Control prin învăţare

Master ICAF, An 1 Sem 2

Lucian Buşoniu

Partea III

Învăţarea prin recompensă

Recap: Soluţie

Obiectiv

Maximizează returnul din orice x_0:

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

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

$\Rightarrow$ Legea de control optimală:

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

Recap: Programarea dinamică

1. Iteraţia pe valoare:
   calculează Q^*, h^* greedy în Q^*
2. Iteraţia 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 învăţare prin recompensă
- Soluţia optimală
- Programarea dinamică (variabile discrete)
- **Învăţarea prin recompensă (variabile discrete)**
- Tehnici de aproximare
- Programarea dinamică cu aproximare (var. continue)
- Învăţarea prin recompensă cu aproximare (var. continue)
- Planificarea online (var. continue şi discrete)

Conţinut partea III

1. Monte Carlo, MC
2. Diferenţe temporale, TD
3. Metode pentru accelerarea TD

Reamintim: Iteraţia pe legea de control

Iteraţia pe legea de control

iniţializează legea de control h_0

repeat la fiecare iteraţie ℓ

1: evaluarea legii de control: găseşte Q^{h_ℓ}

2: îmbunătăţirea legii de control:

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

until convergenţă

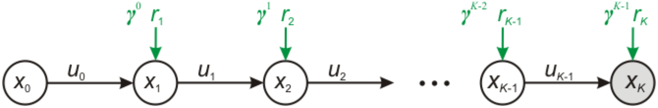
Evaluarea legii de control

Pentru a găsi Q^h:

- Până acum: metode bazate pe model
- Învăţare prin recompensă: **modelul nu este disponibil**
- Învaţă Q^h din date sau prin **interacţiune 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.

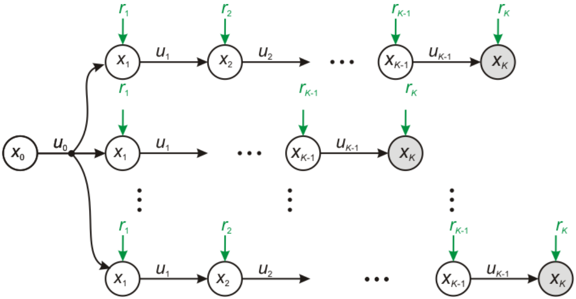
$\Rightarrow$ $Q^h(x_0, u_0) =$ returnul pe traiectorie:

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

- La fiecare pas:

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

Cazul stohastic



- N traiectorii (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 stohastic (cont.)

- Reamintim definiţia valorii Q^h:

$$Q^h(x_0,u_0) = \mathrm{E}_{x_1}\left\{\tilde{\rho}(x_0,u_0,x_1) + \gamma R^h(x_1)\right\}$$

$$= \mathrm{E}_{\text{traj. } x_1,x_2,\dots}\left\{\tilde{\rho}(x_0,u_0,x_1) + \gamma \sum_{j=1}^{\infty} \gamma^j \tilde{\rho}(x_j,h(x_j),x_{j+1})\right\}$$

$$= \mathrm{E}_{\text{traj. } x_1,x_2,\dots}\left\{\sum_{j=0}^{K-1} \gamma^j r_{j+1} \,\middle|\, x_0,u_0\right\}$$

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

$\Rightarrow$ **Convergenţă** la valoarea Q când $N \to \infty$

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

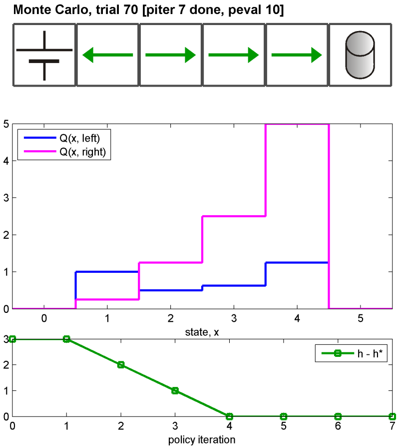
Iteraţia Monte Carlo pe legea de control

Iteraţia Monte Carlo pe legea de control

for fiecare iteraţie ℓ **do**
  efectuează N traiectorii aplicând h_ℓ
  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



Nevoia de explorare

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

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

1. Stări iniţiale x_0 selectate reprezentativ
2. Acţiuni:
   u_0 reprezentative, câteodată diferite de $h(x_0)$
   şi în plus, posibil:
   u_k reprezentative, câteodată diferite de $h(x_k)$

Explorare-exploatare

- **Explorarea** necesară:
  acţiuni diferite de legea curentă de control
- **Exploatarea** cunoştinţelor curente necesară:
  legea de control trebuie aplicată

Dilema explorare-exploatare
– esenţială în toţi algoritmii de RL

(nu doar în MC)

Explorare-exploatare: strategia ε-greedy

- O soluţie simplă: **ε-greedy**

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

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

Îmbunătăţirea optimistă a legii de control

- Legea de control neschimbată pentru N traiectorii

$\Rightarrow$ Algoritmul învaţă încet

- Îmbunătăţirea legii de control după fiecare traiectorie = **optimist**

Metoda Monte Carlo optimistă

Metoda Monte Carlo optimistă

iniţializează 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**

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

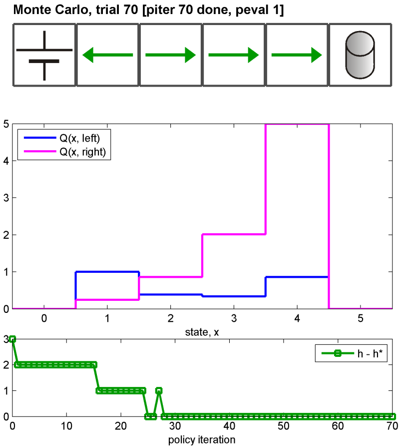
end for

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

end for

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

Robot menajer: Monte Carlo optimist, demo



Reamintim: Evaluarea legii de control

- Transformă ecuaţia 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 iteraţie 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 convergenţă 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 **tranziţia** $(x_k, u_k, x_{k+1}, r_{k+1}, u_{k+1})$:
$$Q(x_k, u_k) \leftarrow r_{k+1} + \gamma Q(x_{k+1}, u_{k+1})$$
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 învăţare

Algoritm intermediar

Diferenţe temporale pentru evaluarea h

for fiecare traiectorie **do**
  iniţializează x_0, alege acţiunea iniţială u_0
  repeat la fiecare pas k
    aplică u_k, măsoară x_{k+1}, primeşte r_{k+1}
    alege acţiunea **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

Diferenţe temporale pentru evaluarea h

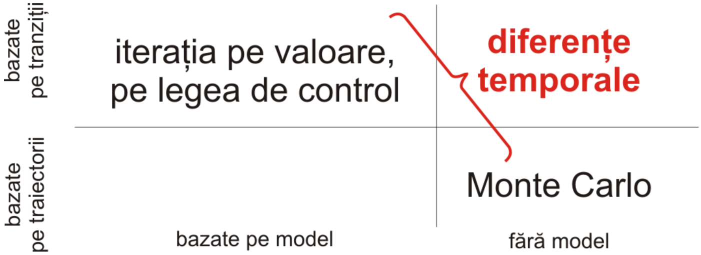
for fiecare traiectorie **do**
    ...
    repeat la fiecare pas k
        aplică u_k, măsoară x_{k+1}, primeşte r_{k+1}
        $Q(x_k, u_k) \leftarrow ...Q...$
    until traiectoria terminată
end for

Monte Carlo

for fiecare traiectorie **do**
    efectuează traiectoria
    ...
    $Q(x, u) \leftarrow A(x, u)/C(x, u)$
end for

Perspective MC şi DP

- Învaţă din interacţiune online: ca şi MC, diferit de DP
- Actualizează după fiecare tranziţie,
  folosind valorile Q precedente: ca şi DP, diferit de MC



Explorare-exploatare

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

- Informaţii despre $(x, u) \neq (x, h(x))$ necesare
  $\Rightarrow$ **explorare**
- h trebuie urmărită
  $\Rightarrow$ **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. Diferenţe temporale, TD
   - Introducere
   - SARSA
   - Învăţarea Q

3. Metode pentru accelerarea TD

Îmbunătăţirea legii de control

- Algoritm precedent: *h* fixată
- Îmbunătăţirea *h*: cel mai simplu, după fiecare tranziţie

$\Rightarrow$ interpretare: iteraţie pe legea de control **optimistă** la nivel de tranziţie

- *h* implicit, greedy în *Q*
  (actualizarea *Q* $\Rightarrow$ implicit îmbunătăţirea *h*)

SARSA

SARSA cu ε-greedy

for fiecare traiectorie **do**

initializează x_0

$$u_0 = \begin{cases} \arg\max_u Q(x_0, u) & \text{cu prob. } (1 - \varepsilon_0) \\ \text{aleatoare} & \text{cu prob. } \varepsilon_0 \end{cases}$$

repeat la fiecare pas k

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

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

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

until traiectoria terminată

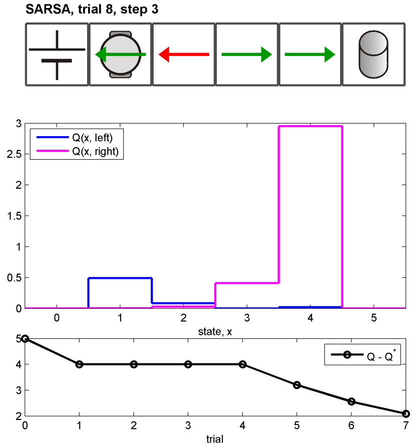
end for

Numele SARSA

$(x_k, u_k, r_{k+1}, x_{k+1}, u_{k+1})$ =
(**S**tare, **A**cţiune, **R**ecompensă, **S**tare, **A**cţiune) = **SARSA**

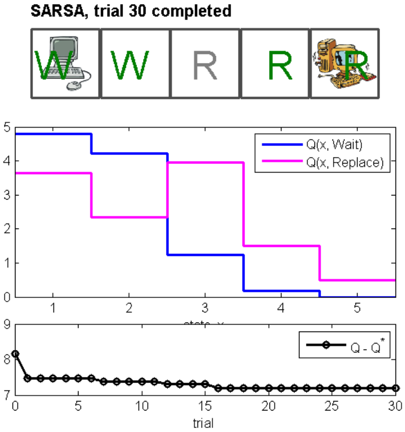
Robot menajer: SARSA, demo

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



Înlocuirea unei maşini: SARSA, demo

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



1 Monte Carlo, MC

2 Diferenţe temporale, TD
- Introducere
- SARSA
- Învăţarea Q

3 Metode pentru accelerarea TD

Reamintim: Iteraţia Q

- Transformă ecuaţia de optimalitate Bellman:

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

într-o procedură iterativă:

repeat la fiecare iteraţie 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 convergenţă la Q^*

Învăţarea Q

1. Similar cu SARSA, pornim de la iteraţia 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 **tranziţia** $(x_k, u_k, x_{k+1}, r_{k+1})$:
$$Q(x_k,u_k) \leftarrow r_{k+1} + \gamma \max_{u'} Q(x_{k+1},u')$$
De notat: $x_{k+1} = f(x_k,u_k)$, $r_{k+1} = \rho(x_k,u_k)$

3. Transformăm într-o actualizare **incrementală**:
$$Q(x_k,u_k) \leftarrow Q(x_k,u_k) + \alpha_k \cdot \left[r_{k+1} + \gamma \max_{u'} Q(x_{k+1},u') - Q(x_k,u_k)\right]$$

Învăţarea Q

Învăţarea Q cu ε-greedy

for fiecare traiectorie **do**
  iniţializează x_0
  repeat la fiecare pas k

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

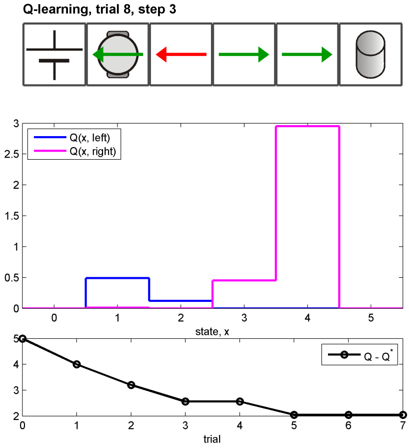
  aplică u_k, măsoară x_{k+1}, primeşte r_{k+1}

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

  until traiectoria terminată
end for

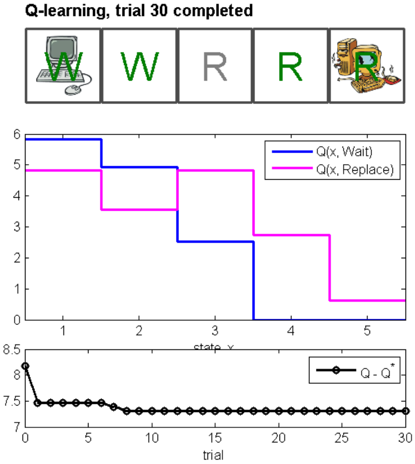
Robot menajer: învăţarea Q, demo

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



Înlocuirea unei maşini: învăţarea Q, demo

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



Convergenţă

Condiţii de convergenţă la Q^*:

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

În plus, pentru SARSA:

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

On-policy / off-policy

SARSA: **on-policy**

- Estimează permanent funcţia Q a legii de control curente

Învăţarea Q: **off-policy**

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

Diferenţe temporale: Discuţie

Avantaje

- Simplu de înţeles, implementat
- Complexitate scăzută $\Rightarrow$ execuţie rapidă

SARSA vs. învăţarea Q

- SARSA mai puţin complex decât învăţarea Q (fără max în actualizarea funcţiei Q)

Secvenţele α_k, ε_k **influenţează semnificativ** performanţa

Dezavantaj principal

- Necesită multe date

1. Monte Carlo, MC
2. Diferenţe temporale, TD
3. Metode pentru accelerarea TD
   - Motivare
   - Urme de eligibilitate
   - Reluarea experienţei

De ce să accelerăm TD?

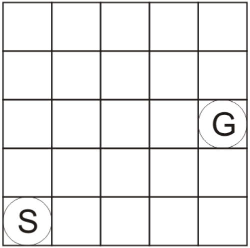
Dezavantaj principal: învaţă încet – **necesită multe date**

În practică, datele costă:

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

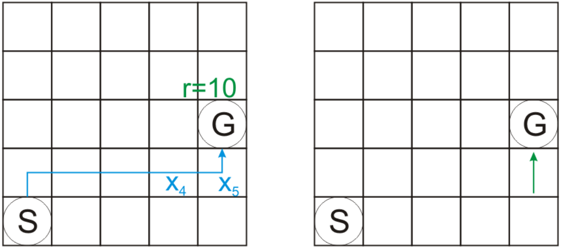
Accelerarea RL = **eficientizarea folosirii datelor**

Exemplu: Navigare 2D



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

Exemplu: TD



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

$$\dots$$

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

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

- O nouă tranziţie de la x_4 la x_5
  necesară pentru a propaga informaţia la x_4!

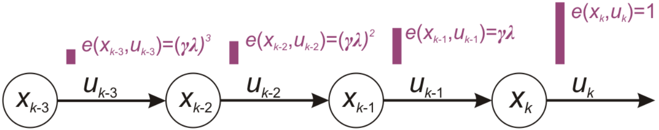
Accelerarea TD: 2 idei

1. Lasă **urme de eligibilitate** de-a lungul traiectoriilor
2. Stochează şi **reia experienţa**

1. Monte Carlo, MC
2. Diferenţe temporale, TD
3. Metode pentru accelerarea TD
   - Motivare
   - Urme de eligibilitate
   - Reluarea experienţei

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**

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

iniţializează x_0, alege acţiunea iniţială u_0

repeat la fiecare pas k

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

alege acţiunea următoare u_{k+1}

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

$e(x_k, u_k) \leftarrow 1$

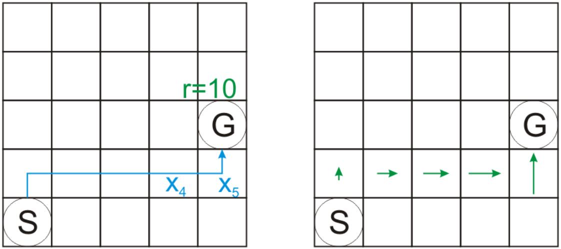
$Q(x, u) \leftarrow Q(x, u) + \alpha_k \cdot e(x, u) \cdot$

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

until traiectoria terminată

end for

Exemplu: Efectul urmei de eligibilitate

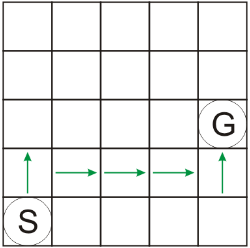


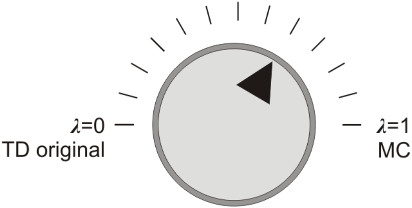
- $\lambda = 0.7$
- Actualizări până la x_4: Q rămâne 0
- La x_5, toată traiectoria actualizată imediat:

$$Q(x_5, u_5) = 10 + \gamma 0 = 10$$
$$Q(x_4, u_4) = (\gamma\lambda)[10 + \gamma 0] = 3.5$$
$$Q(x_3, u_3) = (\gamma\lambda)^2[10 + \gamma 0] = 1.225$$
$$\dots$$

TD versus MC

- $\lambda = 0 \Rightarrow$ algoritmii originali regăsiţi
- $\lambda = 1 \Rightarrow$ TD devine similar cu MC

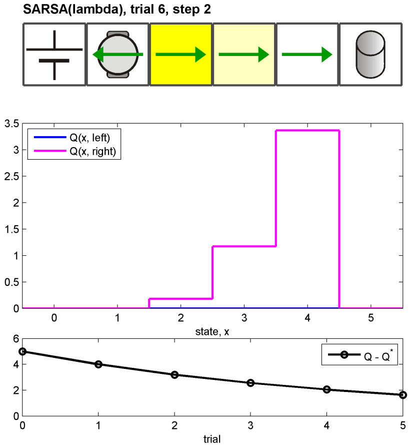




De obicei, $\lambda \in [0.5, 0.8]$ funcţionează rezonabil

Robot menajer: SARSA(λ), demo

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



Învăţarea Q(λ)

- 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: acţiunile de explorare întrerup cauzalitatea $\Rightarrow$ urma poate fi resetată la 0

Algoritmul Q(λ)

Q(λ)

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}, primeşte 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

$e(x_k,u_k) \leftarrow 1$

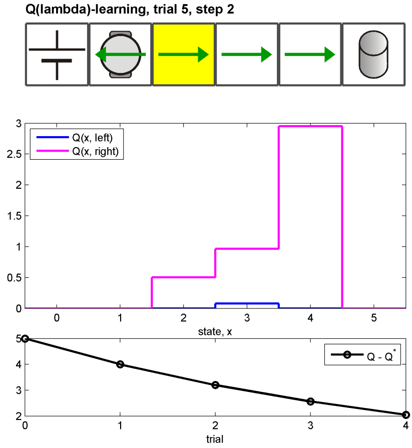
$$Q(x,u) \leftarrow Q(x,u) + \alpha_k \cdot e(x,u) \cdot \left[r_{k+1} + \gamma \max_{u'} Q(x_{k+1},u') - Q(x_k,u_k)\right] \forall x,u$$

until traiectorie terminată

end for

Robot menajer: Q(λ), demo

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



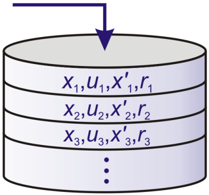
1. Monte Carlo, MC

2. Diferenţe temporale, TD

3. Metode pentru accelerarea TD
   - Motivare
   - Urme de eligibilitate
   - Reluarea experienţei

Reluarea experienţei

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



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

Învăţarea Q cu reluarea experienţei

Învăţarea Q cu reluarea experienţei

for fiecare traiectorie **do**
  iniţializează x_0
  repeat la fiecare pas k
    aplică u_k, măsoară x_{k+1}, primeşte 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
    ReiaExperienţa
  until traiectoria terminată
end for

Procedura ReiaExperienţa

ReiaExperienţa

loop de N ori

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

$$Q(x,u) \leftarrow Q(x,u) + \alpha \cdot \left[r + \gamma \max_{u'} Q(x',u') - Q(x,u)\right]$$

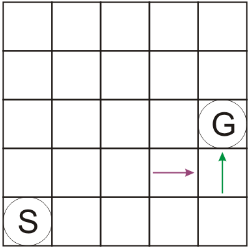
end loop

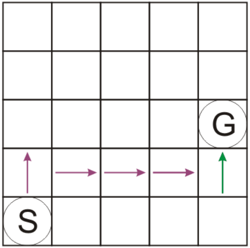
Direcţia de reluare

Ordinea de reluare a tranziţiilor:

1. Înainte
2. Înapoi
3. Arbitrar

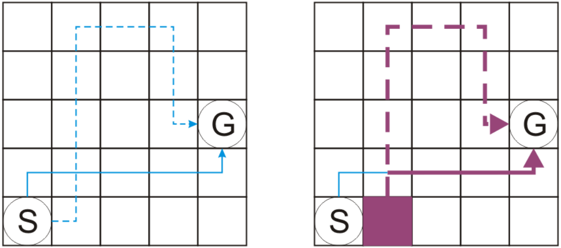
Exemplu: Influenţa direcţiei de reluare





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

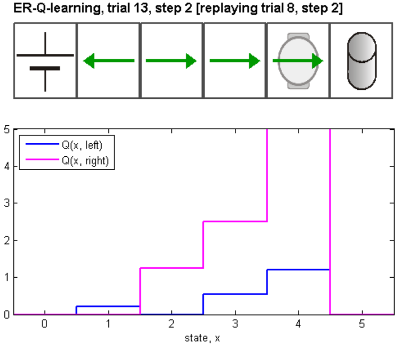
Exemplu: Agregarea informaţiei



- Reluarea experienţei **agregă informaţie** din mai multe traiectorii
- Căsuţa indicată profită de informaţii de-a lungul ambelor traiectorii

Robot menajer: Q cu reluarea experienţei, demo

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



Recapitulare: Metode în Partea III

Metode Monte Carlo, MC:

- Iteraţia Monte Carlo pe legea de control
- Varianta optimistă

Diferenţe temporale, TD:

- SARSA
- Învăţarea Q

Metode pentru accelerarea TD:

- Urme de eligibilitate
- Reluarea experienţei

Terminologie engleză

metode Monte Carlo	=	*Monte Carlo methods*
explorare-exploatare	=	*exploration-exploitation*
diferenţe temporale	=	*temporal differences, TD*
învăţarea Q	=	*Q-learning*
SARSA	=	*SARSA*
urme de eligibilitate	=	*eligibility traces*
reluarea experienţei	=	*experience replay*