Algoritam rojeva čestica u učenju neuronske mreže

Čović, Ivan

SVEUČILIŠTE U ZAGREBU

FAKULTET STROJARSTVA I BRODOGRADNJE

Završni rad

Ivan Čović

Zagreb, 2019.

SVEUČILIŠTE U ZAGREBU

FAKULTET STROJARSTVA I BRODOGRADNJE

Završni rad

Mentor:
Prof. dr. sc. Dubravko Majetić

Student:
Ivan Čović

Zagreb, 2019.

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	Prilog
Klasa:	
Ur.broj:	

ZAVRŠNI ZADATAK

Student: **Ivan Čović** Mat. br.: 0035184336

Naslov rada na hrvatskom jeziku: **Algoritam rojeva čestica u učenju neuronske mreže**

Naslov rada na engleskom jeziku: **Particle Swarm Algorithm in neural network learning**

Opis zadatka:

Optimizacijska metoda rojeva čestica (engl. Particle Swarm Algorithm, PSO) jedan je od algoritama koji se može koristiti za treniranje unaprijedne umjetne neuronske mreže i može poslužiti kao zamjena za algoritam učenja s povratnim prostiranjem pogreške.

U radu treba načiniti sljedeće:

1. Opisati algoritam učenja neuronske mreže s povratnim prostiranjem pogreške.
2. Opisati algoritam metode rojeva čestica.
3. Na zadanom primjeru obaviti učenje neuronske mreže algoritmom povratnog prostiranja pogreške (EBP) i PSO algoritma.
4. Usporediti generalizacijska svojstva neuronske mreže učene s EBP i PSO algoritmom.
5. Izvesti zaključke rada.

Zadatak zadan:

29. studenog 2018.

Rok predaje rada:
1. rok: 22. veljače 2019.
2. rok (izvanredni): 28. lipnja 2019.
3. rok: 20. rujna 2019.

Predviđeni datumi obrane:
1. rok: 25.2. - 1.3. 2019.
2. rok (izvanredni): 2.7. 2019.
3. rok: 23.9. - 27.9. 2019.

Zadatak zadao:

Prof.dr.sc. Dubravko Majetić

Predsjednik Povjerenstva:

Prof. dr. sc. Branko Bauer

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

Zahvaljujem se prof. dr. sc. Dubravku Majetiću na kvalitetnim,stručnim savjetima i na ukazanom povjerenju prilikom izrade ovog rada.

Zahvaljujem se svojoj obitelji i djevojci na bezuvjetnoj podršci i pomoći

Ivan Čović

Sadržaj

Sadržaj……..I

Popis slika……..III

Popis tablica……...IV

Popis oznaka……..IV

Sažetak……...VI

Summary……..VII

Sadržaj……..I
.....I
1. UVOD 1
1.1. Umjetne neuronske mreže 1
1.2. Biološki neuron 2
1.3. Umjetni neuron 3
1.4. Vrste i učenje umjetnih neuronskih mreža 4
2. NEURONSKA MREŽA S POVRATNM PROSTIRANJEM POGREŠKE 5
2.1. Perceptron 5
2.1.1. Učenje perceptrona 5
2.1.2. Delta pravilo 6
2.2. Višeslojne neuronske mreže 7
2.2.1. Model statičkog neurona 7
2.2.2. Model statičke neuronske mreže 9
2.3. Učenje povratnim rasprostiranjem pogreške 10
2.3.1. Unaprijedna faza učenja za statičke mreže 11
2.3.2. Povratna faza učenja 12
2.4. Promjena parametara učenja 14
2.4.1. Promjena težina izlaznog sloja 14
2.4.2. Promjena težina skrivenog sloja 15
2.5. Ocjena točnosti algoritma učenja 17
3. NEURONSKA MREŽA OPTIMIZIRANJEM ROJA ČESTICA 18
3.1. Optimizacijski algoritmi 18
3.2. Općenito o optimiziranju algoritmom roja čestica 18
3.3. Opis algoritma 19
3.4. Podešavanje i odabir parametara 20
4. UČENJE NEURONSKE MREŽE ALGORITMOM POVRATNIM PROSTIRANJEM POGREŠKE I PSO ALGORITMOM 23

4.1. XOR klasifikacijski problem 23
4.2. *Iris* set podataka 23
4.3. Izrada mreže i unaprijedna faza učenja 24
4.3.1. XOR problem 24
4.3.2. Iris set problem 25
4.4. Povratna faza 25
4.4.1. Povratna faza učenja algoritmom povratnog prostiranja pogreške 25
4.4.2. Povratna faza učenja PSO algoritmom 26
4.5. Rezultati 27
4.5.1. Prikaz rezultata za XOR problem 27
4.5.2. Prikaz rezultata za Iris set problem 30
4.6. Analiza rezultata 32
5. ZAKLJUČAK 33
6. LITERATURA 34

Popis slika

Slika 1.1. Struktura biološkog neurona 2
Slika 1.2. Struktura umjetnog neurona..... 3
Slika 2.1. Standardan model statičkog neurona 7
Slika 2.2. Bipolarna sigmoidalna funkcija..... 8
Slika 2.3. Sinusna funkcija..... 8
Slika 2.4. Model statičke unaprijedne neuronske mreže s jednim sakrivenim slojem..... 10
Slika 3.1. Prikaz određivanja novog položaja čestice u koraku algoritma 21
Slika 3.2. Topologije: a) potpuno vezana b) prstenasta c)von Neumannova..... 22
Slika 4.1. Uzorci za učenje problema Iris 24
Slika 4.2. Grafički prikaza učenja povratnim prostiranjem pogreške za slučaj 1..... 27
Slika 4.3. Grafički prikaz učenja PSO algoritmom za slučaj 1..... 27
Slika 4.4. Grafički prikaz učenja povratnim prostiranjem pogreške za slučaj 2..... 28
Slika 4.5. Grafički prikaz učenja PSO algoritmom za slučaj 2..... 28
Slika 4.6. Grafički prikaz učenja PSO algoritmom za slučaj 2..... 29
Slika 4.7. Grafički prikaz učenja PSO algoritmom za slučaj 3..... 29
Slika 4.8. Grafički prikaz učenja povratnim prostiranjempogreške za Iris set problema..... 30
Slika 4.9. Grafički prikaz učenja PSO algoritmom za Iris set problema..... 31
Slika 4.10. Usporedba odziva greške oba algoritma kod učenja mreže..... 32

Popis tablica

Tablica 1. Logičke vrijednosti XOR problema 23
Tablica 2.. zadani parametri tri slučaja kod XOR problema 25
Tablica 3. parametri povratnog prostiranja pogreške u zadanim problemima 25
Tablica 4. parametri PSO algoritma 26
Tablica 5. prikaz rezultata rješavanja XOR problema 30
Tablica 6. prikaz rezultata rješavanja Iris set problema 32

Popis oznaka

A_j – nazivnik prijenosne funkcije dinamičkog člana dinamičkog neurona diskretne forme

c – težinski koeficijent PSO algortima

d_k – željena vrijednost k-tog izlaza neuronske mreže

d_n – željena vrijednost n-tog koraka učenja neuronske mreže

E – funkcija cilja (suma kvadrata pogreške)

I – broj ulaznih neurona, uključujući i *bias*

J – broj neurona u sakrivenom sloju, uključujući i *bias*

K – broj neurona izlaznog sloja (broj izlaza)

L_B – donja dozvoljena granica potrage kod PSO algoritma

l_i – vrijednost globalnog najboljeg položaja kod PSO algoritma

N – broj elemenata u skupu za učenje

n - trenutna promjena parametara učenja, trenutni korak učenja

$NRMS$ – normalizirani korjen srednje kvadratne vrijednosti – mjera točnosti

net – vrijednost funkcije sume

net_{Hj} – vrijednost funkcije sume j-tog neurona sakrivenog sloja

net_{Ok} – vrijednost funkcije sume k-tog neurona izlaznog sloja

O_k – k-ti izlaz neuronske mreže

O_n - dobivena vrijednost izlaza n-tog koraka učenja

p_i – vrijednost osobnog najboljeg položaja kod PSO algoritma

r_l – slučajna vrijednost iz uniformne distribucije [0,1]

r_p – slučajna vrijednost iz uniformne distribucije [0,1]

T_j - željena izlazna vrijednost neurona j

U_B - gornja dozvoljena granica potrage kod PSO algoritma

u_j – vrijednost j-tog ulaza

v_{ji} – težinski koeficijent između j-tog neurona sakrivenog sloja i i-tog neurona ulaznog sloja

v_i - i-ta komponenta vektora brzine

W - matrica težina

w_j - vrijednost j-te težine

w_{ij} – težinski koeficijent veze između i-tog neurona izlaznog sloja i j-tog neurona sakrivenog sloja

$w_{ij}(n)$ – težinski koeficijent u n-tog koraka

$w_{ij}(n+1)$ – težinski koeficijent u slijedećem koraku

x – vektor ulaznih vrijednosti uzoraka skupa za učenje

x_i - i-ta komponenta vektora položaja čestice

y – izlazna vrijednost neurona

y_j – vrijednost izlaza j-tog neurona sakrivenog sloja

z_j – vektor pogreške j-tog neurona sakrivenog sloja

ϑ – parametar učenja (težinski koeficijent, dinamički koeficijent)

$\vartheta(n)$ – trenutna vrijednost parametra učenja

$\vartheta(n+1)$ – nova vrijednost parametra učenja

$\Delta\vartheta(n)$ – tekuća promjena parametra učenja

$\propto$ - vrijednost koeficijenta momentuma prvog reda

Δ_{ij} – iznos promjene težina u jednom koraku

δ_j – parametar algoritma povratnog prostiranja pogreške

β - vrijednost koeficijenta momentuma drugog reda

γ – aktivacijska funkcija neurona

η – koeficijent brzine učenja

σ_{d_n} – standardna devijacija željenih vrijednosti izlaza mreže

∇E – gradijent pogreške

Sažetak

U ovom radu opisani su algoritmi učenja neuronske mreže povratnim prostiranjem pogreške i algoritam optimizacije rojem čestica (PSO). Obrađena su dva klasična primjera (XOR i Iris set) u učenju neuronskih mreža, pomoću dvije navedene metode učenja neuronskih mreža. Uspoređuju se svojstva algoritama nakon izvođenja primjera kao što su: konvergencija greške, veličina greške, te brzina izvođenja algoritma u zadanim parametrima. Uz to, u uvodu se opisuje učenje i glavne značajke neuronskih mreža, a zatim način učenja i rada zadane mreže. Na kraju je izveden zaključak, na temelju obrađenih primjera.

Ključne riječi:

- neuronske mreže
- umjetna inteligencija
- umjetni neuron
- učenje neuronske mreže
- težinski faktori
- povratno prostiranje pogreške
- optimizacija rojem čestica

Summary

This paper describes the neural network training algorithms by backpropagation error and the particle swarm optimization (PSO). Two classical examples (XOR and Iris set) in neural network training were analyzed using the two methods. The properties of algorithms after executing examples are compared, such as: error convergence, magnitude of error, and the velocity of algorithm execution in given parameters. In addition, the introduction describes the training and main features of neural networks, followed by how the default network is trained and operated. Finally, a conclusion is drawn, based on the examples discussed.

Keywords:

- neural networks

- artificial intelligence

- artificial neuron

- training neural network

- neural network weights

- error backpropagation

- particle swarm optimization

1. UVOD

Području umjetne inteligencije cilj je ostvarivanje imitacije ljudskog mozga. U ostvarenju tog cilja pomažu umjetne neuronske mreže koje čine elemente umjetne inteligencije. Osnovna je zadaća umjetnih neuronskih mreža modeliranje biofiziologije mozga. One imaju sposobnost pohrane podataka, obrade informacija i odlučivanja na temelju logičkih pravila, što ih čini sličnima ljudskome mozgu. Također, budući da je ljudski mozak sposoban učiti, učenje je još jedan važan aspekt umjetne inteligencije koji omogućava sustavu da obavlja promjene nad samim sobom. Na temelju takvih usporedbi sustava s radom ljudskog mozga, sustav se naziva inteligentnim.

1.1. Umjetne neuronske mreže

Umjetna je neuronska mreža složen sustav sastavljen od mnoštva elementarnih jedinica, neurona. Svi su neuroni u međusobnoj interakciji i s okolinom grade funkcionalnu cjelinu.

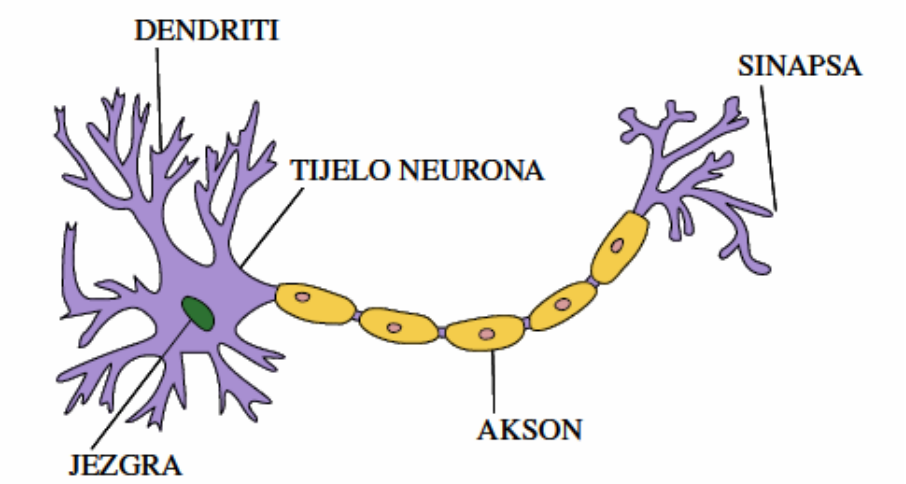
Tvrdnja Williama Jamesa [1] da: aktivnost bilo koje točke ljudskog mozga predstavlja zbroj tendencija svih ostalih točaka da se *"prazne"* u nju. Te tendencije proporcionalne su: a) broju točaka koje djeluju na promatranu točku, b) intezitetu tih uzbuda i c) odsutnosti rivalne točke koja nije u funkcionalnoj vezi s promatranom točkom, a u koju bi se "pražnjenje" ostalih točaka moglo skrenuti, postavlja temelje za izgradnju osnovne strukture umjetnog neurona.

Neuronom se zamjeni točka mozga, zatim se može aktivnost tog neurona modelirati kao zbroj otežanih ulaza neurona. Otežani su ulazi pomnoženi s određenim faktorima koji se nazivaju težinama neurona. Prema tome, aktivnost umjetnog neurona ovisi o: broju ulaza iz okoline neurona, intenzitetu tih veza te o pragu osjetljivosti koji stanje neurona treba dosegnuti prije nego "ispali impuls" preko svog izlaza u okolinu [2].

Formalizaciju aktivnosti umjetnog neurona dali su McCulloh i Pitts 1943. godine [3]. Njihov model neurona i danas služi kao osnovni blok za izgradnju umjetnih neuronskih mreža.

1.2. Biološki neuron

Biološki se neuron sastoji od tijela, aksona i mnoštva dendrita (slika 1.1). Akson se zamišlja kao rep koji povezuje tijelo neurona, najčešće s dendritima drugih neurona. Na krajevima aksona pojavljuju se sinaptički završeci, te se preko sinapsi impulsi šalju na dendrite drugih neurona. Na temelju ovog koncepta odvija se cjelokupni čovjekov misaoni proces.



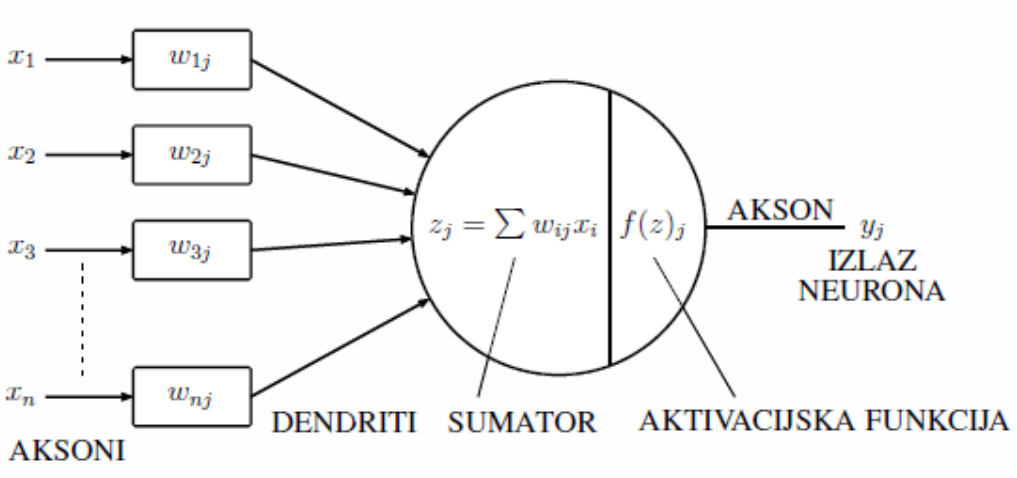
Slika 1.1. Struktura biološkog neurona [4]

Signali koje dendriti prenose mogu biti uzbudni ili smirujući. Matematički se oni opisuju suprotnim predznakom. Neuron će poslati impuls kroz svoj akson ukoliko je njegova uzbuda veća od smirujuće komponente signala za kritičan iznos. Kritičan se iznos naziva prag osjetljivosti neurona.

Neke procjene [5] govore da ljudski mozak posjeduje preko sto milijardi neurona koji su povezani s više od sto trilijuna sinapsi. Zahvaljujući neuronskoj mreži čovjek razmišlja, uči, pokazuje emocije, itd. Ipak, treba naglasiti da još nije dovoljno razumljiv egzaktan način rada neuronske mreže.

1.3. Umjetni neuron

Da bi se opisale osnovne funkcije rada biološkog neurona, dizajnira se umjetni neuron (slika 1.2). Tijelo takvog neurona opisuje se sumatorom: dendriti su ulazi u sumator, a izlaz iz sumatora je akson. U tijelu neurona, osim sumatora, potrebno je postaviti aktivacijsku funkciju čija je uloga opisati prag osjetljivosti biološkog neurona.



Slika 1.2. Struktura umjetnog neurona

Sinaptičke se veze opisuju pomoću težinskih faktora. Oni povezuju izlaze iz drugih neurona s ulazima sumatora (dendriti). Zatim se izlaz sumatora spaja na ulaz aktivacijske funkcije, a na izlazu aktivacijske funkcije producira se izlaz neurona.

Težinski faktori mogu biti pozitivni ili negativni brojevi, čak i neka funkcija, dok aktivacijska funkcija može biti linearna i nelinearna. Kod linearnih, izlaz se sumatora pomnoži s određenim faktorom i tako se dobije izlaz neurona. Nelinearne funkcije mogu biti različitih oblika, a najčešće su: sigmoidalne, hiperbolične, harmoničke i funkcije praga osjetljivosti.

1.4. Vrste i učenje umjetnih neuronskih mreža

Jedna je od kategorizacija umjetnih neuronskih mreža ona po broju slojeva. Umjetne neuronske mreže mogu biti jednoslojne i višeslojne. Kod višeslojnih postoji ulazni i izlazni sloj te barem jedan skriveni sloj.

Sljedeća je kategorizacija kategorizacija prema načinu putovanja signala. Ukoliko su slojevi neuronskih mreža povezani tako da signali putuju isključivo u jednom smjeru (od ulaza do izlaza mreže), tada se govori o unaprijednim neuronskim mrežama (*feedforward neural networks*). Također, u mreži postoji mogućnost postavljanja povratne petlje koje se nazivaju povratne neuronske mreže (*feedback neural networks*). Kategorije problema koje ovakve mreže rješavaju međusobno se isključuju.

Postoji i kategorizacija umjetnih neuronskih mreža prema metodama učenja tih mreža, pa postoje povratno propagirane, suprotno propagirane i statističke neuronske mreže.

Prema principu učenja, razlikuju se supervizorne i nesupervizorne neuronske mreže. Supervizorno se učenje provodi pomoću vanjskog "učitelja", koji promatra i korigira mrežu dok se ne usvoji željeno promatranje mreže. Prvo se odredi struktura mreže: broj ulaza, slojeva, izlaza te težina mreže. Određuju se početne težine neuronske mreže. Na ulaz se dovode ulazne varijable, a zatim mreža generira izlazne varijable. Promatranjem stvarnih izlaza, može se odrediti odstupanje od željenih vrijednosti te se kreira grešku. Potom se pomoću greške računaju nove težine. Postupak se ponavlja iteracijski dok se pogreška ne svede na željeni raspon vrijednosti.

Poslije učenja mreže, mreža se testira. To se čini tako da se uz dobivene, prihvatljive težine zadaju ulazne varijable koje u procesu učenja nisu korištene. Nakon toga, mreža generira izlaze koji se kasnije uspoređuju sa željenim izlazima. Iznos pogreške služi za ocjenu robusnosti.

Nesupervizorno učenje ne koristi vanjskog učitelja. Mreža se sama organizira. Na ulaz se dovode ulazne varijable, a mreža se samoorganizira podešavanjem težina. Željeni izlaz nije specificiran za vrijeme učenja mreže, a rezultat učenja nije predvidiv.

2. NEURONSKA MREŽA S POVRATNM PROSTIRANJEM POGREŠKE

2.1. Perceptron

Umjetni neuron, građen prema **slici 1.2**, te opisan prema poglavlju 1.3, za aktivacijsku funkciju koristi binarnu funkciju praga osjetljivosti, definiranu sljedećim izrazom:

$$y_j = f(z_j) = \begin{cases} 1, & z_j > prag \\ 0, & z_j \leq prag \end{cases} \tag{2.1}$$

Iz izraza se zaključuje da se radi o binarnom neuronu, čija vrijednost može biti 0 ili 1. Vrijednost 1 znači da je neuron aktivan, dok vrijednost 0 znači da je neuron neaktivan. Moguće je povezati više perceptrona u mrežu s ulaznim i izlaznim slojem. Takva mreža naziva se jednoslojna perceptronska mreža (*single-layer perceptron*). Svaki neuron ulaznog sloja predstavlja po jednu značajku, dok neuron izlaznog sloja označava jednu skupinu klasifikacijskog problema. Ulazni sloj prenosi signale izlaznom sloju u kojem se signali množe s težinama w_{ij} i sumiraju. Težine su podesive, a cilj je da se nepoznate težine podese tako da izlazni neuron, koji pripada istoj skupini kao promatrani uzorak skupa za učenje, ima vrijednost 1, a da svi ostali neuroni imaju vrijednost 0.

2.1.1. Učenje perceptrona

Perceptronske se mreže uče iterativnim postupkom prema sljedećem algoritmu:

1. Ulaznim vektorom *x* uzorka iz skupa za učenje izračunava se izlazni vektor *y*:

$$y = x \cdot \mathrm{W}\,. \tag{2.2}$$

W je matrica težina, čiji elementi u prvom koraku poprimaju vrijednost slučajnih brojeva u skupu od 0 do 1.

2. a) Ako je izlaz točan, vraćamo se na točku 1.

b) Ako je pogrešan i ako je 0, dodaju se vrijednosti ulaznog vektora pripadajućim težinama.

c) Ako je pogrešan i ako je 1, od pripadajućih težina oduzmu se vrijednosti ulaznog vektora.

3. Povrat na točku 1.

Postupak učenja iterativno se provodi za sve uzorke skupa za učenje dok svi nisu pravilno razvrstani u pripadajuće skupine. Postupak uspješno konvergira u konačnom broju koraka ukoliko su uzorci linearno separabilni [5].

2.1.2. Delta pravilo

Kod neurona s kontinuiranim ulaznim i izlaznim vrijednostima i generalizacijom postupka učenja dobiva se delta pravilo. Delta pravilo je postalo osnova za učenje većine modela neuronskih mreža. Željena izlazna vrijednost označava se sa T_j, a izračunata izlazna vrijednost s A_j. Razlika između dvije vrijednosti naziva se odstupanje δ_j.

$$\delta_j = T_j - A_j. \tag{2.3}$$

Gdje su:

T_j – željena izlazna vrijednost neurona *j*;

A_j – izračunana izlazna vrijednost neurona *j*.

Težine možemo korigirati pomoću izraza:

$$\Delta_{ij} = \eta \cdot \delta_j \cdot x_i \ , \tag{2.4}$$

$$w_{ij}(n+1) = w_{ij}(n) + \Delta_{ij}. \tag{2.5}$$

Gdje su:

i – indeks neurona ulaznog sloja;

j – indeks neurona izlaznog sloja;

Δ_{ij} – faktor promjena težine;

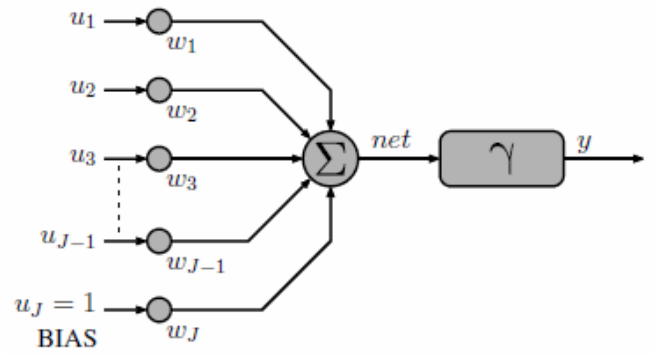
η – koeficijent brzine učenja;

$w_{ij}(n)$ – težina prije podešavanja;

$w_{ij}(n+1)$ – težina veze između neurona *i* ulaznog sloja i neurona *j* izlaznog sloja nakon podešavanja.

2.2. Višeslojne neuronske mreže

2.2.1. Model statičkog neurona



Slika 2.1. Standardan model statičkog neurona

Na slici 2.1 vidi se da neuron posjeduje više ulaza i jedan izlaz. Neuron sadrži dvije funkcije, a to su sumator Σ i aktivacijska funkcija γ. Svaki neuron koji sudjeluje u procesu učenja posjeduje poseban ulaz jedinične vrijednosti koji je u strukturi neuronske mreže realiziran vezom sa zasebnim neuronom, *Bias*-om [7], konstantnog iznosa u vrijednosti 1.

Funkcija sume statičkog neurona jest suma umnožaka ulaza neurona i njihovih težinskih faktora. Koristi se izraz *net* za opisivanje te funkcije te se zapisuje sljedeći izraz:

$$net = \sum_{j=1}^{J} w_j \cdot |u_j \ , \tag{2.6}$$

$$y = \gamma(net). \tag{2.7}$$

Gdje su:

net – rezultat funkcije sume;

w_j – vrijednost *j*-te težine;

u_j – vrijednost *j*-tog ulaza;

γ – aktivacijska funkcija neurona;

y – izlazna vrijednost neurona.

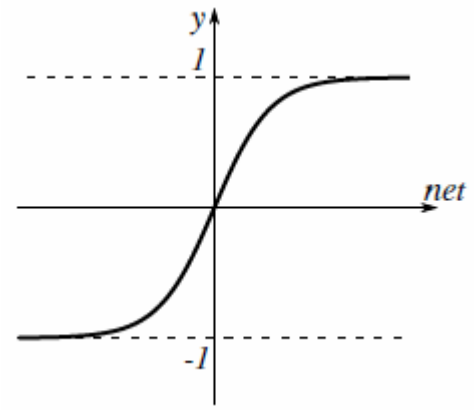
Za aktivacijsku se funkciju odabiru monotone rastuće funkcije sa zasićenjem. Neki su primjeri prikazani u nastavku.

Bipolarna sigmoidalna funkcija (slika 2.2)

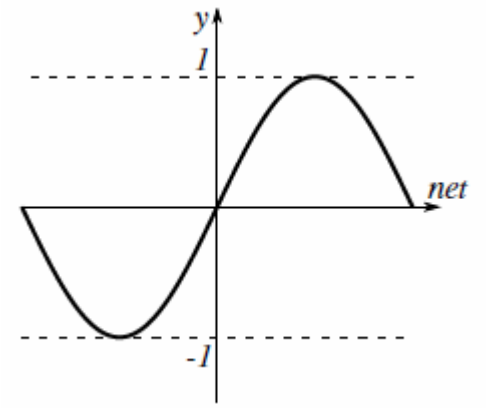
$$y = \frac{2}{1 + e^{-net}} - 1 \ . \tag{2.8}$$

Sinusna funkcija (slika 2.3)

$$y = \sin(net) \quad . \tag{2.9}$$



Slika 2.2. Bipolarna sigmoidalna funkcija



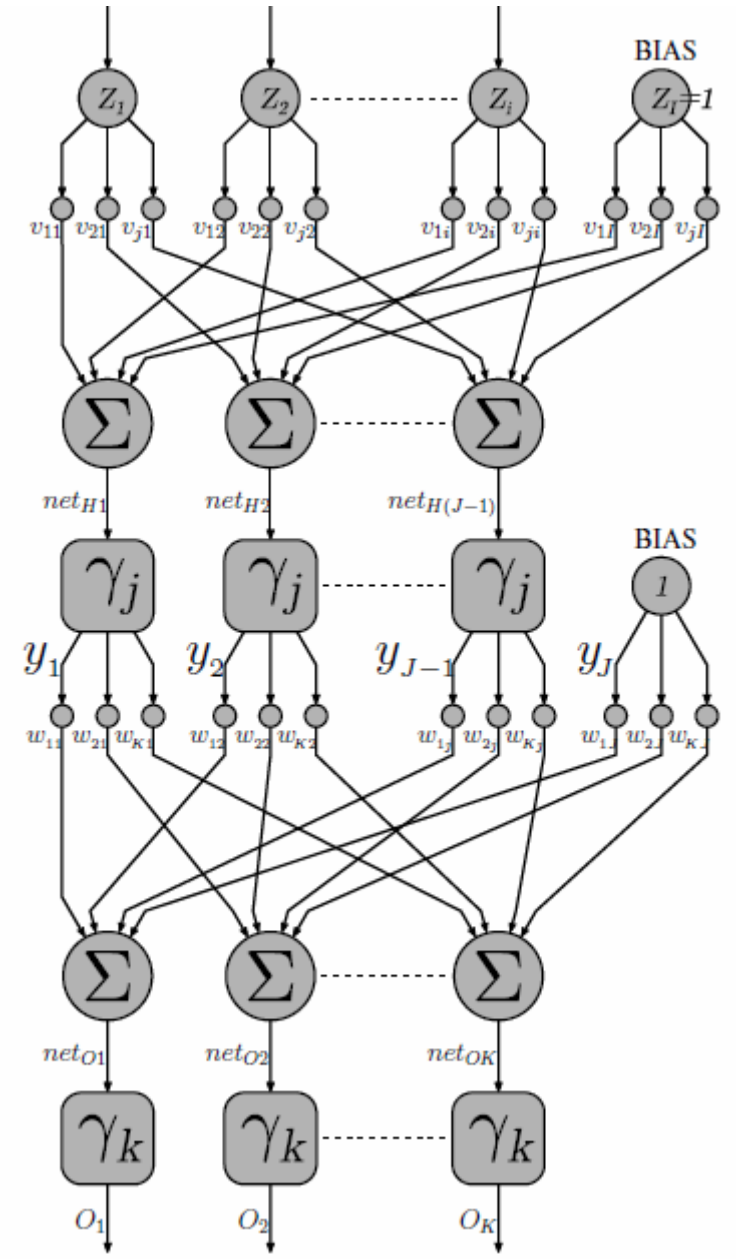
Slika 2.3. Sinusna funkcija

2.2.2. Model statičke neuronske mreže

Kao što je već spomenuto ranije u poglavlju 1.4., neuron sadrži tri sloja: ulazni, sakriveni i izlazni sloj. Ulazni i izlazni slojevi u neposrednoj su interakciji s okolinom, dok sakriveni nije. Slojevi su međusobno povezani vezama koje sadrže težinske koeficijente. Najčešće se koristi statička unaprijedna višeslojna neuronska mreža (SNN, *Static Neural Network*).

Ulazi neurona ulaznog sloja (Z_i) su ulazi u mrežu. Ulazni sloj se povezuje sa sakrivenim slojem pomoću veza opterećenih težinama v_{ji}. Svi su slojevi potpuno umreženi, što znači da je svaki neuron promatranog sloja vezan sa svakim neuronom prethodnog sloja. Jedino se *bias* neuroni ne povezuju na taj način.

Broj je sakrivenih slojeva proizvoljan, iako se najčešće koriste jedan ili dva sakrivena sloja neurona. Cybenko [8] i Funahashi [9] pokazali su u svojim radovima da je moguće s jednim sakrivenim slojem prikazati dobru aproksimaciju bilo koje kontinuirane funkcije. Uvjet za takve neuronske mreže jest dovoljan broj neurona u sakrivenom sloju. Broj tih neurona nije još dokazan pa ga se određuje eksperimentom [10].



Slika 2.4. Model statičke unaprijedne neuronske mreže s jednim sakrivenim slojem

2.3. Učenje povratnim rasprostiranjem pogreške

Postoje tri osnovna karaktera učenja: učenje s učiteljem (*supervised learning*), učenje bez učitelja (*unsupervised learning*) i kombinacija prethodna dva (*reinforcement learning*). Pri učenju povratnim prostiranjem pogreške koristi se učenje s učiteljem.

Postupak učenja je podešavanje težinskih koeficijenata između slojeva mreže. Cilj je da se izlazi mreže što više približe zahtijevanim vrijednostima korištenjem zadanih ulaza. Uglavnom se dobije aproksimirana vrijednost, a kvaliteta izlaza ovisi o više čimbenika kao što su: zadatak učenja, odabrana topologija mreže te odabrani algoritam učenja.

Algoritam se odabire ovisno o broju koraka učenja. Postoje iteracijski algoritmi i algoritmi za jedan korak učenja. U iterativnom postupku učenja, kod adaptacije težinskih koeficijenata, mreža uzastopno prikazuje ulazne i odgovarajuće izlazne veličine. Svaki korak učenja sadrži dvije faze: unaprijednu i povratnu fazu. Parametri učenja mogu se mijenjati pomoću *batch* procedure ili pomoću *pattern,* odnosno *stochastic* procedure. Kod *batch* procedure, parametri se mijenjaju jednom nakon prolaska čitavog ulaznog skupa kroz mrežu, a promjena se odvija srednjom pogreškom koja je proizvod tog ulaznog skupa. Kod *stochastic* procedure, parametri se mijenjaju kod svakog ulazno-izlaznog para skupa. Ova procedura se češće koristi, a bit će korištena i u ovom radu.

2.3.1. Unaprijedna faza učenja za statičke mreže

U ovoj se fazi uzima vrijednost svih ulaza *Z*. Pomoću njih se izračunava izlaz mreže *O*. Također, određuju se početne vrijednosti težinskih koeficijenata *W* i *V*, najčešće pomoću generatora slučajnih brojeva u rasponu vrijednosti ulaza iz skupa učenja. Npr. za ulazne vrijednosti između 1 i -1, početne se vrijednosti težina odabiru u istom rasponu.

2.3.1.1. Sakriveni sloj

Potrebno je izračunati *net* vrijednosti u sakrivenom sloju.

$$net_{Hj} = \sum_{i=1}^{I} v_{ji} \cdot Z_i\,, \;\; j = 1,2,\dots,J-1\;. \tag{2.10}$$

Gdje je:

I – broj ulaznih neurona, uključujući i *bias*,

J – broj neurona u sakrivenom sloju, uključujući i *bias.*

Da bi se izračunale izlazne vrijednosti *y* sakrivenog sloja, potrebno je izračunati aktivacijsku funkciju za dobivene vrijednosti net_{Hj}. Za aktivacijsku funkciju može se uzeti neka od funkcija (2.3), (2.4) ili neka druga. Za primjer će se odabrati sigmoidalna funkcija. Izlaz neurona sakrivenog sloja pomoću takve se funkcije računa prema:

$$y_j = \frac{2}{1+e^{-net_{Hj}}} - 1, \;\; j = 1,2,\dots,J-1\;. \tag{2.11}$$

$$y_J = 1 \tag{2.12}$$

Vrijednosti gore izračunatih izlaza se preko težinskih koeficijenata w_{kj} spajaju na ulaze neurona izlaznog sloja.

2.3.1.2 Izlazni sloj

Sada je *net* neurona izlaznog sloja suma umnožaka težinskih koeficijenata koji spajaju izlaze neurona sakrivenog sloja i vrijednosti izlaza tih neurona.

$$net_{Ok} = \sum_{j=1}^{J} w_{kj} \cdot y_j, \quad k = 1,2, \dots, K\,. \tag{2.13}$$

Gdje je:

K – broj neurona izlaznog sloja (broj izlaza).

Kod odabira aktivacijske funkcije izlaznog sloja nije loše odabrati linearnu funkciju jer se tako ostvaruje vrijednost izlaza mreže veće od 1.

2.3.2. Povratna faza učenja

U ovoj se fazi vrši korekcija vrijednosti težinskih koeficijenata veza između slojeva. Korekcija se odvija na temelju pogreške učenja, koja je ustvari razlika između željenog i dobivenog izlaza. Korekcija se primjenjuje sve dok se ne zadovolji dozvoljena razina pogreške (koju zadaje učitelj).

Suma kvadrata pogreške najčešće je korištena funkcija cilja koja je namijenjena za izračun odstupanja izlaza mreže od željene vrijednosti:

$$E = \frac{1}{2}\sum_{n=1}^{N}(d_n - O_n)^2\ . \tag{2.14}$$

Gdje je:

N – broj elemenata u skupu za učenje;

O_n - dobivena vrijednost izlaza;

d_n – željena vrijednost izlaza.

Dalje se odabranom funkcijom cilja vrši promjena koeficijenata primjenom nekog od algoritama nelinearnog programiranja. Izraz:

$$\vartheta(n+1) = \vartheta(n) + \Delta\vartheta(n). \tag{2.15}$$

Gdje je:

$$n - \ trenutni\ korak\ učenja;$$

$$\Delta\vartheta - \ veličina\ promjene\ parametra\ učenja;$$

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

Grešku $E(\vartheta)$ može se aproksimirati pomoću prva dva člana Taylorovog reda:

$$E(\vartheta + \Delta\vartheta) \approx E(\vartheta) + \Delta E(\vartheta)\ , \tag{2.16}$$

$$\Delta E(\vartheta) = \Delta\vartheta^T\ \nabla E(\vartheta), \tag{2.17}$$

$$\nabla E(\vartheta) = \frac{\partial E(\vartheta)}{\partial \vartheta}\ . \tag{2.18}$$

Gdje je:

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

Poželjno da je se pogreška smanjuje najvećim mogućim iznosom, a za to je potrebno odrediti vrijednost $\Delta\vartheta$ za koju promjena pogreške učenja $\Delta E(\vartheta)$ sadrži najveći negativni iznos. Uvjet se ostvaruje kroz izraz:

$$\Delta\vartheta = -\eta\nabla E(\vartheta)\ . \tag{2.19}$$

Gdje je:

η – koeficijent brzine učenja.

Vrijednost brzine učenja određuje učitelj. Svrha je koeficijenta usmjeriti vrijednosti težina u smjeru najvećeg pada ukupne pogreške učenja. Tako uvrštavanjem (2.19) u (2.15) dobivamo:

$$\vartheta(n+1) = \vartheta(n) - \eta\nabla E(\vartheta(n))\ . \tag{2.20}$$

Ovim je izrazom definiran algoritam povratnog prostiranja pogreške koji predstavlja najpoznatiji i najčešće primjenjivan način promjene parametra učenja. Najveći mu je nedostatak velik broj potrebnih iteracija, odnosno koraka učenja.

2.3.2.1. Momentum

Kako bi se ubrzao proces učenja, odnosno kako bi se smanjio broj koraka učenja, uz istu dozvoljenu pogrešku moguće je modificirati algoritam tako da se ugradi momentum (zamaha). Momentum može biti prvog i drugog reda, a opisan je izrazom:

$$\Delta\vartheta(n) = -\eta\nabla E\big(\vartheta(n)\big) + \propto \Delta\vartheta(n-1) + \beta\Delta\vartheta(n-2). \tag{2.21}$$

Gdje je:

n - trenutna promjena parametara učenja;

(n – 1) - promjena parametra učenja u prošlom koraku;

(n – 2) - promjena parametra učenja u pretprošlom koraku;

$\propto$ - vrijednost koeficijenta momentuma prvog reda. Vrijednost zadaje učitelj (obično u intervalu od 0.1 do 0.9);

β - vrijednost koeficijenta momentuma drugog reda. Vrijednost se računa u odnosu na koeficijent $\propto$.

Uvrštavanjem (2.21) u(2.15) dobiva se konačni izraz za promjenu parametara učenja. Pomoću momentuma brzina se algoritma može povećati i nekoliko puta, ali zato ne garantira konvergenciju.

$$\vartheta(n+1) = \vartheta(n) - \eta\nabla E\big(\vartheta(n)\big) + \propto \Delta\vartheta(n-1) + \beta\Delta\vartheta(n-2). \tag{2.22}$$

2.4. Promjena parametara učenja

Povratnom se fazom učenja dobiva izraz (2.22) za promjenu parametara učenja koji se kasnije koristi za izračun novih težina svakog sloja neuronske mreže.

2.4.1. Promjena težina izlaznog sloja

Kod povratne se faze učenja prvo mijenjaju težine između izlaznog i sakrivenog sloja. To se provodi izrazom:

$$w_{kj}(n+1) = w_{kj}(n) - \eta\nabla E\left(w_{kj}(n)\right) + [\propto \Delta w_{kj}(n-1)]. \tag{2.23}$$

Gradijent pogreške ∇E za težine w_{kj} računa se prema (2.18).

$$\nabla E(n) = \frac{\partial E(n)}{\partial w_{kj}} \quad . \tag{2.24}$$

Problem određivanja pripadajućeg gradijenta pogreške lako se rješava uzastopnim parcijalnim derivacijama:

$$\frac{\partial E(n)}{\partial w_{kj}} = \frac{\partial E(n)}{\partial O_k}\frac{\partial O_k}{\partial net_{Ok}}\frac{\partial net_{Ok}}{\partial w_{kj}} \quad . \tag{2.25}$$

Iz navedenog slijede izrazi za pojedine parcijalne derivacije iz (2.25):

$$\frac{\partial E(n)}{\partial O_k} = \ -(d_k - O_k) \quad , \tag{2.26}$$

$$\frac{\partial O_k}{\partial net_{Ok}} = \gamma' = 1 \quad , \tag{2.27}$$

$$\frac{\partial net_{Ok}}{\partial w_{kj}} = y_j \, . \tag{2.28}$$

Prema [7], karakteristična vrijednost algoritma povratnog prostiranja greške prema definiciji jest:

$$\delta = -\frac{\partial E(N)}{\partial net_{Ok}} \quad . \tag{2.29}$$

Pomoću gornje definicije možemo odrediti δ_{Ok} i (2.26):

$$\delta_{Ok} = \ d_k - O_k. \tag{2.30}$$

Uvrštavanjem (2.26), (2.27), (2.28) i (2.30) u (2.25) izraz za $\nabla E(n)$ na kraju izgleda:

$$\nabla E(n) = \frac{\partial E(n)}{\partial w_{kj}} = -(d_k - O_k) \cdot y_j = -\delta_{Ok} \cdot y_j \, . \tag{2.31}$$

Uvrštavanjem (2.31) u (2.23) dobiva se izraz za algoritam promjene težine izlaznog sloja:

$$w_{kj}(n+1) = w_{kj}(n) + \eta\delta_{Ok}y_j + \left[\propto \Delta w_{kj}(n-1)\right]. \tag{2.32}$$

2.4.2. Promjena težina skrivenog sloja

Nakon promjene težina w_{kj} izlaznog sloja potrebno je promijeniti težine sakrivenog sloja v_{ij}. Analogno promjeni težina izlaznog sloja dobiva se izraz za promjenu težina sakrivenog sloja:

$$v_{ij}(n+1) = v_{ij}(n) - \eta\nabla E\left(v_{ij}(n)\right) + [\propto \Delta v_{ij}(n-1)]. \tag{2.33}$$

Također, analogno izrazu gradijenta pogreške u izlaznom sloju (2.25) postavlja se izraz s parcijalnim derivacijama za gradijent $\nabla E(n)$ u sakrivenom sloju:

$$\frac{\partial E(n)}{\partial v_{ij}} = \frac{\partial E(n)}{\partial y_j}\frac{\partial y_j}{\partial net_{Hj}}\frac{\partial net_{Hj}}{\partial v_{ij}} \quad . \tag{2.34}$$

Na slici 2.4 vidi se da svaki neuroni izlaznog sloja utječu na svaku težinu skrivenog sloja, stoga slijedi izraz:

$$\frac{\partial E(n)}{\partial y_j} = \sum_{k=1}^{K} \left(\frac{\partial E(n)}{\partial O_k}\frac{\partial O_k}{\partial net_{Ok}}\frac{\partial net_{Ok}}{\partial y_j}\right) \quad . \tag{2.35}$$

Pri čemu su parcijalne derivacije u izrazu (2.35) jednake:

$$\frac{\partial E(n)}{\partial O_k} = -(d_k - O_k) \quad , \tag{2.36}$$

$$\frac{\partial O_k}{\partial net_{Ok}} = 1 \quad , \tag{2.37}$$

$$\frac{\partial net_{Ok}}{\partial y_j} = w_{kj} \, , \tag{2.38}$$

$$k = 1,2, \dots , K; \; j = 1,2, \dots , J = 1 \, .$$

Daljnjim uvrštavanjem izraza (2.36), (2.37) i (2.38) u izraz (2.35) dobiva se:

$$\frac{\partial E(n)}{\partial y_j} = -\sum_{k=1}^{K} (d_k - O_k)\, w_{kj} = -\sum_{k=1}^{K} \delta_{Ok}\, w_{kj}. \tag{2.39}$$

Dvije preostale parcijalne derivacije iz izraza (2.34) određuju se iz izraza (2.10) i (2.11). Tako se dobije:

$$\frac{\partial y_j}{\partial net_{Hj}} = \gamma' = \frac{1}{2}\left(1 - y_j^2\right) \, , \tag{2.40}$$

$$\frac{\partial net_{Hj}}{\partial v_{ij}} = Z_i \quad . \tag{2.41}$$

Povratkom (2.39), (2.40) i (2.41) u (2.34) i u (2.33) slijedi izraz:

$$v_{ij}(n+1) = v_{ij}(n) + \frac{1}{2}\eta\left(1 - y_j^2\right)Z_i\left(\sum_{k=1}^{K}\delta_{Ok}\, w_{kj}\right) + \left[\propto \Delta v_{ij}(n-1)\right]. \qquad (2.42)$$

Ovi posljednji izvodi vrijede jedino za mrežu na slici 2.4. te za njoj odabrane aktivacijske funkcije neurona izlaznog i sakrivenog sloja. U slučaju promjene aktivacijske funkcije mijenjaju se samo parcijalne derivacije po funkciji sume neurona, dok se ostale izvode identično ovom primjeru.

Svakim novim sakrivenim slojem izrazi se nešto proširuju, ali numerički postupak ostaje isti. Povećanjem slojeva dobiva se na kvaliteti mreže u odnosu s učenjem i generalizacijom, ali zahtijeva puno više procesorskog vremena.

2.5. Ocjena točnosti algoritma učenja

Kako bi se ocijenila točnost neuronske mreže za vrijeme učenja i nakon njega, za određeni zadatak potrebno je odrediti mjeru točnosti u usporedbi s traženim vrijednostima izlaza mreže. Pa tako postoje srednja kvadratna pogreška, korijen srednje kvadratne pogreške te normalizirani korijen srednje kvadratne pogreške.

U ovom primjeru za mjeru točnosti korišten je normaliziran korijen srednje kvadratne pogreške (*normalized root mean square error*, dalje u tekstu NRMS), a računa se prema izrazu:

$$NRMS = \frac{\sqrt{\frac{\sum_{n=1}^{N}(d_n - O_n)^2}{N}}}{\sigma_{d_n}}. \qquad (2.43)$$

Gdje se σ_{d_n} računa prema izrazu:

$$\sigma_{d_n} = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(d_n - d)^2}, \qquad (2.44)$$

$$d = \frac{\sum_{n=1}^{N} d_n}{N}. \qquad (2.45)$$

NRMS karakterizira bezdimenzionalnost koja osigurava neovisnost mjere o dimenzijama učenih veličina te omogućuje usporedbu izvedenih algoritama učenja s drugim algoritmima.

3. NEURONSKA MREŽA OPTIMIZIRANJEM ROJA ČESTICA

3.1. Optimizacijski algoritmi

Problem optimizacije, odnosno pronalaženje najboljeg rješenja u matematičkom skupu svih mogućih rješenja, u skupini je problema čije se rješavanje omogućilo pojavom računala. Njihovo rješavanje je isključivo iterativne prirode - slijede se koraci kojima se nastoji poboljšati prethodno rješenje. Algoritam tako pronalazi lokalne optimume koji predstavljaju najbolja rješenja u neposrednoj okolini. Često to predstavlja problem za pronalazak globalnog optimuma kada algoritam nije u stanju pretražiti prostor izvan okoline.

S razvijanjem tehnologije računala su bila u stanju povećavati broj operacija i rješavati ih u sve kraćem periodu. No, bez obzira na brzine računala koje danas mogu izračunati stotine trilijuna operacija u sekundi, sistematično pretraživanje svih mogućih rješenja pomoću metode iscrpnog pretraživanja prostora svih mogućih rješenja, za neke probleme bi trajalo više od starosti svemira jer sadrže 10^{100} i više stanja. Zbog toga su se razvili različiti pristupi, a među njima su najpoznatije heuristike, metaheuristike koje definiraju strategiju pretraživanja prostora stanja.

Takve metode ne garantiraju pronalazak optimalnog rješenja, ali omogućuju pronalazak dovoljno dobrog rješenja u razumnom vremenskom periodu. Pristupi heuristike i metaheuristike, iako se temelje na istoj ideji, razlikuju se. Heuristike se definiraju za određene probleme, dok metaheuristike definiraju globalnu strategiju koju je moguće primijeniti na širok spektar različitih problema.

U ovom se radu razmatra metaheuristika roja čestica.

3.2. Općenito o optimiziranju algoritmom roja čestica

Optimiziranje rojem čestica (*particle swarm optimization*, dalje u tekstu PSO) razvili su Kennedy i Eberhart još 1995. godine [11], a temelji se na imitiranju ponašanja jata ptica ili riba u prirodi. Zaključili su da prilikom kretanja populacije jedinki dolazi do razmjene informacija među susjednim jedinkama, što se naziva inteligencijom roja (*swarm intelligence*). Npr. kad pčela pronađe lokaciju bogatu peludom, informira ostatak roja o novoj lokaciji. U međuvremenu je moguće da će jedinke iz roja na putu prema zadanoj lokaciji naići na bolju lokaciju i o tome obavijestiti susjedne jedinke koje dalje dijele informaciju. Pomoću interakcije tog tipa, jedinke, a posredno i roj, dolaze do najbolje lokacije [12]. Algoritam je u određenoj mjeri inspiriran i sociološkim interakcijama između pojedinaca u populaciji, gdje svaki pojedinac pamti svoje, do

tada pronađeno najbolje rješenje problema, te ima uvid u najbolje pronađeno rješenje svojih susjeda, stoga pretraživanje usmjerava uzimajući u obzir te dvije komponente.

Razvijanjem algoritma pokazalo se da se algoritam može jednostavno implementirati i da sadrži relativno malen broj parametara. Također, algoritam je pokazao dobre rezultate na širokom skupu problema.

3.3. Opis algoritma

Na početku rada algoritma inicijaliziraju se čestice, dodjeljuju im se slučajne vrijednosti za položaj i brzinu te se definiraju susjedstva. Čestice pripadaju različitim susjedstvima, čime se omogućava tok informacija među susjedstvima. Izvođenje algoritma provodi se u diskretnim koracima. U svakom koraku za sve čestice se računa novi vektor brzine, novi položaj i vrijednost funkcije za novi položaj. Kod PSO-a, na potencijalna se rješenja gleda kao na čestice u višedimenzionalnom prostoru – n-parametara tvori n-dimenzionalan prostor u koji se postavljaju čestice. Čest je slučaj da se taj prostor ograniči po dimenzijama kako čestice ne bi „lutale“. Ograničenja se postavljaju za svaki specifični problem drugačije.

Budući da je PSO iterativna metoda, kao i kod povratnog prostiranja pogreške, ocjenjuje se kvaliteta učenja uz pomoć funkcije cilja (*fitness function*). Ta funkcija može biti i NRMS kao i kod povratnog prostiranja pogreške ili neka druga funkcija.

Novi vektor brzine (3.1) izačunava se zbrajanjem vektora trenutne brzine, vektora određenog razlikom svojeg najboljeg i trenutnog položaja, te vektora određenog razlikom lokalno najboljeg (u susjedstvu najboljeg pronađenog položaja) i trenutnog položaja. Novi se položaj računa zbrajanjem novog vektora brzine trenutnom položaju (3.2)

Vrijednost se funkcije za novi položaj zamjenjuje ako je novi položaj bolji od prethodnog najboljeg položaja za određenu česticu.

$$v_i^{t+1} = c_1 v_1^t + c_2 r_p (p_i - x_1^t) + c_3 r_l \left(l_i - x_i^t\right) \ , \tag{3.1}$$

$$x_i^{t+1} = x_1^t + v_i^{t+1} \ . \tag{3.2}$$

Gdje su :

$\overrightarrow{v_i^t}$ – *i*-ta komponenta trenutnog vektora brzine;

$\overrightarrow{v_i^{t+1}}$ – *i*-ta komponenta novog vektora brzine;

$\overrightarrow{x_i^t}$ – *i*-ta komponenta radij-vektora trenutnog položaja čestice;

$\overrightarrow{x_i^{t+1}}$ – *i*-ta komponenta radij-vektora novog položaja čestice;

r_p – slučajna vrijednost iz uniformne distribucije [0,1];

r_l – slučajna vrijednost iz uniformne distribucije [0,1];

$\overrightarrow{p_i}$ – radij-vektor osobnog najboljeg položaja;

$\overrightarrow{l_i}$ – radij-vektor globalnog najboljeg položaja;

c_1, c_2, c_3 – faktori povjerenja.

3.4. Podešavanje i odabir parametara

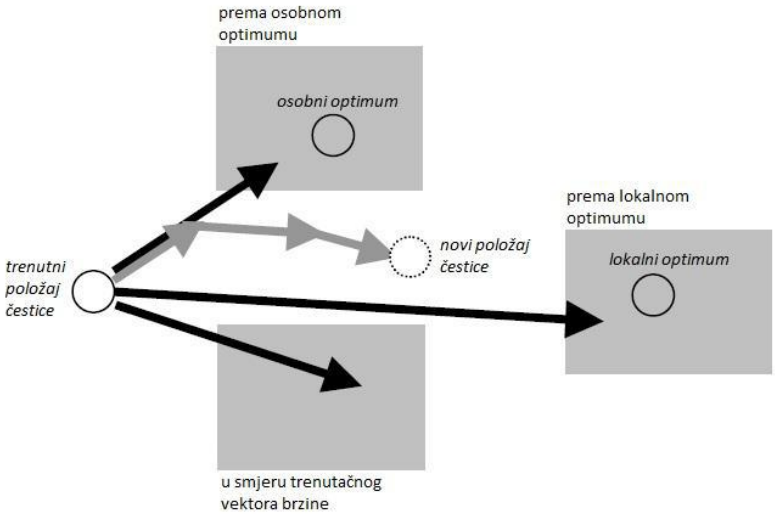
U prethodnim izrazima (3.1) i (3.2) vide se korišteni parametri koji se u ovom poglavlju analiziraju. Analizira se njihov utjecaj na rad i performanse algoritma te odabir njihovih vrijednosti.

Prvi parametri koji se analiziraju su faktori povjerenja. Oni određuju mjeru u kojoj se čestica kreće prema poznatim optimumima [12] (c_2 i c_3), tj. u kojoj mjeri nastavlja kretanje u dotadašnjem parametru (c_1). Razmatraju se granični slučajevi da bi se vidjeli pojedinačni utjecaji tih faktora:

1. slučaj: $c_1 > 0$, $c_2 = c_3 = 0$; čestica se nastavlja kretati u smjeru vektora brzine prethodnog koraka (jačine i smjera). Faktor je dio člana koji se naziva individualna komponenta. Često je se naziva inercijom, a u nekim se literaturama označava ω ili *w*.

2. slučaj: $c_2 > 0$, $c_1 = c_3 = 0$; čestica se kreće u smjeru svojeg najboljeg rješenja te se nastoji vratiti u svoje najbolje rješenje. Faktor je dio člana koji se naziva kognitivna komponenta. U nekim se literaturama označava kao c_1.

3. slučaj: $c_3 > 0$, $c_1 = c_2 = 0$; čestica se kreće prema lokalnom najboljem rješenju, onome koje je pronađeno od drugih čestica. Faktor je dio člana koji se naziva socijalna komponenta. U nekim se literaturama označava kao c_2.



Slika 3.1. Prikaz određivanja novog položaja čestice u koraku algoritma [13]

O vrijednostima gore opisanih parametara ovisi brzina pronalaska i kvaliteta konačnog rješenja. Npr. ako odaberemo malu vrijednost komponente *c1*, čestica neće pretraživati nova područja, već će se kretati gotovo isključivo u smjeru najbolje čestice. Time se povećava mogućnost da algoritam zaglavi u lokalnom optimumu. Također, nije dobro postaviti veliku vrijednost ove komponente zbog mogućnosti da će čestica preletjeti preko potencijalno dobrog rješenja. Važan je i omjer vrijednosti između komponenti, tj. da se parametri ne biraju neovisno jedni od drugih. Prema Van den Berghu [15], da bi se osigurala konvergencija rješenja potrebno je zadovoljiti izraz :

$$(c_2 + c_3)/2 - 1 < c_1 \tag{3.3}$$

Može se koristiti i vremenski promjenjiv faktor c_1, na način da se linearno smanjuje vrijednost faktora. Na taj način na početku pretrage postiže se *eksplorativno* (globalno pretraživanje) ponašanje čestice, dok se pri kraju pretrage dobiva *eksploativno* (lokalno pretraživanje) ponašanje. U ovom se radu to neće razmatrati.

Osim faktora povjerenja, potrebno je ustanoviti broj čestica u algoritmu. Obično se ne mijenja broj čestica do kraja izvođenja algoritma. Složeniji algoritmi koriste razne verzije, npr. povećavaju ili smanjuju broj čestica, izbacuju i ubacuju nove, slučajno generirane čestice itd. Prema Clercu [12], ustanovljen je optimalan broj čestica u roju. Eksperimentima je dokazano da najbolje rezultate daje algoritam s dvadeset do četrdeset čestica. Potrebno je napomenuti da algoritam nasumično generira početnu populaciju.

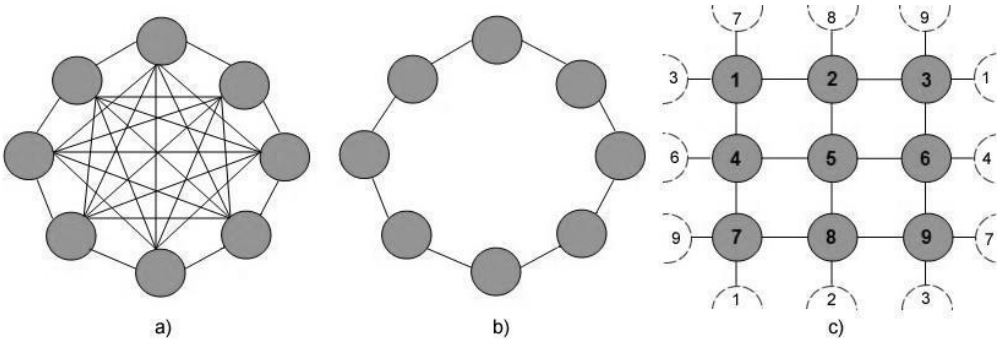
Još je potrebno opisati susjedstvo. Ovim se parametrom definira način na koji su čestice povezane među sobom. Postoje uobičajene topologije povezanosti čestica u susjedstvu, a to su: *potpuno*

povezana, *zvjezdasta* i *prstenasta* topologija. S vremenom se razvila i *von Neumannova*, koja se također često koristi. U ovom radu koristi se prstenasta struktura.

Potpuno povezana – svaka je čestica povezana sa svim ostalim česticama. Informacija o najboljem rješenju je globalna i poznata svakoj čestici.

Prstenasta – svaka je čestica povezana s lijevom i desnom susjednom česticom. Tako povezane čestice tvore krug ili prsten.

von Neumannova – svaka čestica ima četiri susjeda, tako tvore mrežu. Rubne su čestice povezane s rubnim česticama na drugom kraju.



Slika 3.2. Topologije: a) potpuno povezana, b) prstenasta, c) von Neumannova [13]

Dakle, nakon opisa pojedinih dijelova unutar algoritma i odabranih parametara, algoritam se može opisati u sljedećih nekoliko koraka:

- Inicijalizira se roj od N čestica kojima se dodjeljuju početne brzine i pozicije u zadanom intervalu, najčešće slučajnim odabirom;
- Svaka se čestica ocijeni funkcijom cilja, ažuriraju se individualno najbolje pozicije, lokalno najbolje pozicije, te globalno najbolja pozicija;
- Izračunaju se nove brzine i nove pozicije (3.1) (3.2);
- Algoritam se ponavlja dok se ne pronađe zadovoljavajuća vrijednost funkcije cilja ili kad se dosegne maksimalan broj iteracija;
- Iz roja uzimamo najbolju česticu.

4. UČENJE NEURONSKE MREŽE ALGORITMOM POVRATNIM PROSTIRANJEM POGREŠKE I PSO ALGORITMOM

U ovom će se poglavlju prikazati usporedba učenja neuronske mreže pomoću dva algoritma (EBP i PSO). Primjer koji se uzima dosta je jednostavan, ali njime se može prikazati osnovna razlika između dva načina učenja.

4.1. XOR klasifikacijski problem

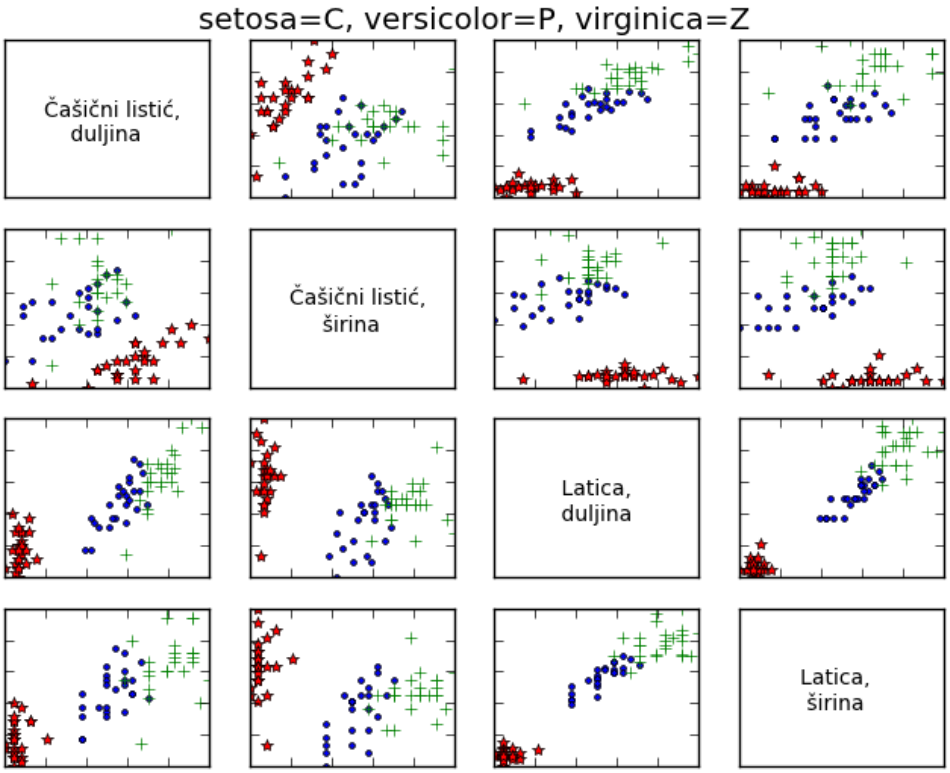
Ekskluzivni ILI (eng. XOR) je logički problem. Jedan je od najjednostavnijih primjera s linearno neseparabilnim uzorcima [16], pa je zbog toga čest primjer za ispitivanje svojstava različitih modela umjetnih neuronskih mreža. Ulazi su određeni s dvije binarne varijable koje mogu biti vrijednosti 0 ili 1, dok se izlaz sastoji od jedne binarne varijable koja također može biti isključivo 0 ili 1. Logičke vrijednosti su opisane u tablici 1.

Tablica 1. Logičke vrijednosti XOR problema

Ulaz		Izlaz
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

4.2. *Iris* set podataka

Drugi problem na kojem se obavlja učenje mreže jest Iris set podataka, prvi put spomenut 1936. godine u radu Ronalda Fishera[17]. Radi se o klasifikaciji tri različita cvijeta roda *Iris(Iris setosa, Iris virginica i Iris versicolor)*. Klasifikacija se izvodi na temelju četiri ulazna podatka: duljina latice, širina latice, duljina čašičnog listića, te širine čašičnog listića. Za svaku od vrste cvijeta daje se 50 uzoraka, tako da je ukupno 150 uzoraka. Također se radi o linearno neseparabilnom problemu. U mrežu se ulazi s četiri ulazna podatka, tj. neurona u ulaznom sloju i jedan u izlaznom sloju. Uzorci za učenje prikazani su 2-D grafom, gdje svaki red prikazuje odnos jedne značajke sa svim ostalima. Na njima je vidljivo da su podaci za cvijet *Iris setosa* (prikazano crvenim zvijezdama) linearno separabilni od podataka za ostala dva cvijeta. Podaci za *Iris virginicu* (zeleni križići) i *Iris versicolor* (plavi kružići) se preklapaju, odnosno linearno su neseparabilni.



Slika 4.1. Uzorci za učenje problema Iris [18]

4.3. Izrada mreže i unaprijedna faza učenja

Broje neurona u ulaznom i izlaznom sloju zadan je problemom koji se riješava. Broj se neurona u skrivenom sloju može podešavati. Početne se težine zadaju slučajnim odabirom vrijednosti između 0 i 1, a *bias*-a je vrijednost 1. Za aktivacijsku funkciju odabrana je sigmoidalna funkcija. Unaprijedna faza učenja je ista za oba načina učenja.

Također je potrebno odabrati broj koraka učenja. Kroz učenje mreže prate se izlazne vrijednosti te vrijednosti greške učenja (NRMS, te funkcija cilja). Potom se te vrijednosti uspoređuju između dva načina učenja. Vrijednosti težina unutar slojeva su nasumično odabrane i različite su kod pojedinog načina učenja.

4.3.1. XOR problem

Mreža se sastoji od dva ulazna neurona, određeni broj neurona plus bias u skrivenom sloju i jednog neurona u izlaznom sloju.

Zamišljena su tri slučaja učenja koja će se pratiti, a prikazani su tablicom 2.

Tablica 2.. zadani parametri tri slučaja kod XOR problema

	1. slučaj	2. slučaj	3. slučaj
Broj koraka	100	200	500
Broj neuron u skrivenom sloju	3	4	5

4.3.2. Iris set problem

Mreža se sastoji od četiri ulazna neurona, određeni broj plus bias u skrivenom sloju i jednog neurona u izlaznom sloju. Od 150 uzoraka koristit ćemo 120 za učenje mreže. Budući da je problem *Iris* prilično zahtjevniji u radu se objavljuje slučaj kada program dosegne zadovoljavajuću konvergenciju. Koristi se tri neurona u skrivenom sloju.

4.4. Povratna faza

4.4.1. Povratna faza učenja algoritmom povratnog prostiranja pogreške

Kod ovog algoritma zadaje se koeficijent brzine učenja η, najčešće vrijednosti $10^{-4} < \eta < 10$. Za XOR problem je odabran veći koeficijent jer je jednostavniji problem i ne očekuje se problem konvergencije. Za drugi, *Iris* set problem je odabrani niži koeficijent jer je problem zahtjevniji, stoga učenje treba biti pažljivije, inače neće doći do željene konvergencije. Odabire se i momentum α, najčešće vrijednosti $0.1 < \alpha < 1$. U ovom se primjeru zanemaruje momentum, tj. njegova će vrijednost biti 0. Kao mjera točnosti algoritma uzima se NRMS opisan u poglavlju 2.5, kojeg se može i zadati ukoliko se želi postići prekid programa kada je rješenje u zadovoljavajućim okvirima. Zadani su parametri prikazani u tablici 3.

Tablica 3. parametri povratnog prostiranja pogreške u zadanim problemima

	XOR	Iris set
η	0.1	0.001
Zadani NRMS	0.01	0.1
Momentum	0	0

4.4.2. Povratna faza učenja PSO algoritmom

Kod ovog algoritma se zadaju težinski koeficijenti opisani u poglavlju 3.2. Inercijski koeficijent, kognitivni, te socijalni koeficijent(c_1,c_2 i c_3). Određuje se broj čestica, te broj dimenzija. Isto tako postavljaju se gornja i donja granica potrage. Početna pozicija se zadaje nasumično unutar dozvoljenih granica potrage(U_B – gornja i L_B-donja) dok se početna brzina zadaje u ovisnosti o početnoj poziciji prema izrazu(4.1). Prikaz konstantnih zadanih parametara prikazano je tablicom 4.

$$v_i^0 = 0.1x_i^0. \tag{4.1}$$

Gdje su:

$\overrightarrow{v_i^0}$ – *i*-ta komponenta vektora brzine nultog koraka

$\overrightarrow{x_i^0}$ – *i*-ta komponenta vektora položaja nultog koraka

Tablica 4. parametri PSO algoritma

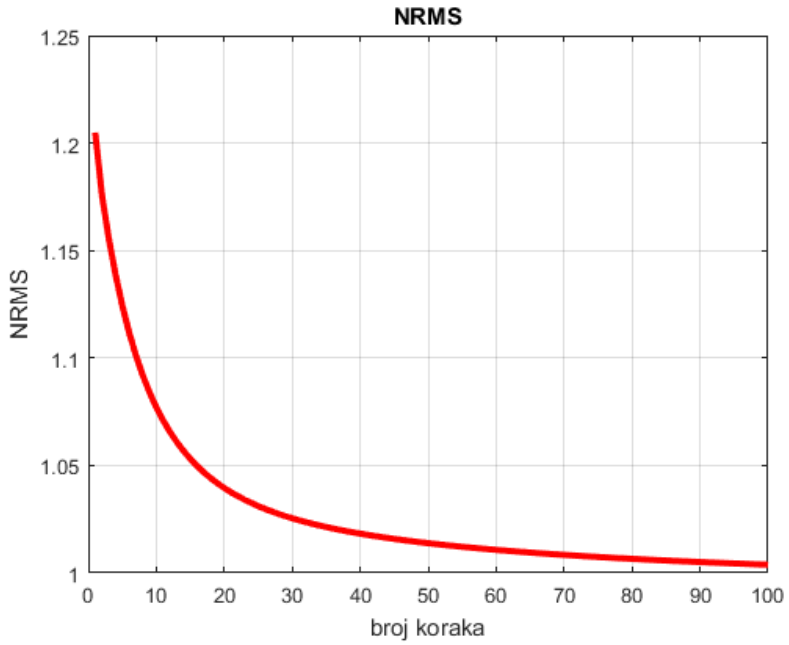
n- broj čestica	20
c_1	0.9
c_2	1.5
c_3	1.5
U_B	10
L_B	-10

4.5. Rezultati

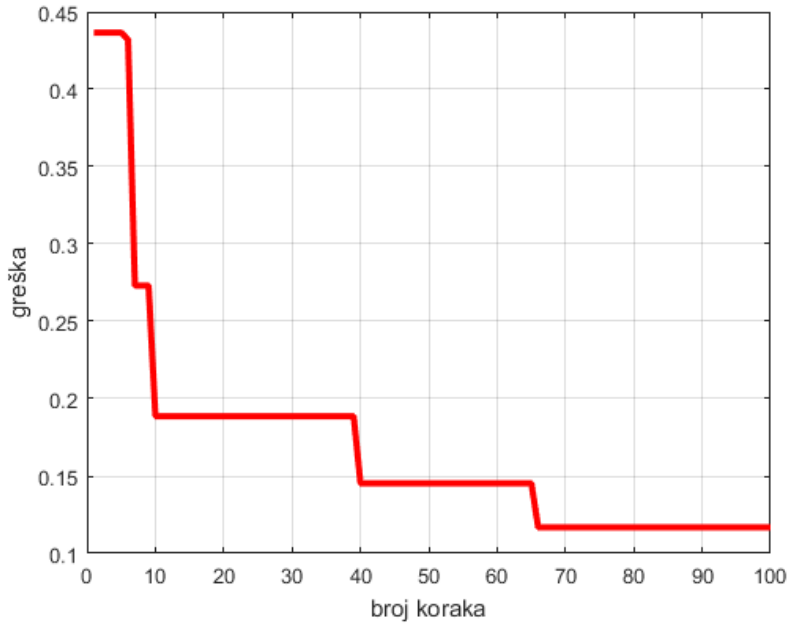
4.5.1. Prikaz rezultata za XOR problem

Rezultate se usporedno prikazuje za nekoliko slučajeva. Ti će se slučajevi razlikovati u broju koraka i broju neurona skrivenog sloja prikazano u tablici 2.

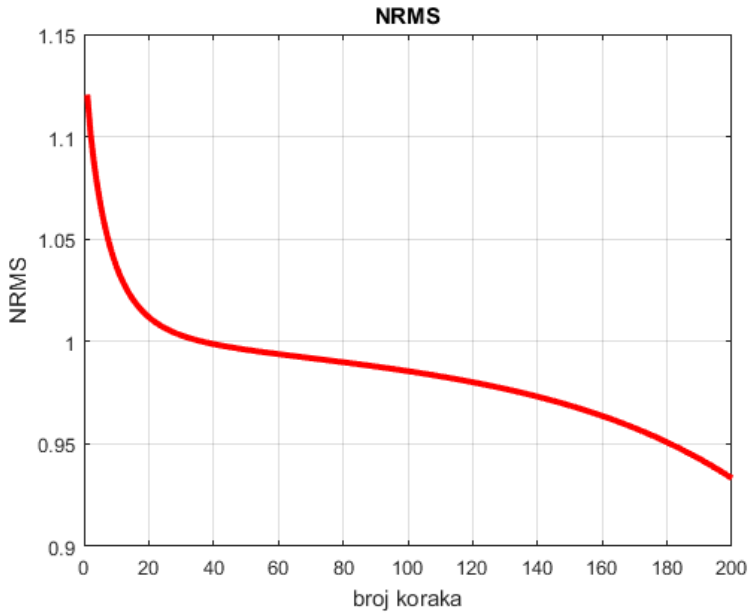
4.5.1.1. Grafički prikaz



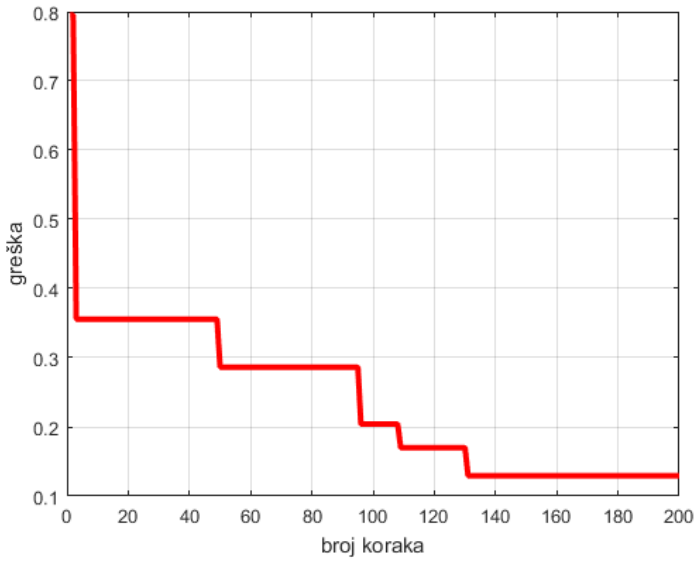
Slika 4.2. Grafički prikaz učenja povratnim prostiranjem pogreške za slučaj 1.



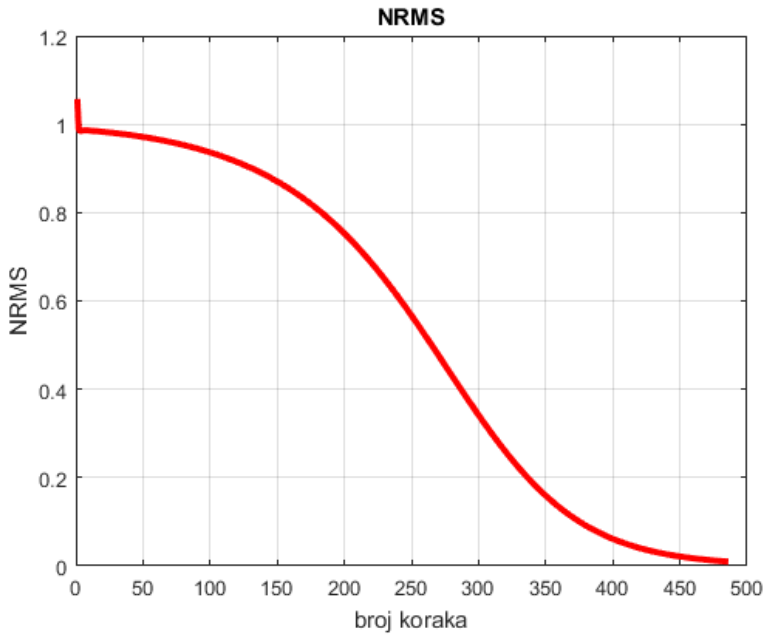
Slika 4.3. Grafički prikaz učenja PSO algoritmom za slučaj 1.



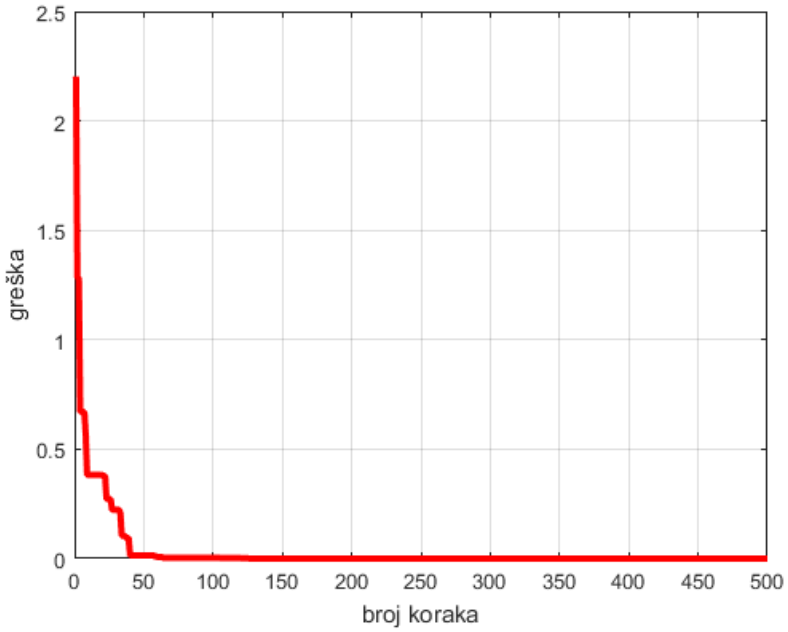
Slika 4.4. Grafički prikaz učenja povratnim prostiranjem pogreške za slučaj 2.



Slika 4.5. Grafički prikaz učenja PSO algoritmom za slučaj 2.



Slika 4.6. Grafički prikaz učenja PSO algoritmom za slučaj 3.



Slika 4.7. Grafički prikaz učenja PSO algoritmom za slučaj 3.

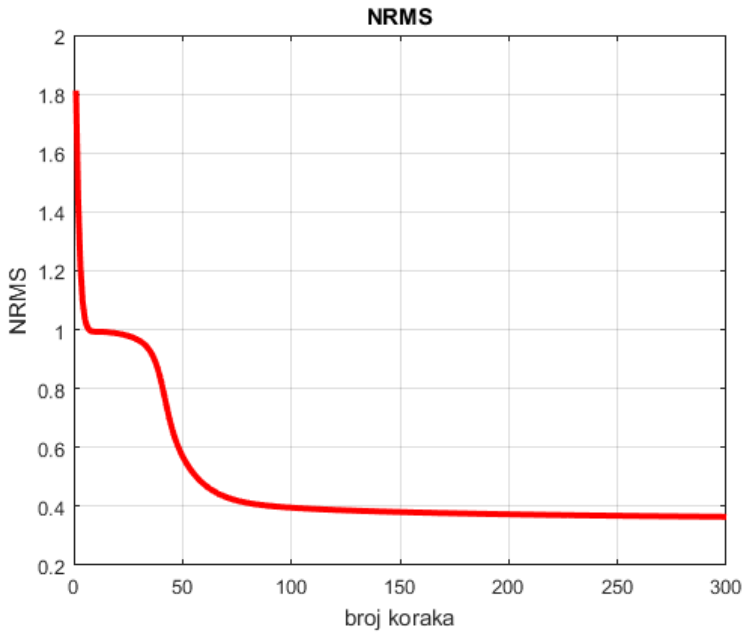
4.5.1.2. Tablični prikaz

Tablica 5. prikaz rezultata rješavanja XOR problema

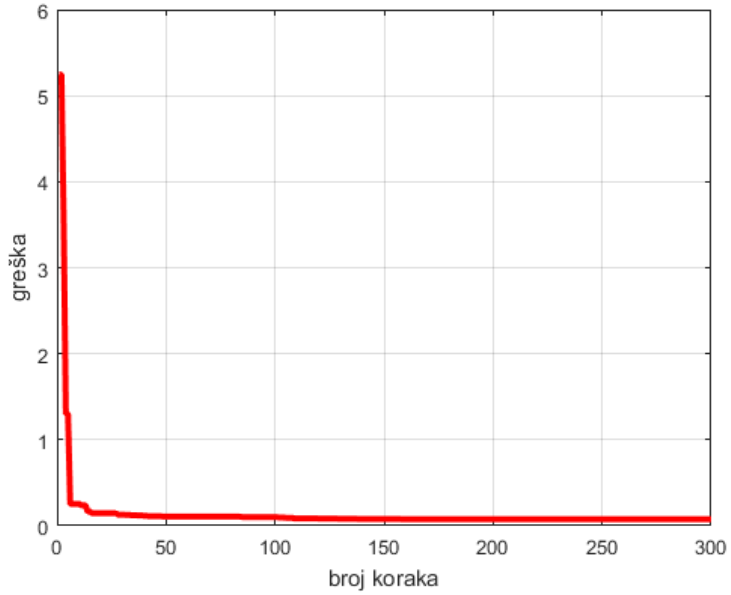
	Greška učenja (%)			Vrijeme učenja (s)		
	1.slučaj	2.slučaj	3.slučaj	1.slučaj	2.slučaj	3.slučaj
EBP	99.72	94.2	0.98	0.29	0.36	0.36
PSO	11.69	12.94	0	43.04	79.45	190.97

4.5.2. Prikaz rezultata za Iris set problem

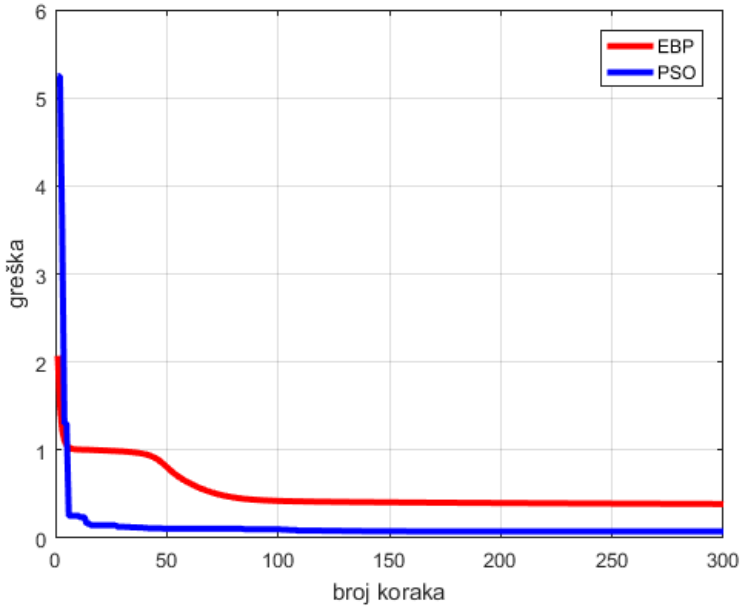
4.5.2.1 Grafički prikaz



Slika 4.8. Grafički prikaz učenja povratnim prostiranjem pogreške za *Iris set* problema



Slika 4.9. Grafički prikaz učenja PSO algoritmom za *Iris set* problema



Slika 4.10. Usporedba odziva greške oba algoritma kod učenja mreže

4.4.2.2. Tablični prikaz

Tablica 6. prikaz rezultata rješavanja Iris set problema

	Greška učenja (%)	Vrijeme učenja (s)
EBP	36.48	0.93
PSO	7.7	138.9

4.6. Analiza rezultata

Za prvi slučaj prvog problema vidljivo je iz tablice (Tablica 5.) kako učenje mreže sa PSO algoritmom već nakon 100 iteracija postiže puno manju grešku, za razliku od učenja povratnim prostiranjem pogreške. To se još bolje vidi iz grafičkog prikaza (Slika 4.2 i slika 4.3). Kod drugog slučaja također postoji velika razlika u vrijednostima greške (Tablica 5.), no vidljivo je da je greška kod povratnog prostiranja mreže tek počela konvergirati prema zadovoljavajućem rješenju (Slika 4.4). Najzad u trećem slučaju oba načina učenja su postigla zadovoljavajuću grešku, uz različite konvergencije (Slika 4.6 i slika 4.7), s time da je PSO algoritmu trebalo puno više vremena (Tablica 5.).

Kod drugog problema (Iris set), vidljivo je kao i kod prvog da učenje mreže pomoću PSO algoritma ponovno brže konvergira prema zadovoljavajućoj vrijednosti greške. Također je vidljivo da u manje koraka dolazi do optimalnog rješenja (Slika 4.8). Povratnim prostiranjem pogreške (Slika 4.9) ne uspijeva se dobiti zadovoljavajuće rješenje sa zadanim parametrima, rješenje zapinje u lokalnom minimumu i ne postiže željenu konvergenciju. Iz usporedbe razultata (Tablica 6.) vidljivo je da je greška učenja puno manja sa PSO algoritmom, no također je vidljivo da je potrebno puno više vremena za provođenje programa. Iz grafa (Slika 4.10), može se dobiti stvarni dojam razlike u konvergenciji greške u učenju.

5. ZAKLJUČAK

Analizom rezultata dva problema učenjem pomoću dva algoritma došlo se do sljedećih zaključaka.

Učenjem pomoću PSO algoritma moguće je postići bržu konvergenciju i manju vrijednost greške. Što znači da nam je potrebno manje koraka učenja za određeni problem, u odnosu na povratno prostiranje pogreške. Što je veći broj parametara (Iris set je imao čak 120 ulaznih parametara u odnosu na XOR koji je imao samo osam), to je veća prednost PSO-a nad povratnim prostiranjem pogreške.

Ipak, velika mana PSO-a je vrijeme učenja, što je velika prednost kod povratnog prostiranja pogreške. Povratno prostiranje pogreške pokazalo se boljim rješenjem kod trećeg slučaja u prvom problemu, kada je uspjela konvergirati grešku u zadovoljavajuće okvire i to za manje od pola sekunde. PSO algoritmu je trebalo više od tri minute. Stoga se nameće zaključak kako kod jednostavnih problema kao što je XOR se ne isplati koristiti PSO algoritam. Za složenije probleme kao što je pokazano za *Iris* set PSO je isplativ, jer unatoč dugom učenju konvergira rješenje i postiže osjetno bolju vrijednost (skoro 30%) pogreške od povratnog prostiranja pogreške.

Ipak potrebno je napomenuti, da ovi zaključci vrijede samo za probleme obrađene u radu, tj. ne mogu se ekstrapolirati na sve probleme koji se koriste u neuronskim mrežama. Isto tako dobro je napomenuti da bi se postigao daljnji napredak u upotrebi PSO algoritma, trebalo bi istraživati PSO algoritam u smjeru dinamički promjenjivih čestica, u kojem već istraživači iz područja modifikacije i unapređenja PSO algoritma idu [19]. Što ustvari znači da bi se promjenom,tj. smanjenjem broja parametara, također optimizirao broj neurona skrivenog sloja, čime bi uz zadržavanje kvalitete odziva minimizirali veličinu mreže.

6. LITERATURA

[1] W.James, The Principles of Psychology, 1890.

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

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

[4] Wikipedia, https://en.wikipedia.org/wiki/File:Neuron_Hand-tuned.svg

[5] P.D. Wasserman, Neural computing: Theory and practice, Van Nostrand Reinhold, New York, 1989.

[6] F. Rosenblatt, Principles of neurodynamics, Spartan Books, New York, 1962.

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

[8] G. Cybenko, Approximations by Superposition of a Sigmoidal Function, Mathematics of Control, Signals and Systems, Vol. 2, pp. 303-314, 1989.

[9] K.Funahashi, On The Approximate Realization of Continuous Mappings by Neural Networks, Neural Networks, Vol. 2, pp. 183-192, 1989.

[10] V.Kecman, D. Majetić, Approximation Capabilities of ANNs as Function of the Numbers of Hidden Layer Neurons and the Type of Activation Function, Techical report TR93-YUSA-01, Massachusetts Institute of Technology, Cambridge, USA, pp. 31-50, 1993.

[11] Kennedy, J. and Eberhart, R. C. (1995) Particle swarm optimization. In Proceedings of the IEEE 1995 International Conference on Neural Networks, pp. 1942-1948. (http://www.engr.iupui.edu/~shi/Coference/psopap4.html)

[12] M.Clerc, Particle Swarm Optimization, 1st edition, London: ISTE, 2006.

[13] D. Ćutić, Programsko ostvarenje algoritama rojeva čestica, diplomski rad, Fakultet elektrotehnike i računarstva, Zagreb, 2014.

[14] M. Omran, Particle Swarm Optimization Methods for Pattern Recognition and Image Processing, Pretoria: University od Pretoria, November 2004.

[15] F. Van den Bergh and A.P. Engelbrecht. A New Locally Convergent Particle Swarm Optimizer. *In Proceedings of the IEEE Conference on Systems, Man, and Cybernetics*, Hammamet, Tunisia, 2002.

[16] D. Shiffman, The Nature Of Code, Chapter 10: Neural Networks. [Internet] Dostupno na: http://natureofcode.com/book/chapter-10-neural-networks/

[17] Fisher, R. A.: The use of multiple measurements in taxonomic problems, Annals of Eugenics 7 (2), str. 179-188, 1936.

[18] M.Vitasović, Optimiranje strukture neuronske mreže s radijalnim baznim funkcijama primjenom algoritmom roja čestica, FSB, Zagreb, 2016.

[19] Zhao, J.; Li, M.; Wang, Z.: Fuzzy Neural Networks Learning by Variable-dimensional Quantum-behaved Particle Swarm Optimization, TELKOMNIKA, Vol. 11, No. 10, str. 6216-6223, 2013.