Control prin învăţare

Master ICAF, An 1 Sem 2

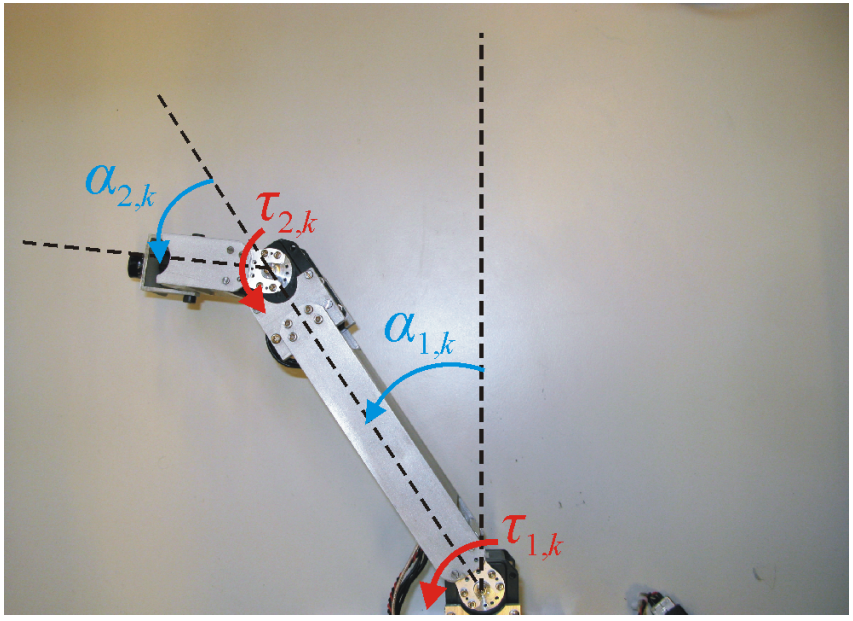
Lucian Buşoniu

Partea V

Învăţarea prin recompensă online cu aproximare

Recap: Nevoia de aproximare

- În aplicaţii reale de control, x, u **continue**!



- Reprezentarea prin tabel **imposibilă**
- **Aproximarea** funcţiilor de interes $Q(x,u)$, $V(x)$, $h(x)$ necesară

Partea 5 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)
- **Partea V: Învăţarea prin recompensă cu aproximare (var. continue)**
- ~~Planificarea online (var. continue şi discrete)~~

Recap: Partea 4 – Algoritmi offline

pornind de la:
– model f, ρ
– sau date (x_s, u_s, r_s, x'_s), $s = 1, \ldots, n_s$

1. găseşte soluţie aproximată $\widehat{Q}(x, u)$, $\widehat{h}(x)$, etc.
2. controlează sistemul folosind soluţia găsită

Algoritmi exemplificaţi:

- iteraţia fuzzy Q
- iteraţia Q bazată pe date
- LSPI, iteraţia lege de control CMMP

Partea 5 în plan: Categorii de algoritmi

După utilizarea unui model:

- Bazat pe model: f, ρ cunoscute
- **Fără model**: doar date (învăţarea prin recompensă)

După nivelul de interacţiune:

- Offline: algoritmul rulează în avans
- **Online**: algoritmul controlează direct sistemul

Exact vs. cu aproximare:

- Exact: x, u număr mic de valori discrete
- **Cu aproximare**: x, u continue (sau multe valori discrete)

Există mulţi algoritmi, doar câţiva sunt selectaţi pentru discuţie

Conţinut partea 5

Reamintim: 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

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

Învăţarea Q aproximată

- Învăţarea Q **scade diferenţa temporală**:

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

- $r_{k+1} + \gamma \max_{u'} Q(x_{k+1}, u')$ înlocuieşte **idealul** $Q^*(x_k, u_k)$

  [Vezi şi Bellman: $Q^*(x, u) = \rho(x, u) + \gamma \max_{u'} Q^*(x', u')$]

$\Rightarrow$ Ideal, scade eroarea $[Q^*(x_k, u_k) - Q(x_k, u_k)]$

Învăţarea Q aproximată (continuare)

Aproximare: folosim $\widehat{Q}(x, u; \theta)$, actualizăm **parametri**

- **Gradient** pe eroarea $[Q^*(x_k, u_k) - \widehat{Q}(x_k, u_k; \theta)]$:

$$\theta_{k+1} = \theta_k - \frac{1}{2}\alpha_k \frac{\partial}{\partial \theta} \left[Q^*(x_k, u_k) - \widehat{Q}(x_k, u_k; \theta_k) \right]^2$$
$$= \theta_k + \alpha_k \frac{\partial}{\partial \theta} \widehat{Q}(x_k, u_k; \theta_k) \cdot \left[Q^*(x_k, u_k) - \widehat{Q}(x_k, u_k; \theta_k) \right]$$

- Foloseşte **estimare** pentru $Q^*(x_k, u_k)$:

$$\theta_{k+1} = \theta_k + \alpha_k \frac{\partial}{\partial \theta} \widehat{Q}(x_k, u_k; \theta_k) \cdot$$
$$\left[r_{k+1} + \gamma \max_{u'} \widehat{Q}(x_{k+1}, u'; \theta_k) - \widehat{Q}(x_k, u_k; \theta_k) \right]$$

(diferenţă temporală aproximată)

Învăţarea Q aproximată: algoritm

Învăţarea Q aproximată cu explorare ε-greedy

for fiecare traiectorie **do**

iniţializează x_0

repeat la fiecare pas k

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

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

$$\theta_{k+1} = \theta_k + \alpha_k \frac{\partial}{\partial \theta} \widehat{Q}(x_k, u_k; \theta_k) \cdot \left[r_{k+1} + \gamma \max_{u'} \widehat{Q}(x_{k+1}, u'; \theta_k) - \widehat{Q}(x_k, u_k; \theta_k) \right]$$

until traiectoria terminată

end for

Desigur, **explorarea** necesară şi în cazul aproximat

Reamintim: Maximizare

Soluţia 1:

- Legea de control nu este reprezentată explicit
- Acţiuni greedy calculate la cerere din $\widehat{Q}$

Soluţia 2:

- Legea de control aproximată explicit

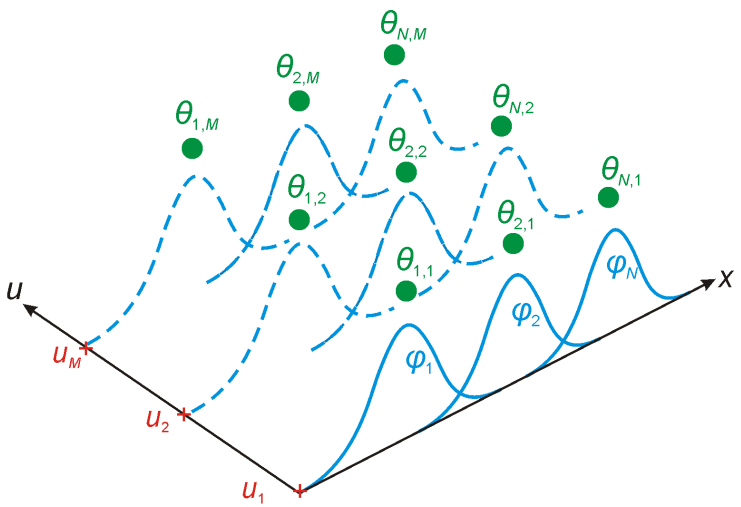
Maximizare în învăţarea Q aproximată

- Acţiuni greedy calculate la cerere din $\widehat{Q}$:

$$\ldots \max_u \widehat{Q}(x, u; \theta) \ldots$$

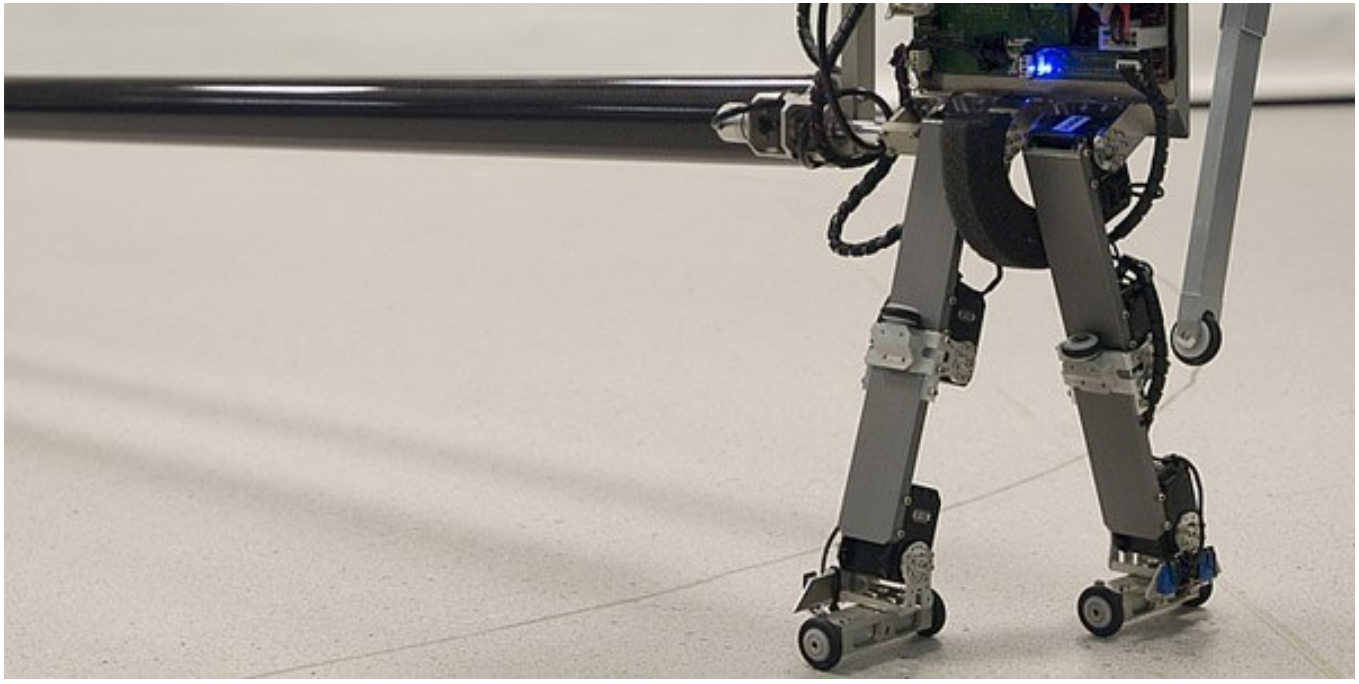
$\Rightarrow$ Soluţia 1: Legea de control reprezentată implicit

- Aproximatorul funcţiei Q trebuie să garanteze **soluţie eficientă pentru** max
- Ex. acţiuni discrete & funcţii de bază în x



Învăţarea Q aprox.: demo mers robotic (E. Schuitema)

Aproximator: **tile coding**



SARSA aproximată

Reamintim SARSA clasică:

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

Aproximare: similar cu învăţarea Q

- actualizăm parametri
- bazat pe gradientul funcţiei Q
- şi diferenţa temporală aproximată

$$\theta_{k+1} = \theta_k + \alpha_k \frac{\partial}{\partial \theta} \widehat{Q}(x_k, u_k; \theta_k) \cdot \left[r_{k+1} + \gamma \widehat{Q}(x_{k+1}, u_{k+1}; \theta_k) - \widehat{Q}(x_k, u_k; \theta_k) \right]$$

SARSA

SARSA aproximată

for fiecare traiectorie **do**
  iniţializează x_0
  alege u_0 (ex. ε-greedy din $Q(x_0, \cdot; \theta_0)$)
  repeat la fiecare pas k
    aplică u_k, măsoară x_{k+1}, primeşte r_{k+1}
    alege u_{k+1} (ex. ε-greedy din $Q(x_{k+1}, \cdot; \theta_k)$)

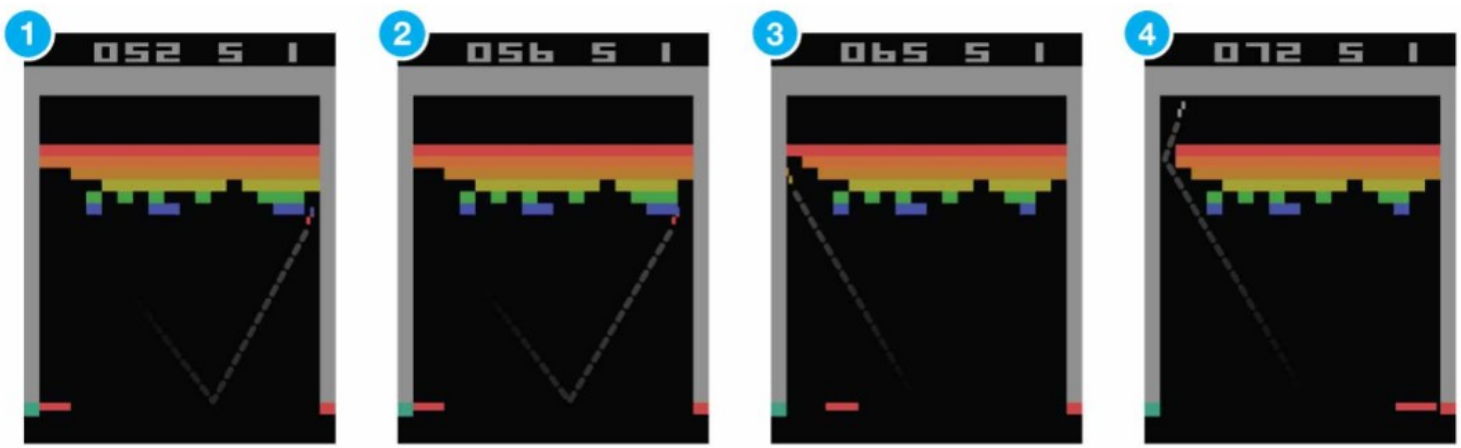
$$\theta_{k+1} = \theta_k + \alpha_k \frac{\partial}{\partial \theta} \widehat{Q}(x_k, u_k; \theta_k) \cdot \left[r_{k+1} + \gamma \widehat{Q}(x_{k+1}, u_{k+1}; \theta_k) - \widehat{Q}(x_k, u_k; \theta_k) \right]$$

  until traiectoria terminată
end for

Învăţarea Q cu “deep neural networks”

- Funcţia Q reprezentată via o reţea neuronală $Q(x_{k+1}, \cdot; \theta_k)$
- Reţea neuronală “deep” cu multe nivele, cu structuri şi funcţii de activare specifice
- Reţeaua antrenată pentru a minimiza diferenţa temporală, similar cu algoritmul standard
- Antrenare pe mini-batch-uri de tranziţii, similar cu iteraţia Q bazată pe date $\Rightarrow$ algoritmul combină RL online şi offline

(DeepMind, Human-level control through deep reinforcement learning, Nature 2015)



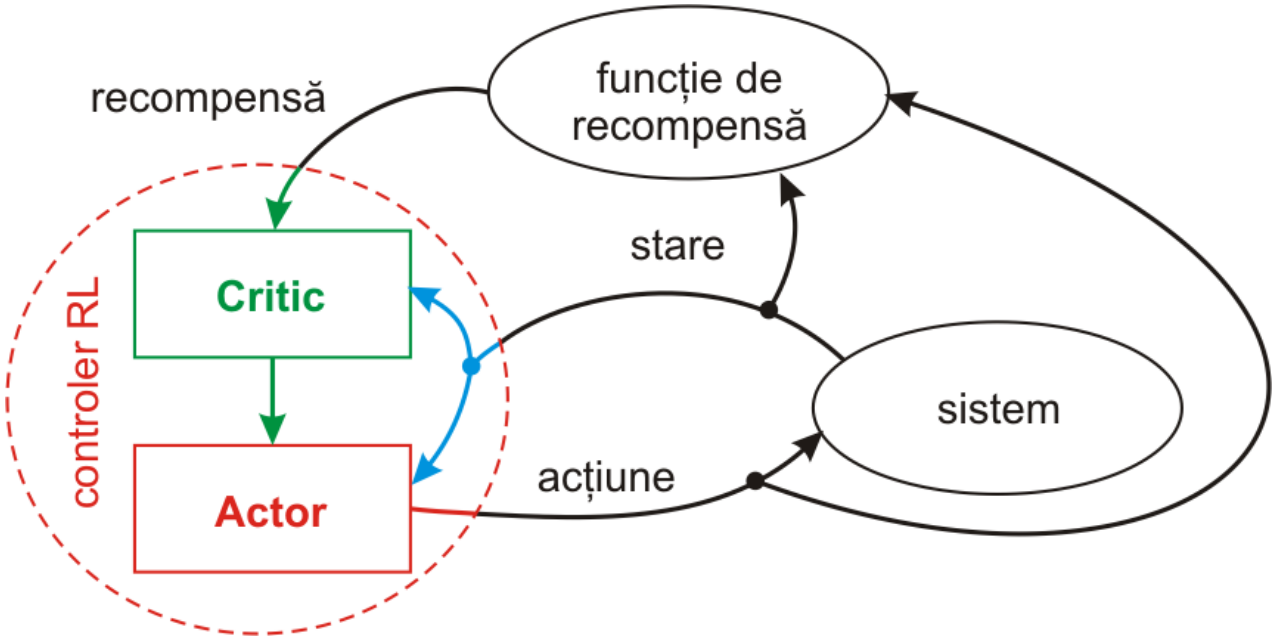
Lege de control explicită

- Soluţia 2: Legea de control **aproximată explicit**: $\widehat{h}(x;\vartheta)$

Avantaje:

- **Acţiuni continue** mai uşor de folosit
- Reprezentarea poate include mai uşor **cunoştinţe expert**

Schema actor-critic



- Actor: legea de control $\widehat{h}(x;\vartheta)$
- Critic: funcţia V, $\widehat{V}(x;\theta)$

Actualizarea criticului

- Gradient pe diferenţa temporală:

$$\theta \leftarrow \theta + \alpha_k^{\text{critic}} \frac{\partial}{\partial \theta} \widehat{V}(x_k;\theta)[r_{k+1} + \widehat{V}(x_{k+1};\theta) - \widehat{V}(x_k;\theta)]$$

$$= \theta + \alpha_k^{\text{critic}} \frac{\partial}{\partial \theta} \widehat{V}(x_k;\theta)\Delta_k$$

- Provine din ecuaţia Bellman pentru V^h:

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

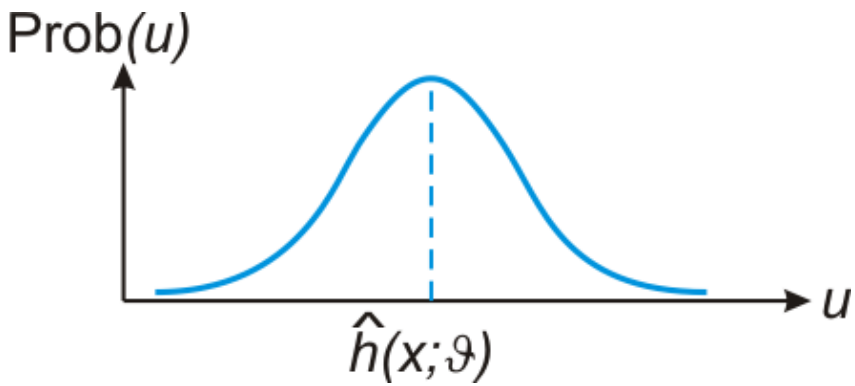
$\Rightarrow$ De fapt, evaluarea legii de control

Explorare-exploatare

- RL online $\Rightarrow$ actor-critic trebuie să exploreze
- Ex. explorare Gaussiană

$$u_k = \widehat{h}(x_k; \vartheta) + u_{\text{explor}}$$

unde termenul explorator u_{explor} Gaussian cu medie 0



Actualizarea actorului

$$\vartheta \leftarrow \vartheta + \alpha_k^{\text{actor}} \frac{\partial}{\partial \vartheta} \widehat{h}(x_k; \vartheta)[u_k - \widehat{h}(x_k; \vartheta)]\Delta_k$$

Intuiţie:

- Dacă $\Delta_k > 0$, adică $r_{k+1} + \widehat{V}(x_{k+1}; \theta) > \widehat{V}(x_k; \theta)$, performanţa mai bună decât cea aşteptată
  $\Rightarrow$ **apropie** de acţiunea u_k exploratoare
- Dacă $\Delta_k < 0$, performanţa mai proastă
  $\Rightarrow$ **îndepărtează** de u_k

Algoritmul actor-critic

Actor-critic

for fiecare traiectorie **do**
  iniţializează x_0
  repeat la fiecare pas k
    $u_k \leftarrow \widehat{h}(x_k; \vartheta)+$ explorare
    aplică u_k, măsoară x_{k+1}, primeşte r_{k+1}
    $\Delta_k \leftarrow r_{k+1} + \widehat{V}(x_{k+1}; \theta) - \widehat{V}(x_k; \theta)$
    $\theta \leftarrow \theta + \alpha_k^{\text{critic}} \frac{\partial}{\partial \theta} \widehat{V}(x_k; \theta) \Delta_k$
    $\vartheta \leftarrow \vartheta + \alpha_k^{\text{actor}} \frac{\partial}{\partial \vartheta} \widehat{h}(x_k; \vartheta)[u_k - \widehat{h}(x_k; \vartheta)] \Delta_k$
  until traiectoria terminată
end for

- De notat: rate de învăţare diferite actor & critic

Actor-critic – iteraţie legea de control optimistă

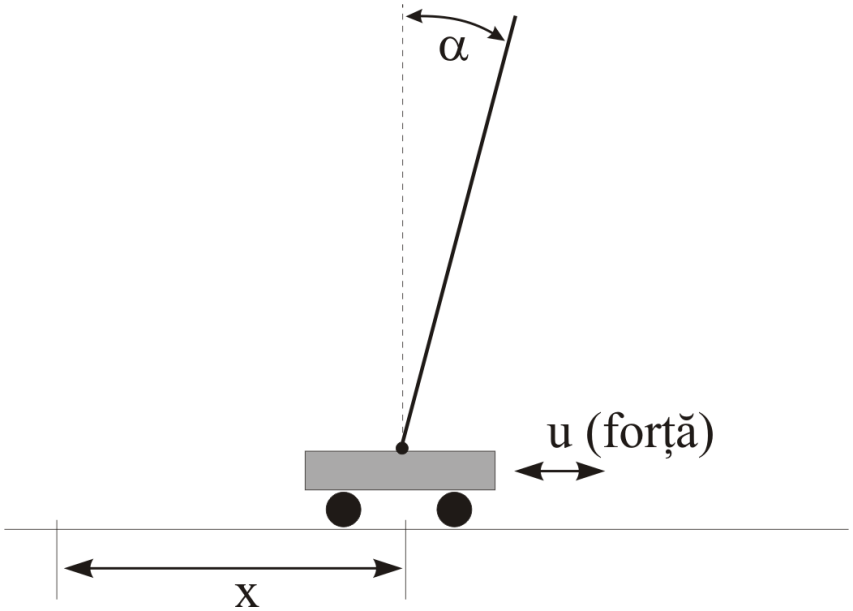
- Reamintim: Actualizare critic
  = 1 pas de evaluare a legii de control
- Actualizare actor
  = îmbunătăţire incrementală a legii de control

$\Rightarrow$ Actor-critic $\approx$ **iteraţie pe legea de control**

- Actualizările alternează la fiecare tranziţie

$\Rightarrow$ Actor-critic $\approx$ iter. legea de control **optimistă**

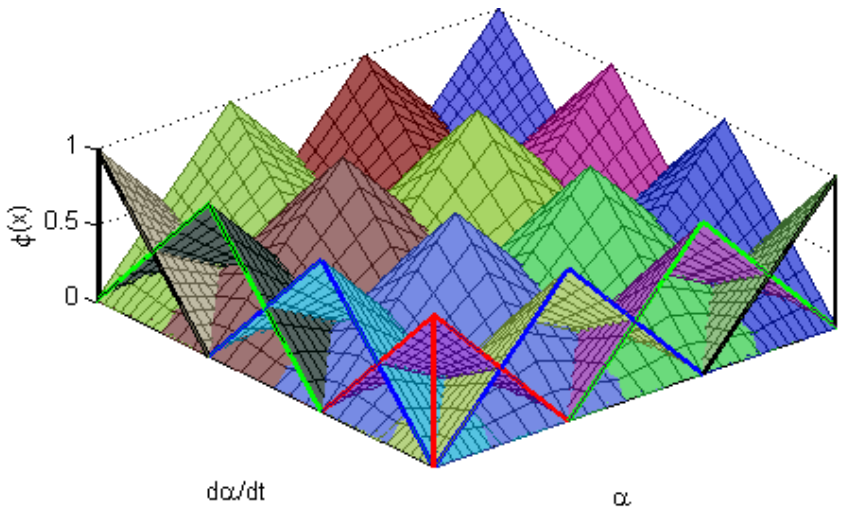
Exemplu: Pendulul inversat cu cărucior



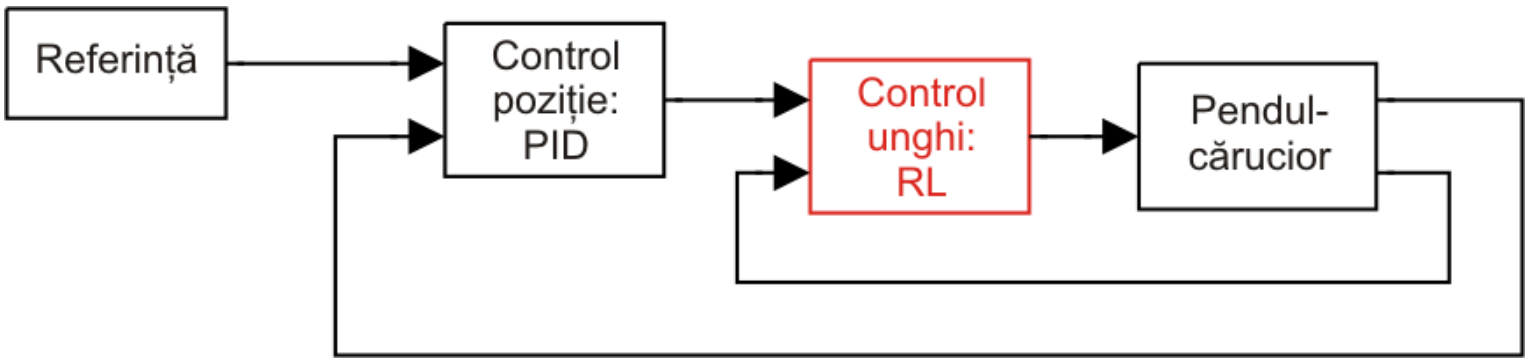
- Forţa transmisă prin accelerarea căruciorului
- Obiectiv: căruciorul la poziţie referinţă, menţinând pendulul vertical
- Nu este nevoie de swingup

Pendul cu cărucior: Aproximator

Atât actor cât şi critic: interpolare (aproximare “fuzzy”)

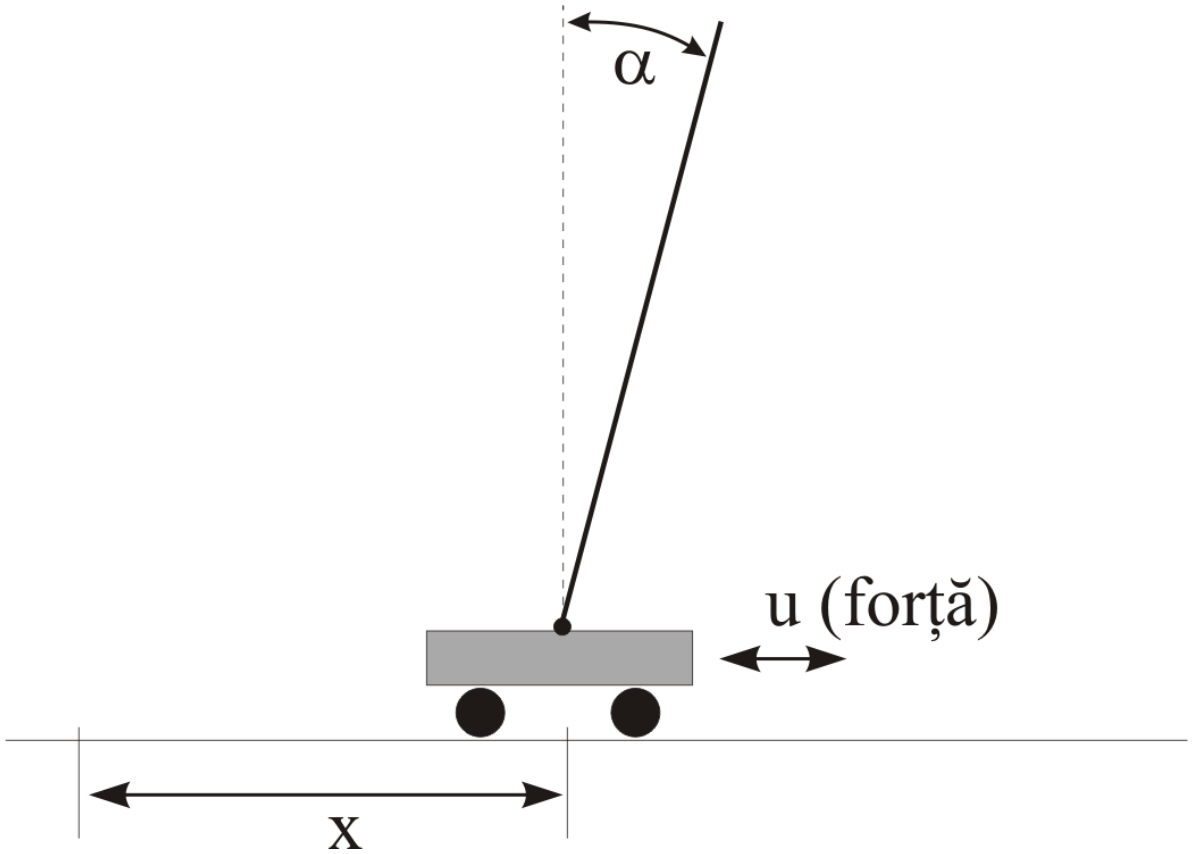


Schema de control



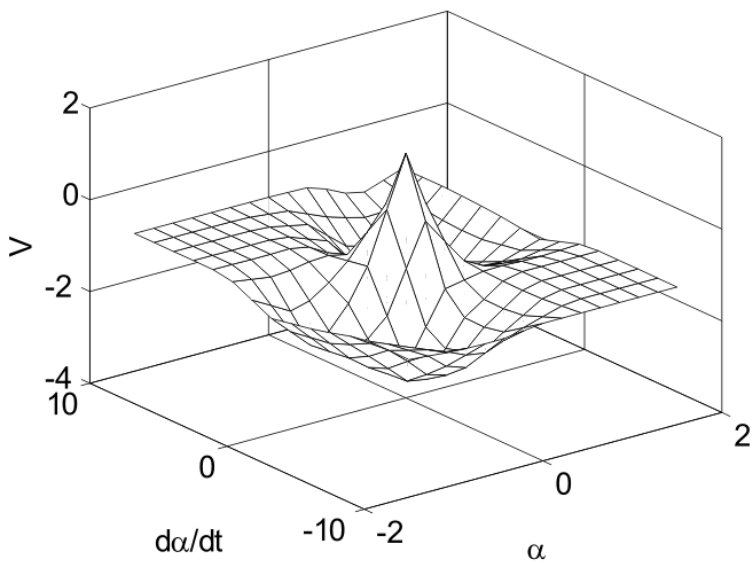
- Bucla externă, control poziţie: PID classic
- Bucla internă, control unghi: **actor-critic**

Pendulul cu cărucior: demo

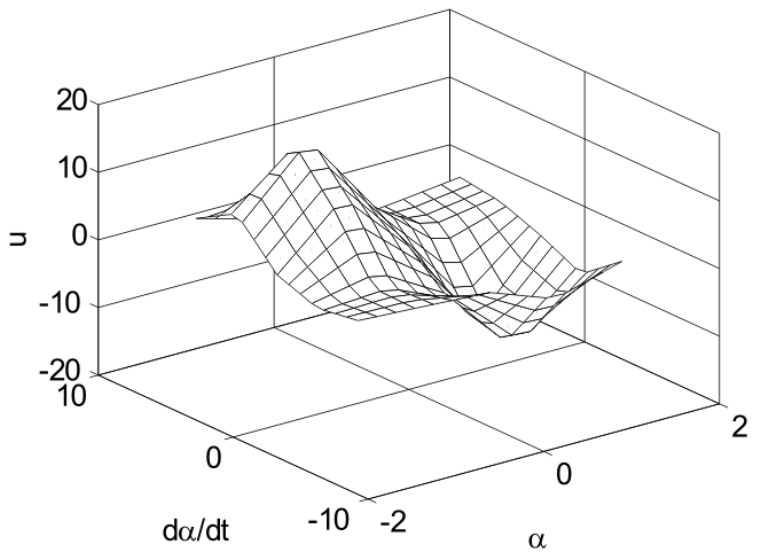


Rezultate

Suprafaţa criticului

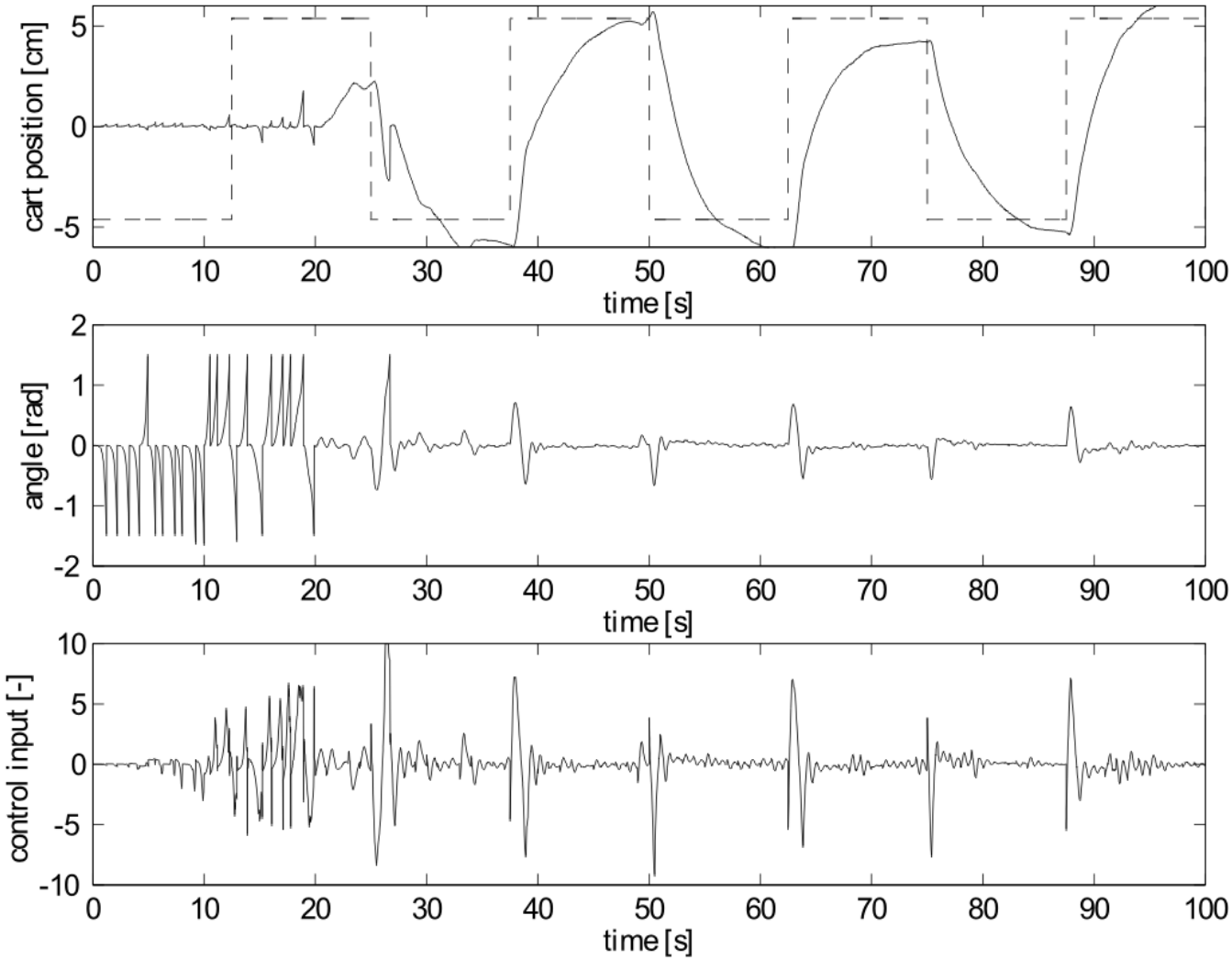


Suprafaţa actorului



Rezultate (continuare)

Traiectorie de-a lungul învăţării



Reamintim: LSPI

Iteraţia pe legea de control CMMP (LSPI)

date fiind (x_s, u_s, r_s, x'_s), $s = 1, \ldots, n_s$
repeat la fiecare iteraţie
Evaluarea legii de control:
$A \leftarrow 0,\ B \leftarrow 0,\ b \leftarrow 0$
for $s = 1, \ldots, n_s$ **do**

actualizează A, B, b folosind (x_s, u_s, r_s, x'_s)

end for
rezolvă $A\theta = \gamma B\theta + b$ găsind θ
Îmbunătăţirea legii de control: $h(x) \leftarrow \arg\max_u \widehat{Q}(x, u; \theta)$
until terminare

LSPI online

- Nu există date în avans, colectează **online**, interactiv
- Îmbunătăţeşte legea de control **optimist**
- De notat: A, B, b refolosite chiar dacă h se schimbă!

$A \leftarrow 0,\ B \leftarrow 0,\ b \leftarrow 0$; iniţializează $h(x)$
for fiecare traiectorie **do**
  repeat fiecare pas k
    aplică u_k, măsoară x_{k+1}, primeşte r_{k+1}
    actualizează A, B, b folosind $(x_k, u_k, r_{k+1}, x_{k+1})$
    if au trecut K tranziţii **then**
      rezolvă $A\theta = \gamma B\theta + b$ găsind θ
      Îmbunătăţeşte $h(x) \leftarrow \arg\max_u \widehat{Q}(x, u; \theta)$
    end if
  until traiectoria terminată
end for

LSPI: Explorare-exploatare

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

- Explorare necesară, ex. ε-greedy:

$$u_k = \begin{cases} h(x_k) & \text{cu prob. } 1 - \varepsilon_k \\ \text{o acţiune aleatoare} & \text{cu prob. } \varepsilon_k \end{cases}$$

Exemplu: Pendul inversat

- $x = [\text{unghi } \alpha, \text{ viteză } \dot{\alpha}]^\top$
- u = voltaj
- $\rho(x, u) = -x^\top \begin{bmatrix} 5 & 0 \\ 0 & 0.1 \end{bmatrix} x - u^\top 1 u$
- Factor de discount $\gamma = 0.98$

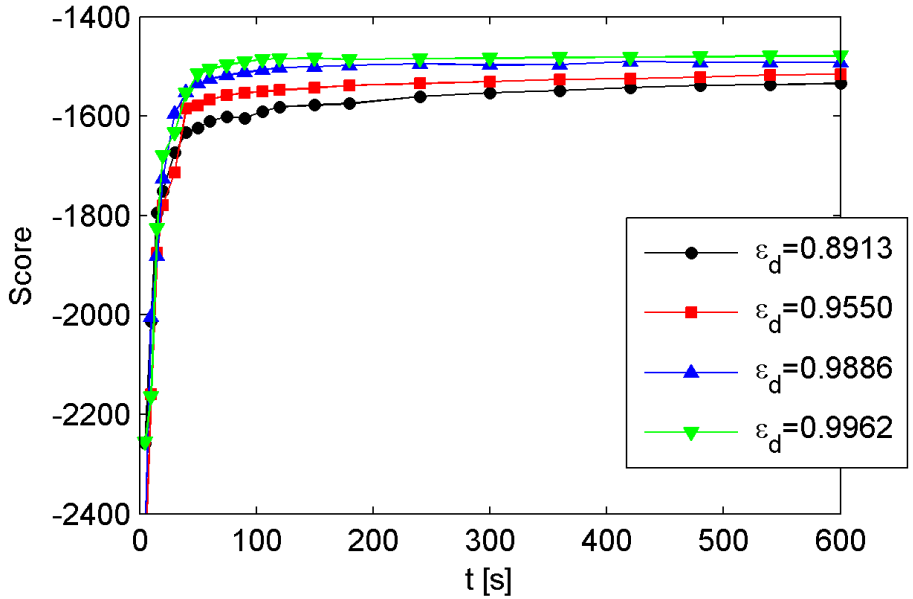
- Obiectiv: stabilizează orientat în sus
- Putere insuficientă $\Rightarrow$ balansează înainte & înapoi

▸ Replay

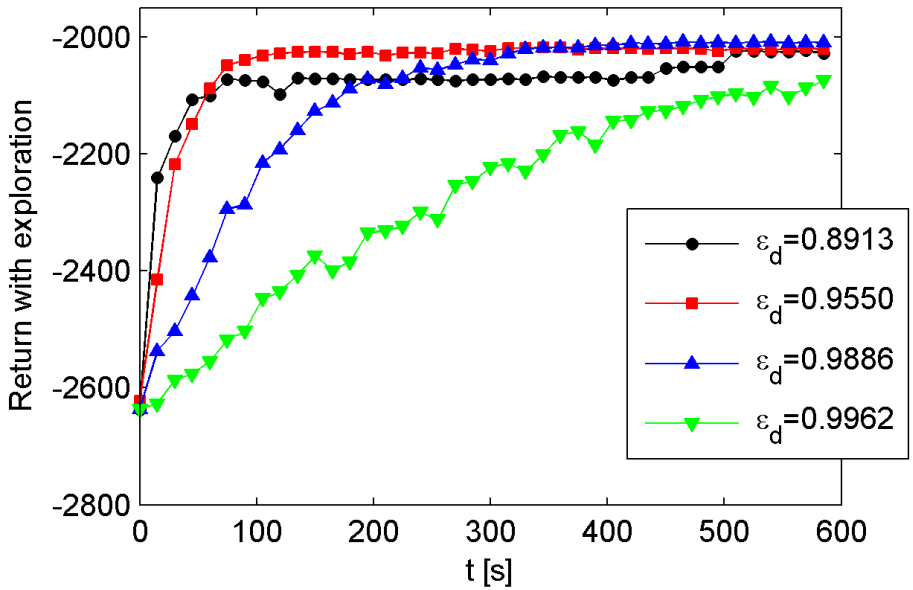
Pendul inversat: LSPI online, demo

Câtă explorare?

Scorul legii de control învăţate (explorarea oprită)



Performanţa **în timpul** învăţării (cu explorare)



$\Rightarrow$ Trebuie găsit un **echilibru**!

Comparaţie între algoritmii prezentaţi

Convergenţă

- Învăţarea Q, SARSA, actor-critic:
  convergenţă garantată pentru **variante modificate**
- LSPI online: nu există garanţii

Complexitate

- Per iteraţie, învăţarea Q, SARSA, actor-critic $<$ LSPI online

Reglaj parametri

- Explorarea **crucială** pentru toate metodele
- Ratele de învăţare α delicate
  (actor-critic are **două** rate de învăţare)
- LSPI online: K – mai uşor de acordat

Motivare

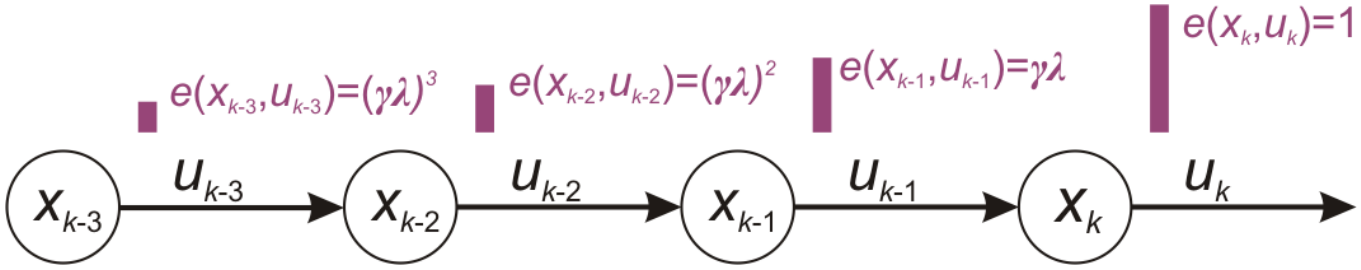
- Dezavantaj metode TD aproximate:
  ca şi în cazul discret, învaţă încet

$\Rightarrow$ Timp, uzură crescute, profituri scăzute

- **Accelerarea învăţării** este necesară

Urme de eligibilitate

- Reamintim cazul discret – urmă $e(x, u)$ de-a lungul traiectoriei:



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

- Când x, u continue, $e(x, u)$ nu se poate reprezenta direct

Urme de eligibilitate în cazul aproximat

- Idee: în actualizarea cu gradient, de ex. învăţarea Q:

$$\theta_{k+1} = \theta_k + \alpha_k \frac{\partial}{\partial \theta} \widehat{Q}(x_k, u_k; \theta_k) \cdot \left[r_{k+1} + \gamma \max_{u'} \widehat{Q}(x_{k+1}, u'; \theta_k) - \widehat{Q}(x_k, u_k; \theta_k) \right]$$

- ...tratăm gradientul $\frac{\partial}{\partial \theta_i} \widehat{Q}(x_k, u_k; \theta_k)$ ca pe o **contribuţie** a parametrului *i* la actualizarea curentă
- Luăm în considerare **contribuţia cumulativă** (scăzând cu $\gamma\lambda$) până la pasul curent:

$$\theta_{k+1} = \theta_k + \alpha_k e_{k+1} \cdot \left[r_{k+1} + \gamma \max_{u'} \widehat{Q}(x_{k+1}, u'; \theta_k) - \widehat{Q}(x_k, u_k; \theta_k) \right]$$

$$e_{k+1} = \sum_{\ell=0}^{k} (\gamma\lambda)^{k-\ell} \frac{\partial}{\partial \theta} \widehat{Q}(x_\ell, u_\ell; \theta_\ell)$$

Urme de eligibilitate în învăţarea Q aproximată

Implementare iterativă în învăţarea Q:

Învăţarea Q(λ) aproximată

for fiecare traiectorie **do**

iniţializează x_0, $e_0 = [0, \ldots, 0]^\top$

repeat la fiecare pas k

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

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

actualizează $e_{k+1} = (\gamma\lambda)e_k + \frac{\partial}{\partial\theta}\widehat{Q}(x_k, u_k; \theta_k)$

$\theta_{k+1} = \theta_k + \alpha_k e_{k+1} \cdot$

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

until traiectoria terminată

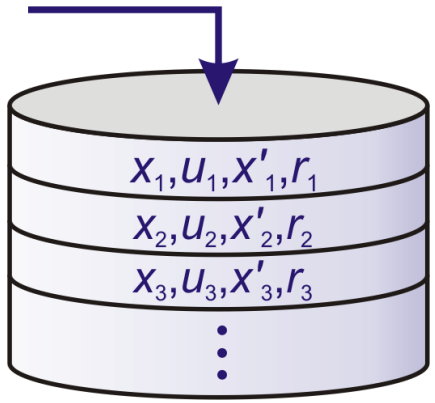
end for

Urme de eligibilitate în alţi algoritmi

- Urmele de eligibilitate se pot adapta simplu la SARSA şi învăţarea criticului din actor-critic
- Ideea se extinde şi la LSPI, dar mai complicat

Reluarea experienţei

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



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

SARSA aproximată cu reluarea experienţei

SARSA aproximată cu reluarea experienţei

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

$$\theta_{k+1} = \theta_k + \alpha_k \frac{\partial}{\partial \theta} \widehat{Q}(x_k, u_k; \theta_k) \cdot \left[r_{k+1} + \gamma \widehat{Q}(x_{k+1}, u_{k+1}; \theta_k) - \widehat{Q}(x_k, u_k; \theta_k) \right]$$

    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

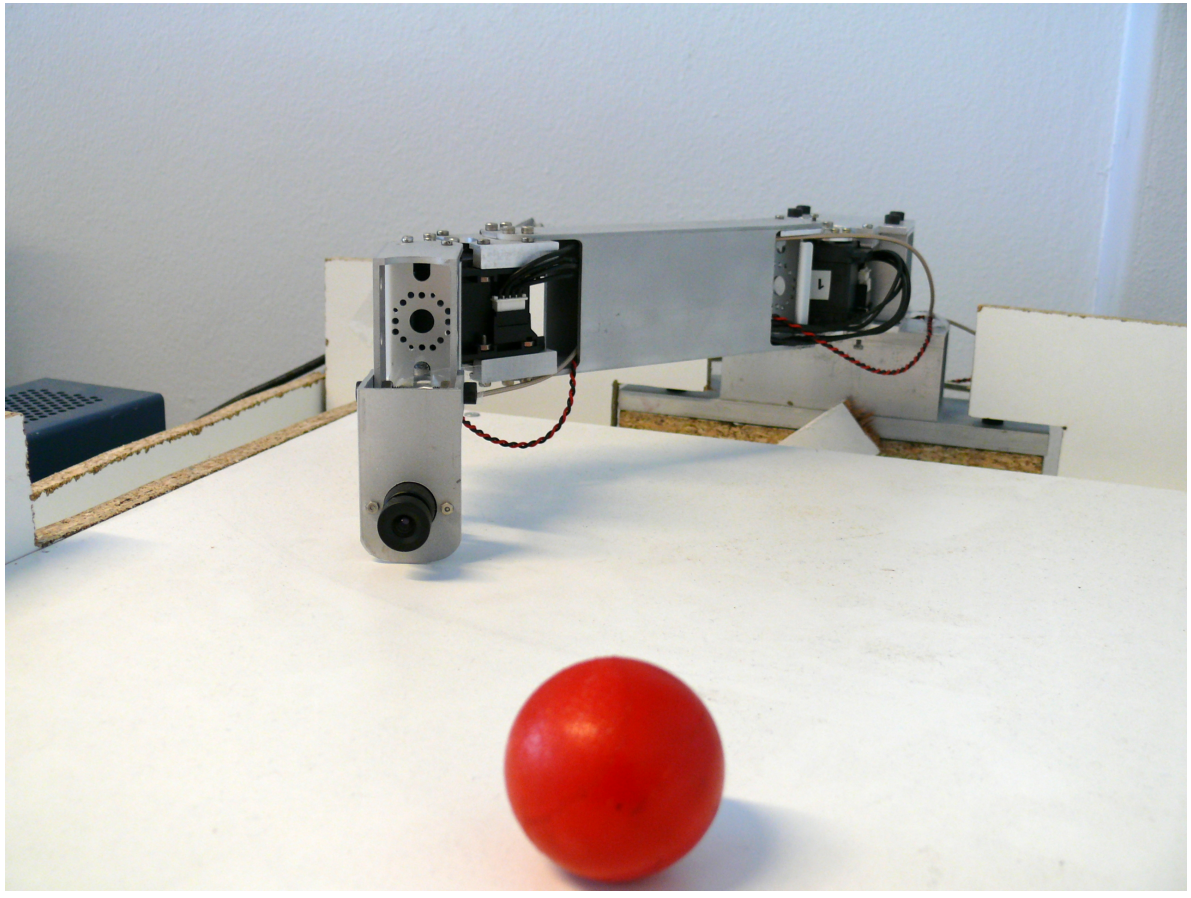
preia o tranziţie (x, u, x', r) din baza de date

$$\theta_{k+1} = \theta_k + \alpha_k \frac{\partial}{\partial \theta} \widehat{Q}(x_k, u_k; \theta_k) \cdot \left[r_{k+1} + \gamma \widehat{Q}(x_{k+1}, u_{k+1}; \theta_k) - \widehat{Q}(x_k, u_k; \theta_k) \right]$$

end loop

Pendul: RL cu reluarea exp., demo (Sander Adam)

Robot portar: RL cu reluarea exp., demo (S. Adam)



Conexiuni: Control optimal

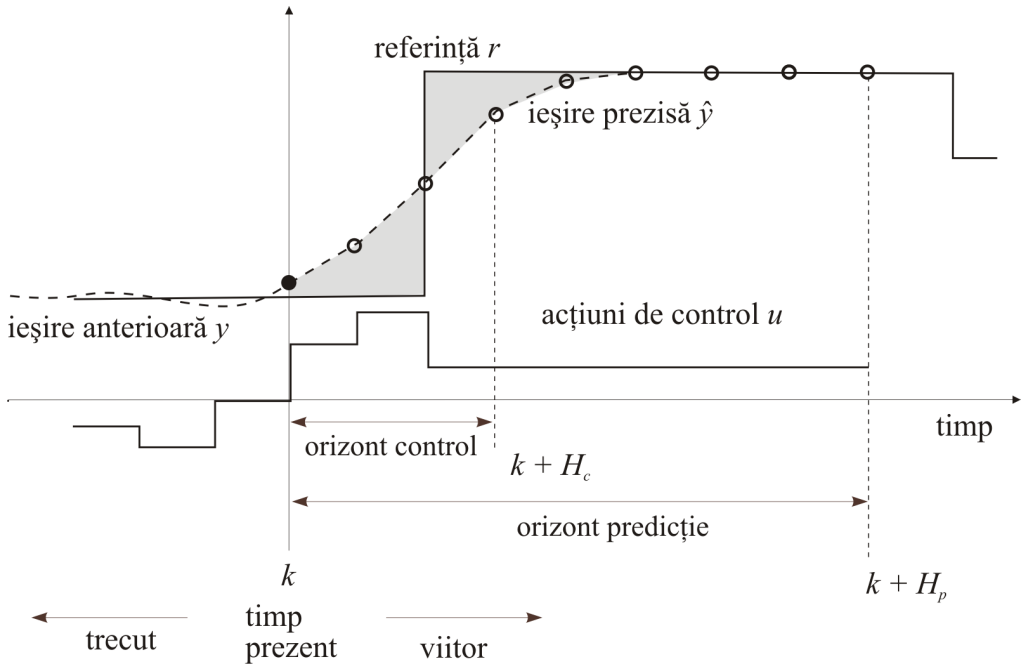
- Controlează un sistem minimizând costul J
- Bazat pe model
- Posibil în timp continuu, orizont finit

RL, DP **sunt control optimal**!

- Controlează un sistem maximizând returnul $R^h(x)$
- În timp discret, orizont infinit
- RL **fără model**, bazat pe date/interacţiune

Conexiuni: Control predictiv

- Bazat pe model, clasic liniar
- Principiu de bază: **receding horizon**



Controlul predictiv este şi el o variantă de control optimal, bazată pe model

Procese de fabricaţie

Focus curs: Controlul sistemelor generice

Procese de fabricaţie

- Legate de logistică şi *operations research*
- Variabile **discrete** sunt importante
  posibil combinate cu variabile continue: hibrid
  Aproximarea rămâne esenţială
- Tranziţii de stare cu **durată variabilă**:
  procese de decizie semi-Markov

DP & RL din perspectiva operations research:

Warren B. Powell, *Approximate Dynamic Programming: Solving the curses of dimensionality*, ed. 2, Wiley, 2011.

Probleme deschise

RL & DP **în curs de dezvoltare**

Probleme deschise:

- Garanţii de siguranţă şi stabilitate
- Stări şi acţiuni cu dimensionalitate mare
- Stări care nu pot fi măsurate
- Strategii de explorare
- Sisteme multiagent