Univerzitet u Beogradu
Fakultet organizacionih nauka

Diplomski rad

Beograd 2010

Univerzitet u Beogradu
Fakultet organizacionih nauka

Supervizorni algoritami za učenje neuronskih mreža sa prostiranjem signala unapred

Diplomski rad

Profesor
Vladan Devedžić

Student
Marko Koprivica

Beograd 2010

SADRŽAJ

1. UVOD

Cilj ovog diplomskog rada je razvoj softvera za jednostavno prikazivanje učenja kod neuronskih mreža sa prostiranjem signal unapred, sa posebnim osvrtom na mreže tipa perceptron i višeslojni perceptron. Softver je realizovan u okviru aplikacije easyNeurons , pomoću softverske biblioteke *Neuroph* koja omogućava jednostano kreiranje i rad sa neuronskim mrežama sa prostiranjem signala unapred. Prvo je izvršena teorijska analiza različitih modela neuronskih mreža sa prostiranjem signala unapred, zatim su definisani zahtevi za svaki od njih, a potom projektovana odgovarajuća rešenja. Na kraju je obavljeno testiranje razlučitih modela višeslojnog perceptrona u rešavanju XOR problema kako bi se utvrdio uticaj arhitekture mreže i parametara učenja na brzinu učenja.
Rad je organizovan u pet celina:

1. U drugom poglavlju su izloženi osnovni pojmovi i objašnjenja u vezi neuronskih mreža sa prostiranjem signala unapred, a zatim su predstavljene opšte arhitekture.
2. U trećem poglavlju je napravljen poseban osvrt na moguća poboljšanja modela a zatim su predstavljeni *use case* model, konceptualni model i isprojektovane su klase softvera.
3. U četvrtom poglavlju je predstavljena implementacija softvera sa delovima koda i ponašanjem softvera u prilikom rešavanja problema.
4. U petom poglavlju su prikazani rezultati testova sa različitim arhitekturama – višeslojnog perceptrona
5. U šestom delu data su zaključna razmatranja

2. NEURONSKE MREŽE – DEFINICIJE I OSNOVNI POJMOVI

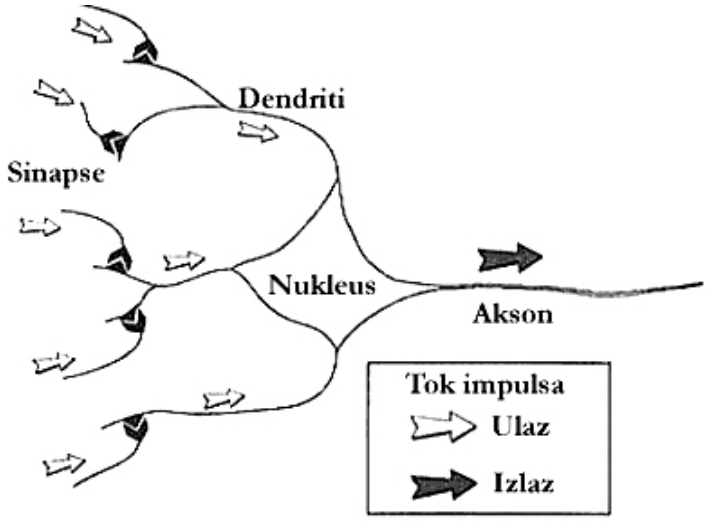
DARPA: Neuronska mreža je sistem koji se sastoji od velikog broja međusobno povezanih, jednostavnih elemenata procesiranja koji rade paralelno. Funkcija NM je određena strukturom mreže, težinom veza, i obradom u elementima procesiranja.

Haykin: Neuronska mreža je paralelni distribuirani procesor koji ima prirodnu sposobnost čuvanja i korišćenja iskustvenog znanja. Sličnost sa mozgom se ogleda kroz dve osobine:
- mreža stiče znanje kroz proces učenja
- znanje se čuva u vezama između neurona (sinaptičkim težinama)

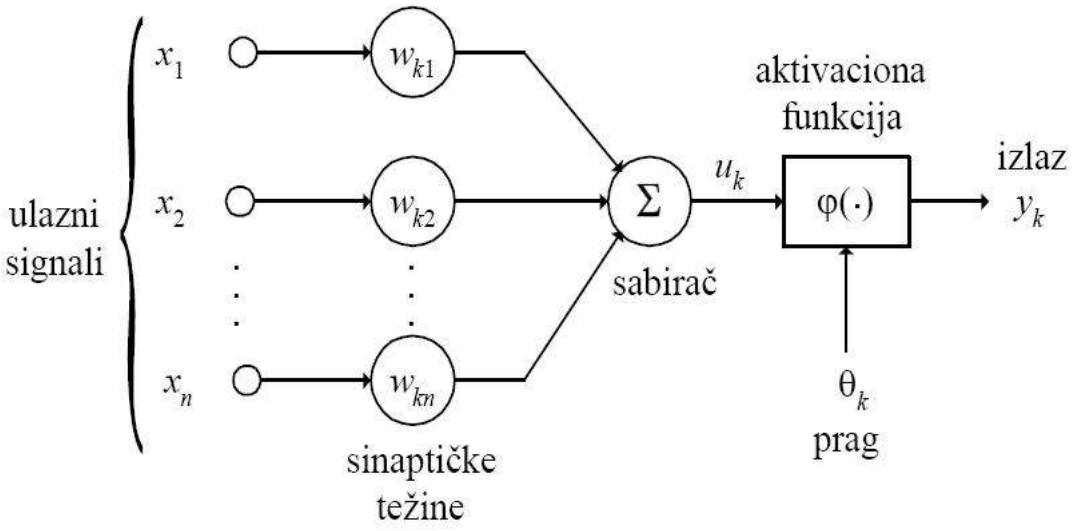
Zurada: Veštački neuro sistemi ili neuronske mreže, su ćelijski sistemi koji mogu da stiču, čuvaju i koriste iskustveno znanje.

U navedenim definicijama data su osnovna strukturna i funkcionalna svojstva NM, a to je da:
1) se sastoje od međusobno povezanih osnovnih jedinica (elemenata, ćelija) procesiranja, koje vrše neku jednostavnu, elementarnu obradu
2) jedinice procesiranja rade paralelno
3) imaju sposobnost učenja, čuvanja i korišćenja znanja

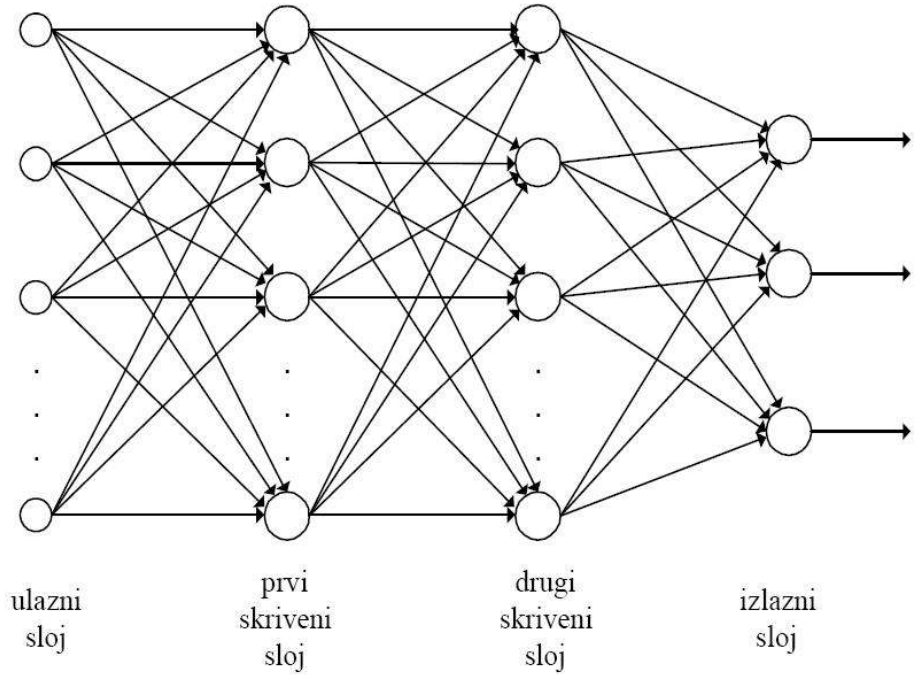
Veštačke neuronske mreže inspirisane su biološkim neuronskim mrežama, i predstavljaju njihov matematički odnosno računarski model. Neuroni su predstavljeni elementima procesiranja, a sinapse težinom veze. Dendriti su ulazi a akson je izlaz elementa procesiranja. Elementi procesiranja povezani su u mrežu tako što je izlaz svakog vezan na ulaz bar jednog od ostalih. Obrada koja se vrši u telu neurona predstavljena je funkcijama ulaza i prenosa.
Analogija između biološkog i veštačkog neurona prikazana je na slikama 1a i 1b Na slici 2 je prikazana veštačka neuronska mreža sa prostiranjem signala unapred.



Slika 1a. Biološki neuron



Slika 1b. Veštački neuron



Slika 2. Veštačka neuronska mreža sa prostiranjem signala unapred

Algoritam za trening (učenje) mreže je proces u kome se vrši podešavanje težina ulaznih veza kako bi mreža imala željeno ponašanje. Podešavanje težina veza se vrši na osnovu određenog skupa podataka - uzoraka odnosno primera. NM dakle uče na osnovu primera, a ispoljavaju sposobnost generalizacije i van podataka koji su korišćeni za učenje.

2.1. MODEL NEURONA

Osnovna komponenta neuronskih mreža je neuron (element procesiranja). To je jedinica koja obrađuje informacije koje dobija na ulazu i kao rezultat daje jedan izlaz. U skladu sa slikom 1b imamo sledeći matematički model:

u – ulazni vektor $[u_1, u_2, \ldots u_n]^T$
w – vektor težina $[w_1, w_2 \ldots w_n]^T$
net – ukupni ulaz iz mreže
g – ulazna funkcija
f – prenosna funkcija
y – izlaz

Izlaz neurona definisan je jednačinom

$$\boldsymbol{y = f(net)} \qquad (1)$$

Ukupni ulaz iz mreže za pojedinačni neuron je vektorska funkcija težina veza i ulaza

$$\boldsymbol{net = g(u, w)} \qquad (2)$$

Najčešće se računa se kao suma ulaza pomnoženih odgovarajućom težinom

$$\boldsymbol{net = \sum u_i w_i} \qquad (3)$$

i predstavlja skalarni proizvod vektora ulaza i vektora težina.

Za funkciju prenosa se biraju funkcije ograničene na intervalima [0, 1] , [-1, 1]
U tabeli 1 prikazane su često korišćene funkcije prenosa.

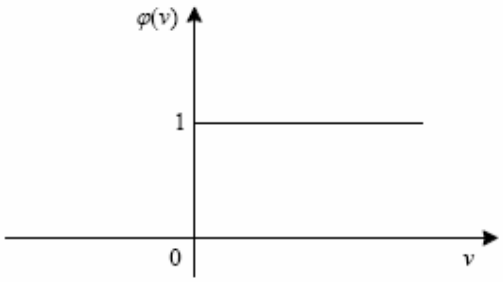
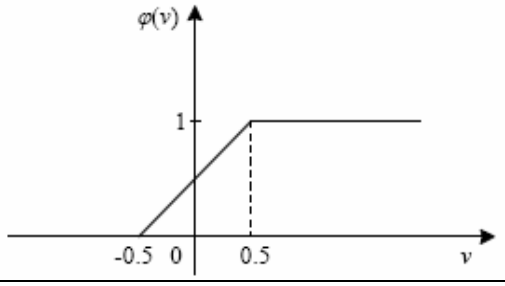
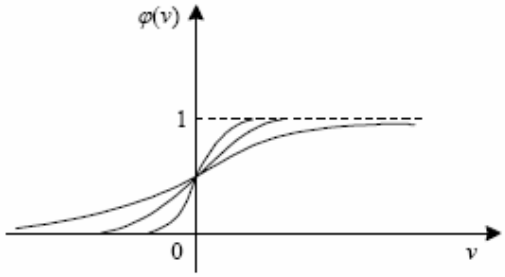
Odskočna	$\varphi(v) = \begin{cases} 0, & v < 0 \\ 1, & v \geq 0 \end{cases}$	
U delovima linearna	$\varphi(v) = \begin{cases} 0, & v < -\frac{1}{2} \\ \frac{1}{2} + v, & -\frac{1}{2} \leq v \leq \frac{1}{2} \\ 1, & v > \frac{1}{2} \end{cases}$	
Sigmoidna	$\varphi(v) = \frac{1}{1+e^{-av}}$	

Tabela 1. Prenosne funkcije

Pored navedenih koriste se i drugi tipovi aktivacionih funkcija. Nekada je poželjno da se se umesto između 0 i 1 izlaz iz aktivacione funkcije menja između –1 i 1. Tada se odskočna funkcija redefiniše na sledeći način:

$$\varphi(v) = \begin{cases} -1, & v < 0 \\ 0, & v = 0 \\ 1, & v > 0 \end{cases}$$

Ovako definisan model neurona funkcioniše na sledeći način:

1. Ulazna funkcija nalazi sumu proizvoda ulaznih signala i težina odgovarajućih ulaznih veza
2. Ova suma je ulaz za funkciju prenosa, čiji izlaz predstavlja izlaz iz neurona

U osnovnom modelu često se pominje i prag *(threshold)* koji predstavlja minimalnu ili maksimalnu vrednost ulazne sume da bi neuron bio aktivan. Ovakvo ponašanje je karakteristično za odskočnu funkciju Ako je

T – prag

tada je

$$Y = f(net-T) \quad (4)$$

pri čemu je f odskočna (step) funkcija.

Uobičajni parametri su $a=1$ i $b=0$

Kod nekih modela imamo i unutrašnju pobudu *(bias)*, koja predstavlja konstantnu vrednost koja se dodaje ulaznoj sumi. Ako sa B označimo bias, tada je za neuron sa bias-om:

$$y=f(net+B) \quad (5)$$

Iz datih formula se vidi da negativan *bias* ima istu ulogu kao prag (*threshold*).

Navedeni model naziva se McCulloch-Pitts (MP) model neurona. Pored njega postoje i razne varijante tzv. modifikovanog MP modela. Originalni MP model ima odskočnu funkciju prenosa, a modeli sa sigmoidnom funkcijom su tzv. modifikovani MP neuroni.

Treba napomenuti još i da ukoliko se uvede vremenska dimenzija, tada jednačine (1) i (3) postaju

$$y(t)=f(net(t)) \quad (6)$$
$$net(t) = \sum u_i(t) w_i \quad (7)$$

Pošto je računarska simulacija NM na jednoprocesorskim sistemima sekvencijalna, koristićemo modele u diskretnom vremenu i umesto t imati k koje označava tekuću iteraciju. Na taj način će se rešiti problem sinhronizacije izračunavanja.

Dati matematički model i uvedeni pojmovi su osnov za dalju analizu i formiranje konceptualnog modela. Pre toga data je analiza osnovnih arhitektura i algoritmi za učenje mreža sa prostiranjem signala unapred.

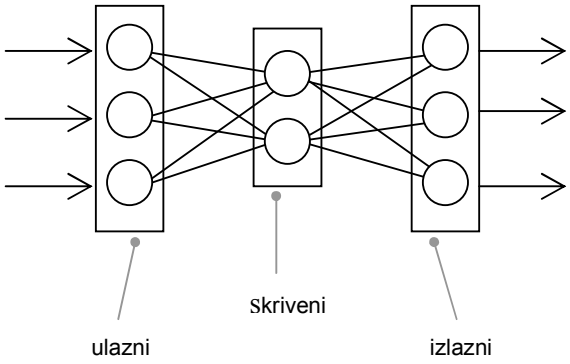
2.2. MODELI NEURONSKIH MREŽA SA PROSTIRANJEM SIGNALA UNAPRED

2.2.1. ARHITEKTURA

Neuroni u mreži grupisani su u slojevima *(layers*). U zavisnosti od uloge u mreži, slojevi se mogu biti:

1) ***Ulazni sloj*** – prima podatke iz okoline
2) ***Izlazni sloj*** – daje rezultate obrade
3) ***Skriveni* sloj**– nalaze se između ulaznog i izlaznog sloja. Nazivaju se skriveni jer njihovi ulazi i izlazi nisu dostupni iz 'spoljnog sveta', već se koriste za interne veze.

Na slici 3 je prikazana arhitektura neuronske mreže sa prostiranjem signala unapred.



Slika 3.Višeslojna mreža sa prostiranjem signala unapred

Veze u mreži su uspostavljene između neurona iz različitih slojeva.

Veza između slojeva je ***jednosmerna veza unapred - feedforward***

Izlazi neurona u svim slojevima šalju signal (vezani su) isključivo na neurone u narednim slojevima.

2.2.2. ALGORITMI ZA UČENJE NEURONSKIH MREŽA SA PROSTIRANJEM SIGNALA UNAPRED

Algoritma za učenje NM sa prostiranjem signala unapred koji koristimo je supervizorno učenje ***(supervised learning)***. Ova vrste algoritama zahteva tzv. učitelja. Učitelj je spoljni kontroler koji vrši podešavanje težina na osnovu razlike stvarnog i željenog izlaza mreže. Ova razlika predstavlja grešku i algoritam teži da je minimizuje.

Opšti matematički model za supervizorno učenje glasi:

$$\boldsymbol{w_{ji}(k+1) = w_{ji}(k) + \Delta w_{ji}(k) = w_{ji}(k) + \mu E(y_j(k), d_j(k))} \qquad (1)$$

gde je:
$w_{ji}(k)$ - težina veze od *i*-og do *j*-og elementa, u *k*-oj iteraciji
$\Delta w_{ji}(k)$ - promena težine veze u *k*-oj iteraciji
u_{ji} – ulaz *j*-og elementa, $u_{ji}=y_i$
$y_j(k)$ - stvarni izlaz *j*-og elementa
$d_j(k)$ - željeni izlaz *j*-og elementa
μ - mala pozitivna konstanta koja se naziva koeficijent učenja
$E(y_j(k), d_j(k))$ - funkcija greške koja obično sadrži i u_{ji}

Učenje je završeno kada korekcija težine $\Delta w_{ji}(k)$, nakon konačnog broja iteracija k teži nuli. To se dešava kada se svi podaci za trening propuste nekoliko puta u slučajnom redosledu.

Nakon uvođenja osnovnih pojmova definisaćemo opšti konceptualni model, a zatim u definisanom okviru analizirati konkretne modele. Modeli koji će biti analizirani izabrani su tako da obuhvate prethodno navedene slučajeve u pogledu arhitekture i algoritama za učenje. Svaki model biće analiziran kroz dva aspekta:

1) Arhitektura
2) Algoritam za učenje

2. 3. 1. ADALINE (ADAptive LInear NEuron)

Arhitektura

Ukoliko je funkcija prenosa $y = net$ tj. izlaz neurona je jednak ukupnom ulazu, takva komponenta se naziva *adaptivni linearni kombinator (adaptive linear combiner - ALC)*. Jednačina ovog elementa je

$$\boldsymbol{y = wu + B}$$

gde je

$w = [w_0, w_1 \ldots w_m]$ – vektor težina
$u = [u_0, u_1, \ldots u_m]^T$ – vektor ulaza
B – unutrašanja pobuda (bias)

Radi jednostavnosti se uzima da je $u_0=1$ *a* $w_0=B$ tako da je $y=wu$

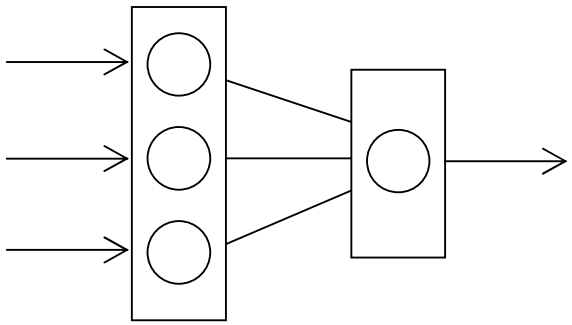
Ukoliko mu na izlazu stavimo prag ili signum funkciju imaćemo:

$$\boldsymbol{y = \begin{cases} wu, wu \geq T \\ 0, wu < T \end{cases}}$$

odnosno

$$\boldsymbol{y = sgn(wu)}$$

Adaline (slika 4) ima dva sloja ovakvih neurona, pri čemu ima samo jedan neuron u izlaznom sloju. Prvi sloj je ulazni i njegova uloga je samo da prosledi ulaze do izlaznog neurona.



Slika 4. Adaline

Adaline se može proširiti u *Madaline (Many Adalines)* kombinovanjem nekoliko *adaline* komponenti.

Algoritam za učenje: Widrow-Hoff (LMS) rule

Algoritam se zasniva na metodi najmanjih kvadrata *(Least Mean Squares)* pa se naziva i *MNK metoda.* Matematička osnova je metoda opadanja gradijenta kojom se vrši iterativno pretraživanje i nalazi minimum funkcije srednje-kvadratnog odstupanja.

Razmotrićemo prvo trening jednog elementa sa linearnom funkcijom prelaza, a zatim dobijeni model uopštiti za više elemenata i nelinearne izlaze.

LMS pravilo se može izraziti kroz sledeće jednačine:

$\varepsilon_p = d_p - y_p$ (1) greška izlaznog neurona za *p*-ti uzorak iz skupa za trening

$$E = \frac{1}{2N}\sum_{p=1}^{n} \varepsilon_p^{\,2}$$ (2) ukupna greška za sve uzorke iz skupa za trening

$$w_{ji}(k+1) = w_{ji}(k) + \mu\varepsilon(k)u_{ji}(k)$$ (3) promena težine

$$\mu(k+1) = \mu(k) - \beta$$ (4) promena koeficijenta učenja

Jednačina (1) predstavlja grešku – razlika između željenog i stvarnog izlaza.
Jednačina (2) je izraz sa srednje kvadratno odstupanje i izračunava ukupnu grešku svih uzoraka za trening. Ukoliko je greška nula ili manja od dozvoljene trening je završen.
Jednačina (3) izračunava i vrši promenu težine veze.
Jednačina (4) postepeno smanjuje koeficijent učenja i na taj način obezbeđuje bolju konvergenciju treninga. Koeficijent učenja može biti i konstantan, ali se ovako postižu bolji rezultati.

Ukoliko imamo više elemenata u izlaznom sloju tada jednačini (2) dodajemo sumu po elementima izlaznog sloja, a jednačina (3) prelazi u vektorski oblik.

Za elemente sa nelinearnim izlazom, pravilo se primenjuje pre nelinearne funkcije prenosa. U slučaju kada imamo prag i/ili odskočnu funkciju, greška se izražava kao razlika nivoa aktivacije i praga.

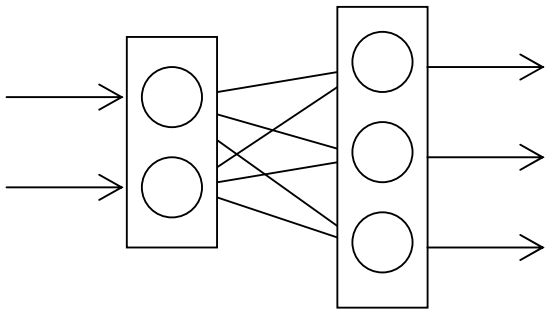
Mogućnosti primene i ograničenja

Neuronske mreže sa prostiranjem signala unapredADALINE se mogu koristiti za rešavanje problema u kojima su ulazni oblici linearno separabilni.

2. 3. 2. PERCEPTRON

Arhitektura

Perceptron se sastoji od dva potpuno povezana sloja neurona – ulaznog i izlaznog. Veza između slojeva je jednosmerna unapred (slika 5).



Slika 5. Perceptron

Elementi procesiranja u perceptronu su neuroni sa pragom, ulaznom funkcijom sume i odskočnom funkcijom prenosa.

Ponašanje neurona u perceptronu opisuje se sledećim jednačinama:

$$net = \sum u_i w_i \qquad \text{- ulazna suma}$$

$$y = \begin{cases} 1, \sum wx - T > 0 \\ 0, \sum wx - T \leq 0 \end{cases} \qquad \text{- izlaz neurona}$$

Iz matematičkog modela se vidi da je izlaz neurona jednak 1 ako je ulazna suma veća od praga, a 0 u suprotnom.

Algoritmi za učenje: Delta pravilo

Prvobitni algoritam za trening perceptrona koji je predložio Rosenblatt - *perceptron learning* glasi ovako:
Težina aktivne veze se povećava kada neuron nije aktivan a treba da bude, a smanjuje kada je aktivan a ne treba da bude. Matematički oblik je:

$$w(k+1) = w(k) + \mu(d(k) - y(k))u(k) \quad (1)$$

Algoritam je skoro identičan LMS algoritmu, ali bitna razlika je što se pri izračunavanju greške uzima izlaz nelinearne funkcije prenosa a ne ukupni ulaz.

Zbog problema i ograničenja koji su uočeni, iz ovoga je proisteklo delta pravilo koje predstavlja proširenje LMS pravila za nelinearne sisteme sa glatkim diferencijabilnim funkcijama prenosa.

$$w(k+1) = w(k) + \mu\varepsilon(k)u(k)f'(net(k)) \quad (2)$$

Delta pravilo dobijeno je primenom pravila za izvod složene funkcije pri izračunavanju parcijalnih izvoda funkcije srednje-kvadratnog odstupanja po težinama. Time se zapravo dobija osetljivost funkcije greške na promenu težina.
Za primenu ovog algoritma potrebno je odabrati pogodnu funkciju prenosa, kojoj se lako (sa malo izračunavanja) može odrediti izvod. Zato se koriste sigmoidne funkcije:

$$y = \frac{1}{1+e^{-x}} \qquad y' = y(1-y) \quad (3)$$

$$y = \frac{1-e^{-x}}{1+e^{-x}} = tanh(x) \qquad y' = 0.5(1-y^2) \quad (4)$$

Ukupna greška mreže, jednaka je sumi srednje-kvadratnih grešaka pojedinačnih neurona iz izlaznog sloja za sve elemente iz skupa za trening:

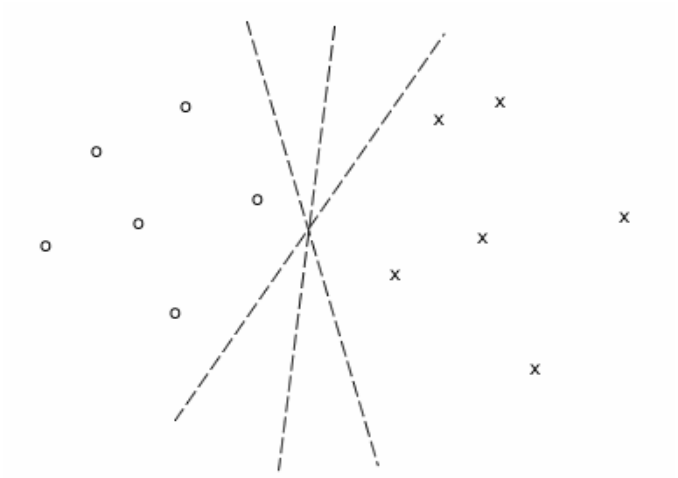
$$E = \frac{1}{2}\sum_{p=1}^{n}\sum_{j=1}^{m} E_{pj}^{\ 2} \quad (5)$$

pri čemu je :
p - indeks elementa za trening
j – indeks neurona u izlaznom sloju

Varijacije perceptrona

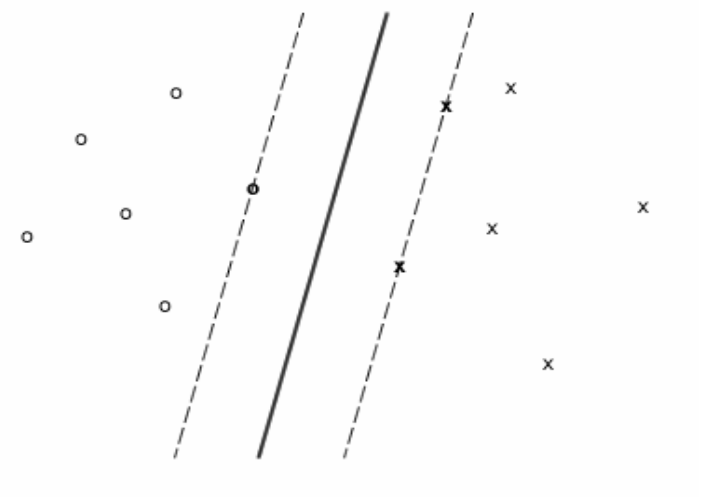
Algoritam za učenje perceptrona se zaustavlja čim se dostigne linearna separabilnost izmedju ulaznih podataka. Ali u mnogim realnim situacijama mi želimo da dostignemo najbolju linearnu podelu iako podaci nisu idealni. **Pocket perceptron** je modifikacija perceptron pravila koja snima najbolji vektor težina u „džep“ (pocket) dok nastavlja da uči. Težine se menjaju samo ako se nađe bolji vektor težina.

Još jedan oblik varijacije perceptrona je **optimal percptron**. Perceptron može da nađe više rešenja linearnog razdvajanja rešenja, sada treba naći ravan koja omogućava najveću toleranciju. Ravan se definiše uz pomoć specijalnih support vectora. Na slici 6 vidimo rešenje bez korišćenja optimal preceptron algoritma.



Slika 6. Rešenje dobijeno bez korišćenja optimal perceptron algoritma

Na slici 7 je prikazano rešenje korišćenjem optimal perceptron algoritma.



Slika 7. Rešenje dobijeno korišćenjem optimal perceptron algoritma

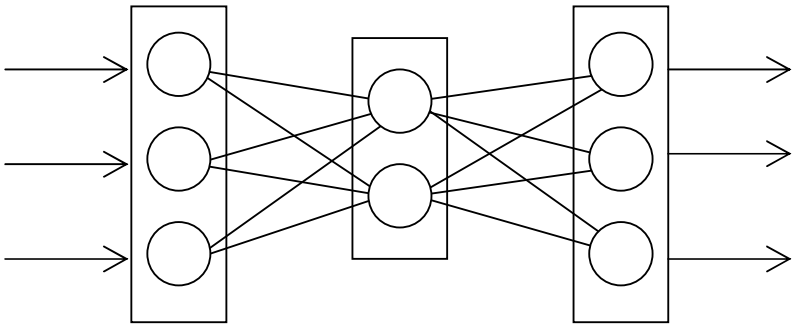
Mogućnosti primene i ograničenja

Kao i ADALINE, i percepton može da rešava samo linearno separabilne problema. To je jedan od osnovnih razloga zbog koga je interesovanje za neuronske mreže splasnulo posle velikog početnog entuzijazma.

2. 3. 3. VIŠESLOJNI PERCEPTRON

Arhitektura

Višeslojni perceptron *(Multi layer perceptron)* pored ulaznog i izlaznog ima i najmanje jedan skriveni sloj neurona. U praksi se koriste mreže sa najviše 3 skrivena sloja. Na slici 8 je prikazan opšti model višeslojnog perceptrona.



Slika 8. Višeslojni perceptron

Algoritam za učenje: Prostiranje greške unazad (Backpropagation of error)

Backpropagation algoritam predstavlja generalizaciju delta pravila i odnosi se na mreže sa skrivenim slojevima.
Promena težina u izlaznom sloju vrši se isto kao kod delta pravila, i koristi se isti izraz za ukupnu grešku. Promena težina u skrivenom sloju (pre izlaznog) izračunava se po sledećem obrascu:

$$w_{ji}(k+1) = w_{ji}(k) + \mu f'(net_j(k)) \left(\sum_a \varepsilon_a(k) f'(net_a(k)) w_{aj}(k) \right) u_{ji} \qquad (1)$$

Izraz u zagradi je suma lokalnih grešaka δ pomnoženih težinom veze, od svih izlaznih neurona. Izraz predstavlja grešku koja se sa izlaznog prenosi na skriveni sloj. To se radi zato jer ne postoji željena vrednost za skriveni sloj, pa nije moguče neposredno izračunati grešku za neurone iz ovog sloja.

MOGUĆA POBOLJŠANJA BACKPROPAGATION ALGORITMA ZA UČENJE:

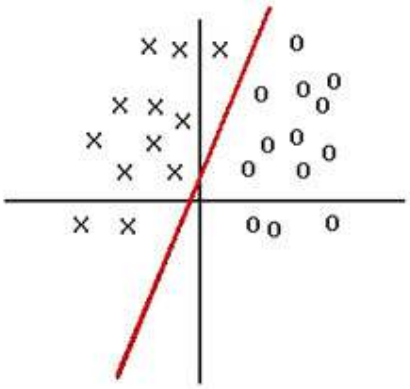
Momentum **–** težine se menjaju u smeru koji je kombinacija trenutnog i prethodnog gradijenta. Prednost ovog algoritma se posebno vidi kada su neki podaci za trening dosta različiti od većine podataka.

Delta bar delta – dozvoljava svakoj težini ponaosob da ima svoju posebnu stopu učenja i da se te stope menjaju sa vremenom kako trening odmiče. Prednost ovog algoritma je znatno smanjenje vremena učenja koje se ogleda u manjem broju iteracija.

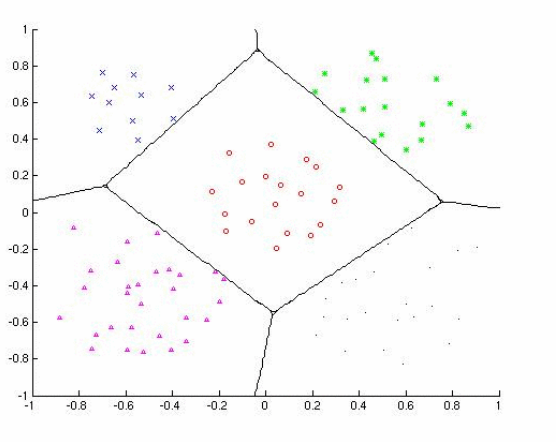
Funkcija transfera – se može takođe poboljšati promenom ciljanih vrednosti. Ukoliko se koristi sigmoidna f-ja, broj iteracija se znatno smanjuje ako kao traženi izlaz ne uzimamo 1 i 0 već vrednosti koje mogu da se dobiju računanjem f-je (sigmoidna f-ja teži 1 i 0 u beskonačnosti).

Mogućnosti primene i ograničenja

U odnosu na mreže koje mogu da se koriste za rešavanje linearno separabilnih problema višeslojni perceptron rešava i dosta komplikovanije probleme. Na slici 9a vidimo rezultat učenja perceptrona a na slici 9b rezultat učenja višeslojnog perceptrona. Uz naprednije učenje (*backpropagation*), ova mreža se koristi za rešavanje širokog spektra različitih naučnih, privrednih i vojnih problema.



Slika 9a. Učenje perceptrona



Slika 9b.Učenje višeslojnog perceptrona

3. ANALIZA I PROJEKTOVANJE

U prvom delu ovog poglavlja su podrobnije opisana moguća poboljšanja *backpropagation* algoritma. U drugom delu je prezentovan konceptualni model softvera za učenja neuronskih mreža sa prostiranjem signala unapred.

3. 1. ANALIZA MATEMATIČKIH I RAČUNARSKIH MODELA ALGORITAMA

3.1.1 MOMENTUM

Kod *backpropagation* algoritma sa momentumom, promena težina je u pravcu koji nastaje iz trenutnog i prethodnog gradijenta f-je prenosa. Ova modifikacija se najbolje vidi kod učenja iz malobrojnih podataka koji su dosta različiti od većinskih. Poželjno je koristiti malu stopu učenja kako bi izbegli velike prekide pravca učenja kada se pojavi par dosta neobičanih paterna za učenje. Na drugoj strani, poželjno je održavati, što je moguće brže učenje, dok god su ulazni podaci slični.

Konvergencija je ponekad brža ako sistem momentum-a dodat formulama za promenu težina. Da bi smo mogli da koristimo momentum, moramo sačuvati jednu ili više grupa prethodnih trening težina. Kod najjednostavnijeg oblika backpopagation-a sa momentumom nove težine za trening na koraku t+1 su bazirane na težinama tening koraka t i t-1. Formula za promenu težina korišćenjem *backpropagation* algoritma sa momentumom su:

$$\omega_{jk}(t+1) = \omega_{jk}(t) + \alpha\delta_k z_j + \mu\left[\omega_{jk}(t) - \omega_{jk}(t-1)\right] \quad (1)$$

Ili

$$\Delta\omega_{jk}(t+1) = \alpha\delta_k z_j + \mu\Delta\omega_{jk}(t) \quad (2)$$

Za izlazne neurone i

$$v_{ij}(t+1) = v_{ij}(t) + \alpha\delta_j x_i + \mu\left[v_{ij}(t) - v_{ij}(t-1)\right] \quad (3)$$

Ili za neurone iz skrivenog sloja,

$$\Delta v_{ij}(t+1) = \alpha\delta_j x_i + \mu\Delta v_{jk}(t) \quad (4)$$

Gde je parametar momentuma **μ** konstanta i treba da bude u rasponu od 0 do 1, ekskluzivno za granične vrednosti.

Momentum dozvoljava mreži da napravi razumljivo velika težinska podešavanja onoliko koliko su korekcije u istom generalnom pravcu za nekoliko ulaza, dok koristeći manji korak učenja sprečavamo veliko povećavanje greške od bilo kod trening paterna. Ono takođe smanjuje verovatnoću da mreža nađe težine koje su lokalni a ne globalni maksimum. Korišćenjem momentuma mreža napreduje ne u pravcu gradijenta prenosne f-je nego u pravcu kombinacije trenutnog gradijenta i prethodnog pravca promene težina. Za razliku od delta-bar-delta poboljšanja, momentum pravi sumu težina koja se eksponencijalno povećava u odnosu na prošle i sadašnje promene.

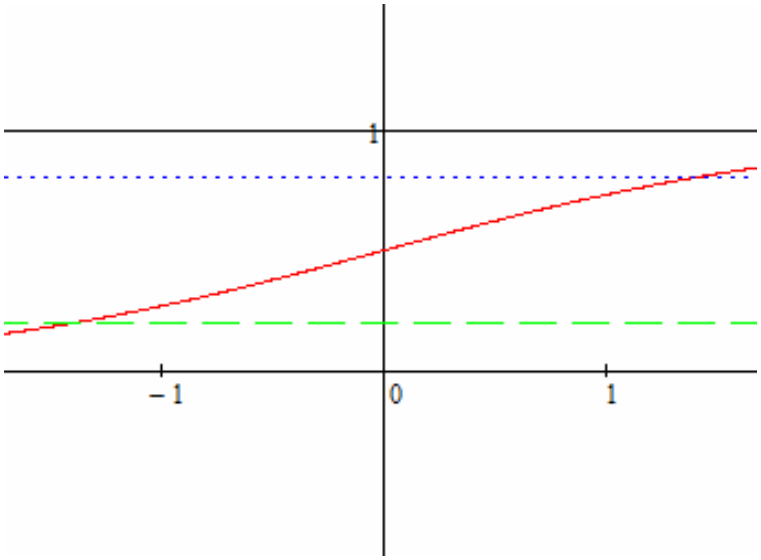
Ograničenja efektivnosti momentuma predstavlja činjenica da korak učenja predstavlja gornju granicu vrednosti za koju se neka težina može promeniti i činjenica da momentum može izazvati da se težina promeni u onom pravcu u kom se povećava greška.

3.1.2. AKUMULATIVNE PROMENE TEŽINA

U nekim slučajevima napredno je akumulisati korekcije na težinama za nekoliko ulaznih uzora (ili čak za čitave epohe ako nema previše uzora) i napraviti jedinstvenu težinsku promenu (jednaku srednjoj vrednosti pojedinačnih promena težina) za svaku težinu ponaosob nego menjati težine posle svakog ulaznog uzora. Ova procedura ima zamućujući efekat na promene. U nekim slučajevima, ovo zamućivanje može povećati šanse za konvergenciju ka lokalnom minimumu. Sa aspekta programiranja je značajn njen uticaj povećanje brzine rada programa koji razvijaju neuronske mreže.

3.1.3. PROMENE VREDNOSTI IZLAZNIH NEURONA

Kada se kao f-ja prenosa korisi sigmoidna f-ja a izlazi iz neurona su granične vrednosti jedan i nula, dostizanje zacrtane minimalne kvadratne greške može da izazove dugotrajno učenje sa mnogo iteracija jer navedene granične vrednosti sigmoidna f-ja dostiže tek u beskonačnosi. Stoga se, radi smanjenja broja iteracija i na taj način ubrzavanja učenja za vrednosti izlaza uzimaju dostižljivi brojevi. Npr. Možemo umesto 0 da uzmemo 0.2 a umesto 1 broj 0.8. Na taj način mreža će brže naučiti zacrtano bez znatnog smanjenja efekta učenja. Na slici 10 je prikazana sigmoidna f-ja i nove vrednosti izlaza neurona.



Slika 10. Sigmoidna f-ja sa promenjenim izlazima neurona

3. 1.4. DELTA-BAR-DELTA

Opšti pristup delta-bar-delta algoritmu kaže da svakoj težini treba omogućiti da ima svoj sopstven korak učenja i dozvoliti da se korak učenja menja sa vremenom kako trening napreduje. Kao dodatak ovome se koriste dve heuristike za određivanje odgovarajućih promena u koraku učenja za svaku težinu. Ako se težina menja u istom pravcu (povećava ili smanjuje) u nekoliko vremenskih koraka, korak učenja za tu težinu bi se trebao povećati (promena težine će biti u istom pravcu ako se parcijalni izvod greške u odnosu na tu težinu ima isti znak nekoliko vremenskih koraka). Pa ipak, ako se pravac promene težine obrnće (promena znaka parcijalnog izvoda) naizmenično korak učenja treba da se smanji. Treba primetiti da se ne tvrdi da će ove heuristike uvek poboljšati performanse mreže, iako će to najčešće raditi.

Ukratko osnovne karakteristike algoritma delta-bar delta su:

- Svaka težina ima svoj korak učenja
- Za svaku težinu se gradijent prethodnog koraka upoređuje sa gradijentom trenutnog koraka
- Ako su gradijenti u istom pravcu onda se korak učenja povećava
- Ako su gradijenti u suprotnim pravcima onda se korak učenja smanjuje
- Delta-bar-delta treba koristiti samo sa akumulativnom promenom težina

Delta-bar-delta pravilo se sastoji od pravila za promenu težina i pravila za promenu koraka učenja. Ako su nam $\omega_{ij}(t)$ težina u vremenu t, $\alpha_{ij}(t)$ korak učenja u vremenu t i E kvadratna greška u vremenu t, delta-bar-delta pravila za promenu težina su sledeæa:

$$\omega_{jk}(t+1) = \omega_{jk}(t) - \alpha_{jk}(t+1)\frac{\partial E}{\partial \omega_{jk}} \quad (1)$$

$$= \omega_{jk}(t) + \alpha_{jk}(t+1)\delta_k z_j$$

Ovo je standardna promena težina kod *backpropagation* algoritma sa dodatkom koji omoguæava svakoj težini da se menja sa različitom merom parcijalnog izvoda greške u odnosu na težinu. Tako, sada pravac promene vektora težina više nije u pravcu negativnog gradijenta f-je prenosa.

Za svaki izlaz mi definišemo ,,delta'':

$$\Delta_{jk} = \frac{\partial E}{\partial \omega_{jk}} = -\delta_k z_j \quad (2)$$

I za svaku skriveni neuron:

$$\Delta_{ij} = \frac{\partial E}{\partial v_{ij}} = -\delta_j x_i \quad (3)$$

Delta-bar-delta pravilo koristi kombinaciju informacija o trenutnom i prošlom izvodu i definiše ,,delta-bar'' za svaki izlazni neuron:

$$\overline{\Delta_{jk}}(t) = (1-\beta)\Delta_{jk}(t) + \beta\overline{\Delta_{jk}}(t-1) \quad (4)$$

I za svaki skriveni neuron:

$$\overline{\Delta_{ij}}(t) = (1-\beta)\Delta_{ij}(t) + \beta\overline{\Delta_{ij}}(t-1) \quad (5)$$

Vrednost parametra β mora biti određena od strane korisnika (0<β<1).

Heuristika koja korak učenja povećava ako se težina menja u istom pravcu u uzastopnim koracima je implementirana povećavanjem koraka učenja (za konstantnu vrednost) ako su $\overline{\Delta_{jk}}(t-1)$ i $\Delta_{jk}(t)$ istog znaka. Korak učenja se smanjuje (za procentualnu vrednost γ od sadašnje vrednosti) ako su $\overline{\Delta_{jk}}(t-1)$ i $\Delta_{jk}(t)$ suprotnog znaka.

Novi korak učenja je dat formulom:

$$\alpha_{jk}(t+1)\begin{cases} \alpha_{jk}(t) + k \;\; ako\; je\; \overline{\Delta_{jk}}(t-1)\Delta_{jk}(t) > 0, \\ (1-\gamma)\alpha_{jk}(t)\; ako\; je\; \overline{\Delta_{jk}}(t-1)\Delta_{jk}(t) < 0, \\ \alpha_{jk}(t) \;\; u\; ostalim\; slučajevima \end{cases} \quad (6)$$

Vrednosti parametara k i γ određuje korisnik. eksperimentalnim putem.

Poređenje metoda je sumirano ovde a dao ih je Jakobs [1988][1]. Rađeno je dvadeset-pet simulacija XOR problema (sa različitim vrednostima početnih težina), korišćenjem mreže sa dva ulaza, dva skrivena sloja i jednim izlaznim neuronom. XOR problem je definisan sa binarnim ulazom, i ciljanim vrednostima 0.1 i 0.9. Korišćene su akumulativne promene težina. Uspešan trening je baziran na kvadratnoj grešci manjoj od (po epohi) od 0.04 prosečno na 50 epoha. Parametri su korišæeni su prikazani u tabeli 2:

	α (korak učenja)	μ (momentum)	K	γ	β
Backpropagation	0.1				
Momentumom	0.75	0.9			
Delta-bar-delta	0.8		0.035	0.333	0.7

Tabela2. Parametri korišćeni za Jacob-ovu simulaciju

Rezultat dat u tabeli 3, sumiran ovde pokazuje da, iako delta-bar-delta modifikacija *backpropagation* algoritma ne postiže uvek konvergenciju brzina učenja algoritma je značajno veća u odnosu na druge algoritme. Treba primetiti da momentum uvek postiže konvergenciju.

Metod	Broj simulacija	Uspešnog učenja	Srednji broj epoha
Backpropagation	25	24	16859.8
Momentum	25	25	2056.3
Delta-bar-delta	25	22	447.3

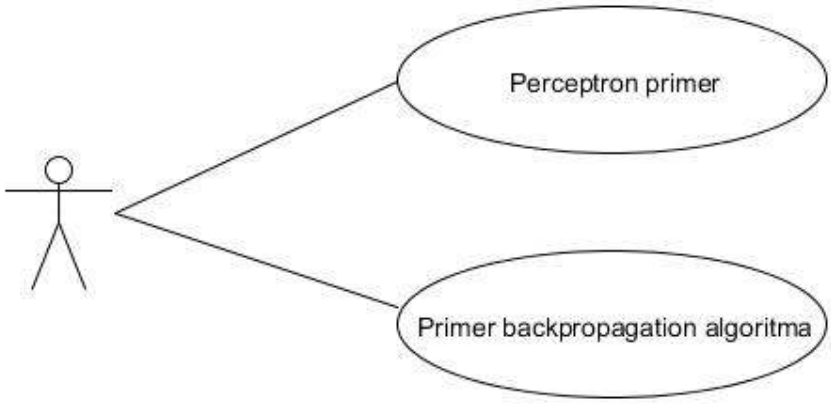
Tabela 3. Rezultat Jacob-ove simulacije

[1] Jacobs, R. A. (1988). „Increased Rates of Convergence Thought Learning Rate Adaption'' Neural Networks

3. 2. KONCEPTUALNI MODEL I DIJAGRAMI KLASA

3.2.1. VERBALNI OPIS I *USE CASE* DIJAGRAM

Realizovati softvare koji na jednostavan način prikazuje učenje neuronskih mreža sa prostiranjem signala u napred sa posebnim osvrtom na mreže tipa perceptron, višeslojni perceptron i *backpropagation* algoritam. Softver ima zadatak da napravi trening set od ulaza koje zada korisnik i da na osnovu njih istrenira mrežu. Na kraju, rezultat treniranja vizuelno prikazati korisniku.
Na slici 11 dat je *use case* dijagram proizišao iz verbalnog opisa:



Slika 11 .Use Case dijagram zasnovan na verbalnom opisu

Softvare treba realizovati da bude *user-friendly* sa grafičkim interfejsom koji:

- Omogućava brzo i lako unošenje parametara učenja
- Prikazuje liniju koja deli prostor kao rezultat učenja u slučaju perceptrona
- Prikazuje težine na vezama između neurona u slučaju perceptrona
- Omogućava lako testiranje mreže u slučaju perceptrona
- Prikazuje učenje i testiranje mreže tipa višesloni perceptron

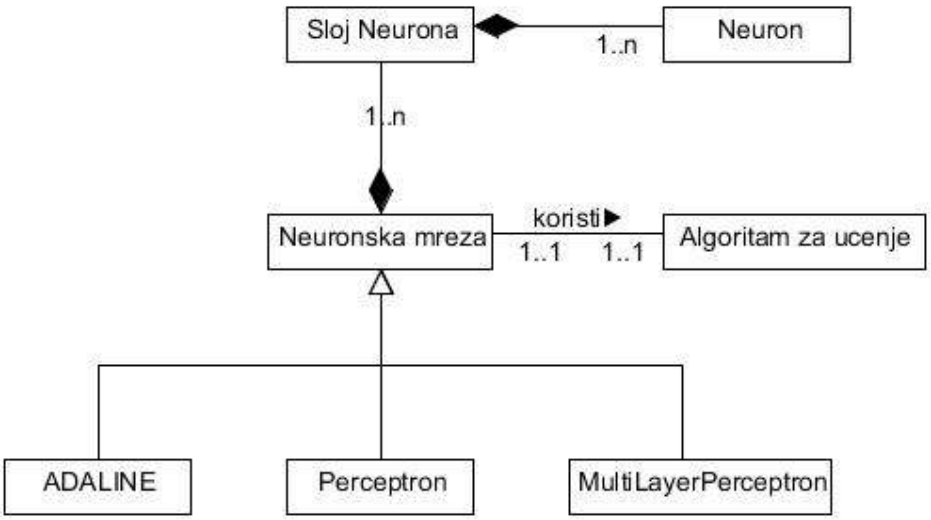
3.2.2. DEFINISANJE FUNKCIONALNIH I STRUKTURNIH ZAHTEVA

Na osnovu date strukturne i funkcionalne specifikacije glavnih predstavnika različitih neuronskih mreža sa prostiranjem signala unapred, definisaćemo opšti model koji će poslužiti kao osnova za njihovo izvođenje.

1) Neuronska mreža se sastoji iz jednog ili više slojeva neurona
2) Sloj je strukturni element mreže i sastoji se od jednog ili više neurona
3) Neuron je osnovni element obrade.
4) Obrada koju neuron vrši određena je funkcijama ulaza i prenosa.
5) Svaka vrsta NM ima odgovarajući algoritam za učenje.

3.2.3. KONCEPTUALNI MODEL

Iz prethodno navedene specifikacije izveden je konceptualni model koji predstavlja osnov za definisanje odgovarajućih klasa. Na slici ispod je dat dijagram klasa izveden iz specifikacije.

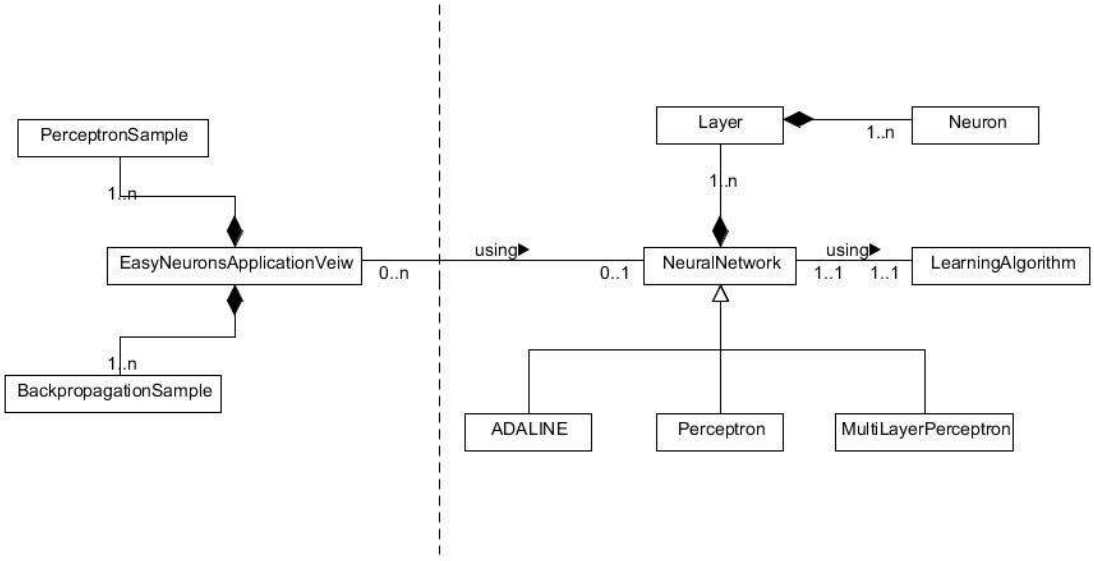


Slika 12.Konceptualni model neuronske mreže

1) **Neuronska mreža** – sastoji se od jednog ili više slojeva međusobno povezanih neurona
2) **Sloj neurona –** skup neurona koji se u simulaciji tretira kao celina
3) **Neuron** - osnovni element obrade u NM. Karakterišu ga ulazna i prenosna funkcija
4) **Algoritam za učenje –** pravilo po kome se vrši podešavanje težina kako bi mreža imala željeno ponašanje
5) **Adaline –** opisuje adaline neuronsku mrežu
6) **Perceptron –** opisuje perceptron neuronsku mrežu
7) **MultiLayerPerceptron –** opisuje višeslojni perceptron neuronsku mrežu

3.2.4. DIJAGRAM KLASA

Na slici 13 dat je dijagram klasa neposredno proizašao iz konceptualnog modela. Prikazana je dvo-nivojska arhitektua sa razdvojenim korisničkim interfejsom i aplikacionom logikom.



Slika 13. Dijagram klasa aplikacije i dvo-nivojska arhitektura

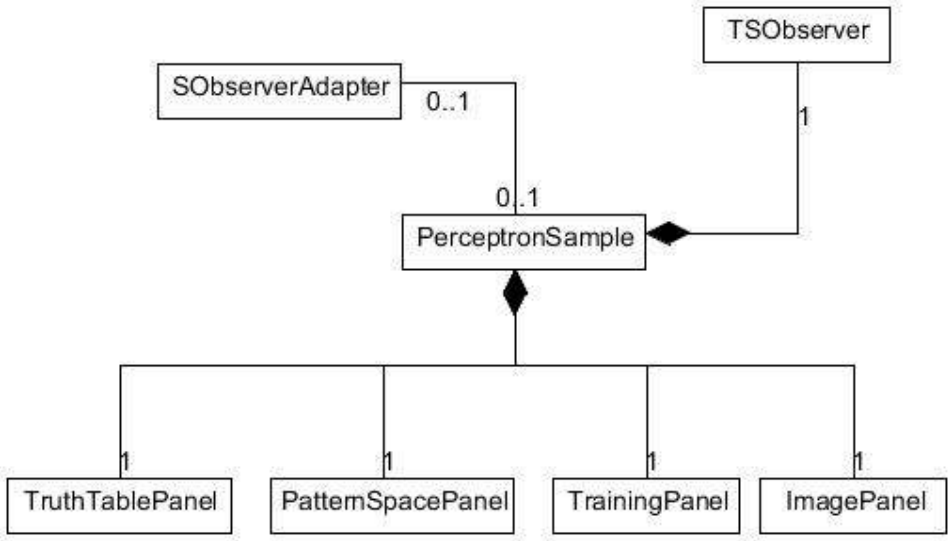
U tabeli 4 ukratko su opisane klase sa slike 13.

Klasa	**Uloga**
NeuralNetwork	Neuronska mreža
Layer	Sloj neurona
LearningAlghoritm	Algoritam za učenje
Adaline	Adaline neuronska mreža
Perceptron	Perceptron neuronska mreža
MultiLayerPerceptron	Višeslojni perceptron neuronska mreža
EasyNeuronsApplicationVew	Desktop frame za grafički prikaz programa
PerceptronSample	Internal frame za prikaz primera perceptrona
BackpropagationSample	Internal frame za prikaz backpropagation učenja

Tabela 4. Klase aplikacije opisane dijagramom sa slike 12.

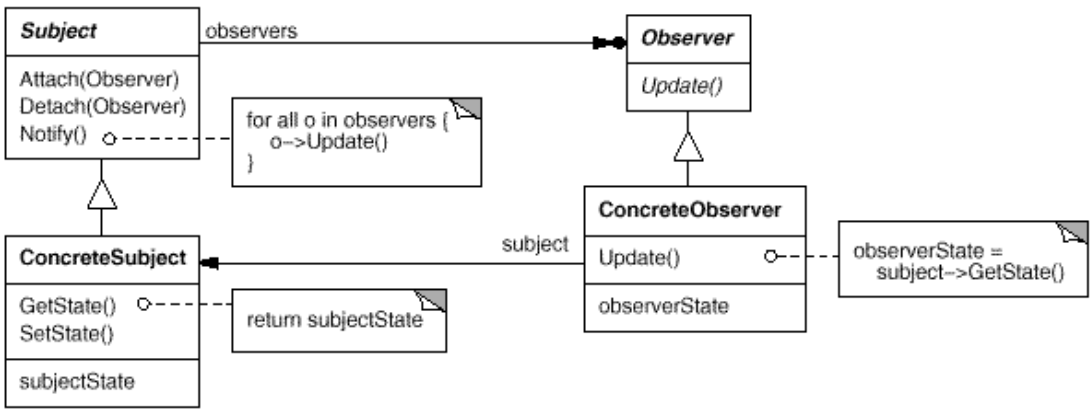
3.2.5. DIJAGRAM KLASA KORISNIČKOG INTERFEJSA

Na slici 14 dat je dijagram klasa dela korisničkog interfejsa koji se odnosi na primer perceptrona.



Slika 14. Dijagram klasa PerceptronSample-a

Paterni korišćeni u projektovanju Perceptron primera su *Observer* kao pomoćni patern koji olakšava implementaciju paterna za razdvajanje korisničkog interfejsa i logike. Klasa *SObserverAdapter* nasleđuje interfejs Observer i prilagođava ga našoj aplikaciji. Ona služi da obavesti klase korisničkog interfejsa da su se trening podaci promenili. Na slici 15 je dat dijagram klasa *Observer* paterna.



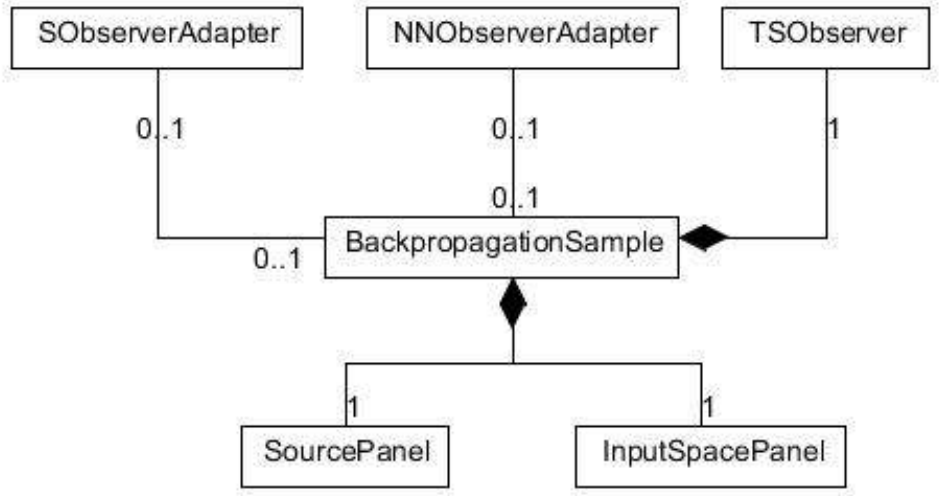
Slika 15. Dijagram klasa Observer Paterna

U tabeli 5 opisane su klase sa slike 14 iz Perceptron primera.

Klasa	Uloga
PerceptronSample	Frame za prikaz mreže
TruthTablePanel	Panel za prikaz training set-a
PatternSpacePanel	Panel za grafički prikaz problema i rešenja
TrainingPanel	Panel za određivanje uslova učenja mreže
ImagePanel	Panel za grafički prikaz mreže
SObserverAdapter	Observer adapter za primere mreže
TSObserver	Observer za training set

Tabela 5. Klase aplikacije opisane dijagramom sa slike 13.

Na slici 16 dat je dijagram klasa dela korisničkog interfejsa iz primera za *backpropagation* algoritam.



Slika 16. Dijagram klasa BackpropagationSample-a

Paterni korišćeni u projektovanju *Backpropagation* primera su takođe *Observer* kao pomoćni patern i patern za razdvajanje korisničkog interfejsa i logike. Klase *SObserverAdapter* i *NNObserverAdapter* nasleđuje interfejs *Observer* i prilagođavju ga našoj aplikaciji. One služi da obavesti klase korisničkog interfejsa da su se trening podaci ili neuronska mreža promenili.

U tabeli 6 opisane su klase sa slike 14 iz *Backpropagation* primera.

Klasa	**Uloga**
BackpropagationSample	Frame za prikaz mreže
SourcePanel	Panel za određivanje uslova učenja mreže
InputSpacePanel	Panel za unos trainig set-a
PSObserverAdapter	Observer adapter za primere mreže
NNObserverAdapter	Observer za neuronsku mrežu
TSObserver	Observer za training set

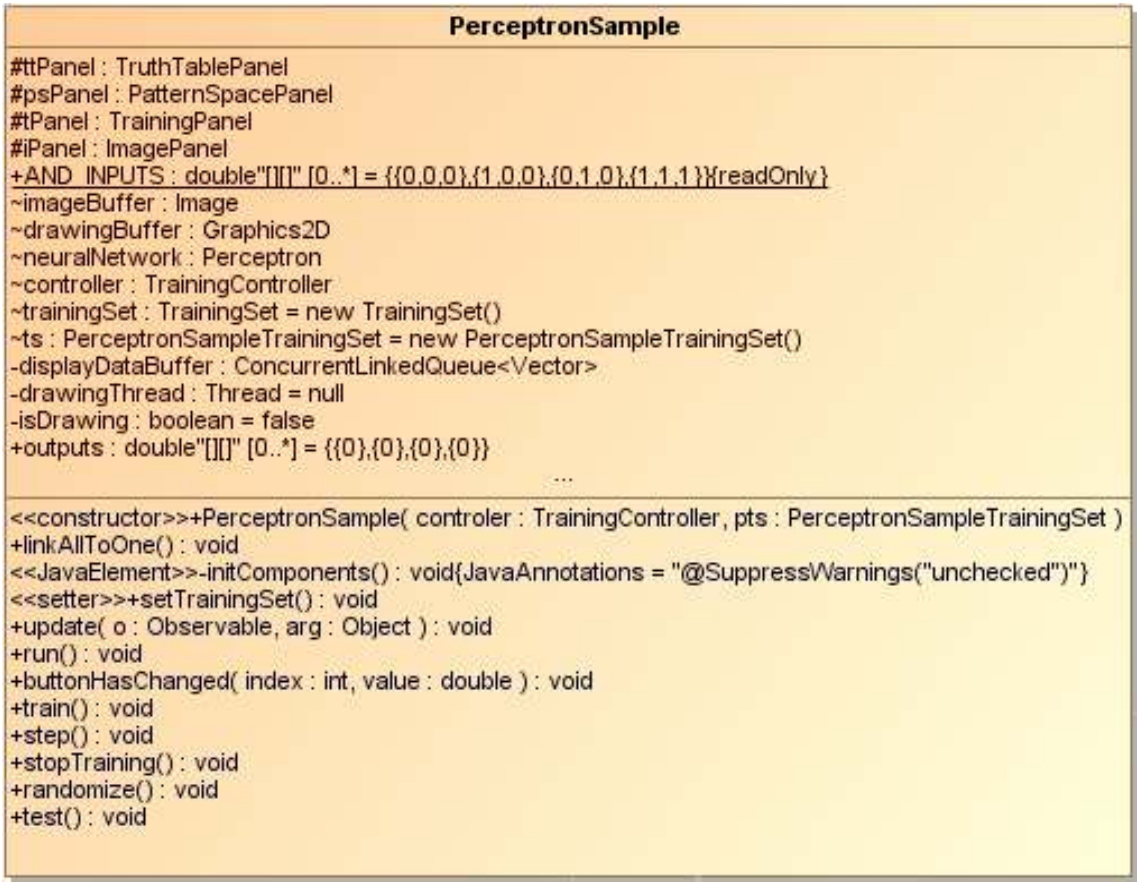
Tabela 6. Klase aplikacije opisane dijagramom sa slike 14.

3.2.6. KLASE APLIKACIONOG OKVIRA

Klasa *PerceptronSample*

Perceptron je neuronska mreža sa prostiranjem signala unapred koja ima dva ili više ulaznih neurona i jedan izlazni neuron. Perceptron možemo posmatrati kao adaptivni ulazno–izlazni sistem, koji uči pomoću odgovarajućeg algoritma koji je određen strukturom mreže.
U Perceptron primeru koristi se neuronska mreža tipa perceptron sa dva ulazna i jednim izlaznim neuronom.

Klasa *PerceptronSample* nasleđuje *JFrame*, sadrži neuronsku mrežu tipa perceptron, agregira objekat koji realizuje grafiku na ekranu i vizualno predstavlja algoritam za učenje. Takođe obezbeđuje izvršenje osnovnih operacija nad mrežom. U okviru klase su implementirani paneli koji grafički prikazuju razne aspekte vezane za perceptron. Na slici 17 je prikazana klasa *PerceptronSample*.



Slika 17. Klasa PerceptronSample

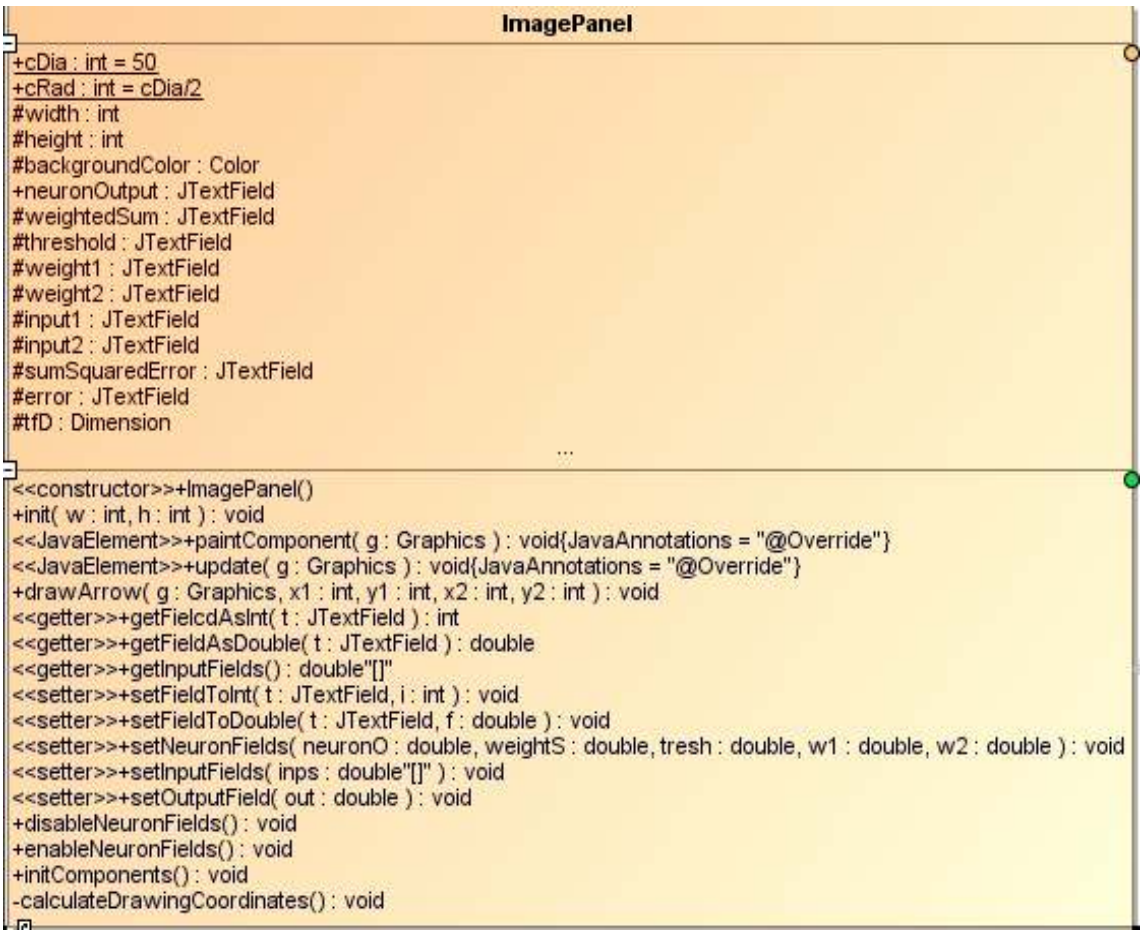
Klasa *TrainingPanel*

Klasa *TrainingPanel nasleđu*je *Java* klasu JPanel, obezbeđuje ubacivanje osnovnih podataka o mreži (korak učenja, maksimalna greška, broj iteracija) i pokreće učenje. Takođe omogućava zaustavljanje učenja i testiranje mreže. Ovde možemo uz pomoć random komande odrediti početne vrednosti težina mreže i na taj način testirati zavisnost dužine učenja u zavisnosti od ovih vrednosti. Na slici 18 je prikazana klasa *TrainingPanel*.

Slika 18. Klasa TrainingPanel

Klasa *ImagePanel*

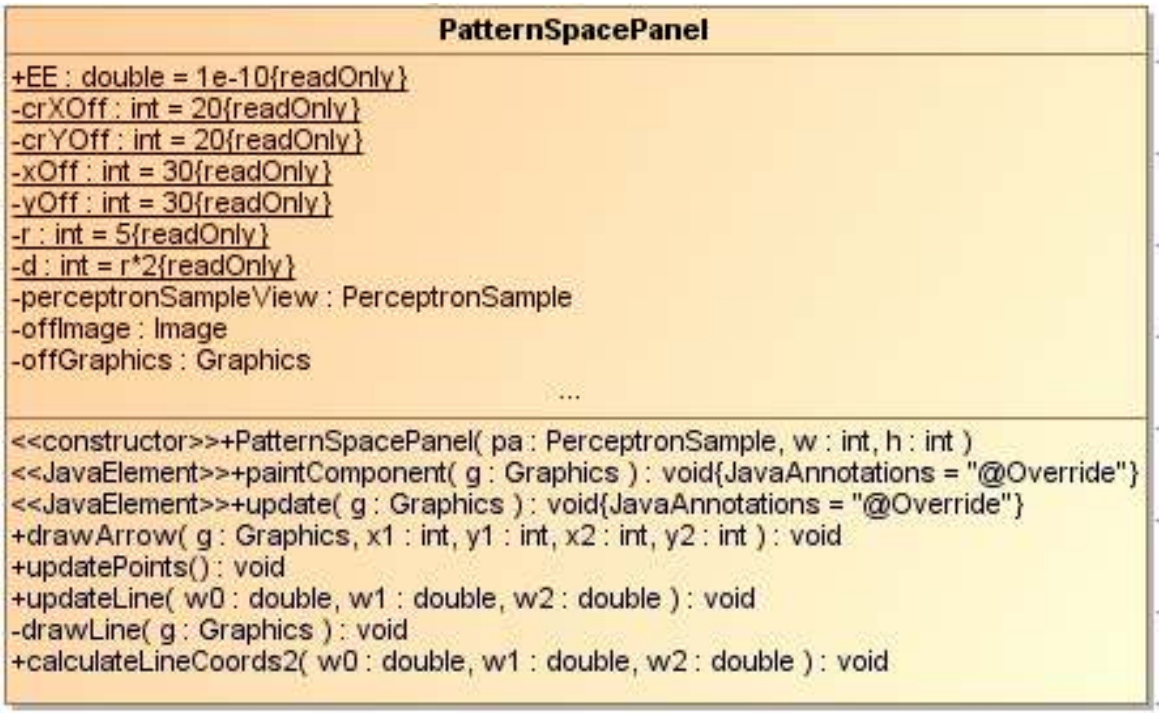
Klasa *ImagePanel* nasleđuje *Java* klasu *JPanel* i obezbeđuje grafički interfejs za kreiranje *training set-a*. Ova klasa omogućava da se za svaki ulazni vektor , koji je dat u parovima vrednosti (po jedna vrednost za svaki ulazni neuron), zada željena vrednost na izlazu neuronske mreže. Na salici 19 je prikazana klasa *ImagePanel*.



Slika 19. Klasa ImagePanel

Klasa *PatternSpacePanel*

Klasa *PatternSpacePanel* nasleđuje *Java* klasu *JPanel*. Ova klasa grafički predstavlja rezultate učenja mreže tj. liniju koja deli prostor trening seta. Ovde je vizuelno predstavljen koordinatni sistem sa ulazima i željenim izlazima iz mreže a nakon svakog koraka učenja formira se linija na osnovu težina mreže koja deli prostor na koordinatnom sistemu. Kod jednostavnijih neuronskih mreža sa prostiranjem signala unapred kao što je perceptron, linijom se deli prostor za čiji ulaz je vrednost izlaza 1 od onog gde je vrednost izlaza 0. Na slici 20 je prikazana klasa *PatternSpacePanel*.

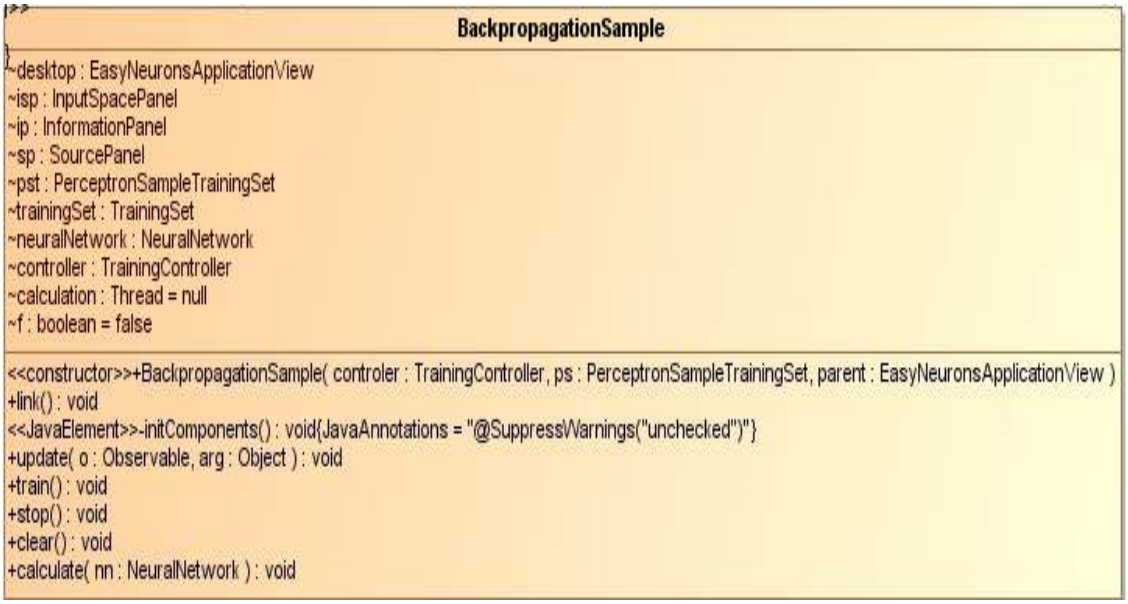


Slika 20. Klasa PaternSpacePanel

Klasa *BackpropagationSample*

Višeslojni perceptron je neuronska mreža sa više slojeva neurona i prostiranjem signala unapred. Ova mreža za učenje koristi *backpropagation* algoritam sa sigmoidnom aktivacionom f-ijom.

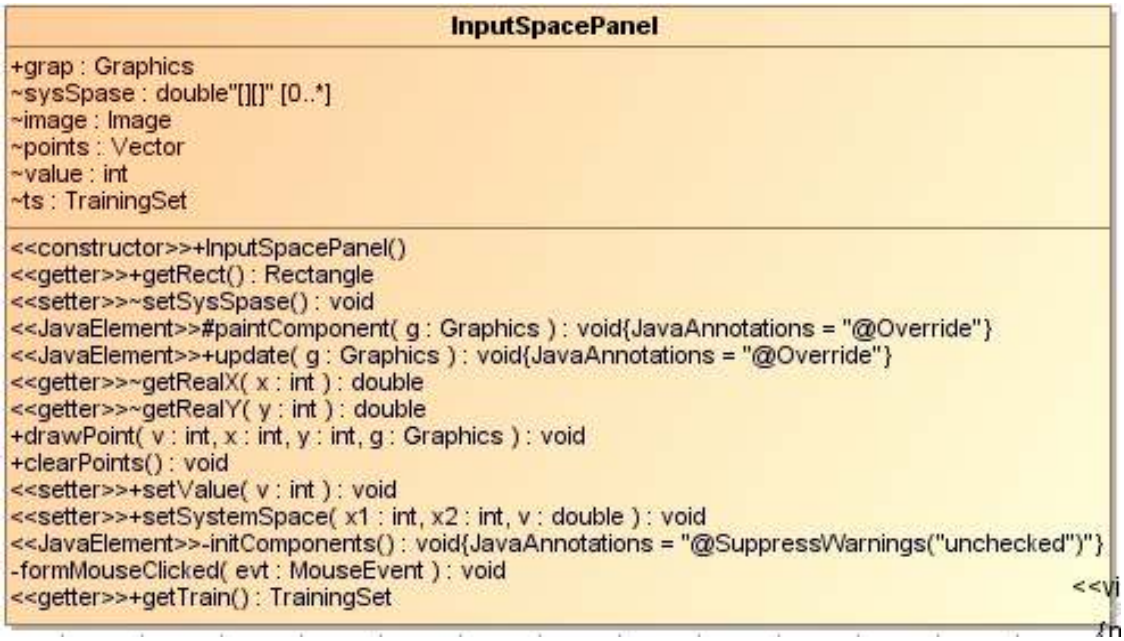
Klasa *BackpropagationSample* nasleđuje *Java* klasu *JFrame*, sadrži neuronsku mrežu višeslojni perceptron i agregira objekat koji realizuje grafiku na i vizualno predstavlje algoritam za učenje. Paneli koji su deo ovog *frame*-a obezbeđuju osnovne operacija nad mrežom. Na slici 21 je prikazana klasa *BackpropagationPanel*.



Slika 21. Klasa BackpropagationSample

Klasa *InputSpacePanel*

Klasa *InputSpacePanel* nasleđuje *Java* klasu *JPanel*. Skup elemenata za trening je skup vektora ulaza, odnosno skup parova vektora (u, y) koji predstavljaju ulaz i željeni izlaz. Na panelu kliktanjem miša definišemo skup ulaza i izlaza. To mogu biti kontinualne vrednosti od 0 do 1 što se tiče ulaza i fiksne 0 ili 1 što se tiče izlaza. Na slici 22 je prikazana klasa *InputSpacePanel.*



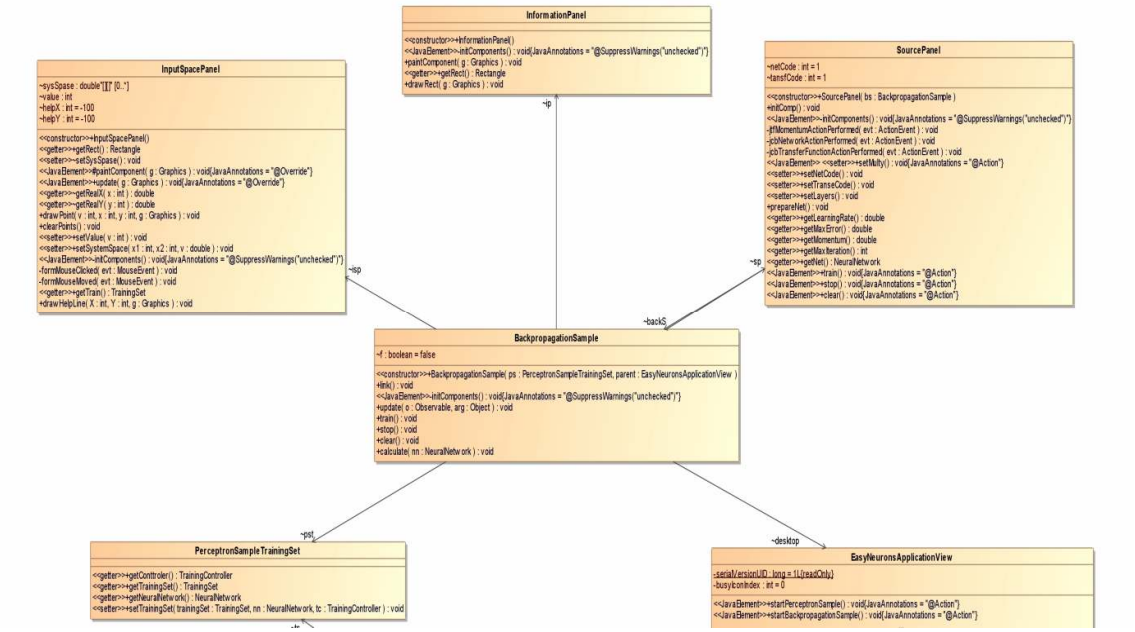
Slika 22. Klasa InputSpacePanel

Klasa *SourcePanel*

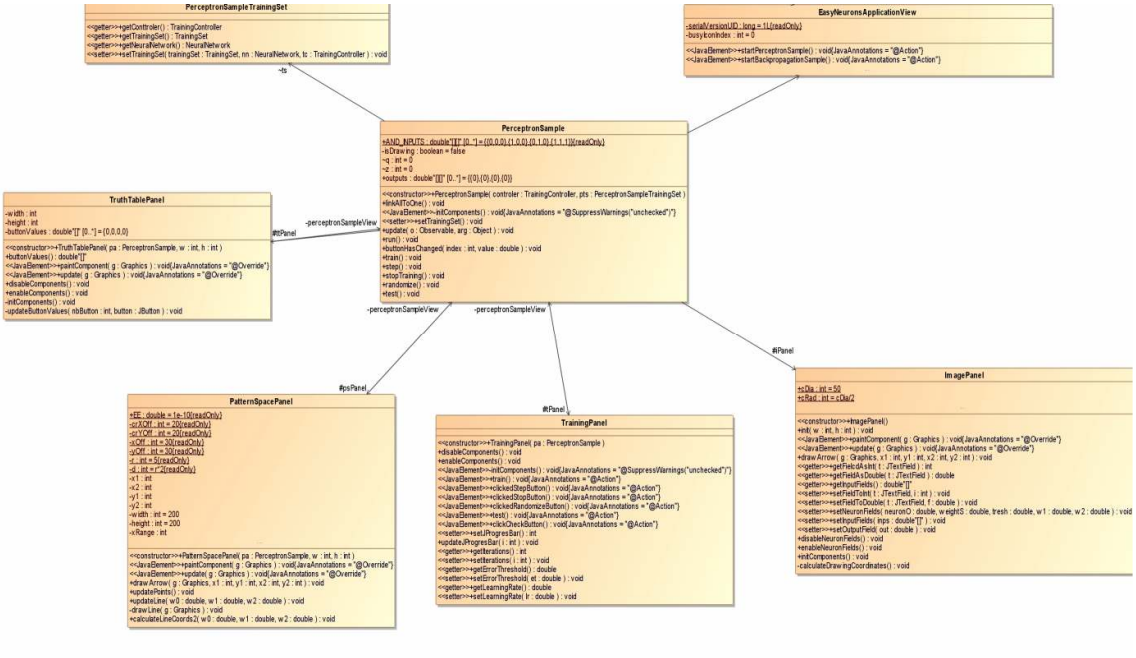
Klasa *SourcePanel* nasleđuje *Java* klasu JPanel. Ova klasa služi za unos podataka neophodnih za specifikaciju mreže: vrsta prenosne f-je, maksimalna greška, broj iteracija, momentum. Takođe ovde pokrećemo i zaustavljamo algoritam za učenje. Na slici 23 je prikazana klasa *SourcePanel.*

Slika 23. Klasa SourePanel

Na slikama 24 i 25 su prikazane sve klase koje čine *PerceptronSample* i *BackpropagationSample* zajedno sa klasom *EasyNeuronsApplicationView* koja ih poziva.

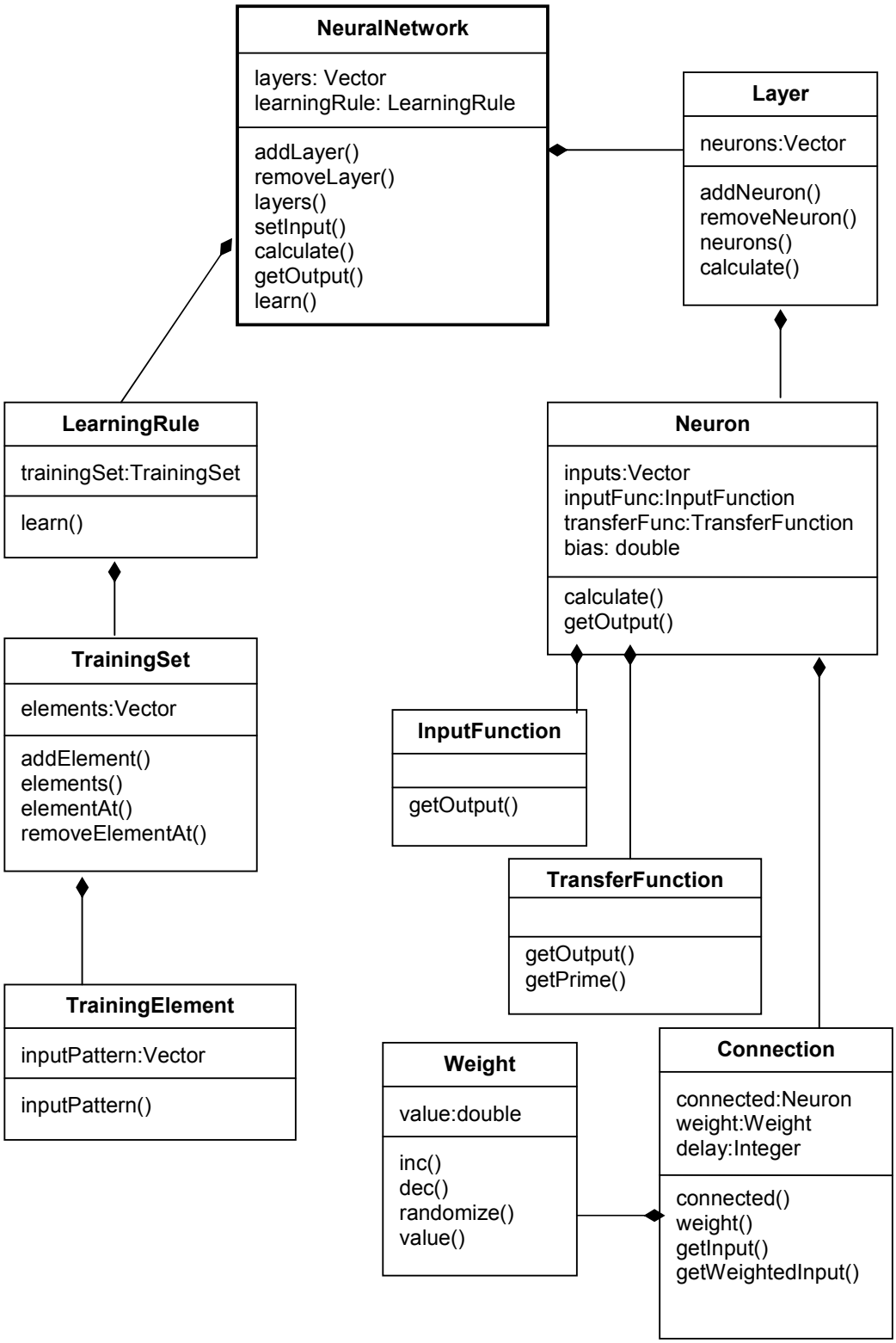


Slika 24. Globalni pogled - klase koje čine Perceptron Sample



Slika 25. Globalni pogled - klase koje čine Backpropagation Sample

Na slici 26 dat je globalni pogled na klase koje čine *Neuroph* biblioteku koja je korišćena pri izradi softvera.



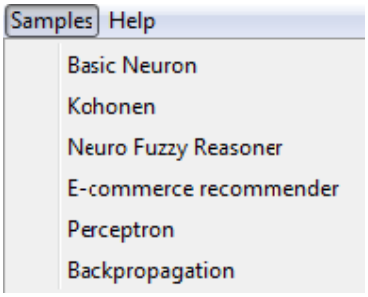
Slika 26. Globalni pogled - osnovne klase biblioteke Neuroph

4. IMPLEMENTACIJA

U ovom poglavlju je opisana implementacija klasa Perceptron primera-a i primera *backpropagation* algoritma. Dati su najzanimljiviji delovi koda i primeri korišćenja.

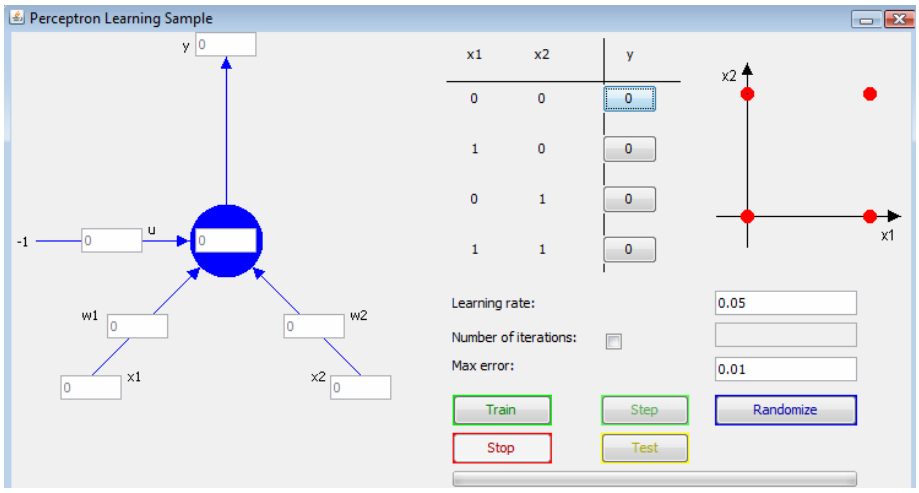
4.1. PERCEPTRON PRIMER

Pomoću biblioteke *Neuroph* napravljen je primer *PerceptronSample* koja demonstrira osnovne mehanizme učenja u radu sa percepton neuronskom mrežma sa prostiranjem signala unapred. Na slici 27 ispod prikazan je način za pozivanje ovog primera.



Slika 27. Sample podmeni

Na slici 28 dat je izgled prozora *PerceptronSample*-a



Slika 28. PerceptronSample

Osnovne operacije koje primer perceptrona izvršava su:

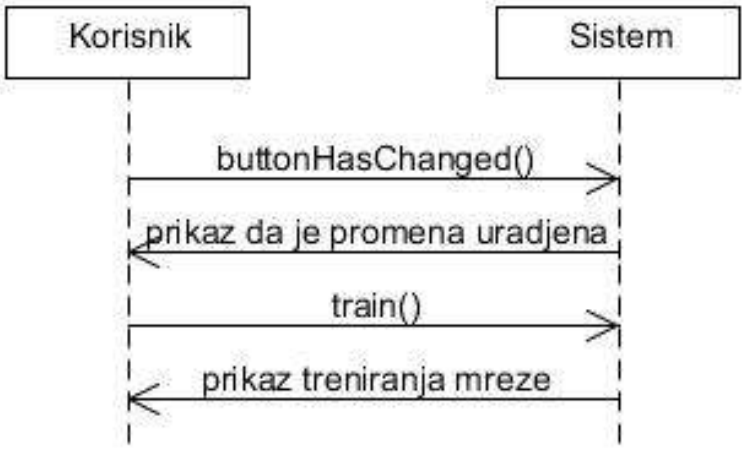
- Učenje mreže
- Testiranje naučene mreže

Na osnovu skupa za trening koji korisnik unosi, *PerceptronSample* uči mrežu sa dva ulaza i jednim izlazom koja koristi step aktivacionu funkciju. Trening ulazi su uvek isti ({0,0}; {1,0}; {0,1}; {1,1}) dok trening izlaze određuje korisnik.

Učenje mreže

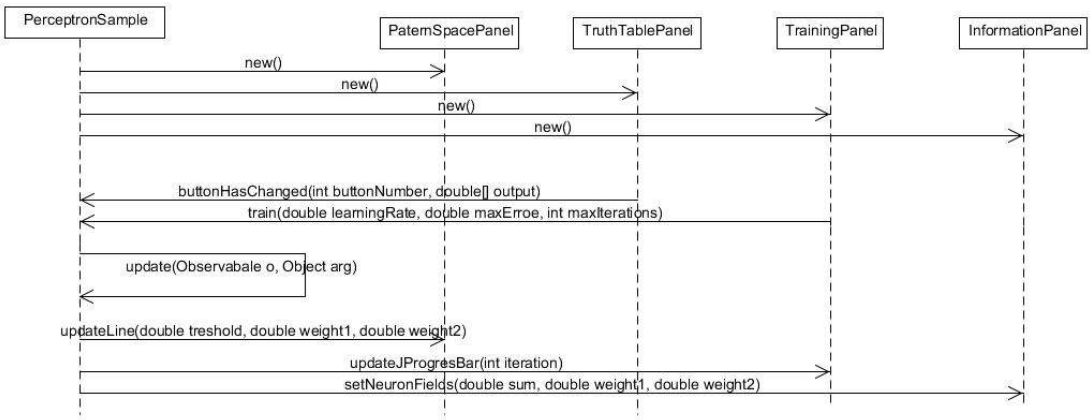
1. Korisnik unosi podatke o željenim izlazima
2. Korisnik poziva sistem da istrenira mrežu
3. Sistem prikazuje korisniku rezultat treniranja mreže

Na slici 29 je prikzan dijagram sekvenci za scenario učenja perceptron mreže.



Slika 29. Dijagram sekvenci za scenario učenja

Na slici 30 je prikazan detaljan dijagram sekvenci za scenario učenja mreže sa svim klasama iz korisničkog interfejsa.



Slika 30. Detaljan dijagram sekvenci za PerceptronSample

U trećem poglavlju je već objašnjena uloga *Observer* paterna. Metoda *update(Observable o, Object arg)* koju nasleđuje klasa *PerceptronSample* je detaljno prikazana na slici 31 zajedno sa kodom koji je značajan za razumevanje rada ove metode.

```
public class PerceptronSample extends javax.swing.JInternalFrame
implements Observer, Runnable{
...
     public double[] [] outputs = {{0},{0},{0},{0}};

     public void buttonHasChanged(int index, double value) {
          outputs[index][0] = value;
                psPanel.updatePoints();
     }

 public void train(double learningRate, double maxError, int
maxIterations) {
        if (!trainingSet.isEmpty()) trainingSet.clear();
        trainingSet.addElement(new SupervisedTrainingElement(
                                              new      double[]{0,      0},
outputs[0]));
        trainingSet.addElement(new SupervisedTrainingElement(
                                              new      double[]{1,      0},
outputs[1]));
        trainingSet.addElement(new SupervisedTrainingElement(
                                              new      double[]{0,      1},
outputs[2]));
        trainingSet.addElement(new SupervisedTrainingElement(
                                              new      double[]{1,      1},
outputs[3]));
        controller.setTrainingSet(trainingSet);
        controller.setStepDRParams(learningRate,maxError,maxIterations);
        controller.train();
     }

   public void update(Observable o, Object arg) {
```

```
        double l =
((ThresholdNeuron)neuralNetwork.getLayerAt(1).getNeuronAt(0)).getThresh();

          double w1 =
neuralNetwork.getLayerAt(1).getNeurons().get(0).getWeightsVector()
                                                          .get(0).getValue();
          double w2 =
neuralNetwork.getLayerAt(1).getNeurons().get(0).getWeightsVector()
                                                          .get(1).getValue();
         psPanel.updateLine(l,w1,w2);
      }
}
```

Slika 31. Segment programskog koda klase PerceptronSample

Iz datog koda sa slike 31 se vidi:

- Da se trening podaci određuju tek pošto se pozove metoda *train(double learningRate, double maxError, int maxIterations)*
- Da se parametri treninga takođe određuju tek što je pozvana metoda *train*
- Da se metoda observer poziva prilikom svake promene na mreži i da ona dalje poziva metodu koja iscrtava liniju rešenja na ekranu

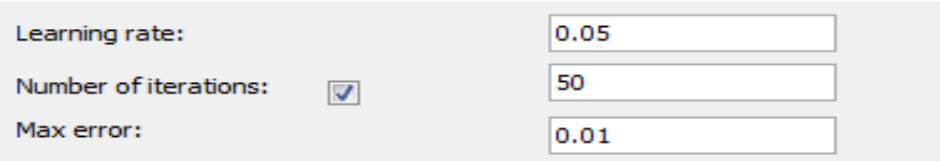
Iscrtavanje linije se obavlja u klasi *PatternSpacePanel* i na slici 32 je dat segment koda iz te klase sa celom metodom *updateLine(double l, double w1, double w2).*

```
public class PatternSpacePanel extends JPanel{
...
int x1,x2,y1,y2;
    public void updateLine(double w0, double w1, double w2) {
        calculateLineCoords(w0,w1,w2);
        drawLine(offGraphics);
    }

     public void calculateLineCoords(double w0, double w1, double
w2)
    {
             x1 = -50;
             x2 = 150;
             y1 = (int) 100-(((w0 - w1 * (-0.5))/ w2)*100);
             y2= (int) 100-(((w0 - w1 * 1.5) / w2)*100);
    }
    private void drawLine(Graphics g) {
        g.setColor(Color.blue);
        g.drawLine(x1,y1,x2,y2);
     }
}
```

Slika 32. Segment programskog koda klase PatternSpacePanel

Na slici 33 dati su parametri učenja neuronske mreže za NOT AND problem, rezultat učenja je dat na slici 34.

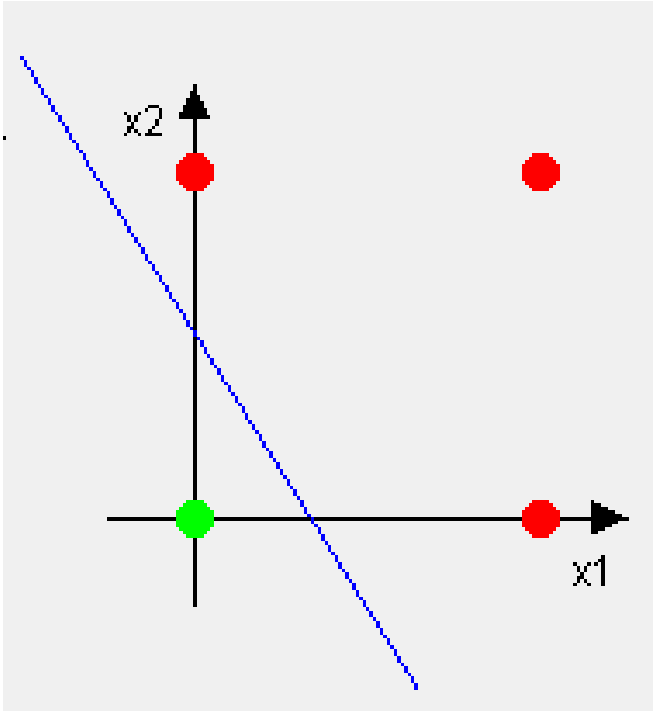


Slika 33. Deo prozora PerceptronSample-a gde se unose parametri

Zadati parametri algoritama za učenje su:

1. Korak učenja je 0.05
2. Maksimalan broj iteracija je 50
3. Maksimalna greška je 0.01

Rezultat iscrtavanja je:



Slika 34. Rezultat učenja perceptrona

Testiranje mreže

Testiranje naučene mreže se vrši pošto se završi proces učenja, tako što se u mrežu ubacuju određeni ulazi i posmatra se da li izlaz iz mreže odgovara traženom izlazu. Ulazi su uvek isti i predstavljaju bilo koji od četiri para: (0, 0), (1, 0), (0, 1), (1, 1) a izlazi mogu biti 0 ili 1.

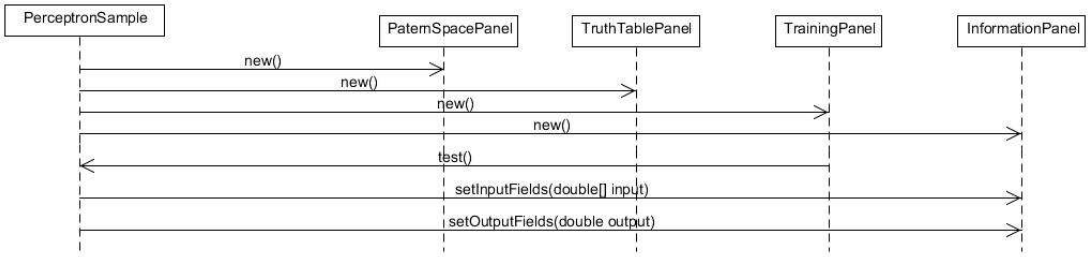
Scenario testiranja mreže

1. Korisnik unosi podatke o trening ulazima
2. Korisnik poziva sistem da testira mrežu
3. Sistem prikazuje korisniku rezultat testiranja mreže

Na slici 35 je prikazan dijagram sekvenci za scenario učenja perceptron mreže.

Slika 35. Dijagram sekvenci za scenario testiranja

Na slici 36 je prikazan detaljan dijagram sekvenci za scenario testiranja mreže sa svim klasama iz korisničkog interfejsa.



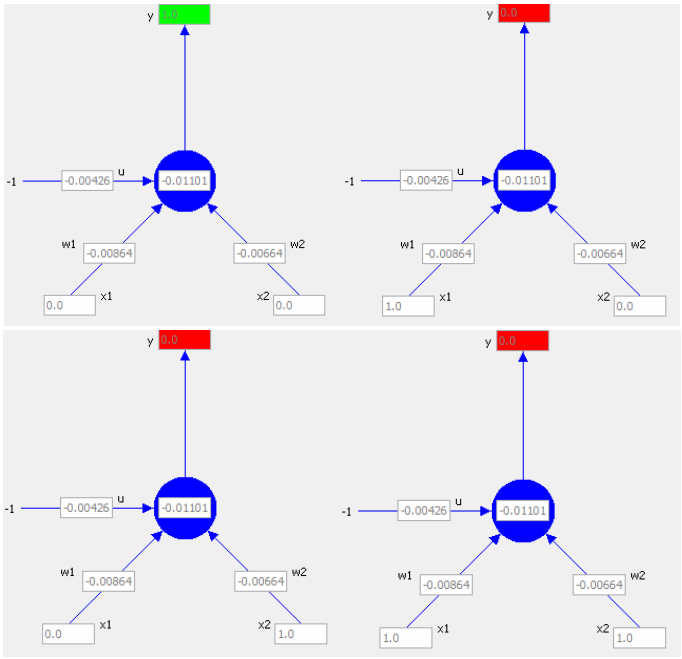
Slika 36. Detaljan dijagram sekvenci za scenario testiranja

Na slici 37 dat je kod metode *test()* klase *PerceptronSample.*

```
public class PerceptronSample extends javax.swing.JInternalFrame
implements Observer, Runnable{
...
int z = 0;
    public void test() {
            if(z>3) z=0;
            switch (z) {
                  case       0:       controller.setInput("0       0");
iPanel.setInputFields(new double[]{0 , 0});break;
                  case       1:       controller.setInput("1       0");
iPanel.setInputFields(new double[]{1 , 0});break;
                  case       2:       controller.setInput("0       1");
iPanel.setInputFields(new double[]{0 , 1});break;
                  case       3:       controller.setInput("1       1");
iPanel.setInputFields(new double[]{1 , 1});break;
            }
       double sum = neuralNetwork.getOutput().firstElement();
      iPanel.setOutputField(sum);
        z++;
    }
}
```

Slika 37. Segment programskog koda klase PerceptronSample

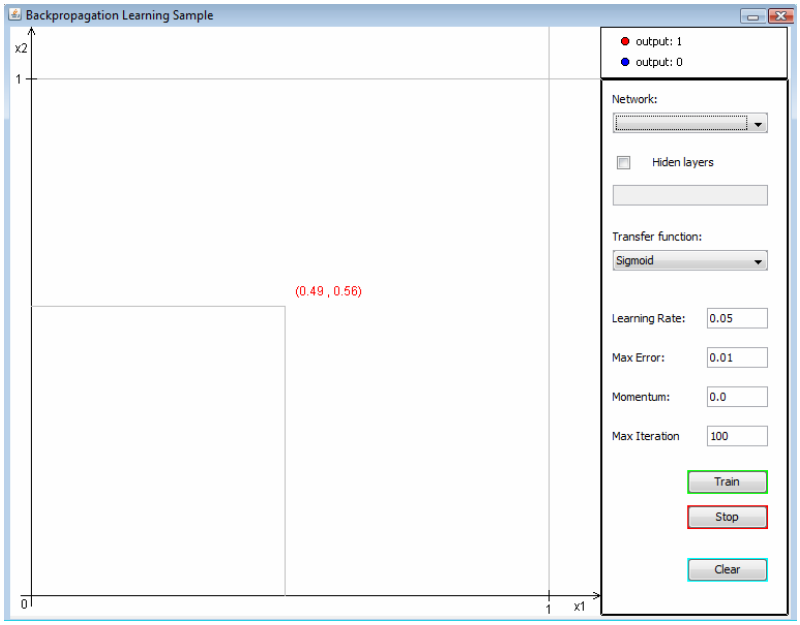
Iz datog koda se može videti da se svakim pozivanjem metode *test()* brojač z povećava za 1 a upravi od vrednosti brojača zavisi koji će od četiri moguća ulaza testirati. U slučaju da je brojač veći broj od 3 njegova vrednost će se vratiti na 0. Na slici 38 dati su rezultati četiri uzastopna poziva metode *test()* u okviru klase *PerceptronSample*.



Slika 38. Tezultat četiri uzastopna poziva metode test()

4.2. BACKPROPAGATION PRIMER

Kao i prethodni Perceptron primer, i *Backpropagation* primer je napravljen korišćenjem biblioteke *Neuroph*. Ovaj primer služi za prikaz mehanizma učenja *backpropagation* algoritmom koji se koristi u višeslojnim neuronskom mrežma sa prostiranjem signala unapred. Na slici 39 dat je prikaz korisničkog interfejsa *BackpropagationSample*-a.



Slika 39. Backpropagation sample

Osnovne operacije koje izvršava primer *backpropagation* algoritma su:

- Učenje mreže
- Prikaz podeljenog prostora kao rezultat testiranja mreže

Učenje mreže se zasniva na ulazima koje korisnik zadaje kliktanjem miša (crtanjem tačaka) na koordinatni sistem koji je dat u okviru korisničkog interfejsa. Broj ulaza koji se može zadati nije ograničen. Trening ulaz je određen koordinatama sa x1 i x2 ose dok je izlaz određen bojom tačke (crvena boja predstavlja vrednost izlaza 1 a plava vrednost izlaza 0). Mreža ima dva ulaza i jedan izlaz a broj srednjih slojeva i neurona u okviru njih takođe zadaje korisnik.

Učenje mreže

Kao što je već rečeno u uvodu ovog poglavlja, učenje se zasniva na podacima koje korisnik unese. Količina tih podataka ničim nije ograničena što je znatna prednost u odnosu na prethodni primer (*PerceptronSample*) jer je interakcija korisnika i sistema veća pa sa tim se povećavaju i mogućnosti za praktičnu primenu ovog primera.

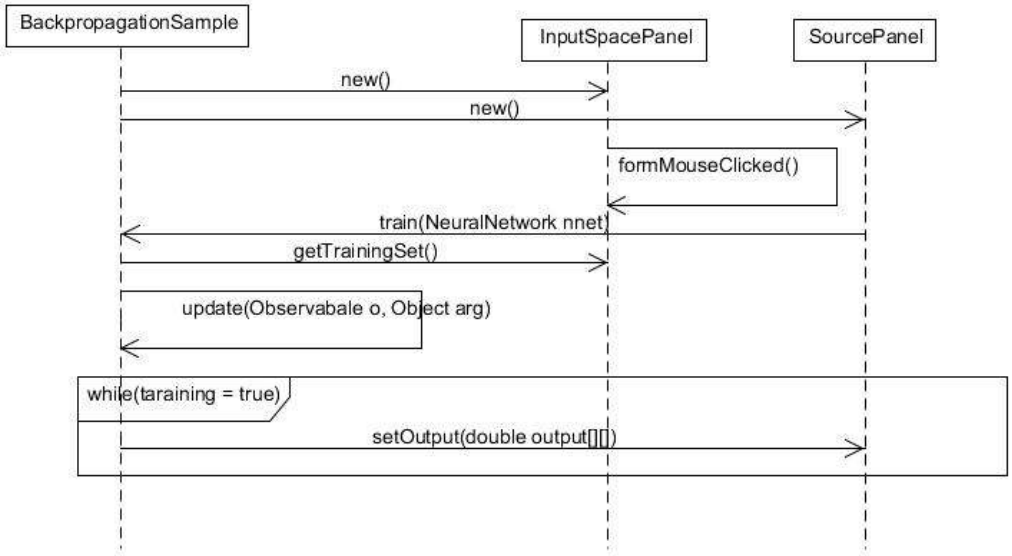
Scenario učenja mreže

1. Korisnik unosi podatke o trening ulazima i izlazima
2. Korisnik poziva sistem da istrenira mrežu
3. Sistem prikazuje korisniku rezultat testiranja mreže

Na slici 40 je prikazan dijagram sekvenci za scenario učenja mreže višeslojni perceptron sa *backpropagation* algoritmom za učenje.

Slika 40. Dijagram sekvenci za scenario učenja mreže

Na slici 41 je prikazan detaljan dijagram sekvenci za scenario učenja mreže, sa svim klasama korisničkog interfejsa.



Slika 41. Detaljan dijagram sekvenci za scenario učenja mreže

Iz detaljnog dijagrama sekvenci se vidi da dok god traje učenje, a na osnovu implementiranog *observer* paterna, *update* metoda prati svaku promenu na neuronskoj mreži (svaki korak učenja) i da se posle toga iscrtava rezultat testiranja mreže između svaka dva koraka. Metoda *update(Observable o, Object arg)* je data na slici 42.

```
public class BackpropagationSample extends javax.swing.JInternalFrame
implements Observer{
   ...
  boolean f = false;
   public void update(Observable o, Object arg) {
        NeuralNetwork nnet = controller.getNetwork();
        this.displayDataBuffer.add(nnet);
         if(!f){
             Thread firstCalculation = new Thread("Calculation"){
                   int ii=0;
                 @Override
                   public void run() {
                      f = true;
                      while(!controller.isStoppedTraining()){
                        NeuralNetwork nn = displayDataBuffer.poll();
                                  calculate(nn);
                      }
                       f = false;
                    }
             };
             firstCalculation.start();
        }
}
```

Slika 42. Deo koda klase BackporpagationSample sa update medom

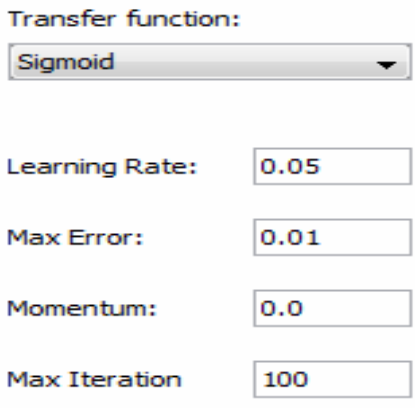
Iz datog koda se vidi da prilikom svake promene na mreži trenutni skup težina (koji se nalazi u objektu *NeuralNetwork*) stavlja u jedan red. Prilikom prvog poziva metode update počinje sa radom jedna nit koja uzima iz reda prvi objekat *NeuralNetwork* klase i prosleđuje ga metodi *calculate(NeuralNetwork nn)* koja je data na slici 43.

```
public class BackpropagationSample extends javax.swing.JInternalFrame
implements Observer{
   ...
   public void calculate(NeuralNetwork nn){
    if (nn!=null){
          for(int i=0; i<50;i++)
               for (int j=0;j<50; j++) {
                       double x = 0.0 + i*0.02;
                       double y = 1.0 - j*0.02;
                      nn.setInput(new double[]{x, y});
                      nn.calculate();
                     double v =
nn.getLayers().lastElement().getNeurons().firstElement().getOutput();
               }
      }
  }
}
```

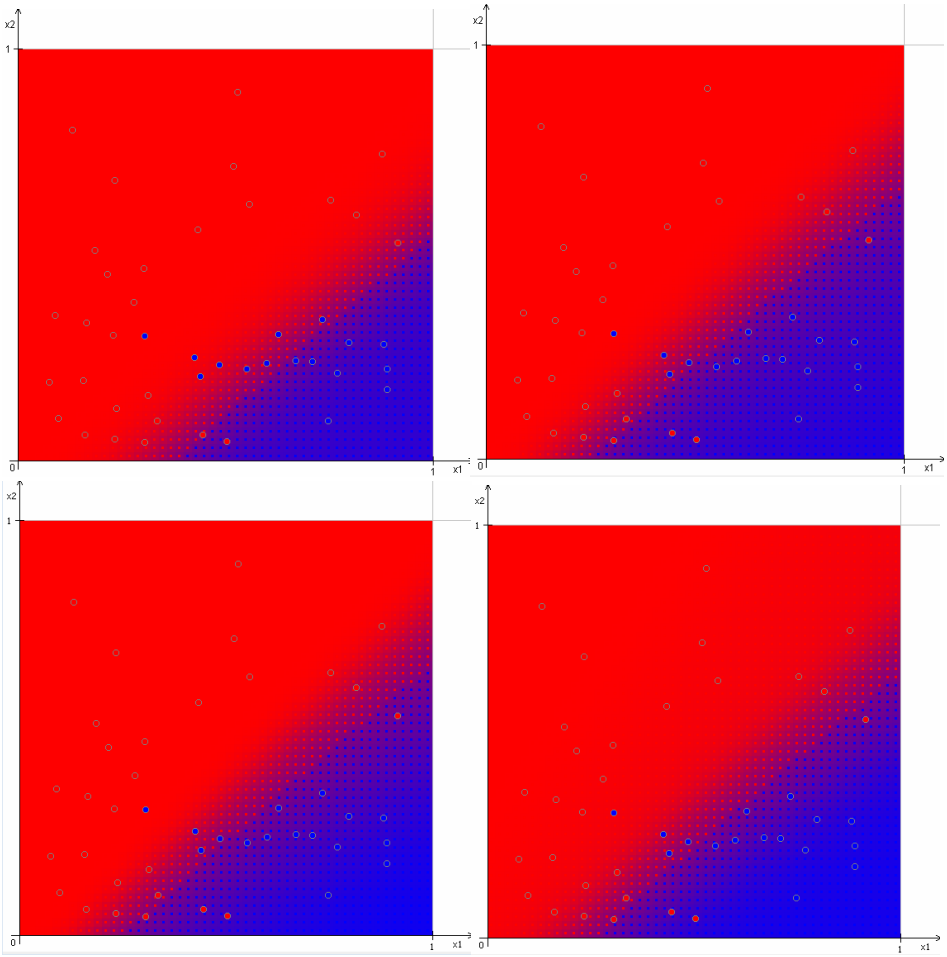
Slika 43. Deo koda klase BackporpagationSample sa calculate medom

Sa slike 43 se vidi da se sa 2500 različitih vrednosti ulaza testira mreža na svakom svom koraku. Rezultati tog testiranja se vide iscrtani na ekranu. Na slici 45 je dat primer testiranja mreže posle 100 iteracija sa zadatim parametrima (slika 44):

1. Prenosna f-ja je sigmoidna
2. Korak učenja je 0.05
3. Maksimalan broj iteracija je 5000
4. Momentum je 0.75
5. Maksimalna greška je 0.01



Slika 44. Deo prozora BackpropagationSample-a gde se unose parametri



Slika 45. Deo prozora BackpropagationSample-a gde se iscrtava rezultat treniranja

U ovom konkretnom primeru sa slike 45 se vidi rezultat učenja mreže posle 500, 1000, 2500 i 5000 iteracija i primećuje se da mreža nije uspela da nauči traženo. Ovaj zaključak izvodimo iz toga što se i dalje u prostoru obojenim crvenom bojom pojavljuju plave tačkice i obratno. Za mrežu bi se reklo da je naučila zadati skup podataka, da se sve tačkice nalaze u onim delovima koji su obojeni kao i tačkice. Ovaj prikaz učenja kroz testiranje je veoma efikasan pokazatelj da li je zadatak koji smo zadali uspešno izvršen.

5. EVALUACIJA

U ovom poglavlju izvršeno je detaljno testiranje *backpropagation* algoritma u zavisnosti od arhitekture mreže koja se trenira i parametara algoritma. Cilj je utvrđivanje uticaja različitog broja neurona u srednjem sloju, početnih težina i koraka učenja na brzinu učenja. Test je izvršen na XOR problemu sa dva i sa tri ulaza.

Problem

Testiranje je izvršeno na neuronskoj mreži sa dva ili sa tri ulaza za XOR problem. U ranijim razmatranjima utvrđeno je da perceptron ne može da reši XOR problem pa on neće biti razmatran, već će testovi biti izvršeni isključivo za višeslojni perceptron. U tabelama ispod (tabela 7 i tabela 8) dati su trening setovi u zavisnosti od broja ulaza.

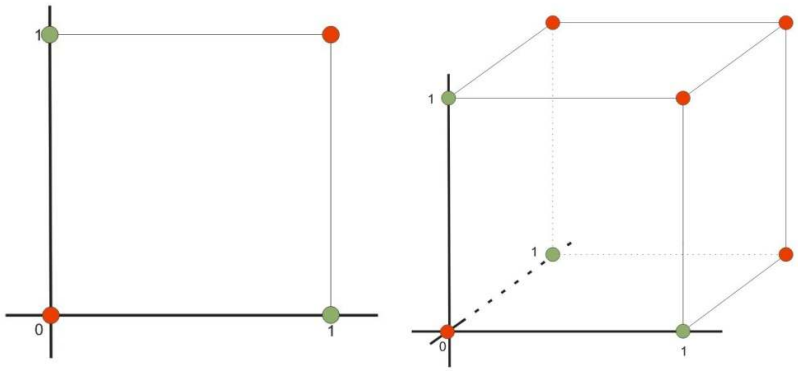
X1	X2	Y
0	0	**0**
1	0	**1**
0	1	**1**
1	1	**0**

Tabela 7. Trening set za dva ulaza

X1	X2	X3	Y
0	0	0	**0**
1	0	0	**1**
0	1	0	**1**
0	0	1	**1**
1	1	0	**0**
1	0	1	**0**
0	1	1	**0**
1	1	1	**0**

Tabela 8. Trening set za tri ulaza

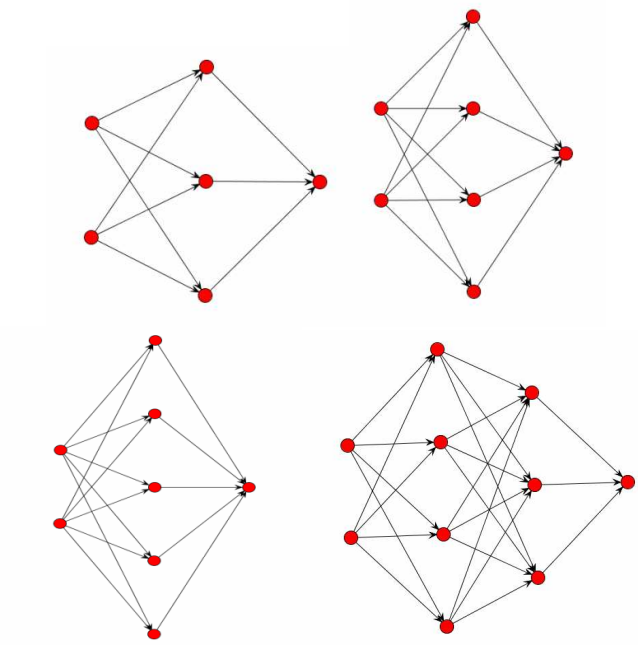
U tabeli 7. i tabeli 8. su sa X obleženi ulazi a sa Y je obležen izlaz. Grafički prikaz XOR problema je dat na slici 46. Crvenom bojom je obeležen izlaz nula a zelenom jedan.



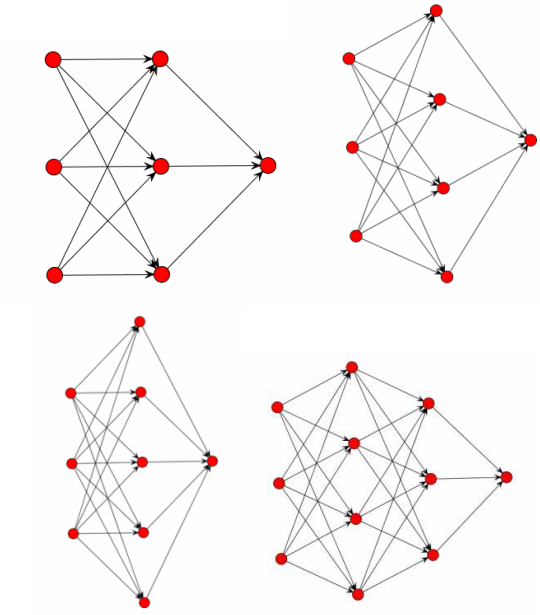
Slika 46. Graficki prikaz trening seta sa dva ulaza (levo) is a tri ulaza (desno)

Broj neurona u srednjem sloju

Jedan od značajnih problema kod projektovanja neuronskih mreža je određivanje broja neurona. U testovima koji slede je utvrđeno kako broj neurona u srednjem sloju može da utiče na brzinu učenja za XOR problem. Broj neurona u ulaznom sloju je konstantan i on će biti dva ili tri a broj neurona u izlaznom sloju je takođe konstantan i iznosi jedan. Na slici 47 su prikazane mreže sa tri, četiri i pet neurona u srednjem sloju i mreža sa dva sloja u sredini a sa dva ulazna neurona. Na slici 48 su prikazane slične mreže sa tri ulazna neurona.



Slika 47. Višeslojni perceptron sa dva ulazna – različitim brojem srednjih – jednim izlaznim neuronom



Slika 48. Višeslojni perceptron sa tri ulazna – različitim brojem srednjih – jednim izlaznim neuronom

Početne vrednosti težina

Postoji znatan uticaj početnih težina na brzinu konvergencije (broj iteracija) [2].Izbor početnih težina će uticati na mrežu sa prostiranjem signala unapred na to da li će ona dostići globalni ili lokalni minimum pri računanju greške i, ako postigne, brzinu konvergencije. Imajući ovo u vidu, testovi su obavljeni sa pet različitih setova početnih težina.
U tabeli ispod su date težine veza između ulaznog i srednjeg sloja za pet različitih testova:

	ω_{11}	ω_{12}	ω_{13}	ω_{21}	ω_{22}	ω_{23}	ω_{31}	ω_{32}	ω_{33}	ω_{41}	ω_{42}	ω_{43}	ω_{51}	ω_{52}	ω_{53}
ω^1	*-0.13*	*-0.38*	*-0.5*	*0.31*	*0.21*	*-0.25*	*-0.48*	*-0.35*	*0.09*	*0.19*	*-0.15*	*-0.41*	*-0.33*	*-0.04*	*0.11*
ω^2	*-0.11*	*-0.04*	*-0.23*	*-0.19*	*0.31*	*0.39*	*0.48*	*0.18*	*0.18*	*0.03*	*-0.4*	*0.25*	*-0.3*	*0.42*	*-0.06*
ω^3	*0.43*	*0.15*	*-0.22*	*-0.04*	*-0.24*	*0.11*	*-0.12*	*0.49*	*0.21*	*0.11*	*0.24*	*-0.31*	*0.46*	*-0.44*	*0.39*
ω^4	*0.13*	*-0.2*	*-0.34*	*0.26*	*0.11*	*-0.38*	*0.35*	*0.21*	*-0.09*	*0.37*	*0.1*	*-0.46*	*0.48*	*-0.23*	*0.48*
ω^5	*-0.24*	*0.39*	*-0.1*	*0.06*	*-0.4*	*0.18*	*0.17*	*0.01*	*-0.37*	*-0.39*	*0.32*	*0.37*	*-0.01*	*-0.43*	*0.07*

Tabela 9. Veze neurona između ulaznog i srednjeg sloja

U tabeli ispod su date težine veza između srednjeg i izlaznog sloja za pet različitih testova:

	z_{11}	z_{12}	z_{13}	z_{14}	z_{15}	z_{21}	z_{22}	z_{23}	z_{24}	z_{25}	z_{31}	z_{32}	z_{33}	z_{34}	z_{35}
z^1	*0.4*	*-0.5*	*0.23*	*0.32*	*-0.16*	*0.18*	*-0.32*	*0.35*	*-0.49*	*0.18*	*-0.48*	*-0.18*	*-0.49*	*0.15*	*0.41*
z^2	*-0.15*	*0.37*	*-0.11*	*-0.34*	*0.02*	*-0.01*	*-0.37*	*-0.11*	*0.2*	*-0.24*	*-0.26*	*0.31*	*0.03*	*-0.39*	*0.33*
z^3	*0.05*	*0.45*	*0.17*	*0.11*	*0.10*	*-0.32*	*-0.27*	*0.27*	*-0.47*	*-0.44*	*0.24*	*-0.47*	*-0.36*	*0.37*	*-0.32*
z^4	*0.47*	*0.19*	*0.23*	*-0.08*	*0.02*	*-0.5*	*0.11*	*-0.33*	*-0.38*	*-0.42*	*0.3*	*-0.1*	*-0.37*	*-0.27*	*0.37*
z^5	*0.06*	*0.09*	*-0.12*	*0.27*	*0.13*	*0.17*	*-0.24*	*0.41*	*-0.13*	*0.4*	*-0.5*	*-0.11*	*-0.24*	*-0.34*	*-0.49*

Tabela 10. Veze neurona između srednjeg i izlaznog sloja

U tabeli ispod date su težine veza između drugog srednjeg sloja, i izlaznog sloja:

	q_1	q_2	q_3
q^1	-0.26	-0.23	-0.49
q^2	0.28	0.45	-0.33
q^3	-0.41	0.4	0.33
q^4	0.00	-0.4	0.3
q^5	0.37	0.49	-0.11

Tabela 11. Veze neurona između drugog srednjeg sloja i izlaznog sloja

[2] Laurene Fausett (1992). „*Fundamentals of neural networks, architectures, algorithms and applications*“

Iz datih tabela formirani su skupovi početnih težina koji su korišćeni za testiranje u zavisnosti od broja neurona arhitekture:

Set 1 početnih težina neurona:
$WS1_{rmsv}$ - (ω_{ij}^{1}, z_{kl}^{1},q_{n}^{1}) za i = 1 .. m , j = 1 .. r , k = 1 .. s , l = 1 .. m , n = 0 .. v

Set 2 početnih težina neurona:
$WS2_{rmsv}$ - (ω_{ij}^{2}, z_{kl}^{2},q_{n}^{2}) za i = 1 .. m , j = 1 .. r , k = 1 .. s , l = 1 .. m , n = 0 .. v

Set 3 početnih težina neurona:
$WS3_{rmsv}$ - (ω_{ij}^{3}, z_{kl}^{3},q_{n}^{3}) za i = 1 .. m , j = 1 .. r , k = 1 .. s , l = 1 .. m , n = 0 .. v

Set 4 početnih težina neurona:
$WS4_{rmsv}$ - (ω_{ij}^{4}, z_{kl}^{4},q_{n}^{4}) za i = 1 .. m , j = 1 .. r , k = 1 .. s , l = 1 .. m , n = 0 .. v

Set 5 početnih težina neurona:
$WS5_{rmsv}$ - (ω_{ij}^{5}, z_{kl}^{5},q_{n}^{5}) za i = 1 .. m , j = 1 .. r , k = 1 .. s , l = 1 .. m , n = 0 .. v

gde su vrednosti u opsegu:

r = {2,3} - predstavlja broj ulaznih neurona
m = {3,4,5} - predstavlja broj neurona u prvom srednjem sloju
s = {1,2,3} - predstavlja broj neurona u drugom srednjem / izlaznom sloju
v = {0,3}- predstavlja broj izlaznih neurona (koristi se isključivo za mrezu sa dva skrivena sloja)

5.1. TESTIRANJE KORAKA UČENJA I POČETNIH VREDNOSTI TEŽINA

U ovom poglavlju su prikazani rezultati testova koji se odnose na broj iteracija učenja višeslojnog perceptrona u odnosu na veličinu koraka učenja. Korak učenja predstavlja vrednost koja kontroliše promenu težina. Testirane su vrednosti koraka učenja od 0.25 , 0.3, 0.35, 0.4, 0.45. Testovi su izvršeni nad mrežama sa različitim brojem neurona u srednjem i ulaznom sloju rešavajući XOR problem.

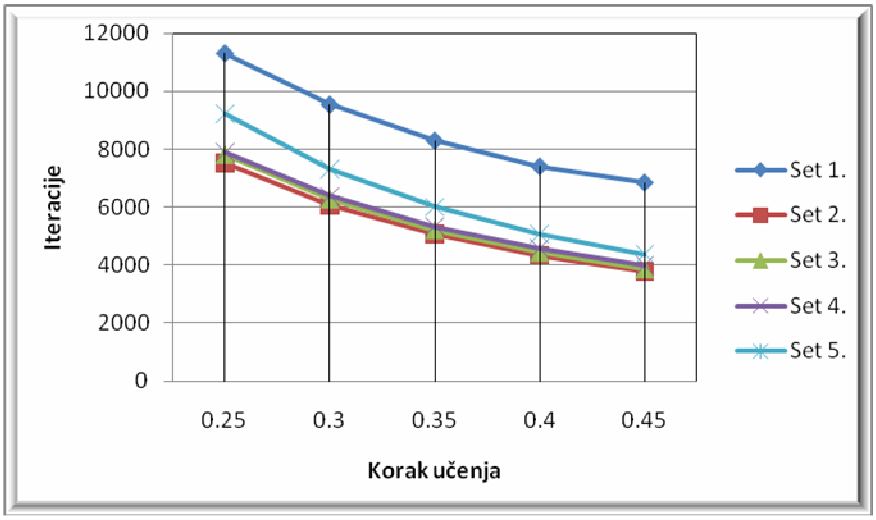
Test 1

U tabeli 12 je prikazan broj iteracija koji je potreban višeslojnom perceptronu sa dva ulaza – tri srednja neurona i jednim izlaznim (slika 47) da nauči dvodimenzioni XOR problem (tabela 7) u zavisnosti od početnih vrednosti težina između neurona (datih u tabelama 9,10 i 11).

Test	Početne težine	Korak učenja				
		0.25	0.3	0.35	0.4	0.45
1.	$WS1_{2310}$	11315	9544	8302	7411	6853
2.	$WS2_{2310}$	7525	6073	5069	4338	3783
3.	$WS3_{2310}$	7801	6260	5204	4438	3860
4.	$WS4_{2310}$	7894	6385	5337	4569	3984
5.	$WS5_{2310}$	9232	7307	6007	5077	4382

Tabela 12. Broj iteracija višeslojnog perceptrona sa dva ulaza i tri neurona u srednjem sloju

Na osnovu rezultata iz tabele 4 napravljen je grafik prikazan na slici 49.



Slika 49. Grafik napravljen na osnovu rezultata iz tabele 1.

Na osnovu podataka iz tabele 12. i grafika sa slike 49. može se videti da je sa setom težina 2 i korakom učenja 0.45 potrebno najmanje iteracija da višeslojni perceptron nauči XOR problem. U svim testovima sa različitim setovima početnih težina povećanjem koraka učenja

se smanjuje broj potrebnih iteracija. Ovaj test pokazuje da početne težine imaju znatan uticaj na brzinu učenja.

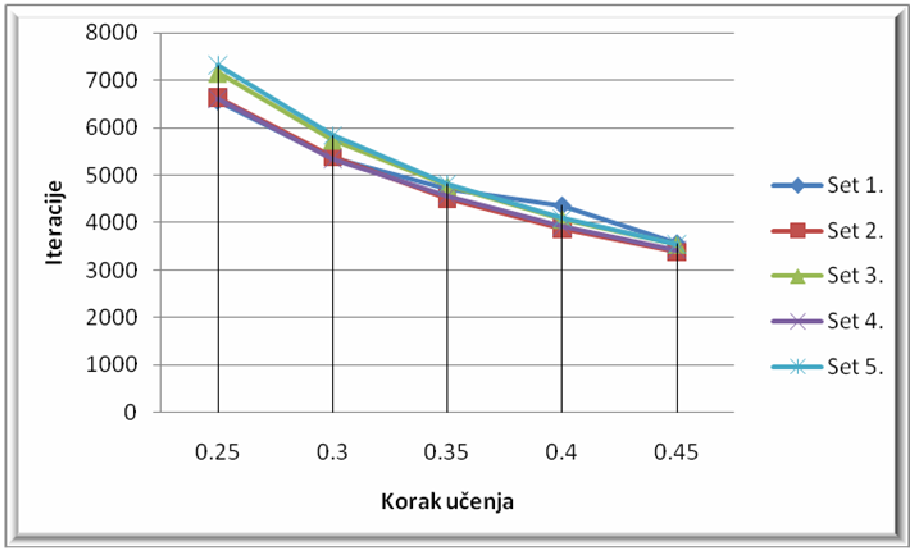
Test 2

U tabeli 13 je prikazan broj iteracija koji je potreban višeslojnom perceptronu sa dva ulazna neurona – četiri neurona u srednjem sloju i jednim izlaznim neuronom (slika 47) da nauči dvodimenzioni XOR problem (tabela 7) u zavisnosti od početnih vrednosti težina između neurona (datih u tabelama 9,10 i 11).

Test	Početne težine	Korak učenja				
		0.25	0.3	0.35	0.4	0.45
1.	$WS1_{2410}$	6551	5376	4696	4358	3562
2.	$WS2_{2410}$	6637	5384	4511	3869	3380
3.	$WS3_{2410}$	7157	5742	4772	4071	3541
4.	$WS4_{2410}$	6610	5338	4564	3913	3426
5.	$WS5_{2410}$	7315	5836	4820	4087	3537

Tabela 13. Broj iteracija višeslojnog perceptrona sa dva ulaza i četiri neurona u srednjem sloju

Na osnovu rezultata iz tabele 13 napravljen je grafik prikazan na slici 50.



Slika 50.Grafik napravljen na osnovu rezultata iz tabele 13.

Na osnovu podataka iz tabele 13 i grafika sa slike 50 se vidi da višeslojni perceptron sa četiri neurona u srednjem sloju najbrže uči sa početnim setom težina 2 i korakom učenja 0.45. Sa povećanjem koraka učenja se smanjuje broj iteracija u svih pet pokušaja sa različitim setovima početnih težina. Rezultati učenja za svih pet setova su veoma bliski, razlika broja iteracija između mreže sa najviše i mreže sa najmanje je manja od 10% za svaki korak učenja a u slučaju koraka učenja 0.45 iznosi 5% što ukazuje da početne težine u ovom slučaju nisu imale značajnu ulogu u učenju.

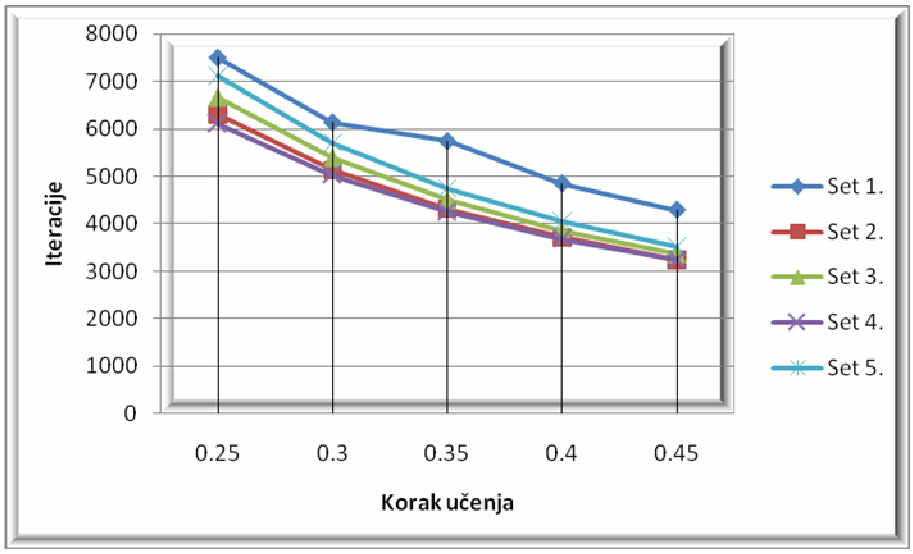
Test 3

U tabeli 14 je prikazan broj iteracija koji je potreban višeslojnom perceptronu sa dva ulazna neurona – pet neurona u srednjem sloju i jednim izlaznim neuronom (slika 47) da nauči dvodimenzioni XOR problem (tabela 7) u zavisnosti od početnih vrednosti težina između neurona (datih u tabelama 9,10 i 11).

Test	Početne težine	Korak učenja				
		0.25	0.3	0.35	0.4	0.45
1.	$WS1_{2510}$	7504	6133	5736	4844	4285
2.	$WS2_{2510}$	6300	5131	4312	3708	3244
3.	$WS3_{2510}$	6658	5380	4494	3846	3354
4.	$WS4_{2510}$	6100	5013	4243	3670	3227
5.	$WS5_{2510}$	7110	5705	4742	4042	3511

Tabela 14. Broj iteracija višeslojnog perceptrona sa dva ulaza i pet neurona u srednjem sloju

Na osnovu rezultata iz tabele 14 napravljen je grafik prikazan na slici 51.



Slika 51.Grafik napravljen na osnovu rezultata iz tabele 14.

Na osnovu podataka iz tabele 14 i grafika sa slike 51 se vidi da višeslojni perceptron sa pet neurona u srednjem sloju najbrže uči (ima najmanje iteracija) sa četvrtim setom početnih težina i korakom učenja 0.45. Sa povećanjem koraka učenja se smanjuje broj iteracija za svaki set početnnih težina. Rezultati testova za setove 2, 3, 4 i 5 početnih težina su veoma bliske dok je broj iteracija učenja za prvi set težina za oko 1000 iteracija veći. U ovom slučaju je primetan značaj početnog seta težina.

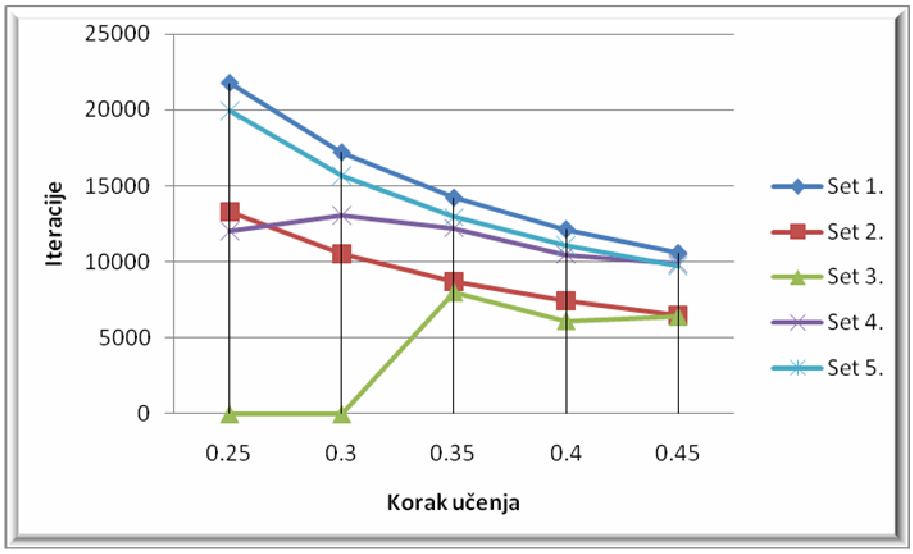
Test 4

U tabeli 15 je prikazan broj iteracija koji je potreban višeslojnom perceptronu sa dva ulazna neurona – četiri neurona u prvom skrivenom sloju – tri neurona u drugom skrivenom sloju i jednim izlaznim neuronom (slika 47) da nauči dvodimenzioni XOR problem (tabela 7) u zavisnosti od početnih vrednosti težina između neurona (datih u tabelama 9,10 i 11).

Test	Početne težine	Korak učenja				
		0.25	0.3	0.35	0.4	0.45
1.	$WS1_{2433}$	21756	17192	14214	12135	10583
2.	$WS2_{2433}$	13283	10531	8700	7409	6459
3.	$WS3_{2433}$	-	-	7982	6107	6382
4.	$WS4_{2433}$	12029	13037	12202	10467	9884
5.	$WS5_{2433}$	19921	15669	12946	11081	9744

Tabela 15. Broj iteracija višeslojnog perceptrona sa dva skrivena sloja

Na osnovu rezultata iz tabele 15 napravljen je grafik prikazan na slici 52.



Slika 52.Grafik napravljen na osnovu rezultata iz tabele 15.

Na osnovu podataka iz tabele 15 i grafika sa slike 52 se vidi da je *višeslojnog perceptrona* sa dva skrivena sloja najbrže uči (ima najmanje iteracija) sa trećim setom težina i korakom učenja 0.45. Takođe se može primetiti da sa korakom učenja 0.25 i 0.3 ta mreža nije uspela na nauči XOR problem što znači da je korak učenja isuviše mali te je dostignut lokalni minimum a ne globalni. Sa povećanjem koraka učenja mreže su brže učile za sve početne setove težina.

Zaključak testova 1 - 4

U testovima 1 - 4 su istestirane različite arhitekture *višeslojnog perceptrona* -a da reše isti XOR problem dat u tabeli 7. Rezultati testiranja su pokazali da bez obzira na broj neurona u skrivenom sloju važi:

- Korak učenja utiče na broj iteracija koje su potrebne višeslojnom perceptronu da nauči XOR problem
- Početne težine imaju uticaj na broj iteracija potrebnih višeslojnom perceptronu da nauči XOR problem.
- Ako je korak isuviše mali broj iteracija će biti veći
- Povećanje koraka učenja u svim testovima dovelo je do smanjenja broj aiteracija
- Ako je korak isuviše veliki mreža neće moći uspešno da završi učenje
- Ako je korak učenja isuviše mali učenje se može zaustaviti u lokalnom minimumu.

Arhitekture sa tri ulaza

Da bi proverili zaključke zasnovane na dvodimenzionom XOR problemu testiraćemo iste arhitekture *višeslojnog perceptrona* (sa istim brojem srednjih neurona) sa trodimenzionim XOR problemom datim u tabeli 8.

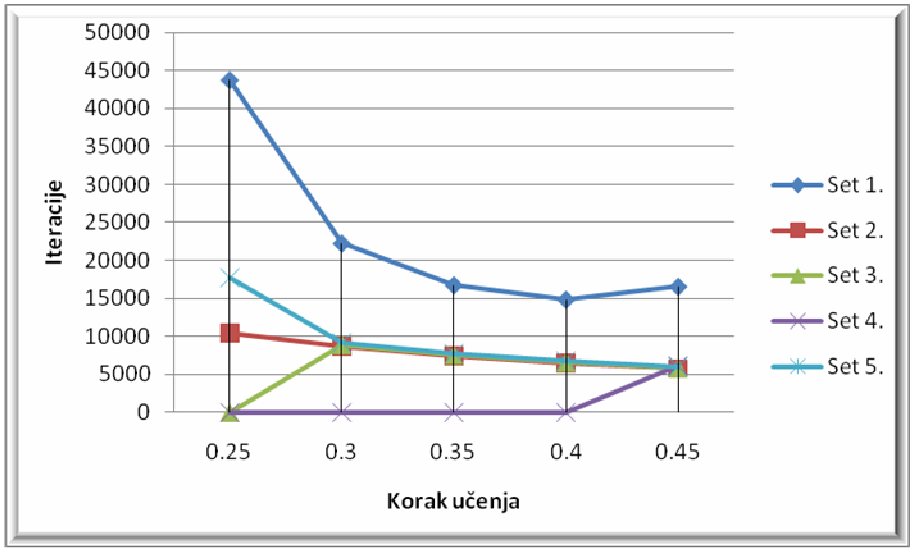
Test 5

U tabeli 16 je prikazan broj iteracija koji je potreban višeslojni perceptron sa tri ulazna neurona – tri neurona u srednjem sloju i jednim izlaznim neuronom (slika 48) da nauči trodimenzioni XOR problem (tabela 8) u zavisnosti od početnih vrednosti težina između neurona (datih u tabelama 9,10 i 11).

Test	Početne težine	Korak učenja				
		0.25	0.3	0.35	0.4	0.45
1.	$WS1_{3310}$	43753	22232	16737	14838	16544
2.	$WS2_{3310}$	10415	8660	7410	6475	5750
3.	$WS3_{3310}$	-	8901	7568	6584	5828
4.	$WS4_{3310}$	-	-	-	-	6157
5.	$WS5_{3310}$	17702	9148	7756	6739	5959

Tabela 16. Broj iteracija višeslojnog perceptrona sa tri ulaza i tri neurona u srednjem sloju

Na osnovu rezultata iz tabele 16 napravljen je grafik prikazan na slici 53.



Slika 53.Grafik napravljen na osnovu rezultata iz tabele 16.

Na osnovu podataka iz tabele 16 i grafika sa slike 53 može se videti da višeslojni perceptron sa tri neurona u srednjem sloju najbrže uči (najmanje iteracija) ima sa drugim setom početnih težina i sa korakom učenja 0.45. Višeslojni perceptron koji je učio sa četvrtim setom početnih težina je postigao konvergenciju tek sa korakom učenja od 0.45 jer se manji koraci zaustave kod lokalnog minimuma. Višeslojni perceptron koji je učio sa prvim setom težina je najmanji broj iteracija postigao sa korakom učenja 0.4 što je odstupanje od zaključaka dosadašnjih testova da sa korakom učenja se smanjuje broj iteracija.

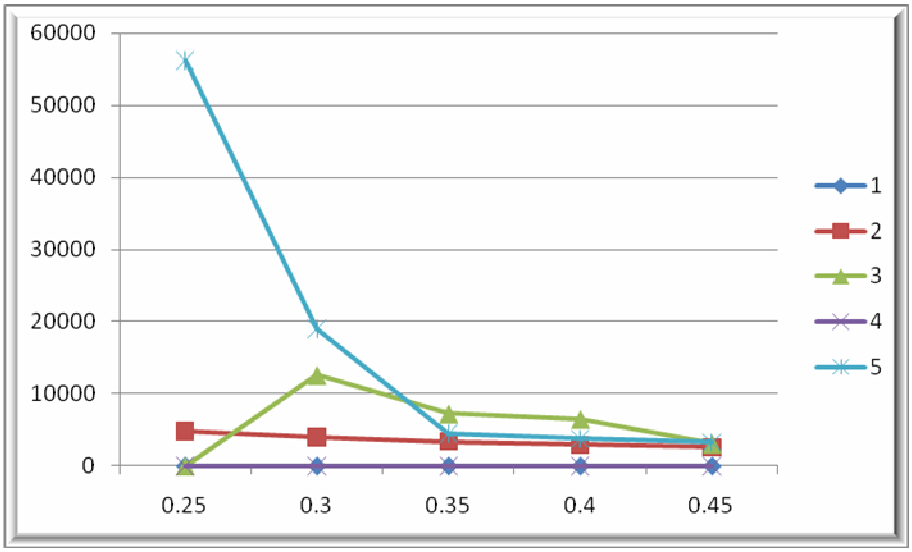
Test 6

U tabeli 17 je prikazan broj iteracija koji je potreban višeslojnom perceptronu sa tri ulazna neurona – pet neurona u srednjem sloju i jednim izlaznim neuronom (slika 48) da nauči trodimenzioni XOR problem (tabela 8) u zavisnosti od početnih vrednosti težina između neurona (datih u tabelama 9,10 i 11).

Test	Početne težine	Korak učenja				
		0.25	0.3	0.35	0.4	0.45
1.	$WS1_{3410}$	-	-	-	-	-
2.	$WS2_{3410}$	4812	3964	3368	2925	2584
3.	$WS3_{3410}$	-	12582	7243	6475	3078
4.	$WS4_{3410}$	-	-	-	-	-
5.	$WS5_{3410}$	56166	18968	4502	3767	3262

Tabela 17. Broj iteracija višeslojnog perceptrona sa dva ulaza i četiri neurona u srednjem sloju

Na osnovu rezultata iz tabele 17 napravljen je grafik prikazan na slici 54.



Slika 54.Grafik napravljen na osnovu rezultata iz tabele 9.

Na osnovu podataka iz tabele 17 i grafika sa slike 54 može se videti da najbrže uči višeslojni perceptron sa drugim setom početnih težina i korakom učenja 0.45. Vidi se da za prvi i četvrti set početnih težina višeslojni perceptron nije postigao konvergenciju ni za jedan od testiranih koraka učenja. Ovde se jasno vidi uticaj početnih težina u mogućoj konvergenciji. Kod drugog, trećeg i petog seta početnih težina višeslojni perceptron brže uči sa povećanjem koraka učenja.

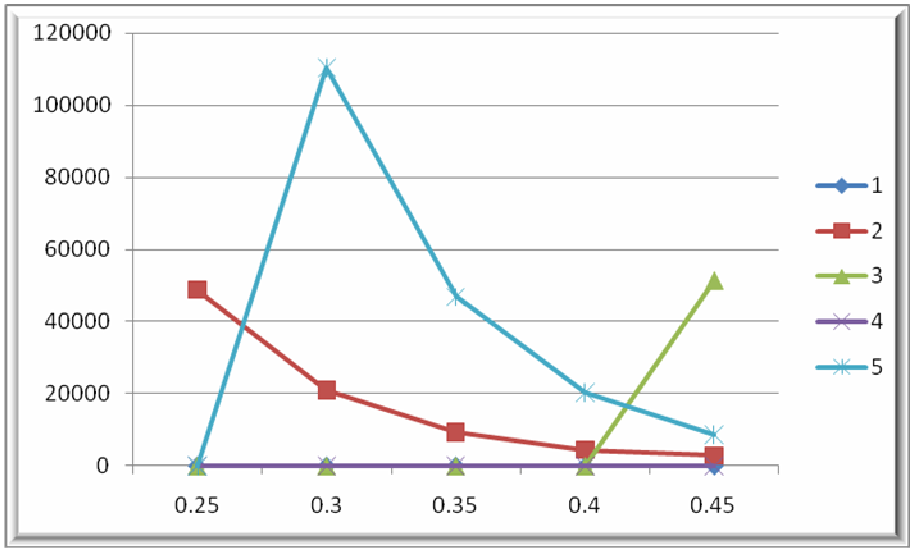
Test 7

U tabeli 18 je prikazan broj iteracija koji je potreban višeslojnom perceptronu sa tri ulazna neurona – pet neurona u srednjem sloju i jednim izlaznim neuronom (slika 48) da nauči trodimenzioni XOR problem (tabela 8) u zavisnosti od početnih vrednosti težina između neurona (datih u tabelama 9,10 i 11).

Test	Početne težine	Korak učenja				
		0.25	0.3	0.35	0.4	0.45
1.	$WS1_{3510}$	-	-	-	-	-
2.	$WS2_{3510}$	48786	21064	9230	4450	2801
3.	$WS3_{3510}$	-	-	-	-	51267
4.	$WS4_{3510}$	-	-	-	-	-
5.	$WS5_{3510}$	-	110302	46854	20215	8542

Tabela 18. Broj iteracija mreže sa dva ulaza i četiri neurona u srednjem sloju

Na osnovu rezultata iz tabele 18 napravljen je grafik prikazan na slici 55.



Slika 55.Grafik napravljen na osnovu rezultata iz tabele 18.

Na osnovu podataka iz tabele 18 i grafika sa slike 55 može se videti da višeslojni perceptron koji najbrže uči (ima najmanje iteracija) ima korak učenja 0.45 sa drugim setom početnih težina. Višeslojni perceptron sa prvim i četvrtim početnim setom težina nije postigao konvergenciju a sa trećim setom početnih težina je postigao konvergenciju tek sa korakom učenja 0.45. To znači da je potreban korak učenja veći od 0.45 jer mreža staje sa učenjem kada naiđe na lokalni minimum. Iz grafika se vidi da sa povećanjem koraka učenja se smanjuje broj iteracija potrebnih višeslojnom perceptronu da nauči.

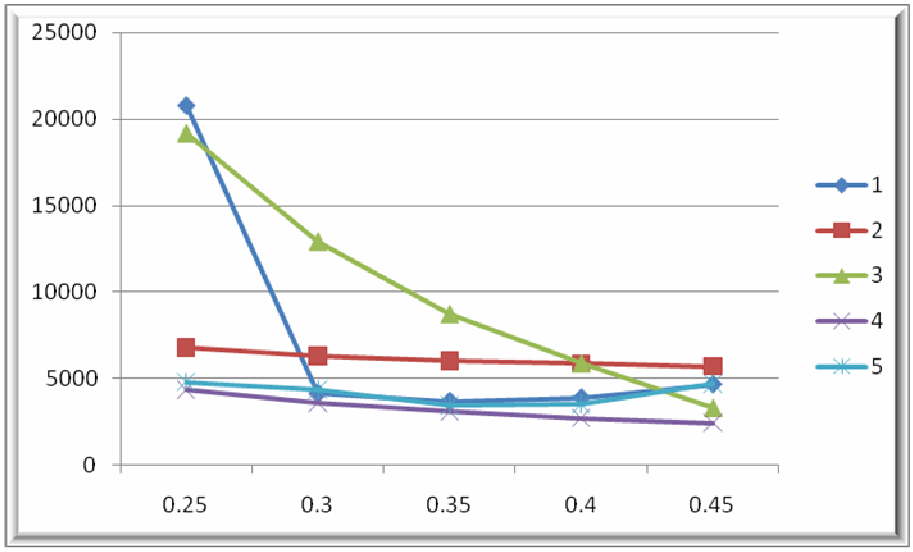
Test 8

U tabeli 19 je prikazan broj iteracija koji je potreban višeslojnom perceptronu sa tri ulazna neurona – četiri neurona u prvom skrivenom sloju – tri neurona u drugom skrivenom sloju i jednim izlaznim neuronom (slika 48) da nauči trodimenzioni XOR problem (tabela 8) u zavisnosti od početnih vrednosti težina između neurona (datih u tabelama 9,10 i 11).

Test	Početne težine	Korak učenja				
		0.25	0.3	0.35	0.4	0.45
1.	$WS1_{3433}$	20766	4155	3701	3890	4659
2.	$WS2_{3433}$	6802	6301	6043	5877	5722
3.	$WS3_{3433}$	19160	12912	8733	5881	3307
4.	$WS4_{3433}$	4355	3619	3108	2735	2453
5.	$WS5_{3433}$	4820	4378	3467	3531	4720

Tabela 19. Broj iteracija mreže sa dva skrivena sloja

Na osnovu rezultata iz tabele 19 napravljen je grafik prikazan na slici 56.



Slika 56.Grafik napravljen na osnovu rezultata iz tabele 19.

Na osnovu podataka iz tabele 19 i grafika sa slike 56 može se videti da višeslojni perceptron koji nabrže uči (najmanji broj iteracija) ima korak učenja 0.45 sa četvrtim setom početnih težina. Broj iteracija učenja višeslojnog perceptrona ne opada nužno sa povećanjem koraka učenja jer kod prvog i četvrtog seta početnih težina broj iteracija opada porastom koraka učenja sa 0.25 na 0.3 pa dalje do 0.35 a onda raste daljim porastom koraka učenja na 0.4 i 0.45, što znači da je globalni minimum lakše pronaći da korakom učenja koji nije ni previše mali a ni previše veliki a konkretna vredost se može dobiti testiranjem.

Zaključna razmatranja

Zaključci testova 5 – 8:

- Na osnovu trodimenzionog XOR problema se mogu potvrditi zaključci dati na osnovu testova 1 – 4 koji se odnose na korak učenja i početne težine.
- Višeslojni perceptron koja ima arhitekturu sa dva skrivena sloja sporije uči u slučaju dva ulaza u mrežu u odnosu na mrežu sa tri ulaza tj. broj skrivenih slojeva ne poboljšava performanse učenja.

U tabelama 20 i 21 date su prosečne vrednosti itereracija potrebnih da višeslojni perceptron nauči XOR problem (dvodimenzioni i trodimenzioni).

Arhitektura	Korak učenja				
	0.25	0.3	0.35	0.4	0.45
MLP 2-3-1	8753,4	7113,8	5983,8	5166,6	4572,4
MLP 2-4-1	6854	5535,2	4672,6	4059,6	3489,2
MLP 2-5-1	6734,4	5472,4	4705,4	4022	3524,2
MLP 2-4-3-1	16747,25	14107,25	11208,8	9439,8	8610,4

Tabela 20. Prosečan broj iteracija višeslojnog perceptron sa dva ulaza

Arhitektura	Korak učenja				
	0.25	0.3	0.35	0.4	0.45
MLP 3-3-1	23956,67	12235,25	9867,75	8659	8047,6
MLP 3-4-1	30489	11838	5037,667	4389	2974,667
MLP 3-5-1	48786	65683	28042	12332,5	20870
MLP 3-4-3-1	11180,6	6273	5010,4	4382,8	4172,2

Tabela 21. Prosečan broj iteracija višeslojnog perceptrona sa tri ulaza

Iz datih tabela 20 i 21 se može zaključiti da dvodimenzioni XOR problem najbrže naći višeslojni perceptron sa 4 neurona u skrivenom sloju i korakom učenja od 0.45 dok kod trodimenzionog XOR problema najbrže uči višeslojni perceptron sa dva skrivena sloja i korakom učenja od 0.45.

6. ZAKLJUČNA RAZMATRANJA

Ciljevi ovog rada su uspešno realizovani i dat je teorijski i praktični prikaz učenja kod neuronskih mreža sa prostiranjem signala unapred. *Perceptron* i *Backpropagation* primeri vizuelno predstavljaju učenje kod neuronskih mreža sa prostiranjem signala unapred. Klase su projektovane na dvo-nivojskoj arhitekturi, pri čemu je korišćen *observer* patern za razdvajanje logike od korisničkog interfejsa.

Kao najbitnije stavke realizovnog softvera izdvajaju se:

- vizuelni prikaz učenja mreže tipa perceptron;
- vizuelni prikaz učenja mreže tipa višeslojni perceptron;
- vizuelni korisnički interfejs za unošenje trening seta;
- vizuelni prikaz podeljenog prostora tokom rada neuronske mreže.

Perceptron primer i *Backpropagation* primer imaju:

- Značaj u obrazovanju ljudi koji žele da nauče kako rade veštačke neuronske mreže sa prostiranje signala unapred

- Praktičan značaj u vizuelnom predstavljanju rešenja problema

Dalji razvoj softvera bi obuhvatio još bolje predstavljanje načina na koji neuronska mreža sa prostiranjem signala unapred deli ulazni prostor. Sa teorijskog aspekta bilo bi interesantno testirati uticaj momentum parametra na učenje.

LITERATURA

[1] Jacobs, R. A, *„Increased Rates of Convergence Thought Learning Rate Adaption",* Neural Networks, 1988.

[2] Laurene Fausett, „*Fundamentals of neural networks, architectures, algorithms and applications*", 1992.

[3] Zoran Ševarac, „*Aplikacioni okvir za razvoj neuronskih mreža",Fakultet organizacionih nauka u Beogradu,* 2004.

[4] Wikipedia, „*Feedforward neural network*", http://en.wikipedia.org/wiki/Feedforward_neural_network

[5] Laboratorie de Microinformatique, „*Neural Java*", http://lcn.epfl.ch/tutorial/english/index.html

[6] Neural Network Forecasting, „*MLP Neural Nets*", http://www.neural-forecasting.com/mlp_neural_nets.htm