

Reinforcement learning (II)

IA 2023/2024

Introducere

Învățarea pasivă

Bazată pe model: ADP

Fără model: TD-learning

Învățarea activă

Q-learning

Deep Q-learning

Concluzii

► Proces de decizie Markov

- Mulțimea de stări S , mulțimea de acțiuni A
- Modelul de tranziții $P(s'|s, a)$ este **cunoscut**
- Funcția de recompensă $R(s)$ este **cunoscută**
- **Calculează** o politică optimă

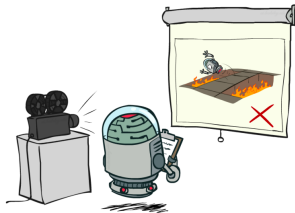
► Învățare cu întărire

- Se bazează pe procese de decizie Markov, dar:
- Modelul de tranziții este **necunoscut**
- Funcția de recompensă este **necunoscută**
- **Învață** o politică optimă

Tipuri de învățare cu întărire

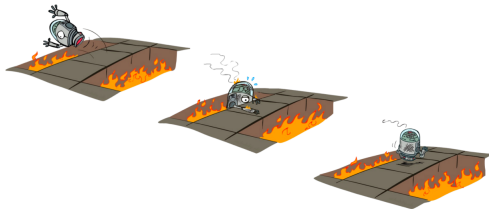
Pasivă/activă

- **Pasivă**: agentul execută o politică **fixă** și o **evaluează**



Agentul nu are control asupra acțiunilor sale; robotică

- **Activă**: agentul își actualizează politica pe măsură ce învață



Tipuri de învățare cu întărire

Bazată pe model/fără model

- ▶ **Bazată pe model:** învață modelul de tranziții și recompense și îl folosește pentru a descoperi politica optimă
- ▶ **Fără model:** descoperă politica optimă fără a învăța modelul

Introducere

Învățarea pasivă

Bazată pe model: ADP

Fără model: TD-learning

Învățarea activă

Q-learning

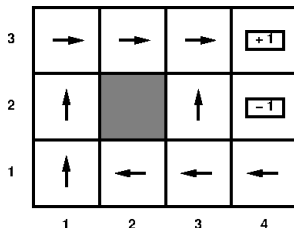
Deep Q-learning

Concluzii

Învățarea pasivă

- ▶ Politica este **fixă**: în starea s execută întotdeauna acțiunea $\pi(s)$
 - ▶ Scopul: învață cât de bună este politica π
 - ▶ învață utilitatea $U^\pi(s)$
- cum?
- execută politica și învață din experiență
- ▶ abordare similară cu pasul (1) de evaluare a politicii din cadrul algoritmului *Iterarea politicilor*; diferența: nu cunoaștem modelul de tranziții $P(s'|s, a)$ și nici $R(s)$

Învățarea pasivă este o modalitate de explorare a mediului.



- ▶ Agentul execută o serie de încercări (*trials*)
 $(1, 1)_{-.04} \rightsquigarrow (1, 2)_{-.04} \rightsquigarrow (1, 3)_{-.04} \rightsquigarrow (1, 2)_{-.04} \rightsquigarrow (1, 3)_{-.04} \rightsquigarrow$
 $(2, 3)_{-.04} \rightsquigarrow (3, 3)_{-.04} \rightsquigarrow (4, 3)_{+1}$
 $(1, 1)_{-.04} \rightsquigarrow (1, 2)_{-.04} \rightsquigarrow (1, 3)_{-.04} \rightsquigarrow (2, 3)_{-.04} \rightsquigarrow (3, 3)_{-.04} \rightsquigarrow$
 $(3, 2)_{-.04} \rightsquigarrow (3, 3)_{-.04} \rightsquigarrow (4, 3)_{+1}$
 $(1, 1)_{-.04} \rightsquigarrow (2, 1)_{-.04} \rightsquigarrow (3, 1)_{-.04} \rightsquigarrow (3, 2)_{-.04} \rightsquigarrow (4, 2)_{-1}$
- ▶ Politica este aceeași, dar mediul este nedeterminist
- ▶ Scopul este să învețe utilitatea așteptată $U^\pi(s) = E[\sum_{t=0}^{\infty} \gamma^t R(S_t)]$

Introducere

Învățarea pasivă

Bazată pe model: ADP

Fără model: TD-learning

Învățarea activă

Q-learning

Deep Q-learning

Concluzii

Programarea dinamică adaptivă (ADP)

1. Învățăm modelul de tranziții

Estimăm $P(s'|s, \pi(s))$ și $R(s)$ din încercări.

Utilizăm un tabel de probabilități (cât de des apare rezultatul unei acțiuni și estimăm probabilitatea de tranziție)

Exemplul 1: acțiunea *Right* este executată de 3 ori în starea (1,3) și în 2 cazuri starea rezultantă este (2,3)

$$\implies P((2,3)|(1,3), \textit{Right}) = 2/3$$

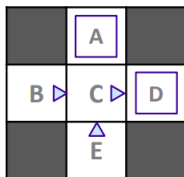
2. Rezolvăm MDP

Programarea dinamică adaptivă (ADP)

1. Învățarea modelului empiric

Exemplu:

Input Policy π



Assume: $\gamma = 1$

Observed (s, a, s', R) Transitions

Episode 1

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 2

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 3

E, north, C, -1
C, east, D, -1
D, exit, x, +10

Episode 4

E, north, C, -1
C, east, A, -1
A, exit, x, -10

Learned Model

$\hat{T}(s, a, s')$

$T(B, \text{east}, C) = 1.00$
 $T(C, \text{east}, D) = 0.75$
 $T(C, \text{east}, A) = 0.25$
...

$\hat{R}(s, a, s')$

$R(B, \text{east}, C) = -1$
 $R(C, \text{east}, D) = -1$
 $R(D, \text{exit}, x) = +10$
...

Programarea dinamică adaptivă

2. Utilizăm programarea dinamică pentru rezolvarea procesului de decizie Markov.

Probabilitățile și recompensele învățate se introduc în ecuațiile Bellman (politica fixă).

$$U^\pi(s) = R(s) + \gamma \sum_{s'} P(s'|s, \pi(s)) U^\pi(s')$$

Se rezolvă sistemul de ecuații liniare cu necunoscutele $U^\pi(s)$.

ADP este inefficientă dacă spațiul stărilor este mare

- ▶ sistem de ecuații liniare de ordin n
- ▶ jocul de table: 10^{50} ecuații cu 10^{50} necunoscute

Introducere

Învățarea pasivă

Bazată pe model: ADP

Fără model: TD-learning

Învățarea activă

Q-learning

Deep Q-learning

Concluzii

1. Estimarea directă a utilității

- ▶ Utilitatea unei stări este recompensa totală așteptată de la acea stare înainte (**reward-to-go**)

Exemplu: $(1, 1)_{-.04} \rightsquigarrow (1, 2)_{-.04} \rightsquigarrow (1, 3)_{-.04} \rightsquigarrow (1, 2)_{-.04} \rightsquigarrow (1, 3)_{-.04} \rightsquigarrow (2, 3)_{-.04} \rightsquigarrow (3, 3)_{-.04} \rightsquigarrow (4, 3)_{+1}$

$(1, 1)_{-.04} \rightsquigarrow (1, 2)_{-.04} \rightsquigarrow (1, 3)_{-.04} \rightsquigarrow (2, 3)_{-.04} \rightsquigarrow (3, 3)_{-.04} \rightsquigarrow (3, 2)_{-.04} \rightsquigarrow (3, 3)_{-.04} \rightsquigarrow (4, 3)_{+1}$

$(1, 1)_{-.04} \rightsquigarrow (2, 1)_{-.04} \rightsquigarrow (3, 1)_{-.04} \rightsquigarrow (3, 2)_{-.04} \rightsquigarrow (4, 2)_{-1}$

Prima încercare produce:

- ▶ în starea (1,1) recompensa totală 0.72 ($1 - .04 \times 7$)
 - ▶ în starea (1,2) două recompense totale 0.76 și 0.84
 - ▶ în starea (1,3) două recompense totale 0.80 și 0.88
-
- ▶ Utilitatea estimată: media valorilor eșantionate
 - ▶ $U(1,1) = 0.72$, $U(1,2) = 0.80$, $U(1,3) = 0.84$ etc.

Estimarea directă a utilității

- ▶ Presupune că utilitățile sunt independente (fals)
Nu ține cont de faptul că utilitatea unei stări depinde de utilitățile stărilor succesoare (constrângerile date de ecuațiile Bellman)
 - ▶ căutarea într-un spațiu mult mai mare
 - ▶ convergența este foarte lentă
- ▶ Avem toate episoadele dinainte

2. Învățarea diferențelor temporale (*Temporal Differences*)

- ▶ Combină avantajele celor două abordări anterioare (*Estimarea directă a utilității* și *Programarea dinamică adaptivă*)
 - ▶ actualizează doar stările direct afectate
 - ▶ satisface aproximativ ecuațiile Bellman
- ▶ Scopul: estimarea utilităților $U^\pi(s)$, date episoadele generate utilizând politica π .
Acțiunile sunt decise de politica π .
- ▶ Utilitățile sunt ajustate **după fiecare tranziție** observată.

Exemplu:

- ▶ După prima încercare: estimările $U^\pi(1, 3) = 0.84$, $U^\pi(2, 3) = 0.92$.
- ▶ Fie tranziția $(1, 3) \rightarrow (2, 3)$ în a doua încercare.
Constrângerea dată de ecuația Bellman impune ca $U^\pi(1, 3) = -0.04 + U^\pi(2, 3) = 0.88$ ($\gamma = 1$).
- ▶ Estimarea inițială $U^\pi(1, 3) = 0.84$ este mai mică \rightarrow se actualizează

Învățarea diferențelor temporale

- **Ecuția diferențelor temporale** utilizează diferența utilităților între stări succesive:

$$U^\pi(s) \leftarrow U^\pi(s) + \alpha(R(s) + \gamma U^\pi(s') - U^\pi(s))$$

α rata de învățare

- Actualizarea implică doar succesorul s' , pe când condițiile de echilibru (ec. Bellman) implică toate stările următoare posibile
- Metoda aplică o serie de corecții pentru a converge
- Obs: metoda nu are nevoie de un model de tranziții P pentru a realiza actualizările

Diferențe temporale: pseudocod

function PASSIVE-TD-AGENT(*percept*) **returns** an action

inputs: *percept*, a percept indicating the current state s' and reward signal r'

persistent: π , a fixed policy

U , a table of utilities, initially empty

N_s , a table of frequencies for states, initially zero

s, a, r , the previous state, action, and reward, initially null

if s' is new **then** $U[s'] \leftarrow r'$

if s is not null **then**

 increment $N_s[s]$

$U[s] \leftarrow U[s] + \alpha(N_s[s])(r + \gamma U[s'] - U[s])$

if $s'.\text{TERMINAL?}$ **then** $s, a, r \leftarrow \text{null}$ **else** $s, a, r \leftarrow s', \pi[s'], r'$

return a

Demo: https://cs.stanford.edu/people/karpathy/reinforcejs/gridworld_td.html

Învățarea diferențelor temporale: exemplu

States

| | | |
|---|---|---|
| | A | |
| B | C | D |
| | E | |

Assume: $\gamma = 1$, $\alpha = 1/2$

Observed Transitions

B, east, C, -2

| | | |
|---|---|---|
| | 0 | |
| 0 | 0 | 8 |
| | 0 | |

C, east, D, -2

| | | |
|----|---|---|
| | 0 | |
| -1 | 0 | 8 |
| | 0 | |

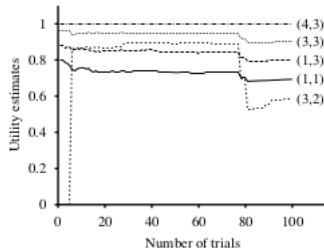
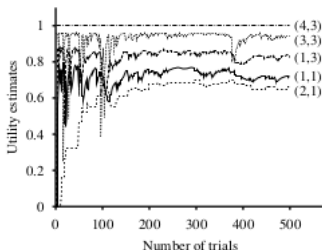
| | | |
|----|---|---|
| | 0 | |
| -1 | 3 | 8 |
| | 0 | |

Învățarea diferențelor temporale

- ▶ Rata de învățare α determină viteza de convergență la utilitatea reală
- ▶ Valoarea medie a $U^\pi(s)$ va converge la valoarea corectă
 - ▶ suficiente încercări, tranzițiile rare apar rar
 - ▶ dacă α este o funcție care scade pe măsură ce nr. de vizitări ale unei stări crește, atunci $U^\pi(s)$ converge la valoarea corectă
 - ▶ funcția $\alpha(n) = 1/n$ sau $\alpha(n) = 1/(1 + n) \in (0, 1]$

Diferențe temporale vs. Programare dinamică adaptivă

- ▶ TD nu are nevoie de model, ADP este bazată pe model
- ▶ TD utilizează doar succesul observat pentru actualizare și nu toți succesorii
- ▶ TD converge mai lent, dar execută calcule mai simple



- ▶ TD poate fi văzut ca o aproximare a ADP

Introducere

Învățarea pasivă

Bazată pe model: ADP

Fără model: TD-learning

Învățarea activă

Q-learning

Deep Q-learning

Concluzii

Învățarea pasivă vs. învățarea activă

- ▶ Agentul pasiv are o politică fixă vs. agentul activ trebuie să **decidă acțiunile**
- ▶ Agentul pasiv învață (probabilitățile tranzițiilor și) utilitățile stărilor și alege acțiunile optime

vs.

Agentul activ își **actualizează politica** pe măsură ce învață

- ▶ scopul este să învețe politica optimă
- ▶ însă, funcția utilitate nu este cunoscută decât aproximativ

Exploatare vs. explorare

Dilema exploatare-explorare a agentului

- ▶ să își maximizeze utilitatea, pe baza cunoștințelor curente, sau
- ▶ să își îmbunătățească cunoștințele

Este necesar un compromis între

- ▶ exploatare
 - ▶ agentul oprește învățarea și execută acțiunile date de politică
- ▶ explorare
 - ▶ agentul învață încercând acțiuni noi

Dilema exploatare - explorare: soluții

Metoda ϵ -greedy

- ▶ Fie $\epsilon \in [0, 1]$
- ▶ Acțiunea următoare selectată va fi:
 - ▶ o acțiune aleatoare, cu probabilitatea ϵ
 - ▶ acțiunea optimă, cu probabilitatea $1 - \epsilon$
- ▶ Implementare
 - ▶ inițial $\epsilon = 1$ (explorare)
 - ▶ când se termină un episod de învățare, ϵ scade (de ex. cu 0.05) - crește progresiv rata de exploatare
 - ▶ ϵ nu scade niciodată sub un prag, de ex. 0.1
 - ▶ agentul are mereu o șansă de explorare, pentru a evita optimele locale

Introducere

Învățarea pasivă

Bazată pe model: ADP

Fără model: TD-learning

Învățarea activă

Q-learning

Deep Q-learning

Concluzii

Algoritmul Q-Learning (Watkins, 1989)

- ▶ Algoritmul Q-Learning învață o funcție acțiune-valoare $Q(s, a)$ (*Q quality*).

$Q(s, a)$ valoarea asociată realizării acțiunii a în starea s .

Relația dintre utilități și valorile Q : $U(s) = \max_a Q(s, a)$.

- ▶ Ecuațiile adevărate la echilibru când valorile Q sunt corecte

$$Q(s, a) = R(s) + \gamma \sum_{s'} P(s'|s, a) \max_{a'} Q(s', a')$$

Acestea pot fi utilizate într-un proces iterativ care calculează valorile Q exacte.

Algoritmul Q-Learning

- ▶ Un agent TD care învață o funcție Q nu are nevoie de un model probabilist $P(s'|s, a)$ (*învățare fără model*).
- ▶ Pentru fiecare eșantion (s, a, s', r) , se actualizează valoarea Q .
Ecuația de actualizare pentru *TD Q-Learning*:

$$Q(s, a) = Q(s, a) + \alpha(R(s) + \gamma \max_{a'} Q(s', a') - Q(s, a))$$

(executând acțiunea a în starea s rezultă s')

Coeficientul de învățare α determină viteza de actualizare a estimărilor; de obicei, $\alpha \in (0, 1)$

Algoritmul Q-Learning: pseudocod

function Q-LEARNING-AGENT(*percept*) **returns** an action

inputs: *percept*, a percept indicating the current state s' and reward signal r'

persistent: Q , a table of action values indexed by state and action, initially zero

N_{sa} , a table of frequencies for state–action pairs, initially zero

s, a, r , the previous state, action, and reward, initially null

if TERMINAL?(s) **then** $Q[s, \text{None}] \leftarrow r'$

if s is not null **then**

 increment $N_{sa}[s, a]$

$Q[s, a] \leftarrow Q[s, a] + \alpha(N_{sa}[s, a])(r + \gamma \max_{a'} Q[s', a'] - Q[s, a])$

$s, a, r \leftarrow s', \operatorname{argmax}_{a'} f(Q[s', a'], N_{sa}[s', a']), r'$

return a

f funcție de explorare

- ▶ Q-learning converge la o politică optimă
- ▶ Q-Learning este mai lent decât ADP

Q-learning: exemplu

Pacman is in an unknown MDP where there are three states [A, B, C] and two actions [Stop, Go]. We are given the following samples generated from taking actions in the unknown MDP. For the following problems, assume $\gamma = 1$ and $\alpha = 0.5$.

(a) We run Q-learning on the following samples:

| s | a | s' | r |
|---|------|----|----|
| A | Go | B | 2 |
| C | Stop | A | 0 |
| B | Stop | A | -2 |
| B | Go | C | -6 |
| C | Go | A | 2 |
| A | Go | A | -2 |

What are the estimates for the following Q-values as obtained by Q-learning? All Q-values are initialized to 0.

$$Q(C, Stop) = ?, Q(C, Go) = ?$$

- Ecuția de actualizare

$$Q(s, a) = Q(s, a) + \alpha(R(s) + \gamma Q(s', a') - Q(s, a))$$

(s', a') perechea (starea următoare, acțiunea următoare)

SARSA utilizează abordarea TD: se actualizează tabelul Q după fiecare pas până când soluția converge/nr. max. de iterații.

- Exemplu aplicație: *Windy Gridworld*

<http://www.incompleteideas.net/book/ebook/node64.html>

Introducere

Învățarea pasivă

Bazată pe model: ADP

Fără model: TD-learning

Învățarea activă

Q-learning

Deep Q-learning

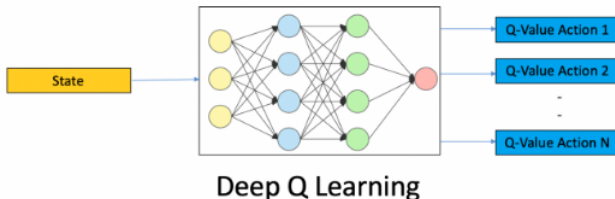
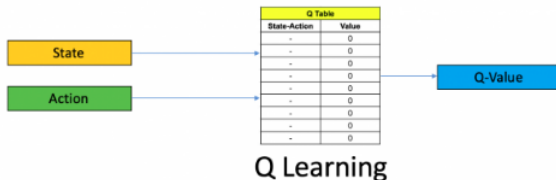
Concluzii

Deep Reinforcement Learning

Utilizează o rețea neurală (profundă) pentru a aproxima valorile Q

► intrare: o stare

ieșire: o estimare a lui Q , pentru fiecare acțiune posibilă



Deep Reinforcement Learning

- Considerăm ec. de actualizare a valorii Q (derivată din ec. Bellman):

$$\begin{aligned}Q(s, a) &= Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)] \\&= (1 - \alpha)Q(s, a) + r + \gamma \max_{a'} Q(s', a')\end{aligned}$$

- Funcția de cost: eroarea medie patratică dintre valoarea Q prezisă și valoarea țintă Q^* (nu se cunoaște).

Valoarea țintă: $target(s') = r + \gamma \max_{a'} Q(s', a')$

Minimizăm $loss(s, a, s') = (Q(s, a) - target(s'))^2$.

- Utilizăm metoda *Gradient descent* pentru a optimiza funcția de cost

Descriere: <https://deeplearningmath.org/deep-reinforcement-learning.html>

Demo: <https://cs.stanford.edu/people/karpathy/reinforcejs/>

Probleme:

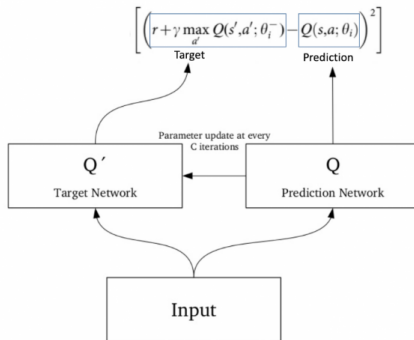
- ▶ eşantioanele sunt corelate \rightarrow rețeaua nu poate generaliza
- ▶ $target(s')$ este o estimare \rightarrow convergență lentă/alg. nu e stabil

Soluții:

- ▶ ϵ -greedy policy
- ▶ *experience replay*: memorăm experiențele (s, a, r, s') și le folosim pentru antrenare (*mini-batch*)

Double Deep Q-network

- ▶ valoarea țintă se modifică la fiecare iterație; soluție: o rețea separată pentru a estima valoarea țintă



- ▶ la fiecare C iterații, parametrii din rețeaua de predicție sunt copiați în rețeaua țintă

Function approximation

$$\hat{U}_\theta(s) = \theta_1 f_1(s) + \theta_2 f_2(s) + \dots \theta_n f_n(s)$$

f_1, \dots, f_n attribute

RL învață valorile parametrilor $\theta = \theta_1, \dots, \theta_n$ a.i. funcția de evaluare \hat{U}_θ aproximează funcția utilitate.

- ▶ actualizează parametrii după fiecare încercare
- ▶ utilizează o funcție de eroare și calculează gradientii în raport cu θ

$$E_j(s) = (\hat{U}_\theta(s) - u_j(s))^2/2$$

$u_j(s)$ recompensa totală observată din starea s pentru încercarea j

$$\theta_i \leftarrow \theta_i - \alpha \frac{\delta E_j(s)}{\delta \theta_i} = \theta_i + \alpha (u_j(s) - \hat{U}_\theta(s)) \frac{\delta \hat{U}_\theta(s)}{\delta \theta_i}$$

- ▶ putere de generalizare (stări vizitate \rightarrow stări nevizitate)

- ▶ Metode de tip *policy gradient*: învață o *politică parametrizată* (fără a decide acțiunea pe baza valorilor).

$$\pi(a|s, \theta) = Pr\{A_t = a | S_t = s, \theta_t = \theta\}$$

Scopul: identificarea parametrilor θ a.i. să maximizăm performanța $J(\theta)$. Parametrii sunt actualizați utilizând *Gradient Ascent*
 $\theta_{t+1} = \theta_t + \alpha \widehat{\nabla J(\theta_t)}$.

Politica poate fi stochastică.

- ▶ Metode de tip *actor-critic*: învață atât politica cât și valorile.

ChatGPT utilizează *Proximal policy optimization*:

<https://openai.com/research/openai-baselines-ppo>.

Conținut

Introducere

Învățarea pasivă

Bazată pe model: ADP

Fără model: TD-learning

Învățarea activă

Q-learning

Deep Q-learning

Concluzii

- ▶ Învăţarea cu întărire este necesară pentru agenţii care evoluează în medii necunoscute
- ▶ Învăţarea **pasivă** presupune evaluarea unei politici date
- ▶ Învăţarea **activă** presupune învăţarea unei politici optime

- ▶ *Artificial Intelligence: A modern Approach*. Ch. 21. Reinforcement Learning
- ▶ Sutton&Barto. *Reinforcement Learning. An introduction*
<http://incompleteideas.net/book/RLbook2020.pdf>