NEURONSKE MREŽE

Do sada obrađeni klasifikatori su u potpunosti određeni ako su poznate statističke osobine uzoraka, njihova srednja vrijednost i varijansa. Estimacija statističkih parametara se vrši na osnovu skupa uzoraka. Skup uzoraka (*training patterns*) za koje se zna kojoj klasi pripadaju i koji se koriste za određivanje ovih parametara se naziva skup uzoraka za obučavanje (*training set (paterns)*), a sam proces se naziva *obučavanje* (*učenje*).

Proces učenja u prethodnim primjerima se svodi samo na određivanje parametara, nakon čega struktura klasifikatora ostaje fiksna.

U praksi su često statističke osobine klasa nepoznate i nemoguće ih je procijeniti. Zato se koriste metodi gdje se funkcija odlučivanja formira kroz proces učenja. U tom slučaju nije neophodno postavljati pretpostavke o statističkim osobinama klasa.

Osnovu ovog razmatranja čini upotreba nelinearnih elemenata (nazvanih *neuroni*), organizovanih u mreže u kojima međukonekcije elemenata podsjećaju na strukturu mozga, te se nazivaju *neuronske mreže*. Međutim, još uvijek ostaje nejasno kako ljudski mozak, koji se sastoji od oko 100 milijardi neurona, pri čemu je svaki neuron je povezan sa više hiljada drugih neurona i gdje je brzina reakcije jednog neurona reda milisekundi, sposoban da sa lakoćom izvršava, za računare još uvijek veoma složene zadatke kao što su akvizicija, pamćenje, reprezentacija, pronalaženje, analiza i prepoznavanje slika.

Perceptron za dvije klase uzoraka

Ovaj perceptron, prikazan na Slici 199, učenjem dolazi do funkcije odlučivanja koja razgraničava dva linearno separabilna seta za obučavanje.

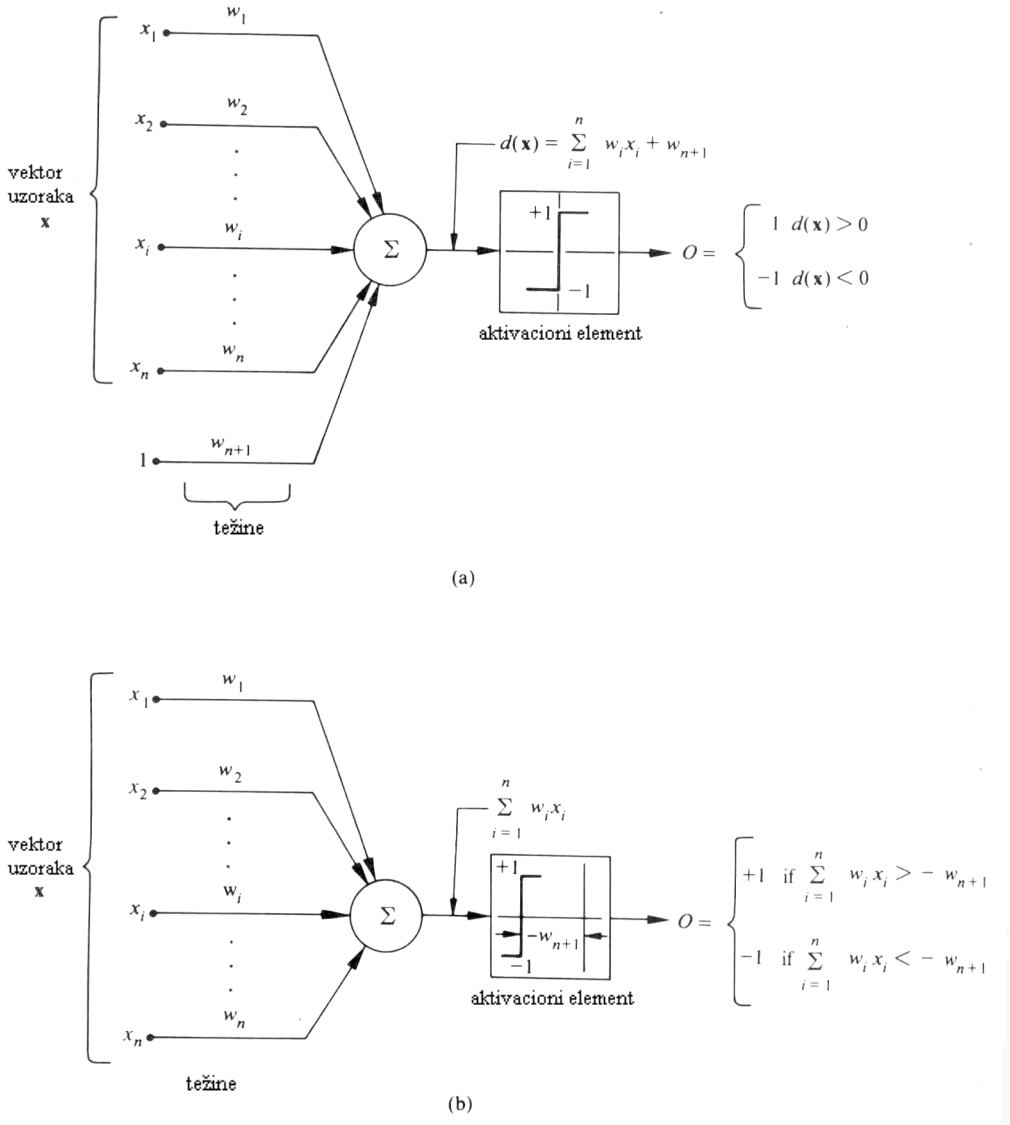
Odziv ovog osnovnog elementa je težinska suma ulaza, odnosno linearna funkcija odlučivanja u odnosu na komponente vektora uzorka:

$$d(\mathbf{x}) = \sum_{i=1}^{n} w_i x_i + w_{n+1} .$$

Težine w_i su analogne sinapsama u čovjekovom nervnom sistemu. Funkcija koja preslikava ovu težinsku sumu u konačni izlaz se naziva *aktivacjska funkcija*. Ako je $d(\mathbf{x}) \rangle 0$ izlaz perceptrona je +1 što označava da uzorak pripada klasi w_1, a ako je $d(\mathbf{x}) \langle 0$ izlaz je jednak –1 i uzorak pripada klasi w_2. Kada je $d(\mathbf{x}) = 0$ $\mathbf{x}$ leži na granici, pa se prema tome granica između klasa koju postavlja perceptron dobija sa:

$$d(\mathbf{x}) = \sum_{i=1}^{n} w_i x_i + w_{n+1} = 0 \quad \text{ili} \quad w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + w_{n+1} = 0$$

što je jednačina hiperravni u *n*-dimenzionalnom prostoru uzoraka. Geometrijski, prvih *n* koeficijenata određuje orijentaciju hiperravni, dok je posljedniji koeficijent w_{n+1} proporcionalan normalnoj udaljenosti hiperravni od ishodišta (ako je $w_{n+1} = 0$ hiperravan razgraničenja prolazi kroz koordinatni početak prostora uzoraka). Slično, ako je $w_j = 0$, hiperravan je paralelna x_j osi.



Slika 199. [1] Preceptron za dvije klase uzoraka

Da bi izbjegli pomijeranje aktivacione funkcije, u praksi se vektoru uzoraka dodaje još jedan element koji je uvijek jednak jedinici, Slika 199(b), tako da se dobija pošireni vektor uzoraka:

$$y_i = x_i,\ i = 1,2,...,n;\ y_{n+1} = 1$$
$$d(\mathbf{y}) = \sum_{i=1}^{n+1} w_i y_i = \mathbf{w}^T \mathbf{y}$$

Osnovni zadatak je pronaći vektor težina koristeći set uzoraka za obučavanje.

Algoritmi obučavanja

Linearno separabilne klase

Neka je $\mathbf{w}(1)$ inicijalni vektor težina, odabran proizvoljno. U *k*-tom koraku iteracije (podešavanja težina):

$$\mathbf{y}(k) \in w_1 \wedge \mathbf{w}^T(k)\mathbf{y}(k) \le 0 \Rightarrow \mathbf{w}(k+1) = \mathbf{w}(k) + c\mathbf{y}(k),$$

$$\mathbf{y}(k) \in w_2 \wedge \mathbf{w}^T(k)\mathbf{y}(k) \ge 0 \Rightarrow \mathbf{w}(k+1) = \mathbf{w}(k) - c\mathbf{y}(k),$$

gdje je *c* pozitivni korekcioni inkrement, za koji ćemo za sada smatrati da je konstantan.

Ove korekcije se, dakle, rade ako je nastupila pogreška pri klasifikaciji. U protivnom, težine se ne mijenjaju:

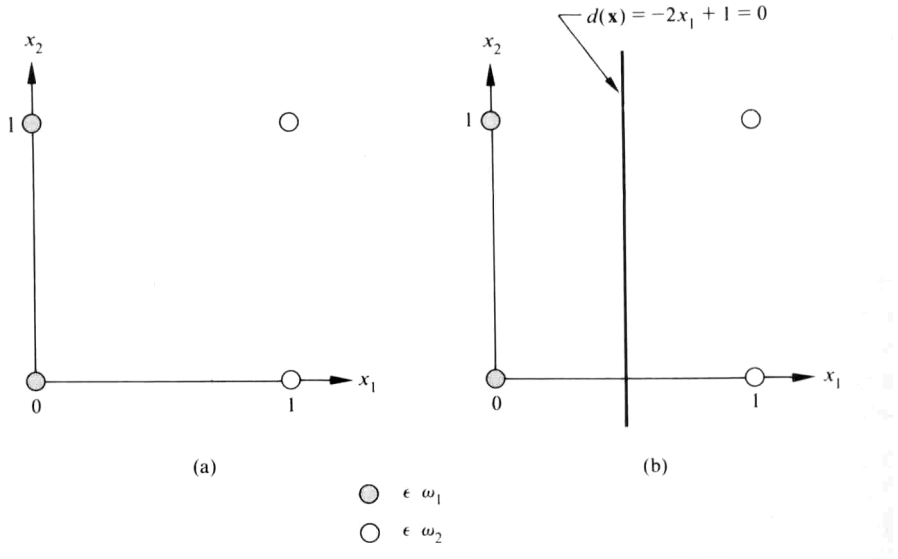
$$\mathbf{w}(k+1) = \mathbf{w}(k).$$

Izloženi algoritam se naziva *pravilo fiksne inkrementalne korekcije*.

Ovaj metod učenja je zasnovan na principu *kazni-nagradi*. Postupak obučavanja se završava kada kompletan set za obučavanje iz obje klase prođe proces klasifikacije bez greške. Algoritam fiksne inkrementalne korekcije konvergira u konačnom broju koraka ako su dva seta za obučavanje linearno separabilna (*perceptron training theorem*).

Primjer:

U primjeru datom na Slici 200 sa dvije klase uzoraka, set za obučavanje je:
$\left\{(0,0,1)^T, (0,1,1)^T\right\}$ za klasu w_1 $\left\{(1,0,1)^T, (1,1,1)^T\right\}$ za klasu w_2.



Slika 200. [1] Ilustracija algoritma za obučavanje: (a) uzorci iz dvije klase, (b) linija razgraničenja određena ovim algoritmom

Stavljajući $c=1,\ \mathbf{w}(1)=\mathbf{0}$, slijede koraci:

$$\mathbf{w}^T(1)\mathbf{y}(1)=\begin{bmatrix}0 & 0 & 0\end{bmatrix}\begin{bmatrix}0\\0\\1\end{bmatrix}\quad \mathbf{w}(2)=\mathbf{w}(1)+\mathbf{y}(1)=\begin{bmatrix}0\\0\\1\end{bmatrix},$$

$$\mathbf{w}^T(2)\mathbf{y}(2)=\begin{bmatrix}0 & 0 & 1\end{bmatrix}\begin{bmatrix}0\\1\\1\end{bmatrix}\quad \mathbf{w}(3)=\mathbf{w}(2)=\begin{bmatrix}0\\0\\1\end{bmatrix},$$

$$\mathbf{w}^T(3)\mathbf{y}(3)=\begin{bmatrix}0 & 0 & 1\end{bmatrix}\begin{bmatrix}1\\0\\1\end{bmatrix}\quad \mathbf{w}(4)=\mathbf{w}(3)-\mathbf{y}(3)=\begin{bmatrix}-1\\0\\0\end{bmatrix},$$

$$\mathbf{w}^T(4)\mathbf{y}(4)=\begin{bmatrix}-1 & 0 & 1\end{bmatrix}\begin{bmatrix}1\\1\\1\end{bmatrix}\quad \mathbf{w}(5)=\mathbf{w}(4)=\begin{bmatrix}-1\\0\\1\end{bmatrix}.$$

Skup za obučavanje se mora "propuštati" kroz algoritam, stavljajući $\mathbf{y}(5)=\mathbf{y}(1)$, $\mathbf{y}(6)=\mathbf{y}(2), \mathbf{y}(7)=\mathbf{y}(3), \mathbf{y}(8)=\mathbf{y}(4)$, itd., sve dok ne dobijemo takve vrijednosti težina da svi uzorci iz is seta za obučavanje budi svrstani u pripadajuće klase bez greške. U ovom primjeru konvergencija nastupa u 14 koraku iteracije, a generisani vektor težina $d(\mathbf{y})=-2y_1+1$. Vraćajući se na originalni prostor uzoraka ($x_i=y_i$) imamo jednačinu $d(\mathbf{x})=-2x_1+1$ koja, kad se izjednači sa nulom, predstavlja liniju razgraničenja klasa.

Neseparabilne klase

U praksi su linearno separabilne klase rijetkost. Jedan od prvih metoda koji je pronađen za klase koje nisu linearno separabilne, originalno LMS delta pravilo za obučavanje perceptrona (Widrow-Hoff), minimizira grešku između trenutnog i željenog odziva u svakom koraku učenja.

Neka je kriterij greške (kriterijumska funkcija) dat sa:

$$J(\mathbf{w})=\frac{1}{2}\left(r-\mathbf{w}^T\mathbf{y}\right)^2,$$

gdje je r željeni odziv ($r=+1$ ako vektor uzorka **y** iz seta za obučavanje koji se trenutno posmatra pripada klasi w_1, a $r=-1$ ako vektor uzorka **y** iz seta za obučavanje koji se trenutno posmatra pripada klasi w_2).

Pronalaženje vektora težina pri kom nastupa najmanja srednjekvadratna greška se svodi na traženje minimuma kriterijumske funkcije, odnosno inkrementalno podešavanje vektora težina u smjeru negativnog gradijenta od $J(\mathbf{w})$. Minimalna greška nastupa za $r=\mathbf{w}^T\mathbf{y}$, što odgovara korektnoj klasifikaciji. Ako $\mathbf{w}(k)$ predstavlja vektor težina u k-tom iterativnom koraku, opšti algoritam se može zapisati sa:

$$\mathbf{w}(k+1)=\mathbf{w}(k)-\alpha\left[\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}\right]_{\mathbf{w}=\mathbf{w}(k)},\ \alpha\rangle 0,$$

gdje α daje magnitudu korekcije. Iz izraza za kriterijumsku funkciju dobijamo:

$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = -\left(r - \mathbf{w}^T \mathbf{y}\right)\mathbf{y},$$

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \alpha\left[r(k) - \mathbf{w}^T(k)\mathbf{y}(k)\right]\mathbf{y}(k),$$

gdje početni vektor $\mathbf{w}(1)$ usvajamo proizvoljno.

Ako se korekcija radi samo u slučaju pogrešne klasifikacije, i definišući promjenu težinskog vektora sa:

$$\Delta\mathbf{w} = \mathbf{w}(k+1) - \mathbf{w}(k),$$

možemo pisati (*delta korekcioni algoritam*):

$$\Delta\mathbf{w} = \alpha e(k)\mathbf{y}(k),$$

gdje je:

$$e(k) = r(k) - \mathbf{w}^T(k)\mathbf{y}(k)$$

greška koja se čini sa trenutnim vektorom težina kada se vrši klasifikacija uzorka $\mathbf{y}(k)$.

Ako promijenimo vektor težina sa $\mathbf{w}(k)$ na $\mathbf{w}(k+1)$ a uzorak ostane isti, greška će biti:

$$e(k) = r(k) - \mathbf{w}^T(k+1)\mathbf{y}(k).$$

Promjena greške je:

$$\Delta e = \left[r(k) - \mathbf{w}^T(k+1)\mathbf{y}(k)\right] - \left[r(k) - \mathbf{w}^T(k)\mathbf{y}(k)\right] = -\left[\mathbf{w}^T(k+1) - \mathbf{w}^T(k)\right]\mathbf{y}(k) = -\Delta\mathbf{w}^T\mathbf{y}(k).$$

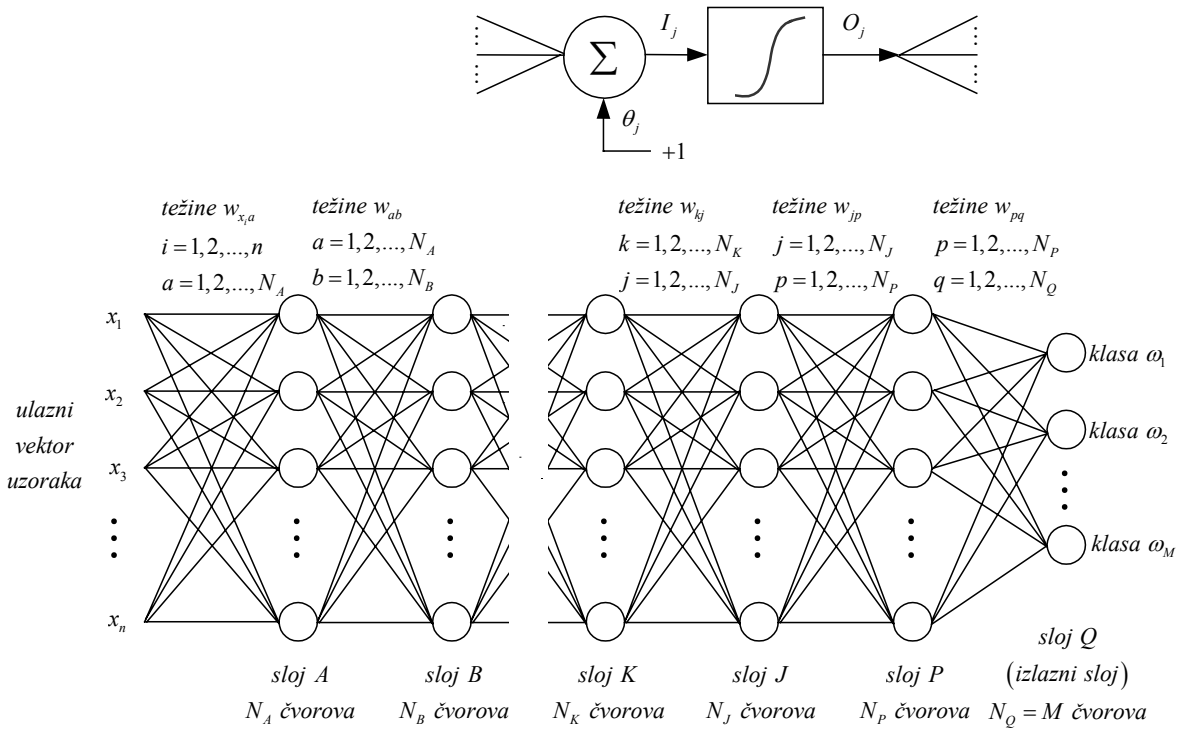
Znajući da je $\Delta\mathbf{w} = \alpha e(k)\mathbf{y}(k)$ imamo:

$$\Delta e = -\alpha e(k)\mathbf{y}^T(k)\mathbf{y}(k) = -\alpha e(k)\left\|\mathbf{y}(k)\right\|^2.$$

Dakle, ovakva promjena vektora težina smanjuje grešku sa faktorom $\alpha\left\|\mathbf{y}(k)\right\|^2$. Dovođenjem sljedećeg uzorka na ulaz počinje novi ciklus adaptacije, koji smanjuje grešku sa faktorom $\alpha\left\|\mathbf{y}(k+1)\right\|^2$, itd… Izbor parametra α kontroliše stabilnost i brzinu konvergencije. Widrow i Stearns su 1985 pokazali da je za stabilnost neophodno da bude $0 \langle \alpha \langle 2$, dok se u praktičnim primjenama bira $0.1 \langle \alpha \langle 1.0$. Algoritam konvergira rješenju koje minimizira srednjekvadratnu grešku na datom set uzoraka za obučavanje. Ako se radi sa separabilnim klasama ovaj algoritam ne mora pronaći istu hiperravan razdvajanja koju bi dao prethodno prikazani algoritam za obučavanje perceptrona u slučaju separabilnih klasa.

Osnovna arhitektura višeslojnih neuronskih mreža

Model višeslojne neuronske mreže prikazan je na Slici 201. Broj neurona u prvom sloju (A) je N_A. Često je N_A=n. Broj neurona u izlaznom sloju (Q) je N_Q i jednak je broju klasa M koje mreža treba da nauči da raspoznaje. Mreža prepoznaje da uzorak **x** pripada klasi w_m ako je m-ti izlaz mreže "visok" dok su ostali izlazi "niski".

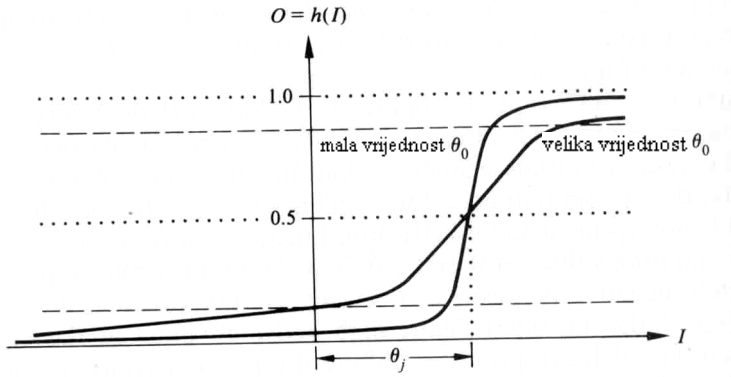


Slika 201. Višeslojna neuronska mreža

Unutar neuronske mreže, svaki neuron ima formu perceptrona, s tim da je aktivacijska funkcija korigovana tako što su oštre granice ublažene izborom aktivacijske funkcije oblika:

$$h_j\left(I_j\right) = \frac{1}{1+\exp\left[-\left(I_j+\theta_j\right)/\theta_0\right]},$$

gdje su I_j, $j = 1,2,...,N_j$ ulazi u aktivacioni element svakog čvora u J-tom sloju mreže, θ_j je ofset, a θ_0 kontroliše oblik aktivacione funkcije. Ovakva aktivacijska funkcija se naziva sigmoidalna aktivacijska funkcija, Slika 202.



Slika 202. [1] Sigmoidalna aktivacijska funkcija

Dakle, izlaz aktivacionog elementa je "visok" za svaku vrijednost ulaza veću od ofseta. Kako je vrijednost aktivacione funkcije uvijek pozitivna i svoje granične vrijednosti 0 i 1 dostiže tek pri beskonačno velikim vrijednostima ulaza, uveden je pojam visokog i niskog izlaza (vrijednosti bliske 0 i 1, npr. 0.05 i 0.95). U principu je moguće koristiti različite aktivacione funkcije za svaki čvor mreže, ali se u praksi uobičajeno bira ista aktivacijska funkcija za cijelu mrežu. Ofset je isto što i težinski koeficijent w_{n+1} i ne prikazuje se posebno već predstavlja integralni dio svakog čvora.

Ulaz u *j*-ti čvor bilo kog sloja (*J*) je težinska suma izlaza iz prethodnog sloja (*K*):

$$I_j = \sum_{k=1}^{N_K} w_{kj} O_k,\ j = 1,2,...,N_J,$$

dok su izlazi *K*-tog sloja:

$$O_k = h_k\left(I_k\right),\ k = 1,2,...,N_K.$$

N_K ulaza u prvi čvor sloja *J* imaju težinske koeficijenta w_{k1}, itd…

Ukupno imamo N_JxN_K koeficijenata kojima se množe izlazi sloja *K* i dovode na ulaze sloja *J*, uz dodatne koeficijente θ_j. Uvrštavajući izraz koji opisuje ulaze sloja *J* u izraz za aktivacionu funkciju imamo:

$$h_j\left(I_j\right) = \frac{1}{1+\exp\left[-\left(\sum_{k=1}^{N_K} w_{kj} O_k + \theta_j\right) \Big/ \theta_0\right]}.$$

Obučavanje propagacijom unazad

Obučavanje neuronske mreže podrazumijeva podešavanje svih težina kroz cijelu mrežu tako da ukupna greška za sve prezentirane uzorke iz seta za obučavanje na svim neuronima izlaznog stepena bude minimalna. Greška jednog (*k*-tog) neurona izlaznog sloja za sve prezentirane uzorke je:

$$E_k = \frac{1}{2}\sum_{d\in D}\left(r_d - O_d\right)^2,$$

gdje je r_d željeni, a O_d stvarni odziv posmatranog neurona izlaznog sloja kad je na ulazu mreže jedan uzorak iz seta za obučavanje $d \in D$.

Ukupna greška, za sve neurone izlaznog sloja je:

$$E_Q = \frac{1}{2}\sum_{d\in D}\sum_{k\in Q}\left(r_{dk} - O_{dk}\right)^2,$$

gdje je Q skup izlaznih neurona.

Gradijent greške:

$$\nabla E_Q(\mathbf{w}) = \left[\frac{\partial E_Q}{\partial w_1}, \frac{\partial E_Q}{\partial w_2}, \ldots, \frac{\partial E_Q}{\partial w_n}\right]$$

ima brojne lokalne minimume. Problem traženja globalnog minimuma se svodi na optimizaciju u višedimenzionalnom prostoru, čija je dimenzija jednaka broju težina u mreži.

Za optimizaciju se koristi LMS gradijentni spust. Negativan gradijent ukazuje na smjer pada pogreške, pa se težine mijenjaju proporcionalno parcijalnom izvodu greške:

$$\mathbf{w} \leftarrow \mathbf{w} + \Delta\mathbf{w},$$

$$\Delta\mathbf{w} = -\alpha\nabla E_Q(\mathbf{w}).$$

Za svaku komponentu vektora vrijedi:

$$\Delta w_{ij} = -\alpha\frac{\partial E_Q}{\partial w_{ij}}.$$

Cilj je pronaći pravilo učenja, slično delta pravilu, koje podešava težine u svakom sloju tako da se minimizira ova funkcija greške. Algoritam podešavanja težina propagacijom greške unazad se zasniva na inkrementalnom (postepenom) podešavanju težina za svaki prezentirani uzorak, tzv. «stohastička aproksimacija gradijentnog spusta». Umjesto da grešku računamo za sve uzorke pa onda podesimo težine, računa se greška za svaki uzorak i odmah podešavaju

težine. Zbog toga u daljem razmatranju posmatramo grešku za jedan uzorak za sve neurone izlaznog sloja:

$$E_Q = \frac{1}{2}\sum_{k\in Q}\left(r_k - O_k\right)^2 .$$

Nakon svih uzoraka (jedne epohe) očekujemo da će ukupna greška za sve uzorke biti manja.

Posmatrajmo jedan neuron. Greška svih neurona izlaznog sloja može da zavisi od izlaza neurona u bilo kom sloju, izlaz neurona u tom sloju zavisi od ulaza u aktivacioni element koji je opet funkcija težina tog neurona:

$$\begin{aligned} E_Q &= f\left(O_j,...\right), \\ O_j &= f\left(I_j,...\right), \\ I_j &= f\left(w_{ij},...\right). \end{aligned}$$

Tako možemo pisati:

$$\Delta w_{ij} = -\alpha\frac{\partial E_Q}{\partial w_{ij}} = -\alpha\frac{\partial E_Q}{\partial I_j}\frac{\partial I_j}{\partial w_{ij}} .$$

Kako je:

$$\frac{\partial I_j}{\partial w_{ij}} = \frac{\partial\sum_i w_{ij}O_i}{\partial w_{ij}} = O_i ,$$

uvodeći oznaku:

$$\delta_j = -\frac{\partial E_Q}{\partial I_j}$$

korekcioni član se može pisati kao:

$$\Delta w_{ij} = \alpha O_i\delta_j .$$

δ_j je pogreška kojom neuron *j* bilo kog sloja utiče na ukupnu grešku na izlazu mreže:

$$\delta_j = -\frac{\partial E_Q}{\partial I_j} = -\frac{\partial\left[\frac{1}{2}\sum_{k\in Q}\left(r_k - O_k\right)^2\right]}{\partial I_j} .$$

Greška je funkcijski ovisna o svim izlazima neurona izlaznog sloja. Zanima nas kako na izlaze pojedinih neurona izlaznog sloja utiče ulaz bilo kog neurona u bilo kom sloju da bi mogli podedsiti njegove težine tako da minimiziramo ukupnu grešku.
Razmotrimo dva slučaja:

1. ulaz neurona utiče direktno na grešku u izlaznom sloju (radi se o jednom od neurona izlaznog sloja, $j \in Q$),
2. ulaz neurona indirektno utiče na grešku u izlaznom sloju (radi se o jednom od neurona u nekom od unutrašnjih slojeva, $j \notin Q$).

Neuron vanjskog sloja

Za neuron vanjskog sloja, $j \in Q$, vrijedi:

$$\delta_j = -\frac{\partial E_Q}{\partial I_j} = -\frac{\partial\left[\frac{1}{2}\sum_{q\in Q}\left(r_q - O_q\right)^2\right]}{\partial I_j}, j = 1,2,...,N_Q.$$

Ako se radi o neuronu izlaznog sloja onda njegov ulaz utiče na izlaz samo jednog neurona vanjskog sloja, odnosno, samo jedan $O_q, q \in Q$ je funkcija I_j:

$$\delta_j = -\frac{\partial\left[\frac{1}{2}\sum_{q\in Q}\left(r_q - O_q\right)^2\right]}{\partial I_j} = -\frac{\partial\left[\frac{1}{2}\left(r_j - O_j\right)^2\right]}{\partial I_j} = \left(r_j - O_j\right)\frac{\partial O_j}{\partial I_j}, j = 1,2,...,N_Q.$$

Izvod sigmoidalne funkcije $\left(\theta_0 = 1\right)$ je:

$$\frac{\partial O_j}{\partial I_j} = \frac{\partial\left(\frac{1}{1+e^{-I_j}}\right)}{\partial I_j} = \frac{e^{-I_j}}{\left(1+e^{-I_j}\right)^2} = O_j\left(1-O_j\right), j = 1,2,...,N_Q,$$

te je konačno:

$$\delta_j = O_j\left(1-O_j\right)\left(r_j - O_j\right), j = 1,2,...,N_Q.$$

Ovo su pogreške kojima neuroni vanjskog sloja $\left(j \in Q\right)$ utiču na ukupnu pogrešku. Zamijenićemo indeks i označiti pogreške sa:

$$\delta_q = O_q\left(1-O_q\right)\left(r_q - O_q\right), q = 1,2,...,N_Q,$$

kako bi bilo jasnije da se odnose na izlazni sloj.

Korekcija težina vanjskog sloja se radi na osnovu:

$$\Delta w_{pq} = \alpha\delta_q O_p.$$

Kada je funkcija $h_q\left(I_q\right)$ specificirana (mi smo pretpostavili sigmoidalnu funkciju), svi elementi za korekciju težina su poznati, ili se mogu dobiti iz mreže. Nakon dovođenja uzorka iz seta za učenje na ulaz mreže, mi znamo željeni odziv r_q u svakom čvoru izlaznog sloja.Vrijednosti O_q i O_p se mogu snimiti. Znači, znamo kako da podesimo težine poslednjeg sloja.

Neuron unutrašnjeg sloja

Kad bi poznavali željeni odziv neurona unutrašnjih slojeva, provedeno razmatranje bi se moglo proširiti i na ove neurone. Međuti, mi poznajemo samo željeni odziv na izlazu cijele mreže.

Posmatrajmo sloj koji direktno prethodi izlaznom sloju. Uticaj neurona *j* iz sloja koji direktno prethodi izlaznom sloju na ukupnu pogrešku je:

$$\delta_j = -\frac{\partial E_Q}{\partial I_j} = -\frac{\partial\left[\frac{1}{2}\sum_{q\in Q}\left(r_q - O_q\right)^2\right]}{\partial I_j}, j = 1,2,...,N_P .$$

Ulaz aktivacionog elementa tog neurona utiče na izlaze onih neurona vanjskog sloja koji su sa prethodnim slojem vezani preko težina w_{pq}. Označimo taj skup izlaznih neurona sa:

$$Q_P = \left\{q \middle| \exists w_{pq}\right\}.$$

Za te neurone je:

$$I_q = f\left(I_j\right), j = 1,2,...,N_P, q \in Q_P ,$$

te imamo:

$$\delta_j = -\frac{\partial E_Q}{\partial I_j} = -\frac{\partial\left[\frac{1}{2}\sum_{q\in Q}\left(r_q - O_q\right)^2\right]}{\partial I_j} = -\sum_{q\in Q_P}\frac{\partial\left[\frac{1}{2}\left(r_q - O_q\right)^2\right]}{\partial I_q}\frac{\partial I_q}{\partial I_j} = \sum_{q\in Q_P}\delta_q\frac{\partial I_q}{\partial I_j}, j = 1,2,...,N_P .$$

Uticaj ulaza aktivacionog elementa sloja koji direktno prethodi izlaznom na ulaze aktivacionog elementa izlaznog sloja je:

$$\frac{\partial I_q}{\partial I_j} = \frac{\partial I_q}{\partial O_j}\frac{\partial O_j}{\partial I_j}, j = 1,2,...,N_P .$$

Kako je:

$$\frac{\partial I_q}{\partial O_j} = \frac{\partial\sum_{k=1}^{N_P} w_{kq}O_k}{\partial O_j} = w_{jq},$$

dobivamo:

$$\delta_j = \sum_{q\in Q_P}\delta_q w_{jq}\frac{\partial O_j}{\partial I_j} = \frac{\partial O_j}{\partial I_j}\sum_{q\in Q_P}\delta_q w_{jq}, j = 1,2,...,N_P .$$

Zamjenom indeksa *j* sa *p* da naglasimo o kom sloju se radi, imamo:

$$\delta_p = \frac{\partial O_p}{\partial I_p} \sum_{q \in Q_P} \delta_q w_{pq}, p = 1,2,..., N_P .$$

Za sigmoidalnu aktivacionu funkciju taj izraz postaje:

$$\delta_p = O_p \left(1 - O_p\right) \sum_{q \in Q_P} \delta_q w_{pq}, p = 1,2,..., N_P .$$

Korekcija težina u stepenu koij direktno prethodi izlaznom stepenu je data sa:

$$\Delta w_{jp} = \alpha \delta_p O_j$$

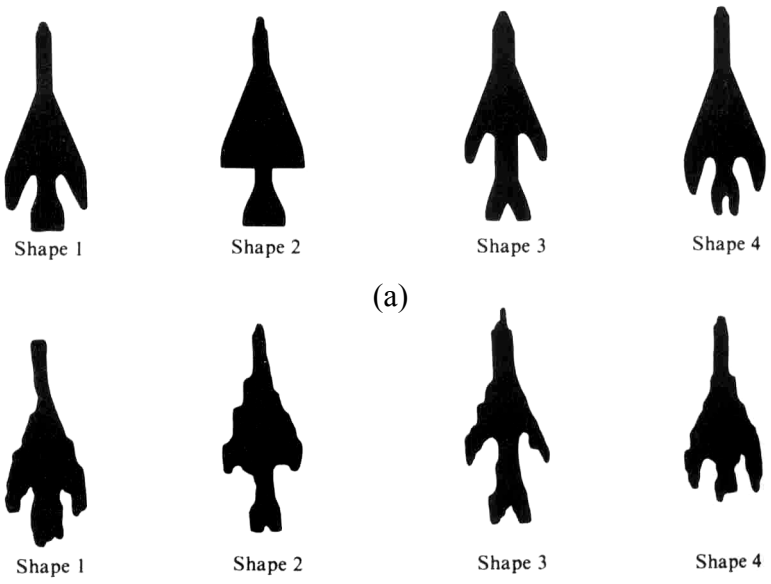
i računa se na osnovu ranije određenih težina vanjskog sloja i mjerljivih izlaza prethodnog sloja. Svaki neuron iz sloja do izlaznog je za ukupnu pogrešku odgovoran onoliko koliko je povezan sa izlazom i u skladu s tim biće provedena korekcija njegovih težina.

Ovaj postupak se može uopštiti. Svaki neuron iz jednog sloja je za ukupnu pogrešku na izlazu sloja koji direktno slijedi odgovoran onoliko koliko je povezan sa tim slojem. Postupak se nastavlja tako što se težine svakog prethodnog sloja računaju na osnovu težina koje ga povezuju sa sljedećim slojem i koje su već određene i izlaza prethodnog sloja. Na ovaj način se vrši propagacija greške unazad po mreži, te se algoritam naziva "*back propagation algorithm*".

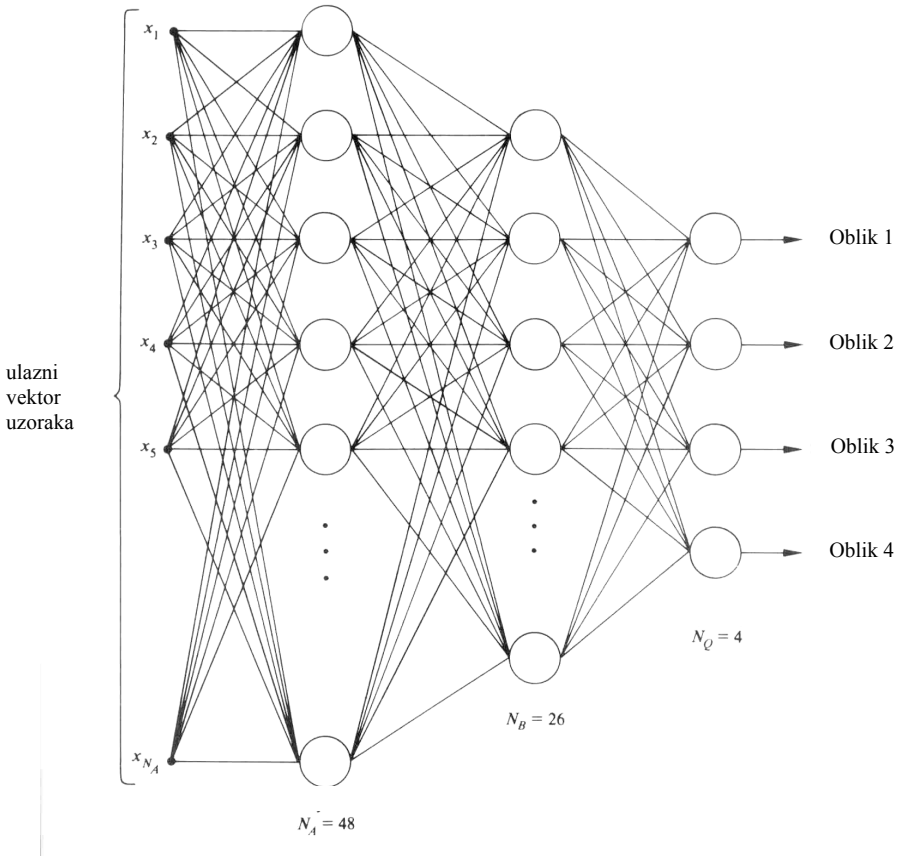
Primjer:

Uzorci za obučavanje su prikazani na Slici 203. Neuronska mreža ima zadatak da vrši razvrstavanjje uzoraka u četiri klase. Struktura mreže je prikazana na Slici 204. Obučavanje se vrši sa uzorcima bez šuma i sa uzorcima sa šumom. P je vjerovatnoća da je rubni piksel test uzorka zadržao svoju vrijednost, dok je vejrovatnoća da će doći do promjene vrijednosti rubnog piksela R=1-P. Indeks t se koristi da označi ovu vjerovatnoću kod test uzoraka. Tako R_t =0.0 označava da su test uzorci bez šuma, dok R_t =0.2 označava da su test uzorci sa šumom generisani tako da je vjerovatnoća promjene vrijednosti rubnog piksela 0.2.

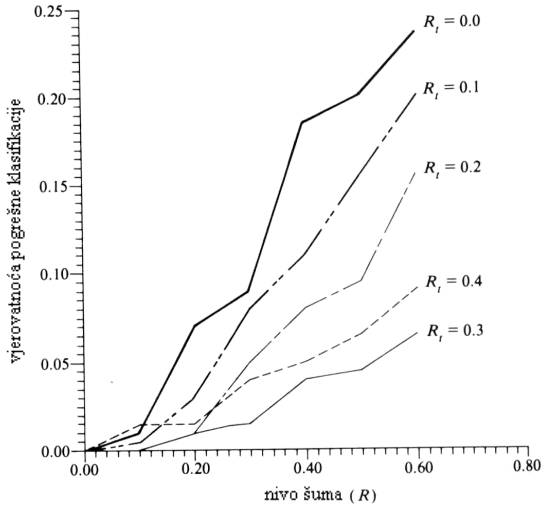
Mreža je obučavana sa 10 uzoraka. Zatim je mreža testirana sa test uzorcima (100 iz svake klase) pri čemu je R varirano između 0.1 i 0.4. Bolji rezultati se postižu pri obučavanju sa uzorcima sa šumom, Slika 205. Međutim, kad su test uzorci generisani sa velikom šumom ovo prestaje da važi, kao što se vidi za slučaj R_t =0.4. Tada je 10 uzoraka suviše malo za obučavanje mreže. Povećavajući broj uzoraka za obučavanje na 40 vidi se da se bolji rezultati postižu pri obučavanju mreže sa uzorcima sa većim nivoom šuma, Slika 206.



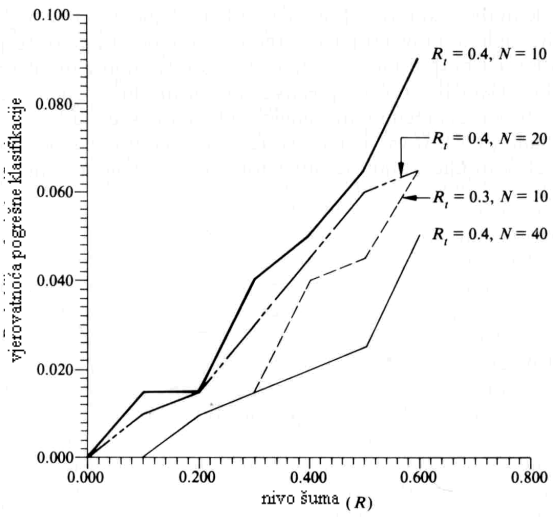
Slika 203. [1] Uzorci za obučavanje: (a) bez šuma, (b) sa šumom



Slika 204. Neuronska mreža sa četiri sloja



Slika 205. [1] Performanse neuronske mreže u zavisnosti od nivoa šuma



Slika 206. [1] Uticaj broja uzoraka za obučavanje na poboljšanje performansi neuronske mreže