SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINAR

Evolucija ponašanja jednostavnih bića genetskim algoritmima

Marko Benačić

Voditelj: Marko Đurasević

Zagreb, svibanj, 2018.

Sadržaj

1. Uvod

Od začetka računarstva pa do sadašnjosti, problemi na koje je računarska znanost nailazila su sve više rasli kako u veličini tako i u svojoj kompleksnosti. Inženjeri su shvatili da su takvi problemi izvan opsega ljudskog mozga, te da neke probleme ne mogu efektivno riješiti klasičnim pristupom i jednostavnim algoritmima. Neki od tih problema su: učenje neuronskih mreža, problem trgovačkog putnika, strategija igara, itd. Kao potencijalni način rješavanja takvih problema, pojavljuju se genetski algoritmi. Oni nam omogućuju pronalaženje sve boljih i učinkovitijih rješenja koristeći vrlo intuitivni i naizgled jednostavni proces koji vuče inspiraciju iz prirode - evoluciju. Evolucija je robustan proces pretraživanja prostora rješenja. Živa bića se tijekom dugog i repetitivnog procesa prirodne selekcije te pokušaja i pogreške prilagođavaju svojoj okolini i vrlo često nepovoljnim uvjetima. Jedinka koja je najbolje prilagođena uvjetima u kojima živi ima najveću šansu preživljavanja te slijedno tome i najveći izgled za prenošenje svojeg genetskog materijala preko potomaka. Analogno tome, genetski algoritmi također koriste proces selekcije i genetske operatore te određenu funkciju cilja kao pokazatelj koliko su dobro ili loše „prilagođeni uvjetima“ koje smo im zadali, tj. kolika je vjerojatnost da će određena jedinka genetskog algoritma „preživiti“ (Golub).

Cilj ovog seminara je objasniti koncept i motivaciju genetskih algoritama te njihovu primjenu u stvaranju vlastite jednostavne neuronske mreže, koju ćemo dalje koristiti kao podlogu za simulaciju ponašanja jednostavnih organizama i njihovu evoluciju unutar virtualnog okruženja.

2. O genetskim algoritmima

2.1. Prirodni evolucijski procesi

Evolucija je neprekidan proces prilagođavanja živih bića svojoj okolini, tj. na uvjete u kojima žive. U prirodi vlada nemilosrdna borba za opstanak u kojoj pobjeđuju najbolji, a loši umiru. Da bi neka vrsta tijekom evolucije opstala, mora se prilagođavati uvjetima i okolini u kojoj živi, jer se i uvjeti i okolina konstantno mijenjaju. Svaka slijedeća generacija neke vrste mora pamtiti **dobra** svojstva prethodne generacije, pronalaziti i mijenjati ta svojstva tako da ostanu dobra u neprekidno izmjenjujućim uvjetima [1].

Svaka jedinka u prirodi može se okarakterizirati nizom svojstava koji ih razlikuju od ostalih jedinki, kao npr. brzina trčanja, dobar vid, sluh, njuh, boja kože, boja očiju itd. Jedinke koje imaju loša svojstva, tj. loše su prilagođena okolini, imaju malu vjerojatnost preživljavanja u borbi za opstanak, te će najvjerojatnije umrijeti i neće prenijeti svoj genetski kod preko svojih potomaka. Dakle, loša svojstva imaju veliku vjerojatnost odumiranja zajedno sa jedinkom koja ih posjeduje, a dobra svojstva imaju veću vjerojatnost prenošenja na slijedeću generaciju.

Raznolikost živih bića i njihova odlična prilagođenost raznim uvjetima, ne bi bila moguća bez **križanja, mutacija i prirodne selekcije**. To su ključni mehanizmi koji pokreću evoluciju i konstantno poboljšavanje i prilagođavanje. Mutacije mogu biti korisne, ali i štetne za jedinku. Križanjem genetskog materijala oba roditelja se genetski materijal prenosi na njihove potomke, te se isti geni roditelja mogu izmiješati na ogroman broj načina, što doprinosi raznolikosti koju vidimo u prirodi.

2.2. Jednostavni genetski algoritam

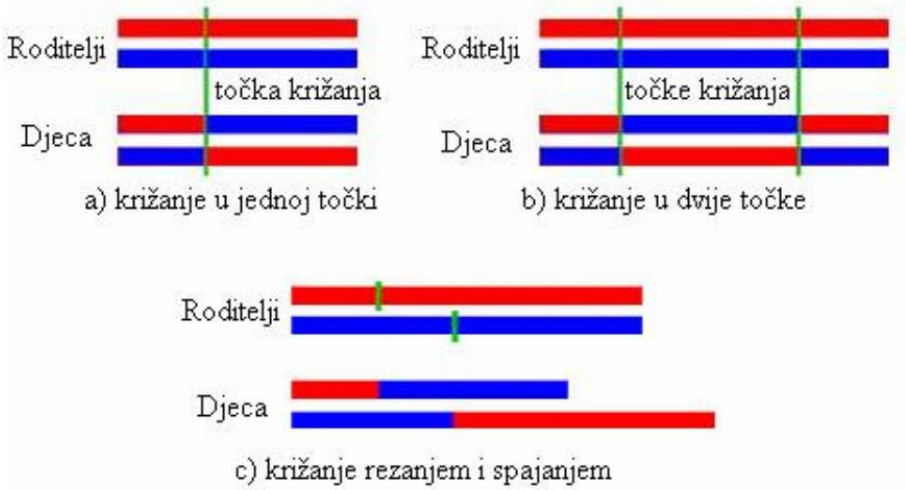
Iz prethodno navedenih prirodnih evolucijskih procesa možemo zaključiti da je evolucija, po svojoj definiciji, neka vrsta metode optimiranja u prirodi. Odatle proizlazi ideja za genetske algoritme, oni sadrže sve što sadrži i mehanizam prirodnog odabira.

J. H. Holland je u svom radu predložio (jednostavni) genetski algoritam kao računarski proces koji imitira evolucijski proces u prirodi i primjenjuje ga na apstraktne jedinke. Svaka jedinka predstavlja potencijalno rješenje problema koji se obrađuje; to može biti matematička funkcija, plan rada neke tvornice, put trgovačkog putnika i sl. Svaka je jedinka predstavljena jednakom podatkovnom strukturom (broj, niz, matrica, stablo itd.).Jedinke mogu biti predstavljene na razne načine, a najjednostavniji način kojim bismo ih predstavili bi bio niz bitova u kojemu svaki slijed bitova predstavlja neku informaciju (genetski kod). Te jedinke se nazivaju **kromosomi**. Svakom rješenju se pridjeljuje određena mjera kvalitete koja se u literaturi obično naziva **dobrota** (engl. *fitness*), dok se funkcija koja tu kvalitetu određuje naziva **funkcija cilja** ili **funkcija dobrote**.Tada se iz stare formira nova populacija izdvajajući, po nekom postupku

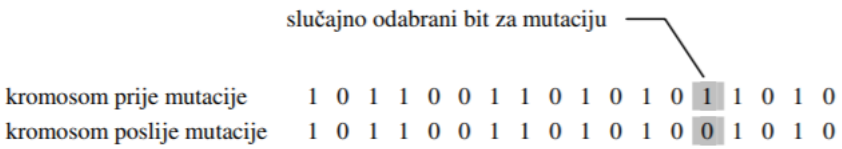
odabira, bolje jedinke iz skupa postojećih (turnirska selekcija, eliminacijska selekcija). Neki članovi ove nove populacije podvrgnuti su utjecajima genetskih operatora (**mutacije, križanje**) koji iz njih formiraju nove jedinke.

Križanje je genetski operator u kojem jedinka nove generacije sadrži po dio genetičkog koda od svakog roditelja. Npr genetski kod (u našem slučaju niz bitova) oba roditelja se presiječe na polovici (ili nasumičnoj lokaciji) te se ta dva (ili više) odsječka spoje u jedan. Primjer križanja prikazan je na slici 1.

Mutacija je genetski operator koji se upotrebljava za dobivanje genetske raznolikosti sljedeće generacije rješenja od one postojeće. Najjednostavniji primjer mutacije je vjerojatnost da se neki bit u genetskom kodu (tj. rješenju) promijeni iz svog originalnog stanja u neko novo stanje, npr. promijeni iz dvobitne vrijednosti 1 u vrijednost 0. To se postiže uvođenjem neke varijable za svaki bit u nizu. Ta varijabla govori kolika je vjerojatnost hoće li neki bit biti modificiran ili neće. Primjer mutacije je vidljiv na slici 2.



Slika 1: Križanje



Slika 2: Mutacija

Nakon određenog broja izvršenih generacija čitav postupak se zaustavlja kada se zadovolji uvjet zaustavljanja, a najbolji član trenutne populacije predstavlja rješenje koje bi trebalo biti sasvim blizu optimuma. Križanjem se prenose svojstva roditelja na djecu. Mutacijom se mijenjaju svojstva jedinke slučajnom promjenom gena. Takvim postupkom se postiže iz generacije u generaciju sve veća i veća prosječna dobrota populacije.

Postoji opasnost da se dobro rješenje dobiveno nakon puno iteracija izgubi ukoliko ga genetski operatori (npr. mutacija ili selekcija) izmijene. Stoga se javlja potreba za mehanizmom zaštite najbolje jedinke od bilo kakve izmjene ili eliminacije tijekom evolucijskog procesa. Takav mehanizam se naziva **elitizam**. Elitizam nam garantira da genetski algoritam u svakom trenutku teži k globalnom optimumu, tj. onemogućava da sljedeća generacija bude lošija od prethodne. Jedini nedostatak je što pronalazak najboljih jedinki zahtjeva procesorsko vrijeme zbog kojeg se može znatno usporiti genetski algoritam.

```
Genetski_algoritam
{
   t = 0
   generiraj početnu populaciju potencijalnih rješenja P(0);
   sve dok nije zadovoljen uvjet završetka evolucijskog procesa
   {
        t = t + 1;
        selektiraj P'(t) iz P(t-1);
        križaj jedinke iz P'(t) i djecu spremi u P(t);
        mutiraj jedinke iz P(t);
   }
   ispiši rješenje;
}
```

Slika 3: Struktura genetskog algoritma

3. O neuronskim mrežama

3.1. Motivacija i inspiracija

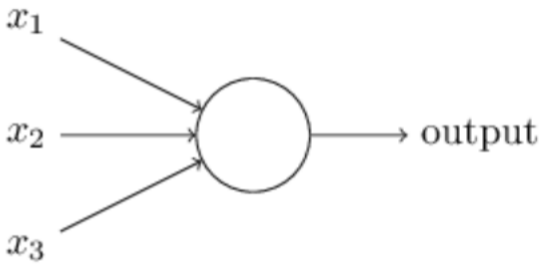
Umjetna neuronska mreža (eng. Artificial Neural Network, skraćeno ANN) je skup umjetnih neurona (najčešće kao apstraktnih pojmova) koji su međusobno povezani i interaktivni kroz operacije obrade signala. Nastale su kao prirodom inspirirana moguća realizacija umjetne inteligencije. Iako neuronske mreže nisu novi koncept (prvi matematički model iz 1943.), njihova upotreba i korisnost se naglo povećala unazad zadnjih par godina, uglavnom jer su računala napokon dostigla brzinu koja je potrebna za njihovo efikasno učenje i korištenje. Neke od područja u kojima se koriste neuronske mreže su: prepoznavanje slike, obrada prirodnog jezika, automatizirano trgovanje (burze), autonomni automobili.

Njihova inspiracija proizlazi iz građe ljudskog živčanog sustava. Živčani sustav se sastoji od 6 milijuna međusobno povezanih živčanih stanica – **neurona**. Njihova uloga se može konceptualizirati kao prihvaćanje, obrađivanje i odašiljanje podataka. Analogno tome neuronske mreže se sastoje od mreže **umjetnih neurona**. Neuron i umjetni neuron oba imaju dio koji služi za primanje informacija, dio za obradu, prijenos te dio za prosljeđivanje obrađene informacije.

Danas postoji nekoliko glavnih vrsta mreža, no temeljno je obilježje svih mreža, bez obzira na oblik i broj veza unutar njih, da se odlikuju svojstvom „učenja", tj. uvježbavanja kroz niz ponavljajućih postupaka analize. Od cijeloga skupa podataka veći dio upotrijebljen je za učenje, a manji za ponovno predviđanje poznatih vrijednosti.

3.2 Struktura neuronske mreže

Povijesno gledano, neuronske mreže se sastoje od više slojeva umjetnih neurona koji se nazivaju **perceptron**. Perceptroni su razvijeni u 50-im i 60-im godinama prošlog stoljeća (Frank Rosenblatt). Iako se moderne neuronske mreže sastoje od drukčijih modela umjetnih neurona kao što je npr. **sigmoidni neuron**, da bi se shvatila osnovna građa mreže potrebno je objasniti građu perceptrona. Perceptroni kao ulazne podatke primaju niz ulaza $x1,x2,x3$... te na izlaz daje samo jedan binarni podatak (output). Primjer jednostavnog perceptrona vidljiv je na slici 4.



Slika 4: Perceptron

Rosenblatt je predložio jednostavno rješenje računanja izlaza (outputa). Izlaz neurona može biti 0 ili 1, a on ovisi o sumi umnožaka ulaza *Xi* i njihovih pripadajućih težina *Wi* (weight). Ako je suma manja od određene granice na izlaz će dati 0, a ako je suma veća na izlaz će dati 1 .Jednostavna izlazna funkcija perceptrona prikazana je na slici 5. Težinu koja je pridodana svakom ulazu možemo shvatiti kao koeficijent važnosti određenog svojstva koje ulazi u perceptron. Što je težina manja, to njoj pridruženi ulaz ima manju važnost pri promjeni izlazne funkcije (outputa). Uzmimo za primjer donošenje neke odluke iz svakodnevnog života: ako želimo donijeti odluku hoćemo li se ići kupati na Jarun, današnja vremenska prognoza ima puno veću **težinu** za našu odluku, nego činjenica da se nismo naspavali i da smo umorni (jer ako pada kiša sigurno se nećemo ići kupati, a ako smo umorni postoji mogućnost da svejedno odemo). Dakle perceptron je model koji donosi odluke na temelju svih ulaznih podataka. Treba imati na umu da težine mogu imati i negativne vrijednosti.

$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{ threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{ threshold} \end{cases}$$

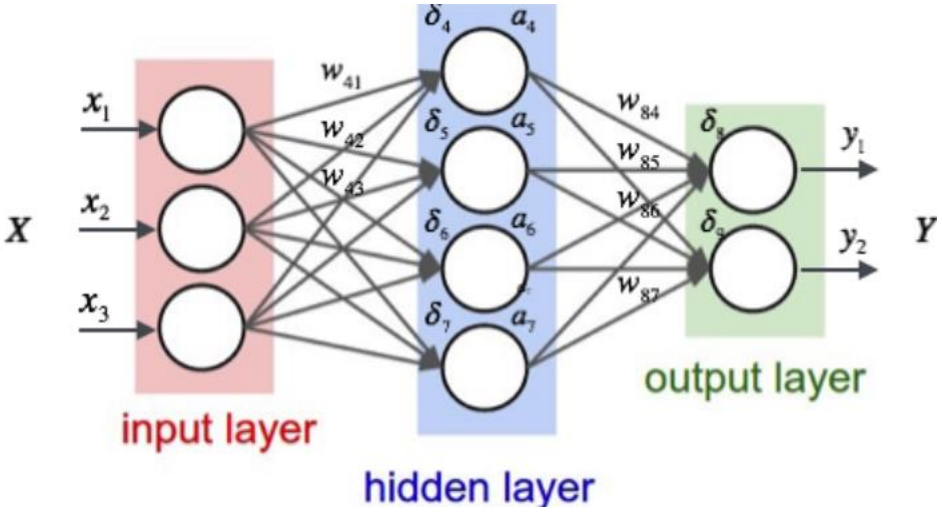
Slika 5: Izlazna funkcija perceptrona

Neuronska mreža se sastoji od tri sloja takvih neurona: **ulazni, skriveni i izlazni sloj**. Svaki sloj može imati veliki broj neurona, ovisno o upotrebi. Izlazi svakog neurona svakog sloja se šalju na ulaz svakog neurona sljedećeg sloja, gdje on onda predstavlja ulazni podatak u tom novom sloju. Ovakav oblik povezivanja naziva se **feed-forward** mreža, jer podatci putuju u samo jednom smjeru, tj. prema izlazu. Izlazi i ulazi se mogu i drugačije organizirano spajati pa se ovisno o pravilima po kojima se to radi mogu klasificirati određene vrste mreža. Skriveni sloj može biti konstruiran od više slojeva različitih veličina, ali se u jednostavnom modelu predstavlja kao jedan sloj. Skriveni sloj je najvažniji te se u njemu odvija većina „razmišljanja“ neuronske mreže.

Osim klasičnih neurona postoje i neuroni koji nemaju ulaz, a na izlazu imaju konstantu, tzv. **bias** (hrv. naklonost). Oni služe kao konstanta na ulazu običnog neurona. Zato se izlaz može pisati kao funkcija varijabli *w*x* + *b*, gdje su *w* – težina, *x* – ulaz, i *b* – naklonost. Pa se izlazna funkcija može pisati i ovako: (Slika 6).

$$\text{output} = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases}$$

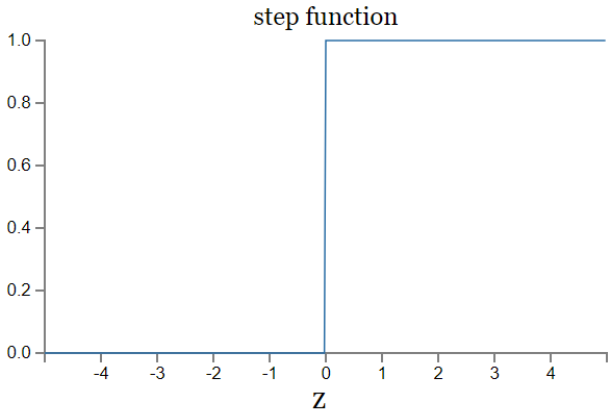
Slika 6: izlazna funkcija sa biasima



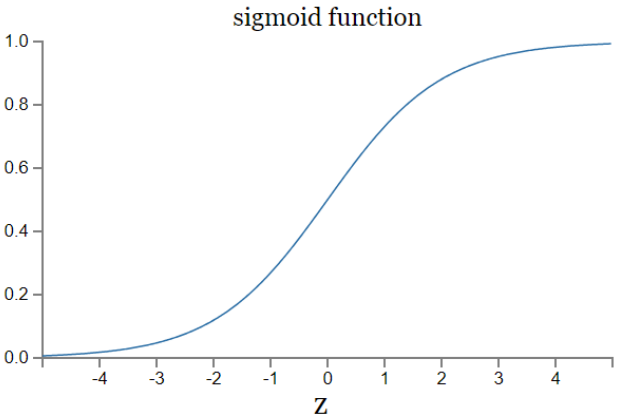
Slika 7: Model neuronske mreže

3.3. Sigmoidni neuroni i backpropagation algorithm

U konkretnim primjenama neuronskih mreža na stvarne probleme, perceptronska mreža nije dovoljno dobra i uglavnom neće davati dobre rezultate. Razlog tome je činjenica što u perceptronu uvijek postoji nagli prijelaz iz outputa 0 i outputa 1. Kod stvarnih problema za dobro rješenje je obično potrebno napraviti puno diskretnije promjene nego nagli prijelaz iz nule u jedinicu kao što imamo kod perceptrona. Dakle izlaz iz sigmoidnog neurona neće izgledati kao **step function** (slika 8), već kao njena izglađena verzija, tzv. **sigmoid** (slika 9). Dakle kod sigmoidnih neurona male promjene Δw_j u težinama te male promjene Δb u biasima, će dati male promjene na outputu neurona. Puno efektivniji način učenja modernih neuronskih mreža je metoda gradijentnog spusta, pomoću *backpropagation* algoritma To su matematički postupci koji su previše složeni za ovaj seminar da bismo uazili u dubinu. Ukratko, svode se na pronalaženje lokalnog ili globalnog minimuma n-dimenzionalne **funkcije cilja**, koja nam je pokazatelj koliko dobro se izlaz mreže slaže sa željenim izlazom.



Slika 8: step funkcija



Slika 9: sigmoidna funkcija

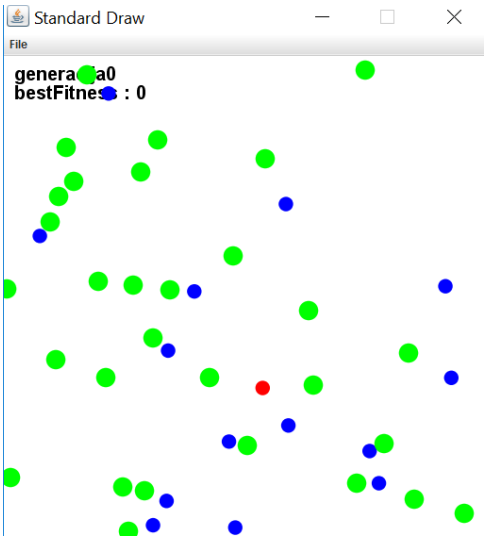
4. Kreiranje vlastitih evolucijskih organizam

U ovom poglavlju ćemo iskoristiti stečeno znanje iz neuronskih mreža i genetskih algoritama pri kreaciji jednostavnih umjetnih organizama, te simulirati njihovo ponašanje i evolucijski razoj. Da bismo mogli simulirati ponašanje i evoluciju tih organizama moramo postaviti skup početnih parametara, njihovo ciljano ponašanje te okolinu u kojoj će živjeti.

4.1 Početni parametri i korištene tehnologije

Za realizaciju simulacije koristiti ćemo programski jezik Java. Od vanjskih libraryja koristiti ćemo La4j i StdDraw. La4j je library za obavljanje jednostavnih operacija linearne algebre koje su nam potrebne za realizaciju neuronske mreže. StdDraw je klasa sa metodama koje su nam potrebne za crtanje virtualne okoline te vizualnu reprezentaciju stvaranja hrane i kretanja umjetnih bića.

Prvo moramo stvoriti okruženje u kojemu će kretati organizmi. Okruženje je jednostavan prozor sa bijelom pozadinom veličine 512 x 512. Osim toga, želimo stvoriti hranu koju će organizmi „konzumirati“. Glavni cilj organizama je konzumacija što većeg broja hrane. Količina konzumirane će nam također služiti kao mjera dobrote organizma. Hrana je predstavljena klasom koja sadrži samo svoju x i y koordinatu unutar okruženja. Prilikom stvaranja svake hrane njihova x i y koordinata se inicijaliziraju na nasumično odabranu vrijednost [0, 512] te stoje tamo dok organizam ne prijeđe preko nje, tj. pojede ju.



Slika 10: Virtualno okruženje; organizmi – plavi, hrana - zelena

Sada kada smo postavili teren za igru, vrijeme je da stvorimo organizme. Kao i hrana, organizmi imaju svoju x i y koordinatu koje određuju njegov položaj unutar okoline. Osim toga, oni imaju mogućnost kretanja po okolini, pa imaju dvije varijable koje određuju njihov smjer kretanja po x i y osima. Prilikom završetka svake jedinice vremena, ogranizmi obave kretanje koje je određeno njihovim varijablama x_smjer i y_smjer. Mehanizam kretanja prikazan je slikom 11. Nadalje, svaki organizam ima u sebi svoju jednostavnu neuronsku mrežu, koja odlučuje koja će biti iduća vrijednost x_smjer i y_smjer, tj. u kojem će se smjeru kretati organizam.

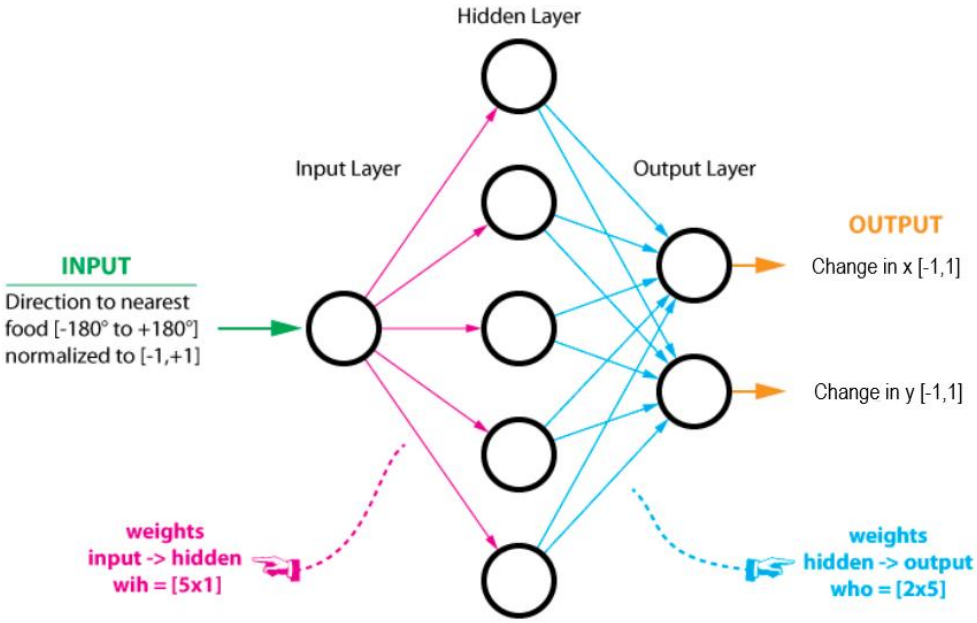
```
public void move() {
    this.x = (this.x + (smjer_x*brzina)) % 512 ;
    if (x < 0) x = 512;
    this.y = (this.y + (smjer_y*brzina)) % 512 ;
    if(y < 0) y = 512;
}
```

Slika 11: Kretanje po ploči

4.2 Implementacija Neuronske mreže

Kao što je već rečeno, glavni cilj svakog organizma je „konzumirati“ što veći broj hrane koja se nalazi na terenu. Cilj optimizacije naše neuronske mreže je da na temelju ulazne informacije neuronska mreža mora donijeti odluku u kojem će smjeru obaviti kretanje u idućoj jedinici vremena. Ulazna informacija u neuronsku mrežu je kut od organizma do njemu najbliže hrane. Na temelju te informacije, želimo da naši organizmi sami donesu odluku koja će ih dovesti do hrane, koju oni potom „konzumiraju“ tako što pređu preko nje. Tu odluku donose unutar metode think, koja je prikazana na slici 13.

Neuronska mreža će se sastojati od 3 sloja; ulazni sloj će imati samo jedan ulaz (kut do najbliže hrane), hidden sloj će imati 5 neurona (iako je moguće promijeniti parametre), te izlazni sloj koji će imati dva neurona koji će nam reći u kojem x i y smjeru se organizam mora pomaknuti. Kao aktivacijsku funkciju naše mreže smo koristili tangens hiperbolni, pa ćemo ulaznu informaciju o kutu do hrane, normalizirati na vrijednosti između -1 i 1.



Slika 12: Shema neuronske mreže

```
public void think(ArrayList<Food> food) {

    Food najHrana = najblizaHrana(food);

    DenseMatrix input = DenseMatrix.zero(4, 1);

    input.set(0, 0, (double) getAngle(najHrana));
    input.set(1, 0, (double) 1);
    input.set(2, 0, (double) 1);
    input.set(3, 0, (double) 1);


    DenseMatrix hiddenLayer = wih.multiply(input).toDenseMatrix();
    for(int i = 0; i < hiddenLayer.rows(); i++) {
        for(int j = 0; j < hiddenLayer.columns(); j++) {
            hiddenLayer.update(new MatricaFunkcija());
        }
    }
    DenseMatrix output = who.multiply(hiddenLayer).toDenseMatrix();
    for(int i = 0; i < hiddenLayer.rows(); i++) {
        for(int j = 0; j < hiddenLayer.columns(); j++) {
            output.update(new MatricaFunkcija());
        }
    }

    smjer_x=(float)output.get(0, 0);
    smjer_y=(float)output.get(1, 0);
```

Slika 13: Razmišljanje

Svaka težina unutar mreže će biti postavljena na nasumične vrijednosti , pa će se početne jedinke u prvoj generaciji kretati nasumično. Tek nakon optimiziranja mreže genetskim algoritmom će organizam znati kako se kretati da bi došao do hrane.

4.3 Optimiziranje neuronske mreže pomoću genetskog algoritma

Glavne dvije informacije koje su nam potrebne da bismo uspješno optimizirali naše neuronske mreže i naučili organizme da konzumiraju hranu su **dobrota** i **genetski kod**. U našem slučaju dobrotu predstavlja količina hrane koju je organizam pojeo, a genetski kod nam predstavljaju težine svakog neurona njihove neuronske mreže.

Prvo moramo odrediti nakon koliko vremena želimo da se jedna iteracija genetskog algoritma provede. Potrebno je ostaviti dovoljno vremena da bi organizmi imali šansu proći određeni put i pojesti neku hranu. Kada je prošao taj interval možemo primjeniti genetski algoritam. Određujemo koji će broj **najboljih** organizama preživjeti te njih stavljamo u listu najboljih (elitizam). Elitni organizmi ne umiru pojavom nove iteracije nego njihov genetski kod ostaje sačuvan. Nakon toga stvaramo nove organizme koji su potomci elitnih dok ne upopunimo željeni broj novih organizama. Nadalje, određujemo kako će izgledati genetski kod novih organizama pomoću križanja nad slučajno izabrana dva elitna organizma.

```
//crossover
int p1 = rand1.nextInt(BROJ_ELIT);
int p2 = rand1.nextInt(BROJ_ELIT);
if(p1==p2) p2 = (p1 + 1) % BROJ_ELIT;

Organism parent1 = topGeneracija.get(p1);
Organism parent2 = topGeneracija.get(p2);

double crossover_weight = Math.random();
DenseMatrix wih_new = parent1.wih.multiply(crossover_weight)
                            .add(parent2.wih.multiply(1 - crossover_weight)).toDenseMatrix();
DenseMatrix who_new = parent1.who.multiply(crossover_weight)
                                 .add(parent2.who.multiply(1 - crossover_weight)).toDenseMatrix();
noviOrganizam.wih = wih_new;
noviOrganizam.who = who_new;
```

Slika 14: Križanje

Zatim provodimo operaciju mutacije. Operacija mutacije se ne provodi nad svim novim organizmima, nego postoji određena vjerojatnost da će se dogoditi mutacija. Tu vjerojatnost smo ranije odredili te je ona također jedan od bitnih parametara pri optimizaciji našeg algoritma. U našem slučaju vjerojatnost mutacije je 10%. Algoritam mutacije prikazan na slici 15.

```
//mutacija
double mutate = Math.random();
if(mutate <= mutation_rate) {   //mutation rate

    int mat_pick = rand1.nextInt(2);

    if(mat_pick == 0) {
        int index_row = rand1.nextInt(5);
        int index_column = rand1.nextInt(4);
        noviOrganizam.wih.set(index_row, index_column,
                noviOrganizam.wih.get(index_row, index_column) * ((double) 1.1 - ((double) rand1.nextInt(3) / (double)10));
        if(noviOrganizam.wih.get(index_row, index_column) > 1)
            noviOrganizam.wih.set(index_row, index_column, 1);
        if(noviOrganizam.wih.get(index_row, index_column) < -1)
            noviOrganizam.wih.set(index_row, index_column, -1);
    }
    if(mat_pick == 1) {
        int index_row = rand1.nextInt(2);
        int index_column = rand1.nextInt(5);
        noviOrganizam.who.set(index_row, index_column,
                noviOrganizam.who.get(index_row, index_column) * ((double) 1.1 - ((double) rand1.nextInt(3) / (double)10));
        if(noviOrganizam.who.get(index_row, index_column) > 1)
            noviOrganizam.who.set(index_row, index_column, 1);
        if(noviOrganizam.who.get(index_row, index_column) < -1)
            noviOrganizam.who.set(index_row, index_column, -1);
```

Slika 15: Mutacija

Svakom iteracijom jedinke će postajati sve bolje u pronalaženju hrane, što znači da smo postigli svoj cilj. Također je važno napomenuti da algoritam koji smo implementirali nije savršen te uvelike ovisi o slučajnosti početnih parametara, ali obavlja dovoljno dobar posao s obzirom na svoju jednostavnost. Unatoč tome, unutar ove jednostavne mreže postoje mnoge mogućnosti za poboljšanje i optimiziranje. Na primjer dodavanje više ulaznih parametara u neuronsku mrežu, određivanje idealnog broja skrivenih neurona, ubrzavanje/usporavanje šanse za mutacijom, povećavanje/smanjivanje pojave hrane itd...

5. Zaključak

Unatoč lošim prognozama u prošlosti, neuronske mreže se brzo razvijaju i omogućuju nam da rješavamo sve više raznolikih problema u stvarnom svijetu, od prepoznavanja rukopisa pa do samovozećih auta i umjetne inteligencije. Zahvaljujući genetskim algoritmima i ostalim matematičkim metodama, neuronske mreže optimiziramo vrlo brzo i efikasno. Smatram da je razvoj umjetne inteligencije vrlo obećavajuće područje i da će imati ogroman utjecaj na daljnji razvoj čovječanstva i tehnologije te je vrlo uzbudljivo i inspirirajuće biti dio toga svijeta budućnosti. S druge strane, pomalo je zastrašujuće razmišljati o činjenici da će jednog dana u budućnosti velika većina ljudske djelatnosti biti zamjenjena raznim vrstama umjetne inteligencije, ali moje osobno mišljenje je da razvitak tehnologije koja je uslužna čovjeku može dovesti sa sobom samo porast životnog standarda i udobnosti.

6. Literatura

1. Genetski algoritmi (Marin Golub)

2. https://en.wikipedia.org/wiki/Artificial_neural_network

3. Neural networks and deep learning (Michael Nielsen)

4. https://gormanalysis.com/introduction-to-neural-networks/

5. What is a neural network? : https://www.youtube.com/watch?v=aircAruvnKk&feature=youtu.be

6. https://nathanrooy.github.io/posts/2017-11-30/evolving-simple-organisms-using-a-genetic-algorithm-and-deep-learning/

7. Reference

7. La4j - http://la4j.org/

8. StdDraw - https://introcs.cs.princeton.edu/java/stdlib/javadoc/StdDraw.html