Control prin învăţare

Master ICAF, An 1 Sem 2

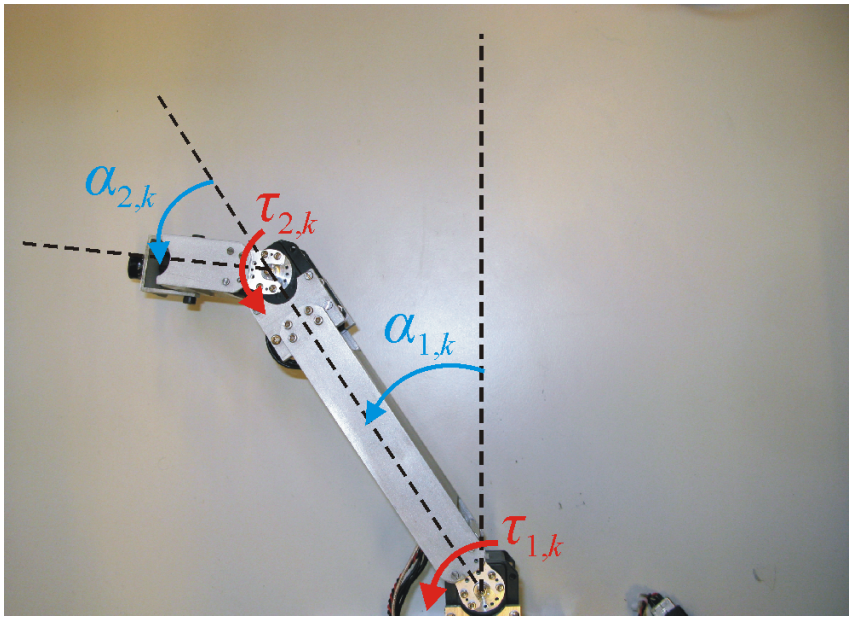
Lucian Buşoniu

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

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

Partea V în plan

- Problema de învăţare prin recompensă
- Soluţia optimală
- Programarea dinamică exactă
- Învăţarea prin recompensă exactă
- Tehnici de aproximare
- Programarea dinamică cu aproximare (var. continue)
- **Învăţarea prin recompensă cu aproximare (var. continue)**

Există mulţi algoritmi, doar 2 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

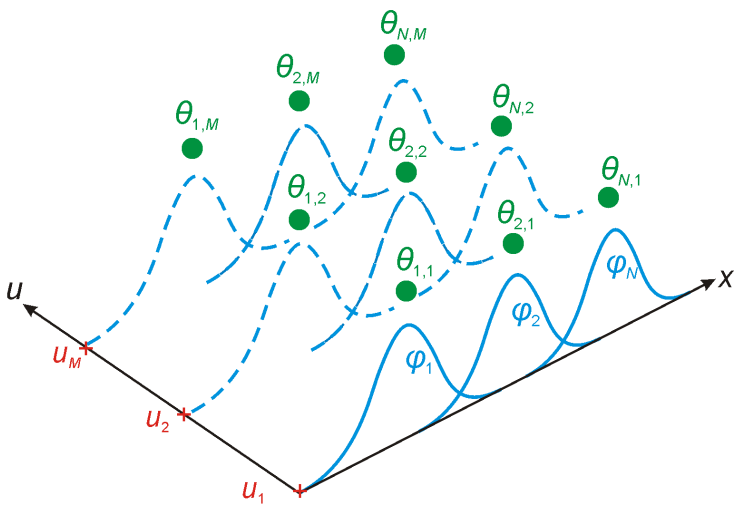
Maximizare în învăţarea Q aproximată

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

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

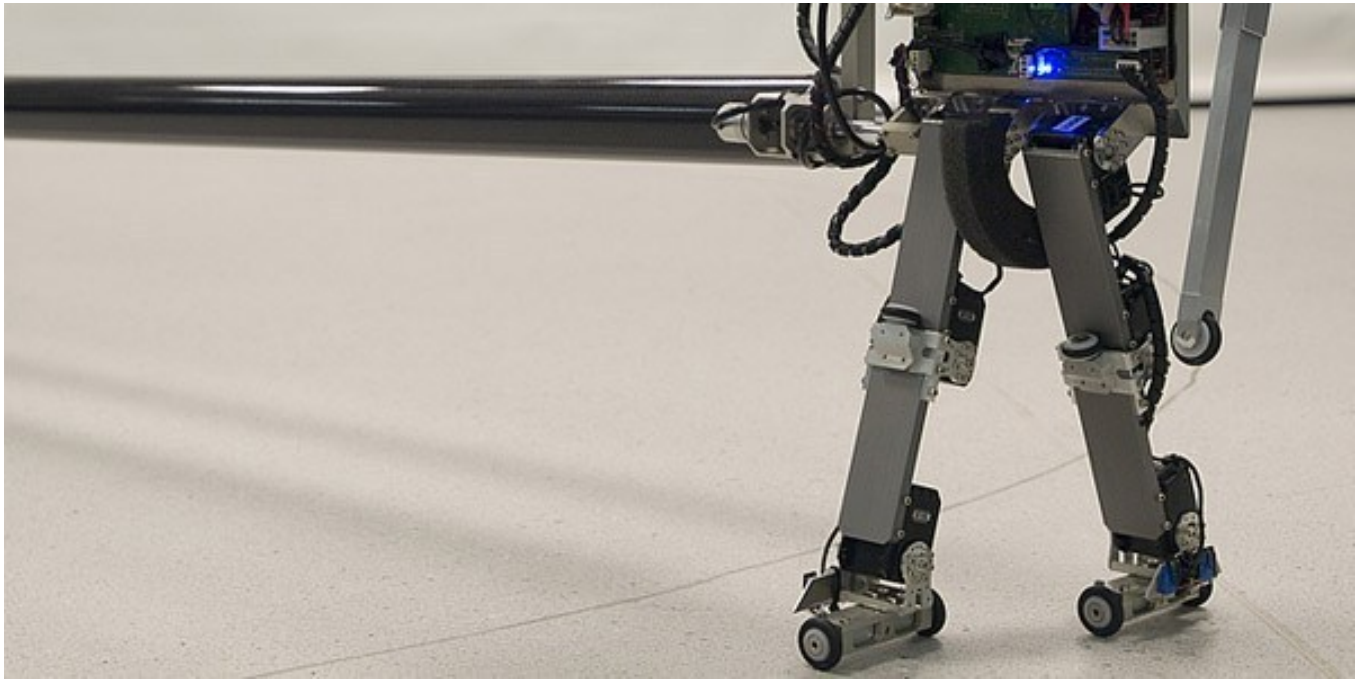
$\Rightarrow$ Soluţia 1: Legea de control reprezentată implicit

- 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

Discuţie învăţarea Q, SARSA

Convergenţă

- Convergenţă garantată pentru **variante modificate**
- Complexitate scăzută
- Explorarea **crucială** pentru toate metodele
- Rata de învăţare α delicată

Motivare

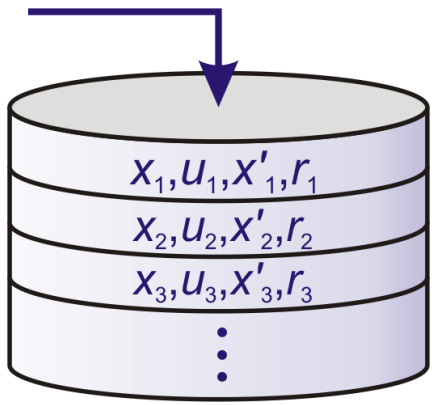
- Dezavantaj metode TD aproximate:
  ca şi în cazul discret, învaţă încet

$\Rightarrow$ Timp, uzură crescute, profituri scăzute

- **Accelerarea învăţării** este necesară

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

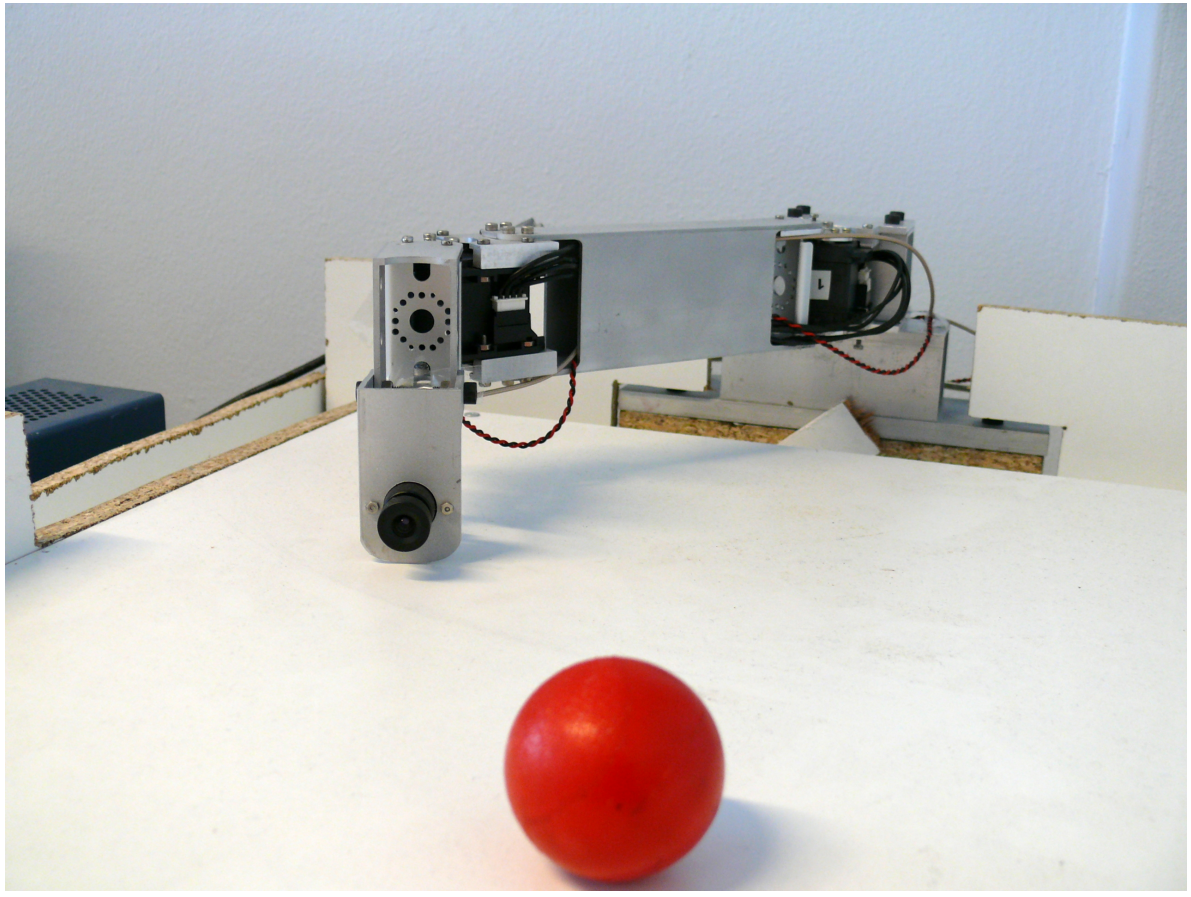
preia o tranziţie (x, u, x', r) din baza de date

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

end loop

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

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



Conexiuni: Control optimal

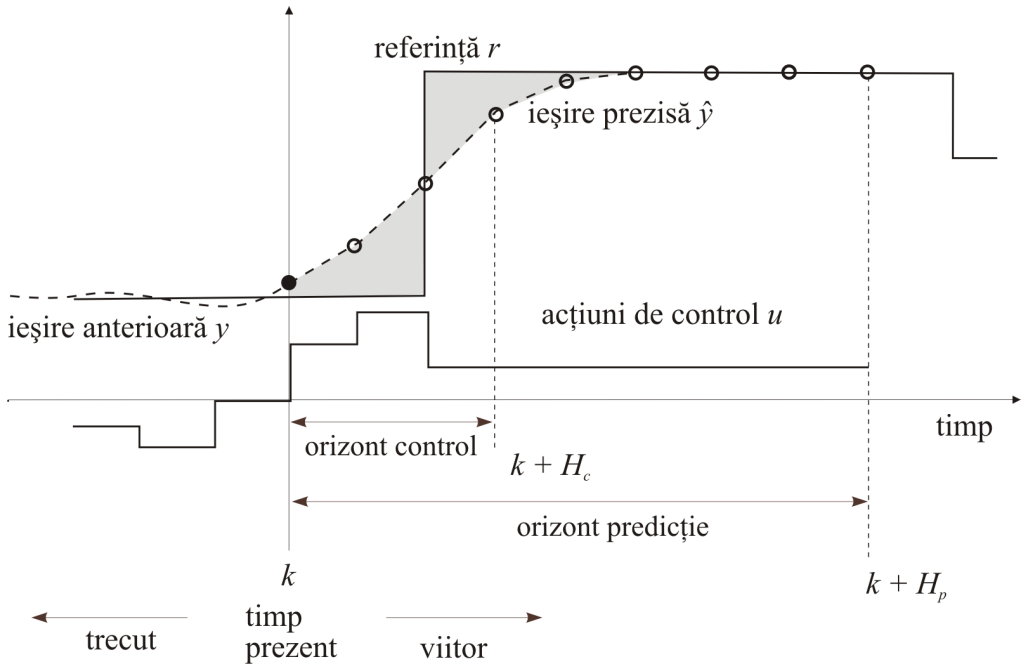
- Controlează un sistem minimizând costul J
- Bazat pe model
- Posibil în timp continuu, orizont finit

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

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

Învăţarea prin recompensă cu reţele adânci – Deep RL

O modalitate de a trata stările cu dimensionalitate mare *când acestea sunt imagini* (sau pot fi asimilate unor imagini)

Deep Q-network, DQN

- 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)