

Control prin învățare: Laborator 1

Procese de decizie Markov. Iterația Q și pe legea de control

Regulament

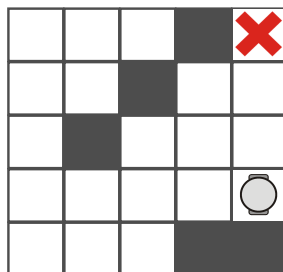
- Acest laborator este o parte obligatorie a cursului “Control prin învățare”. Soluția se notează de la 0 la 10.
- Laboratorul se rezolvă preferabil în grupuri de câte doi studenți (grupurile mai mari nu sunt permise). Un grup predă o singură soluție, comună între cei doi studenți.
- Soluția constă din (1) un scurt raport scris în română sau în engleză și (2) codul MATLAB asociat soluției. Aceste două elemente vor fi trimise prin email la adresa lucian.busoniu@aut.utcluj.ro. (1) Raportul va fi în format PDF. *Este obligatoriu să includeți în raport listinguri complete ale codului Matlab dezvoltat* (doar codul dezvoltat de dvs., nu și codul comun pus la dispoziție de către profesor la începutul laboratorului). Prima pagină a raportului va include numele studenților din grup. Explicați clar în raport diviziunea lucrului între cei doi studenți. (2) Codul MATLAB va fi arhivat într-un fișier ZIP.
- Termenul limită pentru predarea soluției este în **31 martie 2024 până la ora 24:00**. Predarea după termen este acceptată, dar nota maximă scade la jumătate: 5 puncte. Timpul necesar rezolvării laboratorului depinde de experiența cu programarea în Matlab, dar va fi necesar ca o parte din lucru să fie efectuată acasă. Vă este insistent recomandat să nu așteptați până imediat înainte de termen pentru a finaliza soluția.
- Discutarea ideilor între grupuri este încurajată, dar reutilizarea codului sau raportului (fie și parțială) nu este permisă. **Încălcarea acestei reguli (copierea) duce imediat la pierderea dreptului de participare la examen.**
- O sesiune de discuții va fi organizată unde contribuția fiecărui student va fi determinată prin întrebări tehnice detaliate, inclusiv legate de implementarea în Matlab.

Tema de laborator

Descărcați codul ce formează baza primului laborator de pe site-ul cursului, secțiunea “Practical assignments”. Dezarhivați codul într-un director de pe calculatorul dvs, navigați din MATLAB în acest director și rulați scriptul `startup`. Codul poate fi acum folosit.

Partea 1

Obiectivul primei părți este familiarizarea cu interfața pe care o furnizează implementarea în Matlab ale problemei considerate; precum și recapitularea conceptelor de bază în procesele de decizie Markov. A se vedea prezentarea primului curs, care poate fi descărcată la adresa de mai sus.



Vom considera problema reprezentată în figură – o abstractizare a unei probleme reale de navigație robotică. În această problemă, un robot trebuie să găsească cea mai scurtă cale înspre o țintă (X roșu), evitând obstacolele (pătrate gri). Poziția robotului variază pe o grilă de dimensiunea 5×5 , cu starea $x = [x_1, x_2]^T \in \{1, 2, 3, 4, 5\} \times \{1, 2, 3, 4, 5\}$ (T înseamnă că vectorul este transpus). De notat că starea $[1, 1]^T$ reprezintă colțul stânga-jos în figură. Robotul se poate deplasa la fiecare pas câte o celulă într-una dintre cele patru direcții cardinale. Aceste patru acțiuni u sunt reprezentate prin numerele 1 (stânga), 2 (dreapta), 3 (jos), 4 (sus). Orice deplasare care rezultă în lovirea unui perete sau obstacol eșuează și robotul rămâne în aceeași celulă. Robotul primește o recompensă -0.1 la fiecare pas în care nu a atins ținta, și 10 când o atinge. Ținta este o stare terminală, așadar când ea este atinsă episodul se termină și robotul este resetat într-o poziție inițială. Recompensa negativă reprezintă consumul de energie și duce la o soluție de timp minim, i.e. o cale de lungime minimă.

Sarcinile sunt:

- Identificați: variabilele de stare și spațiul stărilor, variabilele de acțiune și spațiul acțiunilor. Scrieți folosind formule matematice funcția de tranziție și funcția de recompensă. **[2p]**
- Familiarizați-vă cu modul de funcționare al funcțiilor MATLAB care creează modelul problemei de navigație, (`gridnav_problem`), care simulează tranzițiile și calculează recompensele (`gridnav_mdp`), și care vizualizează grafic elementele problemei (`gridnav_visualize`). În acest scop, studiați scriptul furnizat ca exemplu `gridnav_example`, precum și comentariile și codul funcțiilor de mai sus.
- Creați o problemă de navigație alegând pozițiile obstacolelor pentru a face problema interesantă. Simulați o traiectorie generând acțiunile după cum doriți, de exemplu aleator. Puteți folosi modul 'reset' al funcției `gridnav_problem` pentru a inițializa starea. Verificați când starea terminală a fost atinsă și opriți traiectoria în acest caz. Limitați de asemenea lungimea traiectoriei la o valoare maximă rezonabilă, pentru cazul în care starea terminală nu este atinsă. La fiecare pas, vizualizați robotul folosind `gridnav_visualize`; mișcarea robotului pe grilă va fi arătată. De notat că trebuie să refolosiți obiectul "view" creat de `gridnav_visualize`! Altfel reprezentarea grafică va fi recreată la fiecare apel, o operație inutilă care ia mult timp. **[1p]**

Partea 2

În a doua parte a laboratorului, vom implementa și testa iterația Q și iterația pe legea de control, pentru problema de navigație descrisă în partea 1.

- Implementați (1) iterația Q și (2) iterația pe legea de control. În cadrul iterației pe legea de control, folosiți algoritmul iterativ de evaluare a legii de control. Acești algoritmi au fost prezentați în detaliu în cursul 2 (prezentarea este disponibilă pe pagina cursului). Algoritmii primesc la intrare factorul de discount și pragurile pentru oprirea algoritmilor (ϵ_{qiter} , ϵ_{hiter} , ϵ_{heval}), și produc la ieșire funcția Q optimă Q^* și legea de control optimă h^* . Funcțiile create trebuie să fie suficient de generale pentru orice configurație de țintă și obstacole. **[3p]**
- Demonstrați că funcțiile MATLAB implementate funcționează corect aplicându-le pentru o valoare $\gamma = 0.95$. Reprezentați grafic funcția Q optimă, folosind de exemplu funcția MATLAB `mesh` (funcția Q are 3 argumente, deci puteți de exemplu să o reprezentați separat pentru fiecare valoare a acțiunii). Produceți de asemenea o reprezentare a legii de control optimale, folosind de exemplu `gridnav_visualize`. **[1p]**

Opțional, pentru iterația Q, cu `gridnav_visualize` puteți vizualiza în timp ce algoritmul rulează o reprezentare sintetică a funcțiilor Q calculate pe figura reprezentând grila 2D a robotului (ca nivele de culoare ale celulelor, fiecare nivel reprezentând $\max_u Q(x, u)$ pentru celula x); precum și legile corespunzătoare de control (ca săgeți în fiecare celulă). Pentru iterația pe legea de control, puteți vizualiza similar legile de control calculate de algoritm și funcțiile lor Q.

Partea 3

În a treia și ultima parte a laboratorului vom investiga comportamentul celor doi algoritmi implementați și al soluțiilor produse.

- Comparați (a) numărul de iterații și (b) timpul de execuție ale iterației Q cu cele ale iterației pe legea de control. Folosiți comenzile MATLAB `tic` și `toc` pentru a măsura timpul de execuție. Pentru iterația pe legea de control, considerați pe de o parte iterațiile majore ℓ , și pe de alta iterațiile minore τ . Interpretați rezultatele în contextul comparației și discuției de la curs. **[1.5p]**
- Schimbați treptat valoarea lui γ de la 0.6 la 0.99. Rulați unul dintre algoritmi pentru a obține funcțiile Q optimale și legile de control optimale pentru valorile γ încercate. Discutați semnificația lui γ și evoluția soluțiilor optimale cu valoarea lui γ . Se schimbă natura legii de control optimale? De ce / de ce nu? **[1.5p]**

În raport, descrieți pe scurt problemele pe care le-ați rezolvat, alegerile semnificative pe care le-ați făcut în implementarea algoritmilor (nu descrieți codul linie cu linie – codul complet trebuie inclus oricum în raport), și includeți câteva figuri reprezentative (de exemplu soluțiile optime, dar nu toată secvența de funcții Q produsă de algoritmi). Nu uitați să includeți studiul și discuțiile din partea 3.