Control prin învăţare: Laborator 1
Procese de decizie Markov. Programarea dinamică

Regulament

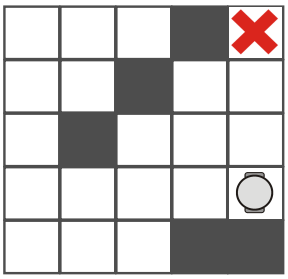
- Acest laborator este o parte obligatorie a cursului "Control prin învăţare". Soluţia se notează de la 0 la 10 şi intră cu pondere de 20% în nota finală.
- Laboratorul se rezolvă preferabil în grupuri de câte doi studenţi (grupurile mai mari nu sunt permise). Studenţii vor stabili în prealabil o diviziune echitabilă a lucrului. Un grup predă o singură soluţie, comună între cei doi studenţi.
- Soluţia constă din (1) un scurt raport scris în română sau în engleză şi (2) codul MATLAB asociat soluţiei. Aceste două elemente vor fi trimise prin email la adresa lucian@busoniu.net. (1) Raportul va fi în format PDF. *Este obligatoriu să includeţi în raport listinguri complete ale codului Matlab dezvoltat* (doar codul dezvoltat de dvs., nu şi codul comun pus la dispoziţie de către profesor la începutul laboratorului). Prima pagină a raportului va include numele studenţilor din grup. Explicaţi clar în raport diviziunea lucrului între cei doi studenţi. (2) Codul MATLAB va fi arhivat într-un fişier ZIP.
- Termenul limită pentru predarea soluţiei este de două săptămâni după efectuarea laboratorului: **24 martie 2021 până la ora 24:00**. Predarea după termen este acceptată, dar nota maximă scade la jumătate: 5 puncte. Timpul necesar rezolvării laboratorului depinde de experienţa cu programarea în Matlab, dar va fi necesar ca o parte din lucru să fie efectuată acasă. Vă este insistent recomandat să nu aşteptaţi până imediat înainte de termen pentru a finaliza soluţia.
- Discutarea ideilor între grupuri este încurajată, dar reutilizarea codului sau raportului (fie şi parţială) nu este permisă. **Încălcarea acestei reguli (copierea) duce imediat la pierderea dreptului de participare la examen**.
- O sesiune de discuţii va fi organizată unde contribuţia fiecărui student va fi determinată prin întrebări tehnice detaliate, inclusiv legate de implementarea în Matlab.

Tema de laborator

Descărcaţi codul ce formează baza primului laborator de pe site-ul cursului, 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.

Partea 1

Obiectivul primei părţi este familiarizarea cu interfaţa pe care o furnizează implementările în Matlab ale problemelor ce vor fi considerate în laboratoarele următoare; precum şi 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 navigaţie robotică. În această problemă, un robot trebuie să găsească cea mai scurtă cale înspre o ţintă (X roşu), evitând obstacolele (pătrate gri). Poziţia 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 colţul stânga-jos în figură. Robotul se poate deplasa la fiecare pas câte o celulă într-una dintre cele patru direcţii cardinale. 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 robotul rămâne în aceeaşi celulă. Robotul primeşte o recompensă -0.1 la fiecare pas în care nu a atins ţinta, şi 10 când o atinge. Ţinta este o stare terminală, aşadar când ea este atinsă episodul se termină şi robotul este resetat într-o poziţie iniţială. Recompensa negativă reprezintă consumul de energie şi duce la o soluţie de timp minim, i.e. o cale de lungime minimă.

Sarcinile sunt:

- Identificaţi: variabilele de stare şi spaţiul stărilor, variabilele de acţiune şi spaţiul acţiunilor. Scrieţi folosind formule matematice funcţia de tranziţie şi funcţia de recompensă. **[2p]**

- Familiarizaţi-vă cu modul de funcţionare al funcţiilor MATLAB care creează modelul problemei de navigaţie, (`gridnav_problem`), care simulează tranziţiile şi calculează recompensele (`gridnav_mdp`), şi care vizualizează grafic elementele problemei (`gridnav_visualize`). În acest scop, studiaţi scriptul furnizat ca exemplu `gridnav_example`, precum şi comentariile şi codul funcţiilor de mai sus.

- Creaţi o problemă de navigaţie alegând poziţiile obstacolelor pentru a face problema interesantă. Simulaţi o traiectorie generând acţiunile după cum doriţi, de exemplu aleator. Folosiţi modul 'reset' al funcţiei `gridnav_problem` pentru a iniţializa starea. Verificaţi când starea terminală a fost atinsă şi opriţi traiectoria în acest caz. Limitaţi de asemenea lungimea traiectoriei la o valoare maximă rezonabilă, pentru cazul în care starea terminală nu este atinsă. La fiecare pas, vizualizaţi robotul folosind `gridnav_visualize`; mişcarea robotului pe grilă va fi arătată. De notat că trebuie să refolosiţi obiectul "view" creat de `gridnav_visualize`! Altfel reprezentarea grafică va fi recreată la fiecare apel, o operaţie inutilă care ia prea mult timp. **[1p]**

Partea 2

În a doua parte a laboratorului, vom implementa şi testa iteraţia Q şi iteraţia pe legea de control, pentru problema de navigaţie descrisă în partea 1.

- Implementaţi (1) iteraţia Q şi (2) iteraţia pe legea de control. În cadrul iteraţiei pe legea de control, folosiţi algoritmul iterativ de evaluare a legii de control. Aceşti algoritmi au fost prezentaţi în detaliu în cursul 2 (prezentarea este disponibilă pe pagina cursului). Algoritmii primesc la intrare factorul de discount şi pragurile pentru oprirea algoritmilor ($\varepsilon_{\text{qiter}}, \varepsilon_{\text{hiter}}, \varepsilon_{\text{heval}}$), şi produc la ieşire funcţia Q optimală Q^* şi legea de control optimală h^*. Funcţiile create trebuie să fie suficient de generale pentru orice configuraţie de ţintă şi obstacole. **[3p]**

- Demonstraţi că funcţiile MATLAB implementate funcţionează corect aplicându-le pentru o valoare $\gamma = 0.95$. Reprezentaţi grafic funcţia Q optimală, folosind de exemplu funcţia MATLAB `mesh`. Produceţi de asemenea o reprezentare a legii de control optimale, folosind de exemplu `gridnav_visualize`. **[1p]**

  Opţional, pentru iteraţia Q, cu `gridnav_visualize` puteţi vizualiza în timp ce algoritmul rulează o reprezentare sintetică a funcţiilor 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 şi legile corespunzătoare de control (ca săgeţi în fiecare celulă). Pentru iteraţia pe legea de control, puteţi vizualiza similar legile de control calculate de algoritm şi funcţiile lor Q.

Partea 3

În a treia şi ultima parte a laboratorului vom investiga comportamentul celor doi algoritmi implementaţi şi al soluţiilor produse.

- Comparaţi (a) numărul de iteraţii şi (b) timpul de execuţie ale iteraţiei Q cu cele ale iteraţiei pe legea de control. Folosiţi comenzile MATLAB `tic` şi `toc` pentru a măsura timpul de execuţie. Pentru iteraţia pe legea de control, consideraţi pe de o parte iteraţiile majore ℓ, şi pe de alta iteraţiile minore τ. Interpretaţi rezultatele în contextul comparaţiei şi discuţiei de la curs. **[1.5p]**

- Schimbaţi treptat valoarea lui γ de la 0.6 la 0.99. Rulaţi unul dintre algoritmi pentru a obţine funcţiile Q optimale şi legile de control optimale pentru valorile γ încercate. Discutaţi semnificaţia lui γ şi evoluţia soluţiilor optimale cu valoarea lui γ. Se schimbă natura legii de control optimale? De ce / de ce nu? **[1.5p]**

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 includeţi câteva figuri reprezentative (de exemplu soluţiile optime, dar nu toată secvenţa de funcţii Q produsă de algoritmi). Nu uitaţi să includeţi explicit studiul şi discuţiile din partea 3.