Monte Carlo Methoden

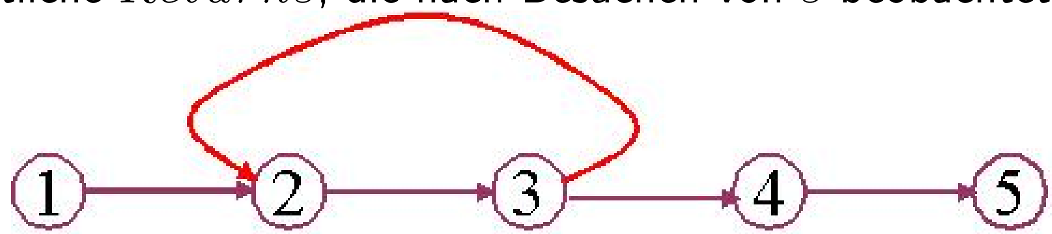
Ziele dieser Vorlesung:

- Monte Carlo Methoden lernen von *vollständigen* Beispiel-*Returns*
  - Sind nur für episodische Aufgaben definiert
- Monte Carlo Methoden lernen direkt aus Erfahrung
  - *Online*: Kein Modell nötig, erreicht trotzdem Optimum
  - *Simuliert*: Kein *vollständiges* Modell nötig

Dieser Teil ist aus “Reinforcement Learning: An Introduction”, Richard S. Sutton and Andrew G. Barto

Monte Carlo *Policy* Evaluierung

- $Ziel$: lerne $V^\pi(s)$
- $Gegeben$: eine Anzahl Episoden für π die s enthalten
- $Idee$: durchschnittliche $Returns$, die nach Besuchen von s beobachtet werden



- $Every - Visit\, MC$: durchschnittliche $Returns$ für $jedes$ Mal, dass s in einer Episode besucht wird
- $First - Visit\, MC$: durchschnittliche $Returns$ für das $erste$ Mal, dass s in einer Episode besucht wird
- Beide konvergieren asymptotisch

$First-Visit$ Monte Carlo $Policy$ Evaluierung

Initialisiere:

$\pi \leftarrow$ zu evaluierende $Policy$

$V \leftarrow$ eine beliebige Zustands-Wertefunktion

$Returns(s) \leftarrow$ eine leere Liste, für alle $s \in S$

Wiederhole unendlich:

(a) Generiere eine Episode mit Hilfe von π

(b) Für jeden Zustand s, der in der Episode vorkommt:

$R \leftarrow\ Return$ nach dem ersten Auftreten von s

Hänge R an $Returns(s)$ an

$V(s) \leftarrow$ Durchschnittliche $(Returns(s))$

Blackjack Beispiel

- Ziel: erhalte eine größere Kartensumme als der Kartengeber ohne 21 zu übersteigen

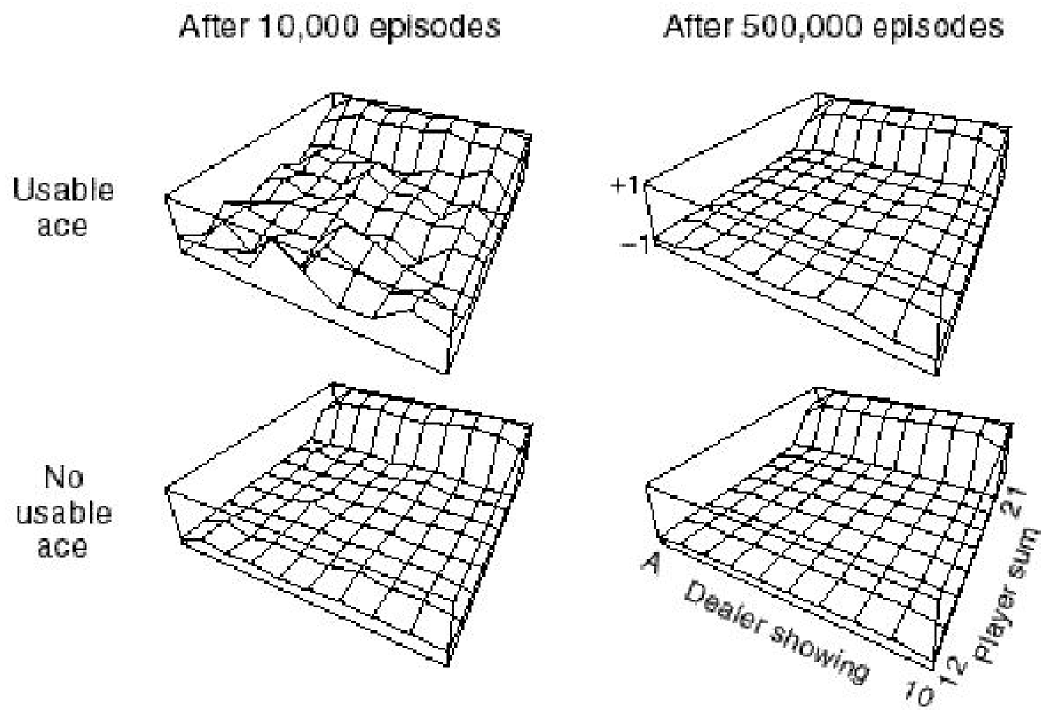
- Zustände (200 gibt es):
  - gegenwärtige Summe (12 - 21)
  - die sichtbaren Karten des Kartengebers (Ass - 10)
  - habe ich ein brauchbares Ass?

- $Reward$: $+1$ für den Sieg, 0 für Unentschieden, -1 bei Niederlage

- Aktionen: Rest (fordere keine Karten mehr), Karte (fordere eine weitere Karte)

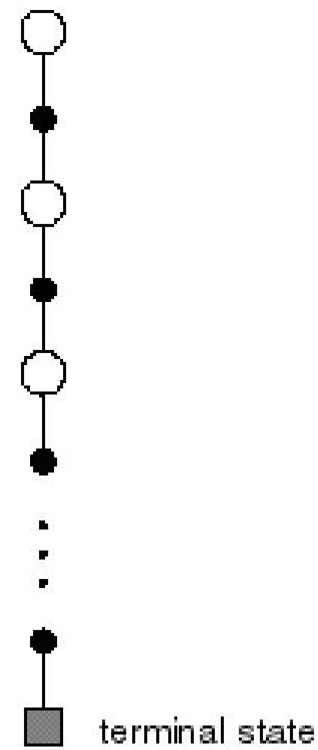
- $Policy$: Rest, wenn meine Summe 20 oder 21 beträgt, ansonsten Karte

Blackjack Wertefunktionen



Backup Diagramm für Monte Carlo

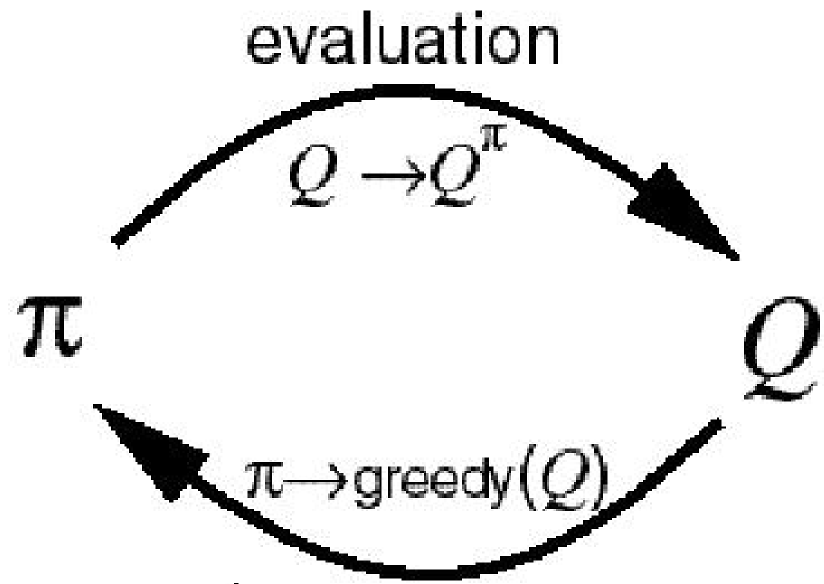
- Die ganze Episode ist enthalten
- Nur eine Wahl in jedem Zustand (anders als bei DP)
- Bei MC gibt es kein *Bootstrapping*
- Die Zeit, die zur Schätzung eines Zustands nötig ist, hängt nicht von der gesamten Zahl der Zustände ab



Monte Carlo Schätzung der Aktions-Werte (Q)

- Monte Carlo ist am nützlichsten, wenn kein Modell zur Verfügung steht
  - Wir wollen Q^* lernen
- $Q^\pi(s, a)$ - durchschnittlicher $Return$ beginnend beim Zustand s und bei Aktion a und Folgen von π
- Konvergiert auch asymptotisch $wenn$ jedes Zustands-Aktions Paar besucht wird
- Exploriere die $Starts$: jedes Zustands-Aktions-Paar hat eine Wahrscheinlichkeit ungleich Null, das Start-Paar zu sein

Monte Carlo *Control*



Verbesserung

- MC *Policy* Iteration: *Policy* Evaluierung mit MC Methoden, gefolgt von *Policy* Verbesserung

- *Policy* Verbesserungs-Schritt: "Greedifiziere" mit Bezug auf die Werte-Funktion (oder die Aktions-Werte-Funktion)

Konvergenz von MC *Control*

- Das $Policy$ Verbesserungs-Theorem sagt uns:

$$\begin{aligned} Q^{\pi_k}(s, \pi_{k+1}(s)) &= Q^{\pi_k}(s, \arg\max_a Q^{\pi_k}(s,a)) \\ &= \max_a Q^{\pi_k}(s,a)) \\ &\geq Q^{\pi_k}(s, \pi_k(s)) \\ &= V^{\pi_k}(s) \end{aligned}$$

- Dies setzt Explorierende $Starts$ und eine unendliche Zahl an Episoden für die MC $Policy$ Evalueriung voraus

- Um Letzteres zu lösen:
  - nur bis zu einem gegebenen Gütegrad updaten
  - wechsle zwischen Evaluierung und Verbesserung per Episode

Monte Carlo Explorierende *Starts*

Initialisiere für alle $s \in S, a \in A(s)$:

$Q(s, a) \leftarrow$ beliebig

$\pi(s) \leftarrow$ beliebig

$Returns(s, a) \leftarrow$ leere Liste

Wiederhole unendlich:

(a) Generiere eine Epsiode mit Hilfe Explorierender Starts und π

(b) Für jedes Paar s, a das in der Episode auftaucht:

$R \leftarrow Return$ nach dem ersten Auftreten von s, a

Hänge R an $Returns(s, a)$ an

$Q(s, a) \leftarrow$ durchschnittliche $(Returns(s, a))$

(c) Für jedes s in der Episode:

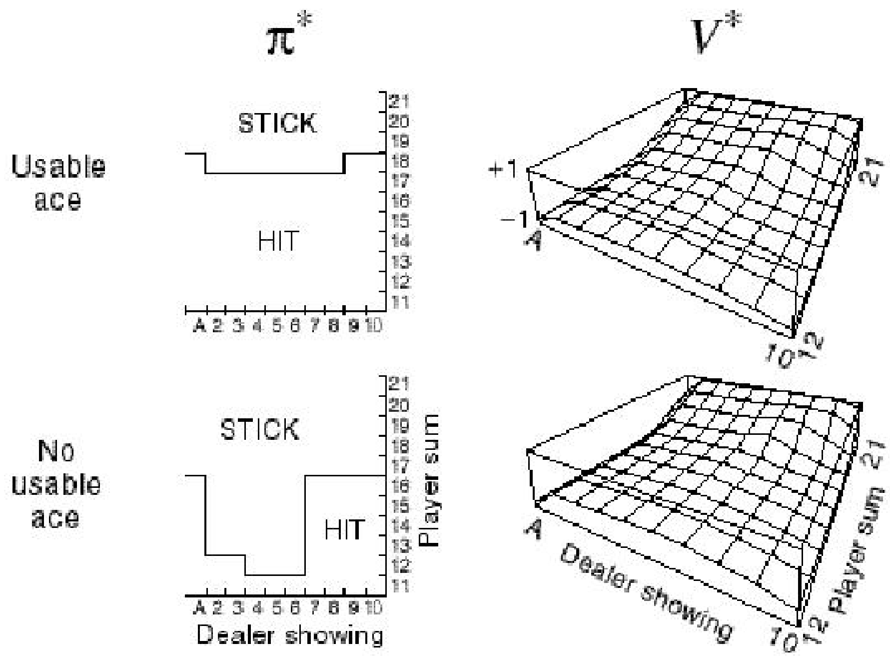
$\pi(s) \leftarrow \arg\max_a Q(s, a)$

Der Fixpunkt ist die optimale $Policy\, \pi^*$

Der Beweis steht noch aus

Blackjack Beispiel Forts.

- Explorierende $Starts$
- Initiale $Policy$ wie zuvor beschrieben



On – Policy Monte Carlo *Control*

- $On - Policy$: erlerne die gegenwärtig ausgeführte $Policy$
- Wie werden wir Explorierende $Starts$ los?
  - Wir brauchen $soft\,Policies$: $\pi(s,a) > 0$ für alle s und a
  - z.B. ϵ-$soft$ $Policy$:

$$\frac{\epsilon}{|A(s)|} \qquad\qquad 1-\epsilon+\frac{\epsilon}{|A(s)|}$$

non-max greedy

- Wie bei GPI: Nähere die $Policy$ der $greedy\,Policy$ an (z.B. ϵ-$soft$)
- Konvergiert zur besten ϵ-$soft$ $Policy$

On – Policy MC *Control*

Initialisiere für alle $s \in S, a \in A(s)$:

$Q(s, a) \leftarrow$ beliebig

$Returns(s, a) \leftarrow$ leere Liste

$\pi \leftarrow$ eine beliebige ϵ-*soft Policy*

Wiederhole unendlich:

(a) Generiere eine Epsiode mit π

(b) Für jedes Paar s, a das in der Episode auftaucht:

$R \leftarrow Return$ nach dem ersten Auftreten von s, a

Hänge R an $Returns(s, a)$ an

$Q(s, a) \leftarrow$ durchschnittliche $(Returns(s, a))$

(c) Für jedes s in der Episode:

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

Für alle $a \in A(s)$: $$\pi(s,a) \leftarrow \begin{cases} 1-\epsilon+\epsilon/|A(s)| & \text{if } a = a^* \\ \epsilon/|A(s)| & \text{if } a \neq a^* \end{cases}$$

$Off - Policy$ MC $Control$

- Verhaltens-$Policy$ generiert Verhalten im Umfeld
- Schätzungs-$Policy$ ist die $Policy$, die man erlernt

π' folgen und dabei π erlernen

Angenommen wir haben n_s $Returns$, $R_i(s)$, vom Zustand s, jeder mit der Wahrscheinlichkeit $p_i(s)$, von π generiert zu werden und der Wahrscheinlichkeit $p_i^{'}$, von $\pi^{'}$ generiert zu werden. Dann können wir schätzen:

$$V^{\pi}(s) = \frac{\sum_{i=1}^{n_s} \frac{p_i(s)}{p_i^{'}(s)} R_i(s)}{\sum_{i=1}^{n_s} \frac{p_i(s)}{p_i^{'}(s)}}$$

dass von den umgebenden Wahrscheinlichkeiten $p_i(s)$ und $p_i^{'}(s)$ abhängt. Allerdings,

$$p_i(s_t) = \Pi_{k=t}^{\tau_i(s)-1} \pi(s_k, a_k) P_{s_k s_{k+1}}^{a_k}$$

und

$$\frac{p_i(s_t)}{p_i^{'}(s_t)} = \frac{\Pi_{k=t}^{\tau_i(s)-1} \pi(s_k, a_k) P_{s_k s_{k+1}}^{a_k}}{\Pi_{k=t}^{\tau_i(s)-1} \pi'(s_k, a_k) P_{s_k s_{k+1}}^{a_k}} = \Pi_{k=t}^{\tau_i(s)-1} \frac{\pi(s_k, a_k)}{\pi'(s_k, a_k)}$$

So hängt das benötigte Gewicht $p_i(s)/p_i^{'}(s)$ nur von den zwei $Policies$ und nicht von der gesamten Dynamik der Umgebung ab.

$Off - Policy$ MC $Control$

Initialisiere für alle $s \in S, a \in A(s)$:

$Q(s,a) \leftarrow$ beliebig

$N(s,a) \leftarrow 0$; Zähler und

$D(s,a) \leftarrow 0$; Nenner von $Q(s,a)$

$\pi \leftarrow$ eine beliebige deterministische $Policy$

Wiederhole unendlich:

(a) Wähle eine $Policy\, \pi'$ und generiere mit ihr eine Episode:

$$s_0, a_0, r_1, s_1, a_1, r_2, \ldots, s_{T-1}, a_{T-1}, r_T, s_T$$

(b) $\tau \leftarrow$ spätester Zeitpunkt, zu dem $a_\tau \neq \pi(s_\tau)$

(c) Für jedes Paar s, a das in der Episode nach τ auftaucht:

$t \leftarrow$ der Zeitpunkt des ersten Auftretens (nach τ) von s, a

$w \leftarrow \Pi_{k=t+1}^{T-1} \frac{1}{\pi'(s_k, a_k)}$

$N(s, a) \leftarrow N(s, a) + wR_t$

$D(s, a) \leftarrow D(s, a) + w$

$Q(s, a) \leftarrow \frac{N(s,a)}{D(s,a)}$

(d) Für jedes $s \in S$:

$\pi(s) \leftarrow \arg\max_a \; Q(s, a)$

Inkrementelle Implementierung

- MC kann inkrementell implementiert werden
  - spart Speicherplatz
- Errechne den gewichteten Durchschnitt jedes $Returns$

$$V_n = \frac{\sum_{k=1}^{n} w_k R_k}{\sum_{k=1}^{n} w_k}$$

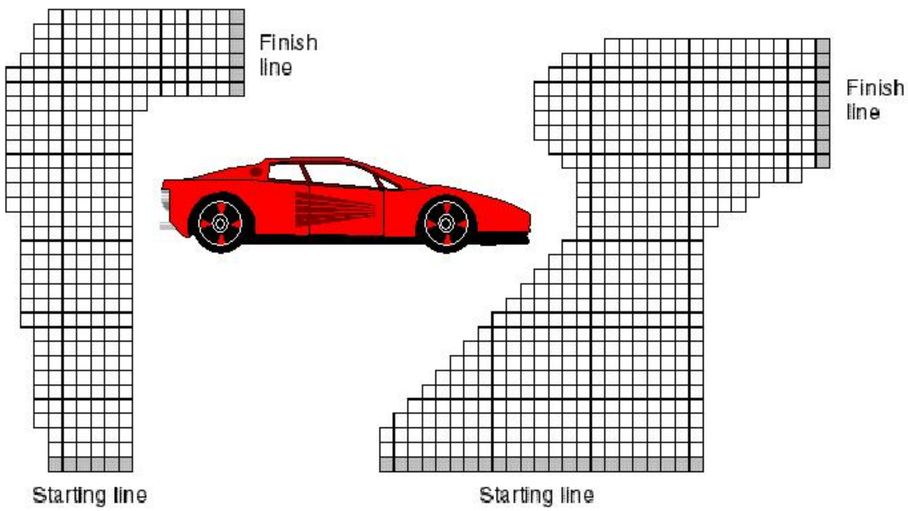
nicht-inkrementell

$$\begin{aligned} V_{n+1} &= V_n + \frac{w_{n+1}}{W_{n+1}}[R_{n+1} - V_n] \\ W_{n+1} &= W_n + w_{n+1} \\ V_0 &= W_0 = 0 \end{aligned}$$

inkrementelles
Äquivalent

Rennbahn Beispiel

- *Zustände*: Planquadrate, Geschwindigkeit horizontal und vertikal
- *Rewards*: -1 auf der Strecke, -5 neben der Strecke
- *Aktionen*: +1, -1, 0 zur Geschwindigkeit
- $0 <$Geschwindigkeit < 5
- Stochastisch: 50% der Zeit bewegt es sich 1 Extra Quadrat hoch oder nach rechts



Zusammenfassung

- MC hat mehrere Vorteile ggü. DP:
  - Es kann direkt von der Interaktion mit der Umgebung lernen
  - Vollständige Modelle sind nicht nötig
  - Man muss nicht alle Zustände für das Lernen benutzen
  - Es entsteht weniger Schaden durch Markovsche Verstöße (dazu später mehr)
- MC Methoden liefern einen alternative $Policy$-Evaluierungs Prozess
- Ein Problem muss man beachten: genügend Exploration beizubehalten
  - Explorieren von $Starts$, $soft\,Policies$
- Kein $Bootstrapping$ (im Gegensatz zu DP)