Korišćenje neuronskih mreža u razvoju video igrica

Seminarski rad u okviru kursa
Računarska inteligencija
Matematički faklutet

Vujičić Zoran, Stefanović Stefan
vzoran96@gmail.com,
stefaniussuperbus@gmail.com

Sažetak

U ovom radu je prikazana prednost korišćenja neuronskih mreza u pravljenju video igrica, konkretno je dizajnirana i programirana igrica SmartHeli koja omogućava primenu metoda veštačke inteligencije i neuronskih mreža i pokazali smo da je AI igrac pametniji od obicnog prosecnog igraca, ma koliko on bio vešt i iskusan u igranuju igrica.

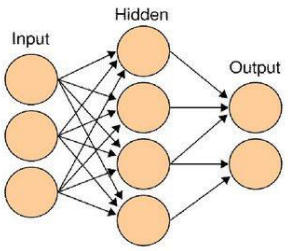
Sadržaj

1 Uvod

Neuronska mreža je jedan oblik implementacije sistema veštačke inteligencije, koji predstavlja sistem koji se sastoji od određenog broja međusobno povezanih procesora ili čvorova, ili procesnih elemenata koje nazivamo veštačkim neuronima.

Telo neurona naziva se čvor ili jedinica. Svaki od neurona ima lokalnu memoriju u kojoj pamti podatke koje obrađuje. Podaci koji se obrađuju su lokalni podaci kao i oni koji se primaju preko veze. Podaci koji se ovim kanalima razmenjuju su obično numerički.

Arhitektura neuronske mreže predstavlja specifično povezivanje neurona u jednu celinu. Struktura neuronske mreže se razlikuje po broju slojeva. Prvi sloj se naziva ulazni, a poslednji izlazni, dok se slojevi između nazivaju skriveni slojevi. Najčešće ih ima tri, ali to se uglavnom odnosi na manje projekte jer sto je veći broj neurona u srkivenom sloju to je više informacija potrebno za učenje, i to je više vremena potrebno, ali se zato kompleksnije stvari mogu naučiti. Prvi sloj, tj. ulazni je jedini sloj koji prima podatke iz spoljašnje sredine, sledeći (skriveni) prosleđuje relevantne podatke do trećeg (izlaznog) sloja. Na izlazu trećeg sloja dobijamo konačan rezultat. Složenije neuronske mreže imaju više skrivenih slojeva. Slojevi su međusobno potpuno povezani.



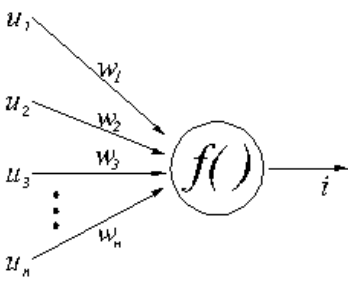
Slojevi komuniciraju tako što se izlaz svakog neurona iz prethodnog sloja povezuje sa ulazima svih neurona narednog sloja. Znači, svaki čvor ima nekoliko ulaza i jedan izlaz. Jačina veza kojom su neuroni povezani naziva se težinski faktor (weight).Slojevi komuniciraju tako što se izlaz svakog neurona iz prethodnog sloja povezuje sa ulazima svih neurona narednog sloja. Znači, svaki čvor ima nekoliko ulaza i jedan izlaz. Jačina veza kojom su neuroni povezani naziva se težinski faktor (weight).

U ovom radu, kada se govori o neuronskim mrežama, misli se prvenstveno na “veštačke neuronske mreže” (engleski termin Artificial Neural Networks skraćeno ANN), zbog toga što se uglavnom govori o modelima neuronskih mreža (skraćeno NM), realizovanim na računarima.

Veštačke neuronske mreže se poslednjih godina uspešno primenjuju u mnogim oblastima. Prednosti koje nude veštačke neuronske mreže učinile su da one postanu nezaobilazne u rešavanju sve složenijih problema koji se javljaju u savremenom svetu. Veštačke neuronske mreže predstavljaju jednu od najpopularnijuh tehnika veštačke inteligencije. Za uspešnu primenu veštačkih neuronskih mreža potrebno je prikupiti što više podataka. Veštačke neuronske mreže su mreže vođene podacima tako da kvalitet modela zavisi i od količine podataka tj. Pogodne su za fuziju podataka. Kako, neurosnke veštačke mreže imaju primenu u savremenoj nauci, njihova primena je širokog spektra.

Podaci iz trening skupa se periodično propuštaju kroz NM. Dobijene vrednosti na izlazu mreže se upoređuju sa očekivanim. Ukoliko postoji razlika između dobijenih i očekivanih podataka, prave se modifikacije na vezama između neurona u cilju smanjivanja razlike trenutnog i željenog izlaza. Ulazno-izlazni skup se ponovo predstavlja mreži zbog daljih podešavanja težina, pošto u prvih nekoliko koraka mreža obično daje pogrešan rezultat. Posle podešavanja težina puta za sve ulazno izlazne šeme u trening skupu, mreža nauči da reaguje na željeni način. NM je obučena ako može tačno da rešava zadatke za koje je obučavana. NM je sposobna da izdvoji važne osobine i šeme u klasi trening primera. Nakon obučavanja sa određenom verovatnoćom, NM može da generalizuje nove ulazne podatke za koje nije obučavana.

Model veštačkog neurona:

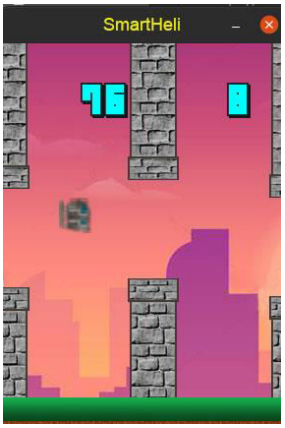


Veštački neuroni, kao i biološki, imaju jednostavnu strukturu i imaju slične funkcije kao biološki neuroni. Telo neurona se naziva čvor ili jedinica.Veštački neuroni mogu da imaju prag jačine tako da samo ako agregatni signal premaši taj prag signal biva poslat. Tipično, veštački neuroni su organizovani u slojeve. Različiti slojevi mogu da izvode različite vrste transformacija na svojim ulazima. Signali putuju od prvog (ulaznog), do zadnjeg (izlaznog) sloja, u nekim slučajevim prolazeći kroz slojeve više puta.

2 Opis rešenja

Programski jezik u kom je rađen projekat je Python 3.6 uz python-pygame pakete za određene funkcije a okruženje za kucanje koda je PyCharm Professional.

U glavnom fajlu **heli.py** se nalazi opšti algoritam po kom funkcioniše igrica. Napomenućemo neke delove glavnog koda i njihovo funkcionisanje.



Glavni cilj ove igrice je da helikopter ostane što duže u životu, tako što će da izbegava prepreke (kamene stubove) uz pomoć ANN (Artificial Neural Network). Bez neuronskih mreža prosečan igrač može ostvariti određen broj poena u ovoj igri, međutim uz pomoć neuronskih mreža helikopter će po generacijama naučiti da ostane veoma dugo u životu, što je za mnoge igrače nedostižno.

Tokom prvih generacija heli će biti "loš" igrač, udaraće u kamene stubove dok ne počne da uči. Dakle, iz generacije u generaciju helikopter "stvara svest" (naravno preko neuronskih mreža) kada treba da poleti, koliko i na kojoj distanci od kamenih stubova.

Svaki od modema je praćen neuronskom mrežom sa 3 ulazna čvora, 3 skrivena i jednim izlaznim čvorom. Svi čvorovi koriste sigmoidnu funkciju aktivacije i potpomažemo im u evoluciji koristeći opadajući gradijent.

```
for i in range(amount_of_models):
    model = Sequential()
    model.add(Dense(output_dim=3, input_dim=3))
    model.add(Activation("sigmoid"))
    model.add(Dense(output_dim=1))
    model.add(Activation("sigmoid"))
    sgd = SGD(lr=0.01, decay=1e-6, momentum=0.3, nesterov=True)
    model.compile(loss="mse", optimizer=sgd, metrics=["accuracy"])
    current_pool.append(model)
    fitness.append(-100)
# ako smo izabrali da nastavimo od vec sacuvanog modela
if load_saved_pool:
    for i in range(amount_of_models):
        current_pool[i].load_weights("Modeli/model_new"+str(i)+".keras")
for i in range(amount_of_models):
    print(current_pool[i].get_weights())
```

Segment koda u kom se generišu/ućitavaju neuronske mreže

Pri pokretanju programa generišemo početnu generaciju čvorova tako što slučajnim izborom izaberemo parametre težina za svaki čvor. Alternativno, možemo nastaviti evoluciju od postojećih mreža.

Sam program radi tako što se dolenavedeni ciklus ponavlja dok ne izađemo iz samog programa, putem ESC dugmeta ili zatvaranjem prozora.

1. Generiši početak igre, sa svim helikopterima u istoj poziciji i par početnih stubova.
2. Počni igru.
3. Pri svakom frejmu, svaki helikopter procenjuje da li da se podigne, unoseći u svoju neuronsku mrežu sopstvenu visinu, udaljenost od najbližih stubova i visinu na kojoj se nalazi prolaz. Ako je izlaz iz mreže veći od 0.5, diži se.
4. Nastavljaj 3. dok su svi helikopteri živi. Generiši nove prepreke periodično. Ako su helikopteri prošli kroz par stubova, dodaj im na rezultat.
5. Skaliraj rezultate helikoptera tako da im je suma 1.
6. Ruletskom selekcijom izaberi 2 helikoptera za ukrštanje.
7. Zameni njihovim mrežama ulazne čvorove i tako generiši 2 primerka nove generacije. Mutiraj ih.
8. Nastavi 6. i 7. dok ne generišemo isti broj mreža nove generacije kao i stare.
9. Ako smo to naznačili, sačuvaj trenutne modele za buduće korišćenje u fajlove, kao i prosečan i maksimalan skor za tu generaciju.
10. Zameni postojeće mreže novom generacijom i vrati se na 1.

```
def predict_action(height, dist, obstacle_height, model_num):
    global current_pool
    height = min(HEIGHT_OF_SCREEN, height) / HEIGHT_OF_SCREEN - 0.5
    dist = dist / 400 - 0.5
    obstacle_height = min(HEIGHT_OF_SCREEN, obstacle_height) / HEIGHT_OF_SCREEN - 0.5
    neural_input = np.asarray([height, dist, obstacle_height])
    neural_input = np.atleast_2d(neural_input)
    output_prob = current_pool[model_num].predict(neural_input, 1)[0]
    if output_prob[0] <= 0.5:
        return 1 # skaci
    return 2  # nemoj
```

Funkcija predviđanja poteza

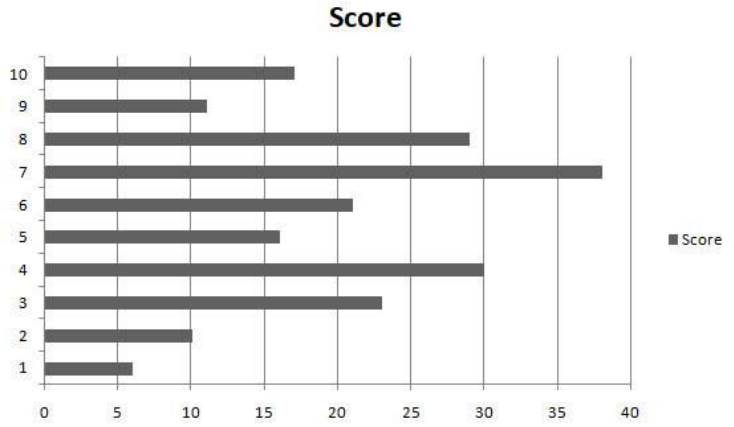
```
def crossover(model_i1, model_i2):
    global current_pool
    weights1 = current_pool[model_i1].get_weights()
    weights2 = current_pool[model_i2].get_weights()
    weights_new1 = weights1
    weights_new2 = weights2
    weights_new1[0] = weights2[0]
    weights_new2[0] = weights1[0]
    return np.asarray([weights_new1, weights_new2])
#mutacija, svaki neuron ima 20% sanse da se promeni u krugu -0.5,0.5
def mutate(weights):
    for xi in range(len(weights)):
        for yi in range(len(weights[xi])):
            if random.uniform(0, 1) > 0.8:
                change = random.uniform(-0.5,0.5)
                weights[xi][yi] += change
    return weights
```

Ukrštanje i mutacija neuronskih mreža

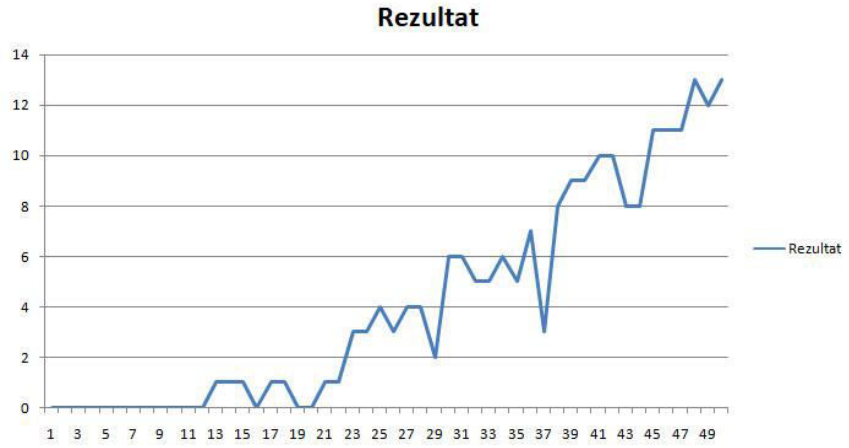
3 Rezultati testiranja

Najbolji način da proverimo kako funkcioniše naš projekat je eksperimentalno testiranje. Dakle, testirano je u više generacija i u određenim vremenskim periodima da bi videli kako helicopter napreduje u učenju i došli smo do određenih zaključaka.

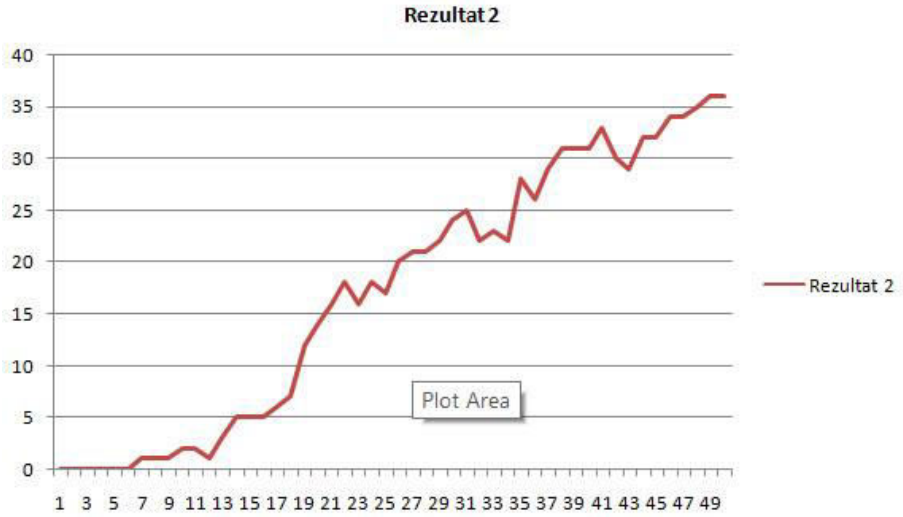
Prvo smo testirali igricu uz pomoć više igrača bez neuronskih mreža, i došli do sledećih rezultata.



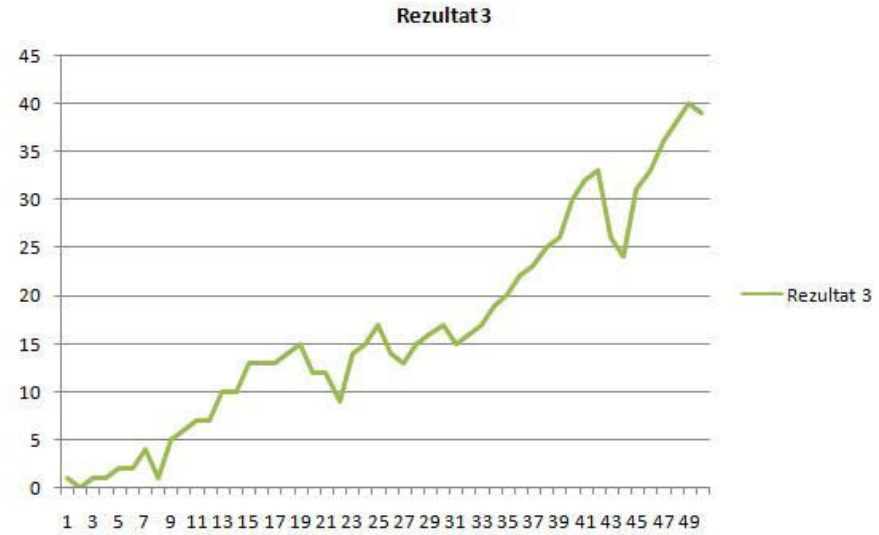
Na ovom grafiku su prikazani ostvareni poeni na 10 igrača. Postignut je prosečan broj od 20.1 poena. Najbolji rezulat je bio 38 poena a najgori 6 poena. (Igrica nije baš toliko jednostavna kako se čini).



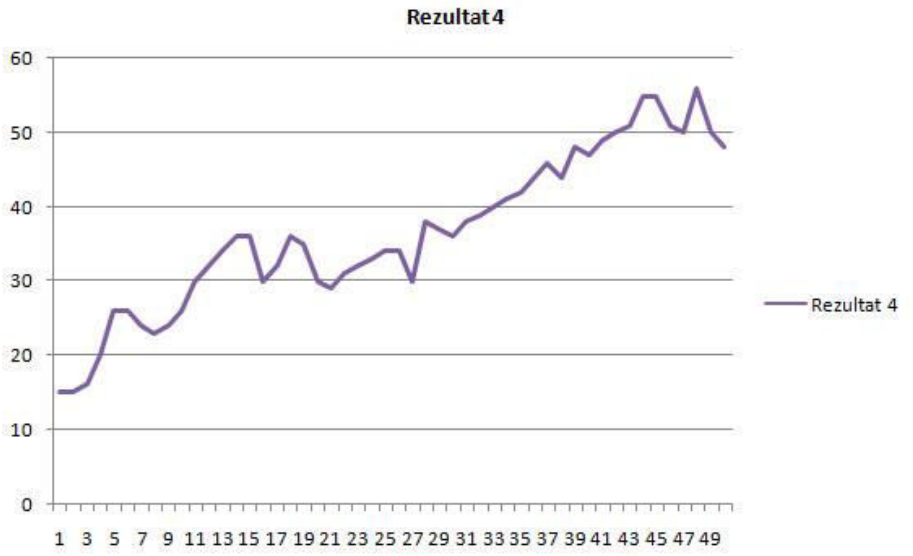
Od ovog grafika kreće korišćenje neuronskih mreža. Na ovom grafiku su prikazani rezultati u prvih 50 generacija. Može se videti da heli u prvih 10 genracija ostvaruje 0 poena, a onda se postepeno taj broj povećava, ali i pada (na primer u 20. generaciji je 0 poena). Generalno gledano skor raste a i veštine helikoptera.



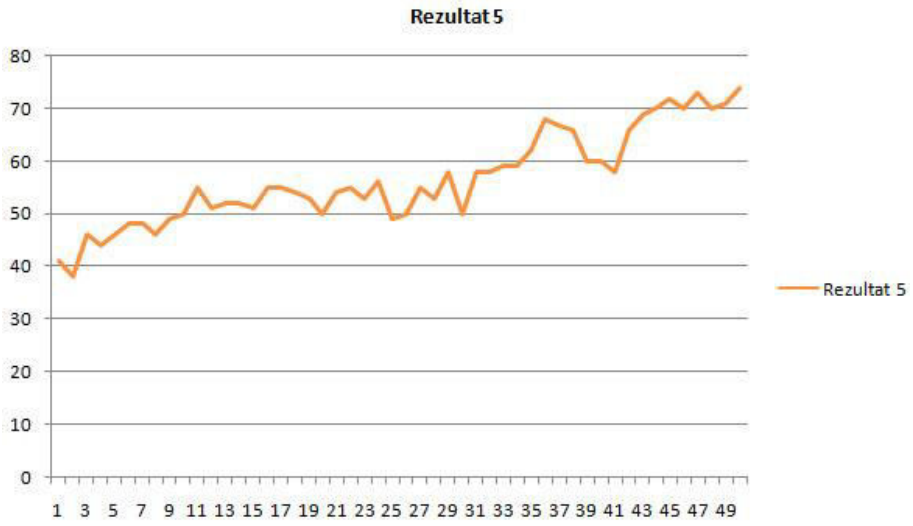
Na ovom grafiku su prikazani rezultati drugih 50 generacija, vidi se da crvena linija polako kreće da raste i da su rezultati bolji od prethodnog grafika.



Na ovom grafiku prikazani su rezultati još 50 generacija, vidi se da ima napredka u učenju helikoptera ali ima i padova, i grafik je približan onom prethodnom, stim da je ovde bio maksimali rezultat 40 poena.



Na ovom grafiku prikazani su rezultati još 50 genracija, jasno se vidi da se kriva (zelena linija) sve više približava jednoj konstantnoj liniji, dakle helicopter i dalje napreduje, ima padova, dešavalo se da posle uspešne generacije počne da pada skor, pa čak i na 0 poena, jer nije baš toliko pametan, ali ipak je napredovao i maksimalni rezultat je bio 50.



Na ovom grafiku prikazani su rezultati još 50 generacija, helicopter sada poseduje određene veštine, naravno pametnije je od prethodnog, opet ima padova ali je kriva (narandžasta linija) sve "pravija" i rezultati su sve bolji.

4 Zaključak

Neuronske mreže kombinovane sa genetskim algoritmom čine fascinirajuće rešenje za ovakav tip igre. Igra nije deterministička, niti postoji jedno optimalno rešenje, tako da kombinovanje postojećih rešenja primenom genetskog algoritma često može da poboljša postojeća rešenja. Neuronske mreže nama omogućavaju da uzmemo u obzir razne faktore u samoj igri, poput udaljenosti od stuba i visina, koje se potom faktorišu na različite načine kroz mrežu da bi se donela odluka o skakanju. Ovo nama omogućava fleksibilan algoritam koji se uz pomoć mutacije dorađuje i time generišemo više različitih vrhunskih rešenja.

Literatura

[1] http://solair.eunet.rs/~ilicv/neuro.html

[2] http://www.gf.uns.ac.rs/~zbornik/doc/ZR20.05.pdf

[3] https://sr.wikipedia.org/sr-ec/Неуронска_мрежа

[4] http://www.tf.ni.ac.rs/images/Neuronske_mreze.pdf

[5] Computational Intelligence - An Introduction, Andries Engelbrecht, John Willey & Sons, 2007.