FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Vorlesung
Grundlagen adaptiver Wissenssysteme

Prof. Dr. Thomas Gabel
Frankfurt University of Applied Sciences
Faculty of Computer Science and Engineering
tgabel@fb2.fra-uas.de

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Vorlesungseinheit 5

Das Wertiterationsverfahren

Das Wertiterationsverfahren

Lernziele

- Kennenlernen des Wertiterationsverfahren: Value Iteration
  - Algorithmus von Bellman, 1957
- Überlegungen zur Konvergenz dieses Algorithmus

Das Wertiterationsverfahren

Überblick

1. Der unendliche Horizont

Das Wertiterationsverfahren

Überblick

1. Der unendliche Horizont
2. Deterministisches Wertiterationsverfahren

Das Wertiterationsverfahren

Überblick

1. Der unendliche Horizont
2. Deterministisches Wertiterationsverfahren
3. Das Wertiterationsverfahren

Das Wertiterationsverfahren

Überblick

1. Der unendliche Horizont
2. Deterministisches Wertiterationsverfahren
3. Das Wertiterationsverfahren
4. Konvergenz der Wertiteration

Das Wertiterationsverfahren

Überblick

1. Der unendliche Horizont
2. Deterministisches Wertiterationsverfahren
3. Das Wertiterationsverfahren
4. Konvergenz der Wertiteration
5. Einordnung und Ausblick

Das Wertiterationsverfahren

Überblick

Motivation

Von endlichen zum unendlichen Horizont

Zwei Arten praktischer Problemstellungen:

- Entscheidungsfindung in MDPs mit sehr großer Anzahl relevanter Entscheidungsstufen
  - z.B.: Einlastung von Aufträgen in laufender Fabrik (Reihenfolgeplanung, *Scheduling*, Roboter (über-)lebt in unbekannter (freundlich und feindlich) Umgebung (lebenslanges Lernen, *life-long learning*), Regelung eines Reaktors, ...

Motivation

Von endlichen zum unendlichen Horizont

Zwei Arten praktischer Problemstellungen:

- Entscheidungsfindung in MDPs mit sehr großer Anzahl relevanter Entscheidungsstufen
  - z.B.: Einlastung von Aufträgen in laufender Fabrik (Reihenfolgeplanung, *Scheduling*, Roboter (über-)lebt in unbekannter (freundlich und feindlich) Umgebung (lebenslanges Lernen, *life-long learning*), Regelung eines Reaktors, ...
- Probleme, die terminieren, bei denen aber der Zeitpunkt unbekannt ist ("kürzester-Pfad"-Probleme)
  - z.B.: skifahrender Roboter will Ende des Hanges erreichen (Endzeitpunkt hängt z.B. von Optimierungskriterium ab), oder durch Gewinn/Niederlage beendetes Schachspiel

$\Rightarrow$ Diese Modellierung ist typischerweise die Grundlage für Lernprobleme, wie wir sie betrachten werden!

Unendlicher Horizont

Problemstellung:

Der Wert eines Zustandes i unter Strategie π sei gegeben durch:

$$V^{\pi}(i) = \lim_{N \to \infty} \mathbb{E}\left[\sum_{t=0}^{N} c(s_t, \pi(s_t)) | s_0 = i\right]$$

Unendlicher Horizont

Problemstellung:

Der Wert eines Zustandes i unter Strategie π sei gegeben durch:

$$V^{\pi}(i) = \lim_{N \to \infty} \mathbb{E}\left[\sum_{t=0}^{N} c(s_t, \pi(s_t)) | s_0 = i\right]$$

Aufgabe des lernenden Agenten: Suche π^* mit

$$V^{\pi^*} = V^* = \min_{\pi \in \Pi} V^{\pi}$$

Problem: Was ist mit der Endlichkeit der Kosten?

Unendlicher Horizont

Problemstellung:

Der Wert eines Zustandes i unter Strategie π sei gegeben durch:

$$V^{\pi}(i) = \lim_{N \to \infty} \mathbb{E}\left[\sum_{t=0}^{N} c(s_t, \pi(s_t)) | s_0 = i\right]$$

Aufgabe des lernenden Agenten: Suche π^* mit

$$V^{\pi^*} = V^* = \min_{\pi \in \Pi} V^{\pi}$$

Problem: Was ist mit der Endlichkeit der Kosten? $\Rightarrow$ Wir betrachten 2 Typen von Problemen mit endlichen Kosten

- Diskontierung
- Stochastische Kürzester-Pfad-Probleme

Betrachtete Problemtypen

Problemtyp 1: Diskontierung $\gamma < 1$

- Einführung eines Diskontierungsfaktors γ, der kleiner als 1 ist.
- Damit ergibt sich

$$V^{\pi}(i) = \lim_{N \to \infty} \mathbb{E}\left[\sum_{t=0}^{N} \gamma^{t} c(s_t, \pi(s_t)) | s_0 = i\right]$$

- Der Diskontierungsfaktor repräsentiert den Unterschied in der Bedeutung zwischen unmittelbaren und später anfallenden Kosten.

Betrachtete Problemtypen

Problemtyp 1: Diskontierung $\gamma < 1$

Betrachtung der Auswirkungen:

- Da S, A endlich, gibt es ein $\mathcal{M} \in \mathbb{R}$ mit $\mathcal{M} \geq |c(i, a)|$.
- Damit sind die maximalen Pfadkosten begrenzt:

Betrachtete Problemtypen

Problemtyp 1: Diskontierung $\gamma < 1$

Betrachtung der Auswirkungen:

- Da S, A endlich, gibt es ein $\mathcal{M} \in \mathbb{R}$ mit $\mathcal{M} \geq |c(i,a)|$.
- Damit sind die maximalen Pfadkosten begrenzt:

$$V(i) \leq \sum_{k=0}^{\infty} \gamma^k \mathcal{M} = \mathcal{M} \frac{1}{1-\gamma}$$

- Der Diskontierungsfaktor γ wird mit in die Definition des MDP M aufgenommen: $M = (T, S, A, p, c, \gamma)$

Frage: Welche Sonderfälle ergeben sich für $\gamma = 0$ und $\gamma \to 1$?

Betrachtete Problemtypen

Problemtyp 2: Stochastische Kürzester-Pfad-Probleme (SKP)
Besonderheit: Annahme, dass es einen Terminalzustand “0” gibt, der die folgenden Eingenschaften hat:

- “0” ist absorbierend (wird nicht mehr verlassen).
- Es fallen (in ihm) keine Kosten mehr an.
- Damit hat eine Strategie, die “0” erreicht, endliche Kosten.

Ein erster Blick aufs Problem (1)

Zielstellung:

Berechnung der optimalen Wertfunktion

$$V^*(i) = \min_{\pi} \lim_{N \to \infty} \mathbb{E}\left[\sum_{t=0}^{N} \gamma^t c(s_t, \pi_t(s_t)) | s_0 = i\right], \quad \gamma \leq 1$$

Bereits bekannt:

Ein erster Blick aufs Problem (1)

Zielstellung:

Berechnung der optimalen Wertfunktion

$$V^*(i) = \min_{\pi} \lim_{N\to\infty} \mathbb{E}\left[\sum_{t=0}^{N} \gamma^t c(s_t, \pi_t(s_t)) | s_0 = i\right], \quad \gamma \leq 1$$

Bereits bekannt: Für das N-stufige Entscheidungsproblem lassen sich die Kosten durch N Iterationen berechnen (mittels Backward-DP).

$$V_N^*(i) = \min_{a\in A(i)} \sum_{j=1}^{n} p_{ij}(a)\left(c(i,a) + V_{N-1}^*(j)\right) \qquad i = 1 \ldots n$$

mit $V_0^*(i) = g(i)\,\forall i$.

Aber: Bei unendlichem Horizont gibt es keine Finalkosten $g(i)$!

Ein erster Blick aufs Problem (2)

Vermutung 1: Die optimalen Kosten für den unendlichen Fall sind die nach obiger Formel berechneten Kosten im Grenzfall $N \to \infty$

$$V^*(i) = \lim_{N \to \infty} V_N^*(i)$$

Ein erster Blick aufs Problem (2)

Vermutung 1: Die optimalen Kosten für den unendlichen Fall sind die nach obiger Formel berechneten Kosten im Grenzfall $N \to \infty$

$$V^*(i) = \lim_{N \to \infty} V_N^*(i)$$

Vermutung 2: Im Grenzfall muss gelten (aus obigen beiden Formeln):

$$V^*(i) = \min_{a \in A(i)} \sum_{j=1}^{n} p_{ij}(a)(c(i,a) + \gamma V^*(j)) \qquad i = 1 \ldots n$$

Bemerkung: Dies ist kein Algorithmus, sondern ein Gleichungssystem, das für die Lösung V^* erfüllt sein muss. Diese Gleichung drückt das Bellman'sche Optimalitätsprinzip (Bellman-Gleichung) aus.

Ein erster Blick aufs Problem (3)

Vermutung 3: Die optimale Strategie ist stationär und es gilt:

$$\pi(i) \in \arg\min_{a \in A(i)} \sum_{j=1}^{n} p_{ij}(a)(c(i,a) + \gamma V^*(j))$$

Ein erster Blick aufs Problem (3)

Vermutung 3: Die optimale Strategie ist stationär und es gilt:

$$\pi(i) \in \arg\min_{a \in A(i)} \sum_{j=1}^{n} p_{ij}(a)(c(i,a) + \gamma V^*(j))$$

- Anders ausgedrückt: Wenn die optimalen Pfadkosten bekannt sind, ist durch obige Gleichung eine optimale Strategie definiert!
- Eine solche Strategie zu finden, ist die zentrale Aufgabe in autonom lernenden Systemen.

Bemerkung: Alle drei Vermutungen sind typischerweise für eine grosse Klasse von MDPs gültig, insbesondere gelten sie bei den hier betrachteten Problemen.

Weitere Vorgehensweise

- Betrachtung des Sonderfalls einer deterministischen Umgebung
- Definition des vollen Algorithmus' zur iterativen Berechnung von V^*
  - Value Iteration
- Betrachtung von dessen Konvergenzeigenschaften

$\Rightarrow$ Konvergenz ist eine wesentliche Grundlage für Lernverfahren!

Das Wertiterationsverfahren

Überblick

Sonderfall: Deterministische Umgebung (1)

Das deterministische Wertiterationsverfahren

Kernidee:

- Zustandsübergänge in der Umgebung seien deterministisch: Wenn in Zustand i die Aktion a ausgeführt wird, erfolgt Übergang nach j mit Wahrscheinlichkeit 1.0.
  - $f(i, a) = j$

Sonderfall: Deterministische Umgebung (1)

Das deterministische Wertiterationsverfahren

Kernidee:

- Zustandsübergänge in der Umgebung seien deterministisch: Wenn in Zustand *i* die Aktion *a* ausgeführt wird, erfolgt Übergang nach *j* mit Wahrscheinlichkeit 1.0.
  - $f(i, a) = j$
- Grundlegende Annahme bei Value Iteration: Wenn eine Lösung $V^*(j)$ für ein "Teilproblem" bereits bekannt ist, so ist es leicht, eine Lösung für $V^*(i)$ zu ermitteln:

$$V^*(i) = \min_{a \in A} (c(i, a) + \gamma V^*(j))$$

Sonderfall: Deterministische Umgebung (1)

Das deterministische Wertiterationsverfahren

Kernidee:

- Zustandsübergänge in der Umgebung seien deterministisch: Wenn in Zustand *i* die Aktion *a* ausgeführt wird, erfolgt Übergang nach *j* mit Wahrscheinlichkeit 1.0.
  - $f(i, a) = j$
- Grundlegende Annahme bei Value Iteration: Wenn eine Lösung $V^*(j)$ für ein "Teilproblem" bereits bekannt ist, so ist es leicht, eine Lösung für $V^*(i)$ zu ermitteln:

$$V^*(i) = \min_{a \in A} (c(i, a) + \gamma V^*(j))$$

- Grundgedanke des Wertiterationsverfahrens ist es, derartige Aktualisierungen wiederholt durchzuführen.
  - So lange, bis sich keine Änderungen mehr ergeben.
- Beispiel: Beginne mit dem Ende (dem Zielzustand) und arbeite dich rückwärts vor.

Sonderfall: Deterministische Umgebung (2)

Beispiel: Finden des kürzesten Pfades zum Ziel

g			

Problem

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

V_1

0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1

V_2

0	-1	-2	-2
-1	-2	-2	-2
-2	-2	-2	-2
-2	-2	-2	-2

V_3

0	-1	-2	-3
-1	-2	-3	-3
-2	-3	-3	-3
-3	-3	-3	-3

V_4

0	-1	-2	-3
-1	-2	-3	-4
-2	-3	-4	-4
-3	-4	-4	-4

V_5

0	-1	-2	-3
-1	-2	-3	-4
-2	-3	-4	-5
-3	-4	-5	-5

V_6

0	-1	-2	-3
-1	-2	-3	-4
-2	-3	-4	-5
-3	-4	-5	-6

V_7

Sonderfall: Deterministische Umgebung (4)

Deterministic Value Iteration

- Wähle V_0 beliebig.
- Setze Zähler $k = 0$.

Sonderfall: Deterministische Umgebung (4)

Deterministic Value Iteration

- Wähle V_0 beliebig.
- Setze Zähler $k = 0$.
- REPEAT

  $k := k + 1$

  FORALL $i \in S$ $(i = 1 \ldots n)$

Sonderfall: Deterministische Umgebung (4)

Deterministic Value Iteration

- Wähle V_0 beliebig.
- Setze Zähler $k = 0$.
- REPEAT

  $k := k + 1$

  FORALL $i \in S$ **(**$i = 1 \ldots n$**)**

$$V_k(i) = \min_{a \in A(i)} (c(i,a) + \gamma V_{k-1}(f(i,a))) \qquad //f(i,a) = j$$

  UNTIL (konvergiert, d.h. $V_k = V_{k-1}$)

Sonderfall: Deterministische Umgebung (4)

Deterministic Value Iteration

- Wähle V_0 beliebig.
- Setze Zähler $k = 0$.
- REPEAT

  $k := k + 1$

  FORALL $i \in S$ **(**$i = 1 \ldots n$**)**

$$V_k(i) = \min_{a \in A(i)} (c(i,a) + \gamma V_{k-1}(f(i,a))) \qquad //f(i,a) = j$$

  UNTIL (konvergiert, d.h. $V_k = V_{k-1}$)

Bemerkung: Der Zähler k korrespondiert zur Iteration und hilft, den iterativen Charakter der Verfahrens hervorzuheben. In einer praktischen Implementierung kommt man hingegen leicht mit zwei Wertfunktionen aus (z.B. V_{alt} und V_{neu}).

Sonderfall: Deterministische Umgebung (4)

Herausforderungen für den allgemeinen (nicht-deterministischen) Fall bzw. für eine entsprechende Variante des Algorithmus:

- In MDPs gibt es meist keinen endlichen Horizont.
- MDPs besitzen in der Regel Schleifen (man gelangt wiederholt in den gleichen Zustand).
- Es gibt kein “Ende” von dem ausgehend man sich rückwärtig voranarbeiten könnte.

Sonderfall: Deterministische Umgebung (4)

Herausforderungen für den allgemeinen (nicht-deterministischen) Fall bzw. für eine entsprechende Variante des Algorithmus:

- In MDPs gibt es meist keinen endlichen Horizont.
- MDPs besitzen in der Regel Schleifen (man gelangt wiederholt in den gleichen Zustand).
- Es gibt kein “Ende” von dem ausgehend man sich rückwärtig voranarbeiten könnte.
- Aber: (Erlernte) Informationen können dennoch “rückwärtig” durch den MDP propagiert werden.
  - Zum Beispiel mit der Bellman-Gleichung und für einzelne Zustandsübergänge von *i* nach *j* (auch wenn diese stochastisch erfolgen).
  - Teilprobleme (die sich vom Folgezustand *j* aus erstrecken) können als “einfacher” zu lösen angesehen werden.
    - Das ist garantiert bei Verwendung eines Diskontierungsfaktors.
  - Iteration kann so lange durchgeführt werden, bis sich Konvergenz einstellt.

Das Wertiterationsverfahren

Überblick

Value Iteration

Das Wertiterationsverfahren

Kernidee:

- berechne die optimale Wertfunktion iterativ
- verwende als Ausgangspunkt die Lösung eines N-stufigen Entscheidungsproblems
- berechne darauf aufbauend die Lösung eines $N + 1$-stufigen Entscheidungsproblems
  - repräsentiert durch eine Wertfunktion V_{N+1} für das $N + 1$-stufige Entscheidungsproblem

Es bleibt dann zu zeigen:

$$\lim_{k \to \infty} V_k = V^*$$

Value Iteration

Das Wertiterationsverfahren

Value Iteration

- Wähle V_0 beliebig.
- Setze Zähler $k = 0$.

Value Iteration

Das Wertiterationsverfahren

Value Iteration

- Wähle V_0 beliebig.
- Setze Zähler $k = 0$.
- Repeat

  $k := k + 1$

  ForAll $i \in S \quad (i = 1 \ldots n)$

Value Iteration

Das Wertiterationsverfahren

Value Iteration

- Wähle V_0 beliebig.
- Setze Zähler $k = 0$.
- REPEAT

  $k := k + 1$

  FORALL $i \in S \quad (i = 1 \ldots n)$

$$V_k(i) = \min_{a \in A(i)} \sum_{j=0}^{n} p_{ij}(a)(c(i,a) + \gamma V_{k-1}(j))$$

UNTIL (konvergiert)

Optimalitätsprinzip in MDPs

Value Iteration nutzt das folgende Theorem aus:

Theorem: Optimalitätsprinzip

Eine Strategie π erzielt die minimalen Kosten ausgehend vom Zustand i, d.h. es gilt $V^{\pi}(i) = V^{*}(i)$, genau dann wenn diese Strategie auch die minimalen Kosten für jeden Zustand j erzielt (d.h. $V^{\pi}(j) = V^{*}(j)$), der von i aus direkt erreichbar ist.

Optimalitätsprinzip in MDPs

Value Iteration nutzt das folgende Theorem aus:

Theorem: Optimalitätsprinzip

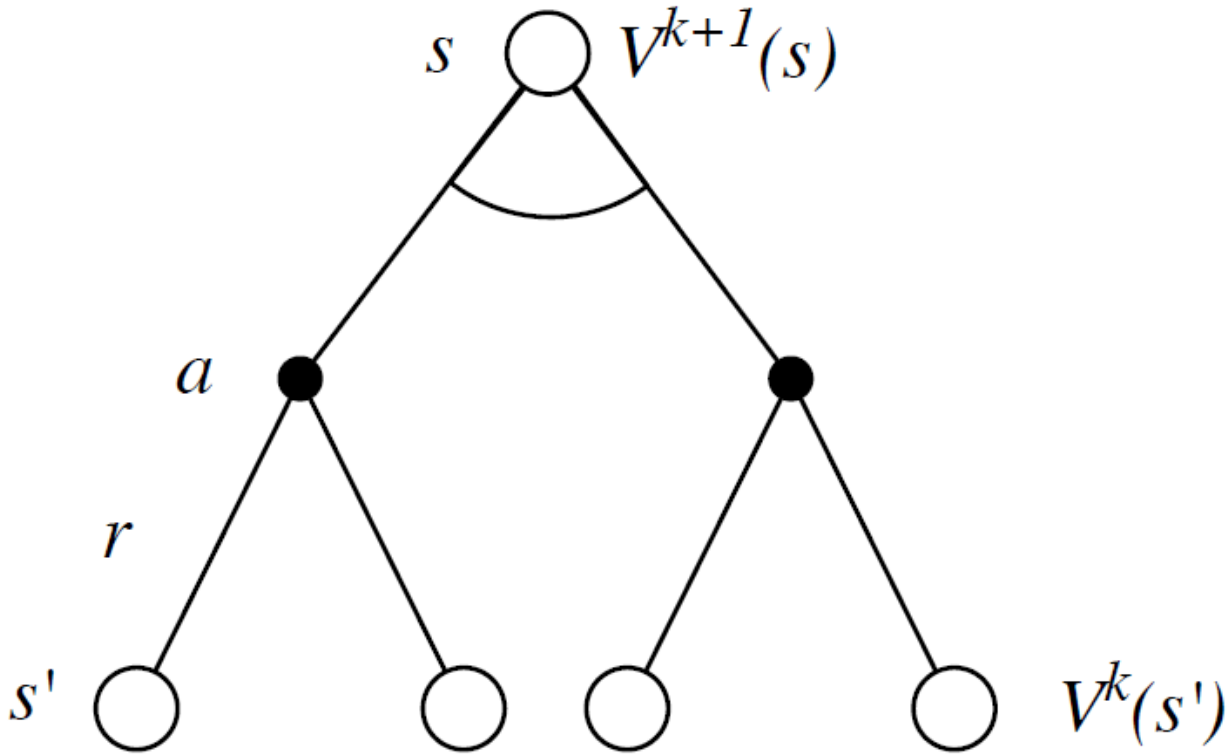
Eine Strategie π erzielt die minimalen Kosten ausgehend vom Zustand i, d.h. es gilt $V^{\pi}(i) = V^{*}(i)$, genau dann wenn diese Strategie auch die minimalen Kosten für jeden Zustand j erzielt (d.h. $V^{\pi}(j) = V^{*}(j)$), der von i aus direkt erreichbar ist.

Demgemäß umfasst eine optimale Strategie π^{*}

- die optimale erste Aktion a^{*}
- sowie im Anschluss daran die optimale Strategie für jeden Folgezustand.

Veranschaulichung von Value Iteration

Darstellung einer einzelnen Aktualisierung für einen Zustand $i = s$ mit Folgezuständen $j = s'$



Zwischenzusammenfassung

Eigenschaften von Value Iteration:

- Problem: Finde die optimale Strategie π^*
- Lösung: Iterative Anwendung von Bellman-Aktualisierungen für alle Zustände
- Eine Folge von Wertfunktionen entsteht: $V_0 \rightarrow V_1 \rightarrow V_2 \rightarrow \cdots \rightarrow V^*$

Zwischenzusammenfassung

Eigenschaften von Value Iteration:

- Problem: Finde die optimale Strategie π^*
- Lösung: Iterative Anwendung von Bellman-Aktualisierungen für alle Zustände
- Eine Folge von Wertfunktionen entsteht: $V_0 \rightarrow V_1 \rightarrow V_2 \rightarrow \cdots \rightarrow V^*$
- zwei ineinander geschachtelte Schleifen
  - äußere Schleife über die Iterationen
  - innere Schleife über alle Zustände $i \in S$
  - asynchrone Aktualisierungen werden vorgenommen: $V_{k+1}(i)$ auf Basis von $V_k(j)$
- Konvergenz zu V^*: wird gleich betrachtet

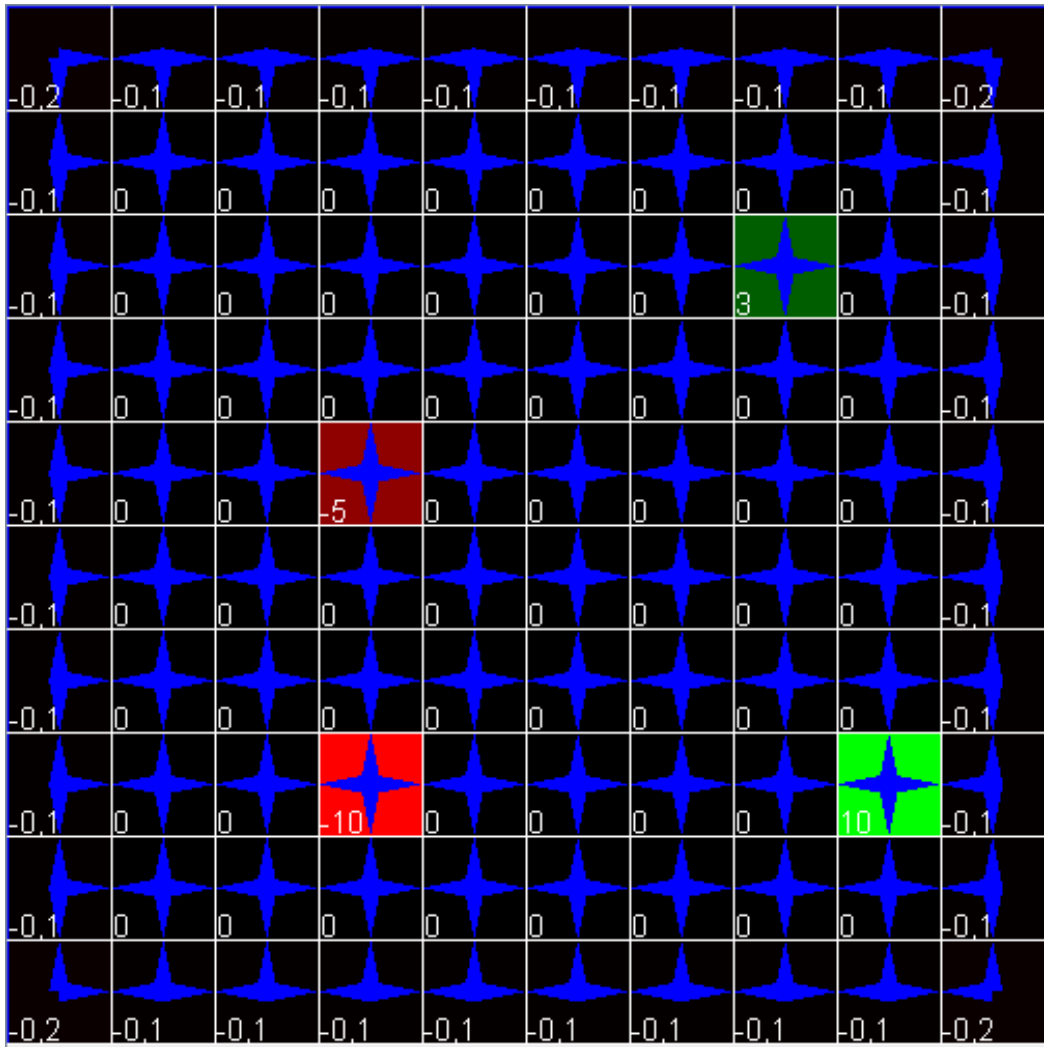
Beispiel für Value Iteration (1)

Gitterwelt

- Agent in einem zweidimensionalen Gitter, Größe 10x10
- jede Zelle ist ein Zustand $\rightarrow$ 100 Zustände
- 4 Aktionen: hoch, runter, links, rechts
- Terminologie: Belohnungen (negative Zahlenwerte sind Kosten)
- Definition direkter Belohnungen
  - Bewegung in eine Wand: -1
  - Zustand (8,9), in dem es positiven Reward gibt: +10
  - Zustand (3,8), in dem es positiven Reward gibt: +3
  - Zustand (5,4), in dem es negativen Reward gibt: -5
  - Zustand (8,4), in dem es negativen Reward gibt: -10
  - alle anderen Zustandsübergänge: kosten-/belohnungsfrei

Beispiel für Value Iteration (2)

Darstellung im Bild:



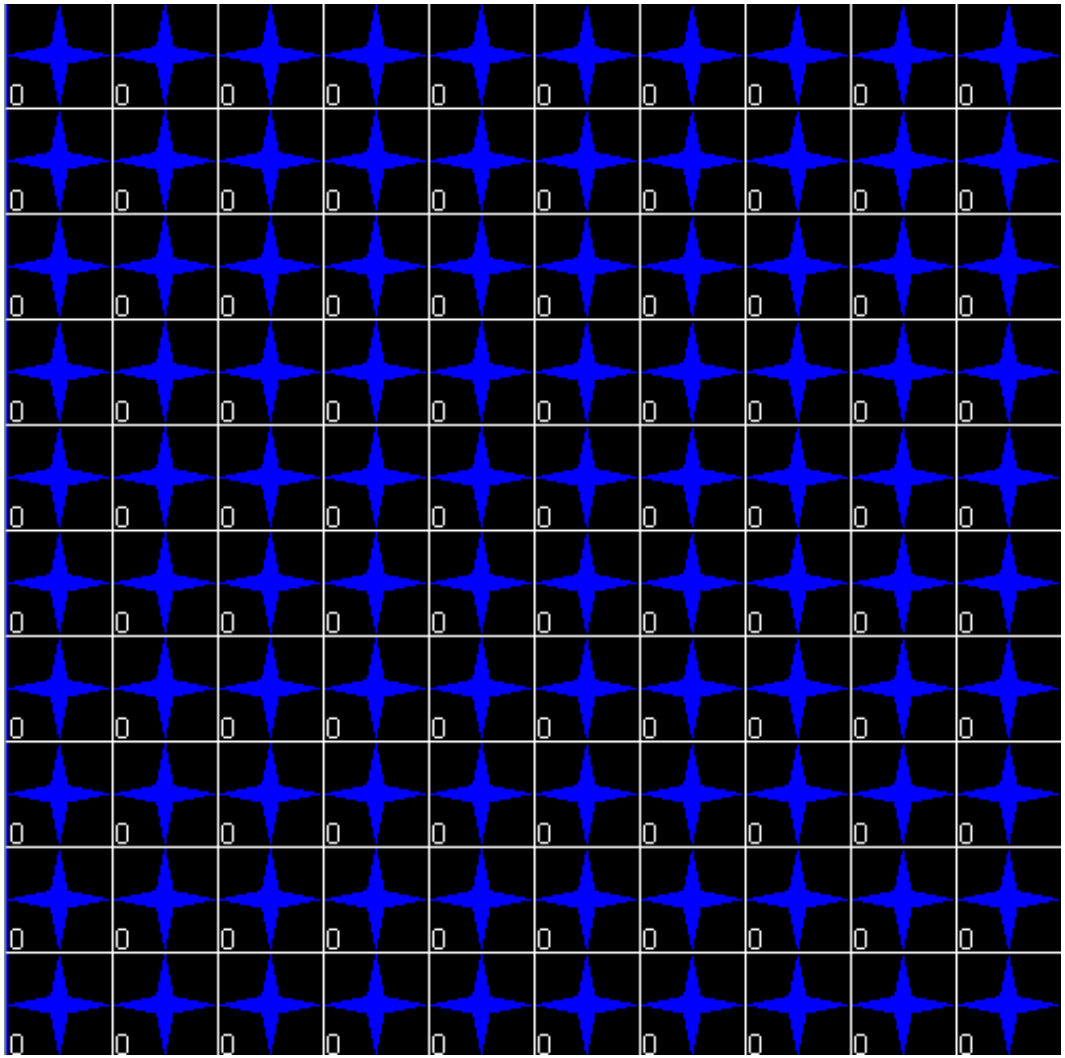
Beispiel für Value Iteration (3)

Gitterwelt (Forts.)

- Zustandsübergänge sind stochastisch
  - Jede Aktion wird mit Wahrscheinlichkeit 0.7 ausgeführt.
  - Aber mit Wahrscheinlichkeit von je 0.1 wird eine der drei anderen Aktionen ausgeführt.
  - Die positiven Reward-Zustände sind absorbierend.
  - Keine Diskontierung ($\gamma = 1$)!
- Visualisierung der optimalen Aktion in blau
- Darstellung des Wertfunktion mit kleinen Zahlen in der jeweiligen Zelle

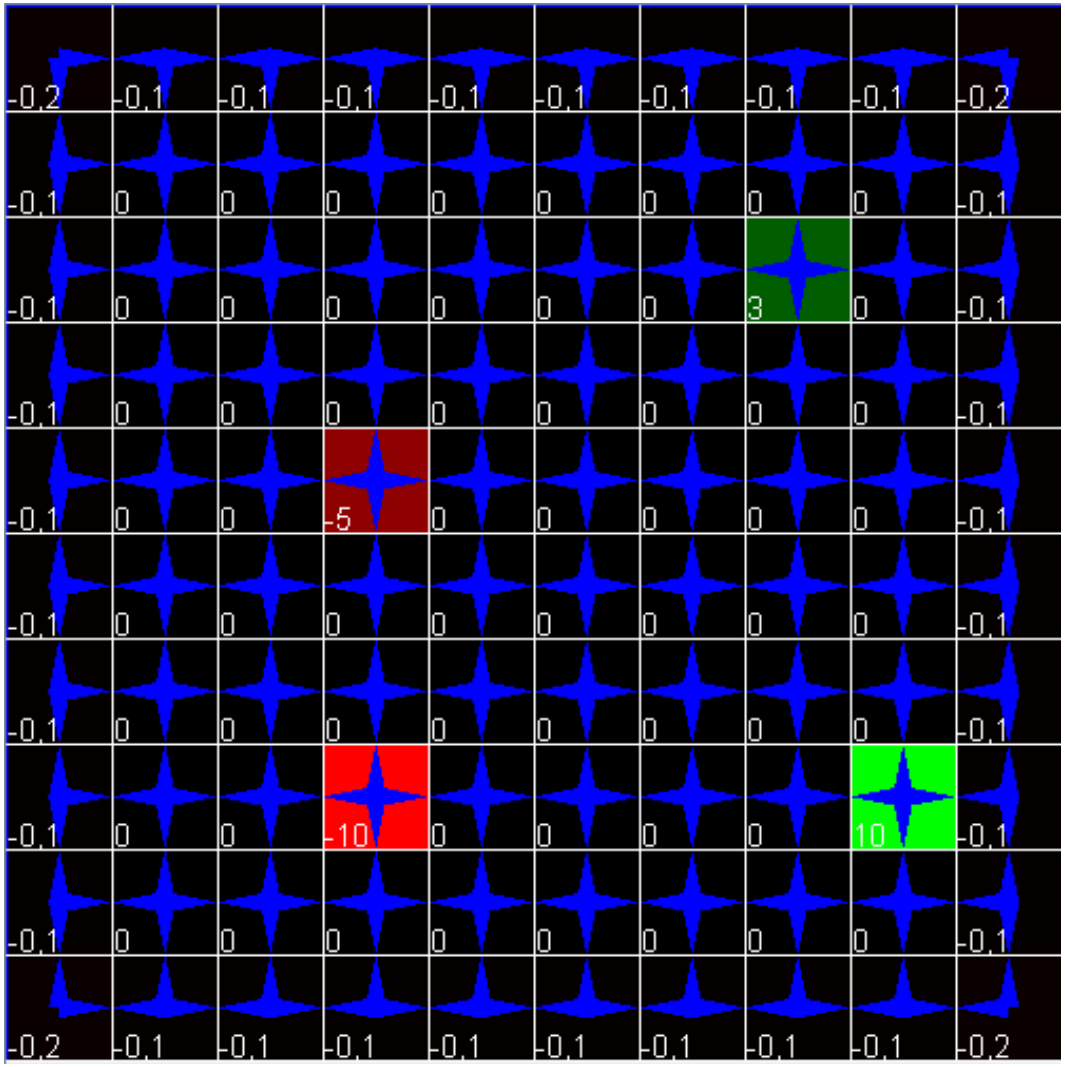
Beispiel für Value Iteration (4)

Initialisierung der Wertfunktion



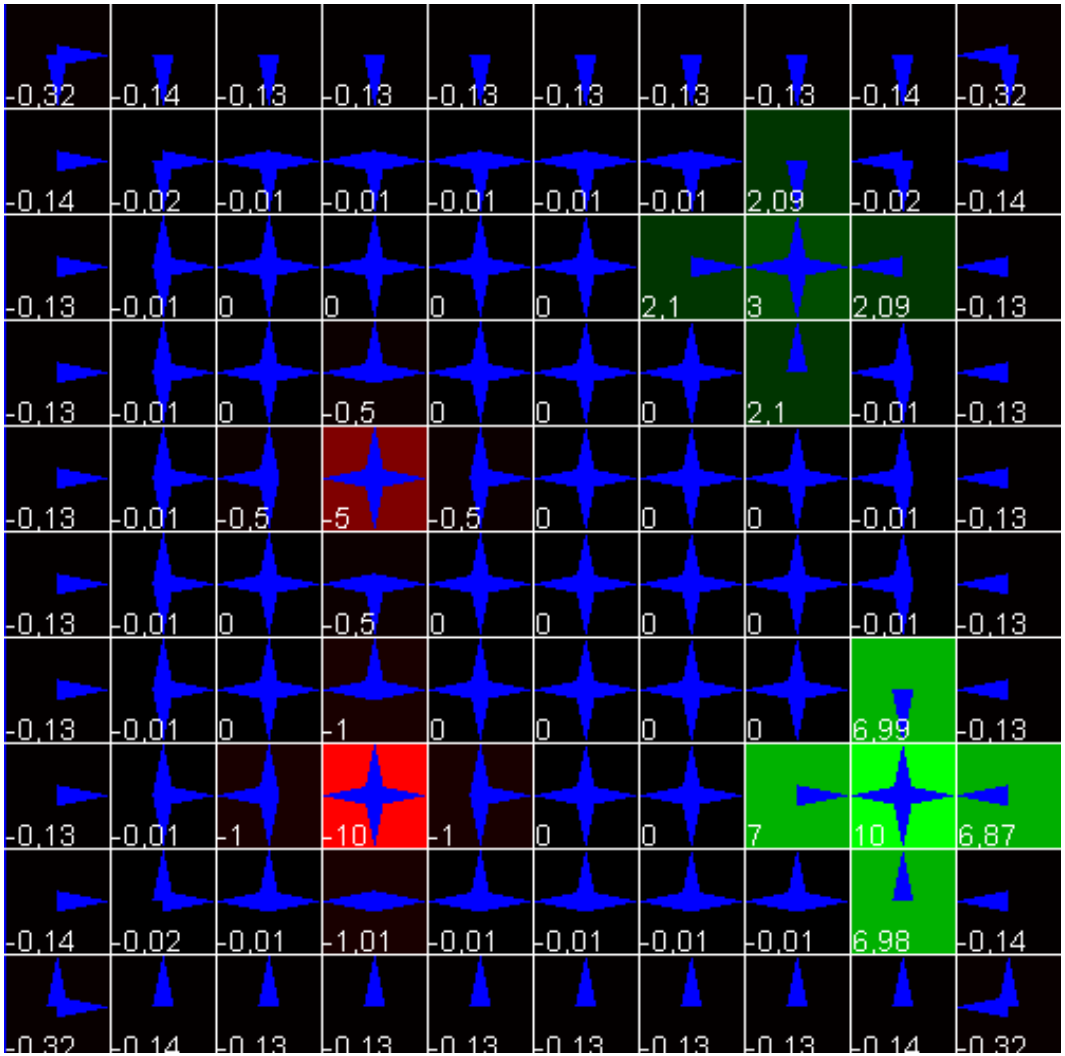
Beispiel für Value Iteration (5)

Erste Iteration



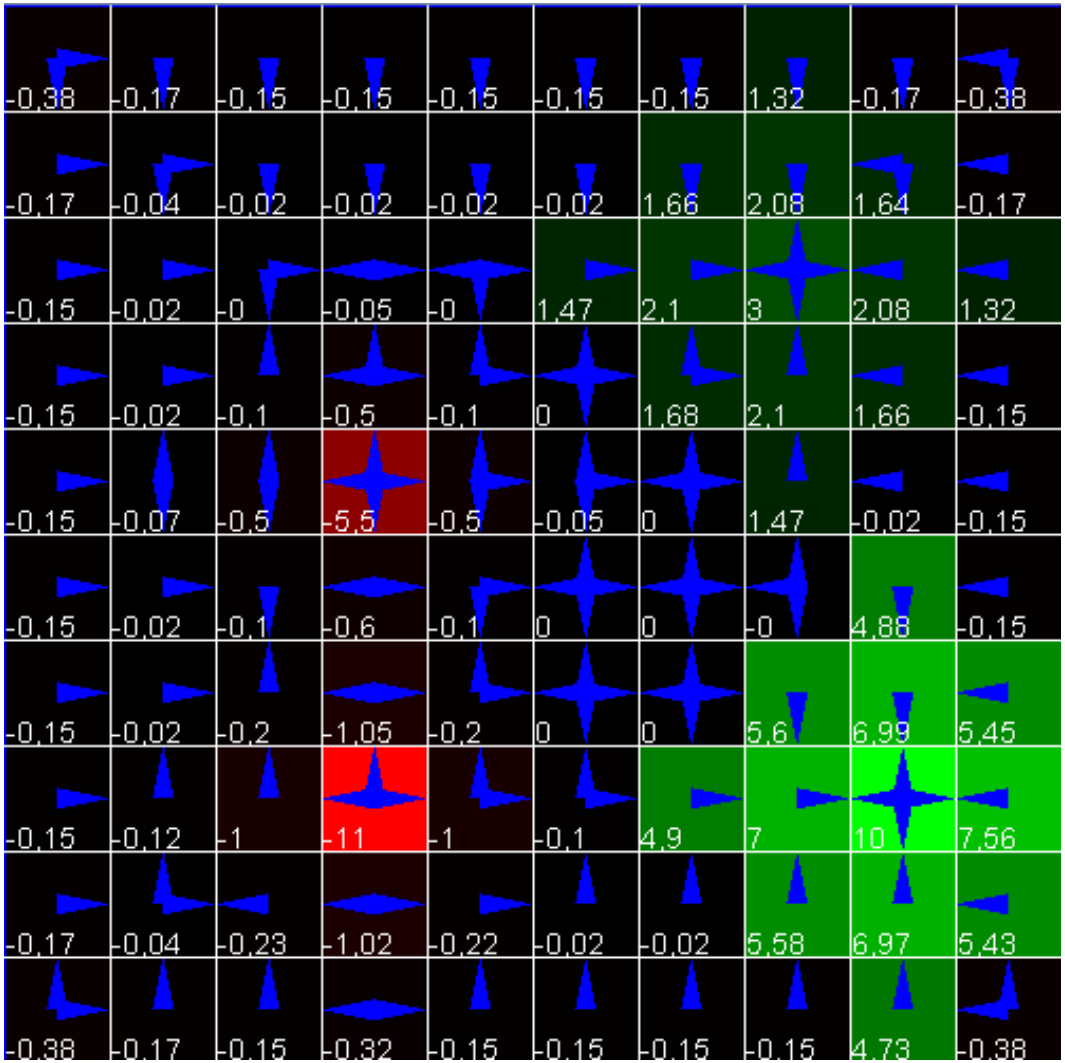
Beispiel für Value Iteration (6)

Zweite Iteration



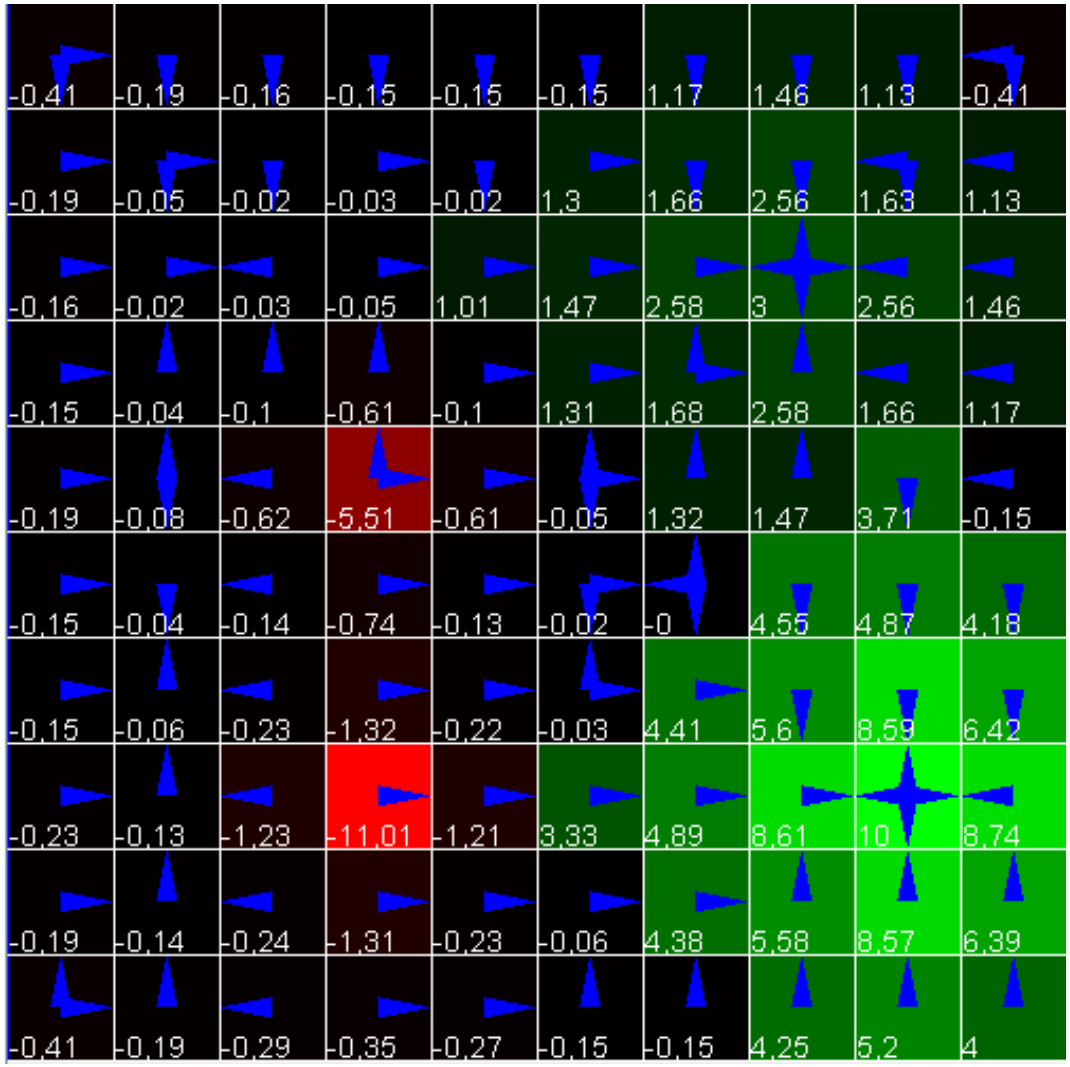
Beispiel für Value Iteration (7)

Dritte Iteration



Beispiel für Value Iteration (8)

Vierte Iteration



Das Wertiterationsverfahren

Überblick

Konvergenzaussagen

Fallunterscheidung nach den beiden Problemtypen

- stochastische Kürzester-Pfad-Probleme (SKP)
- diskontierte Probleme
  (Diskontierung mit Diskontierungsfaktor $\gamma < 1$)

Konvergenz bei SKP-Problemen

Erinnerung:
Ein stochastisches Kürzester-Pfad-Problem liegt vor, wenn es einen Terminalzustand “0” gibt mit

- “0” ist absorbierend (wird nicht mehr verlassen):
  $p_{0j}(a) = 0, \forall a, \forall j \neq 0$
- es fallen keine Kosten mehr an: $c(0, a) = 0, \forall a$
- **Konvention:** Im folgenden soll immer gelten: $V(0) = 0$
  (die Wertfunktion an der Stelle “0” ist 0)

Sonderfälle

Interessanterweise können folgende Problemtypen als Spezialfälle von SKP-Problemen angesehen werden.

- deterministisches Kürzester-Pfad-Problem: finde die minimalen Wegkosten (auch energieoptimal, zeitoptimal, ...)

Sonderfälle

Interessanterweise können folgende Problemtypen als Spezialfälle von SKP-Problemen angesehen werden.

- deterministisches Kürzester-Pfad-Problem: finde die minimalen Wegkosten (auch energieoptimal, zeitoptimal, ...)
- Probleme mit endlichem Horizont können ebenfalls als SKP-Spezialfall formuliert werden:
  - Erweiterung der Zustandsbeschreibung um aktuellen Zeitschritt
  - Zustand $(i, \mathbf{k})$, Übergang $\mapsto (j, \mathbf{k} + \mathbf{1})$
  - Von jedem Zustand im letzten Zeitschritt N, $(i, \mathbf{N})$, erfolgt eine Übergang in den Terminalzustand "0" mit Übergangskosten gemäß Finalkosten $g_N(i)$

Sonderfälle

Interessanterweise können folgende Problemtypen als Spezialfälle von SKP-Problemen angesehen werden.

- deterministisches Kürzester-Pfad-Problem: finde die minimalen Wegkosten (auch energieoptimal, zeitoptimal, ...)
- Probleme mit endlichem Horizont können ebenfalls als SKP-Spezialfall formuliert werden:
  - Erweiterung der Zustandsbeschreibung um aktuellen Zeitschritt
  - Zustand $(i, \mathbf{k})$, Übergang $\mapsto (j, \mathbf{k} + \mathbf{1})$
  - Von jedem Zustand im letzten Zeitschritt N, $(i, \mathbf{N})$, erfolgt eine Übergang in den Terminalzustand "0" mit Übergangskosten gemäß Finalkosten $g_N(i)$

$\Rightarrow$ Vorteil: Wenn Konvergenzaussagen für das Wertiterationsverfahren getroffen werden können, so gelten diese auch für jene Spezialfälle.

Der Begriff der erfüllenden Strategie (1)

Proper Policy

Definition (Erfüllende Strategie)

Eine stationäre Strategie heisst erfüllend (proper), wenn durch ihre Anwendung für jeden beliebigen Startzustand der Terminalzustand "0" in höchstens n Schritten mit positiver Wahrscheinlichkeit erreicht wird, also

$$\rho_\pi :=$$

Der Begriff der erfüllenden Strategie (1)

Proper Policy

Definition (Erfüllende Strategie)

Eine stationäre Strategie heisst erfüllend (proper), wenn durch ihre Anwendung für jeden beliebigen Startzustand der Terminalzustand "0" in höchstens n Schritten mit positiver Wahrscheinlichkeit erreicht wird, also

$$\rho_\pi := \max_{i=1\ldots n} P(s_n \neq 0 | s_0 = i, \pi) < 1$$

Der Begriff der erfüllenden Strategie (2)

Proper Policy

- *Erinnerung: n* bezeichnet die Anzahl der Zustände.
- *Bemerkung 1:* Eine erfüllende Strategie führt mit Wahrscheinlichkeit 1 (irgendwann) in den Terminalzustand. Damit sind die dazugehörigen (Wege-)Kosten für jeden Zustand endlich.
  - Ohne Beweis. □

Der Begriff der erfüllenden Strategie (2)

Proper Policy

- *Erinnerung: n* bezeichnet die Anzahl der Zustände.
- *Bemerkung 1:* Eine erfüllende Strategie führt mit Wahrscheinlichkeit 1 (irgendwann) in den Terminalzustand. Damit sind die dazugehörigen (Wege-)Kosten für jeden Zustand endlich.
  - Ohne Beweis. □
- *Bemerkung 2:* Strategie π ist erfüllend gdw. die zu π gehörende Markov-Kette jeden Zustand *i* mit dem Terminalzustand mit positiven Übergangswahrscheinlichkeiten verbindet
  - Ohne Beweis. □

Zu beantwortende Frage im Folgenden: Welche Bewandnis hat der Begriff der erfüllenden Strategie hinsichtlich der Konvergenzeigenschaften von Value Iteration?

Konvergenzvoraussetzungen für Value Iteration bei SKP (1)

Konvergenz von Value Iteration bei SKP

Das Wertiterationsverfahren konvergiert bei stochastischen Kürzester-Pfad-Problemen, wenn die folgenden beiden Voraussetzungen erfüllt sind:

- Voraussetzung 1: Es gibt mindestens eine erfüllende (proper) Strategie. [SKP-V1]

Konvergenzvoraussetzungen für Value Iteration bei SKP (1)

Konvergenz von Value Iteration bei SKP

Das Wertiterationsverfahren konvergiert bei stochastischen Kürzester-Pfad-Problemen, wenn die folgenden beiden Voraussetzungen erfüllt sind:

- Voraussetzung 1: Es gibt mindestens eine erfüllende (proper) Strategie. [SKP-V1]
- Voraussetzung 2: Für jede nicht erfüllende (improper) Strategie ist für mindestens einen Zustand der Wert der Wertfunktion $V^{\pi}(i)$ unendlich. [SKP-V2]
  - Man spricht auch von “unendlichen Pfadkosten”.

Konvergenzvoraussetzungen für Value Iteration bei SKP (2)

Bemerkungen:

- Bei deterministischen KP-Problemen bedeutet Voraussetzung 1, dass es einen Pfad von jedem Anfangszustand zum Terminalzustand gibt.

Konvergenzvoraussetzungen für Value Iteration bei SKP (2)

Bemerkungen:

- Bei deterministischen KP-Problemen bedeutet Voraussetzung 1, dass es einen Pfad von jedem Anfangszustand zum Terminalzustand gibt.
- Voraussetzung 2 kann auch ersetzt werden durch die stärkere Forderung

$$c(i,a) > 0 \quad \forall i \neq 0, a \in A(i)$$

  Diese Bedingung ist bei vielen Lernproblemen erfüllt, z.B. Reduzierung der Zeit, Reduzierung von Energieaufwand, ...
- Voraussetzungen 1 und 2 sind ebenfalls erfüllt, wenn **alle** möglichen Strategien erfüllend sind.
  → Diese Aussage benötigen wir noch!

Konvergenzvoraussetzungen für Value Iteration bei diskontierten Problemen (1)

Soweit betrachtet: SKP-Probleme

- Nun: diskontierte Probleme

Kernidee:

- Wenn es möglich ist, jedes diskontierte Problem in ein stochastisches Kürzester-Pfad-Problem umzuformulieren, so würden die Konvergenzaussagen für SKP-Probleme auch unmittelbar für diskontierte Probleme gelten.
- Solch eine Umformulierung soll im Folgenden vorgestellt werden.

Diskontierte Probleme als SKP-Probleme

Zur Erinnerung:
Diskontierung mit $\gamma < 1$

$$V^{\pi}(i) = \lim_{N \to \infty} \mathbb{E}\left[\sum_{k=0}^{N} \gamma^k c(s_k, \pi(s_k)) | s_0 = i\right]$$

Bemerkungen:

- In Anwendungen wird der Diskontierungsfaktor so gewählt, dass er den Verlauf über einen bestimmten Zeitraum berücksichtigt.
- Der Diskontierungsfaktor reflektiert eine Abwägung zwischen direkten Kosten und zeitlich später anfallenden Kosten.
- Oft wird er durch Experimente festgelegt.

Umformulierung eines diskontierten Problems in ein SKP-Problem

$$V^*(i) = \min_{\pi} \lim_{N\to\infty} \mathbb{E}\left[\sum_{t=0}^{N} \gamma^t c(s_t, \pi(s_t)) | s_0 = i\right], \quad \gamma \leq 1$$

Ausgangspunkt:

- Diskontierungsproblem mit n Zuständen
- Übergangswahrscheinlichkeiten $p_{ij}(a)$, wobei $a = \pi(s_t)$

Umformulierung eines diskontierten Problems in ein SKP-Problem

$$V^*(i) = \min_{\pi} \lim_{N\to\infty} \mathbb{E}\left[\sum_{t=0}^{N} \gamma^t c(s_t, \pi(s_t)) | s_0 = i\right], \quad \gamma \leq 1$$

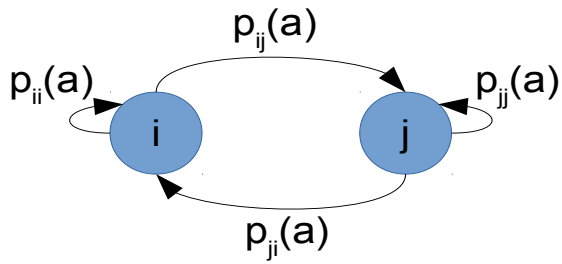
Ausgangspunkt:

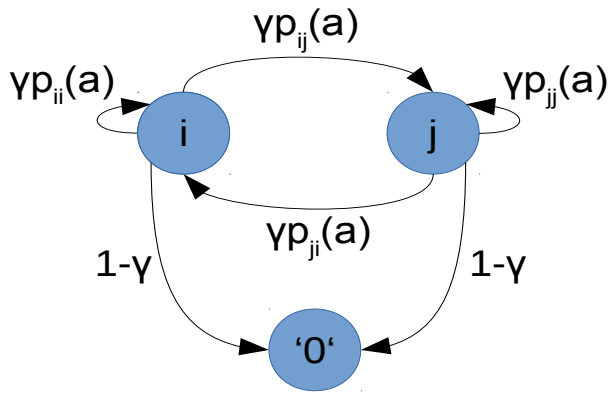
- Diskontierungsproblem mit n Zuständen
- Übergangswahrscheinlichkeiten $p_{ij}(a)$, wobei $a = \pi(s_t)$

Umformulierung:

- Einführung eines Terminalzustand “0”
- Neudefinition der übergangswahrscheinlichkeiten von $i' \to j'$ wie folgt:
  - falls j' Nichtterminalzustand: $p(i', j') := \gamma\, p_{ij}(a)$
  - falls $j' = 0$ (Terminalzustand): $p(i', j') := \mathbf{1} - \gamma$

Diskontiertes Problem als SKP-Problem: Skizze





$$V^*(i) = \min_{\pi} \lim_{N \to \infty} \mathbb{E}\left[\sum_{t=0}^{N} \gamma^t c(s_t, \pi_t(s_t)) | s_0 = i\right], \quad \gamma \leq 1$$

Konvergenz von diskontierten Problemen

Mit dieser Umformulierung lässt sich ein diskontiertes Problem also als SKP darstellen.

- Erfreulich: Damit lässt sich die Konvergenzaussage für SKP-Probleme auch auf diskontierte Probleme anwenden.

Konvergenz von diskontierten Problemen

Mit dieser Umformulierung lässt sich ein diskontiertes Problem also als SKP darstellen.

- Erfreulich: Damit lässt sich die Konvergenzaussage für SKP-Probleme auch auf diskontierte Probleme anwenden.
- Da aus jedem Zustand mit jeder Aktion mit Wahrscheinlichkeit $1 - \gamma$ in den Terminalzustand übergegangen wird, ist die (alternative) Konvergenzvoraussetzung “alle Strategien sind erfüllend” für SKPs erfüllt.

$\Rightarrow$ Ergebnis: Value Iteration konvergiert bei diskontierten Problemen ohne weitere Voraussetzungen als $\gamma < 1$.

Zusammenfassung

Konvergenzaussagen zu Value Iteration

- Das Wertiterationsverfahren konvergiert für SKP-Probleme, wenn die Voraussetzungen SKP-V1 und SKP-V2 erfüllt sind.
- Diskontierte Probleme sind als SKP-Probleme formulierbar.
  $\Rightarrow$ Konvergenz gilt auch für den diskontierten Fall.
- Start mit beliebigem V; Verfahren konvergiert gegen V^*.
- Im Allgemeinen sind allerdings unendlich viele Iterationen nötig.
- Es gibt Ausnahmen, z.B. deterministische kürzester Pfad-Probleme (hier ergibt sich Konvergenz in maximal n (Anzahl der Zustände) Schritten).
- Prinzipielles Vorgehen: Anpassungsschritt wird für alle Zustände i gleichzeitig durchgeführt. Stop, falls sich nichts mehr ändert.

Das Wertiterationsverfahren

Überblick

Einordnung

Value Iteration / Das Wertiterationsverfahren

- ist ein Verfahren des dynamisches Programmierens (dynamic programming).
- ist eine Form der Planung für einen MDP.
- ist (noch) kein Reinforcement Learning!
  - Weil die Kostenfunktion und die Übergangswahrscheinlichkeiten, also das “Modell der Umgebung”, als gegeben angenommen werden.
- hat gute Konvergenzeigenschaften für Probleme mit endlich vielen Zuständen.
- wird sich bei Problemen mit unendlich vielen Zuständen in der vorgestellten Art nicht einfach so anwenden lassen.

Asynchrone Wertiterationsverfahren (1)

Unter dem asynchronen Wertiterationsverfahren versteht man Erweiterungen von Value Iteration im Hinblick auf dessen Anwendung in praktischen Problemstellungen. Insbesondere:

Asynchrone Wertiterationsverfahren (1)

Unter dem asynchronen Wertiterationsverfahren versteht man Erweiterungen von Value Iteration im Hinblick auf dessen Anwendung in praktischen Problemstellungen. Insbesondere:

- Im normalen Wertiterationsverfahren werden alle Aktualisierungen gleichzeitig (parallel) durchgeführt.
- Beim asynchronen Wertiterationsverfahren dürfen Anpassungen in beliebiger Reihenfolge ausgeführt werden.
- Für einen (in welcher Reihenfolge auch immer) ausgewählten Zustand wird die Aktualisierung vorgenommen.
- Dieses Vorgehen kann den Rechenaufwand erheblich reduzieren.

Asynchrone Wertiterationsverfahren (2)

- Man kann zeigen, dass die Zustände auch nacheinander oder in beliebiger Reihenfolge angepasst werden können.
- Value Iteration konvergiert unter den gleichen Voraussetzungen, falls jeder Zustand unendlich oft angepasst wird.

Asynchrone Wertiterationsverfahren (2)

- Man kann zeigen, dass die Zustände auch nacheinander oder in beliebiger Reihenfolge angepasst werden können.
- Value Iteration konvergiert unter den gleichen Voraussetzungen, falls jeder Zustand unendlich oft angepasst wird.
- Asynchrones Value Iteration wird oft in lernenden Systemen verwendet.
- Konkrete Umsetzungen:

Asynchrone Wertiterationsverfahren (2)

- Man kann zeigen, dass die Zustände auch nacheinander oder in beliebiger Reihenfolge angepasst werden können.
- Value Iteration konvergiert unter den gleichen Voraussetzungen, falls jeder Zustand unendlich oft angepasst wird.
- Asynchrones Value Iteration wird oft in lernenden Systemen verwendet.
- Konkrete Umsetzungen:
  - In-Place Dynamic Programming: synchrone Wertiteration auf nur einer einzigen Wertfunktion (anstatt auf zwei Kopien V_{old} und V_{new} beim herkömmlichen VI)
  - Prioritized Sweeping: Auswahl der Zustände, für die als nächstes eine Aktualisierung durchgeführt wird auf Basis des Bellman-Fehlers
  - Real-Time Dynamic Programming (RTDP): Anpassung entlang von Trajektorien, die der Agent in Interaktion mit der Umwelt abläuft ($\rightarrow$ kommt RL am nächsten)