Tema 2 - Invăţare prin recompensa

Data publicare:
Deadline: 9.04.2020
Se acceptă teme trimise până la data de 26.04.2020. Se aplica 0.5 puncte depunctare în prima zi de intarziere, si cate 1 punct pentru fiecare din următoarele două. (din max 10 puncte)

Se consideră o lume de tip grid, de dimensiune NxM. În interiorul gridului pot exista bucăţi de branza (plasate în celulele gridului). In aceasta lume exista un soarece si o pisica. Scopul soarecelui este să adune toate bucăţile de branză fără a fi mancat de pisică. Soarecele si pisica se pot deplasa în direcţiile N, E, S, V, fără a putea trece celulele-obstacol. Dacă pisica ajunge prea aproape de soarece (maxim A paşi între ei) aceasta îl va urmări. În caz contrar, pisica executa mişcări aleatoare.

Modelati lumea descrisă în problemă (gridul şi mişcarea pisicii) şi ajutaţi soarecele să înveţe cum să adune toate bucăţile de branză fără a fi mancat de pisică, folosind algoritmul de învăţare prin recompensă Q-learning. Programul trebuie să poată fi rulat în două moduri: pas cu pas sau execuţie continuă. Afişarea poate fi făcută grafic sau în mod text, dar sa fie cat mai realistă: să fie vizibile celulele gridului, obstacolele, bucăţile de branză, şoarecele şi pisica, precum şi deplasările celor doua animale.

Se vor experimenta 3 strategii de exploatare / explorare:

- Max First: exploatare pură, acţiunea cu utilitatea maximă va fi aleasă de fiecare data
- Random: ignoră tabela de utilitati si alege aleator o actiune posibila
- Exploatare:atata timp cat sunt acţiuni nefolosite într-o stare, se va alege aleator dintre acestea
- Explorare / exploatare ponderata: permite echilibrarea explorării cu exploatarea folosind o probabilitate pentru fiecarea acţiune bazată pe valoarea utilităţii acesteia

Descrierea gridului este preluata dintr-un fisier text cu urmatorul format:

N M
N*M valori 0, 1 sau 2 // 0 reprezintă celula libera, 1 reprezintă obstacol,
//2 reprezintăţ bucata de branză
A //numarul de pasi maxim pentru care putem spune că soarecele este prea aproape de pisică

xs ys // pozitia inițială a soarecelui in grid
xp yp // pozitia inițială a pisicii in grid

Se cer:

- [2.5p] Implementarea jocului (Pentru implementare puteți folosi Gym)
  - Numărul de bucăți de branză si dimensiunea habitatului să poată fi variat
  - Implementarea trebuie sa permita rularea pas cu pas (după antrenare)
- [6.5p] Implementarea sistemului
  - [3.5p] Algoritmul Q-Learning
    - [1p] Parametri variabili (rata de invatare, factor de discount)
    - [0.5p] MaxFirst
    - [0.5p] Random
    - [0.5p] Exploatare
    - [1.5p] Explorarea ponderata
  - [3p] Grafice (pentru toate strategiile de exploatare)
    - [1p] Evolutia scorului în funcție de numărul episodului de antrenament
    - [2p] Procentul de jocuri castigate în funcție de valoarea (jocurile vor fi rulate in batchuri):
      - factorului de învățare
      - factorului de discount
- [1p] Grafice comparative:
    - [0.5p] Cum afectează numărul de episoade de antrenament valorile din tabela de utilitati în cazul strategiei maxfirst?
    - [0.5p] Care sunt diferențele intre tabela de utilitati din cazul strategiei maxfirst și random?

Observație:

- pozițiile inițiale ale prăzii, prădătorilor și capcanelor vor fi aleatoare fără a exista coliziuni inițiale

Bonus [2 puncte]:

Implementarea algoritmului de învățare prin recompensa SARSA și trasarea curbei de învățare Q-learning - SARSA (de exemplu număr episod - recompensa totală pentru acel episod - sau alte reprezentări pentru a putea pune în evidență eventualele diferențe dintre cei doi algoritmi pentru acest caz). Curba de învățare poate fi reprezentată si sub forma tabelară.

Arhiva: Se va încărca o arhivă care conține 2 fișiere:

- codul python / notebook-ul aferent rezolvării temei
- un fișier PDF în care sunt trecute: un readme al implementării, graficele cerute și analiza acestora (text explicativ). Numele arhivei trebuie va fi de forma: Tema2_<nume>_<prenume>_<grupa>.