SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 4699

Neuronske mreže izgrađene maxout neuronima

Luka Žmegač

Zagreb, lipanj 2016.

Velika hvala mentoru doc. dr. sc. Marku Čupiću na strpljenju, pomoći i savjetima kojima je omogućio izradu ovog rada.
Zahvaljujem se prijateljima na pomoći u izradi rada te na podršci koju su mi pružali.
Najveća hvala mojoj obitelji koja mi je pružala potporu svih ovih godina u mojim nastojanjima, porazima i najviše od svega, uspjesima.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA ZAVRŠNI RAD MODULA

Zagreb, 17. ožujka 2016.

ZAVRŠNI ZADATAK br. 4699

Pristupnik: **Luka Žmegač (0036478761)**
Studij: Računarstvo
Modul: Računarska znanost

Zadatak: **Neuronske mreže izgrađene maxout-neuronima**

Opis zadatka:

Maxout-neuronske mreže su vrsta neuronskih mreža koje sadrže novu vrstu neurona: takozvane maxout-neurone. Radi se o jednoj od vrsta neurona kod kojih prijenosna (aktivacijska) funkcija nije fiksna već se može učiti iz podataka. Učenje je moguće klasičnim gradijentnim spustom a tipično se koristi uz uporabu tehnike dropout.

U okviru ovog zadatka potrebno je proučiti ovu vrstu neuronskih mreža, napisati programsku implementaciju mreže i algoritma treniranja uz dropout. Mrežu je potrebno primijeniti na odabrani problem klasifikacije. Za implementirani postupak potrebno je provesti vrednovanje. Radu je potrebno priložiti algoritme, izvorne kodove i rezultate uz potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Zadatak uručen pristupniku: 18. ožujka 2016.
Rok za predaju rada: 17. lipnja 2016.

Mentor:
Doc. dr. sc. Marko Čupić

Djelovođa:
Doc. dr. sc. Tomislav Hrkać

Predsjednik odbora za završni rad modula:
Prof. dr. sc. Siniša Srbljić

SADRŽAJ

1. Uvod

Mnogi problemi računarstva riješeni su optimizacijom algoritama i napretkom računala. Međutim, postoje problemi koji nisu rješivi u konačnom vremenu poput NP-teških[1] ili jednostavno nema algoritama koji bi ih riješili. Teži i kompleksniji problemi klasifikacije, prepoznavanja obrazaca i računalnog vida, spadaju u tu skupinu problema.

Sredinom 20. stoljeća u počecima računalstva pokazalo se kako za rješavanje problema nije dovoljna samo sirova snaga računala, nego i odgovarajuća strategija. Započet je razvoj raznih područja računarstva ali i neuroračunarstva potaknut novim spoznajama o ljudskom mozgu. Zamisao je bila da se oponaša ljudski mozak i njegov rad te da se takvim postupcima riješe neki teški problemi.

Neuronske mreže nude drugačiji način analize podataka i prepoznavanja obrazaca u njima od tradicionalnih računalnih metoda, ali nisu rješenje za sve probleme. Ako se problem može dobro okarakterizirati i ako su podaci iz kojih se treba nešto zaključiti vrlo jasni, tradicionalne metode dovoljno su dobre za njihovo rješavanje. Problemi koji nisu strukturirani i podaci u kojima nisu istaknute neke karakteristike preteški su za klasične metode u smislu da nisu rješivi u konačnom vremenu ili uopće nisu rješivi. Za mnoge probleme nije jasan način algoritamskog rješavanja. Pomoću neuronskih mreža u mogućnosti smo tražiti uzorke za koje možda nismo ni svjesni da postoje.

Količine generiranih podataka svake se godine udvostručuju. Većina tih podataka je nestrukturirana i nemoguće bi bilo iskoristiti ih u korisne svrhe običnim algoritmima. Neuronske mreže su sposobne analizirati te podatke koji inače ne bi bili korisni i nalaziti uzorke u njima, donositi neke korisne zaključke i omogućiti iskorištavanje i tih nestrukturiranih podataka.

[1] NP-teški problemi - problemi za čije je rješavanje potrebno nedeterminističko polinomijalno vrijeme.

2. Umjetne neuronske mreže

Počeci neuronskih mreža u računarstvu počinju 40-ih godina prošlog stoljeća, točnije 1943. godine, pretpostavkama o radu umjetnog neurona (tzv. TLU-perceptrona) u radu *A Logical Calculus of Ideas Immanent in Nervous Activity Warrena* McCullocha i Waltera Pittsa u kojem su modelirali jednostavnu neuronsku mrežu pomoću strujnog kruga. Donald Hebb ojačao je ovaj koncept neurona u svojoj knjizi *The Organization of Behavior* u kojoj je iznio zapažanje da se neuronski putevi ojačavaju svakim prolaskom impulsa kroz njih.

Daljnjim razvojem računarstva u 50-im godinama prošlog stoljeća, postalo je moguće modelirati teorijske pretpostavke. Prve korake u implementiranju računalnih neuronskih mreža napravio je Nathanial Rochester iz IBM-ovih istraživačkih laboratorija.

U sljedećim godinama John von Neumann predlaže izgradnju jednostavnih funkcija neurona pomoću telegrafskih releja ili vakumskih cijevi. Frank Rosenblatt sa sveučilišta Cornell započinje rad na Perceptronima. Rosenblatt je bio fasciniran funkcioniranjem oka muhe koje procesira većinu informacija i koje muhi govori gdje letjeti. Model Perceptrona i algoritam učenja koji su nastali ovim istraživanjem, najstariji su tip neuronskih mreža koje se i danas koriste. Jednoslojni perceptron se pokazao korisnim u klasificiranju kontinuiranih varijabli u dvije klase.

Krajem 50-ih godina Bernard Widrow i Macrian Hoff razvili su ADALINE neurone i neuronsku mrežu MADALINE zbog upotrebe neurona koje su oni nazvali prilagodljivim linearnim elementima (engl. *ADAptive LINear Elements*). MADALINE je bila prva komercijalno primjenjena neuronska mreža za uklanjanje jeke u telefonskim linijama, a u upotrebi je i danas.

Prvi veći uspjesi neuronskih mreža naveli su ljude da misle da su neuronske mreže svemoguće u godinama koje su slijedile. Međutim, zbog ograničenja algoritama i performansi računala došlo je do velikog razočaranja u neuronske mreže te stagnacije istraživanja na području neuronskih mreža. Ponovni uzlet u istraživanju neuronskih mreža dogodio se početkom 80-ih godina utjecajem radova poput *Neural Networks and Physical Systems with Emergent Collective Computational Abilities* autora Johna

Hopfielda, koji je matematičkom analizom pokazao što su neuronske mreže u mogućnosti činiti te otkrića novog algoritma propagiranja pogreške unatrag autora Paula Werbosa.

Napredak u tehnologijama izrade računala i algoritama učenja neuronskih mreža u posljednjem desetljeću omogućio je značajna poboljšanja performansi neuronskih mreža, zbog kojeg su neuronske mreže postale veoma popularne. Zbog ograničenja računalne tehnike pokušavaju se razviti neuro-procesori koji bi značajno ubrzali izvođenje neuronskih mreža.

2.1. Motivacija

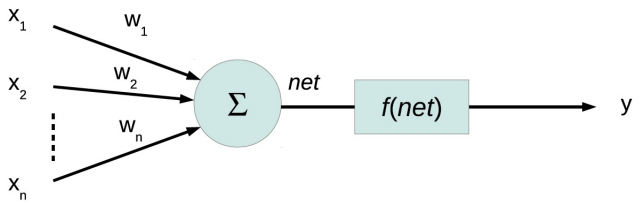
Priroda nam je i u ovom slučaju inspiracija, jer ako uzmemo neke probleme koji su nerješivi za računala, neka ih životinja s vrlo jednostavnim mozgom može vrlo jednostavno riješiti. Zašto je to moguće? Istraživači u biologiji su došli do zaključaka da mozak sprema informacije kao uzorke. Neki od tih uzoraka vrlo su kompleksni i omogućavaju nam na primjer prepoznavanje osoba.

Razvoj umjetnih neuronskih mreža započeo je pokušajem da se objasni i simulira inteligentno ponašanje životinja i ljudi. Rad mozga još je dobrim dijelom misterij ali su pojedini aspekti poznati. Poznato je da su osnovne stanice mozga jedine stanice u tijelu koje se ne obnavljaju i zamjenjuju, pretpostavlja se da nam one omogućuju pamćenje, razmišljanje i prepoznavanje situacije prema već viđenim situacijama. Snaga ljudskog mozga dolazi iz samog broja ovih osnovnih stanica i veza između njih. Podaci koji se šalju u mozak masovno se paralelno obrađuju i omogućuju prepoznavanje mjesta na koje smo došli ili osobe koju smo vidjeli. Prema istraživanjima, ljudski mozak sastoji se od 100 milijardi neurona pri čemu postoji oko 100 različitih vrsta. Svaki neuron povezan je u prosjeku s deset tisuća drugih neurona vezama koje se nazivaju sinapse. Dodatna činjenica koja podupire teoriju o masovnom paralelnom obrađivanju podataka je sporost rada neurona. Vrijeme paljenja neurona je otprilike 1ms, što bi značilo da se podaci moraju paralelno izvoditi, jer bi se inače u slijednoj obradi informacije mogao obraditi tek mali broj koraka u vremenu reakcije.

2.2. Model neurona

Biološki neuron osnovna je gradivna stanica mozga koja ima tijelo s jezgrom, akson i dendrite koji se nalaze na rubovima stanice: to su tanki krakovi koji se granaju u dendritska stabla kojima se neuron spaja s drugim neuronima i preko kojih dobiva električne impulse koji se akumuliraju u tijelu. Nakon što se akumulira količina naboja koja je veća od praga, neuron se aktivira i šalje akumulirani naboj u akson. Akson dodatno izmjenjuje signal i prenosi ga dalje. Moglo bi se reći da su dendriti ulazi za podatke, tijelo stanice dio za generiranje podataka te akson dio koji obrađuje podatke i prosljeđuje ih dalje.

Na temelju biološkog neurona možemo definirati umjetni neuron koji će isto imati tri dijela: ulaze koji predstavljaju dendrite, tijelo neurona koje generira podatke i šalje ih aksonu na daljnju obradu te izlaze koji predstavljaju sinapse. Ulazni podaci dovode se na ulaze x_1 do x_n Težine w_1 do w_n određuju utjecaj svakog od ulaza na neuron. Tijelo neurona računa ukupnu pobudu (označavat ćemo je s net), šalje je dalje aksonu to jest prijenosnoj funkciji $f(net)$ koja obrađuje ukupnu pobudu i šalje je na izlaze y_1 do y_n. Grafički prikaz modela neurona dan je na slici 2.1.



Slika 2.1: Prikaz strukture umjetnog neurona, preuzeto iz (Čupić et al., 2013)

Kako neuron ima prag paljenja (engl. *bias*) θ, moramo i njega uključiti u model. Prag ne trebamo tretirati kao posebnu komponentu već ga možemo ukomponirati u ukupnu pobudu tako da na "fiktivni" ulaz x_0 stavimo konstantnu pobudu 1, a težinu w_0 postavimo na vrijednost praga paljenja.

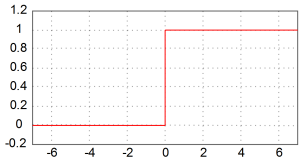
$$net = w_1 \cdot x_1 + w_2 \cdot x_2 + ... + w_n \cdot x_n - \theta \tag{2.1}$$

$$= w_0 \cdot x_0 + w_1 \cdot x_1 + ... + w_n \cdot x_n \tag{2.2}$$

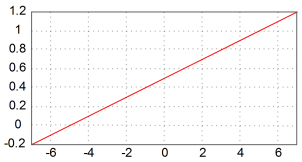
$$= \sum_{i=0}^{n} w_i \cdot x_i \tag{2.3}$$

Prijenosna funkcija modelira ponašanje aksona. Funkcije određuju ulogu neurona i u praksi se koristi nekoliko tipičnih prijenosnih funkcija: jedinična, funkcija skoka,

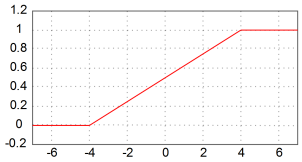
linearna, po dijelovima linearna, sigmoidalna, ReLU[1] i druge. Funkcije možemo vidjeti na slici 2.2. Odluka koja će se funkcija koristiti određuje problem koji se rješava neuronskom mrežom.



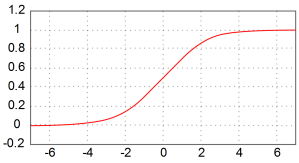
(a) Funkcija skoka



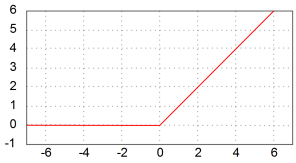
(b) Linearna funkcija



(c) Po dijelovima linearna funkcija



(d) Sigmoidalna funkcija



(e) ReLU funkcija

Slika 2.2: Prikaz različitih prijenosnih funkcija

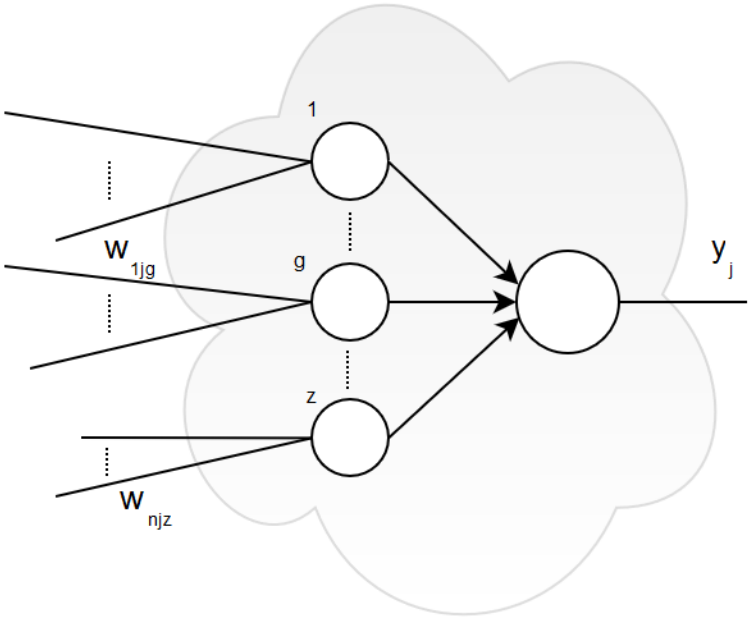
Povezivanjem više neurona, dobivamo umjetne neuronske mreže koje ćemo kasnije spominjati samo kao neuronske mreže ili mreže. Mreža se dijeli na slojeve neurona koji imaju različite funkcije. Ulazni sloj neurona ne obavlja nikakvu obradu podataka i samo prosljeđuje dobivene informacije dubljim slojevima mreže. Izlazni sloj neurona daje izlaz mreže i mora imati onoliko neurona koliko je i dimenzionalnost rješenja problema koji se rješava. Slojevi između ulaznog i vanjskog nazivaju se unutrašnji. Neuroni u slojevima i između slojeva mogu biti povezani vezama koje predstavljamo težinama. U ovom radu obrađivat će se najčešći tip mreža: potpuno povezane unaprijedne mreže u kojima neuroni u istim slojevima nisu međusobno povezani. Susjedni slojevi su potpuno povezani, a neuroni su povezani sa svim neuronima sljedećeg i prethodnog sloja.

2.3. Neuronske mreže tipa Maxout

Maxout-neuroni nastali su kao prirodni par tehnici Dropout koju ćemo kasnije obraditi. Maxout-neuroni pokazali su se vrlo uspješnima u raznim domenama i ne nužno uz korištenje tehnike Dropout. Neuroni ovog tipa razlikuju se od ostalih tipova ne-

[1]ReLU - (engl. *Rectified Linear Unit*) prijenosna funkcija $f(net) = max(0, net)$

urona u tome što imaju novu prijenosnu funkciju, maxout. Ona za aktivaciju odabire maksimalnu vrijednost unutar grupe linearnih dijelova. Ovakva nelinearnost je generalizacija ostalih funkcija s mogućnošću aproksimacije bilo koje aktivacijske funkcije. Aktivacijska funkcija maxout je slična ReLU s razlikom da maxout uspoređuje vrijednosti grupe kandidata linearnih dijelova dok ReLU uspoređuje vrijednost jednog dijela s 0 (Cai et al., 2013). Pokazano je da neuronske mreže građene Maxout-neuronima, mreže tipa maxout, postižu bolje rezultate od mreža s prijenosnim funkcijama tipa sigmoida ili ReLU (Goodfellow et al., 2013).



Slika 2.3: Prikaz strukture Maxout-neurona

Struktura Maxout-neurona vizualno je prikazana na slici 2.3 u obliku skupine više neurona zbog lakšeg razumijevanja i ekspresivnijeg prikaza. Skupovi težina su odvojeni i prikazani kao zasebni neuroni, koji se zatim spajaju u glavni neuron koji računa maksimalnu vrijednost.

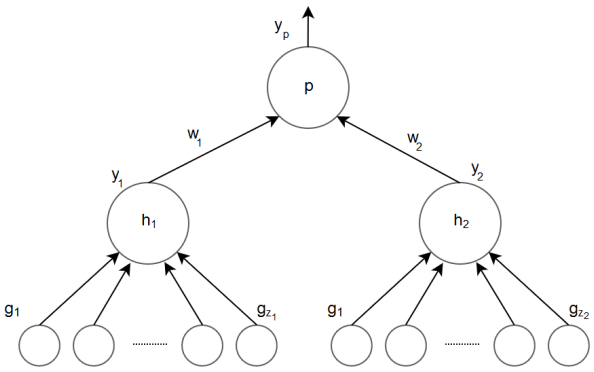
Svaki Maxout-neuron ima grupu z linearnih dijelova. Najveća vrijednost z dijelova odabire se kao aktivacija neurona. Ove grupe zapravo predstavljaju skupovi težina za razliku od običnih neurona koji imaju samo jedan skup težina. Prijenosnu funkciju neurona koji ima z skupova s n ulaza prikazujemo:

$$y = \max_{g \in 1 \ldots z} net_g \tag{2.4}$$

gdje je net_g definiran kao u 2.3 s pragom paljenja:

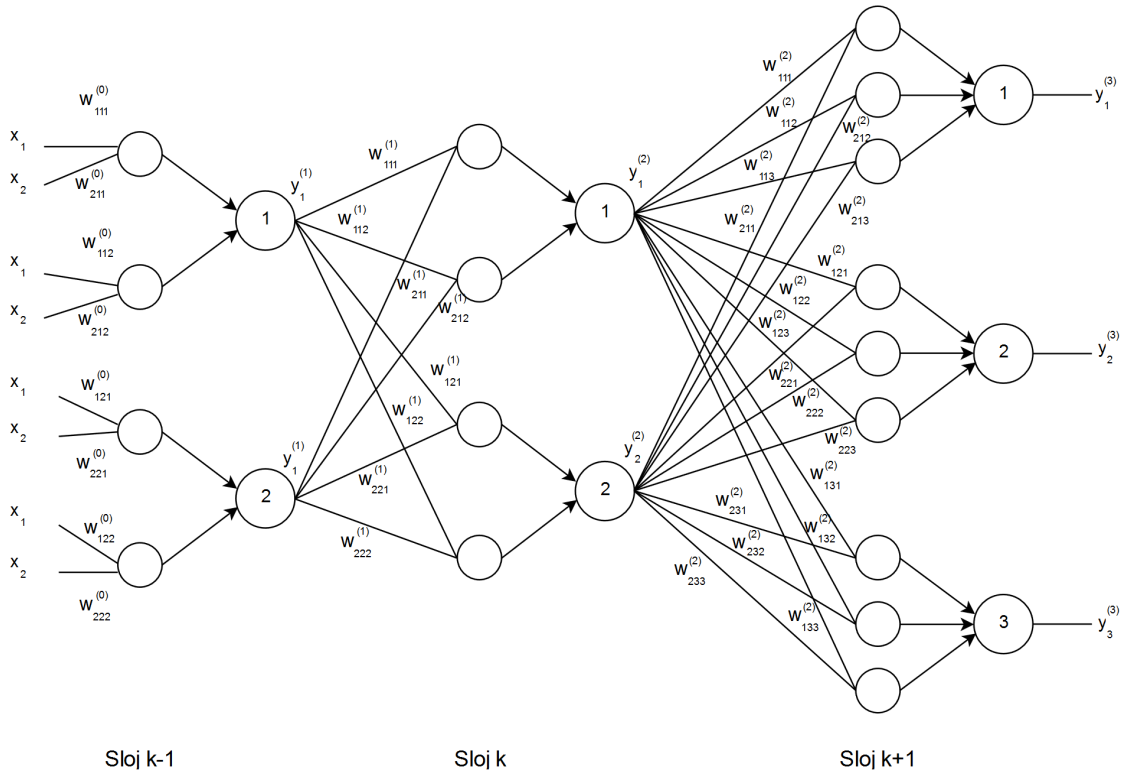
$$net_g = \sum_{i=0}^{n} w_{ig} \cdot x_i$$

Struktura Maxout-neurona omogućuje mu da aproksimira bilo koju kontinuiranu funkciju koja je konveksna. Autori u (Goodfellow et al., 2013) su dokazali da samo dva Maxout neurona mogu proizvoljno precizno oponašati bilo koju kontinuiranu funkciju, ako imaju dovoljno linearnih dijelova, odnosno skupova težina. Tijekom učenja Maxout-neuroni uče prijenosnu funkciju sami od sebe i u teoriji mogu oponašati bilo koju kontinuiranu funkciju što ih čini veoma moćnim pri rješavanju stvarnih problema. Osnovna ideja oponašanja bilo koje kontinuirane funkcije prikazana je na slici 2.4.



Slika 2.4: Neuronska mreža tipa Maxout koristeći samo dva Maxout-neurona može oponašati bilo koju kontinuiranu funkciju. Težine u izlaznom sloju podešene su tako da dobijemo razliku izlaza dva Maxout-neurona h_1 i h_2. Kako svaki od neurona h_1 i h_2 može oponašati bilo koju konveksnu kontinuiranu funkciju, izlaz mreže može stoga oponašati bilo koju kontinuiranu funkciju.

Spajanjem više Maxout-neurona dobivamo neuronsku mrežu tipa Maxout. Slijedi primjer jedne takve mreže s tri sloja u kojima prvi i drugi sloj imaju po dva skupa težina, a treći po tri. Mreža ima dva ulaza i tri izlaza. Struktura mreže prikazana je na slici 2.5.



Slika 2.5: Primjer potpuno povezane unaprijedne neuronske mreže tipa Maxout

3. Učenje neuronskih mreža

Neuronske mreže moraju se trenirati (učiti) da bi se postigle željene funkcionalnosti mreže. Od početaka razvoja neuronskih mreža razvile su se metode koje bismo mogli podijeliti u tri glavne skupine, kako je prikazano u nastavku.

- Nadzirano učenje
  U situacijama kada znamo ulaze i odgovarajuće izlaze neuronske mreže možemo izračunati pogrešku mreže na temelju dobivenih i traženih izlaza. Pogrešku možemo iskoristiti da bismo mijenjali mrežu podešavajući težine te poboljšali karakteristike mreže.
- Nenadzirano učenje
  Odgovarajuće izlaze nemamo uvijek te se u takvim situacijama koristi nenadzirano učenje u kojem se pomoću ulaza pokušavaju naći neki uzorci i svojstva u podacima. U ovom slučaju ne možemo utjecati na mrežu pomoću ciljnih vrijednosti.
- Potporno učenje
  U potpornom učenju isto kao u prethodnom nenadziranom učenju nemamo izlaze ali se može izračunati kvaliteta danog rješenja to jest funkcija dobrote. Cilj algoritama potpornog učenja je maksimizirati nagradu ili minimizirati kaznu ovisno o situaciji, modelom pokušaja i pogrešaka. Mnogi algoritmi iz ove porodice inspirirani su prirodom.

Razvoj neuronskih mreža i pripadnih algoritama u svojoj povijesti imali su uspona i padova. 70-ih godina prošlog stoljeća velik problem u složenijim neuronskim mrežama bio je određivanje odgovornosti za pogreške pojedinih neurona i korekcije takvog ponašanja. Problem je dobio naziv problem dodjele zasluga. U to vrijeme, problem nije uspješno riješen te se konektivistički pristup[1] rješavanju problema sve više napuštao. Značajni porast popularnosti područja neuronskih mreža ponovo se dogo-

[1] Konektivistički pristup - pristup umjetnoj inteligenciji koji se temelji na izgradnji sustava čija se arhitektura temelji na neuronskim mrežama.

dio u drugoj polovici 80-ih godina otkrićem algoritma propagacije pogreške unatrag (engl. *Backpropagation*) koji je riješio velik problem dodjele zasluga za određene vrste mreža. Algoritam propagacije pogreške unatrag ima svojih nedostataka i problema koje ne može riješiti. Stoga su, pogotovo u zadnjem desetljeću, nastale mnoge tehnike i algoritmi rješavanja uočenih problema. Jedna od novijih tehnika je i tehnika *Dropout* koji će omogućiti izradu dubljih mreža i povećati performanse algoritma propagacije pogreške unatrag. O tehnici Dropout više će biti rečeno u kasnijem dijelu rada.

3.1. Algoritam propagacije pogreške unatrag

Jednoslojne mreže ograničene su u mogućnosti učenja. Već je prije rečeno kako višeslojne mreže rješavaju neke od problema. Neuroni u višeslojnoj mreži su povezani u slojevima na način da neuroni u n-tom sloju svoju aktivacijsku vrijednost prenose samo neuronima u $n + 1$-om sloju. Signal svojom propagacijom može pogreške koje se nalaze duboko unutar mreže propagirati i razviti kompleksnim i neočekivanim načinima kroz slojeve te je zbog toga analiza pogreške višeslojne mreže na izlazu kompleksna.

3.1.1. Princip rada algoritma propagacije pogreške unatrag

Algoritam propagacije pogreške unatrag predstavlja jedno od najpopularnijih rješenja ovog problema, raspodjeljujući krivicu za pogreške i mijenjajući težine veza među neuronima. Kao što njegovo ime kaže, algoritam propagacije pogreške unatrag temelji se na računanju pogreške na izlazu te na njenom propagiranju unatrag po mreži uz pomoć tehnike gradijentnog spusta.

Algoritam se može podijeliti u nekoliko faza koje se ponavljaju sve dok svojstva neuronske mreže nisu zadovoljiva. Prva faza je propagiranje ulaznog signala kroz mrežu od početnog sloja do izlaznog i dobivanje izlaznog signala na izlazu. Sljedeća faza je računanje gradijenta funkcije pogreške pomoću propagiranja pogreške od izlaznog sloja pa sve do početnog. Pogreška se može računati na više načina, ali najčešći su razlika izlaza dobivenog na mreži i onog željenog te kvadrat srednje pogreške dobivenog i željenog izlaza. Nakon propagiranja unaprijed i unazad promjene težina unose se u neuronsku mrežu. Da bi se ublažio utjecaj gradijenta uvedena je stopa učenja. Postavljanje vrijednosti parametra stope učenja je proizvoljno i za svaki problem se razlikuje. Povećanjem stope učenja ubrzava se učenje, ali i povećava mogućnost divergiranja te-

žina i izlaza te propadanja mreže. Smanjivanjem stope učenja, učenje je točnije ali se povećava mogućnost konvergencije rezultata u nekom lokalnom minimumu.

Izvedbe algoritma dijelimo prema vremenu unosa promijena težina među neuronima u neuronsku mrežu.

Klasični algoritam propagacije pogreške unatrag - zovemo ga još grupni (engl. *Batch Backpropagation*). Algoritam računa gradijent parova podataka cijelog podatkovnog skupa na neuronskoj mreži nepromijenjenih težina. Nakon prolaska svih podatkovnih parova kroz algoritam tijekom kojeg se računao ukupni gradijent, gradijent promjene neuronske mreže dodaje se težinama neuronske mreže. Učenje ovakvim modelom brže daje rezultate ali postoji dobra vjerojatnost da će oni zapeti u nekom lokalnom minimumu.

Stohastički algoritam propagacije pogreške unatrag - gradijent promjene nakon svakog para podataka se izračunava i dodaje težinama neuronske mreže. Ovakav način učenja unosi smetnje u mrežu koristeći podatke samo jednog para podataka, što smanjuje vjerojatnost da mreža zapne u nekom lokalnom minimumu, ali sporije konvergira od klasičnog algoritma propagacije pogreške unatrag.

Algoritam propagacije pogreške unatrag s mini-grupama - kombinacija grupnog i stohastičkog algoritma propagacije pogreške unatrag u kojem se umjesto ili korištenja svih podataka podatkovnog skupa ili samo jednog podatkovnog para za računanje gradijenta, koristi proizvoljan broj podatkovnih parova za izračun gradijenta koji se koristi pri ispravljanju mreže.

Gradijent je ovisan o prijenosnoj funkciji i tipu neurona. Pomoću gradijenta ćemo doći do promjene težina neurona u svakom sloju za stohastički algoritam propagacije pogreške unatrag. Kriterijska funkcija koja mjeri kvalitetu neuronske mreže definirana je kao polovica srednjeg kvadratnog odstupanja svakog željenog izlaza mreže i stvarne vrijednosti izlaza za jedan uzorak. Funkciju možemo zapisati:

$$E = \frac{1}{2}\sum_{o=1}^{m}(t_o - y_o^{(k+1)})^2$$

gdje indeks o ide po svim izlaznim neuronima neuronske mreže. Oznaka $y_o^{(k+1)}$ predstavlja izlaz o-tog neurona izlaznog sloja a t_o traženi izlaz o-tog neurona za jedan uzorak podataka. Postupak učenja mreže treba promijenom težina u mreži ostvariti minimalnu funkciju E koja je funkcija skupa uzoraka te mreže čiji iznos ovisi o vrijednostima težinskih faktora. Kako mreža ima fiksnu arhitekturu a skup uzoraka se ne

mijenja, minimum funkcije E će se moći postići jedino mijenjanjem težinskih faktora. Po uzoru na gradijentni spust potrebno je izračunati gradijent od E:

$$\frac{\partial E}{\partial w_{ijg}^{(k)}}$$

Težinski faktori (težina) $w_{ijg}^{(k)}$ mijenjat će se u skladu s gradijentnim spustom:

$$w_{ijg}^{(k)} \leftarrow w_{ijg}^{(k)} - \psi \cdot \frac{\partial E}{\partial w_{ijg}^{(k)}} \tag{3.1}$$

gdje će ψ biti mala pozitivna konstanta koju nazivamo *stopa učenja*.

3.1.2. Algoritam propagacije pogreške unatrag primjenjen na neuronskim mrežama tipa Maxout

Algoritam za učenje napravit ćemo za neuronske mreže tipa Maxout. Izvod algoritma napravit ćemo pomoću slojevite neuronske mreže tipa Maxout s c ulaza, m izlaza i $k+1$ slojeva neurona. Podaci za učenje mreže su predstavljeni u obliku N parova $(\vec{x}_i, \vec{t}_i)$ gdje je $\vec{x}_i$ c-dimenzijski vektor ulaza a $\vec{t}_i$ m-dimenzijski vektor odgovarajućeg odziva mreže.

Izlazni sloj Izvod za izlazni sloj različit je od skrivenih slojeva i zbog toga ćemo prvo napraviti izvod za izlazni sloj.

Potrebno je izračunati $\frac{\partial E}{\partial w_{ijg}^{(k)}}$:

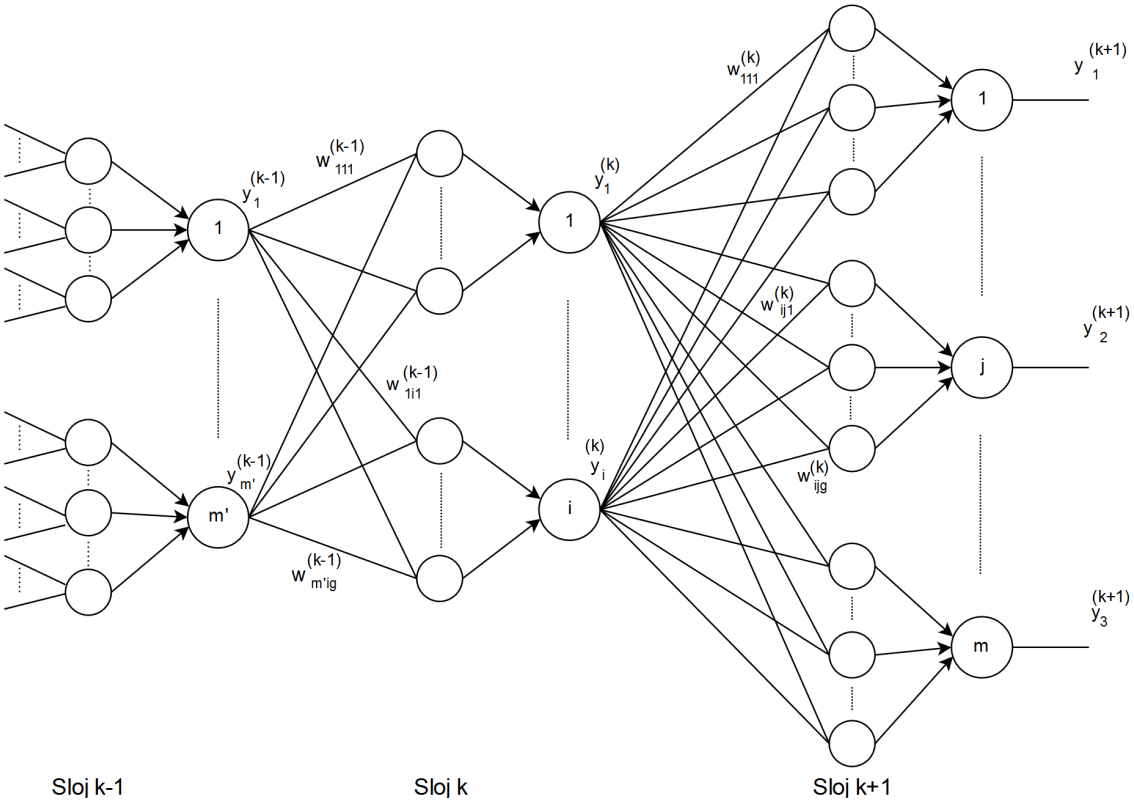
$$\frac{\partial E}{\partial w_{ijg}^{(k)}} = \frac{\partial}{\partial w_{ijg}^{(k)}} \left[\frac{1}{2} \sum_{o=1}^{M} \left(t_o - y_o^{(k+1)} \right)^2 \right] \tag{3.2}$$

$$= \frac{1}{2} \sum_{o=1}^{M} 2 \cdot \left(t_o - y_o^{(k+1)} \right) \cdot (-1) \cdot \frac{\partial y_o^{(k+1)}}{\partial w_{ijg}^{(k)}} \tag{3.3}$$

$$= - \sum_{o=1}^{M} \left(t_o - y_o^{(k+1)} \right) \frac{\partial y_o^{(k+1)}}{\partial w_{ijg}^{(k)}} \tag{3.4}$$

Kako je težina $w_{ijg}^{(k)}$ težina koja spaja i-ti neuron u k-tom sloju i j-ti neuron u $(k+1)$-om sloju, on utječe samo na izlaz neurona j u $(k+1)$-om sloju. Stoga parcijalne derivacije $\frac{\partial y_o^{(k+1)}}{\partial w_{ijg}^{(k)}}$ za slučaj da je $o \neq j$ nemaju utjecaja, iščezavaju te vrijedi:

$$\sum_{o=1}^{M} \frac{\partial y_o^{(k+1)}}{\partial w_{ijg}^{(k)}} = \frac{\partial y_j^{(k+1)}}{\partial w_{ijg}^{(k)}} \tag{3.5}$$



Slika 3.1: Primjer završnih slojeva potpuno povezane unaprijedne neuronske mreže tipa Maxout

Uvrštavanjem u početni izraz dobivamo:

$$\frac{\partial E}{\partial w_{ijg}^{(k)}} = -(t_j - y_j^{(k+1)}) \frac{\partial y_j^{(k+1)}}{\partial w_{ijg}^{(k)}} \tag{3.6}$$

Za izračun derivacije iskoristit ćemo pravilo ulančavanja derivacija:

$$\frac{\partial y_j^{(k+1)}}{\partial w_{ijg}^{(k)}} = \frac{\partial y_j^{(k+1)}}{\partial net_{jg}^{(k+1)}} \cdot \frac{\partial net_{jg}^{(k+1)}}{\partial w_{ijg}^{(k)}} \tag{3.7}$$

Derivacija $\frac{\partial y_j^{(k+1)}}{\partial net_{jg}^{(k+1)}}$ ovisna je o odabranom skupu težina g koje se koriste pri izračunu $net_{jg}^{(k+1)}$ težinske sume $(k+1)$-og sloja. Zbog načina funkcioniranja Maxout-neurona samo skup g_{max} koji daje maksimalnu sumu ima utjecaja na y. Stoga za $\frac{\partial y_j^{(k+1)}}{\partial net_{jg}^{(k+1)}}$ možemo zapisati:

$$\frac{\partial y_j^{(k+1)}}{\partial net_{jg}^{(k+1)}}) = \begin{cases} 0, & g \neq g_{max} \\ 1, & g = g_{max} \end{cases} \tag{3.8}$$

Također, uočimo da je funkcija $net_{jg}^{(k+1)}$ koju računamo kao:

$$net_{jg}^{(k+1)} = y_1^{(k)} \cdot w_{1jg}^{(k)} + y_2^{(k)} \cdot w_{2jg}^{(k)} + ... + y_i^{(k)} \cdot w_{ijg}^{(k)} + ...$$

Zbog svojstava mreže niti jedan od izlaza $y_1^{(k)}, y_2^{(k)}, ...$ pomoću kojih se računa $net_{jg}^{(k+1)}$ ne ovise o težini $w_{ijg}^{(k)}$ te pri deriviranju za nju predstavljaju konstante. Tada za drugi dio parcijalne derivacije dobivamo:

$$\frac{\partial net_{jg}^{(k+1)}}{\partial w_{ijg}^{(k)}} = y_i^{(k)} \tag{3.9}$$

Uvrštavanjem izračunatih parcijalnih derivacija u izraz 3.6 dobivamo:

$$\frac{\partial E}{\partial w_{ijg}^{(k)}} = \begin{cases} 0, & g \neq g_{max} \\ -(t_j - y_j^{(k+1)}) \cdot y_i^{(k)}, & g = g_{max} \end{cases} \tag{3.10}$$

Jednadžbu možemo i prikazati:

$$\frac{\partial E}{\partial w_{ijg}^{(k)}} = \begin{cases} 0, & g \neq g_{max} \\ -\delta_j^{(k+1)} \cdot y_i^{(k)}, & g = g_{max} \end{cases} \tag{3.11}$$

pri čemu je $\delta_j^{(k+1)}$ pogreška j-tog neurona $(k+1)$-og sloja dobivena iz:

$$\delta_j^{(k+1)} = (t_j - y_j^{(k+1)})$$

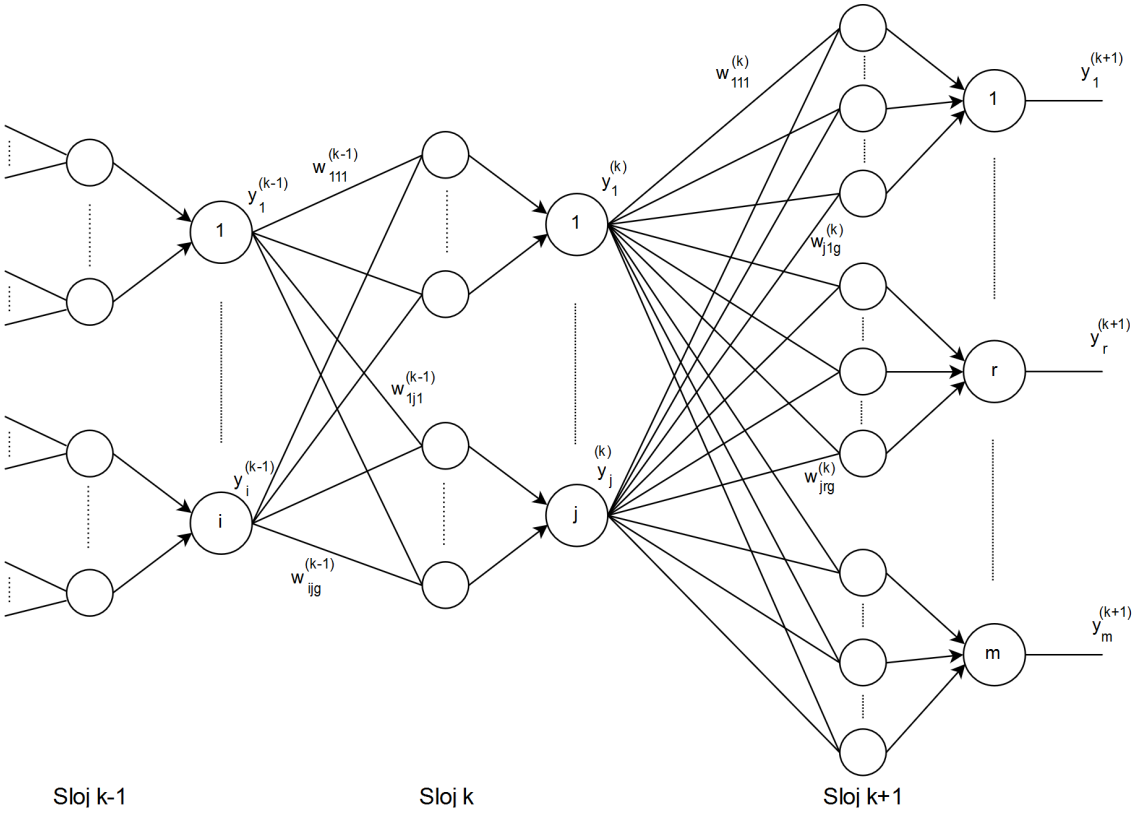
Pravilo ažuriranja težina svodi se samo na mijenjanje skupa težina $w_{ijg_{max}}^{(k)}$ koji je generirao najveći net u j-tom neuronu $(k+1)$-og sloja i ono glasi:

$$w_{ijg}^{(k)} \leftarrow w_{ijg}^{(k)} - \psi \cdot \frac{\partial E}{\partial w_{ijg}^{(k)}} \tag{3.12}$$

$$\leftarrow w_{ijg}^{(k)} - \psi \cdot \left(- \delta_j^{(k+1)} \cdot y_i^{(k)} \right) \tag{3.13}$$

$$\leftarrow w_{ijg}^{(k)} + \psi \cdot \left(\delta_j^{(k+1)} \cdot y_i^{(k)} \right) \tag{3.14}$$

Skriveni slojevi Slučaj za težine skrivenih slojeva ćemo napraviti na primjeru k-tog sloja i težina koje ga povezuju s $(k-1)$-im slojem.



Slika 3.2: Primjer završnih slojeva potpuno povezane unaprijedne neuronske mreže tipa Maxout

Potrebno je izračunati $\frac{\partial E}{\partial w_{ijg}^{(k)}}$:

$$\frac{\partial E}{\partial w_{ijg}^{(k-1)}} = \frac{\partial}{\partial w_{ijg}^{(k-1)}} \left[\frac{1}{2} \sum_{o=1}^{M} \left(t_o - y_o^{(k+1)} \right)^2 \right] \tag{3.15}$$

$$= \frac{1}{2} \sum_{o=1}^{M} 2 \cdot \left(t_o - y_o^{(k+1)} \right) \cdot (-1) \cdot \frac{\partial y_o^{(k+1)}}{\partial w_{ijg}^{(k-1)}} \tag{3.16}$$

$$= - \sum_{o=1}^{M} \left(t_o - y_o^{(k+1)} \right) \frac{\partial y_o^{(k+1)}}{\partial w_{ijg}^{(k-1)}} \tag{3.17}$$

Raspisivanjem $\frac{\partial y_o^{(k+1)}}{\partial w_{ijg}^{(k-1)}}$ pomoću ulančavanja derivacije, formule 3.7 dobivamo:

$$\frac{\partial E}{\partial w_{ijg}^{(k-1)}} = - \sum_{o=1}^{M} \left(t_o - y_o^{(k+1)} \right) \frac{\partial y_o^{(k+1)}}{\partial net_{og}^{(k+1)}} \cdot \frac{\partial net_{og}^{(k+1)}}{\partial w_{ijg}^{(k-1)}} \tag{3.18}$$

Raspisivanjem prve derivacije, derivacije max funkcije po $net_{og}^{(k+1)}$ bodivamo:

$$\frac{\partial y_o^{(k+1)}}{\partial net_{og}^{(k+1)}}) = \begin{cases} 0, & g \neq g_{max} \\ 1, & g = g_{max} \end{cases}$$

Gdje g_{max} označava skup koji je daje maksimalan net za o-ti neuron $(k+1)$-og sloja. Sljedeća derivacija preostaje jedino u slučaju ako je odabran $g = g_{max}$. Da bismo razriješili preostalu parcijalnu derivaciju, fiksirajmo na trenutak o te pogledajmo kako se računa $net_{og_{max}}^{(k+1)}$. Vrijedi:

$$net_{og_{max}}^{(k+1)} = y_1^{(k)} \cdot w_{1og_{max}}^{(k)} + y_2^{(k)} \cdot w_{2og_{max}}^{(k)} + ... + y_j^{(k)} \cdot w_{jog_{max}}^{(k)} + ...$$

Težina $w_{ijg}^{(k-1)}$ koja spaja i-ti neuron u $(k-1)$-om sloju i j-ti neuron u k-tom sloju utječe samo na izlaz j-tog neurona $(k+1)$-om sloju. Zbog toga je parcijalna derivacija $\frac{\partial y_o^{(k)}}{\partial w_{ijg}^{(k-1)}} = 0$ za $o \neq j$. Derivacija $net_{og}^{(k+1)}$ glasi:

$$\frac{\partial net_{og}^{(k+1)}}{\partial w_{ijg}^{(k)}} = \frac{\partial}{\partial w_{ijg}^{(k)}} \Big(y_1^{(k)} \cdot w_{1og_{max}}^{(k)} + ... + y_j^{(k)} \cdot w_{jog_{max}}^{(k)} + ... \Big) \tag{3.19}$$

$$= w_{jog_{max}}^{(k)} \cdot \frac{\partial y_j^{(k)}}{\partial w_{ijg}^{(k-1)}} \tag{3.20}$$

Uvrštavanjem u preostalu derivaciju dobivamo:

$$\frac{\partial E}{\partial w_{ijg}^{(k-1)}} = -\sum_{o=1}^{M} \Big(t_o - y_o^{(k+1)} \Big) w_{jog_{max}}^{(k)} \cdot \frac{\partial y_j^{(k)}}{\partial w_{ijg}^{(k-1)}} \tag{3.21}$$

$$= -\frac{\partial y_j^{(k)}}{\partial w_{ijg}^{(k-1)}} \sum_{o=1}^{M} \Big(t_o - y_o^{(k+1)} \Big) w_{jog_{max}}^{(k)} \tag{3.22}$$

$$= -\frac{\partial y_j^{(k)}}{\partial w_{ijg}^{(k-1)}} \sum_{o=1}^{M} \delta_o^{(k+1)} \cdot w_{jog_{max}}^{(k)} \tag{3.23}$$

Derivaciju $\frac{\partial y_j^{(k)}}{\partial w_{ijg}^{(k-1)}}$ možemo riješiti pomoću ulančavanja derivacija, te rješavanja parcijalnih derivacija koje su već prije riješene:

$$\frac{\partial y_j^{(k)}}{\partial w_{ijg}^{(k-1)}} = \frac{\partial y_j^{(k)}}{\partial net_{jg}^{(k)}} \cdot \frac{\partial net_{jg}^{(k)}}{\partial w_{ijg}^{(k-1)}} \tag{3.24}$$

Kako je $y_j^{(k)}$ ovisan samo o skupu težina g_{max} koji je generirao najveći net u j-tom neuronu k-og sloja, promjena koju ćemo računati će se jedino događati na ovom "maksimalnom" skupu težina. Preostaje derivacija s "maksimalnim" skupom:

$$\frac{\partial y_j^{(k)}}{\partial w_{ijg_{max}}^{(k-1)}} = \frac{\partial net_{jg_{max}}^{(k)}}{\partial w_{ijg_{max}}^{(k)}} \tag{3.25}$$

$$= \frac{\partial}{\partial w_{ijg_{max}}^{(k-1)}} \left(y_1^{(k-1)} \cdot w_{1jg_{max}}^{(k-1)} + ... + y_i^{(k-1)} \cdot w_{ijg_{max}}^{(k-1)} + ... \right) \tag{3.26}$$

$$= y_i^{(k-1)} \tag{3.27}$$

Uvrštenjem u formulu 3.23 dobivamo:

$$\frac{\partial E}{\partial w_{ijg_{max}}^{(k-1)}} = -y_i^{(k-1)} \sum_{o=1}^{M} \delta_o^{(k+1)} \cdot w_{jog_{max}}^{(k)} \tag{3.28}$$

$$= -y_i^{(k-1)} \cdot \delta_j^{(k)} \tag{3.29}$$

gdje $\delta_j^{(k)}$ predstavlja pogrešku j-tog neurona iz k-tog sloja prema $(k+1)$-om sloju:

$$\delta_j^{(k)} = \sum_{o=1}^{M} \delta_o^{(k+1)} \cdot w_{jog_{max}}^{(k)}$$

Pravilo ažuriranja težina svodi se, opet, na mijenjanje skupa težina g_{max} koji je generirao najveći net u j-tom neuronu k-og i ono glasi:

$$w_{ijg_{max}}^{(k-1)} \leftarrow w_{ijg_{max}}^{(k-1)} - \psi \cdot \frac{\partial E}{\partial w_{ijg_{max}}^{(k-1)}} \tag{3.30}$$

$$\leftarrow w_{ijg_{max}}^{(k-1)} - \psi \cdot \left(- y_i^{(k-1)} \cdot \delta_j^{(k)} \right) \tag{3.31}$$

$$\leftarrow w_{ijg_{max}}^{(k)} + \psi \cdot \left(y_i^{(k-1)} \cdot \delta_j^{(k)} \right) \tag{3.32}$$

Razmotrimo li umjesto neurona iz zadnjeg skrivenog sloja neki od neurona iz prethodnih slojeva, dolazi se do strukturno istih formula: za svaki neuron iz skrivenog sloja l, pogreška se računa kao derivacija njegove prijenosne funkcije pomnožena težinskom sumom pogrešaka neurona sljedećeg sloja na koje je promatrani neuron direktno spojen.

Grupni i algoritam propagacije pogreške unatrag s mini-grupama Prije izvedene formule izvedene su za stohastički algoritam propagacije pogreške unatrag u kojem se za svaki par podataka radi promjena težina. Za implementaciju preostale dvije

inačice algoritma izvod je skoro u potpunosti isti osim dodatne sume u kojoj se zbrajaju gradijenti trenutne grupe podataka. Grupa podataka veličine P za učenje trenutnog gradijenta gdje je $P \in \{1...N\}$ po formulama neurona izlaznog sloja:

$$w_{ijg}^{(k)} \leftarrow w_{ijg}^{(k)} + \psi \cdot \left(\frac{1}{P} \sum_{s=1}^{P} \delta_{s,j}^{(k+1)} \cdot y_{s,i}^{(k)} \right) \tag{3.33}$$

$$\leftarrow w_{ijg}^{(k)} + \eta \cdot \left(\sum_{s=1}^{P} \delta_{s,j}^{(k+1)} \cdot y_{s,i}^{(k)} \right) \tag{3.34}$$

i formulama skrivenih slojeva:

$$w_{ijg}^{(k)} \leftarrow w_{ijg}^{(k)} + \psi \cdot \left(\frac{1}{P} \sum_{s=1}^{P} \delta_{s,j}^{(k)} \cdot y_{s,i}^{(k-1)} \right) \tag{3.35}$$

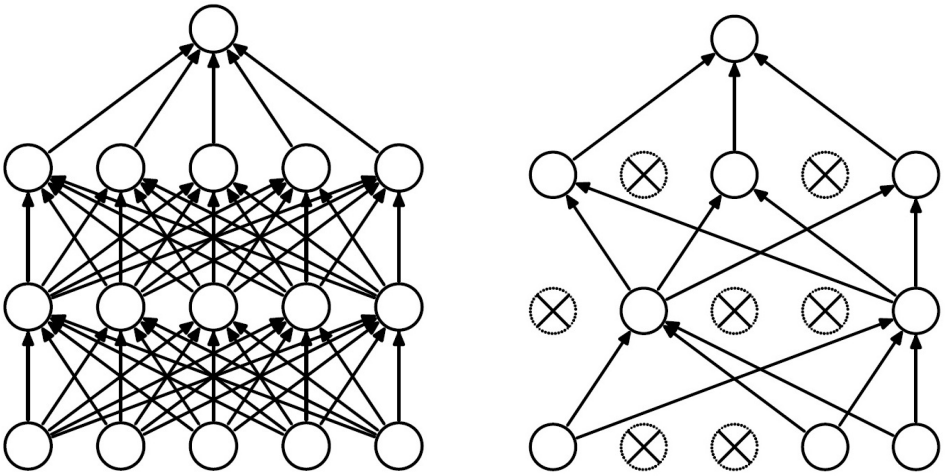
$$\leftarrow w_{ijg}^{(k)} + \eta \cdot \left(\sum_{s=1}^{P} \delta_{s,j}^{(k)} \cdot y_{s,i}^{(k-1)} \right) \tag{3.36}$$

gdje je η novi faktor učenja gradijenta a s oznaka trenutnog skupa podataka veličine P. Uočimo da se smanjenjem parametra P na $P = 1$ algoritam pretvara u stohastički algoritam propagacije pogreške unatrag, dok povećanjem P mini-grupa na $P = N$ dobivamo obični algoritam propagacije pogreške unatrag.

3.2. Tehnika *Dropout*

Duboke neuronske mreže sadrže više nelinearnih skrivenih slojeva, što ih čini vrlo ekspresivnim modelima koji imaju sposobnost učenja vrlo kompliciranih veza između ulaza i izlaza. Skupovi podataka su ograničeni i mnoge kompliciranе veze bit će rezultat smetnji uzorkovanja, kojih nema u stvarnim podacima. Predugo učenje mreža na takvim skupovima podataka naučit će mreže dobro prepoznavati samo skup za učenje i one više neće biti sposobne generalizirati. Ovakav problem nazivamo prekomjernim učenjem mreže (engl. *overffitting*). Neke od metoda koje pokušavaju riješiti ovaj problem su: zaustavljanje treniranja nakon što mreža počne davati sve lošije rezultate na skupu podataka za validaciju, uvođenjem kazni težina poput L1 i L2 regularizacija te dijeljenje težina prema (Nowlan i Hinton, 1992).

Kombiniranje modela gotovo uvijek poboljšava performanse strojnog učenja, ali korištenjem velikih neuronskih mreža jednostavna ideja uravnotežavanja izlaza puno odvojenih i zasebno treniranih mreža je i za današnju tehnologiju previše zahtjevno. Kombiniranje nekoliko modela najkorisnije je kada se zasebni modeli razlikuju. Kako



Slika 3.3: Usporedba potpuno povezane mreže s mrežom dobivenom istanjivanjem pomoću tehnike Dropout, preuzeto iz (Srivastava et al., 2014)

bi se napravile različite mreže trebaju ili imati različite arhitekture ili ulazni skupovi podataka trebaju biti drugačiji. Dodatan problem koji se pojavljuje je namještanje parametara svake neuronske mreže, što je kompliciran proces. Velike neuronske mreže zahtijevaju mnogo podataka za učenje, podataka koji nisu uvijek dostupni u dovoljno velikim količinama. Recimo da smo i uspjeli u razumnom vremenu riješiti ove probleme i istrenirati mreže, korištenje svih mreža odjednom u vrijeme testiranja nije primjenjivo, pogotovo u aplikacijama kod kojih je brzina odluke vrlo važna.

Tehnika Dropout (Hinton et al., 2012) pokušava riješiti probleme prekomjernog učenja mreže i pruža učinkovit i jednostavan način učenja velikih skupova modela dijeljenih parametara i objedinjavanja predikcija ovih modela. Dropout primjenjen na višeslojne perceptrone i duboke neuronske mreže je poboljšao klasifikaciju od velikih skupova objekata pa sve do audio klasifikacije (Cai et al., 2013) i (Cai et al., 2014). Primjena tehnike Dropout bi na skoro svim modelima trebala postići bar skromna poboljšanja performansi (Goodfellow et al., 2013). Dropout je najučinkovitiji pri uzimanju relativno velikih koraka u parametarskom prostoru gdje svaku promjenu možemo gledati kao značajnu promjenu različitih modela na različitim pod-skupovima ulaznih podataka što se razlikuje od klasičnog gradijentnog spusta u kojem jedan model postupno napreduje malim koracima.

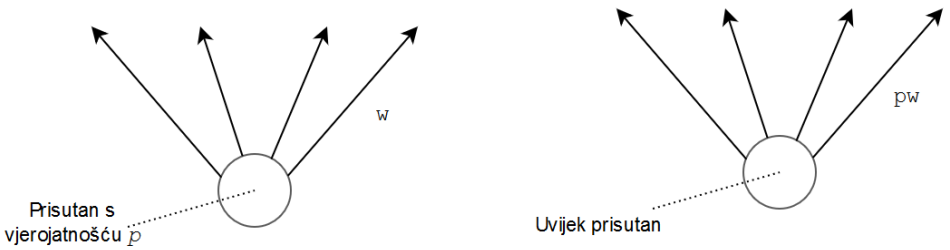
Motivacija Motivacija za tehniku Dropout dolazi iz principa razmnožavanja. Pri seksualnom razmnožavanju dijete dobiva pola gena od svakog od roditelja uz vrlo

malu količinu slučajne mutacije. Aseksualna alternativa je nastanak djece prosljeđivanjem malo mutiranih gena jednog jedinog roditelja. Čini se kako bi aseksualna vrsta razmnožavanja trebala biti bolja jer bolji se genski materijal odmah u cijelosti prosljeđuje dalje djetetu i tako optimira dobrotu jedinke dok je kod seksualne reprodukcije velika vjerojatnost da će se neke uspješne skupine gena prekinuti, pogotovo ako su oni veliki, te smanjiti dobrotu novonastale jedinke čiji su roditelji razvili komplicirana i dobra svojstva.

Seksualno razmnožavanje je unatoč spomenutim pretpostavkama način razmnožavanja većine naprednih organizama. Objašnjenje uspješnosti seksualne reprodukcije mogla bi biti činjenica da tijekom dugog perioda kriterij prirodne selekcije nije individualna uspješnost genskih skupina već sposobnost miješanja gena. Sposobnost skupina gena da uspješno funkcioniraju s drugim potpuno slučajnim skupom čini ih otpornijima na neočekivane okolnosti. Kako se geni ne mogu osloniti na veliku skupinu partnera koji će stalno biti prisutni, prisiljeni su sami naučiti raditi nešto korisno bez obzira jesu li drugi geni prisutni ili ne. Uloga seksualne reprodukcije nije samo da bi se geni proširili populacijom već i smanjivanje kompleksnih ovisnosti koje bi mogle ograničiti poboljšanje sposobnosti jedinke. Neuron u neuronskim mrežama učenim uz tehniku Dropout, slično prethodno spomenutim genima, mora naučiti raditi i pridonositi rezultatu bez obzira na populaciju susjednih jedinica. Neuroni bi ovim postupkom trebali doprinijeti dobrim svojstvima bez oslanjanja na druge neurone koji bi popravljali njegove pogreške.

Opis algoritma Ispuštanjem neurona privremeno ga uklanjamo iz mreže zajedno s pripadnim ulaznim i izlaznim težinama. Odabir neurona koji će se ispustiti iz mreže je slučajan. Možemo reći da je svaki neuron sadržan u mreži s vjerojatnošću p neovisno o drugim neuronima. Ovisno o tipu neuronske mreže može se primjenjivati različita vjerojatnost zadržavanja u mreži ovisno o tome da li se neuron nalazi u ulaznom ili skrivenim slojevima. Izlazni sloj nam je u cijelosti potreban i neuroni u njemu su fiksni. Tehnika Dropout smanjuje broj neurona u mreži i od iste mreže dobiva se velik broj drugih mreža koje dijele težine. Mrežu koja ima n neurona možemo gledati kao kolekciju 2^n istanjenih mreža. Kod svakog testiranja na istom paru podataka, nova mreža se ispituje i trenira. Treniranje neuronske mreže tehnikom Dropout mogli bi vidjeti kao treniranje kolekcije 2^n različitih mreža dijeljenih težina, gdje se svaka od mreža vrlo rijetko ili nikad ne trenira.

Za vrijeme testiranja mreže usrednjavanje nije izvedivo za eksponencijalno mnogo



Slika 3.4: Prikaz neurona za vrijeme učenja i testiranja mreže uz dodatak tehnike Dropout

mreža. Umjesto toga koristit će se aproksimacija, koja se pokazala dosta dobrom i to pomoću jedne mreže. Težine neuronske mreže bit će skalirane verzije istreniranih težina. Ako je neuron sadržan u mreži s vjerojatnošću p tijekom treniranja, za vrijeme testiranja je sigurno sadržan u mreži ali se njegove izlazne težine skaliraju vjerojatnošću p. Ovakvim skaliranjem omogućili smo spajanje 2^n mreža dijeljenih težina u jednu testnu mrežu.

Princip rada algoritma Tehnika Dropout koristi gradijentni spust kao ostale neuronske mreže. Jedina je razlika da se svaki gradijent računa za mini-grupu ulaznih podataka koji se propuštaju kroz promjenjenu početnu mrežu. Koriste se zajedničke težine svih mreža.

Algoritam propagacije pogreške unatrag potpomognut tehnikom Dropout možemo podijeliti na nekoliko koraka. Na početku mini-grupe ulaznih podataka generiramo masku veličine neuronske mreže. Maska predstavlja potpunu mrežu u kojoj su neki neuroni ispušteni. Par podataka iz mini-grupe stavlja se na ulaz istanjene mreže i propagira do kraja. Za dani par podataka gradijent se računa kao i u običnom algoritmu propagacije pogreške unatrag. Kada se odredi doprinos svih parova podataka iz mini-grupe ulaznih podataka, ukupni gradijent se zbraja težinama mreže. Postupak se ponavlja dok nije cijeli ulazni skup podataka iskorišten te ako nakon toga nisu zadovoljeni uvjeti zaustavljanja, navedeni koraci se ponavljaju ponovno za cijeli ulazni skup.

4. Eksperimenti

Implementirana je neuronska mreža tipa Maxout, uključujući i Maxout-neurone kako bi se ispitao njihov rad na regresijskim i klasifikacijskim problemima. Objektno orijentirana paradigma nije u potpunosti primjenjena zbog potrebne jednostavnosti korištenog modela mreže za brže izvođenje algoritama.

Mreže tipa Maxout nemaju problem koji se javlja u mrežama s neuronima koji imaju sigmoidalnu funkciju, a to je sve veće gušenje gradijenta što se on dublje propagira u mrežu. Poznat je pod nazivom problem nestajućeg gradijenta. Problem koji se javljao pri testiranju mreža tipa Maxout velika je osjetljivost težina o stopi učenja koju je potrebno mijenjati pri svakoj promjeni strukture mreže. Računanje gradijenata mreža tipa Maxout zbog utjecaja sume net može divergirati poslije nekoliko promjena težina ukoliko ga stopa učenja ne umanji. Stavljanje sigmoidalne prijenosne funkcije na izlaz mreže, što je objašnjeno u poglavlju klasifikacije, imalo je pozitivnog utjecaja na gradijent koji je u izlaznom sloju bio mali tako da su u propagiranju unatrag svi gradijenti bili manje osjetljivi na promjene strukture mreže.

Rad mreže se ispitivao na problemima regresije i klasifikacije. Primjeri problema i načini na koji su riješeni, objašnjeni su u sljedećim poglavljima.

4.1. Problemi funkcijske regresije

Funkcijska regresija je postupak pri kojem na temelju danih podataka, neuronska mreža pokušava napraviti aproksimaciju funkcije. Maxout-neuron sam po sebi je dobar aproksimator što znači da bi se funkcijska regresija vrlo lako trebala moći ostvariti neuronskim mrežama tipa Maxout. Obrađena su dva problema koji pokazuju mogućnosti Maxout-neurona i mreža građenih od njih.

4.1.1. Jednostavna linearna funkcija

Funkcija koju bi Maxout-neuron trebao s lakoćom aproksimirati je funkcija:

$$f = max \left\{ \begin{array}{l} -2x - 1 \\ 1.5x - 5.2 \end{array} \right\}$$

Mreža koja se koristila za ovaj problem imala je jedan Maxout-neuron s dva skupa težina koja bi učenjem trebala poprimiti vrijednosti koje odgovaraju linearnim dijelovima funkcije f.

Za učenje mreže koristio se algoritam propagacije greške unatrag s mini-grupama. Neki od parametara koji su se koristili pri učenju mreže navedeni su u nastavku.

Veličina grupa: 4

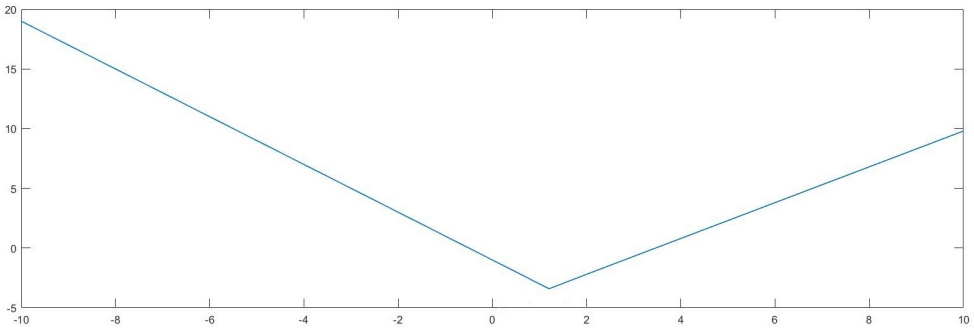
Stopa učenja: 0.015

Broj epoha: 50

Veličina skupa za učenje: 100

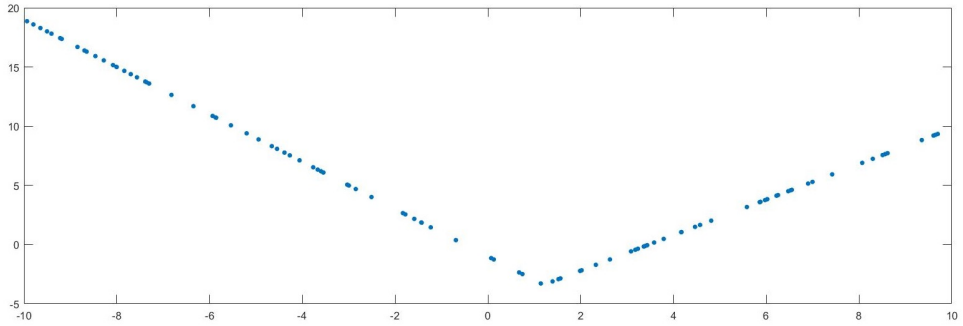
Veličina skupa za testiranje: 50

Skup podataka za učenje koji se sastoji od parova ulaza i izlaza generiran je slučajno za područje ulazne vrijednosti od -10 do 10. Funkciju i skup za učenje možemo vidjeti na slikama 4.1 i 4.2.



Slika 4.1: Stvarni izgled funkcije

Mreža je nakon učenja koje je trajalo 50 epoha, imala srednju kvadratnu pogrešku $1.874 \cdot 10^{-6}$ te omjer izlaznih vrijednosti koje nisu odstupale više od 2.5% od tražene vrijednosti (vrijednosti funkcije f) i stvarnih traženih vrijednosti je $50/50$. Možemo reći da je od 50 uzoraka za svih 50 uzoraka izlazna vrijednost bila unutar 2.5% od



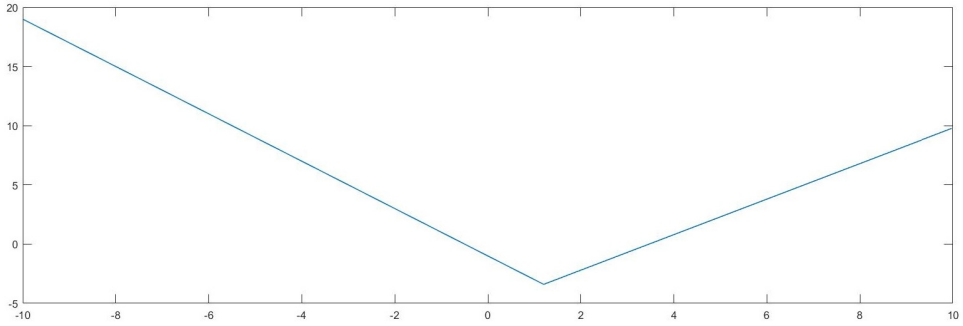
Slika 4.2: Parovi uzoraka za učenje

zadane vrijednost. Težine mreže nakon učenja možemo vidjeti u nastavku.
1. skup težina

$$w_0 = -5.1968$$
$$w_1 = 1.4997$$

2. skup težina

$$w_0 = -0.9999$$
$$w_1 = -1.9999$$



Slika 4.3: Dobivena izlazna funkcija

Možemo vidjeti da je mreža u potpunosti aproksimirala danu funkciju, slika 4.3. Vrijednosti težina za svaki od skupova su poprimili vrijednosti koeficijenata jednog od pravaca koji određuju ulaznu funkciju. Raspon testnih primjera mogao bi se proširiti i mreža bi još davala dobre rezultate.

4.1.2. Sinusoidalna funkcija

Funkciju f bi jedan Maxout-neuron trebao biti u mogućnosti aproksimirati. Preciznost aproksimacije ovisi o broju skupova težina:

$$f = 2 \cdot \sin(x/3).$$

Mreža koja se koristila za ovaj problem imala je dva sloja Maxout-neurona: skriveni koji se sastoji od dva Maxout-neurona s po petnaest skupova težina te izlazni sloj s jednim Maxout-neuronom i dva skupa težina.

Za učenje mreže koristio se algoritam propagacije greške unatrag s mini-grupama. Neki od parametara koji su se koristili pri učenju mreže navedeni su u nastavku.

Veličina grupa: 5

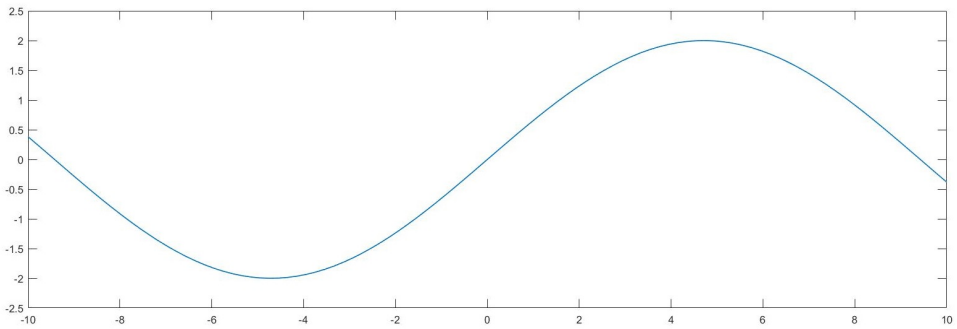
Stopa učenja: 0.01

Broj epoha: 1000000

Veličina skupa za učenje: 100

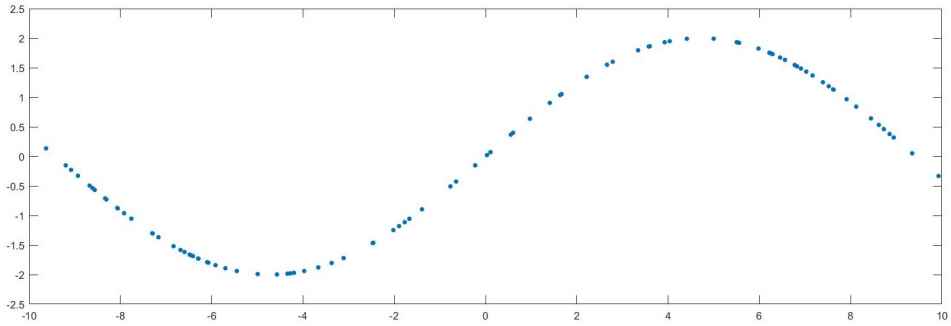
Veličina skupa za testiranje: 50

Skup podataka za učenje koji se sastoji od parova ulaza i izlaza generiran je slučajno za područje ulazne vrijednosti od -10 do 10. Funkcija i skup za učenje možemo vidjeti na slikama 4.4 i 4.5.

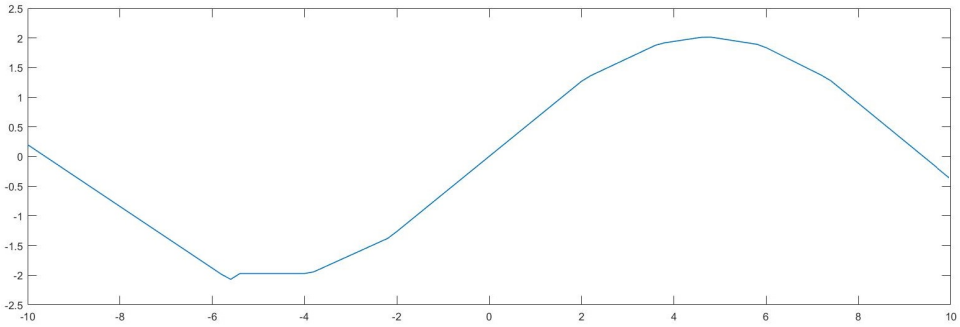


Slika 4.4: Stvarni izgled funkcije

Mreža je nakon učenja koje je trajalo 1000000 epoha, imala srednju kvadratnu pogrešku $1.424 \cdot 10^{-3}$ te omjer izlaznih vrijednosti koje nisu odstupale više od 2.5% od tražene vrijednosti (vrijednosti funkcije f) i traženih vrijednosti je $42/50$.



Slika 4.5: Parovi uzoraka za učenje



Slika 4.6: Dobivena izlazna funkcija

Dobiveni rezultati, prikazani na slici 4.6, odgovaraju pretpostavkama da je Maxout-neuron u mogućnosti proizvoljno približno opisati kontinuiranu krivulju. Što se više skupova težina uzme neuron bi trebao moći preciznije opisati funkciju. Funkcija je kompliciranija pa je trebao puno veći broj iteracija nego u prošlom problemu. Mreža tipa Maxout s više slojeva bi vrlo brzo uspjela aproksimirati funkciju (nekoliko stotina iteracija za rezultate koji su ovdje postignuti).

4.2. Problemi klasifikacije

Klasifikacija kao problem nije najprirodniji za Maxout funkciju koja nije ograničena dok se pri klasifikacijama najčešće uzimaju izlazi koji su ograničeni unutar 0 i 1. Maxout-funkcija kao linearna funkcija vrlo teško dolazi do traženih diskretnih vrijednosti klasifikacije te se pri korištenju mreža tipa Maxout najčešće na izlazni sloj dodaje funkcija Softmax (Goodfellow et al., 2013) ili dodatni sigmoidalni filter.

Sigmoidalna funkcija ograničava izlazne vrijednosti u rasponu od 0 do 1. Algoritmi za računanje gradijenata trebaju mijenjati samo oblik funkcije na izlaznom sloju dok matematički izrazi za unutarnje slojeve ostaju isti. Ulančavanjem nove funkcije u parcijalne derivacije izraza 3.7 dobivamo:

$$\frac{\partial y_j^{(k+1)}}{\partial w_{ijg}^{(k)}} = \frac{\partial y_j^{(k+1)}}{\partial q_j^{(k+1)}} \cdot \frac{\partial q_j^{(k+1)}}{\partial net_{jg}^{(k+1)}} \cdot \frac{\partial net_{jg}^{(k+1)}}{\partial w_{ijg}^{(k)}}. \tag{4.1}$$

Gdje je $q_j^{(k+1)}$ izlaz Maxout funkcije na koju je doveden izlaz Maxout-neurona. Druga derivacija je ista kao u 3.8. Prva derivacija iznosi:

$$\frac{\partial y_j^{(k+1)}}{\partial q_j^{(k+1)}} = y_j^{(k+1)} \cdot (1 - y_j^{(k+1)}) \tag{4.2}$$

Dobivamo izraz za grešku $\delta_j^{(k+1)}$:

$$\delta_j^{(k+1)} = y_j^{(k+1)}(1 - y_j^{(k+1)})(t_j - y_j^{(k+1)}).$$

Sljedeći primjer biti će riješen uz implementaciju sigmoidalne funkcije na izlazu mreže tipa Maxout za što su bile potrebne male izmjene programskog koda.

4.2.1. Klasifikacija dvodimenzionalnog prostora podijeljenog pravcima

U ovom primjeru prostor je podijeljen u pet klasa koje su odvojene linearnim funkcijama. Mreža koja se koristila za ovaj problem imala je tri sloja Maxout-neurona, prvi od deset Maxout-neurona s po šest skupova težina, drugi od deset Maxout-neurona s po pet skupova težina i izlazni sloj od pet Maxout-neurona s po tri skupa težina. Mreža je dovoljno velika da se počne primjenjivati tehniku Dropout uz male vjerojatnosti ispadanja neurona u skrivenim slojevima i zadržavanjem svih neurona u ulaznom sloju u kojem imamo premalo neurona da bi se i oni izbacivali.

Za učenje mreže koristio se algoritam propagacije greške unatrag s mini-grupama i tehnikom Dropout. U nastavku su navedeni neki od parametara koji su se koristili pri učenju mreže.

Vjerojatnost ispadanja neurona ulaznog sloja: 0

Vjerojatnost ispadanja neurona skrivenih slojeva: 0.1

Veličina grupa: 10

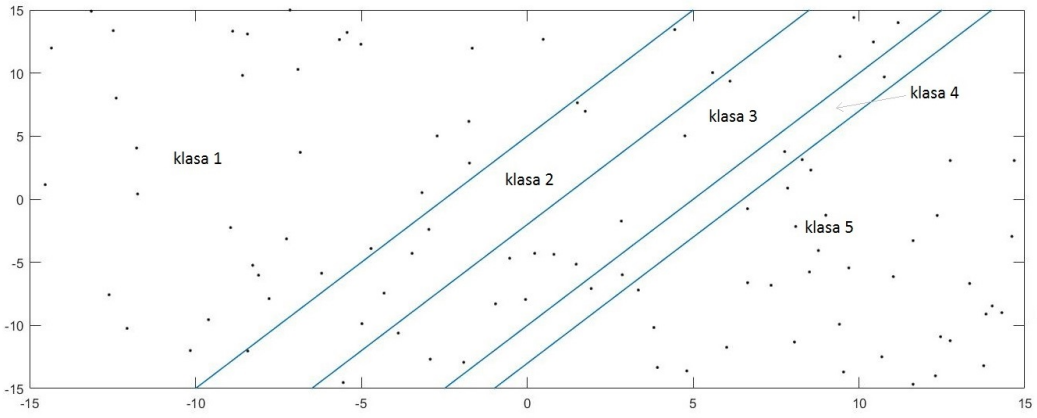
Stopa učenja: 0.007

Broj epoha: 50000

Veličina skupa za učenje: 100

Veličina skupa za testiranje: 50

Skup podataka za učenje koji se sastoji od parova ulaza i izlaza generiran je slučajno za područje ulazne vrijednosti od -15 do 15. Ulazni podaci se sastoje od dvije dimenzije a izlazni od pet, po jedna za svaku od pet klasa. Dobivene točke su klasificirane i dodijeljena im je izlazna vrijednost svake klase, 1 ukoliko točka spada u tu klasu ili 0 u suprotnom. Podjelu prostora na klase i skup za učenje možemo vidjeti na slici 4.7.



Slika 4.7: Podjela kordinata na klase i skup za učenje

Mreža je nakon učenja koje je trajalo 50000 epoha, imala srednju kvadratnu pogrešku $1.6 \cdot 10^{-3}$ te omjer izlaznih vrijednosti koje nisu odstupale više od 5% od tražene vrijednosti (vrijednosti funkcije f) i traženih vrijednosti je $42/50$.

Korištenje tehnike Dropout usporava dolazak mreže do manje kvadratne pogreške zbog neuravnoteženosti koje ispadanje neurona donosi. Kvadratne pogreške na kraju

učenja su u većem broju pokušaja manje uz korištenje tehnike Dropout zbog boljeg istraživanja težina. Najbolje rezultate donosi korištenje tehnike Dropout čiji se utjecaj s vremenom smanjuje. Vjerojatnost ispadanja neurona u tom slučaju nije fiksna nego se postupno smanjuje te tako na početku imamo više istraživanja i poticanja neurona na rad, a kasnije kada je mreža bolja ne uništava joj se kvaliteta ispadanjem neurona nego se pokušava što bolje naučiti poput algoritma propagacije pogreške unatrag bez primjene tehnike Dropout. Mreža tipa Maxout postigla je dobre rezultate i manjim dimenzijama u kojima bi vjerojatnost ispadanja neurona bila vrlo mala.

4.2.2. Klasifikacija dvodimenzionalnog prostora podijeljenog ugnježđivanjem podprostora

Prostor je u ovom primjeru podijeljen u pet klasa koje su predstavljene ugnježđenim područjima. Mreža koja se koristila za ovaj problem imala arhitekturu kao i mreža u prošlom primjeru. Tijekom učenja primjenjivala se tehnika Dropout uz male vjerojatnosti ispadanja neurona u skrivenim slojevima i zadržavanjem svih neurona u ulaznom sloju.

Za učenje mreže koristio se algoritam propagacije greške unatrag s mini-grupama i tehnikom Dropout. Neki od parametara koji su se koristili pri učenju mreže navedeni su u nastavku.

Vjerojatnost ispadanja neurona ulaznog sloja: 0

Vjerojatnost ispadanja neurona skrivenih slojeva: 0.2

Veličina grupa: 10

Stopa učenja: 0.007

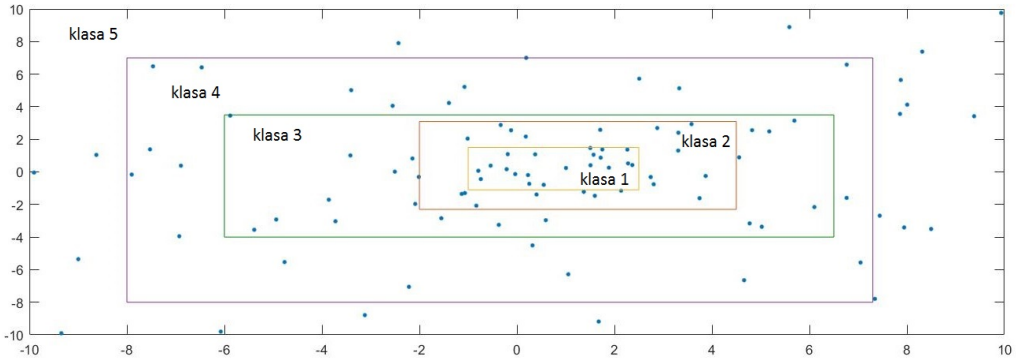
Broj epoha: 20000

Veličina skupa za učenje: 100

Broj predstavnika svake klase: 20

Veličina skupa za testiranje: 50

Skup podataka za učenje koji se sastoji od parova ulaza i izlaza generiran je slučajno za područje ulazne vrijednosti od -10 do 10. Ulazni podaci se sastoje od dvije dimenzije a izlazni od pet, po jedna za svaku od pet klasa. Dobivene točke su klasificirane i dodijeljena im je izlazna vrijednost svake klase, 1 ukoliko točka spada u tu klasu ili 0 u suprotnom. Podjelu prostora na klase i skup za učenje možemo vidjeti na slici 4.8.



Slika 4.8: Podjela kordinata na klase i skup za učenje

Mreža je nakon učenja koje je trajalo 20000 epoha, imala srednju kvadratnu pogrešku $3,353 \cdot 10^{-6}$ te omjer izlaznih vrijednosti koje nisu odstupale više od 5% od tražene vrijednosti (vrijednosti funkcije f) i traženih vrijednosti je $38/50$. Točke u kojima mreža nije dobro klasificirala nalazile su se u graničnim područjima klasa koja mreža nije dobro odredila. Povećanjem podatkovnog skupa za učenje dobili bi točnije klasificiranje.

5. Zaključak

Tehnologija je svakim danom sve brža i sve više napreduje no kako tehnologija napreduje tako ljudi pokušavaju riješiti sve veće i nejasnije probleme za koje nije moguće napraviti algoritme koji bi ih u brzom vremenu riješili. Neuronske mreže koje su inspirirane prirodom dio su računarstva koji je trenutno vrlo popularan i svugdje se koristi. Njihove mogućnosti su velike i rastu razvojem novih vrsta mreža poput mreža tipa Maxout i algoritama koji omogućuju njihovo kvalitetnije učenje.

Ovaj rad se bavio neuronskim mrežama tipa Maxout i njihovim učenjem pomoću algoritma propagacije pogreške unatrag uz pomoć tehnike Dropout. Neuronske mreže tipa Maxout rješavaju problem nestajućeg gradijenta, ali s porastom dubine mreže imaju problema s naglim porastom gradijenta.

Napravljena je implementacija mreža i algoritma učenja te je ispitana na raznim problemima klasifikacije i regresije. Maxout-neuroni su se pokazali vrlo dobrima u problemima regresije. Za klasifikaciju neuronskim mrežama tipa Maxout potrebno je na izlaz mreže staviti sigmoidalnu funkciju koja će ograničiti izlazne vrijednosti i omogućiti lakše učenje mreže.

Tehnika Dropout nastala je kao pokušaj rješavanja problema prekomjernog učenja i pruža jednostavan način učenja različitih skupova modela dijeljenih parametara. Tehnika pospješuje učenje mreža unošenjem slučajnih promjena u mreže. Primjena ima većeg utjecaja na mreže koje imaju veći broj neurona i kompliciraniju arhitekturu.

LITERATURA

M. Cai, Y. Shi, J. Kang, J. Liu, i T. Su. Convolutional maxout neural networks for low-resource speech recognition. U *Chinese Spoken Language Processing (ISCSLP), 2014 9th International Symposium on*, stranice 133–137, Sept 2014. doi: 10.1109/ISCSLP.2014.6936676.

Meng Cai, Yongzhe Shi, i Jia Liu. Deep maxout neural networks for speech recognition. 2013.

Ian J. Goodfellow, Warde-Foley, D. M. Mirza, A. Courville, i Y. Bengio. Maxout networks. 2013.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, i Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012. URL `http://arxiv.org/abs/1207.0580`.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, i Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. URL `http://jmlr.org/papers/v15/srivastava14a.html`.

Marko Čupić, Bojana Dalbelo Bašić, i Marin Golub. *Neizrazito, evolucijsko i neuroračunarstvo*. 2013.

Neuronske mreže izgrađene maxout neuronima

Sažetak

Umjetne neuronske mreže danas pospješuju napredak umjetne inteligencije. Njihova primjena je prisutna u velikoj većini industrija gdje rješavaju različite probleme. Neuronske mreže nisu primjenjive u svim situacijama. Maxout-neuroni koji su obrađeni u ovom radu, nova su vrsta neurona koja uz tehniku Dropout pokušavaju riješiti određene probleme mreža. Napravljena je implementacija neuronske mreže tipa Maxout i algoritma propagacije pogreške unatrag uz korištenje tehnike Dropout. Mreža je istestirana na raznim problemima regresije i klasifikacije gdje je i uz manju kompleksnost pokazala vrlo dobre rezultate.

Ključne riječi: Neuronske mreže, Maxout, neuroni, Dropout, algoritam propagacije pogreške unatrag, univerzalni aproksimator.

Maxout Neural Networks

Abstract

Nowadays artificial neural networks foster great advancements in field of Artificial Inteligence. Its application can be seen in most industries where is employed to solve diverse problems, but still there are some for which neural networks are not working. Maxout neurons which are researched in this work are new type of neurons which in association with Dropout training try to solve some of the problems other types of neural networks encounter. In this work Maxout neural networks and Backpropagation algorithm enhanced with Dropout technique have been implemented. Neural networks have been tested on both regression and classification problems and have achieved great results even with little network complexity.

Keywords: Neural networks, Maxout, neuron, Dropout, Backpropagation algorithm, universal aproximator