Control prin învăţare

Master ICAF, An 1 Sem 2

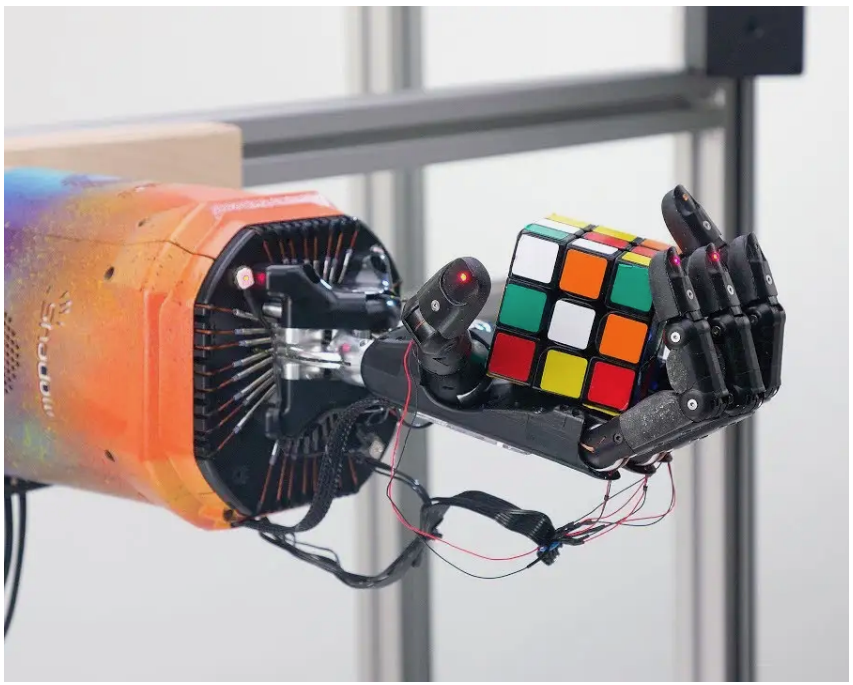
Lucian Buşoniu, Ştefan Pîrje

Partea V

Învăţarea prin recompensă online cu aproximare

Recap: Nevoia de aproximare

- În aplicaţii reale de control, x, u **continue**! (sau discrete cu multe valori)



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

Algoritmi exemplificaţi:

- iteraţia Q cu interpolare
- 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)**
- Reţele neurale profunde
- Învăţarea prin recompensă cu reţele neurale profunde

Partea V în plan: Categorii de algoritmi

După utilizarea unui model:

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

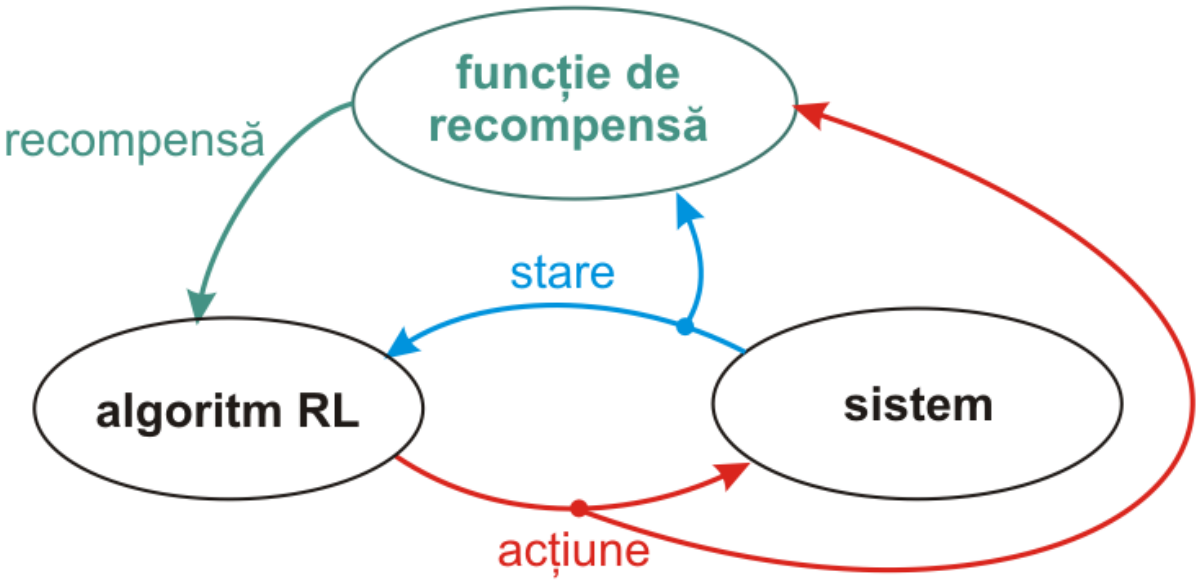
După nivelul de interacţiune:

- 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)

Principiul RL



- Urmărim acum cu adevărat procedura RL de interacţiune online
- Există mulţi algoritmi; discutăm câţiva

Conţinut partea 5

Reamintim SARSA clasică

SARSA cu ε-greedy

for fiecare traiectorie **do**

iniţializează 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

Reamintim: Derivarea actualizării SARSA (on-policy)

- Actualizare:

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

- $\hat{R}_k$ este un estimat cu bootstrap (folosind ecuaţia Bellman) al return-ului Monte Carlo R_k din (x_k, u_k) pentru legea curentă de control h
- R_k este el însuşi un eşantion al $Q^h(x_k, u_k)$, deci în final rulăm o versiune estimată a actualizării ideale:

$$Q(x_k, u_k) \leftarrow Q(x_k, u_k) + \alpha_k[Q^h(x_k, u_k) - Q(x_k, u_k)]$$

SGD pentru cazul aproximat

- Extindem ideea de mai sus la aproximatorul parametric $\hat{Q}(x, u; \theta)$
- Nu mai putem actualiza direct Q, în schimb actualizăm θ folosind **coborârea pe gradient stohastică (SGD)** pe eroarea de aproximare pătratică:

$$\begin{aligned}\theta_{k+1} &= \theta_k - \frac{1}{2}\alpha_k \nabla_\theta \left[Q^h(x_k, u_k) - \widehat{Q}(x_k, u_k; \theta_k)\right]^2 \\ &= \theta_k + \alpha_k \nabla_\theta \widehat{Q}(x_k, u_k; \theta_k) \left[Q^h(x_k, u_k) - \widehat{Q}(x_k, u_k; \theta_k)\right]\end{aligned}$$

- Înlocuim $Q^h(x_k, u_k)$ cu eşantionul Monte Carlo R_k, apoi R_k cu estimatul cu bootstrap $\hat{R}_k = r_{k+1} + \gamma \widehat{Q}(x_{k+1}, u_{k+1}; \theta_k)$:

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

Semigradient

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

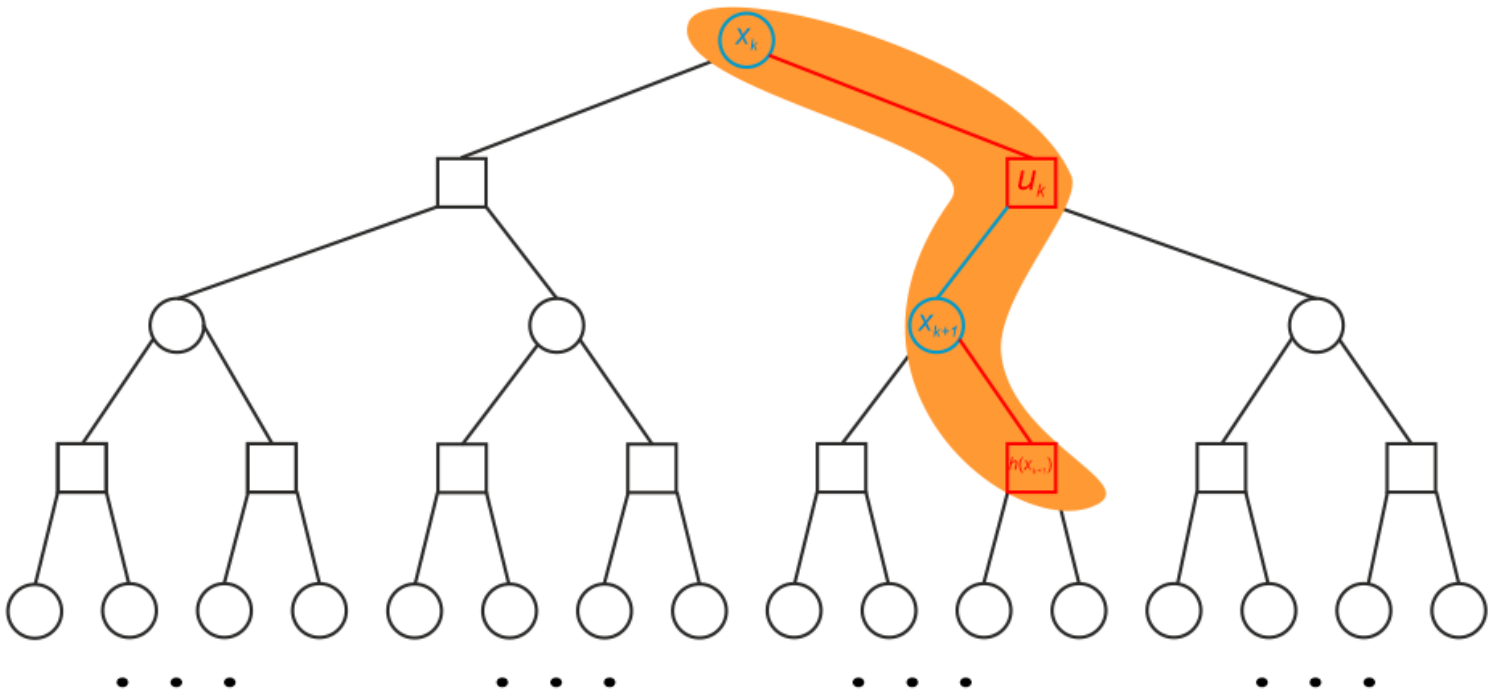
- Actualizarea finală nu este un gradient complet, deoarece $\hat{R}_k$ depinde de θ_k prin $\widehat{Q}(x_{k+1}, u_{k+1}; \theta_k)$, dar doar al doilea termen al erorii este diferenţiat!

$\Rightarrow$ Astfel de metode se numesc de **semigradient**.

- Termenul din paranteza pătrată este o **diferenţă temporală aproximată**.

Ilustraţie

Ilustraţia grafică este similară cazului clasic:



dar acum prezenţa aproximării necesită utilizarea gradientului

Obiectiv şi comparaţie cu metodele fitted

$$\theta_{k+1} \leftarrow \theta_k + \alpha_k \nabla_\theta \widehat{Q}(x_k, u_k; \theta_k) \left[Q^h(x_k, u_k) - \widehat{Q}(x_k, u_k; \theta_k) \right]$$

minimizează următorul obiectiv sub distribuţia generată de eşantioanele (x_k, u_k):

$$\mathcal{L}_{\text{eval}} = \mathrm{E} \left\{ \left[Q^h(x, u) - \widehat{Q}(x, u; \theta) \right]^2 \right\}$$

- Comparăm cu obiectivul "fitted", unde distribuţia este implicit tot cea a eşantioanelor: $\sum_{s=1}^{n_s} \left[\hat{R}_s - \widehat{Q}(x_s, u_s; \theta) \right]^2$
- Dacă $\sum_{k=0}^{\infty} \alpha_k^2$ este finită şi $\sum_{k=0}^{\infty} \alpha_k \to \infty$, SGD converge către un minim local al $\mathcal{L}_{\text{eval}}$.
- Acest lucru rămâne valabil când $Q^h(x_k, u_k)$ este înlocuit cu eşantionul Monte Carlo R_k, dar nu şi pentru estimatul cu bootstrap $\hat{R}_k$, din cauza actualizărilor semigradient!

SARSA aproximată semigradient

SARSA aproximată

for fiecare traiectorie **do**
  iniţializează x_0
  alege u_0 (ex. ε-greedy din $\widehat{Q}(x_0, \cdot; \theta_0)$)
  repeat la fiecare pas k
    aplică u_k, măsoară x_{k+1}, primeşte r_{k+1}
    alege u_{k+1} (ex. ε-greedy din $\widehat{Q}(x_{k+1}, \cdot; \theta_k)$)
    $\theta_{k+1} = \theta_k + \alpha_k \nabla_\theta \widehat{Q}(x_k, u_k; \theta_k) \cdot$
    $\left[r_{k+1} + \gamma \widehat{Q}(x_{k+1}, u_{k+1}; \theta_k) - \widehat{Q}(x_k, u_k; \theta_k) \right]$
  until traiectoria terminată
end for

Explorarea este desigur necesară şi în cazul aproximat.

Actualizarea semigradient pentru învăţarea Q

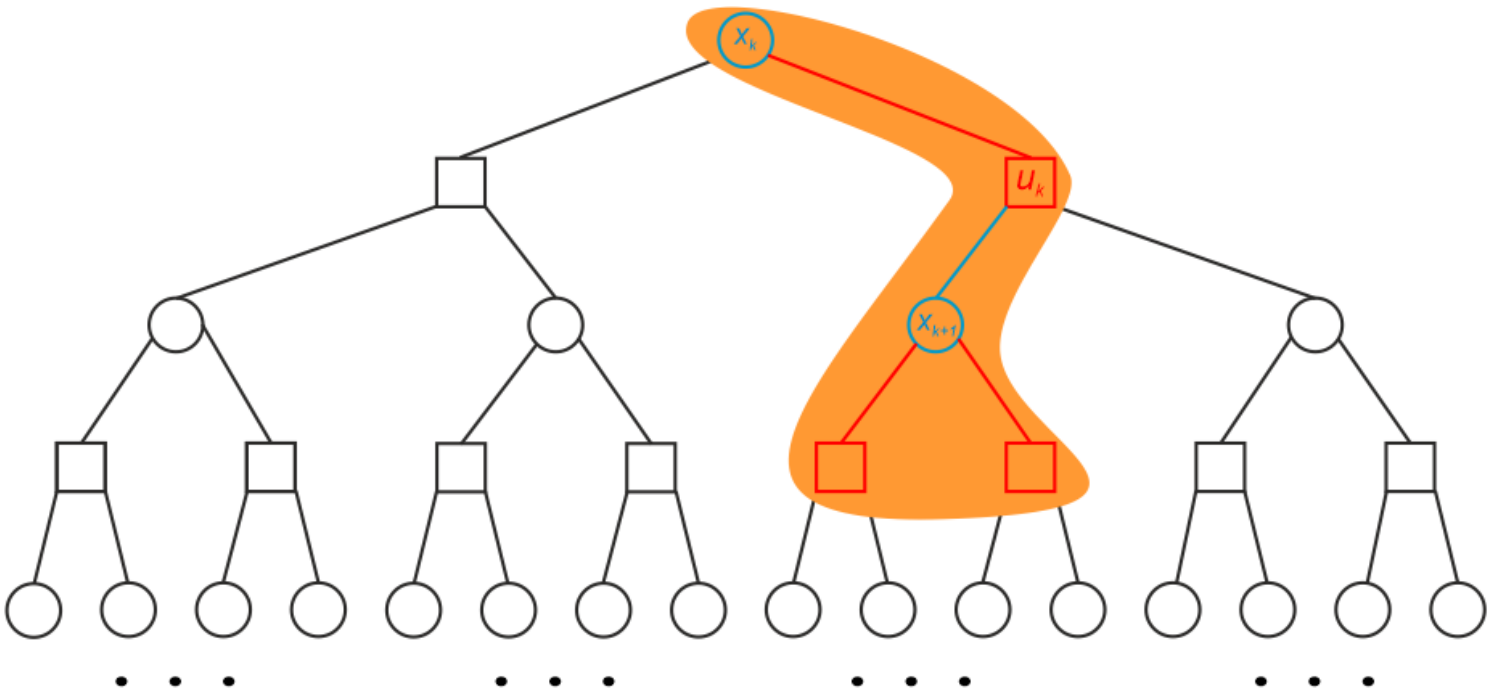
Reamintim actualizare din învăţarea Q clasică:

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

În cazul aproximat, folosim gradientul:

$$\theta_{k+1} = \theta_k - \frac{1}{2}\alpha_k \nabla_\theta \left[Q^*(x_k, u_k) - \widehat{Q}(x_k, u_k; \theta_k)\right]^2$$
$$= \theta_k + \alpha_k \nabla_\theta \widehat{Q}(x_k, u_k; \theta_k) \left[Q^*(x_k, u_k) - \widehat{Q}(x_k, u_k; \theta_k)\right]$$
$$\approx \theta_k + \alpha_k \nabla_\theta \widehat{Q}(x_k, u_k; \theta_k) \cdot$$
$$\left[r_{k+1} + \gamma \max_{u'} \widehat{Q}(x_{k+1}, u'; \theta_k) - \widehat{Q}(x_k, u_k; \theta_k)\right]$$

Ilustraţie



Învăţarea Q aproximată semigradient: algoritm

Învăţarea Q aproximată cu explorare ε-greedy

for fiecare traiectorie **do**
  iniţializează x_0
  repeat la fiecare pas k
    alege u_k (ex. ε-greedy din $\widehat{Q}(x_k, \cdot; \theta_k)$)
    aplică u_k, măsoară x_{k+1}, primeşte r_{k+1}
    $\theta_{k+1} = \theta_k + \alpha_k \nabla_\theta \widehat{Q}(x_k, u_k; \theta_k) \cdot$

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

  until traiectoria terminată
end for

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

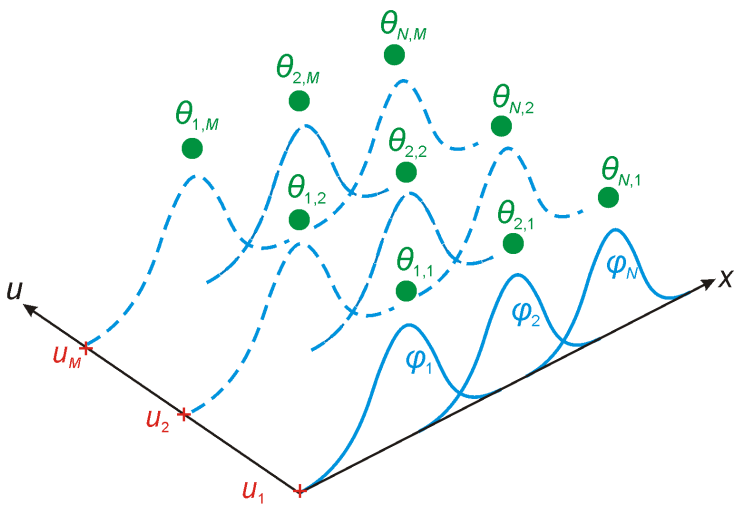
Maximizare în învăţarea Q aproximată

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

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

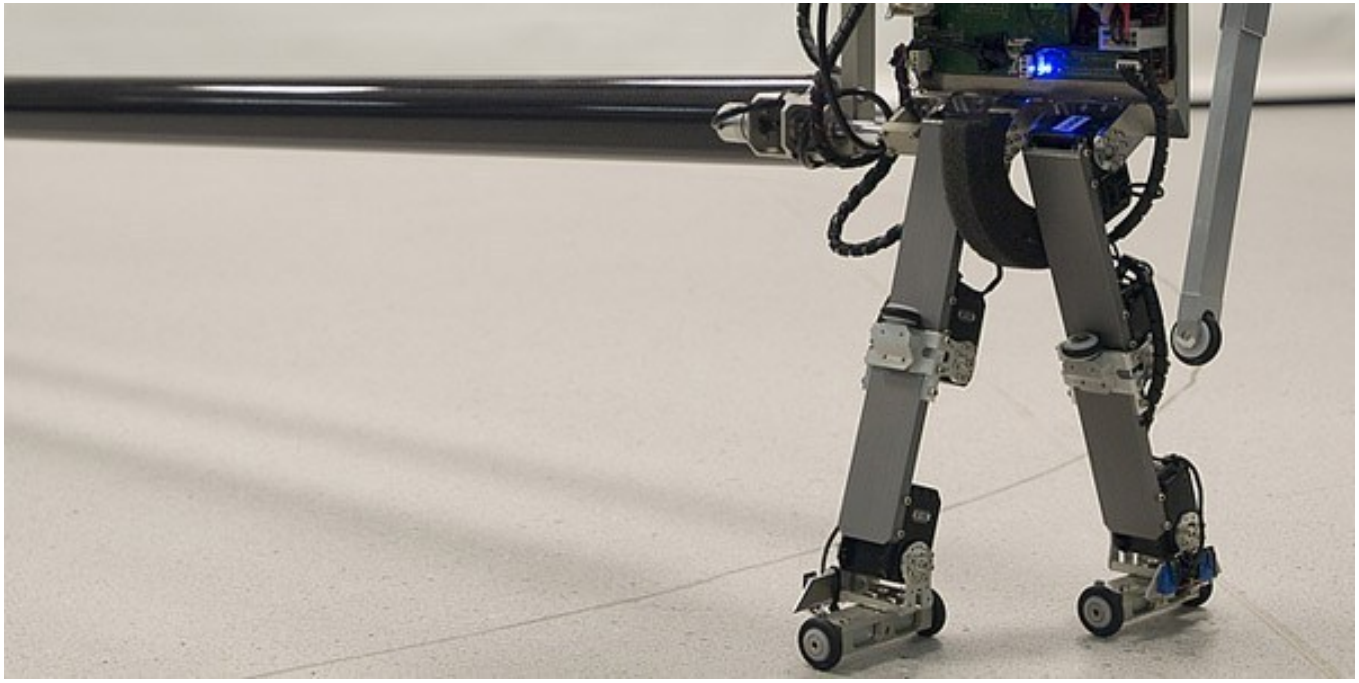
$\Rightarrow$ Soluţia 1: Legea de control reprezentată implicit

- 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**



Discuţie despre metodele TD aproximate

- Complexitate redusă
- Convergenţa este garantată pentru **versiuni modificate**
- Ratele de explorare şi de învăţare trebuie să fie **reglate cu grijă** pentru toate metodele
- La fel ca în cazul clasic, metodele TD aproximate învaţă lent, deci trebuie **accelerate**
- Reluarea experienţei şi return-urile pe *n* paşi sunt aproape direct aplicabile (ultimele pentru SARSA, dar pot fi extinse la Q-learning off-policy)

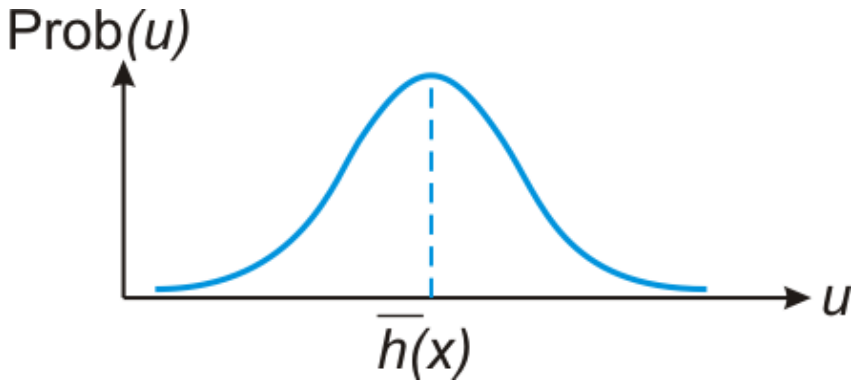
Reprezentarea legii de control

- Tipul 2: Legea de control **aproximată explicit**
- Reamintim avantajele: mai uşor de tratat acţiuni continue şi de integrat cunoştinţe a priori
- De exemplu, lege de control reprezentată folosind atribute (funcţii de bază):

$$\bar{h}(x;\mu) = \sum_{i=1}^{n} \phi_i(x)\mu_i$$

Lege de control cu explorare

- RL online $\Rightarrow$ policy gradient trebuie să exploreze



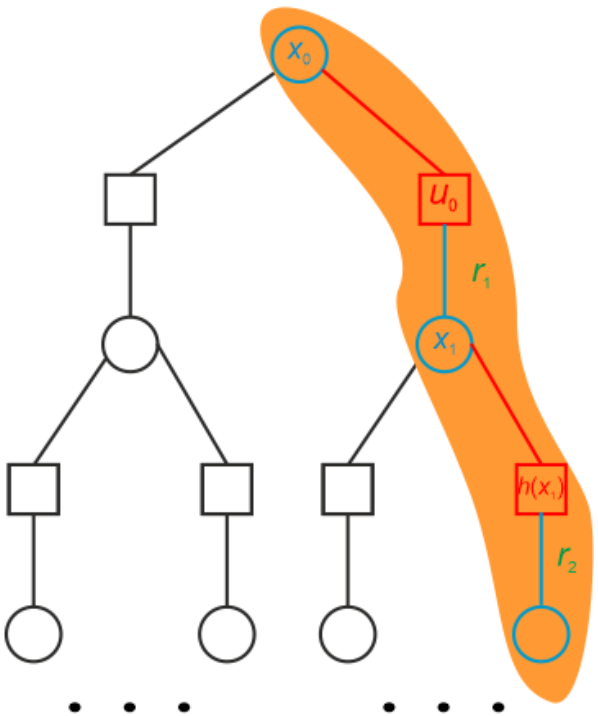
- **Explorarea gaussiană** aplică u în x cu probabilitatea:

$$P(u|x) = \mathcal{N}(\bar{h}(x;\mu),\sigma) =: \widehat{h}(x,u;\vartheta)$$

unde ϑ conţine pe lângă μ şi covarianţele în matricea σ

- Astfel, reprezentăm o lege de control stohastică, incluzând direct explorarea în parametri

Traiectorie



- Traiectoria $\tau := (x_0, u_0, \ldots, x_k, u_k, \ldots)$ generată cu $\widehat{h}$; recompensele rezultate $r_1, \ldots, r_{k-1}, \ldots$
- Considerăm un MDP determinist pentru simplitate. Return-ul de-a lungul traiectoriei:

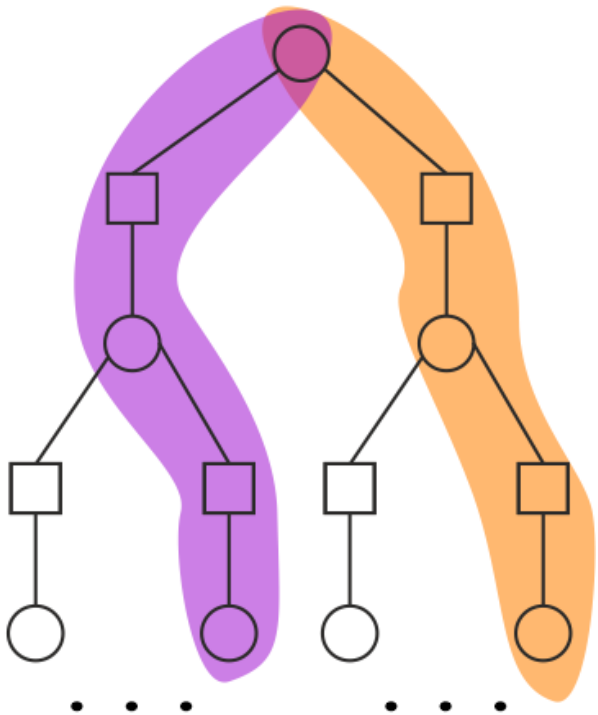
$$R(\tau) = \sum_{k=0}^{\infty} \gamma^k r_{k+1} = \sum_{k=0}^{\infty} \gamma^k \rho(x_k, u_k)$$

- Probabilitatea traiectoriei τ daţi fiind parametrii legii de control ϑ:

$$P_\vartheta(\tau) = \prod_{k=0}^{\infty} \widehat{h}(x_k, u_k; \vartheta)$$

unde $x_{k+1} = f(x_k, u_k)$

Obiectiv



Considerăm x_0 fixat, pentru simplitate

Obiectiv

Maximizarea return-ului aşteptat din x_0 al legii de control $\widehat{h}(\cdot, \cdot; \vartheta)$:

$$J_\vartheta := \mathrm{E}_\vartheta \{R(\tau)\} = \int R(\tau) P_\vartheta(\tau) d\tau$$

Ideea principală

Ascensiune pe gradient pentru maximizarea J_ϑ:

$$\vartheta \leftarrow \vartheta + \alpha \nabla_\vartheta J_\vartheta$$

Derivarea gradientului

Dorim să uşurăm implementarea actualizării folosind date:

$$\begin{aligned}
\nabla_\vartheta J_\vartheta &= \int R(\tau)\nabla_\vartheta P_\vartheta(\tau)d\tau \\
&= \int R(\tau)P_\vartheta(\tau)\nabla_\vartheta \log P_\vartheta(\tau)d\tau \\
&= \mathrm{E}_\vartheta\left\{R(\tau)\nabla_\vartheta \log\left[\prod_{k=0}^{\infty}\widehat{h}(x_k,u_k;\vartheta)\right]\right\} \\
&= \mathrm{E}_\vartheta\left\{R(\tau)\sum_{k=0}^{\infty}\nabla_\vartheta \log\widehat{h}(x_k,u_k;\vartheta)\right\}
\end{aligned}$$

Unde:

- am folosit “likelihood ratio trick”
  $\nabla_\vartheta P_\vartheta(\tau) = P_\vartheta(\tau)\nabla_\vartheta \log P_\vartheta(\tau)$
- am înlocuit integrala cu o aşteptare şi am substituit $P_\vartheta(\tau)$
- am înlocuit logaritmul produsului cu suma logaritmilor

Implementarea gradientului

- Există multe metode de estimare a gradientului, bazate de ex. pe Monte-Carlo
- De exemplu, REINFORCE foloseşte legea de control curentă pentru a executa n_τ traiectorii-eşantion, fiecare cu lungimea finită K, şi estimează:

$$\widehat{\nabla_\vartheta J_\vartheta} = \frac{1}{n_\tau} \sum_{s=1}^{n_\tau} \left[\sum_{k=0}^{K-1} \gamma^k r_{s,k} \right] \left[\sum_{k=0}^{K-1} \nabla_\vartheta \log \widehat{h}(x_{s,k}, u_{s,k}; \vartheta) \right]$$

  (posibil adăugând un baseline pentru a reduce varianţa)
- Comparat cu formula exactă:

$$\nabla_\vartheta J_\vartheta = \mathrm{E}_\vartheta \left\{ R(\tau) \sum_{k=0}^{\infty} \nabla_\vartheta \log \widehat{h}(x_k, u_k; \vartheta) \right\}$$

- Gradientul $\nabla_\vartheta \log \widehat{h}$ preferabil direct calculabil (prin formulă)

Policy gradient: algoritm

Policy gradient cu REINFORCE

iniţializează ϑ

repeat

execută n_τ traiectorii-eşantion cu lungimea K

estimează:

$$\widehat{\nabla_\vartheta J_\vartheta} = \frac{1}{n_\tau} \sum_{s=1}^{n_\tau} \left[\sum_{k=0}^{K-1} \gamma^k r_{s,k} \right] \left[\sum_{k=0}^{K-1} \nabla_\vartheta \log \widehat{h}(x_{s,k}, u_{s,k}; \vartheta) \right]$$

actualizează parametrii: $\vartheta \leftarrow \vartheta + \alpha \nabla_\vartheta J_\vartheta$

until finalizarea experimentului

Scaun rulant asistat electric (cu Feng et al.)

- Sursă de putere hibridă: utilizator uman şi baterie
- Scop: urmărirea referinţei de viteză, optimizând asistenţa pentru:
  - (i) a atinge nivelului dorit de oboseală a utilizatorului
  - (ii) a minimiza consumului de baterie
- Provocare: utilizatorul are **dinamică necunoscută**

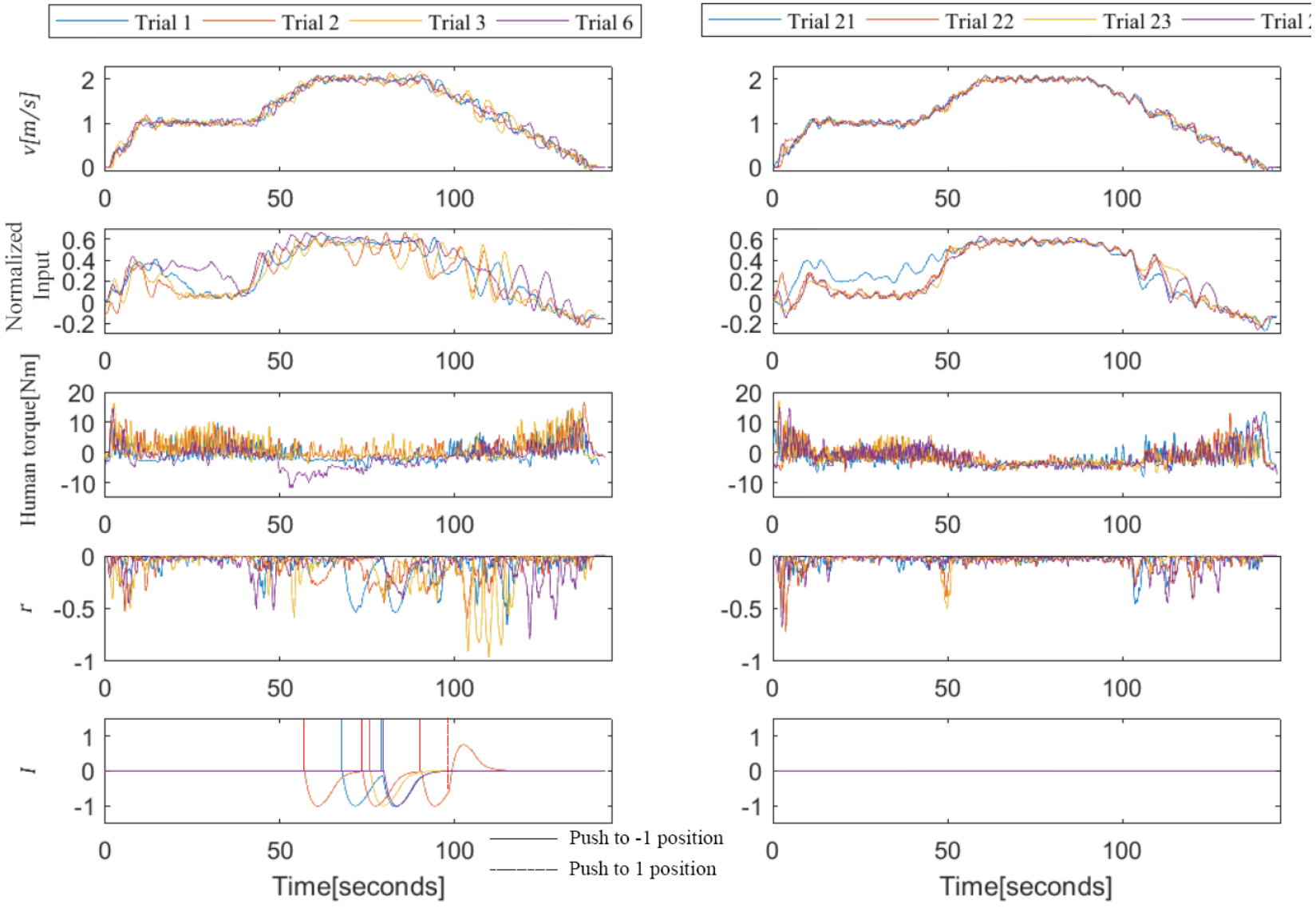
PAW: Configuraţia experimentului

- Utilizatorul setează viteza, trage/împinge joystick-ul când e prea obosit/vrea mai mult efort
- Recompensa penalizează eroarea de viteză, semnalul joystick-ului I, şi magnitudinea asistenţei (pentru economisirea energiei)

$$r = -w_1(v - v_{\text{ref}})^2 - w_2 I^2 - w_3 u^2$$

- Controller PI cu reglarea constantelor folosind policy gradient (specific, algoritmul POWER)

PAW: Rezultate



Terminologie engleză

coborâre pe gradient stohastică	=	stochastic gradient descent, SGD
semigradient	=	semigradient
diferenţă temporală aproximată	=	approximate temporal difference
gradientul pe legea de control	=	policy gradient

Exerciţii

1. Găsiţi actualizările de tip semigradient ale parametrilor θ pentru a aproxima V^h şi V^*.
2. Generalizaţi actualizarea semigradient din SARSA cu return pe n paşi.
3. Încercaţi să găsiţi o formulă completă de coborâre pe gradient pentru SARSA, care ia în considerare dependenţa estimatului cu bootstrap de vectorul de parametri.