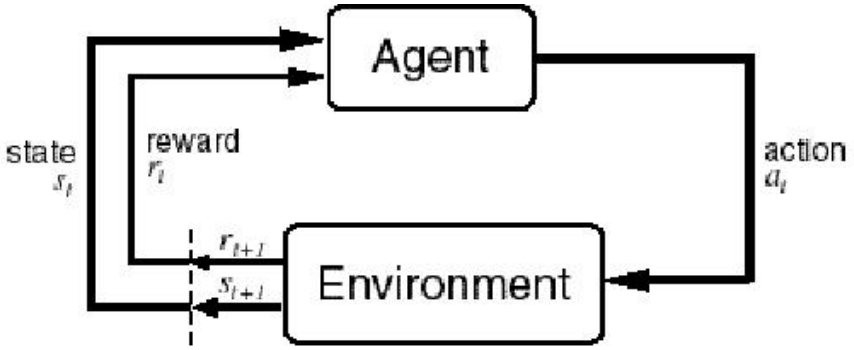
Das Reinforcement Lernproblem

Ziele dieser Vorlesung:

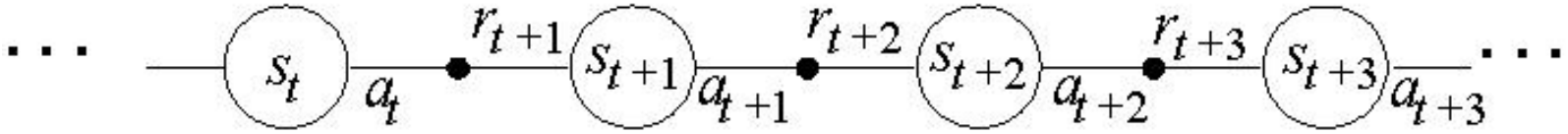
- Beschreibung des RL-Problems das wir in den folgenden Vorlesungen betrachten werden
- Präsentation einer idealisierten Form des RL-Problems für die es konkrete theoretische Betrachtungen gibt
- Einführung der wichtigen mathematischen Komponenten: Wertefunktionen und Bellman-Gleichungen
- Beschreibung des trade-offs zwischen Anwendbarkeit und mathematischer Lenkbarkeit

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

Der Agent in einer Lernumgebung



Agent und Umgebung interagieren zu diskreten Zeitpunkten: $t = 0,1,2...K$
Agent beobachtet Zustand zum Zeitpunkt t : $s_1 \in S$
erzeugt Aktion zum Zeitpunkt t : $a_1 \in A(s_1)$
erhält $Reward$: $r_{t+1} \in \mathcal{R}$
und nächsten Zustand: s_{t+1}



Der Agent lernt eine $Policy$

Policy zum Zeitpunkt t, π_t :
Abbildung von Zuständen auf Aktionswahrscheinlichkeiten
$\pi_t(s, a) =$ Wahrsch., dass $a_t = a$ wenn $s_t = s$

- Reinforcement Lernmethoden beschreiben wie ein Agent seine $Policy$ als Resultat seiner Erfahrung ändert.

- Grob gesagt ist das Ziel des Agenten, auf lange Sicht soviel $Reward$ wie möglich zu erhalten.

Richtiger Abstraktionsgrad

- Zeitschritte müssen keine festen Intervalle der wirklichen Zeit sein.
- Aktionen können $lowlevel$ (z.B., Spannungen von Motoren), oder $highlevel$ sein (z.B., nimm ein Jobangebot an),"mental" (z.B., shift in focus of attention), etc.
- Zustände können $lowlevel$ "Empfindungen", oder abstrakt sein, symbolisch, basierend auf einem Gedächtnis, oder subjektiv (z.B., der Zustand "überrascht zu sein").
- Ein RL Agent ist nicht wie ein ganzes Tier oder ein kompletter Roboter, welche aus mehreren RL Agenten und anderen Komponenten bestehen.
- Die Umgebung ist dem Agenten nicht notwendigerweise unbekannt, nur unvollständig kontrollierbar.
- $Reward$-Berechnung passiert in der Umgebung, da der Agent sie nicht beliebig ändern kann.

Ziele und *Rewards*

- Ist ein skalares $Reward$-Signal eine adäquate Beschreibung für ein Ziel? – vielleicht nicht, aber es ist erstaunlich flexibel.

- Ein Ziel sollte beschreiben **was** wir erreichen wollen und nicht **wie** wir es erreichen wollen.

- Ein Ziel muss außerhalb der Kontrolle des Agenten liegen – also außerhalb des Agenten selbst.

- Der Agent muss Erfolge messen können:
  - explizit;
  - häufig während seiner Lebensdauer.

Returns

Angenommen die Sequenz von Rewards nach Zeitpunkt t ist:

$$r_{t+1}, r_{t+2}, r_{t+3}, \ldots$$

Was wollen wir maximieren?

Generell wollen wir den **erwarteten *Return***, $E\{R_t\}$, in jedem Zeitschritt t maximieren.
Episodische Task : Interaktion teilt sich naturgemäß in Episoden auf, z.B. Durchgänge eines Spiels, Durchläufe durch ein Labyrinth.

$$R_t = r_{t+1} + r_{t+2} + \cdots + r_T$$

wobei T ein finaler Zeitpunkt ist zu dem ein Zielzustand erreicht und die Episode beendet wird.

Returns für kontinuierliche Tasks

Kontinuierliche Task: Interaktion hat keine natürlichen Episoden.

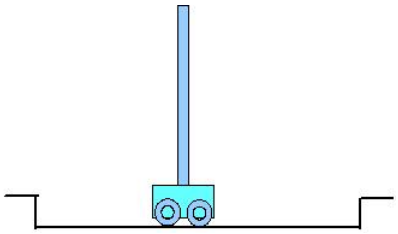
Discounted *Return* :

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1},$$

wobei $\gamma, 0 \leq \gamma \leq 1$, die ***discount rate*** ist.

"kurzsichtig" $0 \leftarrow \gamma \rightarrow 1$ "weitsichtig"

Ein Beispiel



Misserfolg vermeiden: der Stab fällt über einen kritischen Winkel oder der Wagen erreicht das Ende der Strecke

Als **episodische task** wo die Episoden bei Misserfolg enden:

$$\begin{aligned} Reward &= +1 \text{ für jeden Schritt vor dem Misserfolg} \\ \Rightarrow Return &= \text{Anzahl der Schritte bis zum Misserfolg} \end{aligned}$$

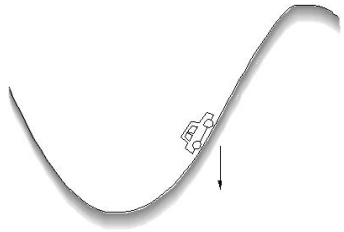
Als **kontinuierliche task** mit $discounted\, Return$:

$$\begin{aligned} Reward &= -1 \text{ bis Misserfolg; 0 sonst} \\ \Rightarrow Return &= -\gamma^k, \text{für } k \text{ Schritte vor dem Misserfolg} \end{aligned}$$

In beiden Fällen wird der Return maximiert, indem Misserfolg
so lange wie möglich vermieden wird.

Ein weiteres Beispiel

Fahre so schnell wie möglich auf die Spitze des Berges.



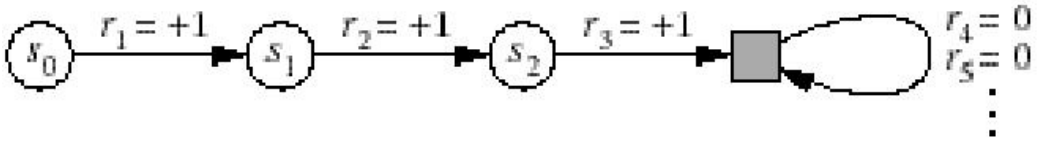
$Reward$ = -1 für jeden Schritt an dem man **nicht** auf der Spitze des Berges ist

$Return$ = $-$Anzahl der Schritte vor Erreichen der Hügelspitze

Der $Return$ wird maximiert indem die Anzahl der Schritte zum Erreichen der Bergspitze minimiert wird.

Vereinheitlichte Notation

- In episodischen Tasks nummerieren wir die Zeitschritte jeder Episode beginnend mit Null.
- Im allgeimeinen differenzieren wir nicht zwischen Episoden. Wir schreiben s_t anstatt $s_{t,j}$ für den Zustand zum Zeitpunkt t in Episode j.
- Betrachte das Ende jeder Episode als einen absorbierenden Zustand der immer einen **Reward** von 0 hat:



- Wir fassen alle Fälle zusammen:

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1},$$

wobei γ nur 1 sein kann wenn immer ein absorbierender Zustand erreicht wird.

Die Markov Wahrscheinlichkeit

- Mit "dem Zustand" zum Zeitpunkt t meinen wir alle Informationen die dem Agenten zum Zeitpunkt t über seine Umgebung zur Verfügung stehen.
- Der Zustand kann sofortige "Wahrnehmungen", weiter verarbeitete Wahrnehmungen und Strukturen, die über eine Sequenz von Wahrnehmungen aufgebaut wurden, beinhalten.
- Idealerweise sollte ein Zustand vergangene Wahrnehmungen zusammenfassen, um alle"wichtigen" Informationen zu erhalten, das heißt er sollte die **Markov Wahrscheinlichkeit** besitzen:

$$Pr\left\{s_{t+1} = s', r_{t+1} = r | s_t, a_t, r_t, s_{t-1}, a_{t-1}, \ldots, r_1, s_0, a_0\right\} = Pr\left\{s_{t+1} = s', r_{t+1} = r | s_t, a_t\right\}$$

Für alle s', r,und $histories\, s_t, a_t, r_t, s_{t-1}, a_{t-1}, \ldots, r_1, s_0, a_0$.

Markov Entscheidungsprozesse

- Wenn eine Reinforcement-Lernaufgabe die Markov Wahrscheinlichkeit besitzt ist dies hauptsächlich ein **Markov Entscheidungsprozess (MDP)**.
- Wenn Zustands- und Aktionsräume endlich sind ist es ein endlicher MDP.
- Um einen endlichen MDP zu definieren braucht man:
  - **Zustands- und aktionsräume**
  - one-step-"Dynamik"definiert über **Übergangswahrscheinlichkeiten:**

$$P^a_{ss'} = Pr\{s_{t+1} = s' | s_t = s, a_t = a\} \forall s, s' \in S, a \in A(s).$$

  - ***Reward*** **Wahrscheinlichkeiten**:

$$R^a_{ss'} = E\{r_{t+1} | s_t = s, a_t = a, s_{t+1} = s'\} \forall s, s' \in S, a \in A(s).$$

Ein Beispiel endlicher MDP

Recycling Roboter

- Zu jedem Zeitpunkt muss der Roboter entscheiden, ob er (1) aktiv eine Dose suchen soll, (2) auf jemanden warten soll, der ihm eine Dose bringt oder (3) zum Aufladen zur Basisstation fahren soll

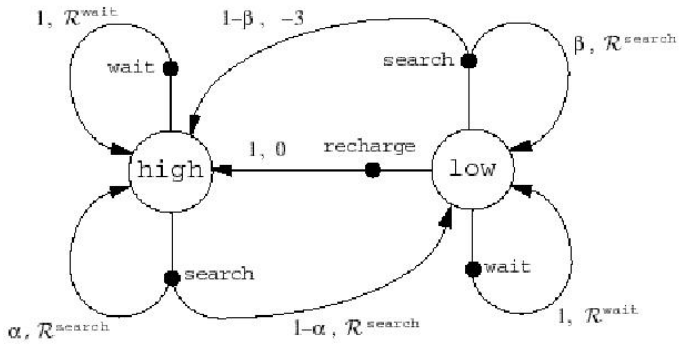
- Suchen ist besser, verbraucht allerdings die Batterien; wenn die Batterien leer laufen während er sucht, muss er "befreit"werden (schlecht)

- Entscheidungen werden auf Grundlage des momentanen Energielevels getroffen: `high, low`

- $Reward =$ Anzahl an gesammelten Dosen

Recycling Roboter MDP

$S = \{\texttt{high}, \texttt{low}\}$
$A\,(\texttt{high}) = \{\texttt{search}, \texttt{wait}\}$
$A\,(\texttt{low}) = \{\texttt{search}, \texttt{wait}, \texttt{recharge}\}$
$R^{\texttt{search}}$ = erwartete Anzahl Dosen während der Suche
$R^{\texttt{wait}}$ = erwartete Anzahl Dosen während des Wartens
$R^{\texttt{search}} > R^{\texttt{wait}}$



Wertefunktionen

- Der **Wert eines Zustandes** ist der erwartete $Return$ beginnend mit diesem Zustand; hängt von der $Policy$ des Agenten ab:

  Zustands-Wertefunktion für $\boldsymbol{Policy\ \pi}$:

$$V^{\pi}(s) = E_{\pi}\left\{R_t | s_t = s\right\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\right\}$$

- Der **Wert wenn eine Aktion in einem Zustand unter der $\boldsymbol{Policy\ \pi}$** gewählt wird ist der erwartete $Return$ beginnend mit diesem Zustand, bei Wählen dieser Aktion und weiterem Verfolgen von π :

  Aktions-Wertefunktion für $\boldsymbol{Policy\ \pi}$:

$$Q^{\pi}(s,a) = E_{\pi}\left\{R_t | s_t = s, a_t = a\right\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\right\}$$

Bellman -Gleichung für *Policy* π

Grundlegende Idee:

$$\begin{aligned} R_t = & \quad r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} + \dots \\ = & \quad r_{t+1} + \gamma \left(r_{t+2} + \gamma r_{t+3} + \gamma^2 r_{t+4} + \dots \right) \\ = & \quad r_{t+1} + \gamma R_{t+1} \end{aligned}$$

Somit:

$$\begin{aligned} V^\pi(s) = & \quad E_\pi \left\{ R_t | s_t = s \right\} \\ = & \quad E_\pi \left\{ r_{t+1} + \gamma V(s_{t+1}) | s_t = s \right\} \end{aligned}$$

Oder, ohne Erwartungswertoperator:

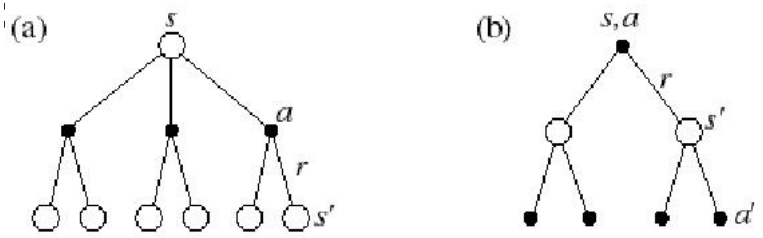
$$V^\pi(s) = \sum_a \pi(s,a) \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma V^\pi(s') \right]$$

Mehr zur Bellman-Gleichung

$$V^\pi(s) = \sum_a \pi(s,a) \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma V^\pi(s') \right]$$

Dies ist eine Menge von Gleichungen (linear), eine für jeden Zustand. Die Wertefunktion für π ist die eindeutige Lösung.

Backup-Diagramme



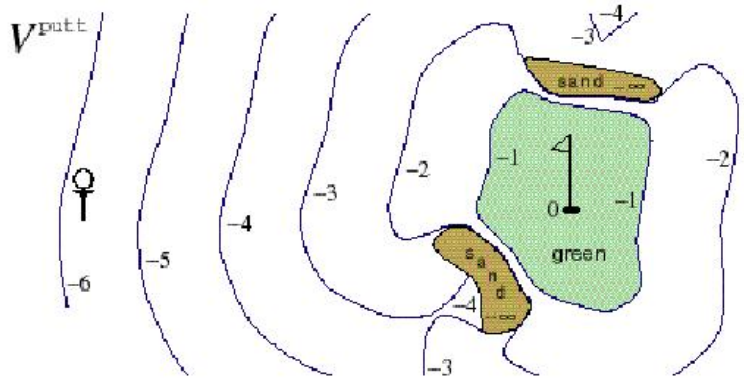
für V^π für Q^π

Gridworld

- Aktionen: norden , süden , osten , westen ; deterministisch.
- Wenn der Agent das Feld verlassen würde: kein Zug, aber $Reward = -1$
- Andere Aktionen geben $Reward = 0$, außer Aktionen die den Agenten aus einem bestimmten Zustand A und B bewegen.

Zustands-Wertefunktion für gleichverteilte Zufalls-$Policy$; $\gamma = 0.9$

Golf



- Zustand ist die Ballposition
- Reward von -1 für jeden Schlag bis der Ball im Loch ist
- Wert eines Zustandes?
- Aktionen: `putt` (benutze Putter) `driver` (benutze Driver)
- `putt` führt auf dem Grün immer zum Erfolg

Optimale Wertefunktionen

- Für endliche MDPs können die $Policies$ **partiell geordnet** werden:

$$\pi \geq \pi' \quad genau\ wenn \quad V^{\pi}(s) \geq V^{\pi'}(s)\ \forall s \in S$$

- Es gibt immer mindestens eine (vielleicht mehrere) $Policies$ die besser als oder gleich allen anderen ist. Dies ist eine **optimale *Policy***. Wir bezeichnen sie alle mit π^*.

- Optimale $Policies$ teilen sich die gleiche **optimale Zustands-Wertefunktion**:

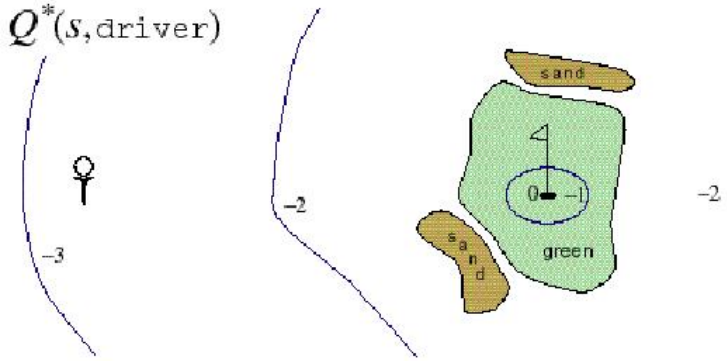
$$V^*(s) = \max_{\pi} V^{\pi}(s)\ \forall s \in S$$

- Optimale $Policies$ teilen sich auch die gleiche **optimale Aktions-Wertefunktion:**

$$Q^*(s,a) = \max_\pi Q^\pi(s,a) \; \forall s \in S \; und \; a \in A(s)$$

Dies ist der erwartete $Return$ nach Wahl der Aktion a in Zustand s und Weiterverfolgen einer optimalen $Policy$.

Optimale Wertefunktionen beim Golf



- Wir können den Ball mit `driver` weiter schlagen als mit `putter`, allerdings mit weniger Genauigkeit

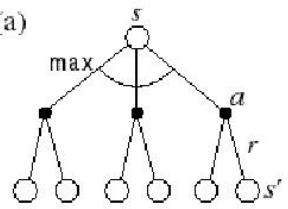
- Q *(s,`driver`) gibt die Werte für die Auswahl von `driver` an, wenn nachfolgend immer die jeweils beste Aktion gewählt wird

Optimale Bellman-Gleichung für V^*

Der Wert eines Zustandes unter einer optimalen $Policy$ ist gleich dem erwarteten $Return$ für die beste Aktion von diesem Zustand an.

$$
\begin{aligned}
V^*(s) &= \max_{a \in A(s)} Q^{\pi^*}(s,a) \\
&= \max_{a \in A(s)} E\left\{ r_{t+1} + \gamma V^*(s_{t+1}) | s_t = s, a_t = a \right\} \\
&= \max_{a \in A(s)} \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma V^*(s') \right]
\end{aligned}
$$

Das entsprechende Backup-Diagramm:

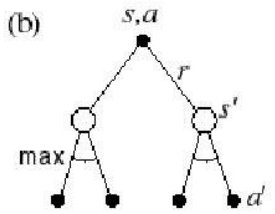


V^* ist die eindeutige Lösung dieses Systems der nichtlinearen Gleichungen.

Optimale Bellman -Gleichung für Q^*

$$
\begin{aligned}
Q^*(s,a) &= E\left\{ r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1}, a') | s_t = s, a_t = a \right\} \\
&= \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma \max_{a'} Q^*(s', a') \right]
\end{aligned}
$$

Das entsprechende Backup-Diagr



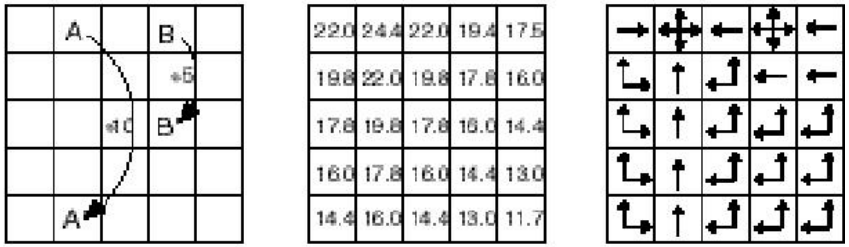
Q^* ist die eindeutige Lösung dieses Systems der nichtlinearen Gleichungen.

Warum optimale Zustands -Wertefunktionen nützlich sind

Eine $Policy$ die mit Rücksicht auf $V^*greedy$ ist, ist eine optimale $Policy$.

Deshalb produziert die $one - step - ahead$ -Suche bei gegebenem V^* auf lange Zeit optimale Aktionen.

z.B., in der gridworld:



a) gridworld b) V^* c) π^*

Was ist mit optimalen Aktions -Wertefunktionen?

Bei gegebenem Q^* muss der Agent sogar keine $one - step - ahead$-Suche machen:

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

Lösen der optimalen Bellman - Gleichung

- Um eine optimale $Policy$ über die Lösung der optimalen Bellman-Gleichung zu finden benötigt man folgendes:
  - exakte Kenntnis der Dynamik der Umgebung;
  - genug Speicherplatz und Berechnungszeit;
  - die Markov-Wahrscheinlichkeit.

- Wieviel Speicherplatz und Zeit benötigen wir?
  - polynomiell mit der Anzahl der Zustände (über $dynamic\,programming$ Methoden; spätere Vorlesung),
  - ABER, normalerweise ist die Anzahl der Zustände groß (z.B., hat Backgammon ungefähr 10^{20} Zustände).

- Wir müssen normalerweise auf Approximationen zurückgreifen.

- Viele RL-Methoden können als approximative Lösung der optimalen Bellman-Gleichung verstanden werden.

Zusammenfassung

- Agent-environment Interaktion
  - Zustände
  - Aktionen
  - $Rewards$
- ***Policy***: stochastiche Aktionsauswahl-Regel
- ***Return***: die Funktion der zukünftigen $Rewards$, die der Agent zu maximieren versucht
- Episodische und kontinuierliche Aufgaben
- Markov Wahrscheinlichkeit
- Markov Entscheidungsprozess
  - Übergangswahrscheinlichkeiten
  - Erwartete $Rewards$
- **Wertefunktionen**
  - Zustands-Wertefunktion für eine $Policy$
  - Aktions-Wertefunktion für eine $Policy$
  - Optimale Zustands-Wertefunktion
  - Optimale Aktions-Wertefunktion

- Optimale Wertefunktionen
- Optimale $Policies$
- Bellman-Gleichungen
- Die Erfordernis von Approximation