Vorlesung
Grundlagen adaptiver Wissenssysteme

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

Vorlesungseinheit 4

Das Bellman-Prinzip

Das Bellman-Prinzip

Lernziele

- Bellman'sches Optimalitätsprinzip kennen lernen
- einen ersten Algorithmus des optimierenden Lernens (konkret des dynamischen Programmierens) kennen lernen

Das Bellman-Prinzip

Überblick

1. Bellman’sches Optimalitätsprinzip

Das Bellman-Prinzip
Überblick

1. Bellman’sches Optimalitätsprinzip

2. Backward Dynamic Programming (Rückwärts-DP)

Das Bellman-Prinzip

Überblick

1. Bellman'sches Optimalitätsprinzip

2. Backward Dynamic Programming (Rückwärts-DP)

Lösung dynamischer Optimierungsprobleme

Zentrale Frage:

- Wie finde ich die Strategie, die (im Mittel) zu den minimalen Kosten führt?
- **Erinnerung:** Man kann das ganze analog auch als Maximierungsproblem (z.B. Maximierung des Gewinns) formulieren und dann von Belohnungen anstatt von Kosten sprechen.

Lösung dynamischer Optimierungsprobleme

Zentrale Frage:

- Wie finde ich die Strategie, die (im Mittel) zu den minimalen Kosten führt?
- **Erinnerung:** Man kann das ganze analog auch als Maximierungsproblem (z.B. Maximierung des Gewinns) formulieren und dann von Belohnungen anstatt von Kosten sprechen.

Lösungsmethodik: Dynamisches Programmieren (Bellman, 1957)

- Backward Dynamic Programming (BDP, Rückwärts-DP)
- Value Iteration (VI, Wertiteration)
- Policy Iteration (PI, Strategieiteration)

Backward Dynamic Programming

Problemkreis:

- stochastische Entscheidungsprobleme
- mehrstufige Entscheidungsprobleme
- Entscheidungsprobleme mit endlichem Horizont

Kernidee:

Backward Dynamic Programming

Problemkreis:

- stochastische Entscheidungsprobleme
- mehrstufige Entscheidungsprobleme
- Entscheidungsprobleme mit endlichem Horizont

Kernidee:

- Berechne die Kosten ausgehend von der letzten Stufe hin zur ersten Stufe.

Beispiel

Suche kürzesten Pfad in einem Graphen

Problemspezifikation

Gegeben:

- endlicher Horizont N
- MDP, d.h.

N diskrete Entsch.zeitpunkte	$t \in T = \{0, 1, \dots, N\}$
Zustandsmenge endlich	$s_t \in S = \{1, 2, \dots, n\}$
Aktionsmenge endlich	$a_t \in A = \{a_1, \dots, a_m\}$
Übergangswkt. $p_{ij}(a)$	$P(s_{t+1} = j \mid s_t = i, a_t = a) = p_{ij}(a)$
direkte Kosten	$c : S \times A \to \mathbb{R}$

- in der letzten Stufe N verursacht jeder Zustand Endkosten $g(s_N) := c_N(s_N)$

Zielsetzung

Gesucht: Eine Strategie π^* mit

$$V^{\pi^*} = \min_{\pi} V_N^{\pi},$$

Zielsetzung

Gesucht: Eine Strategie π^* mit

$$V^{\pi^*} = \min_\pi V_N^\pi,$$

für die gilt $V_N^\pi(i) = \mathbb{E}[g(s_N) + \sum_{t=0}^{N-1} c(s_t, \pi_t(s_t)) | s_0 = i]$

Definition (Optimale Pfadkosten)

Die zu π^* gehörigen Kosten werden als optimale Pfadkosten $V^* := V^{\pi^*}$ bezeichnet.

Zielsetzung

Gesucht: Eine Strategie π^* mit

$$V^{\pi^*} = \min_{\pi} V_N^{\pi},$$

für die gilt $V_N^{\pi}(i) = \mathbb{E}[g(s_N) + \sum_{t=0}^{N-1} c(s_t, \pi_t(s_t))|s_0 = i]$

Definition (Optimale Pfadkosten)

Die zu π^* gehörigen Kosten werden als optimale Pfadkosten $V^* := V^{\pi^*}$ bezeichnet.

Vorgehensweise:

1. Berechnung der optimalen Pfadkosten ("Cost-to-Go") $V_k^*(\cdot)$ für alle Zustände ($V_k^*(\cdot)$ ist ein n-dimensionaler Vektor). k ist die Anzahl der verbleibenden Schritte, n die Anzahl der Zustände.
2. Aus V_k^* ergibt sich die optimale Strategie für das k-Schritt-Problem. (k Schritte bis Prozess terminiert).

Motivation

Behauptung – Bellman'sches Optimalitätsprinzip:

Wenn ich noch k Schritte zu gehen habe, sind die optimalen Kosten für einen Zustand i gegeben durch den minimalen Erwartungswert der Summe aus

Motivation

Behauptung – Bellman'sches Optimalitätsprinzip:

Wenn ich noch k Schritte zu gehen habe, sind die optimalen Kosten für einen Zustand i gegeben durch den minimalen Erwartungswert der Summe aus

- *direkten Übergangskosten*
- **plus**

Motivation

Behauptung – Bellman'sches Optimalitätsprinzip:

Wenn ich noch k Schritte zu gehen habe, sind die optimalen Kosten für einen Zustand i gegeben durch den minimalen Erwartungswert der Summe aus

- *direkten Übergangskosten*
- **plus** *den optimalen Pfadkosten des Folgezustands, wenn von dort aus noch $k-1$ Schritte gemacht werden*

Die Minimierung geht dabei über alle zur Verfügung stehenden Aktionen.

Bellman’sches Optimalitätsprinzip (1)

Principle of Optimality

Bemerkungen:

- Das Bellman’sche Optimalitätsprinzip gilt aufgrund der Markov-Eigenschaft des Prozesses.
- Dieses Prinzip führt uns zu einem einfachen Algorithmus aus dem Bereich des dynamischen Programmierens: den Rückwärts-DP-Algorithmus (Backward Dynamic Programming).

Bellman’sches Optimalitätsprinzip (1)

Principle of Optimality

Bemerkungen:

- Das Bellman’sche Optimalitätsprinzip gilt aufgrund der Markov-Eigenschaft des Prozesses.
- Dieses Prinzip führt uns zu einem einfachen Algorithmus aus dem Bereich des dynamischen Programmierens: den Rückwärts-DP-Algorithmus (Backward Dynamic Programming).
- Bellman’sche Optimalitätsgleichungen gibt es nicht nur beim optimierenden Lernen, sondern sie bilden ein generelles Konzept.
- Für viele Berechnungsprobleme aus der Informatik kann man Bellman-Gleichungen aufstellen, die das Problem in geeigneter Weise zerlegen.

Bellman'sches Optimalitätsprinzip (2)

Principle of Optimality

Formalisierung

- Für die optimalen Pfadkosten des k-stufigen Entscheidungsproblems, $V_k^*(i)$, gilt:

$$V_k^*(i) = \min_{a \in A(i)} \mathbb{E}_{w_k}\{c(i,a) + V_{k-1}^*(f(i,a,w_k))\}$$

Bellman'sches Optimalitätsprinzip (2)

Principle of Optimality

Formalisierung

- Für die optimalen Pfadkosten des k-stufigen Entscheidungsproblems, $V_k^*(i)$, gilt:

$$
\begin{aligned}
V_k^*(i) &= \min_{a \in A(i)} \mathbb{E}_{w_k} \{ c(i,a) + V_{k-1}^*(f(i,a,w_k)) \} \\
&= \min_{a \in A(i)} \sum_{j=1}^{n} \{ p_{ij}(a)(c(i,a) + V_{k-1}^*(j)) \} \qquad i = 1 \dots n
\end{aligned}
$$

Damit können die optimalen Kosten des $N-$stufigen Optimierungsproblems rekursiv beginnend mit $k = 0$ berechnet werden

$\Rightarrow$ **Backward-DP Algorithmus**

Bellman'sches Optimalitätsprinzip (3)

Beweisskizze

Wesentliche Elemente:

- Strategie $\hat{\pi}^{(k)}$ für k-Stufen:

$$\hat{\pi}^{(k)} = (\pi_k, \pi_{k-1}, \pi_{k-2}, \ldots) = (\pi_k, \hat{\pi}^{(k-1)})$$

- Kosten für k Übergänge:
  Summe aus Übergangskosten und Kosten für $k-1$ Übergänge:

$$V_k^{\hat{\pi}^{(k)}}(i) = \sum_{j=1}^{n} p_{ij}(\pi_k(i)) \left(c(i, \pi_k(i)) + V_{k-1}^{\hat{\pi}^{(k-1)}}(j) \right)$$

Bellman'sches Optimalitätsprinzip (4)

Beweisskizze

Bellman'sches Optimalitätsprinzip (4)

Beweisskizze

Damit: $V_k^*(i) \quad = \quad \min_{\hat{\pi}^{(k)}} V_k^{\hat{\pi}^{(k)}}(i)$

Bellman'sches Optimalitätsprinzip (4)

Beweisskizze

$$\text{Damit: } V_k^*(i) = \min_{\hat{\pi}^{(k)}} V_k^{\hat{\pi}^{(k)}}(i)$$

$$= \min_{a \in A(i), \hat{\pi}^{(k-1)}} \sum_{j=1}^{n} p_{ij}(a) \left(c(i, a) + V_{k-1}^{\hat{\pi}^{(k-1)}}(j) \right)$$

Bellman'sches Optimalitätsprinzip (4)

Beweisskizze

$$
\begin{aligned}
\text{Damit: } V_k^*(i) &= \min_{\hat{\pi}^{(k)}} V_k^{\hat{\pi}^{(k)}}(i) \\
&= \min_{a \in A(i), \hat{\pi}^{(k-1)}} \sum_{j=1}^{n} p_{ij}(a) \left(c(i,a) + V_{k-1}^{\hat{\pi}^{(k-1)}}(j) \right) \\
&= \min_{a \in A(i)} \sum_{j=1}^{n} p_{ij}(a) \left(c(i,a) + \min_{\hat{\pi}^{(k-1)}} V_{k-1}^{\hat{\pi}^{(k-1)}}(j) \right)
\end{aligned}
$$

Bellman'sches Optimalitätsprinzip (4)

Beweisskizze

$$\text{Damit: } V_k^*(i) = \min_{\hat{\pi}^{(k)}} V_k^{\hat{\pi}^{(k)}}(i)$$

$$= \min_{a \in A(i), \hat{\pi}^{(k-1)}} \sum_{j=1}^{n} p_{ij}(a) \left(c(i,a) + V_{k-1}^{\hat{\pi}^{(k-1)}}(j) \right)$$

$$= \min_{a \in A(i)} \sum_{j=1}^{n} p_{ij}(a) \left(c(i,a) + \min_{\hat{\pi}^{(k-1)}} V_{k-1}^{\hat{\pi}^{(k-1)}}(j) \right)$$

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

Das Bellman-Prinzip

Überblick

1. Bellman'sches Optimalitätsprinzip

2. Backward Dynamic Programming (Rückwärts-DP)

Backward Dynamic Programming

Algorithmus

Backward-DP-Algorithmus

- $k = 0$:

$$V_0^*(i) = g(i)$$

Backward Dynamic Programming

Algorithmus

Backward-DP-Algorithmus

- $k = 0$:

$$V_0^*(i) = g(i)$$

- FOR $k = 1$ TO N
  FORALL $i \in S$

$$V_k^*(i) = \min_{a \in A(i)} \mathbb{E}_{w_k} \{ c(i,a) + V_{k-1}^*(f(i,a,w_k)) \}$$

bzw.

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

Backward Dynamic Programming

Aktionswahl

Annahme:

- $V_k^*(i)$ ist für alle $k \leq N$ bekannt.
- D.h. ist effektiv berechnet worden und liegt vor.

Frage: Wie wird auf Basis von V^* nun eine Aktion ausgewählt?

Backward Dynamic Programming

Aktionswahl

Annahme:

- $V_k^*(i)$ ist für alle $k \leq N$ bekannt.
- D.h. ist effektiv berechnet worden und liegt vor.

Frage: Wie wird auf Basis von V^* nun eine Aktion ausgewählt?

- Man berechnet für alle möglichen Aktionen die erwarteten Kosten und wählt die beste Aktion (die mit minimalen erwarteten Pfadkosten) aus.

$$\pi_k^*(i) \in \arg\min_{a \in A(i)} \sum_{j=1}^{n} p_{ij}(a) \left(c(i,a) + V_{k-1}^*(j) \right)$$

$\Rightarrow$ die ausgewählte optimale Aktion minimiert die Summe der erwartenden Übergangskosten zuzüglich der für das Restproblem erwarteten Pfadkosten.

Backward Dynamic Programming

Bemerkungen

Eindeutigkeit:

- Mit V_k^* ist eine optimale Strategie definiert.
- Die Strategie ist nicht eindeutig (Warum?), V_k^* aber schon.

Aufwandsbetrachtungen:

- für deterministische Systeme Aufwand $O(N * n)$
- für stochastische Systeme Aufwand $O(N * n^2)$
- geschlossene Lösungen selten berechenbar $\Rightarrow$ numerische Lösung
  - Aber: Sehr aufwändig!

Beispielaufgaben

Aufgabenblatt 1

- Problemmodellierung mit MDPs
- Probleme mit endlichem Horizont
- Anwendung des Backward-DP-Algorithmus

$\Rightarrow$ Separater Foliensatz!