Primjena tehnike mašinskog učenja na igricu Triba

Belmin Ruhotina
Prirodno – matematički fakultet
Odsjek za matematiku / Teorijska kompjuterska nauka
Sarajevo, BiH
belmin.ruhotina@gmail.com

***Abstract*—U ovom radu opisana je Q-learning tehnika koja pripada oblasti mašinskog učenja i njena primjena na igricu Triba. Cilj je bio da naučimo AI agenta da igra ovu igru i bude što uspješniji protiv čovjeka. Na kraju rada su priloženi i rezultati testiranja AI agenta protiv igrača koji igra nasumične poteze.**

Keywords — Q-learning, reinforcement learning

I. UVOD

Igrica Triba sastoji se od mreže m x n tačaka,gdje su m i n parni prirodni brojevi. Dva igrača igraju naizmjenično. Svaki igrač u svom potezu bira trougao tako što izabere tri tačke iz mreže koje ne leže na istoj pravoj. Trouglovi se ne smiju sjeći, te ne smiju dva trougla imati zajednički vrh. Trouglovi se mogu nalaziti jedan unutar drugog. Igra je završena kada jedan igrač ostane bez poteza. Igrač koji je ostao bez poteza gubi igru. Da bi naučili AI agenta kako da igra ovu igru, koristili smo Q-learning tehniku koja pripada podoblasti mašinskog učenja koja se naziva reinforcement learning.

II. KORIŠTENI ALGORITAM

Prije opisa korištenog algoritma, objasnimo nekoliko pojmova koji su potrebni za razumjevanje. Prvo, potrebno je da razumijemo šta je Q-funkcija. Q je skraćenica od Quality što u prevodu znači kvalitet odnosno vrijednost nečega. Q-funkcija će svakom paru (stanje_ploče, potez) dodijeliti realan broj iz intervala [0,1) koji predstavlja koliko je kvalitetan potez koji smo odigrali. Uvedimo oznake S za stanje_ploče i P za potez koje ćemo koristit u nastavku. Optimalno bi bilo da je:

$$Q(S,P) = \max_{p} Q(S', p) \quad (1)$$

Drugim rječima, vrijednost odigravanja poteza P u stanju S treba da bude skoro kao vrijednost odigravanja najboljeg poteza p u stanju S' u koje smo došli odigravanjem poteza P u stanju S. Zbog toga, uvodimo faktor $\gamma \in (0,1)$ i (1) postaje:

$$Q(S,P) = \gamma * \max_{p} Q(S', p) \quad (2)$$

Dalje, ne želimo svaki put pri ažuriranju Q(S,P) zamjeniti kompletnu staru vrijednost sa novom, već želimo novu vrijednost za Q(S,P) da bude između stare i nove. Zbog toga, uvodimo još jedan parametar koji se zove learning rate α. Sada (2) konačno postaje:

$$Q(S,P) = (1 - \alpha)*Q(S,P) + \alpha * \gamma * \max_{p} Q(S', p) \quad (3)$$

Iz formule zaključujemo da α utvrđuje koliko ćemo promijeniti Q(S,P) naspram umanjene maksimalne vrijednosti Q(S',p). Za α = 0, neće se ništa promjeniti. Za α = 1, kompletnu staru vrijednost ćemo zamijeniti sa novom. Za α = 0.5, dobit ćemo srednju vrijednost stare i nove. Vrijednosti Q funkcije ćemo čuvati u hash tabeli, pri čemu će ključ biti (S,P), a vrijednost Q(S,P). Data hash tabela se naziva Q-tabela. Za više informacija o Q-learning tehnici se može pronaći u [1]. Algoritam se zasniva na igranju određenog broja igara AI agenta protiv igrača koji igra nasumične poteze. Nakon svake odigrane igre, AI agent posjeduje listu parova (S_i,P_i) i = $\overline{1,n}$, pri čemu je n ukupan broj poteza koje je AI agent odigrao u toj igri. Za svaki par (S_j,P_j) j = $\overline{n,1}$ izračunat ćemo $Q(S_j,P_j)$ prema (3), pri čemu prvo gledamo da li se dati ključ nalazi u Q-tabeli. Ukoliko se nalazi, uzet ćemo njegovu vrijednost i uvrstiti je za Q(S,P) u (3). U suprotnom, Q(S,P) = 0. Također, za j = n, Q(S',P) = 1, ako je AI agent pobijedio igru, u suprotno je 0. Parametri α i γ imaju vrijednosti 0.2 i 0.9 respektivno.

Algoritam 1 Algoritam za treniranje AI agenta

Ulaz: num_of_rounds, lap

Izlaz: hash table

```
for i = 1 to num_of_rounds do
  if(i % lap == 0) then
    ai_agent.decrease_exp_rate(0.05)
  end if
  while(true) do
    positions = board.get_available_positions()
    move = ai_agent.choose_position(positions)
    board.update(move)
    board_hash = board.get_hash()
    ai_agent.add_state(board_hash)
    if(board.game_over()) do
      ai_agent.feed_reward(1)
      ai_agent.reset()
      random_player.reset()
      board.reset()
      break
    else
      positions = board.get_available_positions()
      move = random_player.choose_position(positions)
      board.update(move)
      board_hash = board.get_hash()
      if(board.game_over()) do
        ai_agent.feed_reward(0)
        ai_agent.reset()
        random_player.reset()
        board.reset()
        break
      end if
    endi if
  end while
end for
return ai_agent.get_states_value()
```

Nakon što smo završili treniranje AI agenta, imamo spremnu Q-tabelu koju koristimo u igranju AI agenta protiv čovjeka. Bitno je još naglasiti kako AI agent bira koji će potez odigrati.

Tu koristimo ε-greedy strategiju. Postavimo ε na vrijednost između 0 i 1. Prije odabira poteza, generišemo slučajan broj između 0 i 1 koristeći uniformu distribuciju. Ukoliko je slučajni broj manji od ε, AI agent će odigrati nasumičan potez i to se zove exploration. U suprotnom, AI agent koristi Q-tabelu i to se zove exploitation. Prije početka algoritma, postavimo ε na 0.3. To znači da će AI agent 30% vremena treniranja igrati nasumične poteze. Međutim, dobra je praksa da tokom vremena treniranja smanjujemo ε, što je i urađeno u našem algoritmu. Razlog zato jeste da AI agent na početku treniranja probava razne opcije, dok se kasnije vremenom usredotočava na korištenje Q-tabele.

III. SIMULACIJSKI REZULTATI

Nakon završenog treninga, imamo spremnu Q-tabelu koju ćemo iskoristiti tako što će AI agent igrati protiv igrača koji igra nasumične poteze određen broj igara da bi vidjeli koliko je AI agent uspješan. Prilikom igranja, AI agent će isključivo koristiti Q-tabelu pri odabiru svojih poteza.

Tabela 1 Parametri treninga AI agenta

Ploča	Broj igara	Vrijeme treninga (u minutama)
4x4	2000	30
4x6	3000	90
4x8	3200	300
6x6	5000	600

Tabela 2 Parametri testiranja AI agenta

Ploča	Broj igara	Uspješnost
4x4	100	100%
4x6	100	100%
4x8	100	65%
6x6	100	77%

Uzimajući u obzir da na ploči 4x8, AI agent prvi potez može odigrati na 4628 različitih načina, dok na ploči 6x6 taj broj iznosi 6768, zaključujemo da je uspješnost zadovoljavajuća s obzirom na broj odigranih igara tokom treninga.

IV. ZAKLJUČAK

Ovim radom se htjelo pokazati kolika je uspješnost primjene Q-learning tehnike na igricu Triba koja je zahtjevna u smislu velikog broja mogućih poteza koje igrač može odigrati u svom potezu. Na osnovu tabele 2, zaključujemo da je tehnika veoma uspješna u slučaju malih dimenzija ploče jer one omogučavaju igranje velikog broja igara u znatno manjem vremenu u odnosu na veće dimenzije ploče. Također, zaključujemo da Q-learning tehnika ima potencijala da bude uspješna i na većim dimenzijama ploče, ali zahtjeva igranje većeg broja igara i samim tim zahtjeva puno vremena za treniranje AI agenta.

LITERATURA

[1] Watkins, C.J.C.H., Dayan, P. Q-learning. Mach Learn 8, 279-292 (1992)