Path planning using reinforcement learning

Luka Cerovć, Lazar Lukić

Matematički fakultet

1. Postavka problema

Problem koji rešavamo porazumeva agenta koja treba da se kreće kroz dati prostor za skladištenje, da pokupi kutiju i da je odnese na zadato mesto. Prostor je predstavljen matricom velicne NxM na kojoj postoje polja po kojima agent može da se kreće, dok ostala polja predstavljaju zidove preko kojih agent ne može da prelazi.

2. Učenje potkrepljivanjem

Za rešavanje ovog problema koristićemo algoritam *Q-learning* koji je iz familije algoritama učenja potkrepljivanjem. Algoritam se zasniva na principu nagrade i kazne koje agent dobija da osnovu trenutnog stanja i akcije koju izvrši iz datog stanja. Cilj nam je da agent posle dovoljno iteracija nauči koja akcija je najbolja u nekom proizvoljnom stanju tako da reši zadati problem, sa ciljem da maksimizuje ukupnu nagradu(u našem slučaju da to uradi najkraćim mogućim putem).

2.1 Stanja i akcije

Jedno stanje agenta ima ukupno 7 dimenzija:

- X koordinata agenta
- Y koordinata agenta
- X koordinata ciljne lokacije
- Y koordinata ciljne lokacije
- X koordinata kutije
- Y koordinata kutije
- Indeks nošenja kutije (da li agent kod sebe ima kutiju)

Ovo znači da je ukupan broj stanja u kojima agent može da se nadje: $N^3 * M^3 * 2$.

Agent iz nekog stanja prelazi u sledeće tako što se pomeri gore, dole, levo ili desno za jedno polje u matrici.

Postoje i takozvana terminalna stanja, koja prekidaju iteraciju ukoliko se agent nadje u jednom od njih. U našem slučaju imamo dva takva stanja:

- Ako je agent udario u zid
- Ako je agent došao na cilj i kod sebe ima kutiju

2.2 Nagrade i kazne

Kada agent izvrši neku akciju on dobija odredjenu nagradu (kaznu) koja se akumulira tokm jedne iteracije. Želimo da agent bude nagradjen kada uradi ono što želimo: kada pokupi kutiju, kada je donese na cilj. Želimo da agent bude kažnjen kada udari u zid.

Takođe, cilj nam je da agent uvek ide najkraćim putem i zato ćemo mu davati malu kaznu svaki put kad se pomeri za jedno polje.

Moramo biti oprezni kod politike davanja nagrada (kazni) jer neka stanja su važnija od drugih. Npr: Ne želimo da ga kažnjavamo previše za pomeraj jer neće nikad stići da istraži svoje okruženje. Tako će kazna za zid biti mnogo veća od ostalih.

3. Treniranje agenta

Agenta ćemo trenirati kroz više iteracija. Jedna iteracija traje dok ne dođe u terminalno stanje ili predje dozvoljeni maksimalni broj poteza u jednoj iteraciji. Na početku iteracije agent se postavlja u nasumično stanje tako da se njegova pozicija ne poklopi sa nekim zidom. To podrazumeva poziciju agenta, poziciju kutije I poticiju cilja.

Tokom jedne iteracije, agent u svakom potezu radi sledeće:

1. Agent bira koju akciju da izvrši u trenutnom stanju
2. Predje u sledeće stanje I dobije odgovarajuću nagradu
3. Ažurira vrednosti *Q-matrice*

3.1 *Q-matrica*

Već smo pomenuli da imamo 7 dimenzija jednog stanja. U svakom od tih stanja agent treba da može da odluči koja akcija je najbolja. Takva matrica koja ima vrednost za svaku akciju u svakom stanju je *Q-matrica.* Njena veličina u ovom sučaju je: $N^3 * M^3 * 2 * 4$

3.2 Izbor akcije (*ε-greedy*)

Sada znamo da će agent koristiti *Q-matricu* za izbor akcije iz nekog stanja. Medjutim, šta ako agent još uvek ništa nije naučio? U tom slučaju možemo da biramo nasumičnu akciju. Pitanje je: kako da znamo kada je agent spreman da primeni naučeno? Zato uvodimo hiperparametar *ε* koji će se smanjivati tokom iteracija. *ε* predstavalja verovatnoću da agent izabere nasumičnu akciju u trenutnom stanju. Na ovaj način podstičemo agenta da istražuje u ranim iteracijama, a kasnije da koristi to što je naučio.

3.3 Belmanova jednačina

Za ažuriranje vrednosti *Q-matrice* koristimo *Belmanovu jednačinu*.

$$Q(S_t) = Q(S_{t,a}) + \alpha*[R_{t+1} + \gamma*maxQ(S_{t+1}) - Q(S_{t,a})]$$

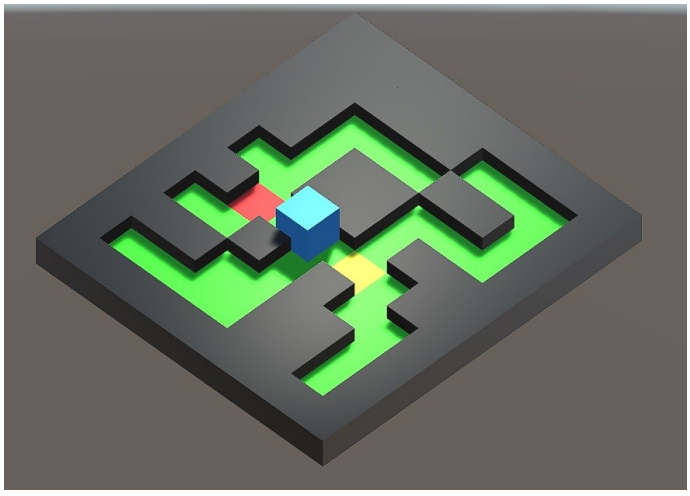
Gde su:

- $Q(S_{t,a})$ – vrednost *Q-matrice* za stanje t i akciju a
- $maxQ(S_{t+1})$ – vrednost *Q-matrice* za najbolju akciju u stanju u koje prelazimo
- R_{t+1} – nagrada za prelaz u stanje t+1
- α – koeficijent učenja (koliko brzo agent uči)
- γ – discount factor (koliko uzimamo od već naučenog)

4. Evaluacija algoritma

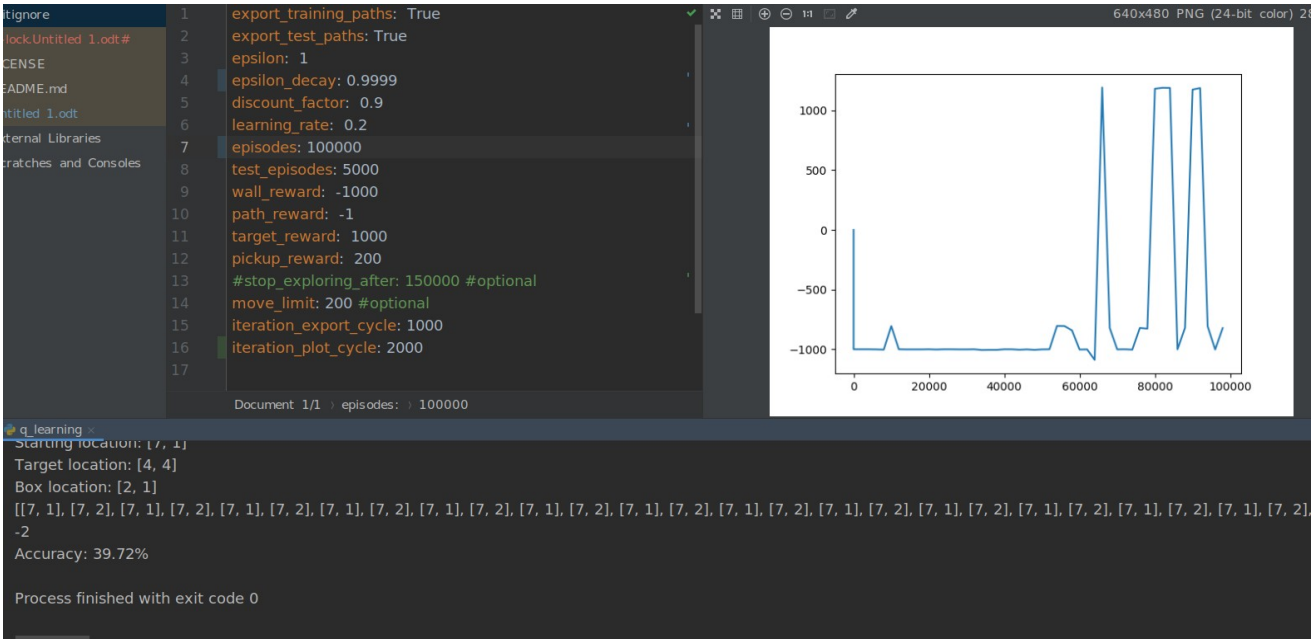
Sada ćemo primeniti algoritam na naš problem i koristeći različite vrednosti parametara.
Na svim testovima prikazaćemo parametre, grafik zavisnosti ukupne nagrade od iteracije kao i preciznost na testu gde se koristi ono što je naučeno.

Mapa okruženja (primer jednog stanja):

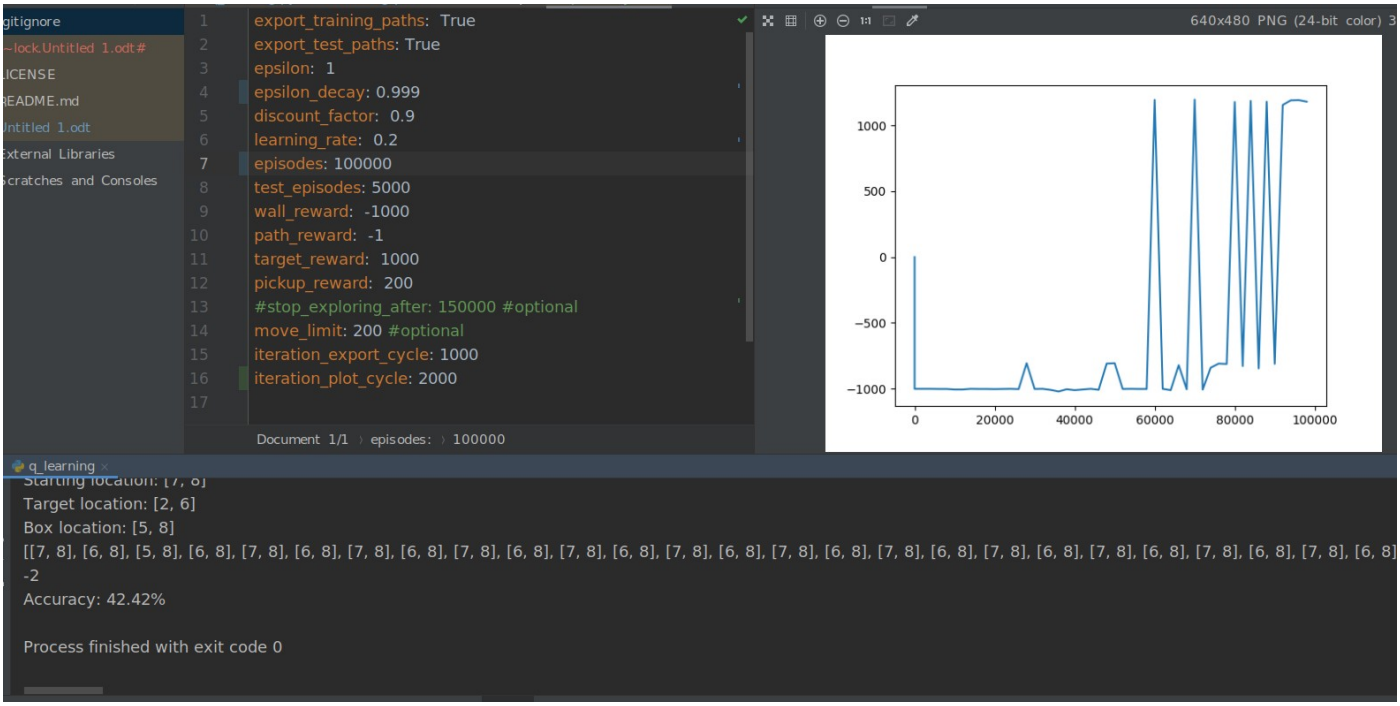


- zidovi su crne boje
- pozicije za kretanje su zelene boje
- agent je kocka
- kutija je crveni kvadrat
- cilj je žuti kvadrat

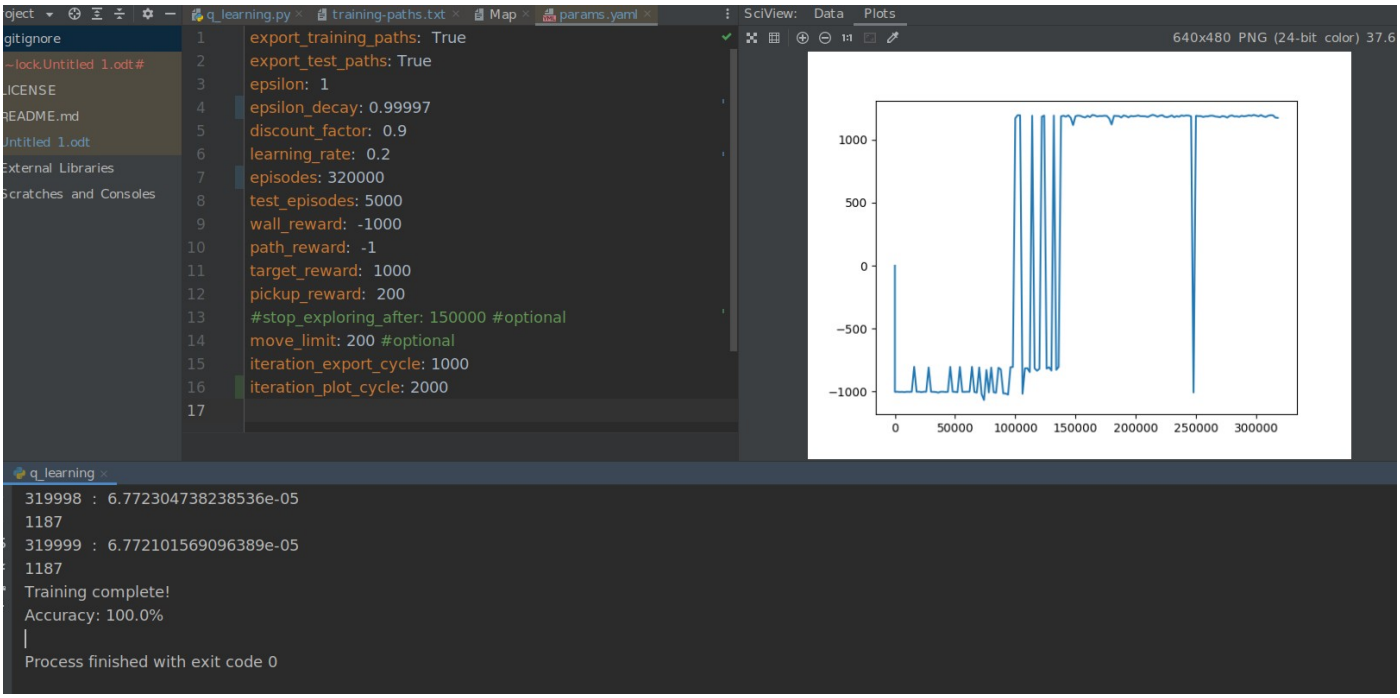
TEST 1 – U ovom slučaju algoritam nema dovoljno iteracija da bi konvergirao. Takodje, ε se previše sporo umanjuje za broj iteracija.



TEST 2 – Ovde je isti broj iteracija ali ε brže opada, malo je bolji rezultat ali daleko od idealnog.



TEST 3 - Ovde smo povećali broj iteracija i usporili opadanje ε. Vidimo da se na vrhu grafika pojavljuju platoi sto znači da su nagrade konstantno visoke. Opadanje koje se nalazi negde oko 250000 se javlja jer ε, čak i u krajnjim iteracijama, daje verovatnoću da se izabere nasumična akcija.



5. Literatura:

- http://incompleteideas.net/book/RLbook2018.pdf