



Katedra za Signale i sisteme
Elektrotehnički fakultet
Univerzitet u Beogradu



Mašinsko učenje

Peti domaći zadatak

Učenje podsticanjem

Student:
Aleksandar Đorđević 2023/3049

Mentor:
dr. Predrag Tadić

1 Uvod

Za razliku od učenja sa nadzorom gde imamo željeni izlaz za svaki ulaz učenje podsticanjem je znatno komplikovanije jer je neophodno na osnovu krajnjeg ishoda (nagrade) odrediti koje su akcije povoljne a koje ne. Ovaj vid učenja se koristi za kreiranje inteligentnih agenata koji su u stanju da igraju igre (npr. šah), ali i da se snalaze u realnim situacija (autonomna vožnja). Neretko se ispostavlja da su ovi agenti znatno uspešnji u obavljanju konkretnog zadatka od bilo kog čoveka.

Okruženje u kome agent živi sastoji se od **stanja** (npr. pozicija figura u šahu), **akcija** (dovoljeni potezi koje agent može povući) i **nagrada**. Duže partije se često penalizuju pomoću faktora umanjenja budućih nagrada $\gamma \in (0, 1)$ (logika iza ovoga je da je pobeda u manjem broju poteza bolja). Agentov cilj je da maksimizuje prosečnu ukupnu nagradu odnosno sledeći izraz.

$$E[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \dots] \quad (1)$$




Rezultat treniranja agenta jeste **politika** $\pi(s)$ (zakon upravljanja) koja za svako stanje govori agentu kako da igra odnosno koju akciju da preduzme. Ona može biti deterministička (u istom stanju agent će uvek preduzeti istu akciju) ili stohastička (za dato stanje politika daje raspodelu verovatnoće mogućih akcija). Determinističke politike su zastupljene u igrama gde nemamo protivnika, dok su stohastičke zbog svoje nepredvidljivosti veoma korisne u igrama sa protivnicima. U zavisnosti od algoritma politika se može direktno učiti (REINFORCE) ili neposredno (Q-learnig) time što će se najpre proceniti koliko su stanja dobra, a politika je da se povuče onaj potez koji nas dovodi u najbolje stanje. U svrhu procene kvaliteta stanja uvodi se Q-vrednost:

$$Q(s, a) = R(s) + \gamma \sum_{s'} P_{sa}(s') V(s') \quad (2)$$

gde $V(s)$ predstavlja očekivanu vrednost nagrade kada je početno stanje s , dok $P_{sa}(s')$ predstavlja verovatnoću da ako u stanju s primenimo akciju a dođemo u stanje s' . Ukoliko nam okruženje nije unapred poznato neophodno ga je najpre pretražiti i na osnovu toga proceniti verovatnoće P_{sa} .

1.1 Okruženje

U ovom zadatku cilj je projektovati agenta koji igra igru na sledećoj slici.

	1	2	3	4	5
A	S				
B					

Slika 1: Okruženje

Agent počinje igru na polju A1. Dostupne akcije su mu gore, dole, levo i desno. Kada primeni akciju agent ima 60% šanse da ode u stranu u koju je hteo, a po 20% da ode u jednu od susednih strana. Ukoliko agent udari u zid ostaje u istom stanju. Red B predstavlja terminalna stanja odnosno stanja u kojima se igra završava. Stanja B1 i B3 predstavljaju poraz (agent dobija nagradu -1), dok je stanje B5 pobedničko (agent dobija nagradu +3).

2 Q-learning

Q-learning se zasniva na proceni Q-vrednosti za svaki par stanje-akcija. Politika se nakon toga dobija izborom akcije koja za dato stanje ima najveću Q-vrednost. Q-learning se često uparuje sa ϵ -gramzivom pretragom. Naime, na početku je okruženje nepoznato odnosno procene za Q-vrednosti su nekvalitetne. Tada nema smisla vući optimalne poteze već je od većeg interesa istraživati okruženje. ϵ -gramziva pretraga funkcioniše tako što postoji ϵ šanse da se povuče nasumičan potez, a inače se povlači optimalan na osnovu do sada procenjenih Q-vrednosti. Na osnovu prethodnog jasno je da ϵ na početku treba biti veliko (uzeto 1) i smanjivati se kroz epizode. Nakon svake epizode smanjeno je za 1% dok ne dođe do 0.01 i nakon toga se ne menja. Kako bi se pratilo kada i da li je algoritam konvergirao računata je V-vrednost za svako neterminalno stanje na svake epizode. Algoritam funkcioniše na sledeći način. Do konvergencije ili u našem slučaju unapred zadat broj epizoda igra se igra. Jedna epizoda igre traje dok se ne stigne do terminalnog stanja. Svaka iteracija se sastoji od izbora da li povlačimo nasumičan ili optimalan potez. Primena tog poteza na okruženje (ažuriranje stanja, provera da li se igra završila i računanje nagrade). Zatim se estimacije za Q-vrednost se ažuriraju po sledećoj formuli:

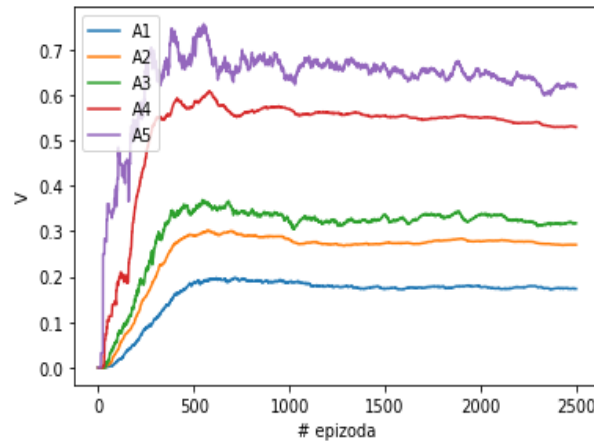
$$Q(s, a) = Q(s, a) + \alpha(R(s) + \gamma \max_{a'}(Q(s', a') - Q(s, a))) \quad (3)$$

gde je α stopa učenja koja može biti konstantna ili promenljiva. Za promenljivu stopu učenja često se koristi sledeća formula.

$$\alpha(e) = \frac{\ln e + 1}{e + 1} \quad (4)$$

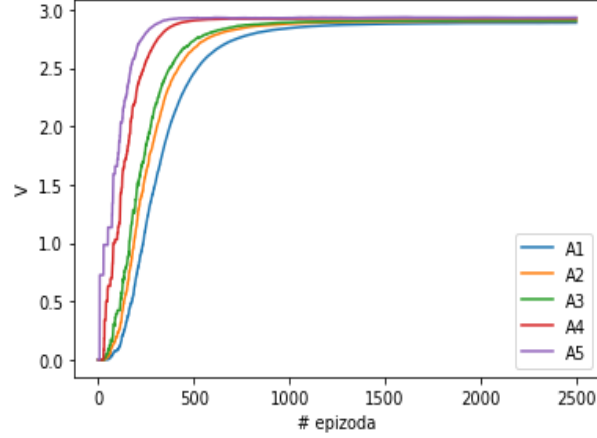
gde je e redni broj epizode.

Uzmimo $\gamma = 0.9$ i α po gorenavedenoj formuli. Politika koja se dobija je sledeća: A1:U, A2:R, A3:U, A4:R, A5:D. Grafik V-vrednosti kroz iteracije izgleda ovako:



Slika 2: V-vrednost kroz epizode

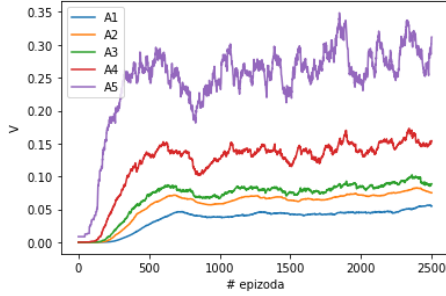
Ako promenimo $\gamma = 0.999$ odnosno praktično ne penalizujemo dodatne poteze politika koja se dobija se: A1:U, A2:R, A3:U, A4:R, A5:R. Zanimljivo je da zbog toga što se ne penalizuje dodatni broj poteza poslednji potez se neretko sa ovim γ dobija da je nadesno što isto dovodi sigurno do poredničkog polja, ali kroz veći broj poteza u proseku. Grafik V-vrednosti izgleda ovako. Možemo primeti da je grafik znatno



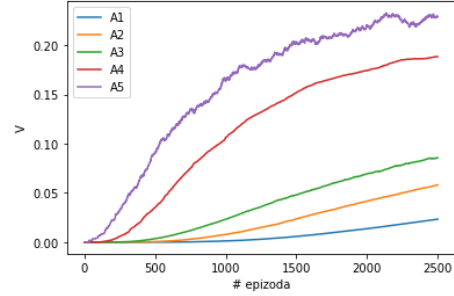
Slika 3: V-vrednost kroz epizode

gladi i brže konvergira što je i logično jer smo smanjili kompleksnost problema. Naime, sada algoritam samo treba da nauči da stigne do poredničkog stanja, ali ne mora da vodi računa koliko brzo.

Za $\gamma = 0.9$ testirano je konstantno α . Evo ih rezultati:



(a) $\alpha = 0.01$



(b) $\alpha = 0.001$

Slika 4: V-vrednosti kroz epizode za konstantnu stopu obučavanja

Za malo α algoritam veoma sporo konvergira, a za veliko se izražavaju velike oscilacije. Zbog ovoga je zaključak da je za ovaj problem promenljivo α bolji izbor.

Testiranje je izvršeno za obe dobijene politike igranjem 10 epizoda. Obe politike uvek ostvare poredbe (nagrada 3), ali politika za $\gamma = 0.9$ to radi u prosečno 14.8 poteza, a politika dobijena za $\gamma = 0.999$ to radi u 20.3 poteza. Ovo je očekivano jer za veće γ se manje penalizuju dodatni potezi.

3 REINFORCE

Za razliku od Q-learning-a REINFORCE algoritam ne uči koja su stanja dobra, već direktno uči politiku koju treba primeniti. Ovo se radi tako što se politika parametrizuje i zatim se treniraju parametri na osnovu gradijenta politike. Prednost ovog algoritma je ta što se u zavisnosti od vrste parametrizacije može primeniti i na problema sa kontinualnim i diskretnim akcijama. Kod problema kontinualnih akcija često se koristi Gausovska parametrizacija, u našem problemu (diskretne akcije) koristimo *softmax* parametrizaciju. Politika je data sledećim izrazom.

$$\pi_{\theta}(s, a) = \frac{e^{\theta_{sa}}}{\sum_{a'} e^{\theta_{sa'}}} \quad (5)$$

Kako imamo pet stanja i četiri moguće akcije potrebno nam je 20 parametara (15 nezavisnih jer za svako stanje jedan parametar se može odrediti na osnovu preostala 3). Prilikom primene algoritma biće nam potreban parcijalni izvod logaritma politike koji izgleda ovako:

$$\frac{\partial}{\partial \theta_{sa}} \ln \pi_{\theta}(s, a) = 1 - \frac{e^{\theta_{sa}}}{\sum_{a'} e^{\theta_{sa'}}} \quad (6)$$

Treniranje pomoću ovog algoritma se zasniva na gradijentnom usponu i teoremi o gradijentu upravljanja.

$$\nabla_{\theta} J(\theta) = E_{\pi_{\theta}} [\nabla_{\theta} \pi_{\theta}(a|s) Q^{\pi_{\theta}}(s, a)] \quad (7)$$

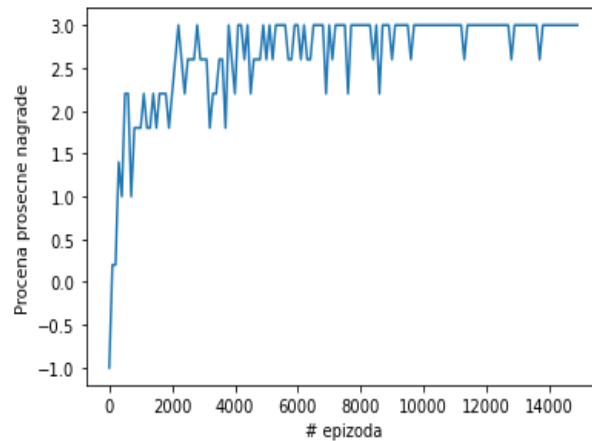
U REINFORCE algoritmu se matematičko očekivanje zanemaruje odnosno procenjuje se kroz jednu iteraciju. Takođe Q-vrednost se procenjuje V-vrednošću.

$$Q^{\pi_{\theta}}(s_t, a_t) \approx v_t = \sum_{\tau=t}^T \gamma^{\tau-t} r_{\tau} \quad (8)$$

Nakon svake epizode za svaki par stanje-akcija koji je odigran u toj epizodi ažuriraju se parametri θ .

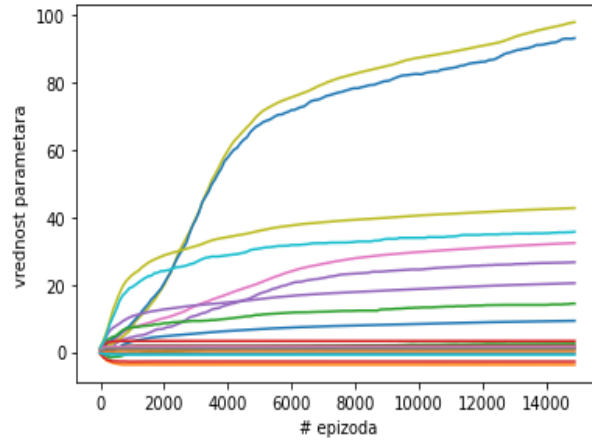
$$\theta \leftarrow \theta + \alpha \nabla_{\theta} \ln \pi_{\theta}(s_t, a_t) v_t \quad (9)$$

gde je α stopa učenja. Za γ je uzeto 0.9, a izabrana stopa učenja je 0.04. Napredak obučavanja je praćen tako što se nako svakih 100 epizoda politika se zamrzne odigra se 10 epizoda i zabeleži se prosečna ostvarena nagrada. Vidimo da algoritam uspeva da ostvari prosečnu nagradu od 3 to jest uspeva uvek da pobedi.



Slika 5: Procena prosečne nagrade kroz obučavanje

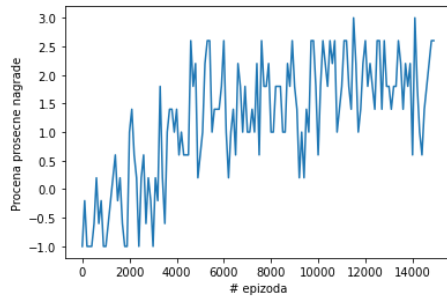
Konvergenciju je zgodno pratiti preko promene vrednosti parametra što je prikazano na sledećoj slici.



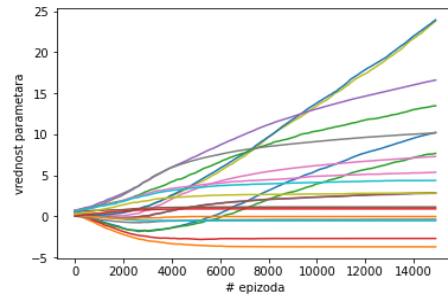
Slika 6: Parametri kroz obučavanje

Vidimo da pojedini parametri još nisu konvergirali iako smo iz prethodnog grafika zaključili da je agent naučio dobro da igra igru. Takođe, može se primetiti da je potrebno mnogo više iteracija nego za Q-learning. Ipak, REINFORCE je podoban za komplikovanije probleme te je očekivano da će se na jednostavnim problemima bolje pokazati jednostavniji algoritmi.

Na sledećim slikama se vidi isti ovi grafici za različite stope obučavanja.



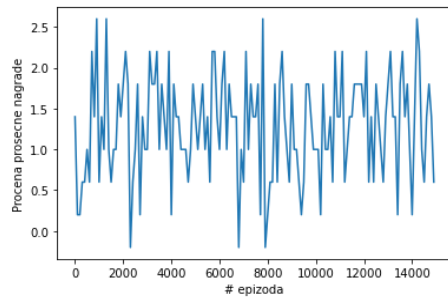
(a) Procena nagrade



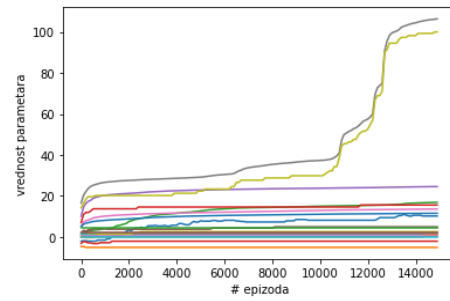
(b) Parametri

Slika 7: $\alpha = 0.004$

Možemo videti da je stopa obučavanja premala te algoritam ne konvergira ni nakon 10000 epizoda.



(a) Procena nagrade



(b) Parametri

Slika 8: $\alpha = 0.4$

Vidimo da je ova stopa prevelika jer parametri konvergiraju, ali agent ne uspeva da ostavri dobre rezultate u igri.