Planning - Programare Dinamica

Luciana Morogan

Academia Tehnica Militara

3 iunie 2025

Cuprins

Intro

Introducere: Programarea Dinamică (PD)

- Dinamicitate:: Secvențialitate sau componentă temporală pt o problemă
- Programare:: optimizarea unui „program" (*politică*)

- Metodă de rezolvare a problemelor complexe prin
  - Împărțirea în subprobleme
  - Rezolvarea subproblemelor
  - Combinarea soluțiilor

Introducere: Programarea Dinamică

- Solutie pentru probleme in care:
  - Solutia optima poate fi descompusa in subprobleme/substructuri peste care poate fi aplicat principiul optimalității
  - Subproblemele sa poata fi suprapuse daca apar de mai multe ori. Solutiile subproblemelor pot fi memorate si reutilizate

Cum

- Ecuația Bellman $\rightarrow$ descompune recursiv
- Funcția de evaloare $\rightarrow$ salvează și reutilizează soluțiile

Atunci

- PD poate fi utilizata pt MDP-uri
- PD presupunere **cunoasterea COMPLETA** a MDP-ului

PD folosita pt planificare (PLANNING) intr-un MDP

Pentru predicție:

- Intrare:
  - MDP: $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ și politică π
  - MRP: $\langle \mathcal{S}, \mathcal{P}_\pi, \mathcal{R}_\pi, \gamma \rangle$
- Ieșire: funcția valoare v_π

Pentru control:

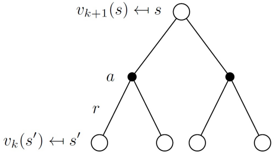
- Intrare: MDP: $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$
- Ieșire: funcția de valoare optimă v^* și politica optimă π^*

Evaluarea si iterarea politicii

Evaluarea Politicii: Evaluare Iterativă

- Problemă: evaluarea unei politici date π
- Soluție: aplicare iterativă a ecuației Bellman (B. expectation equation)

$$v_1 \rightarrow v_2 \rightarrow .. \rightarrow v_\pi$$

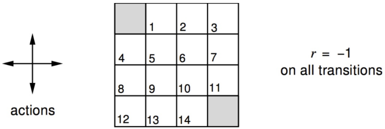


- Se realizeaza backup-uri **sincrone**
  - La fiecare iterație $k+1$ si:
  - Pentru toate stările $s \in S$
    - update $v_{k+1}(s)$ din $v_k(s')$ pentru s' succesor al lui s

$$v_{k+1}(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_k(s') \right)$$

$$\mathbf{v}^{k+1} = \boldsymbol{\mathcal{R}}^{\boldsymbol{\pi}} + \gamma \boldsymbol{\mathcal{P}}^{\boldsymbol{\pi}} \mathbf{v}^k$$

Exemplu: Small Gridworld



- Fie un MDP episodic fara discount ($\gamma = 1$)
- Stari neterminale: 1...14 si terminale: cele gri
- Actiuni: N, S, E, V
- Actiuni care ar duce la depasirea grilei lasa starea neschimbata
- Recompensa este -1 pana la atingerea starii terminale
- Agentul urmareste o politica aleatoare uniforma $\pi(n|.) = \pi(s|.) = \pi(e|.) = \pi(v|.) = 0.25$

Exemplu: Small Gridworld

v_k for the Random Policy

Greedy Policy w.r.t. v_k

$k = 0$

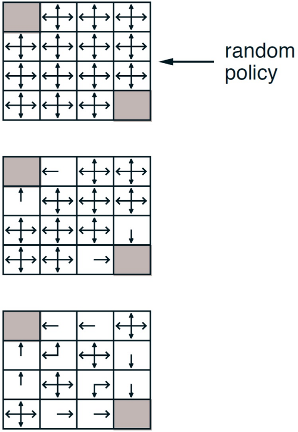
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

$k = 1$

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0

$k = 2$

0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0



Exemplu: Small Gridworld

$k = 3$

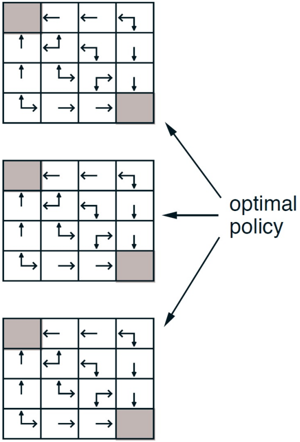
0.0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0.0

$k = 10$

0.0	-6.1	-8.4	-9.0
-6.1	-7.7	-8.4	-8.4
-8.4	-8.4	-7.7	-6.1
-9.0	-8.4	-6.1	0.0

$k = \infty$

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0

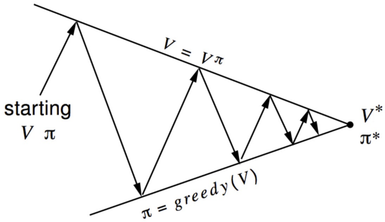


Iterarea Politicii

Scop: Imbunatatirea politicii

- Evaluează politica π: $v_\pi(s) = \mathbb{E}\left[R_{t+1} + \gamma R_{t+2} + \ldots \mid S_t = s\right]$
- Îmbunătățirea politicii: $\pi' = greedy(v^\pi)$
- Repetă până la convergență: $\pi' = \pi^*$

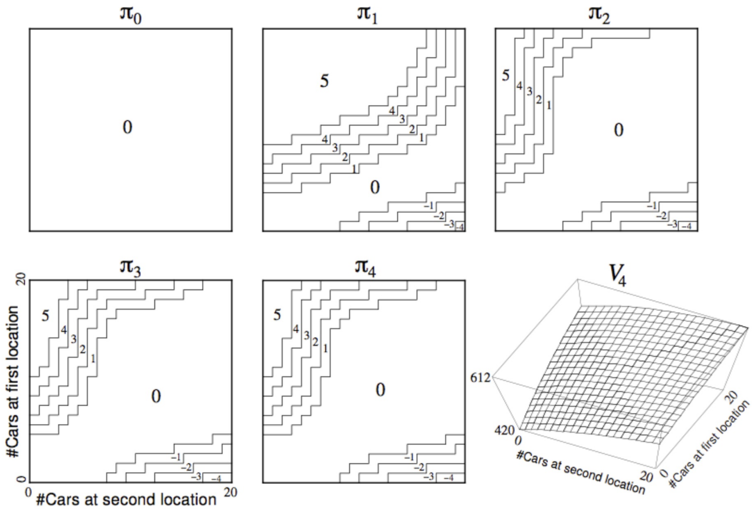
- Evaluarea politicii: Estimeaza v_π
  Evaluarea iterativa a politicii
- Imbunatatirea politicii: Genereaza $\pi' \geq \pi$
  Imbunatatirea greedy a politicii



Exemplu clasic: Jack's Car Rental - Iterarea politicii

- Stări: 2 locații a maxim 20 mașini
- Acțiuni: mutarea a maxim 5 mașini între locații (peste noapte)
- Recompensă: 10$ per mașină închiriată (tb sa fie disponibila in locatie)
- Tranzitii: inchirierea si returnarea random a masinilor pentru o distribuție Poisson, n retururi/cereri cu prob. $\frac{\lambda^n}{n!}e^{-\lambda}$
    - Locatia 1: in medie cu 3 cereri si 3 retururi
    - Locatia2: in medie cu 4 cereri si 2 retururi

Exemplu clasic: Jack's Car Rental - Iterarea politicii



Imbunatatirea politicii

- Fie $a = \pi(s)$ - o politica determinista
- a poate fi *imbunatatita* actionand greedy

$$\pi'(s) = \arg\max_{a \in \mathcal{A}} q_\pi(s, a) \rightarrow$$

- Valoarea oricarei stari s dupa un pas este imbunatatita

$$q_\pi(s, \pi'(s)) = \max_{a \in \mathcal{A}} q_\pi(s, a) \geq q_\pi(s, \pi(s)) = v_\pi(s) \rightarrow$$

- Functia valoare este imbunatatita: $v_{\pi'}(s) \geq v_\pi(s)$

$$\begin{aligned} v_\pi(s) \leq q_\pi(s, \pi'(s)) &= \mathbb{E}_{\pi'}\left[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s\right] \\ &\leq \mathbb{E}_{\pi'}\left[R_{t+1} + \gamma q_\pi(S_{t+1}, \pi'(S_{t+1})) \mid S_t = s\right] \\ &\leq \mathbb{E}_{\pi'}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 q_\pi(S_{t+2}, \pi'(S_{t+2})) \mid S_t = s\right] \\ &\leq \mathbb{E}_{\pi'}\left[R_{t+1} + \gamma R_{t+2} + \ldots \mid S_t = s\right] = v_{\pi'}(s) \end{aligned}$$

Imbunatatirea politicii: Convergenta

- Daca stop, atunci

$$q_\pi(s, \pi'(s)) = \max_{a \in \mathcal{A}} q_\pi(s, a) = q_\pi(s, \pi(s)) = v_\pi(s) \rightarrow$$

Ecuatia optimalitatii a lui Bellman este satisfacuta: $v_\pi(s) = \max_{a \in \mathcal{A}} q_\pi(s, a)$

$$\rightarrow v_\pi(s) = v * (s) \forall s \in \mathcal{S}$$

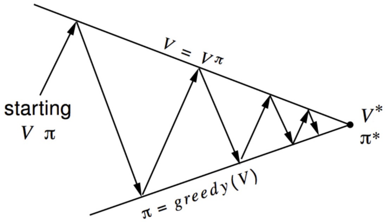
$\rightarrow \pi$ este o politica optimala

Iterarea Politicii Generalizata

Generalizare:

- Evaluarea politicii: Estimeaza v_π
Pentru orice algoritm de evaluare iterativa a politicii

- Imbunatatirea politicii: Genereaza $\pi' \geq \pi$
Pentru orice algoritm de imbunatatire a politicii



Iterarea valorii (Value Iteration)

Principiul Optimalității

- Orice politică optimă poate fi descompusă în:
  - O primă acțiune optimă a^*
  - Urmată de o politică optimă din starea următoare s'

Teoremă (Principiul optimalitatii)

O politica $\pi(a|s)$ atinge valoarea optima din starea s $(v_\pi(s) = v_*(s))$ dacă si numai daca pentru toate stările s' la care se poate ajunge din s, π atinge valoarea optima din starea s' $(v_\pi(s') = v_*(s'))$

Iterarea determinista a valorii (Deterministic Value Iteration)

- Daca se cunoaste solutia subproblemelor $v_*(s')$
- Atunci (one-step lookahead)

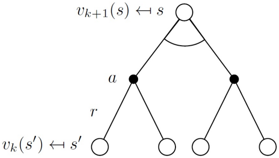
$$v_*(s) \leftarrow \max_{a \in \mathcal{A}} \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_*(s')$$

- Actualizările se aplica iterativ
- Intuitiv: se incepe cu recompensele finale și se lucrează inapoi

Iterarea Valorii: Determinarea π optim

- Problemă: determinarea unei politici π optimale
- Soluție: aplicare iterativă a ecuației Bellman optimale (B. optimality equation)

$$v_1 \to v_2 \to .. \to v_*$$



$$v_{k+1}(s) = \max_{a \in \mathcal{A}} \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_k(s') \right)$$

$$\mathbf{v}_{k+1} = \max_{a \in \mathcal{A}} \boldsymbol{\mathcal{R}}^{\mathbf{a}} + \gamma \boldsymbol{\mathcal{P}}^{\mathbf{a}} \mathbf{v}_k$$

- Se realizeaza backup-uri **sincrone**
  - La fiecare iterație $k+1$ si:
  - Pentru toate stările $s \in S$
    - update $v_{k+1}(s)$ din $v_k(s')$ pentru s' succesor al lui s

PD Sincrona - recap

Problem	Bellman Equation	Algorithm
Prediction	Bellman Expectation Equation	Iterative Policy Evaluation
Control	Bellman Expectation Equation + Greedy Policy Improvement	Policy Iteration
Control	Bellman Optimality Equation	Value Iteration

- Algoritmii se bazează pe funcțiile de evaloare a stării $v_\pi(s)$ si $v_*(s)$
- Pentru m acțiuni și n stări, complexitatea: $O(mn^2)$ per iterație
- Pot fi aplicate si pt functiile de evaluare ale actiunilor: $q_\pi(s,a)$ sau $q_*(s,a)$
- Pentru m acțiuni și n stări, complexitatea: $O(m^2n^2)$ per iterație

Programare Dinamica Asincrona

- PD sincrona (pana acum): toate starile sunt backed up in paralel (sincron)
- PD asincrona: Backup-uri asincrone pentru stari individuale
  - PD tip in-place
  - Sweeping prioritizat
  - PD in timp real