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 7

Zeitliche Differenz-Methoden

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Zeitliche Differenz-Methoden

Lernziele

- Ersetzung des Modells
- Kennenlernen von Verfahren, die ohne Modell zurechtkommen

Zeitliche Differenz-Methoden

Überblick

1. Motivation

Zeitliche Differenz-Methoden

Überblick

1. Motivation
2. Real-Time Dynamic Programming

Zeitliche Differenz-Methoden

Überblick

1. Motivation
2. Real-Time Dynamic Programming
3. Die Monte-Carlo-Methode (MC)

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Zeitliche Differenz-Methoden

Überblick

1. Motivation
2. Real-Time Dynamic Programming
3. Die Monte-Carlo-Methode (MC)
4. Zeitliche-Differenzmethoden (TD) – Ansatz I

Zeitliche Differenz-Methoden

Überblick

1. Motivation
2. Real-Time Dynamic Programming
3. Die Monte-Carlo-Methode (MC)
4. Zeitliche-Differenzmethoden (TD) – Ansatz I
5. Zeitliche-Differenzmethoden (TD) – Ansatz II

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Zeitliche Differenz-Methoden

Überblick

1. Motivation
2. Real-Time Dynamic Programming
3. Die Monte-Carlo-Methode (MC)
4. Zeitliche-Differenzmethoden (TD) – Ansatz I
5. Zeitliche-Differenzmethoden (TD) – Ansatz II
6. Die Familie der TD(λ)-Verfahren

Zeitliche Differenz-Methoden

Überblick

1. Motivation
2. Real-Time Dynamic Programming
3. Die Monte-Carlo-Methode (MC)
4. Zeitliche-Differenzmethoden (TD) – Ansatz I
5. Zeitliche-Differenzmethoden (TD) – Ansatz II
6. Die Familie der TD(λ)-Verfahren
7. Konvergenzaussagen zu TD(λ)

Zeitliche Differenz-Methoden

Überblick

1. Motivation
2. Real-Time Dynamic Programming
3. Die Monte-Carlo-Methode (MC)
4. Zeitliche-Differenzmethoden (TD) – Ansatz I
5. Zeitliche-Differenzmethoden (TD) – Ansatz II
6. Die Familie der TD(λ)-Verfahren
7. Konvergenzaussagen zu TD(λ)

Reinforcement Learning

Reinforcement Learning repräsentiert den Aufgabentyp

Merkmale sind:

- eine a priori unbekannte Umgebung
- ein bewertendes Trainingssignal (Reward, Belohnung/Kosten)
  - z.B. “gut” / “schlecht”
- zeitliche Verzögerungen im Erhalt der Rückmeldung sind der Normalfall
  - Trainingssignal erst nach einer Folge von Entscheidungen

Reinforcement Learning

Reinforcement Learning repräsentiert den Aufgabentyp

Merkmale sind:

- eine a priori unbekannte Umgebung
- ein bewertendes Trainingssignal (Reward, Belohnung/Kosten)
  - z.B. “gut” / “schlecht”
- zeitliche Verzögerungen im Erhalt der Rückmeldung sind der Normalfall
  - Trainingssignal erst nach einer Folge von Entscheidungen

Kennengelernte Lösungsansätze

- Lernproblem als Markov’sches Entscheidungsproblem formulieren
- Lösung finden mit dynamischem Programmieren

$\Rightarrow$ WAS FEHLT?

“Klassische” Lösungsverfahren

- Wertiterationsverfahren (Value Iteration):
  für alle i : $V_{k+1}(i) := \min_a \mathbb{E}_w\{c(i,a) + V_k(f(i,a,w))\}$
- Strategieiterationsverfahren (Policy Evaluation / Iteration):
  für alle i : $V_{k+1}(i) = \mathbb{E}_w\{c(i,\pi(i)) + V_k(f(i,\pi(i),w))\}$

“Klassische” Lösungsverfahren

- Wertiterationsverfahren (Value Iteration):
  für alle i : $V_{k+1}(i) := \min_a \mathbb{E}_w\{c(i,a) + V_k(f(i,a,w))\}$
- Strategieiterationsverfahren (Policy Evaluation / Iteration):
  für alle i : $V_{k+1}(i) = \mathbb{E}_w\{c(i,\pi(i)) + V_k(f(i,\pi(i),w))\}$

$\Rightarrow$ Anwendbar bei relativ wenigen Zuständen (für alle i)

$\Rightarrow$ Außerdem: Ein Modell $f(i,a,w)$ bzw. $p_{ij}(a)$ wird zur Berechnung benötigt.

“Klassische” Lösungsverfahren

- Wertiterationsverfahren (Value Iteration):
  für alle i : $V_{k+1}(i) := \min_a \mathbb{E}_w\{c(i,a) + V_k(f(i,a,w))\}$
- Strategieiterationsverfahren (Policy Evaluation / Iteration):
  für alle i : $V_{k+1}(i) = \mathbb{E}_w\{c(i,\pi(i)) + V_k(f(i,\pi(i),w))\}$

$\Rightarrow$ Anwendbar bei relativ wenigen Zuständen (für alle i)
$\Rightarrow$ Außerdem: Ein Modell $f(i,a,w)$ bzw. $p_{ij}(a)$ wird zur Berechnung benötigt.

Was aber ist mit der Anwendung bei ...

- Schach, Backgammon (10^{20} Zustände), unbekannter Gegner
- autonomen Robotern in unbekannter Umgebung
- Regelungstechnik (kontinuierliche Zustände)

Zwei große Herausforderungen beim optimierenden Lernen

Problemstellungen / Herausforderungen:

- Modell unbekannt
- sehr viele / unendlich viele Zustände
  - Berechnungsvorgang sehr aufwendig / terminiert nach unendlich vielen Iterationen
  - Repräsentation der Pfadkosten speicherintensiv / unmöglich

Zwei große Herausforderungen beim optimierenden Lernen

Problemstellungen / Herausforderungen:

- Modell unbekannt
- sehr viele / unendlich viele Zustände
  - Berechnungsvorgang sehr aufwendig / terminiert nach unendlich vielen Iterationen
  - Repräsentation der Pfadkosten speicherintensiv / unmöglich

Lösungswege:

- Lernen durch direkte Interaktion am Prozess (oder Simulation)
- Verwenden von Approximationsmodellen

Weiteres Vorgehen (“Fahrplan”)

- Methoden zum Schätzen der Kosten einer Strategie ohne Modell (VE 7)
- Bestimmung einer gierigen Strategie auch ohne Modell (VE 8)
- Erlernen der optimalen Wertfunktion ohne Modell (VE 9)
- Repräsentation der Pfadkosten / Wertfunktion (ab VE 10)

Weiteres Vorgehen (“Fahrplan”)

- Methoden zum Schätzen der Kosten einer Strategie ohne Modell (VE 7)
- Bestimmung einer gierigen Strategie auch ohne Modell (VE 8)
- Erlernen der optimalen Wertfunktion ohne Modell (VE 9)
- Repräsentation der Pfadkosten / Wertfunktion (ab VE 10)

- Zunächst (also in der aktuellen Vorlesungseinheit) versuchen wir, die Kosten V^{π} für eine fixe Strategie π zu schätzen.
- Der Übersichtlichkeit halber lassen wir im Folgenden den Diskontierungsfaktor weg ($\gamma = 1$), betrachten also SKP-Probleme.

Zeitliche Differenz-Methoden

Überblick

1. Motivation
2. Real-Time Dynamic Programming
3. Die Monte-Carlo-Methode (MC)
4. Zeitliche-Differenzmethoden (TD) – Ansatz I
5. Zeitliche-Differenzmethoden (TD) – Ansatz II
6. Die Familie der TD(λ)-Verfahren
7. Konvergenzaussagen zu TD(λ)

Lernen am Prozess

Kernideen:

- Durchlaufen von Trajektorien (sogenannter Rollouts) gemäss der aktuellen Strategie π
- Jede einzelne Trajektorie (auch als Sequenz bezeichnet) liefert den Wert der erhaltenen Pfadkosten.

Lernen am Prozess

Kernideen:

- Durchlaufen von Trajektorien (sogenannter Rollouts) gemäss der aktuellen Strategie π
- Jede einzelne Trajektorie (auch als Sequenz bezeichnet) liefert den Wert der erhaltenen Pfadkosten.
- Diese Werte werden verwendet, um die **erwarteten** Kosten zu schätzen.
- Zweck: Iterative Verbesserung der Strategie (z.B. im Rahmen von Policy Iteration)

Verfahren: Real Time Dynamic Programming

- Barto, Sutton, Watkins, 1989

Lernen am Prozess

Kernideen:

- Durchlaufen von Trajektorien (sogenannter Rollouts) gemäss der aktuellen Strategie π
- Jede einzelne Trajektorie (auch als Sequenz bezeichnet) liefert den Wert der erhaltenen Pfadkosten.
- Diese Werte werden verwendet, um die **erwarteten** Kosten zu schätzen.
- Zweck: Iterative Verbesserung der Strategie (z.B. im Rahmen von Policy Iteration)

Verfahren: Real Time Dynamic Programming

- Barto, Sutton, Watkins, 1989

$\Rightarrow$ “Online”-Lernen, Lernen durch Interaktion

$\Rightarrow$ Lernen auf “interessanten” Teilmengen des Zustandsraums

Lernen in Interaktion mit der Umgebung

Beispielalgorithmus

- **Init** Prozess mit Anfangszustand s_0
- **While** nicht in Terminalzustand
  - **Aktionswahl**
    $a_t := \pi(s_t)$
  - **Exploration**
  - **Anwendung auf Prozess**
    $s_{t+1} = f(s_t, a_t, w_t)$
- **EndWhile**

$\Rightarrow$ “Sammeln von Erfahrungen” – Anpassung der Werte der Wertfunktion nach oder während der Trajektorie

Zwischenzusammenfassung

- Ziel ist das Lernen entlang von tatsächlich ausgeführten Regeltrajektorien
- Erfahrungen werden in der Umwelt gesammelt, ohne dass dabei ein Modell benötigt wird

Zwischenzusammenfassung

- Ziel ist das Lernen entlang von tatsächlich ausgeführten Regeltrajektorien
- Erfahrungen werden in der Umwelt gesammelt, ohne dass dabei ein Modell benötigt wird
- Ziel ist außerdem die inkrementelle Verbesserung der Schätzung der Pfadkosten (Wertfunktion) anhand von beobachteten Zustandsübergängen
  - Dies bezeichnet man auch als “Sampling” bzw. Monte-Carlo-Methoden (MC)

Zeitliche Differenz-Methoden

Überblick

1. Motivation
2. Real-Time Dynamic Programming
3. Die Monte-Carlo-Methode (MC)
4. Zeitliche-Differenzmethoden (TD) – Ansatz I
5. Zeitliche-Differenzmethoden (TD) – Ansatz II
6. Die Familie der TD(λ)-Verfahren
7. Konvergenzaussagen zu TD(λ)

Schätzung des Erwartungswerts

Allgemeines Problem

gegeben: Zufallsvariable v mit unbekanntem Erwartungswert $\bar{v}$.
Seien $v_1, v_2, \ldots$ beispielhaft gezogene Werte (Samples) dieser Zufallsvariable.

Ziel: Schätzung des Mittelwerts $\bar{v}$.

Schätzung des Erwartungswerts

Allgemeines Problem

gegeben: Zufallsvariable v mit unbekanntem Erwartungswert $\bar{v}$. Seien $v_1, v_2, \ldots$ beispielhaft gezogene Werte (Samples) dieser Zufallsvariable.

Ziel: Schätzung des Mittelwerts $\bar{v}$.

Definition (Monte-Carlo-Schätzung eines Erwartungswerts)

Die Bildung des Mittelwerts eines Stichprobe vom Umfang N ist definiert als

$$\bar{v}_N = \frac{1}{N} \sum_{k=1}^{N} v_k$$

Schätzung des Erwartungswerts

Allgemeines Problem

gegeben: Zufallsvariable v mit unbekanntem Erwartungswert $\bar{v}$.
Seien $v_1, v_2, \ldots$ beispielhaft gezogene Werte (Samples) dieser Zufallsvariable.

Ziel: Schätzung des Mittelwerts $\bar{v}$.

Definition (Monte-Carlo-Schätzung eines Erwartungswerts)

Die Bildung des Mittelwerts eines Stichprobe vom Umfang N ist definiert als

$$\bar{v}_N = \frac{1}{N} \sum_{k=1}^{N} v_k$$

Eine **rekursive Schätzung** des Erwartungswerts ist gegeben durch

$$\bar{v}_{N+1} = \bar{v}_N + \alpha_{N+1} \left(v_{N+1} - \bar{v}_N \right)$$

mit Lernrate α.

Stochastische Approximation

Für die rekursive Schätzung gibt es zwei äquivalente Darstellungen:

$$\begin{aligned} \bar{v}_{N+1} &= \bar{v}_N + \alpha_{N+1} \left(v_{N+1} - \bar{v}_N \right) \\ &= \end{aligned}$$

Stochastische Approximation

Für die rekursive Schätzung gibt es zwei äquivalente Darstellungen:

$$\begin{aligned} \bar{v}_{N+1} &= \bar{v}_N + \alpha_{N+1} \left(v_{N+1} - \bar{v}_N \right) \\ &= \left(1 - \alpha_{N+1} \right) \bar{v}_N + \alpha_{N+1} \, v_{N+1} \end{aligned}$$

Wahl der Lernrate: z.B. $\alpha_N := \frac{1}{N}$

Stochastische Approximation

Für die rekursive Schätzung gibt es zwei äquivalente Darstellungen:

$$\begin{aligned} \bar{v}_{N+1} &= \bar{v}_N + \alpha_{N+1}\,(v_{N+1} - \bar{v}_N) \\ &= (1 - \alpha_{N+1})\,\bar{v}_N + \alpha_{N+1}\,v_{N+1} \end{aligned}$$

Wahl der Lernrate: z.B. $\alpha_N := \frac{1}{N}$

Bemerkung: Diese Gleichung konvergiert gegen den Erwartungswert von v auch für allgemeinere Wahl von α, solange gilt:

1. $\sum_N^\infty \alpha_N = \infty$ und

Stochastische Approximation

Für die rekursive Schätzung gibt es zwei äquivalente Darstellungen:

$$\begin{aligned} \bar{v}_{N+1} &= \bar{v}_N + \alpha_{N+1}\,(v_{N+1} - \bar{v}_N) \\ &= (1 - \alpha_{N+1})\,\bar{v}_N + \alpha_{N+1}\,v_{N+1} \end{aligned}$$

Wahl der Lernrate: z.B. $\alpha_N := \frac{1}{N}$

Bemerkung: Diese Gleichung konvergiert gegen den Erwartungswert von v auch für allgemeinere Wahl von α, solange gilt:

1. $\sum_N^\infty \alpha_N = \infty$ und
2. $\sum_N^\infty \alpha_N^2 < \infty$

Umgangssprachlich:
α muss gegen null gehen, aber nicht zu schnell.
$\Rightarrow$ Diese Verfahren sind unter dem Namen STOCHASTISCHE APPROXIMATION bekannt.

Strategiebewertung mit MC-Methoden (1)

Im Folgenden soll gelten:

- Betrachtet werden stochastische Kürzester-Pfad-Probleme.
- Es geht um eine fest vorgegebene, erfüllende (proper) Strategie π.
- **Ziel:** Schätze die erwarteten Pfadkosten V^{π}
  - Dies ist der Ausgangspunkt für Strategieiterationsverfahren.

Strategiebewertung mit MC-Methoden (1)

Im Folgenden soll gelten:

- Betrachtet werden stochastische Kürzester-Pfad-Probleme.
- Es geht um eine fest vorgegebene, erfüllende (proper) Strategie π.
- **Ziel:** Schätze die erwarteten Pfadkosten V^{π}
  - Dies ist der Ausgangspunkt für Strategieiterationsverfahren.
- Wir betrachten zwei mögliche Vorgehensweisen:

1. Generiere Trajektorien für jeden Startzustand *i* bis zum Erreichen des Terminalzustands.
   $\Rightarrow$ trajektorienbasierter Ansatz / Monte-Carlo-Methode
2. Benutze auch Informationen entlang der Trajektorien, um so auch noch die Bewertungen von Zwischenzuständen anzupassen.
   $\Rightarrow$ Zeitliche Differenzmethode (Temporal Difference)

Strategiebewertung mit MC-Methoden (2)

Vereinfachungen und Annahmen

- Da die Strategie fix ist, betrachten wir im folgenden nur aktionsunabhängige Übergangswahrscheinlichkeiten p_{ij} und -kosten $c(i)$.
  - Frage: Warum ist das ok?

Strategiebewertung mit MC-Methoden (2)

Vereinfachungen und Annahmen

- Da die Strategie fix ist, betrachten wir im folgenden nur aktionsunabhängige Übergangswahrscheinlichkeiten p_{ij} und -kosten $c(i)$.
  - Frage: Warum ist das ok?
- Jede Trajektorie endet im Terminalzustand ’0’.
- Ihre Länge wird mit N bezeichnet; es gilt also $s_N = 0$.
- Ausserdem schreiben wir kurz V statt V^π (da π ja fix ist).

Ansatz I: Trajektorienbasiert

Monte-Carlo-Strategiebewertung

Wir betrachten Trajektorie t, startend im Zustand i. Pfadkosten:

Ansatz I: Trajektorienbasiert

Monte-Carlo-Strategiebewertung

Wir betrachten Trajektorie t, startend im Zustand i. Pfadkosten:

$$g(i,t) = c(s_0) + c(s_1) + \ldots + c(s_{N-1}), \quad s_0 = i$$

Ansatz I: Trajektorienbasiert

Monte-Carlo-Strategiebewertung

Wir betrachten Trajektorie t, startend im Zustand i. Pfadkosten:

$$g(i,t) = c(s_0) + c(s_1) + \ldots + c(s_{N-1}), \quad s_0 = i$$

Dann gilt für V: $V(i) = \mathbb{E}[$

Ansatz I: Trajektorienbasiert

Monte-Carlo-Strategiebewertung

Wir betrachten Trajektorie t, startend im Zustand i. Pfadkosten:

$$g(i,t) = c(s_0) + c(s_1) + \ldots + c(s_{N-1}), \quad s_0 = i$$

Dann gilt für V: $V(i) = \mathbb{E}[g(i,t)]$

V kann auch per Mittelwertbildung abgeschätzt werden durch

$$V(i) =$$

Ansatz I: Trajektorienbasiert

Monte-Carlo-Strategiebewertung

Wir betrachten Trajektorie t, startend im Zustand i. Pfadkosten:

$$g(i,t) = c(s_0) + c(s_1) + \ldots + c(s_{N-1}), \quad s_0 = i$$

Dann gilt für V: $V(i) = \mathbb{E}[g(i,t)]$

V kann auch per Mittelwertbildung abgeschätzt werden durch

$$V(i) = \frac{1}{K} \sum_{t=1}^{K} g(i,t)$$

Ansatz I: Trajektorienbasiert

Monte-Carlo-Strategiebewertung

Wir betrachten Trajektorie t, startend im Zustand i. Pfadkosten:

$$g(i,t) = c(s_0) + c(s_1) + \ldots + c(s_{N-1}), \quad s_0 = i$$

Dann gilt für V: $V(i) = \mathbb{E}[g(i,t)]$

V kann auch per Mittelwertbildung abgeschätzt werden durch

$$V(i) = \frac{1}{K} \sum_{t=1}^{K} g(i,t)$$

Algorithmus: MC-Strategiebewertung

Es sei $V_0(i) = 0 \, \forall i$ und es werden durch den Agenten wiederholt Trajektorien in Interaktion mit der Umgebung unter Verwendung der Strategie π abgelaufen. Dann wird die Wertfunktion wie folgt aktualisiert:

$$V_t(i) = V_{t-1}(i) + \alpha_t(g(i,t) - V_{t-1}(i))$$

wobei z.B. $\alpha_t = 1/t$ verwendet werden kann.

Zeitliche Differenz-Methoden

Überblick

1. Motivation
2. Real-Time Dynamic Programming
3. Die Monte-Carlo-Methode (MC)
4. Zeitliche-Differenzmethoden (TD) – Ansatz I
5. Zeitliche-Differenzmethoden (TD) – Ansatz II
6. Die Familie der TD(λ)-Verfahren
7. Konvergenzaussagen zu TD(λ)

Zeitliche-Differenzmethoden (1)

Temporal-Difference-Methoden (TD)

Kernideen:

- gehen zurück auf Samuel (“Checker Player”, 1959)
- Idee: Anpassung nach jedem Zustandsübergang
- Zeitliche Differenz: Diskrepanz zwischen zwei aufeinanderfolgenden Zustandsbewertungen

Zeitliche-Differenzmethoden (1)

Temporal-Difference-Methoden (TD)

Kernideen:

- gehen zurück auf Samuel (“Checker Player”, 1959)
- Idee: Anpassung nach jedem Zustandsübergang
- Zeitliche Differenz: Diskrepanz zwischen zwei aufeinanderfolgenden Zustandsbewertungen

Ausgangspunkt:
Bei den Überlegungen zur Monte-Carlo-Strategiebewertung betrachteten wir eine Trajektorie t beginnend im Zustand s_k:

$$g(s_k, t) = c(s_k) + c(s_{k+1}) + \ldots + c(s_{N-1})$$

$$V_t(s_k) = V_{t-1}(s_k) + \alpha_t(g(s_k, t) - V_{t-1}(s_k))$$

Zeitliche-Differenzmethoden (2)

Temporal-Difference-Methoden (TD)

Einsetzen (Betrachtung für eine einzelne Episode):

$$V_t(s_k) = V_{t-1}(s_k) + \alpha_t \left(c(s_k) + c(s_{k+1}) + \ldots + c(s_{N-1}) - V_{t-1}(s_k) \right)$$

Zeitliche-Differenzmethoden (2)

Temporal-Difference-Methoden (TD)

Einsetzen (Betrachtung für eine einzelne Episode):

$$V_t(s_k) = V_{t-1}(s_k) + \alpha_t \left(c(s_k) + c(s_{k+1}) + \ldots + c(s_{N-1}) - V_{t-1}(s_k)\right)$$

Umformungstrick: Addiere und subtrahiere jedes $V_{t-1}(s_l)$

$$\begin{aligned} V_t(s_k) = V_{t-1}(s_k) + \alpha_t(\quad & c(s_k) + V_{t-1}(s_{k+1}) - V_{t-1}(s_k) \\ + \quad & c(s_{k+1}) + V_{t-1}(s_{k+2}) - V_{t-1}(s_{k+1}) \\ + \quad & c(s_{k+2}) + V_{t-1}(s_{k+3}) - V_{t-1}(s_{k+2}) \\ + \quad & \ldots \\ + \quad & c(s_{N-1}) + V_{t-1}(s_N) - V_{t-1}(s_{N-1})) \end{aligned}$$

wobei benutzt wird, dass $V(s_N) = 0$ gilt.

Zeitliche-Differenzmethoden (3)

Temporal-Difference-Methoden (TD)

Definiere nun:

Definition (Zeitlicher Differenzfehler (TD-Fehler))

Die Differenz

$$d_k := c(s_k) + V_{t-1}(s_{k+1}) - V_{t-1}(s_k)$$

zwischen den erwarteten Pfadkosten $V_{t-1}(s_k)$ und der im nächsten Schritt erfahreren "besseren" Schätzung der Pfadkosten $c(s_k) + V_{t-1}(s_{k+1})$ wird als zeitlicher Differenzfehler oder kurz als TD-Fehler (engl. Temporal Difference Error) bezeichnet.

Zeitliche-Differenzmethoden (3)

Temporal-Difference-Methoden (TD)

Definiere nun:

Definition (Zeitlicher Differenzfehler (TD-Fehler))

Die Differenz

$$d_k := c(s_k) + V_{t-1}(s_{k+1}) - V_{t-1}(s_k)$$

zwischen den erwarteten Pfadkosten $V_{t-1}(s_k)$ und der im nächsten Schritt erfahreren "besseren" Schätzung der Pfadkosten $c(s_k) + V_{t-1}(s_{k+1})$ wird als zeitlicher Differenzfehler oder kurz als TD-Fehler (engl. Temporal Difference Error) bezeichnet.

Damit lässt sich die Aktualisierungsvorschrift für V_t, die aus dem Umformungstrick hervorgegangen ist, schreiben als

$$V_t(s_k) = V_{t-1}(s_k) + \alpha_t (d_k + d_{k+1} + \ldots + d_{N-1})$$

Der TD(1)-Algorithmus

Online-Version: Der Anpassungsschritt kann sofort durchgeführt werden, nachdem ein einzelner Übergang $s_l \rightarrow s_{l+1}$ gemacht ist. Dann steht der TD-Fehler (zeitliche Differenz) fest:

$$d_l = c(s_l) + V(s_{l+1}) - V(s_l)$$

Der TD(1)-Algorithmus

Online-Version: Der Anpassungsschritt kann sofort durchgeführt werden, nachdem ein einzelner Übergang $s_l \rightarrow s_{l+1}$ gemacht ist. Dann steht der TD-Fehler (zeitliche Differenz) fest:

$$d_l = c(s_l) + V(s_{l+1}) - V(s_l)$$

Algorithmus: TD(1)

Durchlaufe eine Trajektorie t mit $s_0, s_1, \ldots, s_l, \ldots, s_N$:

FOR alle bereits "durchlaufenen" Zustände $k \leq l$:

$$V_t(s_k) = V_{t-1}(s_k) + \alpha_t \, d_l$$

Wichtige Bemerkung: Der TD(1)-Algorithmus ist äquivalent zur trajektorienbasierten Methode (geht durch algebraische Umformung daraus hervor), d.h. zur Monte-Carlo-Strategiebewertung.

Zeitliche Differenz-Methoden

Überblick

1. Motivation
2. Real-Time Dynamic Programming
3. Die Monte-Carlo-Methode (MC)
4. Zeitliche-Differenzmethoden (TD) – Ansatz I
5. Zeitliche-Differenzmethoden (TD) – Ansatz II
6. Die Familie der TD(λ)-Verfahren
7. Konvergenzaussagen zu TD(λ)

TD-Methoden: Zwei Ansätze

Erinnerung:

1. Generiere Trajektorien für jeden Startzustand *i* bis zum Erreichen des Terminalzustands.
   $\Rightarrow$ trajektorienbasierter Ansatz / Monte-Carlo-Methode
   $\Rightarrow$ äquivalent zu TD(1)
2. Benutze auch Informationen entlang der Trajektorien, um so auch noch die Bewertungen von Zwischenzuständen anzupassen.
   $\Rightarrow$ übergangsbasierter Ansatz
   $\Rightarrow$ Zeitliche Differenzmethode (Temporal Difference)
   $\Rightarrow$ TD(λ) mit $\lambda < 1$

Ansatz II: Übergangsbasiert (1)

Verwendung der Bellman-Gleichung

Kernidee:

- Anwendung einer einzelnen Aktualisierung gemäß des Bellman'schen Optimalitätsprinzips, nach jedem einzelnen Übergang $s_k \rightarrow s_{k+1}$ wie folgt

$$V_t(s_k) = \mathbb{E}_w[c(s_k) + V_{t-1}(s_{k+1})], \quad \text{mit } s_{k+1} = f(s_k, \pi(s_k), w)$$

Problem dabei:

- Wenn $V_{t-1}(s_{k+1})$ fehlerhaft, so ist auch $V_t(s_k)$ fehlerhaft.

Ansatz II: Übergangsbasiert (2)

Verwendung der Bellman-Gleichung

Lösungsidee:

- Die Übergangskosten sind genau, also nimm mehrere Übergangskosten mit in den Aktualisierungsschritt auf.
- zum Beispiel eine 3-Schritt-Vorschau:

$$V_t(s_k) = \mathbb{E}_w[c(s_k) + c(s_{k+1}) + c(s_{k+2}) + V_{t-1}(s_{k+3})]$$

Ansatz II: Übergangsbasiert (2)

Verwendung der Bellman-Gleichung

Lösungsidee:

- Die Übergangskosten sind genau, also nimm mehrere Übergangskosten mit in den Aktualisierungsschritt auf.
- zum Beispiel eine 3-Schritt-Vorschau:

$$V_t(s_k) = \mathbb{E}_w[c(s_k) + c(s_{k+1}) + c(s_{k+2}) + V_{t-1}(s_{k+3})]$$

Verallgemeinerung

- berücksichtige die Kosten der $l + 1$ nächsten Übergänge
- dadurch wird die Schätzung genauer

$$V_t(s_k) = \mathbb{E}[V_{t-1}(s_{k+l+1}) + \sum_{m=0}^{l} c(s_{k+m})]$$

Ansatz II: Übergangsbasiert (3)

Verwendung der Bellman-Gleichung

$$V_t(s_k) = \mathbb{E}[V_{t-1}(s_{k+l+1}) + \sum_{m=0}^{l} c(s_{k+m})]$$

Bemerkung:

- Wenn $V_{t-1} = V^{\pi}$ ist (also die Wertfunktion für π bereits perfekt abgeschätzt wird), so unterscheidet sich der rechte Teil nicht für unterschiedliche l.
- Will man kein fixes l willkürlich festlegen, so kann man z.B. V^{π} für verschiedene l ausrechnen und den Mittelwert bilden.
- Weitere Idee: Gewichte die Werte für verschiedene l (z.B. exponentiell abfallend mit wachsendem l)
  - Idee wird auf den nächsten Folien weiter verfolgt $\Rightarrow$ TD(λ)

Zeitliche Differenz-Methoden

Überblick

1. Motivation
2. Real-Time Dynamic Programming
3. Die Monte-Carlo-Methode (MC)
4. Zeitliche-Differenzmethoden (TD) – Ansatz I
5. Zeitliche-Differenzmethoden (TD) – Ansatz II
6. Die Familie der TD(λ)-Verfahren
7. Konvergenzaussagen zu TD(λ)

Exponentiell abfallende Gewichtung

Ausgangspunkt: Berücksichtigung mehrerer Folgeschritte bei der Aktualisierung

$$V_t(s_k) = \mathbb{E}[V_{t-1}(s_{k+l+1}) + \sum_{m=0}^{l} c(s_{k+m})]$$

Exponentiell abfallende Gewichtung

Ausgangspunkt: Berücksichtigung mehrerer Folgeschritte bei der Aktualisierung

$$V_t(s_k) = \mathbb{E}[V_{t-1}(s_{k+l+1}) + \sum_{m=0}^{l} c(s_{k+m})]$$

Kernidee: Berücksichtige alle Werte für l mit abfallendender Gewichtung: Das bedeutet, multipliziere die obige Gleichung mit $(1-\lambda)\lambda^l$, wobei $0 \leq \lambda < 1$ und $(1-\lambda)$ ist ein Korrekturterm, weil $\sum_{l=0}^{\infty} \lambda^l = \frac{1}{1-\lambda}$):

$$V_t(s_k) = (1-\lambda)\mathbb{E}\left[\sum_{l=0}^{\infty} \lambda^l \left(V_{t-1}(s_{k+l+1}) + \sum_{m=0}^{l} c(s_{k+m})\right)\right]$$

Dies lässt sich in eine TD-Form bringen.

Der $TD(\lambda)$-Algorithmus (1)

Ausgangspunkt: Exponentielle Gewichtung der Pfadkosten-“Datenpunkte”

$$V_t(s_k) = (1-\lambda)\mathbb{E}\left[\sum_{l=0}^{\infty}\lambda^l\left(V_{t-1}(s_{k+l+1}) + \sum_{m=0}^{l} c(s_{k+m})\right)\right]$$

Diese Idee umgeformt in “TD-Form” ergibt den **$TD(\lambda)$-Algorithmus**

(Beweis/Umformung: sh. Literatur)

Der $TD(\lambda)$-Algorithmus (1)

Ausgangspunkt: Exponentielle Gewichtung der Pfadkosten-"Datenpunkte"

$$V_t(s_k) = (1-\lambda)\mathbb{E}\left[\sum_{l=0}^{\infty}\lambda^l\left(V_{t-1}(s_{k+l+1}) + \sum_{m=0}^{l} c(s_{k+m})\right)\right]$$

Diese Idee umgeformt in "TD-Form" ergibt den **$TD(\lambda)$-Algorithmus**

(Beweis/Umformung: sh. Literatur)

Algorithmus: TD(λ)

Durchlaufe Trajektorie t mit $s_0, s_1, \ldots, s_N$ und aktualisiere:

$$V_t(s_k) \quad = \quad V_{t-1}(s_k) + \alpha \sum_{m=0}^{N} \lambda^m d_{k+m}$$

mit zeitlichem Differenzfehler $d_m = c(s_m) + V_{t-1}(s_{m+1}) - V_{t-1}(s_m)$

Der $TD(\lambda)$-Algorithmus (2)

Online-Version: Wie schon bei der Besprechung von TD(1)/MC gilt, dass der Anpassungsschritt sofort durchgeführt werden kann, nachdem ein einzelner Übergang $s_l \rightarrow s_{l+1}$ gemacht ist.
Dann steht der TD-Fehler (zeitlicher Differenzfehler) für Schritt l fest und kann zum Update benutzt werden: $d_l = c(s_l) + V(s_{l+1}) - V(s_l)$

Der $TD(\lambda)$-Algorithmus (2)

Online-Version: Wie schon bei der Besprechung von TD(1)/MC gilt, dass der Anpassungsschritt sofort durchgeführt werden kann, nachdem ein einzelner Übergang $s_l \rightarrow s_{l+1}$ gemacht ist.
Dann steht der TD-Fehler (zeitlicher Differenzfehler) für Schritt l fest und kann zum Update benutzt werden: $d_l = c(s_l) + V(s_{l+1}) - V(s_l)$

Algorithmus: TD(λ), Online-Version

Durchlaufe Trajektorie $s_0, s_1, \ldots, s_l, \ldots, s_N$:

FOR alle bereits "durchlaufenen" Zustände $k \leq l$
{
$V_t(s_k) = V_{t-1}(s_k) + \alpha \, \lambda^{l-k} \, d_l$
}

Diskussion $TD(\lambda)$ (1)

Beispiel:

- kleines Beispiel: gleich
- ausführliches Beispiel: siehe Übungen

Bemerkungen:

- Verfahren geht zurück auf Richard Sutton, 1987: *Learning to predict by the methods of temporal differences* und ist als $TD(\lambda)$-Algorithmus bekannt (Sutton, 1987).

Sonderfälle:

- Für $\lambda = 1$ ergibt sich das trajektorienbasierte Monte-Carlo-Verfahren.
- Für $\lambda = 0$: siehe nächste Folie

Diskussion $TD(\lambda)$ (2)

Sonderfälle:

- Für $\lambda = 0$ ergibt sich (wegen $0^0 = 1$, $\lambda^m = 0^m = 0$ für $m > 0$)

$$V_t(s_k) = V_{t-1}(s_k) + \alpha \left(c(s_k) + V_{t-1}(s_{k+1}) - V_{t-1}(s_k)\right)$$

- Dieses Verfahren bezeichnet man als **TD(0)-Algorithmus**.

Algorithmus: TD(0)

Durchlaufe beliebige Zustandsübergänge $s_k \rightarrow s_{k+1}$ und aktualisiere die Wertfunktion V gemäß:

$$V(s_k) \leftarrow V(s_k) + \alpha \left(c(s_k) + V(s_{k+1}) - V(s_k)\right)$$

Diskussion $TD(\lambda)$ (3)

Sonderfall $\lambda = 0$:

- TD(0) entspricht der Realisierung der Aktualisierung gemäß der Bellman-Gleichung für einen einzelnen Übergang (übergangsbasiert) im Rahmen einer stochastischen Approximation.
- Das Verfahren für $\lambda = 0$ kann für beliebige Zustände im Zustandsraum angewendet werden und ist damit insbesondere *nicht* auf das Durchlaufen ganzer Trajektorien angewiesen.
- Das Verfahren ist auch für diskontierte MDPs einsetzbar. Dann kommt in der Formel der Diskontierungsfaktor γ mit hinzu.

$$V(s_k) \leftarrow V(s_k) + \alpha \left(c(s_k) + \gamma V(s_{k+1}) - V(s_k)\right)$$

Beispiel: für die Anwendung von TD(λ) (1)

TD(λ)-Anwendung der Online-Version

- nach Übergang 1: $(s_0 \rightarrow s_1)$

$$V(s_0) := V(s_0) + \alpha d_0$$

- nach Übergang 2: $(s_1 \rightarrow s_2)$

$$V(s_0) := V(s_0) + \alpha \lambda d_1$$

$$V(s_1) := V(s_1) + \alpha d_1$$

Beispiel: für die Anwendung von TD(λ) (2)

TD(λ)-Anwendung

- nach Übergang 3: $(s_2 \rightarrow s_3)$

$$V(s_0) := V(s_0) + \alpha\lambda^2 d_2$$

$$V(s_1) := V(s_1) + \alpha\lambda d_2$$

$$V(s_2) := V(s_2) + \alpha d_2$$

- allgemein: $(s_k \rightarrow s_{k+1})$

$$V(s_0) := V(s_0) + \alpha\lambda^k d_k$$

$$V(s_1) := V(s_1) + \alpha\lambda^{k-1} d_k$$

...

$$V(s_k) := V(s_k) + \alpha d_k$$

Beispiel: für die Anwendung von TD(λ) (3)

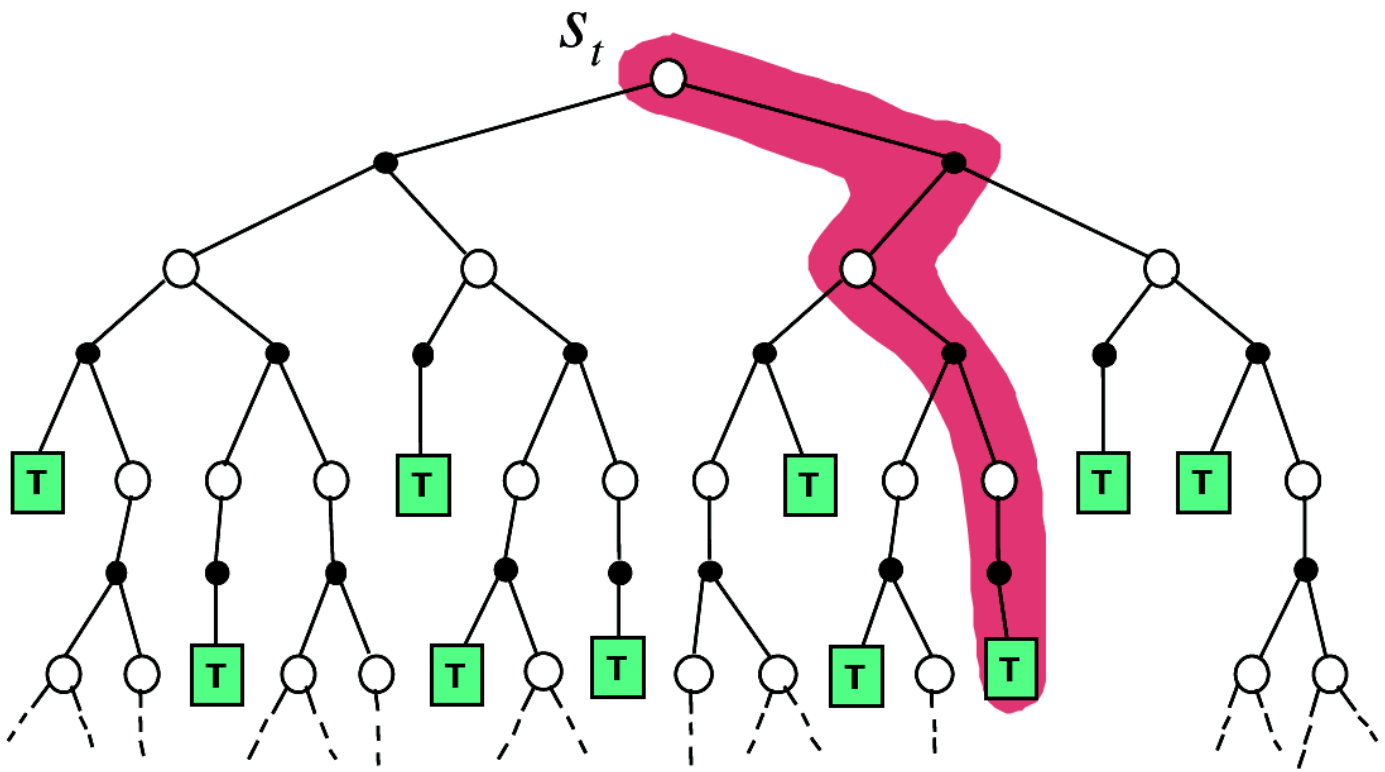
Offline-Variante von $TD(\lambda)$:

- Durchlaufe die komplette Trajektorie
- führe die Anpassung der Werte anschliessend durch
- Ergebnis: Wenn ein Zustand öfters besucht wird, unterscheiden sich die Resultate der Online-Version leicht von denen der Offline-Version.

Zusammenfassung TD-Verfahren (1)

Aktualisierung bei TD(1) (Monte-Carlo-Strategiebewertung)

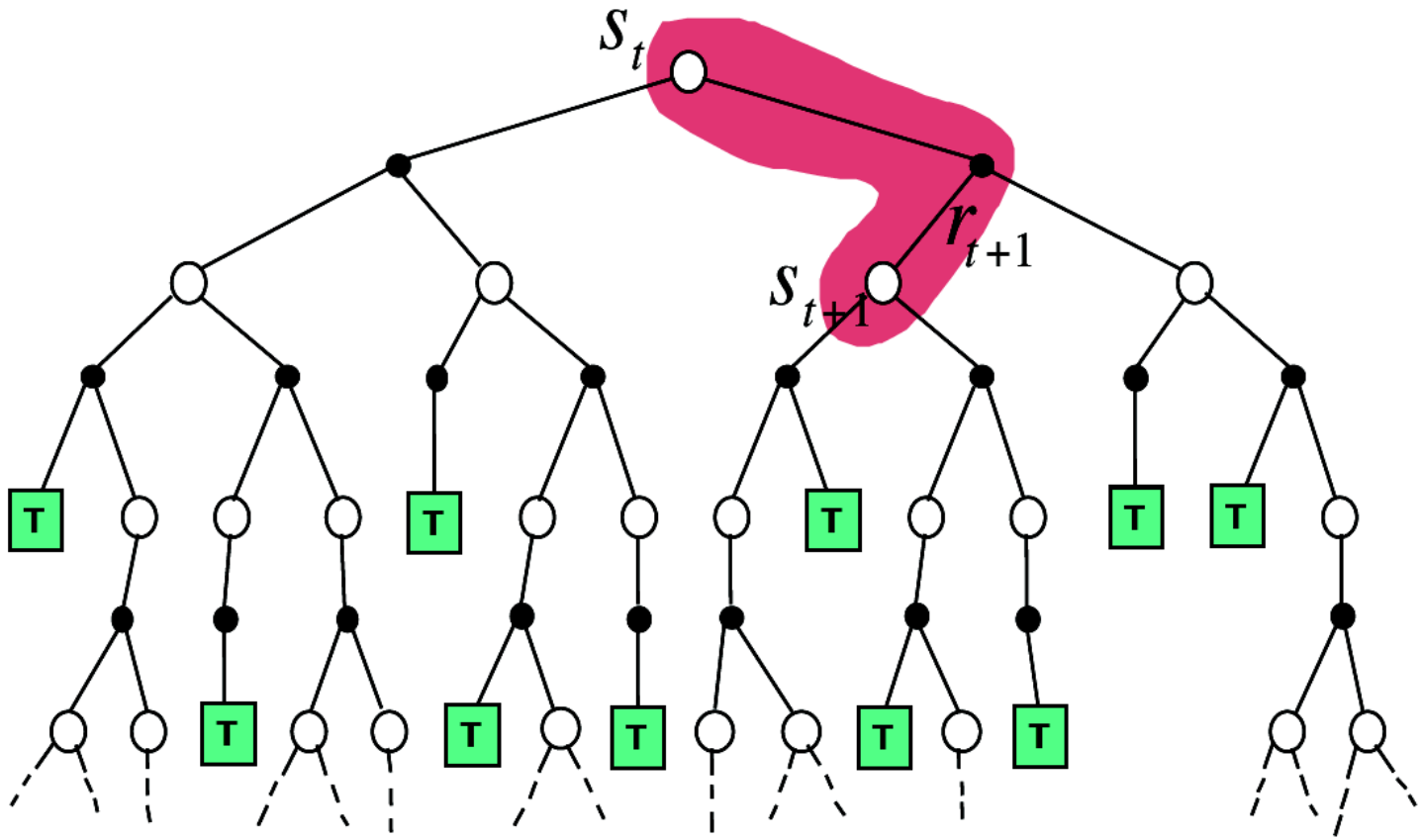
$$V(s_t) \leftarrow V(s_t) + \alpha(g(s_t) - V(s_t))$$



Zusammenfassung TD-Verfahren (2)

Aktualisierung bei TD(0) (Temporal Difference Backup)

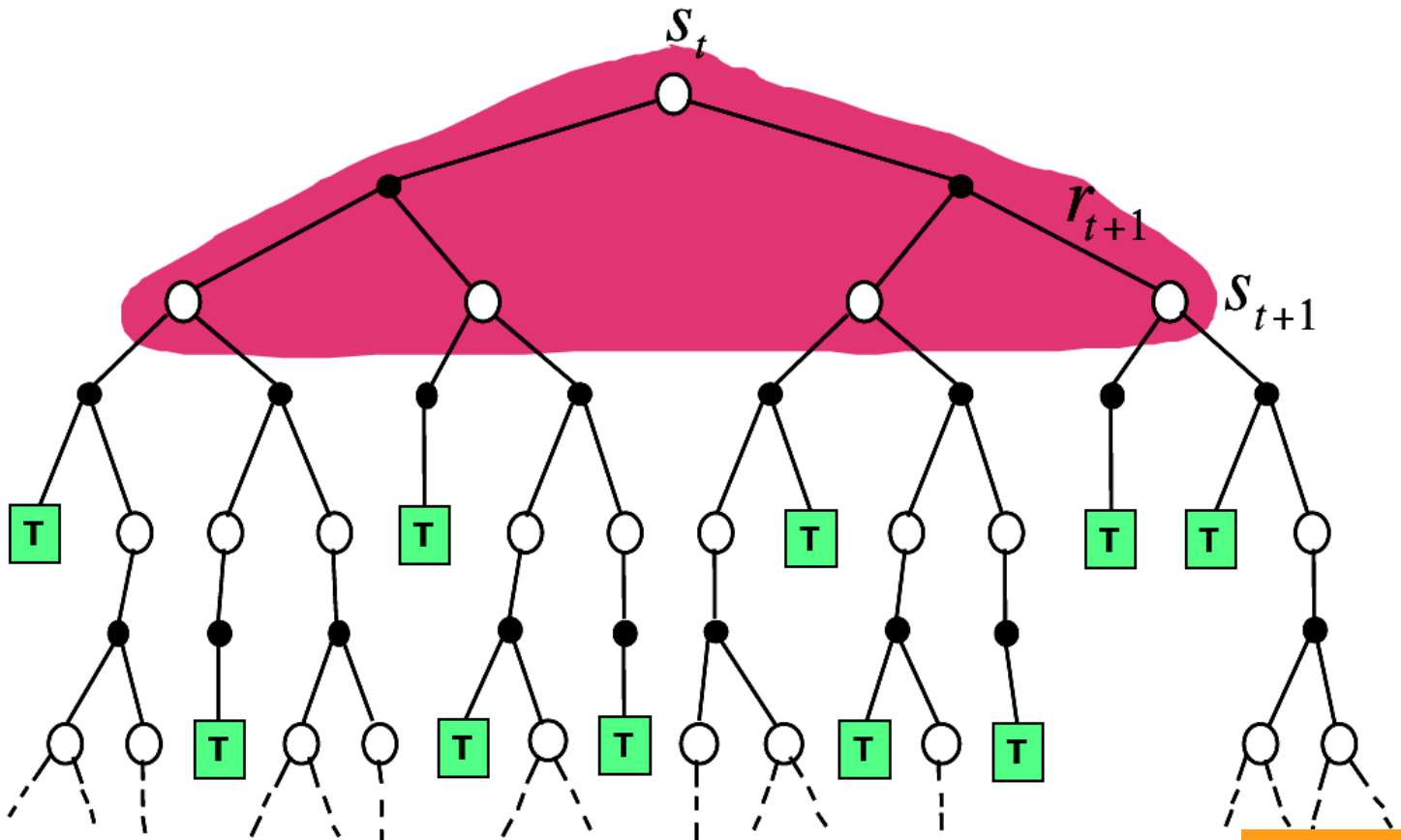
$$V(s_t) \leftarrow V(s_t) + \alpha \left(c(s_t) + \gamma V(s_{t+1}) - V(s_t)\right)$$



Zusammenfassung TD-Verfahren (3)

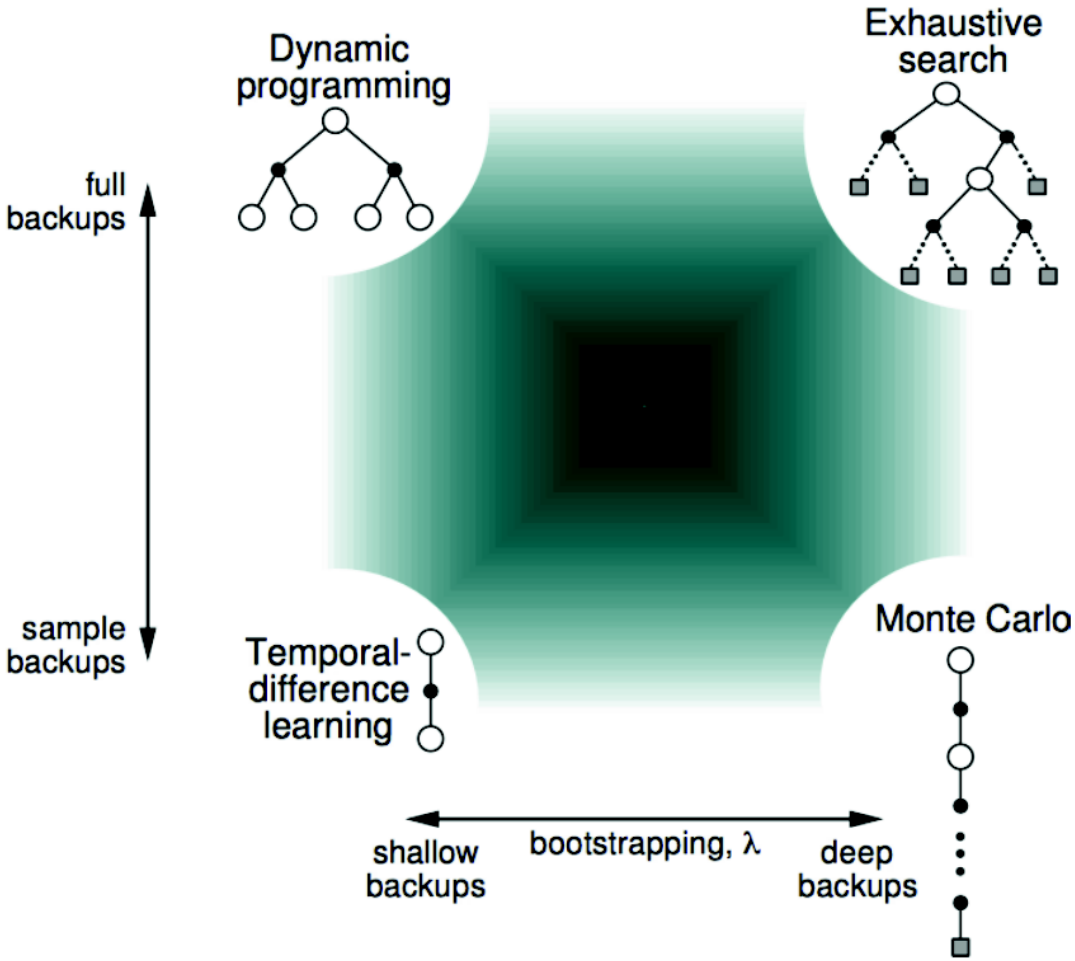
Zum Vergleich: Aktualisierung bei DP (Dynamic Programming)

$$V(s_t) \leftarrow \sum_{s_{t+1} \in S} p_{s_t, s_{t+1}}(\pi(s_t)) \left(c(s_t, \pi(s_t)) + \gamma V(s_{t+1})\right)$$



Zusammenfassung TD-Verfahren (4)

Methodenüberblick



Zeitliche Differenz-Methoden

Überblick

1. Motivation
2. Real-Time Dynamic Programming
3. Die Monte-Carlo-Methode (MC)
4. Zeitliche-Differenzmethoden (TD) – Ansatz I
5. Zeitliche-Differenzmethoden (TD) – Ansatz II
6. Die Familie der TD(λ)-Verfahren
7. Konvergenzaussagen zu TD(λ)

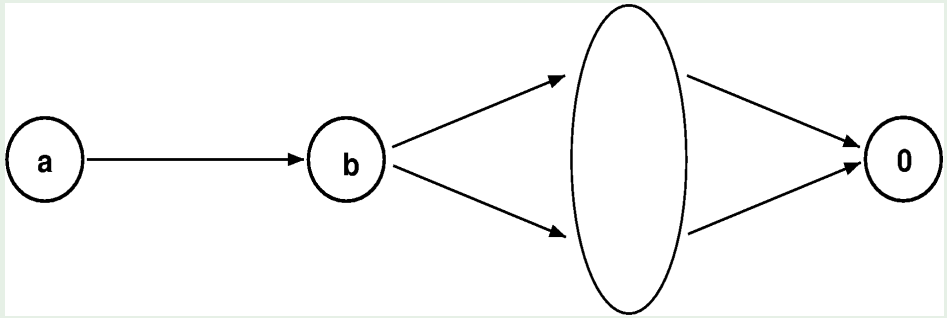
Konvergenz TD(λ)

Man kann zeigen, dass der *TD*(λ)-Algorithmus für beliebige Werte von λ mit Wahrscheinlichkeit 1 gegen V^{π} konvergiert, wenn

1. jeder Zustand unendlich oft besucht wird und
2. die Schrittweite α (Lernrate) “vernünftig” abnimmt
   - Vernünftig bedeutet dabei gemäß der Forderungen der stochastische Approximation.

Überlegungen zur Wahl von λ (1)

Beispiel

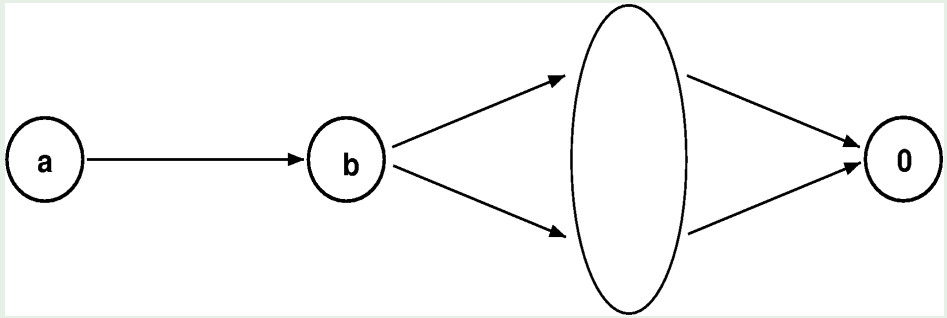


Eigenschaften:

- Übergang von 'a' $\rightarrow$ 'b' sei deterministisch, danach geht es stochastisch weiter.
- Jede Trajektorie starte in $s_0 = a$.
- Seien $g(a, t)$ die Pfadkosten für Trajektorie t.
- Da 'a' nur einmal pro Trajektorie besucht wird, ist der Mittelwert der $g(a, t)$ ein erwartungstreuer Schätzer und das beste, was man hier machen kann.
- Also: TD(λ) mit $\lambda = 1$ und $\alpha = 1/t$ ist gute Wahl.

Überlegungen zur Wahl von λ (2)

Beispiel



Eigenschaften:

- ABER: Annahme, es gebe eine gute Schätzung $V^{\pi}(b)$ für 'b'.
  - Beispielsweise war 'b' schon Ausgangspunkt mehrerer Trajektorien.
- $TD(1)$ benutzt diese Information überhaupt nicht!
- Falls $V^{\pi}(b)$ eine gute Schätzung des Erwartungswerts ab 'b' liefert, kann der TD-Algorithmus mit $\lambda = 0$ und $\alpha = 1$ den richtigen Wert für 'a' in einem einizen Schritt bestimmen!
  - Da der Übergang von 'a' nach 'b' ja deterministisch ist.

Überlegungen zur Wahl von λ (3)

Weitere Aspekte:

- Diskussion in [Singh, Dayan, 1996]: Analytisch hergeleitete Formeln, um den mittleren quadratischen Fehler vorherzusagen. Ganz grob: Mittlere Werte für λ (zwischen 0 und 1) funktionieren i.A. am besten, aber komplizierter Zusammenhang mit der Wahl der Schrittweite α.
- Wenn die Werte im Laufe des Verfahrens exakter werden, ist eine plausible Strategie, den Wert von λ gegen '0' gehen zu lassen.
- Der Konvergenzbeweis für $TD(\lambda)$ erlaubt variable $\lambda-$Werte, deshalb steht einem solchen Vorgehen nichts im Weg; allerdings ist bis heute auch kein mathematischer Beweis für den Vorteil eines solchen Verfahrens bekannt.

Zusammenfassung

- Lernen von Kosten durch Beispieltrajektorien mit fester Strategie
- Grundprinzip: Anwendung von Monte-Carlo-Methoden und stochastischer Approximation
- Variante: $TD(\lambda)$-Algorithmus;
  $\lambda = 1$ entspricht trajektorienbasierter Evaluierung
  $\lambda = 0$ entspricht Übergangsbasierter Evaluierung
- Online-/Offline-Version

Aus- und Überblick

Bisheriger Stand

- Approximiere V^{π} **ohne Modell** durch Durchlaufen vieler Trajektorien mit fester Strategie
- Beherrschung sehr grosser Zustandsräume durch Lernen auf “interessanten” Teilmengen

Aus- und Überblick

Bisheriger Stand

- Approximiere V^{π} **ohne Modell** durch Durchlaufen vieler Trajektorien mit fester Strategie
- Beherrschung sehr grosser Zustandsräume durch Lernen auf “interessanten” Teilmengen

Nächste Schritte

- Verbessere π durch gierige (“greedy”) Auswertung von $V\pi$ (Strategieiteration)
  - Problem: Hierzu braucht man immer noch ein Modell ...
- Repräsentation der Pfadkosten für kontinuerliche Zustände?