Usporedba algoritama neuronskih mreža

Đuranec, Erik

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Erik Đuranec

USPOREDBA ALGORITAMA NEURONSKIH MREŽA

ZAVRŠNI RAD

Varaždin, 2021.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Erik Đuranec

Matični broj: 0016137387

Studij: Poslovni sustavi

USPOREDBA ALGORITAMA NEURONSKIH MREŽA

ZAVRŠNI RAD

Mentor/Mentorica:

Doc. dr. sc. Dijana Oreški

Varaždin, lipanj 2021.

Erik Đuranec

Izjava o izvornosti

Izjavljujem da je moj završni/diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Ovaj rad se bavi tematikom neuronskih mreža od teoretskog aspekta pa sve do uspoređivanja konkretnih algoritama učenja. Preciznije, ovaj rad se dotiče fundamentalnih osnova i principa funkcioniranja neuronske mreže od umjetnog neurona preko aktivacijskih funkcija i mrežnih topologija pa sve do algoritama učenja. Naravno, postoje mnogi algoritmi koji su zaduženi za funkcioniranje neuronske mreže, ali mi ćemo u ovome radu fokus staviti na četiri algoritma učenja. Konkretno, usporedit ćemo performanse algoritma gradijentnog spusta, konjugiranog gradijentnog spusta, Quasi-Newton metode te Levenberg-Marquardt algoritma kao algoritama treniranja ili učenja neuronske mreže. Pritom je arhitektura neuronske mreže koju koristimo fiksna kako bi ostale varijable, izuzev samih algoritama učenja, ostale kontrolirane. Naposljetku se izvodi zaključak o performansama pojedinog algoritma uz preporuke za odabir istog.

Ključne riječi: neuronske mreže, algoritmi, strojno učenje, klasifikacija, regresija, arhitektura neuronske mreže, gradijentni spust, konjugirani gradijentni spust, Quasi-Newton metoda, Levenberg-Marquardt algoritam

Sadržaj

1. Uvod

Tema ovog završnog rada je uspoređivanje algoritama neuronskih mreža, odnosno ovaj rad će se baviti teoretskom podlogom neuronskih mreža i njihovih algoritama uz pripadno uspoređivanje realizirano putem alata za izgradnju neuronskih mreža. Iako neuronske mreže kao ideja imaju korijenje u prošlome stoljeću, tek porastom računalne snage i razvojem boljih računalnih arhitektura možemo reći da su neuronske mreže doživjele ponovni procvat. Danas neuronske mreže omogućavaju realizaciju kompleksnih sustava uz pomoć različitih arhitektura, ali i algoritama koji predstavljaju srž njihove funkcionalnosti. Kompleksni sustavi koji su spomenuti omogućavaju postojanje pametnih asistenata, autonomne vožnje, ali i mnogih drugih „pametnih“ značajki koje su ostvarene putem okvira neuronskih mreža. Nadasve, izgleda da je implementacija neuronskih mreža otvorila put industriji, ali i akademskoj zajednici u daljnjem istraživanju i unapređenju klasičnih arhitektura i algoritama. Shodno rečenom, možemo samo nagađati kakva budućnost iščekuje ovakav pristup rješavanju složenih problema, ali jedno je sigurno, neuronske mreže danas čine vrlo zanimljivo područje. Tvrtke koje se upravljaju na temelju podataka, ali i računalni programi samo su neki od trendova koji potiču daljnji razvoj ovakvih sustava, a zasigurno ne i konačna granica primjene.

Iako neuronske mreže danas predstavljaju sustave specijalizirane primjene, odnosno sustave koji su jako dobri samo u određenoj domeni ili skupu domena, njihov potencijal je iznimno velik. Budući da neuronske mreže koriste iste fundamentalne principe kod rješavanja problema kao i ljudski mozak. Potonje me nagnalo na istraživanje ovog područja i njegove povijesti, primjene, budućnosti i trenutnih dostignuća jer osobno smatram da će neuronske mreže daljnjim razvojem tehnologije, teorije i novih rješenja pružiti zapanjujuće rezultate u budućnosti. Nadalje, pitanje ljudske svijesti i njezine manifestacije je osobna fascinacija koja je također vezana uz područje neuronskih mreža. Iako opća umjetna inteligencija ili samosvjesno biće izrađeno od strane čovjeka još uvijek spada u domenu znanstvene fantastike, smatram da je nužno daljnje istraživanje i inovacije u ovome području kako bismo mogli iskoristiti veliki potencijal ovakvih sustava. Naravno, s ciljem poboljšanja kvalitete, sigurnosti i velikog broja ostalih područja ljudskog života gdje neuronske mreže mogu biti primijenjene.

Pri izradi ovog rada polazišna točka je teoretska podloga, odnosno proučavanje i izučavanje literature vezane uz neuronske mreže, strojno učenje, algoritme, priručnike implementacije modela i sl. Nakon detaljne analize i objašnjenja ključnih koncepata neuronskih mreža usporedit ćemo algoritme koji se koriste u „treninzima“ istih. Prvo ćemo detaljno opisati odabrane algoritme za rješavanje konkretnog problema, te ih teoretski obraditi

U drugom dijelu rada koristit ćemo vrlo moćan alat Neural Designer koji će nam omogućiti dizajniranje arhitekture neuronske mreže pogodne za primjenu teoretski obrađenih algoritama. Pritom ćemo koristiti javno dostupne skupove podataka koji će poslužiti u treningu i evaluaciji pojedinog algoritma. Neural Designer ima iznimno bogatu paletu popratnih izvještaja vezanih uz sam trening konkretne mreže s vrijednim podacima koji će nam olakšati evaluaciju pojedinog algoritma, ali i komparaciju svih algoritama koje ćemo implementirati u neuronsku mrežu.

Naposljetku, dobivene rezultate komparacije algoritama putem alata Neural Designer usporedit ćemo s polazišnim teoretskim osnovama izabranih algoritama. Na temelju rezultata izvest ćemo zaključak o korištenim algoritmima .

2. Teoretska podloga neuronskih mreža

Ovo poglavlje je posvećeno osnovnoj gradivnoj jedinici neuronske mreže, umjetnom neuronu. Nadalje, uz pomoć jednostavnih koncepata postepeno ćemo graditi percepciju o umjetnoj neuronskoj mreži, analizirat ćemo različite aspekte kojima možemo definirati neuronske mreže te ćemo spomenuti njihove karakteristike i raspraviti o mogućim klasifikacijama neuronskih mreža. U završetku poglavlja spomenut ćemo algoritme koje možemo koristiti u treniranju i probleme koje možemo riješiti korištenjem ovakvih sustava.

2.1. Neuron – osnovna jedinica neuronske mreže

U ovome dijelu krenut ćemo od osnovne gradivne jedinice neuronske mreže, neurona. Važno je razumjeti koncept neurona budući da je on preduvjet izgradnje općenitih modela neuronskih mreža. Naravno, prije detaljne analize umjetnog neurona standardni je pristup poći od biološkog modela koji je poslužio kao inspiracija matematičkoj reprezentaciji biološke jedinice.

2.1.1. Biološki neuron

Krenut ćemo od objašnjenja biološkog neurona budući da je analoški vrlo sličan umjetnom neuronu kojeg koristimo kako bi izgradili umjetnu neuronsku mrežu. Valja naglasiti da ćemo u ovom poglavlju pod terminom „neuron“ smatrati biološku jedinicu, dok će se u svim ostalim poglavljima navedeni termin odnositi na umjetnu jedinicu (matematički model).

Biološki neuron ima mnogo jedinica od kojih je građen, međutim kako bi shvatili koncept umjetnog neurona nama su od posebnog interesa tri jedinice: dendriti, soma te akson (iako postoji još tjelešaca od kojih se neuron sastoji). Naš mozak, ali i cijeli živčani sustav (koji se sastoji od neurona) funkcionira na principu međusobne interakcije neurona. Naime, informacije se našim tijelom prijenose putem bilijuna neurona, odnosno živčanih stanica. Pojedini neuroni su povezani u kompleksnu biološku mrežu, a aktiviraju se podražajem uzrokovanim električnim impulsima. Spomenuti impulsi su iznimno brzi, odnosno impulsi putuju kroz neurone brzinom od 400 km/h što omogućava mozgu dotok iznimne količine informacija iz perifernih neurona (onih koji ne sačinjavaju mozak). Mozak sadržava oko 100 bilijuna neurona koji su međusobno povezani, a svaka skupina veza između neurona može biti zadužena za različite kognitivne funkcije. Budući da je fokus ovog rada na neuronu nećemo detaljnije ulaziti u moždane regije i trenutna dostignuća neuroznanosti. Nastavno, **dendriti** su poput malih ticala neurona, odnosno primatelji električnih impulsa neurona koji okidaju u neposrednoj blizini. Ovi električni impulsi do dendrita dolaze putem sinapsi koje sadržavaju neurotransmitere, a upravo

neurotransmiteri omogućavaju prijenos kemijskim putem (kemijskim procesom). Neurotransmiteri su zaduženi za okidanje neurona, odnosno oni su svojevrsni modulatori dolazećih električnih impulsa. **Soma** ili jednostavnije tijelo neurona je zaduženo za sumiranje svih dolaznih signala (putem dendrita) te ukoliko je suma dolaznih signala veća od određene točke okidanja spomenuti neuron šalje električni impuls prema sljedećem neuronu. Naravno, ukoliko suma dolaznih signala ne prelazi točku okidanja neuron ne prosljeđuje električni impuls sljedećem. Naposljetku, akson je produžetak some koji prenosi električni impuls iz some ostalim neuronima koji su spojeni na akson sa svojim dendritima. [1][2]

Ono što je važno izlučiti iz bioloških neurona kao svojstva važna kod umjetnih neuronskih mreža jesu: [1]

1. Procesni modul (soma) prima višestruke ulazne signale (električne impulse).
2. Ulazni signali mogu biti modulirani putem sinapse (težina ulaznog signala).
3. Procesni modul sumira sve ulazne signale i njihove pripadne težine.
4. Ukoliko je uvjet ukidanja zadovoljen neuron prenosi samo jedan impuls (jedan izlaz).
5. Iako ima jedan izlaz, putem aksona neuron može jedan izlaz prenijeti višestrukim neuronima.

Također, važno svojstvo koje moramo spomenuti je i tolerancija na greške. Neuroni mogu raspoznati greške na dva načina. Prvi način je prepoznavanje višestrukih ulaza koji su različitiji od prethodno primljenih ulaza, na principu ovoga načina funkcionira i temeljno raspoznavanje različitih objekta putem vidnog živca primjerice. Drugi način, puno svježiji u neuroznanosti je pojam neuroplastičnosti, odnosno sposobnosti neurona da promijene svoje veze kako bi mogle izvršavati nove funkcionalnosti. Temeljem navedenoga, mozgu je omogućeno da bez obzira na gubitak neurona (uzrokovan starenjem) može učiti nove stvari i procesuirati podražaje koji su degeneracijom neurona ugroženi. [1] [3]

2.1.2. Umjetni neuron

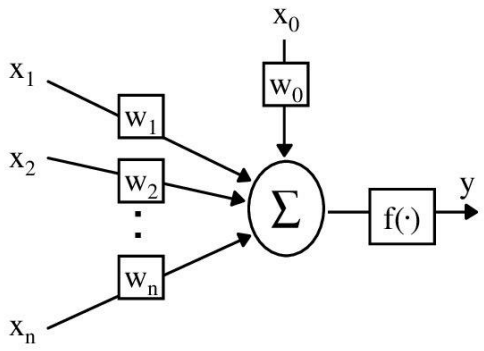
Nakon kratkog objašnjenja vitalnih dijelova bioloških neurona koji nam služe kao pretpostavka za gradnju umjetnih neurona možemo krenuti dalje u definiciju umjetnog neurona kojim gradimo neuronske mreže. Umjetni neuron je u svojoj srži način reprezentacije biološkog neurona, odnosno profinjeni način simuliranja aktivnosti stvarnog neurona korištenjem matematičkog modela.

Na isti način na koji smo opisali i biološki neuron možemo pristupiti izgradnji matematičkog modela neurona. Za početak, naš umjetni neuron će se također sastojati od

dendrita, u ovome kontekstu možemo ih nazvati *x* ulaznim varijablama. Nadalje, prisjetimo se da smo spomenuli neurotransmitere koji moduliraju ulazne signale (težine ulaza), u umjetnom neuronu težine ulaza ćemo označiti slovom *w*. Procesna jedinica (soma u biološkom neuronu) je u našem općenitom modelu umjetnog neurona reprezentirana aktivacijskom funkcijom. Postoje razne vrste aktivacijskih funkcija, međutim ovdje ćemo govoriti o generalnom modelu umjetnog neurona pa ćemo jednostavno reći da je aktivacijska funkcija jednostavno reprezentacija općenite funkcije, a kasnije ćemo spomenuti neke od često korištenih funkcija. Naposljetku, naš model neurona ima i izlaz, jedan jedinstveni izlaz koji ćemo označiti slovom *y* i tek sada možemo krenuti u pisanje matematičke formule našeg neurona. Prije nego krenemo u definiciju formule, valja se prisjetiti da je aktivacija određenog neurona uvjetovana vrijednošću sume ulaza *x* i njihovih pripadnih težina *w* te da funkcija koju koristimo (u ovom primjeru općenita funkcija) *f* određuje vrijednost izlaza neurona. Slijedom navedenog proizlazi da će suma umnožaka ulaza i njihovih pripadnih težina biti korištena od strane funkcije aktivacije $f(x)$, a rezultat izlaza, odnosno vrijednost izlaznog aksona će biti jednaka $y(x)$. Dakle, matematički možemo modelirati neuron na sljedeći način: [4]

$$y(x) = f\left(\sum_{i=1}^{n} w_i x_i\right)$$

Sada imamo osnovni matematički model koji koristimo kao gradivni element za konstrukciju kompleksnijih modela.



Slika 1: Prikaz modela umjetnog neurona(Prema: Szablowski, B. , 2020)

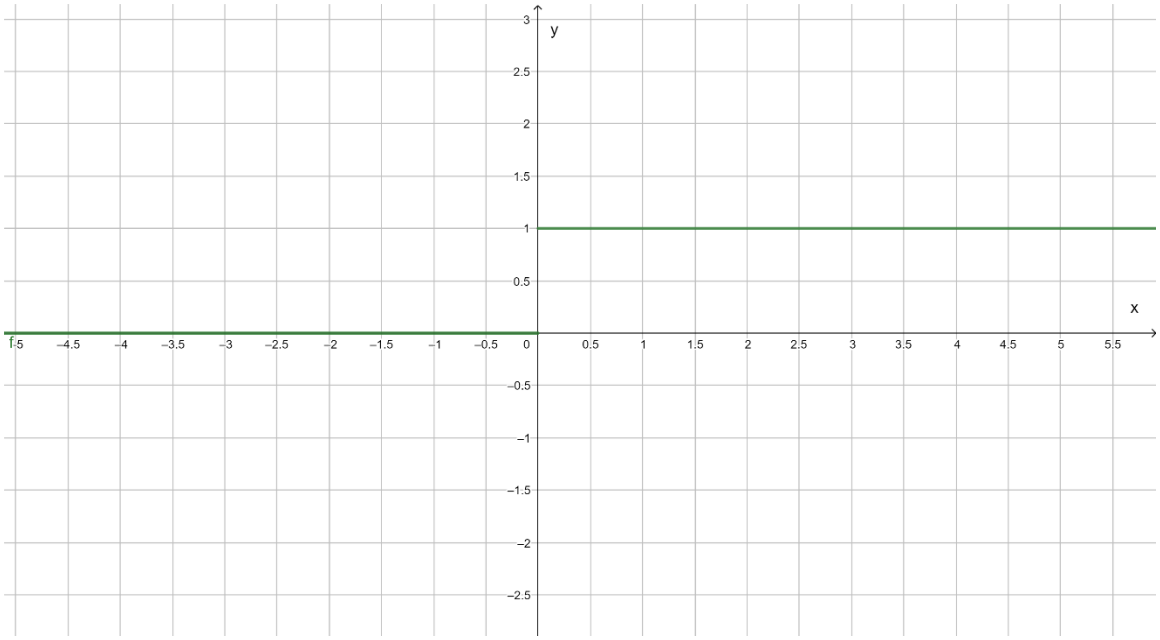
Na gornjoj slici dana je grafička reprezentacija matematičkog modela s ciljem bolje predodžbe strukture neurona, a posljedično i boljim razumijevanjem dane formule. [4]

Model sličan ovomu kojeg smo prikazali formulom, izuzev pripadnih težina, je prikaz samo jednog neurona i u literaturi se često ovakva reprezentacija naziva McCulloch-Pitts neuron, a karakteriziraju ga višestruki ulazi te samo jedan izlaz, što i odgovara reprezentaciji aksona u biološkom neuronu. Mi ćemo se ovdje ipak baviti drugim vrstama neurona koje su često korištene u sustavima današnjice. [5]

U nastavku ćemo detaljnije proučiti neke od mogućih funkcija aktivacije koje se mogu primjenjivati u umjetnim neuronima, a koriste se u realnim sustavima neuronskih mreža. Izbor pojedine funkcije aktivacije neurona ovisi o tipu problema koji se želi riješiti primjenom konkretne neuronske mreže, ali i o izboru te iskustvu arhitekta ovakvih sustava s obzirom na poznavanje problemske domene. Također, odabir aktivacijske funkcije jedan od ključnih čimbenika kojima razlikujemo neuronske mreže, ali o klasifikacijama neuronskih mreža biti će riječi u sljedećem poglavlju.

2.1.2.1. Stepeničasta binarna aktivacijska funkcija

Krenut ćemo s opisom jedne od tipičnih aktivacijskih funkcija, a to je binarna stepeničasta funkcija. Svoje ime ova funkcija duguje svome grafu koji podsjeća na stepenicu, a ova funkcija je ujedno i vrlo točna reprezentacija aktivacije some u biološkom neuronu.



Slika 2: Graf stepeničaste binarne funkcije (Izrađeno alatom Geogebra)

Stepeničasta binarna aktivacijska funkcija je vrlo jednostavna, na gornjoj slici je prikaz spomenute funkcije. Ukoliko je vrijednost koja ulazi u funkciju (u kontekstu rada, suma umnožaka ulaza i pripadnih težina) manja od 0, onda će rezultat funkcije biti 0 (bez aktivacije).

S druge strane, ukoliko je vrijednost ulaza veća od ili jednaka 0, onda će rezultat funkcije biti 1 (aktivacija neurona). Spomenuta funkcija može biti prikazana i sljedećom formulom:

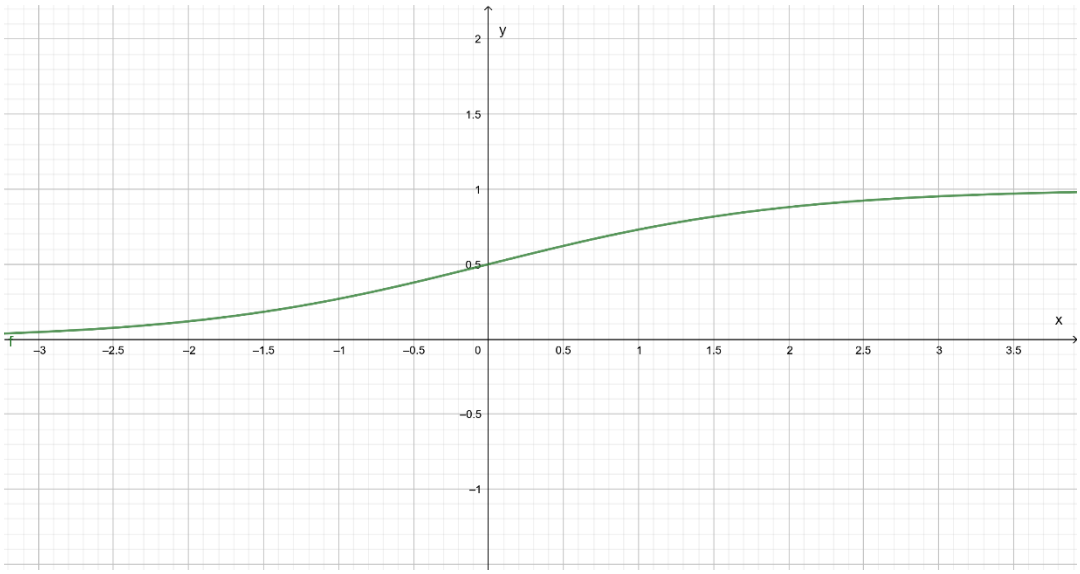
$$f(x) = \begin{cases} 1 \; ako \; x \geq 0 \\ 0 \; ako \; x \; < 0 \end{cases}$$. [1]

Važno je naglasiti i očiti nedostatak ove funkcije, a to je da je njezin izlaz može dati samo dvije moguće vrijednosti 0 ili 1. Ovo ograničenje nameće nemogućnost aktivacije više klasa, što utječe na loše rezultate klasifikacije korištenjem ove vrste aktivacijske funkcije. [5]

2.1.2.2. Sigmoidna (logistička) aktivacijska funkcija

Vjerojatno jedna od najčešće korištenih aktivacijskih funkcija u neuronima je sigmoidna ili logistička funkcija kao alternativa prethodno spomenutoj aktivacijskoj funkciji. Ključna razlika između sigmoide i stepeničaste binarne funkcije je da kod sigmoide izlazne vrijednosti mogu biti u rasponu od 0 do 1.

Logistička funkcija je također karakteristična po svome grafu u obliku slova S, a funkcija je dana formulom: $f(x) = \frac{1}{1+e^{-x}}$ .Dakle, jedan od najvažnijih razloga zašto se ova funkcija koristi je spomenuti izlaz u rasponu od 0 do 1. Naime, sigmoidna logistička funkcija nema derivacija koje su, odnosno ne postoji derivacija iznosa 0 za bilo koju točku funkcije. Upravo zbog toga određene metode, poput jednostavnog kontrasta (eng. *Simple Gradient Method*), mogu postići bolje rezultate za svaki korak optimizacije. O koracima optimizacije i konkretnim algoritmima bavit ćemo se u kasnijim poglavljima. Za sada je važno spomenuti da su metode koje koristimo za optimizaciju ustvari metode koje omogućavaju „treniranje“ neuronske mreže. Problem ove aktivacijske funkcije naziva se problemom nestajućeg kontrasta koji nastaje zbog same prirode transformacije ulaza u izlaze u rasponu od 0 do 1. Konkretno, ukoliko ulazni podaci teže u beskonačnost (bilo pozitivnu ili negativnu) derivacija funkcije je vrlo blizu 0, to može rezultirati toliko malim kontrastom da proces učenja postaje neefikasan. [1] [5]



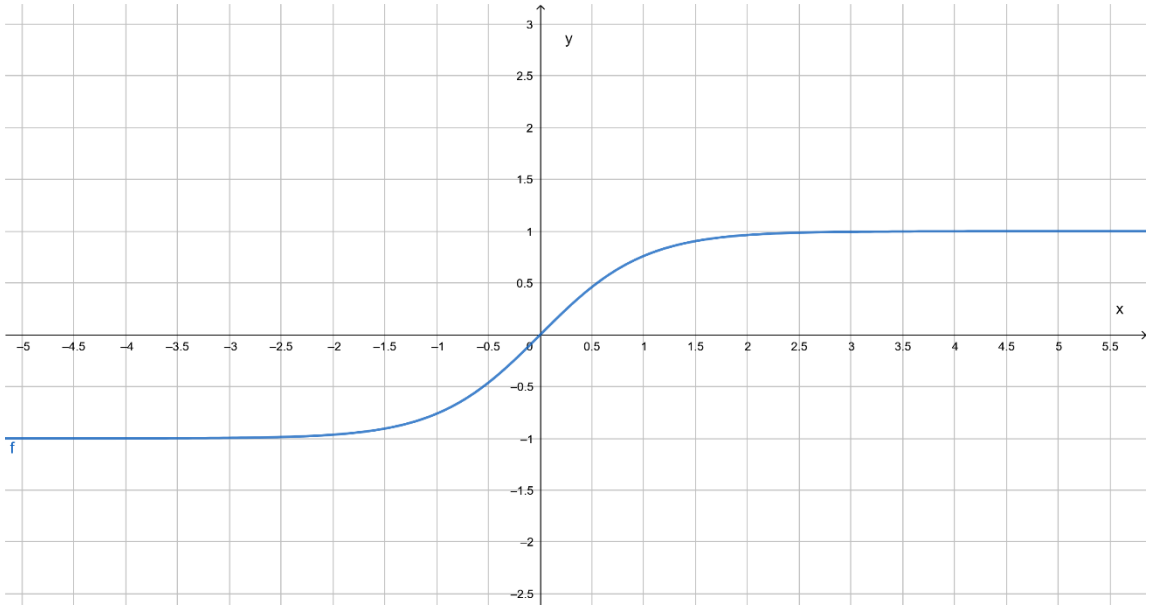
Slika 3: Graf logističke funkcije (Izrađeno alatom Geogebra)

Kao što je i vidljivo na gornjoj slici grafa logističke funkcije, što više ulazi u funkciju teže u beskonačnost to je derivacija (smjer kretanja funkcije) bliže 0. Ponovno, diferencijabilnost aktivacijske funkcije je vrlo važna, a to ćemo posebno i detaljnije objasniti u poglavlju koje će se baviti algoritmima neuronskih mreža.

2.1.2.3. Hiperbolni tangens ili TanH aktivacijska funkcija

Funkcija hiperbolnog tangensa je vrlo slična logističkoj funkciji, međutim razlika je u tome da TanH funkcija vraća rezultate u rasponu od -1 do 1 što će biti jasnije nakon objašnjenja njene formule i prikaza grafa kao i za prethodne aktivacijske funkcije.

Za početak, TanH funkcija je kao i logistička funkcija neprekidna i diferencijabilna funkcija, što znači da je dobar kandidat za mnoge algoritme optimizacije. Graf funkcije TanH ima središte u ishodištu koordinatnog sustava. Drugim riječima TanH možemo zamisliti kao pomaknutu verziju logističke funkcije s proširenom kodomenom. Preopruka je da se TanH koristi kod binarnih podataka umjesto uobičajene logističke sigmoide. Ovdje je važno spomenuti i postojanje bipolarne sigmoide koja, za razliku od logističke sigmoide, može poprimiti vrijednosti izlaza u rasponu od -1 do 1. Formula za TanH aktivacijsku funkciju je: $f(x) = \frac{sinhx}{coshx} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$. [1][5]



Slika 4: Graf TanH funkcije (Izrađeno alatom Geogebra)

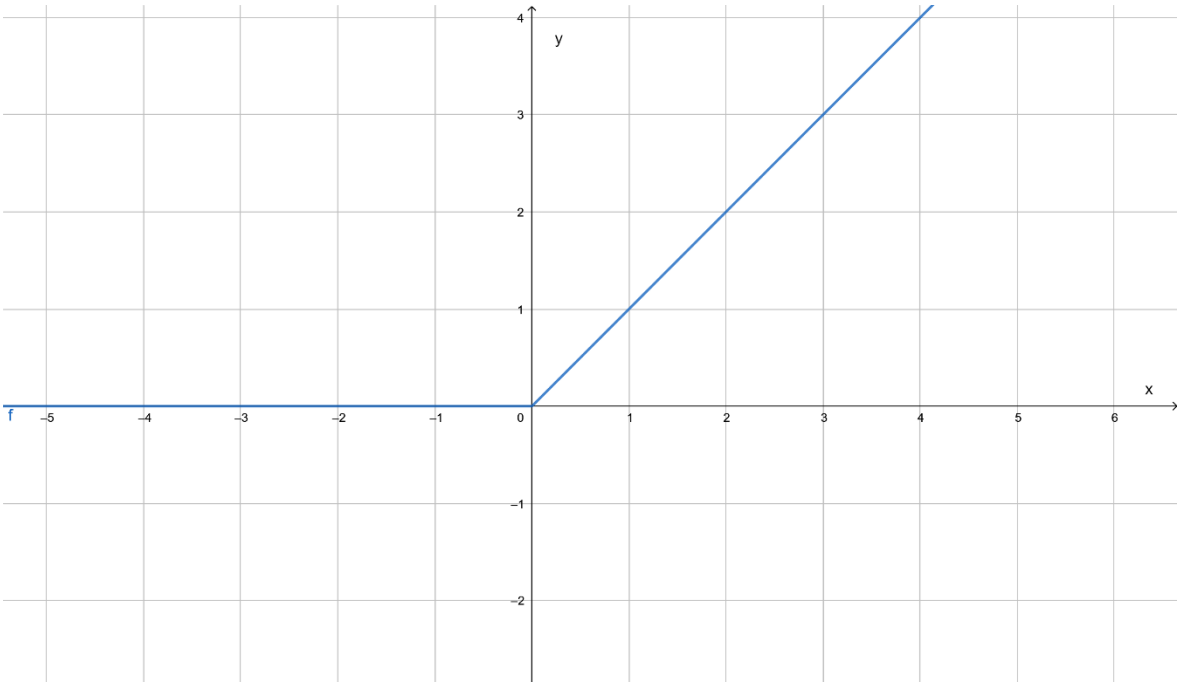
2.1.2.4. ReLU aktivacijska funkcija

ReLU aktivacijska funkcija neurona je danas često korištena funkcija u sustavima koji su namijenjeni za duboko učenje, međutim ovdje je spominjemo iz razloga jer je njezina učestalost korištenja u porastu.

ReLU funkcija je kombinacija uobičajene funkcije praga i linearne funkcije, a njezina formula je:

$$y(x) = \begin{cases} 0 \; ako \;\; x < 0 \\ x \; ako \; x \;\geq 0 \end{cases}.$$ [5]

Možemo primijetiti da je matematički oblik ove funkcije iznimno jednostavan, ali to ne znači da nije i efektivan. Naime, kod ReLU aktivacijske funkcije ne dolazi do problema nestajućeg kontrasta uzrokovanog nedostatkom maksimalne izlazne vrijednosti. Naravno, ReLu ima i svojih nedostataka poput „ReLU umiranja“ do kojeg dolazi zbog izlazne vrijednosti 0 za sve ulaze koji su negativni. Međutim, postoje razne varijante ReLu aktivacijske funkcije koje zaobilaze spomenuti nedostatak, ali u okviru ovoga rada u njih nećemo ulaziti. U nastavku slijedi prikaz grafa ReLu aktivacijske funkcije gdje se jasno može vidjeti dualna priroda ove funkcije. [5]



Slika 5: Graf ReLu funkcije (Izrađeno alatom Geogebra)

Nakon pregleda nekih od aktivacijskih funkcija važno je spomenuti i dodatne stvari koje se tiču odabira same funkcije kod izgradnje modela neuronske mreže. U nastavku ćemo se osvrnuti na bitne zaključke ovog dijela te ćemo spomenuti osnovne preporuke kod odabira aktivacijske funkcije.

Općenito, aktivacijska funkcija bi trebala biti nelinearna funkcija, upravo zbog toga jer korištenjem linearne aktivacijske funkcije mreža ustvari postaje model linearne regresije. Stoga, budući da neuronske mreže nisu stvorene da budu ekvivalent postojećim modelima preporuka je izbjegavati korištenje ovakvih funkcija. Nadalje, većina problema iz realnog svijeta u kojima koristimo neuronske mreže kao alat imaju nelinearno preslikavanje ulaza u izlaze. Važno je reći da ćemo odabirom određene aktivacijske funkcije istu koristiti kroz sve neurone u neuronskoj mreži, što je i uobičajeno. Autori se slažu da su TanH, sigmoidna logistička funkcija te stepeničasta binarna funkcija česti izbor kod dizajniranja arhitekture neuronske mreže. Također, funkcije koje ograničavaju velike pozitivne ili negativne vrijednosti u određeni raspon, poput logističke, nazivaju se funkcije gnječenja iz očitog razloga. Dakako, korištenje pojedinačnih neurona nije efikasno, budući da bi se njihovo korištenje svelo na linearni model ili logističku regresiju (ovisno o odabranoj aktivacijskoj funkciji). Zato neurone spajamo u mrežu, konkretnije u neuronsku mrežu o kojoj ćemo raspravljati u sljedećem potpoglavlju. [5] [6] [7]

2.2. Neuronska mreža

Ovo potpoglavlje će se baviti osnovnim konceptima koji su važni za razumijevanje modela neuronske mreže. Budući da sada imamo osnovno razumijevanje temeljne gradivne jedinice – neurona, možemo graditi kompleksnije koncepte. Također, u ovome dijelu ćemo se posvetiti i ključnim karakteristikama koje čine određuju neuronsku mrežu, a u sljedećem poglavlju ulazimo u detalje vezane uz algoritme, arhitekture te vrste primjena pojedinih modela neuronskih mreža.

2.2.1. Definicije neuronske mreže

Neuronsku mrežu možemo definirati iz više aspekata, u nastavku ćemo spomenuti nekoliko različitih definicija neuronske mreže iz relevantnih izvora i donijeti zaključak o aspektu s kojeg ćemo promatrati neuronsku mrežu u ovome radu.

Neuronska mreža predstavlja generalizaciju ljudske kognicije i neurobiologije pomoću matematičkih modela. Pretpostavke koje su pritom zadovoljene jesu: [1]

1. Obrada informacija se odvija u puno jednostavnih jedinica obrade – neuronima
2. Signali između neurona se prenose putem komunikacijskih veza
3. Svaka komunikacijska veza ima pripadnu težinu veze
4. Svaki neuron ima aktivacijsku funkciju za određivanje izlaza temeljem sume ulaza

Dakle, ovdje proučavamo neuronsku mrežu kao sustav za obradu podataka koji ima određene karakteristike performansi zajedničke s biološkom neuronskom mrežom. [1]

Sljedeći pristup govori o neuronskim mrežama kao alatu za rješavanje problema uz pomoć simuliranja funkcionalnosti moždanih aktivnosti. Iako se ovdje naglasak stavlja na simuliranje moždanih aktivnosti putem adekvatnih heurističkih modela ponašanja, a ne na replikaciji biološkog sustava putem neuronske mreže. Shodno rečenome, možemo zaključiti da je ovakav pristup umjetnoj neuronskoj mreži nije ograničen biološkim karakteristikama, već za cilj ima adekvatnost modela. Drugim riječima, vjernost replikacije moždane aktivnosti ovdje nije bitna. [4]

Neuronsku mrežu možemo promatrati i kroz aspekt sustava, odnosno kao sustav jednostavnih jedinica koje povezane na prikladan način mogu dati zanimljive rezultate i manifestirati kompleksna ponašanja. Drugim riječima, promatramo li neuronsku mrežu kao sustav možemo zaključiti da njegova snaga ne proizlazi iz pojedinačnih jednostavnih procesorskih jedinica (neurona). Snaga sustava krije se u kombinaciji i povezivanju pojedinačnih jedinica u sustav sposoban za rješavanje kompleksnih problema. Pritom se

specijalizacija neuronske mreže realizira korištenjem različitih mrežnih topologija o kojima će biti riječi u sljedećem poglavlju te korištenjem različitih aktivacijskih funkcija koje smo već ranije spomenuli. [6]

Naposljetku, neuronsku mrežu možemo promatrati iz aspekta informatičko kibernetičke teorije. Ključni aspekti funkcija u neuronskoj mreži koji su vezani uz kibernetičku perspektivu jesu kompleksnost neuralne arhitekture, prostorno-vremenska (eng. *Spatiotemporal randomness*) slučajnost, efikasnost samoorganizacije povezanih neurona te entropija sustava povezana sa prostorno-vremenskom slučajnošću. [7]

Budući da je cilj ovog rada uspoređivanje performansi različitih algoritama neuronskih mreža bilo bi dobro krenuti od promatranja neuronske mreže kao sustava koji omogućava rješavanje konkretne problemske domene. Ne zanima nas vjernost replikacije biološkog sustava, već smo fokusirani na aspekt korektnosti modela u rješavanju zadanog problema. Odnosno, zanima nas kako pojedini algoritam optimizacije ili treniranja neuronske mreže utječe na njezine performanse.

2.2.2. Karakteristike neuronske mreže

Iako se na prvu možda čini da neuronske mreže predstavljaju jedinstveni model koji koristi različite varijacije algoritama za rješavanje problema, to je vrlo daleko od istine. Danas postoji ogroman broj različitih varijacija neuronskih mreža, a sve zahvaljujući kombinacijama karakteristika koje ćemo izdvojiti u ovom potpoglavlju.

Kao što smo i ranije spomenuli, postoji ogromni broj varijacija neuronskih mreža, ali svaku pojedinu mrežu možemo okarakterizirati ili opisati specifičnim skupom karakteristika koje istu određuju. Neuronsku mrežu karakterizira **aktivacijska funkcija**, a ranije smo spomenuli neke od mogućih funkcija i njihovu ulogu u modelu neuronske mreže. Nadalje, važno je spomenuti **mrežnu topologiju** ili **mrežnu arhitekturu** koja predstavlja način na koji povezujemo neurone unutar mreže te njihov broj, ali nam govori i o broju slojeva mreže o kojima će biti više riječi kasnije. Naposljetku i s obzirom na temu rada nama jedna od najzanimljivijih karakteristika, **algoritam treniranja** koji specificira na koji način se određuju težine pojedinih ulaza kako bi se neuroni korektno inhibirali ili aktivirali proporcionalno s ulaznim signalima. [4]

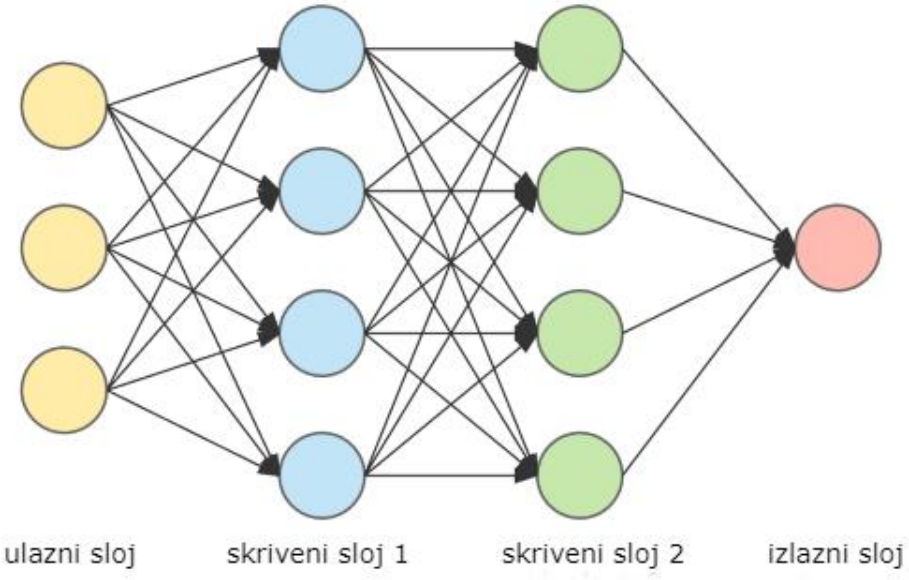
Budući da smo aktivacijske funkcije obradili u prethodnom poglavlju u ovome poglavlju ćemo se detaljnije posvetiti mrežnim topologijama, odnosno arhitekturama neuronskih mreža.

2.2.2.1. Mrežne topologije

Topologija mreže uvelike određuje njenu sposobnost za učenjem budući da neuroni kao zasebne jedinice nisu dovoljni za rješavanje kompleksnih problema. Slično kao što smo definirali specifične karakteristike koje određuju neuronsku mrežu, tako ćemo i definirati skup karakteristika koje određuju mrežnu topologiju.

Kapacitet učenja neuronske mreže je uvelike određen njenom topologijom, odnosno u obrascima i strukturama međusobno povezanih neurona. Karakteristike koje određuju pojedinu topologiju jesu broj slojeva mreže, smjer putovanja informacija unutar mreže (putuju li informacije samo unaprijed ili je dozvoljeno i vraćanje) te broj neurona unutar pojedinog sloja mreže. [4]

Mrežnu topologiju možemo definirati i kao uređeni skup neurona razvrstanih u slojeve s pripadnim obrascem povezivanja pojedinih neurona. Nadalje, slojevi mreže određuju dva osnovna tipa neuronske mreže temeljem slojeva, a to su neuronske mreže s jednim slojem ili višeslojne neuronske mreže. Općenito, slojeve neuronske mreže možemo podijeliti na tri različita tipa: ulazni sloj, skriveni sloj te izlazni sloj. Ulazni sloj je sloj koji prima inicijalne podatke koje dajemo neuronskoj mreži. Skriveni sloj jednostavno predstavlja sloj (ili slojeve jer kod određenih arhitektura ih može biti više) između ulaznog i izlaznog sloja gdje se odvija procesuiranje podataka. Izlazni sloj je sloj koji daje rezultate za dani set ulaznih podataka. [1] [9]



Slika 6: Slojevi neuronske mreže (Prema: G. Ognjanovski, 2019)

Nadalje, spomenuli smo i način na koji informacije putuju neuronskom mrežom kao važnu odrednicu topologije. Naime, smjer strujanja informacija kroz mrežu može bitno utjecati na performanse učenja. U prijašnjoj ilustraciji podaci kroz mrežu putuju isključivo od ulaza prema izlazu, odnosno mreža širenja prema naprijed (eng. *feedforward network*). S druge strane, postoje i mreže kod kojih je moguće širenje unatrag, odnosno rekurzivne mreže (eng. *recurrent network*). Potonje mreže puno vjernije preslikavaju biološku neuronsku mrežu, ali i omogućavaju rješavanje kompleksnijih problema. Konkretno, rekurzivne mreže su primjerice vrlo dobre kod prepoznavanja ovisnosti između pojedinih riječi unutar teksta i predviđanja sljedeće riječi. [4] [10]

Što se broja neurona unutar mreže tiče, nema generalnog pravila o određenom broju neurona. Dakle, broj neurona unutar mreže je proizvoljan izbor arhitekta ili dizajnera mreže a ovisi o nizu parametara poput problema za koji se mreža trenira, kompleksnost problemske domene, ograničenost računalnih resursa. Ono što valja imati na umu je da veći broj neurona može rezultirati zrcaljenjem trening skupa podataka, odnosno naša mreža može izgubiti točnost uvođenjem novih njoj nepoznatih podataka (ovo je vrlo popularan problem kod treniranja mreža nadziranim učenjem). Preporuka je koristiti što manji mogući broj neurona koji daje zadovoljavajuće rezultate kod validacijskog seta podataka. [4]

Algoritme neuronskih mreža (učenja) ćemo obraditi u sljedećem potpoglavlju, međutim i oni su bitna karakteristika koja je nama najzanimljivija u kontekstu ovog rada. Također, u nastavku ćemo se posvetiti relevantnim klasifikacijama neuronskih mreža temeljene na ranije spomenutim karakteristikama, spomenuti ćemo i načine učenja neuronskih mreža.

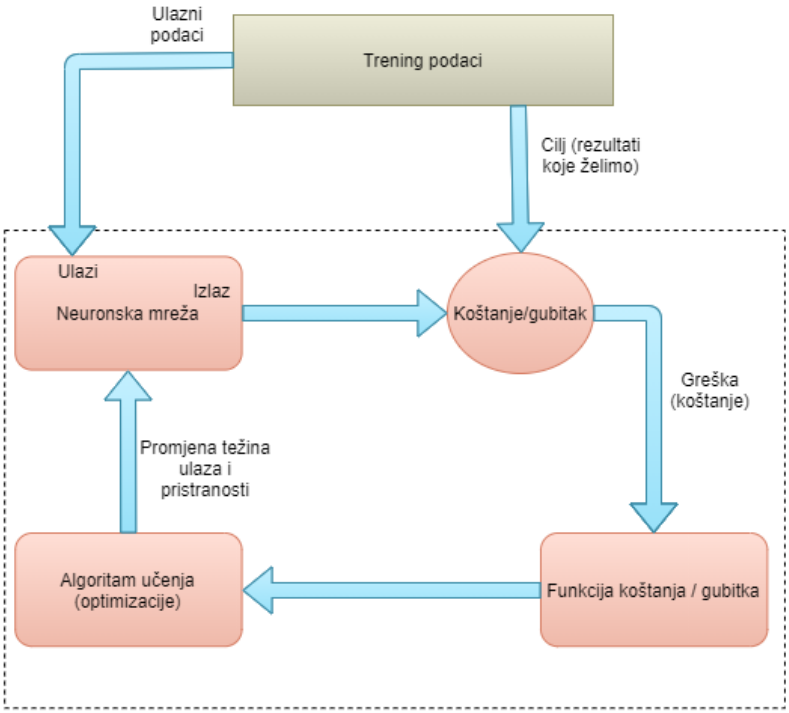
2.3. Algoritmi učenja

Algoritmi učenja predstavljaju sasvim jednu zasebnu cjelinu u svijetu neuronskih mreža a o istima su napisane stotine radova i izdano mnogo knjiga jer oni uvelike određuju karakteristike mreže. Način na koji mreža uči predstavlja inherentni problem upotrebe i popularizacije neuronskih mreža. Naime, pojavom nekih novih algoritama pojavilo se i veće zanimanje za rad s neuronskim mrežama upravo zbog određenih poboljšanja koje prijašnji algoritmi nisu imali. U ovome potpoglavlju ćemo se baviti osnovnim konceptom algoritma učenja, odnosno njegovom teoretskom podlogom i intuitivnim pristupom, dok će u praktičnome dijelu biti opisani pojedini algoritmi koje ćemo koristiti na konkretnom skupu podataka radi uspoređivanja. Međutim, prije nego krenemo na sam koncept algoritama učenja moramo se posvetiti načinima na koji neuronske mreže uče kako bi mogli istinski razumjeti koncepte učenja.

2.3.1. Načini učenja

Da, uistinu postoje različiti načini na koji neuronske mreže mogu učiti, iako se ova tema čini kompleksna ustvari je vrlo laka i jednostavna. Osnovna razlika u načinima učenja neuronskih mreža je u postojanju ili nepostojanju ljudskih naputaka o rezultatima. Iako se možda ne čini jasno na prvu u nastavku ćemo detaljno proći kroz različite načine učenja i objasniti pojedinosti za svaki način.

Nadzirano učenje (eng. *Supervised learning)* je učenje kod kojeg je izlaz poznat mreži, odnosno mreža zna koji skup vrijednosti može očekivati na izlazu. Preciznije, neuronskoj mreži je dan set podataka za treniranje (ulaznih vektora) sa željenim izlaznim vrijednostima (izlaznim vektorom). Ova vrsta učenja seže do početaka razvoja neuronskih mreža gdje su najjednostavnije neuronske mreže pretežito bile suočene sa zadacima klasifikacije gdje je svrha bila odrediti pripada li neki element skupu ili ne. Dakle, kod nadziranog učenja neuronskoj mreži jasno dajemo do znanja s kojim vrijednostima barata i koje vrijednosti očekujemo kao rezultat, odnosno izlaz iz neuronske mreže. Također, nadzirano učenje generalno predstavlja problem aproksimacije funkcije. Primjerice, ukoliko imamo set ulaznih podataka *x* te set ciljanih izlaza *t* naša zadaća je pronaći funkciju *y(x)* koja će zadovoljiti ciljane izlaze za sve dane ulaze. Ovo možda zvuči kao trivijalni zadatak, međutim moramo imati na umu da se neuronske mreže mogu sastojati od velikog broja neurona te slojeva, a shodno tome i ulaza i pripadnih težina. Ova činjenica čini problem pronalaska optimalne funkcije itekako računalno zahtjevnim zadatkom. Budući da ovakav problem nikako nije trivijalan vrijeme je da spomenemo i funkcije koštanja (eng. *cost functions*). Funkcije koštanja nam pomažu utvrditi koliko je precizna vrijednost dobivena upotrebom modela od stvarne očekivanje vrijednosti, ustvari je suma kvadrata razlike dobivene i tražene vrijednosti. Za sada je dovoljno samo reći da nam funkcije koštanja pomažu usmjeriti algoritam treniranja, odnosno govore nam kreće li se treniranje u dobrom smjeru ili ne. O funkcijama koštanja ili funkcijama grešaka govorit ćemo u zasebnom potpoglavlju, a u nastavku slijedi dijagram nadziranog učenja koji nam može pomoći u shvaćanju gore spomenute teorije, također ovo je način učenja kojim ćemo se mi baviti. [1] [6] [11] [12]



Slika 7: Dijagram nadziranog učenja (Prema: Reedu i Marksu, 1998)

Nenadzirano učenje (eng. *Unsupervised learning*) predstavlja antipod nadziranom učenju kao što se i iz naziva može pretpostaviti. U nenadziranom učenju dajemo mreži ulazni set podataka za učenje, ali ne i željeni izlaz, dakle na mreži je da klasificira ulazne podatke i podesi težine ulaza. Možemo reći da u nenadziranom učenju ne postoji učitelj, kao kod nadziranog učenja. Pritom, važno je naglasiti da niti ulazni trening podaci nisu označeni tako da mreža ustvari ne zna s kakvim podacima radi. Postavlja se pitanje kako neuronske mreže uče kod nenadziranog učenja? Radi se o svojstvu adaptabilnosti neuronskih mreža, a vezano uz sličnosti s biološkim neuronima. Dakle, mreža se kod nenadziranog učenja prilagođava pravilnostima unutar skupa podataka prema implicitnim pravilima implementiranima u sam dizajn arhitekture mreže. Možemo reći da je sad učitelj implementiran u sam dizajn neuronske mreže. Ovakav tip učenja je koristan jer često imamo lako dostupne skupove neoznačenih podataka nauštrb označenim setovima podataka (poput planetarno popularnog MNISTA). Za implementaciju nenadziranog učenja često se koriste samo organizirajuće neuronske mreže (tzv. Kohonen samo-organizirajuće mape). Također, ovaj tip učenja je iznimno kompleksan i zahtjevan za računalne resurse, ali unatoč tome pronalazi mnoge aplikacije u realnoj domeni. Nenadzirano učenje predstavlja i uzbudljivo područje za razvoj i eksperimentiranje neuronskih

mreža, ali mi u opsegu ovoga rada nećemo detaljnije ulaziti u područje nenadziranog učenja. [1] [6] [13]

Učenje s potporom (eng. *Reinforcement learninq*) predstavlja kombinaciju nadziranog i nenadziranog učenja. Kod ovog tipa učenja neuronska mreža nema poznate tražene vrijednosti izlaza, ali postoji potpora u smislu točnosti ili netočnosti izlaza. Također se naziva i polu-nadzirano učenje iz očitih razloga. Ovaj tip učenja predstavlja suštinski način učenja kod ljudi i životinja po principu sustava nagrađivanja/kažnjavanja. Da preciziramo, uzmimo analogiju sa dresurom psa, ukoliko je pas uspješno izvršio zadatak dobit će poslasticu kao nagradu za uspjeh, odnosno dobit će pozitivni podražaj. S druge strane ukoliko pas ne izvrši zadanu naredbu, onda će ostati bez poslastice, odnosno ulaz je ovdje negativni podražaj. Na ovaj način pas uči kako da izvrši određenu naredbu uz pomoć pozitivnih i negativnih podražaja. Slično uče i djeca uz pomoć pokušaja i pogrešaka kod upoznavanja svijeta oko sebe od malih nogu. Cilj učenja je minimizirati sumu negativnih podražaja neuronske mreže (pogrešaka) što je više moguće. Postoji učenje s potporom temeljeno na modelu kod kojeg sustav uz pomoć iskustva konstruira interni model tranzicija i pripadnih posljedica pojedine tranzicije za okolinu na koje se referira kako bi odabrao prikladnu akciju, dok su podražaji eksterni (van sustava). S druge strane imamo i učenje s potporom bez modela gdje sustav koristi iskustvo kako bi naučio pravilo ili vrijednost funkcije bez korištenja internog modela okoline. Ključna razlika je da u potonjem sustav zna samo moguća stanja i akcije u okolini, ali ne zna ništa o prijelazima stanja ili funkcijama podraživanja. [6] [11] [14]

Naveli smo tri osnovna tipa učenja za neuronske mreže, iako danas postoji i još varijacija na temu. Primjerice, možemo spomenuti i nadzirano učenje s udaljenim učiteljem, online učenje i offline učenje, ali za naše potrebe tri opisana tipa učenja će biti dovoljna za razumijevanje teme ovoga rada. Nadalje, moramo obavezno spomenuti i pojam pogreške u kontekstu neuronskih mreža jer su vrlo važne za same algoritme optimizacije (učenja). Dakle, pogreške u predikcijama naših modela čine ključan element u svakom novom koraku optimizacije performansi neuronske mreže. Upravo iz tog razloga u sljedećem potpoglavlju ćemo ukratko opisati što su konkretno greške u kontekstu neuronskih mreža, kako ih mjerimo te naposljetku zašto su one ključne za algoritme učenja.

2.3.2. Analiza pogrešaka

Analiza pogrešaka predstavlja jedan od koraka kod općenitog treniranja ili učenja neuronske mreže za rješavanje danog zadatka. Dakle, analizu pogrešaka možemo opisati kao proces analize skupova podataka za treniranje čije vrijednosti algoritam nije dobro aproksimirao (kod nadziranog učenja). Ili jednostavnije rečeno, analiza pogrešaka je metoda koju koristimo kako bi sastavili listu problema u mreži na koje trebamo obratiti pažnju i koji

drastično utječu na performanse mreže. Koji su to izvori pogrešaka koji mogu utjecati na performanse mreže? Postoji ih nekoliko, a najčešći je zasigurno **loša kvaliteta podataka**, odnosno nepravilno označeni podaci za treniranje. Sveprisutnost ovog problema je naveliko polemizirana i u drugim područjima znanosti a sročena je konceptom „Smeće unutra, smeće van.“ koji se reflektira na prethodni izvor pogrešaka. Drugi izvor pogrešaka je tzv. **zamućena linija demarkacije** ili jednostavno nesposobnost mreže da uvijek točno i jednoznačno odredi rezultat. Primjerice, ukoliko neuronska mreža mora razlikovati psa od čovjeka veća je vjerojatnost da će to učiniti dobro, nego ukoliko mora razlikovati sliku žene od slike muškarca. Razlozi za to su i sami po sebi jasni, čak i ljudi imaju problema kod jednoznačnog određivanja spola pojedinca temeljem vizualnog podražaja. Na posljednjem mjestu je već prethodno spomenuti koncept **pretjeranog korištenog ili zapostavljenog atributa (dimenzije)** skupu podataka za trening. Naime, ukoliko kod jednostavnih mreža za klasifikaciju imamo zadatak određivanja razlike između mačke i psa, a sustav za svijetlo-dlake mačke govori da su psi i za tamno-dlake pse kao mačke onda se očito radi o grešci. Ova pogreška može biti vezana uz nedovoljno primjera unutar skupa podataka za trening, a koji su vezani uz atribut boje dlake pojedine životinje ili jednostavno nedostatnim brojem trening primjera. [15] [16]

Jednom kada se približimo određenom skupu problema ili uočimo analizom da su pogreške uniformne pitanje je što učiniti dalje. Ukoliko samo nahranimo neuronsku mrežu s više trening podataka možemo prouzrokovati pretjeranu prilagođenost modela skupu podataka za trening i slične neželjene pojave. Glavni parametri koji nam govore u kojem bi smjeru trebali krenuti za razrješavanje pogrešaka su **pristranost** (eng. *bias)* te **varijanca.** Ako imamo pogreške u klasifikaciji ili predikciji u testnom i skupu podataka za treniranje, onda to znači da imamo visoku pristranost. Ukoliko imamo puno pogrešaka u testnom skupu podataka, a malo u trening skupu, onda imamo visoku varijancu. Generalno, možemo reći da ako mreža ima visoku pristranost da će izlaz iz mreže biti loš i za testne i za trening podatke, a ukoliko je varijanca visoka output je dobar za neki skup podataka, ali loš za drugi skup. Visoka pristranost može upućivati na nedovoljno bogat model, dok visoka varijanca može upućivati na nedovoljno trening primjera. Pristranost je iznimno važna za razumijevanje koncepta algoritma učenja. [15] [17]

Pristranost koristimo kako bi premjestili rezultat aktivacijske funkcije prema pozitivnoj ili negativnoj strani prostora rješenja. Korištenjem pristranosti efektivno smanjujemo varijancu, uvodimo fleksibilnost u model te postižemo bolju generalizaciju neuronske mreže. U svojoj srži pristranost nam omogućava bolju kontrolu nad aktivacijom same aktivacijske funkcije a radi se o nekom realnom broju. Sada kad smo spomenuli pristranost i njezinu funkciju vrlo lako je možemo integrirati u formulu iz poglavlja 2.1.2. koja sada izgleda ovako:

$y(x) = f(\sum_{i=1}^{n} w_i x_i) + pristranost.$ [18]

Ovime smo kompletirali osnovnu formulu koja nam je potrebna za razumijevanje algoritama učenja. [15] [18]

Postoje razni načini kako možemo utjecati na smanjenje broja pogrešaka u modelu. Za smanjenje pristranosti možemo: [15]

- povećati veličinu modela,
- dozvoliti modelu da koristi više atributa (pretjerana selekcija atributa),
- smanjenje regularizacije (regularizacija modela održava model balansiranim – atribute modela drži u susjedstvu 0),
- zaobilazak lokalnog minimuma (problem o kojem ćemo više pričati u nastavku)
- te bolja mrežna topologija.

Za smanjenje varijance možemo: [15]

- povećati broj trening podataka
- dodati regularizaciju (L1 i L2 metode koje efektivno onemogućavaju pretjeranu prilagođenost algoritma, tzv. overfitting)
- rano zaustavljanje učenja (veliki broj iteracija nad istim podacima može izazvati pretjeranu prilagođenost)
- smanjiti broj atributa (primjerice PCA metodom)
- smanjiti veličinu modela
- koristiti pomoćni model
- te promijeniti mrežnu topologiju.

Nećemo se detaljnije baviti pojmovima navedenima u prethodnom nizu mogućih rješenja za smanjenje pogrešaka budući da bi se o svakom pojedinom mogao napisati cjeloviti završni rad, a to nam za daljnje razumijevanje algoritama nije toliko važno. [15] [18]

I konačno možemo krenuti s općenitim opisom algoritma učenja. Ako zamislimo svaki neuron unutar mreže i svih njezinih slojeva kao mali kotačić čijim finim zakretanjem postižemo bolju aproksimaciju ciljane funkcije (kotačić s pripadnim ulazima, težinama te pristranošću) onda je algoritam učenja maestro koji umjesto nas okreće kotačiće. Ustvari, način na koji neuronske mreže uče sve više počinje zvučati kao matematičko modeliranje, a ne svijet znanstvene fantastike, što ono ustvari i jest. Sve se svodi na pronalazak minimuma određene funkcije. Prisjetimo se ranije objašnjenih pogrešaka unutar mreže, njih dobivamo uz pomoć funkcija gubitaka poput srednje kvadratne pogreške, Huber gubitka i sl. Pretpostavimo da je početno stanje težina za pojedini ulaz i pristranosti slučajno, proizlazi da će i prvi rezultat nad trening skupom podataka također biti slučajan. **Funkcije koštanja** su način na koji mreži govorimo da je dobiveni rezultat loš i da još treba poraditi na točnosti. Primjerice kod srednje

kvadratne pogreške vrijednosti sume kvadrata dobivenih i traženog rezultata će biti veliki ukoliko je mreža jako pogriješila ili mali ukoliko je mreža uspješno klasificirala objekt. Naravno, gore navedena činjenica vrijedi za jednu iteraciju kroz trening set podataka (bavimo se nadziranim učenjem), neuronska mreža može imati tisuće trening iteracija. U nastavku ćemo pobliže i matematički objasniti koncept funkcija koštanja. [19]

2.3.3. Funkcije koštanja

Ove vrste funkcija su daljnje prodiranje s više razine analize pogrešaka u mehanizme koji nam omogućavaju samu analizu općenito. Funkcije koštanja možemo pobliže intuitivno objasniti korištenjem analogije planine. Zamislimo li da smo na vrhu planine, a naš je zadatak spustiti se do nizine, kojim ćemo putem krenuti? Logičan prvi korak je pogledati sve moguće smjerove kojima se možemo kretati i odbaciti one smjerove ili puteve koji nas vode ponovo prema vrhu. Naposljetku, od onih puteva koji vode do nizine odabrat ćemo onaj kojim ćemo najbrže stići do nizine, odnosno odabrat ćemo najstrmiji put. Ukoliko odabiremo put koji vodi prema gore on će nas koštati puno više energije i vremena, ali ukoliko odaberemo ići nizbrdo biti ćemo na dobitku pa taj put nizbrdo ima negativno koštanje. Upravo je koštanje baza za funkcioniranje algoritama za nadzirano učenje, a cilj spomenutih algoritama je pronaći minimalni trošak za svaku pojedinu iteraciju u procesu treniranja (učenja) neuronske mreže. U svojoj suštini, funkcije koštanja uzimaju kao ulaze (u funkciju) sve ulaze i pripadne težine neuronske mreže, a kao izlaz daju jedan jedinstveni broj koji označava koštanje, a vezan je uz sve moguće iteracije treniranja (dakle tražimo vrijednost koja će opisati skup svih iteracija). Osnovna podjela funkcija koštanja ovisna je o tipu problema koji rješavamo neuronskom mrežom, odnosno radi li se o klasifikaciji ili regresiji. Kod klasifikacije varijablu koju tražimo ili „predviđamo“ je kategorička, dok je kod regresije ona kvantitativna. Klasifikacija i regresija spadaju u domenu prediktivnog modeliranja koje je usko vezano uz neuronske mreže i njihovu osnovnu namjenu. [20] [21]

2.3.3.1. Regresijske funkcije koštanja

Regresijske funkcije koštanja su **srednja apsolutna greška** ili **L1 gubitak**, **srednja kvadratna pogreška** ili **L2 gubitak** te **Huber gubitak**. Budući da su L1 i L2 gubitak najčešće korišteni za njih ćemo dati definiciju i formule. Srednja apsolutna greška predstavlja prosjek apsolutnih razlika između predviđenih vrijednosti (od strane mreže) te stvarnih vrijednosti (ciljeva). Rečeno zapisano matematičkom formulom izgleda ovako: [21]

$$MAE = \frac{1}{n}\sum_{j=1}^{N} |\, y_j - x_j|.$$

Gdje je y vrijednost dobivena modelom, a x stvarna vrijednost koja bi trebala biti postignuta od strane mreže. Naravno, možemo zaključiti prema formuli da ova funkcija ima potencijalni problem, a to je da ukoliko su neke vrijednosti pretjerane (pozitivna vrijednost greške), a neke previše male (negativna vrijednost greške) može doći do njihovog poništavanja. Nadalje, ovaj problem rješava korištenje L2 gubitka ili funkcije srednje kvadratne pogreške koja predstavlja prosjek kvadrata razlika dobivenih vrijednosti i traženih vrijednosti. Budući da sad radimo s kvadratima, efektivno izbjegavamo problem poništavanja koji imamo kod L1 gubitka. L2 gubitak formuliramo na sljedeći način: [21]

$$MSE = \frac{1}{n}\sum_{j=1}^{N}(y_j - x_j)^2.$$

Također, L2 gubitak se koristi kod slučajeva gdje ulazi i izlaze imaju male razlike, budući da kvadriranjem vrijednosno velike greške imaju puni veći utjecaj. [21] [22]

2.3.3.2. Klasifikacijske funkcije koštanja

Klasifikacijske funkcije koštanja su **gubitak unakrsne entropije** (eng. *cross-entropy loss*) i **Hinge gubitak.** Gubitak unakrsne entropije ima prvenstveno veze s entropijom, kao što mu i ime govori. Ustvari, entropija predstavlja vrijednost kojom izražavamo mjeru „nereda“ ili nesigurnosti u skupu podataka. Matematički entropija je negativna suma produkata vjerojatnosti nekog događaja i njegovog logaritma s bazom 2 kroz sve moguće događaje. Odnosno, $H = \sum_i p_i(\log_2 p_i)$ respektabilno. Sada, gubitak unakrsne entropije ustvari je definiran kao negativna suma produkta očekivane klase i prirodnog logaritma predviđene klase od svih mogućih klasa. Negativni predznak se koristi zato jer pozitivni logaritam, kao što znamo, brojeva manjih od 1 daje negativne vrijednosti s kojima je modelu teže raditi. Rečeno formuliramo ovako: [21]

$$L_{cross-entropy}(x, y) = -\sum_i y_i \log(x_i).$$

Važno je napomenuti da kod binarnih problema klasifikacije gubitak unakrsne entropije za kompletni set trening primjera bi bio nešto drugačiji zbog redukcije prethodno navedene formule. Hinge gubitak se ponajviše koristi kod SVM (eng. *Support Vector Machines*) metode strojnog učenja, a predstavlja zgodan način na koji možemo kazniti loše predviđene vrijednosti, ali i vrijednosti koje su možda točne a opet nepouzdane. Hinge gubitak je dan sljedećom formulom:

$$L_i = \sum_{j \neq j_i} \max(0, s_j - s_{ji} + 1). \text{ [21] [22]}$$

Funkcije koštanja ili gubitaka čine posljednji dio mozaika koji nam je potreban kako bi koncept učenja neuronske mreže bio potpun. Sada, možemo decidirano opisani algoritme učenja. Dakle, algoritam učenja nam ustvari omogućava pronaći globalni minimum funkcije koštanja. Odnosno, algoritmi učenja prilagođavaju težine i pristranost mreže, tako da je za izlaz postignuti minimalni rezultat funkcije koštanja. Vrlo je važno napomenuti da se radi o globalnom minimumu funkcije, budući da mnogi algoritmi nailaze na problem s lokalnim minimumom. Konkretno, pojedine funkcije ne znaju da su zapele u lokalnom minimumu, međutim postoje algoritmi i metode koje rješavaju navedeni problem stanke kod lokalnog minimuma funkcije. Prisjetimo li se prije spomenutog primjera s planinom i najbržim putem prema dolje, algoritmi učenja ili optimizacije su ništa drugo doli iskusni planinari koji nam omogućavaju da se spustimo do doline najbržim putem i uz najmanji utrošak resursa. U usporedbi s prvom analogijom algoritma kao maestra koji okreće kotačiće ulaznih težina i pristranosti sada možemo nazvati pravim imenom, optimizator parametara. Možda na prvi pogled trivijalni problem pronalaska minimuma neke funkcije, dakako da nam derivacije govore o kretanju funkcije. Međutim, zamislite li kompleksnu multivarijabilnu funkciju kao površinu u koordinatnom sustavu s mnogo ulaznih parametara i pristranosti, onda problem pronalaska minimuma itekako postaje zahtjevan posao. S matematičke perspektive algoritmi učenja se odnose na pronalazak specifičnog vektora koji nas vodi do globalnog minimuma neke funkcije te ponavljanja određenog seta koraka do postizanja minimuma. [19] [20]

Zaključno i nezavisno od implementacijskih detalja, važno je razumjeti da je glavni problem „učenja“ neuronske mreže ustvari minimizacija neke od funkcije koštanja (gubitaka). Naravno, posljedično tome zaključku poželjno je spomenuti da funkcija koštanja koju koristimo mora imati uglađeni spust (sjetimo se derivabilnosti) kako bi pronašli lokalni minimum krećući se malim koracima. Upravo zbog ovakve matematičke prirode neuronskih mreža, svaki pojedini neuron ima raspon aktivacija, umjesto binarne vrijednosti 0 ili 1 kao biološki neuron. Mozaik neuronske mreže je sada potpun, uz osnovni **model** tu je i **funkcija koštanja (gubitaka)** te **algoritam učenja** koji čine srž funkcionalnosti mreže.

2.4. Pregled prijašnjih istraživanja

Kako bi imali konkretne smjernice dobre prakse osvrnut ćemo se na neka od prijašnjih istraživanja iz područja algoritama neuronskih mreža. Konkretno nas zanima po kojim parametrima možemo vršiti usporedbu pojedinog algoritma, ali i kako ćemo vrednovati rezultate.

2.4.1. Modeliranje svojstava armiranog betona

Prvo istraživanje je za kao cilj imalo odrediti performanse različitih algoritama treniranja neuronskih mreža (algoritama učenja) u modeliranju svojstava betona armiranog čelikom. U istraživanju je korištena arhitektura višeslojnog perceptrona, a informacije putuju isključivo širenjem prema unaprijed. Zadatak neuronske mreže jest razviti prediktivni model korištenjem atributa omjera sastojaka, elastične snage i ostalih specifičnih svojstava. Dakle, ovdje se radi o nadziranom učenju budući da su i sami podaci za treniranje podijeljeni u set podataka za treniranje i set podataka za testiranje. Korišteno je nekoliko različitih algoritama učenja kako bi se usporedile performanse, a pripadaju tri skupine algoritama: gradijentni spust, Levenberg Marquardt (quasi-Newton) te genetički algoritmi. Sama struktura mreže je izrazito jednostavna a mrežu čine ulazni sloj s tri neurona, skriveni sloj s 10 neurona, te izlazni sloj s 5 neurona. Ova studija je usporedila prediktivna svojstva svakog pojedinog algoritma, a parametri mjerenja se tiču brzine konvergencije algoritma učenja ka globalnom minimumu funkcije koštanja. Ukupne performanse pojedinog algoritma su mjerene koeficijentom determinacije koji ustvari govori o jačini povezanosti ulaznih varijabli. Nadalje, korištena je i poznata statistička metoda apsolutne frakcije varijance i dakako korijen srednje kvadratne pogreške (eng. *RMSE- Root Mean Square Error*). Ne smijemo zaboraviti spomenuti ni prosječni postotak pogreške pojedinog algoritma (eng. *MPE – Mean Percentage Error*) Također, osim numeričkih vrijednosti u ovoj studiji korištene su i grafičke reprezentacije rezultata, odnosno dijagrame koji prikazuju suodnos predviđenih i očekivanih vrijednosti. [23]

2.4.2. Predikcija mehaničkih svojstava i obradivosti bakrenih legura

Drugo istraživanje se također bavilo usporedbom različitih algoritama učenja u treniranju neuronske mreže, a tiče se predikcije mehaničkih svojstava i obradivosti bakrenih legura. U radu je korištena arhitektura višeslojnog perceptrona, a korištena su i dva pristupa treniranju neuronske mreže. Prvi pristup obuhvaća obitelj algoritama gradijentnog spusta, dok drugi pristup treniranju obuhvaća genetske algoritme učenja. Kako bi pojedine razvijene modele mogli uspoređivati autori studije koristili su korijen srednje kvadratne pogreške, apsolutnu frakciju varijance te prosječnu apsolutnu postotnu pogrešku. I ova studija također

koristi grafičke reprezentacije dobivenih rezultata istovjetne kao i u prethodno spomenutom primjeru, a budući da možemo uočiti ponavljanje statističkih metoda za vrednovanje pojedinog algoritma u nastavku ćemo ih definirati i formulom radi boljeg razumijevanja njihovog funkcioniranja. [24]

Korijen srednje kvadratne pogreške je dan formulom: [24]

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N} |d_i - \hat{y}_i|} \ ,$$

gdje N predstavlja broj podatkovnih točaka, odnosno trening podataka, a ustvari nam predstavlja korijen kvadratnih razlika između predviđenih i očekivanih vrijednosti. Druga mjera je apsolutna frakcija varijance koja je ustvari korelacijski koeficijent, odnosno govori o smjernu linearne povezanosti između dvije varijable i dana je formulom: [24]

$$r^2 = 1 - \left(\frac{\sum_{i=1}^{N}(d_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(\hat{y}_i)^2)}\right).$$

Naposljetku, imamo prosječnu apsolutnu pogrešku koja nam kao mjera jednostavno govori koliko je precizan sustav predikcije, a dana je formulom: [24]

$$MAPE = \frac{1}{N} * \sum_{i=1}^{N} \left|\frac{d_i - \hat{y}_i}{d_i}\right| * 100\% \ .$$

Valja imati na umu da se pregled ovih istraživanja bavi regresijskom predikcijom, odnosno kvantitativnim varijablama pa su shodno tome i mjere koje se koriste za usporedbu performansi one iz domene regresijske analize. Iz presjeka spomenutih istraživanja također možemo zaključiti da je osim standardnih mjera usporedbe poželjno koristiti grafičke prikaze koje nam omogućavaju alati kako bi mogli preciznije i točnije reprezentirati dobivene rezultate.

3. Uspoređivanje algoritama neuronskih mreža

Nakon definiranja općih teoretskih koncepata koji su iznimno važni za razumijevanje funkcioniranja neuronskih mreža krećemo na same algoritme učenja (optimizacije). Prvo ćemo spomenuti njihove klasifikacije, a zatim opisati nekoliko različitih vrsta algoritama optimizacije koji su dostupni za treniranje neuronske mreže unutar alata Neural Designer. [31] Također, svakome algoritmu posvetit ćemo nekoliko riječi opisa te prednosti i slabosti svakog pojedinog algoritma.

3.1. Podjele algoritama učenja

O podjeli algoritama učenja ovisi i točka s koje ih promatramo, a pojedinih ih autori razvrstavaju generalno u 2 velike skupine. U prošlosti se često spominjala podjela na deterministčke i stohastičke algoritme optimizacije neuronskih mreža. Deterministčke algoritme karakterizira činjenica da u svakom koraku procesa optimizacije akcije koje izvršava su eksplicitno određene formulama. Dakle, deterministički algoritmi mogu relativno efikasno doći do minimuma funkcije koštanja. Međutim, ako funkcija ima oveći broj lokalnih minimuma ovakvi algoritmi mogu naići na veliki problem i jednostavno zapeti u jednom od lokalnih minimuma efektivno promašujući globalni minimum koji je cilj. Iako, danas postoje mnoge metode i unaprijeđene verzije ovakvih algoritama koje taj problem mogu relativno uspješno anulirati. S druge strane imamo stohastičke algoritme učenja koji predstavljaju alternativnu skupinu algoritama, a ključna razlika je da ova skupina generira i koristi slučajne varijable. Takav pristup možda čini ovu skupinu algoritama sporijom, ali omogućava veću vjerojatnost pronalaska pravog globalnog minimuma funkcije. Nadalje, kroz perspektivu korištenja reda derivacije za pronalazak najbolje „nizbrdice“ funkcije koštanja, možemo spomenuti i podjelu na algoritme optimizacije gradijenta prvog reda i drugog reda. Ovom podjelom ćemo se i detaljnije baviti u sklopu teme ovoga rada u nastavku. [25][26]

3.1.1. Algoritmi optimizacije gradijenta prvog reda

Ova skupina algoritama minimizira (ili maksimizira) funkciju koštanja koristeći vrijednosti samog gradijenta, naravno u relaciji s parametrima. Odnosno, informacije o gradijentu nam uvijek govore koja će točno promjena parametra minimizirati pogrešku (funkciju koštanja) najviše u danom trenutku. Jasno je da nam je potreban gradijent, a derivacije prvog reda nam doslovno daju tangentu (linija koja dotiče krivulju) za točku na površini funkcije koštanja. Gradijent funkcije je ustvari vektor koji predstavlja multivarijabilnu generalizaciju

derivacije $\frac{dy}{dx}$ koja predstavlja trenutnu brzinu promjene *y* nad *x*. Međutim, naš gradijent ovisi o više varijabli i zato koristimo parcijalne derivacije. Dakle, možemo reći da je rezultat gradijenta neke funkcije polje vektora. U ovoj skupini algoritama učenja gradijent je predstavljen uz pomoć Jacobijeve matrice, odnosno matrice koja se sastoji od parcijalnih derivacija prvog reda ili gradijenata. Ako je Jacobijeva matrica kvadratna, onda je multivarijabilna generalizacija gradijenta. Ključno je da za multivarijabilne funkcije koristimo gradijent, a ne uobičajene derivacije. Ovoj obitelji algoritama pripadaju metoda gradijentnog spusta te metoda konjugiranog gradijenta koje ćemo uspoređivati u okviru rada. [26]

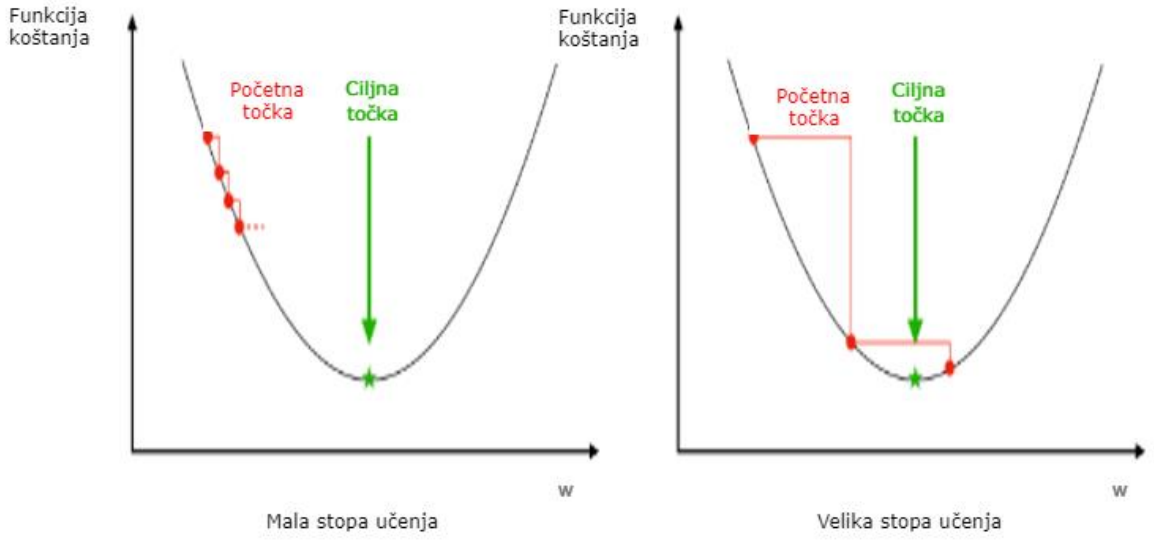
3.1.2. Algoritmi optimizacije gradijenta drugog reda

Kao što im i samo ime govori, ova skupina algoritama učenja ili optimizacije koristi derivacije drugog reda za minimiziranje funkcije koštanja. Umjesto Jacobijeve matrice u ovom slučaju koristimo tzv. Hessianovu matricu. Hessianova matrica je matrica parcijalnih derivacija drugog reda, za razliku od parcijalnih derivacija prvog reda koje se nalaze unutar Jacobijeve matrice. Poznato je da su parcijalne derivacije drugog reda kompleksnije za izračunati od onih prvog reda, stoga se algoritmi optimizacije drugog reda rjeđe koriste, upravo zbog većeg trošenja računalnih resursa. Zašto koristimo matricu parcijalnih derivacija drugog reda? Upravo zato jer nam parcijalne derivacije drugog reda govori da li se derivacija prvog reda povećava ili smanjuje. A potonja informacija nam može biti od koristi budući da nam može ukazati na zakrivljenost funkcije. Konkretno, uz pomoć parcijalnih derivacija drugog reda možemo doći do kvadratne površine koja dodiruje zakrivljenost površine grafa funkcije koštanja. Iako, Jacobijeva matrica može, ali i ne mora biti kvadratna, Hessianova matrica je uvijek kvadratna matrica parcijalnih derivacija drugog reda. Budući da daljnji uvidi u spomenute matrice zahtijevaju dodatnu obradu nekoliko srodnih polja iz vektorske matematike njihovom daljnjom analizom se nećemo baviti. [26]

U nastavku ćemo se detaljnije posvetiti pojedinom algoritmu učenja koji ćemo kasnije implementirati putem Neural Designera. Spomenuti pregled će biti kratak i fokusiran na ključne karakteristike algoritama, kao i njihove prednosti te nedostatke.

3.2. Algoritam gradijentnog spusta

Vjerojatno jedan od najpoznatijih algoritama učenja današnjice i de facto standard za uvod u neuronske mreže kod velikog broja autora. Jednostavnim riječima algoritam gradijentnog spusta se kreće prema negativnom gradijentu. Nove parametre dobivamo tako da trenutnim parametrima oduzmemo umnožak gradijenta (negativan jer ga u pravilu želimo smanjiti) i stope učenja. Možda je bolje rečeno zapisati konkretnom formulom: $w(t+1) = w(t) - \eta g$, gdje g predstavlja gradijent: $g = \triangle f(w(t))$, a η predstavlja brzinu ili stopu učenja. [27] Za vjerodostojnu aproksimaciju $\eta \to 0$, ali naravno da bi to nepotrebno usporavalo konvergenciju algoritma prema lokalnom minimumu i zato se u praksi koriste veće vrijednosti. Stopa učenja je jednostavno najmanji mogući put koji algoritam uzima kako bi se primicao dnu nizbrdice (globalni minimum funkcije koštanja). Gornjom formulom smo ustvari objasnili suštinu funkcioniranja gradijentnog spusta jer za svaku prethodnu težinu mi možemo odrediti novu vrijednost težine uz pomoć gradijenta i stope učenja. Pogledom na formulu možemo zaključiti da algoritam gradijentnog spusta ima dva posla koja obavlja, prvi je poboljšanje parametara neuronske mreže (težina i pristranosti), a drugi je izračun negativnog gradijenta (za smjer kretanja) te stope učenja. Neki softverski alati ostavljaju stopu učenja fiksnu, međutim danas se općenito koristi dinamička stopa učenja koja se prilagođava kroz svaku pojedinu iteraciju. Prirodno se javlja pitanje zašto uopće koristimo malu stopu učenja, odnosno korake prema kretanju nizbrdo? Pa, ukoliko je stopa učenja prevelika, a imamo funkciju s vrlo uskim područjem lokalnog minimuma, onda se može dogoditi da prevelikim korakom promašimo lokalni minimum, što je i prikazano donjom ilustracijom na jednostavnoj funkciji.



Slika 8: Usporedba stopi učenja za gradijentni spust

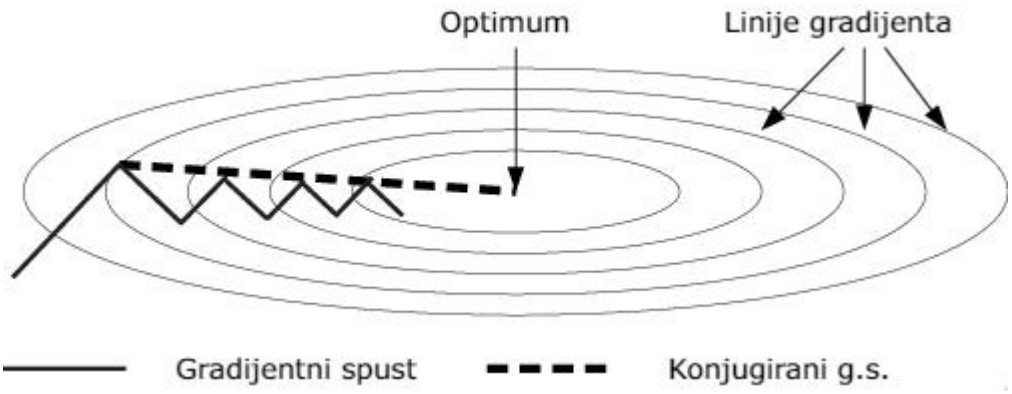
Koraci algoritma gradijentnog spusta jesu:

- Evaluacija indeksa funkcije koštanja
- Ako kriterij zaustavljanja nije zadovoljen
    - Izračunaj stopu treniranja
    - Izračunaj negativni smjer gradijenta
    - Poboljšaj parametre (w)
- Ako je kriterij zadovoljen zaustavi, inače ponovi prethodne korake

Prednosti gradijentnog spusta kao algoritma učenja neuronske mreže jesu jednostavnost implementacije i lakoća razumijevanja samog algoritma. S druge strane, ovaj algoritam itekako ima svojih **nedostataka** i nije omiljen pretežito zbog sporosti konvergencije prema lokalnom minimumu. Nadalje, budući da ovaj algoritam izračunava gradijent za cijeli skup podataka u svakoj iteraciji (a iteracija može biti na stotine i tisuće) kalkulacije su vrlo spore. Također, gradijentni spust ima problema s funkcijama koje imaju duge i uske strukture, nešto nalik fjordovima. Naime, za spust do globalnog minimuma je u spomenutom slučaju potrebno mnoštvo iteracija jer negativni gradijent je smjer u kojemu se funkcija koštanja najbrže smanjuje, ali to ne znači da ćemo imati i najbržu konvergenciju. Općenito se ovaj algoritam koristi u slučajevima masivnih neuronskih mreža s tisućama parametara, a glavni razlog za to je korištenje Jacobaijanove matrice koja je prvog reda i manje računalno zahtjevna od matrice drugog reda. [6] [27] [28]

3.3. Algoritam konjugiranog gradijentnog spusta

Algoritam konjugiranog gradijentnog spusta predstavlja bolju inačicu prethodnog algoritma i vrlo dobar pokušaj postizanja brzine konvergencije ka globalnom minimumu kao i algoritmi drugog reda. Pretežito se ovaj algoritam koristi za optimizaciju diferencijabilnih funkcija s puno varijabli, neki autori ističu ovaj algoritam pridjevom čaroban, a u nastavku ćemo se dotaknuti detalja. Cijeli algoritam se bazira na konceptu konjugiranih vektora, odnosno na konceptu konjugiranih vektora prema nekoj kvadratnoj funkciji. Međutim, ovaj koncept može biti nejasan pa ćemo ga uz pomoć usporedbe s gradijentnim spustom pokušati objasniti. Dakle, kod uobičajenog gradijentnog spusta u svakom novom koraku izračunavamo gradijent ispočetka, budući da će taj novi gradijent biti okomit (vektor) na trenutni gradijent i spustit će se niz padinu funkcije koštanja ponovno. Međutim, što ako pokušamo izračunati konjugirani smjer s obzirom na trenutni smjer i minimizirati putem te linije? Prema samoj definiciji konjugacije svaka promjena gradijenta koja je rezultat tog koraka minimizacije će biti ortogonalna prethodnom smjeru kretanja. Ili jednostavnije, ako dostignemo minimum u tom prethodnom smjeru, onda ćemo i ostati u minimumu u tom smjeru.



Slika 9: Usporedba konjugiranog i uobičajenog gradijentnog spusta

Na gornjoj slici možemo lakše uočiti spomenuto poboljšanje konjugiranog spusta. Naime, možemo vidjeti da za konvergenciju uobičajenom gradijentu treba čak 9 iteracija, dok konjugiranom trebaju samo dvije. Upravo ova slika jasno ističe zašto mnogi preferiraju konjugirani gradijentni spust. Optimum naravno predstavlja globalni minimum funkcije koštanja, a linije gradijenta ustvari predstavljaju točke iste udaljenosti od središta, nešto poput izohipsa u geografiji, iako se ovdje radi o dvodimenzionalnom prikazu. Što se tiče samog algoritma, nove parametre dobivamo tako da od starih parametara oduzmemo umnožak konjugiranog gradijenta i stope učenja. Odnosno, ako je inicijalni vektor treniranja smjera dan s $d = -g$, gdje je g ponovno gradijent, ova metoda konstruira sekvencu trening vektora smjera kao $d^{(i+1)} = g^{(i+1)} + d^{(i)} * \Upsilon^{i}$. Gdje je Υ parametar konjugacije, a postoje različiti načini njegova izračuna u koje u ovome radu nećemo ulaziti. Tek sada možemo formulom zapisati način dobivanja novih parametara: $w^{(i+1)} = w^{(i)} + d^{(i)} * \eta^{i}$. [27] Koraci algoritma konjugiranog gradijentnog spusta u alatu Neural Designer jesu:

- Evaluacija indeksa funkcije koštanja
- Ako kriterij zaustavljanja nije zadovoljen
    - Izračunaj stopu treniranja
    - Izračunaj konjugirani smjer gradijenta
    - Poboljšaj parametre (w)
- Ako je kriterij zadovoljen zaustavi, inače ponovi prethodne korake

Prednosti ovog algoritma smo već nekoliko puta spomenuli, pa ćemo ukratko ponoviti da ovaj algoritam može brže stići do minimuma funkcije koštanja koristeći konjugirane vektore i na taj način trošeći manje vremena. **Nedostaci** uključuju poveći utrošak računalnih resursa, budući

da metode za izračun parametra konjugacije, ovisno o problemu, mogu biti zahtjevne. Također, za multivarijabilne probleme može biti neefikasno izračunavati parametar konjugacije jer iako na gornjoj slici imamo samo 2 izračuna do minimuma u stvarnosti ih možemo imati i nekoliko desetaka, ovisno o kompleksnosti modela. Algoritam konjugiranog gradijentnog spusta se također preporuča za velike neuronske mreže. [6] [25] [27]

3.4. Quasi-Newton metoda

Ova metoda se u literaturi i ponekad naziva metoda varijabilnih metrika, a funkcionira na temelju aproksimacije inverzne Hessianove matrice koristeći samo podatke dobivene izračun gradijenta prvog reda. Možemo reći da je Quasi-Nweton algoritam učenja hibridno rješenje i zlatna sredina između metoda prvog i drugog reda. Ovom metodom zaobilazimo potrebu za eksplicitnim računanjem matrice gradijenata drugog reda (Hessian), što uvelike rasterećuje računalne resurse. Međutim, i dalje moramo pohraniti informacije o aproksimaciji inverzne Hessianove matrice. Spomenuta aproksimacija se odvija u svakoj novoj iteraciji algoritma, a u nastavku ćemo se baviti implementacijskim detaljima algoritma. Kao što smo ranije već spomenuli, Hessianova matrica se sastoji od parcijalnih derivacija drugog reda od funkcije koštanja, a Quasi-Newton metoda aproksimira Hessianovu matricu uz pomoć matrice *G*, i to koristeći samo parcijalne derivacije prvog reda. Odnosno, nove parametre dobivamo tako da od starih parametara oduzmemo umnožak aproksimacije inverzne matrice, gradijenta te stope učenja. Stoga, Quasi-Newton formula glasi: $w^{(i+1)} = w^{(i)} - (G^{(i)} * g^{(i)}) * \eta^{i}$, gdje je *G* aproksimirana matrica, *g* predstavlja ranije spomenuti gradijent, a η i dalje ostaje stopa učenja. [27] Iako se gornja formula naizgled čini jednostavnom, moramo spomenuti da aproksimacija inverzne Hessianove matrice može doći u više varijanti, ovisno o metodi dobivanja iste. Dvije najpoznatije metode izračuna G matrice jesu Davidon-Fletcher-Powell formula (DFP) i Broyden-Fletcher-Goldfarb-Shano formula (BFGS). Pritom je potonja preporučena i često korištena unutar zajednice, a aproksimaciju ćemo dati pojednostavljenim oblikom BFGS formule: [6]

$$G_{k+1} = G_k + \frac{1}{g_k^T p_k} g_k g_k^T + \frac{1}{\alpha_k y_k^T p_k} y_k y_k^T ,$$

gdje je G_{k+1} nova aproksimacija, G_k predstavlja aproksimaciju Hessiana iz prošlog koraka. g_k je gradijent vektora iz prethodnog koraka, a $y_k = g_{k+1} - g_k$, dakle razlika između gradijent vektora iz trenutnog i prethodnog koraka. Nadalje, p_k predstavlja smjer traženja, odnosno smjer linijske pretrage unutar skupa, a $p_k = \frac{g_k}{G_k}$. Eksponenti *T* služe kao oznaka za transponirane matrice budući da moramo imati na umu da radimo s matricama cijelo vrijeme. Također, početna aproksimacija matrice *G* je često matrica identiteta (jedinična matrica) pa je

prvi korak ustvari iteracija najboljeg koraka strmog spusta (algoritma koji ovdje ne spominjemo). Iz BFGS formule možemo razlučiti da zadržava simetričnost, a za stabilnost je iznimno važno sačuvati ograničenost s pozitivne strane. Zato je važno biti oprezan kod implementacije BFGS formule u Quasi-Newton metodi jer gubitak pozitivne ograničenosti je moguć zbog ograničene preciznosti linijskog pretraživanja i naravno grešaka zbog zaokruživanja decimalnih vrijednosti. Također postoje varijacije BFGS formule poput **BFGS s ograničenom memorijom**, ali one izlaze van okvira ovog rada i njima se nećemo baviti. Važno je steći okvirno razumijevanje funkcioniranja izračuna aproksimacije inverzne Hessianove matrice i kako istu primjenjujemo u poznati okvir za izračun novih težina, što smo i objasnili. A sada slijede koraci Quasi-Newton metode realizirane u alatu Neural Designer:

- Evaluacija indeksa funkcije koštanja
- Ako kriterij zaustavljanja nije zadovoljen
    - Izračunaj stopu treniranja
    - Izračunaj quasi-Newton smjer treninga
    - Poboljšaj parametre (w)
- Ako je kriterij zadovoljen zaustavi, inače ponovi prethodne korake

Prednosti Quasi-Newton metode su da kod manjih do srednjih neuronskih mreža postiže puno nižu ukupnu pogrešku od ustaljenog algoritma nazadne propagacije, konjugiranog gradijentnog spusta, te simplex pretrage za problem aproksimacije funkcije (regresija). Jedino je Levenberg-Marquardt algoritam postigao manju pogrešku od svih prethodno spomenutih algoritama, uključujući i Quasi-Newtona. Također, Quasi-Newton metoda može imati brzu konvergenciju ka minimumu, a u nekim istraživanjima se pokazao kao najbrži i najpouzdaniji algoritam za manje problemske domene. S druge strane, **nedostatci** su implicitno vidljivi iz prednosti algoritma. Naime, za velike i kompleksne neuronske mreža računalna složenost izračuna aproksimacije postaje puno veća i čini ovaj algoritam iznimno sporim. Također, postoje slučajevi brze konvergencije Quasi-Newton metode, ali u lokalni minimum, a ne globalni. U većim skupovima javlja se i problem s pretjeranom generalizacijom kod nadziranog učenja. Dakle, Quasi-Newton metodu je dobro koristiti za manje do srednje skupove kada želimo postići nisku grešku mreže, a bez ovećeg utroška računalnih resursa koje koriste „prave“ metode drugog reda. [6] [27] [29]

3.5. Levenberg-Marquardt algoritam

I posljednji algoritam kojim ćemo se baviti ima posebnosti kod pristupa rješavanju problema minimuma funkcije koštanja. Ponovno, radi se o algoritmu koji na pametni način koristi informacije iz parcijalnih derivacija prvog reda kako bi došao do konačnog rješenja, tj.

globalnog minimuma funkcije koštanja. Prije nego krenemo u objašnjenje algoritma, važno je znati da za njegovo funkcioniranje računalo mora riješiti veliki sustav linearnih jednadžbi za svaku pojedinu iteraciju. To može preopteretiti računalo ukoliko imamo iznimno veliki broj težina koje moramo izračunati. Dakle, Levenberg-Marquardt algoritam je dizajniran specifično za rad s funkcijama koštanja, konkretno s funkcijom srednje kvadratne pogreške. Ovo se možda čini kao nedostatak, međutim MSE je ionako široko korišten i generalno dobar izbor za izračun funkcije koštanja. Kod ovog algoritma gradijent se koristi samo kao alat koji potvrđuje da algoritam napreduje dobro, dok je glavna zadaća ovog algoritma, s matematičke perspektive, svesti gradijent vektor na 0. Nove težine dobivamo tako da od starih težina oduzmemo umnožak faktora prigušivanja, Jacobiana te gradijenta. Uzmemo li funkciju koštanja kao [27]

$$f = \sum_{i=1}^{m} e_i^2,$$

gdje je m broj instanci unutar skupa podataka, a e je greška za pojedinu instancu. Možemo definirati Jacobianovu matricu funkcije koštanja kao onu koja sadrži derivacije grešaka s obzirom na parametre (težine) na sljedeći način: [27]

$$J_{i,j} = \frac{de_i}{dw_j},$$

za svaki i=1,…,m i j=1,…,n. Gdje je m broj instanci seta podataka, n je broj parametara unutar neuronske mreže, a veličina Jacobian matrice je m*n. Gradijent vektor funkcije koštanja dobivamo na sljedeći način: [27]

$$\nabla f = 2J^T \cdot e,$$

gdje je e vektor pogrešaka. Konačno, uz sve prethodne kriterije zadovoljene možemo aproksimirati Hessianovu matricu sa sljedećim izrazom: [27]

$$H_f \approx 2J^T \cdot J \cdot \lambda I,$$

gdje je λ faktor prigušivanja koji osigurava pozitivnost Hessianove funkcije, a I je jedinična matrica. Naposljetku, možemo definirati izračun težina za pojedinu iteraciju algoritma učenja formulom: [27]

$$w^{(i+1)} = w^{(i)} - \left(J^{(i)T} \cdot J^{(i)} + \lambda^{(i)} I\right) \cdot \left(2J^{(i)T} \cdot e^{(i)}\right).$$

Kada je faktor prigušivanja lambda jednak 0, onda ova metoda postaje čista Newtonova metoda koja koristi samo aproksimaciju Hessianove matrice, dok kod velikog faktora prigušivanja LM algoritam postaje gradijentni spust s vrlo malom stopom učenja. Stoga je faktor prigušivanja inicijaliziran kao velika vrijednost, kako bi početni koraci bili mali u smjeru gradijentnog spusta. Ako bilo koja od iteracija rezultira neuspjehom, lambda se povećava za određeni faktor, inače kako se gubitak smanjuje (koštanje) parametar prigušivanja se također smanjuje tako da se LM algoritam približava Newtonovoj metodi. Upravo ovaj proces od

gradijentnog spusta do Newtonove metode ubrzava konvergenciju algoritma prema minimumu. Rezimirajmo korake LM algoritma u alatu Neural Designer:

- Evaluacija indeksa funkcije koštanja
- Ako kriterij zaustavljanja nije zadovoljen
    - Poboljšaj faktor prigušivanja λ
    - Izračunaj gradijent
    - Izračunaj aproksimaciju Hessiana
    - Poboljšaj parametre (w)
- Ako je kriterij zadovoljen zaustavi, inače ponovi prethodne korake.

Prednosti Levenberg-Marquardt algoritma se vide kod treniranja mreža koje kao funkciju koštanja imaju neki tip sumiranih kvadrata pogrešaka. Također, ovaj algoritam postiže najmanju ukupnu grešku treniranja za aproksimaciju funkcije od svih prethodno spomenutih algoritama, a u manjim do srednjim razinama kompleksnosti neuronske mreže pokazuje se kao i najbrži. **Nedostatci** također postoje, a prvi je očita nemogućnost primjere na funkcije poput korijena sume kvadrata pogreške te funkcije unakrsne entropije. Također, za velike skupove podataka i za velike neuronske mreže Jacobianova matrica postaje ogromna, i shodno tome zahtjeva veliku količinu memorije. Stoga, ovaj algoritam nije preporučeno koristiti za velike skupove podataka ili velike neuronske mreže. [25] [27] [29]

Zaključno je važno napomenuti nekoliko činjenica koje su bitne kod evaluacije rezultata implementacije spomenutih algoritama u arhitekturu mreže. Prva činjenica je da spomenuti algoritmi mogu imati različite varijacije, primjerice za izračun aproksimacije Hessianove matrice, stoga prethodno navedeni algoritmi su implementirani točno kako su i opisani u alat Neural Designer. Drugo, podaci čine integralni dio neuronske mreže i njihovom upoznavanju te analizi ćemo posvetiti vrijeme u zasebnom poglavlju. Naposljetku, o specifičnostima seta podataka ovisit će i performanse pojedinog algoritma, pravi podatkovni znanstvenici slijede se dobrom praksom i algoritme biraju za podatke, a ne obrnuto. Također, važno je naglasiti da su spomenuti algoritmi samo kap u moru bogatstva svijeta neuronskih mreža i da danas brojimo puno veći broj algoritama, ali i njihovih varijacija koje su dostupne za korištenje.

4. Podaci i rezultati implementacije algoritama

U ovome se poglavlju bavimo upoznavanjem i detaljnom analizom podataka koje ćemo koristiti u istraživanju, odnosno kod uspoređivanja performansi pojedinog algoritma. Također, posvetit ćemo vrijeme i opisu neuronske arhitekture koju ćemo koristiti za sve algoritme. Vrlo je važno provesti kontrolu varijabli, a to možemo samo ako svi uvjeti ispitivanja, osim algoritama učenja, ostanu nepromijenjeni. Stoga ćemo se posvetiti i nekim implementacijskim specifičnostima koje nisu spomenute u teoretskom djelu. Nadalje, za svaki pojedini algoritam diskutirat ćemo rezultate i usporediti ih s prethodno iznesenim teorijskim znanjem. Naposljetku, objedinit ćemo rezultate pojedinih algoritama i izvesti zaključak o ukupnim performansama naspram očekivanja, odnosno početnih pretpostavki.

4.1. Set podataka

Set podataka koji je odabran tiče se obnovljivih izvora energije, konkretno prognoziranja izlazne snage solarne elektrane. Prije analize podataka dobro je spomenuti da će naša neuronska mreža rješavati problem prediktivne regresije. Dakle, set podataka u .csv datoteci se sastoji od 10 atributa, a pritom su svi atributi kontinuirani, tj. numerički. Od desetak atributa jedan će biti korišten kao ciljni atribut, odnosno kao cilj koji želimo aproksimirati funkcijom, naravno radi se o izlaznoj snazi solarne elektrane. Što se tiče instanci, odnosno različitih primjera koji upotpunjavaju navedene atribute, imamo ih 2920. Ovaj skup podataka nije odveć veliki, a nije niti premalen u smislu broja instanci. Skup ove veličine je odabran zbog same prirode algoritama koje smo naveli u prethodnom poglavlju.
Sljedeće varijable su sadržane unutar skupa podataka o parametrima za solarnu elektranu: distance-to-solar-noon, temperature, wind-direction, wind-speed, sky-cover, visibility, humidity, average-wind-speed-(period), average-pressure-(period), power generated. Slijede semantički opisi pojedinog atributa radi lakšeg razumijevanja problemske domene:

- **distance-to-solar-noon** predstavlja udaljenost elektrane od solarnog podneva izraženo u radijanima
- **temeperature** predstavlja dnevnu prosječnu temperaturu izraženu u stupnjevima Celzijevim
- **wind-direction** predstavlja dnevno prosječno kretanje vjetra u stupnjevima od 0 do 360
- **wind speed** dnevna prosječna brzina vjetra u m/s
- **sky-cover** je atribut iskazan u rasponu od 0 do 4, gdje je 0-nebo bez oblaka, a 4-potpuno oblačno nebo

- **visibility** predstavlja vidljivost u kilometrima
- **humidity** predstavlja vlažnost zraka izraženu u postotcima
- **average-wind-speed-(period)** je prosječna brzina vjetra u trosatnom vremenskom intervalu u m/s
- **average-pressure-(period)** je prosječni barometarski pritisak u trosatnom vremenskom intervalu izražen u inčima žive (eng. *mercury inches*)
- **power-generated** predstavlja generiranu snagu solarne elektrane za trosatni period izraženu u džulima

Slijedi prikaz osnovnih statističkih pokazatelja za dani skup podataka radi boljeg razumijevanja skupa. Statistički pokazatelji koje ćemo prikazati sljedećom slikom jesu minimalna i maksimalna vrijednost te prosjek i devijacija skupa.

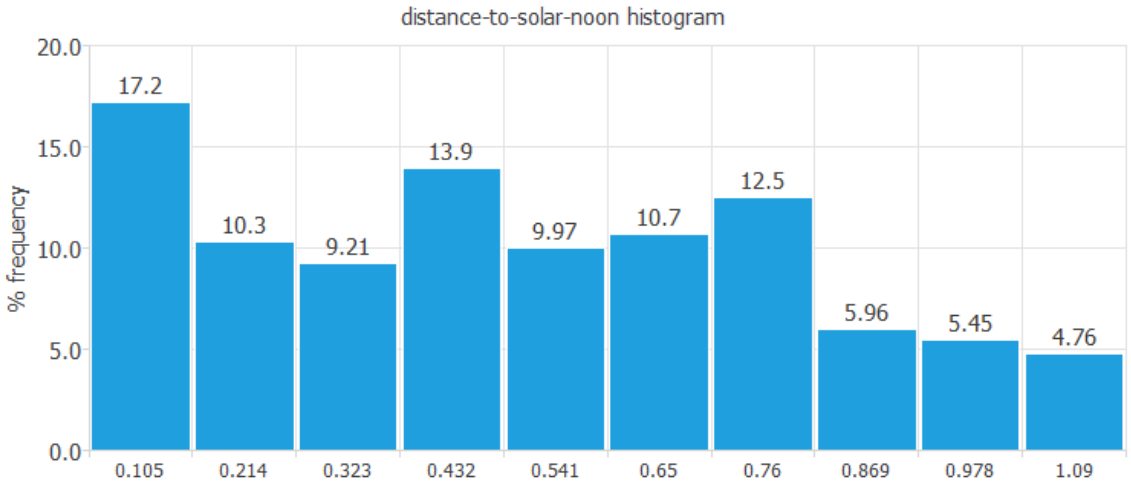
	Minimum	Maksimum	Prosjek	Devijacija
distance-to-solar-noon	0.0504	1.14	0.503	0.298
temperature	42	78	58.5	6.84
wind-direction	1	36	25	6.92
wind-speed	1.1	26.6	10.1	4.84
sky-cover	0	4	1.99	1.41
visibility	0	10	9.56	1.38
humidity	14	100	73.5	15.1
average-wind-speed-(period)	0	40	10.1	7.26
average-pressure-(period)	29.5	30.5	30	0.142
power-generated	0	3.66e+4	6.98e+3	1.03e+4

Slika 10: Osnovne statističke mjere skupa podataka

Naravno, niti jedna analiza bilo kojeg skupa podataka nije potpuna suhoparnim statističkim mjerama nad skupom, već skup moramo potkrijepiti i određenim dijagramima. Konkretno, slijede histogrami za svaki pojedini ulazni atribut.

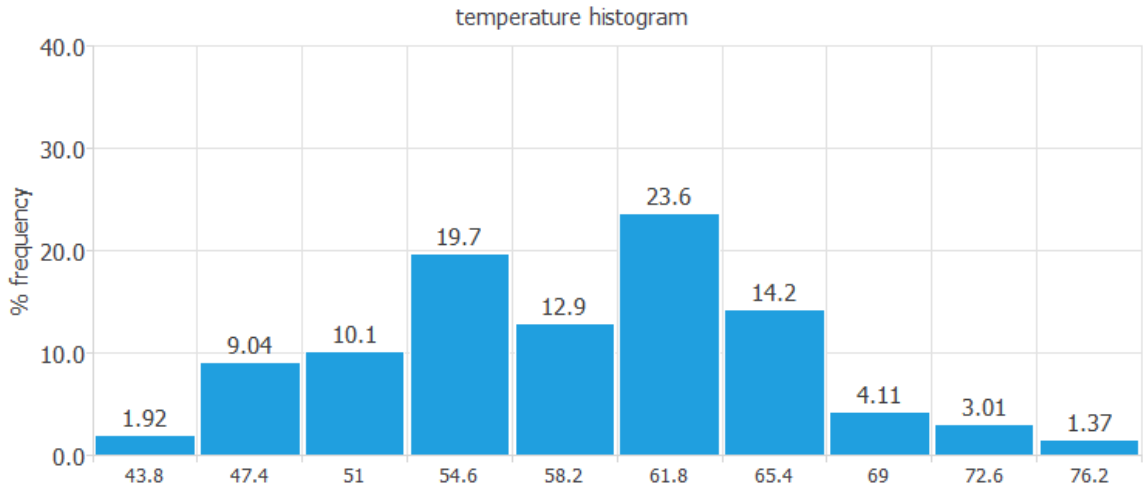
4.1.1. Histogrami atributa

Histogrami nam daju uvid u samu distribuciju podataka unutar cijelog seta podataka. U problemima aproksimacije uniformna distribucija za pojedini atribut je poželjna jer ako imamo raštrkanu distribuciju model će biti lošije kvalitete.



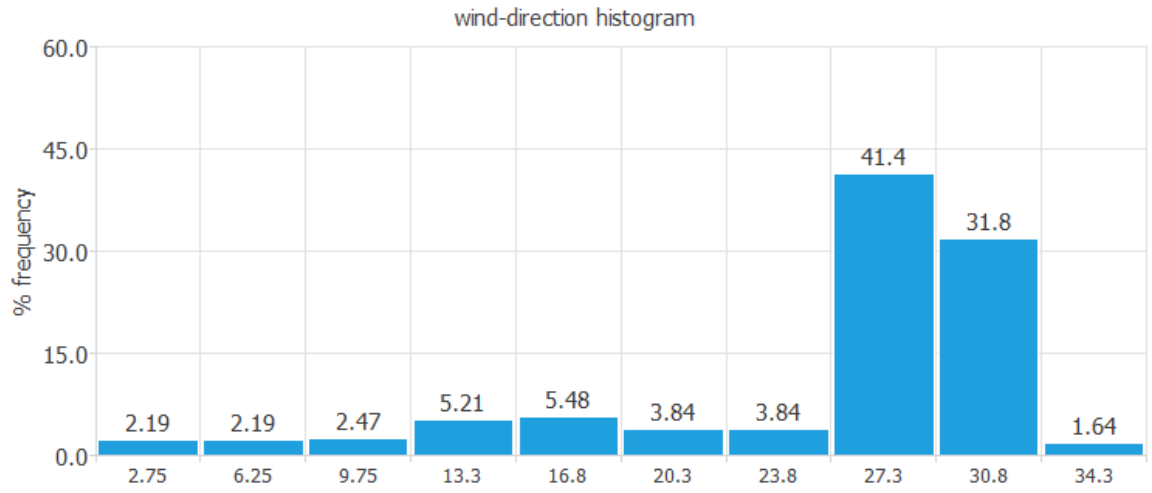
Slika 11: Histogram atributa distance-to-solar-noon

Gornji histogram prikazuje atribut distance-to-soolar-noon. Pritom su na apscisi iskazane vrijednosti u radijanima, a na ordinati su prikazane frekvencije pojavljivanja pojedine vrijednosti radijana. Možemo uočiti da je maksimalna frekvencija 17.2% za vrijednost radijana 0.105, a minimalna 4.76% za iznos radijana 1.09. Možemo reći da je ovaj histogram multimodalne distribucije.



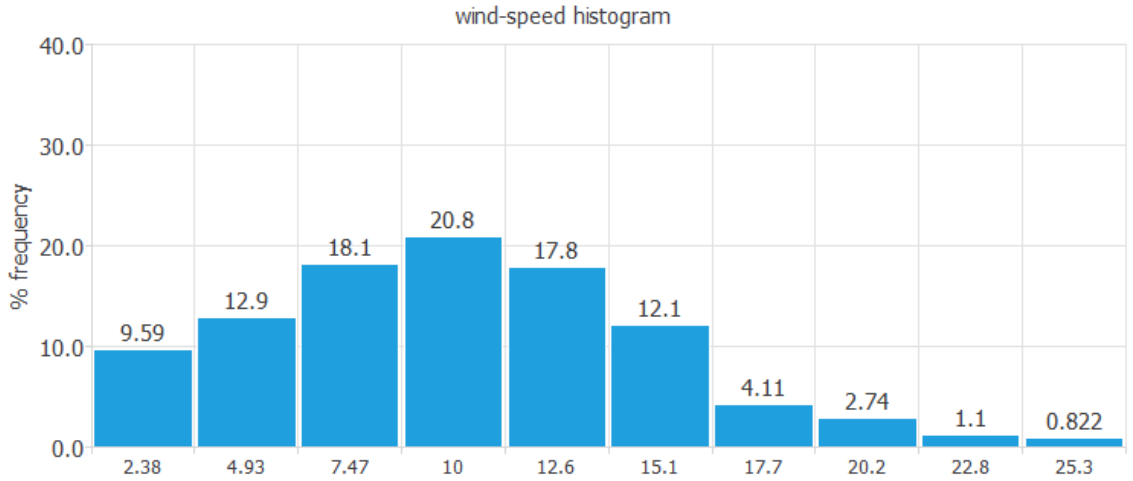
Slika 12: Histogram atributa temperature

Gornji histogram prikazuje atribut temperature. Vidimo da je maksimalna frekvencija 23.6% za vrijednost temperature 61.8, a minimalna 1.37% za iznos temperature 76.2. Možemo reći da je ovaj histogram također multimodalne distribucije.



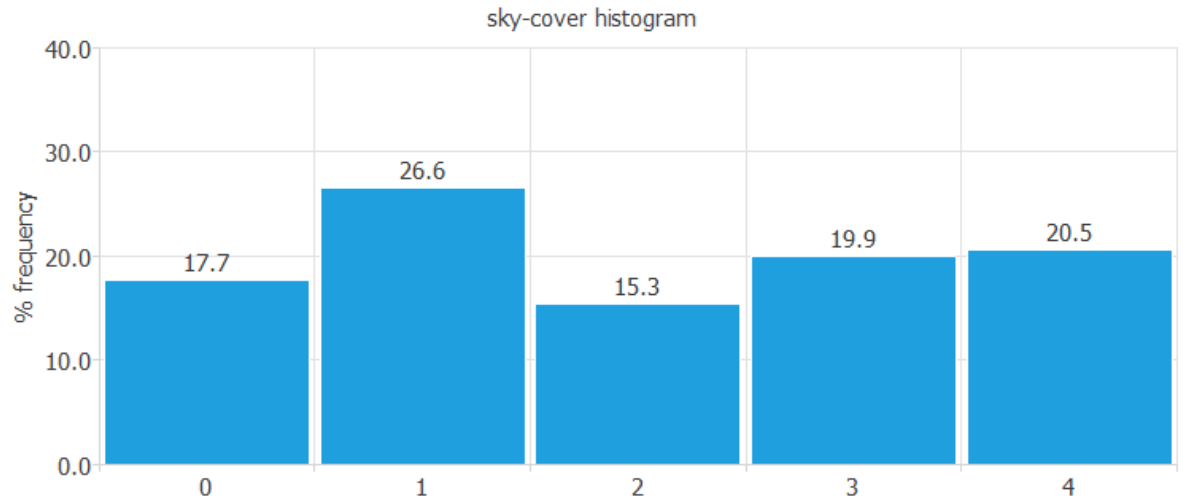
Slika 13: Histogram atributa wind-direction

Gornji histogram prikazuje atribut wind-direction. Iz njega je vidljivo da je maksimalna frekvencija 41.4% za vrijednost stupnjeva 27.3, a minimalna 1.64% za iznos stupnjeva 34.3. Možemo reći da je ovaj histogram multimodalne distribucije.



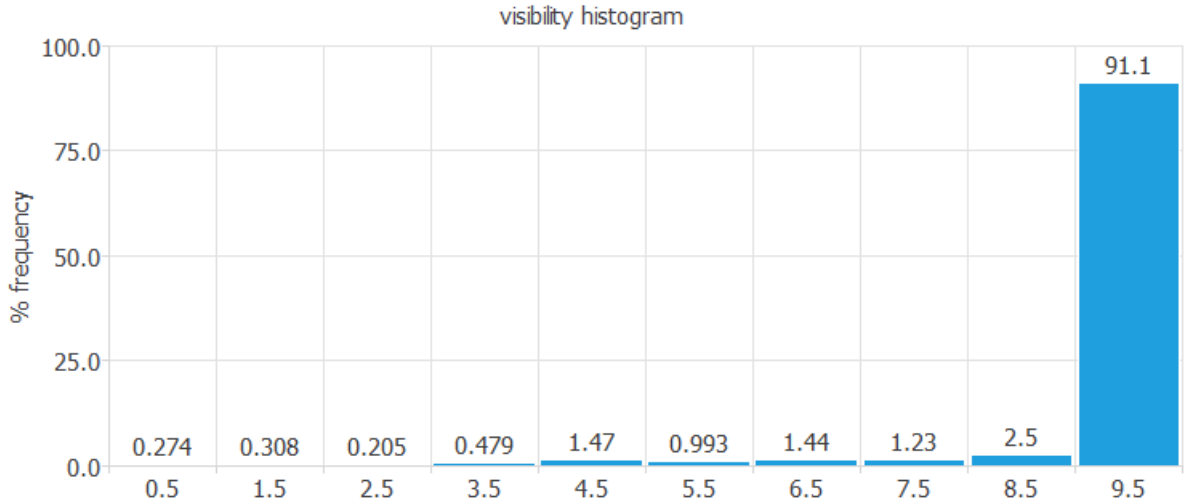
Slika 14: Histogram atribta wind-speed

Gornji histogram prikazuje atribut wind-speed. Možemo vidjeti da je maksimalna frekvencija 20.8% za vrijednost brzine 10 m/s, a minimalna 0.822% za vrijednost brzine vjetra od 34.3 m/s. Ovaj histogram je unimodalno distribuiran i to prema desno.



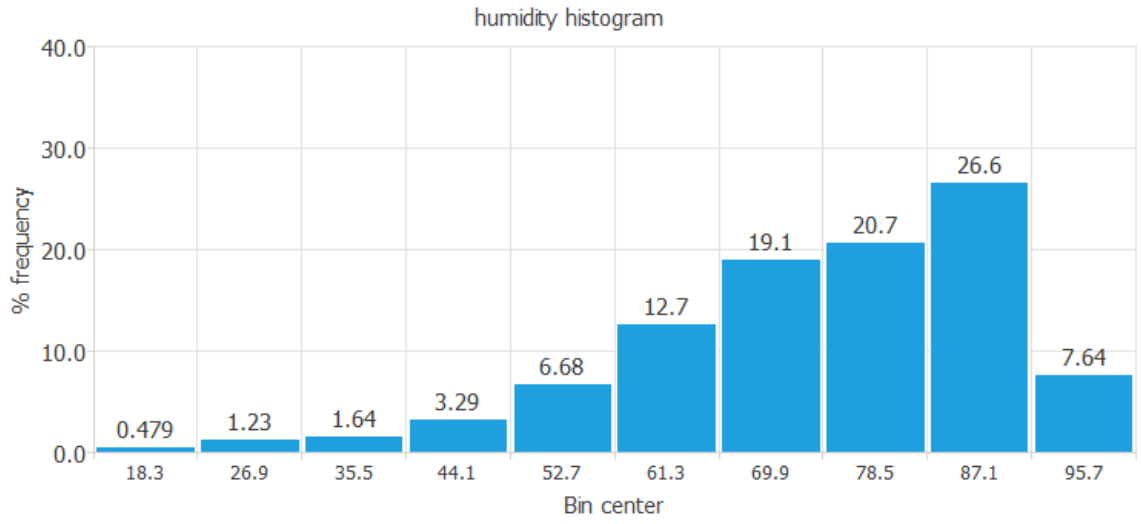
Slika 15: Histogram atributa sky-cover

Gornji histogram prikazuje atribut sky-cover. Možemo uočiti da je maksimalna frekvencija 26.6% za vrijednost 1 na mjernoj skali (0-4), a minimalna 15.3% za vrijednost 2 na skali. Ovaj histogram je uniformno distribuiran.



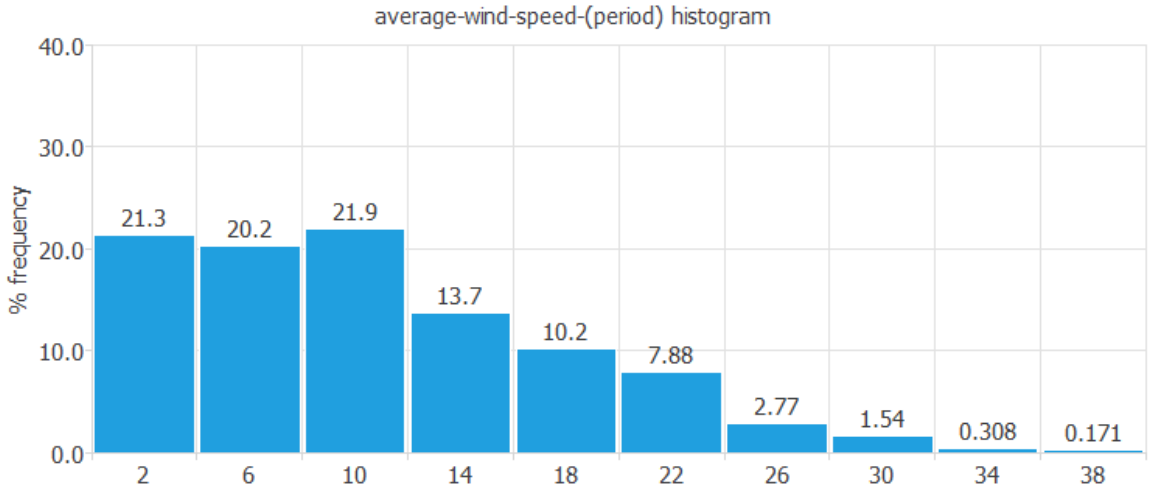
Slika 16: Histogram atributa visibility

Gornji histogram prikazuje atribut visibility. Iz njega vidmo da je maksimalna frekvencija 91.1% za udaljenost od 9.6 km, a minimalna 0.205% za udaljenost od 2.5 km. Ovaj histogram je eksponencijalne distribucije.



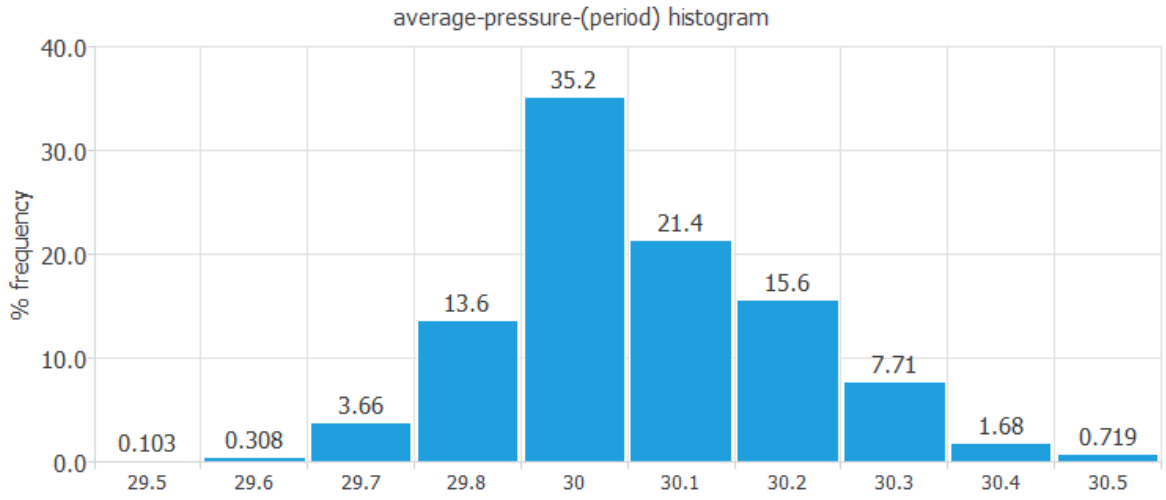
Slika 17: Histogram atributa humidity

Gornji histogram prikazuje atribut humidity. Uočavamo da je maksimalna frekvencija 26.6% za vlažnost 87.1%, a minimalna 0.479% za vlažnost zraka od 18.3%. Ovaj histogram je također unimodalno distribuiran na lijevo.



Slika 18: Histogram atributa average-wind-speed-(period)

Gornji histogram prikazuje distribucije vrijednosti za atribut average-wind-speed-(period). Iz njega možemo iščitati da je maksimalna frekvencija 21.9% za vrijednost 10 te minimalna 0.171% za vrijednost 38. Ovaj histogram je također unimodalno distribuiran, ali na desno.

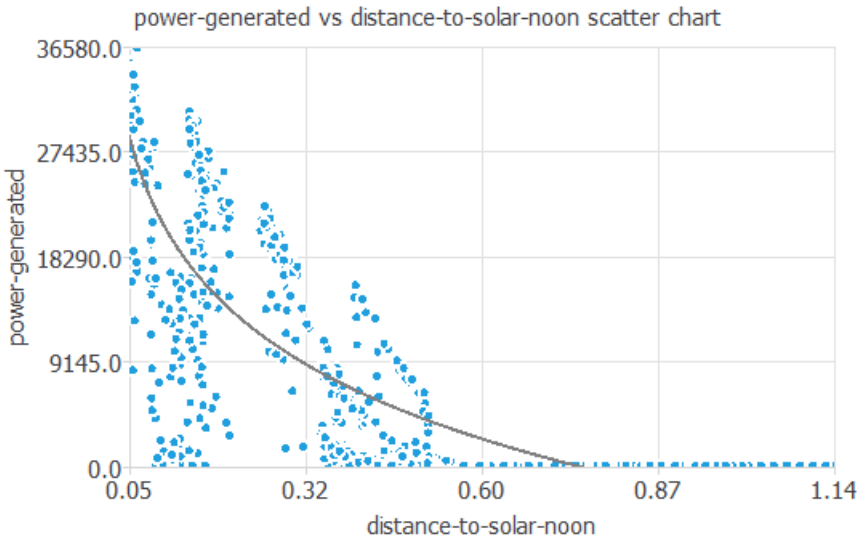


Slika 19: Histogram atributa average-pressure-(period)

Gornji histogram prikazuje distribucije vrijednosti za atributa average-pressure-(period). Iz njega možemo vidjeti da je maksimalna frekvencija 35.2% za vrijednost 30 te minimalna 0.103% za vrijednost 29.5. Ovaj histogram je normalne distribucije.

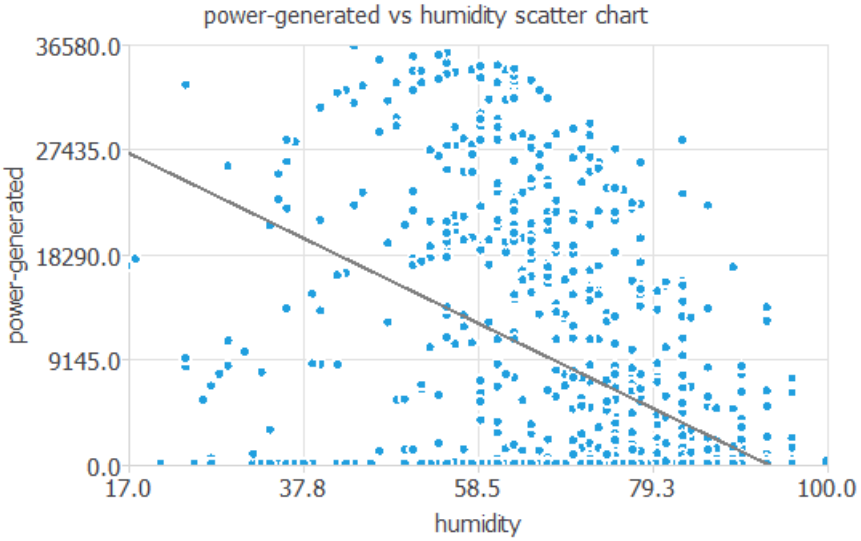
4.1.2. Scatter dijagrami

Scatter dijagrami nam daju uvid u distribuciju vrijednosti između pojedinih ulaznih varijabli i izlazne varijable, a u nastavku ćemo prikazati nekoliko zanimljivih scatter dijagrama.



Slika 20: Scatter dijagram za atribute power-generated i distance-to-solar-noon

Na gornjem dijagramu možemo vidjeti suodnos između varijabli power-generated i distance-to-solar-noon. Važno je napomenuti da gornji dijagram sadržava prikaz 1000 slučajnih instanci odabrani iz cjelokupnog skupa podataka. Možemo vidjeti liniju regresije koja poprima oblik logaritamske funkcije. A na grafu je jasno reprezentirano da što je udaljenost od solarnog podneva u radijanima veća, produkcija energije u elektrani će biti manja.



Slika 21: Scatter dijagram za atribute power-generated i humidity

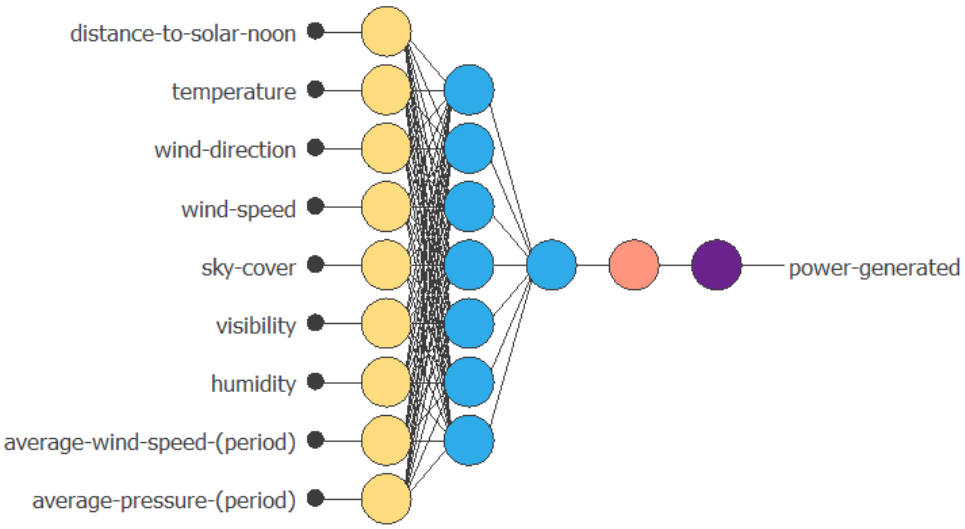
Na gornjem dijagramu možemo vidjeti suodnos između varijabli power-generated i humidity. Možemo vidjeti liniju regresije koja ovaj puta poprima oblik llinearne funkcije. Naravno,

produkcija električne energije će biti manja što je vlažnost veća, budući da kapljice vode u atmosferi sprečavaju dopiranje svjetlosti do kolektora u određenoj mjeri.

Nakon upoznavanja sa setom podataka, te podrobnijom analizom istog, možemo reći da je ovaj set podataka relativno dobra podloga za treniranje neuronske mreže. U nastavku se bavimo samom arhitekturom neuronske mreže koju ćemo koristiti u implementaciji algoritama.

4.2. Arhitektura neuronske mreže

Naravno, arhitektura neuronske mreže uvelike ovisi o problemskoj domeni za koju modeliramo neuronsku mrežu, ali i specifičnosti podataka s kojima će mreža raditi. U nastavku ćemo proći slojeve, brojeve i vrste neurona unutar naše mreže te objasniti osnovne parametre koji će biti fiksni za sve algoritme učenja. Sve implementiramo unutar alata Neural Designer. [31]



Slika 22: Arhitektura neuronske mreže korištene

Na gornjoj slici možemo vidjeti arhitekturu naše neuronske mreže za koju ćemo implementirati različite algoritme učenja. Vidljivo je da se mreža sastoji od pet različitih slojeva prikazanih žutom, plavom, crvenom i ljubičastom bojom. Žuti sloj se naziva sloj skaliranja i sadrži 9 neurona, prvi plavi sloj se naziva sloj perceptrona i sadrži 7 neurona, drugi plavi sloj je također perceptron sloj s jednim neuronom, crveni sloj se naziva sloj deskaliranja. Posljednji sloj zovemo povezničkim (eng. *bounding*) slojem. U nastavku ćemo zasebno objasniti svaki pojedinačni sloj.

4.2.1. Sloj skaliranja

Iako ga u teoretskom djelu nismo spomenuli, u praksi je dobro koristiti ovakav sloj. Naime, uloga sloja skaliranja je pobrinuti se da svi ulazi budu normalizirani u nekom fer rasponu za sve vrijednosti. Možemo gledati na sloj skaliranja kao sloj normalizacije ulaznih podataka jer u kontekstu neuronskih mreža on to i jest. U detalje metoda kojima provodimo skaliranje nećemo ulaziti, ali je važno znati da ovaj sloj koristi osnovne statističke vrijednosti ulaza u svome „radu“. Budući da se u praksi koriste metode skaliranja poput minimum i maksimum metode, skaliranja putem prosjeka i standardne devijacije jasno je zašto nam navedene vrijednosti trebaju. Naš sloj koristi metodu skaliranja prema prosjeku i standardnoj devijaciji koja je dana formulom: $skaliraniUlaz = \frac{ulaz - prosjekUlaza}{standardnaDevijacija}$.[30]

4.2.2. Prvi sloj perceptrona

Najvažnija vrsta sloja u neuronskoj mreži, itekako dobro obrađena u početnim poglavljima teoretskog uvoda. Slojevi perceptrona se također nazivaju i gustim slojevima budući da oni omogućavaju neuronskoj mreži da uči. O detaljima perceptrona, odnosno umjetnog neurona smo rekli već jako puno. Što se tiče konkretne implementacije neuronske mreže važno je spomenuti da neuroni u prvom sloju perceptrona koriste tanH aktivacijsku funkciju.

4.2.3. Drugi sloj perceptrona

Drugi sloj perceptrona je nešto drugačiji, vidimo da ima samo jedan neuron, pritom je njegova aktivacijska funkcija linearna. Iako koristimo više-manje standardne postavke u programu Neural Designer, uobičajeno je da neuronske mreže imaju više slojeva. Gdje je svaki pojedini sloj zadužen za određenu zadaću, odnosno komadić ukupnog problema.

4.2.4. Sloj deskaliranja

Logično je da se ovaj sloj bavi vraćanjem skaliranih vrijednosti u originalne, nešto poput inverzne funkcije. Također, postoje slične metode za deskaliranja skupa podataka kao što postoje i za skaliranje. Budući da za skaliranje koristimo metodu skaliranja prema prosjeku i standardnoj devijaciji, za deskaliranje idemo istim koracima. Metoda za deskaliranje po prosjeku i standardnoj devijaciji glasi: $deskaliraniIzlaz = prosjek + skaliraniIzlaz * standardna\ devijacija$. [30]

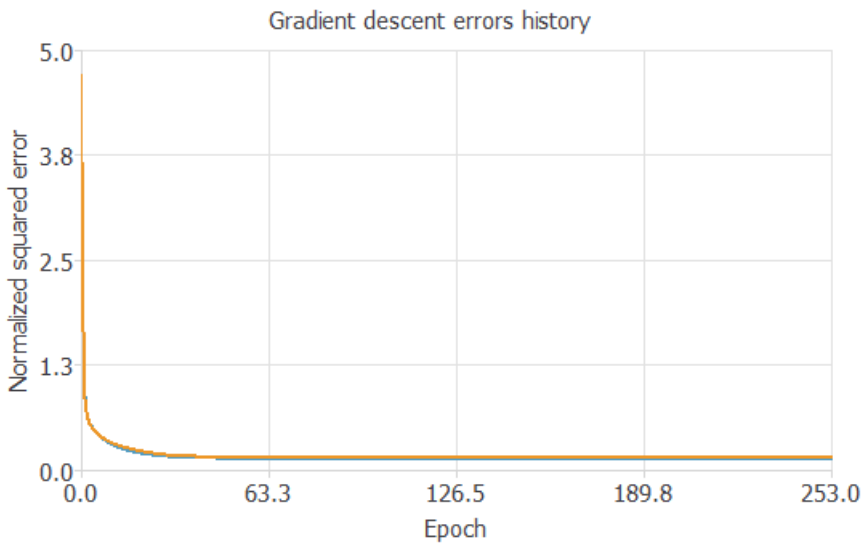
4.2.5. Poveznički sloj

Poveznički sloj je zadužen za transkribiranje svojih ulaza u vrijednosti koje se nalaze unutar zadanog intervala. Primjerice, ukoliko imamo zadovoljstvo korisnika nekom uslugom izraženo mjernom skalom od 1 do 5. U suštini ovaj sloj preslikava ulazne vrijednosti u zadani interval vrijednosti koje želimo dobiti. [30]

Nadalje, važno je reći da ćemo za izračun funkcije koštanja koristiti metodu Normalizirane kvadratne pogreške NSE. Također, za regularizaciju o kojoj smo pričali ranije ćemo koristiti metodu L2. Ovime završavamo pregled arhitekture neuronske mreže i prelazimo na rezultate algoritama učenja.

4.3. Rezultati algoritma gradijentnog spusta

Za početak ćemo se osvrnuti na općenite podatke o treniranju provedenom metodom gradijentnog spusta. Konkretno, prikazat ćemo dijagram funkcije koštanja i tablicu rezultata za korišteni algoritam.



Slika 23: Prikaz smanjenja funkcije koštanja

Na navedenom dijagramu reprezentirana je vrijednost funkcije koštanja (kao realni broj) na ordinati, dok se na apscisi nalazi broj epoha ili jednostavno iteracija koje je algoritam prošao u procesu treniranja. Možemo vidjeti da se povećanjem broja treninga ukupni iznos funkcije koštanja smanjuje, što i želimo vidjeti. Naime, naš je cilj minimizirati vrijednost funkcije koštanja i to je dobar rezultat. Iz grafa možemo iščitati da je negdje oko 30. epohe ili iteracije algoritam postigao vrijednost blisku minimumu. Također, na grafu je vidljiva narančasta linija koja predstavlja funkciju koštanja za selekcijski ili produkcijski set podataka, a plava za trening set podataka. Uglavnom, radi se o tome da narančasta linija pokazuje koliko je naša mreža dobra van treninga (simulacija realnih primjera). Možemo vidjeti da je podjednako dobra u oba slučaja.

	Vrijednost
Trening greška	0.13
Selekcijska greška	0.146
Broj epoha	253
Proteklo vrijeme	00:00:00
Kriterij zaustavljanja	Porast maksimalne selekcijske greške

Slika 24: Rezultati gradijentnog spusta

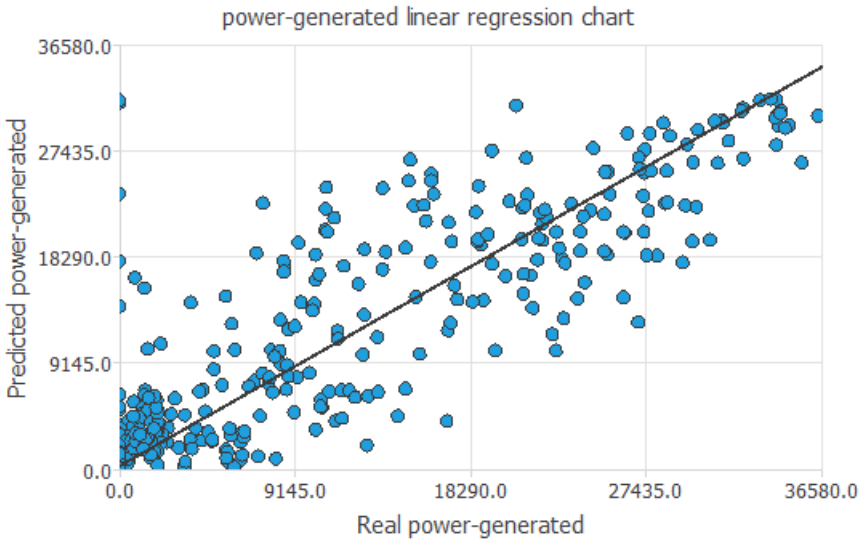
Gornja slika prikazuje podatke iz prethodnog dijagrama, a ovdje je najvažnija vrijednost selekcijske greške. Naime, budući da koristimo funkciju koštanja normalizirane kvadratne pogreške, vrijednost selekcijske greške od 1 bi značila da naša neuronska mreža daje prosječne vrijednosti za izlaz (garbage out). Dok broj bliži 0 znači da naša NM može dati precizniji izlaz za dane ulaze (realnije preslikavanje stvarne domene). Možemo vidjeti da je s ovaj set podataka računalno lagan budući da je vrijeme provedeno u simulaciji 0. Također, kriterij zaustavljanja algoritma je točka u kojoj se selekcijska greška počinje povećavati.

4.3.1. Analiza pogrešaka

O analizi pogrešaka smo s teoretskog aspekta rekli nekoliko rečenica, a sada je vrijeme da u praksi analiziramo rezultate algoritma gradijentnog spusta. Prvo ćemo se osvrnuti na analizu linearne regresije, zatim ćemo se osvrnuti na statistiku pogrešaka uz pripadajući histogram.

4.3.1.1. Analiza linearne regresije

Analiza linearne regresije je jedna od najvažnijih metoda kod analize pogrešaka neuronske mreže. Najbolje ćemo to objasniti na primjeru prikazanom sljedećom slikom.



Slika 25: Analiza linearne regresije

Analizom linearne regresije stječemo uvid u pouzdanost našeg modela, odnosno vidimo koliko je naš model precizan u predviđanjima s obzirom na prave podatke. Ovakva analiza rezultira trima parametrima za svaku izlaznu varijablu, pritom prvi parametar odgovara odsječku na osi y, a drugi parametar nagibu pravca. Treći parametar je $\boldsymbol{R^2}$ ili **koeficijent korelacije** o kojem smo pričali ranije. Za konkretni primjer koeficijent korelacije iznosi 0.8980, što je vrlo dobro, budući da vrijednost 1 označava stopostotnu točnost predikcije modela.

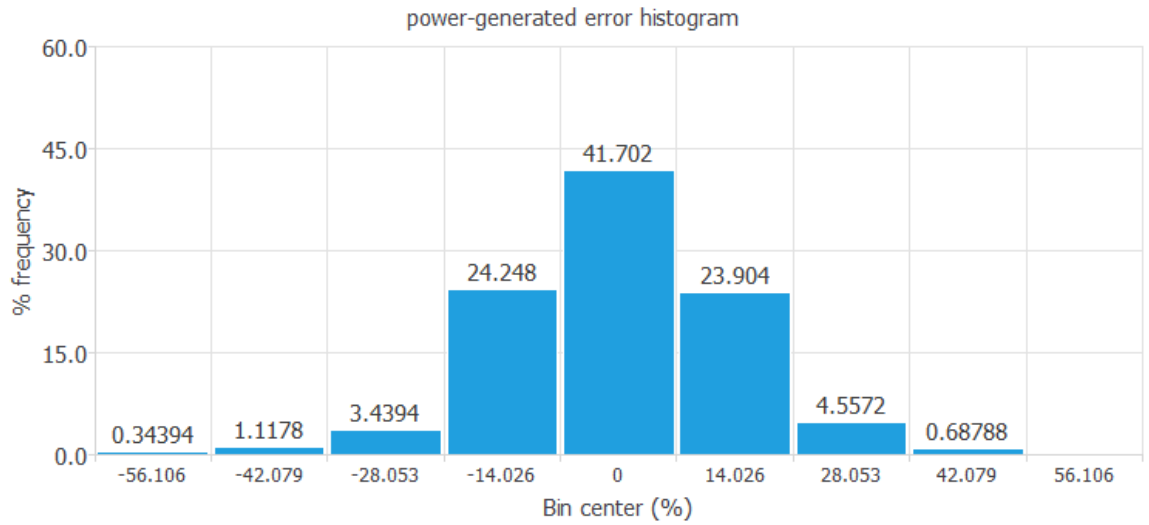
4.3.1.2. Statistika grešaka

Statistikom pogrešaka dobivamo uvid u kvalitetu našeg modela, konkretno mjerimo minimume, maksimume, prosjeke i standardne devijacije relativnih, apsolutnih i postotnih grešaka.

	Minimum	Maksimum	Prosjek	Devijacija
Apsolutna greška	13.022	31796.3	2636.68	3591.45
Relativna greška	0.000355986	0.869226	0.0720797	0.0981807
Postotna greška	0.0355986	86.9226	7.20797	9.81807

Slika 26: Statistika grešaka

Najvažnija vrijednost u prikazanoj tablici je prosječna postotna greška, tzv. Mean Percentage Error koja nam govori koliko naša neuronska mreža griješi u svojim predviđanjima. Možemo vidjeti da je za ovaj primjer prosječna postotna pogreška 7.2% što znači da od 100 pokušaja naša će mreža pogriješiti u predikciji njih 7. Ovo je iznimno dobar rezultat, a ako ga želimo pogledati s vedrije strane možemo reći da naša mreža predviđa s 92.8% točnosti.

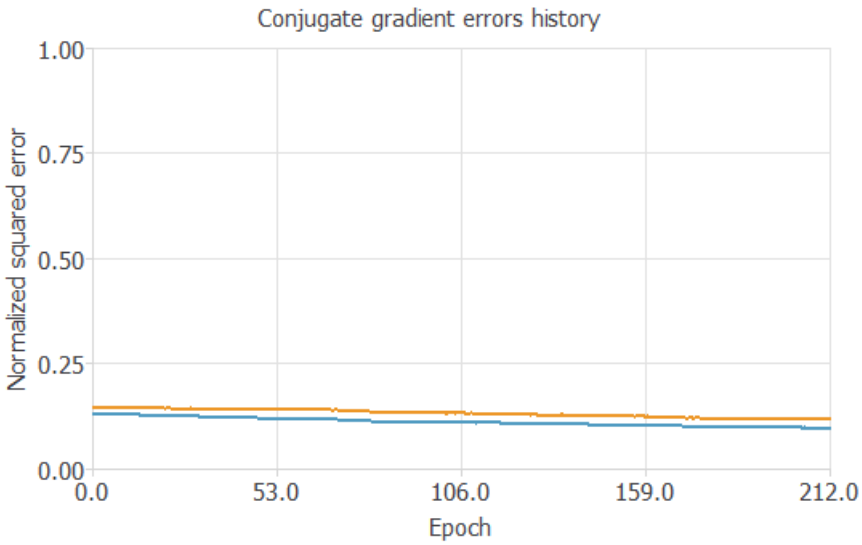


Slika 27: Histogram grešaka

Na gornjem histogramu možemo vidjeti da je najfrekventniji iznos pogreške 0, što je vrlo dobro, a normalna distribucija je očekivana u ovom slučaju. Generalno, što je frekventnost pogreške iznosa 0 veća to je mreža bolja u predviđanju.

4.4. Rezultati algoritma konjugiranog gradijentnog spusta

Rezultati čarobnog algoritma koji predstavlja dostojnu zamjenu za algoritme gradijentnog spusta drugog reda uistinu impresionira svojim rezultatima s obzirom na uobičajeni algoritam gradijentnog spusta što ćemo vidjeti temeljem analize u nastavku.



Slika 28: Prikaz smanjenja funkcije koštanja za CG algoritam

Na dijagramu prikaza smanjenja funkcije koštanja jasno je vidljivo da algoritam konjugiranog gradijenta postiže bržu konvergenciju ka lokalnom minimumu. Naime, ovaj algoritam stiže u lokalni minimum u samo 212 epoha, odnosno iteracija algoritma. Što je bolji rezultat od prethodnog algoritma uobičajenog gradijentnog spusta. Također, vidljivo je da je konvergencija ka minimumu ovdje izražena funkcijom nalik linearnoj i to vrlo blagog nagiba, što znači da je početna aproksimacija ovog algoritma vrlo dobra. Nadalje, na grafu možemo vidjeti da je početna vrijednost NSE-a u susjedstvu 0.12 što je višestruko bolje od prethodnog algoritma.

	Vrijednost
Trening greška	0.0972
Selekcijska greška	0.118
Broj epoha	212
Proteklo vrijeme	00:00:00
Kriterij zaustavljanja	Porast maksimalne selekcijske greške

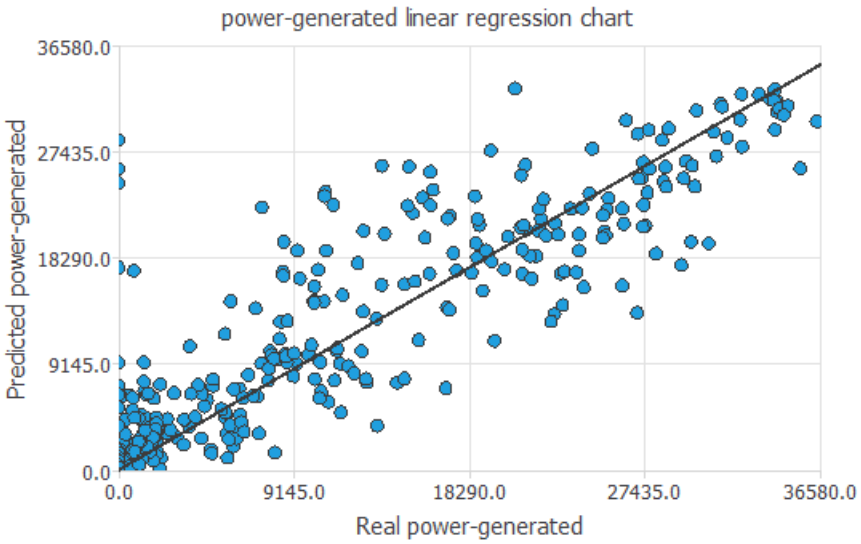
Slika 29: Rezultati konjugiranog gradijentnog spusta

Ponovno, najvažnija vrijednost u tabličnom prikazu je selekcijska greška koja u našem slučaju iznosi 0.118 što je opet bolje nego kod prethodnog algoritma, također u manjem broju epoha završavamo proces treninga što se podudara sa svojstvom algoritma konjugiranog gradijentnog spusta da brže konvergira u minimum funkcije koštanja.

4.4.1. Analiza pogrešaka

Kao i do sad, proći ćemo analizu linearne regresije uz pregled statistike grešaka te pripadni histogram za algoritam konjugiranog gradijentnog spusta.

4.4.1.1. Analiza linearne regresije



Slika 30: Analiza linearne regresije konjugiranog gradijentnog spusta

Analizom linearne regresije konjugiranog gradijentnog spusta možemo zaključiti da je ovaj model za određenu mjeru bolji od uobičajenog algoritma gradijentnog spusta. Naime, je $\boldsymbol{R^2}$ ili **koeficijent korelacije** za konjugirani gradijentni spust iznosi 0.9180, što je bolje za otprilike jedan postotni poen od prethodnog algoritma.

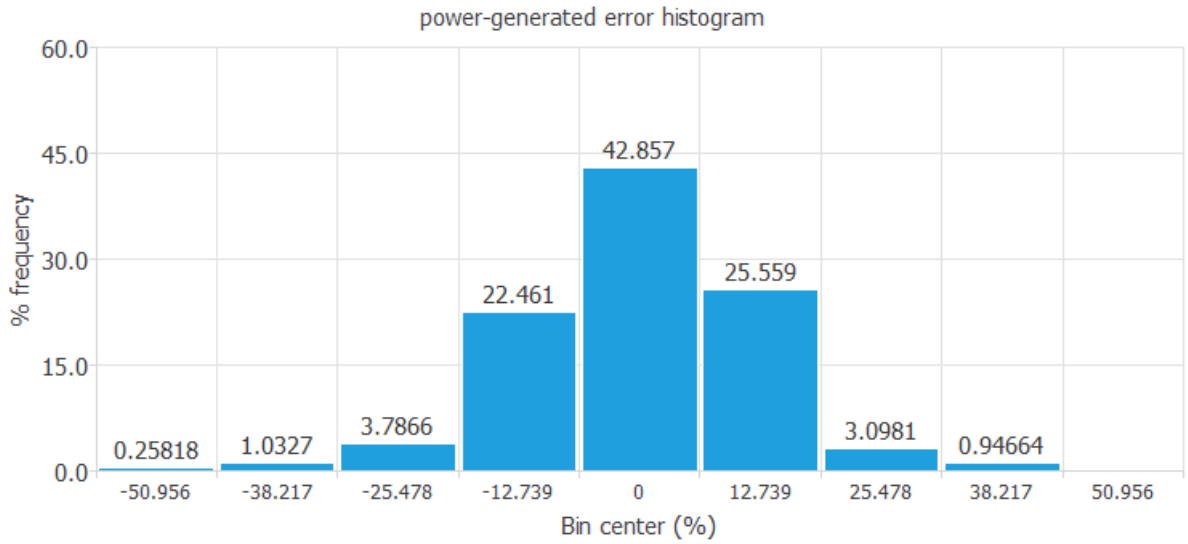
4.4.1.2. Statistika grešaka

Statistikom pogrešaka dobivamo uvid u kvalitetu našeg modela, a već smo ranije spomenuli koje sve statističke mjere uzimamo u obzir kod evaluacije. I kod algoritma konjugiranog gradijentnog spusta priča je istovjetna.

	Minimum	Maksimum	Prosjek	Devijacija
Apsoultna greška	5.92969	28419.3	2332.03	3266.34
Relativna greška	0.000162102	0.776909	0.0637515	0.0892932
Postotna greška	0.0162102	77.6909	6.37514	8.92932

Slika 31: Statistika grešaka za konjugirani gradijentni spust

Dakle, najvažnija vrijednost unutar danog tabličnog prikaza je ponovno prosječna postotna greška za konjugirani gradijentni spust. Ona u ovom konkretnom slučaju iznosi 6.37%, što znači da je ovaj algoritam omogućio modelu manji stupanj pogreške. Preciznije, algoritam konjugiranog gradijentnog spusta poboljšao je (smanjio) postotak prosječne pogreške za skoro jedan postotni poen naspram algoritma gradijentnog spusta. Drugim riječima, mreža trenirana algoritmom konjugiranog gradijentnog spusta ima točnost od 93.63%., što je ponovno poboljšanje s obzirom na prethodni algoritam.



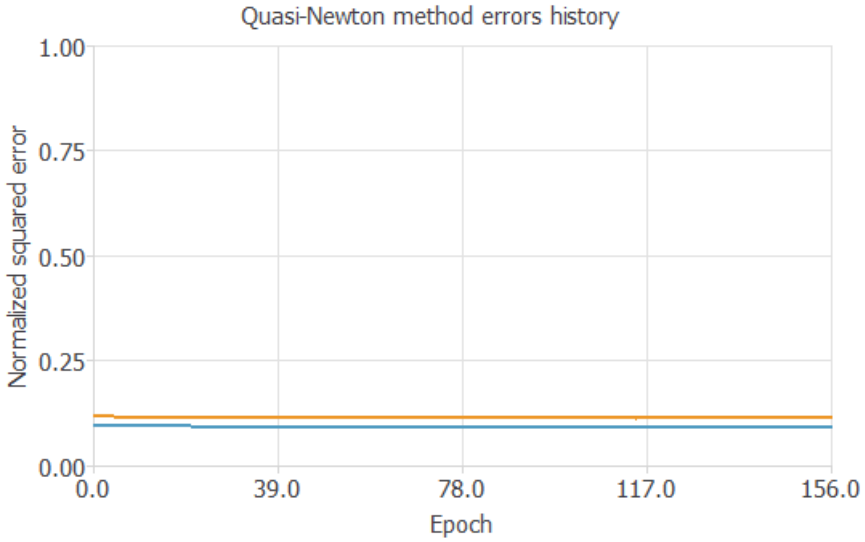
Slika 32: Histogram grešaka konjugiranog gradijentnog spusta

Na gornjem histogramu možemo vidjeti da je najfrekventniji iznos pogreške 0, a to je dobro. Iako razlika u frekventnosti greške s iznosom 0 nije značajno povećana s obzirom na prethodni

algoritam, poboljšanje je ipak vidljivo. Točnije, radi se o poboljšanju od 1.155% naspram uobičajenog gradijentnog spusta.

4.5. Rezultati Quasi-Newton metode

Kako utječe aproksimacija inverzne Hessianove matrice na performanse neuronske mreže vidjeti ćemo u nastavku. U pravilu očekujemo ponovno poboljšanje rezultata kroz sve parametre za koje vršimo mjerenje.



Slika 33: Prikaz smanjenja funkcije koštanja za QN algoritam

Na dijagramu prikaza smanjenja funkcije koštanja uočavamo još veću linearnost trenda smanjenja funkcije koštanja. To nam govori da je početna aproksimacija minimuma funkcije koštanja Quasi-Newton metode još bliža konkretnom minimumu nego prethodne dvije funkcije. Također, možemo vidjeti da je NSE vrijednost i dalje u susjedstvu 0.12, što znači da su i

prethodne aproksimacije točne, kao i ova, ali da niti jedan od algoritama nije zapeo u nekom lokalnom minimumu.

	Vrijednost
Trening greška	0.0914
Selekcijska greška	0.114
Broj epoha	156
Proteklo vrijeme	00:00:00
Kriterij zaustavljanja	Smanjenje minimalnog gubitka

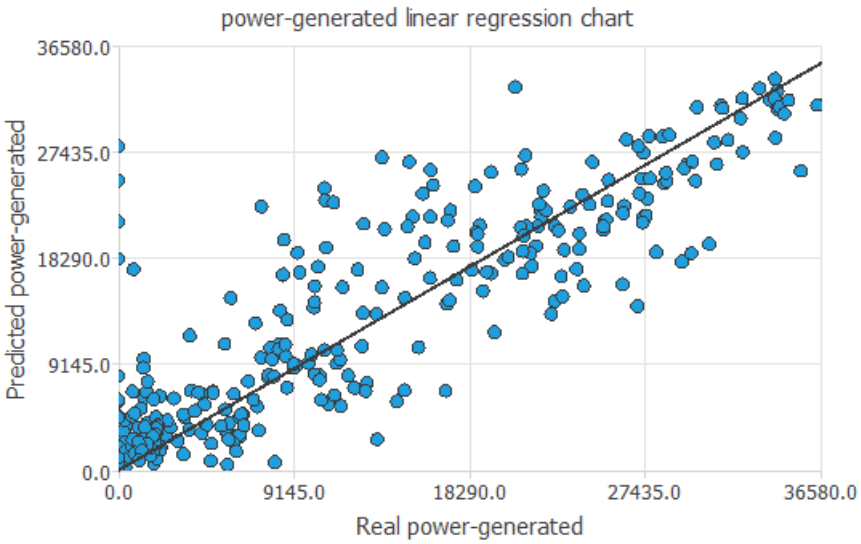
Slika 34: Rezultati Quasi-Newton metode

Selekcijska greška kod Quasi-Newton metode iznosi 0.114, što kao prvo ne iznenađuje. Budući da prema svim ranije navedenim karakteristika ovog algoritma očekujemo bolje performanse s obzirom na „simuliranje“ metoda drugog reda. Iako je poboljšanje skoro pa zanemarivo, moramo naglasiti da je ovo relativno maleni skup podataka s manjim brojem ulaza. Pretpostavka je da se vrijednosti poboljšanja za primjenu na kompleksnijim setovima podatka višestruko skaliraju. Također, broj epoha potreban za treniranje je drastično smanjen naspram algoritma gradijentnog i konjugiranog gradijentnog spusta.

4.5.1. Analiza pogrešaka

Kao i do sad, proći ćemo analizu linearne regresije za Quasi-Newton algoritam učenja uz pripadni pregled statistike grešaka te histogram.

4.5.1.1. Analiza linearne regresije



Slika 35: Analiza linearne regresije Quasi-Newton metode

Analizom linearne regresije konjugiranog Quasi-Newton algoritma dobivamo je $\boldsymbol{R^2}$ ili **koeficijent korelacije** iznosa 0.9215, što je ponovno poboljšanje s obzirom na prethodno korištene algoritme učenja.

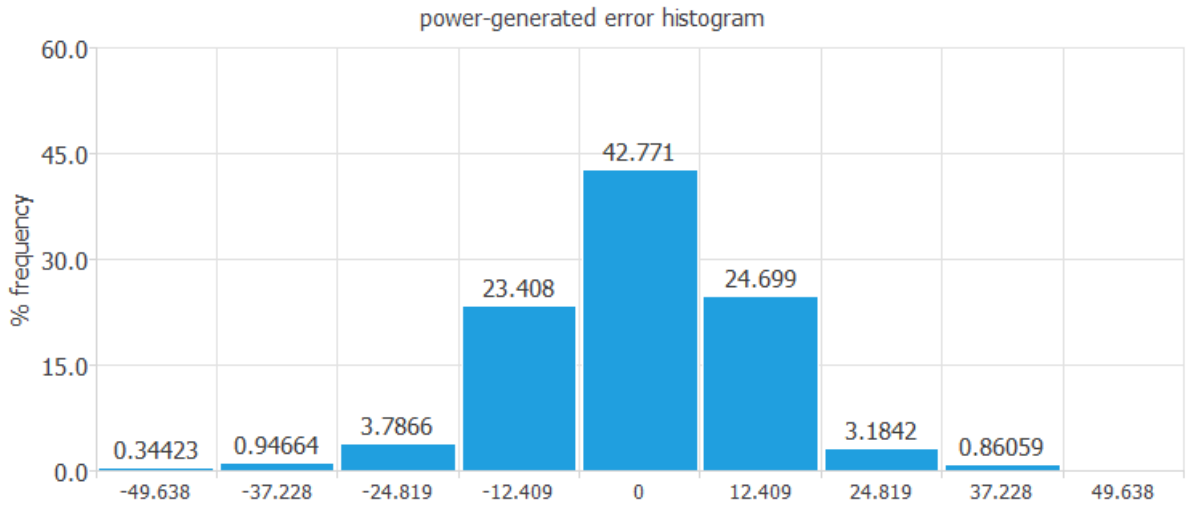
4.5.1.2. Statistika grešaka

Što se tiče kvalitete modela kojeg smo dobili uz pomoć Quasi-Newton algoritma koristit ćemo već nam znanu statistiku grešaka.

	Minimum	Maksimum	Prosjek	Devijacija
Apsolutna greška	0.669922	27929.7	2229.75	3232.86
Relativna greška	1.83139e-5	0.763524	0.0609554	0.0883777
Postotna greška	0.00183139	76.3524	6.09554	8.83776

Slika 36: Statistika grešaka za Quasi-Newton metodu

Ponovno, centralna vrijednost kod statistike grešaka koju gledamo je prosječna postotna greška našeg modela. U ovom slučaju, korištenjem Quasi-Newton metode učenja postigli smo poboljšanje od 0.28%, odnosno prosječna postotna greška sada iznosi 6.09% naspram 6.37% iz konjugiranog gradijentnog spusta. Odnosno, naš model neuronske mreže treniran dotičnom metodom postiže točnost od 93.91%.

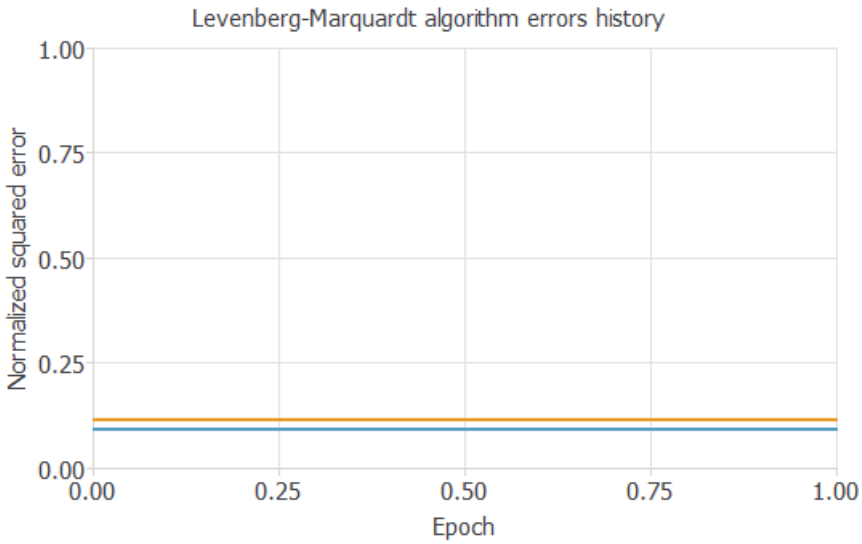


Slika 37: Histogram grešaka Quasi-Newton metode

Ponovno, imamo normalnu distribuciju što je indikator da je algoritam treniranja uspješno minimizirao funkciju koštanja. Najfrekventnija vrijednost je ponovno 0, što je dobro, ali imamo malo smanjenje frekventnosti s obzirom na konjugirani gradijentni spust. Međutim, kod ovog histograma je važna distribucija, dok frekventnost i nema nekog utjecaja na performanse. Rečeno možemo argumentirati preostalim rezultatima koji su bolji za Quasi-Newton metodu naspram ostalih algoritama učenja.

4.6. Rezultati Levenberg-Marquardt algoritma

Posljednji u nizu algoritama učenja koje uspoređujemo je Levenberg-Marquardt algoritam. Za ovaj algoritam možemo očekivati minorna poboljšanja s obzirom na Quasi-Newton metodu budući da se oba algoritma koriste pametnim načinima imitiranja algoritama drugog reda. Jedino područje u kojem možemo očekivati značajnija poboljšanja jesu greške, odnosno trebalo bi ih biti manje.



Slika 38: Prikaz smanjenja funkcije koštanja za LM algoritam

Što nam odmah upada u oči je **dramatično poboljšanje konvergencije** ka lokalnom minimumu. Naime, Levenberg-Marquardt algoritam je odmah na početku u potpunosti dobro aproksimirao inicijalnu minimalnu vrijednost funkcije koštanja. Drugim riječima, naš model je prošao kroz samo jednu iteraciju ili epohu, što je nevjerojatno!

	Vrijednost
Trening greška	0.0914
Selekcijska greška	0.114
Broj epoha	1
Proteklo vrijeme	00:00:00
Kriterij zaustavljanja	Smanjenje minimalnog gubitka

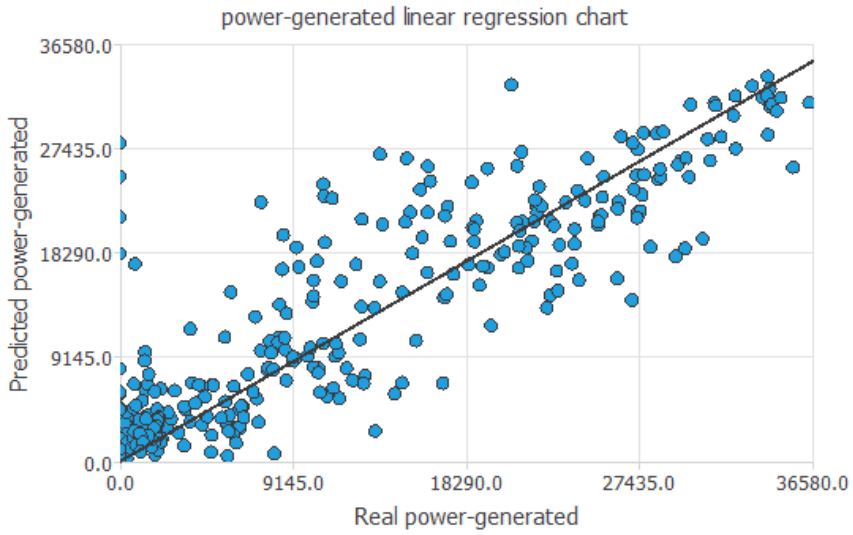
Slika 39: Rezultati Levenberg-Marquardt algoritma

Selekcijska greška kod Levenberg-Marquardt algoritma iznosi 0.114, istovjetno kao i za Quasi-Newton metodu. Ovdje ne očekujemo neko veliko poboljšanje, kao što smo već rekli, ali zapanjujuće malen broj epoha ili iteracija kroz koje je Levenberg-Marquardt algoritam prošao ovdje je i brojčano prikazan. Ustvari, broj iteracija predstavlja brzinu treniranja algoritma i što je taj broj manji algoritam treniranja je brži. Dakle, Levenberg-Marquardt algoritam je definitivno najbrži algoritam koji smo testirali.

4.6.1. Analiza pogrešaka

Kao i do sad, proći ćemo analizu linearne regresije za Levenberg-Marquardt algoritam učenja, a posebno pažnju ćemo posvetiti statistici grešaka budući da tamo očekujemo moguća poboljšanja.

4.6.1.1. Analiza linearne regresije



Slika 40: Analiza linearne regresije Levenberg-Marquardt algoritma

Rezultati analize linearne regresije za Levenberg-Marquardt algoritam su gotovo istovjetni rezultatima koje smo dobili korištenjem Quasi-Newtom metode. Dakle, **koeficijent korelacije** i za Levenberg-Marquardt algoritam iznosi 0.9215. Ovi rezultati ne iznenađuju budući da su poboljšanja dotičnog algoritma vidljiva u ekstremnoj brzini konvergencije ka lokalnom minimumu. U nastavku preostaje vidjeti imamo li poboljšanja u statistici grešaka.

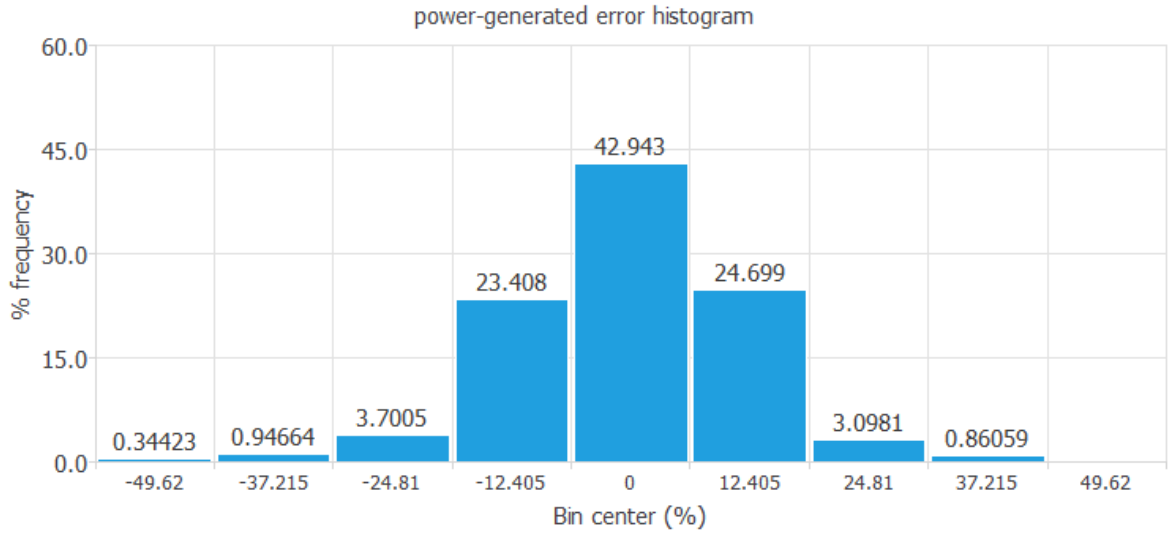
4.6.1.2. Statistika grešaka

Ne možemo očekivati drastično poboljšanje kvalitete modela, ali korištenjem Levenberg-Marquardt algoritma bi u teoriji trebali vidjeti poboljšanje s obzirom na prethodne algoritme učenja.

	Minimum	Maksimum	Prosjek	Devijacija
Apsolutna greška	0.626953	27916.3	2229.5	3232.3
Relativna greška	1.71392e-5	0.763158	0.0609486	0.0883624
Postotna greška	0.00171392	76.3158	6.09486	8.83624

Slika 41: Statistika grešaka za Levenberg-marquardt algoritam

Budući da želimo globalni dojam o prosječnim greškama kroz sve parametre, jer ovaj algoritam bi teorijski to trebao pokazati, promatrat ćemo kompletni stupac prosječnih vrijednosti. Naravno, naglasak je i dalje na prosječnoj postotnoj pogrešci koja je u ovom slučaju zanemarivo bolja. Nadalje, možemo vidjeti poboljšanje kod prosječne apsolutne pogreške za 0.25, što i nije tako zanemarivo s obzirom da je poboljšanje prosječne greške vidljivo tek na 5. decimalnom mjestu. Kvaliteta modela je generalno ostala ista kao i kod Quasi-Newton metode.



Slika 42: Histogram grešaka Levenberg-Marquardt algoritma

Ponovno, imamo normalnu distribuciju, dakle algoritam treniranja je odradio dobar posao. Možemo zamijetiti vrlo malu preraspodjelu frekvencija distribucije, odnosno vidimo da je

Levenberg-Marquardt algoritam postigao neznatnu veću frekventnost vrijednosti grešaka izlaza naspram Quasi-Newton metode.

Naposljetku, možemo zaključiti da empirijski dobivene vrijednosti uz pomoć alata Neural Designer koreliraju s početno iznesenom teorijom. Svaki od implementiranih algoritama se ponašao u skladu sa svojim teoretskim pretpostavkama, čak je i Levenberg-Marquardt algoritam iznenadio svojom brzinom učenja. Naravno, algoritme smo implementirali kako bi riješili problem predviđanja regresije, odnosno naša neuronska mreža je na temelju devet atributa uspjela aproksimirati generaciju električne energije od strane solarne elektrane (power-generated).

Preostaje pitanje, od svih algoritama učenja, koje je algoritme najbolje koristiti? Odgovor na to uvelike ovisi o performansama koje želimo od algoritama dobiti. Ukoliko želimo da naš algoritam bude brz u pronalasku minimuma funkcije koštanja, onda je dobro odabrati Levenberg-Marquardt algoritam. S druge strane, ukoliko želimo sačuvati računalnu snagu za druge zadatke, onda je bolje koristiti Quasi-Newton metodu. Dakle, sve ovisi o problemskoj domeni za koju razvijamo neuronsku mrežu, o našim željama ili potrebama. Ono najvažnije, odabir algoritma je odabir podatkovnog znanstvenika koji se treba temeljiti na stručnom znanju, iskustvu i dobroj praksi.

5. Zaključak

Neuronske mreže golicaju granice ljudske mašte još od 50.-ih godina 20. stoljeća i danas, možda više nego ikada prije, čine sastavni dio ljudske svakodnevice. Od biološkog neurona koji je služio kao osnova za razvoj mnoštva matematičkih modela koji preslikavaju principe funkcioniranja uma u računalo. Šarolikost algoritama, metoda i znanja vezanih uz neuronske mreže dugujemo ogromnoj zajednici akademika, ali i entuzijasta koji omogućavaju evoluciju neuronske paradigme i dalje. Štoviše, porastom računalne snage te eksplozijom podataka koju je omogućila globalna umreženost, čine iznimno dobar temelj za metode koje su namijenjene podacima.

Od izbora pojedinih aktivacijskih funkcija poput stepeničaste binarne ili hiperbolnog tangensa pa sve do odabira mrežne topologije, znanstvenici i entuzijasti danas imaju širom otvoren horizont. Veliki broj dostupne literature, online članaka te kompletnih softverskih rješenja uz pripadne video primjere te edukatora omogućava svima uvid u ljepotu kompleksnosti tematike neuronskih mreža. Da, čak je i izbor algoritama učenja danas velik, ali nećemo ulaziti u detalje izbora. Ovdje ćemo samo naglasiti da bez obzira koristimo li konjugirani gradijentni spust ili Levenberg-Marquardt algoritam u svojim mrežama, važno je sagledati cjelinu. Ono što neuronsku mrežu karakterizira jesu njene aktivacijske funkcije, topologija i algoritam učenja, ali moramo spomenuti i važnost podataka za koje mrežu dizajniramo. Naime, podaci su itekako bitni jer naposljetku o podacima ovisi kako će naša neuronska mreža „disati“. Upravo zato moramo posvetiti vrijeme podacima, njihovoj analizi i upoznavanju domene iz koje dolaze jer to su odlike koje čine kvalitetnog podatkovnog znanstvenika, akademika ili entuzijasta. Prisjetite se da neuronske mreže uče na temelju podataka, stoga za kvalitetno modeliranje uz pomoć neuronske mreže trebamo i kvalitetne podatke. Bez obzira radi li se o nadziranom, nenadziranom ili učenju s potporom, podaci ostaju preduvjet kvalitete modela. Naravno, kod sagledavanja cjeline trebamo obratiti pažnju i na rezultate, pritom ne mislim samo na standardne brojevne pokazatelje analize pogrešaka. Ovdje treba sagledati puno širu sliku, jer za stvarnu implementaciju neuronske mreže kao programskog rješenja nekog problema trebamo imati razumijevanje podataka koje modeliramo. Broj nam može biti indikator ili mjera točnosti, međutim ljudsko razumijevanje je ključno za uspješnu implementaciju neuronske mreže. To je razina inteligencije koju danas, nažalost, još uvijek ne možemo aproksimirati minimiziranjem funkcija.

S porastom digitalizacije svijeta, ali i uređaja koji generiraju podatke te dolaska novih i bržih načina umrežavanja dolazi doba gdje neuronske mreže i strojno učenje imaju svoje mjesto. U budućnosti gdje će hodanje po cesti biti apsolutno sigurno jer će svi automobili voziti cestom uz pomoć sustava dubokog učenja. U budućnosti gdje će neuronske mreže možda biti

temelj za stvaranje sustava predikcije s apsolutnom točnošću. U budućnosti gdje će ljudi shvatiti da se u simuliranju rada ljudskog mozga ne skriva zli robot koji želi apokalipsu svijeta, već se u njega skriva beskrajno prostranstvo mogućnosti napretka ljudske vrste. Naravno, na pojedincima, a možda čak i na svima nama, je da odlučimo u kojem ćemo smjeru razvijati tehnologiju i koji predznak ćemo joj dati.

Popis literature

[1] L. Fausett, *Fundamentals of Neural Networks: Architectures, Algorithms, and Applications*. USA: Prentice-Hall, Inc., 1994.

[2] B. Morgan, D. K. P. Staff, P. Smith, Z. Shalev, i D. Banerjee, *First Human Body Encyclopedia*. Dorling Kindersley Limited, 2016.

[3] C. A. Simpkins i A. M. Simpkins, „Neuroplasticity and Neurogenesis: Changing Moment-by-Moment“, u *Neuroscience for Clinicians: Evidence, Models, and Practice*, New York, NY: Springer New York, 2013, str. 165–174.

[4] B. Lantz, *Machine Learning with R - Third Edition: Expert Techniques for Predictive Modeling*. Packt Publishing, 2019.

[5] B. Szablowski, „What is an artificial neuron and why does it need an activation function?“, 2020. [Na internetu]. Dostupno: https://towardsdatascience.com/what-is-an-artificial-neuron-and-why-does-it-need-an-activation-function-5b4c1e971d80 [Pristupano 23.6.2021.]

[6] R. D. Reed i R. J. Marks, *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks*. Cambridge, MA, USA: MIT Press, 1998.

[7] J. D. Kelleher, „Deep Learning“. The MIT Press, 2019.

[8] P. S. Neelakanta, *Information-Theoretic Aspects of Neural Networks*. CRC-Press, 1999.

[9] G. Ognjanonvski, „Everything you need to know about Neural Networks and Backpropagation – Machine Learning Easy and Fun“, 2019. [Na internetu]. Dostupno: https://towardsdatascience.com/everything-you-need-to-know-about-neural-networks-and-backpropagation-machine-learning-made-easy-e5285bc2be3a [Pristupano 1.7.2021]

[10] A. Pai, „CNN vs. RNN vs. ANN – Analyzing 3 Types of Neural Network sin Deep learning“, 2020. [Članak na internetu]. Dostupno: https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-in-deep-learning/ [Pristupano 1.7.2021]

[11] N. S. Gill, „Artifical Neural Networks Applications and Algorithms“, 2021. [Članak na internetu]. Dostupno: https://www.xenonstack.com/blog/artificial-neural-network-applications [Pristupano 13.7.2021]

[12] M. Z. Mulla, „Cost, Activation, Loss Function|| Neural Network|| Deep Learning. What are these?“, 2020. [Blog post]. Dostupno:https://medium.com/@zeeshanmulla/cost-activation-loss-function-neural-network-deep-learning-what-are-these-91167825a4de [Pristupano 13.7.2021]

[13] Guru99 (bez dat.), „Unsupervised Machine Learning: What is, Algorithms, Example“ [Na internetu]. Dostupno: https://www.guru99.com/unsupervised-machine-learning.html [Pristupano 13.7.2021]

[14] K. Chandrakant, „Reinforcement Learning with Neural Network“, 2020. [Članak na internetu]. Dostupno: https://www.baeldung.com/cs/reinforcement-learning-neural-network [Pristupano 13.7.2021]

[15] V. Solegaonkar, „Error Analysis in Neural Netwroks“, 2019. [Članak na internetu]. Dostupno: https://towardsdatascience.com/error-analysis-in-neural-networks-6b0785858845 [Pristupano 13.7.2021]

[16] D. Oates, „Put garbage in, get garbage out: machine learning is only as powerful as the data it feeds on“, 2021. [Članak na internetu]. Dostupno: https://www.unissu.com/proptech-resources/Put-garbage-in-get-garbage-out-machine-learning-is-only-as-powerful-as-the-data-it-feeds-on [Pristupano 13.7.2021]

[17] F. Malik, „Understanding Neural Network Neurons“, 2019. [Članak na internetu]. Dostupno: https://medium.com/fintechexplained/understanding-neural-network-neurons-55e0ddfa87c6 [Pristupano 16.7.2021]

[18] F. Malik, „Neural Networks Bias And Weights“, 2019. [Članak na internetu]. Dostupno: https://medium.com/fintechexplained/neural-networks-bias-and-weights-10b53e6285da [Pristupano 16.7.2021]

[19] G. Sanderson, (16.10.2017) „Gradient descent, how neural networks learn | Chapter 2, Deep Learning“, Youtube [Video datoteka]. Dostupno: https://www.youtube.com/watch?v=IHZwWFHWa-w&list=PL_h2yd2CGtBHEKwEH5iqTZH85wLS-eUzv&index=2&ab_channel=3Blue1Brown [Pristupano 16.7.2021]

[20] D. J. Hand, P. Smyth, i H. Mannila, *Principles of Data Mining*. Cambridge, MA, USA: MIT Press, 2001.

[21] N. Sharma, „ Exploring Activation and Loss Functions in Machine Learning“, 2020. [Članak na internetu]. Dostupno: https://heartbeat.fritz.ai/exploring-activation-and-loss-functions-in-machine-learning-39d5cb3ba1fc [Pristupano 21.7.2021]

[22] K. Mahendru, „A Detailed Guide to 7 Loss Functions for Machine Learning Algorithms with Python Code“, 2019. [Članak na internetu]. Dostupno: https://www.analyticsvidhya.com/blog/2019/08/detailed-guide-7-loss-functions-machine-learning-python-code/ [Pristupano 21.7.2021]

[23] T. F. Awolusi, O. L. Oke, O. O. Akinkurolere, A. O. Sojobi, i O. G. Aluko, „Performance comparison of neural network training algorithms in the modeling properties of steel fiber reinforced concrete“, *Heliyon*, sv. 5, izd. 1, str. e01115, sij. 2019, Dostupno: https://www.sciencedirect.com/science/article/pii/S2405844018346036#sec2 .[Pristupano 21.7.2021]

[24] M. MADIĆ, G. RADENKOVIĆ i M. RADOVANOVIĆ, "Evaluation of ANN-BP and ANN-GA Models Performance in Predicting Mechanical Properties and Machinability of Cast Copper Alloys", *Strojarstvo*, vol.54, br. 2, str. 169-174, 2012. [Online]. Dostupno na: https://hrcak.srce.hr/93641 . [Pristupano: 5.08.2021.]

[25] T. Masters, „Advanced algorithms for neural networks“, Toronto, Canada: John Wiley & Sons. 1995.

[26] A. S. Walia, „Types of Optimization Algorithms in Neural Networks and Ways to Optimize Gradient Descent“, 2021. [Na internetu]. Dostupno: https://medium.com/nerd-for-tech/types-of-optimization-algorithms-used-in-neural-networks-and-ways-to-optimize-gradient-descent-1e32cdcbcf6c [Pristupano 10.8.2021]

[27] A. Quesada, (bez dat.), „5 algorithms to train a neural network“ [Na internetu]. Dostupno: https://www.neuraldesigner.com/blog/5_algorithms_to_train_a_neural_network [Pristupano: 13. 8.2021]

[28] „Gradient Descent“ (bez dat.). ML-CheatSheet [Na internetu]. Dostupno:https://ml-cheatsheet.readthedocs.io/en/latest/gradient_descent.html [Pristupano: 17.8.2021]

[29] „Training strategy“ (bez dat.). Neural Designer. [Na internetu].Dostupno: https://www.neuraldesigner.com/learning/tutorials/training-strategy [Pristupano: 18.8.2021.]

[30] „Neural network“ (bez dat.). Neural Designer [Na internetu]. Dostupno: https://www.neuraldesigner.com/learning/tutorials/neural-network [Pristupano:18.8.2021]

[31] Artificial Intelligence Techniques Ltd. Neural Designer (verzija 591) (2021) [Na i nternetu]. Dostupno: https://www.neuraldesigner.com/ [Pristupano 2.7.2021]

Popis slika