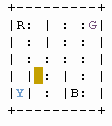
**TAXI V3-ucenje sa podsticajem**



OPIS:

Postoje 4 dizajnirane lokacije u polju(R(ed), G(reen), Y(ellow), B(lue)). Kada pocne epizoda , taksi se nalazi na random poziciji sto vazi takodje i za putnika. Taksi vozi do putnikove lokacije , pokupi putnika , vozi ga do zeljene destinacije (preostala od 4 specijalno dizajnirane lokacije) I onda ostavi putnika. Kada je putnik jednom ostavljen epizoda je zavrsena.

Postoji 500 diskretnih stanja . Imamo 25 mogucih lokacija za taksi , 5 mogucih lokacija putnika (ukljucujuci i slucaj da je putnik u taksiju) i 4 krajnjih destinacija.

PUTNIKOVA LOKACIJA:

1. Red
2. Green
3. Yellow
4. Blue
5. In taxi

KRAJNJE DESTINACIJE:

1. Red
2. Green
3. Yellow
4. Blue

NAGRADE:

-1 po svakom koraku osim ako se ne pokrene druga nagrada

+20 isporuka putnika do zeljene lokacija

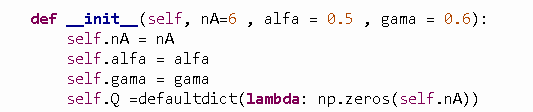
-10 napravilno izvrsavanje akcija “pokupi” I “odbaci”

Za realizaciju ovog ucenja sa podsticajem koristili smo Q-ucenje. Napravili smo fajlove: agent.py, env.py, monitor.py i main.py uz sve to smo koristili gym okruzenje .

Q-ucenje nastoji da nauci kako da maksimizuje ukupnu nagradu. Tacnije u nasem slucaju poenta je da u sto manjem broju koraka dodjemo do cilja I osvojimo sto veci broj poena.

AGENT.PY

Agent random bira akciju koju ce izvrsiti. Agent u interakciji sa oktuzenjem uci koje su akcije bolje sto je bazirano na nagradama koje dobija. On koristi svoje iskustvo da azurisa vrednosti.



nA- number of Actions (broj akcija)

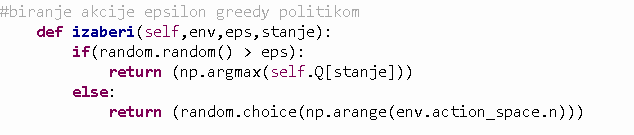
nA=6

1. move south
2. move north
3. move east
4. move west
5. pickup passenger
6. drop of passenger

Alfa – brzina ucenja Lr(learning rate) ili stopa ucenja . Definise kako prihvatamo novu vrednost u odnosu na staru.

Gama- factor popusta. Koristi se za balansiranje trenutne ili buduce nagrade. Vrednost se obicno krece od 0.8 do 0.99

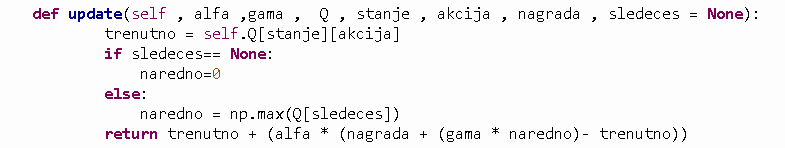
Pravljenje q-tabele tj matrice oblika  *[state, action]*



Agent reaguje sa okruzenjem na 2 nacina:

1. Koristi q-tabelu kao reference I vidi sve moguce kacije za dato stanje. Agent onda bira akciju na osnovu maksimalne(max) vrednosti tih akcija. To nazivamo EXPLOATING – iskoriscavanje jer koristimo informacije koje sun am dostupne za donosenje odluke
2. Preduzima akcije slucajno (random). To se naziva EXPLORING – istrazivanje. Tako nasumicno biramo akciju. Nasumicno delovanje je vazno jer omogucava agentu da otkriva nova stanja koja inace ne bi bila izabrana tokom eksploatacije.

Istrazivanje se moze uravnoteziti koristeci epsilon (eps) – Epsilon greedy policy



Azuriranje Q tabele:

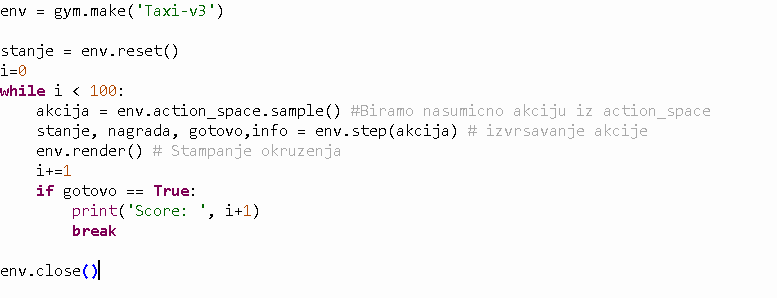
Azuriranje se desava nakon svakog koraka i zavrsavaju se kada se epizoda zavrsi. U ovom slucaju to znaci da dolazi do neke krajnje tacke agenta. To stanje moze biti bilo sta sto ce dovesti do kraja igre tj kraja ispunjavanja zeljenog cilja.

Agent nece nauciti mnogo posle jedne episode , ali na kraju sa dovoljno istrazivanja (koraka i epizoda ) napredovace i nauciti optimalne vrednosti Q ili Q\*

Osnovni koraci:

1. Agent pocinje u stanju (s1), preduzima akciju (a1) I dobija nagradu (r1)
2. Agent bira akciju pozivajuci se na q-tabelu sa najvecom vrednoscu (max) ili random
3. Azuriranje q vrednosti  *Q[state, action]= Q[state,action] +lr\*(reward + gamma\* np.max(Q[new\_state])- Q[state,action] )*

ENV.PY



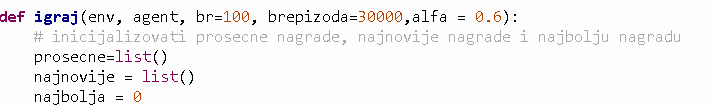
Koristimo gym okruzenje za pravljenje naseg env okruzenja za ‘Taxi-v3’

U rangu od 100 epizoda prvo izaberemo nasumicno akciju, izvrsavamo je , prikazujemo okruzenje I stampamo broj poena.

Nagrada- vrednost dobijena nakon zavrsene radnje u datom stanju. Moze se desiti u bilo kom vremenskom koraku ili krajnjem koraku.

MONITOR.PY

Ovaj fajl sluzi za prikaz agentovih performansi.



Parametri

- env: primer OpenAI Gym's Taxi-3 okruzenja

- agent:primer klase Agent (iz fajla agent.py)

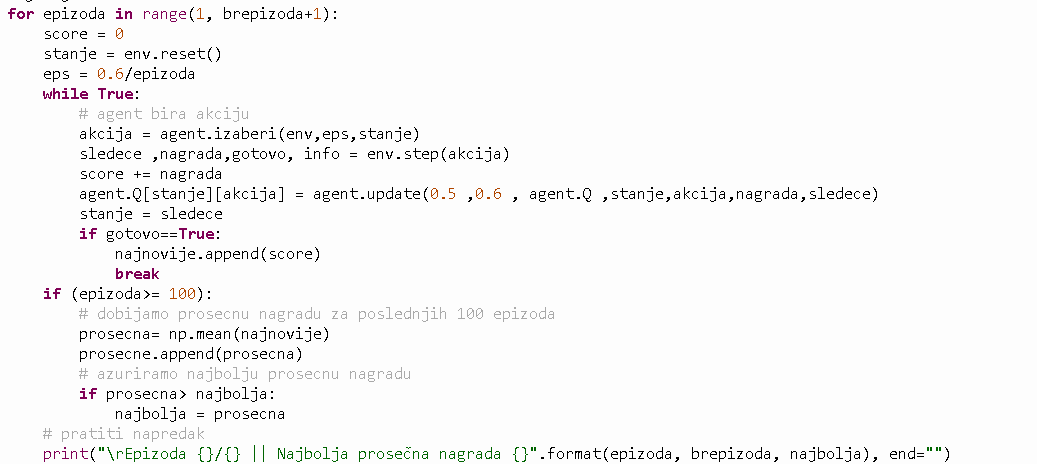
- brepizoda :broj epizoda interakcije agent-okruzenje

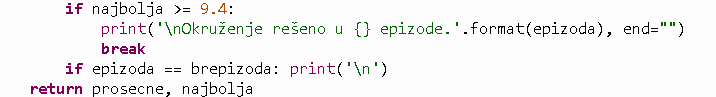
- br: broj epizoda koje treba uzeti u obzir pri izracunavanju prosecnih nagrada

Vraca:

- prosecne: prosecne nagrade

- najbolja: najveca vrednost u nizu prosecne





Kod Q-ucenja koraci idu po sledecem redosledu:

1. Inicijalizacija stanja S

Za svaki korak u epizodi :

1. Izabrati akciju A uz pomoc nekog istrazivacke politike (iz Agent.py) iz stanja S
2. Preduzeti akciju A i posmatrati R-reward I S’(next\_state)
3. Izvrsavamo azuriranje
4. Azuriramo stanje S=S’ I nastavimo da ponavljamo od koraka 2 do kraja epizode

MAIN.PY

U fajlu main.py podesavamo okruzenje , kreiramo agenta I treniramo ga da ostvari sto bolji rezultat u igri u sto manje koraka.

Render funkciju koristimo za prikaz .

Plava- putnik

Ljubicasta-lokacija

Zuta- prazan taksi

Zelena- pun taksi

