

Control prin învățare

Master ICAF, An 1 Sem 2

Lucian Bușoniu



Partea III

Învățarea prin recompensă



Recap: Soluție

Obiectiv

Maximizează **returnul** din orice x_0 :

$$R^h(x_0) = \sum_{k=0}^{\infty} \gamma^k r_{k+1} = \sum_{k=0}^{\infty} \gamma^k \rho(x_k, h(x_k))$$

- Funcția Q: $Q^h(x, u) = \rho(x, u) + \gamma R^h(x')$
- Funcția Q optimală: $Q^* = \max_h Q^h$

⇒ Legea de control optimală:

$$h^*(x) = \arg \max_u Q^*(x, u)$$

Recap: Programarea dinamică

- 1 Iterația pe valoare:
calculează Q^* , h^* greedy în Q^*
- 2 Iterația pe legea de control:
calculează Q^{h_ℓ} , $h_{\ell+1}$ greedy în Q^{h_ℓ} , repetă la $\ell + 1$

Partea III în plan

- Problema de învățare prin recompensă
- Soluția optimală
- Programarea dinamică (variabile discrete)
- **Învățarea prin recompensă (variabile discrete)**
- Tehnici de aproximare
- Programarea dinamică cu aproximare (var. continue)
- Învățarea prin recompensă cu aproximare (var. continue)
- Planificarea online (var. continue și discrete)



Conținut partea III

- 1 Monte Carlo, MC
- 2 Diferențe temporale, TD
- 3 Metode pentru accelerarea TD

- 1 Monte Carlo, MC
- 2 Diferențe temporale, TD
- 3 Metode pentru accelerarea TD

Reamintim: Iterația pe legea de control

Iterația pe legea de control

inițializează legea de control h_0

repeat la fiecare iterație ℓ

1: **evaluarea legii de control**: găsește Q^{h_ℓ}

2: **îmbunătățirea legii de control**:

$$h_{\ell+1}(x) \leftarrow \arg \max_u Q^{h_\ell}(x, u)$$

until convergență

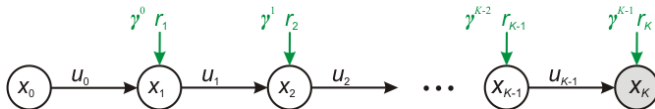
Evaluarea legii de control

Pentru a găsi Q^h :

- Până acum: metode bazate pe model
- Învățare prin recompensă: **modelul nu este disponibil**
- Învăță Q^h din date sau prin **interacțiune online cu sistemul**

Evaluarea Monte Carlo a legii de control

Reamintim: $Q^h(x_0, u_0) = \rho(x_0, u_0) + \gamma R^h(x_1)$



- Traiectorie din (x_0, u_0) până în x_K (**terminală**) folosind $u_1 = h(x_1)$, $u_2 = h(x_2)$ etc.

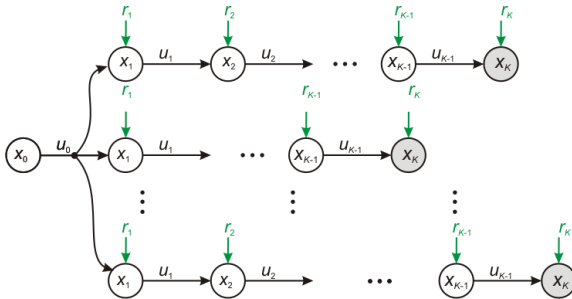
$\Rightarrow Q^h(x_0, u_0) = \text{returnul pe traiectorie:}$

$$Q^h(x_0, u_0) = \sum_{j=0}^{K-1} \gamma^j r_{j+1}$$

- La fiecare pas:

$$Q^h(x_k, u_k) = \sum_{j=k}^{K-1} \gamma^{j-k} r_{j+1}$$

Cazul stochastic



- N traectorii (diferă datorită naturii stohastice)
- Valoarea Q estimată = **media** returnurilor, ex.

$$Q^h(x_0, u_0) = \frac{1}{N} \sum_{i=1}^N \sum_{j=0}^{K_i-1} \gamma^j r_{i,j+1}$$

Cazul stochastic (cont.)

- Reamintim definiția valorii Q^h :

$$\begin{aligned} Q^h(x_0, u_0) &= E_{x_1} \left\{ \tilde{\rho}(x_0, u_0, x_1) + \gamma R^h(x_1) \right\} \\ &= E_{\text{traj. } x_1, x_2, \dots} \left\{ \tilde{\rho}(x_0, u_0, x_1) + \gamma \sum_{j=1}^{\infty} \gamma^j \tilde{\rho}(x_j, h(x_j), x_{j+1}) \right\} \\ &= E_{\text{traj. } x_1, x_2, \dots} \left\{ \sum_{j=0}^{K-1} \gamma^j r_{j+1} \mid x_0, u_0 \right\} \end{aligned}$$

(cu **presupunerea** că traiectoriile se termină după un #finit de pași K)

⇒ **Convergență** la valoarea Q când $N \rightarrow \infty$

$$\frac{1}{N} \sum_{i=1}^N \sum_{j=0}^{K_i-1} \gamma^j r_{i,j+1} \longrightarrow Q^h(x_0, u_0)$$

Iterația Monte Carlo pe legea de control

Iterația Monte Carlo pe legea de control

```

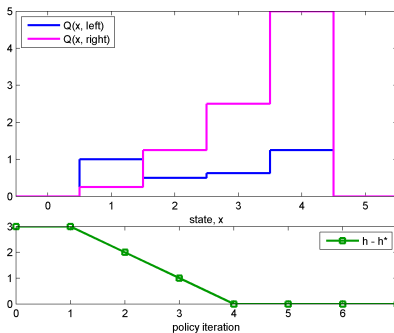
for fiecare iterație  $\ell$  do
    efectuează  $N$  traiectorii aplicând  $h_\ell$ 
    resetează la 0 acumulator  $A(x, u)$ , counter  $C(x, u)$ 
    for fiecare pas  $k$  din fiecare traiectorie  $i$  do
         $A(x_k, u_k) \leftarrow A(x_k, u_k) + \sum_{j=k}^{K_i-1} \gamma^{j-k} r_{i,j+1}$  (return)
         $C(x_k, u_k) \leftarrow C(x_k, u_k) + 1$ 
    end for
     $Q^{h_\ell}(x, u) \leftarrow A(x, u) / C(x, u)$ 
     $h_{\ell+1}(x) \leftarrow \arg \max_u Q^{h_\ell}(x, u)$ 
end for
    
```

De notat: **atingerea stării terminale** trebuie garantată!



Robot menajer: Monte Carlo, demo

Monte Carlo, trial 70 [piter 7 done, peval 10]



Nevoia de explorare

$$Q^h(x, u) \leftarrow A(x, u) / \mathbf{C(x, u)}$$

Cum asigurăm $C(x, u) > 0$ – **informație** despre fiecare (x, u) ?

- 1 Stări inițiale x_0 selectate reprezentativ
- 2 Acțiuni:
 u_0 reprezentative, câteodată diferite de $h(x_0)$
 și în plus, posibil:
 u_k reprezentative, câteodată diferite de $h(x_k)$

Explorare-exploatare

- **Explorarea** necesară:
acțiuni diferite de legea curentă de control
- **Exploatarea** cunoștințelor curente necesară:
legea de control trebuie aplicată

Dilema explorare-exploatare

– esențială în toți algoritmi de RL

(nu doar în MC)

Explorare-exploatare: strategia ϵ -greedy

- O soluție simplă: **ϵ -greedy**

$$u_k = \begin{cases} h(x_k) = \arg \max_u Q(x_k, u) & \text{cu probabilitatea } (1 - \epsilon_k) \\ \text{o acțiune aleatoare} & \text{cu probabilitatea } \epsilon_k \end{cases}$$

- Probabilitatea de explorare $\epsilon_k \in (0, 1)$ scade de obicei în timp



Îmbunătățirea optimistă a legii de control

- Legea de control neschimbată pentru N traiectorii
- ⇒ Algoritmul învață încet
- Îmbunătățirea legii de control după fiecare traiectorie
= **optimist**



Metoda Monte Carlo optimistă

Metoda Monte Carlo optimistă

inițializează la 0 acumulator $A(x, u)$, counter $C(x, u)$

for fiecare traiectorie **do**

efectuează traiectoria, ex. aplicând ε -greedy:

$$u_k = \begin{cases} \arg \max_u Q(x_k, u) & \text{cu prob. } (1 - \varepsilon_k) \\ \text{aleatoare} & \text{cu prob. } \varepsilon_k \end{cases}$$

for fiecare pas k **do**

$$A(x_k, u_k) \leftarrow A(x_k, u_k) + \sum_{j=k}^{K-1} \gamma^{j-k} r_{j+1}$$

$$C(x_k, u_k) \leftarrow C(x_k, u_k) + 1$$

end for

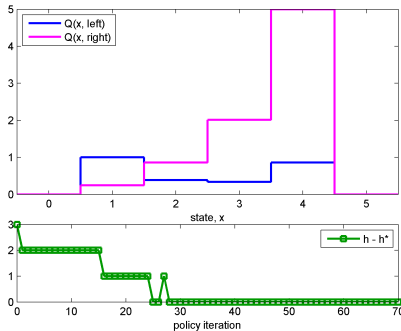
$$Q(x, u) \leftarrow A(x, u) / C(x, u)$$

end for

- h implicit, greedy în Q
- actualizarea $Q \Rightarrow$ implicit îmbunătățirea h

Robot menajer: Monte Carlo optimist, demo

Monte Carlo, trial 70 [piter 70 done, peval 1]



- 1 Monte Carlo, MC
- 2 Diferențe temporale, TD
 - Introducere
 - SARSA
 - Învățarea Q
- 3 Metode pentru accelerarea TD

Reamintim: Evaluarea legii de control

- Transformă ecuația Bellman pentru Q^h :

$$Q^h(x, u) = \rho(x, u) + \gamma Q^h(f(x, u), h(f(x, u)))$$

într-o procedură iterativă:

repeat la fiecare iterație τ

for all x, u **do**

$$Q_{\tau+1}(x, u) \leftarrow \rho(x, u) + \gamma Q_{\tau}(f(x, u), h(f(x, u)))$$

end for

until convergență la Q^h



Perspectiva DP

- 1 Pornim de la evaluarea legii de control:

$$Q_{\tau+1}(x, u) \leftarrow \rho(x, u) + \gamma Q_{\tau}(f(x, u), h(f(x, u)))$$

- 2 În loc de model, folosim la fiecare pas k **tranziția**

$(x_k, u_k, x_{k+1}, r_{k+1}, u_{k+1})$:

$$Q(x_k, u_k) \leftarrow r_{k+1} + \gamma Q(x_{k+1}, u_{k+1})$$

De notat: $x_{k+1} = f(x_k, u_k)$, $r_{k+1} = \rho(x_k, u_k)$, $u_{k+1} \sim h(x_{k+1})$

- 3 Transformăm într-o actualizare **incrementală**:

$$Q(x_k, u_k) \leftarrow Q(x_k, u_k) + \alpha_k \cdot$$

$$[r_{k+1} + \gamma Q(x_{k+1}, u_{k+1}) - Q(x_k, u_k)]$$

$\alpha_k \in (0, 1]$ rata de învățare

Algoritm intermediar

Diferențe temporale pentru evaluarea h

for fiecare traiectorie **do**

inițializează x_0 , alege acțiunea inițială u_0

repeat la fiecare pas k

aplică u_k , măsoară x_{k+1} , primește r_{k+1}

alege acțiunea **următoare** $u_{k+1} \sim h(x_{k+1})$

$$Q(x_k, u_k) \leftarrow Q(x_k, u_k) + \alpha_k \cdot$$

$$[r_{k+1} + \gamma Q(x_{k+1}, u_{k+1}) - Q(x_k, u_k)]$$

until traiectoria terminată

end for

Perspectiva MC

Diferențe temporale pentru evaluarea h

for fiecare traiectorie **do**

...

repeat la fiecare pas k

 aplică u_k , măsoară x_{k+1} , primește r_{k+1}

$Q(x_k, u_k) \leftarrow \dots Q \dots$

until traiectoria terminată

end for

Monte Carlo

for fiecare traiectorie **do**

 efectuează traiectoria

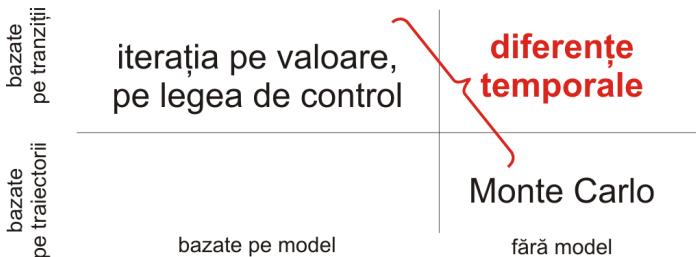
...

$Q(x, u) \leftarrow A(x, u) / C(x, u)$

end for

Perspective MC și DP

- Învăță din interacțiune online: ca și MC, diferit de DP
- Actualizează după fiecare tranziție, folosind valorile Q precedente: ca și DP, diferit de MC



Explorare-exploatare

alege acțiunea următoare $u_{k+1} \sim h(x_{k+1})$

- Informații despre $(x, u) \neq (x, h(x))$ necesare
⇒ **explorare**
- h trebuie urmărită
⇒ **exploatare**
- Ex. ε -greedy:

$$u_{k+1} = \begin{cases} h(x_{k+1}) & \text{cu prob. } (1 - \varepsilon_{k+1}) \\ \text{aleatoare} & \text{cu prob. } \varepsilon_{k+1} \end{cases}$$

- 1 Monte Carlo, MC
- 2 Diferențe temporale, TD
 - Introducere
 - **SARSA**
 - Învățarea Q
- 3 Metode pentru accelerarea TD

Îmbunătățirea legii de control

- Algoritm precedent: h fixată
- Îmbunătățirea h : cel mai simplu, după fiecare tranziție
⇒ interpretare: iterație pe legea de control
optimistă la nivel de tranziție
- h implicit, greedy în Q
(actualizarea $Q \Rightarrow$ implicit îmbunătățirea h)

SARSA

SARSA cu ϵ -greedy

for fiecare traiectorie **do**

inițializează x_0

$$u_0 = \begin{cases} \arg \max_u Q(x_0, u) & \text{cu prob. } (1 - \epsilon_0) \\ \text{aleatoare} & \text{cu prob. } \epsilon_0 \end{cases}$$

repeat la fiecare pas k

aplică u_k , măsoară x_{k+1} , primește r_{k+1}

$$u_{k+1} = \begin{cases} \arg \max_u Q(x_{k+1}, u) & \text{cu prob. } (1 - \epsilon_{k+1}) \\ \text{aleatoare} & \text{cu prob. } \epsilon_{k+1} \end{cases}$$

$$Q(x_k, u_k) \leftarrow Q(x_k, u_k) + \alpha_k \cdot$$

$$[r_{k+1} + \gamma Q(x_{k+1}, u_{k+1}) - Q(x_k, u_k)]$$

until traiectoria terminată

end for



Numele SARSA

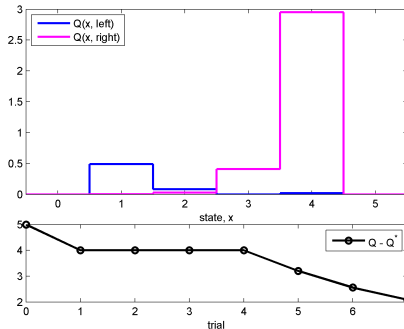
$(x_k, u_k, r_{k+1}, x_{k+1}, u_{k+1}) =$

(**S**tare, **A**cțiune, **R**ecompensă, **S**tare, **A**cțiune) = **SARSA**

Robot menajer: SARSA, demo

Parametri: $\alpha = 0.2$, $\varepsilon = 0.3$ (constanți)
 $x_0 = 2$ sau 3 (aleator)

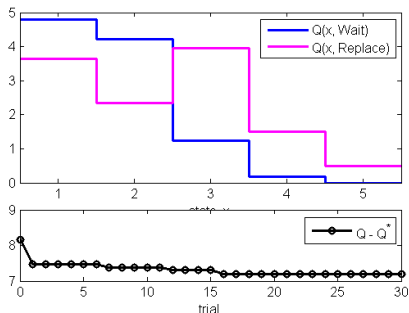
SARSA, trial 8, step 3



Înlocuirea unei mașini: SARSA, demo

Parametri: $\alpha = 0.1$, $\varepsilon = 0.3$ (constanți), 20 pași pe traiectorie
 $x_0 = 1$

SARSA, trial 30 completed



- 1 Monte Carlo, MC
- 2 Diferențe temporale, TD
 - Introducere
 - SARSA
 - Învățarea Q
- 3 Metode pentru accelerarea TD

Reamintim: Iterația Q

- Transformă ecuația de optimalitate Bellman:

$$Q^*(x, u) = \rho(x, u) + \gamma \max_{u'} Q^*(f(x, u), u')$$

într-o procedură iterativă:

repeat la fiecare iterație ℓ

for all x, u **do**

$$Q_{\ell+1}(x, u) \leftarrow \rho(x, u) + \gamma \max_{u'} Q_{\ell}(f(x, u), u')$$

end for

until convergență la Q^*

Învățarea Q

- 1 Similar cu SARSA, pornim de la iterația Q:

$$Q_{\ell+1}(x, u) \leftarrow \rho(x, u) + \gamma \max_{u'} Q_{\ell}(f(x, u), u')$$

- 2 În loc de model, folosim la fiecare pas k **tranziția**

$(x_k, u_k, x_{k+1}, r_{k+1})$:

$$Q(x_k, u_k) \leftarrow r_{k+1} + \gamma \max_{u'} Q(x_{k+1}, u')$$

De notat: $x_{k+1} = f(x_k, u_k)$, $r_{k+1} = \rho(x_k, u_k)$

- 3 Transformăm într-o actualizare **incrementală**:

$$Q(x_k, u_k) \leftarrow Q(x_k, u_k) + \alpha_k \cdot$$

$$[r_{k+1} + \gamma \max_{u'} Q(x_{k+1}, u') - Q(x_k, u_k)]$$

Învățarea Q

Învățarea Q cu ε -greedy

for fiecare traiectorie **do**

inițializează x_0

repeat la fiecare pas k

$$u_k = \begin{cases} \arg \max_u Q(x_k, u) & \text{cu prob. } (1 - \varepsilon_k) \\ \text{aleatoare} & \text{cu prob. } \varepsilon_k \end{cases}$$

aplică u_k , măsoară x_{k+1} , primește r_{k+1}

$$Q(x_k, u_k) \leftarrow Q(x_k, u_k) + \alpha_k \cdot$$

$$[r_{k+1} + \gamma \max_{u'} Q(x_{k+1}, u') - Q(x_k, u_k)]$$

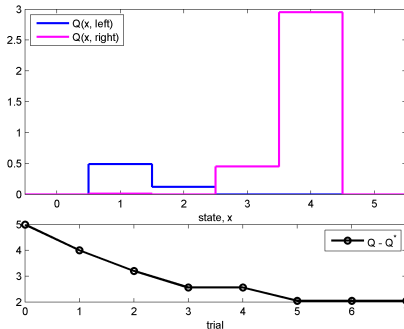
until traiectoria terminată

end for

Robot menajer: învățarea Q, demo

Parameteri – ca și SARSA: $\alpha = 0.2$, $\varepsilon = 0.3$ (constanți)
 $x_0 = 2$ sau 3 (aleator)

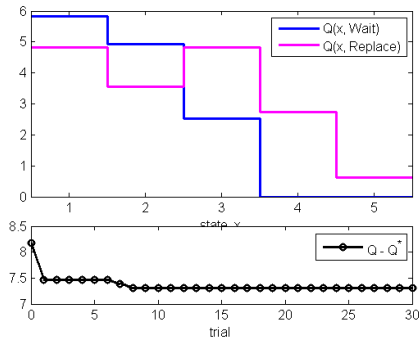
Q-learning, trial 8, step 3



Înlocuirea unei mașini: învățarea Q, demo

Parametri: $\alpha = 0.1$, $\varepsilon = 0.3$ (constanți), 20 pași pe traiectorie
 $x_0 = 1$

Q-learning, trial 30 completed



Convergență

Condiții de convergență la Q^* :

- 1 Toate perechile (x, u) continuă să fie actualizate:
asigurat de **explorare**, ex. ε -greedy
- 2 Condiții tehnice pentru α_k (scade spre 0, $\sum_{k=0}^{\infty} \alpha_k^2 = \text{finit}$,
dar nu prea repede, $\sum_{k=0}^{\infty} \alpha_k \rightarrow \infty$)

În plus, pentru SARSA:

- 3 Legea de control trebuie să devină greedy la infinit
ex. $\lim_{k \rightarrow \infty} \varepsilon_k = 0$

On-policy / off-policy

SARSA: **on-policy**

- Estimează permanent funcția Q a legii de control curente

Învățarea Q: **off-policy**

- Indiferent de legea de control curentă, estimează funcția Q optimală

Diferențe temporale: Discuție

Avantaje

- Simplu de înțeles, implementat
- Complexitate scăzută $\Rightarrow$ execuție rapidă

SARSA vs. învățarea Q

- SARSA mai puțin complex decât învățarea Q (fără max în actualizarea funcției Q)

Secvențele α_k, ε_k **influențează semnificativ** performanța

Dezavantaj principal

- Necesită multe date



- 1 Monte Carlo, MC
- 2 Diferențe temporale, TD
- 3 Metode pentru accelerarea TD
 - Motivare
 - Urme de eligibilitate
 - Reluarea experienței

De ce să accelerăm TD?

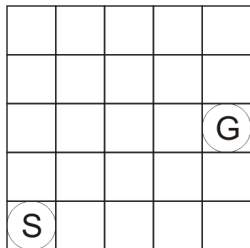
Dezavantaj principal: învață încet – **necesită multe date**

În practică, datele costă:

- timp
- profit (performanță scăzută datorită explorării)
- uzură a sistemului

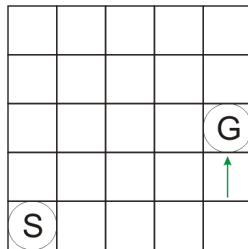
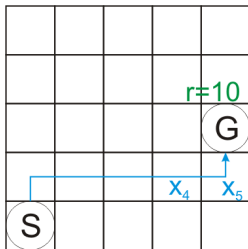
Accelerarea RL = **eficientizarea folosirii datelor**

Exemplu: Navigare 2D



- Navigare într-o lume 2D discretă de la **Start** la **Goal**
- Singura recompensă = 10 la atingerea G (stare terminală)

Exemplu: TD



- Alegem SARSA, $\alpha = 1$; inițializăm $Q = 0$
- Actualizări de-a lungul traiectoriei din stânga:

...

$$Q(x_4, u_4) = 0 + \gamma \cdot Q(x_5, u_5) = 0$$

$$Q(x_5, u_5) = 10 + \gamma \cdot 0 = 10$$

- O nouă tranziție de la x_4 la x_5 necesară pentru a propaga informația la x_4 !

Accelerarea TD: 2 idei

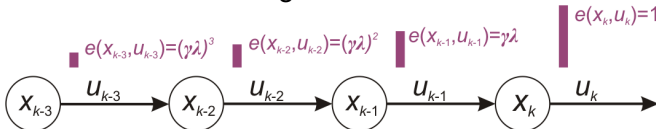
- 1 Lasă **urme de eligibilitate**
de-a lungul traiectoriilor
- 2 Stochează și **reia experiența**



- 1 Monte Carlo, MC
- 2 Diferențe temporale, TD
- 3 Metode pentru accelerarea TD
 - Motivare
 - Urme de eligibilitate
 - Reluarea experienței

Urme de eligibilitate

- Idee: Lasă o **urmă** de-a lungul traiectoriei:



- $\lambda \in [0, 1]$ rată de scădere

Urme de eligibilitate “cu înlocuire”

$e(x, u) \leftarrow 0$ pentru toate x, u

for fiecare pas k **do**

$e(x, u) \leftarrow \lambda\gamma e(x, u)$ pentru toate x, u

$e(x_k, u_k) \leftarrow 1$

end for

SARSA(λ)

- Reamintim SARSA original actualizează doar $Q(x_k, u_k)$:

$$Q(x_k, u_k) \leftarrow Q(x_k, u_k) + \alpha_k \cdot [r_{k+1} + \gamma Q(x_{k+1}, u_{k+1}) - Q(x_k, u_k)]$$

- SARSA(λ) actualizează **toate perechile eligibile**:

$$Q(x, u) \leftarrow Q(x, u) + \alpha_k \cdot e(x, u) \cdot [r_{k+1} + \gamma Q(x_{k+1}, u_{k+1}) - Q(x_k, u_k)] \quad \forall x, u$$



Algoritmul SARSA(λ)

SARSA(λ)

for fiecare traiectorie **do**

$e(x, u) \leftarrow 0 \quad \forall x, u$

inițializează x_0 , alege acțiunea inițială u_0

repeat la fiecare pas k

aplică u_k , măsoară x_{k+1} , primește r_{k+1}

alege acțiunea următoare u_{k+1}

$e(x, u) \leftarrow \lambda \gamma e(x, u) \quad \forall x, u$

$e(x_k, u_k) \leftarrow 1$

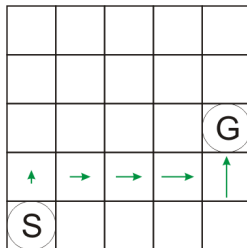
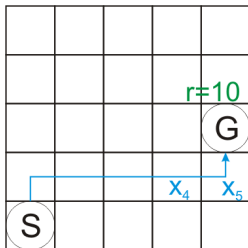
$Q(x, u) \leftarrow Q(x, u) + \alpha_k \cdot e(x, u) \cdot$

$[r_{k+1} + \gamma Q(x_{k+1}, u_{k+1}) - Q(x_k, u_k)]$ **for all** x, u

until traiectoria terminată

end for

Exemplu: Efectul urmei de eligibilitate



- $\lambda = 0.7$
- Actualizări până la x_4 : Q rămâne 0
- La x_5 , toată traiectoria actualizată imediat:

$$Q(x_5, u_5) = 10 + \gamma 0 = 10$$

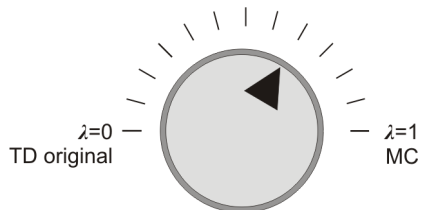
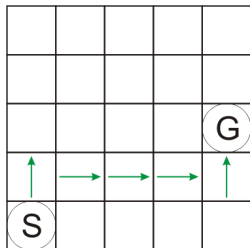
$$Q(x_4, u_4) = (\gamma \lambda)[10 + \gamma 0] = 3.5$$

$$Q(x_3, u_3) = (\gamma \lambda)^2[10 + \gamma 0] = 1.225$$

...

TD versus MC

- $\lambda = 0 \Rightarrow$ algoritmi originali regăsiți
- $\lambda = 1 \Rightarrow$ TD devine similar cu MC

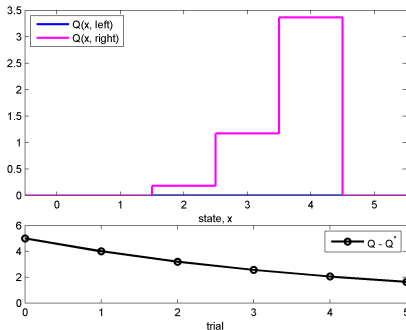


De obicei, $\lambda \in [0.5, 0.8]$ funcționează rezonabil

Robot menajer: SARSA(λ), demo

Parametri: $\alpha = 0.2$, $\varepsilon = 0.3$ (ca SARSA original), $\lambda = 0.5$
 $x_0 = 2$ or 3 (aleator)

SARSA(lambda), trial 6, step 2



Învățarea $Q(\lambda)$

- Similar cu SARSA, actualizarea originală:

$$Q(x_k, u_k) \leftarrow Q(x_k, u_k) + \alpha_k \cdot [r_{k+1} + \gamma \max_{u'} Q(x_{k+1}, u') - Q(x_k, u_k)]$$

- Se transformă în:

$$Q(x, u) \leftarrow Q(x, u) + \alpha_k \cdot e(x, u) \cdot [r_{k+1} + \gamma \max_{u'} Q(x_{k+1}, u') - Q(x_k, u_k)] \quad \forall x, u$$

- De notat: acțiunile de explorare întrerup cauzalitatea
 $\Rightarrow$ urma poate fi resetată la 0

Algoritmul $Q(\lambda)$

$Q(\lambda)$

for fiecare traiectorie **do**

$e(x, u) \leftarrow 0 \quad \forall x, u$

initialize x_0

repeat la fiecare pas k

aplică u_k , măsoară x_{k+1} , primește r_{k+1}

if u_k exploratoare **then** $e(x, u) \leftarrow 0 \quad \forall x, u$

else $e(x, u) \leftarrow \lambda \gamma e(x, u) \quad \forall x, u$

end if

$e(x_k, u_k) \leftarrow 1$

$Q(x, u) \leftarrow Q(x, u) + \alpha_k \cdot e(x, u) \cdot$

$[r_{k+1} + \gamma \max_{u'} Q(x_{k+1}, u') - Q(x_k, u_k)] \quad \forall x, u$

until traiectorie terminată

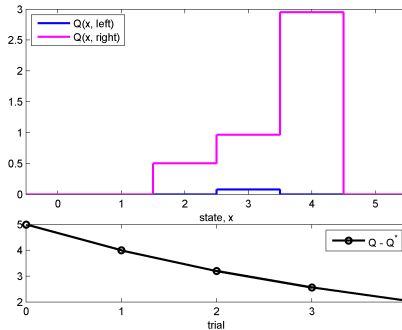
end for



Robot menajer: $Q(\lambda)$, demo

Parametri: $\alpha = 0.2$, $\varepsilon = 0.3$, $\lambda = 0.5$
 $x_0 = 2$ or 3 (aleator)

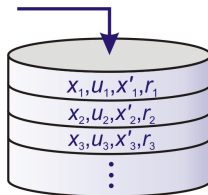
$Q(\lambda)$ -learning, trial 5, step 2



- 1 Monte Carlo, MC
- 2 Diferențe temporale, TD
- 3 Metode pentru accelerarea TD
 - Motivare
 - Urme de eligibilitate
 - Reluarea experienței

Reluarea experienței

- Stochează fiecare tranziție $(x_k, u_k, x_{k+1}, r_{k+1})$ într-o bază de date



- La fiecare pas, **reia** n tranziții din baza de date (pe lângă actualizările normale)

Învățarea Q cu reluarea experienței

Învățarea Q cu reluarea experienței

```

for fiecare traiectorie do
    inițializează  $x_0$ 
    repeat la fiecare pas  $k$ 
        aplică  $u_k$ , măsoară  $x_{k+1}$ , primește  $r_{k+1}$ 
         $Q(x_k, u_k) \leftarrow Q(x_k, u_k) + \alpha_k \cdot$ 
             $[r_{k+1} + \gamma \max_{u'} Q(x_{k+1}, u') - Q(x_k, u_k)]$ 
        adaugă  $(x_k, u_k, x_{k+1}, r_{k+1})$  la baza de date
        ReiaExperiența
    until traiectoria terminată
end for
    
```

Procedura ReiaExperiența

ReiaExperiența

loop de N ori

 preia o tranziție (x, u, x', r) din baza de date

$$Q(x, u) \leftarrow Q(x, u) + \alpha \cdot$$

$$[r + \gamma \max_{u'} Q(x', u') - Q(x, u)]$$

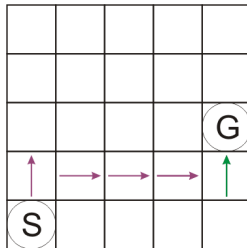
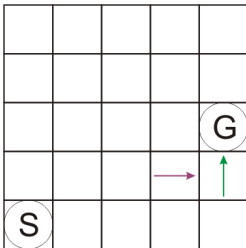
end loop

Direcția de reluare

Ordinea de reluare a tranzițiilor:

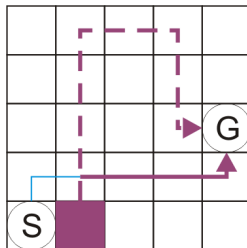
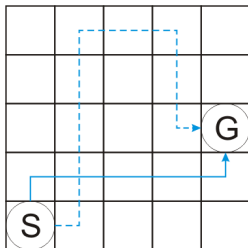
- 1 Înainte
- 2 Înapoi
- 3 Arbitrar

Exemplu: Influența direcției de reluare



- Verde: **actualizările normale**, mov: **reluarea experienței**
- Stânga: reluare înainte; dreapta: reluare înapoi
- **Reluarea înapoi** în general preferabilă

Exemplu: Agregarea informației

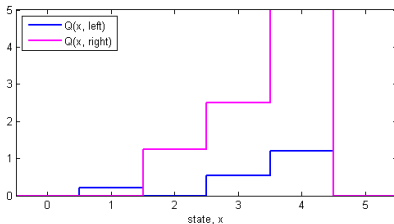


- Reluarea experienței **agregă informație** din mai multe traiectorii
- Căsuța indicată profită de informații de-a lungul ambelor traiectorii

Robot menajer: Q cu reluarea experienței, demo

Parametri: $\alpha = 0.2$, $\varepsilon = 0.3$, $N = 5$, direcția înapoi
 $x_0 = 2$ or 3 (aleator)

ER-Q-learning, trial 13, step 2 [replaying trial 8, step 2]



Recapitulare: Metode în Partea III

Metode Monte Carlo, MC:

- Iterația Monte Carlo pe legea de control
- Varianta optimistă

Diferențe temporale, TD:

- SARSA
- Învățarea Q

Metode pentru accelerarea TD:

- Urme de eligibilitate
- Reluarea experienței



Terminologie engleză

metode Monte Carlo	=	<i>Monte Carlo methods</i>
explorare-exploatare	=	<i>exploration-exploitation</i>
diferențe temporale	=	<i>temporal differences, TD</i>
învățarea Q	=	<i>Q-learning</i>
SARSA	=	<i>SARSA</i>
urme de eligibilitate	=	<i>eligibility traces</i>
reluarea experienței	=	<i>experience replay</i>