Control prin învăţare – Laborator 2: Învăţarea Q

31 martie 2016

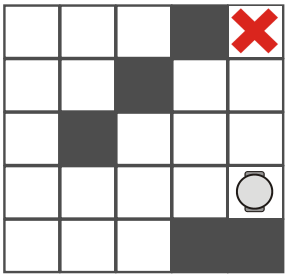
Regulament – acelaşi ca şi la laboratorul precedent, cu excepţia termenului limită pentru predarea soluţiei, care este **joi 14 aprilie 2016**.

Introducere

Codul care formează baza laboratorului este acelaşi ca şi la primul laborator; în plus, un şablon pentru algoritmul de învăţare Q, care poate servi ca punct de plecare pentru implementare, este disponibil în fişierul `qlearning.m`; şi un script pentru a calcula funcţia Q optimală, `gridnav_nearoptsol`, de asemenea furnizat. Codul poate fi descărcat de la:

`http://busoniu.net/teaching/ci2016`

secţiunea "Practical assignments". Dezarhivaţi codul într-un director de pe calculatorul dvs, navigaţi din MATLAB în acest director şi rulaţi scriptul `startup`. Codul poate fi acum folosit.



Vom considera din nou problema de navigaţie 2D, în care un robot trebuie să găsească cea mai scurtă cale înspre o ţintă (X roşu), evitând obstacolele (pătrate gri). Starea robotului este $x = [x_1, x_2]^T \in \{1, 2, 3, 4, 5\} \times \{1, 2, 3, 4, 5\}$; $x = [1, 1]^T$ reprezintă colţul stânga-jos în figură. Robotul se poate deplasa într-una dintre cele patru direcţii cardinale, iar aceste patru acţiuni u sunt reprezentate prin numerele 1 (stânga), 2 (dreapta), 3 (jos), 4 (sus). Orice deplasare care rezultă în lovirea unui perete sau obstacol eşuează, şi în acest caz robotul rămâne în poziţia precedentă. Robotul primeşte o recompensă -0.1 la fiecare pas în care nu a atins ţinta (inclusiv la coliziuni), şi 10 când o atinge, scopul fiind atingerea ţintei în cel mai scurt timp. Ţinta este o stare terminală, aşadar când ea este atinsă episodul se termină şi robotul este resetat într-o poziţie iniţială. Reamintim funcţiile MATLAB care creează modelul problemei de navigaţie, (`gridnav_problem`), simulează tranziţiile şi calculează recompensele (`gridnav_mdp`), şi vizualizează grafic elementele problemei (`gridnav_visualize`).

Partea 1

Implementaţi învăţarea Q cu explorare greedy, introdusă în cursul 4. Algoritmul trebuie să ruleze pentru un număr dat de traiectorii (en. *trials*), T. Fiecare traiectorie se termină la atingerea stării ţintă terminale, sau după un număr maxim de paşi K. Această a doua limitare este impusă în cazul în care starea terminală nu este atinsă într-un interval rezonabil.

Algoritmul trebuie implementat într-o funcţie care acceptă la intrare parametrii experimentului şi ai algoritmului de învăţare Q: rata de învăţare, probabilitatea de explorare, factorul de discount, numărul de

traiectorii, numărul maxim de paşi pe traiectorie etc.[1]. Algoritmul trebuie să returneze (a) *o **secvenţă** de funcţii Q*, conţinând funcţiile Q găsite la sfârşitul fiecărei traiectorii, şi (b) *returnurile cu discount* $R(x_0)$ acumulate de-a lungul fiecărei traiectorii. Fiecare return este calculat relativ la starea iniţială x_0 a traiectoriei, iar valorile în traiectorii multiple sunt comparabile doar dacă x_0 este acelaşi pentru fiecare traiectorie. Suma de recompense, neputând fi infinită în practică, se opreşte la ultimul pas al traiectoriei indiferent dacă starea terminală este atinsă sau nu. **[3p]**

Puteţi porni implementarea de la şablonul furnizat pentru algoritmul de învăţare Q, `qlearning.m`.

Demonstraţi că funcţia creată funcţionează corect aplicând-o pentru un discount γ între 0.9 şi 0.99 la problema de navigaţie 2D. Valori sugerate pentru parametri: $T = 100$, $K = 1000$, $\alpha = 0.1$ (constant), $\varepsilon = 0.3$ (constant). Starea poate fi iniţializată la o valoare care necesită traiectorii dificile şi informative, de exemplu în colţul stânga-jos $x_0 = [1, 1]^{\mathrm{T}}$ (cel mai departe de ţintă). Vizualizaţi online poziţia robotului în timp ce acesta învaţă. Opţional, vizualizaţi de asemenea funcţiile Q de la sfârşitul fiecărei traiectorii, precum şi legile de control greedy corespunzătoare. Folosiţi funcţia `gridnav_visualize`. De reamintit că trebuie să refolosiţi obiectul "view" creat. **[1p]**

Partea 2

Implementaţi o secvenţă de explorare care descreşte exponenţial. Mai exact, probabilitatea de explorare ε este iniţializată la o valoare mare, de exemplu 1, ţinută constantă de-a lungul fiecărei traiectorii, iar la sfârşitul unei traiectorii scăzută cu formula $\varepsilon \leftarrow \varepsilon \cdot d$ unde d este rata de scădere.

Încercaţi câteva rate de scădere în intervalul 0.9 – 0.99. Folosiţi o rată de învăţare $\alpha = 0.1$ constantă, $T = 100$ traiectorii, şi $\gamma = 0.98$. Iniţializaţi robotul tot timpul în aceeaşi stare. Pentru a economisi timp, dezactivaţi vizualizarea când rulaţi aceste experimente (experimentele vor rula mult mai rapid dacă vizualizarea nu trebuie actualizată). Pentru fiecare valoare a lui d, folosiţi secvenţele de funcţii Q şi de returnuri produse de algoritm pentru a crea două grafice:

1. Primul grafic arată distanţa între funcţiile Q găsite la sfârşitul fiecărei traiectorii şi funcţia Q optimală. Graficul are numărul traiectoriei pe orizontală şi distanţa între funcţiile Q pe verticală. **[1p]** Pentru a găsi funcţia Q optimală, folosiţi scriptul `gridnav_nearoptsol`. Pentru a calcula distanţa între două funcţii Q, folosiţi norma 2 transformând întâi tabelul Q într-un vector, vezi funcţia `norm` în Matlab. (Norma 2 este mai informativă în acest context decât norma ∞.)
2. Al doilea grafic arată returnul obţinut de robot de-a lungul fiecărei traiectorii. Acest grafic are numărul traiectoriei pe orizontală şi returnul pe verticală. **[1p]**

Discutaţi care rată de scădere ar fi mai bună: pentru precizia funcţiei Q, pentru return, şi pentru un compromis rezonabil între ambele. Explicaţi relaţia între cele două grafice şi motivul pentru care intervine acest compromis. **[2p]**

În raport, descrieţi pe scurt problemele pe care le-aţi rezolvat, alegerile semnificative pe care le-aţi făcut în implementarea algoritmilor (nu descrieţi codul linie cu linie – codul complet trebuie inclus oricum în raport), şi mai în detaliu studiile cerute. Nu uitaţi mai ales să includeţi studiul şi discuţiile din partea 2, cu toate graficele pentru valorile alese ale ratei de scădere.

[1] O sugestie opţională: Pentru mai multă flexibilitate şi simplitate, puteţi folosiţi o structură Matlab (`struct`) pentru a furniza toţi aceşti parametri, în loc de variabile separate