Neuronska mreža u učenju kvadriranja brojeva

Špoljarić, Andreja

Undergraduate thesis / Završni rad

2015

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Mechanical Engineering and Naval Architecture / Sveučilište u Zagrebu, Fakultet strojarstva i brodogradnje**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:235:838695

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-19**

Repository / Repozitorij:

Repository of Faculty of Mechanical Engineering and Naval Architecture University of Zagreb

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

Završni rad

Andreja Špoljarić

Zagreb, 2015.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

Završni rad

Neuronska mreža u učenju kvadriranja brojeva

Mentor:
Prof. dr. sc. Dubravko Majetić

Student:
Andreja Špoljarić

Zagreb, 2015.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za završne ispite studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala i mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum 02 -03- 2015	Prilog
Klasa: 602-04/15-6/3	
Ur.broj: 15-1703-15-154	

ZAVRŠNI ZADATAK

Student: **Andreja Špoljarić** Mat. br.: 0035188655

Naslov rada na hrvatskom jeziku: **Neuronska mreža u učenju kvadriranja brojeva**

Naslov rada na engleskom jeziku: **Neural Network in Squaring Numbers**

Opis zadatka:

Zadatak je načiniti interaktivni program kojim se pokazuje način kreiranja datoteke učenja, provođenje samog učenja i testiranja naučene mreže, a za primjer kvadriranja cijelih brojeva. Rad treba poslužiti kao nastavno pomagalo za demonstraciju rada statičke neuronske mreže.

U radu je potrebno načiniti slijedeće:

1. Izvesti matematički model učenja statičke neuronske mreže s više različitih aktivacijskih funkcija temeljen na povratnom prostiranju pogreške.
2. U proces učenja uključiti metodu momentuma prvog i drugog reda.
3. Za zadani problem izraditi interaktivnu programsku podršku, s naglaskom na vizualizaciji i jednostavnom korištenju.
4. Programsku podršku načiniti u nekom od dostupnih matematičkih programskih paketa.
5. Izvesti zaključke rada.

Zadatak zadan:
25. studenog 2014.

Rok predaje rada:
1. rok: 26. veljače 2015.
2. rok: 17. rujna 2015.

Predviđeni datumi obrane:
1. rok: 2., 3., i 4 . ožujka 2015.
2. rok: 21., 22., i 23. rujna 2015.

Zadatak zadao:

Prof. dr. sc. Dubravko Majetić

Predsjednik Povjerenstva:

Prof. dr. sc. Zoran Kunica

Izjavljujem da sam ovaj rad izradila samostalno koristeći stečena znanja tijekom studija i navedenu literaturu.

Posebno se zahvaljujem voditelju rada prof.dr.sc. Dubravku Majetiću na prihvaćanju mentorstva, pružanju korisnih savjeta te stručne pomoći pri izradi rada.

Također zahvaljujem baki, roditeljima te kolegama na pruženoj podršci i potpori.

Andreja Špoljarić

Sadržaj:

Popis slika II
Popis tablica III
Sažetak IV
Ključne riječi IV
Summary V
Key words V
Popis oznaka VI
1. Uvod 1
1.1 Umjetne neuronske mreže 1
1.2 Biološki neuron 3
1.3 Umjetni neuron 4
1.4 Učenje umjetne neuronske mreže 6
2. Programski paket MATLAB 7
2.1 Rad u MATLAB-u 7
2.2 Grafičko korisničko sučelje (eng. Grafic User Interface GUI) 8
2.2.1 GUIDE (Graphical User Interface development enviroment) 9
3. Statička unaprijedna neuronska mreža s povratnim prostiranjem pogreške (eng. EBP- Error-Back Propagation) 11
3.1 Perceptron 11
3.1.1 Učenje perceptrona 12
3.1.2 Delta pravilo 13
3.2 Višeslojne neuronske mreže 14
3.2.1 Model statičkog neurona 14
3.2.2 Model statičke neuronske mreže 16
3.2.3 Učenje povratnim rasprostiranjem pogreške 17
3.2.3.1 Unaprijedna faza učenja statičke mreže 17
3.2.3.2 Povratna faza učenja statičke mreže 18
3.2.4 Ubrzanje iterativnog učenja 20
3.2.4.1 Zamah prvog reda 20
3.2.4.2 Zamah drugog reda 21
3.2.5 Promjena težina izlaznog sloja 21
3.2.6 Promjena težina skrivenog sloja 23
3.3 Ocjena točnosti algoritma učenja 24
4. Neuronska mreža za kvadriranje prirodnih brojeva 27
4.1 Princip rada mreže 27
4.2 Topologija mreže 30
4.3 Normiranje 30
4.4 Primjer rada mreže 31
5. Zaključak 36
6. Literatura 37

Popis slika:

Slika 1.1. Pojednostavljena struktura biološkog neurona 3
Slika 1.2. Struktura umjetnog neurona..... 5
Slika 2.1. Matlabov glavni radni prozor..... 8
Slika 2.2. GUIDE Layout Editor..... 9
Slika 2.3. Poravnavanje odabranih objekata..... 9
Slika 2.4. Pregled grafičkog sučelja za testiranje mreže 10
Slika 3.1. Mreža perceptrona..... 12
Slika 3.2. Model statičkog neurona..... 14
Slika 3.3 Bipolarna sigmoidalna funkcija..... 15
Slika 3.4 Sinusna funkcija..... 15
Slika 3.5. Model statičke unaprijedne neuronske mreže 16
Slika 4.1 Odabir brojeva za kvadriranje..... 27
Slika 4.2 Odabir parametara mreže..... 28
Slika 4.3 Prozor za iscrtavanje NRMS dijagram..... 28
Slika 4.4 Prozor za ispis koraka i vremena učenja..... 29
Slika 4.5 Prozor za testiranje mreže..... 29
Slika 4.6 Odabir brojeva za kvadriranje..... 31
Slika 4.7 Prikaz prozora za učenje mreže - 1000 koraka učenja..... 31
Slika 4.8 Testiranje mreže – 1000 koraka učenja..... 32
Slika 4.9 Testiranje mreže – 5000 koraka učenja..... 32
Slika 4.10 Prikaz prozora za učenje - 1000 koraka, moment 1. reda..... 33
Slika 4.11 Testiranje mreže – 1000 koraka, momentum 1. reda..... 33
Slika 4.14 Prikaz prozora za učenje - 1000 koraka, momentum 2. reda..... 34
Slika 4.15 Testiranje mreže – 1000 koraka, momentum 2. reda 34

Popis tablica:

Tablica 4.1 Rezultati učenja- sigmoidalna funkcija, proizvoljne težine..... 35
Tablica 4.2 Rezultati učenja- sinusna funkcija, proizvoljne težine 35

Sažetak:

U ovom radu prikazan je postupak učenja neuronske mreže s povratnim rasprostiranjem pogreške u zadatku prikaza generalizacijskog svojstva neuronske mreže za primjer kvadriranja prirodnih brojeva. Općenito gledano neuronske mreže spadaju u jedno mnogo šire područje nazvano umjetna inteligencija. Glavna karakteristika umjetnih neuronskih mreža je u tome što se one ne oslanjaju isključivo na determinsitičke matematičke postupke zbog čega svi ulazi u sustav ne moraju biti u potpunosti točni da bi on mogao pravilno raditi. Osim samog postupka učenja u ovom je radu prikazano i testiranje mreže koje se provodi nakon samog procesa učenja, a cijela programska podrška, uključujući i prikladno grafičko sučelje, načinjeni su pomoću programskog paketa MATLAB. Opis, učenje i glavne karakteristike umjetnih neuronskih mreža prikazane su i detaljno opisane u uvodu, a zatim u nastavku slijedi prikaz i način učenja i rada zadane mreže.

Ključne riječi

- neuronske mreže
- umjetna inteligencija
- umjetni neuron
- kvadriranje
- učenje neuronske mreže
- testiranje neuronske mreže
- težinski faktori
- zamah prvog reda
- zamah drugog reda
- povratno rasprostiranje pogreške

Summary:

This paper deals with the functioning of artificial neural networks with error-back propagation whose goal is to square numbers. Generally neural networks belong to much wider area that is called artificial intelligence. The main characteristic of artificial neural networks is that they do not rely solely on deterministic mathematical procedures for which all inputs into the system may not be completely accurate so that it could work properly. In addition to the learning process in this paper, showing and testing the network carried out after the learning process, and the entire program support, including appropriate graphical interface, are made using the software package MATLAB. Description, teaching and main characteristics of artificial neural networks are shown and described in detail in the introduction. After that a presentation of the learning mode and the default operation network is given.

Key words

- neural networks
- artificial intelligence
- artificial neuron
- squaring
- neural network learning
- neural network testing
- weighting factors
- first order momentum
- second order momentum
- regenerative propagation errors

Popis oznaka

Z - matrica ulaznih neurona
D - matrica izlaznih neurona
V - matrica težina sakrivenih nerona
W - matrica težina izlaznih neurona
BIAS - neuron bez ulaza konstantnog izlaza jednakog jedan
NRMS - normalizirani korijen srednje kvadratne pogreške - mjera točnosti

1. Uvod

Umjetne neuronske mreže dio su mnogo šireg područja, umjetne inteligencije. Kad govorimo u umjetnoj inteligenciji mislimo na znanosti koja se bavi razvojem računarskih tehnika za rješavanje zadaća povezanih s ljudskom inteligencijom. Do ideje razvoja neuronskih mreža došlo je nakon niza pokušaja modeliranja biofiziologije mozga čovjeka u svrhu lakšeg razumijevanja i objašnjenja njegovog funkcioniranja. Takav model trebao bi imati sposobnost pohrane i obrade informacija na analogan način kako to vrši ljudski mozak prilikom obrade podataka.

1.1 Umjetne neuronske mreže

Kad govorimo o umjetnim neuronskim mrežama, mislimo prije svega na složen sustav sastavljen od elemenata koje nazivamo neuronima. Svi elementi sustava, neuroni, u međusobnoj su interakciji i tako s okolinom sustava grade funkcionalnu cjelinu.

Prvi model umjetnog neurona, temeljne jedinice svake neuronske mreže, predložili su McCulloch i Pitts [4] 1943. godine prema tvrdnji Williama Jamesa [3] iz 1890. godine koja glasi: "Aktivnosti bilo koje točke mozga predstavljaju zbroj tendencija svih ostalih točaka da se "prazne" u nju. ".
Pojednostavljeno gledajući, ako određenu točku u mozgu čovjeka zamijenimo neuronom, onda aktivnost neurona možemo modelirati kao zbroj otežanih ulaza neurona, a pod time mislimo na ulaze pomnožene određenim faktorima koje nazivamo težinama neurona. Dakle, aktivnost svakog umjetnog neurona ovisi o broju ulaza iz okruženja koju još nazivamo okolinom neurona, zatim o intezitetu tih veza koje još nazivamo težinskim faktorima, te o pragu osjetljivosti koje stanje neurona mora dosegnuti prije nego što pošalje impuls preko svog izlaza u okolinu neurona. Okruženje neurona, u koje odlazi ispaljen impuls, čine ostali neuroni umjetne neuronske mreže sa ili bez okruženja te mreže.

Neuronske mreže razlikujemo prema strukturi veza među neuronima i neurona s okruženjem mreže , te po metodologiji određivanja tih veza, što predstavlja proces učenja mreže.

Umjetne neuronske mreže mogu biti jednoslojne i višeslojne. Glavna karakteristika jednoslojnih nerunskih mreža je da imaju jedan ulaz i jedan izlaz, a između skriveni sloj. U slučaju povezivanja slojeva neuronskih mreža tako da signal putuje samo u jednom smjeru, od ulaza prema izlazu, mislimo na unaprijedne neuronske mreže ili eng. Feedforward Neural Networks. Kod nekih neuronskih mreža može postojati i povratna petlja, znači mogućnost suprotnog smjera gibanja signala, pa tada govorimo o povratnim neuronskim mrežama ili eng. Feedback or Recurrent Neural Networks. Odabir neuronske mreže koja će se koristiti ovisi o problemu kojeg treba riješiti.

Najpoznatiji predstavnik umjetnih mreža je MADALINE-mreža [5], koju su prvi predstavili Widrow i Hoff 1962. godine, dok je najpoznatiji predstavnik povratnih neuronskih mreža

Hopfieldova neuronska mreža [6] nazvana po svom autoru Hopfieldu, prvi put predstavljena 1984. godine.

Neuronske mreže imaju i nazive ovisno o metodi koju koriste za učenje, pa se tako razlikuju povratno propagirane, suprotno propagirane i statičke neuronske mreže. Također, neke se zovu prema svojim autorima, pa tako imamo Kohonenove i već spomenute Hopfieldove neuronske mreže. Različit je i kriterij implementacije neuronskih mreža, pa tako imamo i podjelu na softverske, hardverske te optičke. Važna je i podjela ovisno o području primjene neuronskih mreža, pa tako imamo perceptronske, asocijativne, dvostruko asocijativne, adaptivne te kognitronske i neokognitronske neuronske mreže.

Uz sve navedene podjele postoji i podjela na vremensko-kontinuirane i na vremenski-diskretne neurosnke mreže. Ovakva sistematizacija je nužna zato što uvođenjem povratne petlje u mrežu treba uključiti i vrijeme.

Kod implementacije neuronskih mreža može se koristiti analogna tehnika koju koristimo kod povratnih mreža, digitalna tehnika vezana za unaprijedne mreže, hibridna tehnika, tehnika implusne modulacije i optoelektronika. Analognu tehniku karakterizira visok stupanj kompaktnosti i velika brzina rada, uz manju preciznost i podložnost neželjenoj izmjeni parametara i smetnjama. Veća preciznost i mogućnost izmjene parametara glavna je karakteristika digitalne tehnike, dok je nedostatak u povećanju dimenzija sklopova. Prednost optoelektroničke implementacije je mogućnost promjena težina mreže, dok je nedostatak opto-elektronička i elektro-optička transformacija signala.

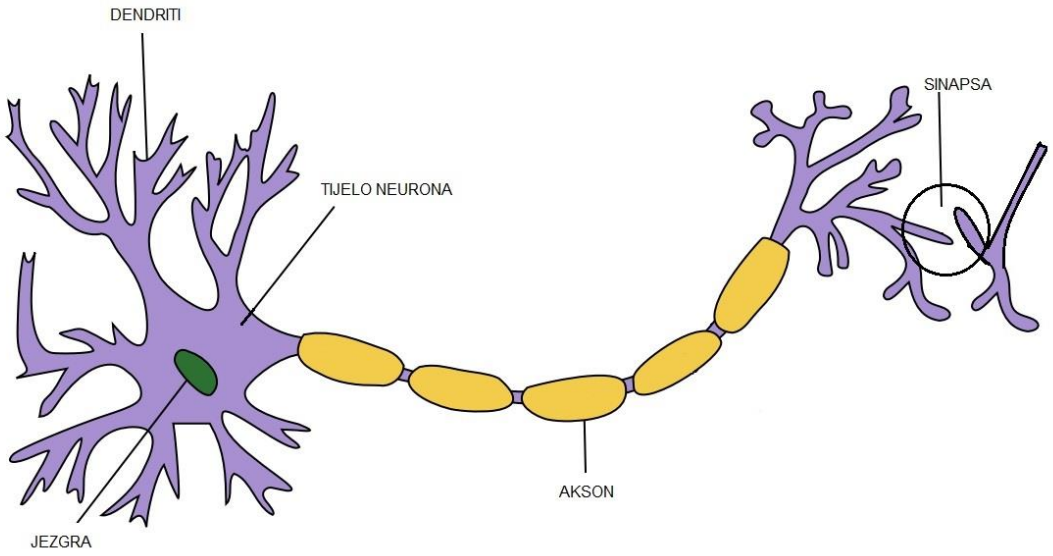
Danas u svijetu nalazimo dva različita pristupa modeliranja umjetnih neuronskih mreža. Prvi takav pristup ima kao glavni cilj realizaciju modela koji dovoljno točno oponašaju aktivnosti mozga čovjeka, što u perspektivi rezultira sustavom umjetne inteligencije. Drugi pristup podrazumijeva razvoj umjetnih neuronskih mreža koje će imati velike računske sposobnosti koje će biti orijentirane rješavanju konkretnih zadataka praktične prirode. Između ova dva pristupa nema stroge granice, jer oba koriste zajedničke elemene.

Iako je razvoj neuronski mreža započeo relativno nedavno, metode koje su se razvile iz njih nalaze široku primjenu kako u tehničkim, tako i u društvenim znanostima. Tako neuronske mreže danas primjenjujemo kod procesa optimiranja, linearnog programiranja, komuniciranja i donošenja odluka i zaključaka. U medicini neuronske mreže koristimo kod dijagnosticiranja te kod ultrazvučnog prikaza. Također, neuronske mreže mogu se primijeniti i kod gramatičkog zaključivanja, a i u vremenskoj prognozi.

Jedno od glavnih područja primjene neuronskih mreža je robotika i to zahavaljujući njihovoj mogućnosti paralelnog procesuiranja signala s ulaza na izlaz mreže. Do danas su razvijene mreže za identifikaciju dinamike robota, točnije kod rješavanja inverznog kinematičkog problema, zatim kod automatskog planiranja trajektorija robota, optimalnog i adaptivnog vođenja robota u prostoru i vremenu itd [7].

1.2 Biološki neuron

Nakon brojinih istraživanja došlo je do zaključka da postoje velike sličnosti između umjetnog i biološkog neurona, zbog čega u nastavku slijedi detaljniji prikaz svakog od njih [1,3].



Slika 1.1. Pojednostavljena struktura biološkog neurona

Ako gledamo pojednostavljeno, vidimo da se biološki neuron sastoji od tijela neurona, aksona i dendrita.

Akson možemo zamisliti kao tanku cjevčicu na kojoj se na jednom kraju nalazi tijelo neurona, a na drugom mnoštvo grana koje potom dodiruju dendrite drugog neurona. Mali razmak između završetka aksona prethodnog neurona i i dendrita ili tijela sljedećeg neurona naziva se sinapsa. Impuls putuje kroz akson, koji formira sinaptičke veze s mnoštvom drugih neurona, sve do sinapsi odakle se signali različitog intenziteta šalju kroz dendrite direktno na tijelo drugih neurona. Neuron prenosi svoj impuls kroz akson samo ako je njegova uzbuda veća od smirujućeg utjecaja na kritični iznos, koji predstavlja prag osjetljivosti neurona.

Prema procjeni mozak čovjeka ima preko 100 milijardi neurona, koji su naravno mnogo složeniji nego na ovom prikazu. Takvi, biološki neuroni, komuniciraju preko neuronskih vlakana, gradeći na taj način preko 100 trilijuna veza , odnosno sinapsa. Upravo ta mreža neurona je odgovorna za naše razmišljanje, učenje, emocije , strpljenje, percepciju i slične aktivnosti. Kako točno radi biološka neuronska mreža još uvijek nije točno razjašnjeno.

Do sad je otkriveno nekoliko stotina bioloških neurona koji se međusobno razlikuju prema svojem obliku, a upravo taj oblik određuju njihovu funkciju. Na žalost, funkcije tih neurona do danas nisu u potpunosti poznate, ali zato znamo da jedan neuron u nekoj neuronskoj mreži može aktivirati ili smiriti ako treba i tisuće drugih neurona, a istovremeno

svaki od njih može uzbuđivati tisuće drugih neurona. Upravo taj podatak nam govori kolika je zapravo složenost ovakvih mreža i veza između pojedinih neurona koji grade takav sustav.

Kada se usmjerimo na mozak, vidimo da se komunikacija u njemu odvija kemijskim signalima preko sinapsi i električnim signalima unutar neurona. Upravo zahvaljujući električnim signalima možemo primjetiti određenu vezu između ljudskog mozga i računala. Uz to i mozak i računalo imaju veliki broj jednostavnih elemenata, a i izvode funkcije koje se općenito nazivaju računskim funkcijama. Međutim, postoji bitna razlika između ljudskog mozga i računala, a to je da za rješavanja nekog problema mozak uči, a računalo treba programirati. Dakako, bitne su i razlike u rezultatima, točnije u ispravnost rezultata kojeg će dati čovjek i onog kojeg će dati računalo. Jasno je da je računalo manje sklono pogrškama, a uz to će brže i bez umaranja dati konačno rješenje. Ljudski mozak također ima prednosti , a to su veće mogućnosti zaključivanja i bolja aproksimacija i kod nekompletnih ulaza.

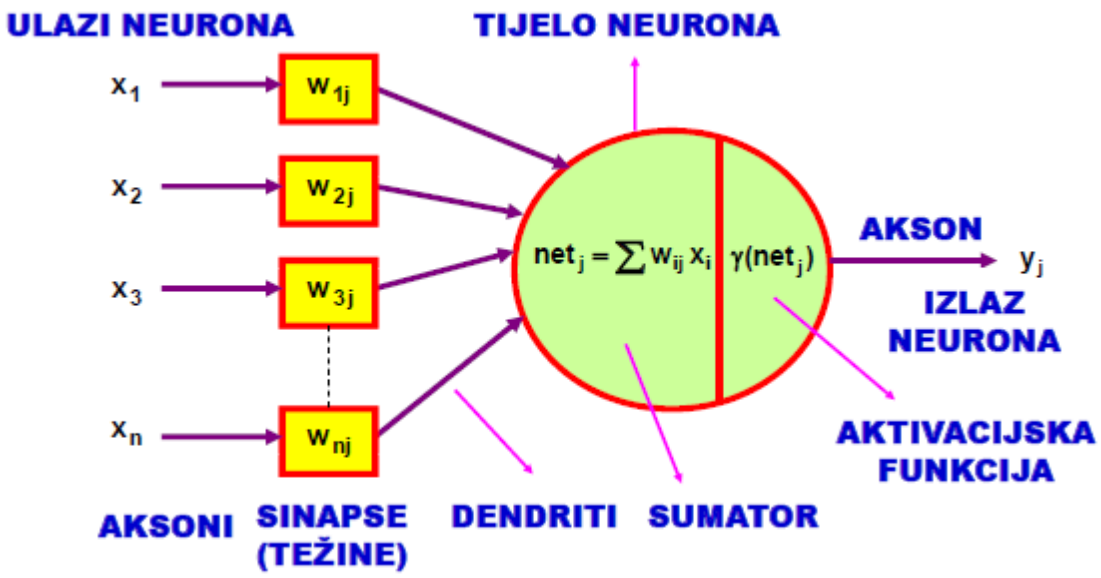
Pretpostavlja se da će u budućnosti umjetne neuronske mreže moći oponašati funkcije mozga čovjeka, ali da bi to ostvarili treba nam za početak dublji uvid u funkcije koje ljudski mozak obavlja, što još nije u potpunosti razumljivo i do kraja istraženo.

1.3 Umjetni neuron

Glavna ideja vezana za umjetni neuron je da u konačnici oponaša osnovne funkcije biologškog neurona. Da bi se iz biološkog neurona dobio umjetni neuron potrebno je tijelo biološkog neurona zamijeniti sumatorom , zatim ulogu dendrita na sebe trebaju preuzeti ulazi u sumator, dok je izlaz iz sumatora akson umjetnog neurona [1, 3].

Što se tiče praga osjetljivosti bioloških neurona, on se kod umjetnog neurona preslikava u aktivacijske funkcije. Funkcije sinaptičke veze bioloških neurona s njegovom okolinom preslikavaju se na težinske faktore, preko kojih se ostvaruje veza umjetnog neurona s njegovom okolinom. Težinski faktori mogu biti pozitivan i negativan broj, a kod modernih mreža može biti i funkcija. U slučaju da je težniski faktor jednak nuli, tada veze sa okolinom neurona nema, te se težinski faktor na shemi neuronske mreže ne ucrtava.

Općenito gledano, težinski faktori imaju istu ulogu kao i sinapse, točnije oni povezuju izlaze iz drugih neurona i iz okoline neurona sa ulazima sumatora koji su kod biološkog neurona nazvani dendritima, a intenzitet veze ovisi o iznosu, dok karakter ovisi o predznaku težinskog faktora.



Slika 1.2. Struktura umjetnog neurona [1, 4]

Izlaz iz sumatora povezuje se na aktivacijske funkcije koje mogu biti linearne i nelinearne. Linearne karakterizira množenje izlaza sumatora s nekim faktorom pojačanja kako bi se dobio izlaz neurona, dok nelinearna funkcija provodi izlaz sumatora na izlaz neurona preko nelinearnog pojačanja. Nelinearne aktivacijske funkcije mogu biti različitog oblika poput funkcije praga osjetljivosti, sigmoidalne, hiperboličke i harmoničke funkcije.

Drugi naziv umjetnog neurona je McCulloch-Pittsov neuron [4], a naziv potječe od njegovih autora. U slučaju da je aktivacijska funkcija oblika praga osjetljivosti , opisani neuron naziva se još Perceptron.

Ovakva umjetni neuron zanemaruje mnoge karakteristike biološkog neurona, jer ne vodi računa o kašnjenju signala što djeluje na dinamiku sustava, iako to daje veću brzinu procesuiranja ulaznih signala mreže što je opet dobra karakteristika. Također, ovaj neuron uključuje efekt sinkronizacije, funkcije frekventne modulacije biološkog neurona, koja je izrazito bitna po mišljenju znanstvenika.

Pravilnim povezivanjem umjetnih neurona u mrežu možemo ostvariti funkcije koje sam neuron ne bi bio u stanju realizirati.

1.4 Učenje umjetne neuronske mreže

Razlikujemo učenje neuronskih mreža uz nadzor i bez nadzora, točnije supervizorno i nesupervizorno učenje.

Glavna karakteristika supervizornog učenja je u tome što neuronska mreža zahtijeva vanjskog učitelja, koji promatra ponašanje mreže i ima mogućnost njene korekcije sve dok ne dobije željeno rješenje. Kod takvog učenja najprije se usvoji određena struktura u koju spadaju broj ulaza, broj neurona, broj slojeva, broj izlaza te broj težina mreže. Najprije se preko generatora slučajnih brojeva usvoje početne težine neuronske mreže, a zatim se na ulaz dovodi skup ulaznih varijabli. Mreža na temelju tih podataka producira određen skup izlaznih varijabli, te ga uspoređuje sa željenim skupom izlaznih varijabli. Razlika željenih i stvarnih izlaza neuronske mreže gradi pogrešku mreže, koju korisitmo kod računanja novih težina preko usvojenog algoritma. Cijeli postupak ponavlja se interakcijski dok pogreška mreže ne bude manja od unaprijed zadanog iznosa.

Nakon procesa učenja slijedi testiranje neuronske mreže, a to se radi s novim skupom ulaza neuronske mreže koji nije bio zadan u ulaznom skupu za vrijeme procesa učenja. Mreža nakon procesa testiranja izbacuje nove izlaze koji se potom uspoređuju sa željenim izlazima, a da se pritom ne mijenja parametar mreže. Iznos pogreške koja se javlja u procesu testiranja služi za procjenu robusnosti mreže, točnije njenoj sposobnosti da daje zadovoljavajuće rezultate i za onaj skup učenja kojim nije bila učena.

Kod nesupervizornog učenja nema vanjskog učitelja , pa se u ovom slučaju mreža sama organizira te se zato ovakve mreže nazivaju još i samoorganizirajuće mreže. Na ulaz mreže dovodi se skup ulaznih varijabli, a mreža se potom samoorganizira podešavanjem svojih parametara, težina, po definiranom algoritmu. Rezultat učenja nije u ovom slučaju predvidiv upravo zbog toga što željeni izlazi nisu specificirani za vrijeme učenja. I u ovom slučaju, nakon učenja provodi se proces testiranja.

Važno je spomenuti i različite algoritme učenja kojih ima cijeli niz, a neki od najvažnijih su generalizirano delta pravilo, povratno propagiranje, Grossbergovo učenje, optimalno učenje, adaptivno učenje i drugi.

2. Programski paket MATLAB

Matlab je high-performance programski jezik koji se najčešće primjenjuje kod tehničkih proračuna. Glavna karakteristika mu je ta što objedinjuje računanje, vizualizaciju i programiranje koje se vrši u lako uporabljivoj okolini. Ovaj program možemo koristiti kod rješavanja različitih matematičkih problema i izračuna, kod razvoja algoritama, modeliranja, simulacije, analize, razvoja aplikacija i drugih sličnih djelatnosti.

Prva verzija ovog programskog paketa potječe iz 1970. godine, napisana na sveučilištima Universitiy of New Mexico i Stanford University [2], s ciljem da nađe širu primjenu u matričnoj teoriji, linearnoj algebri te numeričkoj analizi. Budući da se od samih početaka ovaj program temeljio na kompleksnoj matrici kao osnovnom tipu podataka, imenovan je kraticom od **Mat**rični **lab**oratorij. Još jedno bitno svojstvo mu je to da se može povezati sa programom pisanim u C jeziku ili Fortranu [2], a upravo zbog te karakteristike može se koristiti kod vrlo složenih zadataka. Zahvaljujući formi koja je slična našem zapisivanju matematičkih formula, jednim redom u Matlabu zamjenjujemo stotine redaka napisanih u programskim jezicima opće namjene, zbog čega ga smatramo jezikom vrlo visoke učinkovitosti što se tiče tehničkog računanja.

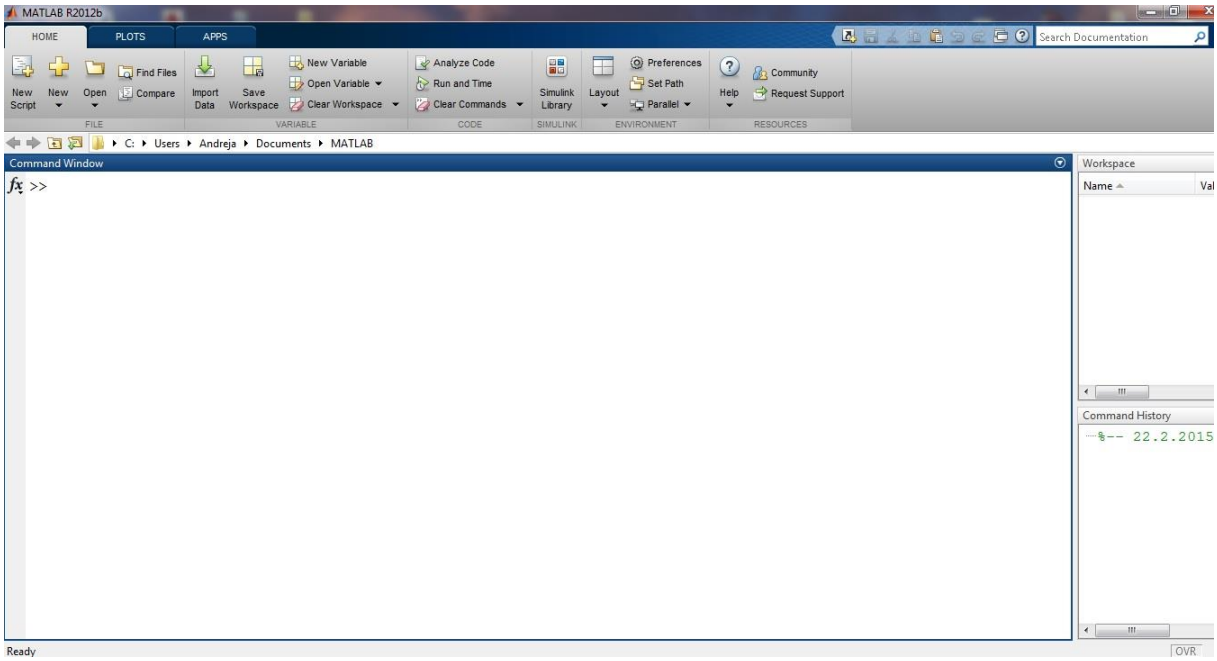
Jedna od glavnih karakteristika ovog programskog paketa je i ta što omogućava izgradnju različitih vlastitih alata za višekratnu uporabu. Funkcije i programi uz pomoć ovog programskog paketa mogu se vrlo lako kreirati, a njihovim spremanjem dobivamo m-datoteke. Skup specijaliziranih m-datoteka za rad u određenoj klasi je Toolbox. Svaki Toolbox kojim Matlab raspolaže rezultat je istraživanja vrhunskih stručnjaka iz područja upravljanja, obrade signala, identifikacije procesa i sličnih.

U odnosu na druge programske pakete, Matlab karakterizira elegancija, preglednost, praktičnost i fleksibilnost, a uz to ima i prigodnu On-line potporu. Također, složeniji programi mogu biti postavljeni u kraćem vremenu, a uz to Matlab posjeduje i vrlo snažnu grafičku potporu koja nudi brojne mogućnosti. Zbog svih svojih pozitivnih karakteristika Matlab nalazi primjenu u inženjerskim, privrednim i prirodnim znanostima.

2.1 Rad u MATLAB-u

Ovisno o namjeni, Matlab možemo koristiti kao "džepno računalo", kao programski jezik i kao vrlo složen matematički program.

Nakon dvostrukog klika na Matlabovu ikonu otvara nam se radni prostor koji se naziva Command Window. Matlab odmah u tom radnom prostoru ispisuje upit sa oznakama >> kojim naznačava da očekuje korisnikovu naredbu. Nakon upisa i pritiskom tipke Enter započinje izvršenje naredbe. Potom slijedi ispis ili iscrtavanje rješenja te zahtjev za sljedećom naredbom.



Slika 2.1. Matlabov glavni radni prozor

Matlabovi prozori podijeljeni su na više područja:

- Command Window - koristi se kod komunikacije s korisnikom
- Command History - služi za spremanje naredbi u tzv. povijest naredbi iz koje se može ponovo pokrenuti
- Launch pad - služi kako bi ubrzao pristup alatima i dokumentaciji
- Current directory - u ovom području nalaze se sve datoteke koje se očitavaju, spremaju i pokreću
- Workspace – radni prostor koji pokazuje skup trenutačno raspoloživih varijabli

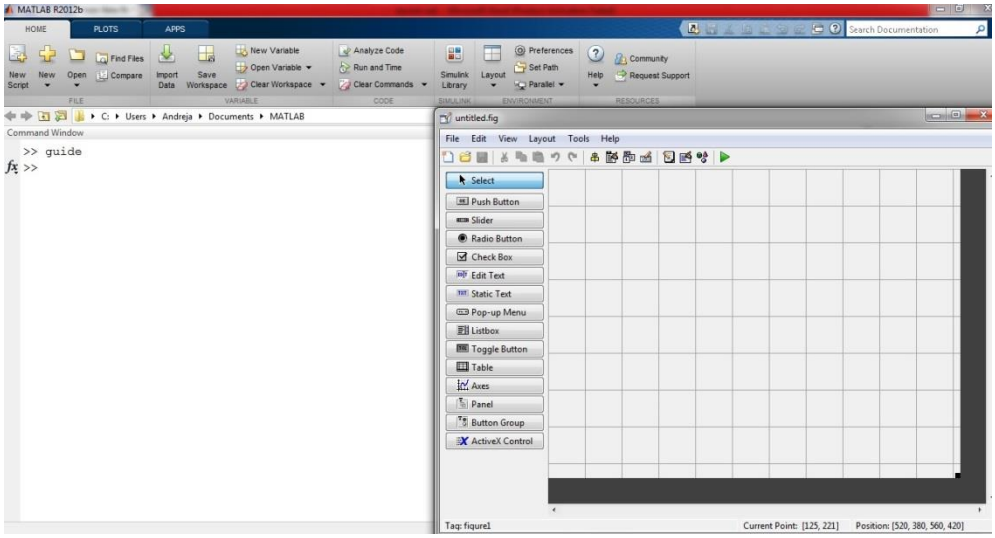
U Matlabu postoji 14 vrsta podataka, a svaki od njih pojavljuje se u obliku polja, array. Najmanje takvo polje je reda 1x1, a može narasti i do veličine n-te dimenzije. Razlikuju se jednodimenzionalne i dvodimenzionalne verzije polja. Jednodimenzionalne verzije polja su vektori, a dvodimenzionalne su matrice. Polja mogu biti potpuna i slabo popunjena, a označavaju se alfanumeričkim imenima koja ne smiju počinjati znamenkama. Svako polje ima i svojstvo automatskog proširivanja veličina, što je jedna od glavnih karakteristika Matlaba.

2.2 Grafičko korisničko sučelje (eng. Grafic User Interface GUI)

GUI je dio Matlab-a koji omogućava izradu grafičkih sučelja programiranjem odgovarajućih naredbi direktno u M-datoteku. Takav način programiranja grafičkog sučelja izuzetno je dugačak i kompliciran, te je u tu svrhu je napravljen GUIDE, odnosno alat za izradu grafičkog sučelja ubacivanjem gotovih objekata korisničkog sučelja.

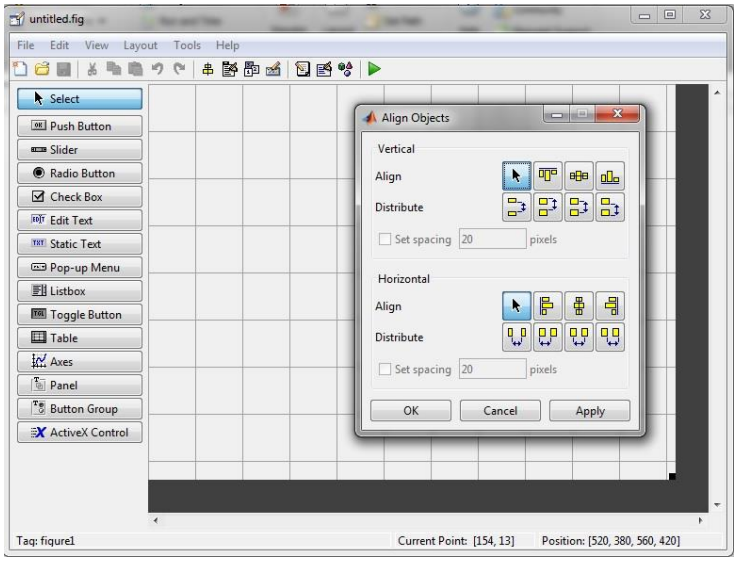
2.2.1 GUIDE (eng. Graphical User Interface development enviroment)

Grafičko sučelje GUIDE u Matlabu otvaramo preko glavnog prozora upisom naredbe *guide* iza oznake za unos. Nakon toga otvara nam se GUIDE Layout Editor, novi prozor, koji na svojoj lijevoj traci sadrži temeljne objekte potrebne za izgradnju željenog korisničkog sučelja.



Slika 2.2. GUIDE Layout Editor

Kod izgradnje korisničkog sučelja, željene elementa postavljamo na površinu na jednostavan način, „drag and drop" postupkom. Osnovni objekti koje možemo koristiti, a vidljivi su i na Slika 2.3, su različite tipke, pop-up izbornici, okviri za unos teksta, statički tekst, tablice, slikovni sadržaji i slično. Radi jednostavnijeg postavljanja objekata u traženi položaj u glavnom prozoru nalaze se linije, grid, koje ovisno o našoj želji možemo preko izbornika smanjivati i povećavati.



Slika 2.3. Poravnavanje odabranih objekata

Elemente možemo poravnavati i preko izbornika Tools koji se nalazi na glavnoj liniji, u kojem potom kliknemo na Align objects te nam se potom otvara prozor za podešavanje razmaka i udaljenost odabranih objekata.

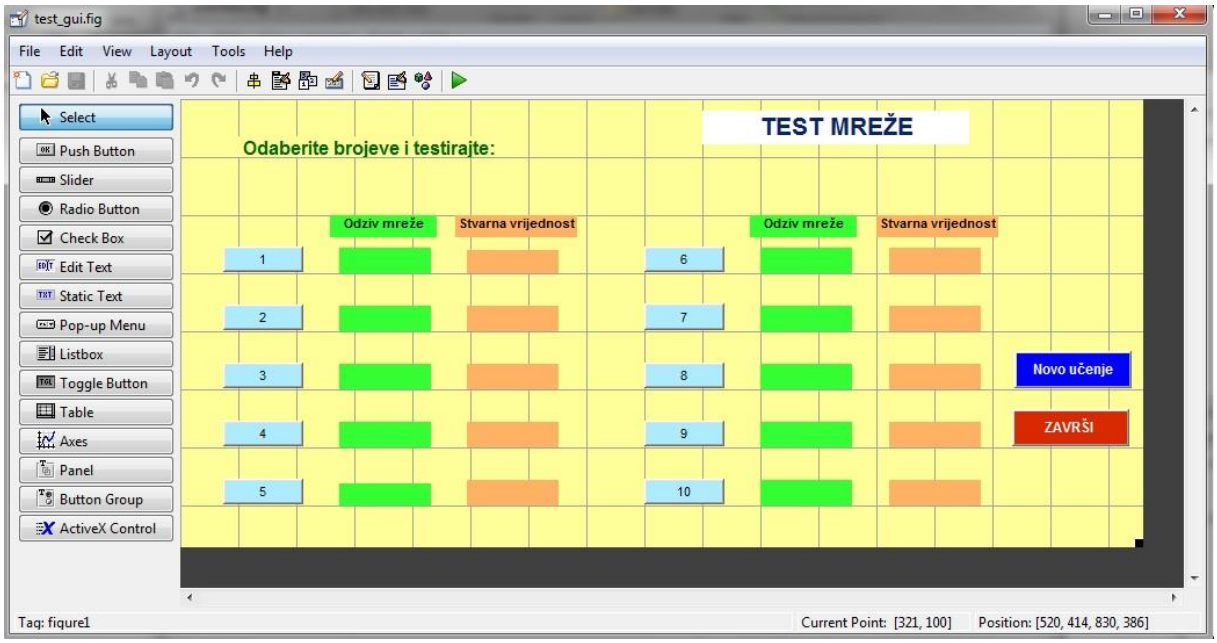
Moguća je i promjena boje, veličine, položaja i drugih karakteristika, a to se vrši preko izbornika koji nam se otvori nakon dvostrukog klika na željeni element, tzv. Property Insperctor. Osim navedenih, ovdje nalazimo i druge bitne parametre svakog objekta, poput oznake (tag), povratnog poziva (callback) i vrijednosti (value). Sve vrijednosti parametara spremaju se automatski, a u M-datoteci nose naziv prema onome što smo unijeli u polje tag.

Nakon što izradimo željeno sučelje, slijedi spremanje datoteke koja u ovom slučaju ima nastavak .fig. Prilikom spremanja automatski se stvara i druga, M-datoteka, koja sadrži inicijalizaciju svih objekata te se u njoj nalaze sve informacije i funkcije koje izvode elementi koji su odabrani.

Izvršenje željene naredbe postiže se tako da se u M-datoteci napravi povratni poziv, te unutar te funkcije definira naredba koja će se izvršiti pritiskom na tu tipku ili neki drugi objekt. Ako funkcije nisu definirane ili su krive, do izvršenja naredbe neće doći. Također je važno da se sve vrijednosti koje se koriste proglase stringovima. String je niz znakova koji je predstavljen kao vektor ili jednoredna ili jednostupčana matrica.

Samo pokretanje grafičkog sučelja uslijedit će nakon pritiska na tipku Run koja se nalazi u glavnom izborniku GUIDE Layout Editora, a označena je zelenim trokutom.

.



Slika 2.4. Pregled grafičkog sučelja za testiranje mreže

3. Statička unaprijedna neuronska mreža s povratnim prostiranjem pogreške (eng. EBP- Error-Back Propagation)

Kao što je ranije spomenuto, kod unaprijednih neuronskih mreža signal putuje samo u jednom smjeru, od ulaza prema izlazu, dok u slučaju postojanja povratne petlje govorimo o povratnim neuronskim mrežama. Kako bi se objasnilo funkcioniranje statičke unaprijedne neuronske mreže s povratnim prostiranjem pogreške prvo će biti objašnjen osnovni dio neuronske mreže odnosno Perceptron, zatim učenje neuronske mreže te sama matematika po kojoj neuronska mreža uči.

3.1 Perceptron

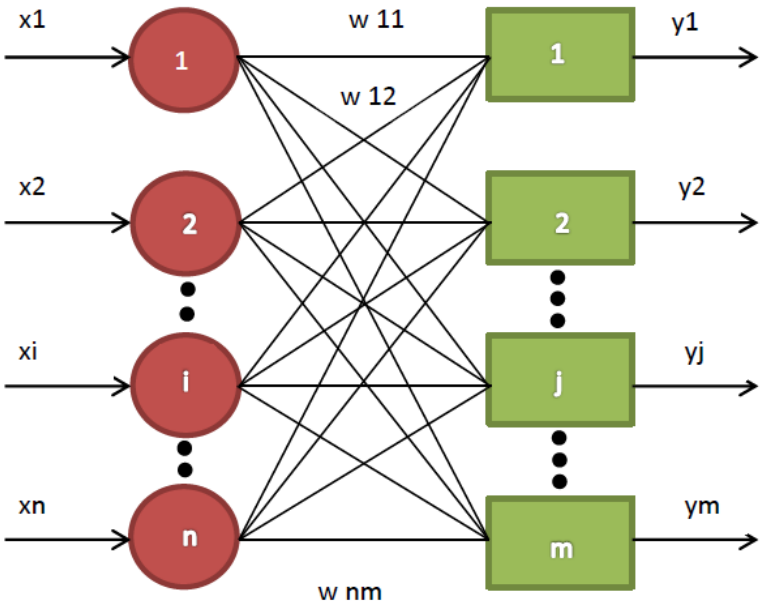
Iza pojma Perceptron stoji umjetni neuron kod kojeg kao aktivacijsku funkciju koristimo binarnu funkciju praga osjetljivosti definiranu izrazom:

$$y_i = f(z_i) = \begin{cases} 1 \; ako \; z_i > prag \\ 0 \; ako \; z_i \leq prag \end{cases} \tag{3.1}$$

Može se reći da je Perceptron binarni neuron koji može poprimiti samo vrijednosti 0 i 1, pri čemu ako je vrijednost 1 neuron je aktivan i šalje signal susjednim neuronima, a kod vrijednosti 0 neuron je neaktivan, miruje. Veća količina Perceptrona može se lako spojiti u neuronsku mrežu.

Sama struktura mreže određena je problemom, broj neurona ulaznog sloja n jednak je broju značajki koje opisuju problem, a broj neurona na izlazu je jednak broju skupina za koje se provodi klasifikacija. Neuroni ulaznog i izlaznog sloja međusobno su povezani, dok neuroni unutar istog sloja nisu povezani

Glavna uloga ulaznog sloja jest da prenosi signale do izlaznog sloja, gdje se ulazne vrijednosi množe sa pridruženim težinama i sumiraju. Oznaka težina je w_{ij} , a one su dakle podesivi parametri i predmet učenja. Konačni je cilj učenja podešavanje nepoznate težine w_{ij} na način da binarni izlazni neuron, koji pripada istoj skupini kao i promatrani uzorak skupa za učenje , poprimi vrijednost 1, a svi ostali neuroni vrijednost 0 [1, 8]. Nakon što se učenje uspješno provede, mreže se mogu koristiti kao klasifikatori.



Slika 3.1. Mreža perceptrona

3.1.1 Učenje Perceptrona

Mreže nastale spajanjem perceptrona uče postupkom iteracije. Za ulazni vektor x uzorka iz skupa za učenje izračuna se izlazni vektor y prema sljedećoj formuli [1, 5, 8] :

$$y = x \cdot W \tag{3.2}$$

W-matrica težina- u 1. koraku slučajni brojevi od 0 do1
x – ulazni vektor
y – izlazni vektor

U slučaju točnog izlaza dolazi do povrata u točku 1. Ako je izlaz pogrešan i 0, vrijednosti ulaznog vektora nadodaju se pripadajućim težinama, dok u slučaju pogrešnog vektora i 1, vrijednosti ulaznog vektora oduzmu se od pripadajućih težina. Slijedi povratak u točku 1. Postupak se provodi za sve uzorke skupa za učenje dok svi nisu pravilno razvrstani u pripadajuće skupine.

3.1.2 Delta pravilo

Delta pravilo osnova je za učenje većine modela neuronskih mreža, a dobiveno je generalizacijom postupka učenja Perceptrona i to kod neurona s kontinuiranim ulaznim i izlaznim vrijednostima. Radi se dakle o jednom od prvih postupaka namještanja težina [1].

Odstupanje (delta) nosi oznaku δ_j , a definirano je izrazom:

$$\delta_j = T_j - A_j \tag{3.3}$$

T_j – željena izlazna vrijednost neurona j
A_j- izračunata vrijednost izlaznog neurona j

Korekcija podesive težine dobiva se iz izraza:

$$w_{ij}(n+1) = w_{ij}(n) \cdot \Delta_{ij} \tag{3.4}$$

I – indeks neurona ulaznog sloja
J – indeks neurona izlaznog sloja
Δ_{ij} –faktor promjene težine
$w_{ij}(n+1) -$ težina veze između neurona i ulaznog sloja i j izlaznog sloja
$w_{ij}(n)$ - težina prije podešavanja

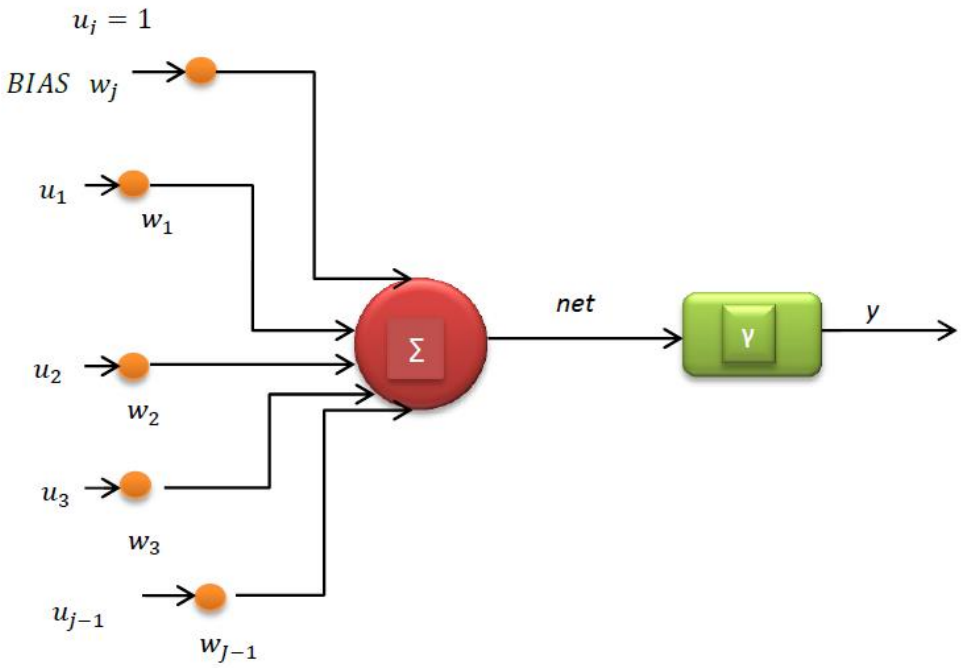
Prikazano Delta pravilo ne može se primijeniti kod neurona skrivenog sloja, jer za njih nije poznat željen vektor. Minsk i Papert prvi su dokazali da postoji vrlo velika skupina jednostavnih problema koje nije moguće riješiti jednoslojnom mrežom Perceptrona. Najpoznatiji takav problem je XOR ili ILI problem, a tu spadaju i drugi linearno neseparabilni uzorci.

Problem linearno nesaparabilnih uzoraka uspješno se rješava uvođenjem dodatnih slojeva, koji se nazivaju skriveni slojevi, uz uvjet da koriste nelinearnu aktivacijsku funkciju. XOR problem koristi se kod ispitivanja svojstava različitih modela umjetnih neuronskih mreža, a ulaz je određen s dvije binarne varijable koje poprimaju vrijednosti nula i jedan. Ovakav problem može se uspješno riješiti uvođenjem RBF mreže koja se sastoji od dva neurona ulaznog, četri neurona skrivenog i jednog neurona izlaznog sloja. RBF mrežom smatra se ona neuronska mreža s radijalnim baziranim funkcijama.

3.2 Višeslojne neuronske mreže

Najpoznatija višeslojna neuronska mreža je diskretna statička neuronska mreža sa povratnim rasprostiranjem pogreške (eng. Error-Back Propagation Neural Networks with Static and Dynamic Neuron Models), a služi za raspoznavanje uzoraka. Ovakva mreža koristi model statičkog neurona [8] detaljno prikazan u nastavku.

3.2.1 Model statičkog neurona



Slika 3.2. Model statičkog neurona [4]

Iz *Slika 3.2* jasno je vidljivo sa se statički neuron sastoji od više ulaza, a samo jednog izlaza. Također je vidljivo da ovakav oblik neurona sadrži dvije temeljne podfunkcije, a to su funkcija sume ∑ i aktivacijska funkcija γ. Svaki neuron koji sudjeluje u procesu učenja posjeduje poseban ulaz jedinične vrijednosti koji je u strukturi neuronske mreže vezan zasebnom vezom, a nosi oznaku Bias konstantnog izlaza jednakog jedinici.

Net je rezultat funkcije sume koja predstavlja ulaz neurona i pripadajućih težinskih faktora, dok aktivacijska funkcija vrši preslikavanje vrijednosti net u izlaznu vrijednost neurona.

Vrijednost net računamo prema izrazu:

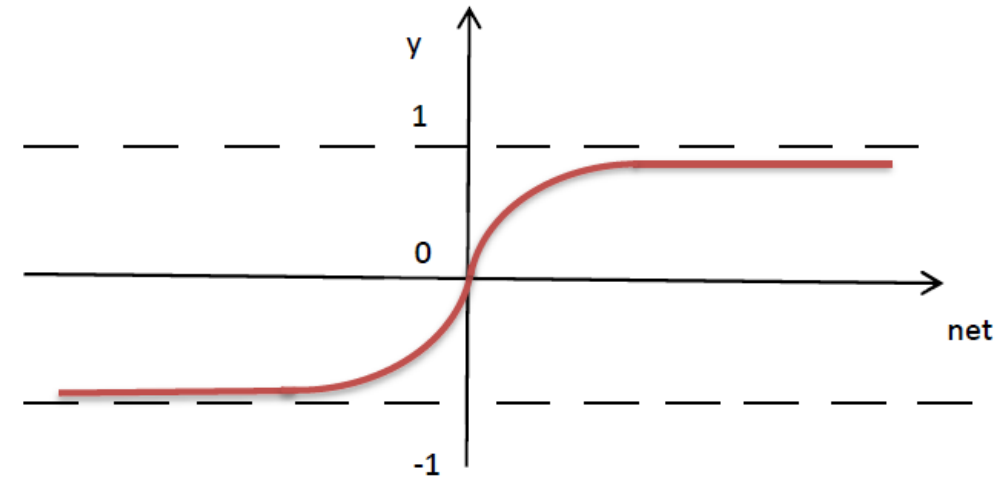
$$net = \sum_{j=1}^{J} w_j u_j$$

dok aktivacijsku funkciju dobivamo preko formule:

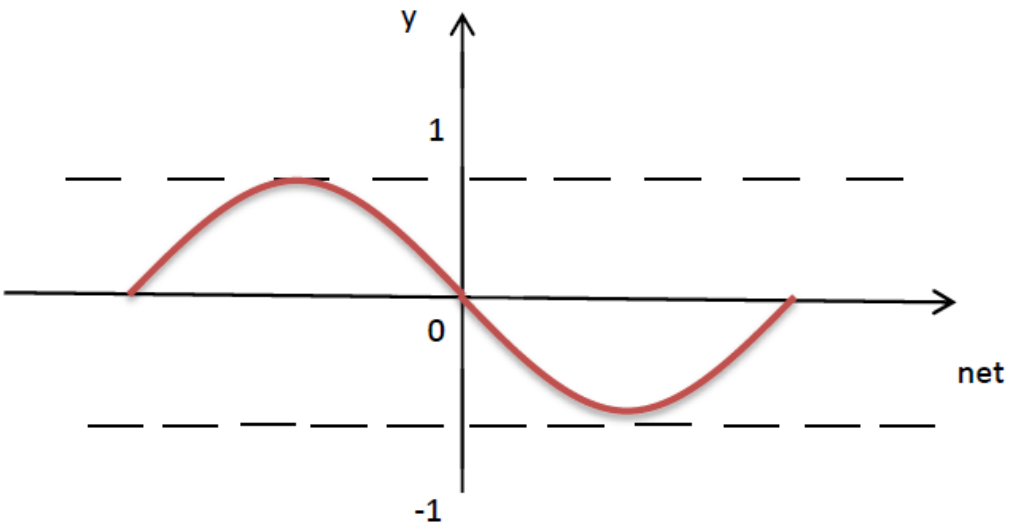
$$y = \gamma\,(net) \tag{3.6}$$

w – težinski faktori
u – ulazi
y –izlaz
γ – aktivacijska funkcija

Najčešće korištena aktivacijska funkcija je nelinearna bipolarna sigmoidalna funkcija, a zatim slijedi sinusna funkcija.



Slika 3.3 Bipolarna sigmoidalna funkcija



Slika 3.4 Sinusna funkcija

Aktivacijske funkcije rastuće su monotone funkcije sa zasićenjem, a odabir prigodne funkcije ovisi o skupu učenja i o skupu testiranja neuronske mreže. Gotovo sve aktivacijske funkcije, kako je vidljivo na *Slika 3.3* i *Slika 3.4*, zbog karakterističnog zasićenja normiraju vrijednosti izlaza neurona na vrijednosti 1 i -1.

Iznos izlaza sinusne funkcije dobiva se preko formule:

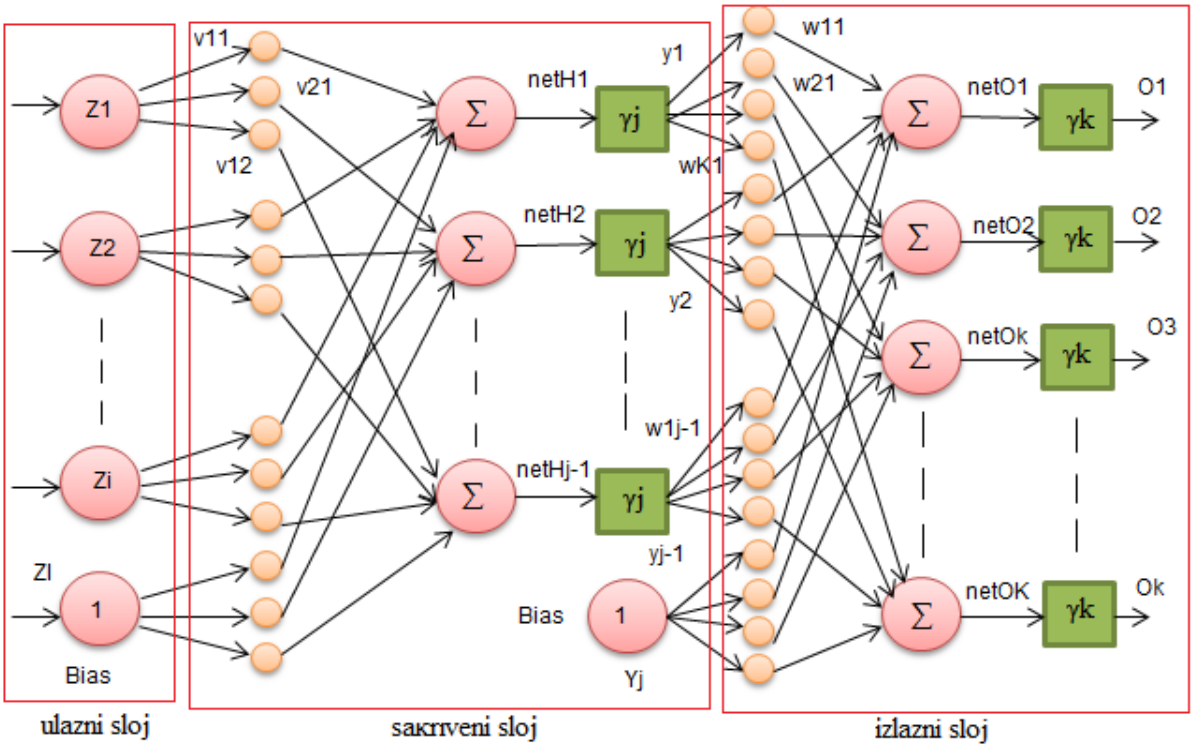
$$y = \sin(net) \tag{3.7}$$

dok se izlaz bipolarne sigmoidalne funkcije dobiva preko izraza:

$$y = \frac{2}{1+e^{-net}} - 1 \tag{3.8}$$

3.2.2 Model statičke neuronske mreže

Kako bi dobili pravilnu strukturu neuronske mreže, za početak neurone treba organizirati u spojeve koje je potom potrebno povezati vezama koje su opterećene težinskim koeficijentima.



Slika 3.5. Model statičke unaprijedne neuronske mreže

Iz *Slika 3.5* vidljivo je da se razlikuju tri tipa slojeva neuronskih mreža , a to su ulazni sloj, sakriveni sloj i izlazni sloj neurona. Ulazni i izlazni sloj u direktnoj su interakciji s okolinom.

Oznaka Z predstavlja ulaze u mrežu, koji su sa skrivenim slojem povezani preko veza opterećenim težinama v_{ji}. Važno je naglasiti da je svaki neuron promatranog sloja vezan sa svakim neuronom prethodnog sloja. Broj ulaznih i izlaznih neurona određen je karakterom preslikavanja koji mreža obavi. Izlazni sloj povezan je sa sakrivenim slojem neurona težinskim vrijednostima w_{kj} te on definiran izlaze neuronske mreže označene sa O_k. Što se tiće skrivenih slojeva, njihov broj je proizvoljan, a najčešće se koriste jedan do dva takva skrivena sloja neurona. Jedan takav skriveni sloj omogućava dovoljno dobru aproksimaciju kontinuiranih funkcija, a jedini uvjet je da sadrži dovoljan broj neurona.

3.2.3 Učenje povratnim rasprostiranjem pogreške

Postoje tri vrste učenja, a to su učenje s čitateljem, učenje bez čitatelja, te učenje koje je kombinacija prva dva. Najčešće se koristi učenje s učiteljem.

Svaka mreža uči tako da joj se podese težinski koeficijenti veza među slojevima i to s ciljem da se izlazi mreže za odgovarajuće ulaze skupa za učenje što više približe zadanim vrijednostima. Dakle, krajnji rezultat učenja ne mora biti potpuno točan, već je on aproksimacija nekog željenog izlaza.

Razlikujemo algoritme jednog koraka učenja i iteracijske algoritme učenja, dok se postupak učenja za svaki korak obavlja u dvije osnovne faze, unaprijedna i povratna faza, a one će biti opisane detaljno u nastavku.

Promjena parametara odvija se na dva načina, uz pomoć "batch" procedure pri kojoj se parametri mijenjaju jednom nakon prolaska čitavog ulaznog skupa datoteke učenja kroz mrežu, a drugi je "pattern" postupak kod kojeg se parametri učenja mijenjaju za svaki ulazno izlazni par skupa učenja.

3.2.3.1 Unaprijedna faza učenja statičke mreže

Prva faza učenja je unaprijedna faza, u kojoj se iz trening zapisnika uzimaju vrijednosti svih ulaza mreže Z, te se u njima izračunava izlaz mreže O. Da bi se faza mogla provesti potrebno je odrediti iznose težinskih faktora W i V. Kod prvog učenja obično se kao početne težine uzimaju slučajne vrijednosti brojeva, a u slučaju normiranih ulaznih i izlaznih vrijednosti zapisnika učenja potrebno je i početne težine birati slučajno. Dakle, ako su početne vrijednosti zapisnika učenja normirane u vrijednostima od -1 do 1, onda se i početne vrijednosti težina trebaju odabrati generatorom slučajnih brojeva u intervalu između -1 i 1. [1] Ukoliko u samom početku učenja pojavljuju problemi sa lokalnim minimumima pogreške, tada i interval početnih vrijednosti treba smanjiti za red veličine tj. između -0,1 i 0,1.

Unaprijedna faza učenja započinje funkcijom sume net neurona skrivenog sloja H, koja potom dobiva indeks oznake sloja netH te za svaki j-ti neuron dobiva drugi indeks, a računa se pomoću formule:

$$net_{Hj} = \sum_{i=1}^{I} v_{ji} Z_i \tag{3.9}$$

gdje je j=1,2,...J-1 i i=1,2,..I.

I – broj neurona ulaznog sloja + 1 (bias)
J- broj neurona skrivenog sloja uvećan za bias

U slučaju odabira nelinearne bipolarne sigmoidalne funkcije, vrijednost izlaznog sloja dobivamo prema izrazu:

$$y_j = \frac{2}{1+e^{-net_{Hj}}} - 1 \tag{3.10}$$

gdje je j= 1,2, ... J-1, a u slučaju odabira sinusne funkcije vrijednost izlaznog sloja dobiva se prema:

$$y_j = \sin(net) \tag{3.11}$$

Funkcija sume net izlaznog sloja O dobiva se iz izraza:

$$net_{ok} = \sum_{j=1}^{J} w_{kj} y_j \tag{3.12}$$

gdje je k=1,2,...K, pri čemu K predstavlja broj neurona izlaznog sloja.

U slučaju odabira linearne funkcije kod koje postoji mogućnost ostvarenja vrijednosti izlaza mreže veće od 1, aproksimacijski izraz za izračun izlaza mreže glasi:

$$O_k = K_p \cdot net_{ok} \tag{3.13}$$

gdje je k=1,2,..., K_p, pri čemu je K_p nagib linearne aktivacijske funkcije.

3.2.3.2 Povratna faza učenja statičke mreže

Druga faza učenja naziva se povratnom fazom, kod koje se na osnovi ostvarenog izlaza mreže i željenog izlaza mreže izračunava pogreška učenja. Na osnovi očitane pogreške vrši se korekcija vrijednosti težinskih koeficijenata veza između slojeva. Postupak se ponavlja sve dok se ne postigne pogreška manja ili jednaka dozvoljenoj pogrešci.

Uobičajena statistička metoda regresijske analize, suma kvadrata pogreške kao mjera odstupanja izlaza mreže od željene vrijednosti izlaza najčešće je korištena funkcija cilja :

$$E = \frac{1}{2}\sum_{n=1}^{N}(d_n - O_n)^2 \tag{3.14}$$

N- broj elemenata u skupu za učenje

$\frac{1}{2}$ – potrebite derivacije funkcije cilja

O_n- dobiveni izlaz mreže

d_n – željeni izlaz mreže

Odabranom funkcijom cilja se vrši promjena koeficijenata težina primjenom nekog od algoritma nelinearnog optimiranja. Forma promjene parametara učenja težinskih koeficijenata ϑ:

$$\vartheta(n+1) = \vartheta(n) + \Delta\vartheta(n) \tag{3.15}$$

n- trenutni broj koraka učenja

$\Delta\vartheta(n)$-veličina promjene parametara učenja

$\vartheta(n+1)$- nova vrijednost parametara učenja

v=ϑ – skriveni sloj

w=ϑ – izlazni sloj

Porešku $E(\vartheta)$ moguće je u okolišu točke ϑ aproksimirati u prva dva člana Taylorovog reda prema sljedećim izrazima:

$$E(\vartheta + \Delta\vartheta) \approx E(\vartheta) + \Delta E(\vartheta) \tag{3.16}$$

$$\Delta E(\vartheta) = \Delta\vartheta^T \nabla E(\vartheta) \tag{3.17}$$

$$\nabla E(\vartheta) = \frac{\partial E(\vartheta)}{\partial \vartheta} \tag{3.18}$$

$\nabla E(\vartheta)$- gradijent pogreške

Kako bi se pogreška smanjivala maksimalnim mogućim iznosom, potrebno je odrediti Δϑ za koji pogreška učenja ΔE(ϑ) poprima najveći negativni iznos, a to se ostvaruje uz uvjet:

$$\Delta\vartheta = -\eta\nabla E(\vartheta) \tag{3.19}$$

η – koeficijent brzine učenja

Koeficijent brzine učenja određuje već ranije spomenuti učitelj, a njegova vrijednost kreće se od 10^{-3} do 10.

Na temelju prikazanih formula, može se izraziti izraz za izračun algoritma najstrmijeg pada ili algoritam povratnog prostiranja pogreške koji glasi:

$$\vartheta(n+1) = \vartheta(n) - \eta \nabla E(\vartheta(n)) \tag{3.20}$$

Najveći nedostatak ovog algoritma je u velikom broju potrebnih iteracija, točnije koraka učenja. Da bi proces učenja tekao brže, uz smanjen broj koraka i dozvoljenu pogrešku, koriste se različite modifikacije osnovnog oblika. Jedan takav modifikacijski algoritam poznat je pod nazivom momentum , odnosno zamah, detaljnije objašnjen u nastavku.

Sve prikazane formule odnose se na povratnu fazu koja se vrši po uzorku. Postoji još jedna povratna faza, a to je povratna faza po skupu.

Kao što je već objašnjeno povratna faza po skupu se odvija na način da se sve greške ulazno-izlaznih parova zbrajaju te da se nakon toga pristupa promjeni parametara učenja. U ovom sklučaju vrijedi sljedeći izraz:

$$\Delta\vartheta = -\eta \sum_{i=1}^{n} \nabla E(\vartheta) \tag{3.21}$$

n- broj ulazno-izlaznih parova

te se dobiva izraz

$$\vartheta(n+1) = \vartheta(n) - \eta \sum_{i=1}^{n} \nabla E(\vartheta) \tag{3.22}$$

3.2.4 Ubrzanje iterativnog učenja

Kao što je spomenuto u prethodnom ulomku, algoritme učenja karakterizira veliki broj potrebnih iteracija, koraka učenja. Glavni modifikator navedenih iteracija je zamah, a u nastavku će biti opisani zamah 1. reda i zamah 2. reda.

3.2.4.1 Zamah prvog reda

U jednadžbu za algoritam povratnog prostiranja pogreške dodaje se dio u kojem se koristi pogreška učenja u prethodnom korakom pomnožena sa koeficijentom zamaha α. Iznos koeficijenta zamaha određuje učitelj, a on se kreće od 0.1 do 0.9.

Vrijednost momentuma prvog reda računa se preko izraza:

$$\Delta\vartheta(n) = \alpha\Delta\vartheta(n-1) - \eta \nabla E(\vartheta(n)) \tag{3.23}$$

N – trenutna promjena parametara učenja
(n-1) – prethodna promjena parametara učenja
α-koeficijent momentuma

Primjenom ovog izraza brzina algoritma učenja može se povećati i do 10 puta.

Konačna promjena parametara učenja poprima oblik:

$$\Delta\vartheta(n+1) = \vartheta(n) + \alpha\Delta\vartheta(n-1) - \eta\nabla E(\vartheta(n)) \tag{3.24}$$

3.2.4.2 Zamah drugog reda

U slučaju potrebe za dodatnim ubrzanjem procesa koristi se zamah drugog reda. Ovaj zamah ne koristi se tako često zato što zamah drugog redam može znatno usporiti brzinu učenja. Zamah drugog reda se oduzima od jednadžbe za algoritam povratnog prostiranja pogreške umnoškom koeficijenata zamaha drugog reda β i greške učenja u predzadnjem koraku.

Zamah drugog reda dobiva se iz sljedećeg izraza:

$$\beta = \frac{\alpha-1}{3} \tag{3.25}$$

Izraz za algoritam povratnog prostiranja pogreške glasi:

$$\Delta\vartheta(n+1) = \vartheta(n) + \alpha\Delta\vartheta(n-1) - \eta\nabla E\big(\vartheta(n)\big) - \beta\Delta\vartheta(n-2) \tag{3.26}$$

Navedena formula vrijedi kod povratne faze koja se provodi po uzorku. Ukoliko se radi o povratnoj fazi po skupu, treba koristiti sljedeći izraz:

$$\Delta\vartheta(n+1) = \vartheta(n) + \alpha\Delta\vartheta(n-1) - \eta\sum_{i=1}^{n}\nabla E\big(\vartheta(n)\big) - \beta\Delta\vartheta(n-2) \tag{3.27}$$

3.2.5 Promjena težina izlaznog sloja

Promjena parametara učenja, ukoliko imamo povratnu fazu s povratnim prostiranjem pogreške, odvija se od izlaznog prema ulaznom sloju mreže. Promjena težinskih faktora između ulaznog i izlaznog sloja odvija se na sljedeći način:

$$w_{kj}(n+1) = w_{kj}(n) - \eta\nabla E(n) + \alpha\Delta w_{kj}(n-1) \tag{3.28.}$$

Gradijent pogreške $\nabla E(n)$ za težine w_{kj} računa se prema:

$$\nabla E(n) = \frac{\partial E(n)}{\partial w_{kj}} \tag{3.29}$$

n – broj ulazno – izlaznih parova

Uzastopnim parcijalnim deriviranjem dobivamo:

$$\frac{\partial E(n)}{\partial w_{kj}} = \frac{\partial E(n)}{\partial O_k} \cdot \frac{\partial O_k}{\partial net_{Ok}} \cdot \frac{\partial net_{Ok}}{w_{kj}} \tag{3.30}$$

Iz izraza 3.30 postupkom deriviranja možemo izračunati iznose pojedinih članova:

$$\frac{\partial E(n)}{\partial O_k} = -(d_k - O_k) \tag{3.31}$$

$$\frac{\partial O_k}{\partial net_{Ok}} = \gamma'_k = 1 \tag{3.32}$$

Množenjem izraza (3.31) i (3.32) pokazuje se karakteristična vrijednost algoritma povratnog prostiranja pogreške koja se prvo računa za izlazni sloj i koja je po definiciji:

$$\delta = -\frac{\partial E(n)}{\partial net} \tag{3.33}$$

$$\partial O_k = d_k - O_k \tag{3.34}$$

Derivacijom izraza (3.13) dobiva se:

$$\frac{\partial net_{Ok}}{\partial w_{kj}} = y_j \tag{3.35}$$

Nakon uvrštavanja izraza (3.31), (3.32) i (3.34) u (3.28) dobiva se:

$$\nabla E(n) = \frac{\partial E(n)}{\partial w_{kj}} = -(d_k - O_k) y_j = -\delta_{Ok} \cdot y_j \tag{3.36}$$

Uvrštavanjem izraza (3.36) u (3.28) dobiva se:

$$w_{kj}(n+1) = w_{kj}(n) - \eta\delta_{Ok}y_i + \alpha\Delta w_{kj}(n-1) \tag{3.37}$$

3.2.6 Promjena težina skrivenog sloja

Osim težina izlaznog sloja potrebno je promijeniti i težine skrivenog sloja v_{ji}.

Glavni izraz po kojem se vrši promjena težina skrivenog sloja je:

$$v_{ji}(n+1) = v_{ji}(n) - \eta\nabla E(n) + \alpha\Delta v_{ji}(n-1) \tag{3.38}$$

I u ovom slučaju potrebno je izračunati gradijent pogreške $\nabla E(n)$, također primjenom parcijalne derivacije, iz čega slijedi izraz:

$$\frac{\partial E(n)}{\partial v_{ji}} = \frac{\partial E(n)}{\partial y_i} \cdot \frac{\partial y_i}{\partial net_{Hj}} \cdot \frac{\partial net_{Hj}}{\partial v_{ji}} \tag{3.39}$$

Na svaku težinu skrivenog sloja utječu svi neuroni izlaznog sloja, pa iz toga slijedi da je izraz $\frac{\partial E(n)}{\partial y_i}$ jednak:

$$\frac{\partial E(n)}{\partial y_i} = \frac{\partial E(n)}{\partial O_1} \cdot \frac{\partial O_1}{\partial net_{O1}} \cdot \frac{\partial net_{O1}}{\partial y_j} + \frac{\partial E(n)}{\partial O_2} \cdot \frac{\partial O_2}{\partial net_{O2}} \cdot \frac{\partial net_{O2}}{\partial y_j} + \ldots\ldots + \frac{\partial E(n)}{\partial O_K} \cdot \frac{\partial O_K}{\partial net_{OK}} \cdot \frac{\partial net_{OK}}{\partial y_j} \tag{3.40}$$

gdje je:

$$\frac{\partial E(n)}{\partial O_k} = -(d_k - O_k), \quad k = 1,2,\ldots K \tag{3.41}$$

$$\frac{\partial O_k}{\partial net_{OK}} = 1, \quad k = 1,2,\ldots K \tag{3.42}$$

$$\frac{\partial net_{OK}}{\partial y_i} = w_{kj}, \quad k=1,2,\ldots K, \quad j=1,2,\ldots\ldots,J-1 \tag{3.43}$$

Slijedi uvrštavanje izraza (3.41), (3.42) i (3.43) u izraz (3.40) nakon čega se dobiva sljedeća jednadžba:

$$\frac{\partial E(n)}{\partial y_i} = -\sum_{k=1}^{K}(d_k - O_k)\, w_{kj} \tag{3.44}$$

U izraz (3.44) možemo uvrstiti izraz (3.34) korišten u prethodnom ulomku vezanom za težine izlaznog sloja, iz čega se dobiva:

$$\frac{\partial E(n)}{\partial y_j} = -\sum_{k=1}^{K}\delta_{Ok}\, w_{kj} \tag{3.44}$$

Pomoću izraza (3.9) i (3.10) dobivamo i vrijednosti drugog i trećeg razlomka u jednakosti za izračun pogreške gradijenta , (3.39). Dobivaju se slijedeći izrazi:

$$\frac{\partial y_j}{\partial net_{Hj}} = \gamma'_j = \frac{1}{2}(1 - y_j^2) \tag{3.45}$$

$$\frac{\partial net_{Hj}}{\partial v_{ji}} = Z_i \tag{3.46}$$

Izraze (3.44), (3.45) i (3.46) sad treba vratiti u jednadžbu (3.39), nakon čega se dobiva konačni oblik algoritama promjene težinskih koeficijenata sakrivenog sloja koji glasi:

$$v_{ji}(n+1) = v_{ji}(n) + \frac{1}{2}\eta(1 - y_j^2)Z_i(\sum_{k=1}^{K}\delta_{Ok}w_{kj}) + \alpha\Delta v_{ji}(n-1) \tag{3.47}$$

Kad bi mreža sadržavala još jedan sakriveni sloj, postupak proračuna bio bi jednak, samo bi se mreža malo više proširila. Svaki novi sloj jamči bolju kvaliteti učenja, ali to zahtijeva i mnogo više procesorskog vremena. Također će i mreža imati sporiji odziv od mreže sa samo jednim slojem sakrivenih neurona.

3.3 Ocjena točnosti algoritma učenja

Uspješnost učenja neuronske mreže za određeni zadatak određuje se tako da se najprije definira mjera točnosti. Razlikuju se 3 mjere točnosti učenja, a to su srednja kvadratna pogreška, korijen srednje kvadratne pogreške i normalizirani korijen srednje kvadratne pogreške (NRMS). U nastavku slijede izrazi za njihov izračun.

- Izraz za izračun srednje kvadratne pogreške - MS (eng. Mean Squre Error):

$$MS = \frac{\sum_{n=1}^{N}(d_n - o_n)^2}{N} \tag{3.48}$$

n- broj pojedinog izlaza
N- ukupan broj izlaza

- Izraz za izračun korijena srednje kvadratne pogreške – RMS (eng. Root Mean Squre Error):

$$RMS = \sqrt{MS} = \sqrt{\frac{\sum_{n=1}^{N}(d_n - o_n)^2}{N}} \tag{3.49}$$

n- broj pojedinog izlaza
N- ukupan broj izlaza

- Izraz za izračun normaliziranog korijena srednje kvadratne pogreške – NRMS (eng. Normalized Root Mean Squre Error):

$$NRMS = \frac{RMS}{\sigma_{d_n}} = \frac{\sqrt{\frac{\sum_{n=1}^{N}(d_n - o_n)^2}{N}}}{\sigma_{d_n}} \tag{3.50}$$

pri čemu se σ_{d_n} dobiva iz izraza

$$\sigma_{d_n} = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(d_n - \bar{d})^2} \quad , \tag{3.51}$$

a $\bar{d}$ iz

$$\bar{d} = \frac{1}{N}\sum_{n=1}^{N} d_n \tag{3.52}$$

n- broj pojedinog izlaza
N- ukupan broj izlaza

Kao mjera točnosti učenja u ovom programskom radu odabran je NRMS, tj. normalizirani korijen srednje kvadratne pogreške. NRMS karakterizira bezdimenzionalnost koja osigurava neovisnost mjere o dimenzijama učenih veličina i koja omogućuje usporedbu izvedenih algoritama učenja s drugim algoritmima, neovisno o korištenoj sklopovskoj ili programskoj podršci.

4. Neuronska mreža za kvadriranje prirodnih brojeva

Cilj ovog rada bio je napraviti statičku unaprijednu neuronsku mrežu koju će odlikovati sposobnost kvadriranja cijelih brojeva. Također, ovaj rad rađen je kako bi poslužio kao nastavno pomagalo pri demostraciji rada statičke neuronske mreže. Postupak kreiranja datoteka učenja, provođenja i testiranja naučene mreže bit će detaljno prikazan u nastavku. Kao što je već ranije navedeno, cijeli rad mreže napravljen je uz pomoć programskog paketa MATLAB.

4.1 Princip rada mreže

Početak rada zadane neuronske mreže, nazvane "Kvadriralica", započinje otvaranjem .m datoteke. Pritiskom tipke „run", otvara se prigodno korisničko sučelje. U njemu na samom početku treba odabrati brojeve koji se žele kvadrirati, a potom pritisnut tipku '*Spremanje*'.

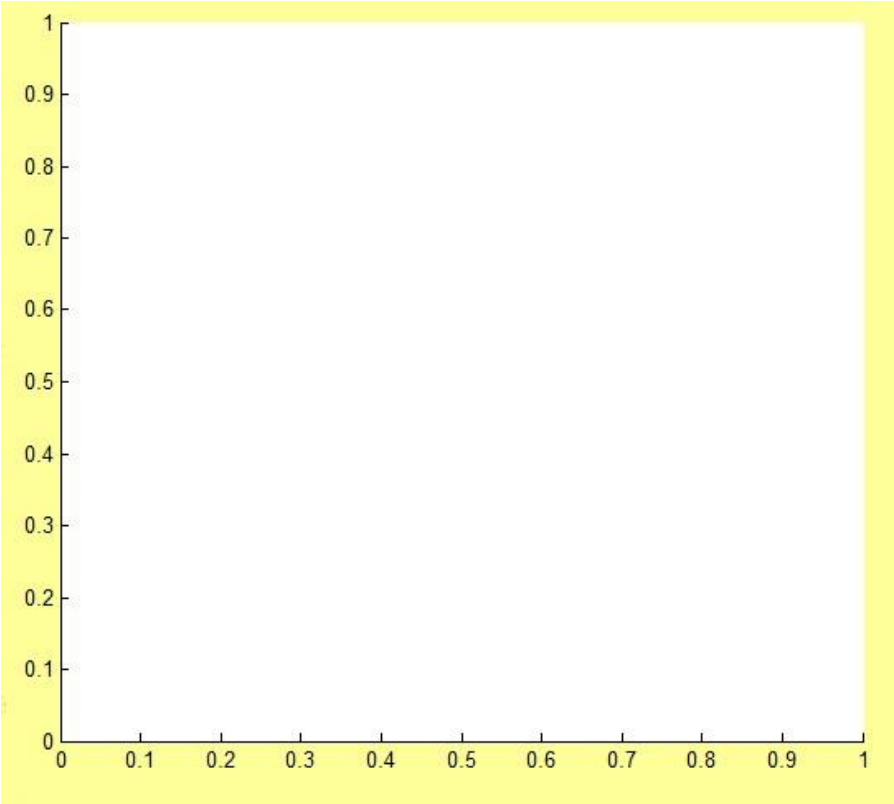
Slika 4.1 Odabir brojeva za kvadriranje

Slijedi odabir parametara mreže:

1. željena funkcija – sigmoidalna ili sinusna funckcija
2. odabir težine – postojeća ili proizvoljna
3. odabir momentuma – bez momentuma, moment prvog reda ili moment drugog reda
4. unos brzine učenja
5. unos željenog NRMS-a
6. unos koraka učenja

Slika 4.2 Odabir parametara mreže

Potom je potrebno pritisnuti na tipku *Učenje* kako bi se započeo proces i pokrenuo ispis na desnom ekranu. Također, u izborniku se nalazi i tipka *Ispis* koja crta pripadni NRMS dijagram.

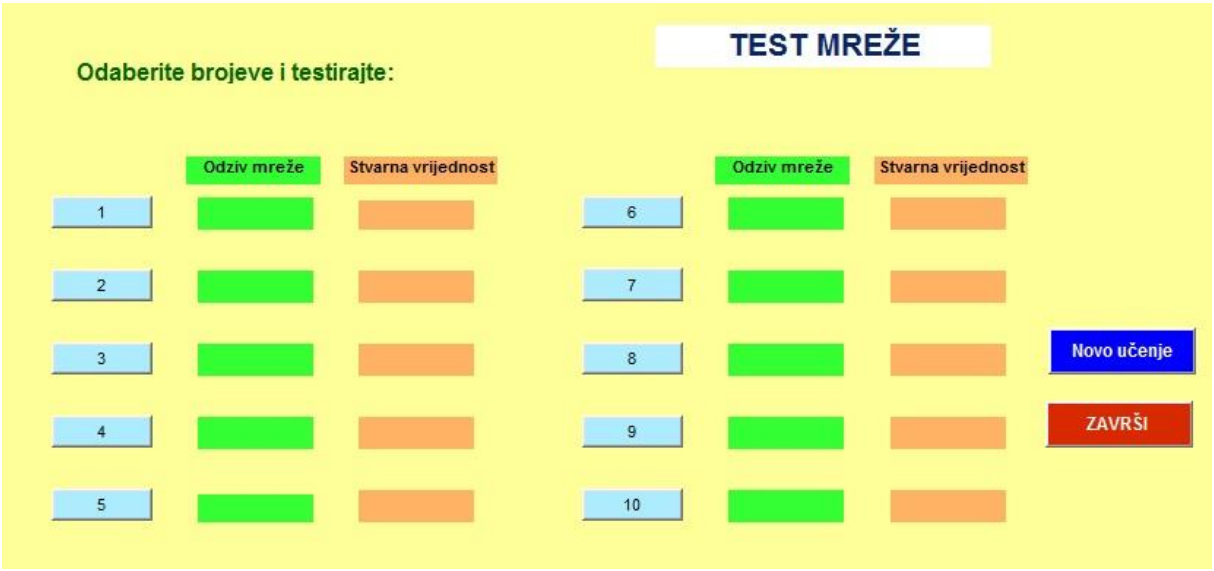


Slika 4.3 Prozor za iscrtavanje NRMS dijagram

Osim odgovarajućeg NRMS dijagrama, pritiskom na *Učenje* dobivamo tablicu u kojoj se nalaze vrijednosti NRMS-a za svaki korak učenja, te vrijeme učenje i ukupan broj koraka koji su bili potrebni da se učenje provede.

Slika 4.4 Prozor za ispis koraka i vremena učenja

Nakon što smo proveli učenje, na redu je testiranje, koje se pokreće klikom na gumb *Testiranje.* Automatski se otvara novi prozor unutar kojeg je moguć ponovan odabir brojeva kako bi se provjerilo kako mreža radi i što je naučila.



Slika 4.5 Prozor za testiranje mreže

Odabirom nekog broja prikazuju se vrijednosti u oba prozora, jedan prikazuje odziv mreže (neka cjelobrojna vrijednost), dok je drugi stvarna vrijednost (najčešće decimalna vrijednost). Nakon provedenog testiranja može se odabrati jedna od ponuđene dvije opcije. Klikom na *Novo učenje* otvara se prethodni prozor za učenje mreže te je moguće ponoviti postupak, dok se klikom na *Završi* zatvara prozor i mreža prestaje s radom.

4.2 Topologija mreže

Broj ulaza u mrežu ovisi o tome koliko se brojeva odabere, a za svaki odabrani broj dobiva se po jedan izlaz. Nakon što je broj označen i pritisnuta tipka *Spremanje*, mreža ga pohrani u obliku .txt datoteke u memoriju te automatski počinje stvarati Z matricu, tj. matricu ulaznih vrijednosti. Analogno dolazi do stvaranja jedinične matrice D, točnije izlazne matrice, kojoj veličina ovisi o tome koliko brojeva mreža treba kvadrirati tj. koliko rješenja treba dati.
Prije samog početka učenja zadajemo broj neurona skrivenog sloja. Prozor za unos neurona skrivenog sloja dobivamo tako da pod opcijom *Odabir težina* odaberemo *Proizvoljne težine,* te u polju koje se otvori upišemo željenu vrijednost. Nakon unošenja željenog broja sakrivenih neurona, generiranjem slučajnih vrijednosti tvore se matrice V i W, gdje je matrica V matrica težina sakrivenih neurona i ovisna je o broju ulaznih neurona i broju sakrivenih neurona, a matrica W je matrica težina izlaznih neurona te je ovisna o broju sakrivenih neurona plus BIAS te o broju izlaznih neurona. Kao što se kod izbora težina pojavljuje dodatno polje za unos podatka, tako se i u slučaju odabira *Momentum 1. reda* ili *Momentum 2. reda* pojavljuje polje za unos u kojem unešenu vrijednost mreža pohrani i koristi kroz cijeli proces učenja.

4.3 Normiranje

Brojeve koje želimo da mreža kvadrira odabiremo na način prikazan na *Slika 4.1*, te njihovim spremanjem formiramo ulaznu matricu Z kojom krećemo u proces učenja. Važno je da ti brojevi budu normirani, kao i brojevi vezani za izlaznu matricu D. Normiranje je potrebno kako bi svi neuroni sustava imali jednake šanse utjecati na ishod učenja.
Prema tome , za matricu Z vrijedi izraz:

$$Z(i,1) = \frac{broj(i)}{10} \tag{4.1}$$

Kod normalizacije smo uzeli najveću vrijednost koja se može kvadrirati, broj 10. Podjednak postupak primjenimo i na matricu D koju moramo podijeliti sa brojem 100, budući da je to kvadratna vrijednost broja 10.

$$D(i,1) = \frac{kavd_br(i)}{100} \tag{4.2}$$

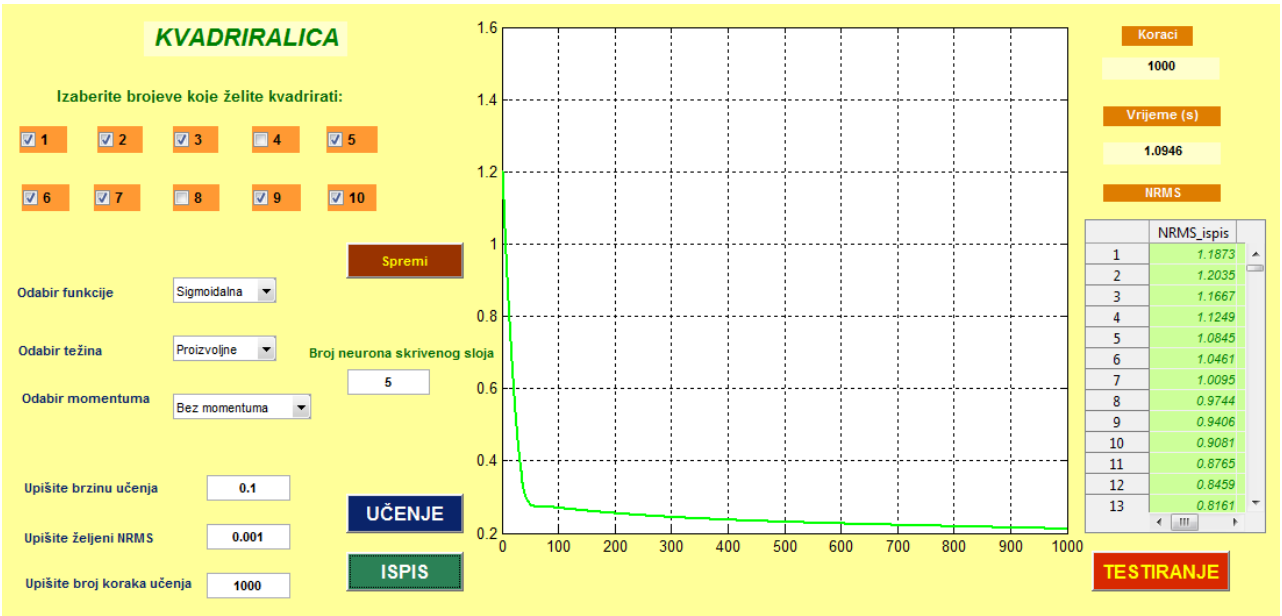
Oznake u zagradi označavaju gdje će se dobivena vrijednost spremti, a u ovom slučaju radi se o prvom stupci. Matrica Z tvori se redom kako smo kliknuli na neki broj, a ne po njegovom iznosu (npr. od većeg prema manjem ili obratno) .

4.4 Primjer rada mreže

Kao što je rečeno u prethodnih nekoliko ulomaka, kako bi mrežu mogli učili na samom početku trebamo odabrati brojeve koje želimo kvadrirati. Uzmimo u ovom slučaju 1,2,3,5,6,7,9 ,10 te stavimo oznaku kvačice u kućice ispred njih.

Slika 4.6 Odabir brojeva za kvadriranje

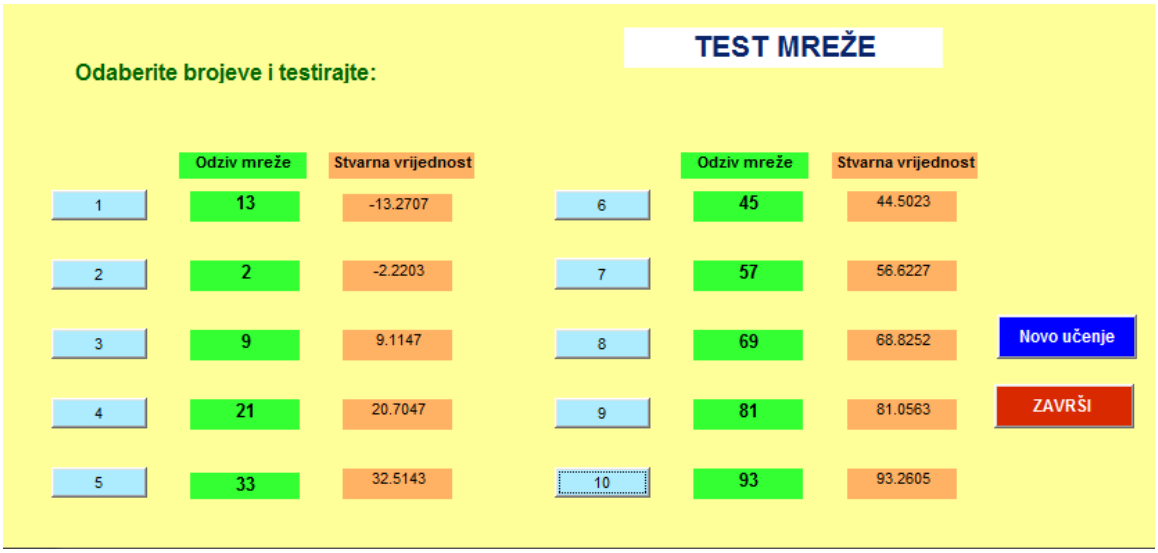
Sljedeće odabiremo parametre. U ovom slučaju uzet ćemo sigmoidalnu funkciju, proizvoljne težine sa 5 sakrivenih neurona, te bez momentuma. Kao vrijeme učenja uzmimo 0,1 sekundu, koraka 1000, dok NRMS nastojimo zadati što niži, npr. 0.001. Slijedi pritisak na tipku *Učenje* , a potom *Ispis* kako bi dobili rezultat.



Slika 4.7 Prikaz prozora za učenje mreže - 1000 koraka učenja

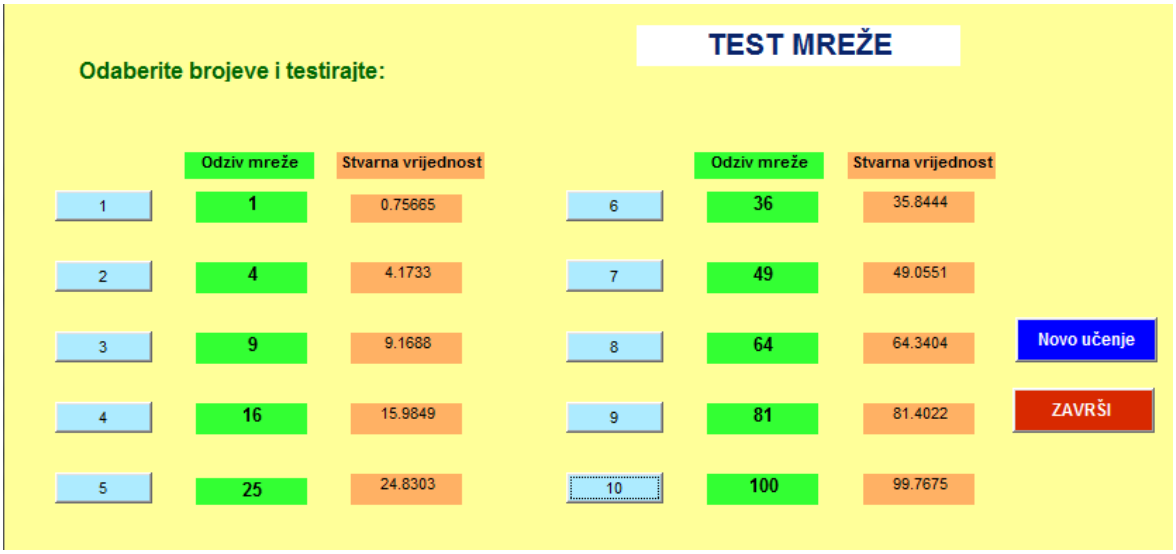
Na prikazanoj slici vidimo da se pritiskom na *Učenje* stvorila spomenuta tablica koja sadrži iznose NRMS-a u svakom pojedinom koraku. Jasno je vidljivo da povećanjem broja koraka NRMS, točnije normalizirani korijen srednje kvadratne pogreške se smanjuje.

Smanjivanje NRMS-a još jasnije je predočeno na dijagramu na *Slika 4.7*. Tako u 1. koraku NRMS za dati primjer 1,1873, u 500. iznosi 0,2307 , dok u krajnjem 1000. koraku NRMS isnosi 0,2215. Jasno je vidljivo iz *Slika 4.7* da je na samom početku pad NRMS izraženiji, a kako se broj koraka povećava, tako je njegova razlika za svaki korak sve manja, što se podudara s tvrdnjom da se povećanjem broja koraka pogreška smanjuje. Još jedan važan podatak je vrijeme koje je bilo potrebno da mreža obavi traženi zadatak. Vidmo da u ovom slučaju ono iznosi 1,0946 sekundi i da ga je obavljala punih 1000 koraka.



Slika 4.8 Testiranje mreže – 1000 koraka učenja

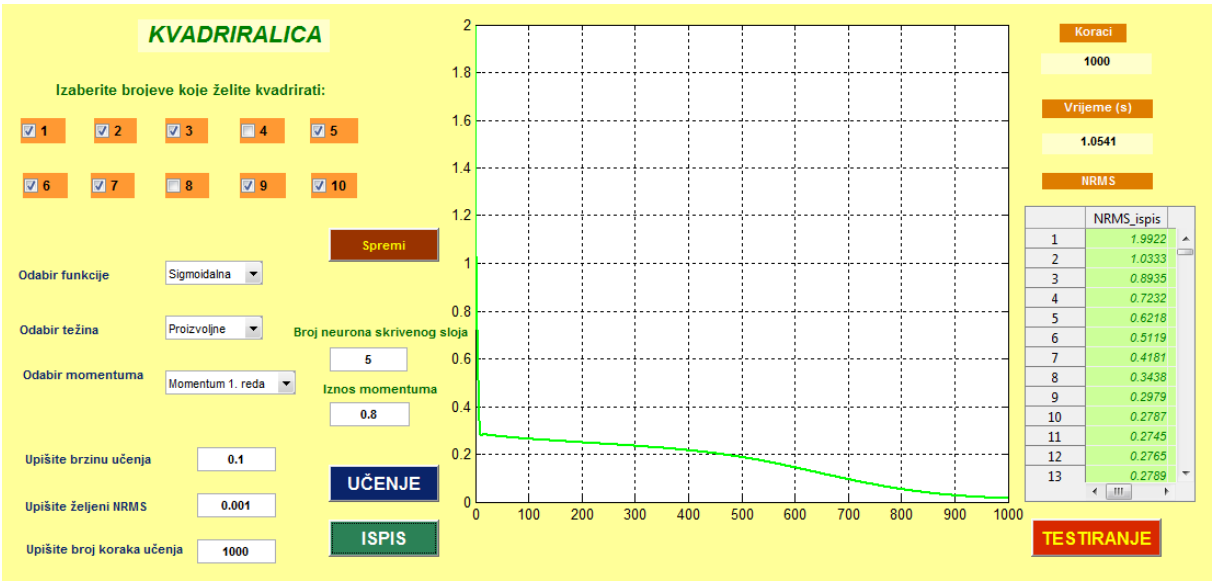
U prozoru za testiranje mreže vidimo da mreža nije dala u potpunosti zadovoljavajuća rješenja. Vidljivo je da je odziv mreže uvijek pozitivan, dok stvarna vrijednost koju mreža daje može biti negativna. U nastavku ćemo za iste parametre mreže i iste naučene brojeve pokušati dobiti točnije rješenje na način da ćemo povećati broj koraka učenja.



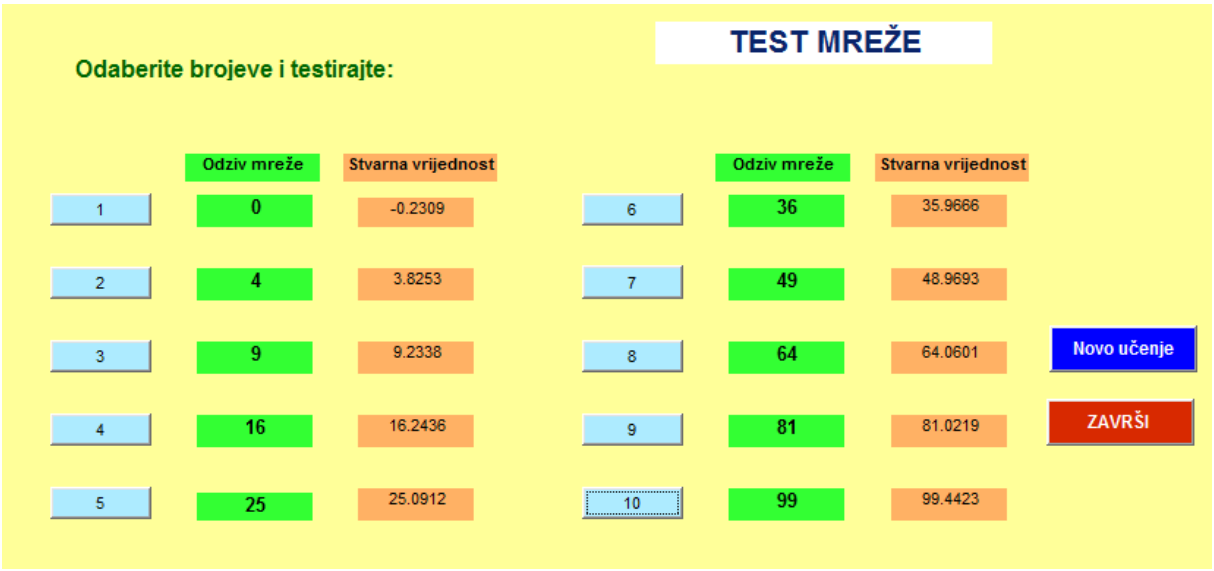
Slika 4.9 Testiranje mreže – 5000 koraka učenja

Povećanjem broja koraka učenja na 5000 koraka vidljivo je da je krajnji rezultat učenja daleko točniji nego kod računanja sa 1000 koraka. Ovime je dokazano kako brojem koraka povećavamo točnost procesa. Učenje u ovom slučaju traje 5,1746 sekundi, što je osjetno duže nego kod 1000 koraka. NRMS i u ovom slučaju pada, početna vrijednost 0,8197 , dok konačna 0,0123.

Kao što je već spomenuto, proces učenja možemo ubrzati uvođenjem momentuma prvog ili drugog reda. Slijedi prikaz rada sa 1000 koraka i momentumom 1. reda u vrijednosti od 0,8, dok ostali parametri ostaju jednaki.



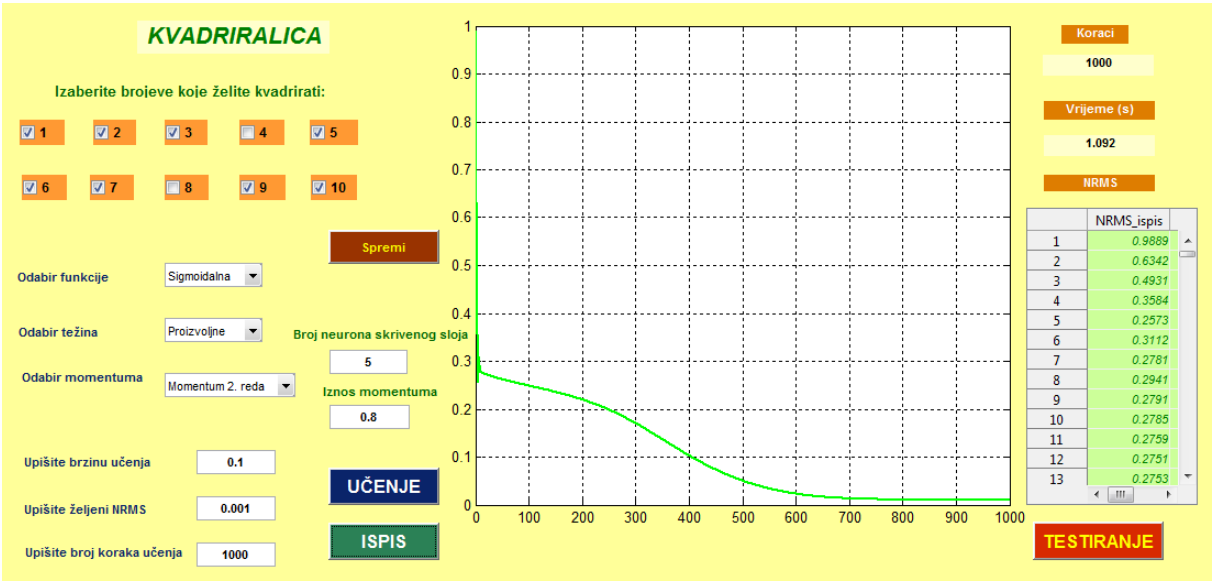
Slika 4.10 Prikaz prozora za učenje - 1000 koraka, moment 1. reda



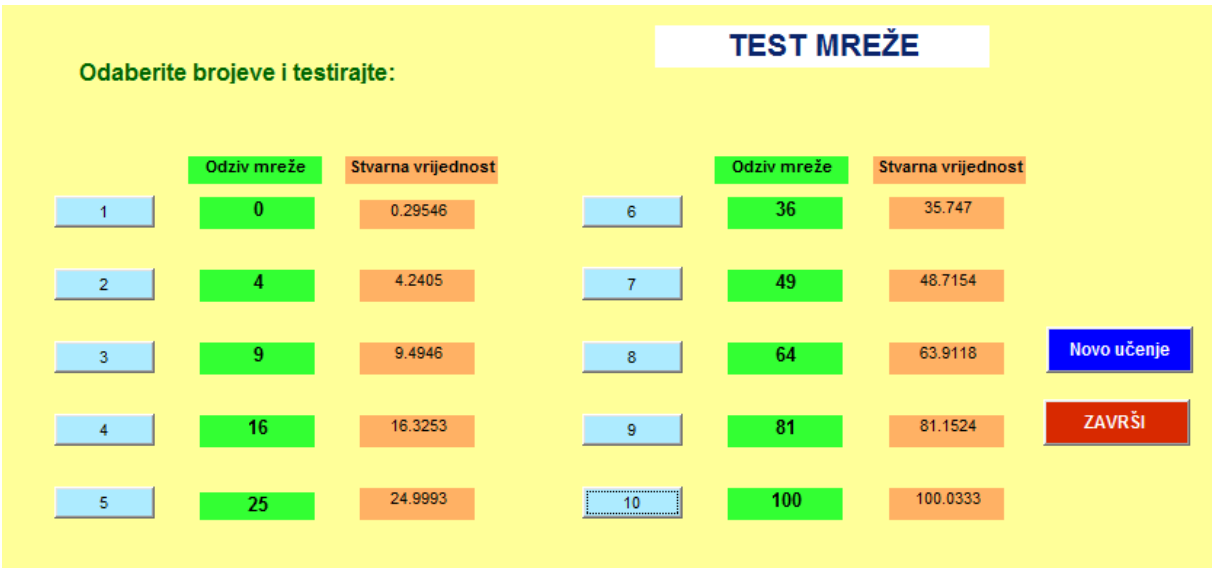
Slika 4.11 Testiranje mreže – 1000 koraka, momentum 1. reda

Iz *Slika 4.11* vidljivo je da smo momentumom 1. reda jako povećali brzinu učenja, a rješenje se približilo već u 1000. koraku onim vrijednostima koje je mreža davala bez uvođenja momentuma i to tek u 5000. koraku. Dok je mreži kako bi se približila točnijim vrijednostima trebalo 5,1746 sekundi, uvođenjem momentuma prvog reda, do sličnih rješenja već u 1000. koraku dolazi za 1,0541 sekundu. Početni NRMS učenja iznosi 0,9936, dok je konačni 0,0268.

Slijedi provjera rezultata uvođenjem momentuma 2. reda.



Slika 4.14 Prikaz prozora za učenje - 1000 koraka, momentum 2. Reda



Slika 4.15 Testiranje mreže – 1000 koraka, momentum 2. reda

I u ovom slučaju mreža je dala zadovoljavajuće rezultate, za još kraće vrijeme, a za jednak broj koraka učenja i jednake početne parametre.
Početni iznos NRMS-a glasi 0,9889, dok je konačni 0,0102. Vrijeme potrebno da se provede 1000 koraka učenja je 1,092 sekunde .

Radi preglednosti, slijedi prikaz rezultata u tablici.

- funkcija – sigmoidalna
- težina – proizvoljna
- broj neurona skrivenog sloja – 5
- momentum - prvog reda - 0,8
  - drugog reda – 0,8
- vrijeme - 0,1 sekunda
- NRMS – 0,001
- koraci 1000

	Broj koraka	NRMS(konačni)	Vrijeme(s)
Bez momentuma	1000	0,2215	1,0946
Momentum prvog reda	1000	0,0268	1,0541
Momentum drugog reda	1000	0,0102	1,0920

Tablica 4.1 Rezultati učenja- sigmoidalna funkcija, proizvoljne težine

Analogno vrijedi i za sigmoidalnu funkciju s postojećim težinama. U nastavku slijedi prikaz rezultata dobivenih korištenjem sinusne funkcije i proizvoljnih težina.

- funkcija – sinusna
- težina – proizvoljne
- broj neurona sakrivenog sloja - 5
- momentum - prvog reda - 0,8
  - drugog reda
- vrijeme - 0,1 sekunda
- NRMS – 0,001
- koraci 1000

	Broj koraka	NRMS(konačni)	Vrijeme(s)
Bez momentuma	1000	1,0716	1,0848
Momentum prvog reda	1000	1,0448	1,1107
Momentum drugog reda	1000	1,0064	1,0817

Tablica 4.2 Rezultati učenja- sinusna, proizvoljne težine

5. Zaključak

U ovom zadatku pobliže je objašnjeno što su neuronske mreže, njihov način rada i učenja te je uz pomoć "Kvadriralice" prikazana jedna od primjena ovakvih umjetnih neuronskih mreža s povratnim rasprostiranjem pogreške. Samo učenje je iterativno te ga je moguće provesti po skupu ili po uzorku. Aktivacijske funkcije koje se ponuđene kod učenja su sinusna i nelinearna bipolarna sigmoidalna aktivacijska funkcija. Kako bi se ubrzalo samo učenje, odnosno smanjio broj potrebnih iteracija u algoritam učenja ugrađen je zamah prvog i zamah drugog reda.

Sva programska podrška napravljena je uz pomoć MATLAB programskog paketa, a uz pomoć MATLAB-ovog programskog alata GUDE načinjeno je korisniku prilagođeno grafičko sučelje. Da bi mreža mogla ispravno raditi zahtijeva da se minimalno poznavanje matematičkih algoritama učenja i testiranja neuronske mreže.

Na temelju dobivenih rezultat vidljivo je kako zamah prvog reda jako smanjuje broj iteracija i brzinu učenja povećava nekoliko puta. Momentum drugog reda je u ovom primjeru usporio samo učenje, što je pokazatelj da će njegova primjena ovisiti o samom problemu učenja.

Također je iz svih primjera vidljivo da se povećanjem broja koraka smanjuje NRMS parametar učenja, neovisno o tome koju aktivacijsku funkciju odaberemo i sa kojim početnim težinama krećemo u učenje.

Uz navedeno je jasno vidljivo da neuronska mreža ne funkcionira kao kalkulator, tj. ona nikad ne daje cijele vrijednosti odziva, već su to decimalne vrijednosti koje na izlazu, a prije pokazivanja odziva korisniku, treba zaokružiti na cijele brojeve.

6. Literatura

[1] Branko Novaković, Dubravko Majetić, Mladen Široki, *Umjetne neuronske mreže,* FSB, Zagreb, 2011.

[2] M. Essert, T. Žilić, *Matlab- Matrični laboratorij,* Zagreb, 2004.

[3] W. James, *Psihology (Briefer Course),* Holt, New York, 1890.

[4] W. McCulloch and W. Pitts, *A logical calculator of the ideas immanent in nervous activity,* Bulletin of Mathematical Biophysics 5, pp. 115-133, 1943.

[5] B. Widrow and M. Hoff, *Generalization and information storage in networks of Adaline Neurons,* In M.C. Yovitz, G.T. Jacobi and G.Goldstein, editors, Self- Organizating Systems, Spartan Books, Washington, DC, 1962.

[6] J. Hopfield, *Neurons with graded response have collective computational properties like those of two-state neurons,* Proc. of the National Academy of Science, 81., pp. 3088-3092, May 1984.

[7] B. Novaković, *Stanje i trend umjetnih neuronskih mreža i robotike,* Vjesnik HAZU, vol. 1-3, str. 125-147, Zagreb, 1993.

[8] J.M.Zurada, *Artificial Neural Systems,* W.P Company, USA, 1992.