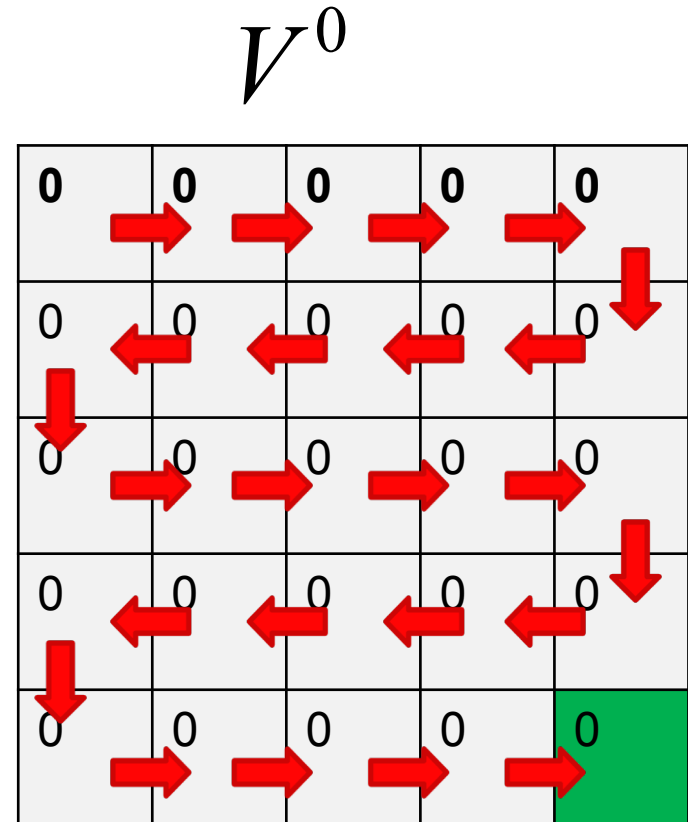
Beispiel: Gridworld

- Discountfaktor $\gamma = 0,9$
- Start Policy π_1 
- Policy Iteration:
 - ◆ Berechne V^{π_1}
durch Folge von
Approximationen V^k




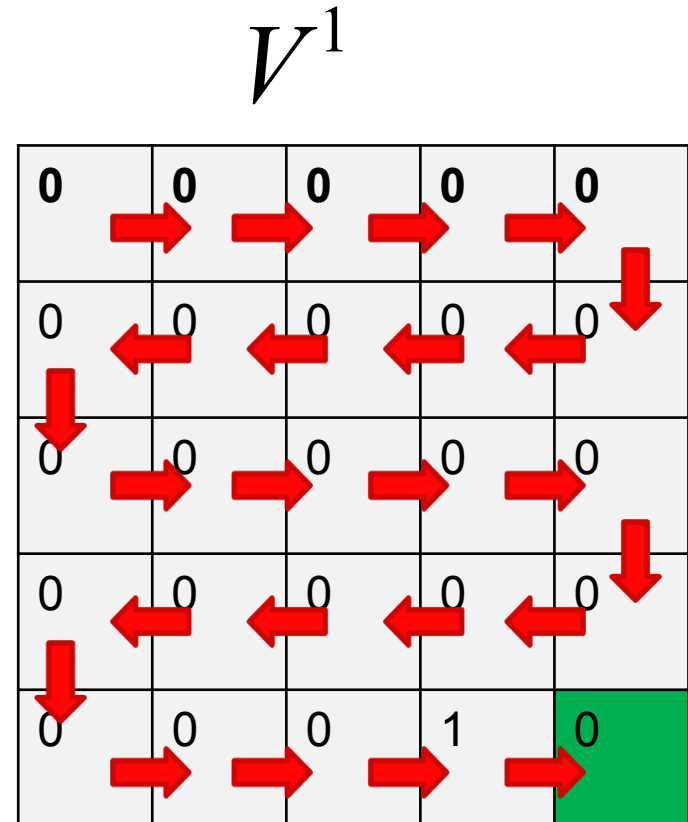
Beispiel: Gridworld

- Discountfaktor $\gamma = 0,9$
- Start Policy π_1 
- Policy Iteration:
 - ◆ Berechne V^{π_1}
durch Folge von
Approximationen V^k




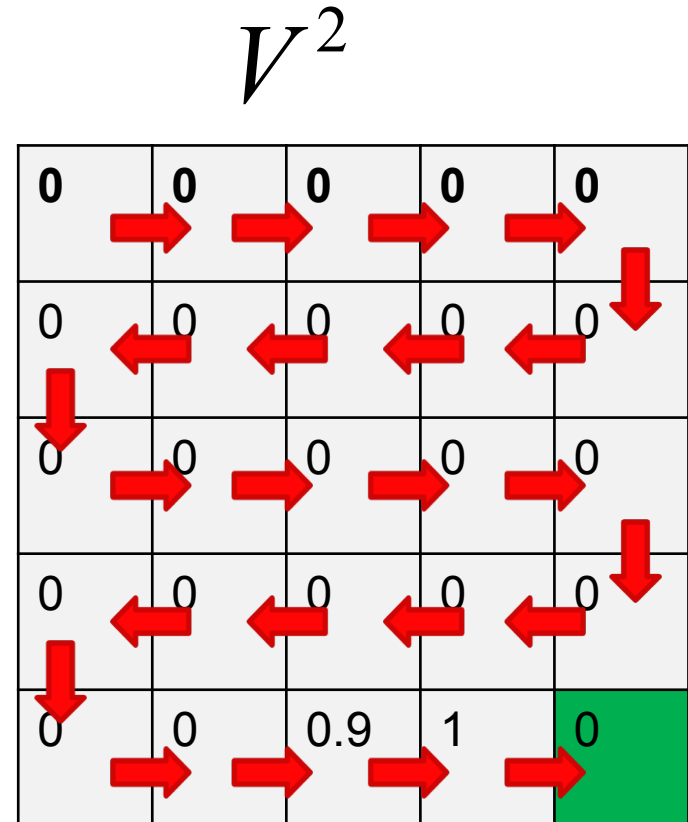
Beispiel: Gridworld

- Discountfaktor $\gamma = 0,9$
- Start Policy π_1 
- Policy Iteration:
 - ◆ Berechne V^{π_1}
durch Folge von
Approximationen V^k




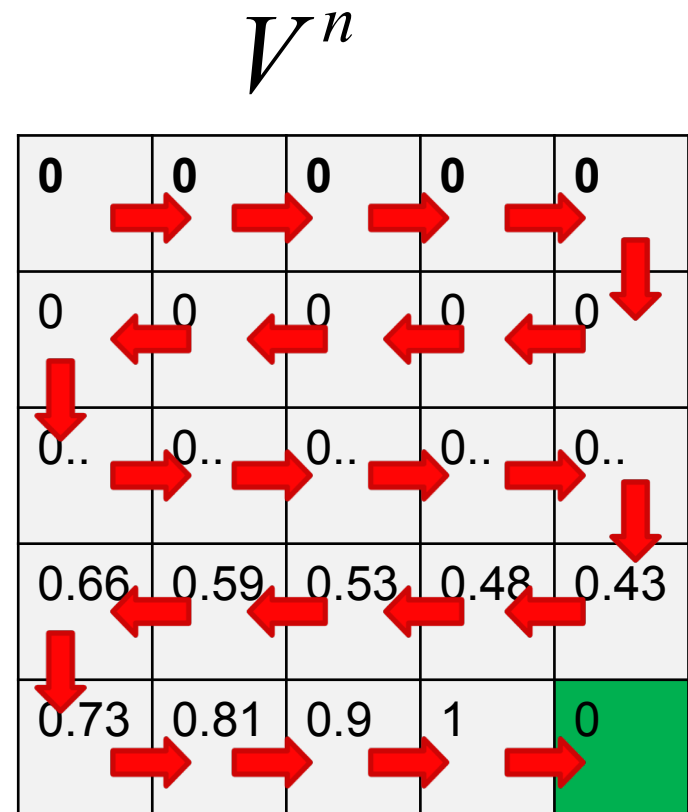
Beispiel: Gridworld

- Discountfaktor $\gamma = 0,9$
- Start Policy π_1 
- Policy Iteration:
 - ◆ Berechne V^{π_1}
durch Folge von
Approximationen V^k




Beispiel: Gridworld


























- Discountfaktor $\gamma = 0,9$
- Start Policy π_1 
- Policy Iteration:
 - ◆ Berechne V^{π_1} durch Folge von Approximationen V^k



Beispiel: Gridworld

- Discountfaktor $\gamma = 0,9$
- Start Policy π_1 
- Policy Iteration:
 - ◆ Berechne V^{π_1} durch Folge von Approximationen V^k
 - ◆ Policy Improvement: Berechne greedy Policy π_2

$$V^{\pi_1}$$

0	0	0	0	0
				
0	0	0	0	0
				
0..	0..	0..	0..	0..
				
0.66	0.59	0.53	0.48	0.43
				
0.73	0.81	0.9	1	0
				

Policy Evaluation

- Im Limit $k \rightarrow \infty$ konvergiert V_k zu V^π . Konvergenzrate $O(\gamma^k)$: $\|V_k - V^\pi\| = O(\gamma^k)$
- Beweis z.B. über Banach'schen Fixpunktsatz.
- Sei $B=(B, \|\cdot\|)$ ein Banachraum.
- Sei T ein Operator $T:B \rightarrow B$, so dass $\|TU - TV\| \leq \gamma \|U - V\|$ mit $\gamma < 1$.
 T nennt man dann eine γ -Kontraktion.
- Dann hat T einen eindeutigen Fixpunkt V und es gilt, dass für alle $V_0 \in B$ die Folge $V_{k+1} = T V_k$, $k \rightarrow \infty$ gegen V konvergiert. Außerdem gilt $\|V_k - V\| = O(\gamma^k)$

Policy Evaluation

- Es gilt, dass der Bellman-Operator T^π

$$(T^\pi V)(s) = R(s, \pi(s)) + \gamma \sum_{s' \in S} P(s'|s, \pi(s))V(s')$$

eine γ -Kontraktion ist. Daraus folgt, dass die iterative Anwendung des Operators

$$V_{k+1}(s) = (T^\pi V_k)(s) = R(s, \pi(s)) + \gamma \sum_{s' \in S} P(s'|s, \pi(s))V_k(s')$$

gegen V^π konvergiert.

Policy Evaluation: Kontraktion

- Was bedeutet Kontraktion?

$$\|T^\pi V_{k+1} - T^\pi V_k\| \leq \gamma \|V_{k+1} - V_k\|$$

- Anwenden der Iterationsvorschrift:

$$\|V_{k+2} - V_{k+1}\| \leq \gamma \|V_{k+1} - V_k\|$$

- Die Distanz zur echten Value Function verringert sich pro Iteration mit Faktor γ (im sup-Norm Sinne).

$$\|V^\pi - V_{k+1}\| = \|T^\pi V^\pi - T^\pi V_k\| \leq \gamma \|V^\pi - V_k\|$$

Policy Improvement

- Greedy Policy Improvement

$$\pi_{k+1}(s) = \arg \max_a Q^{\pi_k}(s, a)$$

- Policy Improvement Theorem:

Seien π' und π deterministische Policies für die gilt, dass für alle $s \in S$: $Q^\pi(s, \pi'(s)) \geq V^\pi(s)$.

Dann $V^{\pi'}(s) \geq V^\pi(s)$

Value Iteration

- Value Iteration für das Kontrollproblem:

$$V_{k+1}(s_t) = \max_a \left[R(s_t, a) + \gamma \sum_{s_{t+1}} T(s_{t+1}|s_t, a) V_k(s_{t+1}) \right]$$

$$Q_{k+1}(s, a) = R(s, a) + \gamma \sum_{s'} T(s'|s, a) \max_{a'} Q_k(s', a')$$

- Konvergiert gegen Q^* für $k \rightarrow \infty$
- Ähnlicher Beweis.

MDP unvollständig — Reinforcement Learning

- Indirektes Reinforcement Learning: Modelbasiert
- Lerne Modell des MDP:
 - ◆ Rewardfunktion R
 - ◆ Transitionswahrscheinlichkeiten P
- Anschließend Planen.
- Exploration / Exploitation Problem:
um R und P gut schätzen zu können, müssen alle Zustände und Zustandsübergänge beobachtet werden. (Im tabellarischen Fall)

Monte-Carlo Methoden

- Lernen von episodischen Interaktionen mit der Umgebung.
- Ziel: Lernen von $Q^\pi(s,a)$.
- Monte-Carlo Schätzung von $Q^\pi(s,a)$: Mittelwert bilden über gesampelte kumulative Rewards.
- Erwartungstreue Schätzung des echten erwarteten Rewards. Varianz fällt mit $1/n$.

Monte-Carlo Methoden

- Berechnungszeit der Schätzung ist unabhängig von der Größe des Zustandsraums.
- Problem: Falls π deterministisch werden viele state-action-Paare $Q(s,a)$ nie beobachtet.
- Probleme beim Policy Improvement-Schritt
- Lösung: stochastische Policies, z.B. ϵ -greedy Policies.

Temporal Difference Learning

- Idee: Updates von Zuständen auf Grund von *Schätzungen* von anderen Zuständen.
- Natürliche Formulierung als Online-Methoden.
- Anwendbar auch für unvollständige Episoden.
- Nachteil gegenüber Monte-Carlo:
 - ◆ Stärkerer Schaden durch Verletzung der Markov-Eigenschaft.

Q-Learning

- Q-Learning Update-Schritt:

$$Q(s_t, a_t) \leftarrow (1 - \alpha_t)Q(s_t, a_t) + \alpha_t \left(R(s_t, a_t) + \gamma \max_{a'} Q(s_{t+1}, a') \right)$$

- Konvergiert gegen Q^* falls

- ◆ Jeder Zustand unendlich oft besucht wird
- ◆ Für den Schrittweitenparameter gilt:

$$\sum_{t=0}^{\infty} \alpha_t = \infty \qquad \sum_{t=0}^{\infty} \alpha_t^2 < \infty$$

- ◆ Beweis folgt durch Theorie der stochastischen Approximation aus dem Beweis für DP-Fall.

Q-Learning

- Off-Policy-Methode. Dadurch wird das Exploration / Exploitation Problem gelöst.
- Lernen einer optimalen Policy π^* während nach einer anderen Policy π' entschieden wird.
- Die Policy π' kann z.B. eine stochastische Policy mit $\pi(s,a) > 0$ für alle s und a sein, damit Q konvergiert.

SARSA

- SARSA: On-Policy Temporal Difference Methode.

$$Q(s_t, a_t) \leftarrow (1 - \alpha_t)Q(s_t, a_t) + \alpha_t (R(s_t, a_t) + \gamma Q(s_{t+1}, a_{t+1}))$$

- Exploration / Exploitation Problem.
- SARSA vollzieht einen 1-Schritt Temporal-Difference Updateschritt.

N-step Returns

- Allgemeine Updaterregel:

$$V(s_t) \leftarrow (1 - \alpha_t)V(s_t) + \alpha_t \Delta V(s_t)$$

- Temporal Difference Methoden machen 1-Schritt Updates:

$$\Delta_1 V(s_t) = R_t + \gamma V(s_{t+1})$$

- Monte-Carlo-Methoden machen dagegen Updates, die auf der gesamten Episode basieren:

$$\Delta_\infty V(s_t) = R_0 + \gamma R_1 + \gamma^2 R_2 + \gamma^3 R_3 + \dots$$

- N-Schritt-Updates:

$$\Delta_n V(s_t) = R_0 + \gamma R_1 + \gamma^2 R_2 + \dots + \gamma^{n-1} R_{n-1} + \gamma^n V(s_n)$$

TD(λ)

$$\begin{array}{cccccccc} & R_0 & R_1 & R_2 & R_3 & \dots & R_k & R_{k+1} & \dots \\ \Delta_1 : & R_0 + \gamma V(s_1) & & & & & & & \\ \Delta_2 : & R_0 + \gamma R_1 + \gamma^2 V(s_2) & & & & & & & \\ \Delta_3 : & R_0 + \gamma R_1 + \gamma^2 R_2 + \gamma^3 V(s_3) & & & & & & & \\ & & & \vdots & & & & & \\ \Delta_k : & R_0 + \gamma R_1 + \gamma^2 R_2 + \gamma^3 R_3 + \dots + \gamma^k V(s_k) & & & & & & & \\ & & & \vdots & & & & & \\ \Delta_\infty : & R_0 + \gamma R_1 + \gamma^2 R_2 + \gamma^3 R_3 + \dots + \gamma^k R_k + \gamma^{k+1} R_{k+1} + \dots & & & & & & & \end{array}$$

TD(λ)

$$\begin{array}{rcl} & & R_0 \quad R_1 \quad R_2 \quad R_3 \quad \dots \quad R_k \quad R_{k+1} \quad \dots \\ w_1 \quad \Delta_1 : & & R_0 + \gamma V(s_1) \\ w_2 \quad \Delta_2 : & & R_0 + \gamma R_1 + \gamma^2 V(s_2) \\ w_3 \quad \Delta_3 : & & R_0 + \gamma R_1 + \gamma^2 R_2 + \gamma^3 V(s_3) \\ & & \vdots \\ w_k \quad \Delta_k : & & R_0 + \gamma R_1 + \gamma^2 R_2 + \gamma^3 R_3 + \dots + \gamma^k V(s_k) \\ & & \vdots \\ w_\infty \quad \Delta_\infty : & & R_0 + \gamma R_1 + \gamma^2 R_2 + \gamma^3 R_3 + \dots + \gamma^k R_k + \gamma^{k+1} R_{k+1} + \dots \end{array}$$

- Idee: gewichtete Summe aller n-step Returns

$$\Delta = \sum_{k=1}^{\infty} w_k \Delta_k V(s_0)$$

TD(λ)

$$\begin{array}{rcl} & & R_0 \quad R_1 \quad R_2 \quad R_3 \quad \dots \quad R_k \\ (1 - \lambda) & \Delta_1 : & R_0 + \gamma V(s_1) \\ (1 - \lambda)\lambda & \Delta_2 : & R_0 + \gamma R_1 + \gamma^2 V(s_2) \\ (1 - \lambda)\lambda^2 & \Delta_3 : & R_0 + \gamma R_1 + \gamma^2 R_2 + \gamma^3 V(s_3) \\ & & \vdots \\ (1 - \lambda)\lambda^{k-1} & \Delta_k : & R_0 + \gamma R_1 + \gamma^2 R_2 + \gamma^3 R_3 + \dots + \gamma^k V(s_k) \\ & & \vdots \end{array}$$

- **TD(λ) Update:**
$$\Delta(\lambda) = \sum_{k=1}^{\infty} (1 - \lambda)\lambda^{k-1} \Delta_k V(s_0)$$
- $0 \leq \lambda \leq 1$ interpoliert zwischen 1-step und MC.

Bias-Variance-Tradeoff

	R_0	R_1	R_2	R_3	\dots	R_k	R_{k+1}	\dots
$\Delta_1 :$	$R_0 + \gamma V(s_1)$							
$\Delta_2 :$	$R_0 + \gamma R_1 + \gamma^2 V(s_2)$							
$\Delta_3 :$	$R_0 + \gamma R_1 + \gamma^2 R_2 + \gamma^3 V(s_3)$							
Weniger Bias			\vdots					
$\Delta_k :$	$R_0 + \gamma R_1 + \gamma^2 R_2 + \gamma^3 R_3 + \dots + \gamma^k V(s_k)$							
			\vdots					
$\Delta_\infty :$	$R_0 + \gamma R_1 + \gamma^2 R_2 + \gamma^3 R_3 + \dots + \gamma^k R_k + \gamma^{k+1} R_{k+1} + \dots$							

Mehr Varianz

Eligibility Traces

- Algorithmische Sicht auf TD(λ)
- Einführung eines zusätzlichen Speichers $e(s)$ für jeden Zustand $s \in S$.
- Nach Beobachtung $\langle s_t, a_t, R_t, s_{t+1} \rangle$, berechne

$$\delta_t \leftarrow R_t + \gamma V(s_{t+1}) - V(s_t)$$

$$e(s_t) \leftarrow e(s_t) + 1$$

- Update für alle Zustände

$$V(s) \leftarrow V(s) + \alpha_t \delta_t e(s)$$

$$e(s) \leftarrow \lambda \gamma e(s)$$

