

Moderne Methoden der KI: Maschinelles Lernen

Prof. Dr. sc. Hans-Dieter Burkhard
Vorlesung Sommer-Semester 2008

Verstärkungs-Lernen
(Reinforcement Learning)

Literatur:
R.S.Sutton, A.G.Barto
Reinforcement Learning
MIT Press 1998

Optimales Verhalten des Agenten

Steuerstrategie lernen:

- Aktionen abhängig von aktuellem Umwelt-Zustand
- Umwelt-Zustand verändert durch Aktionen
- Umwelt-Zustand wird bewertet („rewards“)
- Umwelt nur partiell erfahrbar
- Erfolgsbewertung verzögert

Umwelt i.a. nicht explizit gegeben
(aber bei dynamischer Programmierung/Optimierung)

Explorationsstrategie: Experimentieren mit möglichen Aktionen

Modelle für Interaktion Umwelt/Agent (Wdh.)

Dynamische Beschreibung in diskreter Welt

- Zusammenhänge zwischen Zuständen und Ereignissen
- Wie verändert sich die Welt durch Ereignisse

Agent beeinflusst die Welt durch Aktionen

- Strategie π legt die Aktionen fest
(z.B. abhängig vom aktuellen Zustand/Ziel)

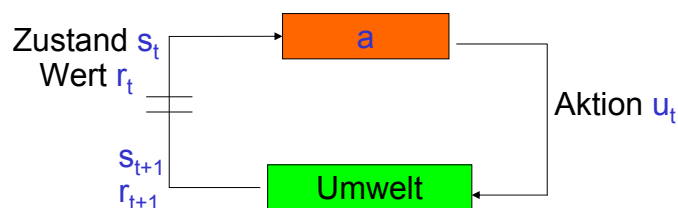
Modelle für Interaktion Umwelt/Agent

Ablaufbeschreibung mit Ereignissen e_t für diskrete lineare Zeit t

$$s_0 \rightarrow e_0 \rightarrow r_1 s_1 \rightarrow e_1 \rightarrow r_2 s_2 \rightarrow e_2 \rightarrow \dots$$

oder mit Aktionen u_t eines Agenten a

$$s_0 \rightarrow u_0 \rightarrow r_1 s_1 \rightarrow u_1 \rightarrow r_2 s_2 \rightarrow u_2 \rightarrow \dots$$



Markov-Bedingung

Abfolge

$$s_0 - u_0 \rightarrow r_1 s_1 - u_1 \rightarrow r_2 s_2 - u_2 \rightarrow \dots u_{t-1} \rightarrow r_t s_t - u_t \rightarrow$$

i.a. Zustand/Wert $r_{t+1} s_{t+1}$ zur Zeit $t+1$

abhängig von gesamter Vergangenheit:

$$P(s_{t+1} = s', r_{t+1} = r \mid s_0 u_0 r_1 s_1 u_1 r_2 s_2 u_2 \dots u_{t-1} r_t s_t u_t)$$

Markov-Bedingung: $= P(s_{t+1} = s', r_{t+1} = r \mid s_t u_t)$

d.h. nur aus letztem Zustand/letzter Aktion berechenbar

Markov-Bedingung

Dynamik des Systems für **stochastisch** beschriebene Welt mit Markov-Bedingung :

$F: S \times U \rightarrow \text{Verteilungen über } S \times \mathcal{R}$

$$P(s' = s_{t+1}, r = r_{t+1} \mid s_t u_t = su) = F(s, u)(s', r)$$

daraus abgeleitet

Übergangswahrscheinlichkeit $\mathcal{P}_{s u s'} := P(s' = s_{t+1} \mid s_t u_t = su)$

Reward-Wahrscheinlichkeit $\mathcal{P}_{s u r} := P(r = r_{t+1} \mid s_t u_t s_{t+1} = s u s')$

Erwartungswert für Reward $r_{s u s'} := E(r_{t+1} \mid s_t u_t s_{t+1} = s u s')$

Werte (rewards), analog: Kosten

Für einen Agenten zur Zeit t : r_t bzw. $r_{\text{sus}'}$

Kumulativer Wert

für feste Folge $s_0 -u_0 \rightarrow r_1 s_1 -u_1 \rightarrow r_2 s_2 -u_2 \rightarrow \dots$

Wert ab Zeit t : $R_t := \sum_{i=0 \dots \infty} \gamma^i r_{t+1+i}$
mit Discount-Faktor $0 < \gamma \leq 1$

In stochastischer Welt:

Erwartungswert für R_t bei gegebener Strategie π

H.D.Burkhard
Sommer-Semester 2008

MMKI: Reinforcement Learning

7

Markov-Entscheidungsprozess

Markov Decision Process (MDP) ist gegeben durch

- Diskrete lineare Zeitskala $t=0,1,2,\dots$
- (Endliche) Menge S von Zuständen s ,
Agent kann s vollständig und zuverlässig beobachten
- (Endliche) Menge U von Aktionen u des Agenten
- Dynamik beschrieben mittels
 - Übergangswahrscheinlichkeiten $P_{\text{sus}'}$
 - Erwartungswerten $r_{\text{sus}'} \in \mathcal{R}$

Der Agent muss sich zu jedem Zeitpunkt t für eine Aktion entscheiden.

H.D.Burkhard
Sommer-Semester 2008

MMKI: Reinforcement Learning

8

Strategie (policy)

Deterministische MDP mit deterministischer Strategie:

$$\pi: S \rightarrow U$$

Agent entscheidet sich für die Aktion $u=\pi(s)$ in Abhängigkeit vom aktuellen Zustand s

Damit ergibt sich die Abfolge

$$s_0 \xrightarrow{-u_0} r_1 s_1 \xrightarrow{-u_1} r_2 s_2 \xrightarrow{-u_2} \dots \xrightarrow{-u_{t-1}} r_t s_t \xrightarrow{-u_t} \dots$$

auf der Basis von Reward-/Übergangsfunktionen

$$u_t = \pi(s_t)$$

$$r_{t+1} = f_r(s_t, u_t)$$

$$s_{t+1} = f_s(s_t, u_t)$$

H.D.Burkhard
Sommer-Semester 2008

MMKI: Reinforcement Learning

9

Wertfunktion (value function) V

In deterministischem MDP ist bei gegebener Strategie π für jeden Zustand s der Wert berechenbar:

$$V^\pi: S \rightarrow \mathcal{R}$$

$V^\pi(s)$ ist der Wert, der bei Start in s und Arbeit gemäß π insgesamt erreicht wird:

$$V^\pi(s_t) := R_t = \sum_{i=0 \dots \infty} \gamma^i r_{t+1+i}$$

H.D.Burkhard
Sommer-Semester 2008

MMKI: Reinforcement Learning

10

Strategie (policy)

Für stochastische MDP:

$$\pi: \mathcal{S} \times \mathcal{U} \rightarrow [0,1]$$

Agent entscheidet sich mit Wahrscheinlichkeit $\pi(s,u)$ für die Aktion u in Abhängigkeit vom aktuellen Zustand s

Daraus ergeben sich Wahrscheinlichkeiten für die Abfolgen

$$s_0 \xrightarrow{-u_0} r_1 s_1 \xrightarrow{-u_1} r_2 s_2 \xrightarrow{-u_2} \dots \xrightarrow{-u_{t-1}} r_t s_t \xrightarrow{-u_t}$$

auf der Basis von

$$\pi(s_i, u_i), \mathcal{P}_{s u s'}, \mathcal{P}_{s u r}$$

Wertfunktion (state-value function) V

Bei gegebener Strategie π ist für jeden Zustand s der Erwartungswert bestimmt:

$$V^\pi: \mathcal{S} \rightarrow \mathcal{R}$$

$V^\pi(s)$ ist der Erwartungswert, der bei Start in s und Arbeit gemäß π insgesamt erreicht wird:

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

$$\begin{matrix} r_t(s) \\ \mathcal{P}_{s u s'} \\ R_t(s) \\ V^\pi(s) \end{matrix}$$

Bellmann-Gleichung

$$\begin{aligned}
 V^\pi(s) &:= E(R_t | s = s_t, \pi) \\
 &= E\left(\sum_{i=0}^{\infty} \gamma^i r_{t+1+i} \mid s = s_t, \pi\right) \\
 &= E\left(r_{t+1} + \gamma \sum_{i=0}^{\infty} \gamma^i r_{t+2+i} \mid s = s_t, \pi\right) \\
 &= \sum_u \pi(s, u) \sum_{s'} \mathcal{P}_{s, u}(s') (r_{s, u} + \gamma E(\sum_{i=0}^{\infty} \gamma^i r_{t+2+i} \mid s' = s_{t+1}, \pi)) \\
 &= \sum_u \pi(s, u) \sum_{s'} \mathcal{P}_{s, u}(s') (r_{s, u} + \gamma V^\pi(s'))
 \end{aligned}$$

Iterative Beziehungen zwischen den Werten $V^\pi(s)$

Bei $n = \text{card}(S)$ gibt es n lineare Gleichungen für n Werte.

Nach bekannten Verfahren lösbar.

Optimale Strategie π^*

π besser oder gleichwertig bzgl. π' ,
falls $V^\pi(s) \geq V^{\pi'}(s)$ für alle $s \in S$

Es gibt maximale Strategien (muss nicht eindeutig sein).

Maximale Strategie: π^*

Wertfunktion für maximale Strategie: V^*

$$V^*(s) := \max_{\pi} V^\pi(s)$$

Optimale Strategie π^*

Bellmann-Gleichung

für Wertfunktion V^* der optimalen Strategie π^*

$$V^*(s) = \max_u \sum_{s'} \mathcal{P}_{sus'} (r_{sus'} + \gamma V^*(s'))$$

$n := \text{card}(S)$

n nicht-lineare Gleichungen für n Werte $V^*(s)$

Lösung mit Methoden der dynamischen Programmierung

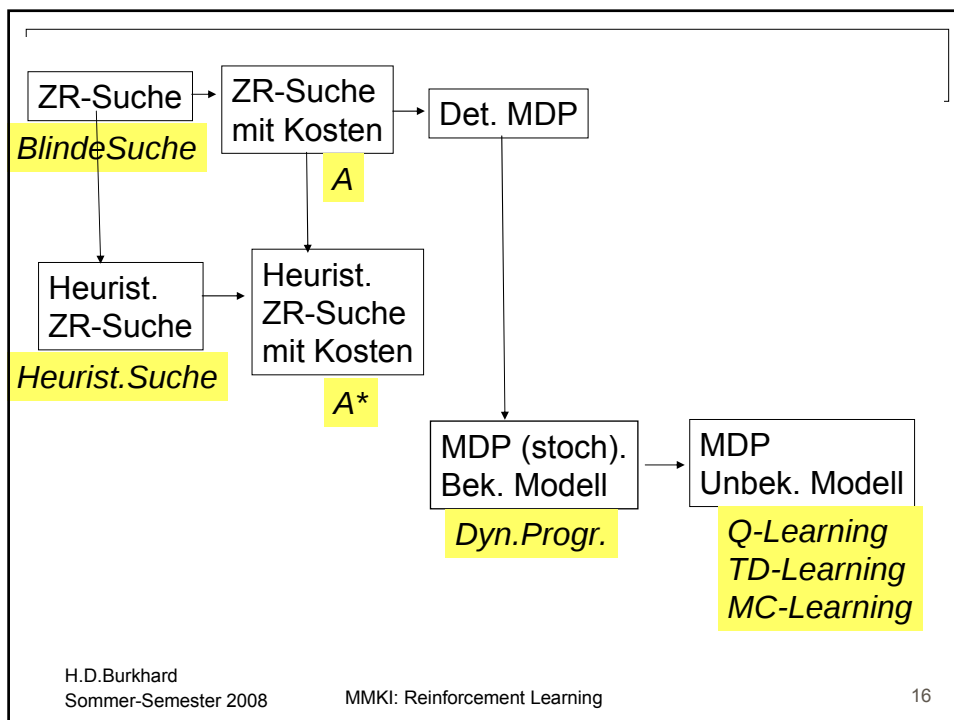
Voraussetzung: Modell ist bekannt

Spezialfälle: Suchverfahren (vgl. EKI)

H.D.Burkhard
Sommer-Semester 2008

MMKI: Reinforcement Learning

15



H.D.Burkhard
Sommer-Semester 2008

MMKI: Reinforcement Learning

16

Q-Funktion (action-value function)

$$Q^\pi: S \times U \rightarrow \mathcal{R}$$

$Q^\pi(s,u)$ ist der Erwartungswert der rewards,
der insgesamt erreicht wird,
bei Start in s mit Aktion u und Arbeit gemäß π :

$$Q^\pi(s, u) := E(R_t | s=s_t, u=u_t, \pi) = E\left(\sum_{i=0}^{\infty} \gamma^i r_{t+1+i} | s=s_t, u=u_t, \pi\right)$$

Bellmann-Gleichung für Q-Funktion

$$\begin{aligned} Q^\pi(s,u) &:= E(R_t | s = s_t, u=u_t, \pi) \\ &= E\left(\sum_{i=0}^{\infty} \gamma^i r_{t+1+i} | s = s_t, u=u_t, \pi\right) \\ &= E\left(r_{t+1} + \gamma \sum_{i=0}^{\infty} \gamma^i r_{t+2+i} | s = s_t, u=u_t, \pi\right) \\ &= \sum_{s'} \mathcal{P}_{s s'} (r_{s s'} + \gamma \sum_{u'} \pi(s', u') E(\sum_{i=0}^{\infty} \gamma^i r_{t+2+i} | s'=s_{t+1}, u'=u_{t+1}, \pi)) \\ &= \sum_{s'} \mathcal{P}_{s s'} (r_{s s'} + \gamma \sum_{u'} \pi(s', u') Q^\pi(s', u')) \end{aligned}$$

Iterative Beziehungen zwischen den Werten $Q^\pi(s,u)$

Bei $n = \text{card}(S)$ und $m = \text{card}(U)$ gibt es $n \cdot m$ lineare
Gleichungen für $n \cdot m$ Werte.

Nach bekannten Verfahren lösbar (vgl. Wertfunktion V).

Zusammenhänge zwischen Q und V

$$V^\pi(s) := \sum_u \pi(s,u) Q^\pi(s,u)$$

$$Q^\pi(s,u) := \sum_{s'} \mathcal{P}_{s u s'} (r_{s u s'} + \gamma V^\pi(s'))$$

Optimale Strategie π^*

Q-Funktion für maximale Strategie π^* : $Q^*(s,u) := \max_\pi Q^\pi(s,u)$

Bellmann-Gleichung für Q^* :

$$Q^{\pi^*}(s,u) = \sum_{s'} \mathcal{P}_{s u s'} (r_{s u s'} + \gamma \max_{u'} Q^{\pi^*}(s',u'))$$

$n := \text{card}(S)$, $m = \text{card}(U)$:

$n \cdot m$ nicht-lineare Gleichungen für $n \cdot m$ Werte $Q^*(s,u)$

Lösung mit Methoden der dynamischen Programmierung

Voraussetzung: Modell ist bekannt

Betrachtete Funktionen (Zusammenfassung)

MDP mit

Übergangswahrscheinlichkeit $\mathcal{P}_{\text{sus}'} := P(s' = s_{t+1} \mid s_t u_t = \text{sus})$

Reward-Wahrscheinlichkeit $\mathcal{P}_{\text{sus}'r} := P(r = r_{t+1} \mid s_t u_t s_{t+1} = \text{sus}')$

Erwartungswert für Reward $r_{\text{sus}'} := E(r_{t+1} \mid s_t u_t s_{t+1} = \text{sus}')$

Strategie $\pi: S \rightarrow U$ bzw. stochastisch $\pi: S \times U \rightarrow [0,1]$

Wert-Funktion (state-value-function) $V^\pi: S \rightarrow \mathcal{R}$ für Strategie π

Q-Funktion (action-value function) $Q^\pi: S \times U \rightarrow \mathcal{R}$ für Strategie π

Optimale Funktionen (Zusammenfassung)

Optimale Strategie $\pi^*: S \rightarrow U$

dazu:

Wert-Funktion (state-value-function) $V^*(s) = \max_{\pi} V^\pi(s)$

Q-Funktion (action-value function) $Q^*(s,u) = \max_{\pi} Q^\pi(s,u)$

Bestimmung von π^* anhand von bekanntem V^* bzw. Q^* :

$\pi^*(s) = \max_u Q^*(s,u)$ Anwendbar ohne Kenntnis des Modells

$= \max_u (r_{\text{sus}'} + \gamma V^*(s'))$ Modell: Übergänge, Rewards

Kenntnis des Modells notwendig

Statt π^* kann auch V^* bzw. Q^* bestimmt (gelernt) werden

Dynamische Programmierung

Voraussetzung: Modell bekannt, d.h.

Übergangswahrscheinlichkeit $\mathcal{P}_{s'us'} := P(s' = s_{t+1} \mid s_t u_t = su)$

Reward-Wahrscheinlichkeit $\mathcal{P}_{sus'r} := P(r = r_{t+1} \mid s_t u_t s_{t+1} = sus')$

Erwartungswert für Reward $r_{sus'} := E(r_{t+1} \mid s_t u_t s_{t+1} = sus')$

Aufgaben:

- Für gegebene Strategie π die Wertfunktion V^π berechnen
- Optimale Strategie π^* bestimmen (alternativ V^* bzw. Q^*)

Strategie-Bewertung: V^π berechnen

Aufgabe:

Gegeben π , berechne V^π in bekanntem MDP
(Vorhersage-Problem)

1. Variante:

Lösen der Bellmann-Gleichung (Lineares System):

$$V^\pi(s) = \sum_u \pi(s, u) \sum_{s'} \mathcal{P}_{sus'} (r_{sus'} + \gamma V^\pi(s'))$$

Iterative Strategie-Bewertung

2. Variante: Iterative Strategie-Bewertung

Iteration anhand der Bellmann-Gleichung:

$$V_0(s) := 0$$

$$V_{k+1}(s) := \sum_u \pi(s,u) \sum_{s'} \mathcal{P}_{s|s,u} (r_{s|s,u} + \gamma V_k(s'))$$

Konvergenz gegen V^π

(Fixpunkt, Bellmann-Gleichung, Voraussetzung: V^π existiert)

bei "full backup": alle Nachfolge-Zustände berücksichtigen

Iterative Strategie-Bewertung

Implementation:

- a) Zwei Tabellen für V_k und V_{k+1} ,
Einträge für V_{k+1} anhand der Werte für V_k
(entspricht simultaner Berechnung)
- b) Nur eine Tabelle
überschreiben der Werte
konvergiert schneller
(Resultat für V_{k+1} abhängig von Reihenfolge)

Iteration abbrechen,

wenn Abstand zwischen V_k und V_{k+1} hinreichend gering

Strategie-Verbesserung

Zunächst deterministisch:

Greedy-Verfahren

Ausgehend von Strategie π wird neue Strategie π' definiert:

$$\pi'(s) := \operatorname{argmax}_u Q^\pi(s,u) \quad (\text{falls } u \text{ nicht eindeutig: w\u00e4hlen})$$

Dann gilt f\u00fcr alle s : $V^{\pi'}(s) \geq V^\pi(s)$

und $V^{\pi'} = V^\pi \rightarrow V^{\pi'} = V^*$

F\u00fcr stochastische Strategien analog.

Strategie-Iteration

Sukzessive neue Strategien π^{k+1} aus Strategie π^k definieren:

$$\pi^{k+1}(s) := \operatorname{argmax}_u Q^{\pi^k}(s,u)$$

Dann gilt f\u00fcr alle s und alle k : $V^{\pi_{k+1}}(s) \geq V^{\pi_k}(s)$

$$V^{\pi_0} \leq V^{\pi_1} \leq V^{\pi_2} \leq V^{\pi_3} \leq \dots \rightarrow V^*$$

Konvergenz nach endlich vielen Schritten f\u00fcr endliche MDP.

Strategie-Iteration

Implementation:

Abwechselnd Strategie-Bewertung und Strategie-Verbesserung ausführen (Start beliebig, Abbruch bei Gleichheit bzw. geringer Verbesserung):

$$\pi^0 \rightarrow_{\text{Bew.}} V^{\pi_0} \rightarrow_{\text{Verb.}} \pi^1 \rightarrow_{\text{Bew.}} V^{\pi_1} \rightarrow_{\text{Verb.}} \pi^2 \rightarrow_{\text{Bew.}} V^{\pi_2} \dots \rightarrow_{\text{Verb.}} \pi^* \rightarrow_{\text{Bew.}} V^*$$

Problem:

Strategie-Bewertung durch Iteration ist unendlicher Prozess.
Man muss aber nicht vollständig bewerten (endl. viele k).

Wert-Iteration

Kombinierte Strategie-Bewertung und Strategie-Verbesserung, wobei nur Wertfunktion V explizit iteriert wird:

$$\begin{aligned} V_{k+1}(s) &:= \max_u E(r_{t+1} + \gamma V_k(s_{t+1}) \mid s_t = s, u_t = u) \\ &= \max_u \sum_{s'} \mathcal{P}_{sus'} (r_{sus'} + \gamma V_k(s')) \end{aligned}$$

Aus Ergebnis V dann Strategie π bestimmen.
(Abbruch wenn nur noch geringe Verbesserung)

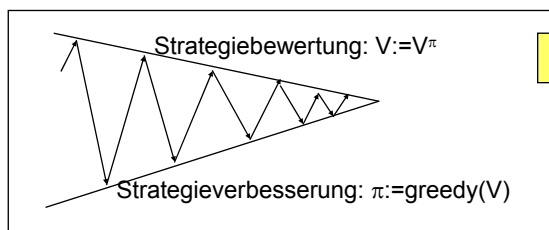
Verallgemeinerte Strategie-Iteration (GPI)

Interagierende Prozesse:

- Strategie-Bewertung
- Strategie-Verbesserung

Varianten dabei bzgl. Ausführlichkeit der jeweiligen Prozesse.

Sukzessive Annäherung an optimale Strategie.



Ende bei $\pi = \text{greedy}(V^\pi)$

H.D.Burkhard
Sommer-Semester 2008

MMKI: Reinforcement Learning

31

Lernen ohne Modell

Dynamische Programmierung/Optimierung:

optimale Strategie ermitteln unter Verwendung des Modells:

π^* bzw. V^* oder Q^*

Jetzt: Exploration

Modell unbekannt, aber durch Aktionen erkunden

Voraussetzung: Umwelt (Zustand, Reward) beobachtbar

H.D.Burkhard
Sommer-Semester 2008

MMKI: Reinforcement Learning

32

Iteration ohne Modell

Grundidee:

Mit Aktionen Modell (partiell) erkunden und
mittels Iterationen Funktion bestimmen

Bei stochastischer Umwelt: Wiederholte Erkundung

Entspricht GPI-Ansatz:

- Aktionsauswahl nach jeweils verbesserter Strategie
- (Verkürzte) Bewertung der Strategie

Beispiel

Q-Learning für unbekanntes **deterministisches** MDP

$$f: S \times U \rightarrow S \times \mathcal{R}$$
$$[s_{t+1}, r_{t+1}] = f(s_t, u_t)$$

kann man aufspalten in :

Übergangsfunktion $f_s: S \times U \rightarrow S$

Rewardfunktion $f_r: S \times U \rightarrow \mathcal{R}$

$$s_{t+1} = f_s(s_t, u_t)$$

$$r_{t+1} = f_r(s_t, u_t)$$

Beispiel

Lernziel: optimale Aktions-Wert-Funktion Q^*

daraus ableitbar:

$$\pi^*(s) = \text{ArgMax}_u Q^*(s,u) ,$$

$$V^*(s) = \text{Max}_u Q^*(s,u)$$

Bellmann-Optimalitäts-Gleichung (Iteration):

$$Q^*(s,u) = f_r(f_s(s,u)) + \gamma \text{Max}_{u'} Q^*(f_s(s,u), u')$$

ermöglicht iteratives Lernen

Q-Learning-Algorithmus (det. Modell)

Folge von Hypothesen Q_k :

$$Q_{k+1}(s,u) = f_r(f_s(s,u)) + \gamma \text{Max}_{u'} Q_k(f_s(s,u), u')$$

Beschreibung z.B. in Tabellenform

update durch Auswertung von Explorationen:

$$s' = f_s(s, u) , f_r(s')$$

Lernen in aufeinander folgenden Episoden.

Q-Learning-Algorithmus (det. Modell)

Initial: Q_0 : beliebig

Repeat for ever (für jede Episode)

- Q_i : Werte aus voriger Episode
- Anfangs-Zustand s bestimmen

Mit s iterieren:

- Aktion u **bestimmen** und ausführen .
- $s' = f_s(s, u)$, $r = f_r(s, u)$ beobachten
- Aktualisieren: $Q_{i+1}(s, u) := r + \gamma \max_{u'} Q_i(s', u')$
- $s := s'$

Q-Learning: Konvergenz (det. Modell)

Konvergenz von Q_k gegen Q^*

gilt auch in unendlichen Modellen, falls

- $0 \leq \gamma < 1$
- $f_r(s, u)$ beschränkt durch Konstante c
- beliebige endliche initiale Werte
- alle Paare (s, u) werden unendlich oft betrachtet.

(Beweis)

Bei nicht-negativen rewards und Initialisierung mit 0

gilt stets $0 \leq Q_k(s, u) \leq Q(s, u)$

Q-Learning: Aktionsauswahl

Generalized Strategy Iteration (GPI):

Aktionsauswahl nach jeweils verbesserter Strategie
(Verkürzte) Bewertung des erreichbaren Q -Werts

Auswahl-Strategie (nächste Aktion u):

Möglichkeit: „gierige (greedy) Auswahl“

$$u := \operatorname{argmax}_{u'} (f_r(s, u') + \gamma Q_k(s', u'))$$

Damit bestes bekanntes Resultat erzielt.

Problem: evtl. bessere Zustände werden nicht probiert.

Explorationstrategien

Problem (Exploration vs. Exploitation) :

Auswahl zwischen bekannten „guten“ Aktionen und
Experiment mit anderen Aktionen

(Konvergenzbedingung: alle (s, u) unendlich oft)

Auswahl-Wahrscheinlichkeiten (stochastische Strategie!)

$$\text{z.B. } \pi(s, u) := \frac{c^{Q_k(s, u)}}{\sum_v c^{Q_k(s, v)}} \\ \text{für eine Konstante } c$$

Explorationstrategien

Weitere Varianten

Spezielle (günstige) Folgen verwenden
(ausgehend von den Zustandsübergängen mit rewards)

Simulation:

partiell: über bereits bekannte Folgen bei neuen Q_k -Werten

total: bei bekannten f_s und f_r (dynamische Programmierung)

Q-Learning für stochastische Modelle

Einfaches update (Überschreiben) nicht sinnvoll
(keine Konvergenz)

Stattdessen: Inkrementelle Annäherung an Erwartungswert

$$\text{neu} := \text{alt} + \text{Schrittweite} \cdot (\text{Messwert} - \text{alt})$$

Beispiel: n-armiger Bandit

Beispiel: „n-armiger Bandit“

Ohne Zustand, n Aktionen

$Q^*(u)$: Unbekannter (Erwartungs-)Wert für Aktion u

$Q_t(u)$: geschätzter Wert für u zur Zeit t

z.B. Mittelwert über bisher erhaltene rewards:

$$Q_t(u) = 1/t \sum_{i=1 \dots t} r_i \quad (\text{konvergiert gegen } Q^*(u))$$

Auswahl einer Aktion zur Zeit t

Exploitation: nächste Aktion gemäß $\operatorname{argmax}_u Q_t(u)$

Exploration: andere Aktion

(momentan schlechter, aber falls noch weitere Aktionen folgen sollen: evtl. bessere Alternativen finden)

n-armiger Bandit

greedy-Auswahl: immer Exploitation

ϵ -greedy : mit Wkt. $\epsilon > 0$ andere Aktion

Softmax: $P(u) = e^{Q_t(u)/\tau} / \sum_v e^{Q_t(v)/\tau}$

$\tau > 0$ „Temperatur“ (Boltzmann-Verteilung)

Optimistischer Anfangswert

führt dazu, dass zunächst exploriert wird

n-armiger Bandit

Inkrementale Berechnung:

$$\begin{aligned}Q_{t+1}(u) &= (t+1)^{-1} \sum_{i=1 \dots t+1} r_i \\&= (t+1)^{-1} (r_{t+1} + \sum_{i=1 \dots t} r_i) \\&= (t+1)^{-1} (r_{t+1} + t Q_t(u) + Q_t(u) - Q_t(u)) \\&= (t+1)^{-1} (r_{t+1} + (t+1) Q_t(u) - Q_t(u)) \\&= Q_t(u) + (t+1)^{-1} (r_{t+1} - Q_t(u))\end{aligned}$$

Allgemeine Form:

$$\begin{aligned}\text{neu} &:= \text{alt} + \text{Schrittweite} \cdot (\text{Messwert} - \text{alt}) \\&= \text{alt} + \text{Schrittweite} \cdot \text{Fehler}\end{aligned}$$

n-armiger Bandit

$$Q_{t+1}(u) = Q_t(u) + (t+1)^{-1} (r_{t+1} - Q_t(u))$$

$$\text{Allgemeiner } Q_{t+1}(u) = Q_t(u) + \alpha_t (r_{t+1} - Q_t(u))$$

Schrittweite α_t variieren für bessere Konvergenz,

Konvergenz mit Wkt. 1 falls $\sum_{i=0 \dots \infty} \alpha_i = \infty$ und $\sum_{i=0 \dots \infty} \alpha_i^2 < \infty$

Q-Learning für stochastisches Modell

Initial: Q_0 : beliebig

Repeat for ever (für jede Episode)

- Q_i : Werte aus voriger Episode
- Anfangs-Zustand s bestimmen

Mit s iterieren:

- Aktion u **wählen** (z.B. ϵ -greedy) und ausführen .
- s' , r beobachten
- Aktualisieren:

$$Q_{i+1}(s,u) := Q_i(s,u) + \alpha (r + \gamma \max_{u'} Q_i(s', u') - Q_i(s,u))$$

- $s := s'$

Dabei ist $r + \gamma \max_{u'} Q_i(s', u')$ der neue Messwert
Temporal difference: $r + \gamma \max_{u'} Q_i(s', u') - Q_i(s,u)$

TD-Lernen

TD = Temporal difference

(entspricht der Abweichung vom bisher angenommen Wert)

(Später mehr dazu)

Auch für andere Probleme anwendbar

Allgemeine Formel: $\text{update} = \text{Lernrate} * \text{Fehler}$

(vgl. delta-Regel für NN, Gradientenverfahren usw.)

Haben bis jetzt für stochastische Modelle:

	Modell bekannt (DP)	Modell unbekannt (RL)
V^π	• Lin. Glgssystem • Iteration	
Q^π		
π^*	Strategie-Iteration	
V^*	Iteration	
Q^*		Q-Lernen

Für deterministische Modelle auch einfachere Iterationsverfahren

Monte Carlo (MC) Lernen

- Modell nicht bekannt
- Lernen in Episoden (!)
- Mittlung der interessierenden Werte aus Episoden

Prinzip GPI für Steuerung des Agenten

MC: Strategiebewertung für π

Initial: Liste $\text{Returns}(s_i, u_i)$ leer, V beliebig

Repeat for ever

- Episode (endl. Zustandsfolge $s_0, s_1, s_2, \dots, s_n$ mit rewards) gemäß π generieren und beobachten
- Für alle s_i den dabei erreichten Wert R_i ermitteln und an Liste $\text{Returns}(s_i)$ anhängen
- $V(s_i) := \text{Mittelwert}(\text{Returns}(s_i))$

MC: Strategiebewertung (Policy evaluation) für geg. π

Varianten:

- jedes Auftreten eines Zustands s berücksichtigen
- nur das erste Auftreten („first visit“)
- Evtl. nicht alle Zustände in den Episoden berücksichtigen (dadurch attraktiv)
- Konvergenz gemäß Gesetz der großen Zahlen (unendliche viele Besuche der Zustände)

MC: Strategiebewertung für Q^π -Funktion

Analog zu Strategie-Bewertung.

Problem:

Bei deterministischer Strategie π wird im Zustand s stets nur Aktion $u=\pi(s)$ ausgeführt, d.h. keine vollständige Exploration.

Explorationsproblem (vgl. n-armiger Bandit)

Möglichkeit:

Mit jedem Zustands-Aktions-Paar (s,u) unendlich oft anfangen

Optimale Strategie

In unbekanntem Modell kann optimale Strategie π^* aus Q^* bestimmt werden:

$$\pi^*(s) = \max_u Q^*(s,u)$$

(nicht möglich aus V^*)

MC: Optimale Q-Funktion lernen

Initial: $\text{Returns}(s_i, u_i)$ leer, Q beliebig, π beliebig (stoch.)

Repeat for ever

- Episode (endl. Zustandsfolge $s_0, s_1, s_2, \dots, s_n$ mit rewards) gemäß aktueller Strategie π generieren und beobachten
- Für alle (s_i, a_i) den dabei erreichten Wert R_i ermitteln und an Liste $\text{Returns}(s_i, u_i)$ anhängen
- $Q(s_i, u_i) := \text{Mittelwert}(\text{Returns}(s_i, u_i))$
- Verbesserte Strategie π z.B. gemäß (ϵ -greedy)
 $\pi(s, u) := 1 - \epsilon + \epsilon / \text{card}(U)$ für $u := \arg\max_u Q(s, u)$
☐ $\pi(s, u) := \epsilon / \text{card}(U)$ sonst

folgt dem GPI-Schema

H.D.Burkhard
Sommer-Semester 2008

MMKI: Reinforcement Learning

55

MC: Optimale Q-Funktion lernen

folgt dem GPI-Schema:

Es erfolgen abwechselnd Schritte

- zur Strategieverbesserung (neue Strategie π)
- Strategiebewertung (Q^π), aber nicht vollständig
- ϵ -greedy statt Start mit unterschiedlichen s_0, u
- Approximation der optimalen Q-Funktion bzw. Strategie.
- Zusätzliche Analyse bzgl. Stabilität des Prozesses für deterministische Strategie (Abbruch bei Stabilität: Optimale Strategie gefunden)

H.D.Burkhard
Sommer-Semester 2008

MMKI: Reinforcement Learning

56

MC: Off-Policy

On-Policy:

Wertfunktion/Q-Funktion für Strategie π wird gelernt durch Episoden, die mit π erzeugt werden

Alternative: Off-Policy:

Wertfunktion/Q-Funktion für Strategie π wird gelernt durch Episoden, die mit einer anderen Strategie π' erzeugt werden

MC: Off-Policy

Vorteile:

Strategie π kann greedy (deterministisch) sein
Strategie π' erzeugt die notwendigen Beispiele

Dafür notwendig:

1. $\pi(s,u) > 0 \rightarrow \pi'(s,u) > 0$
2. Ausgleich der unterschiedlichen Wahrscheinlichkeiten für die durchlaufenen Folgen bei π bzw. π'

MC: inkrementelles Vorgehen

$V(s_i) := \text{Mittelwert}(\text{Returns}(s_i))$ inkrementell berechnen

Verallgemeinerung: gewichteter Mittelwert (für off-policy)

$$V_n = \sum_{i=1 \dots n} w_i R_i / \sum_{i=1 \dots n} w_i$$

Inkrementell:

$$V_{n+1} = V_n + w_{n+1} / W_{n+1} (R_{n+1} - V_n)$$

mit $W_{n+1} = W_n + w_{n+1}$ und $V_0 = W_0 = 0$

$$\begin{aligned} & V_n + w_{n+1} / W_{n+1} (R_{n+1} - V_n) \\ &= 1 / W_{n+1} (W_{n+1} V_n + w_{n+1} R_{n+1} - w_{n+1} V_n) \\ &= 1 / W_{n+1} ((W_{n+1} - w_{n+1}) V_n + w_{n+1} R_{n+1}) \\ &= 1 / W_{n+1} (W_n V_n + w_{n+1} R_{n+1}) \\ &= 1 / W_{n+1} (\sum_{i=1 \dots n} w_i R_i + w_{n+1} R_{n+1}) \\ &= 1 / W_{n+1} (\sum_{i=1 \dots n+1} w_i R_i) = V_{n+1} \end{aligned}$$

H.D.Burkhard

Sommer-Semester 2008

MMKI: Reinforcement Learning

59

Temporal Difference Learning (TD)

Allgemeine Idee:

- Differenz zwischen letzter Schätzung und aktuellem Messwert zur Korrektur verwenden: Richtung der Änderung
- Schrittweite der Änderung durch Wichtungsfaktor α gegeben mit der Zeit abnehmend, z.B.

$$\alpha := 1 / (1 + \text{Anzahl der bisherigen Aktionen u in s})$$

H.D.Burkhard

Sommer-Semester 2008

MMKI: Reinforcement Learning

60

TD(0) für Strategiebewertung V^π

Initial: V beliebig

Für jede Episode:

- Startzustand s_0 bestimmen
 V : Werte aus vorheriger Episode
- Wiederhole für jeden Zustand s der Episode bis zum Ende
Aktion $u := \pi(s)$ ausführen
Folgezustand s' und reward r beobachten
 $V(s) := V(s) + \alpha (r + \gamma V(s') - V(s))$
 $s := s'$

Konvergenz mit Wkt. 1 bei unendlich vielen Besuchen

H.D.Burkhard
Sommer-Semester 2008

MMKI: Reinforcement Learning

61

Vergleich MC und TD

Update jeweils für besuchten Zustand (analog Zustand/Aktion):

Nach Besuch von s_t erfolgt update von $V(s_t)$

- MC: mittels R_t vom Ende der Episode (gemittelt bei mehrmaligem Auftreten)
- TD: mittels r_{t+1} und $V(s_{t+1})$
(„Bootstrapping-Verfahren“)

H.D.Burkhard
Sommer-Semester 2008

MMKI: Reinforcement Learning

62

Vergleich MC, DP und TD

Bellmann-Glg.:

$$V^\pi(s) = E(R_t | s = s_t, \pi)$$

Update für MC

$$= E\left(\sum_{i=0}^{\infty} \gamma^i r_{t+1+i} | s = s_t, \pi\right)$$

$$= E\left(r_{t+1} + \gamma \sum_{i=0}^{\infty} \gamma^i r_{t+2+i} | s = s_t, \pi\right)$$

$$= E\left(r_{t+1} + \gamma V^\pi(s_{t+1}) | s = s_t, \pi\right)$$

Update für DP

Tatsächlich werden jeweils nur Approximationen verwendet:

MC: Beispielwert für R_t (gemittelt über bisherigen Episoden)

DP: bisherige Schätzung V_i^π für V^π (bootstrapping)

TD: Beispielwert für r_{t+1} und bisherige Schätzung V_i^π

Q-Learning (Wdh.)

Initial: Q : beliebig

Repeat for ever (für jede Episode)

- Q : Werte aus voriger Episode
- Anfangs-Zustand s bestimmen

Mit s iterieren:

- Aktion u wählen (z.B. ϵ -greedy) und ausführen.
- s' , r beobachten
- Aktualisieren (mit bereits neueren Q -Werten):
 $Q(s,u) := Q(s,u) + \alpha (r + \gamma \max_{u'} Q(s',u') - Q(s,u))$
- $s := s'$

Off-policy

Sarsa (anderes GPI-Schema)

Initial: Q beliebig

Repeat for ever (für jede Episode)

- Anfangs-Zustand s und
- erste Aktion u (z.B. ϵ -greedy gemäß Q) bestimmen

Wiederhole für jeden Zustand s der Episode

- u ausführen, s' , r beobachten
- Aktion u' für s' wählen (z.B. ϵ -greedy gemäß Q)
- Aktualisieren:

$$Q(s,u) := Q(s,u) + \alpha (r + \gamma Q(s',u') - Q(s,u))$$

$$s := s', u := u'$$

statt bei off-policy:

$$r + \gamma \text{Max}_{u'} Q(s',u') - Q(s,u)$$

TD(λ): Vorwärtsbetrachtung

$$s_0 \rightarrow u_0 \rightarrow r_1 s_1 \rightarrow u_1 \rightarrow r_2 s_2 \rightarrow u_2 \rightarrow \dots$$

Wert ab Zeit t : $R_t := \sum_{i=0.. \infty} \gamma^i r_{t+1+i}$

Approximation gemäß 1-Schritt-Vorausschau: $r_{t+1} + \gamma V_t(s_{t+1})$

Alternativ: Approximation mit Vorausschau von n Schritten ab s_t
n-Schritt>Returns $R_t^{(n)}$

$$n=1: R_t^{(1)} := r_{t+1} + \gamma V(s_{t+1})$$

$$n=2: R_t^{(2)} := r_{t+1} + \gamma r_{t+2} + \gamma^2 V(s_{t+2})$$

...

$$n: R_t^{(n)} := r_{t+1} + \gamma r_{t+2} + \dots + \gamma^{n-1} r_{t+n-1} + \gamma^n V(s_{t+n})$$

USW.

TD(λ): Vorwärtsbetrachtung

Gleichzeitige Benutzung aller Vorausschaumöglichkeiten
bei Kombination mit Wichtungsfaktor λ (recency factor) :
($0 \leq \lambda \leq 1$, Gesamtgewicht = 1)

$$\begin{aligned} R_t^\lambda &:= (1-\lambda) (R_t^{(1)} + \lambda R_t^{(2)} + \lambda^2 R_t^{(3)} + \dots) \\ &= (1-\lambda) \sum_{i=1, \dots} \lambda^{i-1} R_t^{(i)} \end{aligned}$$

TD(λ): Vorwärtsbetrachtung

Bei einer Episode der Länge T wird abschließender Return R_t
erreicht bei : $R_t^{(T-t+1)} = R_t$
Danach weiter für $i > 0$: $R_t^{(T-t+1+i)} = R_t$

$$\begin{aligned} R_t^\lambda &= (1-\lambda) \sum_{i=1, \dots, T-t-1} \lambda^{i-1} R_t^{(i)} + (1-\lambda) \sum_{i=T-t, \dots, T} \lambda^{i-1} R_t \\ &= (1-\lambda) \sum_{i=1, \dots, T-t-1} \lambda^{i-1} R_t^{(i)} + \lambda^{T-t} R_t \end{aligned}$$

Falls $\lambda = 1$: $R_t^\lambda = R_t$ entspricht MC-Lernen

Falls $\lambda = 0$: $R_t^\lambda = R_t^{(1)}$ entspricht bisherigem TD-Lernen
(daher Bezeichnung TD(0))

Insgesamt bei $0 < \lambda < 1$: Kombination von beidem

TD(λ): Rückwärtsbetrachtung

Nach jeder Aktion erfolgt update der bereits früher besuchten Zustände.

Dabei Wichtung der Zustände entsprechend der Häufigkeit der Besuche („eligibility trace“ = Spur) mit den Parametern γ, λ

Für jeden Zeitpunkt t : $e_t: S \rightarrow \mathcal{R}$

$$e_t(s) := \gamma \lambda e_{t-1}(s) \quad \text{für } s \neq s_t$$

$$e_t(s) := \gamma \lambda e_{t-1}(s) + 1 \quad \text{für } s = s_t$$

TD(λ): Strategiebewertung V^π

Initial: $V(s)$ beliebig, $e(s)=0$ für alle $s \in S$

Repeat (für jede Episode)

Initial: s wählen

Repeat (für jeden Schritt der Episode)

Aktion $u=\pi(s)$ ausführen,

reward r und Folgezustand s' beobachten

Korrekturwert $\delta := r + \gamma V(s') - V(s)$

e-Wert für s erhöhen: $e(s) := e(s) + 1$

α ist Lernrate

Für alle $s \in S$ neue Schätzung: $V(s) := V(s) + \alpha \cdot \delta \cdot e(s)$

Alle e-Werte anpassen: $e(s) := \gamma \cdot \lambda \cdot e(s)$

$s := s'$

TD(λ)

TD(1) d.h. $\lambda = 1$:

entspricht MC-Lernen, aber inkrementelle Realisierung

Variationen von λ

- Zeitlich
- Zustandsabhängig

$$e(s) := \gamma \cdot \lambda(s) \cdot e(s)$$

Bei guter Schätzung in s : $\lambda(s) \rightarrow 1$ (hoher Einfluss)

Bei schlechter Schätzung: $\lambda(s) \rightarrow 0$ (geringer Einfluss)

Sarsa(λ)

Initial: Q beliebig, $e(s,u)=0$ für alle s, u

Repeat (für jede Episode)

- s, u (z.B. ϵ -greedy gemäß Q) bestimmen

Repeat Aktion (für jeden Schritt der Episode)

- u ausführen .
- s', r beobachten
- u' für s' wählen (z.B. ϵ -greedy gemäß Q)
- Korrektur-Wert $\delta := r + \gamma Q(s', u') - Q(s, u)$
- e-Wert für s, u erhöhen: $e(s, u) := e(s, u) + 1$
- für alle s, u neue Schätzung: $Q(s, u) := Q(s, u) + \alpha \delta e(s, u)$
- Alle e-Werte anpassen: $e(s, u) := \gamma \cdot \lambda \cdot e(s, u)$
- $s := s', u := u'$

$Q(\lambda)$: Q-Learning erweitert

Off-Policy-Verfahren

Problem:

„Exploitation“-Aktionen (nicht greedy gewählt)
können Verfahren verfälschen

Eligibility-Trace entsprechend modifizieren:

Verfahren von Watkins

H.D.Burkhard
Sommer-Semester 2008

MMKI: Reinforcement Learning

73

$Q(\lambda)$: Verfahren von Watkins

Initial: Q beliebig, $e(s,u)=0$ für alle s, u

Repeat (für jede Episode)

- s, u (z.B. ϵ -greedy gemäß Q) bestimmen

Repeat (für jeden Schritt der Episode)

- Aktion u ausführen .
- s', r beobachten
- u' für s' wählen (z.B. ϵ -greedy gemäß Q)
- Greedy-Aktion bestimmen: $u^* := \operatorname{argmax}_v Q(s', v)$
- Korrektur-Wert $\delta := r + \gamma Q(s', u^*) - Q(s, u)$
- e-Wert für s, u erhöhen: $e(s, u) := e(s, u) + 1$
- für alle s, u neue Schätzung: $Q(s, u) := Q(s, u) + \alpha \delta e(s, u)$
- e-Werte anpassen: falls $u' = u^*$: $e(s, u) := \gamma \cdot \lambda \cdot e(s, u)$
sonst: $e(s, u) := 0$

$s := s', u := u'$

Für große/kontinuierliche Modelle

Tabelle für $V(s)$ bzw. $Q(s,a)$ ersetzen durch Approximation

- Funktionen
- Neuronale Netze, ...

Statt verbessern der Tabellenwerte:

Lernverfahren für die Approximationen

(Gradientenabstieg, BackPropagation, ...)

Trainingsbeispiele aus den beobachteten Werten

Konvergenz nicht gesichert

H.D.Burkhard
Sommer-Semester 2008

MMKI: Reinforcement Learning

75

Behandelte Probleme

	Mit Modell DP	Ohne Modell MC	Ohne Modell Temporal Difference	Ohne Modell Eligibility Traces
Bewertung (prediction) V^π	Lin.-Glg. Iteration	Episoden	TD(0)	TD(λ)
Bewertung Q^π				
Strategie- Iteration π^*	Iteration			
Wert-Iteration V^*	Iteration			
Q-Learning Q^* , Control (GPI)		Episoden	Q-Learning Sarsa	Watkins Sarsa(λ)

H.D.Burkhard
Sommer-Semester 2008

MMKI: Reinforcement Learning

76

Modellierung beim Verstärkungslernen

MDP festlegen:

- Beschreibungen für Zustände, Aktionen: S, U
- Übergangsfunktion: Tabelle, Funktion, Neuron. Netz, ...
- Rewardfunktion: Tabelle, Funktion, Neuron. Netz, ...

Lernverfahren:

- Episoden
- Akkumulation von rewards: Parameter γ
- on-policy vs. off-policy
- Exploration vs. Exploitation: ϵ -greedy, Parameter ϵ
- Schrittweite α für TD-Lernen
- Recency-Faktor λ für TD(λ)-Lernen

H.D.Burkhard
Sommer-Semester 2008

MMKI: Reinforcement Learning

77

Konvergenz-Aussagen

Notwendig:

Alle Zustands-Aktionspaare werden hinreichend oft besucht

- in endl. stochastischen MDP: unendlich oft
- in endl. Determinist. MDP: endlich Anzahl von Besuchen
- in unendl. MDP: Problem der Approximationsmodelle für Zustandsübergangs-Funktion/Reward-Funktion

Beispiel:

Konvergenz beim Q-Lernen gegen Q^* , falls

- $0 \leq \gamma < 1$
- $f_r(s,u)$ beschränkt durch Konstante c
- beliebige endliche initiale Werte
- alle Paare (s,u) werden unendlich oft betrachtet

H.D.Burkhard
Sommer-Semester 2008

MMKI: Reinforcement Learning

78