Hibrid neuronske mreže, genetskog algoritma i reinforcement learninga za igranje top-down shooter igre

Seminarski rad u okviru kursa
Računarska inteligencija
Matematički fakultet

Đaković Branko, Filip Kristić, Krčmarević Mladen
brankodjakovic08@gmail.com, filip.kristic96@gmail.com
mladenk@twodesperados.com

3. septembar 2019.

Sažetak

U radu će biti prikazana upotreba "reinforcement learing"-a i genetskog algoritma za generisanje težina neuronske mreže koja igra top-down shooter igru "Shrodinger's shooter". Ovaj pristup spada u Neuro-evoluciju[3], s tim što se menjaju samo težine neruonske mreže dok je struktura fiksna. Za pisanje kodova je korišćen programski jezik C++ i biblioteka FANN sa omotačem za C++. Daćemo kratak opis igre kao i dva rešenja koja se razlikuju po načinu reprezentacije ulaza za neuronsku mrežu.

Sadržaj

1 Uvod **2**

2 Opis rešenja **2**
2.1 Selekcija, ukrštanje i mutacija 4

3 Upoređivanje rešenja **5**
3.1 Prvo rešenje 5
3.2 Drugo rešenje 6

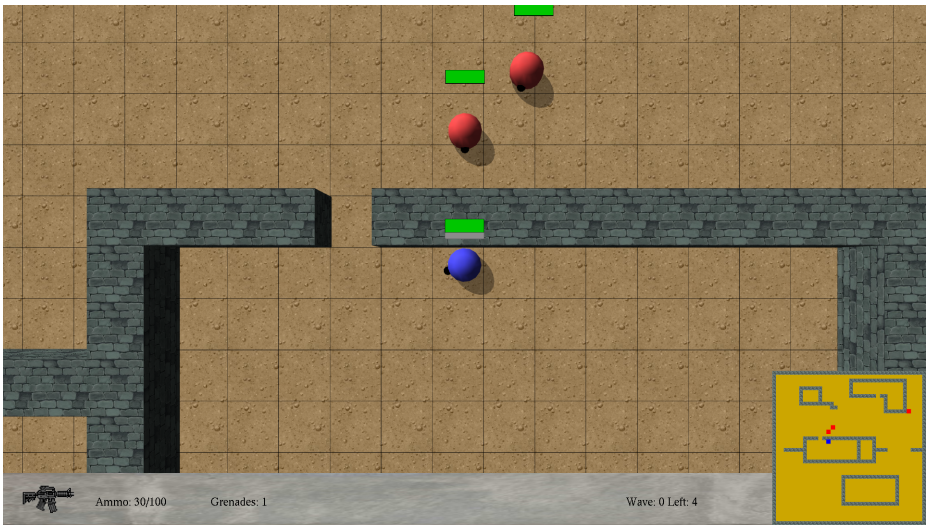
4 Zaključak **7**

Literatura **7**

A Uputstvo **7**

1 Uvod

"Shrodinger's shooter" je shooter igra koju smo razvili za predmet Razvoj softvera u kojoj je cilj igrača da što duže preživi nalete protivnika. Igrica koristi 2D fiziku uz 3D grafiku, a mapa je kvadratnog oblika i sadrži zidove. Moguće akcije igrača su: "idi gore", "dole", "levo", "desno" kao i dijagonalno kretanje njihovom kombinacijom, podešavanje pozicije nišana, repetiranje oružja i pucanje. Zbog kompleksnosti su izbačeni prva pomoć, pancir, granate i različito oružje. Cilj neuronske mreže[2] je da na osnovu trenutne situacije igrača da njegovu sledeću akciju.



Slika 1: Schrodinger's shooter

2 Opis rešenja

Program ima tri načina izvršavanja:

```
./Schshooter.out
```

koji obuhvata učitavanje rešenja u vidu neuronske mreže i igranje igrice uz grafički prikaz,

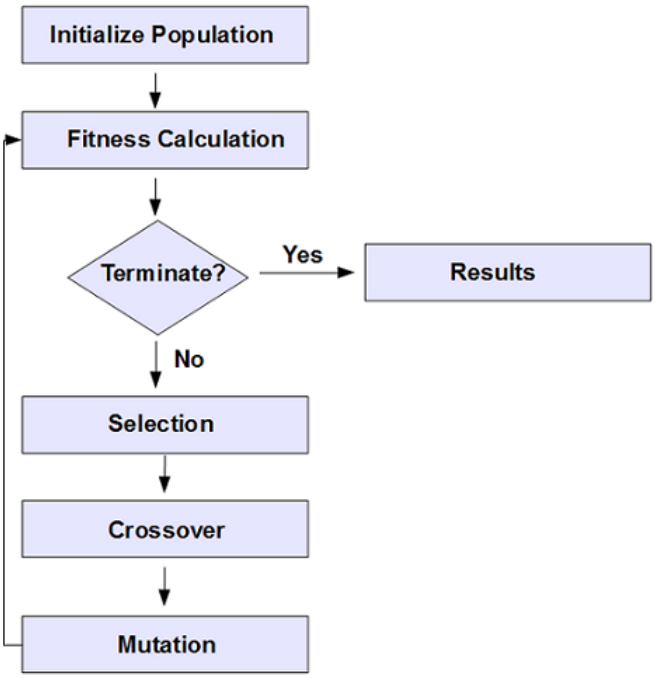
```
./Schshooter.out -t brojIteracija
```

koji vrši treniranje sa brojem iteracija "brojIteracija" bez grafičkog prikaza i

```
./Schshooter.out -tv brojIteracija
```

koji vrši treniranje sa brojem iteracija "brojIteracija" sa grafičkim prikazom.

Trening započinje kreiranjem početne generacije genetskog algoritma[1] (slika 2.[1]) kod koje svaki hromozom ima prilagođenost (fitness) jednak nuli. Hromozomi su predstavljeni sadržajem tj. nizom brojeva u pokretnom zarezu koji predstavlja težine svih veza mreže i jednim brojem u pokretnom zarezu koji predstavlja prilagođenost tog hromozoma. Težine se uzimaju nasumično iz intervala [-10, 10] koji je takođe nasumično izabran. Broj hromozoma u generaciji kao i broj generacija u oba rešenja iznose po nekoliko stotina. O ulazu će više biti rečeno u narednom poglavlju.



Slika 2: Genetski algoritam

Nakon kreiranja generacije redom se uzimaju hromozomi i težine veza neuronske mreže se postavljaju na sadržaj hromozoma. Veličina ulaznog sloja u prvom rešenju je 191, a 10 u drugom. oba rešenja imaju samo jedan skriven sloj koji je veličine 10 u prvom zbog ogromnog broja veza usled veličine ulaza i 8 u drugom. Za aktivacionu funkciju je korišćena linearna aproksimacija sigmoidne funkcije. Sigmoidna funkcija je oblika: [2]

[1] Slika preuzeta sa: https://apacheignite.readme.io/docs/genetic-algorithms

[2] Preuzeto iz dokumentacije fann biblioteke. http://leenissen.dk/fann/html/files/fann_cpp-h.html

x - ulaz
y - izlaz
s - nagib i
d - derivacija
raspon: $0 < y < 1$

$$y = \frac{1}{1 + e^{-2 \times s \times x}} \quad (1)$$

$$d = 2 \times s \times y \times (1 - y) \quad (2)$$

Stopa učenja nije podešena pošto nema nikakvog uticaja na dati problem. Izlazni sloj sadrži pet neurona čiji izlazi imaju vrednosti u intervalu [0,1] a koji regulišu kretanje gore-dole, levo-desno, treći ugao pod kojim igrač nišani,da li da puca i da li da dopuni municiju. Prva dva izlaza su podeljena u tri intervala [0,0.33], (0.33-0.66] i (0.66-1] koji redom odgovaraju kretanju gore (levo), bez kretanja i dole (desno). Treći izlaz se skalira na [-π, π] i ugao igrača se postavlja na datu vrednost a četvrti i peti daju potvrdan odgovor za vrednosti manje od 0.5, a negativan u suprotnom. U svakoj iteraciji programa se ažuriraju pozicija i akcije igrača u zavisnosti od izlaza mreže. "Reinforcement learing" se koristi prilikom izračunavanja prilagođenosti svakog hromozoma tako što za svaku dobru akciju u vidu eliminacija ili nanošenja štete hromozom dobija nagradu u vidu poena. Prilagođenost se računa na osnovu naredne formule, a najbolje ocenjen hromozom se čuva. U drugom rešenju je dodatno implementirano višestruko izvršavanje za svaki hromozom, a zatim uzimanje srednje vrednosti dobijenih prilagođenosti.

fintess = eliminacije * 50 + šteta * 0.5

2.1 Selekcija, ukrštanje i mutacija

Po obradi svih hromozoma generacije, ukoliko ima još iteracija, vrši se selekcija hromozoma koji će učestvovati u izradi nove generacije. Selekcija se vrši ruletskim pristupom, tako što se računa zbir prilagođenosti svih hromzoma i svaki ima šansu da bude izabran srazmerno odnosu njegove i celokupne prilagođenosti. Nakon izbora hromozoma koji učestvuju u reprodukciji nasumično se biraju dva roditelja i vrše se ukrštanje i mutacija koji su implementirani na različite načine u prvom i drugom rešenju. Svaki par roditelja kreira dva deteta i to se vrši sve dok ne bude ispunjena nova populacija.

Ukrštanje u prvom rešenju:

i - nasumičan broj od 0 do veličine sadržaja hromozoma;
dete1 = sadržaj roditelja1 do i + sadržaj roditelja2 od i do kraja;
dete2 = sadržaj roditelja2 do i + sadržaj roditelja1 od i do kraja;

Mutacija u prvom rešenju:

t - nasumičan broj u pokretnom zarezu iz intervala [0,1];
Ukoliko je t manje od stope mutacije:
Promeniti nasumičan element sadržaja hromozoma;

Ukrštanje u drugom rešenju:

Za svaki element i sadržaja hromozoma uraditi:
t - nasumičan broj u pokretnom zarezu iz intervala [0,1];
Ako je t $<$ 0.5:
dete1[i] = roditelj1[i];
dete2[i] = roditelj2[i];
Inače:
dete1[i] = roditelj2[i];
dete2[i] = roditelj1[i];

Mutacija u drugom rešenju:

Za svaki element i sadržaja hromozoma uradi:
t - nasumičan broj u pokretnom zarezu iz intervala [0,1];
Ukoliko je t manje od stope mutacije:
Promeniti i-ti element sadržaja hromozoma;

Celokupan navedeni postupak se zatim ponavlja dok nije zadovoljen kriterijum zaustavljanja, tj. dok se ne premaši zadati broj iteracija. Najbolja jednika kao i svi hromozomi poslednje generacije se čuvaju u vidu neuronskih mreža u tekstualnim datotekama radi čuvanja progresa i nastavka treniranja.

3 Upoređivanje rešenja

U ovom poglavlju biće opisana dva eksperimentalna rešenja problema. Njihova suštinska razlika je način na koji je predstavljen ulaz neuralne mreze igrača, tj. način na koji se opisuje trenutno stanje okoline igrača.

3.1 Prvo rešenje

U prvom rešenju pokušali smo sa pristupom predstavljanja celokupne okoline igrača, naime svako polje na ekranu predstavlja jedan ulazni čvor koji može imati vrednosti: 0 ako je polje prazno, -1 ako se na njemu nalazi protivnički igrač i 1 ako je u pitanju zid. Ovaj unos je poprilično velik jer na ekranu, u datom trenutku, igrač može da vidi 190 polja, te se to preslikava u 190 ulaznih čvorova i još jedan za količinu trenutne municije. Igrač se uvek nalazi na polju (5, 9), tj. u sredini mape koja je na primeru obeleženma sa "P" radi preglednosti, a prilikom izvršavanja je 0.

1000000000000000000
1000000000000000000
1000000000000000000
1000000011110111111
1000000010000000-100
100000001P000000000
10000000000-10000000
1000000010000000000
1000000010000000000
1000000011111111111

Trening je vršen na populaciji od 300 jedinki po generaciji u 300 iteracija.

Karakteristike hardvera na kome je vršen trening:

CPU: intel-i5 2500k
GPU: AMD Radeon 6850
RAM: 4GB DDR3
OS: Ubuntu 16.04

Vreme trajanja je oko 10h posle čega je dobijena konfiguracija koja nije davala dobre rezultate. Zaključeno je da ovaj pristup suštinski ne konvergira ka dobrom rešenju te je primenjen drugi pristup sa drastično smanjenjim ulazom neuralne mreze.

3.2 Drugo rešenje

U drugom rešenju znatno je smanjen broj ulaznih čvorova. Prvi ulaz je ugao prema najblizem vidljivom protivniku, a drugi trenutna kolicina municije. Okolina je predstavljena preko senzora. Postoji 8 senzora, po jedan u svakom pravcu od igrača (gore, dole, levo, desno i dijagonale). Senzori proveravaju broj praznih mesta u svojim pravcima, postavljaju se na određen broj i skaliraju na interval [0, 1] u odnosu na ukupan broj polja do kraja ekrana.

Ocekivano je da ce ovaj pristup dati smislenije rezultate zbog drastično manje kolicine ulaznih podataka. Uprkos smanjenom ulazu su dobijeni slični rezultati kao kod prvog rešenja, tj. igrači koji se nasumično kreću i pucaju.

Trening za ovaj pristup vršen je na populaciji od 100 jedinki po generaciji u 300 iteracija, s tim što se za svaku jedinku fitnes računao po pet puta pa je uzeta srednja vrednost.

Karakteristike hardvera na kome je vršen trening:

CPU: intel-i5 7300hq
GPU: NVIDIA GeForce 1060
RAM: 16GB DDR4
OS: Windows Subsystem for Linux - Ubuntu 18.04

4 Zaključak

Tema ovog rada bila je da se prikaže upotreba tehnika "reinforcement learing"-a, neuronskih mreza i genetskog algoritma u svrhu igranja kompjuterskih igrica tj konkretno igrice "Shrodinger's shooter". Postoji više načina na koji moze da se realizuje "računarski igrač", predstavljeni pristup nije dao dobre rezultate. Podešavanjem parametara i načina predstavljanja unosnih podataka, kao i različitim dužinama treninga pokusano je da se sistem navede na bolja rešenja ali idalje nije dobijeno zadovoljavajuće ponašanje. Analizom dobijenih rezultata zaključujemo da za dati problem nije dovoljno samo menjanje težine i da bi se, za ovaj konkretan problem, verovatnije dobili bolji rezultati primenom neke druge metode poput NEAT[4] Algoritma gde se menja cela struktura neuronske mreze u procesu učenja.

Literatura

[1] Janičić dr Predrag and Nikolić dr Mladen. *Veštačka Inteligencija*. Beograd, 2019.

[2] Kantardzic Mehmed. *Data Mining: Concepts, Models, Methods, and Algorithms, Second Edition*. 2011.

[3] Kenneth Stanley O., Clune Jeff, Lehman Joel, and Risto Miikkulainen. Designing neural networks through neuroevolution. *Nature Machine Intelligence*, 2019. on-line at: http://www.evolvingai.org/files/s42256-018-0006-z.pdf.

[4] Kenneth Stanley O. and Risto Miikkulainen. Evolving Neural Networks through Augmenting Topologies. *The MIT Press Journals*, 2002. on-line at: http://nn.cs.utexas.edu/downloads/papers/stanley.ec02.pdf.

A Uputstvo

Pre pokretanja bilo kog programa potrebno je instalirati neophodne pakete tako što se izvrše sledeće naredbe u terminalu:

```
sudo apt-get install freeglut3-dev
sudo apt-get install libbox2d-dev
sudo apt-get install libalut-dev
sudo apt-get install libfann2
sudo apt-get install libfann-dev
```