Matematičke osnove neuronskih mreža

Kalinić, Matea

Master's thesis / Diplomski rad

2017

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Split, University of Split, Faculty of science / Sveučilište u Splitu, Prirodoslovno-matematički fakultet**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:166:112258

Rights / Prava: Attribution-NonCommercial-NoDerivatives 4.0 International/Imenovanje-Nekomercijalno-Bez prerada 4.0 međunarodna

Download date / Datum preuzimanja: **2024-03-28**

Repository / Repozitorij:

Repository of Faculty of Science

zir.nsk.hr

UNIVERSITY OF SPLIT

dabar

DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

PRIRODOSLOVNO–MATEMATIČKI FAKULTET
SVEUČILIŠTA U SPLITU

MATEA KALINIĆ

MATEMATIČKE OSNOVE NEURONSKIH MREŽA

DIPLOMSKI RAD

Split, lipanj 2017.

PRIRODOSLOVNO–MATEMATIČKI FAKULTET
SVEUČILIŠTA U SPLITU

ODJEL ZA MATEMATIKU

MATEMATIČKE OSNOVE NEURONSKIH MREŽA

DIPLOMSKI RAD

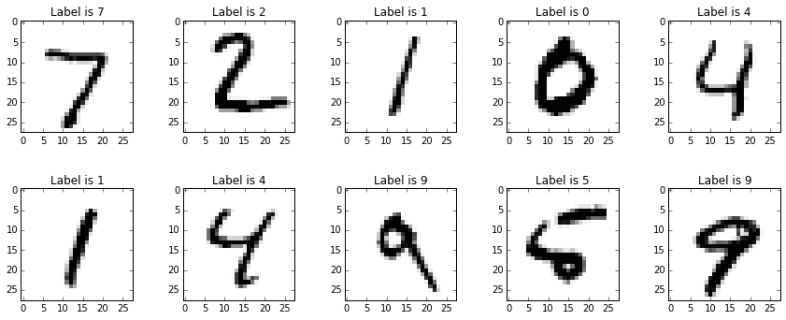
Studentica:
Matea Kalinić

Mentor:
doc.dr.sc. Ivo Ugrina

Split, lipanj 2017.

Uvod

Usporedno s nastankom računala, nastajala je i razvijala se i čovjekova želja za strojevima koji će razmišljati, biti inteligentni. Inteligencijom smatramo mentalnu karakteristiku koja se sastoji od sposobnosti za učenje iz iskustva, prilagodbe na nove situacije, razumijevanja i korištenja apstraktnih pojmova te korištenja stečenih znanja za snalaženje u okolini. Iako su moderna računala sposobna brže i uspješnije od čovjeka riješiti mnoge zahtjevne probleme predstavljene kao niz matematički složenih pravila i dalje ne uspijevaju nadmašiti čovjeka u njemu izrazito jednostavnim, svakodnevnim zadaćama kao što je uočavanje lica na fotografiji ili čitanje tuđeg rukopisa. Potreba za razvojem strojeva sa ciljem reproduciranja čovjekove inteligencije oblikovala je novu granu računarske znanosti - umjetnu inteligenciju.



Slika 1: Rukom pisani brojevi [6]

Vratimo se sada spomenutom problemu. Zamislimo da je dan niz brojeva

prikazanih na Slici 1, napisanih rukom od strane nekog čovjeka. Velika većina ljudi s lakoćom će pročitati napisano, no to za računalo nije slučaj. Jedan od ciljeva ovog rada bit će objasniti matematički model za *učenje* računala kako prepoznavati rukom napisane brojeve.

U tu svrhu, prisjetimo se naših prvih susreta s brojevima. Uglavnom, djeca se kroz odrastanje prvo upoznaju s pojmom broja i broje koristeći se prstima ruku, a tek potom uče iste zapisati. Pritom im je prvo *pokazano* kako se koji broj piše, a kasnije je *zahtjevano* od njih samih da znakom izraze željeni broj i prepoznaju zapisani. Ovaj proces učenja zove se **nadzirano učenje** ili **učenje s nadzorom**. Dakle, to je vrsta učenja kod kojeg je subjektu koji uči prvo *dan* određeni broj primjera koje čine uzorci upareni s točnim odgovorima, a onda je od njih zahtjevana primjena znanja na nepoznato. Grana umjetne inteligencija koja se bavi ovakvim procesima naziva se strojno učenje.

Jedan od najčešćih zadatka koji se rješavaju pomoću strojnog učenja je klasifikacija. Taj tip zadatka traži od računala da specificira kojoj od k kategorija pripada određeni ulaz. Kako bi se ovaj zadatak riješio potrebno je pronaći funkciju

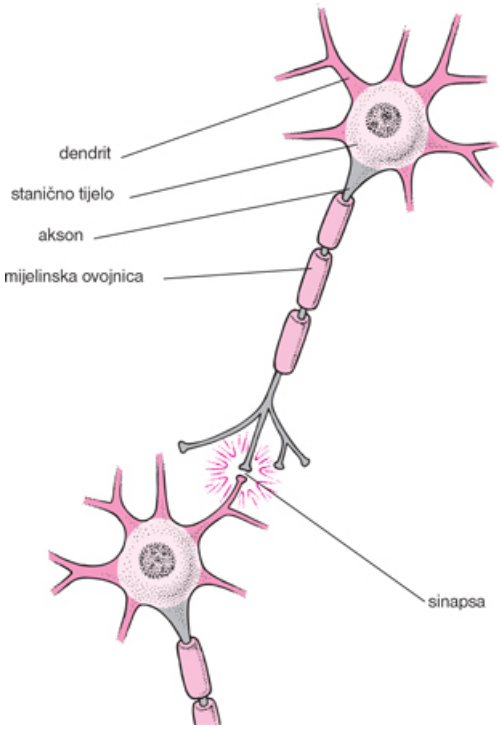
$$f : \mathbb{R}^n \to \{1, \ldots, k\}$$

koja izrazom

$$y = f(x)$$

govori da je ulaz x klasificiran kao izlaz y. Prije spomenuti primjer prepoznavanja brojeva je upravo primjer klasifikacijskog problema.

S obzirom da karakteristike ljudske inteligencije želimo na neki način prenijeti na računala, ima smisla pogledati kako zaista funkcionira ljudski živčani sustav — sustav koji je zadužen za primanje, prijenos i pohranjivanje informacija te za obavljanje misaonih radnji i reakcija na podražaje.



Slika 2: Građa živčane stanice [13]

Temeljna građevna jedinica živčanog sustava je **živčana stanica**(neuron) koja se sastoji od tijela (some) s kojeg se pruža veći broj kraćih živčanih vlakana zvanih dendriti i jedno duže živčano vlakno zvano akson koje završava malim zadebljanjima s pomoću kojih se stvara veza između pojedinih neurona ili neurona i mišičnog vlakna koju zovemo sinapsa. Osjetilna tjelešca na dendritima primaju podražaje koji onda mijenjaju električni naboj stanice. Promijenjeni električni potencijal stanice prenosi se tijelom neurona do završetaka aksona, a onda se izlučivanjem neurohormona podražaj prenosi na drugi neuron ili vlakno.

Građa i funkcija živčanog sustava čovjeka inspirirala je model za strojno učenje - umjetnu neuronsku mrežu.

Sadržaj

Uvod **iii**

Sadržaj **vi**

1 Neuronska mreža **1**

1.1 Jednostavni g-perceptron 1

1.2 Geometrijska interpretacija i primjeri 5

1.3 Jednostavna neuronska mreža 7

1.4 Višeslojni perceptron 10

2 Proces učenja **16**

2.1 Funkcija gubitka 17

2.2 Metoda gradijentnog spusta 18

2.3 Algoritam s povratnim proslijeđivanjem pogreške 21

2.3.1 Arhitektura mreže 22

2.3.2 Primjena algoritma na jednoslojnu neuronsku mrežu . 31

2.3.3 Matrična forma algoritma 36

3 Naprednije arhitekture **38**

3.1 Elmanova mreža 39

3.2 Povratni višeslojni perceptron 43

3.3 Algoritam povratnog proslijeđivanja pogreške kroz vrijeme . . 44

4 Zaključak **47**

Literatura **48**

Poglavlje 1

Neuronska mreža

1.1 Jednostavni g-perceptron

Za razumijevanje arhitekture i funkcije neuronskih mreža od iznimine je važnosti njihovo grafičko predočavanje. Figura sastavljena od točaka i linija koje ih povezuju je geometrijska interpretacija apstraktnog matematičkog objekta - grafa.

Definicija 1.1 ***Graf*** *je uređena trojka* (V, E, ϕ)*, gdje je* V *neprazan konačan skup čije elemente nazivamo* ***vrhovima****,* E *konačan skup čije elemente nazivamo* ***bridovima*** *i* ϕ *preslikavanje*

$$\phi : E \to \mathscr{P}(V)$$

koje svakom bridu $e \in E$ *pridružuje par (dvočlani skup) vrhova* $\{u, v\} \in \mathscr{P}(V)$*.*

Definicija 1.2 ***Usmjereni graf*** *je uređena trojka iz Definicije 1.1 kod koje je* ϕ *preslikavanje*

$$\phi : E \to V \times V$$

koje svakom bridu $e \in E$ *pridružuje uređeni par vrhova* $(u, v) \in V \times V$*.*

Definicija 1.3 ***(Usmjereni) težinski graf*** *je uređeni par (G, w), gdje je $G = (V, E, \phi)$ (usmjereni) graf i*

$$w : E \to \mathbb{R}$$

funkcija koja svakom bridu $e \in E$ pridružuje broj $w(e) \in \mathbb{R}$ koji nazivamo ***težinom brida****.*

Napomena 1.4 *S obzirom da ćemo ovdje promatrati samo usmjerene težinske grafove, nadalje ćemo ih, radi kratkoće, zvati samo grafovima. Još, umjesto $\phi(e) = (u, v)$, pisat ćemo samo $e = (u, v)$.*

Inspirirani organizacijom neuronskih mreža u živčanom sustavu čovjeka, uvodimo sljedeću definiciju:

Definicija 1.5 ***Jednostavni g-perceptron*** *je uređena četvorka $(X, Y, g, s_{(w_1,\ldots,w_n,\theta)})$, gdje je $X \subseteq \mathbb{R}^n$, za neki $n \in \mathbb{N}$, $Y \subseteq \mathbb{R}$, $s_{(w_1,\ldots,w_n,\theta)}$ funkcija koju zovemo* ***integralnom funkcijom***

$$s_{(w_1,\ldots,w_n,\theta)} : X \to \mathbb{R}$$

dana pravilom

$$s_{(w_1,\ldots,w_n,\theta)}(x) = s_{(w_1,\ldots,w_n,\theta)}(x_1, \ldots, x_n) := \sum_{j=1}^{n} w_j x_j - \theta,$$

gdje su $w_1, \ldots, w_n, \theta \in \mathbb{R}$ parametri te g proizvoljna funkcija

$$g : \mathbb{R} \to Y$$

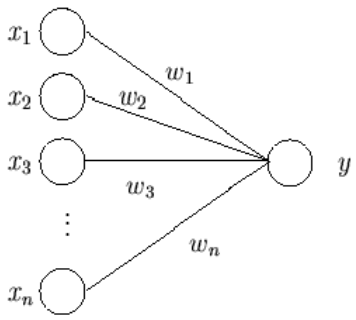
koju zovemo ***aktivacijskom funkcijom****.*

Napomena 1.6 *Radi jednostavnosti zapisa ubuduće ćemo pisati $s = s_{(w_1,\ldots,w_n,\theta)}$.*

Inutitivno, X je skup svih mogućih uređenih n-torki $(x_1, \ldots, x_n)$ koje promatramo, a Y je skup svih mogućih realnih brojeva dobivenih na način

$$y = g\left(s(x_1, \ldots, x_n)\right) = g\left(\sum_{j=1}^{n} w_j x_j - \theta\right).$$

Uređenu n-torku $x = (x_1, \ldots, x_n) \in X$ najčešće ćemo zvati ulazom, a realni broj $y \in Y$, dobiven kako je navedeno, izlazom. Nadalje, g-perceptron ćemo najčešće predočavati, i štoviše, poistovjećivati s grafom (G, w), $G = (V, E, \phi)$, gdje je $V = \{v_1, \ldots, v_n, u\}$, a $E = \{(v_i, u) : i \in \{1, \ldots, n\}\}$. Pritom će $x_1, \ldots, x_n$ i y predstavljati vrijednosti vrhova grafa, ali zbog jednostavnosti i tradicije, govorit ćemo da su to *vrhovi grafova*, a ne *vrijednosti vrhova grafova*, što bi bilo pravilnije.



Slika 1.1: g-perceptron [12]

Vrijednosti $x_1, \ldots, x_n$ možemo zamišljati kao signale koji se putem bridova grafa, to jest sinapsa u živčanom sustavu čovjeka, prenose od jednog vrha do drugog, to jest od jedne živčane stanice do druge. Jačina signala pri prijenosu ne ovisi samo o njegovoj vrijednosti, već i o jakosti veze između živčanih stanica koje komuniciraju. Tu jakost, odnosno težine bridova, zovemo sinaptičkim težinama i označavamo sa w_j. Formalno, $w_j = w((v_j, u))$, za $j \in \{1, \ldots, n\}$. Parametar θ je računarski ekvivalent praga podražljivosti u živčanoj stanici. Često ćemo govoriti da je *funkcija g aktivacijska funkcija*

pridružena vrhu y danog grafa G te da je θ *parametar vrha y* umjesto da je *dan jednostavni g-perceptron s parametrom* θ *predočen grafom G*. Sve vrhove grafa uglavnom ćemo zvati neuronima. Za one koji predstavljaju ulaz reći ćemo da čine ulazni sloj (engl. *input layer*), a za one koji predstavljaju izlaz reći ćemo da čine izlazni sloj (engl. *output layer*).

Zamislimo da g-perceptronom želimo opisati sljedeći način komunikacije: n neurona šalje impulse vrijednosti x_j, $j = 1, \ldots, n$ vezama jačina w_j, $j = 1, \ldots, n$ jednom neuronu u izlaznom sloju koji producira izlaz 1 ako je dobiveni ulaz $\sum_{j=1}^{n} w_j x_j$ prešao neki unaprijed zadani parametar θ, a 0 inače. U tom slučaju spomenuta funkcija g je step funkcija, to jest

$$g : \mathbb{R} \to Y = \{0, 1\}$$

$$g(z) = \begin{cases} 1 & \text{, ako je } z \geq 0 \\ 0 & \text{, ako je } z < 0 \end{cases},$$

to jest vrijedi

$$g(s(x)) = g(s(x_1, \ldots, x_n)) = \begin{cases} 1 & \text{, ako je } \sum_{j=1}^{n} w_j x_j \geq \theta \\ 0 & \text{, ako je } \sum_{j=1}^{n} w_j x_j < \theta \end{cases}.$$

Definicija 1.7 *Ako je funkcija g u Definiciji 1.5 step funkcija, onda g-perceptron nazivamo samo* ***jednostavnim perceptronom*** *ili* ***McCulloch-Pitts neuronom****.*

Često podatak koji govori ima li ili nema podražaja nije dovoljan. Želimo znati koliki je intezitet podražaja u rasponu na primjer $\langle 0, 1\rangle$. Tada koristimo sigmoidne funkcije, odnosno ograničene diferencijabilne funkcije g definirane nad $\mathbb{R}$ kod kojih postoji nenegativna derivacija u svakoj točki.

Najčešći primjeri takvih sigmoidnih funkcija su:

1. $k : \mathbb{R} \to \langle 0, 1\rangle$, $k(x) = \frac{1}{1+e^{-\alpha x}}$, gdje je $\alpha > 0$,
2. $h : \mathbb{R} \to \langle 0, 1\rangle$, $h(x) = \frac{1}{2}(1 + \tanh(x))$.

1.2 Geometrijska interpretacija i primjeri

Definicija 1.8 *Za skup $A \subseteq \mathbb{R}^n$ kažemo da je* ***afino separabilan*** *od skupa $B \subseteq \mathbb{R}^n$ ako postoji $(w_1, \ldots, w_n, \theta) \in \mathbb{R}^{n+1}$ tako da je*

$$\sum_{i=1}^{n} w_i x_i - \theta \begin{cases} \geq 0 & , \text{ ako je } x \in A \\ < 0 & , \text{ ako je } x \in B \end{cases}.$$

Skup $H = \{(x_1, \ldots, x_n) \in \mathbb{R}^n : \sum_{i=1}^{n} w_i x_i = \theta\}$ nazivamo ***separabilnom hiperravninom****.*

Sada ćemo formulirati nužan i dovoljan uvjet za realizaciju logičkih funkcija pomoću McCulloch-Pitts neurona.

Teorem 1.9 *Neka je $X \subseteq \mathbb{R}^n$. Funkcija $f : X \to \{0, 1\}$ može se realizirati pomoću McCulloch-Pitts neurona ako i samo ako je X_+ afino separabilan od X_-, gdje je*

$$X_+ := f^{-1}(1) \subseteq X \text{ i } X_- := f^{-1}(0) \subseteq X.$$

Dokaz. Funkcija f se može realizirati pomoću McCulloch-Pitts neurona ako i samo ako postoji $(w_1, \ldots, w_n, \theta) \in \mathbb{R}^{n+1}$ tako da je

$$f(x_1, \ldots, x_n) = g\left(\sum_{i=1}^{n} w_i x_i - \theta\right),$$

za svaki $(x_1, \ldots, x_n) \in X$, gdje je g step funkcija, što vrijedi ako i samo ako je

$$\sum_{i=1}^{n} w_i x_i - \theta \begin{cases} \geq 0 & , \text{ ako je } x \in X_+ \\ < 0 & , \text{ ako je } x \in X_- \end{cases},$$

to jest ako i samo ako je X_+ afino separabilan od X_-. ■

Dakle, funkcija $g \circ s$, to jest McCulloch-Pitts neuron, dijeli elemente ulaznog prostora X na dvije klase odvojene hiperravninom danom pravilom $\sum_{i=1}^{n} w_i x_i = \theta$.

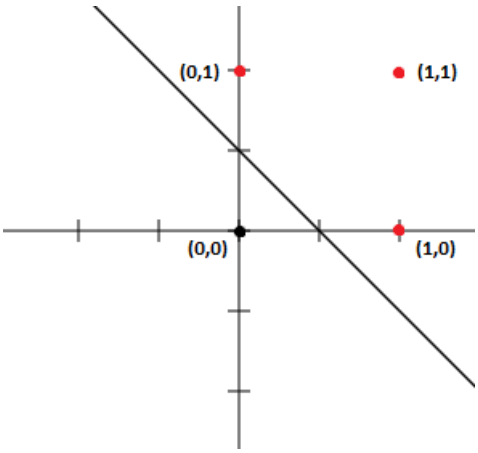
Primjer 1.10 *Neka je dana step funkcija g i logička NOT funkcija f tablicom*

x	$-x$
0	*1*
1	*0*

Funkciju f možemo realizirat pomoću jednostavnog perceptrona sa ulazom x, težinom $w = -1$ i pragom $\theta = -\frac{1}{2}$, to jest $f(x) = g(-x + \frac{1}{2})$.

Primjer 1.11 *Neka je dana step funkcija g i logička OR funkcija f tablicom*

x_1	x_2	$x_1 \vee x_2$
0	*0*	*0*
0	*1*	*1*
1	*0*	*1*
1	*1*	*1*



Slika 1.2: Linearna separabilnost za OR

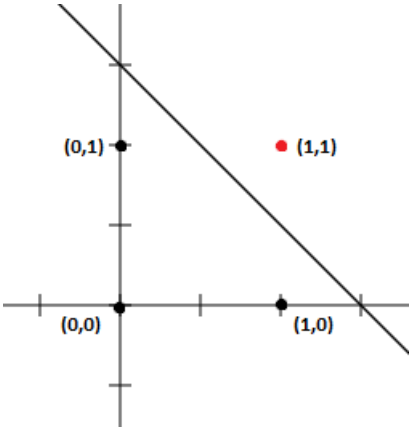
Funkciju f možemo realizirati pomoću jednostavnog perceptrona sa ulazima x_1 i x_2, težinama $w_1 = 1$ i $w_2 = 1$ te pragom $\theta = \frac{1}{2}$, to jest $f(x_1, x_2) = g(x_1 + x_2 - \frac{1}{2})$. Na Slici 1.2 prikazana je linearna separabilnost skupa $X_+ =$

$\{(1,0),(1,1),(0,1)\}$ *od skupa* $X_- = \{(0,0)\}$ *pravcem (hiperravninom)* $y = -x + \frac{1}{2}$.

Primjer 1.12 *Neka je dana step funkcija* g *i logička AND funkcija* f *tablicom*

x_1	x_2	$x_1 \wedge x_2$
0	*0*	*0*
0	*1*	*0*
1	*0*	*0*
1	*1*	*1*

Funkciju f *možemo realizirati pomoću jednostavnog perceptrona sa ulazima* x_1 *i* x_2*, težinama* $w_1 = 1$ *i* $w_2 = 1$ *te pragom* $\theta = \frac{3}{2}$*, tj.* $f(x_1, x_2) = g(x_1 + x_2 - \frac{3}{2})$. *Na Slici 1.3 prikazana je linearna separabilnost skupa* $X_+ = \{(1,1)\}$ *od skupa* $X_- = \{(0,1),(0,0),(1,0)\}$ *pravcem (hiperravninom)* $y = -x + \frac{3}{2}$.



Slika 1.3: Linearna separabilnost za AND

1.3 Jednostavna neuronska mreža

Uvedimo još neke osnovne pojmove iz teorije grafova:

Definicija 1.13 *Neka je $G = (V, E, \phi)$ graf. Skup $P(i) = \{j \in V : (j, i) \in E\}$ nazivamo skupom svih* ***direktnih prethodnika*** *vrha $i \in V$, a skup $S(i) = \{j \in V : (i, j) \in E\}$ nazivamo skupom svih* ***direktnih sljedbenika*** *vrha $i \in V$. Vrh $i \in V$ za koji je $P(i) = \emptyset$ nazivamo* ***izvorištem grafa****, a vrh $j \in V$ za koji je $S(j) = \emptyset$ nazivamo* ***ponorištem grafa****.*

Definicija 1.14 *Neka je $G = (V, E, \phi)$ graf i neka su $i_0, \ldots, i_l \in V$ te neka je $e_j = (i_{j-1}, i_j) \in E$, za $j = 1, \ldots, l$. Uređenu l-torku $(e_1, \ldots, e_l)$ nazivamo* ***šetnjom*** *od i_0 do i_l, a broj bridova l nazivamo* ***duljinom*** *šetnje. Šetnju takvu da je $i_0 = i_l$ nazivamo* ***ciklusom****. Graf koji nema cikluse nazivamo* ***acikličkim*** *grafom.*

Na Slici 1.1, g-perceptron se sastoji samo od ulaznog i izlaznog sloja te izlazni sloj samo od jednog neurona. Ako izlazni sloj ima više neurona, nastali objekt zovemo jednostavnom neuronskom mrežom, a za njeno uvođenje nam je potreban sljedeći teorem.

Teorem 1.15 *Svaki aciklički graf sadrži izvorište i ponorište.*

Dokaz. Neka je $G = (V, E, \phi)$ aciklički graf. Pretpostavimo suprotno, to jest da V ne sadrži nijedno izvorište. Tada za svaki vrh $i \in V$ vrijedi da je $P(i) \neq \emptyset$, to jest svaki vrh ima direktnog prethodnika pa prema tome možemo pronaći šetnju proizvoljne duljine. Neka je $l \in \mathbb{N}$ takav da je $l > card(E)$ i neka je $(e_1, \ldots, e_l)$ šetnja duljine l. Zbog uvjeta $l > card(E)$, to postoje $1 \leq i < j \leq l$ takavi da je $e_i = e_j$, no tada je šetnja $(e_i, \ldots, e_{j-1})$ ciklus što je kontradikcija s pretpostavkom da je graf G aciklički. Dualna tvrdnja za ponorište dokazuje se potpuno analogno. ■

Definicija 1.16 *Neka je $G = (V, E, \phi)$ aciklički graf i neka je $V = V_0 \cup V_1$, gdje je V_0 skup svih izvorišta grafa G, a $V_1 = V \setminus V_0$.* ***Jednostavnu***

neuronsku mrežu $\mathscr{F}$ *čine graf* G *i familija* $((X_i, Y_i, g_i, s_i) : i \in V_1)$ g_i*-perceptrona, gdje za svaki* $i \in V_1$ *jest* $X_i \subseteq \mathbb{R}^{n_i}$*, za* $n_i = card(P(i))$*,* $Y_i \subseteq \mathbb{R}$ *te*

$$s_i : X_i \to \mathbb{R}$$

integralna funkcija iz Definicije 1.5 i

$$g_i : \mathbb{R} \to Y_i$$

aktivacijska funkcija tako da je:

1. $\prod_{j \in P(i)} Y_j \subseteq X_i$*, za svaki* $i \in V_1$
2. $i \in V_0 \Rightarrow S(i) \subseteq V_1$
3. $i \in V_1 \Rightarrow P(i) \subseteq V_0$*,* $S(i) = \emptyset$

Za elemente skupa V_0 kažemo da čine **ulazni sloj**, a za elemente skupa V_1 kažemo da čine **izlazni sloj**. Primjetimo da po Teoremu 1.15 vrijedi $V_0 \neq \emptyset$ i $V_1 \neq \emptyset$.

Napomena 1.17 *Uvjet kompatibilnosti (uvjet 1.) iz definicije 1.16 bit će zadovoljen ako stavimo*

$$X_i = Q^{n_i},\ Y_i = Q,$$

za svaki $i \in V_1$ *i za proizvoljni* $Q \subseteq \mathbb{R}$*, pa će to ubuduće i biti slučaj.*
Nadalje, promatrat ćemo samo slučajeve kod kojih će u uvjetima 2. i 3. definicije 1.16 vrijediti skupovne jednakosti. Naime, to će inutitivno značiti da će svi neuroni iz ulaznog sloja biti povezani sa svim neuronima iz izlaznog sloja. Ukidanje neke veze formalno ćemo rješavati postavljanjem njene težine na nulu. No, u tom slučaju uvjet $X_i \subseteq R^{n_i}$ *također prelazi u* $X_i \subseteq R^n$*, gdje je* $n = card(V_0)$*.*

Kao i u g-perceptrona, jednostavna neuronska mreža ima ulaz x predstavljen sa $(x_1, \ldots, x_n)$, no izlaz y je ovdje predstavljen sa $(y_1, \ldots, y_m)$. Svaki element izlaza y_i zajedno sa $(x_1, \ldots, x_n)$ čini jedan g_i-perceptron. Prema tome, svaki y_i ima jednoznačno određene sinaptičke težine i parametar. Parametar od y_i ćemo značavati sa θ_i, a sinaptičku težinu brida (x_j, y_i) sa w_{ji}.

Napomena 1.18 *Često se u literaturi namijenjenoj tehničkoj svrsi g-perceptron, McCulloch-Pitts neuron i jednostavna neuronska mreža jednim imenom nazivaju jednostavnim perceptronom.*

1.4 Višeslojni perceptron

Dosada smo se bavili proučavanjem jednostavnih neuronskih mreža koje su imale samo dva sloja - jedan ulazni i jedan izlazni. Međutim, postoje funkcije koje takve mreže ne mogu realizirati, na primjer, logička funkcija XOR dana tablicom:

x_1	x_2	$x_1 \overline{\vee} x_2$
0	0	0
0	1	1
1	0	1
1	1	0

Teorem 1.19 *Ne postoji jednostavni perceptron koji realizira logičku funkciju XOR.*

Dokaz. Pretpostavimo suprotno, to jest da postoje $w_1, w_2, \theta \in \mathbb{R}$ takvi da je za svaki $(x_1, x_2) \in \{0, 1\}^2$

$$x_1 \overline{\vee} x_2 = g(w_1 x_1 + w_2 x_2 - \theta),$$

gdje je g step funkcija.

Tada imamo

$$0 = 0\overline{\vee}0 = g(-\theta) \Rightarrow -\theta < 0$$

$$1 = 0\overline{\vee}1 = g(w_2 - \theta) \Rightarrow w_2 - \theta \geq 0$$

$$1 = 1\overline{\vee}0 = g(w_1 - \theta) \Rightarrow w_1 - \theta \geq 0$$

$$0 = 1\overline{\vee}1 = g(w_1 + w_2 - \theta) \Rightarrow w_1 + w_2 - \theta < 0$$

Zbrajanjem druge i treće nejednakosti dobijamo

$$w_1 + w_2 - 2\theta \geq 0,$$

a zbrajanjem prve i četvrte nejednakosti dobijamo

$$w_1 + w_2 - 2\theta < 0$$

što je kontradikcija.

Dakle, ne postoji jednostavni perceptron koji realizira logičku funkciju XOR.

■

S obzirom da jednostavnim perceptronom ne možemo realizirati logičku funkciju XOR, a istu ne možemo realizirati ni jednostavnom neuronskom mrežom — naime, izlaz je 0 ili 1 pa nam je potreban samo jedan vrh u izlaznom sloju — pojavljuje se potreba za složenijom strukturom.

Neka je $G = (V, E, \phi)$ aciklički graf. Tada postoji cijeli broj $k \geq -1$ i particija od V

$$V = V_0 \cup \ldots \cup V_{k+1}$$

konstruirana na sljedeći način: Neka V_0 sadrži sva izvorišta grafa G. Po Teoremu 1.15 vrijedi $V_0 \neq \emptyset$. Uklonimo iz skupa E sve elemente $e \in E$ takve da postoje $v_1 \in V_0$ i $v_2 \in V \setminus V_0$ takvi da je $e = (v_1, v_2)$ te iz skupa V sve elemente $v \in V_0$. Dobiveni graf je ponovno acikličan pa ponovimo postupak.

Definicija 1.20 *Neka je $G = (V, E, \phi)$ aciklički graf i neka je $V = V_0 \cup \ldots \cup V_{k+1}$ prethodno dobivena particija.* ***K-slojnu neuronsku mrežu*** $\mathscr{F}$ *čine graf G i familija $((X_i, Y_i, g_i, s_i) : i \in V \setminus V_0)$ g_i-perceptrona, gdje za svaki $i \in V \setminus V_0$ jest $X_i \subseteq \mathbb{R}^{n_i}$, za $n_i = card(P(i))$, $Y_i \subseteq \mathbb{R}$,*

$$s_i : X_i \to \mathbb{R}$$

integralna funkcija iz Definicije 1.5 i

$$g_i : \mathbb{R} \to Y_i$$

aktivacijska funkcija tako da vrijedi:

1. $\prod_{j \in P(i)} Y_j \subseteq X_i$, *za svaki* $i \in V \setminus V_0$,
2. $i \in V_j \Rightarrow P(i) \subseteq V_{j-1}$ *i* $S(i) \subseteq V_{j+1}$, *za svaki* $j = 0, \ldots, k+1$, *gdje je* $V_{-1} = V_{k+2} = \emptyset$.

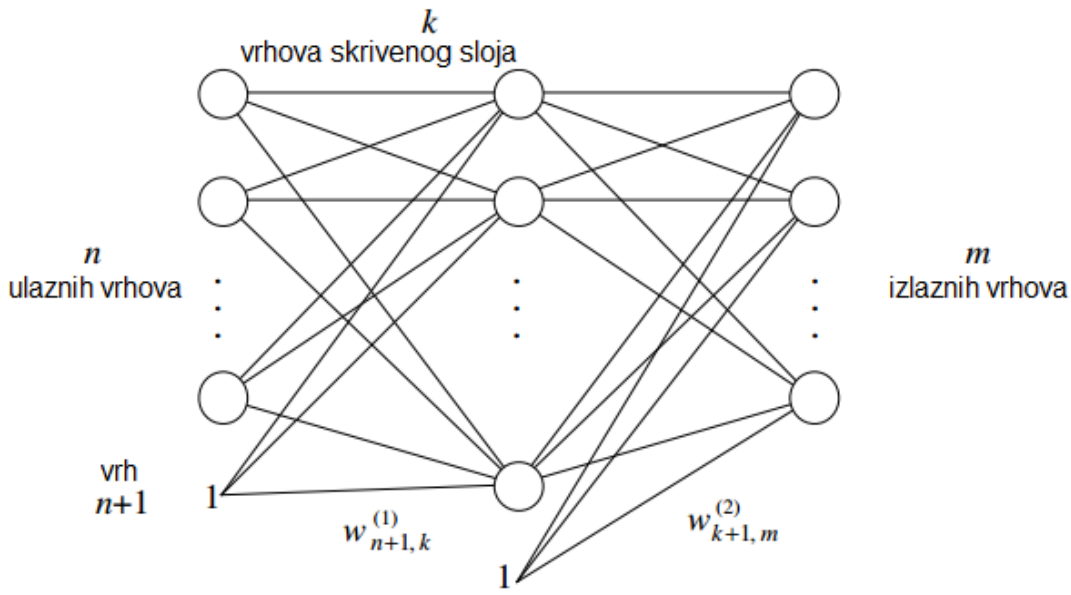
Skup V_j općenito ćemo zvati **j-tim slojem**. Posebno, skup V_0 ćemo zvati **ulaznim**, skup V_{k+1} **izlaznim slojem** te skupove $V_1, \ldots, V_k$ **skrivenim slojevima** (engl. *hidden layer*). Primjetimo da po Teoremu 1.15 i konstrukciji vrijedi $V_j \neq \emptyset$, za svaki $j \in \{0, \ldots, k+1\}$.

Napomena 1.21 *Uvjet kompatibilnosti (uvjet 1.) iz definicije 1.20 bit će zadovoljen ako stavimo*

$$X_i = Q^{n_i},\ Y_i = Q,$$

za svaki $i \in V_1$ i za proizvoljni $Q \subseteq \mathbb{R}$, pa će to ubuduće i biti slučaj.
Nadalje, promatrat ćemo samo slučajeve kod kojih će u uvjetima 2. i 3. definicije 1.20 vrijediti skupovne jednakosti. Naime, to će inutitivno značiti da će svi neuroni iz svakog sloja sloja biti povezani sa svim neuronima iz sljedećeg sloja (ako ga ima). Ukidanje neke veze, formalno ćemo rješavati postavljanjem njene težine na nulu.

Napomena 1.22 *K-slojna neuronska mreža još se naziva i k-slojni ili višeslojni perceptron.*



Slika 1.4: Jednoslojna neuronska mreža [7]

Promotrimo sada jednoslojnu neuronsku mrežu, danu Slikom 1.4. Osim ulaznog i izlaznog sloja, ova mreža ima još jedan skriveni sloj. Radi sugestivnosti oznaka, ulaz mreže ćemo označavati sa $o = (o_1, \ldots, o_n)$, izlaz prvog sljedećeg sloja sa $o^{(1)}$, i tako dalje. Za svaki $i \in V \setminus V_0$ neka je, od sada pa nadalje, osim ako nije naglašeno drugačije,

$$g_i(x) = g(x) = \frac{1}{1 + e^{-x}}.$$

Ovu funkciju odabiremo zbog njene neprekidnosti, diferencijabilnosti te jednostavne derivacije, o čemu će više riječi biti kasnije. Težinu brida između i-tog vrha ulaznog sloja i j-tog vrha skrivenog sloja označavat ćemo sa $w^{(1)}_{ij}$, a težinu brida između j-tog vrha skrivenog sloja i l-tog vrha izlaznog sloja označavat ćemo sa $w^{(2)}_{jl}$. Navedene težine predstavljat će parametre integralne

funkcije s. Prag podražljivosti prikazat ćemo također kao sinaptičku težinu. U tu svrhu, skup ulaznih i skrivenih vrhova proširit ćemo svaki sa jednim vrhom čija će vrijednost biti 1. Ako je broj ulaznih vrhova n, a skrivenih k, tada je vrijednost $o_{n+1} = 1$ te $o_{k+1}^{(1)} = 1$. Uvest ćemo oznake $\hat{o} = (o_1, \ldots, o_n, 1)$ i $\hat{o}^{(1)} = (o_1^{(1)}, \ldots, o_k^{(1)}, 1)$.

Neka je $j \in \{1, \ldots, k\}$. Imamo: .

$$\begin{aligned} o_j^{(1)} &= g\left(\sum_{i=1}^{n} w_{ij}^{(1)} o_i - \theta_j\right) = \\ &= g\left(\sum_{i=1}^{n} w_{ij}^{(1)} o_i + (-\theta_j)1\right) = \\ &= \{w_{n+1,j}^{(1)} := -\theta_j\} = \\ &= g\left(\sum_{i=1}^{n+1} w_{ij}^{(1)} o_i\right). \end{aligned}$$

Analogan račun možemo provesti za $o_l^{(2)}$, za svaki $l \in \{1, \ldots, m\}$, uz $w_{k+1,l}^{(2)} = -\theta_l$ pa imamo

$$o_l^{(2)} = g\left(\sum_{j=1}^{k+1} w_{jl}^{(2)} o_j^{(1)}\right) = g\left(\sum_{j=1}^{k+1} w_{jl}^{(2)} g\left(\sum_{i=1}^{n+1} w_{ij}^{(1)} o_i\right)\right).$$

Nadalje, neka je $\overline{W}_1$ matrica takva da je

$$[\overline{W}_1]_{(i,j)} = w_{ij}^{(1)} \text{ za } i = 1, \ldots, n+1,\ j = 1, \ldots, k,$$

i neka je W_1 matrica takva da je

$$[W_1]_{(i,j)} = w_{ij}^{(1)} \text{ za } i = 1, \ldots, n,\ j = 1, \ldots, k.$$

Još, neka je $\overline{W}_2$ matrica takva da je

$$[\overline{W}_2]_{(i,j)} = w_{ij}^{(2)} \text{ za } i = 1, \ldots, k+1,\ j = 1, \ldots, m,$$

i neka je W_2 matrica takva da je

$$[W_2]_{(i,j)} = w_{ij}^{(2)} \text{ za } i = 1, \ldots, k,\ j = 1, \ldots, m.$$

Sada imamo:

$$o^{(1)} = g(\hat{o}\overline{W}_1)$$

i

$$o^{(2)} = g(\hat{o}^{(1)}\overline{W}_2).$$

Primjer 1.23 *XOR funkciju možemo realizirati kao jednoslojnu neuronsku mrežu. XOR funkcija dana je logičkom tablicom*

x_1	x_2	$x_1 \overline{\vee} x_2$
0	*0*	*0*
0	*1*	*1*
1	*0*	*1*
1	*1*	*0*

Definiramo li vrhove skrivenog sloja sa

$$y_1 = g(-x_1 + x_2 - \frac{1}{2}),$$

$$y_2 = g(x_1 - x_2 - \frac{1}{2}),$$

i vrhove izlaznog sloja sa

$$z = g(y_1 + y_2 - \frac{1}{2}),$$

gdje je g step funcija, dobijamo mrežu koja realizira funkciju XOR.

Poglavlje 2

Proces učenja

U uvodnom poglavlju postavili smo zanimljiv problem — želimo *naučiti* računalo prepoznavati rukom napisane brojeve u rasponu $0 - 9$. U prethodnom poglavlju, inspirirani građom i funkcijom neurona u ljudskom živčanom sustavu koji su zaduženi za učenje i koje želimo oponašati, definirali smo neuronsku mrežu. Sumirano, naš cilj je ustvari naučiti neuronsku mrežu klasificirati dani ulaz u jednu od 10 klasa. U teoriji računarstva, **učenje neuronske mreže** podrazumijeva proces promjene parametara mreže (sinaptičkih težina) pod utjecajem okoline koji onda rezultira drugačijim ponašanjem mreže. Već smo spomenuli jednu paradigmu učenja koja određuje odnos neuronske mreže prema okolini — **učenje pod nadzorom**. Tu je glavna ideja da je dan skup

$$T = \{(x_i, t_i) : 1 \leq i \leq p\}$$

od konačno mnogo ulaza i njima pridruženih odgovarajućih izlaza na temelju kojih se moraju prilagoditi parametri mreže tako da daju ispravan odgovor na nepoznati primjer. Taj skup se zove **skup (podataka) za treniranje**. Osim ove, ostale paradigme učenja su učenje bez nadzora i učenje s podrškom, no

u njihovo proučavanje nećemo ulaziti jer bi to prešlo okvire ovog rada. Dakle, srž problema je ispitati na koji način trebamo prilagođavati sinaptičke težine kako bismo dobili željeni rezultat?

2.1 Funkcija gubitka

Neka je dan višeslojni perceptron s proizvoljno odabranim težinama w_j i skup za treniranje $T = \{(x_i, t_i) : 1 \leq i \leq p\}$. Dovedemo li element x_i, za neki $i \in \{1, \ldots, p\}$, na ulazne vrhove mreže dobijamo izlaz mreže koji ćemo označavati sa o_i i koji je općenito različit od t_i. Kao kriterij promjene sinaptičkih težina nameće se najveće moguće smanjenje razlike elemenata o_i i t_i. U tu svrhu, potrebno je definirati funkciju koju ćemo zvati **funkcija gubitka** ili **funkcija greške** (engl. *loss function* ili *error function*) koja će formalno izražavati razliku dobivenog izlaza o_i od željenog izlaza t_i i biti funkcija sinaptičkih težina w_j. Jedna takva funkcija koju ćemo koristiti u ovom radu je

$$E = \sum_{i=1}^{p} E_i,$$

gdje je

$$E_i = \frac{1}{2} \|o_i - t_i\|^2 .$$

Osim ovako definirane funkcije E_i običavaju se koristiti i sljedeće:

1. $E_i = max(0, 1 - o_i t_i)$
2. $E_i = \log(1 + e^{-o_i t_i})$

Ne postoji univerzalna funkcija koja predstavlja najbolji izbor za svaki problem. Odabir funkcije mijenja se promjenom prirode promatranog problema.

2.2 Metoda gradijentnog spusta

Ključan korak u procesu učenja neuronske mreže je određivanje globalnog minimuma funkcije klase C^2

$$f : \mathbb{R}^n \to \mathbb{R}$$

jer, kako je prethodno rečeno, želimo minimizirati E koja je realna funkcija svih sinaptičkih težina mreže. U praksi se najčešće, zbog vremenske i memorijske povoljnosti, koriste metode iterativnog tipa. Osnovna ideja tih metoda je generiranje niza točaka $(x_k)_{k\in\mathbb{N}\cup\{0\}}$ tako da vrijedi

$$x_{k+1} = x_k + \alpha_k d_k, \text{ za svaki } k \in \mathbb{N} \cup \{0\},$$

gdje je $x_0 \in \mathbb{R}^n$ proizvoljna točka, $d_k \in \mathbb{R}^n$ vektor koji određuje smjer kretanja iz točke x_k, a $\alpha_k > 0$ realni parametar koji određuje duljinu koraka iz točke x_k u smjeru vektora d_k. Niz vektora $(d_k)_{k\in\mathbb{N}\cup\{0\}}$ i parametara $(\alpha_k)_{k\in\mathbb{N}\cup\{0\}}$ obično biramo tako da je niz $(f(x_k))_{k\in\mathbb{N}\cup\{0\}}$ monotono padajući te da konvergira nekom od lokalnih minimuma funkcije f, po mogućnosti baš globalnom minimumu.

Neka je $x \in \mathbb{R}^n$ proizvoljna točka i $\nabla f(x)$ gradijent funkcije f u točki x. Promatrajmo $\langle \nabla f(x), d\rangle$ za $\|d\| = 1$. Jer je $\|d\| = 1$, prema Cauchy-Schwartzovoj nejednakosti je

$$\langle \nabla f(x), d\rangle \leq \|\nabla f(x)\| .$$

Ako stavimo $d = \frac{\nabla f(x)}{\|\nabla f(x)\|}$, dobijamo

$$\langle \nabla f(x), \frac{\nabla f(x)}{\|\nabla f(x)\|}\rangle = \|\nabla f(x)\| .$$

Dakle, $\nabla f(x)$ je smjer najvećeg rasta funkcije f u točki x pa je prema tome $-\nabla f(x)$ smjer najvećeg pada funkcije f u točki x, to jest minimum funkcije

ćemo tražiti u smjeru negativnog gradijenta i tu iterativnu metodu zvati **metoda gradijentnog spusta**.

Algoritam metode je sljedeći:

Neka je $x_0 \in \mathbb{R}^n$ proizvoljna početna točka i $\alpha_0 > 0$ realni broj. Prema Taylorovom teoremu vrijedi

$$f(x_0 - \alpha_0 \nabla f(x_0)) = f(x_0) - \alpha_0 \left\|\nabla f(x_0)\right\|^2 + R_1(-\alpha_0 \nabla f(x_0)),$$

gdje je $R_1(-\alpha_0 \nabla f(x_0))$ Lagrangeov oblik ostatka funkcije f u točki x_0. Ako je $\nabla f(x_0) \neq 0$, onda za dovoljno mali $\alpha_0 > 0$ imamo

$$f(x_0 - \alpha_0 \nabla f(x_0)) < f(x_0),$$

to jest točka $x_1 = x_0 - \alpha_0 \nabla f(x_0)$ predstavlja poboljšanje u odnosu na točku x_0 u smislu traženja minimuma. Pretpostavimo sada da smo u k-tom koraku, za neki $k \in \mathbb{N}$ pronašli točku x_k. Kako bismo pronašli točku x_{k+1} od točke x_k odmičemo u smjeru vektora $-\nabla f(x_k)$ za konkretan iznos od $\alpha_k \nabla f(x_k)$, gdje je $\alpha_k > 0$ dovoljno mali realan broj nazvan veličina koraka, to jest

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k).$$

Time je algoritam završen.

Metoda najbržeg spusta i metoda konstantnog spusta su podvrste metode gradijentnog spusta. Kod metode najbržeg spusta, veličina koraka α_k bira se tako da je u svakoj iteraciji opadanje funkcije f maksimalno moguće. Formalno,

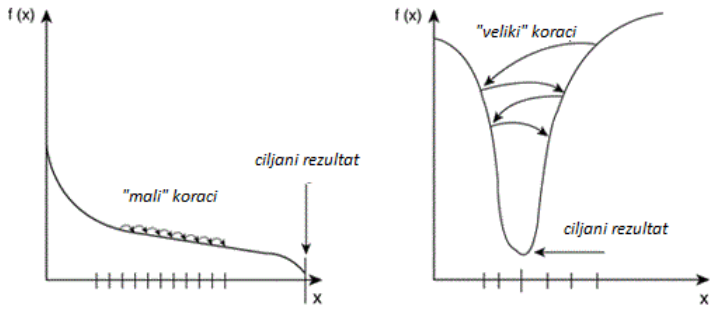
$$\alpha_k = \underset{\alpha \geq 0}{arg\ min} f(x_k - \alpha \nabla f(x_k))$$

Dakle, navedena metoda počinje od proizvoljne točke x_0, a zatim se u svakom k-tom koraku bira točku na polupravcu smjera $-\nabla f(x_k)$ s početkom u x_k za koju je vrijednost funkcije najmanja moguća. Ako je za svaki $k \in \mathbb{N} \cup \{0\}$

ispunjeno $\alpha_k = \alpha$, za neki unaprijed dani i fiksirani $\alpha > 0$, onda se metoda najbržeg spusta svodi na metodu konstantnog spusta. Dakle, u potonjoj, za proizvoljnu točku $x_0 \in \mathbb{R}^n$, k-ti korak iteracije je dan sa

$$x_{k+1} = x_k - \alpha \nabla f(x_k),$$

za svaki $k \in \mathbb{N} \cup \{0\}$. Metoda konstantnog spusta je uglavnom jednostavnija za implementaciju od metode najbržeg spusta, no najčešće zahtjeva više iteracija te u slučaju da veličina koraka α nije pogodno odabrana, ne mora nužno dovesti do pronalaska minimuma. U tom slučaju, najčešće se algoritam provodi nekoliko puta — s različitim početnim točkama x_0 i veličinama koraka α — pa se između dobivenih rješenja svih ponavljanja bira ono za koje je vrijednost funkcije najmanja. Što se tiče odabira veličine koraka α, matematičari je dijele u dvije sugestivne kategorije — „mali" α i „veliki" α. Što se očekuje od veličine koraka α ovisno o kategoriji prikazuje Slika 2.1.



Slika 2.1: Različite veličine koraka [15]

Važno je napomenuti da ponavljanje algoritama, mijenjanje početnih točaka i veličina koraka ne mora dovesti do pronalaska točke u kojem se postiže globalni minimum funkcije f. Upravo je u tome težina i ljepota ove metode. Još, s obzirom na vremensku i prostornu ograničenost, potrebno je definirati

kriterije zaustavljanja kontrukcije niza $(x_k)_{k\in\mathbb{N}\cup\{0\}}$.
Primjeri nekih kriterija su:

1. $\|\nabla f(x_k)\| \leq \epsilon$, za neki unaprijed dani $\epsilon > 0$,
2. $|f(x_{k+1}) - f(x_k)| \leq \epsilon$, za neki unaprijed dani $\epsilon > 0$,
3. $\|x_{k+1} - x_k\| < \epsilon$, za neki unaprijed dani $\epsilon > 0$.

2.3 Algoritam s povratnim proslijeđivanjem pogreške

Vratimo se sad srži problema — problemu učenja, to jest načinu prilagođavanja sinaptičkih težina kako bismo dobili željeni rezultat. **Algoritmom učenja** nazivat ćemo način izračunavanja promjena težina, Δw_i, u svrhu dobijanja težina koje minimiziraju funkciju greške E, a te težine nazvat ćemo **rješenjem problema učenja**. Kod učenja pod nadzorom najčešći algoritmi su LMS algoritam ili algoritam najmanjih kvadrata (engl. *Least mean square algorithm*) i BP algoritam ili **algoritam s povratnim proslijeđivanjem pogreške** (engl. *Backpropagation algorithm*). U ovom ćemo poglavlju BP algoritmom tražiti minimum funkcije greške uz pomoć *metode gradijentnog spusta*. Način traženja bit će grafičke prirode i prednost pred matematičkim formalizmom davat će kratkoći i slikovitosti. Ovaj pristup je zanimljiv i popularan jer usputno sugerira implementaciju algoritma u sklopovlje računala.
Već smo naveli da ćemo za funkciju greške za dani skup za treniranje

$$T = \{(x_i, t_i) : 1 \leq i \leq p\},\ p \in \mathbb{N}$$

gdje je $x_i = (x_{i1}, \ldots, x_{in})$, $t_i = (t_{i1}, \ldots, t_{im})$, n,$m \in \mathbb{N}$, koristiti funkciju

$$E = \sum_{i=1}^{p} E_i,$$

gdje je

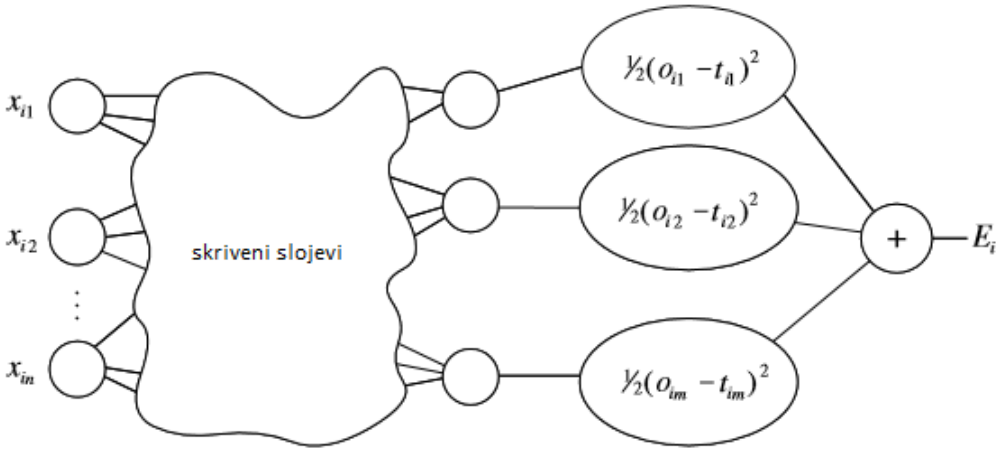
$$E_i = \frac{1}{2} \|o_i - t_i\|^2,$$

a za aktivacijsku funkciju

$$g(x) = \frac{1}{1 + e^{-x}}.$$

Ovi odabiri vrijede nadalje.

2.3.1 Arhitektura mreže



Slika 2.2: Proširena mreža za računanje funkcije greške [7]

Prvi korak je proširiti danu neuronsku mrežu vrhovima i bridovima tako da za ulaz

$$x = (x_1, \ldots, x_p), \text{ gdje je } x_i = (x_{i1}, \ldots, x_{in}), \text{ za svaki } i \in \{1, \ldots, p\}$$

izlaz mreže bude

$$E = \frac{1}{2} \sum_{i=1}^{p} \|o_i - t_i\|^2 = \sum_{i=1}^{p} E_i,$$

to jest implementirati funkciju greške u neuronsku mrežu. U tu svrhu, izlazni vrh koji računa o_{ij}, j-tu komponentu izlaza $o_i = (o_{i1}, \ldots, o_{im})$, spojimo

bridom sa vrhom koji računa funkciju $\frac{1}{2}(o_{ij} - t_{ij})^2$, za svaki $j \in \{1, \ldots, m\}$, te sve tako dodane vrhove — koje ćemo ubuduće zvati *vrhovima proširenog sloja* — spojimo bridom s vrhom čija je pripadna aktivacijska funkcija funkcija zbrajanja m varijabli. Posljednji dodani vrh, dakle, računa

$$E_i = \frac{1}{2}(o_{i1} - t_{i1})^2 + \ldots + \frac{1}{2}(o_{im} - t_{im})^2,$$

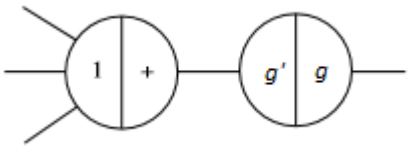
baš kako prikazuje Slika 2.2. Nadalje, ponovimo postupak p puta i izlazne vrhove svih p tako dobivenih mreža spojimo bridovima u vrh čija je pripadna aktivacijska funkcija funkcija zbrajanja p varijabli. Ovako je konstruirano traženo proširenje mreže.

Sada, po metodi gradijentnog spusta, znamo da možemo minimizirati E iterativno mijenjajući sve sinaptičke težine mreže označene sa w_i za iznos

$$\Delta w_i = -\alpha \frac{\partial E}{\partial w_i},$$

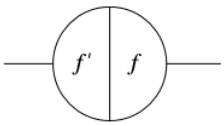
gdje je $\alpha > 0$ unaprijed odabrana veličina koraka te da ćemo za to trebati izračunati gradijent funkcije E.

Dakle, sada se problem učenja svodi na problem računanja gradijenta funkcije E, to jest na računanje gradijenta funkcije koju predstavlja proširena neuronska mreža.



Slika 2.3: Podjela jednog vrha na dva vrha [7]

Za rješavanje problema definirajmo **B-dijagram** (od engl. *Backpropagation diagram*) koji podrazumijeva proširenje neuronske mreže u građi i



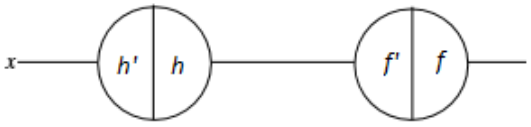
Slika 2.4: Podjela vrha na dva dijela [7]

funkciji opisanim postupkom koji slijedi. Prvo, svaki vrh koji pripada izlaznom sloju ili nekom od skrivenih slojeva, rastavimo na dva **vrha** tako da jedan bude pridružen integralnoj funkciji s, a drugi aktivacijskoj funkciji g. Sada, kada su vrhovi i pridružene funkcije u 1-1 korespodenciji, rastavimo svaki novi vrh na dva **dijela**, lijevi i desni. Neka se u desni dio pohranjuje vrijednost pridružene funkcije na ulazu koji je sdesna doveden u vrh, a u lijevom dijelu neka se pohranjuje vrijednost derivacije pridružene funkcije na spomenutom ulazu. Ubuduće, kada iz konteksta bude jasno da nije potrebno pohranjivati vrijednost u desni dio vrha, nećemo to ni raditi bez prethodnog naglašavanje. Dalje, kada je ulaz doveden sdesna u vrh, neka je izlaz vrha vrijednost pridružene funkcije na tom ulazu. Ovaj korak zovemo *korak proslijeđivanja unaprijed*. Kada je ulaz doveden slijeva u vrh, neka je izlaz umnožak ulaza i vrijednosti pohranjene u lijevom dijelu vrha. Ovaj korak zovemo *korak proslijeđivanja unazad*. Nadalje, ukoliko u koraku proslijeđivanja unazad jedan vrh ima više ulaznih bridova, neka se svi ulazi zbroje prije množenja sa vrijednosti pohranjenoj u lijevom dijelu vrha. Primjetimo da bridovi svakog vrha mijenjaju ulogu izlaza i ulaza ovisno o tome odvija li se protok informacija slijeva nadesno ili sdesna nalijevo, to jest ovisno o kojem od dva navedena koraka je riječ. Još, ako aktivacijska funkcija nekog vrha nije istaknuta, podrazumijevamo identitetu.

Spomenute promjene mreže u svrhu definicije B-dijagrama ilustrirane su na Slikama 2.3 i 2.4.

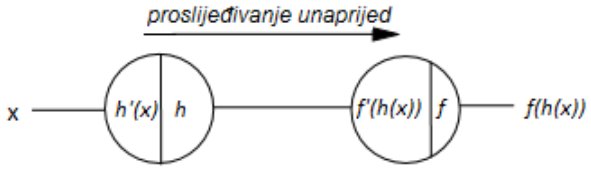
Kompozicija funkcija

Neka su sad dane dvije funkcije h i f i B-dijagram koji se sastoji od dva vrha, jednog pridruženog funkciji h i drugog pridruženog funkciji f, kao na Slici 2.5. Pokažimo da ovaj dijagram računa kompoziciju $(f \circ h)(x)$ i derivaciju te kompozicije $(f \circ h)'(x)$, za proizvoljni $x \in \mathbb{R}$.



Slika 2.5: Mreža za kompoziciju dviju funkcija [7]

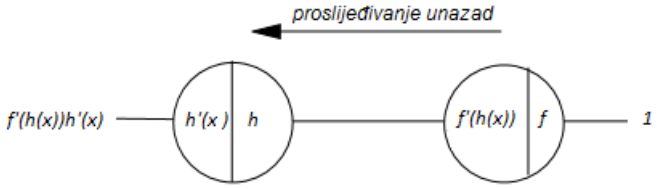
Dovedimo prozvoljni x kao ulaz u vrh pridružen funkciji h. Desna strana vrha računa $h(x)$, a lijeva računa i pohranjuje $h'(x)$ te se izlaz vrha $h(x)$, prema koraku proslijeđivanja unaprijed, proslijeđuje vrhu pridruženom funkciji f. Tu se postupak ponavlja, to jest desna strana vrha računa $f(h(x))$, lijeva računa i pohranjuje $f'(h(x))$, a traženi izraz $f(h(x))$ dobija se kao izlaz promatranog vrha. Stanje mreže nakon ovog koraka ilustrirano je na Slici 2.6.



Slika 2.6: Korak proslijeđivanja unaprijed za kompoziciju dviju funkcija [7]

Dovedimo sada konstantu 1 na brid koji je ulaz u vrh pridružen funkciji f. Naravno, taj brid smo u koraku proslijeđivanja unaprijed zvali izlazom vrha, a sada, u koraku proslijeđivanja unazad, zovemo ga ulazom navedenog vrha. Dalje, konstanta 1 množi se sa vrijednosti pohranjenoj u lijevom dijelu vrha,

to jest sa $f'(h(x))$ i proslijeđuje se unazad vrhu priduženom funkciji h. Tu se ponavlja postupak, to jest $f'(h(x))$ se množi sa $h'(x)$ jer je to vrijednost pohranjena u lijevom dijelu vrha pridruženog funkciji h. Taj se umnožak proslijeđuje unazad i dobijamo traženi rezultat, to jest $f'(h(x))h'(x)$, kako prikazuje Slika 2.7. Dakle, korak proslijeđivanja unazad primjenjuje pravilo za derivaciju kompozicije funkcija.



Slika 2.7: Korak proslijeđivanja unazad za kompoziciju dviju funkcija [7]

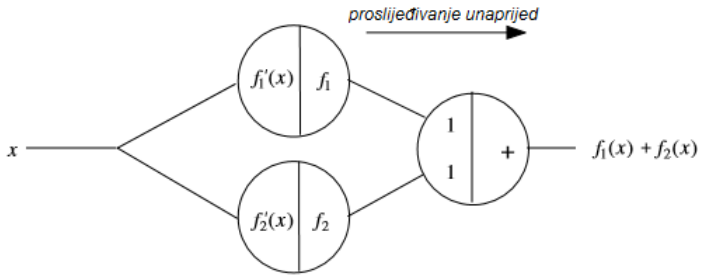
Ovdje smo pokazali kako konstruirati B-dijagram koji računa kompoziciju dviju funkcija i derivaciju kompozicije. Analogno se konstruira B-dijagram koji računa kompoziciju proizvoljnog broja funkcija i derivaciju te kompozicije.

Zbrajanje funkcija

Neka su sada dane dvije funkcije f_1 i f_2 i B-dijagram sastavljen od tri vrha od kojih je jedan pridružen funkciji f_1, drugi pridružen funkciji f_2 te posljednji pridružen funkciji zbrajanja dviju varijabli čija je parcijalna derivacija s obzirom na bilo koju varijablu jednaka 1. Pokažimo da ovaj dijagram računa zbroj $(f_1 + f_2)(x)$ i derivaciju $(f_1 + f_2)'(x)$, za proizvoljni $x \in \mathbb{R}$.

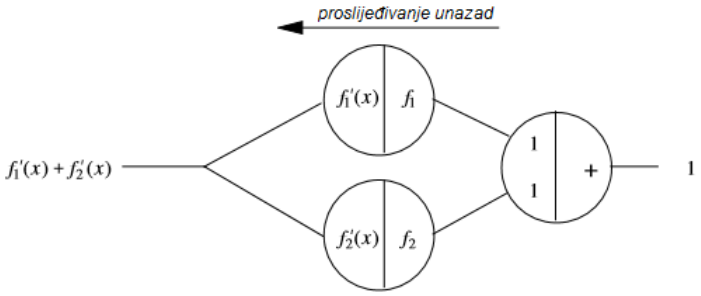
Dovedimo element x na ulaz mreže. Vrhovi pridruženi funkcijama f_1 i f_2 računaju vrijednosti funkcija i njihovih derivacija na x — vrijednosti derivacija funkcija $f_1'(x)$ i $f_2'(x)$ se pohranjuju u lijevim dijelovima vrhova, a vrijednosti funkcija $f_1(x)$ i $f_2(x)$ se proslijeđuju naprijed te se zbrajaju u posljednjem vrhu. Izlaz mreže je traženi $f_1(x) + f_2(x)$, kako prikazuje Slika

2.8.



Slika 2.8: Korak proslijeđivanja unaprijed za zbroj dviju funkcija [7]

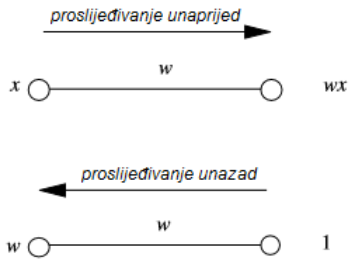
Dovedimo sada konstantu 1 na izlaz mreže. Ona se množi s obje vrijednosti pohranjenje u lijevom dijelu vrha pridruženog funkciji zbrajanja dvije varijable. Te se vrijednosti dalje proslijeđuju unazad, jedna prema vrhu pridruženom funkciji f_1, a druga prema vrhu pridruženom funkciji f_2, gdje se množe sa pohranjenim vrijednostima $f_1'(x)$ i $f_2'(x)$, respektivno. Kada se izlazni bridovi vrhova pridruženih funkcijama f_1 i f_2 (te iste bridove zovemo ulaznim bridovima u koraku proslijeđivanja unaprijed) sastaju, dobijamo vrijednost $f_1'(x) + f_2'(x)$, što je prikazano na Slici 2.9.



Slika 2.9: Korak proslijeđivanja unazad za zbroj dviju funkcija [7]

Ovdje smo pokazali kako konstruirati B-dijagram koji računa zbroj dviju funkcija i derivaciju zbroja. Analogno se konstruira B-dijagram koji računa zbroj proizvoljnog broja funkcija i derivaciju tog zbroja.

Množenje skalarom



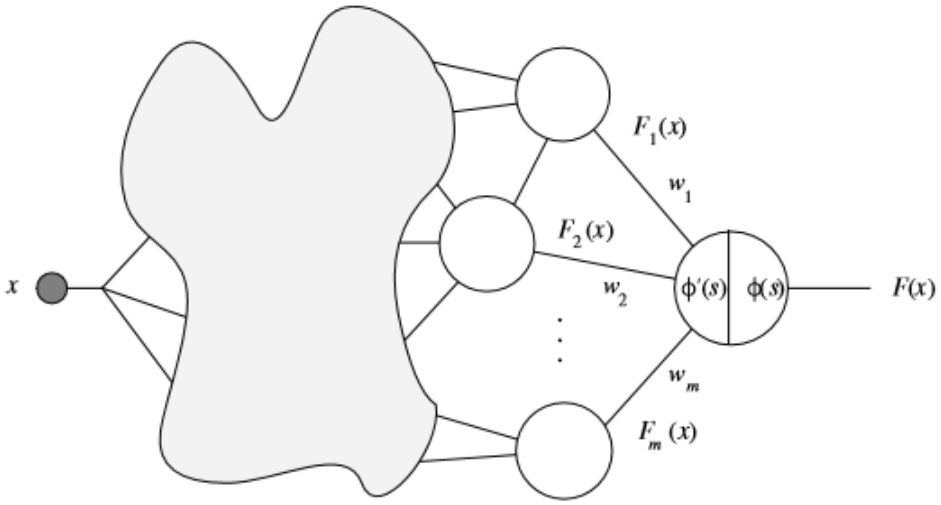
Slika 2.10: Korak proslijeđivanja unaprijed i unazad za množenje skalarom[7]

Neka je sada dana funkcija $r(x) = wx$, gdje su $x, w \in \mathbb{R}$ i B-dijagram koji se sastoji od dva vrha povezana bridom kojem ćemo pridružit težinu w. Pokažimo da ovaj dijagram računa funkciju $r(x)$ i njenu derivaciju $r'(x)$, za proizvoljni $x \in \mathbb{R}$. Iako ovo možemo riješiti kompozicijom funkcija, proširit ćemo svojstva B-dijagrama kako ne bismo morali mijenjati građu mreže.
Neka korak proslijeđivanja unaprijed množi izlaz prvog vrha x sa težinom brida w, a korak proslijeđivanja unazad neka množi konstantu 1 s težinom brida kako je prikazano na Slici 2.10. Pritom su aktivacijske funkcije vrhova identitete. Na ovaj način dobija se traženo, to jest $r(x)$ i $r'(x)$, za proizvoljni $x \in \mathbb{R}$.

Napomena 2.1 *S obzirom da sada znamo kako u mrežu implementirati kompoziciju funkcija, zbroj funkcija i množenje skalarom više nema potrebe za rastavljanjem vrhova grafa na dva vrha, jedan za funkciju s, a drugi za funkciju g pa to ubuduće nećemo ni raditi. Dakle, ako neki vrh ima n ulaznih bridova težina $w_1, \ldots, w_n$ koji prenose vrijednosti $x_1, \ldots, x_n$ i vrhu je pridružena funkcija $g \circ s$, izlaz mu je jednak*

$$g(w_1x_1 + \ldots + w_nx_n).$$

Teorem 2.2 *Neka je B-dijagramom dana neuronska mreža koja za ulaz i izlaz ima jednu realnu varijablu (ima samo jedan vrh u izlaznom sloju i samo jedan vrh u ulaznom sloju) i koja računa funkciju F. Neka je na ulaz mreže doveden element x i neka su provedeni koraci proslijeđivanja unaprijed i unazad. Tada na ulazu mreže dobijamo vrijednost $F'(x)$.*



Slika 2.11: Korak indukcije [7]

Dokaz. Dokaz provodimo matematičkom indukcijom po broju vrhova mreže. Po definiciji B-dijagrama, tvrdnja vrijedi za $n = 1$. Pretpostavimo da iskazana tvrdnja vrijedi za mrežu s n ili manje vrhova i neka je dana mreža s $n + 1$ vrhom. Provedimo korak proslijeđivanja unaprijed. Na izlazu mreže dobijamo $F(x)$ jer je mreža implementacija te funkcije. Neka je vrh izlaza pridružen funkciji ϕ i neka je broj vrhova povezanih bridom s vrhom izlaza jednak m te neka su izlazi tih vrhova $F_1(x), \ldots, F_m(x)$ i neka su težine tih bridova $w_1, \ldots, w_m$, kao na Slici 2.11. Sada, prema Napomeni 2.1 slijedi

$$F(x) = \phi(w_1 F_1(x) + \ldots + w_m F_m(x)).$$

Prema tome, zaključujemo da na ulazu mreže moramo dobiti

$$F'(x) = \phi'(s)(w_1 F_1'(x) + \ldots + w_m F_m'(x)),$$

gdje je $s := w_1 F_1(x) + \ldots + w_m F_m(x)$. Promatrajmo sada mrežu koja se sastoji od ulaznog vrha i svih šetnji kojima je on početak, a kojima je kraj vrh pridružen funkciji F_i, za $i \in \{1, \ldots, m\}$. Ta mreža ima manje od $n+1$ vrhova pa možemo primjeniti pretpostavku indukcije. Dovedemo li kao ulaz spomenutom vrhu, za korak proslijeđivanja unazad, konstantu 1, na ulazu mreže dobijamo $F_i'(x)$. Prema tome, dovedemo li kao ulaz vrhu $\phi'(s)w_i$, na ulazu dobijamo $w_i F_i'(x)\phi'(s)$. Sada, dovedimo konstantu 1 na izlazni vrh mreže i provedimo korak proslijeđivanja unazad. Prvo se konstanta 1 u izlaznom vrhu množi sa $\phi'(s)$, a zatim se proslijeđuje vrhovima pridruženim funkcijama $F_1, \ldots, F_m$ i pritom množi sa težinama $w_1, \ldots, w_n$. Kako je spomenuto, prema pretpostavci indukcije, svaki od tih vrhova daje na ulaznom vrhu mreže $w_1 F_1'(x)\phi'(s), \ldots, w_m F_m'(x)\phi'(s)$, a oni se onda zbrajuju po definiciji B-dijagrama. Dakle, na ulazu dobijamo

$$w_1 F_1'(x)\phi'(s) + \ldots + w_m F_m'(x)\phi'(s) = F'(x)$$

čime je dokazana tvrdnja. ■

Navedeni algoritam može se poopćiti za slučaj kada ulazni sloj mreže ima više vrhova, a izlazni sloj i dalje jedan vrh, što je slučaj kod mreže koja realizira funkciju E. Neka ulazni sloj ima n vrhova. Tada za ulaz $(x_1, \ldots, x_n)$ možemo računati proizvoljnu parcijalnu derivaciju funkcije mreže $F(x_1, \ldots, x_n)$, to jest $\frac{\partial}{\partial x_i} F(x_1, \ldots, x_n)$, za proizvoljni $i \in \{1, \ldots, n\}$. Korak proslijeđivanja unaprijed ostaje potpuno isti. Korak proslijeđivanja unazad ponavlja se n puta — kako bismo dobili $\frac{\partial}{\partial x_i} F(x_1, \ldots, x_n)$, za proizvoljni $i \in \{1, \ldots, n\}$, provodimo standardni korak proslijeđivanja unazad, ali za podmrežu dane mreže

koju čine sve šetnje koje povezuju onaj vrh ulaznog sloja koji ima vrijednost x_i sa izlaznim vrhom. Na spomenutom vrhu ulaza dobijamo $\frac{\partial}{\partial x_i}F(x_1, \ldots, x_n)$.

Vratimo se sada na problem učenja. Rekli smo da želimo minimizirati funkciju greške E koja je funkcija svih težina mreže. Za potrebe sljedećeg razmatranja označimo sa w_{ij} težinu brida koji spaja neki vrh i proizvoljnog sloja sa vrhom j sljedećeg sloja. U svrhu minimizacije od E moramo izračunati $\frac{\partial E}{\partial w_{ij}}$. Prvo izvedemo algoritam s povratnim prosljeđivanjem pogreške; oba njegova koraka, a potom se fokusiramo na vrh j i promatramo podmrežu dane mreže koju čine sve šetnje s početkom u vrhu j kojima je kraj vrh izlaznog sloja. Ulaz takve podmreže je upravo $o_i w_{ij}$, gdje je o_i vrijednost pohranjenja u desnom dijelu vrha i. Korak proslijeđivanja unazad zaustavljen na vrhu j daje upravo $\frac{\partial E}{\partial o_i w_{ij}}$ pa dobijamo

$$\frac{\partial E}{\partial w_{ij}} = o_i \frac{\partial E}{\partial o_i w_{ij}}$$

i uz oznaku

$$\delta_j = \frac{\partial E}{\partial o_i w_{ij}}$$

imamo željenu promjenu

$$\Delta w_{ij} = -\alpha o_i \delta_j.$$

δ_j ćemo zvati pogreškom povratnog proslijeđivanja u vrhu j.

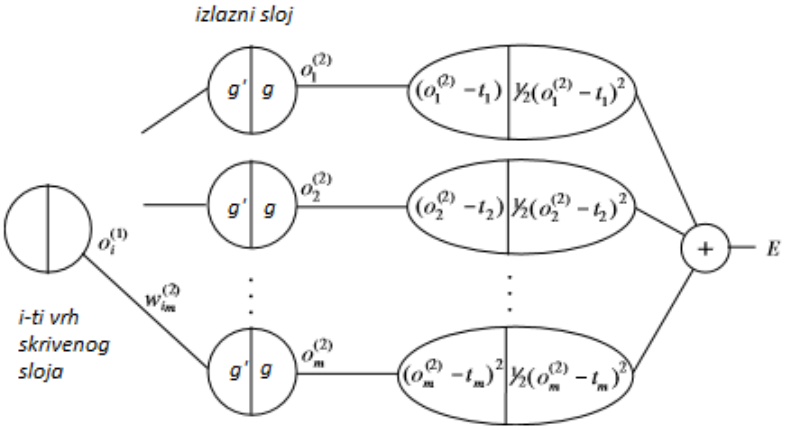
2.3.2 Primjena algoritma na jednoslojnu neuronsku mrežu

Neka je sada dana jednoslojna neuronska mreža koja ima n ulaznih vrhova, k vrhova u svom jedinom skrivenom sloju i m izlaznih vrhova sa svim oznakama navedenim u 1.4 i neka je dan skup za treniranje $T = \{(o_i, t_i) : i = 1, \ldots, p\}$. Pokazat ćemo, korak po korak, algoritam s povratnim proslijeđivanje greške primjenjen na navedenoj mreži.

Prvo, proširimo mrežu na već spomenuti način koristeći samo jedan element skupa za treniranje $(o, t) := (o_j, t_j)$, za proizvoljni $j \in \{1, \ldots, p\}$. Navedeno prikazuje Slika 2.12. Sada je $E = E_j$, a kasnije se stvar generalizira za svih p elemenata.

Za početak, odaberimo proizvoljne realne brojeve za sinaptičke težine bridova. Algoritam možemo podijeliti u četiri osnovna koraka:

1. Proslijeđivanje unaprijed
2. Proslijeđivanje unazad za izlazni sloj
3. Proslijeđivanje unazad za skriveni sloj
4. Promjena težina



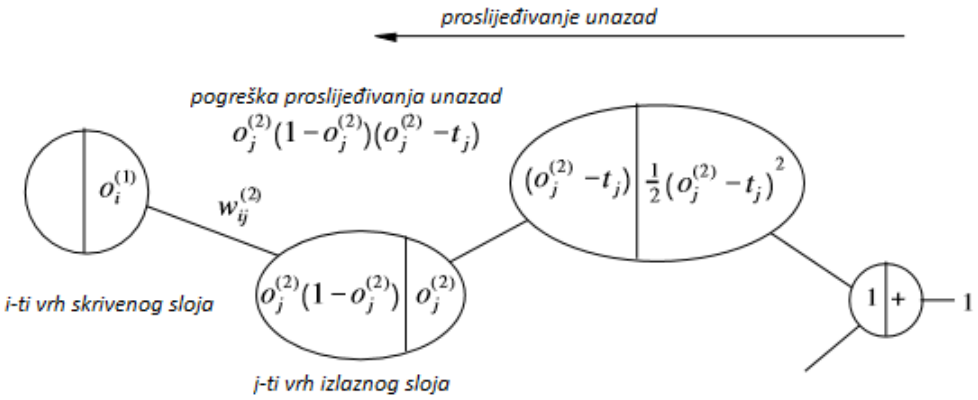
Slika 2.12: Proširena jednoslojna neuronska mreža [7]

Prvi korak: Proslijeđivanje unaprijed

Na ulaz mreže dovodimo $\hat{o} = (o_1, \ldots, o_n, 1)$. Mreža računa vektore $\hat{o}^{(1)} = (o_1^{(1)}, \ldots, o_k^{(1)}, 1)$ i $o^{(2)} = (o_1^{(2)}, \ldots, o_m^{(2)})$ i pohranjuje njihove komponente u desnim dijelovima vrhova skrivenog, odnosno izlaznog sloja jer su te komponente rezultat djelovanja aktivacijske funkcije g na ulaz u odgovarajući

vrh. Rezultat djelovanja derivacije aktivacijske funkcije g' , za koju vrijedi $g'(z) = g(z)(1 - g(z))$, na ulaz u odgovarajuće vrhove daje vektore $(o_1^{(1)}(1 - o_1^{(1)}), \ldots, o_k^{(1)}(1 - o_k^{(1)}), 0)$ i $(o_1^{(2)}(1 - o_1^{(2)}), \ldots, o_m^{(2)}(1 - o_m^{(2)}))$ koji se također pohranjuju u lijevim dijelovima vrhova skrivenog, odnosno izlaznog sloja.

Drugi korak: Proslijeđivanje unazad za izlazni sloj



Slika 2.13: Drugi korak: proslijeđivanje unazad za izlazni sloj [7]

U ovom koraku želimo izračunati $\frac{\partial E}{\partial w_{ij}^{(2)}}$, za $i \in \{1, \ldots, k+1\}$, $j \in \{1, \ldots, m\}$. Kako $w_{ij}^{(2)}$ označava težinu brida između i-toga vrha skrivenog sloja i j-tog vrha izlaznog sloja, promatramo podmrežu koja uključuje j-ti vrh izlaznog sloja i šetnju kojoj je on početak, a kraj izlazni vrh proširene mreže i na njoj provodimo korak proslijeđivanja unazad. Dakle, dovodimo konstantu 1 na izlazni vrh proširene mreže i na j-tom vrhu dobijamo

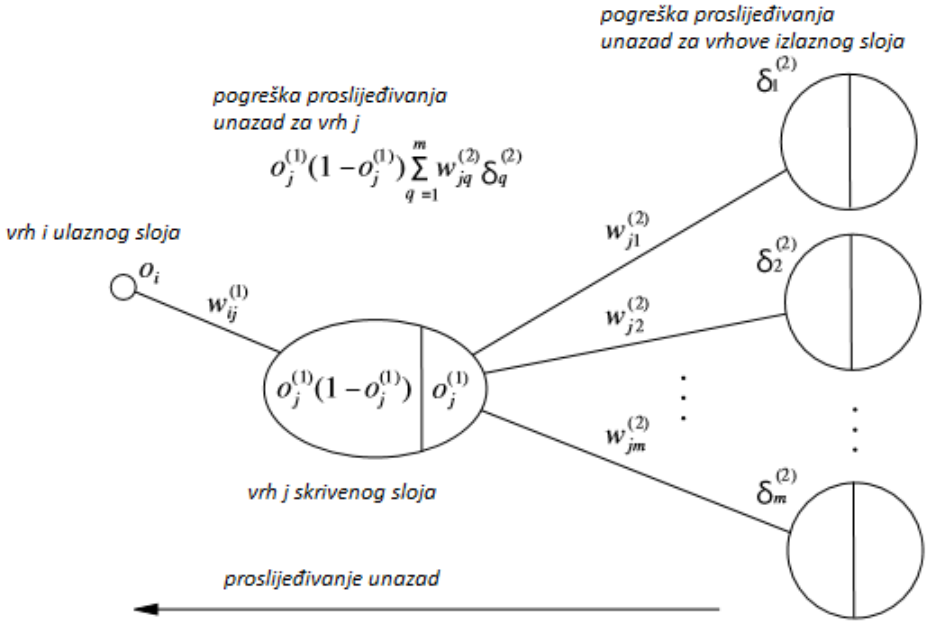
$$\delta_j^{(2)} := o_j^{(2)}(1 - o_j^{(2)})(o_j^{(2)} - t_j),$$

što zovemo greškom povratnog proslijeđivanja u vrhu j, a onda i traženo

$$\frac{\partial E}{\partial w_{ij}^{(2)}} = \delta_j^{(2)} o_i^{(1)}.$$

Navedeni postupak ilustriran je na Slici 2.13 i ponavljamo ga za sve sinaptičke težine bridova između skrivenog i izlaznog sloja.

Treći korak: Proslijeđivanje unazad za skriveni sloj



Slika 2.14: Treći korak: proslijeđivanje unazad za skriveni sloj [7]

U ovom koraku želimo izračunati $\frac{\partial E}{\partial w_{ij}^{(1)}}$, za $i \in \{q, \ldots, n{+}1\}$, $j \in \{1, \ldots, k\}$. Kako $w_{ij}^{(1)}$ označava težinu brida između i-toga vrha ulaznog sloja i j-tog vrha skrivenog sloja, promatramo podmrežu koja uključuje j-ti vrh skrivenog sloja i sve šetnje kojima je on početak, a kraj izlazni vrh proširene mreže. Vrh j povezan je bridom težine $w_{jg}^{(2)}$ sa vrhom q izlaznog sloja, za svaki $q \in \{1, \ldots, m\}$. Prema tome, pogreška proslijeđivanja unazad za vrh j je

$$\delta_j^{(1)} = o_j^{(1)}(1 - o_j^{(1)}) \sum_{q=1}^{m} w_{jq}^{(2)} \delta_q^{(2)},$$

gdje su $\delta_q^{(2)}$ pogreške proslijeđivanja unazad vrhova izlaznog sloja. Tražena

parcijalna derivacija je, prema tome,

$$\frac{\partial E}{\partial w_{ij}^{(1)}} = \delta_j^{(1)} o_i.$$

Pogreška proslijeđivanja unazad može se na isti način izračunati za proizvoljni broj skrivenih čvorova.

Četvrti korak: Promjena težina

Nakon izračunavanja svih parcijalnih derivacija, mijenjamo težine mreže

$$\Delta w_{ij}^{(2)} = -\alpha o_i^{(1)} \delta_j^{(2)}, \text{ za } i = 1, \ldots, k+1,\ j = 1, \ldots, m$$

i

$$\Delta w_{ij}^{(1)} = -\alpha o_i \delta_j^{(1)}, \text{ za } i = 1, \ldots, n+1,\ j = 1, \ldots, k,$$

uz oznake

$$o_{n+1} = o_{k+1}^{(1)} = 1.$$

■

Vratimo se sada skupu za treniranje

$$T = \{(o_i, t_i) : 1 \leq i \leq p\},$$

gdje je $o_i = (o_{i1}, \ldots, o_{in})$, $t_i = (t_{i1}, \ldots, t_{im})$. Dosad smo promatrali samo jedan element tog skupa i izračunali potrebne promjene sinaptičkih težina. S obzirom da je ipak dano p primjera za učenje mreže, ponovimo postupak za svaki od njih, to jest za svaki $i \in \{1, \ldots, n+1\}$ i $j \in \{1, \ldots, k\}$ provedemo navedeni algoritam p puta i dobijamo

$$\Delta_1 w_{ij}^{(1)}, \ldots, \Delta_p w_{ij}^{(1)},$$

gdje je $\Delta_l w_{ij}^{(1)}$, za $l \in \{1 \ldots, p\}$, promjena izračunata navedenim algoritmom koristeći primjer (o_l, t_l) , pa onda, jer je $E = \sum_{l=1}^{p} E_l$, vrijedi

$$\Delta w_{ij}^{(1)} = \Delta_1 w_{ij}^{(1)} + \ldots + \Delta_p w_{ij}^{(1)}.$$

Također, za svaki $i \in \{1, \ldots, k+1\}$ i $j \in \{1, \ldots, m\}$ dobijamo

$$\Delta_1 w_{ij}^{(2)}, \ldots, \Delta_p w_{ij}^{(2)}$$

pa je onda

$$\Delta w_{ij}^{(2)} = \Delta_1 w_{ij}^{(2)} + \ldots + \Delta_p w_{ij}^{(2)},$$

uz sve već dogovorene oznake.

2.3.3 Matrična forma algoritma

Dosad smo se bavili grafičkim pristupom algoritmu s povratnim proslijeđivanjem pogreške, a sad ćemo pristupiti istom problemu matrično. U tu svrhu, koristit ćemo sve oznake uvedene u Poglavlju 1.4 za jednoslojnu neuronsku mrežu.

Neka je D_2 dijagonalna matrica reda m čiji su elementi na dijagonali derivacije pohranjene u lijevim dijelovima vrhova izlaznog sloja, to jest

$$D_2 = \begin{bmatrix} o_1^{(2)}(1-o_1^{(2)}) & 0 & \ldots & 0 \\ 0 & o_2^{(2)}(1-o_2^{(2)}) & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & o_m^{(2)}(1-o_m^{(2)}) \end{bmatrix}$$

i neka je D_1 dijagonalna matrica reda k čiji su elementi na dijagonali derivacije pohranjene u lijevim dijelovima vrhova skrivenog sloja, to jest

$$D_1 = \begin{bmatrix} o_1^{(1)}(1-o_1^{(1)}) & 0 & \ldots & 0 \\ 0 & o_2^{(1)}(1-o_2^{(1)}) & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & o_k^{(1)}(1-o_k^{(1)}) \end{bmatrix}.$$

Dalje, neka je sa e označen m-dimenzionalni vektor čiji su elementi derivacije pohranjene u lijevim dijelovima vrhova proširenog sloja, to jest

$$e = \begin{bmatrix} o_1^{(2)} - t_1 \\ o_2^{(2)} - t_2 \\ \vdots \\ o_m^{(2)} - t_m \end{bmatrix}.$$

Neka je sa $\delta^{(2)}$ označen m-dimenzionalni vektor pogrešaka proslijeđivanjem unazad za izlazni sloj, to jest

$$\delta^{(2)} = \begin{bmatrix} \delta_1^{(2)} \\ \delta_2^{(2)} \\ \vdots \\ \delta_m^{(2)} \end{bmatrix}.$$

Tada je

$$\delta^{(2)} = D_2 e.$$

Slično, neka je sa $\delta^{(1)}$ označen k-dimenzionalni vektor pogrešaka proslijeđivanjem unazad za skriveni sloj, to jest

$$\delta^{(1)} = \begin{bmatrix} \delta_1^{(1)} \\ \delta_2^{(1)} \\ \vdots \\ \delta_k^{(1)} \end{bmatrix}.$$

Tada je

$$\delta^{(1)} = D_1 W_2 \delta^{(2)}.$$

Uz to, za matrice $\Delta\overline{W}_1$ i $\Delta\overline{W}_2$ vrijedi

$$\Delta\overline{W}_2^T = -\alpha\delta^{(2)}\hat{o}^{(1)} \quad \text{i} \quad \Delta\overline{W}_1^T = -\alpha\delta^{(1)}\hat{o}.$$

Poglavlje 3

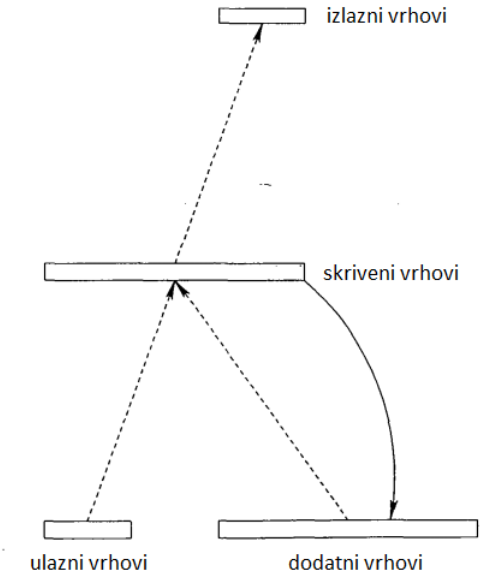
Naprednije arhitekture

U prethodnim poglavljima promatrali smo mreže kod kojih se ulazni podaci prenose od ulaznog sloja preko skrivenih slojeva sve do izlaznog sloja tako da se svaki vrh aktivira samo jednom. Takve mreže zovu se engl. ***feed forward*** mreže. Ako bismo htjeli istrenirati spomenutu mrežu tako da može prepoznati rukom pisane brojeve, nakon učenja na danom skupu za treniranje, pokazali bismo joj dotad neviđene slike i očekivali točan odgovor. Pritom, slijed slika koje mreža prima ne bi bio važan, to jest ako je mreži kao ulaz prvo dana slika koja prikazuje broj 4, a nakon toga slika koja prikazuje broj 5, ponašanje mreže prilikom obrade prve slike ne bi utjecalo na ponašanje mreže prilikom obrade druge slike. Dakle, engl. *feed forward* mreža nema nikakvih saznanja o redoslijedu kroz vrijeme i njen izlaz ovisi samo o trenutnom ulazu (i, naravno, težinama mreže).

Nešto drugačije mreže su **povratne neuronske mreže** (engl. *reccurent neural networks*). To su mreže kod kojih postoji barem jedan povratni brid između neka dva vrha, to jest za neka dva vrha mreže u i v postoje bridovi (u, v) i (v, u). Za njihov razvoj zaslužna je prirodna potreba implementiranja vremena, koje je važan čimbenik u procesu čovjekovog učenja i usko vezano s

čovjekovom inteligencijom, u arhitekturu neuronskih mreža. U povratnim neuronskim mrežama vrijeme je implicitno implementirano, to jest prikazano je preko utjecaja koji ima na procesuiranje i to koristeći povratne bridove koji omogućuju protok informacija u oba smjera i daju mreži memoriju. Za razliku od engl. *feed forward* mreža, izlaz povratnih mreža ovisi ne samo o trenutnom ulazu nego i o mrežinim prethodnim rezultatima, što joj omogućuje spomenuta memorija. Postoji velik broj podvrsta povratne mreže, ali većina je poseban slučaj potpune povratne mreže kod koje između svaka dva vrha postoji povratni brid. Ovdje ćemo pobliže proučiti Elmanovu mrežu i povratni višeslojni perceptron.

3.1 Elmanova mreža



Slika 3.1: Elmanova mreža [1]

Što se arhitekture tiče, Elmanova mreža, prikazana na Slici 3.1, osim ulaz-

nog, izlaznog i jednog skrivenog sloja, ima i **dodatne vrhove** (engl. *context units*) koji proširuju ulazni sloj i čija je početna vrijednost $\frac{1}{2}$ te čiji je broj jednak broju skrivenih vrhova s kojima su vezani korespodentnim povratnim bridovima fiksnih težina postavljenih na 1. Ovi vrhovi su također skriveni vrhovi u smislu da ne komuniciraju izravno s okolinom, nego samo s ostalim vrhovima mreže. Neka je sada dan *niz* ulaznih podataka. U vremenskom trenutku t, mreža prima prvi ulazni podatak, to jest prvi element danog niza podataka. Dodatni vrhovi u tom trenutku postavljeni su na $\frac{1}{2}$. Ulazni i dodatni vrhovi aktiviraju skrivene vrhove čiji se rezultat proslijeđuje izlaznim vrhovima i kopira povratnim vezama u dodatne vrhove. U vremenskom trenutku $t+1$ na ulazne vrhove dovodi se sljedeći element niza i zajedno sa dodatnim vrhovima, to jest rezultatima skrivenih vrhova iz trenutka t, aktivira skriveni sloj. Kod procesa učenja, nakon svakog vremenskog trenutka provodi se algoritam s povratnim proslijeđivanjem pogreške i to u istom obliku u kojem je već naveden zato što se težine povratnih veza postavljaju na 1 i ne mijenjaju kroz proces učenja, prema definiciji Elmanove mreže.

Dakle, povratne veze između dodatnih i skrivenih vrhova omogućuju skrivenom sloju pristup prethodnom rezultatu koji oblikuje trenutni, to jest daju mreži memoriju. Dakle, neka je x_t ulazni podatak u trenutku t, W matrica težina između ulaznog i skrivenog sloja i h_t rezultat skrivenog sloja u trenutku t. Danas postoje malo promijenjene Elmanove mreže kod kojih težine bridova između dodatnih i skrivenih vrhova nisu uvijek 1, nego su promjenjive. U tom je slučaju U matrica spomenutih težina pa je

$$h_t = g(Wx_t + Uh_{t-1}),$$

gdje je g aktivacijska funkcija, a prag svakog vrha je uključen u ulazne vrhove, kako je objašnjeno u prethodnim poglavljima. Formule za izlazne vrhove su iste kao prethodno. U slučaju dodavanja matrice težina U, jasno je da za

proces učenja ne moze biti korišten standardni algoritam povratnog prosljeđivanja. O drugom načinu učenja više će riječi biti kasnije.

Pokažimo sada kako je Elman iskoristio definiranu mrežu za realizaciju logičke funkcije XOR dane sa:

x_1	x_2	$x_1 \overline{\vee} x_2$
0	0	0
0	1	1
1	0	1
1	1	0

Dosad je XOR funkcija bila realizirana mrežom kod koje su ulazi bili dvodimenzinalni vektori $(0,0)$, $(1,1)$, $(0,1)$ i $(1,0)$, a izlazi 0, 0, 1 i 1 redom. Kako želimo realizirati XOR Elmanovom mrežom, bavit ćemo se nizovima konstruiranim tako da je dvodimenzionalni ulaz za funkciju XOR predstavljen kao dva uzastopna elementa niza, a izlaz funkcije kao treći element niza. Nakon toga, istim obrascem se nastavlja konstruirati niz, s tim da je idući dvodimezionalni vektor odabran slučajno. Na primjer:

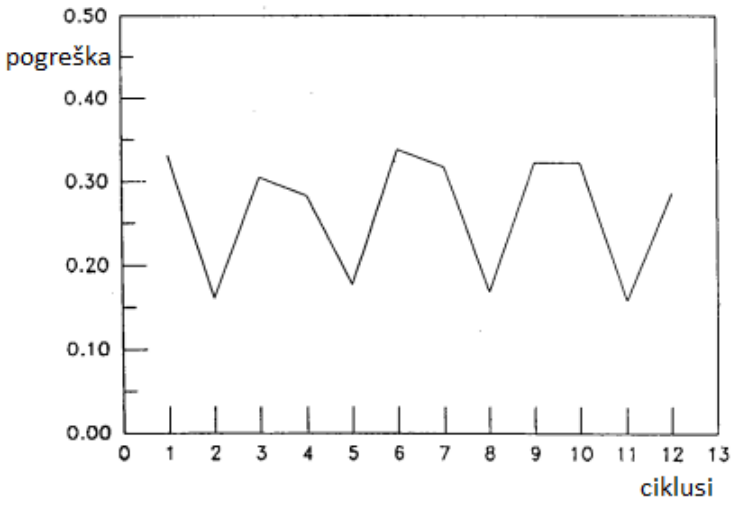
$$1\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 1\ \ldots$$

Dakle, primjenom XOR funkcije na prva dva elementa navedenog niza, 1 i 0, dobijamo treći element niza 1. Četvrti i peti elementi, to jest 0 i 0, ne ovise o prethodnim elementim niza niti međusobno, ali šesti element 0 niza ovisi o njima.

Elman je navedenim postupkom konstruirao niz koji se sastojao od 3000 elemenata. Pomoću njega je trenirao mrežu koja se sastojala od jednog ulaznog vrha, jednog izlaznog vrha, dva skrivena vrha i dva dodatna vrha s ciljem da ona nauči u svakom trenutku predvidjeti idući element niza.
Za naš primjer:

ulazi: 1 0 1 0 0 0 0 1 1 ...

izlazi:0 1 0 0 0 0 1 1 x ...



Slika 3.2: Greška Elmanove mreže za XOR [1]

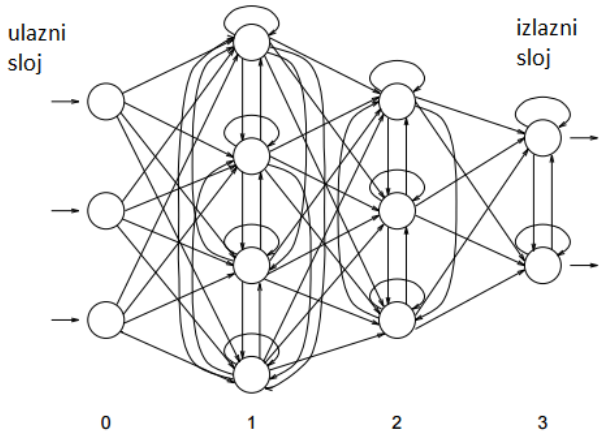
Zbog konstrukcije Elmanove mreže, točnije povratnih veza između dodatnih i skrivenih vrhova, predviđanje se ne temelji samo na trenutnom ulaznom podatku na ulaznom vrhu nego i na prethodnom stanju skrivenih vrhova mreže. Primjetimo da je u našem primjeru kao prvi ulaz u mrežu doveden broj 1 te da mreža ne može predvidjeti sljedeći element jer on nije ni na kakav način vezan s 1. Ne postoji pravilo koje veže ta dva elementa i za koje bismo htjeli da ga mreža nauči. Eventualno ispravno predviđanje mreže u tom trenutku je čista slučajnost (vjerojatnost da mreža ispravno predvidi je $\frac{1}{2}$). Tek nakon što dovedemo dva elementa niza na ulaz mreže, ima smisla zahtjevati od mreže točno predviđanje. Elman je trenirao mrežu na spomenutom nizu od 3000 elementa i to tako da je mreži pokazan cijeli niz 600 puta. Nakon

toga, mreža je uglavnom predviđala po opisanom obrascu. Testiranje mreže na neviđenim podacima dalo je grešku prikazno na Slici 3.2.

Greška je mala na onim mjestima gdje je moguće točno predviđanje, a velika gdje nije. To pokazuje da je mreža *naučila* ponešto o strukturi niza i da predviđa na temelju trenutnog ulaza i prethodnih vrijednosti skrivenih vrhova.

3.2 Povratni višeslojni perceptron

Povratni višeslojni perceptron (engl. *recurrent multilayer perceptron*) drugi je primjer povratne neuronske mreže i sastoji se od niza kaskadnih podmreža, gdje je svaka podmreža zasebni višeslojni perceptron i to takav da su povratni bridovi dozvoljeni samo među vrhovima posljednjeg sloja. Bridovi između podmreža također nisu povratni.



Slika 3.3: Povratni višeslojni perceptron sa dva skrivena sloja [8]

Iako je ovakva arhitektura mreže izrazito moćna, često se za uspješnu primjenu promatra s određenim ograničenjima. U ovom radu proučavat ćemo samo one povratne višeslojne perceptrone kod kojih se svaka podmreža sas-

toji samo od jednog sloja. Dakle, takav perceptron će i dalje imati tri vrste slojeva, ulazni, izlazni i skriveni, kao i kod engl. *feed forward* mreža, ali svi vrhovi svakog skrivenog sloja bit će povezani međusobno i sami sa sobom povratnim bridovima. Također, to će vrijediti i za vrhove izlaznog sloja, kako prikazuje Slika 3.3.

Kao i dosad, ulaz u svaki vrh j sloja i u trenutku t biti će težinska suma vrijednosti vrhova iz prethodnog sloja, to jest suma umnožaka vrijednosti vrhova iz prethodnog sloja i pripadnih sinaptičkih težina, no sada će se dodavati i suma umnožaka vrijednosti vrhova tog istog sloja i u trenutku $t-1$ i pripadnih sinaptičih težina. Dakle,

$$\sum_{k=1}^{N_{i-1}} w_{k,j}^{F,i} y_{i-1,k}(t) + \sum_{k=1}^{N_i} w_{k,j}^{R,i} y_{i,k}(t-1)$$

pa je

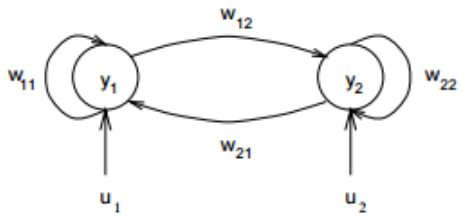
$$y_{i,j}(t) = g(\sum_{k=1}^{N_{i-1}} w_{k,j}^{F,i} y_{i-1,k}(t) + \sum_{k=1}^{N_i} w_{k,j}^{R,i} y_{i,k}(t-1))$$

gdje je $y_{i,j}(t)$ izlaz vrha j u sloju i u trenutku t, $w_{k,j}^{F,i}$ sinaptička težina brida između vrha k u sloju $i-1$ i vrha j u sloju i, $w_{k,j}^{R,i}$ sinaptička težina brida između vrha k i vrha j u sloju i te N_i broj vrhova u sloju i i g aktivacijska funkcija.

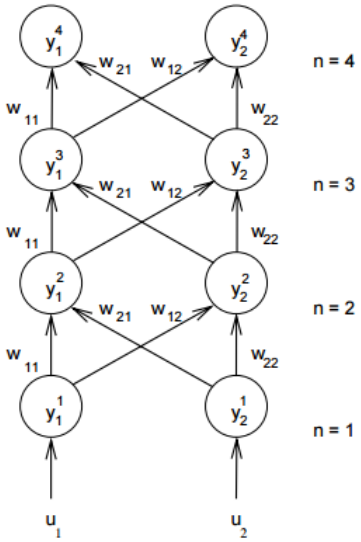
3.3 Algoritam povratnog proslijeđivanja pogreške kroz vrijeme

Proces učenja spomenutih povratnih neuronskih mreža temelji se na metodi gradijentnog spusta baš kao i proces učenja engl. *feed forward* mreža, no sami algoritam se ipak razlikuje zbog postojanja povratnih bridova. **Algoritam povratnog proslijeđivanja pogreške kroz vrijeme** je adaptacija algori-

tam povratnog proslijeđivanja pogreške koja podrazumijeva transformaciju povratne mreže u engl. *feed forward* mrežu. Provedbu ideje algoritma ilustrirat ćemo na jednostavnoj potpuno povezanoj povratnoj mreži s dva vrha, prikazanoj na Slici 3.4.



Slika 3.4: Jednostavna potpuno povezana povratna mreža [8]



Slika 3.5: 4 kopije jednostavne potpuno povezane povratne mreže [8]

Neka je dana niz duljine p koji predstavlja skup podataka za treniranje. Prvo, transformirajmo povratnu mrežu tako da je kopiramo p puta, kako je prikazano na Slici 3.5, uz oznake $y_i^n = y_i(n)$, gdje je $y_i(n)$ vrijednost vrha y_i

u n-tom trenutku. Sada, primjena algoritma s povratnim proslijeđivanjem pogreške daje promjene svake pojedine težine novonastale mreže. Međutim, bridovi koji su nastali kopijom jednog te istog brida povratne mreže moraju se promijeniti za isti iznos pa se zato mijenjaju za zbroj promjena težina dobivenih algoritmom.

Poglavlje 4

Zaključak

Neuronske mreže jedan su od najjačih alata umjetne inteligencije — u većini problema su efikasnije i primjenjivije od dosadašnjih tradicionalnih rješenja koje je nudila računarska znanost. U ovome radu obrađen je samo uvod u građu i funkciju neuronskih mreža te nekoliko konkretnih primjera. Broj različitih arhitektura je velik i primjena istih je široka — neuronske mreže koriste se u prepoznavanju uzoraka, pretvorbi teksta u govor, u ekonomiji...

Već ovakav kratak uvod u iznimno široko područje umjetne inteligencije nametnuo je neka pitanja. S obzirom da postoje razne vrste neuronskih mreža, razni algoritmi učenja mreže, primjeri aktivacijskih funkcija i funkcija greške, kako odabrati one za koje će mreža raditi optimalno s obzirom na zadani problem? Također, postavlja se pitanje koliko puta mreža treba vidjeti podatke za treniranje kako bi naučila esencijalna svojstva tih podataka, a ne „napamet" zapamtila dano? Na većinu ovih pitanja nema jednoznačnog odgovora. Računarska znanost kao rješenje uvijek sugerira ono za koje se eksperimetalnim putem pokazalo da je dovoljno dobro. No već je i to omogućilo široku primjenu neuronskih mreža u svakodnevnom životu, koja će, kako se čini, u budućnosti još više rasti.

Literatura

[1] Elman, J.L. (1990.) *Finding Structure in Time*

[2] Gerstner, W. *Supervised Learning for Neural Networsk: A Tutorial with JAVA exercises*

[3] Goodfellow, I., Bengio,J. i Courville, A. (2016.) *Deep Learning*

[4] Golemac, A. (2014.)*Osnove teorije grafova*

[5] Makin, J.G. (2006.) *Backpropagation*

[6] Rashid, T. (2016.) *Make Your Own Neural Network*

[7] Rojas, R.(1996.) *Neural Networks*

[8] Tutschku, K. (1995.) *Reccurent Multilayer Perceptrons for Identification and Control: The Road to Applications*

[9] Springer, O. (1995.) *Čovjek i zdravlje*

[10] Źak, S.H. i Chong, E.K.P.(2001.) *An Introduction to Optimization*

[11] Zern,E., Helmke,U. i Prätzel-Wolters,D. (2001.) *Mathematical Theory of Neural Networks*

[12] Matlab Geeks - Matlab blog, tutorials, and expertise
http://matlabgeeks.com/tips-tutorials/
neural-networks-a-perceptron-in-matlab/

[13] Neuron - Wikipedija
https://hr.wikipedia.org/wiki/Neuron

[14] Hrvatska Mensa
http://www.mensa.hr/glavna/cesto-postavljana-pitanja/
inteligencija

[15] Yaldex
http://www.yaldex.com/game-development/1592730043_
ch18lev1sec4.html

TEMELJNA DOKUMENTACIJSKA KARTICA

PRIRODOSLOVNO–MATEMATIČKI FAKULTET
SVEUČILIŠTA U SPLITU
ODJEL ZA MATEMATIKU

DIPLOMSKI RAD

MATEMATIČKE OSNOVE NEURONSKIH MREŽA

Matea Kalinić

Sažetak:

Glavni cilj ovog rada objasniti je matematički model po kojem računalo oponaša proces čovjekovog učenja. U tu svrhu definirane su jednostavne i višeslojne neuronske mreže kao objekt podložan učenju te je oprimjerena njihova upotreba na logičkim funkcijama. Također, objašnjen je algoritam s povratnim proslijeđivanjem pogreške, čija je glavna ideja utemeljena na metodi gradijentnog spusta, koji mijenjanjem parametara neuronskih mreža omogućava učenje istih. Algoritam je proveden na jednoslojnoj neuronskoj mreži. Naposljetku, uvedena je Elmanova mreža kao najjednostavniji primjer povratnih neuronskih mreža i primjerom pokazano učenje te mreže.

Ključne riječi:

jednostavni g-perceptron, jednostavna neuronska mreža, k-slojna neuronska mreža, algoritam s povratnim proslijeđivanjem pogreške, metoda gradijentnog spusta, Elmanova mreža

Podatci o radu:

47 stranica, 25 slika, 15 literaturnih navoda, pisano na hrvatskom jeziku

Mentor: *doc.dr.sc. Ivo Ugrina*

Članovi povjerenstva:

TEMELJNA DOKUMENTACIJSKA KARTICA

doc.dr.sc. Ivo Ugrina

doc.dr.sc. Snježana Braić

dr.sc. Goran Erceg

Povjerenstvo za diplomski rad je prihvatilo ovaj rad 6. lipnja 2017.

TEMELJNA DOKUMENTACIJSKA KARTICA

FACULTY OF SCIENCE, UNIVERSITY OF SPLIT
DEPARTMENT OF MATHEMATICS

MASTER'S THESIS

MATHEMATICAL FOUNDATIONS OF NEURAL NETWORKS

Matea Kalinić

Abstract:

The main objective of this master's thesis is to introduce a mathematical model which enables computers to imitate human's learning process. For this purpose, simple and multi-layer neural networks are defined as objects that can be trained and a few examples for their usage are provided. Also, Backpropagation algoritam, whose main idea is based on gradient descent method and which enables neural network to learn by changing their parametar, is explained. The algorithm was implemented on a single-layer neural network. Lastly, Elman's network was introduced as the simplest example of reccurent neural networks and its learning is demonstrated.

Key words:

simple g-perceptron, simple neural network, multi-layer neural network, Backpropagation algorithm, gradient descent method, Elman's network

Specifications:

47 pages, 25 figures, 15 references, written in Croatian

Mentor: *Ivo Ugrina, PhD, Associate Professor*

Committee:

Ivo Ugrina, PhD, Associate Professor

TEMELJNA DOKUMENTACIJSKA KARTICA

Snježana Braić, PhD, Associate Professor

Goran Erceg, PhD

This thesis was approved by the Thesis commettee on June 6, 2017.