Neuronske mreže i neuro-fuzzy sustavi

Zadaci za vježbu

Zadatke i rješenja pripremili: Goran Glavaš i Mladen Karan

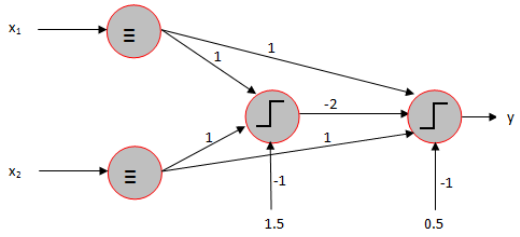
Zadaci

1. Prikažite osnovni model neurona. Što je ukupna pobuda neurona i kako se računa? Što je to prijenosna funkcija? Nacrtajte prijenosnu funkciju praga i logističku prijenosnu funkciju. Koja od tih funkcija je derivabilna? Izvedite izraz za derivaciju te funkcije.

2. Analizirajte rješavanje XOR problema neuronskom mrežom s neuronima s prijenosnom funkcijom praga.

   (a) Neka je zadan osnovni neuron s prijenosnom funkcijom praga (jednostavni perceptron). Može li ovakav perceptron riješiti XOR problem? Objasnite zašto.

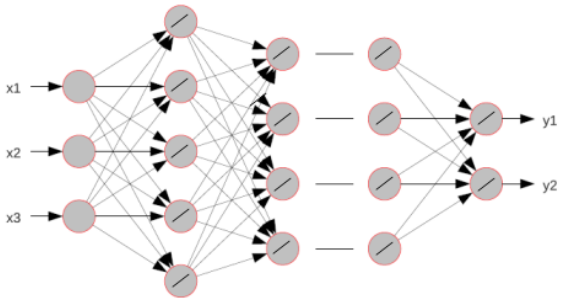
   (b) Zadana je neuronska mreža prikazana na slici 1. Oba neurona imaju



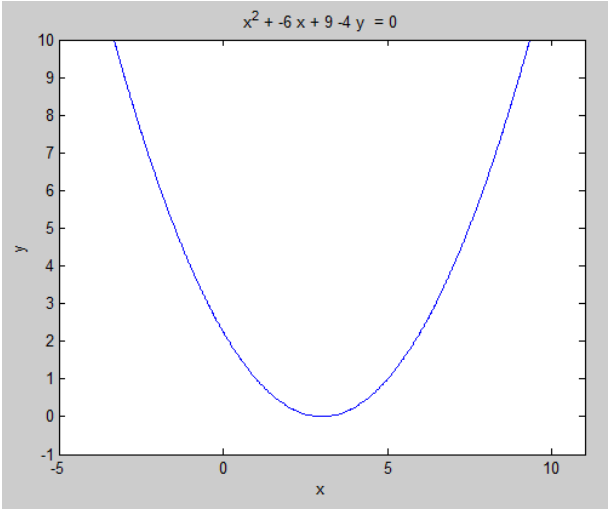
Slika 1: Neuronska mreža s neuronima s pr. funkcijom praga

   prijenosnu funkciju praga. Rješava li ovakva mreža XOR problem? Prikažite grafički decizijske pravce koje određuje ova mreža.

3. Neka je zadana unaprijedna višeslojna neuronska mreža s linearnom funkcijom praga $f(x) = k \cdot x + l$ (slika 2). Pokažite da je takva mreža (bez obzira na strukturu - broj slojeva i broj neurona po slojevima) ekvivalentna mreži bez skrivenih slojeva gdje su ulazi spojeni izravno na izlazne neurone (koji imaju linearnu prijenosnu funkciju).



Slika 2: Unaprijedna neuronska mreža s neuronima s linearnom prijenosnom funkcijom



Slika 3: Graf funkcije $f(x) = \frac{1}{4}(x-3)^2$

4. Zadana je funkcija $f(x) = \frac{1}{4}(x-3)^2$ (Slika 3). Metodom gradijentnog spusta želimo pronaći minimum ove funkcije. Slučajna početna točka iz koje krećemo jest $x_0 = 7$.

   (a) Provedite nekoliko iteracija gradijentnog spusta uz proizvoljno odabranu stopu korekcije ψ. Konvergiraju li vrijednosti dobivene za x u pojedinim iteracijama prema stvarnom minimumu? Alterniraju li predznaci derivacija funkcije u tim točkama? Objasnite relaciju između konvergencije gradijentnog spusta, alternacije predznaka derivacije i iznosa konstante korekcije ψ.

   (b) Odredite graničnu vrijednost konstante korekcije ψ_{gr} pri kojoj metoda

gradijentnog spusta još neće divergirati (ograda divergencije) uz:

i. početno odabranu točku $x_0 = 7$.

ii. početno odabranu točku $x_0 = -1$.

(c) Odredite graničnu vrijednost konstante korekcije ψ_{alt} uz koju vrijednosti dobivene u iteracijama metode gradijentnog spusta još neće alternirati (ograda monotonosti) uz:

i. početno odabranu točku $x_0 = 7$.

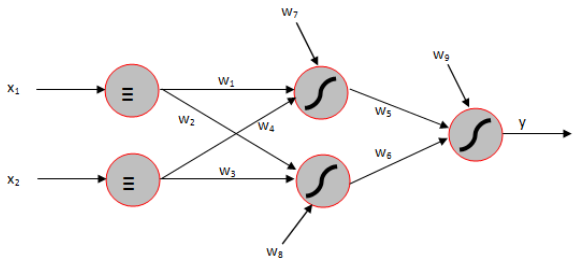
ii. početno odabranu točku $x_0 = -1$.

5. Još jedna često korištena iterativna metoda optimizacije (uz gradijentni spust) jest *Newtonova metoda*. Newtonova metoda pronalazi iterativno (najbližu) nul-točku funkcije pri čemu je iteracija metode definirana s:

$$x^{(i+1)} = x^{(i)} - \frac{f(x)}{f'(x)}.$$

Kako bismo mogli iskoristiti Newtonovu metodu za pronalaženje minimuma funkcije (pomoć: minimum funkcije ujedno je i nul-točka njene derivacije)? Provedite nekoliko iteracija optimizacije Newtonovom metodom za pronalaženje minimuma funkcije $f(x) = \frac{1}{4}(x-3)^2 + 7$ iz prethodnog zadataka uz početnu (slučajno odabranu) točku $x_0 = 6$.

6. Neka je zadana unaprijedna neuronska mreža s jednim skrivenim slojem prikazana na slici 4. Svi neuroni (osim onih u ulaznom sloju) imaju logističku prijenosnu funkciju. Pomoću ove neuronske mreže potrebno je naučiti XOR funkciju (dana su, dakle, 4 primjera za učenje: $(0,0) \to 0$, $(0,1) \to 1$, $(1,0) \to 1$ i $(1,1) \to 0$) Neka su početne vrijednosti težina



Slika 4: Unaprijedna neuronska mreža s jednim skrivenim slojem i logističkom prijenosnom funkcijom neurona

između neurona sve jednake 1. Provedite najmanje 2 iteracije *Backpropagation* algoritma te izračunajte i napišite ažurirane vrijednosti težina (w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8 i w_9) nakon svake iteracije. Koristite faktor korekcije $\eta = 0.1$. Provedite traženi zadatak koristeći sljedeće inačice algoritma:

(a) osnovni *Backpropagation* algoritam (grupno učenje).

(b) stohastički *Backpropagation* algoritam.

(c) stohastički *Backpropagation* algoritam s dodavanjem inercije (uz faktor inercije $\alpha = 0.15$).

7. Prikažite *Winner-takes-all* mrežu na koju se dovode četverodimenzionalni ulazni vektori. Kako se iterativno ažuriraju vrijednosti izlaza pojedinih neurona u ovoj mreži? Neka je na ulaz *Winner-takes-all* mreže doveden ulazni vektor (0.7, 0.6, 0.2, 0.4). Koji će neuron pobijediti i zašto? Prikažite nekoliko iteracija ažuriranja izlaza neurona (koristite $\epsilon = \frac{1}{8}$).

8. Dan je skup od 9 ulaznih vektora iz 2D prostora: $\{(0,0), (0.5,0.5), (1,1), (2,2), (2.5,2.5), (3,3), (4,4), (4.5,4.5), (5,5)\}$. Ako je poznato da primjeri pripadaju trima grupama, kreirajte INSTAR mrežu koja će grupirati primjere u grupe te pronaći reprezentante svake od grupa. Grupiranje provedite na temelju sljedećih mjera sličnosti vektora:

   (a) Sličnost ulaznih vektora s vektorima težina neurona potrebno je mjeriti kosinusom kuta između njih. Pobjednik je onaj neuron koji izbaci van najveći izlaz (uputa: potrebno je prvo normirati ulazne vektore).

   $$cos(\phi) = \frac{\vec{W}\vec{X}}{|\vec{W}||\vec{X}|} \tag{1}$$

   (b) Sličnost ulaznih vektora s vektorima težina neurona potrebno je mjeriti Euklidskom udaljenošću između njih. Pobjednik je onaj neurona koji izbaci van najmanji izlaz.

   $$d(\vec{X}, \vec{W}) = \sqrt{(\vec{X} - \vec{W})(\vec{X} - \vec{W})} \tag{2}$$

   (c) Sličnost ulaznih vektora s vektorima težina neurona potrebno je mjeriti *Jaccardovim* koeficijentom između ta dva vektora. Pobjednik je onaj neuron koji izbaci van najveći izlaz.

   $$Jacc(\vec{X}, \vec{W}) = \frac{\vec{W}\vec{X}}{\sum_{w_i \in \vec{W}} w_i + \sum_{x_i \in \vec{X}} x_i - \vec{W}\vec{X}} \tag{3}$$

   Na temelju kojih mjera sličnosti je moguće uspješno provesti grupiranje, a na temelju kojih nije? Početne vrijednosti vektora težina neurona odaberite slučajno. Objasnite kako početni slučajni odabir vektora težina neurona može negativno utjecati na grupiranje INSTAR mrežom? Kako možemo eliminirati negativan utjecaj nepovoljno odabranih početnih težina?

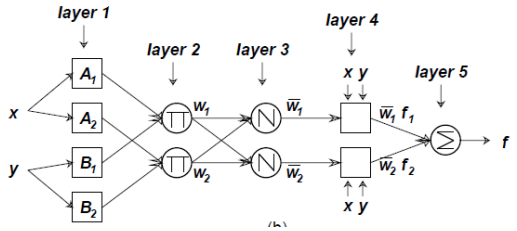
9. Kako mogu biti povezani neuroni (odnosno procesni elementi) u Kohonenovoj samoorganizirajućoj mapi. Koje dvije faze učenja Kohonenovih mapa razlikujemo? Po čemu se razlikuju te dvije faze?

(a) Napišite izraz za postupno smanjivanje stope učenja tijekom inicijalne faze učenja Kohonenove samoorganizirajuće mape. Ako je inicijalna vrijednost stope učenja $\eta_0 = 0.05$, a želimo da konačna vrijednost stope učenja nakon 10000 iteracija inicijalne faze bude $\eta = 0.01$ izračunajte vrijednost konstante K koja određuje tempo smanjivanja stope učenja.

(b) Napišite pravilo učenja za Kohonenovu samoorganizirajuću mapu. Radi li se o čvrstom ili mekom natjecanju?

(c) Dan je skup od 10 ulaznih vektora iz 2D prostora: $\{(0.5, 0.5), (0.5, 1), (1, 0.5), (2, 2.5), (2, 3), (2.5, 2.5), (2.5, 3), (3.5, 3.5), (4, 3.5), (4, 4)\}$. Ako je poznato da primjeri pripadaju trima grupama, kreirajte Kohonenov SOM (kao 1D lanac) koji će grupirati primjere u grupe te pronaći reprezentante svake od grupa. Početna stopa učenja je $\eta_0 = 0.05$. Početne vektore težina procesnih elemenata odaberite slučajno. Provedite 3 iteracije učenja sa svim ulaznim primjerima, smanjujući u svakoj iteraciji stopu učenja za 0.01. Neka je širina susjedstva početno definirana s $\sigma_0 = \frac{1}{\sqrt{2ln2}}$ te neka se po iteraciji smanjuje s $\sigma(n) = \sigma_0 \left(1 - \frac{n}{N_0}\right)$.

10. Zadana je ANFIS mreža koja ostvaruje sustav neizrazitog upravljanja sa zaključivanjem tipa 3 (Takagi-Sugeno-Kang). Arhitektura takve mreže prikazana je na slici 5. Neizraziti sustav upravljanja koji modeliramo



Slika 5: ANFIS mreža koja ostvaruje sustav neizrazitog upravljanja sa TSK zaključivanjem

ovom mrežom sačivanjaju pravila oblika:

$\mathcal{R}_1 : Ako\ x_1\ je\ A_1\ i\ x_2\ je\ B_1\ onda\ y\ je\ z_1$
$\mathcal{R}_2 : Ako\ x_1\ je\ A_2\ i\ x_2\ je\ B_2\ onda\ y\ je\ z_2$
...
$\mathcal{R}_m : Ako\ x_1\ je\ A_m\ i\ x_2\ je\ B_m\ onda\ y\ je\ z_m.$

Konjunkciju antecedenata pravila računajte operatorom umnoška.
Parametri koje je potrebno odrediti učenjem jesu sve vrijednosti z_i te parametri funkcija pripadnosti neizrazitih skupova A_i i B_i. Funkcije pripadnosti svih neizrazitih skupova A_i i B_i definirane su kao normirane

Gaussove funkcije:

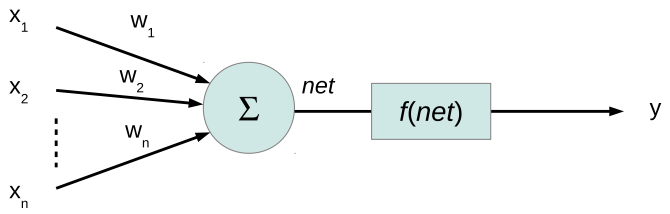
$$f(x) = e^{-\frac{(x-\mu)^2}{2\sigma^2}}. \tag{4}$$

Svaki neizraziti skup A_i i B_i tada ima dva parametra koje je potrebno odrediti učenjem - srednju vrijednost i standardnu devijaciju pripadne Gaussove funkcije. Ukupno, za svako pravilo i postoji, dakle, 5 parametara koje je potrebno odrediti učenjem: z_i, μ_{A_i}, σ_{A_i}, μ_{B_i} i σ_{B_i}.

Odredite pravila za iterativno ažuriranje svih navedenih parametara za metodu stohastičkog gradijentnog spusta (uputa: ključno je odrediti $\frac{\partial E_k}{\partial z_i}$, $\frac{\partial E_k}{\partial \mu_{A_i}}$, $\frac{\partial E_k}{\partial \sigma_{A_i}}$, $\frac{\partial E_k}{\partial \mu_{B_i}}$ i $\frac{\partial E_k}{\partial \sigma_{B_i}}$).

Rješenja

1. Osnovni model neurona prikazan je na slici 6.



Slika 6: Osnovni model neurona

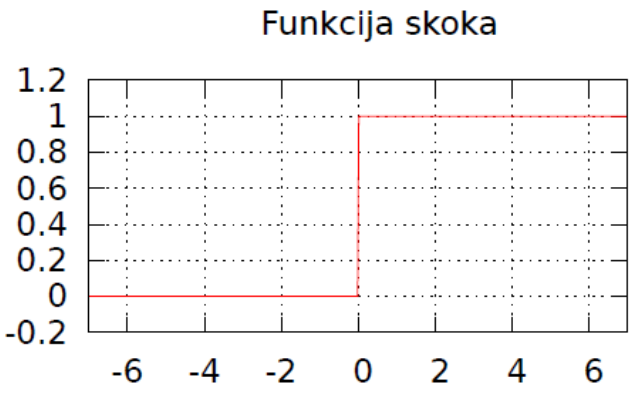
Ukupna pobuda *net* računa se prema izrazu:

$$net = w_1 \cdot x_1 + w_2 \cdot x_2 + \ldots + w_n \cdot x_n - \theta$$

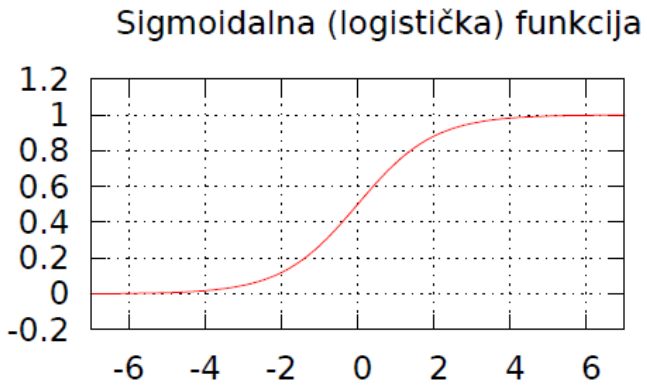
pri čemu θ predstavlja prag paljenja neurona. Kako se iznos praga ne bi morao posebno tretirati u izrazima za *net*, najčešće se definira da neuron ima još jedan "fiktivni" ulaz x_0 na koji je uvijek dovedena vrijednost 1, a prag θ tada se zamjenjuje težinom w_0 čime se dobije konačni izraz:

$$\begin{aligned} net &= w_0 \cdot x_0 + w_1 \cdot x_1 + w_2 \cdot x_2 + \ldots + w_n \cdot x_n \\ &= \textstyle\sum_{i=0}^{n} w_i \cdot x_i \\ &= \vec{w} \cdot \vec{x} \end{aligned}$$

pri čemu je $\vec{x} = (1, x_1, x_2, \ldots, x_n)$ $(n+1)$-dimenzijski vektor koji predstavlja ulaz u neuron a $\vec{w} = (w_0, \ldots, w_n)$ $(n+1)$-dimenzijski vektor težina. Prijenosna funkcija definira na koji način se ulazna pobuda preslikava u izlazni odziv neurona (modelira ponašanje aksona) i u praksi se koristi nekoliko tipičnih prijenosnih funkcija. Na slici 7 prikazana je prijenosna funkcija praga. Funkcija praga prekinuta je u točki $x = 0$ i stoga nije derivabilna. Na slici 8 prikazana je logistička prijenosna funkcija. Logistička



Slika 7: Prijenosna funkcija praga



Slika 8: Logistička prijenosna funkcija

je funkcija neprekidna i derivabilna. Slijedi izvod derivacije logističke

funkcije.

$$\begin{aligned}
h'(x) &= \frac{d(h(x))}{dx} \\
&= \frac{d\left(\frac{1}{1+e^{-x}}\right)}{dx} \\
&= -\frac{1}{(1+e^{-x})^2} \cdot \frac{d\,(1+e^{-x})}{dx} \\
&= -\frac{1}{(1+e^{-x})^2} \cdot e^{-x} \cdot (-1) \\
&= \frac{e^{-x}}{(1+e^{-x})^2} \\
&= \frac{1}{1+e^{-x}} \cdot \frac{e^{-x}}{1+e^{-x}} \\
&= \frac{1}{1+e^{-x}} \cdot \frac{1+e^{-x}-1}{1+e^{-x}} \\
&= \frac{1}{1+e^{-x}} \cdot \left(1 - \frac{1}{1+e^{-x}}\right) \\
&= h(x)\,(1-h(x))
\end{aligned}$$

2. (a) a) Izlaz neurona sa prijenosnom funkcijom praga koji ima i ulaza je dan sa:

$$y = \begin{cases} 1 & \sum_i w_i x_i \geq 0 \\ 0 & \text{inače} \end{cases}$$

Vidimo da on u prostoru značajki odvaja uzorke koristeći linearnu ravninu odluke. Budući da znamo da XOR problem nije linearno odvojiv ovakav neuron ga neće moći riješiti.

(b) b) Promotrimo prvo neuron u skrivenom sloju. Njegov izlaz je dan sa:

$$y_s = \begin{cases} 1 & x_1 + x_2 - 1.5 \geq 0 \\ 0 & \text{inače} \end{cases}$$

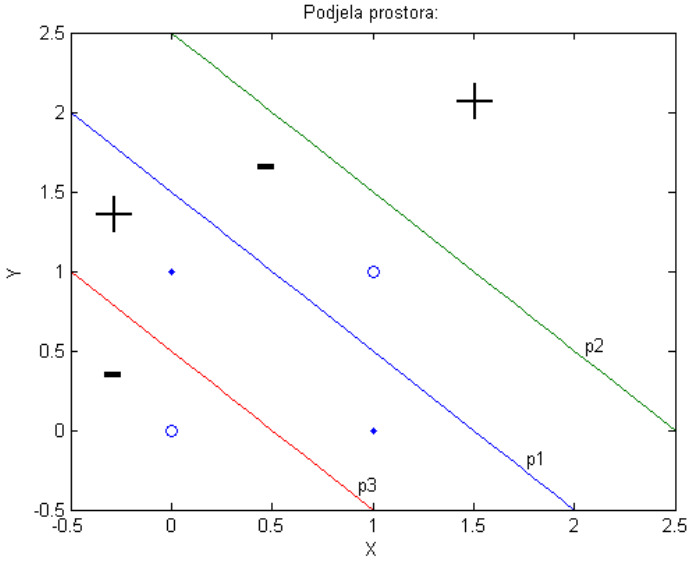
On dijeli ravninu na dvije poluravnine, iznad i ispod pravca $p1$ na slici.

Za izlazni neuron možemo razlikovati dva slučaja:

i. ako je izlaz neurona u skrivenom sloju $y_s = 0$, izlazni neuron daje: $y_{izl} = \begin{cases} 1 & x_1 + x_2 - 0.5 \geq 0 \\ 0 & \text{inače} \end{cases}$
Ovo dijeli poluravninu ispod pravca $p1$ na dva dijela koji su odjeljeni pravcem $p3$.

ii. ako je izlaz neurona u skrivenom sloju $y_s = 1$, izlazni neuron daje: $y_{izl} = \begin{cases} 1 & x_1 + x_2 - 2.5 \geq 0 \\ 0 & \text{inače} \end{cases}$
Ovo dijeli poluravninu iznad pravca $p1$ na dva dijela koji su odjeljeni pravcem $p2$.

Izlaz mreže za pojedina područja prikazan je na slici. Područja gdje je izlaz mreže 1 označena su sa + dok su područja gdje je izlaz mreže 0 označena sa –. Iz slike se vidi da mreža stvarno ispravno rješava XOR problem.



3. Promotrimo neka dva susjedna sloja takve neuronske mreže $k-1$-vi i k-ti. Neka prvi ima prijenosnu funkciju $f_1(x) = k_1 \cdot x + l_1$ a drugi $f_2(x) = k_2 \cdot x + l_2$.

   Izlazi neurona $k-1$-vog sloja dani su sa:

$$y_j = f_1(\sum_i w_{ij} x_i)$$

   gdje su y_j izlazi u sloju $k-1$, w_{ij} težine između pripadajućih neurona $k-2$ i $k-1$ sloja a x_i izlazi neurona $k-2$ sloja. Sumacija ide po svim neuronima $k-2$ sloja.

   Analogno izlazi neurona k-tog sloja dani su sa:

$$z_k = f_2(\sum_j w_{jk} y_j)$$

gdje su z_k izlazi u sloju k, w_{jk} težine između pripadajućih neurona $k-1$ i k sloja a y_j izlazi neurona $k-1$ sloja. Sumacija ide po svim neuronima $k-1$ sloja.

Sada možemo u izraz za z_k uvrstiti izraz za y_j te pojednostaviti izraz koristeći svojstvo linearnosti f_1

$$z_k = f_2(\sum_j w_{jk} f_1(\sum_i w_{ij} x_i)) = f_2(f_1(\sum_j w_{jk} \sum_i w_{ij} x_i))$$
$$= f_2(f_1(\sum_j \sum_i w_{jk} w_{ij} x_i)) = f_2(f_1(\sum_i x_i \sum_j w_{jk} w_{ij}))$$

Uvedimo sada $f^*(x) = f_2(f_1(x))$ i $w^*_{ik} = \sum_j w_{jk} w_{ij}$, gornji izraz je sada ekvivalentan:

$$z_k = f^*(\sum_i w^*_{ik} x_i)$$

Ovime smo pokazali da je izračun rezultata z_k uz zadana dva sloja i prijenosne funkcije f_1 i f_2 identičan slučaju da smo sloj $k-2$ spojili izravno na sloj k koristeći težine w^*_{ik} i prijenosnu funkciju f^*. Zbog toga takva dva sloja možemo spojiti u jedan. Pri tome smo koristili svojstvo linearnosti funkcije f_1. Ako su f_1 i f_2 obje linearne onda je f^* kao kompozicija linearnih funkcija također linearna.

Općenito u mreži sa n slojeva sa linearnim prijenosnim funkcijama moći ćemo kao što je gore pokazano spojiti dva susjedna sloja. Spajanje možemo ponavljati dok ne ostane mreža sa samo jedim slojem (izlaznim) u kojoj su ulazni i izlazni neuroni izravno spojeni.

4. (a) Iz slike se može vidjeti da je minimum funkcije u točki $x^* = 3$. Isti rezultat možemo dobiti izjednačavanjem derivacije funkcije f sa 0. Pogledajmo kako taj isti minimum tražimo postupkom gradijentnog spusta. Trebati će nam derivacija funkcije f:

$$f'(x) = \frac{1}{2} \cdot (x - 3)$$

Sada nam je izraz za korekciju:

$$x_{n+1} = x_n - \frac{\psi}{2} \cdot (x_n - 3)$$

Prve 3 iteracije uz $\psi = 0.5$:

$$x_1 = x_0 - \frac{\psi}{2} \cdot (x_0 - 3) = 7 - \frac{0.5}{2} \cdot (7 - 3) = 6$$
$$x_2 = x_1 - \frac{\psi}{2} \cdot (x_1 - 3) = 6 - \frac{0.5}{2} \cdot (6 - 3) = 5.25$$
$$x_3 = x_2 - \frac{\psi}{2} \cdot (x_2 - 3) = 5.25 - \frac{0.5}{2} \cdot (5.25 - 3) = 4.6875$$

Dobivene vrijednosti se stvarno po iteracijama približavaju minimumu. Predznaci derivacija u ovom slučaju ne alterniraju (u svim iteracijama derivacija ima pozitivan predznak). Odnosi između konvergencije, alterniranja predznaka derivacije i konstante korekcije mogu se opisati tablicom.

ψ	predznaci alterniraju	konvergencija
$\psi \leq \psi_{alt}$	ne	postupak konvergira
$\psi_{alt} < \psi < \psi_{gr}$	da	postupak konvergira
$\psi = \psi_{gr}$	da	postupak oscilira
$\psi > \psi_{gr}$	da	postupak divergira

Gdje su ψ_{alt} i ψ_{gr} ograda monotonosti i ograda divergencije koje ćemo izračunati u b) i c).

(b) Pretpostavimo da je ψ takav da rješenja alterniraju. Postupak još neće divergirati ako vrijedi:

$$x_1 = x_0 - \frac{\psi}{2} \cdot (x_0 - 3)$$
$$x_2 = x_1 - \frac{\psi}{2} \cdot (x_1 - 3) = x_0$$

To znači da se pri povratku na početnu stranu vratimo u točku iz koje smo krenuli (x_0). Uvrštavanjem izraza za x_1 u izraz za x_2 dobivamo:

$$x_2 = x_0 - \frac{\psi}{2} \cdot (x_0 - 3) - \frac{\psi}{2} \cdot (x_0 - \frac{\psi}{2} \cdot (x_0 - 3) - 3)$$

Potom izjednačavanjem x_2 s x_0:

$$x_0 = x_0 - \frac{\psi}{2} \cdot (x_0 - 3) - \frac{\psi}{2} \cdot (x_0 - \frac{\psi}{2} \cdot (x_0 - 3) - 3)$$

Rješavanjem kvadratne jednadžbe uz $x_0 = -1$ dobivamo $\psi_{gr} = 4$. Uz $x_0 = 7$ dobivamo također $\psi_{gr} = 4$. Ograde su jednake jer jer su točke -1 i 7 simetrične s obzirom na optimum a funkcija f je također simetrična.

Primijetimo da smo pri izračunu ψ_{gr} rješavali kvadratnu jednadžbu čije drugo rješenje je bilo 0, ono odgovara slučaju kada se u oba koraka pomaknemo za 0 i tako stvarno završimo tamo od kuda smo krenuli tj. $x_2 = x_0$. Taj slučaj nas ne zanima jer ćemo realno uvijek postaviti ψ na pozitivnu vrijednost.

(c) Ograda monotonosti ψ_{alt} je onaj ψ koji će u jednom koraku prebaciti x točno do optimuma ali ne preko njega (za sve $\psi > \psi_{alt}$ rješenje alternira). Izraz za x_1 je:

$$x_1 = x_0 - \frac{\psi}{2} \cdot (x_0 - 3)$$

Za ψ_{alt} mora vrijediti $x_1 = x^*$ gdje je x^* optimalno rješenje:

$$x^* = x_0 - \frac{\psi}{2} \cdot (x_0 - 3)$$

Optimalno rješenje x^* je 3. Uz $x_0 = -1$ dobivamo $\psi_{alt} = 2$. Uz $x_0 = 7$ dobivamo također $\psi_{alt} = 2$. Razlog istih rješenja je ponovno simetričnost.

5. Zadani postupak traži nul-točku funkcije. Znamo da minimumu funkcije f odgovara nul-točka njene derivacije pa ćemo upravo na f' tražiti nultočku. Izraz za f' je:

$$f'(x) = \frac{1}{2} \cdot (x - 3)$$

Za postupak nam treba i derivacija funkcije na kojoj tražimo nul-točku, to je zapravo druga derivacija od f:

$$f''(x) = \frac{1}{2}$$

Sada možemo napisati korak iteracije:

$$x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)} = x_n - \frac{1}{4} \cdot (x_n - 3)$$

Provedimo prva 3 koraka:

$$x_1 = x_0 - \frac{1}{4} \cdot (x_0 - 3) = 6 - \frac{1}{4} \cdot (6 - 3) = 5.25$$
$$x_2 = x_1 - \frac{1}{4} \cdot (x_1 - 3) = 5.25 - \frac{1}{4} \cdot (5.25 - 3) = 4.6875$$
$$x_3 = x_2 - \frac{1}{4} \cdot (x_2 - 3) = 4.6875 - \frac{1}{4} \cdot (4.6875 - 3) = 4.2656$$

Očekivano, vrijednosti se stvarno približavaju optimalnom rješenju.

6. (a) Odaberimo (slučajno) početne težine kao $w_1 = w_2 = ... = w_9 = 1$. Provedimo sada jednu epohu (prolaz kroz sve uzorke) backpropagation algoritma.

Krenimo računati korekcije sa prvim uzorkom ($x_1 = 0$ i $x_2 = 0$ te željeni izlaz neurona u izlaznom sloju $d_1 = 0$). Prvo nam trebaju izlazi svih neurona za dani ulazni uzorak:

$$y_1^2 = w_1 x_1 + w_4 x_2 - w_7 = 0.2689$$
$$y_2^2 = w_2 x_1 + w_3 x_2 - w_8 = 0.2689$$
$$y_1^3 = w_5 x_1 + w_6 x_2 - w_9 = 0.3865$$

Sada za prvi (i jedini) neuron u izlaznom (trećem) sloju možemo izračunati lokalni gradijent:

$$\delta_1^3 = y_1^3 \cdot (1 - y_1^3) \cdot (d_1 - y_1^3) = -0.0916$$

A pomoću njega i lokalne gradijente neurona u skrivenom sloju:

$$\delta_1^2 = y_1^2 \cdot (1 - y_1^2) \cdot w_5 \cdot \delta_1^3 = -0.0180$$
$$\delta_2^2 = y_2^2 \cdot (1 - y_2^2) \cdot w_6 \cdot \delta_1^3 = -0.0180$$

Primijetimo su članovi $w_5 \cdot \delta_1^3$ i $w_6 \cdot \delta_1^3$ u gornjim izrazima zapravo utežana suma po svim neuronima izlaznog sloja. Ona ima samo jedan član jer je u izlaznom sloju samo jedan neuron. Sada imamo sve što je potrebno za računanje korekcija težina za zadani uzorak:

$$\Delta w_1(1) = \delta_1^2 \cdot x_1 = 0$$
$$\Delta w_4(1) = \delta_1^2 \cdot x_2 = 0$$
$$\Delta w_7(1) = \delta_1^2 \cdot (-1) = 0.0180$$

$$\Delta w_2(1) = \delta_2^2 \cdot x_1 = 0 \qquad \Delta w_5(1) = \delta_1^3 \cdot y_1^2 = -0.0246$$
$$\Delta w_3(1) = \delta_2^2 \cdot x_2 = 0 \qquad \Delta w_6(1) = \delta_1^3 \cdot y_2^2 = -0.0246$$
$$\Delta w_8(1) = \delta_2^2 \cdot (-1) = 0.0180 \qquad \Delta w_9(1) = \delta_1^3 \cdot (-1) = 0.0916$$

Time smo obradili prvi uzorak, težine još nećemo mijenjati već identičnim postupkom obrađujemo preostale uzorke. Za drugi uzorak ($x_1 = 0$ i $x_2 = 1$ te $d_1 = 1$):

$$\Delta w_1(2) = \delta_1^2 \cdot x_1 = 0$$
$$\Delta w_4(2) = \delta_1^2 \cdot x_2 = 0.0313$$
$$\Delta w_7(2) = \delta_1^2 \cdot (-1) = -0.0313$$

$$\Delta w_2(2) = \delta_2^2 \cdot x_1 = 0 \qquad \Delta w_5(2) = \delta_1^3 \cdot y_1^2 = 0.0625$$
$$\Delta w_3(2) = \delta_2^2 \cdot x_2 = 0.0313 \qquad \Delta w_6(2) = \delta_1^3 \cdot y_2^2 = 0.0625$$
$$\Delta w_8(2) = \delta_2^2 \cdot (-1) = -0.0313 \qquad \Delta w_9(2) = \delta_1^3 \cdot (-1) = -0.125$$

Za treći uzorak ($x_1 = 1$ i $x_2 = 0$ te $d_1 = 1$):

$$\Delta w_1(3) = \delta_1^2 \cdot x_1 = 0.0313$$
$$\Delta w_4(3) = \delta_1^2 \cdot x_2 = 0$$
$$\Delta w_7(3) = \delta_1^2 \cdot (-1) = -0.0313$$

$$\Delta w_2(3) = \delta_2^2 \cdot x_1 = 0.0313 \qquad \Delta w_5(3) = \delta_1^3 \cdot y_1^2 = 0.0625$$
$$\Delta w_3(3) = \delta_2^2 \cdot x_2 = 0 \qquad \Delta w_6(3) = \delta_1^3 \cdot y_2^2 = 0.0625$$
$$\Delta w_8(3) = \delta_2^2 \cdot (-1) = -0.0313 \qquad \Delta w_9(3) = \delta_1^3 \cdot (-1) = -0.125$$

Za četvrti uzorak ($x_1 = 1$ i $x_2 = 1$ te $d_1 = 0$):

$$\Delta w_1(4) = \delta_1^2 \cdot x_1 = -0.0286$$
$$\Delta w_4(4) = \delta_1^2 \cdot x_2 = -0.0286$$
$$\Delta w_7(4) = \delta_1^2 \cdot (-1) = 0.0286$$

$$\Delta w_2(4) = \delta_2^2 \cdot x_1 = -0.0286 \qquad \Delta w_5(4) = \delta_1^3 \cdot y_1^2 = -0.1063$$
$$\Delta w_3(4) = \delta_2^2 \cdot x_2 = -0.0286 \qquad \Delta w_6(4) = \delta_1^3 \cdot y_2^2 = -0.1063$$
$$\Delta w_8(4) = \delta_2^2 \cdot (-1) = 0.0286 \qquad \Delta w_9(4) = \delta_1^3 \cdot (-1) = 0.1455$$

Sada kada smo prošli sve uzorke dobijemo akumuliranu korekciju zbrajajući korekcije za svaki pojedini uzorak i nju dodamo početnoj težini. Izraz za modifikaciju težina je:

$$w_i \leftarrow w_i + \eta \sum_{s=1}^{4} \Delta w_i(s)$$

Gdje $\Delta w_i(s)$ označava korekciju dobivenu za uzorak s a sumacija ide po svim uzorcima za učenje. Uvrštavanjem konkretnih vrijednosti u gornji izraz dobivaju se nove vrijednosti težina:

$$w_1 = 1.0003 \qquad w_2 = 1.0003 \qquad w_3 = 1.0003$$
$$w_4 = 1.0003 \qquad w_5 = 0.9994 \qquad w_6 = 0.9994$$
$$w_7 = 0.9984 \qquad w_8 = 0.9984 \qquad w_9 = 0.9987$$

Time je završena jedna iteracija batch verzije backpropagation algoritma. Sljedeću iteraciju radimo identičnim postupkom (koji opet prolazi sve uzorke) ali koristeći dobivene ažurirane težine.

(b) Odaberimo opet početne težine $w_1 = w_2 = ... = w_9 = 1$ i provedimo 2 iteracije algoritma.

Krenimo računati korekcije sa prvim uzorkom ($x_1 = 0$ i $x_2 = 0$ te željeni izlaz neurona u izlaznom sloju $d_1 = 0$). Postupak je identičan postupku za prvi uzorak u a) dijelu zadatka. Dobivamo i iste iznose korekcija:

$$\Delta w_1 = \delta_1^2 \cdot x_1 = 0$$
$$\Delta w_4 = \delta_1^2 \cdot x_2 = 0$$
$$\Delta w_7 = \delta_1^2 \cdot (-1) = 0.0180$$

$$\Delta w_2 = \delta_2^2 \cdot x_1 = 0 \qquad \Delta w_5 = \delta_1^3 \cdot y_1^2 = -0.0246$$
$$\Delta w_3 = \delta_2^2 \cdot x_2 = 0 \qquad \Delta w_6 = \delta_1^3 \cdot y_2^2 = -0.0246$$
$$\Delta w_8 = \delta_2^2 \cdot (-1) = 0.0180 \qquad \Delta w_9 = \delta_1^3 \cdot (-1) = 0.0916$$

Time smo obradili prvi uzorak. Za razliku od batch verzije algoritma sada ne čekamo prolazak kroz sve uzorke već težine odmah ažuriramo prema izrazu:

$$w_i \leftarrow w_i + \eta \Delta w_i$$

Ažurirane težine su:

$$w_1 = 1 \qquad w_2 = 1 \qquad w_3 = 1$$
$$w_4 = 1 \qquad w_5 = 0.9975 \qquad w_6 = 0.9975$$
$$w_7 = 1.0018 \qquad w_8 = 1.0018 \qquad w_9 = 1.0092$$

Sada krećemo u obradu drugog uzorka ($x_1 = 0$ i $x_2 = 1$ te $d_1 = 1$) identično kao u a) zadatku no ovaj put koristeći ažurirane težine. Izlazi neurona su sada:

$$y_1^2 = w_1 x_1 + w_4 x_2 - w_7 = 0.4995$$
$$y_2^2 = w_2 x_1 + w_3 x_2 - w_8 = 0.4995$$
$$y_1^3 = w_5 x_1 + w_6 x_2 - w_9 = 0.4969$$

Lokalni gradijenti su:

$$\delta_1^3 = y_1^3 \cdot (1 - y_1^3) \cdot (d_1 - y_1^3) = 0.1258$$
$$\delta_1^2 = y_1^2 \cdot (1 - y_1^2) \cdot w_5 \cdot \delta_1^3 = 0.3137$$
$$\delta_2^2 = y_2^2 \cdot (1 - y_2^2) \cdot w_6 \cdot \delta_1^3 = 0.3137$$

Te korekcije za drugi uzorak:

$$\Delta w_1 = \delta_1^2 \cdot x_1 = 0$$
$$\Delta w_4 = \delta_1^2 \cdot x_2 = 0.0314$$
$$\Delta w_7 = \delta_1^2 \cdot (-1) = -0.0314$$

$$\Delta w_2 = \delta_2^2 \cdot x_1 = 0 \qquad \Delta w_5 = \delta_1^3 \cdot y_1^2 = 0.6283$$
$$\Delta w_3 = \delta_2^2 \cdot x_2 = 0.0314 \qquad \Delta w_6 = \delta_1^3 \cdot y_2^2 = 0.6283$$
$$\Delta w_8 = \delta_2^2 \cdot (-1) = -0.0314 \qquad \Delta w_9 = \delta_1^3 \cdot (-1) = -0,1258$$

Primjetimo da korekcije nisu identične onima koje smo dobili u obradi drugog uzorka u zadatku a). To je zato što radimo s već ažuriranim težinama. Nakon obrade drugog uzorka opet možemo ažurirati težine.

$$w_1 = 1 \qquad w_2 = 1 \qquad w_3 = 1.0031$$
$$w_4 = 1.0031 \qquad w_5 = 1.0038 \qquad w_6 = 1.0038$$
$$w_7 = 0.9987 \qquad w_8 = 0.9987 \qquad w_9 = 0.9966$$

Sa novim ažuriranim težinama dalje nastavljamo postupak sljedećom iteracijom koja obrađuje treći uzorak.

(c) Postupak je identičan zadatku b) no ažuriranje se radi prema izrazu:

$$w_i \leftarrow w_i + \eta \Delta w_i + \alpha \Delta w_i' \tag{5}$$

Gdje je sa $\Delta w_i'$ označena korekcija u prethodnom koraku.

Odaberimo opet početne težine $w_1 = w_2 = ... = w_9 = 1$. Obradom prvog uzorka dobivamo iste korekcije kao u b):

$$\Delta w_1 = \delta_1^2 \cdot x_1 = 0$$
$$\Delta w_4 = \delta_1^2 \cdot x_2 = 0$$
$$\Delta w_7 = \delta_1^2 \cdot (-1) = 0.0180$$

$$\Delta w_2 = \delta_2^2 \cdot x_1 = 0 \qquad \Delta w_5 = \delta_1^3 \cdot y_1^2 = -0.0246$$
$$\Delta w_3 = \delta_2^2 \cdot x_2 = 0 \qquad \Delta w_6 = \delta_1^3 \cdot y_2^2 = -0.0246$$
$$\Delta w_8 = \delta_2^2 \cdot (-1) = 0.0180 \qquad \Delta w_9 = \delta_1^3 \cdot (-1) = 0.0916$$

Budući da je u prvom koraku prethodna korekcija 0 dobivamo i iste težine kao u b)

$$w_1 = 1 \qquad w_2 = 1 \qquad w_3 = 1$$
$$w_4 = 1 \qquad w_5 = 0.9975 \qquad w_6 = 0.9975$$
$$w_7 = 1.0018 \qquad w_8 = 1.0018 \qquad w_9 = 1.0092$$

Obradom drugog uzorka također dobivamo iste iznose korekcija kao u b):

$$\Delta w_1 = \delta_1^2 \cdot x_1 = 0$$
$$\Delta w_4 = \delta_1^2 \cdot x_2 = 0.0314$$
$$\Delta w_7 = \delta_1^2 \cdot (-1) = -0.0314$$

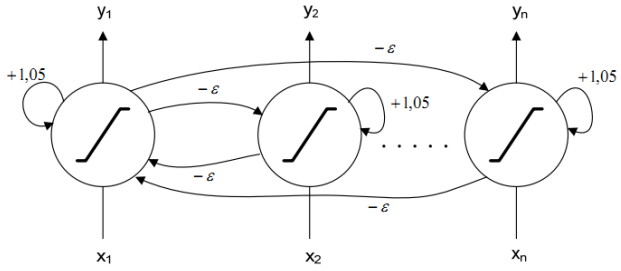
$$\Delta w_2 = \delta_2^2 \cdot x_1 = 0 \qquad \Delta w_5 = \delta_1^3 \cdot y_1^2 = 0.6283$$
$$\Delta w_3 = \delta_2^2 \cdot x_2 = 0.0314 \qquad \Delta w_6 = \delta_1^3 \cdot y_2^2 = 0.6283$$
$$\Delta w_8 = \delta_2^2 \cdot (-1) = -0.0314 \qquad \Delta w_9 = \delta_1^3 \cdot (-1) = -0,1258$$

Budući da su prethodne korekcije iz prvog koraka sada različite od 0 uvrštavanjem u izraz (5) uz $\alpha = 0.15$ dobivamo težine:

$$w_1 = 1 \qquad w_2 = 1 \qquad w_3 = 1.0031$$
$$w_4 = 1.0031 \qquad w_5 = 1.0001 \qquad w_6 = 1.0001$$
$$w_7 = 1.0014 \qquad w_8 = 1.0014 \qquad w_9 = 1.0103$$

One se razlikuju od težina dobivenih obradom drugog uzorka u b) zadatku zbog toga što uzimaju u obzir i korekciju iz prethodnog koraka. Postupak možemo nastaviti analogno b) zadatku, sljedećom iteracijom u kojoj se obrađuje treći uzorak.

7. Općenita Winner-takes-all mreža je prikazana slikom:



Vrijednosti se računaju iterativno u diskretnim trenucima. U prvoj iteraciji se mreži predoče ulazni uzorci koji se potom maknu.

Kroz iteracije onaj neuron koji je bio pobuđen sa najvećim ulazom će nakon prijelazne pojave prigušiti sve ostale neurone. Nakon nekog vremena izlaz se stabilizira, svi neuroni imaju izlaz 0 osim onaj koji odgovara najvećem ulazu.

Za zadani ulazni vektor (0.7, 0.6, 0.2, 0.4) samo prvi neuron će na kraju imati izlaz 1 jer je prvi element ulaznog vektora najveći. Izrazi za računanje izlaza u sljedećoj (n+1) iteraciji su:

$$
\begin{aligned}
y_1(n+1) &= \phi(1.05y_1 - \epsilon y_2(n) - \epsilon y_3(n) - \epsilon y_4(n) + x_1) \\
y_2(n+1) &= \phi(1.05y_2 - \epsilon y_1(n) - \epsilon y_3(n) - \epsilon y_4(n) + x_2) \\
y_3(n+1) &= \phi(1.05y_3 - \epsilon y_1(n) - \epsilon y_2(n) - \epsilon y_4(n) + x_3) \\
y_4(n+1) &= \phi(1.05y_4 - \epsilon y_1(n) - \epsilon y_2(n) - \epsilon y_3(n) + x_4)
\end{aligned}
$$

Gdje je ϕ prijenosna funkcija neurona. Pretpostavimo da je ona definirana sa:

$$
\phi(x) = \begin{cases} 0 & x < 0 \\ x & 0 \leq x \leq 1 \\ 1 & x > 1 \end{cases}
$$

U prvom koraku mreži predočimo ulazne uzorke $x_1 ... x_4$, pri tome uzimamo da su izlazi u prošlom koraku $y_1(0) = \cdots = y_4(0) = 0$. Uvrštavanjem ovih vrijednosti u gornje izraze slijedi:

$$y_1(1) = 0.7 \qquad y_2(1) = 0.6$$
$$x_3(1) = 0.2 \qquad y_4(1) = 0.4$$

U drugom koraku, maknemo ulazne uzorke tj. postavimo $x_1 = \cdots = x_4 = 0$. Kao izlaze prošlog koraka koristimo vrijednosti $y_1(1) \ldots y_4(1)$. Uvrštavanjem ovih vrijednosti izlazi u drugom koraku su kako slijedi:

$$y_1(2) = 0.585 \qquad y_2(2) = 0.4675$$
$$y_2(2) = 0 \qquad y_4(2) = 0.2325$$

Analogno koraku 2 se računaju izlazi za sve sljedeće korake, napišimo još korak 3:

$$y_1(3) = 0.52675 \qquad y_2(3) = 0.3887$$
$$y_3(3) = 0 \qquad y_4(3) = 0.1126$$

Na kraju u 26. koraku sustav se stabilizira, u njemu i svim koracima nakon njega dobivamo očekivani rezultat:

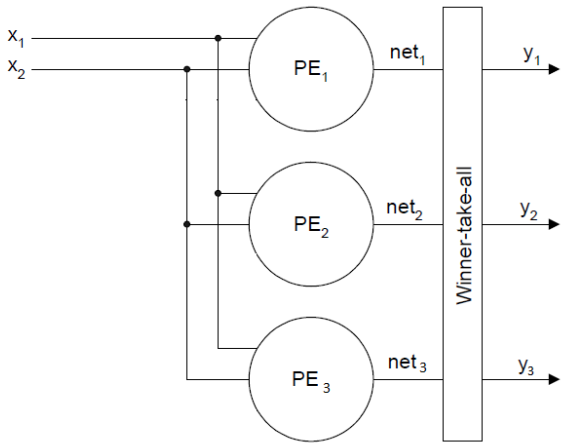
$$y_1(26) = 1 \qquad y_2(26) = 0$$
$$y_3(26) = 0 \qquad y_4(26) = 0$$

8. Budući da je poznato da podaci pripadaju trima grupama, INSTAR mreža imat će 3 neurona (slika 9). Neka su odabrane sljedeći inicijalni vektori težina tih neurona: $(0.5, 2)$, $(2, 3.5)$ i $(4, 6)$. Zadani ulazni primjeri $\{(0.25, 0.25), (0.5, 0.5), (1, 1), (2, 2), (2.5, 2.5), (3, 3), (4, 4), (4.5, 4.5), (5, 5)\}$ svi se nalaze na istom pravcu $y = x$ (slika 10).

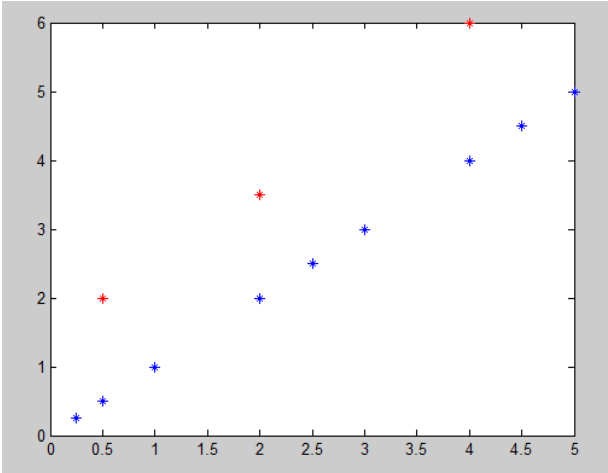
   (a) Kosinusna mjera sličnosti dvaju vektora ne uzima u obzir veličinu (apsolutni iznos duljine vektora) već je bitan samo kut među vektorima. Analitički, kosinus kuta između dva vektora računamo kao skalarni umnožak njihovih normiranih inačica:

$$cos(\phi) = \frac{\vec{W}\vec{X}}{|\vec{W}||\vec{X}|} \tag{6}$$

   Kako se svi ulazni primjeri nalaze na istom pravcu $y = x$, normiranjem se oni svi svode na isti normirani vektor $(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$. Normirani vektori težina neurona iznose redom $(\frac{1}{\sqrt{17}}, \frac{4}{\sqrt{17}})$, $(\frac{4}{\sqrt{65}}, \frac{7}{\sqrt{65}})$ i



Slika 9: INSTAR mreža s tri neurona i dvodimenzionalnim ulaznim primjerima



Slika 10: Ulazni primjeri i inicijalni vektori težina neurona INSTAR mreže

$(\frac{2}{\sqrt{13}}, \frac{3}{\sqrt{13}})$. Budući da su svi ulazni vektori predstavljeni istim normiranim vektorom, jasno je kako će upravo jedan od početnih vektora težina neurona biti najbliži tom normiranom vektoru (tj. zatvarati najmanji kut s tim vektorom). Vektor neurona 3 $(\frac{2}{\sqrt{13}}, \frac{3}{\sqrt{13}})$ zatvara najmanji kut sa svim ulaznim vektorima i stoga će upravo neuron 3 biti uvijek pobjednik i samo će on ažurirati svoje težine. Ažuriranje težina provodi se prema modificiranom Hebbovom pravilu:

$$w_j(n+1) = w_{ij}(n) + \eta \cdot y_j(n) \cdot (x_j(n) - w_{ij}(n)).$$

Uz $\eta = 0.05$ prvih nekoliko iteracija daje sljedeće ažurirane vrijednosti vektora težina pobjedničkog neurona 3:

$w_3(0) = (4, 6)$
$w_3(1) = (3.81, 5.71)$
$w_3(2) = (3.64, 5.45)$
$w_3(3) = (3.51, 5.22)$
...
$w_3(900) = (2.74, 2.74)$

Očekivano, vektor težina pobjedničkog neurona na kraju dolazi na pravac $y = x$ budući da je kut između takvog vektora i svih ulaznih vektora jednak 0 (tj. postiže se maksimalna vrijednost kosinusa kuta). Zanimljivo je primijetiti da vektor težina pobjedničkog neurona neće iskonvergirati u jednu konačnu točku, već se neprekinutim dovođenjem ulaznih vektora na mrežu vektor težina neurona 3 "šeće" sljedećim skupom od 9 točaka (upravo onoliko koliko je ulaznih primjera): $(2.61, 2.61) \rightarrow (2.51, 2.51) \rightarrow (2.43, 2.43) \rightarrow (2.41, 2.41) \rightarrow (2.42, 2.42) \rightarrow (2.45, 2.45) \rightarrow (2.52, 2.52) \rightarrow (2.62, 2.62) \rightarrow (2.74, 2.74)$ Sve se ove točke, naravno, nalaze na pravcu $y = x$ i normiraju u isti vektor $(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$ kao i svi ulazni primjeri. Grupiranje temeljem kosinusne mjere sličnosti vektora neće, dakle, prepoznati tri postojeće grupe u ulaznim podacima niti će pronaći adekvatne reprezentante za svaku od grupa.

(b) Korištenje euklidske udaljenosti kao mjere sličnosti dvaju vektora otklanja prethodno uočeni nedostatak kosinusne mjere. Za računanje euklidske udaljenosti nije potrebno normirati ulazne vektore (kao ni vektore težina neurona). Pogledajmo prvih nekoliko iteracija algoritma (uz $\eta = 0.05$):

$w_1(0) = (0.5, 2), w_2(0) = (2, 3.5), w_3(0) = (4, 6)$
1. za ulaz $(0.25, 0.25)$ pobjednik je neuron 1
ažurirane vrijednosti težina neurona 1 $w_1(1) = (0.4875, 1.91)$
2. za ulaz $(0.5, 0.5)$ pobjednik je neuron 1
ažurirane vrijednosti težina neurona 1 $w_1(2) = (0.4881, 1.84)$
3. za ulaz $(1, 1)$ pobjednik je neuron 1
ažurirane vrijednosti težina neurona 1 $w_1(3) = (0.5137, 1.799)$
4. za ulaz $(2, 2)$ pobjednik je neuron 1
ažurirane vrijednosti težina neurona 1 $w_1(4) = (0.588, 1.809)$
5. za ulaz $(2.5, 2.5)$ pobjednik je neuron 2
ažurirane vrijednosti težina neurona 2 $w_2(5) = (2.025, 3.45)$
6. za ulaz $(3, 3)$ pobjednik je neuron 2
ažurirane vrijednosti težina neurona 2 $w_2(6) = (2.074, 3.4275)$
7. za ulaz $(4, 4)$ pobjednik je neuron 3
ažurirane vrijednosti težina neurona 3 $w_3(7) = (4, 5.9)$

8. za ulaz $(4.5, 4.5)$ pobjednik je neuron 3
ažurirane vrijednosti težina neurona 3 $w_3(8) = (4.025, 5.83)$
9. za ulaz $(5, 5)$ pobjednik je neuron 3
ažurirane vrijednosti težina neurona 3 $w_3(9) = (4.074, 5.7885)$
...

Nakon nekoliko epoha (treniranja mreže svim ulaznim primjerima) vektori težina svih triju neurona počnu alternirati, svaki između 3 vrijednosti. Očekivano, te su vrijednosti opet na pravcu $y = x$. Vrijednosti među kojima nakon "konvergencije" alterniraju težine neurona su sljedeće:

$w_1(k-2) = (0.596, 0.596)$
$w_1(k-1) = (0.579, 0.579)$
$w_1(k) = (0.575, 0.575)$

$w_2(k-2) = (2.517, 2.517)$
$w_2(k-1) = (2.4912, 2.4912)$
$w_2(k) = (2.4917, 2.4917)$

$w_3(k-2) = (4.517, 4.517)$
$w_3(k-1) = (4.4912, 4.4912)$
$w_3(k) = (4.4917, 4.4917)$

Vidljivo je da vrijednosti težina neurona alterniraju oko vrijednosti koje bi intuitivno bile očekivane kao reprezentanti pojedinih grupa - $(0.6, 0.6)$ za neuron 1, $(2.5, 2.5)$ za neuron 2 te $(4.5, 4.5)$. Bilo koju od tri alternirajuće vrijednosti za svaki od neurona možemo uzeti kao reprezentanta pojedine grupe. Vidljivo je da je grupiranje temeljem euklidske udaljenosti, za razliku od grupiranja kosinusnom mjerom, ispravno grupiralo ulazne primjere i dobro odredilo reprezentante razreda.

(c) Mjerenje sličnosti vektora Jaccardovim koeficijentom sličnije je kosinusnoj mjeri sličnosti nego euklidskoj udaljenosti kao mjeri sličnosti dvaju vektora. Iako ne normira vektora, Jaccardov koeficijent također ima veću vrijednost što su vektori sličniji (ali nažalost nema tako jasnu geometrijsku interpretaciju kao kosinusna sličnost). Računajući nekoliko početnih iteracja ažuriranja vektora težina neurona na temelju Jaccard sličnosti vektora dobiva se(uz $\eta = 0.05$):

$w_1(0) = (0.5, 2), w_2(0) = (2, 3.5), w_3(0) = (4, 6)$
1. za ulaz $(0.25, 0.25)$ pobjednik je neuron 3
ažurirane vrijednosti težina neurona 1 $w_3(1) = (3.81, 5.71)$
2. za ulaz $(0.5, 0.5)$ pobjednik je neuron 3
ažurirane vrijednosti težina neurona 1 $w_3(2) = (3.64, 5.45)$

3. za ulaz $(1,1)$ pobjednik je neuron 3
ažurirane vrijednosti težina neurona 1 $w_3(3) = (3.51, 5.23)$
4. za ulaz $(2,2)$ pobjednik je neuron 1
ažurirane vrijednosti težina neurona 1 $w_1(4) = (0.575, 2)$
5. za ulaz $(2.5, 2.5)$ pobjednik je neuron 1
ažurirane vrijednosti težina neurona 2 $w_1(5) = (0.67, 2.025)$
6. za ulaz $(3,3)$ pobjednik je neuron 2
ažurirane vrijednosti težina neurona 2 $w_1(6) = (0.787, 2.073)$
7. za ulaz $(4,4)$ pobjednik je neuron 3
ažurirane vrijednosti težina neurona 3 $w_3(7) = (3.54, 5.17)$
8. za ulaz $(4.5, 4.5)$ pobjednik je neuron 3
ažurirane vrijednosti težina neurona 3 $w_3(8) = (3.59, 5.134)$
9. za ulaz $(5,5)$ pobjednik je neuron 3
ažurirane vrijednosti težina neurona 3 $w_3(9) = (3.66, 5.127)$
...

Nakon "konvergencije" konačne vrijednosti težina neurona 1,2 i 3 redom će biti $(1.994, 2.0008)$, $(2.48, 2.48)$ i $(2.79, 2.79)$. Vidljivo je da su i u ovom slučaju vektori težina (približno) na pravcu $y = x$. INSTAR mreža temeljena na Jaccard sličnosti između vektora, ipak, nije uspjela prepoznati grupe podataka niti pronaći adekvatne reprezentante.

Slučajni početni odabir vektora težina neurona može biti nepovoljan na način da je neki vektor težina nekog neurona (tj. procesnog elementa) predaleko od bilo koje grupe podataka. Takav procesni element nikada neće biti pobjednik i nikad neće ažurirati svoje težine. Takvi neuroni su mrtvi neuroni. Postupak učenja u tom će u tom slučaju uzrokovati da jedan neuron bude određen kao predstavnik više od jednog razreda. Najjednostavnije rješenje jest dopustiti svakom od neurona da obavi ažuriranje težinskih faktora. Pri tome se kod pobjednika korekcija množi samo s faktorom η i obavlja prema uobičajenom izrazu napisanom prethodno u ovom zadatku. Ostali neuroni ažuriraju svoje težine "prigušeno" odnosno faktor η skalira se faktorom prigušenja λ za koji vrijedi $\lambda \ll \eta$. Ovakva modifikacija uzrokovat će da i mrtvi neuroni malo po malo izađu iz područja u kojem nikada nisu pobjednici i počnu povremeno pobjeđivati čime će im se omogućiti specijalizacija za određeni dio ulaznog prostora.
Drugo rješenje (spomenuto na predavanjima) zasniva se na konceptu "savjesti". Svaki neuron procjenjuje postotak slučajeva u kojima je pobjednik. Što je taj iznos veći, kod neurona se javlja grižnja savjesti i zbog toga mu se udaljenosti do uzoraka koji se dovode na ulaz mreže prividno povećavaju te ih on doživljava većima no što stvarno jesu. S druge pak strane, neuroni koji malo pobjeđuju, žele pobjeđivati više pa im se udaljenosti do uzoraka koji se dovode na ulaz mreže prividno smanjuju. Ovo će dovesti do toga da nakon određenog vremena do tada mrtvi neuron počne pobjeđivati jer će njegova prividno umanjena udaljenost do ulaznog uzorka postati manja od prividno uvećanih udaljenosti do svih ostalih neurona, i takav će neuron

početi korigirati svoje težinske faktore.

9. Kohonenove samoorganizirajuće mape su jednoslojne linearne mreže u kojima se procesni elementi tipično povezuju u 1D lanac ili 2D polje. 1D lanac je jednodimenzijska topologija, u kojoj su neuroni međusobno povezani u lanac pa svaki neuron osim krajnjih ima dva direktna susjeda (jednog lijevo i jednog desno). 2D polje je dvodimenzijska rešetka, pri čemu svaki neuron (osim rubnih) ima četiri direktna susjeda (jednog iznad, jednog ispod, jednog lijevo i jednog desno).

   Algoritam učenja Kohonenove samoorganizirajuće mape sastoji se od dvije faze: inicijalne faze i faze finog podešavanja. *Inicijalna faza* osigurava grubo podešavanje samoorganizirajuće mape i služi za stvaranje grubog razmještaja neurona po prostoru primjera za učenje. U toj fazi kreće se od relativno velikog susjedstva (primjerice pola širine mape ili više) te se susjedstvo postupno smanjuje prema malom susjedstvu od sveka nekoliko (ili čak jednog) neurona. Stopa učenja također treba krenuti s relativno velikim iznosom i potom se postupno smanjivati (npr. od 0.5 do 0.01). Tijekom *faze finog podešavanja* (koja tipičo ima znatno veći broj iteracija od inicijalne faze) susjedstvo se drži vrlo malenim (ili se čak ograničava samo na pobjednikčki neuron). Stopa učenja obično je malena (tipično se nastavlja na vrijednost na kojoj je stala inicijalna faza, npr. 0.01) i postupno se još više smanjuje.

   (a) Tijekom inicijalne faze učenja Kohonenove samoorganizirajuće mape stopa učenja iterativno se ažurira prema sljedećem izrazu:

   $$\eta(n) = \eta_0 \left(1 - \frac{n}{N_0 + K}\right)$$

   Jednostavnim algebarskim transformacijama dobivamo izraz za vrijednost konstante K koja određuje tempo smanjivanja stope učenja:

   $$K = \frac{n \cdot \eta_0}{\eta_0 - \eta(n)} - N_0$$

   Uvrštavanjem zadanih vrijednosti $\eta_0 = 0.05$, $n = N_0 = 10000$, $\eta(10000) = 0.01$ lako je izračunati traženu vrijednost za K:

   $$\begin{aligned} K &= \frac{10000 \cdot 0.05}{0.05 - 0.01} - 10000 \\ &= 2500 \end{aligned}$$

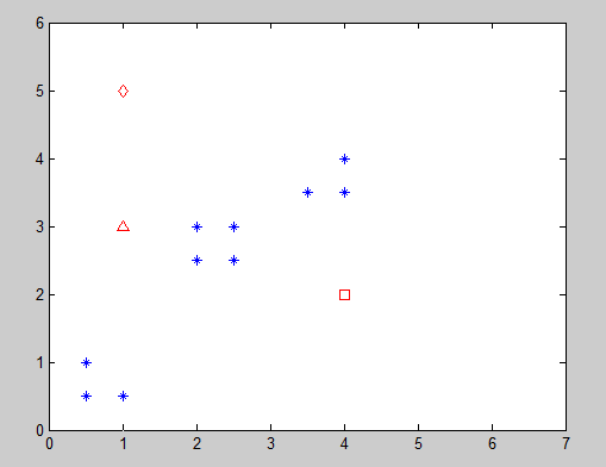
   (b) Pravilo ažuriranja težina koje se koristi kod Kohonenove samoorganizirajuće mape glasi:

   $$\vec{w}_i \leftarrow \vec{w}_i + \eta(n) \cdot \Lambda_{i,i^*}(n) \cdot (\vec{X} - \vec{w}_i)$$

   gdje je $\vec{w}_i$ vektor težina i-tog neurona, $\eta(n)$ iznos stope učenja u n-toj iteraciji, $\vec{X}$ ulazni primjer za učenje, i^* indeks neurona pobjednika

(dakle neurona koji je najbliži ulaznom uzorku za učenje) te $\Lambda_{i,i^*}(n)$ iznos vrijednosti funkcije susjednosti između neurona i i pobjedničkog neurona i^* u n-toj iteraciji. Budući da osim pobjedničkog neurona uče i njegovi susjedni neuroni (iako u manjoj mjeri) ovdje se radi o mekom natjecanju. Ipak, tijekom faze finog podešavanja često se uvodi čvrsto natjecanje na način da se funkcija susjednosti Λ definira tako da poprimi vrijednost 0 za sve neurone u susjedstvu pobjedničkog neurona. Na taj način težine ažurira samo pobjednički neuron. Dakle, tijekom inicijalne faze Kohonenove samoorganizirajuće mape radi se o mekom natjecanju dok se kod faze finog podešavanja može primijeniti i čvrsto i meko natjecanje.

(c) Neka su odabrani sljedeći početni vektori težina neurona: $(1, 3)$, $(1, 5)$, $(4, 2)$. Ulazni primjeri i početni vektori težina neurona prikazani su na slici 11. Neka inicijalna faza traje $N_0 = 100$ iteracija, s time



Slika 11: Ulazni primjeri i početni vektora težina neurona Kohonenove samoorganizirajuće mape

da vrijednost stope učenja η ne može u nijednoj iteraciji biti manja od 0.01. Prilikom svakog dovođenja nekog ulaznog primjera na SOM kao pobjednik se odabire onaj neuron kojemu je vektor težina najbliži ulaznom primjeru. Sličnost se mjeri euklidskom udaljenošću između ulaznog primjera i vektora težina neurona.

Ažuriranje težina neurona (pobjedničkog, ali i neurona u njegovoj okolini) obavlja se na temelju prethodno spomenutog izraza:

$$\vec{w}_i \leftarrow \vec{w}_i + \eta(n) \cdot \Lambda_{i,i^*}(n) \cdot (\vec{X} - \vec{w}_i)$$

Ažuriranje stope učenja u svakoj iteraciji iznosi $\eta(n) = \eta_0 - n \cdot 0.01$ za $n \in \{1, 2, 3, 4\}$ odnosno $\eta(n) = 0.01$ za $n \geq 5$ uz $\eta_0 = 0.05$.

Ažuriranje širine susjedstva u pojedinoj iteraciji računa se pomoću $\sigma(n) = \sigma_0 \left(1 - \frac{n}{N_0}\right)$ uz $\sigma_0 = \frac{1}{\sqrt{2ln2}}$ i $N_0 = 100$.

Iteracija 1 - $\eta = 0.05$, $\sigma = \frac{1}{\sqrt{2ln2}} = 0.849$
za ulaz $(0.5, 0.5)$ pobjednik je neuron 1.
Ažuriranje težina neurona 1: $(1, 3) \rightarrow (0.975, 2.875)$;
Ažuriranje težina neurona 2: $(1, 5) \rightarrow (0.9875, 4.8875)$;
Ažuriranje težina neurona 3: $(4, 2) \rightarrow (3, 989, 1, 995)$;

za ulaz $(0.5, 1)$ pobjednik je neuron 1.
Ažuriranje težina neurona 1: $(0.975, 2.875) \rightarrow (0.951, 2.781)$;
Ažuriranje težina neurona 2: $(0.9875, 4.8875) \rightarrow (0.975, 4.79)$;
Ažuriranje težina neurona 3: $(3, 989, 1, 995) \rightarrow (3.978, 1.992)$;

za ulaz $(1, 0.5)$ pobjednik je neuron 1.
Ažuriranje težina neurona 1: $(0.951, 2.781) \rightarrow (0.9537, 2.67)$;
Ažuriranje težina neurona 2: $(0.975, 4.79) \rightarrow (0.976, 4.683)$;
Ažuriranje težina neurona 3: $(3.978, 1.992) \rightarrow (3.969, 1.987)$;

za ulaz $(2, 2.5)$ pobjednik je neuron 1.
Ažuriranje težina neurona 1: $(0.9537, 2.67) \rightarrow (1.006, 2.66)$;
Ažuriranje težina neurona 2: $(0.976, 4.683) \rightarrow (1.001, 4.628)$;
Ažuriranje težina neurona 3: $(3.969, 1.987) \rightarrow (3.963, 1.989)$;

za ulaz $(2, 3)$ pobjednik je neuron 1.
Ažuriranje težina neurona 1: $(1.006, 2.66) \rightarrow (1.056, 2.676)$;
Ažuriranje težina neurona 2: $(1.001, 4.628) \rightarrow (1.026, 4.587)$;
Ažuriranje težina neurona 3: $(3.963, 1.989) \rightarrow (3.957, 1.992)$;

za ulaz $(2.5, 2.5)$ pobjednik je neuron 1.
Ažuriranje težina neurona 1: $(1.056, 2.676) \rightarrow (1.128, 2.667)$;
Ažuriranje težina neurona 2: $(1.026, 4.587) \rightarrow (1.063, 4.535)$;
Ažuriranje težina neurona 3: $(3.957, 1.992) \rightarrow (3.952, 1.994)$;

za ulaz $(2.5, 3)$ pobjednik je neuron 1.
Ažuriranje težina neurona 1: $(1.128, 2.667) \rightarrow (1.196, 2.683)$;
Ažuriranje težina neurona 2: $(1.063, 4.535) \rightarrow (1.099, 4.497)$;
Ažuriranje težina neurona 3: $(3.952, 1.994) \rightarrow (3.947, 1.997)$;

za ulaz $(3.5, 3.5)$ pobjednik je neuron 3.
Ažuriranje težina neurona 1: $(1.196, 2.683) \rightarrow (1.204, 2.686)$;
Ažuriranje težina neurona 2: $(1.099, 4.497) \rightarrow (1.159, 4.472)$;
Ažuriranje težina neurona 3: $(3.947, 1.997) \rightarrow (3.925, 2.072)$;

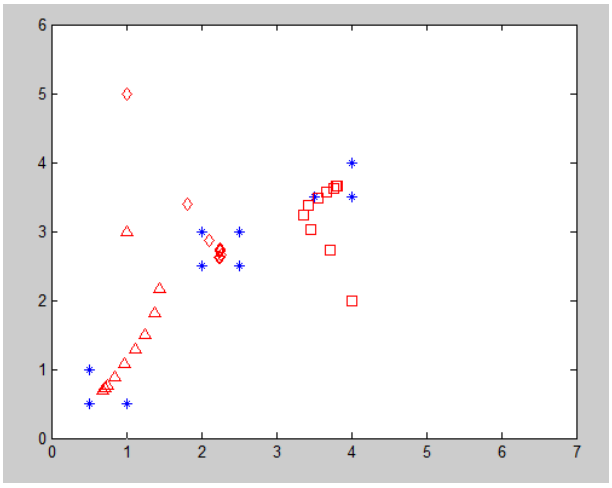
za ulaz $(4, 3.5)$ pobjednik je neuron 3.
Ažuriranje težina neurona 1: $(1.204, 2.686) \rightarrow (1.212, 2.689)$;
Ažuriranje težina neurona 2: $(1.159, 4.472) \rightarrow (1.23, 4.448)$;
Ažuriranje težina neurona 3: $(3.925, 2.072) \rightarrow (3.929, 2.144)$;

za ulaz $(4, 4)$ pobjednik je neuron 3.
Ažuriranje težina neurona 1: $(1.212, 2.689) \rightarrow (1.221, 2.692)$;
Ažuriranje težina neurona 2: $(1.23, 4.448) \rightarrow (1.299, 4.437)$;
Ažuriranje težina neurona 3: $(3.929, 2.144) \rightarrow (3.932, 2.236)$;

Iteracija 2 - $\eta = 0.04$, $\sigma = 0.845$
Analogno kao u prethodnoj iteraciji ažuriraju se težine svih neurona za svaki ulazni primjer.

Iteracija 3 - $\eta = 0.03$, $\sigma = 0.84$
Analogno kao u prethodne dvije iteracije ažuriraju se težine svih neurona za svaki ulazni primjer.
...

Promotrimo kretanje vektora težina neurona kroz nešto veći broj iteracija. Na slici 12 vidimo pozicije vektora težina sva tri neurona nakon 10, 25, 50, 75, 100, 125, 150, 175 i 200 iteracija. Vidljivo je



Slika 12: Konvergencija vektora težina neurona Kohononenove samoorganizirajuće mape

kako je Kohonenova samoorganizirajuća mapa uspjela prepoznati tri grupe prisutne u podacima te jako dobro odabrati reprezentante tih grupa.

10. Neka nam je dostupan niz od N uzoraka za učenje: $\{(x_1,y_1), \ldots, (x_N,y_N)\}$. Potrebno je izvesti *online* verziju algoritma koju ćemo temeljiti na gradijentnom spustu. Neka je izlaz sustava neizrazitog upravljanja za predočeni k-ti uzorak označen s o_k. Funkciju pogreške za taj uzorak tada možemo definirati na sljedeći način:

$$E_k = \frac{1}{2}(y_k - o_k)^2$$

Ako parametre ažuriramo u skladu s algoritmom gradijentnog spusta, za ažuriranje nekog proizvoljnog parametra ψ koristi se izraz:

$$\psi(t+1) = \psi(t) - \eta\frac{\partial E_k}{\partial \psi}$$

Naš je zadatak stoga utvrditi parcijalne derivacije funkcije E_k po svim parametrima o kojima ovisi rad promatranog sustava ANFIS. ($\frac{\partial E_k}{\partial z_i}$, $\frac{\partial E_k}{\partial \mu_{A_i}}$, $\frac{\partial E_k}{\partial \sigma_{A_i}}$, $\frac{\partial E_k}{\partial \mu_{B_i}}$ i $\frac{\partial E_k}{\partial \sigma_{B_i}}$).

Izlaz sustava neizrazitog upravljanja definiran je kao težinska suma:

$$o = \frac{\sum_{i=1}^{m} \alpha_i \beta_i z_i}{\sum_{i=1}^{m} \alpha_i \beta_i}$$

pri čemu je α_i jakost paljenja prvog člana konjunkcije antecedenata i-tog pravila, a β_i jakost paljenja drugog člana konjunkcije antecedenata i-tog pravila. U promatranom slučaju α_i jednak je:

$$\alpha_i = e^{-\frac{(x-\mu_{A_i})^2}{2\sigma_{A_i}^2}}.$$

, a β_i jednak je:

$$\beta_i = e^{-\frac{(x-\mu_{B_i})^2}{2\sigma_{B_i}^2}}.$$

Za potrebe izračuna parcijalne derivacije funkcije pogreške po parametru z_i poslužit ćemo se pravilom ulančavanja:

$$\frac{\partial E_k}{\partial z_i} = \frac{\partial E_k}{\partial o_k}\frac{\partial o_k}{\partial z_i}.$$

Svaku od pojedinih parcijalnih derivacija sada je jednostavno izračunati:

$$\begin{aligned}\frac{\partial E_k}{\partial o_k} &= \frac{\partial}{\partial z_i}\left(\frac{1}{2}(y_k - o_k)^2\right) \\ &= -(y_k - o_k)\end{aligned}$$

$$\begin{aligned}\frac{\partial o_k}{\partial z_i} &= \frac{\partial}{\partial z_i}\left(\frac{\sum_{j=1}^{m}\alpha_j\beta_j z_j}{\sum_{j=1}^{m}\alpha_j\beta_j}\right) \\ &= \frac{\alpha_i\beta_i}{\sum_{j=1}^{m}\alpha_j\beta_j}\end{aligned}$$

Uvrštavanjem izvedenoga u početni izraz dolazi se do konačnog izraza za traženu parcijalnu derivaciju:

$$\frac{\partial E_k}{\partial z_i} = -(y_k - o_k)\frac{\alpha_i \beta_i}{\sum_{j=1}^{m} \alpha_j \beta_j}$$

iz čega lako dobivamo pravilo ažuriranja za algoritam stohastičkog gradijentnog spusta za parametar z_i:

$$z_i(t+1) = z_i(t) + \eta \cdot (y_k - o_k)\frac{\alpha_i \beta_i}{\sum_{j=1}^{m} \alpha_j \beta_j}$$

.

Slično, korištenjem ulančavanja računamo i parcijalnu derivaciju pogreške po μ_{A_i}, $i \in 1, ..., m$ (gdje je m broj pravila neizrazitog upravljanja):

$$\frac{\partial E_k}{\partial \mu_{A_i}} = \frac{\partial E_k}{\partial o_k}\frac{\partial o_k}{\partial \alpha_i}\frac{\partial \alpha_i}{\partial \mu_{A_i}}.$$

Parcijalnu derivaciju $\frac{\partial E_k}{\partial o_k}$ već smo izračunali. Izračunajmo sada i ostale dvije parcijalne derivacije:

$$\begin{aligned}
\frac{\partial o_k}{\partial \alpha_i} &= \frac{\partial}{\partial \alpha_i}\left(\frac{\sum_{j=1}^{m} \alpha_i \beta_i z_i}{\sum_{j=1}^{m} \alpha_i \beta_i}\right) \\
&= \frac{1}{\left(\sum_{j=1}^{m} \alpha_i \beta_i\right)^2} \cdot \left(\frac{\partial}{\partial \alpha_i}\left(\sum_{j=1}^{m} \alpha_i \beta_i z_i\right) \cdot \sum_{j=1}^{m} \alpha_i \beta_i - \frac{\partial}{\partial \alpha_i}\left(\sum_{j=1}^{m} \alpha_i \beta_i\right) \cdot \sum_{j=1}^{m} \alpha_i \beta_i z_i\right) \\
&= \frac{\beta_i z_i \cdot \sum_{j=1}^{m} \alpha_i \beta_i - \beta_i \cdot \sum_{j=1}^{m} \alpha_i \beta_i z_i}{\left(\sum_{j=1}^{m} \alpha_i \beta_i\right)^2}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial \alpha_i}{\partial \mu_{A_i}} &= \frac{\partial}{\partial \mu_{A_i}}\left(e^{-\frac{(x_1 - \mu_{A_i})^2}{2\sigma^2}}\right) \\
&= \left(e^{-\frac{(x_1 - \mu_{A_i})^2}{2\sigma^2}}\right) \cdot \frac{\partial}{\partial \mu_{A_i}}\left(-\frac{(x_1 - \mu_{A_i})^2}{2\sigma^2}\right) \\
&= \left(e^{-\frac{(x_1 - \mu_{A_i})^2}{2\sigma^2}}\right) \cdot \frac{x_1 - \mu_{A_i}}{\sigma^2} \\
&= \frac{\alpha_i}{\sigma^2} \cdot (x_1 - \mu_{A_i})
\end{aligned}$$

Uvrštavanjem izvedenoga u početni izraz dolazi se do konačnog izraza za traženu parcijalnu derivaciju:

$$\frac{\partial E_k}{\partial \mu_{A_i}} = -(y_k - o_k) \cdot \frac{\beta_i z_i \cdot \sum_{j=1}^{m} \alpha_i \beta_i - \beta_i \cdot \sum_{j=1}^{m} \alpha_i \beta_i z_i}{\left(\sum_{j=1}^{m} \alpha_i \beta_i\right)^2} \cdot \frac{\alpha_i}{\sigma^2} \cdot (x_1 - \mu_{A_i})$$

iz čega lako dobivamo pravilo ažuriranja za algoritam stohastičkog gradijentnog spusta za parametar z_i:

$$\mu_{A_i}(t+1) = \mu_{A_i}(t)+\eta\cdot(y_k-o_k)\cdot\frac{\beta_i z_i\cdot\sum_{j=1}^{m}\alpha_i\beta_i-\beta_i\cdot\sum_{j=1}^{m}\alpha_i\beta_i z_i}{\left(\sum_{j=1}^{m}\alpha_i\beta_i\right)^2}\cdot\frac{\alpha_i}{\sigma^2}\cdot(x_1-\mu_{A_i})\,.$$

Slično, korištenjem ulančavanja računamo i parcijalnu derivaciju pogreške po σ_{A_i}, $i\in 1,...,m$:

$$\frac{\partial E_k}{\partial \mu_{A_i}} = \frac{\partial E_k}{\partial o_k}\frac{\partial o_k}{\partial \alpha_i}\frac{\partial \alpha_i}{\partial \sigma_{A_i}}.$$

Prve dvije parcijalne derivacije u ovom izrazu prethodno su već izračunate. Izračunajmo još $\frac{\partial \alpha_i}{\partial \sigma A_i}$:

$$\begin{aligned}
\frac{\partial \alpha_i}{\partial \sigma_{A_i}} &= \frac{\partial}{\partial \sigma_{A_i}}\left(e^{-\frac{(x_1-\mu_{A_i})^2}{2\sigma^2}}\right)\\
&= \left(e^{-\frac{(x_1-\mu_{A_i})^2}{2\sigma^2}}\right)\cdot\frac{\partial}{\partial \sigma_{A_i}}\left(-\frac{(x_1-\mu_{A_i})^2}{2\sigma^2}\right)\\
&= \left(e^{-\frac{(x_1-\mu_{A_i})^2}{2\sigma^2}}\right)\cdot\frac{-(x_1-\mu_{A_i})^2}{2}\cdot\frac{-1}{\sigma^4}\cdot 2\sigma\\
&= \alpha_i\cdot\frac{(x_1-\mu_{A_i})^2}{\sigma^3}
\end{aligned}$$

Uvrštavanjem izvedenoga u početni izraz dolazi se do konačnog izraza za traženu parcijalnu derivaciju:

$$\frac{\partial E_k}{\partial \sigma_{A_i}} = -(y_k-o_k)\cdot\alpha_i\cdot\frac{\beta_i z_i\cdot\sum_{j=1}^{m}\alpha_i\beta_i-\beta_i\cdot\sum_{j=1}^{m}\alpha_i\beta_i z_i}{\left(\sum_{j=1}^{m}\alpha_i\beta_i\right)^2}\cdot\frac{(x_1-\mu_{A_i})^2}{\sigma^3}$$

iz čega lako dobivamo pravilo ažuriranja za algoritam stohastičkog gradijentnog spusta za parametar z_i:

$$\sigma_{A_i}(t+1) = \sigma_{A_i}(t)+\eta\cdot\alpha_i(y_k-o_k)\cdot\frac{\beta_i z_i\cdot\sum_{j=1}^{m}\alpha_i\beta_i-\beta_i\cdot\sum_{j=1}^{m}\alpha_i\beta_i z_i}{\left(\sum_{j=1}^{m}\alpha_i\beta_i\right)^2}\cdot\frac{(x_1-\mu_{A_i})^2}{\sigma^3}.$$

Izrazi za pravila ažuriranja parametara μ_{B_i} i σ_{B_i} izvode se potpuno analogno pravilima za μ_{A_i} i σ_{A_i}, uzimajući u obzir simetričnost uloga α_i i β_i u izrazu za o_k. Pravila koja se dobiju glase:

$$\mu_{B_i}(t+1) = \mu_{B_i}(t)+\eta\cdot(y_k-o_k)\cdot\frac{\alpha_i z_i\cdot\sum_{j=1}^{m}\alpha_i\beta_i-\alpha_i\cdot\sum_{j=1}^{m}\alpha_i\beta_i z_i}{\left(\sum_{j=1}^{m}\alpha_i\beta_i\right)^2}\cdot\frac{\beta_i}{\sigma^2}\cdot(x_2-\mu_{B_i})\,.$$

$$\sigma_{B_i}(t+1) = \sigma_{B_i}(t)+\eta\cdot\beta_i(y_k-o_k)\cdot\frac{\alpha_i z_i\cdot\sum_{j=1}^{m}\alpha_i\beta_i-\alpha_i\cdot\sum_{j=1}^{m}\alpha_i\beta_i z_i}{\left(\sum_{j=1}^{m}\alpha_i\beta_i\right)^2}\cdot\frac{(x_2-\mu_{B_i})^2}{\sigma^3}.$$