

Control prin învățare

Master ICAF, An 1 Sem 2

Lucian Bușoniu, Ștefan Pîrje

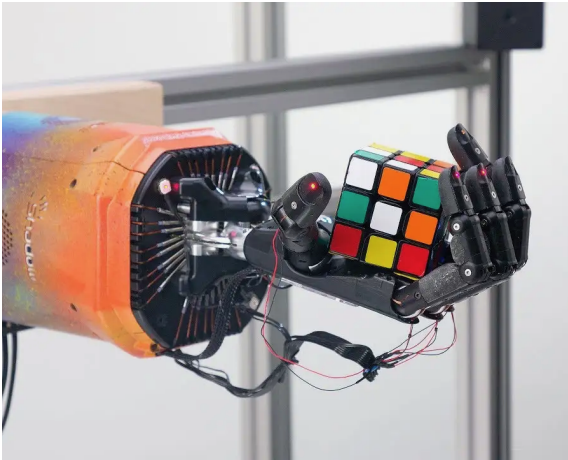


Partea IV

Programarea dinamică și învățarea prin
recompensă offline cu aproximare

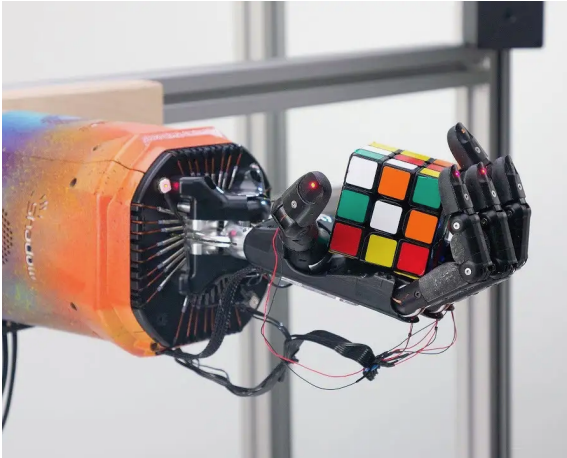


RL pentru manipularea unui cub Rubik (OpenAI)



Nevoia de aproximare

- RL clasică – reprezentare sub formă de tabel, de ex. $Q(x, u)$ separat pentru toate valorile x și u
- În aplicații reale de control, x, u **continue!**



Nevoia de aproximare (continuare)

In aplicații reale de control,
funcțiile de interes trebuie **aproximate**



Partea IV în plan

- Problema de învățare prin recompensă
- Soluția optimală
- Programarea dinamică exactă
- Învățarea prin recompensă exactă
- **Tehnici de aproximare**
- **Programarea dinamică cu aproximare**
- Învățarea prin recompensă cu aproximare
- Rețele neurale profunde
- Învățarea prin recompensă cu rețele neurale profunde



Conținut partea IV

- 1 Metode de aproximare
- 2 Aproximarea în DP&RL
- 3 Iterația Q cu interpolare
- 4 Iterația Q bazată pe date



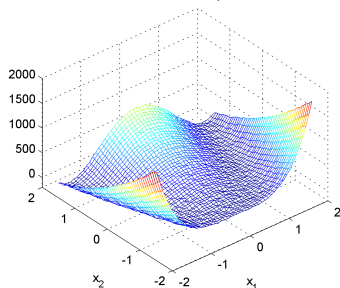
Aproximare

Aproximare:

funcție cu un număr infinit de valori

→ reprezentare printr-un număr mic de valori

$f(x)$
True values y



$\hat{f}(x)$

?

Aproximare parametrică

Aproximare parametrică: $\hat{f}(x)$ formă fixată,
valoare determinată de vector de **parametri** θ :

$$\hat{f}(x; \theta)$$

- 1 Aproximare liniară – combinație ponderată de **attribute** (funcții de bază) ϕ :

$$\begin{aligned}\hat{f}(x; \theta) &= \phi_1(x)\theta_1 + \phi_2(x)\theta_2 + \dots + \phi_n(x)\theta_n \\ &= \sum_{i=1}^n \phi_i(x)\theta_i = \phi^\top(x)\theta\end{aligned}$$

De notat: liniară în parametri, poate fi neliniară în x

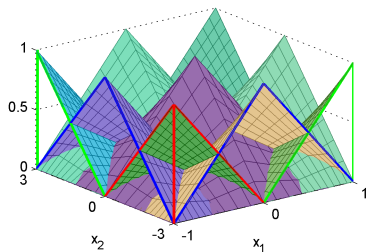
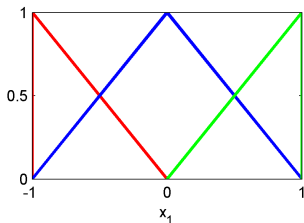
- 2 Aproximare neliniară: rămâne în forma generală



Aproximare parametrică liniară: Interpolare

Interpolare:

- Grilă D-dimensională de puncte
- Interpolare multiniară între puncte
- Echivalentul atributelor **piramidale**



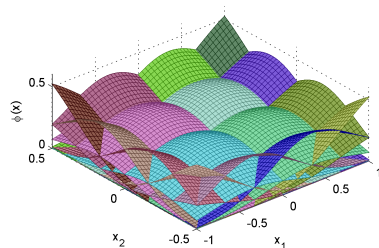
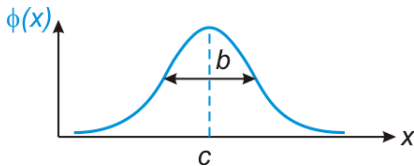
Aproximare parametrică liniară: RBF

Funcție de bază radială (Gaussiană):

$$\phi(x) = \exp\left[-\frac{(x-c)^2}{b^2}\right] \quad (1\text{-dim});$$

$$= \exp\left[-\sum_{d=1}^D \frac{(x_d - c_d)^2}{b_d^2}\right] \quad (D\text{-dim})$$

Eventual, normalizare: $\tilde{\phi}_i(x) = \frac{\phi_i(x)}{\sum_{i' \neq i} \phi_{i'}(x)}$



Antrenarea aproximatoarelor liniare: CMMP

- n_s puncte $(x_j, f(x_j))$, obiectivul descris ca un sistem de ecuații:

$$\hat{f}(x_1; \theta) = \phi_1(x_1)\theta_1 + \phi_2(x_1)\theta_2 + \dots + \phi_n(x_1)\theta_n = f(x_1)$$

...

$$\hat{f}(x_{n_s}; \theta) = \phi_1(x_{n_s})\theta_1 + \phi_2(x_{n_s})\theta_2 + \dots + \phi_n(x_{n_s})\theta_n = f(x_{n_s})$$

- Formă matriceală:

$$\begin{bmatrix} \phi_1(x_1) & \phi_2(x_1) & \dots & \phi_n(x_1) \\ \dots & \dots & \dots & \dots \\ \phi_1(x_{n_s}) & \phi_2(x_{n_s}) & \dots & \phi_n(x_{n_s}) \end{bmatrix} \cdot \theta = \begin{bmatrix} f(x_1) \\ \dots \\ f(x_{n_s}) \end{bmatrix} \quad A\theta = b$$

CMMP (continuare)

- Sistemul este supradeterminat ($n_s > n$), ecuațiile nu vor fi toate satisfăcute cu egalitate.
- Definim o **funcție de pierdere** de tip eroare medie pătratică:

$$\mathcal{L}(\theta) = \frac{1}{n_s} \sum_{j=1}^{n_s} \left| f(x_j) - \hat{f}(x_j; \theta) \right|^2$$

- Rezolvare în sensul celor mai mici pătrate:

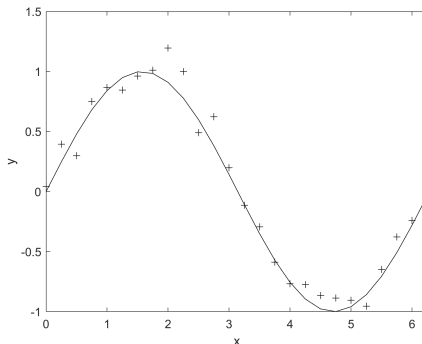
$$\theta^* = \arg \min_{\theta} \mathcal{L}(\theta)$$

...algebră și analiză liniară...

- $\theta^* = (A^T A)^{-1} A^T b$ ($\Leftrightarrow (A^T A)\theta^* = A^T b$)

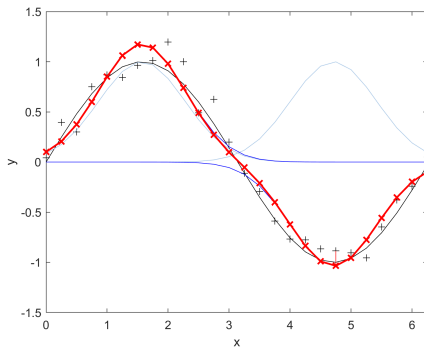


Exemplu de funcționare al aproximatoarelor liniare



Aproximăm $\sin(x)$ pe intervalul $[0, 2\pi]$. De notat: Metoda nu are access la funcție, ci doar la eșantioane cu zgomot!

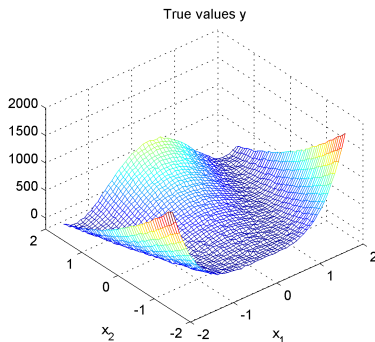
Exemplu de funcționare al aproximatoarelor liniare



- Cum datele au un “deal” și o “vale”, **două RBFuri** ar trebui să fie suficiente
- Parametrii θ_1, θ_2 **scalează & reflectă RBFurile** în așa fel încât **soluția aproximată** minimizează suma erorilor pătratice între datele reale și cele approximate.



Exemplu 2D: Funcția “banană” Rosenbrock



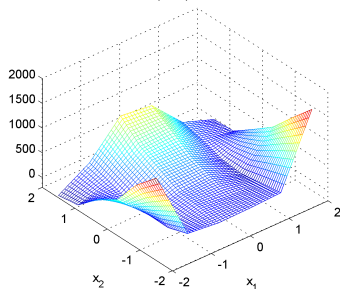
- $f(x) = (1 - x_1)^2 + 100[(x_2 + 1.5) - x_1^2]^2$,
- **Antrenare:** 200 puncte distribuite aleator
- **Validare:** grilă de 31×31 puncte

$$x = [x_1, x_2]^T$$

Funcția Rosenbrock: Rezultat cu aproximatoare liniare

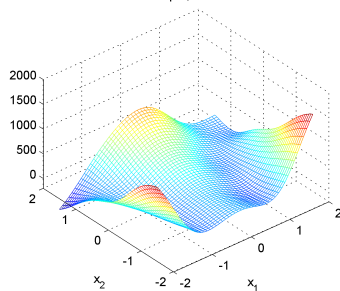
Interpolare pe grilă 6x6:

linearinterp output; MSE=4175



6x6 RBFuri:

RBFs output; MSE=5399

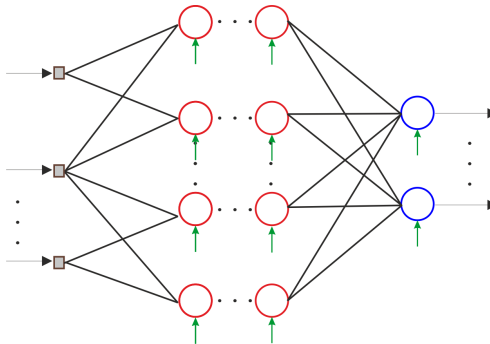


- Interpolarea = colecție de suprafețe multiliniare
- Aproximarea RBF mai netedă (RBFuri late)

Aproximare parametrică neliniară: rețea neurală

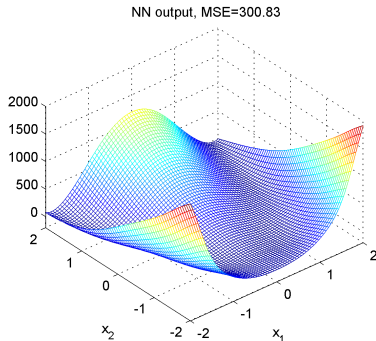
Rețea neurală:

- Neuroni cu funcții de activare (ne)liniare
- Interconectați pe mai multe straturi **ascunse** + 1 strat de **ieșire**
- Prin **legături ponderate** și **bias-uri**



Funcția Rosenbrock: Rezultat cu rețea neurală

Un nivel ascuns cu 10 neuroni și funcții de activare tangent-sigmoidale (liniare pentru nivelul de ieșire). 500 epoci de antrenare.



Datorită flexibilității mai bune a rețelei neurale, rezultatele sunt mai bune decât cu aproximatoarele liniare.

Rețea neurală pentru identificarea atributelor

De obicei, ultimul strat al unei rețele neuronale este liniar, conducând din nou la un aproximator bazat pe atribute:

$$\hat{f}(x; \theta) = \phi^\top(x; \theta_{\text{atr}})\theta_{\text{lin}}$$

unde $\theta = [\theta_{\text{atr}}^\top, \theta_{\text{lin}}^\top]^\top$.

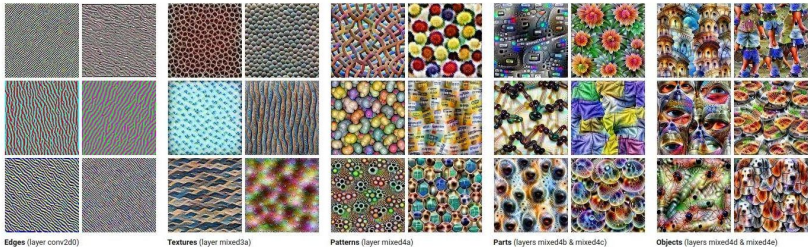
Diferența cheie față de aproximarea liniară: atributele ϕ sunt acum parametrizate – prin θ_{atr} – și **găsite automat de către rețeaua neurală**. În aproximarea liniară, atributele sunt proiectate manual.

Per ansamblu, aproximatorul este neliniar!



Atribute folosind rețele neuronale

Exemplu: Atribute găsite de o rețea neuronală convoluțională:



Detalii mai târziu

Comparație între aproximatoare

- aproximatoarele liniare mai ușor de tratat teoretic decât neliniare
- aproximatoarele neliniare sunt mai flexibile decât liniare



- 1 Metode de aproximare
- 2 Aproximarea în DP&RL**
- 3 Iterația Q cu interpolare
- 4 Iterația Q bazată pe date

Aproximarea în RL

Probleme de rezolvat:

- 1 **Reprezentare:** $Q(x, u)$, $V(x)$, $h(x)$
Folosind metodele de aproximare discutate
- 2 **Maximizare:** ex. $\max_u Q(x, u)$



Maximizare soluția 1: h implicit

- Legea de control nu este reprezentată explicit
- Acțiuni greedy calculate la cerere din \hat{Q} :

$$h(x) = \arg \max_u \hat{Q}(x, u)$$

- Problema principală: **aproximarea funcției Q**
- Aproximatorul trebuie să garanteze **soluție eficientă pentru arg max**



Maximizare soluția 2: h explicit

- Legea de control aproximată explicit: $\hat{h}(x)$

Avantaje:

- **Acțiuni continue** mai ușor de folosit
- Reprezentarea poate include mai ușor **cunoștințe a priori**



Focus: Soluția 1, h implicit

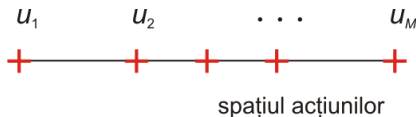
În acest curs:

- Legea de control nu este reprezentată explicit
- Problema principală: **aproximarea funcției Q**



Discretizarea acțiunilor

- Aproximatorul trebuie să garanteze soluție eficientă pentru $\arg \max$
- ⇒ Tipic: **discretizare acțiuni**
- Alege M acțiuni discrete $u_1, \dots, u_M \in U$
Calculează “ $\arg \max$ ” folosind enumerare explicită
- Exemplu: **discretizare pe o grilă**

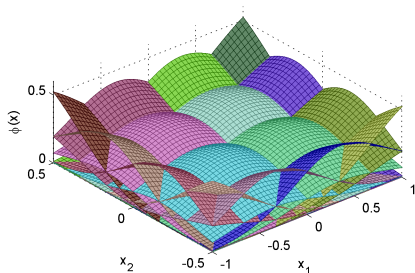
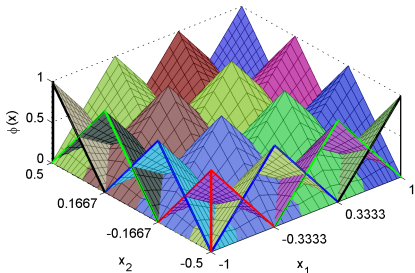


Aproximare în spațiul stărilor

- Tipic: atribute (funcții de bază)

$$\phi_1, \dots, \phi_N : X \rightarrow [0, \infty)$$

- Ex. piramidale, RBF



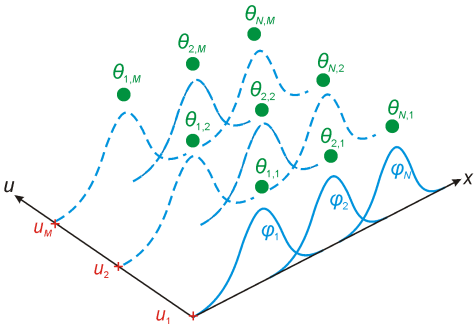
Funcția Q aproximată cu acțiuni discrete

Date fiind:

- 1 N atribute ϕ_1, \dots, ϕ_N
- 2 M acțiuni discrete u_1, \dots, u_M

Stochează:

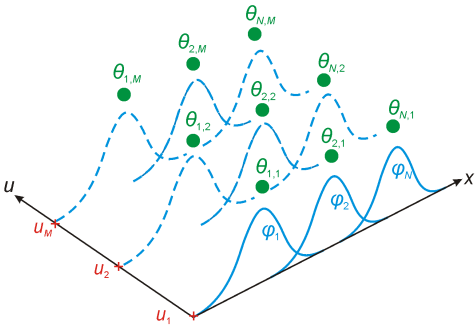
- 3 $N \cdot M$ parametri θ
(pentru fiecare pereche atribut – acțiune discretă)



Funcția Q aproximată cu acțiuni discrete (continuare)

Funcția Q aproximată:

$$\widehat{Q}(x, u_j; \theta) = \sum_{i=1}^N \phi_i(x) \theta_{i,j} = [\phi_1(x) \dots \phi_N(x)] \begin{bmatrix} \theta_{1,j} \\ \vdots \\ \theta_{N,j} \end{bmatrix}$$



Beneficiul aproximării în RL

Aproximarea permite aplicarea RL
în probleme realiste de control



Exemplu: Pendul inversat



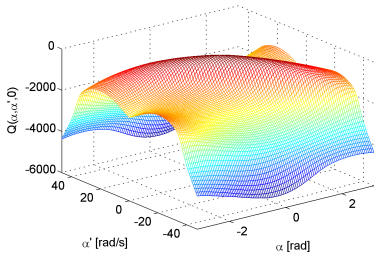
- $x = [\text{unghi } \alpha, \text{ viteză } \dot{\alpha}]^T$
- $u = \text{voltaj}$
- $\rho(x, u) = -x^T \begin{bmatrix} 5 & 0 \\ 0 & 0.1 \end{bmatrix} x - u^T 1 u$
- Factor de discount $\gamma = 0.98$

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

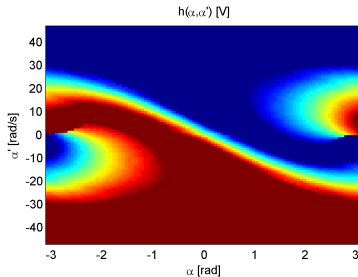
Pendul inversat: Soluție optimă

Stânga: Funcția Q pentru $u = 0$

Dreapta: legea de control



▶ Replay



Întrebări ridicate de aproximare

- 1 **Convergență**: rămâne algoritmul convergent?
- 2 **Calitatea soluției**: la o distanță controlată de optim?
- 3 **Consistență**: pentru un aproximator ideal, de precizie infinită, este soluția optimală regăsită?



Algoritmii din partea IV în taxonomie

După utilizarea unui model:

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

După nivelul de interacțiune:

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

Exact vs. cu aproximare:

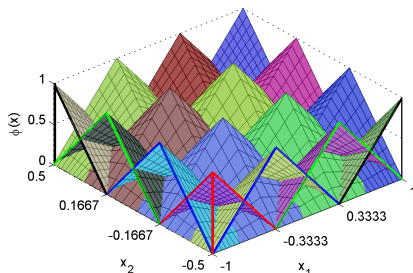
- **Exact:** x, u număr mic de valori discrete
- **Cu aproximare:** x, u continue (sau multe valori discrete)



- 1 Metode de aproximare
- 2 Aproximarea în DP&RL
- 3 Iterația Q cu interpolare**
- 4 Iterația Q bazată pe date

Approximator cu interpolare

- Interpolare = BF piramidale



- Fiecare BF i are centrul x_i
- $\theta_{i,j}$ poate fi văzut ca $\widehat{Q}(x_i, u_j)$, din motivul:
 $\phi_i(x_i) = 1, \phi_{i'}(x_i) = 0$ pentru $i' \neq i$

Iterația Q cu interpolare

Reamintim iterația Q clasică:

```

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
    
```

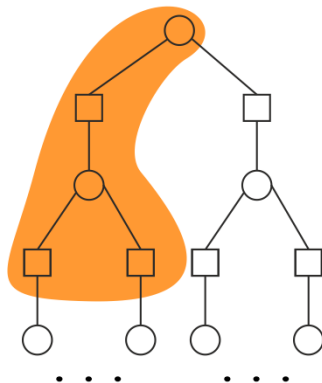
Iterația Q cu interpolare

```

repeat la fiecare iterație ℓ
  for all centrele  $x_i$ , acțiunile discrete  $u_j$  do
     $\theta_{\ell+1,i,j} \leftarrow \rho(x_i, u_j) + \gamma \max_{j'} \hat{Q}(f(x_i, u_j), u_{j'}; \theta_{\ell})$ 
  end for
until convergență
    
```



Ilustrație



Lege de control

- Reamintim legea optimală de control:

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

- În iterația Q cu interpolare:

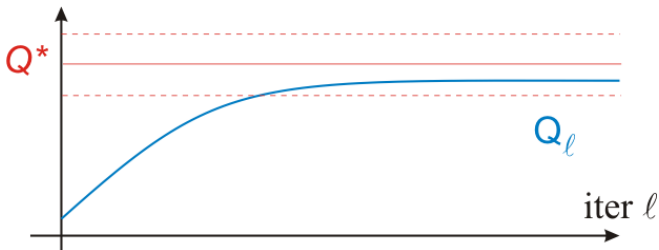
$$\hat{h}^*(x) = \arg \max_{u_j, j=1, \dots, M} \hat{Q}(x, u_j; \theta^*)$$

θ^* = parametrii la convergență



Convergență în imagini

Convergență monotonă la o soluție **aproape-optimală**



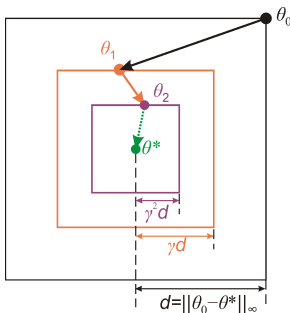
Convergență

Similar cu iterația Q clasică:

- Fiecare iterație este o contracție cu factor γ :

$$\|\theta_{\ell+1} - \theta^*\|_{\infty} \leq \gamma \|\theta_{\ell} - \theta^*\|_{\infty}$$

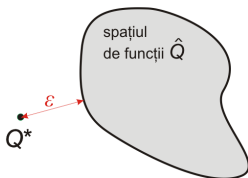
⇒ **Convergență monotonă** la θ^*



Calitatea soluției

Caracterizăm aproximatorul prin distanța minimă până la Q^* :

$$\varepsilon = \min_{\theta} \left\| Q^*(x, u) - \widehat{Q}(x, u; \theta) \right\|_{\infty}$$



Avem:

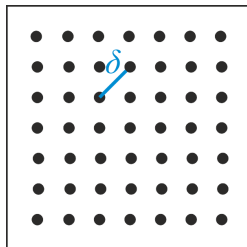
- 1 Suboptimalitatea rezultatului $\widehat{Q}(x, u; \theta^*)$ mărginită:

$$\left\| Q^*(x, u) - \widehat{Q}(x, u; \theta^*) \right\|_{\infty} \leq \frac{2\varepsilon}{1-\gamma}$$

- 2 Suboptimalitatea legii de control \widehat{h}^* mărginită, $\frac{4\varepsilon}{(1-\gamma)^2}$

Consistență

- Consistență: $\widehat{Q}^{\theta^*} \rightarrow Q^*$ pe măsură ce precizia crește
- Precizia măsurată prin rezoluția δ_x a grilei de interpolare în x , δ_u a grilei de discretizare în u

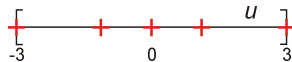
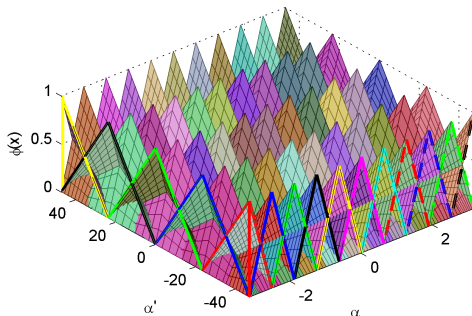


- Date fiind anumite condiții tehnice,
 $\Rightarrow \lim_{\delta_x \rightarrow 0, \delta_u \rightarrow 0} \widehat{Q}^{\theta^*} = Q^*$ — consistență

Pendul inversat: Iterația Q cu interpolare, demo

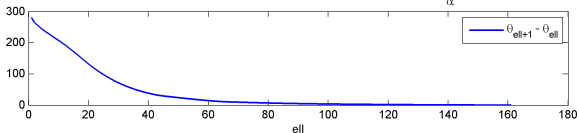
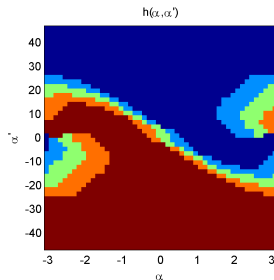
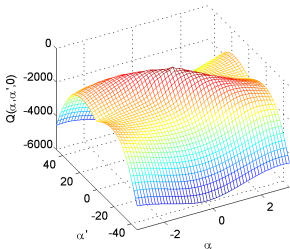
BF: grilă echidistantă 41×21

Discretizare: 5 acțiuni, distribuite în jurul lui 0



Pendul inversat: Iterația Q cu interpolare, demo

Fuzzy Q-iteration, ell=161



- 1 Metode de aproximare
- 2 Aproximarea în DP&RL
- 3 Iterația Q cu interpolare
- 4 Iterația Q bazată pe date**



Iterația Q bazată pe date

Pornim de la iterația Q cu interpolare și o extindem:

- folositoare cu alte aproximatoare decât interpolare
- **fără model** – RL



Algoritm intermediar bazat pe model

Reamintim iterația Q cu interpolare:

for all x_i, u_j $\theta_{\ell+1,i,j} \leftarrow \rho(x_i, u_j) + \gamma \max_{j'} \widehat{Q}(f(x_i, u_j), u_{j'}; \theta_\ell)$ **end for**

- 1 Folosim eșantioane stare-acțiune **arbitrare**
- 2 Extindem la **aproximator generic**
- 3 Găsim parametrii folosind **cele mai mici pătrate**

date fiind (x_s, u_s) , $s = 1, \dots, n_s$

repeat la fiecare iterație ℓ

for $s = 1, \dots, n_s$ **do**

 calculează ținta cu bootstrap pentru $\widehat{Q}(x_s, u_s; \theta)$:

$q_s \leftarrow \rho(x_s, u_s) + \gamma \max_{u'} \widehat{Q}(f(x_s, u_s), u'; \theta_\ell)$

end for

$\theta_{\ell+1} \leftarrow \arg \min \sum_{s=1}^{n_s} |q_s - \widehat{Q}(x_s, u_s; \theta)|^2$

until terminare

Iterația Q cu interpolare echivalentă cu algoritmul generalizat dacă eșantioanele sunt toate combinațiile x_i, u_j



Iterația Q bazată pe date: Algoritm

- 4 Folosim **tranziții** în loc de model

Iterația Q bazată pe date

date fiind (x_s, u_s, r_s, x'_s) , $s = 1, \dots, n_s$

repeat la fiecare iterație ℓ

for $s = 1, \dots, n_s$ **do**

calculează ținta cu bootstrap pentru $\widehat{Q}(x_s, u_s; \theta)$:

$$q_s \leftarrow r_s + \gamma \max_{u'} \widehat{Q}(x'_s, u'; \theta_\ell)$$

end for

$$\theta_{\ell+1} \leftarrow \arg \min \sum_{s=1}^{n_s} |q_s - \widehat{Q}(x_s, u_s; \theta)|^2 =: \mathcal{L}(\theta)$$

until oprire

\mathcal{L} este **funcția de pierdere**



Determinist versus stochastic

- În cazul determinist, $x'_s = f(x_s, u_s)$, $r_s = \rho(x_s, u_s)$
– înlocuirile sunt exacte
- În cazul stochastic, $x'_s \sim \tilde{f}(x_s, u_s, \cdot)$, $r_s = \tilde{\rho}(x_s, u_s, x'_s)$

⇒ Algoritmul **rămâne valid**; intuiție:

- Ideal, $Q(x, u) \leftarrow E_{x'} \left\{ r + \gamma \max_{u'} \hat{Q}(x', u'; \theta_\ell) \right\}$
- Presupunând n_s eșantioane, toate în $(x_s, u_s) = (x, u)$:

$$\min_{\theta} \sum_{s=1}^{n_s} \left| r_s + \gamma \max_{u'} \hat{Q}(x'_s, u'; \theta_\ell) - \hat{Q}(x, u; \theta) \right|^2$$

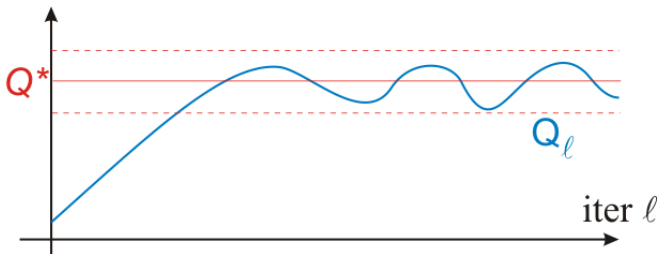
duce la $\hat{Q}(x, u; \theta) \approx E \{ \dots \}$

- Chiar dacă (x_s, u_s) nu se repetă, CMMP aproximează valoarea așteptată
- Similar cu faptul că actualizările incrementale $Q(x_k, u_k) \leftarrow Q(x_k, u_k) + \alpha_k [R_k - Q(x_k, u_k)]$ converg la valorile Q corecte



Iterația Q bazată pe date: Convergență

Convergență la o **secvență** de soluții,
fiecare din ele **aproape-optimală**

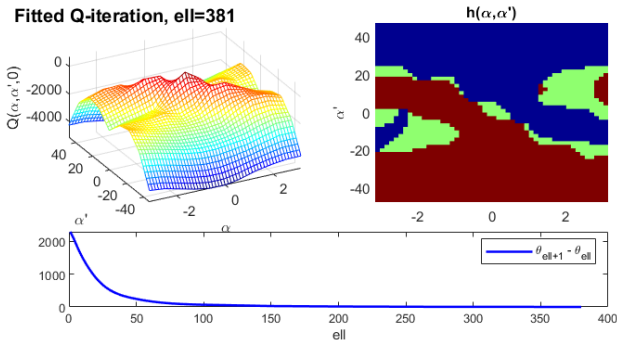


Pend. invers: iterația Q ajustată cu interpolare, demo

Atribute: grilă echidistantă 11×9

Acțiuni discretizate: 3 valori, +/- tensiune maximă și 0

Set de date: 10000 de eșantioane, distribuite uniform în spațiul continuu x - discretizat u



Terminologie engleză

atribut	=	<u>feature</u>
funcție de bază	=	<u>basis function</u>
rețea neurală	=	<u>neural network</u>
strat, neuron, bias	=	<u>layer, neuron, bias</u>
funcție de pierdere	=	<u>loss function</u>
cele mai mici pătrate	=	<u>least squares</u>
iterația Q cu interpolare	=	<u>interpolated Q-iteration</u>
iterația Q bazată pe date	=	<u>fitted Q-iteration</u>
consistență	=	<u>consistency</u>



Exerciții

- 1 Demonstrați că actualizările iterației Q cu interpolare:

$$\theta_{\ell+1,i,j} \leftarrow \rho(x_i, u_j) + \gamma \max_{j'} \widehat{Q}(f(x_i, u_j), u_{j'}; \theta_\ell)$$

sunt contractive cu factorul γ . Indiciu: este esențial că attributele se însumează la 1 și, prin urmare, valoarea aproximativă a funcției Q este o medie a parametrilor!

- 2 Care este limita distanței ε dintre Q^* și spațiul funcțiilor Q reprezentabile atunci când $\delta_x \rightarrow 0, \delta_u \rightarrow 0$? Explicați de ce.
- 3 Scrieți o metodă iterație V aproximată cu interpolare și demonstrați că actualizările sale sunt contractive.
- 4 Scrieți o metodă de iterație V bazată pe date. Este această metodă independentă de model?

