

Universität Potsdam

Institut für Informatik
Lehrstuhl Maschinelles Lernen



Reinforcement Learning 2

Uwe Dick

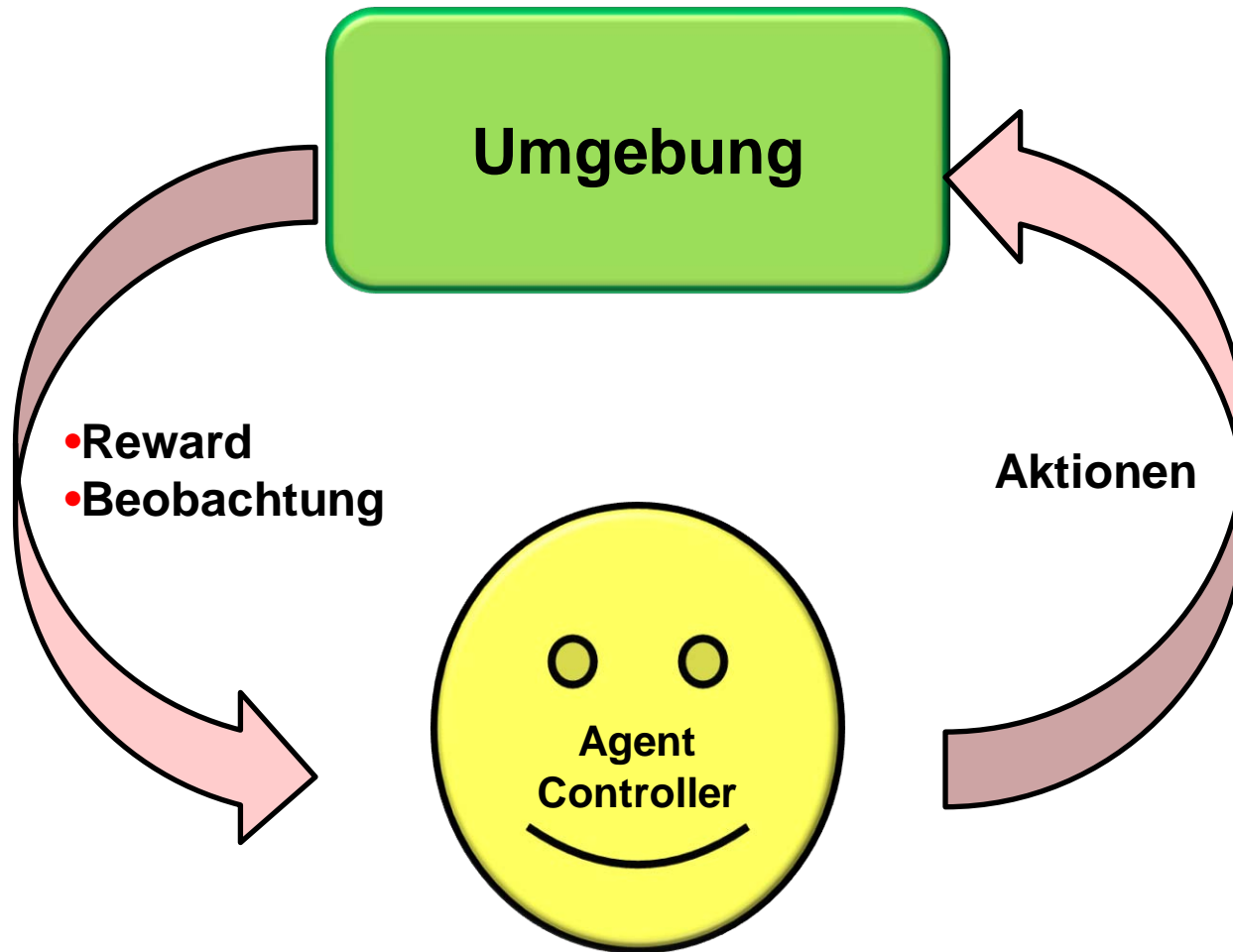
Inhalt

- Erinnerung:
 - ◆ Bellman-Gleichungen, Bellman-Operatoren
 - ◆ Policy Iteration
- Sehr große oder kontinuierliche Zustandsräume
- Monte-Carlo Sampling, UCT
- Diskretisierung
- Approximate Policy Iteration
 - ◆ Bellman Residual Minimization
 - ◆ Least Squares Temporal Difference

Literatur

- Reinforcement Learning. An Introduction.
von Richard S. Sutton und Andrew G. Barto
<http://www.cse.iitm.ac.in/~cs670/book/the-book.html>
- Tutorials auf *videlectures.net*
 - ◆ z.B. von Csaba Szepesvari oder Satinder Singh

Lernen aus Interaktionen



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.

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

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.

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

- Bewertungsfunktion für Zustand-Aktions-Paar:

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

Bellman-Gleichungen

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

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

- Zustand-Aktions-Bewertungsfunktion:

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

Bellman-Operatoren

- In (linearer) Operatorschreibweise:

$$Q^\pi = T^\pi Q^\pi$$

- Mit linearem Operator T^π :

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

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

- Iteration: $Q_{k+1} = T^\pi Q_k$

Bellman-Gleichungen

- Optimale Value Function:

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

- Optimale Policy:

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

- Rekursive Beziehung:

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

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

Bellman-Optimalitätsgleichungen

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

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

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

Policy Iteration

- Allgemeines Verfahren zum Bestimmen der optimalen Policy.
- Iteriere:
 - ◆ Policy Evaluation:
 - ★ Gegeben Policy π_k , bestimme Q^{π_k}
 - ◆ Policy Improvement:
 - ★ Inferiere verbesserte Policy π_{k+1} aus Q^{π_k}
 - ★ z.B. greedy Policy:

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

Value Iteration

- Iteratives Verfahren zur Berechnung
- von V^π

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

- bzw. Q^π

$$Q_{k+1}(s, a) = R(s, a) + \gamma \sum_{s'} P(s' | s, a) Q_k(s', \pi(s'))$$

- Konvergiert gegen V^π bzw. Q^π für $k \rightarrow \infty$

Value Iteration

- Value Iteration für das Kontrollproblem.
- Für V^* :

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

- für Q^* :

$$Q_{k+1}(s, a) = R(s, a) + \gamma \sum_{s'} P(s'|s, a) \max_{a'} Q_k(s', a')$$

- Konvergiert gegen V^* bzw. Q^* für $k \rightarrow \infty$

TD(λ)

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

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

- **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.

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

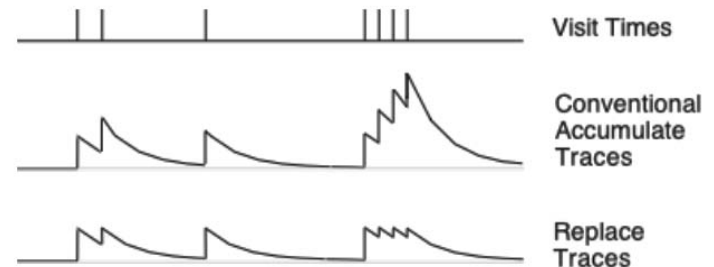
$$\delta_t \leftarrow R_t + \gamma V(s_{t+1}) - V(s_t)$$

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

- Update für alle Zustände

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

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



Problemstellungen

- Lernen einer optimalen Policy.
 - ◆ Oder bestmögliche Approximation.
- Optimales Lernen: Möglichst wenige Fehler während des Lernen.
 - ◆ Exploration / Exploitation Problem.

Exploration / Exploitation Problem

- Tradeoff zwischen
 - ◆ Verfolgen der derzeit besten Policy, um den (greedy) Gewinn zu maximieren.
(Exploitation)
 - ◆ und Erkunden derzeit suboptimaler Aktionen, über deren Wert noch Unsicherheit besteht, um eine potentiell bessere Policy zu finden.
(Exploration)

Bandit Problem

- n-armed bandit Problem:
 - ◆ n Aktionen (Hebel) .
 - ◆ Jede Aktion anderen erwarteten Gewinn.
 - ◆ Erwartete Gewinne unbekannt.
 - ◆ Problem: Finde beste Aktion durch Ausprobieren, ohne dabei zu viel zu verlieren.

- Erwarteter Gewinn für Aktion a ist $Q^*(a)$.

- Schätzung des erwarteten Gewinns nach t

Versuchen:

$$Q_t(a) = \frac{R_1 + R_2 + \dots + R_{n_a(t)}}{n_a(t)}$$

Greedy und ϵ -greedy Policies

- Greedy:

$$\pi = \arg \max_a Q_t(a)$$

- ϵ -greedy

$$\pi = \begin{cases} \arg \max_a Q_t(a) & \text{mit Wahrscheinlichkeit } 1 - \epsilon \\ \text{zufällige Aktion} & \text{mit Wahrscheinlichkeit } \epsilon \end{cases}$$

- ϵ -greedy lässt zufällige Explorationsschritte zu.

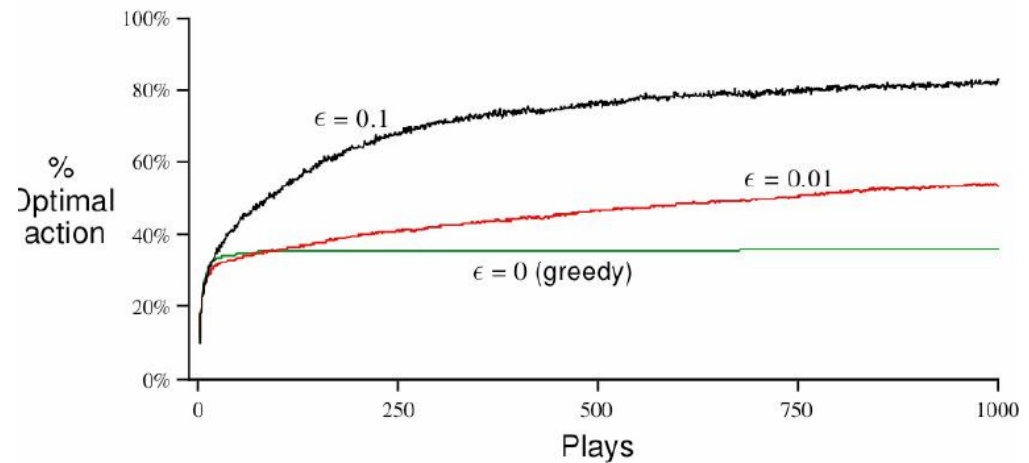
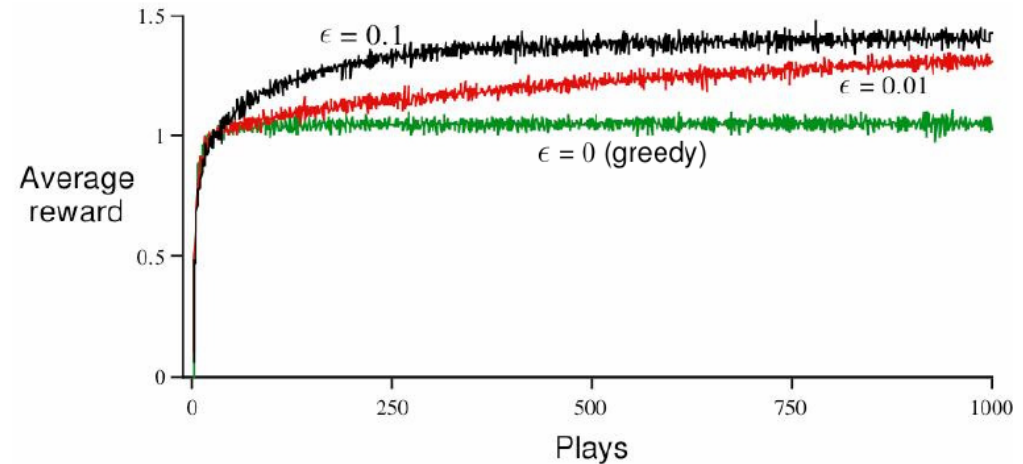
ϵ -greedy Policies

- 10-armed bandit
- 2000 Wiederholungen
- Zufälliges Ziehen von $Q^*(a)$ für alle a :

$$Q^*(a) \sim \mathcal{N}(0, 1)$$

- Rewards werden gezogen aus

$$R_t(a) \sim \mathcal{N}(Q^*(a), 1)$$



Stochastische Policy: Softmax

- π stochastische Policy.
- Schätzungen sollen Einfluss auf Auswahlwahrscheinlichkeit haben.
→ Softmax

- Beispiel: Gibbs-Verteilung:

$$\pi(a) = \frac{e^{Q_t(a)/\tau_t}}{\sum_{i=1}^{|A|} e^{Q_t(a_i)/\tau_t}}$$

- τ_t ist Temperaturparameter.

Optimismus bei Unsicherheit

- Ein Prinzip zur Auflösung des Exploration/Exploitation Dilemmas ist „Optimismus bei Unsicherheit“.
- Existieren auch Umgebungen, in denen Performance schlecht ist.
- Implementierbar z.B. über sehr hohe Initialisierungswerte von Q .

Optimismus bei Unsicherheit

- Upper Confidence Bound (UCB): [Auer et al. 02]
 - ◆ Angenommen, Rewards sind in $[0,1]$.

$$\pi = \arg \max_a \left[Q_t(a) + \sqrt{\frac{c_e \log(t)}{2n_a(t)}} \right], c_e \geq 2$$

- Für stationäre Umgebungen und iid Rewards sehr gute Ergebnisse.

Problemstellungen

- P, R bekannt. $P(s'|s, a)$ können abgefragt werden.
- P, R nicht explizit bekannt. Aber aus den Verteilungen $P(s'|s, a)$ kann gesampelt werden. Annahme: Generatives Modell von P und R .
- P, R nicht oder teilweise bekannt. Es kann Erfahrung gesammelt werden durch Interaktion mit der Umgebung.
→ Reinforcement Learning.
- Batch Reinforcement Learning: Es muss von einer fixen Menge von Beispielen gelernt werden.

Große und unendliche Zustandsräume

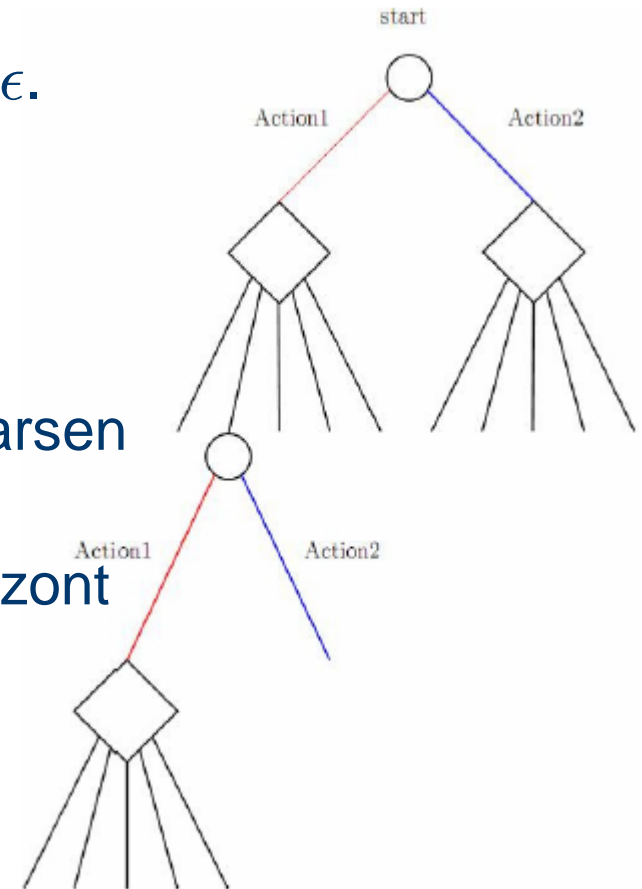
- In realistischen Anwendungen sind Zustandsräume i.A. sehr groß bzw. kontinuierlich.
- Bisherige Annahme: tabellarische Repräsentation der Value Function.
- Mögliche Lösungen:
 - ◆ Planen:
 - ★ Monte-Carlo Sampling
 - ★ Diskretisierung und anschließend z.B. Value Iteration
 - ◆ Approximation der Value Function durch Funktionsapproximationsmethoden.
 - ◆ Direktes Lernen der Policy.

Monte-Carlo Sampling

- Angenommen, S sehr groß
- Ziel: Finde Q , so dass $\|Q - Q^*\|_\infty < \epsilon$.

- Sparse Lookahead Trees:
[Kearns et al. 02]

- ◆ Monte-Carlo: Samplen eines sparsen Aktions-Zustands-Baums.
- ◆ Tiefe des Baums: Effektiver Horizont
 $H(\epsilon) = R_{\max} / (\epsilon(1-\gamma))$
- ◆ MC unabhängig von $|S|$
- ◆ Aber exponentiell in $H(\epsilon) |A|^{H(\epsilon)}$
min. Größe des Baums



Upper Confidence Bounds for Trees

- Besser: Nur solche Teilbäume genauer untersuchen, die vielversprechend sind.
- Optimismus bei Unsicherheit!
 - ◆ Nutze das gleiche Prinzip wie bei Bandit Problem.
 - ◆ UCT: UCB for Trees.
[Kocsis & Szepesvári 06]

$$\pi(s, a) = \arg \max_a \left[Q_t(s, a) + \sqrt{\frac{c_e \log(t)}{2n_{s,a}(t)}} \right], c_e \geq 2$$

UCT Performance: Go

- Sehr gute Resultate in Go.
- 9x9 & 19x19
- Computer Olympiade 2007 - 2009:
 - ◆ 2007 & 2008: 1.-3. Platz verwenden Varianten von UCT.
 - ◆ Im Allgemeinen: Monte-Carlo Search Trees (MCST).
 - ◆ 2009: Mindestens 2. und 3. verwenden Varianten von UCT.

Diskretisierung

- Kontinuierlicher Zustandsraum S .
- Random Discretization Method: [Rust 97]
 - ◆ Sampling von Zuständen S' nach uniformer Verteilung über den Zustandsraum.
 - ◆ Value Iteration.
- Kontinuierliche Value Iteration:

$$V_{t+1}(s) = \max_a \left[R(s, a) + \gamma \int_{s'} p(s'|s, a) V_t(s') ds' \right]$$

- Diskretisierung: Weighted Importance Sampling

$$\frac{\sum_{i=1}^N p(s_i|s, a) V(s_i)}{\sum_{j=0}^N p(s_j|s, a)} \rightarrow \int_{s'} p(s'|s, a) V(s') , \text{ für } N \rightarrow \infty$$

Diskretisierung

- Berechnen der Value Function $V(s)$ für Zustände, die nicht in der Samplingmenge S' sind:
- Bellman-Update –Schritt

$$V^*(s) = \max_a \left[R(s, a) + \gamma \sum_{i=1}^N \frac{p(s_i | s, a)}{\sum_{j=0}^N p(s_j | s, a)} V^*(s_i) \right]$$

- Garantierte Performance: [Rust97]
Annahme: $S=[0,1]^d$

$$E[\|V_N(s) - V^*(s)\|_\infty^2] \leq \frac{Cd|A|^{5/4}}{(1-\gamma)^2 N^{1/4}}$$

Diskretisierung

- Kontrollproblem:

$$\pi^*(s) = \arg \max_a \left[R(s, a) + \gamma \sum_{i=1}^N \frac{p(s_i | s, a)}{\sum_{j=0}^N p(s_j | s, a)} V(s_i) \right]$$

- Ähnliche Garantien. [Sze01]

Funktionsapproximation

- Darstellen der Value Function als parametrisierte Funktion aus dem Funktionsraum \mathcal{F} mit Parametervektor θ .

$$\hat{V}(s; \theta) \qquad \hat{Q}(s, a; \theta)$$

- Vorhersageproblem: Finde Parametervektor θ , so dass V^π , bzw. Q^π am besten approximiert wird.

$$\theta^\pi = \arg \min_{\theta} \sum_s \mu(s) |V^\pi(s) - \hat{V}(s; \theta)|^2$$

$$\theta^\pi = \arg \min_{\theta} \sum_s \sum_a \mu(s) |Q^\pi(s, a) - \hat{Q}(s, a; \theta)|^2$$

Fitted Value Iteration mit Samples

- [Szepesvári & Munos 05]
- $V = 0$.
- Ziehe N Zustände s aus $\mu(s)$.
- Für jedes s und $a \in A$, Ziehe M Nachfolgezustände s' aus $P(\cdot|s,a)$ und Rewards $R(s,a)$.
- Iteriere:
 - ◆ Mit diesen Samples $\langle s, a, R, s' \rangle$ wird ein Bellman-Update-Schritt durchgeführt:

$$V_k(s) \leftarrow \max_a \left[\frac{1}{M} \sum_{i=1}^M R_i(s, a) + \gamma V(s'_i) \right]$$

- ◆ Dann least-squares Fitting:

$$V \leftarrow \arg \min_{f \in \mathcal{F}} \sum_{i=1}^N |V_k(s_i) - f(s_i)|^2$$

Fehlerabschätzung

$$\|V^* - V^{\pi_K}\|_{p,\rho} \leq \frac{2\gamma}{(1-\gamma)^2} \left\{ C(\mu)^{1/p} \left[d(T\mathcal{F}, \mathcal{F}) + c_1 \left(\frac{\mathcal{E}}{N} (\log(N) + \log(K/\delta)) \right)^{1/2p} + c_2 \left(\frac{1}{M} (\log(N|A|) + \log(K/\delta)) \right)^{1/2} \right] + c_3 \gamma^K K_{\max} \right\}$$



FA für Reinforcement Learning

- Das Reinforcement Learning Problem:
 - ◆ Beispiele $\langle s_t, a_t, R_t, s_{t+1} \rangle$ aus Interaktion mit der Umgebung.
 - ◆ Annahme: Interaktion folgt der zu lernenden Policy
 - ◆ On-policy-Verteilung von Zuständen $\mu(s)$.
- Vorhersageproblem: Finde

$$\theta^\pi = \arg \min_{\theta} \sum_s \mu(s) |V^\pi(s) - \hat{V}(s; \theta)|^2$$

FA für Reinforcement Learning

- Online Updates: Anpassen von θ_t nach jeder Interaktion $\langle s_t, a_t, R_t, s_{t+1} \rangle$.

$$\hat{V}(\cdot; \theta_t) \rightarrow V^\pi, \text{ für } t \rightarrow \infty$$

- Gradientenabstieg:

$$\begin{aligned}\theta_{t+1} &= \theta_t - \frac{1}{2} \alpha_t \nabla_{\theta} [V^\pi(s_t) - \hat{V}(s_t; \theta)]^2 \\ &= \theta_t + \alpha_t [V^\pi(s_t) - \hat{V}(s_t; \theta)] \nabla_{\theta} \hat{V}(s_t; \theta)\end{aligned}$$

FA für Reinforcement Learning

- Value Function V^π unbekannt. Ersetze mit Schätzung.
 - ◆ Monte-Carlo: Erwartungstreue Schätzung von V^π .
 - Konvergenz zu lokalem Optimum.
(Unter Bedingungen für α_t)
 - ◆ Temporal Difference (TD(0)): Gebiaste Schätzung.
 - keine Konvergenz zu lokalem Optimum beweisbar.
- Spezialfall: lineare Methoden.

$$\hat{V}(s; \theta_t) = \phi(s)^T \theta_t$$

FA für Reinforcement Learning

- TD(λ)

- ◆ Eligibility traces:

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

$$e_t \leftarrow \gamma \lambda e_{t-1} + \nabla_{\theta} V(s_t)$$

$$\theta_{t+1} \leftarrow \alpha_t \delta_t e_t$$

- ◆ Lineare Methode: Konvergenzgarantie nur für on-policy.
- ◆ Fehlerabschätzung:

$$\begin{aligned} \lim_{t \rightarrow \infty} \sum_s \mu(s) |V^{\pi}(s) - \hat{V}(s; \theta_t)|^2 \\ \leq \frac{1 - \gamma \lambda}{1 - \gamma} \sum_s \mu(s) |V^{\pi}(s) - \hat{V}(s; \theta^*)|^2 \end{aligned}$$

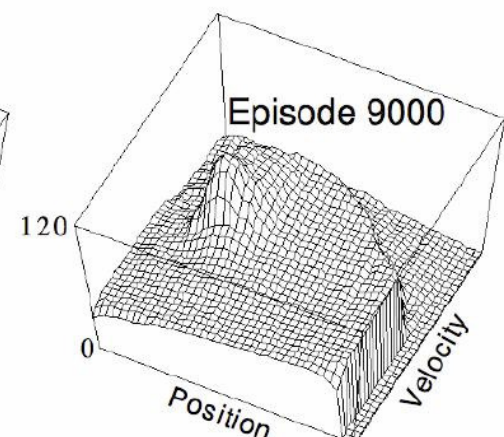
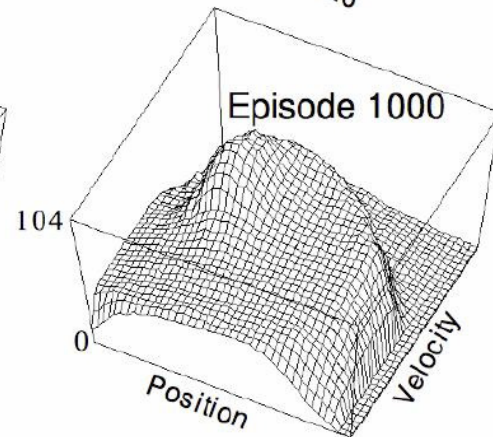
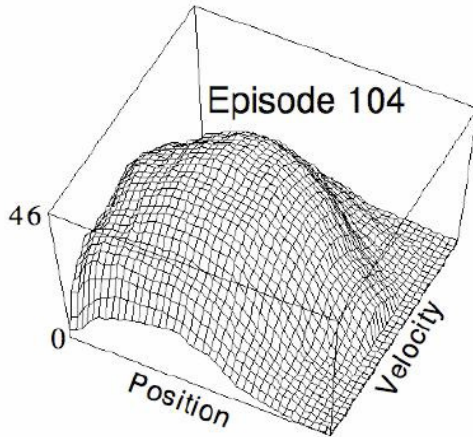
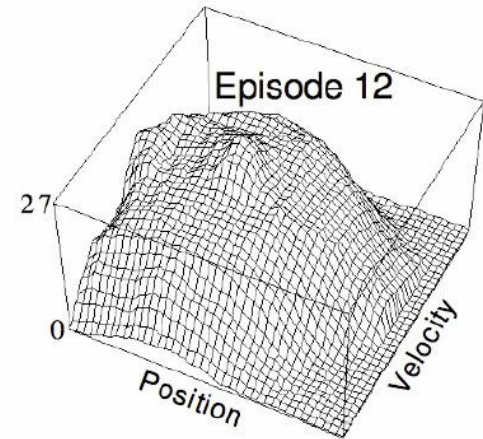
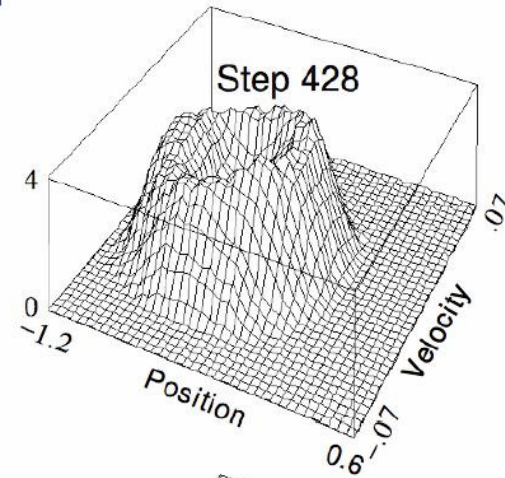
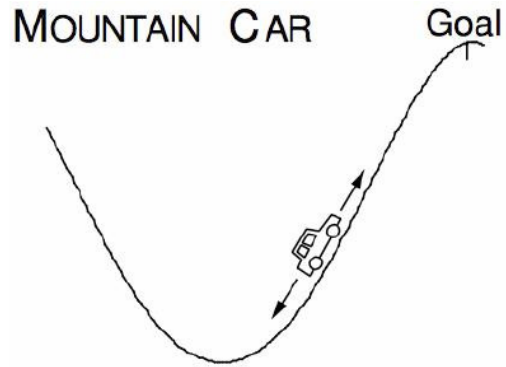
SARSA(λ)

- Kontrollproblem: SARSA(λ) (On-Policy)

$$\begin{aligned}\delta_t &\leftarrow R(s_t, a_t) + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \\ e_t &\leftarrow \gamma \lambda e_{t-1} + \nabla_{\theta} Q(s_t, a_t) \\ \theta_{t+1} &\leftarrow \alpha_t \delta_t e_t\end{aligned}$$

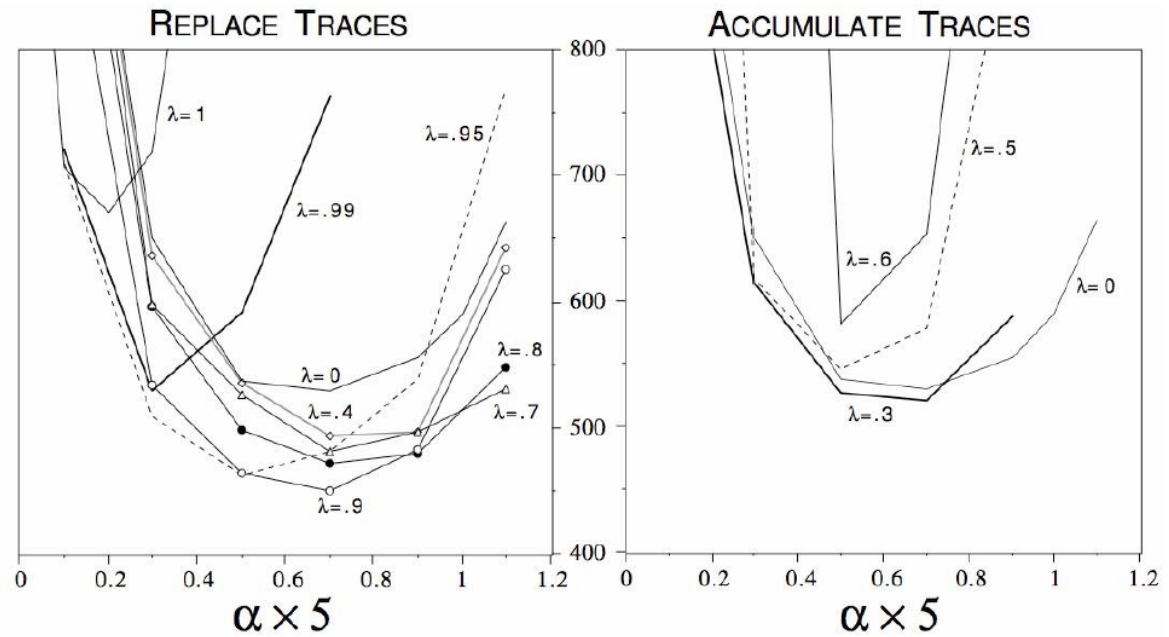
- Off-policy kann divergieren.

SARSA(λ)



SARSA(λ)

Steps per episode
averaged over
first 20 trials
and 30 runs



Approximate Policy Iteration

- Im Folgenden: lineares Modell.

$$\hat{Q}(\phi(s, a); \theta) = \phi(s, a)^T \theta$$

- Approximate Policy Evaluation:
 - ◆ Lernen der optimalen state-action value function \hat{Q}^π von
 - ★ Interaktion
 - ★ fester Trainingsmenge
- Policy Improvement

Approximate Policy Iteration

- Falls Samples von $Q^\pi(s,a)$ bekannt, lerne Q^π vom Trainingssample mit Hilfe eines überwachten Regressionsmethode.
- Problem: Oft off-policy, d.h. Trainingsbeispiele werden beobachtet während einer Verhaltenspolicy gefolgt wird.
 - ◆ Sample Selection Bias (Unterschiedliche Training- und Testverteilungen)

Bellman-Residuen-Minimierung

- Temporal Difference Methode.
- Bellman-Gleichung als Fixpunkt-Gleichung.

$$Q^\pi - T^\pi Q^\pi = 0$$

- Linke Seite als Fehler interpretieren: Bellman Residuum. μ stationäre Verteilung von Zuständen.

$$L_{BRM}(Q; \pi) = \|Q - T^\pi Q\|_\mu^2$$

- Empirisch:

$$\begin{aligned} \hat{L}_{BRM}(Q; \pi, n) \\ = \frac{1}{n|A|} \sum_{t=1}^n \left[Q(s_t, a_t) - (R(s_t, a_t) + \gamma Q(s_{t+1}, \pi(s_{t+1}))) \right]^2 \end{aligned}$$

Bellman-Residuen-Minimierung

- Problem: Schätzer \hat{L}_{BRM} nicht erwartungstreu.

$$E[\hat{L}_{BRM}(Q; \pi, n)] \neq L_{BRM}(Q; \pi)$$

- Denn

$$L_{BRM}(Q; \pi) = E_{s \sim \mu, a} \left[\left(Q(s, a) - T^\pi Q(s, a) \right)^2 \right]$$

$$(T^\pi Q)(s, a) = E_{s' \sim P} \left[R(s, a) + \gamma Q(s', \pi(s')) \right]$$

- Es folgt:

$$L_{BRM}(Q; \pi)$$

$$= E_{s \sim \mu, a} \left[\left(Q(s, a) - E_{s' \sim P} \left[R(s, a) + \gamma Q(s', \pi(s')) \right] \right)^2 \right]$$

Bellman-Residuen-Minimierung

- Aber für $E[\hat{L}(Q; \pi)] = \hat{L}(Q; \pi, n)$, für $n \rightarrow \infty$ gilt:

$$\begin{aligned} & E_{s \sim \mu, a, s' \sim P} [\hat{L}_{BRM}(Q; \pi)] \\ &= E_{s \sim \mu, a, s' \sim P} \left[\left(Q(s, a) - R(s, a) + \gamma Q(s', \pi(s')) \right)^2 \right] \\ &= E_{s \sim \mu, a} \left[E_{s' \sim P} \left[\left(Q(s, a) - R(s, a) + \gamma Q(s', \pi(s')) \right)^2 \right] \right] \end{aligned}$$

- Es gilt aber für Erwartungswerte über Zufallsvariablen X :

$$E[X^2] = E[X]^2 + \text{Var}[X]$$

Bellman-Residuen-Minimierung

- Anwendung auf inneren Erwartungswert:

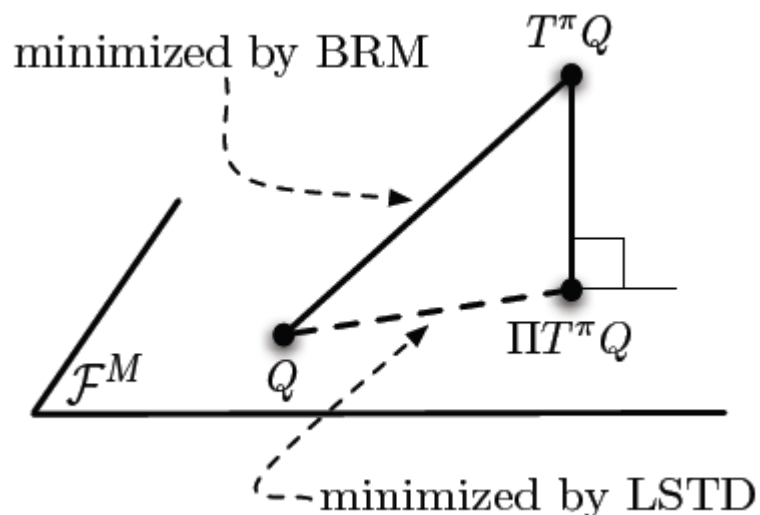
$$\begin{aligned} & E_{s'} \left[\left(Q(s, a) - (R(s, a) + \gamma Q(s', \pi(s'))) \right)^2 \right] \\ &= E_{s'} \left[Q(s, a) - (R(s, a) + \gamma Q(s', \pi(s'))) \right]^2 \\ &\quad + \text{Var}_{s'} \left[Q(s, a) - (R(s, a) + \gamma Q(s', \pi(s'))) \right] \\ &= \left(Q(s, a) - (T^\pi Q)(s, a) \right)^2 \\ &\quad + \text{Var}_{s'} \left[R(s, a) + \gamma Q(s', \pi(s')) \right] \end{aligned}$$

- Der Varianzterm wirkt ähnlich wie ein Regularisierer
→ Bias.

BRM

- Vorschlag: [Antos et. al. 07]
Erwartungstreue durch Einführung einer Hilfsfunktion $h \in \mathcal{F}$.

$$L_{BRM}(Q, h; \pi) = \|Q - T^\pi Q\|_\mu^2 - \|h - T^\pi Q\|_\mu^2$$



Least-Squares Temporal Difference

- Q ist aus Funktionsraum \mathcal{F} .
- $T^\pi Q$ aber nicht notwendigerweise.
- LSTD minimiert den quadratischen Abstand zwischen Q und der Projektion von $T^\pi Q$ auf \mathcal{F} .

$$L_{LSTD}(Q; \pi) = \|Q - \Pi T^\pi Q\|_\mu^2$$

$$\Pi f = \arg \min_{h \in \mathcal{F}} \|h - f\|_\mu$$

- Unbiased.
- LSTD oft bessere Ergebnisse.

Batch Reinforcement Learning

- Episode gesampelt nach π_b
- Zum Trainingszeitpunkt nur Zugang zu dieser einen Episode.

$$\begin{aligned} & \hat{L}_{BRM}(Q; \pi, n) \\ &= \frac{1}{n|A|} \sum_{t=1}^n \frac{1}{\pi_b(a_t|s_t)} \left[Q(s_t, a_t) - (R(s_t, a_t) + \gamma Q(s_{t+1}, \pi(s_{t+1}))) \right]^2 \end{aligned}$$

Literatur

- [Auer et al. 02]: P.Auer, N.Cesa-Bianchi and P.Fischer: Finite time analysis of the multiarmed bandit problem. Machine Learning 47, 2002.
- [Kearns et al. 02]: M.J. Kearns, Y. Mansour, A.Y. Ng: A sparse sampling algorithm for near-optimal planning in large Markov decision processes. Machine Learning 49: 2002.
- [Kocsis & Szepesvári 06]: L. Kocsis and Cs. Szepesvári: Bandit based Monte-Carlo planning. ECML, 2006.
- [Rust 97]: J. Rust, 1997, Using randomization to break the curse of dimensionality, Econometrica, 65:487—516, 1997.
- [Szepesvári & Munos 05]: Cs. Szepesvári and R. Munos: Finite time bounds for sampling based fitted value iteration, ICML, 2005.
- [Antos et. al. 07]: A. Antos, Cs. Szepesvari and R. Munos: Learning near-optimal policies with Bellman-residual minimization based fitted policy iteration and a single sample path, Machine Learning Journal, 2007