UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Aleš Uršič

EVOLUCIJA NEVRONSKIH MREŽ

DIPLOMSKO DELO
NA UNIVERZITETNEM ŠTUDIJU

Mentor: prof. Marko Robnik Šikonja

Solkan, 2012

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01853/2012
Datum: 17.05.2012

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **ALEŠ URŠIČ**

Naslov: **EVOLUCIJA NEVRONSKIH MREŽ**
EVOLUTION OF NEURAL NETWORKS

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Umetno življenje je področje računalništva, ki poskuša z računalniškimi modeli simulirati določene fenomene, ki so sicer značilni za življenjske oblike. Zasnujte model preprostega umetnega okolja, v katerem poteka evolucija umetnih organizmov, predstavljenih z nevronskimi mrežami. Preučite ustrezno literaturo in že izdelana okolja ter implementirajte lasten prototip takšnega okolja. Okolje in potek življenja ustrezno grafično vizualizirajte. Simulirajte potek evolucije pod različnimi okoliščinami ter rezultate statistično ovrednotite.

Mentor:
prof. dr. Marko Robnik Šikonja

Dekan:
prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

I Z J A V A O A V T O R S T V U

diplomskega dela

Spodaj podpisani Aleš Uršič,

z vpisno številko 63060302,

sem avtor diplomskega dela z naslovom:

Evolucija nevronskih mrež

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom prof. dr. Marka Robnik Šikonje
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.

V Solkanu, dne 5. junij 2012 Podpis avtorja:

Znanje je pomembno. Toda še veliko pomembnejša je njegova koristna uporaba. Ta je odvisna od srca in uma človeka.

Dalaj Lama

Zahvala

Najprej se zahvaljujem staršem za pomembno podporo tekom študija, tako čustveno in spodbujevalno kot tudi finančno. Še bolj pa cenim, da so mi v času pisanja diplome omogočili nemoteno in kakovostno delo.

Hvaležen sem sestri in starim staršem na obeh straneh, kjer ni manjkalo motivacije in pozitivnih dejanj.

Hvala dekletu Maji Primožič, ki je stala z menoj od prvega leta študija.

Zelo lepo se zahvaljujem odličnemu mentorju, s katerim sva z zanimivimi diskusijami dobro dodelala in obogatila diplomsko delo ter še marsikatero drugo znanje.

Ravno tako se zahvaljujem sostanovalcu Matjažu Bizjaku in prijateljem Sandiju, Silvotu ter Mitji, kajti marsikatera ideja za diplomo se je rodila pri globokih pogovorih z njimi.

Kazalo vsebine

Povzetek

Cilj diplomske naloge je postavitev ustreznega modela umetnega življenja in simulacija organizmov v okolju s hrano. Organizmi preživijo, če učinkovito iščejo hrano. Z evolucijo in učenjem organizmi razvijejo nevronsko mrežo, ki jim to omogoča.

Najprej predstavim nevronske mreže, njihov zgodovinski razvoj, razložim osnovne koncepte kot so model nevrona, povezava nevronov v nevronsko mrežo, preklopne funkcije, topologije, učenje in vrste nevronskih mrež. Opišem vzvratno pravilo učenja na večnivojski nevronski mreži in ga ponazorim na dveh izbranih primerih. Podam model umetnega življenja ter razvijem več možnih topologij nevronskih mrež za organizme. Analiziram delovanje različnih nevronskih mrež, ki predstavljajo organizme in jih primerjam po kriteriju srednje kvadratne napake. Ugotovim vpliv začetnih uteži in učne množice na kvaliteto učenja organizmov in pokažem, da so možne velike razlike v kvaliteti učenja med nevronskimi mrežami. Simuliram eno generacijo organizmov (brez evolucije), spremljam in beležim obnašanje ter rezultate statistično obdelam. Analiziram uspešnost preživetja pri različnih parametrih simulacije kot so poraba hrane, velikost začetne populacije, količina hrane v okolju, hitrost učenja in ostalo.

Darwinova teorija evolucije je povezana z evolucijskim računanjem. Pojasnim osnovne pojme kot so predstavitev osebkov, selekcija, razmnoževanje, mutacije in preživetvena strategija. Demonstriram uporabo evolucijskega računanja na dveh primerih z različnimi metodami selekcije, križanja in preživetja. Ugotovim najboljšo metodo za specifičen primer in to preverim s statističnim testom.

Razvijem lasten model evolucije nevronskih mrež. Z numerično simulacijo evolucije najdem dobro strukturo nevronske mreže. Pokažem, da evolucija ne najde vedno enako dobre rešitve. Za boljši vizualni vtis dodam nevronski mreži še del za pretvorbo polarnih koordinat premikanja organizma v kartezične zato, da postane napredek organizmov bolj opazen.

Razvite koncepte predstavim v demonstracijskem programu, ki vsebuje grafično simulacijo evolucije in uporabniški vmesnik za nastavitev parametrov. Simuliram evolucijo z različnimi kriteriji in parametri ter analiziram razvoj kompleksnosti nevronskih mrež organizmov, natančnost pretvorbe polarnih koordinat v kartezične, umrljivost organizmov in učinkovitost iskanja hrane.

Črpanje znanja iz narave je v računalništvu in ostalih področjih zelo koristno, saj model evolucije organizmov, v mojem primeru nevronskih mrež, poleg zanimivih simulacij umetnega življenja omogoča reševanje številnih znanstvenih problemov, za katere ne poznamo učinkovitih algoritmov.

Ključne besede:

Nevronske mreže, evolucijsko računanje, umetno življenje, evolucija strukture nevronske mreže.

Abstract

The goal of this work is construction of an artificial life model and simulation of organisms in an environment with food. Organisms survive if they find food successfuly. With evolution and learning organisms develop a neural network which enables that.

First neural networks and their history are introduced with the basic concepts like a neuron model, a network, transfer functions, topologies and learning. I describe the backpropagation learning on multilayer feed forward network and demonstrate it on two examples. I present a model of artificial life and several topologies of neural networks for organisms. I analyse the mean squared error of different organisms. I analyse impact of learning set and initial weights on quality of learning and I show potentially large differences between different neural networks. A survival of organisms with different simulation parameters like food consumption rate, size of initial population, quantity of food in enviroment, learning speed are analyzed.

Darwin's theory is related to evolutionary computation. I explain it's basic concepts like representation of genome, selection, reproduction, mutation and survival strategy. I demonstrate the usage of evolutionary computation on two examples with different types of selection, crossover and survival strategies. The best method for each specific case is determined with the Student's t-test.

I develop a model for evolution of neural networks. With numerical simulation I search a good structure of neural network. The evolution does not allways find the optimal solution. To improve visualisation a new structure is added to a neural network for calculation of cartezian coordinates from polar ones. This makes the progress of organisms visible through generations.

The developed concepts are included in a demonstration with graphical simulation of evolution and a user interface for setting the parameters. I simulate the evolution with different parameters and analyse complexity of neural networks, accurancy of coordinate conversion, mortality of organisms and efficency in food searching.

Using the ideas from nature is useful and efficient in computer science. The evolution models (in my case neural networks) give interesting simulations of artificial life. They also help solving scientific problems where efficient algorithms are unknown.

Key words:

Neural networks, evolutionary computation, artificial life, evolution of neural network topology.

Poglavje 1

Uvod

1.1. Evolucijski pristop k računanju

Umetne nevronske mreže so matematičen model, ki temelji na strukturi in funkcionalnosti bioloških nevronskih mrež. Nevronska mreža je sestavljena iz medsebojno povezanih nevronov. V večini primerov je to adaptivni sistem, ki spreminja svojo strukturo v času učenja. Struktura se spreminja na osnovi zunanjih in notranjih informacij, ki jih mreža procesira.

Nevronske mreže se uporabljajo za modeliranje kompleksnih relacij ali za iskanje zakonitosti v podatkih. To pomeni iskanje predhodno neznane funkcije, ki je ne znamo določiti analitično.

Evolucijsko računanje je v računalništvu način računanja, ki črpa ideje iz narave. Naravno evolucijo si predstavljamo kot populacijo posameznikov v danem okolju, ki se borijo za obstanek in se ohranjajo z reprodukcijo. Okolje določa stopnjo prilagajanja posameznikov pri preživetju.

Darwinova teorija je ponudila razlago za razvoj življenja na Zemlji, kjer igra naravni izbor osrednjo vlogo. V okolju, kjer je mogoče omejeno preživetje števila posameznikov, je selekcija neizbežna. Selekcija, ki jo povzroča tekmovanje v prilagajanju okolju (naravi), je naravna in predstavlja boj za preživetje. Darwin je verjel, da se pri prenosu na potomce pojavljajo manjše fenotipske modifikacije oz. mutacije, zaradi česar prihaja do sprememb v obnašanju posameznikov. Posamezniki v populaciji so torej elementi selekcije, katerih reprodukcija je odvisna od njihove prilagoditve okolju. Ker se bolj prilagodljivi posamezniki reproducirajo, imajo njihovi potomci večjo možnost za še boljšo prilagoditev okolju.

V evolucijskem računanju je okolje problem, ki ga rešujemo, posameznik predstavlja možno rešitev problema, prilagajanje okolju pa je kvaliteta rešitve. Prednost evolucijskega računanja pri reševanju problemov iz najrazličnejših področij je njegova univerzalnost in neodvisnost od narave problema.

V nalogi analiziram evolucijski razvoj nevronskih mrež. Na začetku vsake generacije je množica organizmov predstavljena z nevronsko mrežo brez znanja. Nevronska mreža omogoča organizmu odločanje in prilagajanje. Organizmi so postavljeni v okolje z omejeno količino hrane. Iskanje hrane je nujno za njihovo preživetje. Organizmi najprej naključno tavajo v okolju, trkajo ob druge organizme ali stene in jedo. Ko organizem najde hrano, dobi pozitiven dražljaj, kar si njegova nevronska mreža zapomni. Več takšnih dražljajev dobi, bolj razume, kaj mora početi. Če organizem dobi premalo hrane, umre. V vsaki generaciji preživijo organizmi, ki so se okolju prilagodili, ostali umrejo. Nova generacija nastane s križanjem staršev. Genski material se med življenjem ne spreminja, zato novi organizmi podedujejo le lastnosti prednikov, ne pa tudi v življenju pridobljenega znanja. Organizem se mora sam učiti in prilagajati okolju. Pri križanju in mutaciji nastane nov genski material. Ta omogoči novim organizmom boljše ali slabše učenje. Potomci z boljšim učenjem imajo večje možnosti preživetja, kot počasneje učeči se organizmi, ker se okolju hitreje prilagajajo. Svoj genski material prenesejo na naslednjo generacijo. To imenujemo naravni izbor. Pričakujemo, da bodo sčasoma organizmi boljši. Dobili bomo generacije, kjer bo večina organizmov sposobna učinkovitega učenja in preživetja.

1.2. Motivacije in zasnova modela

Modela evolucije in učenja organizmov sem poskušal narediti na čimbolj naraven način. Razmišljal sem, kako človek poišče najbližji predmet v vidnem polju, ki ustreza skupini predmetov (hrana v primeru organizmov). Izkazalo se je, da to ni najbolj enostaven problem. Poskusil sem nekaj različnih primerov vizualne zaznave, izmed katerih sem izbral najboljšega. Podobno mojemu modelu delujejo roboti sesalci.

Najprej je bilo potrebno sestaviti nevronsko mrežo, ki bo na osnovi aktivnosti senzorjev določila tistega, ki je zaznal najbližji predmet in na izhod postavila kot premika, ustrezen lokaciji senzorja. Navadna naprej povezana nevronska mreža je dajala slabe rezultate. Za takšen problem bi bilo potrebno imeti vsaj 6 skritih slojev, kar pa prinaša veliko težav pri učenju. Zato sem začel iskati alternativne rešitve. Po daljšem razmisleku sem prišel do ideje, da bi procesiranje razdelili na dva nivoja, kjer bi en nivo samo določil najbolj aktiven senzor, ostale pa bi naredil neaktivne, drugi nivo bi pa vrnil preslikavo kota. S poizkusi sem ugotovil, da je drugi nivo takšne mreže možno realizirati z naprej povezano nevronsko mrežo. Prvi nivo nevronske mreže sem našel iskati v Kohonenovih nevronskih mrežah, ki uporabljajo tekmovalni proces za določitev zmagovalnega nevrona. Sprevidel sem, da je iskanje najbolj aktivnega nevrona nevrona določanje zmagovalnega nevrona. Tako je bilo treba le še povezati nevrone v ustrezno topologijo (vsak povezan z vsakim) in začeti tekmovalni proces. Rezultati (srednja kvadratna napaka) takšne nevronske mreže so odlični.

Z evolucijskim računanjem sem se spoznaval na primerih. Za vsakega sem napisal več različnih rešitev in določal najboljšo. Največja problema sta bila v začetni inicializaciji populacije in pri ohranjanju raznolikosti posameznikov. Za ohranjanje raznolikosti sem uporabil turnirsko selekcijo. Za dobro začetno inicializacijo nisem ugotovil recepta.

Ideje sem črpal iz Darwinove knjige *"O nastanku vrst"*. Če posamezniki v boju za obstanek preživijo, pomeni da so naravno odbrani. Navadno so to boljši posamezniki (bolje prilagojeni razmeram, v katerih bivajo) ni pa nujno. Zato obstaja še dodaten pritisk spolne selekcije, ki odbira posameznike znotraj vrste na način, da najboljši ustvarijo več potomcev, najslabši pa manj ali nobenega. V moji simulaciji tako preživijo posamezniki, ki so bili dovolj uspešni pri iskanju hrane, a med njimi ustvarijo več potomcev tisti, ki so bili v tem osnovnem dejanju boljši. Slabši ustvarijo manj ali nič potomcev.

Darwin mehanizmov dedovanja in genskih mutacij ni poznal in je do teorije naravnega odbiranja prišel preko opazovanja živalskih vrst, zbiranja fosilnih ostankov in logičnega sklepanja. Z razumevanjem DNK vijačnice smo prišli do mehanizmov dedovanja, ki so še dodatna potrditev pravilnosti Darwinove hipoteze.

Pri mojem modelu evolucije nevronskih mrež je bistvena predstavitev genoma organizmov in informacije, ki jo posamezni geni kodirajo. Organizmu se ob rojstvu zgradijo "možgani" kodirani v genomu. Kriterij, ki sem ga zastavil, je dednost strukture nevronske mreže, medtem ko pridobljeno znanje staršev ni dedno. Na takšen način sem zasnoval genom, kjer vsak posamezen gen kodira tri različne tipe nevronov (vhodni, skriti, izhodni) in povezave med njimi. Vrednosti uteži ne sme predstavljati znanja staršev pridobljenega z učenjem, zato se dedujejo le začetne vrednosti. Od staršev organizem pridobi kombinacijo njihovih nevronskih mrež. Za drugačenje nevronske mreže organizma sem določil mutacije genov. Problem je, kako beležiti oziroma odkrivati mutacije, ki so se v preteklosti že pojavile. To je pomembno zaradi rekombinacije genov.

Uvedel sem identifikacijsko številko genov, ki označi, kdaj se je mutacija prvič pojavila in genski bazen vseh obstoječih mutacij, preko katerega lahko ugotovimo, če se je mutacija v preteklosti že pojavila. Organizem tako ne more imeti dveh genov za isto povezavo z

različnimi parametri (utežmi), ker bi bilo to nesmiselno. Pri križanju se potomcu dodeljujejo geni v naraščajočem zaporedju inovacijskih številk. Če sta inovacijski številki genov obeh staršev enaki, potomec naključno pridobi enega izmed genov (tako se prepreči podvojevanje genov).

Naloga je poskus izgradnje modela in simulacije umetnega življenja. Diplomsko delo mi je prineslo nova znanja ne le o računalniških področjih, ampak tudi boljšo predstavo o biologiji in evoluciji.

1.3. Napoved vsebine po poglavjih

V drugem poglavju opišem osnove umetnih nevronskih mrež, zgodovino razvoja in metodologijo delovanja z opisom algoritmov, ki se uporabljajo pri učenju nevronskih mrež. Podam dva tipična primera in rešitve s pomočjo nevronskih mrež.

V tretjem poglavju predstavim okolje in organizme. Organizmi so predstavljeni z nevronsko mrežo. Predstavim idejo učenja organizma s pomočjo učitelja in motivacijo za to predstavitev. Podane so različne konstrukcije nevronskih mrež.

V četrtem poglavju empirično analiziram obnašanje organizmov v več različno generiranih okoljih. Naredim tri različne simulacije, kjer spreminjam parametre hitrosti učenja, število organizmov, število enot hrane in porabo hrane za premikanje.

V petem poglavju opišem osnove evolucijskega računanja. Podrobneje opišem genetske algoritme.

V šestem poglavju primerjam učinkovitost različnih metod na dveh primerih.

V sedmem poglavju predstavim model evolucije nevronskih mrež, ga implementiram in nato naredim tri različne simulacije, ki jih vrednotim in primerjam rezultate.

V osmem sklepnem poglavju podam glavne ugotovitve diplomske naloge in ideje za izboljšave.

Poglavje 2

Nevronske mreže

Človek in računalnik sta sposobna doseči, česar nobeden od njiju sam ne zmore.

Hubert L. Dreyfus

V tem poglavju najprej opišemo zgodovinski razvoj nevronskih mrež, nato natančneje izbrano vrsto nevronske mreže za moj problem, podrobnejši opis algoritmov za učenje le-te in na koncu še dva primera uporabe podprta s primerjavo.

2.1. Zgodovina

Leta 1943 sta nevropsihologa Warren McCulloch in matematik Walter Pitts napisala članek v katerem sta predstavila hipotezo o delovanju nevronov. Modelirala sta en sam nevron, ki je predstavljal analogno digitalno transformacijo vhodov v izhode. Če je bila uteženа vrednost vhodov večja od 0 je nevron dal na izhod binarno 1 sicer pa 0. Matematično je bil model takole zapisan:

$$y = f\left(\sum_i x_i \cdot w_i\right) = \begin{cases} 1, & \check{c}e \quad \sum_i x_i \cdot w_i > 0 \\ 0, & sicer \end{cases}$$

Leta 1949 je Donald Hebb napisal članek o postopku učenja, kot utrjevanju sinaptičnih poti med nevroni vsakič, ko se prek njih pretaka informacija. ″Če se dva nevrona aktivirata v istem času, se povezava med njima utrdi″, je zapisal. Ta koncept je zelo pomemben pri razumevanju načina, kako se uči človek .

Ko so v petdesetih letih računalniki postali zmogljivejši, se je končno odprla možnost simulacije hipotetične nevronske mreže. Prvi, a na žalost neuspešen, korak v tej smeri je naredil inženir Nathanial Rochester v IBM-ovih raziskovalnih laboratorijih.

Leta 1959 sta znanstvenika Bernard Widrow and Marcian Hoff razvila modela imenovana ADELINE in MADELINE. Razvita sta bila za razpoznavanje binarnih vzorcev tako, da sta brala bitni tok iz telefonske linije in napovedovala vrednost naslednjega bita. MADALINE je bila prva nevronska mreža, ki se je uporabila v praktični aplikaciji in sicer kot filter za izločanje odmeva v telefonskih linijah. Sistem se, kljub starosti še vedno uporablja.

Leta 1962 sta ista znanstvenika razvila pravilo oz. kriterij učenja imenovan metoda najmanjših kvadratov ali delta pravilo. Še danes je to najbolj pogosto uporabljeno pravilo učenja. Na kratko deluje tako: za dani vhodni vektor se izračuna izhodni vektor, ki se primerja s pravilnim izhodnim vektorjem. Če razlike med tema vektorjema ni, pomeni da je bil odgovor nevronske mreže pravilen in mreža ostane enaka, sicer pa pride do sprememb uteži za zmanjšanje te razlike:

$$\Delta w_{ji} = \alpha \cdot (t_j - y_j) \cdot g'(h_j) \cdot x_i$$

Spremenljivke pomenijo:

- α: hitrost spreminjanja uteži,

- *g(x):* aktivacijska funkcija nevrona,
- t_j: želena izhodna vrednost nevrona,
- y_j: izhodna vrednost nevrona,
- h_j: utežena vsota vhodov v nevron,
- x_i: vhodni nevron.

Kasneje je razvoj nevronskih mrež za približno dvajset let zaspal in v računalništvu je prevladovala tradicionalna von Neumannova arhitektura. V tem času je tudi nastala raziskava, da iz enonivojske nevronske mreže ni možno razviti večnivojsko nevronsko mrežo. Še en problem, ki je pestril znanstvenike v tistem obdobju, je bila uporaba neodvedljive preklopne funkcije.

V poljudni znanosti so se rodile zanimive ideje o nevronskih mrežah in umetni inteligenci. Ideja, da bi računalnik samega sebe programiral je bila osupljiva. Mogoče je razmišljati o tem, da bi na primer Windows 95 reprogramiral samega sebe in tako odpravil vse napake in hrošče, ki so jih razvijalci spregledali. Nastale so knjige o strojih, ki razmišljajo, in o posledicah na človeštvo. Te ideje so zelo težke za realizacijo.

Leta 1972 sta Kohonen in Anderson neodvisno drug od drugega razvila na las podobni nevronski mreži. Za opis vseh operacij sta uporabila matrično algebro. Naredila sta množico analognih vezij ADALINE. Kot izhod nevronske mreže je bila aktivacija množice izhodov namesto samo enega kot pri ADALINE.

Prva večnivojska nevronska mreža je bila razvita leta 1975. Uporabljala je nenadzorovano učenje.

Leta 1982 je tematika nevronskih mrež dobila ponoven zagon, ko je John Hopfield objavil svojo raziskavo na Ameriški akademiji znanosti. Njegov pristop je bil uvedba dvosmernih povezav med nevroni namesto enosmernih kot poprej. Istega leta sta Reilly in Cooper razvila hibridno nevronsko mrežo, kjer je vsak nivo zase uporabljal različne strategije reševanja problemov. Na pobudo Japoncev je prišlo do mednarodne konference o nevronskih mrežah in umetni inteligenci, kjer so ZDA spoznale potrebo po pospešenem razvoju na tem področju.

Leta 1986 se je postavilo vprašanje kako posplošiti delta pravilo učenja na večnivojsko nevronsko mrežo. Tri neodvisne skupine raziskovalcev so se lotile vprašanja. Ena izmed teh z Davidom Rumelhartom je prišla do ideje, ki se danes imenuje vzvratno pravilo učenja. Mreže do takrat so uporabljale dva nivoja, nove nevronske mreže z vzvratnim pravilom učenja pa tri ali več nivojev (vhodni, izhodni in en ali več skritih nivojev). Zaradi večanja števila nivojev se še bolj povečuje število povezav med nevroni in zato je potrebno visoko število iteracij (na tisoče), da se nove nevronske mreže česa naučijo. Zato so to počasi učeče se nevronske mreže [13].

V današnjem času je razvoj na področju nevronskih mrež relativno počasen v primerjavi z ostalimi področji računalništva. Problem je v tem, da za kompleksne probleme v industriji nevronske mreže rabijo tudi nekaj tednov preden začnejo dajati dobre rezultate. Razvijalci skušajo razviti tako imenovane silikonske prevajalnike, ki bi ustvarili specifičen tip integriranega vezja optimiziran za posebne probleme. Hitrost učenja bi bila tako višja. Poleg tega se trudijo z izdelavo treh vrst nevro-čipov: digitalnih, analognih in optičnih. Če se bo z njimi res dalo izdelati nevronske mreže, se tehnologiji obeta svetla prihodnost. Kar se tiče uporabe nevronskih mrež v industriji smo v fazi razvoja.

2.2. Osnovni koncepti nevronskih mrež

V tem podpoglavju bom opisal, na kakšen način je kompleksna nevronska mreža sestavljena iz najosnovnejših elementov, kako so ti elementi povezani med seboj in matematično formulacijo vseh sklopov nevronske mreže.

2.2.1. Model nevrona

Model nevrona je zasnovan na biološkem nevronu tako, da povzame vse njegove glavne značilnosti, ki jih matematično opiše. Sestavljen je iz poljubnega števila vhodov, ki so povezani s seštevalnikom. Povezava je definirana z utežjo, ki je realno število med 0 in 1. Seštevalnik sešteje vrednosti vseh vhodov pomnoženih z utežmi. Ta vrednost se preslika v argument aktivacijske funkcije, ki določi izhodno vrednost nevrona na osnovi utežene vsote.

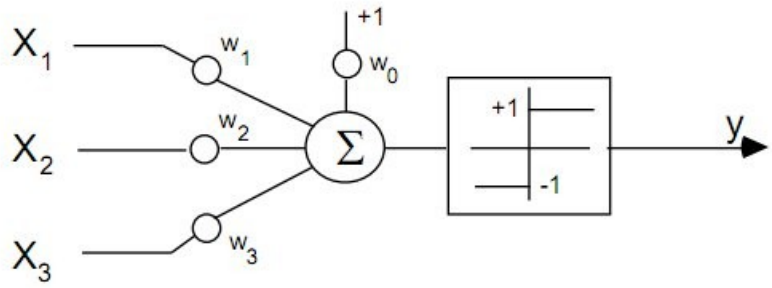
Zgornjo povezavo matematično zapišemo takole:

$$y = f\left(\sum_{i=1}^{n} x_i \cdot w_i\right)$$

Kjer posamezne spremenljivke pomenijo:

- *n:* velikost vhodnega vzorca in število uteži
- x_i: *:* vrednost vhodnega vzorca
- w_i *:* vrednost uteži
- *f:* aktivacijska funkcija

Na spodnji sliki je prikazan najstarejši model nevrona imenovan perceptron, ki uporablja digitalno aktivacijsko funkcijo.



Slika 1: Model umetnega nevrona.

2.2.2. Vrste preklopnih funkcij

Preklopna funkcija preslika uteženo vsoto v izhodno vrednost nevrona. Lahko si jo zamislimo kot odločitveno funkcijo. Obstoji nekaj vrst teh funkcij:

1) Digitalna ali stopničasta preklopna funkcija

   Ta funkcija priredi nevronu izhodno vrednost 1, če je utežena vsota večja od vrednosti imenovane prag, sicer pa 0. Izhod si lahko predstavljamo kot aktivno ali neaktivno stanje nevrona. Definicija funkcije:

$$f\left(\sum_i x_i \cdot w_i\right) = \begin{cases} 1, & \check{c}e \ \sum_i x_i \cdot w_i > 0 \\ 0, & sicer \end{cases}$$

2) Linearna preklopna funkcija

   Ta funkcija zagotavlja, da je vrednost oziroma jakost izhoda nevrona premo sorazmerna z uteženo vsoto vhoda nevrona. Definicija funkcije:

   $$f\left(\sum_i x_i \cdot w_i\right) = \alpha \cdot \sum_i x_i \cdot w_i$$

   Kjer je α razmerje med vhodom in izhodom.

3) Sigmoidna preklopna funkcija

   Je najpogosteje uporabljena funkcija zaradi zvezne odvedljivosti. Ta je pomembna pri delta pravilu učenja, ki ga bomo natančneje spoznali kasneje. Sigmoidna funkcija je po vrednostih med digitalno (stopničasto) in linearno preklopno funkcijo. Definicija funkcije:

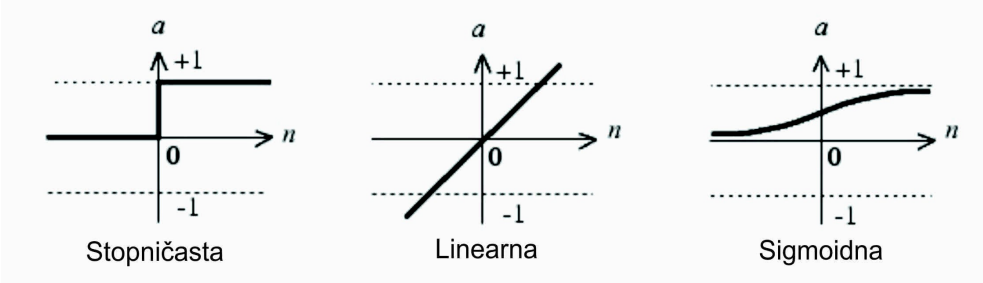
   $$f(\text{sum}) = \frac{1}{1+e^{-\text{sum}}}$$
   $$\text{sum} = \sum_i x_i \cdot w_i$$

   Matematična definicija izhaja iz logistične funkcije in ima obliko "S" krivulje.

4) Gaussova preklopna funkcija

   Spada v poseben razred aktivacijskih funkcij. Uporablja se pri RBF nevronskih mrežah.

Za boljšo predstavo so spodaj dani grafi vseh treh preklopnih funkcij.



Slika 2: Aktivacijske funkcije.

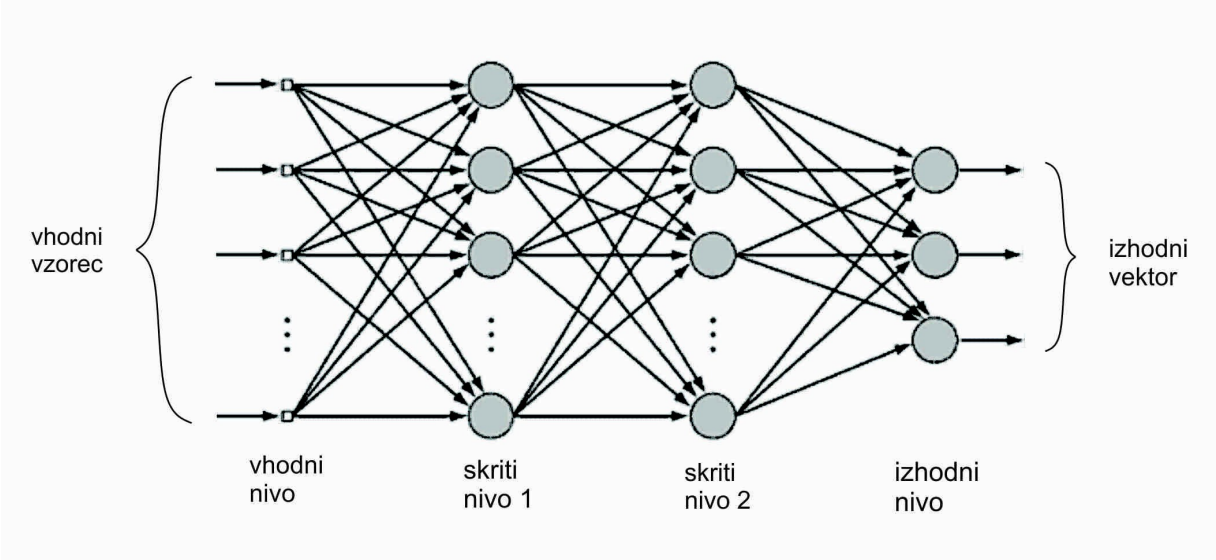
2.2.3. Topologija nevronskih mrež

Ko več nevronov povežemo dobimo nevronsko mrežo. Nevrone lahko združujemo tudi v skupine, znotraj katerih nevroni niso medsebojno povezani. Nivoje povezujemo skupaj na več načinov. Najpogosteje je vsak nevron iz nekega nivoja povezan z vsemi ostalimi nevroni iz sosednjega nivoja. Na tak način dobimo večnivojsko nevronsko mrežo. Struktura take mreže je:

- eden vhodni nivo nevronov,
- eden ali več skritih nivojev nevronov,

- eden izhodni nivo nevronov, kot to prikazuje slika 3.

Nivoji so lahko povezani v obe smeri ali pa samo naprej. Naprej povezana nevronska mreža je pogostejša. Ko se informacija pojavi na vhodu, se aktivirajo nevroni na vhodnem nivoju, kjer se utežena vsota za vsak nevron izračuna na osnovi vhodnega vektorja. Nato se aktivirajo nevroni na skritem nivoju, kjer se utežena vsota za vsak nevron izračuna na osnovi nevronov na vhodnem nivoju itd. Vsak nivo nevronov je odvisen samo od prejšnjega nivoja in neodvisen od naslednjega nivoja, saj informacija teče samo v smeri od vhoda proti izhodu nevronske mreže.



Slika 3: Večnivojski perceptron.

Večnivojski perceptron je potrebno matematično formulirati, če ga želimo uporabiti v računalniškem programu.

Vhod je predstavljen z vektorjem: $\boldsymbol{x} = \{x_1, x_2, .., x_n\}$, kjer je *n* velikost vhodnega vzorca. Izhod je predstavljen z drugim vektorjem: $\boldsymbol{y} = \{y_1, y_2, .., y_m\}$, kjer je *m* velikost izhodnega vektorja. Uteži so zapisane v dveh matrikah. Prva je velikosti *n* x *r*, kjer je *r* število nevronov skritega nivoja, druga pa velikosti *r* x *m*. Matriki izgledata takole:

$$W_1 = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,r} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,r} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n,1} & w_{n,2} & \cdots & w_{n,r} \end{bmatrix} \qquad W_2 = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,m} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ w_{r,1} & w_{r,2} & \cdots & w_{r,m} \end{bmatrix}$$

V prvi matriki vsaka vrstica predstavlja svoj nevron in vsak stolpec svoj element v vhodnem vektorju. V drugi matriki je ravno obratno: vrstica je izhod, stolpec pa skriti nevron.

Vektor nevronov skritega nivoja in izhodni vektor se z matričnim množenjem izračunata po obrazcu: $\boldsymbol{h} = W_1' \cdot \boldsymbol{x}$, $\boldsymbol{y} = W_2' \cdot \boldsymbol{h}$, kjer je $\boldsymbol{h}$ vektor skritega nivoja velikosti *r*.

2.2.4. Algoritmi za učenje

V prejšnjem podpoglavju smo ugotovili, da moramo za pravilno delovanje nevronske mreže poznati vrednosti uteži in aktivacijske funkcije vsakega nevrona. Aktivacijsko funkcijo izberemo glede na vrsto nevronske mreže in problema, ki ga rešuje. Določitev uteži nevronom je proces, ki ga imenujemo učenje nevronske mreže. To je postopek, ki ga večkrat ponavljamo, da delovanje nevronske mreže daje zadovoljive rezultate. V tem podpoglavju se bomo ukvarjali s problemom, kako izračunati uteži nevronske mreže, in kakšni so kriteriji, ki jih moramo izpolniti.

Poznamo tri principe učenja nevronske mreže, ti so nadzorovano, nenadzorovano in spodbujevalno učenje. Različni principi učenja so določeni glede na podatke, ki so na voljo.

2.2.4.1. Nadzorovano učenje

Pri tej vrsti učenja poznamo poleg vhodnih vektorjev še želene izhodne vektorje. Pripravimo množico vhodnih vektorjev in množico izhodnih vektorjev, ki predstavlja želeno delovanje nevronske mreže. Uniji obeh množic pravimo učna množica. Nevronsko mrežo učimo tako, da ji za vsak vhodni vektor podamo še želeni izhodni vektor. Mreža z uporabo algoritma za učenje spremeni uteži tako, da se odgovor nevronske mreže približa želenemu delovanju.

Poglejmo kriterij minimalne napake in postopek za izračun uteži nevronske mreže, ki je iz tega kriterija izpeljan.

Kriterijska funkcija za učenje nevronske mreže je običajno podana takole:

$$J=\frac{1}{2}\cdot E[e^2]=\frac{1}{2}\cdot E[(d-y)^2]$$

Pomen spremenljivk je:

- *E:* operator za matematično upanje,
- *e:* napaka na izhodu nevronske mreže,
- *y:* izhod nevronske mreže,
- *d:* želeni izhod nevronske mreže.

Želimo si, da bi bil izhodni vektor čim bolj podoben želenemu izhodnemu vektorju. Kriterijska funkcija je povprečna napaka nevronske mreže, ki jo želimo minimizirati. Cilj je iskanje uteži nevronske mreže, za katere bo vrednost razlike najmanjša. Matematično kriterij tako zapišemo:

$$\frac{\partial J}{\partial w_i}=0$$

Metoda za izračun uteži ob upoštevanju gornjega kriterija se imenuje metoda najmanjših kvadratov. Izračun natančnih vrednosti uteži po tej metodi je kompleksen, zato uporabimo približek. Nova vrednost uteži se izračuna kot:

$$\hat{w}_i(n+1)=\hat{w}_i(n)+\eta\cdot[d(n)-y(n)]\cdot x_i(n),\quad i=1,..,m$$

To formulo imenujemo delta pravilo in je osnova za številna druga pravila učenja.

Spremenljivke pomenijo:

- *n:* diskreten čas,
- $\hat{w}_i(n)$: ocena uteži v času n,

- η: učni parameter, ki opisuje hitrost spreminjanja uteži.

Najpogosteje je cilj učenja odkrivanje splošne relacije med vhodi in izhodi. Nevronsko mrežo naučimo s poznanimi vhodno-izhodnimi pari vektorjev. Ko je postopek učenja končan, pričakujemo, da je nevronska mreža osvojila funkcijo, ki preslika vsak vhod v pravilen izhod (seveda z odstopanjem).

2.2.4.2. Nenadzorovano učenje

Pri nenadzorovanem učenju mreži podamo samo vhodne podatke brez želenih rezultatov. Tako mora mreža sama poiskati značilnosti, po katerih lahko loči vhode med seboj. Mrežam, ki uporabljajo takšen postopek učenja, pravimo samoorganizirajoče mreže. Ta vrsta učenja se naprej deli na tri veje:

1) Tekmovalni proces

   Za vsak vhodni vektor nevroni v mreži izračunajo vrednosti ustreznih diskriminantnih funkcij, ki predstavljajo osnovo za tekmovanje med nevroni. Nevron z največjo vrednostjo diskriminantne funkcije je zmagovalec tekmovanja, ki diktira ureditev nevronske mreže npr. postane edini nevron, ki se aktivira.

2) Kooperativni proces

   Zmagovalni nevron določa center topološke sosednosti sodelujočih nevronov. Pri tem topološko sosednost določajo povezave bližnjih nevronov v okolici zmagovitega nevrona.

3) Adaptivni proces

   Skladno z vhodnim vektorjem se spreminjajo uteži zmagovalnega nevrona in sosednjih nevronov. Izhaja iz Hebbovega učenja, ki ga je potrebno prirediti, saj se pri samoorganizaciji nevronske mreže uteži spreminjajo samo v eni smeri. Ta težava se reši z vpeljavo novega člena pozabljivosti. Adaptivni proces je razdeljen v dve fazi – faza urejanja in faza konvergiranja. V fazi urejanja nevronska mreža dobi grobo topološko urejenost, v fazi konvergiranja pa natančen končni izgled.

Obstoji še Hebbovo učenje, kjer nevroni črpajo vso informacijo od svojih sosedov, zato rečemo, da je to učenje lokalno. Bistvo učenja je, da se utež med nevronoma poveča, če sta oba aktivna. Ker pa to pomeni, da gredo določene uteži v neskončnost, je Oja predlagal izboljšavo, kjer se uteži v vsakem koraku normalizirajo. Matematični zapis učenja brez normalizacije uteži:

$w(n+1)=w(n)+\rho \cdot y(t) \cdot \boldsymbol{x}(t)$, kjer je *y(t)* utežena vsota vhodov in ρ realno število. Bistvo tega učenja je korelacija med vhodnim in izhodnim vektorjem pomnožena z nekim faktorjem.

2.3.4.3. Spodbujevalno učenje

Tukaj nevronska mreža dobiva pozitivne ali negativne dražljaje iz okolja, ki vzpodbujajo ali zavirajo neko vrsto aktivnosti nevronske mreže. Za učenje živali je ta postopek zelo uporabljen npr. nagrada za dobro obnašanje psa in kazen za neubogljivost. Ta vrsta učenja je dober model učenja v realnem okolju. Pogosta implementacija je v sklopu z algoritmom učečih se avtomatov.

2.2.5. Vrste nevronskih mrež

Nevronske mreže se med seboj zelo razlikujejo. Različnost se pozna po strukturi, velikosti, številu nivojev, aktivacijski funkciji, pravilu učenja itd. Dejansko je edina podobnost med različnimi vrstami model nevrona s povezavami. Za približno predstavo obsežnega področja nevronskih mrež bom orisal nekaj najbolj značilnih vrst. Poglobil se bom samo v nevronsko mrežo, ki jo v diplomski nalogi uporabim.

1) Perceptron

   Je najenostavnejša nevronska mreža, ki je sestavljena le iz enega nevrona in poljubnega števila vhodov. Opisana je bila že v poglavju 2.2.1.

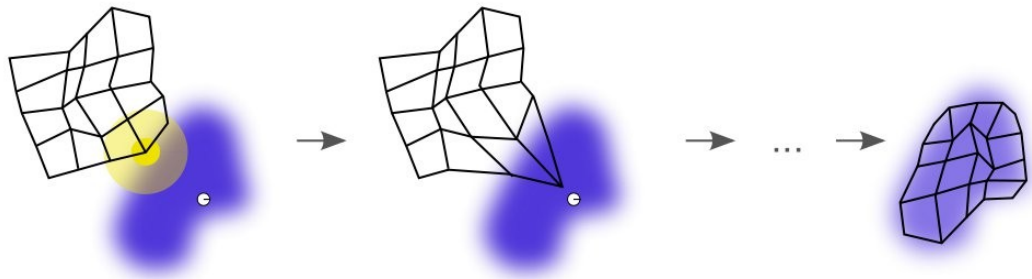
2) Večnivojski perceptron z vzvratnim pravilom učenja

   Ta nevronska mreža je najpogosteje uporabljeni primer naprej povezane nevronske mreže. Vsebuje tri ali več plasti nevronov. Za izračun uteži uporablja vzvratno pravilo učenja, ki je izpeljano iz delta pravila, ki temelji na metodi najmanjših kvadratov. To nevronsko mrežo sem v diplomski nalogi uporabil skupaj z rekurzivno nevronsko mrežo in jo bom zato podrobno opisal v razdelku 2.2.6.

3) Kohonenova samo-organizacijska nevronska mreža ali SOM (angl. Self Organizing Map)

   Ta nevronska mreža je nenadzorovana, torej ne dobiva povratnih informacij iz okolja. Pogoj za želeno delovanje mreže je prilagoditev stanja oz. uteži na vhodni prostor. Mreža deluje tako, da na osnovi vhodnega vektorja in uteži nevronov po tekmovalnem procesu aktivira t.i. zmagovalni nevron. Tekmovalni proces je računanje skalarnega produkta uteži posameznega nevrona z vhodnim vektorjem. Ta je največji, ko so komponente vhodnega vektorja enake utežem. Nevron z največjo vrednostjo skalarnega produkta je zmagovalen, saj so njegove uteži najbolj prilagojene vhodnemu vektorju. Zmagovalni nevron predstavlja grozd oziroma skupino podobnih vektorjev.

   Ta nevronska mreža je naučena na preslikuje točke iz vhodnega prostora v koordinate izhodnega prostora. Vhodni prostor ima lahko različne dimenzije in topologijo od izhodnega prostora. Mreža pri preslikavi vhodnega v izhodni prostor ohranja njuni topološki lastnosti, med katerimi je najpomembnejša ohranjanje sosednosti.



Slika 4: Ilustracija učenja SOM nevronske mreže.

Na zgornji sliki je prikazan postopek učenja-prilagoditve uteži mreže na vhodni prostor. Modri oblaček predstavlja porazdelitev vhodnih podatkov učne množice, majhen beli krogec je pa trenutni vhodni vektor. Na začetku so vozlišča (uteži) nevronske mreže naključno razporejena po podatkovnem prostoru. Rumeno označeno

vozlišče, ki je najbližje vhodnemu vektorju je izbrano in se po nekaj iteracijah učenja premakne v bližino vhodnega vektorja, ravno tako se delno premaknejo tudi vozlišča v njegovi bližini. Po mnogih iteracijah z uporabo celotne učne množice se mreža vozlišč prilagodi vhodnemu prostoru. Mreža je tako naučena, saj najbolje aproksimira vhodne podatke.

4) Rekurzivna nevronska mreža

Rekurzivna nevronska mreža vsebuje tudi povratne povezave izhodov iz nevronov, kar omogoča mreži hranjenje informacije o starem stanju oziroma z drugimi besedami, daje ji kratkoročen spomin. Zanimiv primer te nevronske mreže je Hopfieldova nevronska mreža, ki zelo dobro modelira biološke možgane.

2.2.6. Večnivojski perceptron z vzvratnim pravilom učenja

Za to vrsto nevronskih mrež ne velja več omejitev enonivojskega perceptrona, saj je dokazano, da je možno tri ali večnivojsko mrežo naučiti katerekoli logične funkcije (seveda tudi ostale funkcije). Posvetimo se konceptu učenja te nevronske mreže.

Na vhod nevronske mreže postavimo vhodni vektor. Na osnovi vhodnega vektorja in uteži mreža izračuna izhod. Izhodu mreže podamo vektor, ki predstavlja želeni izhod. Mreža izračuna razliko med izhodnim vektorjem in želenim izhodnim vektorjem in popravi uteži na izhodnem nivoju. Potem mreža na skritem nivoju popravi uteži upoštevajoč nove uteži izhodnega nivoja. Algoritem prodira iz izhoda proti vhodu strani in sproti popravlja uteži, zato mu pravimo vzvratni algoritem.

S podobno matematično izpeljavo, kot je metoda najmanjših kvadratov pridemo do enačbe za spremembo uteži nevronov skrite plasti pred izhodno plastjo:

$\Delta w_{ji}(n) = \eta \cdot \delta_j(n) \cdot y_i(n)$, kjer je sedaj funkcija delta vsota preko vseh izhodnih nevronov:

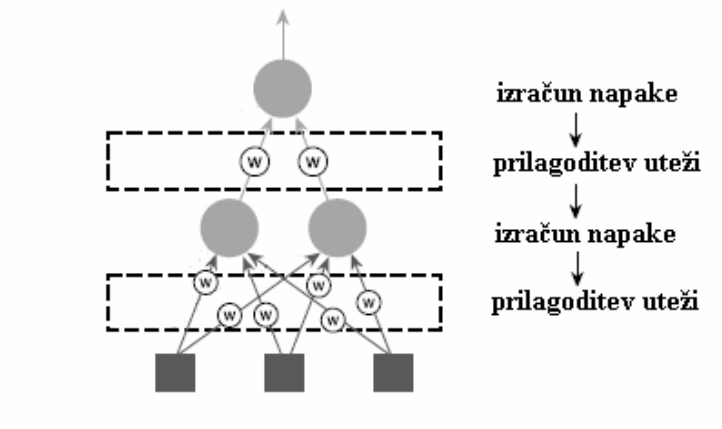
$\delta_j(n) = f'(v_j(n)) \cdot (\sum_k \delta_k(n) \cdot w_{kj}(n))$ in odvod aktivacijske funkcije enak:

$f'(v_j(n)) = y_j(n) \cdot (1 - y_j(n))$.

V primeru, da je več skritih plasti, se spreminjanje uteži njihovih nevronov izračuna na enak način kot zgoraj, le da namesto izhodnih nevronov nastopajo nevroni predhodne skrite plasti. Za lažjo predstavo bom podal psevdo kodo vzvratnega učilnega algoritma za trinivojsko nevronsko mrežo:

1. Naključno inicializiraj uteži nevronske mreže
2. Delaj:
3. Za vsak vhodni vektor iz učne množice:
4. Izračunaj izhodni vektor nevronske mreže
5. Izračunaj razliko med želenim in dejanskim izhodnim vektorjem
6. Izračunaj delta_wh za vse uteži skritega nivoja proti izhodnemu nivoju
7. Izračunaj delta_wi za vse uteži vhodnega nivoja proti skritemu nivoju
8. Uveljavi nove vrednosti uteži nevronske mreže na osnovi izračunanih delt
9. Dokler ni vsak vhodni vektor pravilno klasificiran-razlika med želenim in dejanskim izhodom ni ....minimalna ali je dosežen kakšen drugi kriterij (dovolj veliko število iteracij)
10. Vrni naučeno nevronsko mrežo

Na sliki 5 ilustriram vzrvatno učenje na primeru trinivojske nevronske mreže s šestimi nevroni.



Slika 5: Ilustracija vzvratnega učenja.

2.3. Uporaba nevronskih mrež na praktičnih aplikacijah

2.3.1. Primer učenja vrednosti funkcije

Nevronske mreže z vzvratnim pravilom učenja se lahko uporabljajo za aproksimacijo funkcij, za katere poznamo vrednosti le v določenem številu točk. Te točke predstavljajo učno množico. Predstavljajmo si, da imamo sistem, za katerega želimo ugotoviti zakonitosti. Vhod v sistem je koordinata x, izhod pa koordinata y. Podano imamo končno množico parov (x, y). V našem primeru generiramo to množico s funkcijo:

$y = f(x) = 0{,}7 \cdot x^2 - 1{,}5 \cdot \sin(2 \cdot x)$ na intervalu $x = [-\pi, \pi]$.

Ker ne poznamo oblike funkcije, nas ocenjevanje parametrov funkcije ne pripelje nikamor. Zato prevedemo problem na iskanje uteži nevronske mreže, ki bo za vsak x iz intervala $[-\pi, \pi]$ dala na izhodu vrednost, ki bo približno enaka pravi vrednosti. Torej tudi za vhode, ki niso bili podani v učni množici. Na takšen način poiščemo preslikavo $x \rightarrow y$, ki je namesto s funkcijskim predpisom predstavljena z utežmi nevronske mreže in njenim delovanjem.

Nevronska mreža je sestavljena iz enega vhodnega, enega izhodnega ter *n* skritih nevronov. Kot zanimivost bom prikazal uspešnost delovanja s povprečno vrednostjo kvadratov napake ob spreminjanju naslednjih parametrov:

- število skritih nevronov,
- število iteracij vzvratnega algoritma,
- velikost učne množice.

Za oceno točnosti delovanja izračunamo povprečje kvadratnih razlik med želeno in izhodno vrednostjo nevronske mreže. Matematično je metrika definirana:

$$MSE = \sqrt{\frac{1}{N} \cdot \sum_{i=1}^{N} (f(x_i) - \tilde{f}(x_i))^2}$$

Kjer pomeni:

- *N:* število točk oz. velikost testne množice,
- $f(x_i)$: prava vrednost funkcije,
- $\tilde{f}(x_i)$: približna vrednost funkcije kot izhod nevronske mreže.

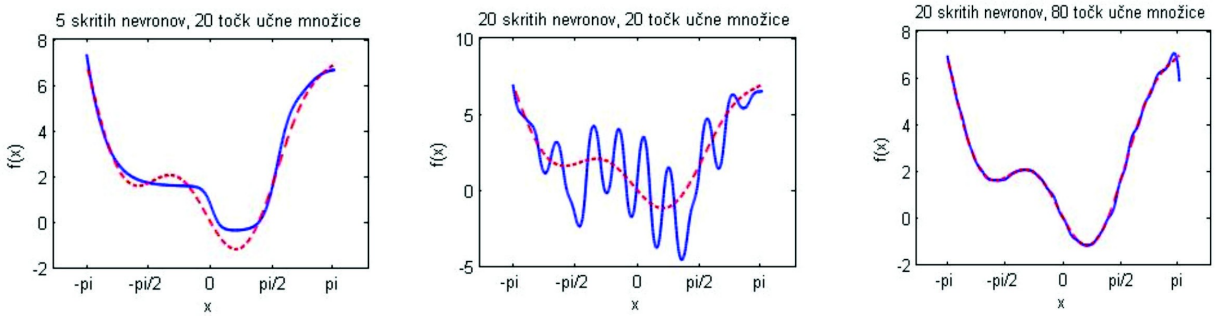
Oglejmo si, kako se funkcija napake obnaša ob spremembah zgoraj naštetih parametrov. Generiral bom več različnih nevronskih mrež in več različnih učnih množic ter eno testno množico, s katero bom preizkusil delovanje različnih nevronskih mrež.

Spreminjanje števila skritih nevronov

Z večanjem števila skritih nevronov kvadratna napaka izračunana na učni množici upada. Nevronska mreža si bolje zapomni, česa je bila naučena. Z večanjem števila skritih nevronov kvadratna napaka na testni množici upada do prelomne točke, kjer doseže najmanjšo vrednost, nato pa začne naraščati. To se zgodi zato, ker se mreža preveč prilagodi učnim podatkom.

To lahko vidimo na sliki 6. V prvem primeru je število skritih nevronov in velikost učne množice majhna, zato nevronska mreža dokaj dobro aproksimira funkcijo. V drugem primeru

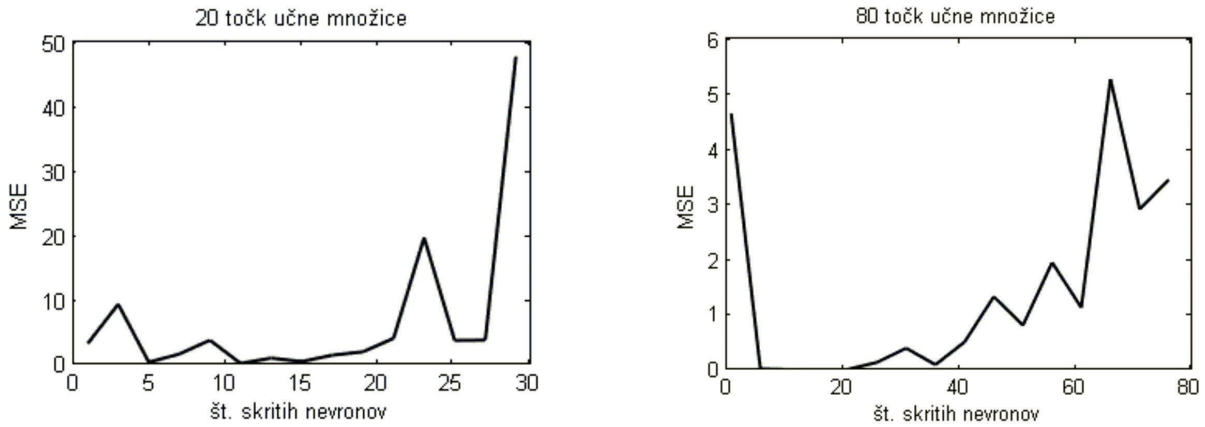
povečamo število skritih nevronov ob nespremenjeni velikosti učne množice in opazimo pretirano prilagoditev. V tretjem primeru povečamo velikost učne množice in delovanje mreže postane še boljše kot v prvem primeru. Mreža z večjim številom skritih nevronov potrebuje za učenje večjo učno množico, saj se sicer pretirano prilagodi učni množici in ne posploši naučenega.



Slika 6: Aproksimacija funkcije z nevronsko mrežo. Prava funkcija je črtkana rdeče barve, aproksimirana funkcija pa modre barve.

Ustvarim 15 nevronskih mrež s številom nevronov med 1 in 29 v prvem primeru ter med 1 in 78 v drugem primeru. V prvem primeru bo velikost učne množice 20 v drugem pa 80 vzorcev. Za oba primera sem izračunal srednjo kvadratno napako za vseh 15 nevronskih mrež in rezultate prikazal na sliki 7. Vidimo, da na začetku pri relativno majhnem številu skritih nevronov kvadratna napaka pada, nato pa začne z večanjem števila skritih nevronov naraščati z nihanji. Ta nihanja nastopijo zato, ker vsakič, ko dodam nove skrite nevrone nevronski mreži spremenim strukturo in moram na novo inicializirati začetne uteži. Delovanje mreže je do neke mere odvisno tudi od začetne vrednosti uteži kar analiziram v razdelku 3.4.

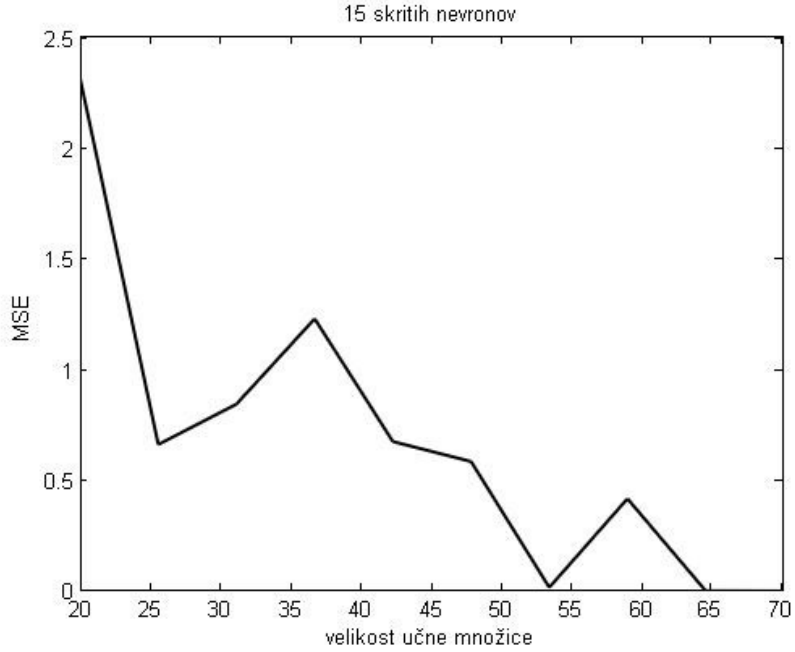
Na grafih vidimo, da je kvadratna napaka v drugem primeru precej manjša kot v prvem primeru. Opazimo, da se začne strmo naraščanje kvadratne napake v prvem primeru že pri 20 skritih nevronih, v drugem primeru pa pri 37.



Slika 7: Funkcija napake nevronske mreže v odvisnosti od števila skritih nevronov.

Spreminjanje velikosti učne množice

Poglejmo si še spreminjanje kvadratne napake ob različno velikih učnih množicah. Na spodnji sliki lahko vidimo padanje kvadratne napake z večanjem učne množice. Do nihanj pride zaradi razlike v boljše ali slabše reprezentativnosti učne množice. Točke so za vsako učno množico postavljene drugače na intervalu funkcije, ki jo aproksimiramo. V manjši učni množici je lahko več točk postavljenih na predelih, kjer se funkcija močno spreminja, kot v večji učni množici. Spreminjanje kvadratne napake je prikazana na sliki 8.



Slika 8: Funkcija napake nevronske mreže v odvisnosti od velikosti učne množice.

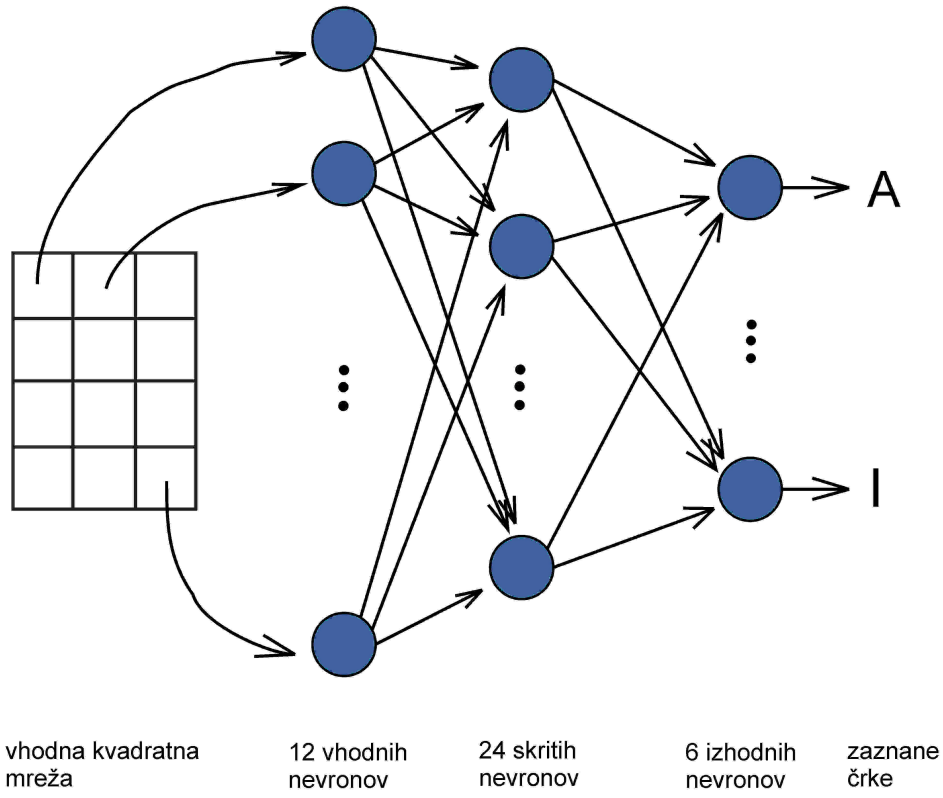
2.3.2. Primer razpoznavanja črk

Zelo pogost problem v realnem življenju je razpoznavanje vzorcev. Recimo, da želimo razpoznati črke angleške abecede. Sestavljena je iz 26 različnih črk. Cilj razpoznave vzorca je za vsak ročno zapisan vzorec ugotoviti, katero črko je pisec imel v mislih. Vsak človek piše nekoliko drugače. Manjše razlike v zapisu iste črke pri različnih ljudeh si lahko predstavljamo kot šum. Za sestavo učne množice potrebujemo sto ali več ljudi ter njihov zapis abecede. Ko nevronsko mrežo naučimo s pomočjo učne množice, postane ta sposobna razpoznati črke različnih pisav, ki jih do tedaj še ni srečala. V tem primeru je potrebna robustnost na šum, po kateri so nevronske mreže v splošnem znane.

Problem bom poenostavil in učil nevronsko mrežo samo za en način pisave in se tako izognil šumu.

Abeceda je sestavljena iz šestih različnih črk $X = \{A, C, D, F, H, I\}$. Črke so zapisane na kvadratni mreži velikosti 4x3. V vsakem kvadratu je zapisana 0 ali 1 (pisava odsotna ali prisotna), zato je število vseh različnih vrednosti kvadratne mreže enako $2^{12} = 4096$. Nevronska mreža je na vhodni strani sestavljena iz dvanajstih nevronov. Vsak zaznava vrednost enega kvadrata vhodne mreže. Na izhodni strani ima nevronska mreža šest nevronov, kjer aktiven nevron ponazarja črko glede na svojo pozicijo npr. izhodni nevron številka 5 je aktiven, ko mreža na vhodu zazna črko H, ostali pa so neaktivni.

Zaradi poenastavitve problema mreža deluje brez napake. Bolj zanimiva pa je topologija nevronske mreže, ki je na sliki 9.



Slika 9: Topologija nevronske mreže za razpoznavo vzorcev.

Poglavje 3

Nevronska mreža za umetne mikroorganizme

V tem poglavju predstavim idejo simuliranega organizma in več različnih rešitev z nevronsko mrežo. S primerjavami rešitev argumentiramo izbiro za implementacijo. Demonstriram delovanje izbrane rešitve v več različnih umetno postavljenih okoljih, prikažem uspešnost delovanja in statistično analiziram rezultate.

3.1. Umetni mikroorganizmi

Raziskujem model obnašanja mikroorganizmov v okolju. Vsak organizem je predstavljen z lastno nevronsko mrežo. Organizem lahko vizualno zaznava lokalno okolje ter zaznava neposreden dotik, s predmeti v okolju. Ko organizem sprejme podatke iz okolja se odloči, kam se bo premaknil. Odločitev je izhod nevronske mreže organizma.

Organizem potrebuje za svoj obstoj hrano. Ob konzumaciji hrane dobi potrebno energijo, ki se s časom troši. Potrošnja kalorij je definirana kot linearna časovna funkcija. Če organizem konzumira novo enoto hrane v času, ko ima še kaj energije, nadaljuje z življenskim ciklom, sicer umre.

Okolje v katerem se organizmi gibljejo je zagrajeno s stenami, v njem je hrana naključno postavljena. Energijska vrednost je enaka za vse enote hrane. Ko organizem s svojim telesom prekrije več kot polovico površine hrane, jo konzumira in hrana se odstrani iz okolja. Ko v okolju začne primanjkovati hrane, se naključno postavi nova hrana.

Cilj vsakega organizma je preživetje, za kar mora ves čas imeti dovolj energije za življenje, zato mora ves čas iskati hrano. Organizem se ob stvaritvi ne ve kaj mora početi in se zato naključno giblje v okolju. Ko poje hrano, občuti zadovoljstvo in počasi ugotovi, da je tisto, kar mora početi, iskanje hrane. Ugotovi, da je najboljši način za preživetje, potovanje v smeri proti najbližji hrani. Organizmi, ki so to strategijo osvojili, so se prilagodili okolju in so po metodi naravne selekcije izbrani za prenos svojega znanja na potomce. Ko se čas ene generacije konča, preživeli organizmi s križanjem in mutacijo prenesejo svoj genski material na potomce in začne se življenje nove generacije. Tisti, ki se niso naučili učinkovitega iskanja hrane, ne preživijo. V novi generaciji se učenje začne znova, le da je genska osnova drugačna kot v prejšnji generaciji. Ker se z vsako generacijo obdržijo dobre lastnosti organizmov, pričakujemo, da bodo sčasoma organizmi napredovali.

3.2. Predstavitev možnih arhitektur nevronskih mrež

Vektor hitrosti organizma je definiran s polarnimi koordinatami. Prva komponenta je magnituda oz. hitrost, druga pa kot premika. Hitrost je dana ob stvaritvi organizma in se s časom ne spreminja. Zato je edini izhod nevronske mreže kot premika. Nevronska mreža ima na vhodnem nivoju nevrone za zaznavo okolja, na izhodnem nivoju pa nevron, ki predstavlja kot premika.

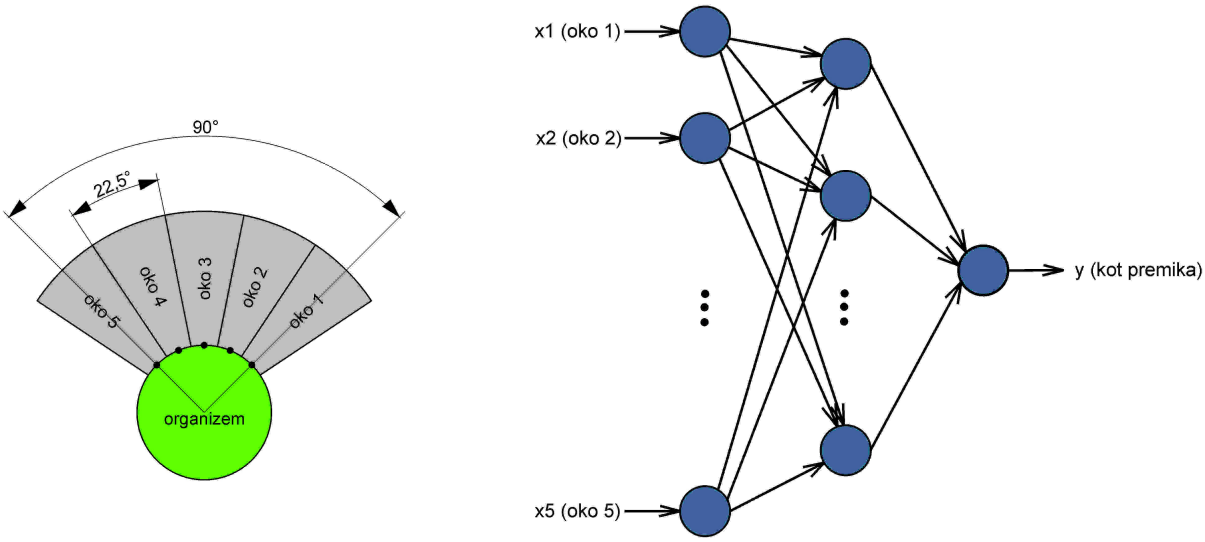
V tem podpoglavju bom opisal štiri načine zaznave okolja. Vsak način zaznave okolja spremlja drugačna senzorika, ki prinaša drugačno topologijo nevronske mreže. Opisal bom tudi različne topologije.

3.2.1. Topologija 1

Organizem ima pet oči postavljenih na različnih kotih. Kot med prvim in zadnjim očesom je 90°, med sosednjimi očmi pa 22.5°. Vsako oko zazna predmet, ki je najbližji organizmu in je kot predmeta v vidnem območju tega očesa, to je 22.5°. Organizem ima omejeno največjo možno razdaljo do predmeta, ki ga je še sposoben zaznati. Oči organizma zaznajo odbito svetlobo predmeta. Če je ta na največji razdalji, je zaznana odbita svetloba najmanjša. Z bližino se delež zaznane svetlobe povečuje. Ko je predmet tik ob očesu, je zaznana vsa odbita svetloba.

Vhod v mrežo je 5 dimenzionalni vektor, kjer vsaka komponenta ponazarja jakost odbite svetlobe za vsako oko. To je realno število, ki je za vso odbito svetlobo enako 1, za nič odbite svetlobe enako 0, ostale vrednosti pa so med 0 in 1.

Izhod nevronske mreže je kot najbližjega predmeta kot to prikazuje slika 10.



Slika 10: Senzorika organizma in topologija njegove nevronske mreže.

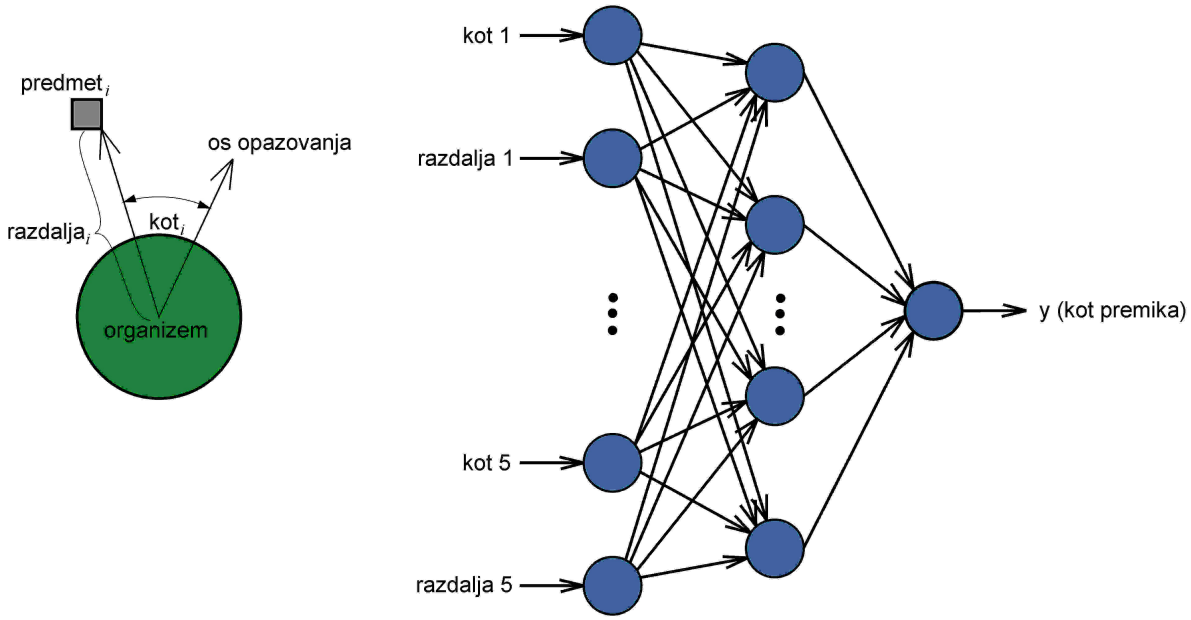
3.2.2. Topologija 2

Organizem ima samo eno oko, ki opazuje celotno sliko okoli njega do neke razdalje. Oko opazi 5 najbližjih predmetov, če so v njegovem dometu.

Vhodni vektor je sestavljen iz 10 komponent. Prva komponenta pomeni kot najbližjega predmeta, ki ga je organizem zaznal in druga pomeni razdaljo do tega predmeta. Tretja komponenta je kot drugega najbližjega predmeta in četrta razdaljo itd. Vhodni sloj je sestavljen iz 10 nevronov, kjer je vsak nevron odgovoren za svojo komponento vektorja.

Od nevronske mreže pričakujemo, da bo na izhod preslikala kot najbližjega predmeta.

Topologija nevronske mreže in organizem sta ilustrirana na sliki 11.

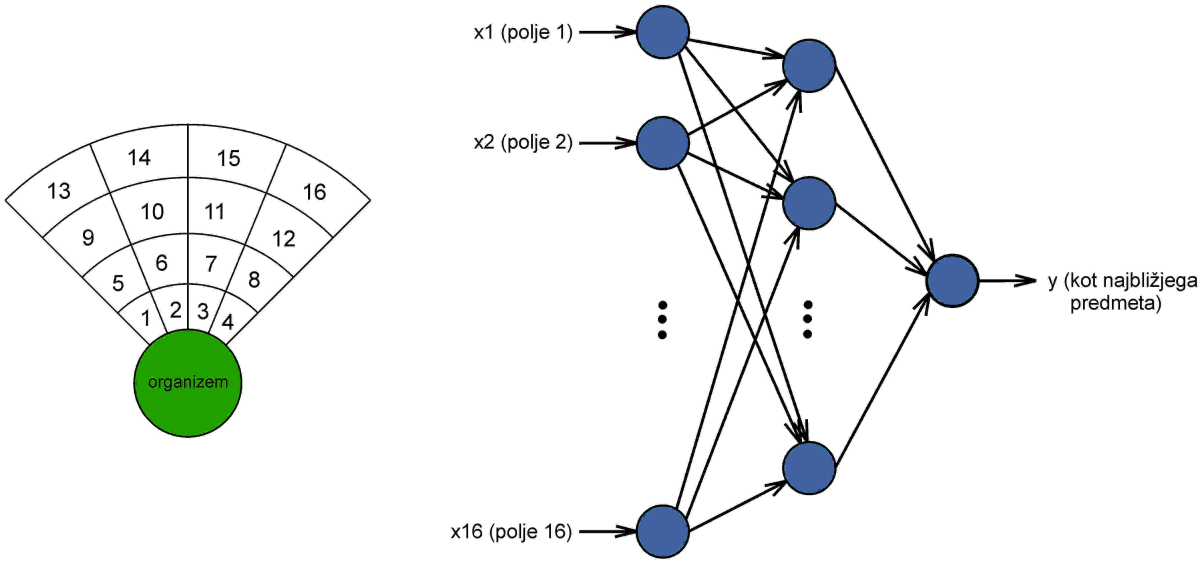


Slika 11: Organizem in topologija njegove nevronske mreže.

3.2.3. Topologija 3

Vidno polje organizma razdelimo v mrežo kvadratov enake dolžine. Kvadratna mreža je sestavljena iz 4 kvadratov v horizontalni smeri in 4 kvadratov v vertikalni smeri. Če se predmet nahaja znotraj kvadrata, je v vhodnem vektorju zapisana 1 sicer pa 0. Kvadrate oštevilčimo od 1 do 16. Dimenzija vhodnega vektorja je 16, kot to prikazuje slika 12.

Na vhodnem nivoju je 16 nevronov, vsak zaznava vrednost v enem izmed kvadratov. Želeni izhod nevronske mreže je izbira kota za najbližji predmet.



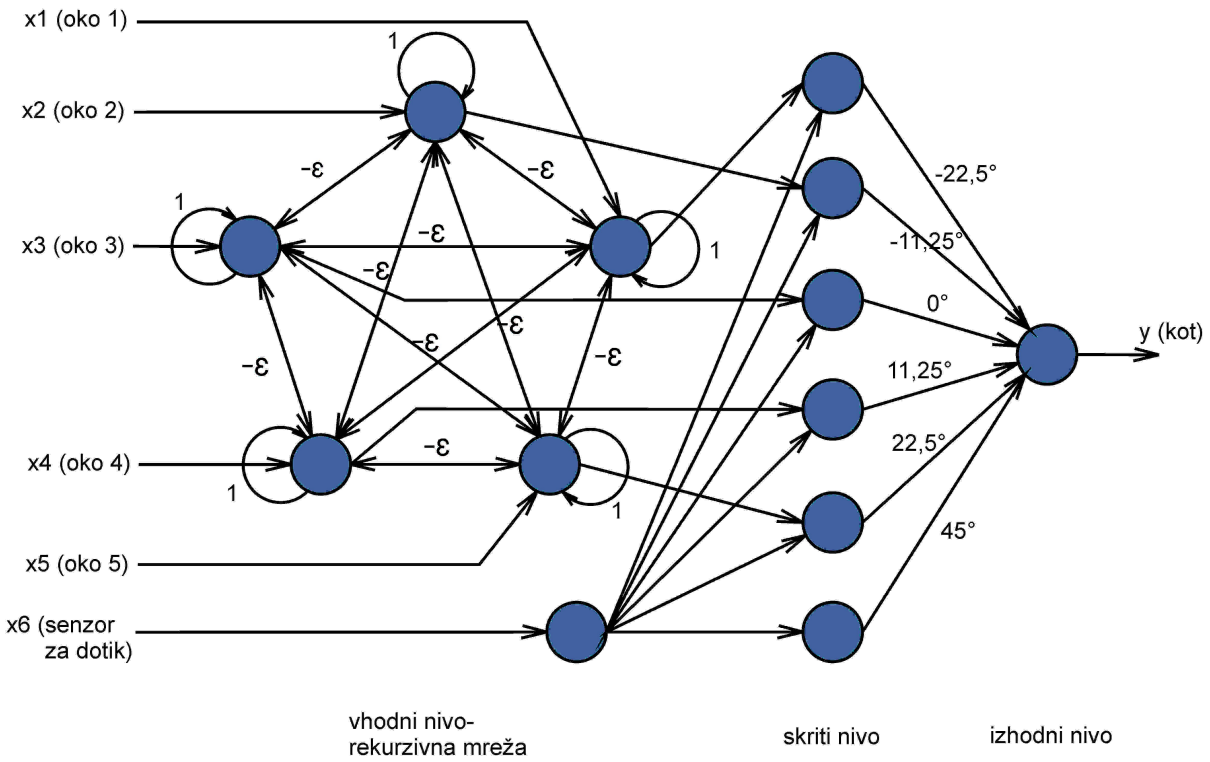
Slika 12: Vidno polje organizma in topologija njegove nevronske mreže za tretji način predstavitve.

3.2.4. Topologija 4

Percepcija organizma je enaka kot pri organizmu 1, topologija nevronske mreže pa je kompleksnejša. Na vhodnem nivoju imamo 5 nevronov, ki so medsebojno povezani vsak z vsakim. Ta del nevronske mreže je rekurziven. Vse uteži v tem delu so majhna negativna realna števila, povezava nevrona samega nase pa ima utež enako 1. Mreža deluje na tekmovalen način, kjer vsak nevron prispeva negativni del ostalim nevronom. Nevron z najvišjo začetno vrednostjo po koncu tekmovalnega procesa obdrži vrednost nad 0 vsi ostali pa so enaki 0. Ta nevron je zmagovalni nevron.

Ta del nevronske mreže določi lokacijo največjega nevrona, ki predstavlja lokacijo najbližjega predmeta organizmu. Ti nevroni so povezani na skriti nivo s 6 skritimi nevroni. Nevroni v vhodni plasti so povezani z nevroni v izhodni plasti. Skrita plast ugotovi položaj nevrona, kjer je bil zaznan najbližji predmet in potem na izhodni nevron preslika kot, ki temu položaju ustreza. Poleg tega nevronska mreža vsebuje še nevron za detekcijo trka s predmeti ali stenami. Zgoraj opisano je primer idealno naučene nevronske mreže. Za lažjo predstavo je primer prikazan na sliki 13.

Ko organizem zazna trk s steno, predmetom ali sosednjim organizmom, bi se moral zarotirati za kot 45°. To je pričakovan izhod nevronske mreže ob detekciji trka, ne glede na to, ali je v istem trenutku organizem vidno zaznal še kakšen predmet. Zato so povezave vhodnega nevrona za trke na skrite nevrone inhibirne oz. imajo negativne uteži. Tako bo utežena vsota vhodov negativna in se skriti nevroni ne bodo aktivirali.



Slika 13: Topologija kombinirane rekurzivne in naprej povezane nevronske mreže. ε je majhna vrednost običajno 0,1.

3.3. Simulacija predstavljenih topologij v okolju in primerjava rezultatov

Za opisane topologije nevronskih mrež je zaželeno delovanje izračun kota do najbližjega predmeta. Opisane topologije se med seboj precej razlikujejo. Za izbiro najboljše je najpomembnejši dejavnik uspešnost delovanja v okolju s predmeti (hrano). Za primerjavo topologij med seboj bom nevronske mreže naučil na množici velikosti 500 vzorcev in nato preveril delovanje v testnem okolju s 100 vzorci. Za primerjavo bom vzel srednjo kvadratno napako. Tako bom ovrednotil izbiro najprimernejše oz. najboljše nevronske mreže. Ker je delovanje odvisno od naključne postavitve uteži ob kreiranju nevronske mreže, bom izračunal povprečje kvadratnih napak za 15 primerov. Rezultati so v tabeli spodaj.

Topologija	Srednja kvadratna napaka
1	287,97
2	402,39
3	229,66
4	4,25

Tabela 1: Povprečje srednjih kvadratnih napak na petnajstih različnih nevronskih mrežah.

Vidimo, da je srednja kvadratna napaka vseh topologij razen četrte ogromna. Njihovo delovanje je primerljivo z naključnim delovanjem. Nevronsko mrežo je težko naučiti iskanja najmanjše ali največje komponente v vhodnem vektorju, če ne uporabimo drugačnih pristopov, npr. kot v topologiji 4. Iskanje minimuma ali maksimuma je ključno pri preslikavi vhodnega vektorja v kot premika.

V topologiji 1 mreža dobi na vhod 5 dimenzionalni vektor. Poiskati mora največjo komponento vektorja in na osnovi pozicije te komponente prirediti izhodu pravilen kot. Naprej povezana nevronska mreža deluje tako, da sešteva produkte uteži z vhodnimi nevroni. Vsako funkcijo, ki jo takšna mreža lahko realizira, lahko predstavimo kot računanje vsote produktov. Iskanje minimuma ali maksimuma ne moremo izraziti na ta način.

V topologiji 2, je problem še večji, saj mora mreža preslikati kot najbližjega predmeta. Torej najprej poišče največji element in nato preslika njegovo sosednjo vrednost-kot. Vhodni vektor je definiran kot: $\boldsymbol{x}$ = *{kot 1, razdalja 1, ..., kot 5, razdalja 5}*. Zaradi tega je delovanje te topologije najslabše.

Topologija 3 je teoretično gledano boljša od prejšnjih dveh, ker je princip delovanja drugačen. Mreža dobi na vhod 16 dimenzionalni vektor. Vrednost komponent vektorja je lahko 0 ali 1 in zato je število vseh možnih vektorjev enako Mreža ne more iskati minimuma, ampak se ga uči na pamet. Če bi bila učna množica dovolj velika, bi se mreža naučila delovati pravilno. Ta topologija nas spominja na mrežo za razpoznavo pisave.

Topologija 4 je edina, ki daje zadovoljive rezultate, kar je posledica njene drugačne arhitekture. Sestavljena je iz rekurzivne in naprej povezane nevronske mreže. Rekurzivni del prejme vhodni vektor. Deluje na tekmovalen način in rezultat tekmovanja je zmagovalni nevron, ki ima vrednost večjo od 0, vsi ostali pa enako 0. Zmagovalni nevron predstavlja lokacijo največje komponente vhodnega vektorja. Naprej povezana nevronska mreža mora nato samo priredit kot na osnovi lokacije zmagovalnega nevrona. Ta del je enostaven, saj mora mreža v utežeh povezav shraniti samo pet različnih kotov, ki ustrezajo petim različnim

zmagovalnim nevronom.

Zaradi gornjih rezultatov sem za implementacijo nevronske mreže izbral topologijo 4.

3.4. Vpliv učne množice in začetnih uteži na učenje

Namen poglavja je ugotoviti, zakaj se srednje kvadratne napake nevronskih mrež močno razlikujejo med seboj. Učenje je deterministično, zato sta edini možnosti za konvergenco srednje kvadratne napake v različne lokalne minimume naključno generirane učne množice in naključne začetne uteži. Merim vpliv učne množice in začetnih uteži na kvaliteto učenja ter na srednjo kvadratno napako nevronske mreže. Izračunam razliko med naučeno nevronsko mrežo in nenaučeno.

3.4.1. Opis problema

Organizem, predstavljen z lastno nevronsko mrežo je postavljen v naključno okolje. Uteži nevronske mreže, ki predstavljajo znanje organizma, so ob rojstvu organizma postavljene naključno in se kasneje spreminjajo v procesu učenja. Namen učenja je uspešno iskanje hrane, ki je odvisno od učnega pravila, topologije nevronske mreže, okolja in začetnih uteži. Učno pravilo in topologija sta ves čas enaki, spreminjata se okolje in začetne uteži. V nadaljevanju analiziram vpliv teh dveh parametrov na rezultat učenja.

3.4.2. Merilne metode

Merim vrednosti kvadratne napake nevronske mreže pri različnih učnih množicah in začetnih utežeh. Kvadratna napaka včasih ne doseže zadovoljive vrednosti, kljub velikemu številu iteracij. Razlog tiči v slabih začetnih utežeh in učni množici. Zanima me, kaj izmed tega bolj vpliva na konvergenco kvadratne napake.

V prvem poskusu sem naredil 15 nevronskih mrež in 15 učnih množic. Velikost učne množice je 500 vzorcev. Vsako nevronsko mrežo sem učil na vsaki od učnih množici. Ker me je zanimala učinkovitost mreže po vsakem učenju brez predhodnega znanja, sem naučene uteži zavrgel takoj po testiranju. Testiranje je bilo izvedeno na naključnem okolju, ki je bilo izbrano na začetku in je bilo za vse organizme enako. V poskusu sem izračunal povprečje 225 (15 x 15) srednjih kvadratnih napak za različne nevronske mreže in učne množice.

V drugem poskusu sem naredil 1 nevronsko mrežo in 1000 učnih množic. Velikost učne množice je 500 vzorcev. Iz izmerjenih povprečnih kvadratnih napak sem izračunal delež leteh v primerjavi s povprečno kvadratno napako nenaučene nevronske mreže (mreža takoj po inicializaciji). Nato sem narisal histogram deležev napak in skušal ugotoviti verjetnostno porazdelitev, ki bi se najbolje prilegala vrednostim histograma.

V tretjem poskusu sem naredil 1 učno množico in 1000 nevronskih mrež. Vse ostalo je enako kot v drugem poskusu.

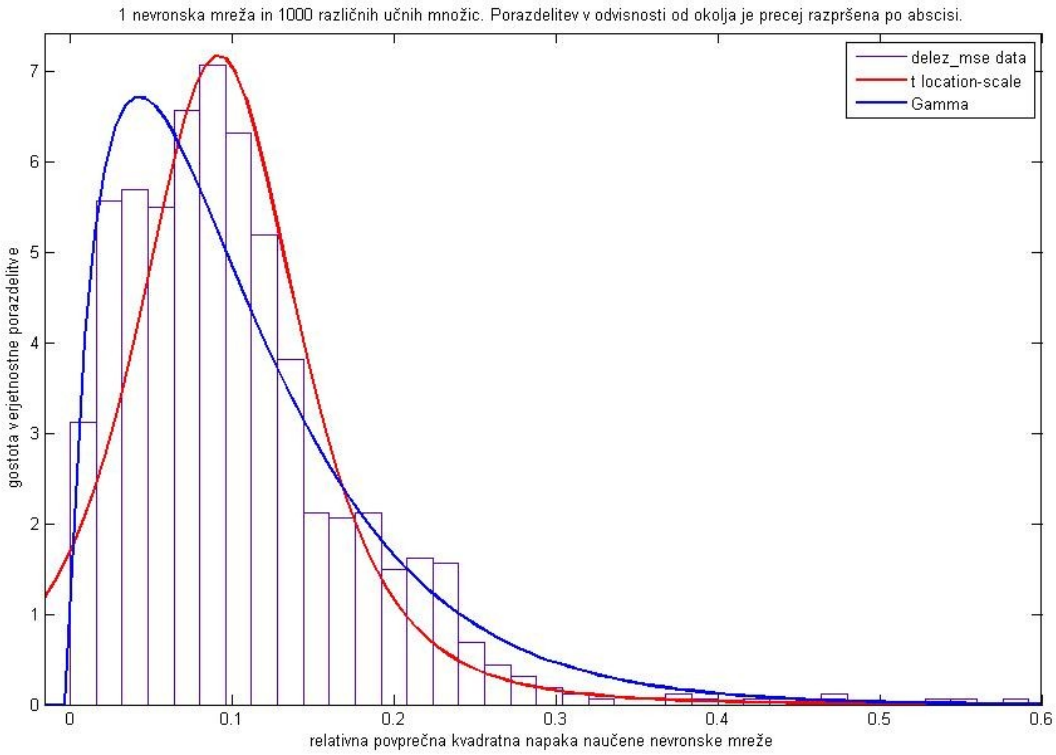
3.4.3. Rezultati meritev

Za prvi poizkus sem izračunal povprečje srednjih kvadratnih napak v razmerju s srednjo kvadratno napako nenaučene nevronske mreže. Razmerje znaša 8,9%. To pomeni, da je delovanje mreže z učenjem v povprečju izboljšano za 91,1%.

Za drugi in tretji poizkus sem izrisal graf histograma in verjetnostne porazdelitve, ki najbolj ustreza danim podatkom.

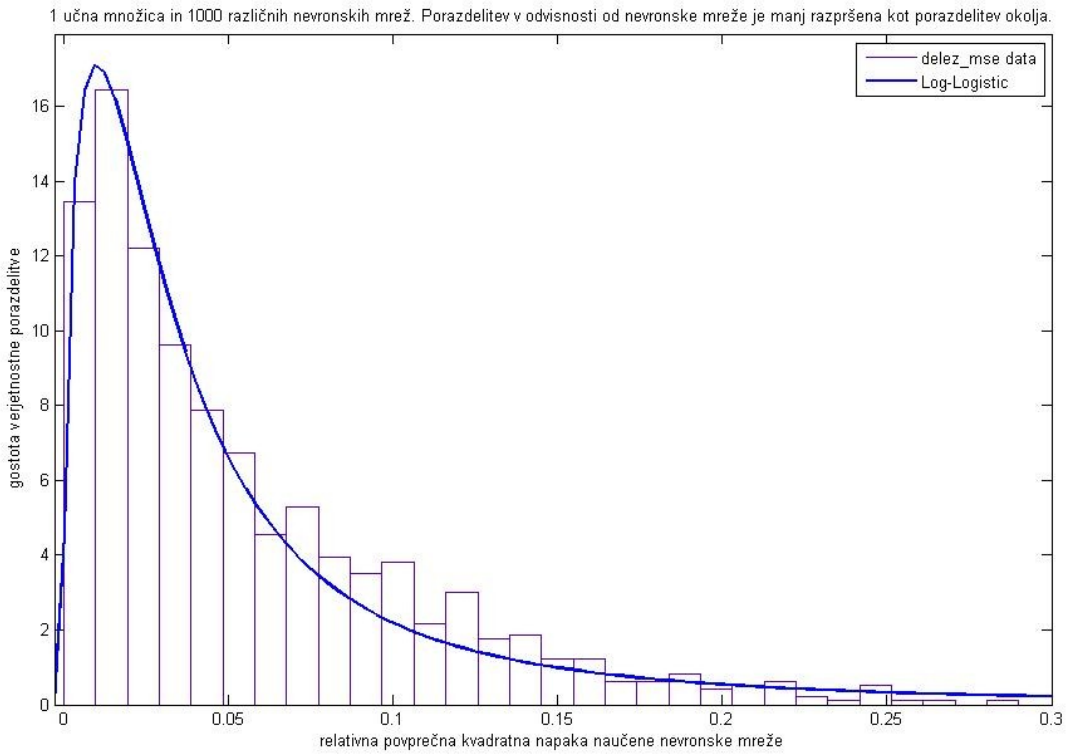
Na sliki 14 je narisan rezultat meritev za drugi poskus. Verjetnostne porazdelitvi sta dve, saj prva porazdelitev (rdeča barva) bolje opiše drugi del histograma medtem ko druga porazdelitev (modra barva) bolje opiše prvi del histograma.

Na slikah 14 in 15 je abscisna os razmerje med srednjo kvadratno napako naučene in nenaučene nevronske mreže. Ordinatna os je gostota verjetnostne porazdelitve. Določeni integral omenjene funkcije na intervalu a in b pomeni verjetnost, da ima naključno izbrana nevronska mreža relativno srednjo kvadratno napako na tem intervalu.



Slika 14: Prikaz gostote verjetnostne porazdelitve učne množice pri eni nevronski mreži. Funkcija modre barve je Gamma porazdelitev, ki se bolje prilega prvemu delu histograma. Funkcija rdeče barve je Studentova t porazdelitev in se bolje prilega drugemu delu histograma.

V zadnjem primeru ni bilo problemov pri iskanju verjetnostne porazdelitve, saj se slednja lepo prilega histogramu. To vidimo na sliki 15.



Slika 15: Prikaz gostote verjetnostne porazdelitve nevronske mreže pri eni učni množici. Gostota verjetnostne porazdelitve je logistična funkcija.

3.4.4. Interpretacija rezultatov

Natančnost odločanja nevronske mreže je povečana za 91,1% v primerjavi z nenaučeno nevronsko mrežo.

Na sliki 14 vidimo, da je verjetnostna porazdelitev precej bolj široka od verjetnostne porazdelitve na sliki 15. To pomeni, da učna množica (okolje) kot naključna spremenljivka bolj varira od začetnih uteži kot naključne spremenljivke. Iz tega sklepam, da je za dobre rezultate učenja bolj pomembno naključno okolje kot začetne uteži nevronske mreže.

Dobra učna množica oz. okolje pomeni enakomerno porazdeljeno hrano v okolju brez kopičenja v raznih predelih. Če imamo srečo z okoljem, so začetne uteži nevronske mreže manj pomembne, saj bodo rezultati boljši, kot v primeru dobre nevronske mreže in neugodnega okolja.

Razlika variance naključnega okolja in začetnih uteži ni velika, zato je za uspešen organizem pomembno oboje: okolje in naravna danost.

Poglavje 4

Simulacija življenja organizmov v eni generaciji

Ustvaril sem množico organizmov v naključno generiranem okolju s hrano in simuliral njihovo obnašanje v eni generaciji. Čas življenja generacije je 5 minut. Časovni korak pri simulaciji je 0,05 sekunde. Na vsakem časovnem koraku organizmi zaznajo predmete v okolju in temu primerno reagirajo. Ob vsaki reakciji organizem dobi iz okolja odgovor in nevronska mreža sproži eno iteracijo vzvratnega učenja. Če je organizem po času 1 minute še živ, je poteklo 1200 iteracij vzvratnega učenja na različnih vhodnih vektorjih. V eni generaciji organizmov poteče 6000 iteracij vzvratnega učenja.

Za zadovoljivo aproksimacijo npr. polinoma stopnje 5 je potrebna učna množica velikosti 50 vzorcev. Na njej se izvede 300 iteracij vzvratnega učenja. Skupaj je izvedenih 15000 iteracij vzvratnega učenja. V času ene generacije organizmov je izvedenih 40% iteracij potrebnih za aproksimacijo funkcije. To sem ilustriral za boljšo predstavo o številu iteracij v drugih praktičnih primerih.

Za vrednotenenje simulacije je zanimivo število preživelih organizmov v določenih časovnih točkah npr. po 30s, 45s, 1min, 90s, 2min, 3min, 5min. Takšno vrednotenje sem naredil za različno količino hrane v okolju, za različno začetno število organizmov ter za različna okolja. Simulacijo primerjam z organizmi, ki se naključno gibljejo in se ne učijo. Prehranjevanje pri njih je posledica naključnega premika na položaj, kjer se nahaja hrana.

Pomembno je poudariti, da smrt organizma pogosto ni povezana z njegovim razmišljanjem. Taki primeri se zgodijo, ko v bližini organizma ni hrane in je zato v vidnem polju ne more zaznati. V takem primeru se giblje naprej, dokler ne pride do območja hrane ali zadane v predmet. Če je večino hrane v predelih odmaknjenih od organizma, bo ta lahko dolgo taval, dokler ne pride do hrane. Večinoma se zgodi, da v takem primeru umre, saj mu hrana daje energije za samo približno 15 sekund življenja. Druga možnost za smrt organizma je, ko imamo v predelu s hrano še druge organizme, ki so hitrejši in pojedo vso hrano.

4.1. Simulacija hitrosti učenja

Parametri, ki jih za vsako posamezno simulacijo spreminjam so količina hrane v okolju, poraba energije na časovno enoto, število organizmov in število časovnih korakov med učenjem organizmov.

Začetni parametri za prve štiri simulacije:

- 15 organizmov
- 60 enot hrane
- 1 enota hrane zadostuje za 15 sekund premikanja

št. organizma	čas smrti 1 [s]	čas smrti 2 [s]	čas smrti 3 [s]	čas smrti 4 [s]
1	7,99	7,67	8,94	59,17
2	8,92	9,06	21,47	70,68
3	10,91	12,69	23,37	300,79
4	15,67	16,24	26,72	
5	18,29	16,89	31,03	
6	24,27	20,95	43,81	
7	25,5	22,03	167,74	
8	30,27	22,97	265,33	
9	32,63	23,85		
10	33,18	25,12		
11	38,33	26,37		
12	40,05	87,28		
13	48,63	94,29		
14	120,31	139,25		
15	151,77			

Tabela 2: Čas smrti posameznih organizmov v sekundah za prve štiri simulacije. Prazen prostor pomeni, da je organizem preživel 5 minut simulacije.

Za vsak stolpec v tabeli 2 je bilo izbrano drugačno število časovnih korakov med učenjem. Stolpci pomenijo:

1. naključno premikanje organizmov,
2. 10 časovnih korakov med učenjem,
3. 5 časovnih korakov med učenjem,
4. učenje organizma na vsakem časovnem koraku.

Za zgornje simulacije sem izbral veliko količino hrane v okolju, večje začetno število organizmov in veliko porabo energije za premikanje organizmov. Parameter, ki sem ga spreminjal za vsako izmed štirih simulacij, je število časovnih korakov med učenjem organizmov. Organizem tako potrebuje več časa, da se nauči iskanja hrane.

V prvi simulaciji se organizmi naključno premikajo. Večina jih umre v manj kot minuti, dva pa v naslednji minuti in pol. Ostale simulacije primerjamo s to in tako ugotavljamo izboljšavo organizmov pri učenju z nevronskimi mrežami.

V drugi simulaciji umrejo vsi organizmi razen enega. To se zgodi zato, ker je poraba energije za premikanje velika in ker se organizmi počasi učijo. En organizem je le preživel 5 minut simulacije. To se je zgodilo zaradi sreče s postavitvijo hrane v njegovi bližini, hitrega premikanja, saj ga ostali organizmi niso prehiteli in mu pobrali hrane v okolici ter dobre inicializacije začetnih uteži nevronske mreže. Ko je potekel čas učenja, se je organizem okolju prilagodil in sam preživel še preostali dve minuti in pol.

V tretji simulaciji preživi skoraj polovica organizmov, ker se organizmi hitreje učijo se dovolj hitro prilagodijo okolju. Dva organizma umreta precej za ostalimi šestimi organizmi, ki so umrli v začetni fazi učenja.

V četrti simulaciji preživi večina - 12 organizmov. Hitrost učenja je tako velika, da v začetni fazi umreta samo dva organizma. Tretji umre na koncu simulacije - po 5 minutah. To se zgodi, ker se je slabo naučil iskanja hrane (odvisno od lokalnega okolja in začetnih uteži) in delno zato, ker je bil prepočasen, v bližini ni bilo hrane in so ga drugi organizmi prehiteli in pojedli vso hrano v okolici. Po petih minutah organizem dobi dovolj iteracij vzvratnega učenja, da se je ob ustreznih začetnih pogojih sposoben naučiti učinkovitega iskanja hrane.

Po koncu vsake izmed zgornjih simulacij z izjemo prve smo dobili množico organizmov, ki je preživela začetno fazo učenja in nato obstala celoten čas simulacije. Ta množica organizmov je učinkovito vsrkala znanje iz okolja ter poleg tega imela še nekaj sreče. Opazimo, da se s hitrostjo učenja ta množica veča. Tako imamo v drugi simulaciji samo en preživeli organizem, v tretji 7 in v četrti 12. Za podoben rezultat bi lahko izbrali manjšo porabo energije za premikanje. Ugotovimo, da če želimo imeti množico dobrih organizmov, moramo ustvariti precej večjo začetno množico organizmov, kajti zaradi naključne inicializacije bomo vedno imeli nekaj slabih organizmov. Zaradi načela naravnega izbora dobri organizmi preživijo, slabi pa ne.

4.2. Simulacija manjše porabe hrane

V prvih štirih simulacijah je bilo število organizmov, količina hrane in poraba energije velika. Sedaj si pa poglejmo, kako izgleda življenje organizmov z nastavitvijo parametrov na nizke vrednosti.

Začetni parametri za peto in šesto simulacijo:

- 12 organizmov
- 20 enot hrane
- 1 enota hrane zadostuje za 50 sekund premikanja
- 10 časovnih korakov med učenjem

št. organizma	čas smrti 5 [s]	čas smrti 6 [s]
1	31,98	26,37
2	34,43	27,21
3	44,54	32,47
4	57,33	54,22
5	60,26	71,68
6	65,58	80,62
7	68,1	94,29
8	70,52	114,87
9	128,73	
10	139,86	
11	188,6	
12	197,72	

Tabela 3: Čas smrti posameznih organizmov simulaciji.

Legenda številk prve vrstice stolpcev:

5. naključno premikanje organizmov
6. organizmi z nevronskimi mrežami

V peti simulaciji opazimo, da so organizmi živeli dlje kot v prvi simulaciji, čeprav so se v obeh primerih naključno premikali. Organizmov je manj in prav tako tudi hrane. Iz tega sledi, da je najpomembnejši parameter za daljše življenje organizmov nizka poraba energije na časovno enoto, ki je v peti in šesti simulaciji majhna.

V šesti simulaciji je bilo treba žrtvovati dve tretjini organizmov, da so se naučili preživetja. Tistim, ki je to uspelo, so preživel opazovani čas simulacije.

Organizmi imajo na začetku shranjene različne količine energije za premikanje, zato nekateri umrejo pred 50s. Šele ko organizem poje hrano, se do konca nasiti.

4.3. Simulacija srednje porabe hrane

Za konec si poglejmo še dve simulaciji organizmov s srednjo porabo energije za premikanje. Pričakujemo, da bodo rezultati med prvimi štirimi in peto ter šesto simulacijo.

Začetni parametri za sedmo in osmo simulacijo:

- 10 organizmov
- 20 enot hrane
- 1 enota hrane zadostuje za 30 sekund premikanja
- 10 časovnih korakov med učenjem

št. organizma	čas smrti 7 [s]	čas smrti 8 [s]
1	11,43	54,66
2	12,07	77,42
3	13,72	80,47
4	21,51	215,19
5	30,14	229,2
6	33,16	
7	37,79	
8	40,31	
9	48,14	
10	52,38	

Tabela 4: Čas smrti posameznih organizmov za simulaciji.

Legenda številk prve vrstice stolpcev:

7. naključno premikanje organizmov,
8. organizmi z nevronskimi mrežami.

V simulaciji naključno premikajočih se organizmov vidimo, da so časi smrti relativno podobni prvi simulaciji. Prvih deset organizmov prve simulacije dejansko umre kasneje kot v sedmi simulaciji.

V osmi simulaciji trije organizmi umrejo v času učenja. Naslednja dva organizma umreta v fazi tekmovanja in izpopolnjevanja iz podobnih razlogov, kot sem jih naštel v nekaterih prejšnjih simulacijah.

V zgornjih simulacijah smo dobili občutek, kako življenje organizmov poteka, kdaj umirajo in koliko jih preživi v odvisnosti od začetnih parametrov. Pomembno je, da znamo nastaviti parametre tako, da po koncu generacije preživi želeno število organizmov in da se uspešno naučijo preživetja.

Poglavje 5

Evolucijsko računanje

V prihodnosti, v stoletjih merjeno ne daleč proč, bodo civilizirane človeške rase skoraj zagotovo iztrebile in zamenjale podivjane rase po celem svetu.

Charles Darwin

5.1. Darwinovo naravno odbiranje in evolucija

Skozi zgodovino življenja so se na Zemlji ustvarjali in izumirali živalski ter rastlinski rodovi, vrste, zvrsti itd. Vrste se nenehno spreminjajo, razvijajo in napredujejo. Nekateri posamezniki znotraj vrste predrugačijo svojo zgradbo in se razmeram v okolju bolje prilagodijo, nekateri pa ne. Za napredek življenja v naravi je potrebno izboljševanje, torej morajo boljši posamezniki ustvariti potomce in nadomestiti slabše posameznike ter ostale prednike znotraj vrste. Vidimo da so živalske in rastlinske vrste danes kompleksnejše oblike, kot so bile epoho pred tem. Napredek torej je, zakaj je temu tako, je prvi razložil Charles Darwin s teorijo *naravnega odbiranja.*

Naravni nagon posameznikov vsake vrste je ustvarjanje čim večjega števila potomcev. Za ohranitev vrsta potrebuje številno potomstvo. V naravi sobiva mnogo živalskih in rastlinskih vrst. Zaradi omejitve naravnih virov, morajo vrste med seboj tekmovati za obstanek. Če bi se vrsta nenadzorovano razmnoževala in če bi vsi potomci preživeli ter ustvarili nove potomce, bi velikost vrste narasla preko vseh meja in nobena pokrajina tega ne bi mogla vzdržati.

Zaradi boja za preživetje bo vsaka sprememba, naj si bo še tako drobna in ne glede na vzrok svojega izvora, pomagala ohranjati posameznika, če mu vsaj malo koristi pri zapletenih razmerjih z drugimi bitji in naravo, njegovi potomci pa jo bodo večinoma dedovali. Do sprememb posameznika pride zaradi križanja staršev in naključnih mutacij. Spremembe so lahko tudi negativne, torej posameznik postane zaradi podedovanih lastnosti in zgradbe manj sposoben od ostalih v vrsti. Tak posameznik bo imel manj možnosti za preživetje in bo manj verjetno ustvaril potomce. Na takšen način se škodljive spremembe zavračajo. Po drugi strani bodo imeli posamezniki, s prednostjo pred ostalimi, večjo možnost, da preživijo in nadaljujejo vrsto. Ohranjanje koristnih in zavračanje škodljivih sprememb je Darwin poimenoval *naravno odbiranje*.

Ko se rodi posameznik z boljšimi sposobnostmi od ostalih, bo verjetneje ustvaril potomce. Nekateri potomci bodo najbrž podedovali iste navade ali zgradbo in morda bo nastala nova zvrst, ki bo spodrinila prejšnjo obliko ali sobivala z njo. Če se nekatere izmed vrst predrugačijo in izboljšajo, se bodo morale primerno izboljšati tudi druge, ali pa bodo izumrle. Nove oblike, ki so nastale na velikih območjih in so že premagale številne tekmice, se bodo najbolj razširile, ustvarile večino novih zvrsti in vrst ter tako prispevale pomembno vlogo v spremenljivi zgodovini organskega sveta.

Zanimivo je tudi s*polno odbiranje,* ki je med samcem in samico povzročilo ločitev po zgradbi, okrasju ali barvi. Pri spopadanju samcev za samice, premaganega tekmeca ne doleti smrt, zato pa ima malo potomcev ali pa sploh nobenega. Favorizirani so močnejši samci z boljšim orožjem, dobro obrambo ali bolj očarljivi (primer pava).

Preko spolnega odbiranja se vrši selekcija znotraj vrste, preko naravnega odbiranja pa tudi

med rodovi, vrstami in zvrstmi. Torej obstojita dva nivoja selekcije, ki predrugačita vrste.

Naravno odbiranje daje torej odgovor na vprašanje, zakaj je danes živalski in rastlinski svet tako raznolik in bogat.

5.2. Zgodovina

Uporaba Darwinovih principov v računanju izvira iz petdesetih let prejšnjega stoletja. Šele v šestdesetih letih so se razvile tri različne interpretacije te ideje.

Lawrence J. Fogel iz ZDA je predstavil koncept evolucijskega programiranja, medtem ko je John Henry Holland poimenoval svojo metodo genetski algoritem. V Nemčiji sta Ingo Rechenberg in Hans-Paul Schwefel odkrila evolucijske strategije. Ta področja so se neodvisno razvijala petnajst let. V začetku devetdesetih let so bile te metode imenovane evolucijsko računanje. V istem obdobju se je razvila še četrta metoda, genetsko programiranje. Od devetdesetih let dalje je evolucijsko računanje postalo uporabljeno v tehnične namene.

Nils Aall Barricelli je v šestdesetih letih začel z simulacijo evolucijskih algoritmov in umetnega življenja. Njegovo delo je nadaljeval Alex Fraser, ki je pisal literaturo o umetni selekciji. Umetna evolucija je postala široko prepoznana optimizacijska metoda v šestdesetih letih zaradi dela Inga Rechenberga, ki je uporabil evolucijske strategije za reševanje kompleksnih inženirskih problemov.

Sčasoma se je povečalo zanimanje za področje evolucijskega računanja, zmogljivost računalnikov je narasla in nastale so prve aplikacije, med drugim tudi avtomatska evolucija računalniških programov. Evolucijski algoritmi se danes uporabljajo za reševanje večdimenzionalnih problemov in so v nekaterih primerih bolj učinkoviti kot deterministični algoritmi. Primer je optimizacija arhitekture računalniških sistemov.

5.3. Evolucijske strategije

V tem podpoglavju bom opisal evolucijske strategije, ki sem jih uporabil.

Evolucijske strategije sta prva definirala Rechenberg in Schwefel v šestdesetih letih. Glavna razlika v primerjavi z drugimi algoritmi je samo adaptacija strateških parametrov. Strateški parametri definirajo velikost koraka mutacije, metodo križanja in izbiro preživetja. Samo adaptacija pomeni, da so strateški parametri vključeni v kromosom kot ostali geni, in se spreminjajo zaradi križanja in mutacije.

Tipična uporaba je v primeru optimizacije zveznih parametrov. V diplomski nalogi so to uteži nevronske mreže, prag za preklopno funkcijo, strmina preklopne funkcije, število nevronov v posameznih plasteh ter število povezav med nevroni.

Bistvena stvar pri evolucijskih strategijah je naključna inicializacija osebkov, ki mora biti dobro porazdeljena. Za vse osebke ocenimo, kako dobra je njihova prilagojenost okolju (problemu), s pomočjo funkcije kvalitete. Če je osebek v začetni populaciji po izbranem kriteriju dovolj dober, potem predstavlja rešitev problema. Če takega osebka ni (večinoma je temu tako), se izberemo skupino dobrih osebkov na osnovi njihove kvalitete. Ta skupina ustvari nasledstvo – bila je naravno odbrana. Metod za izbor staršev je več in jih obravnavali kasneje. Osebki znotraj skupine se križajo in dobimo njihove potomce. Ti z neko verjetnostjo mutirajo. Ta verjetnost je strateški parameter evolucijskih strategij, ki tudi deduje. Za potomce izračunamo stopnjo prileganja okolju in nato v skupini staršev ter njihovih potomcev izberemo skupino kandidatov za novo generacijo, ki gre ponovno skozi proces boja za preživetje.

Za boljšo predstavitev podajam splošno shemo družine evolucijskih algoritmov:

1. Inicializiraj populacijo z naključno izbiro kandidatov
2. Ocena vseh kandidatov – funkcija kvalitete
3. Pogoj zaustavitve: rešitev dovolj dobra ali dovolj veliko št. korakov
3. Ponavljaj dokler (pogoj zaustavitve):
4. Izberi pare staršev na osnovi ocen
5. Križaj pare staršev
6. Mutiraj dobljene potomce
7. Oceni nove kandidate
8. Izberi kandidate za novo generacijo

V naslednjih podpoglavjih bom bom natančneje opisal vse faze genetskega algoritma. Nato bom podal zglede praktičnih aplikacij za različne implementacije faz z različnimi parametri. Faze algoritma so ocenjevanje kandidatov, metoda izbire staršev – selekcija, križanje, mutacija in izbira kandidatov za novo generacijo.

5.3.1. Predstavitev osebkov

Dejanski je genom sestavljen iz dolgega niza genov. Geni kodirajo lastnosti organskega bitja. V evolucijskem računanju je genom lahko sestavljen iz niza realnih števil. Del genoma predstavljajo geni, ki kodirajo lastnosti in zgradbo posameznika v populaciji, del pa geni, ki kodirajo strateške parametre za nadziranje evolucije.

Izgled genotipa:

$$\{x_1, x_2, \ldots, x_n, \sigma_1, \sigma_2, \ldots, \sigma_m, \alpha_1, \alpha_2, \ldots, \alpha_k\}$$

Pomen spremenljivk:

- $x_1, x_2, \ldots, x_n$: geni, ki kodirajo lastnosti in zgradbo posameznika,
- $\sigma_1, \sigma_2, \ldots, \sigma_m, \alpha_1, \alpha_2, \ldots, \alpha_k$: strateški parametri npr. korak mutacije.

Velikost populacije je ponavadi med 20 in 100 osebkov. Biti mora dovolj obsežna, da zadovoljivo pokrije prostor možnih rešitev.

5.3.2. Selekcija

Za vsakega posameznika v populaciji izračunamo kvaliteto, ki jo predstavimo z realnim številom. Na osnovi ocen izberemo skupino posameznikov, ki ustvarijo potomce. Poznamo več različnih metod selekcije. Glavni značilnosti metod, ki ju želimo, sta hitrost konvergence k rešitvi problema in diverziteta populacije. Različne metode ustrezajo za reševanje različnih problemov.

Najpogostejše metode selekcije so:

- proporcionalna izbira,
- izbira z rangiranjem,
- ruletna izbira,
- turnirska izbira.

5.3.2.1. Proporcionalna izbira

Verjetnost za izbor posameznika je sorazmerna vrednosti njegove kvalitete. Ta verjetnost je normalizirana z vsoto kvalitete vseh osebkov:

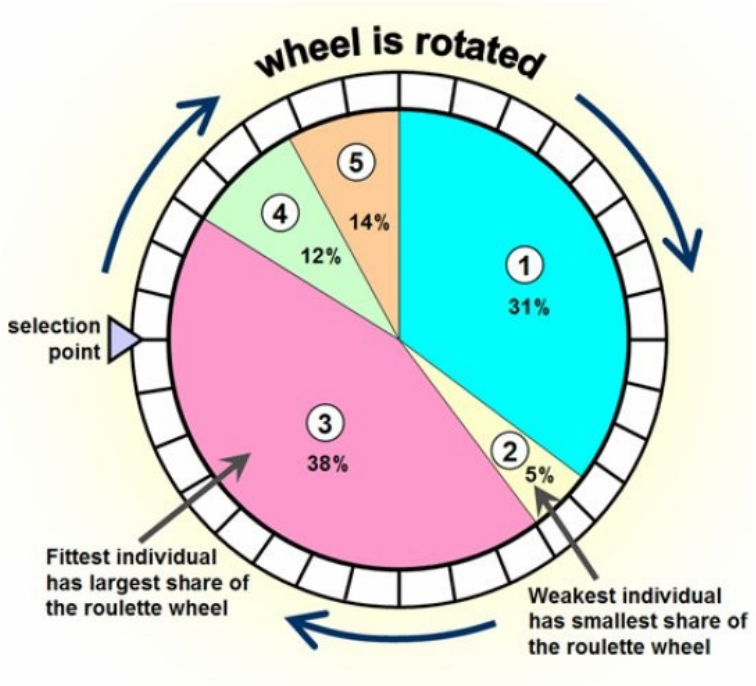
$$p_i = \frac{f_i}{\sum_j^n f_j}$$

Pomen posameznih spremenljivk:

- *i:* indeks osebka,
- *n:* velikost populacije,
- *f:* stopnja prileganja posameznika,
- *p:* verjetnost za izbor posameznika.

5.3.2.2. Ruletna izbira

Ruletna izbira sledi neposredno iz proporcionalne izbire, saj uporabimo isto formulo za izračun verjetnosti izbire posameznika. Verjetnost izbire posameznika si lahko predstavljamo kot površino ruletnega kolesa. Večja kot je verjetnost, večja je površina. Izbiro posameznika si lahko zamislimo kot zasuk ruletnega kolesa in indeks površine, kjer se je kolo ustavilo, predstavlja indeks izbranega kandidata. Ilustracijo vidimo na sliki 16.



Slika 16: Demonstracija delovanja ruletnega postopka [10].

Pri tem postopku imamo veliko mero stohastičnosti. V skupino za razmnoževanje ne bodo prišli samo najboljši kandidati, ampak tudi nekaj slabih. Seveda bo v povprečju boljših kandidatov več, saj je tudi verjetnost za njihov izbor višja. Ta model selekcije ustreza biološkemu svetu, saj ni nujno, da samo najboljša bitja preživijo in ustvarijo potomce.

5.3.2.3. Izbira z rangiranjem

Osebke razvrstimo po naraščajočem vrstnem redu glede na njihovo kvaliteto. Verjetnost izbora se izračuna na osnovi zaporedne številke posameznika v urejenem seznamu. Verjetnost se izračuna podobno kot pri proporcionalni izbiri:

$p_i = \frac{r_i}{\sum_j^n r_j}$, kjer r_i predstavlja zaporedno številko posameznika v urejenem seznamu (rang).

Rezultati te metode so podobni kot pri ruletnemu postopku, torej visoka stopnja stohastičnosti. Najboljši osebki hitro nadomestijo ostale osebke, kar lahko pripelje do prehitre konvergence in lokalnega optimuma.

5.3.2.4. Turnirska izbira

Poznamo več različnih tipov turnirjev. Opisal bom enoturnirsko izbiro. Pri tej metodi naključno izbiramo posameznike iz populacije in tvorimo majhne skupine, navadno velikosti

okoli 5 posameznikov. Iz skupine izberemo dva osebka z najvišjo vrednostjo funkcije prileganja, ki ustvarita dva potomca. Potomca nadomestita najslabša osebka iz skupine. Prednost enoturnirske izbire je, da tekmovanje preživijo vsi osebki razen najlabši. Na takšen način ne izgubljamo prehitro dobrih osebkov, obenem pa kvaliteta populacije ne pada. Najboljši posamezniki počasi nadomeščajo najslabše posameznike. Strategija zagotavlja, da še tako dober osebek nima več kot dveh potomcev, zato ohranja raznolikost populacije. Konvergenca k rešitvi problema je počasnejša, a metoda dobro preišče prostor možnih rešitev in se manj verjetno ustavi v lokalnem minimumu kriterijske funkcije.

5.3.3. Rekombinacija ali križanje

V naravi se pri spolnem razmnoževanju organizmov kromosomi staršev naključno mešajo. Pri genski rekombinaciji se izmenjujejo tudi manjši odseki DNK med enakovrednimi kromosomi. Takšen proces zagotavlja, da se bodo potomci razlikovali od staršev, obenem pa bodo imeli veliko lastnosti podobnih. Raznolikost populacije se tako veča, koristne spremembe so naravno odbrane in se prenesejo naprej, škodljive pa zavrnejo.

V evolucijskem računanju rekombinacijo simuliramo na več različnih načinov. Pri diskretni rekombinaciji je element genotipa potomca izbran naključno med elementoma obeh prednikov, pri vmesni rekombinaciji pa s povprečenjem vrednosti obeh prednikov. Vrednost posameznega gena v genomu novo ustvarjenega potomca je enaka:

$$z_i = \begin{Bmatrix} x_i \, ali \, y_i; & diskretna \, rekombinacija \\ \frac{x_i + y_i}{2}; & vmesna \, rekombinacija \end{Bmatrix}$$

Pomen spremenljivk:

- *i:* lokacija gena,
- *x, y:* gena staršev,
- *z:* gen potomca.

Evolucijske strategije tipično uporabljajo diskretni tip za aplikacijske parametre in vmesni tip rekombinacije za strateške parametre. Aplikacijski parametri predstavljajo rešitev problema, strateški parametri pa so pomembni za iskanje rešitve. Primer je korak mutacije.

5.3.4. Mutacija

Mutacije so trajne dedne spremembe genoma, ki jih povzročijo različni dejavniki, npr. kemijski (vpliv spojin), biološki (virusi) ali fizikalni (sevanje). Njihovo bistvo je sprememba zgradbe DNK. Mutacije so redke in večinoma škodljive, pogosto povzročajo propad mutiranega osebka. Obstajajo pa tudi mutacije, ki so za osebek koristne in povečajo možnost za preživetje in se tako prenesejo naprej na potomce.

Operator mutacije pri genetskih algoritmih bazira na normalni verjetnostni porazdelitvi. Srednja vrednost verjetnostne porazdelitve je enaka nič, standardni odklon pa je navadno vključen v genom osebka kot strateški parameter in se zato s časom spreminja. Mutacija se izvede tako, da se nekaterim naključno izbranim genom doda vrednost iz normalne verjetnostne porazdelitve. Število genov, ki mutirajo, je stvar izbire.

Vrednost gena na lokaciji *i* je po mutaciji enaka:

$x_i' = x_i + N(0, \sigma)$, koder N pomeni normalno verjetnostno porazdelitev, σ pa standardni

odklon mutacije. Standardni odklon se manjša premo sorazmerno z ožanjem prostora rešitev.

5.3.5. Preživetvena strategija

Iz množice prednikov in potomcev je potrebno izbirati množico osebkov, ki ustvarijo novo generacijo. V naravi velikost populacije upada, če se organizmi niso dovolj prilagodili okolju. V skrajnem primeru vrsta izumre.

Tendenca vsake vrste je naraščanje in širjenje, kar izhaja iz boja za preživetje (številčnejši so močnejši). Vrsta se s časom širi, če je sestavljena iz posameznikov z boljšimi sposobnostmi od ostalih vrst.

Vrste, ki živijo prosto v naravi, ne ohranjajo velikosti, kar pa ne velja za udomačene vrste, saj tukaj človek izvaja selekcijo.

Pri evolucijskih strategijah poznamo več različnih metod izbire preživetja, ki so podobne tistim v naravi. V *naključni zamenjavi* potomci zamenjajo vse starše. Če zamenjamo prednike z verjetnostjo zamenjave obratno sorazmerne s funkcijo kvalitete, uporabljamo *ruletno zamenjavo.* Podobna je metoda *rangirne zamenjave,* kjer so najverjetnejši organizmi za zamenjavo tisti, ki so na dnu seznama urejenega po kvaliteti posameznikov.

Metoda zamenjave po kvaliteti *(absolute fitness replacement)* zamenja najslabše posameznike z najboljšimi otroci. V *lokalni elitni zamenjavi,* se prednika primerjata z njunima potomcema in najboljša posameznika preživita. V *naključni elitni zamenjavi* se vsak potomec primerja z naključno izbranim prednikom. Če je njegova funkcija kvalitete višja, potem ga zamenja.

Z metodami selekcije in zamenjave določamo, koliko potomcev se ustvari in koliko jih zamenja prednike v vsaki generaciji evolucijskega algoritma. V ekstremnem primeru potomci popolnoma nadomestijo prednike. To so *generacijski* evolucijski algoritmi. Drug ekstrem so pari staršev, ki ustvarijo potomce, ki jih vstavimo v populacijo kot novo generacijo. Velikost generacije se tukaj veča.

Izbira preživetja pomembno vpliva na hitrost konvergence k rešitvi problema in na raznolikost populacije. Od tega je odvisno ali bomo obtičali v lokalnem ekstremu in ne bomu našli prave globalne rešitve problema. Zgodi se lahko, da so si posamezniki zelo podobni. V tem primeru s križanjem dobimo potomce, ki so še bližje lokalni rešitvi, raznolikost populacije pade in nimamo več posameznikov, ki bi lahko pripeljali do dobre rešitve razen z mutacijo, kar pa je malo verjetno.

5.4. Uporaba genetskih algoritmov

Evolucijsko računanje je metoda za reševanje optimizacijskih problemov v mnogih inženirskih področjih. Uporablja se v primerih, koder ne poznamo dovolj hitrih determinističnih algoritmov. Razlog za to je navadno prevelik prostor rešitev. Problem je treba matematično modelirati in razviti funkcijo, ki opisuje kakovost rešitve. Evolucijski algoritmi s spreminjanjem parametrov funkcijo minimizirajo ali maksimizirajo in skušajo najti globalni ekstrem. Množica parametrov, ki jih na takšen način dobimo, predstavlja več možnih rešitev problema.

Področja uporabe evolucijskega računanja:

1) Oblikovanje v strojništvu

   Uporablja se za oblikovanje mehanskih delov z višjo trdnostjo in nosilnostjo, za iskanje kvalitetnih materialov, optimiziranje delovnih procesov, konstrukcijo delov za boljši prenos toplote na strojih, povečanje izkoristka turbin. Poleg optimizacije in iskanja rešitev problemov se uporabljajo še za odkrivanje pomanjkljivosti in napak izdelkov.

2) Razvoj elektro-strojne opreme (Evolvable Hardware)

   Genetski algoritmi uporabljajo stohastične operatorje (križanje, mutacija, selekcija) za razvoj boljših konfiguracij električnih vezij na osnovi starih konfiguracij. Načrtovalec definira cilje in lastnosti, ki jih mora končna konfiguracija vsebovati. Ostalo je prepuščeno evoluciji. Zanimiva je ideja o električnem vezju, ki se s pojavitvijo novih problemov, ki jih ne zna rešiti, reprogramira, dobi nove funkcije in se prilagodi novemu okolju. Takšne funkcije so npr. samo adaptacija in samodejno odpravljanje napak.

3) Računalniške igre

   Genetski algoritmi se uporabljajo v simulacijskih igrah, kjer igralec ustvarja civilizacije in jih razvija v naprednejše. Imajo vlogo nasprotnika igralcu. Genetski algoritmi črpajo znanje iz odigranih iger in iščejo najuspešnejše strategije. Programi se učijo vključevanja znanja iz teorije iger v oblikovanje novih strategij.

4) Finančne strategije

   Genetski algoritmi skupaj z nevronskimi mrežami se uporabljajo za analizo finančnih trgov, napovedovanje dogodkov in iskanje strategij vlaganja. Pomanjkljivost tega je slaba napovedljivost redkih dogodkov in drastičnih sprememb v kratkem času kot npr. začetek gospodarske krize leta 1929 ali 2007.

5) Kemija in biologija

   Genetski algoritmi se uporabljajo za razumevanje zvitja proteinov, analize učinkov ob spremembah proteinov in za napovedovanje vezave elementov na proteine, ki so jih razvile farmacevtske družbe za zdravljenje nekaterih bolezni. Podobne algoritme najdemo pri oblikovanju kemijskih spojin. Genetski algoritmi so koristni tudi pri analizi izražanja genov, kjer uporabljamo časovno zaporedje različnih konfiguracij aktivnosti genov v celici. Naloga analize je klasifikacija pomena genov, kar lahko privede do odkritja genskih vzrokov za nekatere bolezni.

Poglavje 6

Demonstracija delovanja genetskih algoritmov

V tem poglavju prikažem delovanje genetskih algoritmov na dveh hipotetičnih primerih. Prvi je iskanje globalnega minimuma funkcije in drugi je iskanje pravega zaporedja znakov iz niza naključnih znakov. Namen je pokazati, kako genetski algoritmi rešujejo probleme, kako problem pretvoriti v primerno obliko in kako primerjati kvaliteto rešitve ob uporabi različnih metod selekcije, izbora preživetja, križanja in mutacije.

6.1. Iskanje minimuma funkcije

Podano imamo funkcijo, ki ima poleg globalnega še en lokalni minimum. Lokalni minimum je v točki $x = 0$, globalni pa v točki $x = 21$. Funkcija narisana na sliki 17 je sledeča:

$$y = \begin{cases} -e^{-(x/20)^2}; & x \leqslant 20 \\ -e^{-1} + (x-20)\cdot(x-22); & x > 20 \end{cases}$$

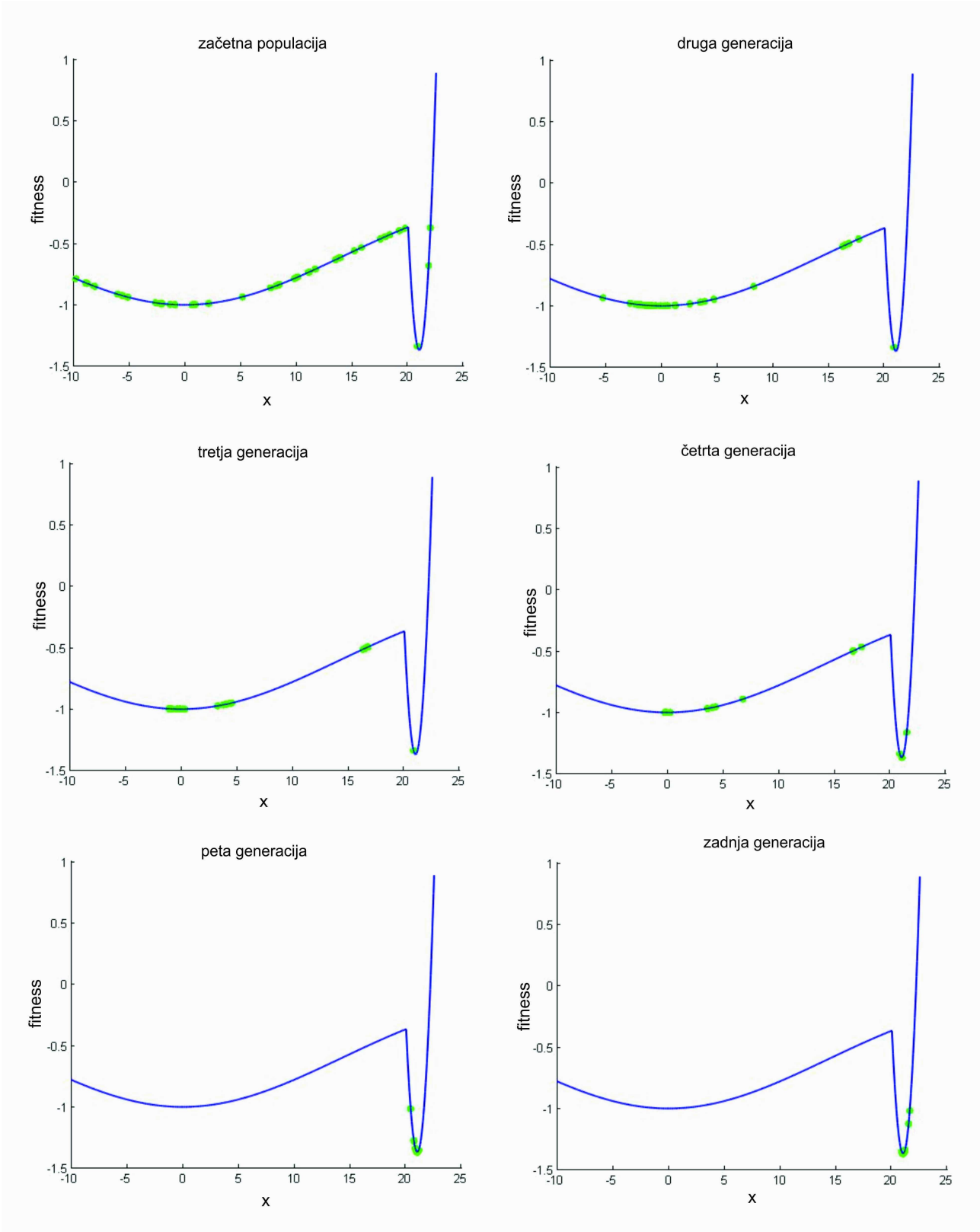
Ker ima funkcija samo eno spremenljivko je genom sestavljen iz samo enega gena. Gen je spremenljivka *x,* predstavljeno v plavajoči vejici. Funkcija kvalitete je vrednost *y.* Kvaliteta rešitve oz. posameznika je obratno sorazmerna z vrednostjo y.

Izbral sem metodo **turnirske izbire**, ker zagotavlja raznolikost populacije in tako poveča možnosti za iskanje globalnega minimuma.

Velikost populacije se ne spreminja in je enaka 40 posameznikov. Velikost turnirja je 8. Iz 40 tekmovalnih skupin izberemo 40 zmagovalcev turnirjev. To so kandidati za razmnoževanje. Kandidate paroma križamo. Vsak par ustvari dva potomca, kjer prvi potomec dobi 20% gena od enega starša in 80% od drugega, drugi potomec pa obratno. Dobili smo 40 potomcev, ki zamenjajo vse prednike. 5 izmed njih mutira tako, da se vrednosti gena prišteje vrednost normalne verjetnostne porazdelitve z disperzijo 0,3. Pogoj zaustavitve algoritma je relativna razlika med staro in novo povprečno kvaliteto generacije, ki mora biti manjša od 0,01. V tem primeru vsi kandidati predstavljajo dobro približno rešitev. Že po 5 generacijah se algoritem ustavi in vrednost najboljšega posameznika v enem izmed poizkusov je enaka 20,9939. To je zelo natančen rezultat, prikazan na sliki 17.

V večini poizkusov je rešitev globalni minimum, včasih pa temu ni tako. To je odvisno od razpršenosti začetne populacije, saj mora vsaj eden izmed posameznikov biti postavljen v bližino globalnega minimuma, da bo dobro ocenjen in zmagovalec v tekmovanju. Na takšen način bo ustvaril nekaj potomcev in število potomcev v okolici globalnega minimuma se bo tako večala. Potomci se bodo hitro približevali točni vrednosti pravilne rešitve.

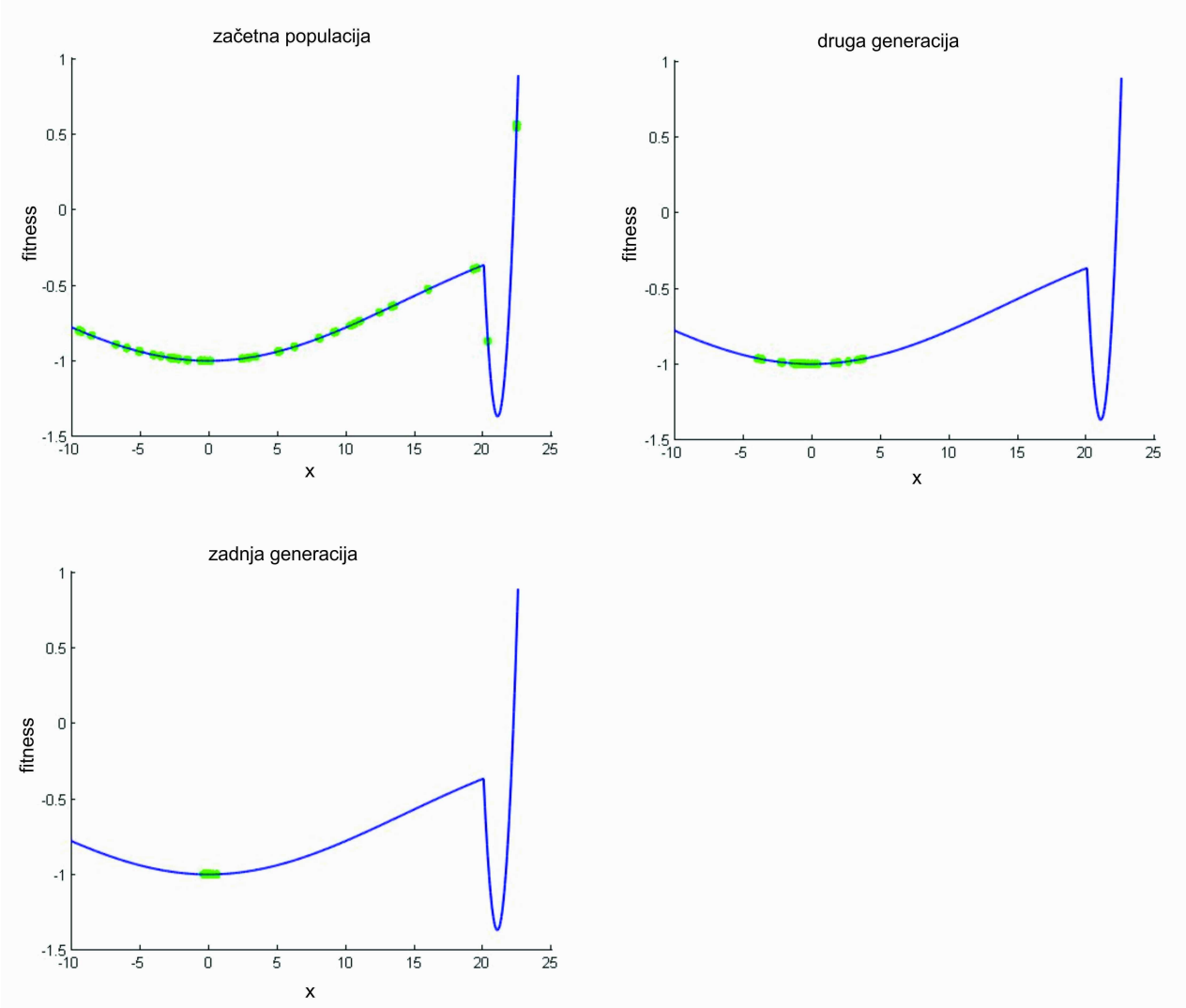
Grafični prikaz delovanja algoritma je na sliki 17.



Slika 17: Delovanje aplikacije za iskanje minimuma funkcije. Posamezniki so zelene barve.

Posamezniki so v začetni populaciji dobro razporejeni po prostoru rešitev. En posameznik se nahaja v neposredni bližini globalnega minimuma, njegova ocena bo najvišja in zato bo zmagovalec vsakega turnirja, če bo v turnir izbran. V drugi generaciji se del posameznikov postavi okoli lokalnega minimuma, majhen del okoli globalnega minimuma in nekaj

posameznikov nekje vmes. Posamezniki vmes so križanci med posamezniki globalnega in lokalnega minimuma. V naslednjih generacijah posamezniki, ki se nahajajo na območju med dvema minimuma, ne bodo izbrani za nadaljevanje vrste, kajti njihova kvaliteta je nižja od ostalih. V naslednjih generacijah opazimo, da se izoblikujejo skupine zelo podobnih posameznikov. Ko je posameznikov v globalnem minimumu dovolj, bo naslednja generacija sestavljena samo iz njihovih potomcev. S časom, bo vsa populacija konvergirala v globalni minimum. V gornjem primeru se to zgodi v šesti generaciji.



Slika 18: Delovanje algoritma v primeru, ko najde samo lokalni minimum.

V neuspešnem primeru iskanja globalnega minimuma na sliki 18 v začetni populaciji ni nobenega posameznika v okolici globalnega minimuma. Na grafu začetne populacije vidimo, da ima najbližji posameznik globalnemu minimumu slabšo kakovost od posameznikov v okolici lokalnega minimuma, ki so izbrani za razmnoževanje. Preiskovanje hitro konvergira v lokalni minimum.

6.2. Iskanje zaporedja znakov

6.2.1. Opis problema in rešitve

Podano imamo zaporedje 16 ASCII znakov. Začetna populacija je sestavljena iz naključnih nizov znakov. Genom posameznika je sestavljen iz 16 genov oz. znakov, gen ima 256 možnih vrednosti. Cilj je poiskati dano zaporedje znakov. Funkcija kvalitete vrne število znakov, ki se ujemajo z genomom posameznika in s podanim zaporedjem. Algoritem se ustavi, ko je vrednost cenilne funkcije najboljšega posameznika enaka 16.

V tabeli 5 je prikazano izvajanje algoritma vsakih nekaj generacij, dokler ne najdemo končne rešitve. Velikost populacije za poizkus je 60 osebkov, uporabljena je turnirska izbira, kjer najboljši osebek v skupini 4 osebkov zamenja najslabšega.

Najboljši posameznik	Fitness	Številka generacije
HadDe Q‘/--<jlm’	3	5
HadDe.em3m/<Ijm-	4	52
HadDe,em3m/<Ijm-	5	54
HadDm,ex3m/#Ijmj	6	73
HadDm,eI8m/#Ijmj	7	86
HadDm,eI8m[Ajjmt	8	118
HadDm,UI8m[Ajjm.	9	135
MadDm,zI8m4AJ1m.	10	154
Madam,zIXm4AJ1m.	11	163
Madam, InmqAJym.	12	256
Madam, I’mqArHm.	13	327
Madam, I’m AC˜m.	14	473
Madam, I’m APam.	15	512
Madam, I’m Adam.	16	647

Tabela 5: Ilustracija iskanja 16 znakovnega niza.

Iskanje pravega zaporedja je dolgotrajen postopek odvisen večinoma od mutacij. Po nekaj generacijah se raznolikost populacije zelo zooži in tako s križanjem ne ustvarjamo več velikih razlik v genskem materialu. Po 100 generacijah so vsi osebki enaki, razlikujejo se le v genih, ki mutirajo. Zato je potrebnih še 500 generacij, da najdemo iskani niz. V tabeli 5 vidimo, da je razlika v generacijah premo sorazmerna z naraščanjem funkcije kvalitete.

6.2.2. Primerjava različnih metod selekcije in izbire preživetja

Primerjam različne metode selekcije in izbire preživetja. Naredil sem 10 poizkusov za 5 različnih metod. Velikost populacije je 60 osebkov in križanje je dvotočkovno. Verjetnost mutacije znaka v nizu je pri prvi metodi 0,6, v ostalih treh pa 0,8. Verjetnosti so zelo visoke

(običajno so manjše od 0,1), to je zaradi precejšnjega pomena naključnega iskanja rešitve. V zadnjih treh metodah se v eni generaciji zamenja polovica osebkov. Ostali parametri so specifični za vsako metodo posebej.

Opis metod:

1) Enoturnirska selekcija

   Turnir je sestavljen iz štirih osebkov, najboljša osebka ustvarita dva potomca s križanjem. Potomca nadomestita najslabša osebka v turnirski skupini. Za vsako generacijo se zvrsti 15 turnirjev.

2) Ruletna selekcija in lokalna elitna zamenjava

   Izbere se polovica (30) najboljših prednikov, ki se paroma ustvarijo po dva potomca. Temu rečem družina. Nato se družina oceni in dva najboljša posameznika preživita in v populaciji nadomestita prednika. Če sta potomca slabša od prednikov, do zamenjave ne pride.

3) Ruletna selekcija in zamenjava najslabših prednikov s potomci

   Polovica najboljših prednikov ustvari enako število potomcev, ki zamenjajo slabšo polovico posameznikov.

4) Rangirna selekcija in lokalna elitna zamenjava

   Metoda je zelo podobna drugi metodi, le da namesto ruletne selekcije zamenjujemo kandidate za razmnoževanje glede na njihov vrstni red v seznamu urejenem glede na kvaliteto

V tabeli 6 vidimo, kako se število generacij potrebnih za rešitev problema razlikuje pri različnih metodah in v različnih poizkusih. Za primerjavo sem izračunal povprečje in standardni odklon števila generacij.

št. poizkusa	končno število generacij			
	metoda 1	metoda 2	metoda 3	metoda 4
1	237	782	524	>1000
2	210	734	818	916
3	253	910	944	>1000
4	483	350	280	575
5	243	408	>1000	>1000
6	751	719	919	>1000
7	282	>1000	730	974
8	960	821	961	775
9	878	>1000	570	929
10	321	>1000	>1000	910
povprečje	461,8	772,4	774,6	907,9
standardni odklon	291,2	233,9	244,7	136,3

Tabela 6: Število generacij potrebnih za iskanje 16 znakovnega niza. Metoda 1 je enoturnirska selekcija, metoda 2 je ruletna selekcija in lokalna elitna zamenjava, metoda 3 je ruletna selekcija in zamenjava najslabših osebkov in metoda 4 je izbira z rangiranjem ter lokalna elitna zamenjava.

Število generacij za rešitev problema se pri posameznih poizkusih razlikujejo zaradi naključne inicializacije populacije stohastičnosti algoritma. S statističnim testom preverim značilnosti

razlik metode 1.

Postavim sledeči hipotezi:

H_0: *povprečje metode* 1 $\geqslant$ *povprečje metode* 2
H_1: *povprečje metode* 1 $<$ *povprečje metode* 2

Ničelna hipoteza pravi, da metoda 1 ni boljša od metode 2. To hipotezo bom preveril s statističnim testom. Za primerjanje sem izbral metodo 2, saj je zaradi najnižjega povprečja najboljša med ostalimi metodami.

S Fisherjevim F-testom poskušamo ovreči ničelno hipotezo, da sta varianci vzorcev enaki in prevzeti alternativno hipotezo, da sta varianci vzorcev *signifikantno* različni. Izračunana F-vrednost preizkusa je 0,52, kar je večje od stopnje zaupanja 0,05 in zato ničelne hipoteze ne zavržemo. V t-testu predpostavimo enakost variance (zaradi rezultata F-testa).

Z enostranskim t-testom vzorcev z enako varianco izračunamo *p-vrednost*=0,01, pri stopnji zaupanja $\alpha = 0{,}05$. Ker je $p - vrednost < \alpha$, ničelno hipotezo zavrnemo.

Torej je povprečje enoturnirske selekcije statistično manjše od ruletne selekcije, kar pomeni, da je to najboljša metoda od vseh preizkušenih za iskanje zaporedja znakov.

6.3. Interpretacija

Vidimo, da sta selekcija in preživetje pomembna dejavnika, ki vplivata na čas izvajanja programa. Pomemben je tudi način križanja. Korak mutacije (disperzija) ni bil vključen v genom kot strateški parameter, saj je v drugem primeru osebka predstavljen s celimi števili, in je prištevanje decimalnih števil nesmiselno. Mutacija je zato izbira enega izmed 256 možnih znakov.

Pri iskanju minimuma funkcije v poglavju 6.1. bi lahko uvedel korak mutacije kot strateški parameter, a sem ga zaradi enostavnosti izpustil.

Za izbiro dobrih parametrov je potrebno problem dobro poznati in ugotoviti, kateri dejavniki najbolj vplivajo na hitrost konvergence in tudi kako ohranjati raznolikost populacije.

Poglavje 7

Evolucija nevronskih mrež

To poglavje je najpomembnejše, saj tu prikažem delovanje končnega programa, to je evolucija nevronskih mrež. Napredovanje oz. razvoj nevronskih mrež umetnih mikroorganizmov se kaže v večanju števila skritih nevronov in sinaps, kar pomeni boljše spominske in računske sposobnosti mreže. Program začne s populacijo nevronskih mrež enostavne strukture, to je mreža s samo enim skritim nevronom. Občasno prihaja do mutacij nevronskih mrež, ki spreminjajo strukturo. Pozitivne mutacije izboljšajo delovanje nevronske mreže (oz. fitnessa) in taki organizmi so naravno odbrani. Te lastnosti se prenesejo na potomce, ki imajo ponovno večje možnosti, da bodo naravno odbrani. Tako se potomci organizma, ki je bil mutiran razširjajo čez dobršen del populacije. Opisano delovanje je zgolj teoretično, saj se v simulaciji srečamo z nekaterimi problemi, zaradi katerih organizmi z boljšo strukturo nevronske mreže oz. večjim potencialom niso nujno naravno odbrani. Ti problemi so raznolikost lokalnega okolja, v katerem je bil organizem postavljen (lahko se več hrane nahaja v njegovi okolici) in hitrost organizma dana ob rojstvu. Zatorej bom v prvih nekaj podpoglavjih naredil numerične simulacije evolucije brez opisanih problemov, nato šele pravo grafično simulacijo razvoja organizmov v okolju s hrano.

7.1. Opis algoritma

Algoritem je podoben algoritmu NEAT [9], vendar sem do ideje prišel sam in šele nato ugotovil, da podobna rešitev že obstaja.

Imamo populacijo velikosti n, ki je razdeljena na turnirske skupine velikosti m. Posameznike naključno razporedimo po skupinah, kjer jih razvrstimo po kvaliteti. Izračun funkcije kvalitete je sestavljen iz dveh delov. V prvem delu nevronsko mrežo učimo na naključno izbrani učni množici, ki ustreza lokalnemu okolju organizma. V drugem delu izračunamo napako vseh organizmov na isti testni množici, kar predstavlja kvaliteto organizmov.

Najboljša organizma sta izbrana za razmnoževanje in ustvarjena potomca nadomestita dva najslabša organizma. Ostali organizmi se ne spremenijo. Pri križanju posamezniki podedujejo celoten genski material staršev preko rekombinacije genov. Bistven je gen, ki kodira povezave, kjer je zapisana inovacijska številka gena, vhodni nevron, izhodni nevron, vrednost uteži sinapse in možnost vklopa ali izklopa gena. Gen za nevron nosi informacijo o zaporednem številu nevrona in njegovo lokacijo (vhodna, skrita ali izhodna plast).

Inovacijska številka pove, kdaj v zgodovini se je gen pojavil zaradi mutacije. Če se enaka mutacija ponovi, je gen že identificiran (gene shranjujem v genski bazen). Organizem tako ne more imeti dveh genov za isto povezavo z različnimi parametri (utežmi), ker bi bilo to nesmiselno. Z mutacijami organizem pridobiva nove gene za nevrone ali povezave.

Pri križanju se potomcu dodeljujejo geni v naraščajočem zaporedju inovacijskih številk. Če sta inovacijski številki genov obeh staršev enaki, potomec naključno pridobi enega izmed genov (tako se prepreči podvojevanje genov).

V novi generaciji se iz genoma ustvarijo nevronske mreže potomcev, ki se znova začnejo učiti, kot so to počeli njihovi predniki, saj znanje in spomin nista dedna. Deduje se samo informacija zapisana v genomu posameznikov, to sta struktura nevronske mreže in začetne vrednosti uteži. Nekatere strukture in začetne uteži omogočajo hitrejše učenje in boljši končni

rezultat, zato se populacija s časom preko mehanizmov naravnega odbiranja in razmnoževanja razvija.

7.2. Numerične simulacije

V prvem delu podpoglavja bom pokazal, kakšna je optimalna struktura nevronske mreže organizmov za iskanje hrane. V drugem delu bom pognal simulacijo evolucije in videli bomo, kakšno strukturo algoritem najde. Nato bomo lahko primerjali strukturi med seboj in ugotavljali razlike.

7.2.1. Optimalna struktura nevronske mreže

Optimalna struktura nevronske mreže učinkovito išče hrano in nima velikega števila nevronov in sinaps. Nevronsko mrežo razdelimo na dva dela. Prvi del nevronske mreže je rekurziven in služi za iskanje vhodnega nevrona z najvišjo vrednostjo. Tako ugotovimo, pod katerim kotom je bil zaznan predmet, ki je najbližji organizmu. Evolucija tega dela nevronske mreže je zelo zahtevna, saj je učenje takšne mreže z vzvratnim pravilom otežkočeno, ker moramo rekurzivno mrežo razviti v večplastno naprej povezano mrežo. Plasti mora biti veliko (več kot 7), zato je učenje dolgotrajno. Ta del se med evolucijo ne spreminja.

Drugi del mreže je preslikava kota premika organizma, ki ustreza vhodnemu nevronu, ki je zaznal najbližji predmet (razen v primeru, ko se organizem dotakne stene ali sosednjega organizma, v tem primeru je izhod nevronske mreže kot 45°). Optimalna struktura je narisana v poglavju 3.2.4. na strani 19.

Matriki idealne nevronske mreže za preslikavo kota (samo drugi del):

$$W_{input} = \begin{bmatrix} w_{1,1} & 0 & \cdots & 0 \\ 0 & w_{2,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ -w_{6,1} & -w_{6,2} & \cdots & w_{6,6} \end{bmatrix} \quad \text{in} \quad W_{output} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_6 \end{bmatrix}$$

Naredil sem nekaj različnih nevronskih mrež z enim skritim nevronom, jih učil na učni množici velikosti 100 s 500 iteracijami vzvratnega učenja in nato izračunal srednjo kvadratno napako na testni množici. To sem storil tudi za dva, tri, štiri, pet in šest skritih nevronov.

Mreža ne bo vedno polno povezana. Začetne uteži nevronskih mrež so pred učenjem enake utežem prejšnje enostavnejše strukture. Torej nevronska mreža s tremi skritimi nevroni in 13 utežmi ima v matriki W_{input} 12 uteži enakih kot nevronska mreža z dvema skritima nevronoma. Vsaka kompleksnejša nevronska mreža je zgrajena na osnovi enostavnejše, saj se tako izognemo problemu različnih konvergenc zaradi različnih začetnih uteži. V tabeli 7 lahko vidimo rezultate numerične simulacije.

Št. skritih nevronov	Št. vseh sinaps	Kvadratna napaka	Povprečna vrednost kvadratnih napak
1	1 + 1	1.21129224	0.836241
	2 + 1	1.23320600	
	6 + 1	0.06422724	
2	7 + 2	0.01765036	0.010314
	8 + 2	0.01103915	
	12 + 2	0.00225305	
3	13 + 3	0.00135750	0.007174
	14 + 3	0.01021135	
	18 + 3	0.00995407	
4	19 + 4	0.00213005	0.001997
	20 + 4	0.00227189	
	24 + 4	0.00159123	
5	25 + 5	0.00389982	0.005266
	26 + 5	0.00541418	
	30 + 5	0.00636627	
6	31 + 6	0.01101406	0.011848
	32 + 6	0.00477756	
	36 + 6	0.01975366	

Tabela 7: Izmerjene kvadratne napake nevronskih mrež različnih topologij. Drugi stolpec pomeni n + m sinaps, kjer je n sinaps iz vhodnega na skriti nivo in m sinaps iz skritega na izhodni nivo.

Ko je število sinaps 1 ali 2 je delovanje nevronske mreže napačno, saj vsi vhodni nevroni niso povezani s skrito plastjo. Izmerjena kvadratna napaka nam služi samo za primerjavo.
Zanimivo je, da se delovanje nevronske mreže ne izboljšuje nujno z večanjem števila skritih nevronov in povezav. Najboljše nevronska mreža deluje s tremi skritimi nevroni in 13 sinapsami (plus 3 izhodne). Zelo dobro deluje še s štirimi skritimi nevroni in 24 sinapsami. Če pogledamo stolpec povprečnih vrednosti za posamezna števila skritih nevronov vidimo, da povprečna kvadratna napaka upada in doseže najmanjšo vrednost, ko je število skritih nevronov 4, nato pa začne naraščati.
Sklepam, da bo nevronska mreža dajala najboljše rezultate pri štirih skritih nevronih in da se bo evolucija na tem koraku ustavila.

Vrstni red	Št. skritih nevronov	Št. vseh sinaps	Kvadratna napaka
1	3	13 + 3	0.00135750
2	4	24 + 4	0.00159123
3	4	19 + 4	0.00213005
4	2	12 + 2	0.00225305

Tabela 8: Po vrstnem redu glede kvadratne napake razporejene nevronske mreže. Vidimo, da je četrta najboljša mreža s samo dvema skritima nevronoma, ki je le za malenkost slabša od nevronske mreže s štirimi skritimi nevroni.

7.2.2. Evolucijsko iskanje optimalne strukture

V tem podpoglavju iščem optimalno strukturo nevronske mreže z genetskim algoritmom. Dano imam začetno populacijo enostavnih nevronskih mrež s samo enim skritim nevronom in utežmi, ki se razlikujejo. Uteži in struktura nevronske mreže, to je povezave med nevroni so del genskega zapisa. Ob rojstvu vsakega organizma se informacija v genomu uporabi za inicializacijo nevronske mreže organizma. Organizmi tekmujejo v različnih okoljih in se izboljšujejo z vzvratnim pravilom učenja. Selekcija je enoturnirska, kjer je velikost turnirja 4. Dva najboljša ustvarita potomca s križanjem in mutacijo. Potomca nadomestita najslabša prednika v turnirju.

S križanjem potomec prejme celotno strukturo nevronskih mrež prednikov, zato je njegova mreža najmanj toliko kompleksna kot mreža prednikov. Mutacija lahko dodatno doprinese k kompleksnosti.

Polovica najboljših posameznikov preživi boj za obstanek in ustvari potomce, ki nadomestijo polovico najslabših posameznikov. Potomci s pozitivnimi mutacijami so naravno odbrani.

Zanimivo je, da mutacije nekaj časa povečujejo kompleksnost od neke točke naprej pa mutacije ne povečujejo pozitivnih lastnosti. Potomca posameznikov z različnima topologijama mrež nimata nujno boljših lastnosti od staršev, kar je posledica povečane kompleksnosti, saj podedujeta vse gene od obeh staršev, ki kodirajo strukturo nevronske mreže.

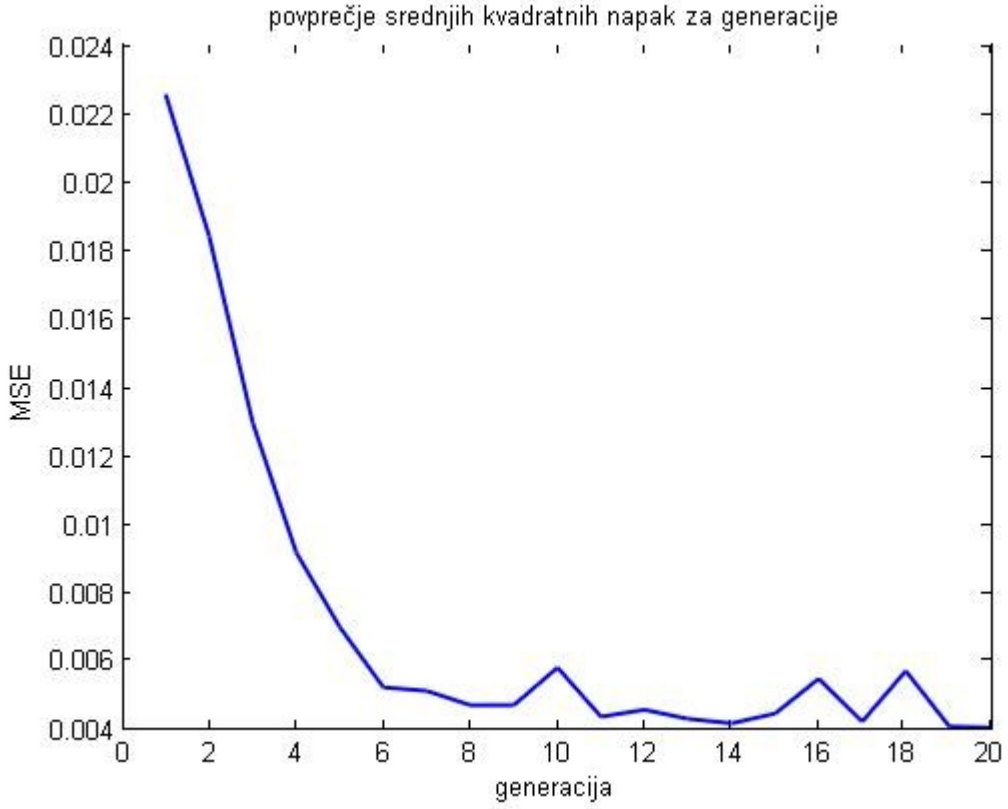
Uvedel sem tudi možnost izražanja genov. To pomeni, da lahko preko mutacij ali križanja nekatere gene izklopimo ali vklopimo, kar je enako kot brisanje ali dodajanje sinaps v nevronski mreži. Na takšen način lahko kompleksno nevronsko mrežo, ki se pod pritiskom naravne selekcije ne razvija več, poenostavimo in omogočimo iskanje potencialno boljše nevronske mreže. To se v naravi tudi dogaja, ko gre razvoj za nekaj korakov nazaj in potem začne novo pot, ki je morda uspešnejša.

Najprej ponazorim primer brez možnosti izklapljanja genov in sinaps. Velikost populacije je 20 osebkov, velikost turnirja pa 4 osebki. Imamo učno množico velikosti 100 posameznikov. Mutira eden izmed potomcev na turnir. Verjetnost mutacije je zelo velika, a na takšen način populacija hitreje konvergira k rešitvi. Število generacij je 20. Nevronska mreža ima v začetni generaciji samo 1 skriti nevron.

Ilustriram dva programa, kjer je eden uspel drugi pa ne. Za uspešnost je med drugim pomembna sestava začetne generacije.

7.2.2.1. Primer manj uspešne evolucije

Prvi poskus je bil manj uspešen kot nekateri ostali, saj program v 20 generacijah najde mrežo s samo 2 skritima nevronoma in 13 sinapsami. Srednja kvadratna napaka je enaka 0,003746. Najprej si na sliki 19 poglejmo, kako se srednja kvadratna napaka izboljšuje z generacijami.



Slika 19: Povprečje srednjih kvadratnih napak nevronskih mrež v evolucijskem razvoju.

Povprečna kvadratna napaka doseže nizko vrednost že po šestih generacijah niha okoli te vrednosti. Povišane povprečne kvadratne napake so posledica zamenjave najslabših prednikov v turnirju s še slabšimi potomci (ni lokalne elitne zamenjave).

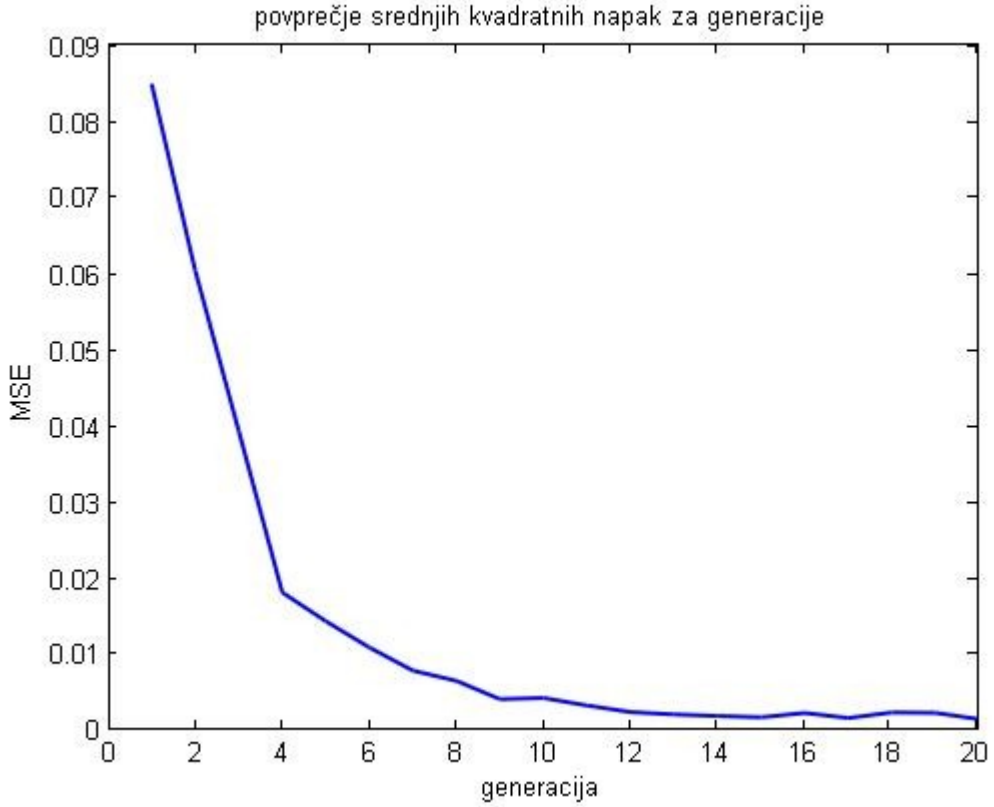
Poskus je bil relativno neuspešen, saj je povprečna kvadratna napaka 20 generacije enaka 0,00406 in je razmerje med povprečno kvadratno napako prve in zadnje generacije enako 5,49. Razmerje si lahko predstavljamo kot izboljšavo, doseženo z evolucijo. V tabeli 9 vidimo nekatere podrobnosti dobljenih organizmov.

Generacija	Najboljši posameznik			Najslabši posameznik			Razmerje MSE med najslabšim in najboljšim
	Št. skritih nevronov	Št. sinaps	Srednja kvadratna napaka	Št. skritih nevronov	Št. sinaps	Srednja kvadratna napaka	
1	1	6 + 1	0.021931	1	6 + 1	0.022814	1,040
10	2	8 + 2	0.004914	2	10 + 2	0.006977	1,420
20	2	11 + 2	0.003746	2	12 + 2	0.005027	1,342

Tabela 9: Podatki o najboljših in najslabših posameznikih v generaciji med evolucijo.

7.2.2.1. Primer uspešne evolucije

Zaradi ugodne začetne inicializacije nevronskih mrež oz. genomov je evolucija po 20 generacijah našla nevronsko mrežo s srednjo kvadratno napako 0,001002. To je najboljša nevronska mreža, kar sem jih našel. Kvadratno napako v odvisnosti od generacije razvoja vidimo na sliki 20.



Slika 20: Povprečje srednjih kvadratnih napak nevronskih mrež v evolucijskem razvoju.

Srednja kvadratna napaka večinoma upada in prihaja le do majhnih nihanj. Kakovost generacij ves čas napreduje.

Struktura najboljše nevronske mreže v tem poskusu je dokaj kompleksna, saj ima 4 skrite nevrone in kar 26 sinaps.

Struktura najboljše nevronske mreže:

```
Vhodni nevroni: x1, x2, x3, x4, x5 in x6.
Skriti nevroni: h1, h2, h3 in h4.
Izhodni nevron: o.
Povezave med nevroni:
x1->h1, x2->h1, x3->h1, x4->h1, x5->h1, x6->h1, h1->o, x1->h2, h2->o, x5->h2, x4-
>h2, x3->h2, x2->h2, x6->h2, x2->h3, h3->o, x4->h3, x6->h3, x3->h3, x5->h3, x1-
>h3, x1->h4, h4->o, x3->h4, x6->h4, x4->h4.
```

V tabeli 10 so podrobnosti najboljših in najslabših posameznikov v generacijah med tekom evolucije. Če naredimo primerjavo s tabelo 9 ugotovimo, da je raznolikost med posamezniki v začetni generacijah precej večja v drugem poskusu, kajti razmerje med srednjo kvadratno napako najslabšega in najboljšega posameznika je enako 2,372, v prvem poskusu pa je enako 1,040, kar pomeni, da je bil najslabši organizem v prvem poskusu le malenkostno slabši od najboljšega.

Generacija	Najboljši posameznik			Najslabši posameznik			Razmerje MSE med najslabšim in najboljšim
	Št. skritih nevronov	Št. sinaps	Srednja kvadratna napaka	Št. skritih nevronov	Št. sinaps	Srednja kvadratna napaka	
1	1	6 + 1	0.076967	1	6 + 1	0.182556	2,372
10	3	17 + 3	0.0012113	3	14 + 3	0.008641	7,134
20	4	22 + 4	0.001002	4	19 + 4	0.002824	2,818

Tabela 10: Podatki o najboljših in najslabših posameznikih v generaciji med evolucijo.

Uspešnost evolucije je torej v veliki meri odvisna od začetne populacije, ki je naključna. Podobno sem ugotovil, ko sem iskal minimum funkcije, saj se je evolucija večkrat ustavila v lokalnem minimumu in ni našla globalnega. Tudi število generacij za iskanje danega zaporedja je zelo variralo, saj so bili nekateri začetni nizi (populacije) bolj podobni danemu nizu kot drugi.

7.3. Grafične simulacije

V podpoglavju opišem grafično simulacijo evolucije nevronskih mrež, kjer vidimo življenje in napredovanje organizmov. Da bi bilo razlike med prvo – enostavno generacijo in kasnejšimi naprednejšimi, bolje opazne, sem organizmom dodal še eno nevronsko mrežo, ki jo opišem.

Za enostavno uporabo programa je potreben uporabniški vmesnik, katerega glavne značilnosti opisujem. Sledi analiza posameznih zanimivih simulacij z različnimi parametri.

7.3.1. Nevronska mreža za preslikavo polarnih koordinat v kartezične

Nevronska mreža v prvem in drugem delu strukture izračuna kot premika. Topologija te mreže je opisana v poglavju 3.2.4. na strani 22. Hitrost premikanja je statična in jo organizem dobi ob rojstvu. V novem delu nevronske mreže želimo preslikati polarne koordinate hitrosti: $\{|\vec{v}|, \Theta\}$ v kartezične koordinate: $\vec{v}=\{v_x, v_y\}$. Te se nato neposredno uporabijo v implementaciji premikanja organizmov v simulaciji.

Preslikava je matematično definirana kot:

$$v_x = |\vec{v}| \cdot \cos\Theta$$
$$v_y = |\vec{v}| \cdot \sin\Theta \quad ,$$

kjer je $|\vec{v}|$ absolutna vrednost hitrosti in Θ kot.

Pretvorba koordinat je dejansko računanje sinusa, kosinusa in množenje. Ker je množenje težko za nizko nivojsko nevronsko mrežo, vsebuje tretji del mreže le računanje sinusa in kosinusa kota. V programu se ti dve vrednosti, ki sta izhod nevronske mreže, pomnožita z absolutno vrednostjo hitrosti.

Nevronska mreža si zapomni vrednosti funkcij sinusa in kosinusa na osnovi učne množice. Ko med delovanjem dobi nepoznan vhod, interpolira naučene točke. Mreža potrebuje dovolj veliko število skritih nevronov, da si lahko v sinaptičnih povezavah zapomni dovolj točk, potrebnih za dobro interpolacijo funkcije.

Ker imajo organizmi v prvi generaciji samo en skriti nevron v vsaki plasti nevronske mreže, bo organizem slabo usmerjal svojo pot zaradi slabega izračuna sinusa in kosinusa.

S časom bodo predrugačenja mreže prinesla boljše delovanje in napredek organizmov bo vizualno bolj opazen.

7.3.2. Uporabniški vmesnik

Za večjo uporabnost simulacije, je bilo potrebno narediti razumljiv grafični uporabniški vmesnik, ki omogoča vnos parametrov in spremljanje simulacije, to so razni grafi, statistike, struktura nevronske mreže itd.

Implementacija vmesnika je zapletena zaradi uporabe procesov za vsakega izmed modulov programa in komunikacije med procesi. V nadaljevanju se na podrobnosti vmesnika ne bom osredotočal, saj za samo uporabo niso pomembne.

Parametri, ki jih uporabnik lahko spreminja so:

- velikost populacije,
- količina hrane v okolju,
- poraba hrane na sekundo,
- medgeneracijski čas,
- pričakovano število mutacij na generacijo.

Možnosti spremljanja simulacije:

- količina zaužite hrane na generacijo in na posameznika (povprečje),
- graf približkov funkcije sinus in kosinus,
- umrljivost,
- struktura nevronske mreže,
- povprečno in največje število nevronov ter povezav na generacijo.

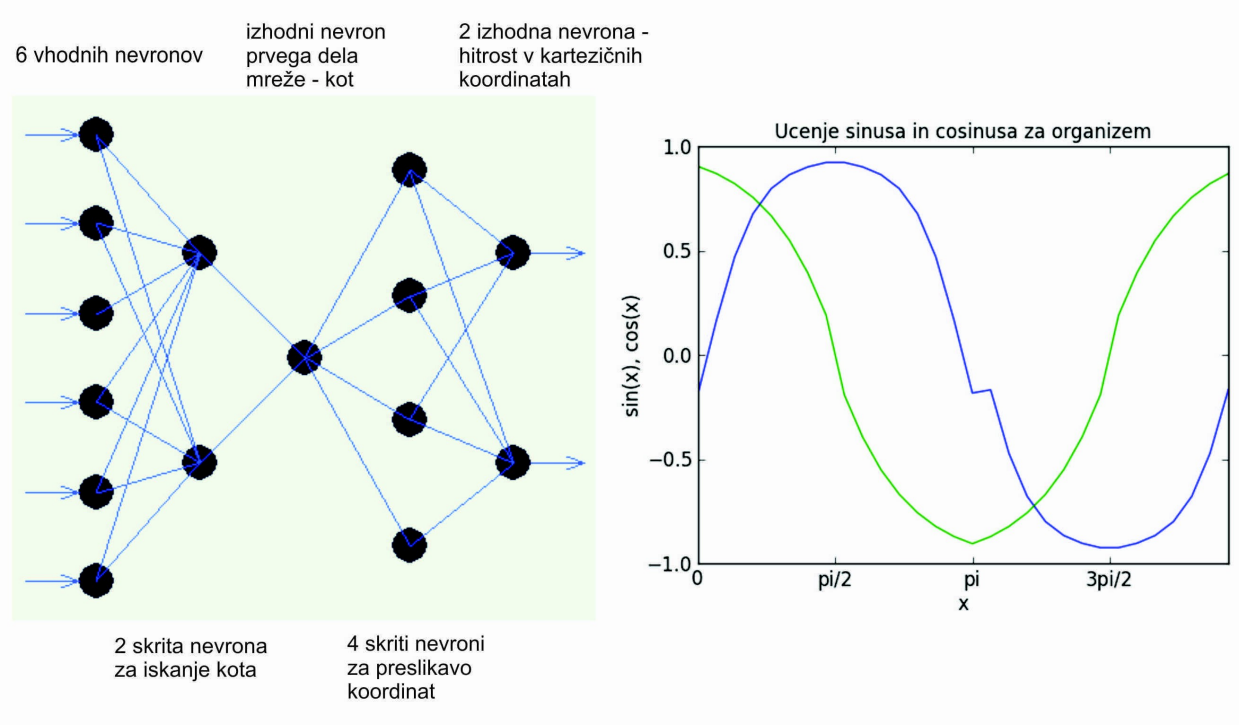
Ukazi za upravljanje simulacije:

- zagon,
- zaustavitev,
- ponastavitev parametrov,
- izhod.

Na začetku vsake generacije se naključno izbere nov organizem izmed potomcev, ki ga analiziramo. To se naredi zato, ker želimo spremljati organizem, ki je drugačen po genotipskih in posledično fenotipskih lastnostih. Za boljši vpogled v celotno populacijo pa rišemo grafa povprečnega in največjega števila nevronov ter povezav za vsako generacijo.

7.3.3. Simulacija razvoja strukture nevronske mreže

Simuliral bom razvoj nevronske mreže na začetni populaciji 20 organizmov, 30 enot hrane, poraba 0,1 hrane na sekundo, medgeneracijski čas 30 sekund in pričakovano število mutacij en gen na generacijo. Zanima nas topologija nevronske mreže po npr. 20 generacijah. Simulacija 20 generacij traja 10 minut.

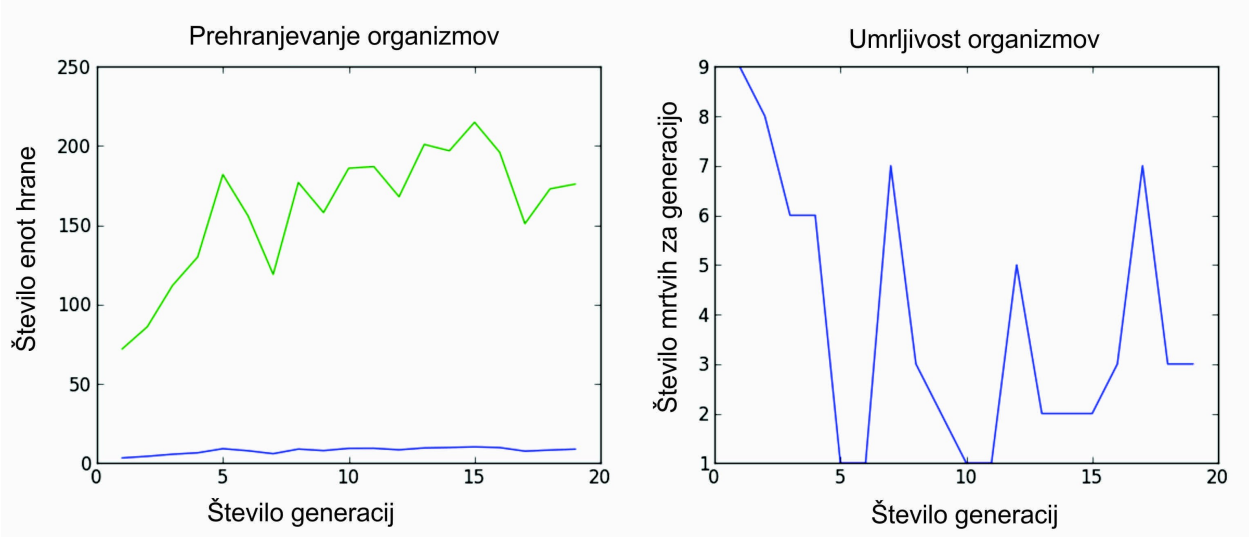


Slika 21: Nevronska mreža po 20 generacijah evolucije in pripadajoči izračun sinusa ter kosinusa. Zelena črta predstavlja je kosinus in modra sinus.

Na sliki 21 je prikazana nevronska mreža organizma 20 generacije. Ta generacija je bila uspešna, saj je dolgo časa ohranila. Izračun sinusa in kosinusa je za potrebe organizma zadovoljiv.

7.3.4. Simulacija prehranjevanja in umrljivosti – manj uspešen primer

Izbral sem enake parametre kot v prejšnjem razdelku, le da spremljam porabo hrane na generacijo, za posameznike (povprečje) in umrljivost posameznikov v generacijah. Umrljivost je definirana kot število posameznikov, ki umre v času ene generacije, zato ker ne najde dovolj hrane. Pričakujem, da bodo posamezniki vedno bolje iskali hrano in se bo zato umrljivost zmanjšala.

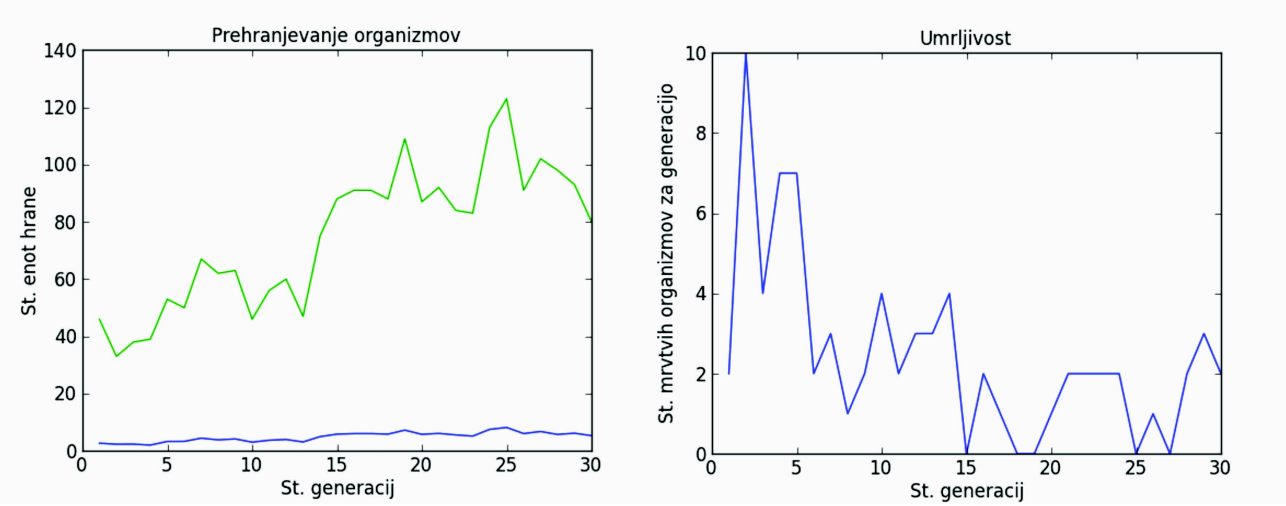


Slika 22: Grafična prikaza prehranjevanja in umrljivosti organizmov po 19 generacijah evolucije. Na levem grafu zelena barva prikazuje nabrano hrano za celotno generacijo, modra pa povprečje za posameznike.

Na levem grafu na sliki 22 vidimo, da se prehranjevanje organizmov izboljšuje, a z velikimi nihanji. Torej naučeni in kompleksnejši organizmi le učinkoviteje iščejo hrano. Na desnem grafu na pa opazimo, da umrljivost organizmov ne sledi nobenemu pravilu, saj so nihanja velika. V peti, šesti, deseti in enajsti generaciji umrljivost doseže najmanjšo vrednost, a ta kasneje skokovito naraste. Nihanje je pričakovano, saj umrejo organizmi prejšnjih generacij - predniki. Zakaj trend ne gre k zmanjšanju umrljivosti, je zanimivo vprašanje. Del odgovora je, da organizem ne razvija hitrosti premikanja. Menim tudi, da se ustvarijo velike razlike v sposobnostih in boljši organizmi "kradejo" hrano slabšim. Kljub temu, da v boju za obstanek preživijo najboljši, se v populaciji opazijo razlike med posamezniki.

7.3.5. Simulacija prehranjevanja in umrljivosti – bolj uspešen primer

Naredil sem še simulacijo 15 organizmov, 25 enot hrane v okolju, medgeneracijski čas 20 sekund in 30 generacij. Rezultate si poglejmo na sliki 23.



Slika 23: Grafična prikaza prehranjevanja in umrljivosti organizmov po 30 generacijah evolucije. Na levem grafu zelena barva prikazuje nabrano hrano za celotno generacijo, modra pa za posameznika.

Na sliki 23 vidimo, da organizmi v povprečju bolje nabirajo hrano in zato tudi manj umirajo. Tukaj je izid evolucije relativno boljši. Kot smo že v veliko primerih opazili, je evolucija vedno vsaj do neke mere stvar naključja.

7.3.6. Epilog simulacij

Ker starša ustvarita dva potomca, se število posameznikov vsako generacijo podvoji. Število posameznikov je omejeno z največjo velikostjo populacije, ki je v mojih grafičnih simulacijah enako začetni velikosti populacije.

V vseh treh primerih je umrljivost manjša ali enaka ustvarjanju potomcev, zato je velikost generacij približno konstantna. Z drugimi besedami to pomeni, da organizmi preživijo boj za obstanek dolg tudi do 2h – 240 generacij, kar sem tudi preizkusil.

V vseh primerih so organizmi v prvih generacijah neuspešni pri iskanju hrane. Nekaj jih kljub temu preživi in ustvarijo potomce, ki iščejo hrano učinkoviteje. Zaradi naključja redno prihaja do nihanj v količini najdene hrane. Organizmi približno v dvajseti generaciji dosežejo najvišjo količino najdene hrane, pozneje okoli te vrednosti le nihajo. Nevronska mreža se dovolj razvije, da omogoča organizmu zaznati najbližjo hrano v vidnem polju in ga tja usmeriti. Zanimiv dodatek k nalogi bi bil razvijanje ostalih parametrov npr. večje vidno polje ali hitrost organizma, saj bi tako količina zaužite hrane na generacijo še naprej naraščala s časom.

Umrljivost je zaradi večjih nihanj in manj izrazitega trenda precej težje razložljiv pojav od količine zaužite hrane. Vsekakor pa se vsaj delno drži enakih mehanizmov kot učinkovitost iskanja hrane in je zato v nekaterih primerih približno obratno sorazmeren. Zanimivo bi bilo razvijanje še ostalih parametrov organizma, saj je umrljivost precej nepredvidljiva.

Poglavje 8

Sklep

V diplomski nalogi predstavim delovanje nevronskih mrež, evolucijskega računanja in razvijem aplikacijo, ki vizualizira evolucijo nevronskih mrež. V posameznih fazah postavim model in ga implementiram. Rezultat je program, ki služi kot študijski pripomoček za lažjo predstavo o umetnem življenju.

Raziščem nevronske mreže in se odločim za uporabo vzvratnega učnega pravila ter večnivojske nevronske mreže zaradi dobrega delovanja na testnih primerih in enostavne implementacije. Pokažem, kakšne so dobre vrednosti parametrov, kot npr. velikost učne množice ali število iteracij za kvalitetno učenje. Tri nivojsko nevronsko mrežo učim iskanja kota najbližjega predmeta in ugotovim, da je natančnost delovanja slaba. Naredim novo mrežo s kombinacijo rekurzivne in naprej povezane tri nivojske nevronske mreže. Natančnost delovanja je veliko boljša od prejšnjih mrež. Ugotovim, da ima učna množica večji vpliv na kvaliteto učenja nevronske mreže kot začetne uteži. Simuliram eno generacijo z različnimi parametri kot so količina hrane v okolju, število organizmov, poraba hrane za premikanje in število časovnih korakov (iteracij glavne zanke) med dvema zaporednima učenjema in opazim, da organizmi manj umirajo, ko se naučijo iskanja hrane.

Za evolucijsko računanje ugotovim, da je največ težav pri začetni inicializaciji populacije osebkov, v ohranjanju diverzitete populacije in v hitrosti konvergence k rešitvi problema. Razvijem nov model za evolucijo nevronskih mrež in najprej z numeričnimi simulacijami poiščem dobro strukturo nevronske mreže. Numerična simulacija najde izvrstno delujočo nevronsko mrežo s 26 povezavami in 4 skritimi nevroni. Z grafičnim prikazom sem učinkovito pokazal, kako so v prvi generaciji organizmi nesposobni in jih samo nekaj preživi. Po nekaj generacijah se opazi napredek in izumrtje slabše prilagojenih organizmov. Umrljivost se z generacijami zmanjšuje, a še vedno niha in nikoli ne pade na 0. Količina hrane, ki jo organizmi najdejo z generacijami praviloma veča in doseže neke vrste limito. Natančnost računanja oz. odločanja mreže je dovolj velika za uspešno iskanje hrane.

Uspelo mi je postaviti model, ki simulira osnovne zakonitosti narave in obnašanja organizmov, ter ga implementirati. To je temelj, ki ga je možno nadgraditi na zanimive načine. En način je model plenilca in plena. Plenilec, plen in "rastlinska" hrana so v trikotnem razmerju, kjer plen beži pred plenilcem in išče hrano, plenilca pa zanima samo plen, ki je njegova hrana. Po pripadnosti skupinam bi se razlikovali v barvah. Organizmi bi razvili nevronsko mrežo, ki bi bolje zaznavala določene barve, poleg tega bi se organizmi morali zavedati pripadnosti skupini. Organizmi bi poleg nevronske mreže razvijali tudi npr. orožje ali hitrost. Plenilci bi se lahko povezali v skupine, ki bi skupaj lovile plen učinkoviteje (npr. stisnili plen v obroč ali kot). Razvila bi se organizacija skupin in strategija lova.

Uspešnost evolucije mojega modela je zadovoljiva, saj organizmi v vsakem poskusu simulacije preživijo kljub hudemu boju za preživetje. Na takšen način se razmeram prilagodijo, kar si lahko predstavljamo kot rešitev iskanja dobre strukture nevronske mreže.

Literatura

1. Charles Darwin, *On The Origin fo Species by Means of Natural Selection, or The Preservation of Favoured Races in the Struggle for Life*, London, 1859
2. Andrej Dobnikar, Branko Šter, *Mehko računanje*, FRI, Ljubljana, 2008
3. Ben Krose, Patrick van der Smagt, *An Introduction To Neural Networks*, The University of Amsterdam, 1996
4. Klemen Kregar: *Analiza uporabe umetnih nevronskih mrež za potrebe klasifikacij v deformacijski analizi.* Diplomsko delo, FGG, Ljubljana, 2009
5. Zoran Kancler: *Razpoznavanje podpisov z nevronsko mrežo.* Diplomsko delo, FERI, Maribor, 2005
6. Dario Floreano in Claudio Mattiussi: *Bio-Inspired Artificial Intelligence*, MIT, USA 2008
7. Daniel Ashlock: *Evolutionary Computation for Modeling and Optimization*, Springer. University of Guelph, Canada, 2006
8. Johan Andersson, *Applications of a Multi-Objective Genetic Algorithm to Engineering Design Problems.* Linköping University, Sweeden, 2000
9. Kenneth O. Stanley, Risto Miikkulainen: *Evolving Neural Networks through Augmenting Topologies,* Massachusetts Institute of Technology, USA, 2002
10. Sandi Gec: *Tridimenzionalna vizualizacija genetskih algoritmov.* Diplomsko delo, FRI, Ljubljana, 2011
11. Mitja Peruš: *Biomreže, mišljenje in zavest.* Založba Satjam, Ljubljana, 2001.
12. Igor Kononenko: *Strojno učenje.* Založba FE in FRI, Ljubljana, 2005.
13. (januar 2012) *Neural Networks, History: The 40's to the present*. Dostopno na: http://www-cs-faculty.stanford.edu/~eroberts/courses/soco/projects/2000-01/neural-networks/History/history1.html
14. (februar 2012) *Artificial Neural Network*. Dostopno na: http://en.wikipedia.org/wiki/Artificial_neural_network
15. (marec 2012) *Evolutionary Computation*. Dostopno na: http://en.wikipedia.org/wiki/Evolutionary_computation
16. (april 2012) *15 Real-World Uses of Genetic Algorithms*. Dostopno na: http://brainz.org/15-real-world-applications-genetic-algorithms