Reinforcement Learning

Volker Tresp

Überwachtes und unüberwachtes Lernen

- **Überwachtes Lernen:** Zielgrößen sind im Trainingsdatensatz bekannt; Ziel ist die Verallgemeinerung auf neue Daten
- **Unüberwachtes Lernen:** Erkenntnis über Beziehungen zwischen Variablen (Spalten) oder/und zwischen Objekten (Zeilen); Clusteranalyse, PCA, ...

Reinforcement Lernen

- **Reinforcement Lernen:** Ein Agent handelt in einer Umwelt und erhält als Rückkopplung Belohnungen und Bestrafungen; der Agent soll lernen, anhand der Interaktionen mit der Umwelt und der Belohnungen (bzw Bestrafungen) sein Verhalten zu optimieren
- Eltern belohnen ihr Kind, wenn es etwas gut gemacht hat; das Kind lernt, das Gutgemachte zu wiederholen (um weitere Belohnungen zu erhalten)
- Kleinkinder belohnen und bestrafen ihre Eltern durch ein Lächeln bzw durch unkontrolliertes Schreien; Eltern lernen ihr Verhalten zu optimieren, so dass Belohnungen maximiert werden und Bestrafungen minimiert werden; hier ist es offensichtlich, dass Kinder nur unspezifische Signale geben, und die Eltern trainiert werden, herauszufinden, welche Aktion optimal ist (Windel wechseln, füttern, schlafen legen, herumtragen ...)

Beispiele

- Spiele wie Schach oder Backgammon: nur eine Belohnung am Ende vieler Aktionen
- Frage der Kreditzuordnung: welche Aktionen waren richtig, welche Aktionen waren falsch?
- Finanzmarkt: welche Aktionen soll ich unternehmen, um meinen Gewinn zu maximieren?
- Admission control (Telekommunikation, Hotel, Autoverleih): soll ich jeden Gast aufnehmen und mein Hotel voll belegen, oder soll ich Zimmer für zahlungskräftige Kunden freihalten
- Regelung und Steuerung schlecht verstandener Systeme
- Technischer Reiz: man muss nicht aufwendig ein Prozessmodell erstellen; das System optimiert sich selber anhand von Belohnung und Bestrafung

Kommentare

- Im Vergleich zu den bisher behandelten Themen, zwei weitere wesentlich neue Aspekte:
- Zeit: Zustände sind zeitlich gekoppelt; Zeitreihenmodellierung, Zeitreihenprognose
- Kontrolle: das Lernende System beeinflusst den gegenwärtigen Zustand der Welt und ebenso zukünftige Zustände; Regelung und Steuerung

Wesentliche Bestandteile

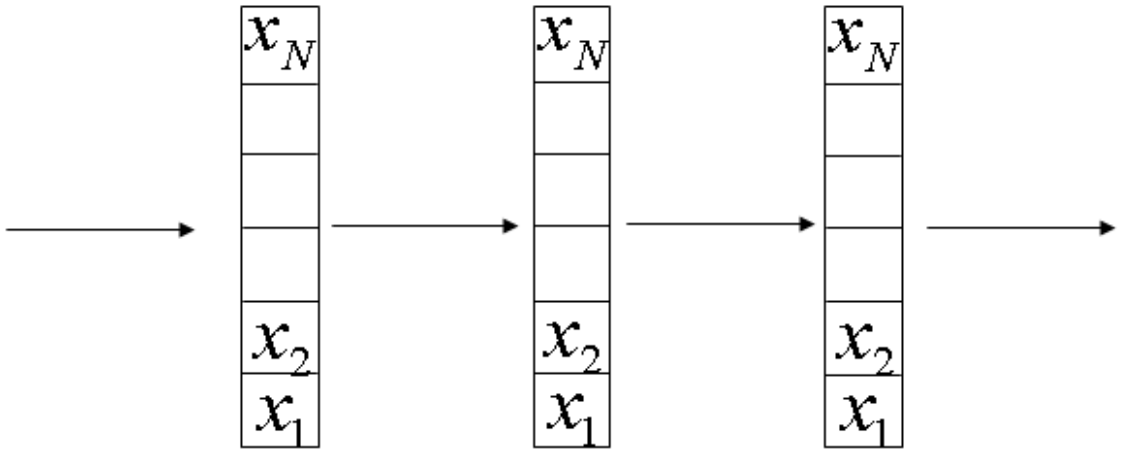
- $X_t = x_i$ bedeutet, dass der Zustand der Welt zum Zeitpunkt t gleich x_i ist. Man nimmt typischerweise an, dass es nur endlich viele Zustände gibt, die sich gegenseitig Ausschließen (z.B. Brettpositionen im Schach); nehmen wir an, dass es N Zustände gibt, d.h. $X_t \in \{x_1, \ldots x_N\}$.
- Markov Eigenschaft: Die Wahrscheinlichkeit eines Folgezustandes ist nur vom gegenwärtigen Zustand abhängig; ist dieser bekannt, ist die Zukunft unabhängig von der Vergangenheit
- Markov Decision Process (MDP)
- Agent: Kennt den Zustand der Welt $X_t = x_i$. Der Agent kann eine Aktion $A_t \in \{a_1, \ldots, a_M\}$ durchführen.

Kontrollgesetz und Belohnung

- Die Strategie des Agenten spiegelt sich wieder in einem Kontrollgesetz (policy)

$$A_t = \pi(X_t)$$

- (In gewissen Fällen ist das Kontrollgesetz stochastisch und $\pi(A_t|X_t)$ ist die Wahrscheinlichkeit, dass der Agent Aktion A_t wählt, wenn sich das System zum Zeitpunkt t sich im Zustand X_t befindet)
- Wird Aktion A_t durchgeführt und befindet sich das System zum Zeitpunkt $t + 1$ in X_{t+1}, so erhält der Agent die Belohnung (reward) $r(A_t, X_t, X_{t+1})$



$$P(X_{t+1} \mid X_t, \pi(X_t))$$

$$r(A_t, X_t, X_{t+1})$$

Spezialfall: $r(X_t)$

Discounted Return

- Es geht nun darum, die Güte verschiedener Kontrollgesetze zu bewerten
- Der "discounted return" ist der zukünftige gewichtete Gewinn zum Zeitpunkt t für eine gegebene Sequenz von Aktionen und Zuständen

$$R_t = \sum_{k=0}^{\infty} \gamma^k r(A_{t+k}, X_{t+k}, X_{t+1+k})$$

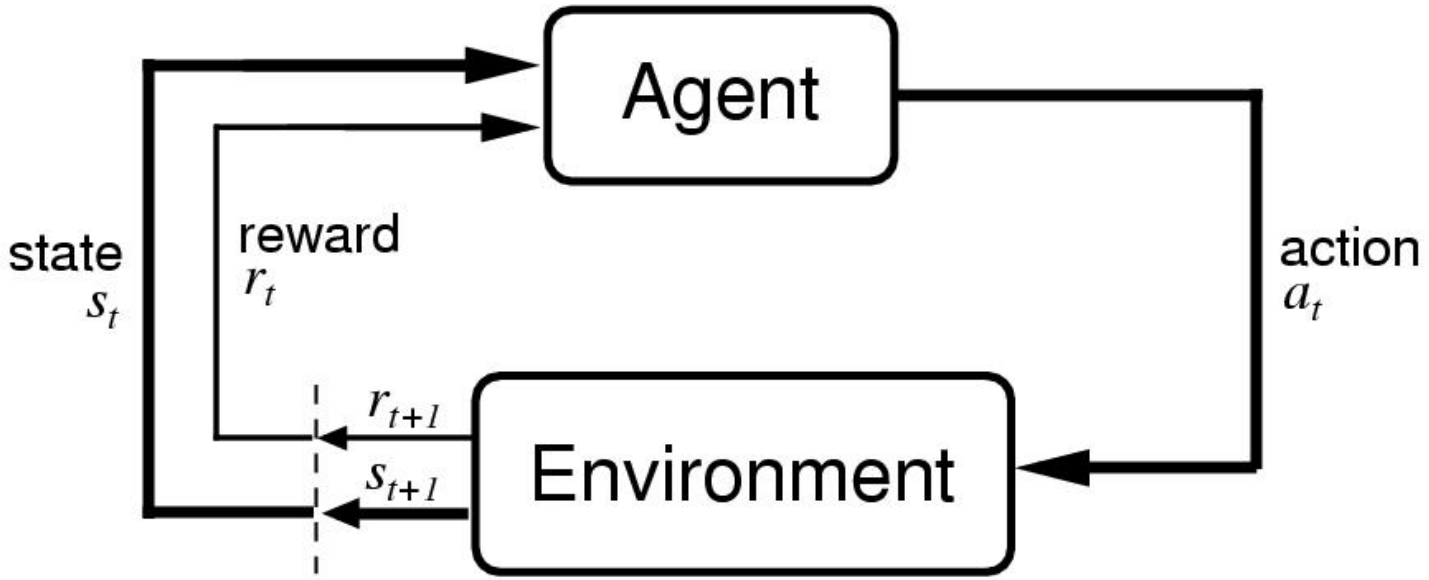
- $0 < \gamma^k < 1$ ist der Discountfaktor; Belohnungen weit in der Zukunft werden weniger stark gewichtet

Bewertungsfunktion

- Die Bewertungsfunktion (value function) ist definiert als der erwartete Gewinn, den ich erziele, wenn ich ein festes Kontrollgesetz π verwende und mich gegenwärtig im Zustand $X_t = x_i$ befinde

$$V^{\pi}(x_i) = E_{\pi}\{R_t | X_t = x_i\} = E_{\pi}\left\{ \sum_{k=0}^{\infty} \gamma^k r(A_{t+k}, X_{t+k}, X_{t+1+k}) \middle| X_t = x_i \right\}$$

- $1 \geq \gamma > 0$ ist der Discount Faktor
- Will ich verschiedene Kontrollgesetzte vergleichen benötige ich deren Bewertungsfunktionen. Die effiziente Berechnung der Bewertungsfunktionen ist Thema der folgenden Folien


Agent
state
s_t
reward
r_t
action
a_t
r_{t+1}
s_{t+1}
Environment

Temporal Difference (TD) Learning

- Man kann umformen

$$V^{\pi}(x_i) = E_{\pi}\{R_t | X_t = x_i\} = E_{\pi}\left\{ \sum_{k=0}^{\infty} \gamma^k r(A_{t+k}, X_{t+k}, X_{t+1+k}) \middle| X_t = x_i \right\}$$

$$= E_{\pi}\left\{r(A_t, X_t = x_i, X_{t+1})\right\} + E_{\pi}\left\{ \sum_{k=1}^{\infty} \gamma^k r(A_{t+k}, X_{t+k}, X_{t+1+k}) \middle| X_t = x_i \right\}$$

$$= E_{\pi}\left\{r(A_t, X_t = x_i, X_{t+1})\right\} + \gamma E_{\pi}\left\{V(X_{t+1}) \middle| X_t = x_i\right\}$$

- Diese Gleichung kann man als Zuordnung auffassen

TD(0)

- Alternativ kann man einen Schritt der Länge ρ in die Richtung des neuen Wertes gehen

$$V^{\pi}(x_i) = V^{\pi}(x_i) +$$

$$\rho \left[E_{\pi} \left\{ r(A_t, X_t = x_i, X_{t+1}) \right\} + \gamma E_{\pi} \left\{ V(X_{t+1}) \middle| X_t = x_i \right\} - V^{\pi}(x_i) \right]$$

- Ein Problem ist, dass man zur Berechnung der Erwartungsgwerte das Prozessmodell exakt kennen muss

- Die Idee ist nun, dass man das tatsächliche System beobachtet oder zumindest realistisch simuliert und die Erwartungswerte durch die tatsächlich beobachteten Übergänge approximiert

- Angenommen ich beobachte einen Übergang von x_i nach x_j unter der festen Policy π und erhalte dabei Belohnung r. Im TD(0) Lernen adaptiert man

$$V(x_i) \longleftarrow V(x_i) + \rho \left[r + \gamma V(x_j) - V(x_i) \right].$$

TD(λ)

- Der vorige Algorithmus heisst $TD(0)$ und berücksichtigt nur den neuen Zustand
- Man kann auch Zustände berücksichtigen, die vor Kurzem besucht wurden; man kann annehmen, dass auch deren Bewertungsfunktion korrigiert werden muss; dazu führt man eine *Eligibility-Trace* ein
- Die akkumulierte *Eligibility-Trace* ist definiert als (für alle Zustände!)

$$e_t(x) = 1 + \lambda e_{t-1}(x) \quad wenn \;\; x = x_t$$

und

$$\lambda e_{t-1}(x) \;\; sonst$$

TD(λ)

- Die *Eligibility-Trace* ist eine Art Kurzzeitgedächnis und man adaptiert für alle Zustände x

$$V(x) \longleftarrow V(x) + \rho \left[r + \gamma V(x_j) - V(x_i) \right] e_t(x).$$

- Ein neuer Parameter $0 \leq \lambda \leq 1$ wird eingeführt und der Algorithmus heisst $TD(\lambda)$
- Gute Performanz wird erreicht, wenn man mit $\lambda = 1$ started und zur Konvergenz λ gegen Null gehen lässt

TD(λ) mit Neuronalem Netz

- In vielen interessanten Anwendungen explodiert der Zustandsraum: gibt es M Dimensionen, die man in L Stufen pro dimension diskretisiert, so wird das System mit L^N Zuständen beschrieben
- Dies ist der Bellman'sche Fluch der Dimensionen (*curse of dimensionality*); dieser tritt in zweierlei Form auf: Speicherbedarf und Rechenzeit
- Der Durchbruch wurde dadurch erzielt, dass die Bewertungsfunktion durch ein Neuronales Netz approximiert wurde

$$V(x) \approx NN_w(x)$$

- Dadurch müssen nur noch die Parameter des Neuronalen Netzes adaptiert werden, und diese Zahl ist typischerweise im Bereich von einigen Hundert
- Es gab einen spektakulären Erfolg, nämlich die erfolgreiche Anwendung auf Backgammon

- Ein Nachteil ist allerdings, dass die Konvergenzbeweise, die für die explizite Zustandsrepräsentation (look-up table) existieren, sich auf Neuronale Netze nicht übertragen lassen

TD(λ) mit Neuronalem Netz (2)

- Der modifizierte Algorithmus lautet folgendermaßen
- Die Eligibility-Trace ist nun ein Vektor der Dimensionalität der Anzahl der Parameter des Neuronalen Netzes:

$$\mathbf{e}_t = \lambda \mathbf{e}_{t-1} + \frac{\partial}{\partial \mathbf{w}} NN_w(x_{t-1})$$

- Das Neuronale Netz wird adaptiert gemäß

$$\mathbf{w}_{new} = \mathbf{w}_{old} + \rho \left[r + \gamma NN_w(x_j) - NN_w(x_i) \right] \mathbf{e}_t$$

Von TD(λ) zum Kontrollgesetz

- Man startet mit einer Policy π und berechnet mit den vorgestellten Algorithmen die Bewertungsfunktion
- Während der Adaption der Bewertungsfunktion verbessert man die Policy; in einem gierigen Ansatz wählt man die Aktion, die das System in den Zustand bringt, welcher nach der (gegenwärtigen) Bewertungsfunktion optimal ist
- Im Algorithmus wird somit gleichzeitig die Policy optimiert als auch die Bewertungsfunktion der adaptierten Policy nachadaptiert (bootstrapping)

Varianten

- Es gibt auch Varianten, in denen mit einer gewissen Wahrscheinlichkeit auch suboptimale Aktionen erlaubt sind, damit das System seinen Zustandsraum sinnvoll erforscht (Exploration)
- Wenn die Aktion den Folgezustand nicht deterministisch bestimmt sondern auch hier Stochastik im Spiel ist, dann kann man das verwandte Q-Lernen (*Q-learning*) verwenden
- Reinforcement Lernen ist sehr verwandt zum Dynamischen Programmieren
- Die Folien zum TD-Gammon kann man finden unter http://www.inf.fu-berlin.de/lehre/SS04/19577-S/06_Back_Gammon.pdf

Bellman's Optimalitätsgleichung

- Value Iteration: Das optimale Kontrollgesetz $\pi = *$ und die dazugehörige Value Funktion $V^*(x)$ können gefunden werden durch Iteration der Gleichung

$$V^*(x) := \max_a \left\{ \sum_{s'} P(x'|x,a) \left(r(a,x,x') + \gamma V^*(x') \right) \right\}$$

- Nach Konvergenz wird aus der Zuordnung eine Gleichung: Bellman's Optimalitätsgleichung