

Control prin învățare

Master ICAF, An 1 Sem 2

Lucian Bușoniu

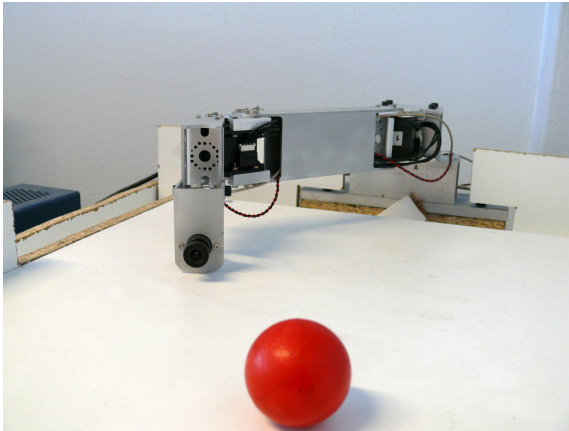


Partea IV

Aproximare. Programarea dinamică cu
aproximare. Învățarea prin recompensă
offline

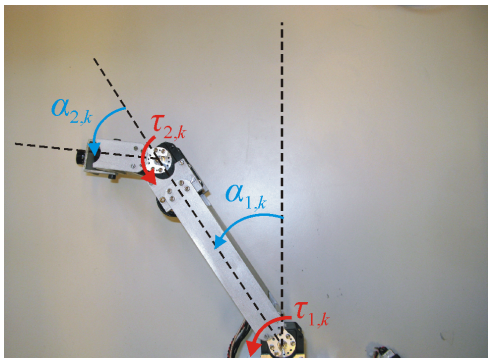
RL pentru un robot-portar (TUDelft)

Învăță să prindă mingea folosind camera video



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!**



- **Reprezentarea prin tabel imposibilă**

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



Conținut partea IV

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

Aproximare

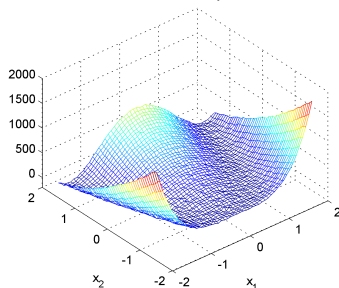
Aproximare:

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 **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

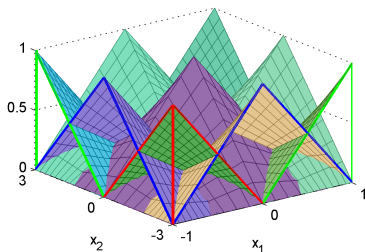
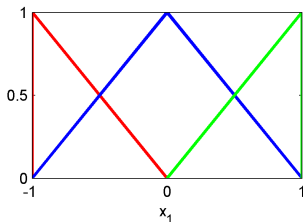
- 2 **Aproximare neliniară:** rămâne în forma generală



Aproximare parametrică liniară: Interpolare

Interpolare:

- Grilă D-dimensională de puncte
- Interpolare multiniară între puncte
- Echivalent cu funcții de bază **piramidale**



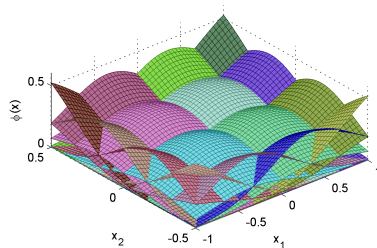
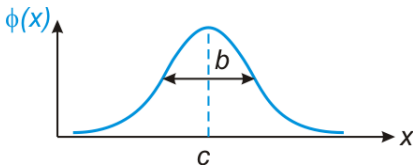
Aproximare parametrică liniară: RBF

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:

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

- Regresie liniară, vezi SysID

CMMP (continuare)

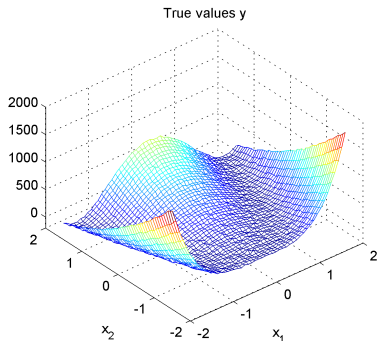
- Sistemul este supradeterminat ($n_s > n$), ecuațiile nu vor fi toate satisfăcute cu egalitate.
 $\Rightarrow$ Rezolvare în sensul celor mai mici pătrate:

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

...algebră și analiză liniară...

- $\theta = (A^{\top} A)^{-1} A^{\top} b$ (Intuiție: $(A^{\top} A)\theta = A^{\top} b$)

Exemplu: Funcția “banană” Rosenbrock



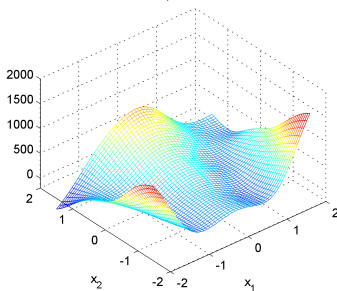
- $f(x) = (1 - x_1)^2 + 100[(x_2 + 1.5) - x_1^2]^2$,
- **Antrenare:** 200 puncte distribuite aleator
- **Validare:** grilă de 31×31 puncte

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

Funcția Rosenbrock: Rezultat cu aproximatoare liniare

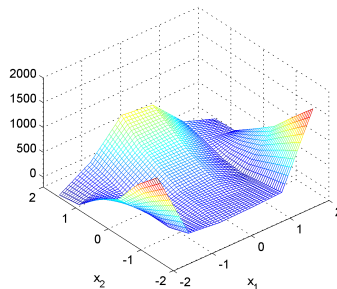
6x6 RBFuri:

RBFs output; MSE=5399



Interpolare pe grilă 6x6:

linearinterp output; MSE=4175

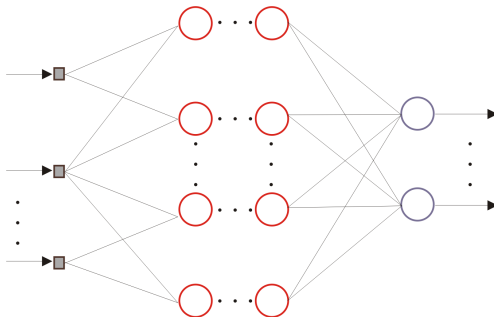


- Aproximarea RBF mai netedă (RBFuri late)
- Interpolarea = colecție de suprafețe multiliniare

Aproximare parametrică neliniară: rețea neuronală

Rețea neuronală:

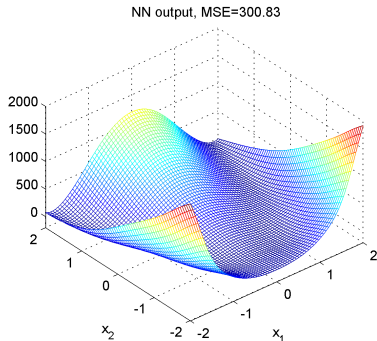
- **Neuroni** cu funcții de activare (ne)liniare
- **Interconectați** pe nivele multiple, prin legături ponderate



Va urma...

Funcția Rosenbrock: Rezultat cu rețea neuronală

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 neuronale, rezultatele sunt mai bune decât cu aproximatoarele liniare.

Aproximare neparametrică

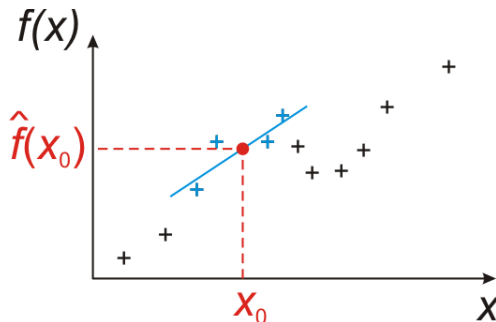
Aproximare parametrică:
formă, număr de parametri fixate

Aproximare neparametrică:
formă, număr de parametri **depind de date**

Aproximare neparametrică: LLR

Regresie liniară locală, LLR:

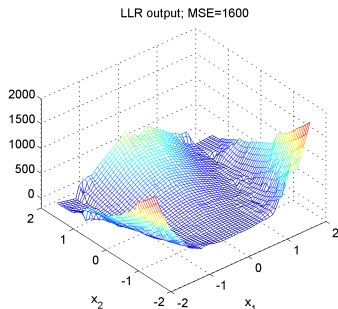
- Bază de date cu puncte de forma $(x, f(x))$
- Pentru x_0 dat, găsește **k cei mai apropiați vecini**
- Rezultat calculat cu **regresie liniară** (CMMP) pe vecini



Funcția Rosenbrock: Rezultat cu LLR

Baza de date = 200 de puncte de antrenament; $k = 5$

Validare: grilă de 31×31 puncte



- Performanța între aproximatoarele liniare și rețeaua neuronală

Comparație între aproximatoare

În combinație cu RL

- liniare mai ușor de tratat teoretic decât neliniare
- parametrice mai ușor de tratat teoretic decât neparametrice

Flexibilitate

- neliniare mai flexibile decât liniare
- nonparametrice mai flexibile decât parametrice:
forma aprox. parametrice trebuie adaptată manual
- nonparametrice se adaptează la date:
complexitatea trebuie controlată când #date crește



- 1 Metode de aproximare
- 2 Aproximarea în DP&RL
- 3 Iterația Q cu interpolare (fuzzy Q)
- 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**

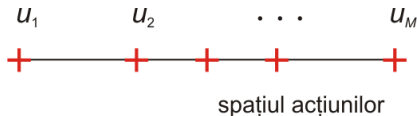


Discretizarea acțiunilor

- 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

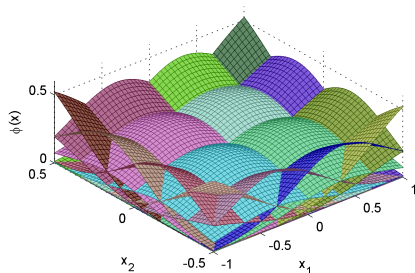
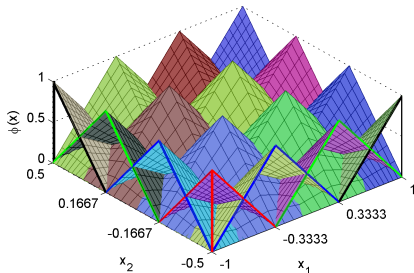


Aproximare în spațiul stărilor

- Tipic: funcții de bază

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

- Ex. **piramidale**, **RBF**



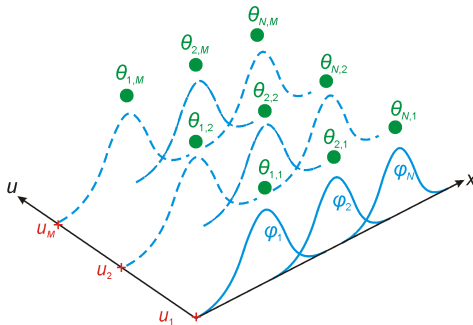
Funcția Q aproximată cu acțiuni discrete

Date fiind:

- 1 N funcții de bază $\phi_1, \dots, \phi_N$
- 2 M acțiuni discrete $u_1, \dots, u_M$

Stocchează:

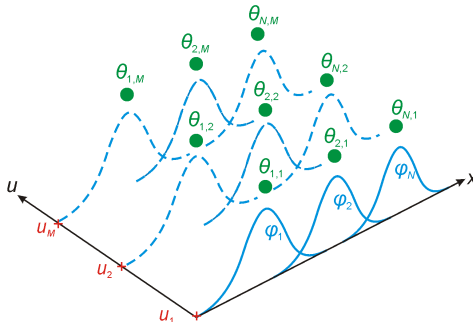
- 3 $N \cdot M$ parametri θ
(pentru fiecare pereche funcție de bază–acțiune discretă)



Funcția Q aproximată cu acțiuni discrete (continuare)

Funcția Q aproximată:

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



Beneficiul aproximării în RL

Aproximarea permite aplicarea RL
în probleme realiste de control

Exemplu: Pendul inversat



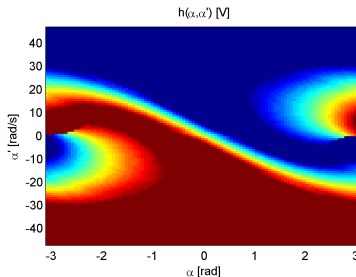
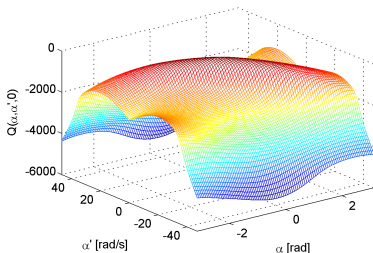
- $x = [\text{unghi } \alpha, \text{ viteză } \dot{\alpha}]^T$
- $u = \text{voltaj}$
- $\rho(x, u) = -x^T \begin{bmatrix} 5 & 0 \\ 0 & 0.1 \end{bmatrix} x - u^T 1 u$
- Factor de discount $\gamma = 0.98$

- **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:

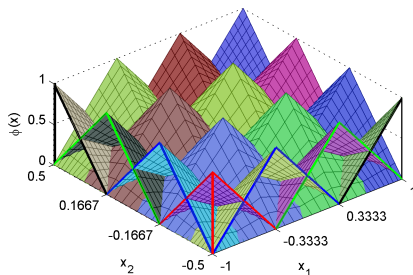
- **Exact:** x , u număr mic de valori discrete
- **Cu aproximare:** x , u continue (sau multe valori discrete)



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

Approximator cu interpolare (“fuzzy”)

- Interpolare = BF piramidale



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

Iterația Q cu interpolare (fuzzy Q)

Reamintim iterația Q clasică:

```
repeat la fiecare iterație  $\ell$ 
  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 fuzzy Q

```
repeat la fiecare iterație  $\ell$ 
  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
```

Lege de control

- Reamintim legea optimală de control:

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

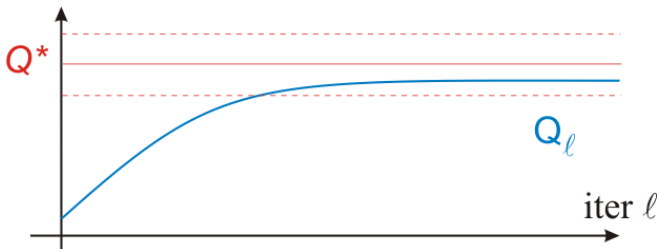
- În iterația fuzzy Q:

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

θ^* = parametrii la convergență

Convergență în imagini

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



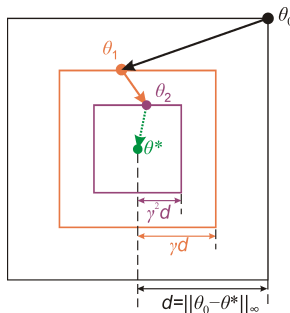
Convergență

Similar cu iterația Q clasică:

- Fiecare iterație este o contracție cu factor γ :

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

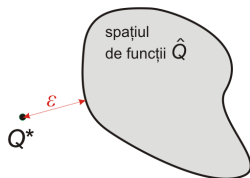
⇒ **Convergență monotonă** la θ^*



Calitatea soluției

Caracterizăm aproximatorul prin distanța minimă până la Q^* :

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



Avem:

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

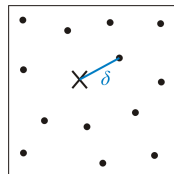
$$\left\| Q^*(x, u) - \hat{Q}(x, u; \theta^*) \right\|_{\infty} \leq \frac{2\varepsilon}{1-\gamma}$$

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

Consistență

- Consistență: $\hat{Q}^{\theta^*} \rightarrow Q^*$ pe măsură ce precizia crește

- Precizia:
$$\begin{cases} \delta_x = \max_x \min_i \|x - x_i\|_2 \\ \delta_u = \max_u \min_j \|u - u_j\|_2 \end{cases}$$

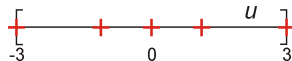
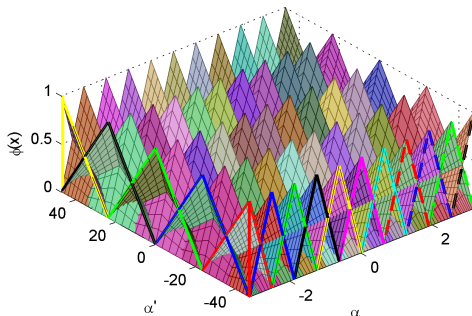


- Date fiind anumite condiții tehnice,
 $\Rightarrow \lim_{\delta_x \rightarrow 0, \delta_u \rightarrow 0} \hat{Q}^{\theta^*} = Q^*$ — consistență

Pendul inversat: Iterația fuzzy Q, demo

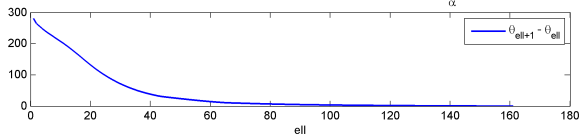
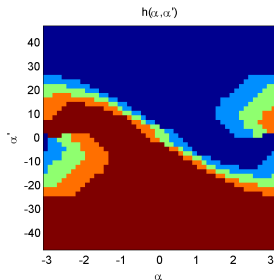
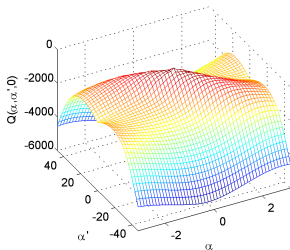
BF: grilă echidistantă 41×21

Discretizare: 5 acțiuni, distribuite în jurul lui 0



Pendul inversat: Iterația fuzzy Q, demo

Fuzzy Q-iteration, ell=161



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

Iterația Q bazată pe date

Pornim de la iterația fuzzy Q și o extindem:

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



Algoritm intermediar bazat pe model

Reamintim iterația fuzzy Q:

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

- ❶ Folosim eşantioane stare-acțiune **arbitrare**
- ❷ Extindem la **aproximator generic**
- ❸ 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**

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

end for

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

until terminare

Iterația fuzzy Q echivalentă cu algoritmul generalizat dacă eşantioanele 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**

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

end for

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

until terminare

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 $(x_s, u_s) = (x, u)$:

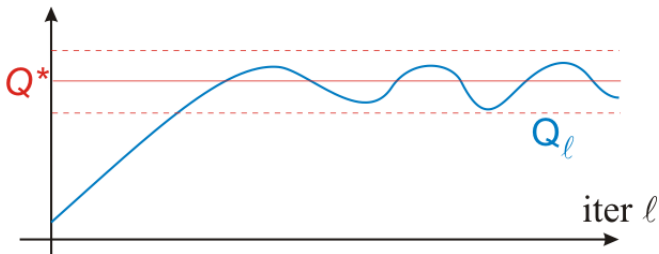
$$\min_{\theta} \sum_{s=1}^{n_s} \left| r_s + \gamma \max_{u'} \hat{Q}(x'_s, u'; \theta_\ell) - \hat{Q}(x, u; \theta) \right|^2$$

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

- Chiar dacă (x_s, u_s) nu se repetă, CMMP aproximează valoarea așteptată

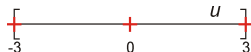
Iterația Q bazată pe date: Convergență

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



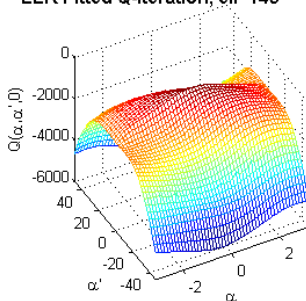
Pendul inversat: Iterația Q cu LLR, demo

Discretizare: 3 acțiuni

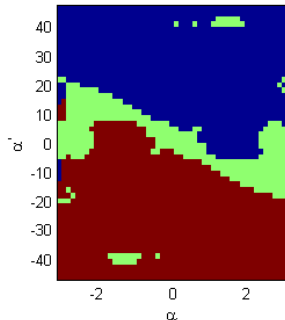


Baza de date: grilă $31 \times 15 \times 3$; $k = 5$

LLR Fitted Q-iteration, ell=146



$h(\alpha, \alpha')$



Terminologie engleză

funcție de bază	=	<u>basis function</u>
rețea neuronală	=	<u>neural network</u>
regresie liniară locală	=	<u>local linear regression, LLR</u>
iterația fuzzy Q	=	<u>fuzzy Q-iteration</u>
iterația Q bazată pe date	=	<u>model-free/fitted Q-iteration</u>

