Blackjack by Monte Carlo

Blackjack by Monte Carlo

U ovom tekstu dat je opis funkcija i funkcionalnosti za domaći zadatak.

Funkcije koje se koriste:

- DrawCard() - vraća, na slučajan način, jednu kartu iz špila, poštujući vreovatnoće izbora svake vrednosti
- EpsGreedy() - funkcija koja implementira politiku odlučivanja, u ovom slučaju epsilon greedy
- PlayGame() - funkcija koja simulira jednu celu partiju igre
- UpdateQ() - funkcija koja dopisuje podatke u Q tabelu

Početak algoritma:

Na početku se inicijalizuje nekoliko promenljivih koje se koriste za procenu efikasnosti algoritma posle 10.000 partija. Q tabela se inicijalizuje sa [4, [1], [-1]], gde je:

- 4 – stanje,
- [1] - lista rezultata partija kada se pri stanju 4 igrao potez *HIT*,
- [-1] - lsita rezultata partija kada se pri stanju 4 igrao potez *HOLD*,

što je opšta forma jednog podatka u Q tabeli. Sam algoritam je predefinisan da se izvršava na 10.000 iteracija, odnosno 10.000 parija.

Jedna iteracija počinje tako što se pozove funkcija *playGame()*. Ulaskom u funkciju *playGame()* prvo se inicijalizuju potrebne promenljive. Nakon toga se poziva funkcija *drawCard()* koja vraća vrednost izvučene karte. Prva karta se daje igraču, druga dileru, treća igraču, četvrta dileru. Kada su podeljene karte proveravaju se stanja u kojima se nalaze igrač i diler tako što se saberu sve karte koje imaju u rukama. Poštoje je prvi potezu igrač prvo se proverava da li poseduje *˝iskoristivog keca˝*. Ukoliko ga poseduje te se pamti u *usableAce*. Nakon provere formira se tuple state koji nosi informacije o trenutnoj sumi karata, posedovanju keca i prvoj karti dilera, koja je otkrivena.

Posle toga se ulazi u petlju koja se izvršava sve dok se ne desi da algoritam kaže *HOLD* ili dok suma igračevih karata ne pređe 21. Prvo što se izvršava u petlji jeste pozivanje politike odlučivanja tako što se funkciji *epsGreedy()* proslede vrednosti *Q* i *state* a kao povratne vrednosti dobijamo akciju i trenutnu vrednosti koja je korišćena da se proračuna akcija, što je zgodno jer tako znamo da li je kec iskorišćen kao 1 ili 11. Ukoliko je politika rekla da se odigra akcija *HIT*, poziva se funkcija *drawCard()* koja daje novu kartu igraču. Posle dodele nove karte proverava se vrednso karata koje poseduje igrač i,

ukoiko je suma peko 21, petlja se prekida i vraćaju se vrednosti. Ukoliko smo i dalje u igri ponovo se poziva politika odlučivana koja daje novu akciju.

Funkcija *epsGreedy()*, koja predstavlja politiku odlučivanja, se izvršava tako što se na početku inicijalizuju promenljive potrebne za rad i razdvoji se infrmacija koja dolazi iz state. Ukoliko imamo iskoristivog keca a suma nam je ispod 11 onda računamo njega kao 11, obratno računamo ga kao 1. Sa takvim promenama prvo se pitamo da li smo pobedili, odnosno da li sada imamo 21 ako keca računamo kao 11. Ukoliko je to tačno politika će se odlučiti za potez *HOLD*.

Ukoliko se ništa nije izvršilo od prethodno navedenih linija, prvo što se pitamo je da li radimo *ekspoataciju* ili *eksploraciju* tako što proverimo da li je radnom generisani broj veći ili manj od epsilon. Ukoliko je manji radi se *eksploracija* tako što se na slobodan način bira potez *HIT* ili *HOLD*. Ukoliko radimo *eksploataciju* prvo gledamo šta se nalazi kod dilera a šta kod nas. Ako smo sigurni da ne možemo da izgubimo ako odigramo *HIT* akciju onda biramo nju, u suprotnom traži se trenutno stanje u kom se igrač trenutno nalazi u *Q* tabeli. Kada ga pronađemo potreben su nam infomacije o rezultatima prethodnih partija kada se u datom stanju igrao HIT ili HOLD potez. Pošto smo označili pobedu kao ˝1˝a gubitak sa ˝-1˝, prostim sabiranjem svih vrednosti možemo da vidimo da li je bolje igrati *HIT* ili *HOLD* potez. Ukoliko je stanje u kome se igrač nalazi novo, odnosno da ga nema u *Q* ili nije moguće pronaći liste za *HIT* ili *HOLD* poteze (što se dešava na pocetku jer nemamo informacije za sva stanja), politika će da radi *ekslporaciju* tako što će da kaže da se odigra *HIT* ili *HOLD* potez, u zavisnosti od informacije koja nam fali.

Kada se završi igranje igrača, pod uslovom da je i dalje u igri (<=21) red je na dilera da odigra svoje poteze. Politika odlučivanja dilera je veoma jednostavna, ukoliko u zbiru ima manje od 17 od će izvlačiti novu kartu dokle god ne bude imao 17 ili više. Takođe, činilo mi se da je logično da diler nastavi da igra ako pređe 17 a igrač ima preko 17 a manje od 21 kako bi pokušao da ga uhvati i pobedi jer je, ako ostane na 17 a igrač ima više od 17 svakako izgubio pa nema šta dodatno da izgubi ako pokuša da pobedi igrača. Najgore što može da se desi jeste da prebaci 21, što opet nije promena u odnosu na početno stanje.

Posle igranja dilera vrši se provera pobednika jednostavni poređenjem vednosti.

Rezultat parije i stanja kroz koja je igrač prošao se vraćaju i koriste za osvežavanje vrednosti *Q* tabele.

Posle 10.000 partija se ispisuje odnos dobijenih i izgubljenih partija.

U konačnom odnosnu pobeda i poraza izbačene su partije koje su završene nerešenim rezultatom a kojih, u proseku, ima oko 10% od upunih partija.