Primjena neuronskih mreža na probleme regresije i klasifikacije u strojnom učenju

Mišetić, Diego

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1439

PRIMJENA NEURONSKIH MREŽA NA PROBLEME REGRESIJE I KLASIFIKACIJE U STROJNOM UČENJU

Diego Mišetić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1439

PRIMJENA NEURONSKIH MREŽA NA PROBLEME REGRESIJE I KLASIFIKACIJE U STROJNOM UČENJU

Diego Mišetić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

ZAVRŠNI ZADATAK br. 1439

Pristupnik:	**Diego Mišetić (0036543343)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentorica:	izv. prof. dr. sc. Josipa Pina Milišić

Zadatak: **Primjena neuronskih mreža na probleme regresije i klasifikacije u strojnom učenju**

Opis zadatka:

Cilj ovog završnog rada je predstaviti korištenje neuronskih mreža u strojnom učenju. U uvodnom dijelu dat će se pregled osnovnih pojmova vezanih uz linearnu regresiju koja će se zatim generalizirati na dvoslojnu neuronsku mrežu, te dalje i na duboku neuronsku mrežu. Posebno će se istražiti mogućnosti treniranja i regularizacije dobivenih neuronskih mreža. Konačno, u radu će se prikazati primjena dubokih neuronskih mreža na problem detekcije transakcijskih prijevara s kreditnim karticama. Ilustrativni primjeri implementirat će se pomoću TensorFlow biblioteke.

Rok za predaju rada: 14. lipnja 2024.

Zahvaljujem se obitelji i mentorici izv. prof. dr. sc. Josipi Pini Miličić na podršci kroz rad

Sadržaj

1. Uvod

Još od davnih vremena, izumitelji i znanstvenici težili su stvaranju strojeva koje mogu razmišljati. Isto tako, stotinu godina prije izlaska prvog računala, ljudi su razmatrali mogućnost stvaranja inteligentnog stroja.

Na početku razvoja umjetne inteligencije, ovo područje se najprije dotaklo i riješilo neke vrlo zahtjevne i teške zadatke za ljude, ali jednostavne za računala zbog postojanja strogih formalnih pravila koja opisuju takve probleme, npr. igranje šaha (pobjeda IBM-og računala protiv Garryja Kasparova 1997. godine). Ali unatoč tome, postoje i druge stvari kojima su ljudima vrlo jednostavne, a računalima izuzetno teške, poput prepoznavanja lica.

Umjetna inteligencija sve više postaje sastavni dio našeg svakodnevnog života. Postoje mnoge primjene umjetne inteligencije u našim životima kao što su pametni asistenti (Siri i Alexa), autonomna vozila (poboljšava sustave pomoći vozačima i optimizira upravljanje prometom), sustavi za prepoznavanje lica, zdravstva skrb (UI se koristi za analiziranje medicinskih podataka i dijagnosticiranje bolesti), financijske usluge, pametni satovi, u svakom industrijskom sektoru te mnoge druge.

Čovjek je izložen velikom broju informacija iz svijeta koji nas okružuje, čime konstantno dobiva iskustvo i nova znanja te ima sposobnost subjektivnog i intuitivnog razmišljanja u svakakvim situacijama. Mnogo je pokušaja da se svijet oko nas opiše preciznim formalnim jezikom, ali ti pokušaji nisu davali zadovoljavajuće rezultate. Računalni sustavi u UI najčešće su sposobni raditi jednu ili nekoliko stvari vrlo dobro, ali im nedostaje sposobnost učenja. Dakle zaključak je da čovjek puno toga zna što je intuitivno, a računalo ne.

Strojno učenje je grana umjetne inteligencije te ga možemo opisati kao programiranje računala tako da optimiziraju neki kriterij uspješnosti na temelju podatkovnih primjera ili prethodnog iskustva. Strojno učenje široko se primjenjuje u raznim podru-

čjima obrade signala, računalstvu, detekciji prijevara, prepoznavanju slika itd. Također umjetne neuronske mreže i strojno učenje primjenjuje se rješavanje problema gdje na postoji ljudsko razumijevanje procesa, u sustavima koji se dinamički mijenjaju te koji koriste ogromne količine podataka. Tako dolazimo do umjetne neuronske mreže koja nastoji simulirati postupak učenja i obrade podataka (kao ljudski mozak). Neuronske mreže si vrsta strojnog učenja, a samim time i umjetne inteligencije. Neuronski se sustav razvija pomoću treninga na velikom broju primjera. Zajedničko svim neuronskim mrežama je primanje ulaznih informacija i pretvaranje u izlazne. Umjetne neuronske mreže koriste se za probleme klasifikacije i regresije. Kod klasifikacije pokušavamo pridodijeliti oznaku uzorcima na temelju oznake uzoraka, dok se kod regresije na temelju ulaza traži neki realan broj kojim će se zamijeniti ulaz.

Neuronske mreže imaju vrlo široku primjenu u raznim područjima kao što su ekonomija, tehničke znanosti, financije, biometrijskoj autentifikaciji te mnogim drugim. Najnoviji primjer primjene umjetnih neuronskih mreža je Wolfram Alpha, javno dostupna tražilica.

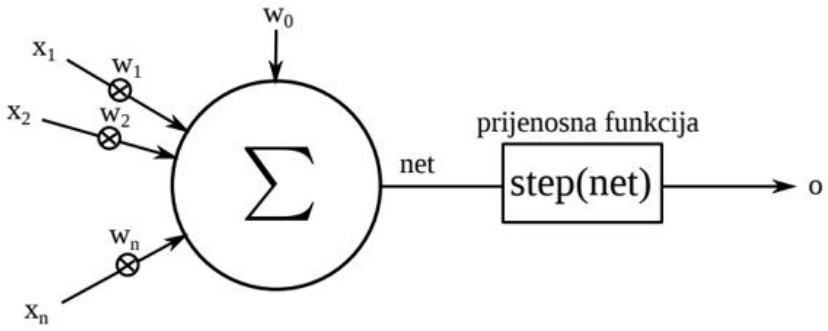
2. Glavni dio

2.1. Model neuronske mreže

Za nelinearne parametarske funkcije, koje su matematički model koji opisuje odnose između ulaza i izlaza na nelinearan način, modeliraju se veze između ulaznih varijabli $x_1, \dots x_p$ i izlaza y u obliku predviđanja označavanom kao na 2.1

$$y = f_\theta(x_1, \dots, x_p) \tag{2.1}$$

U izrazu (2.1) funkcija f je parametrizirana pomoću parametra θ. Nelinearna funkcija može biti parametrizirana na mnoge načine. Strategija u neuronskoj mreži je korištenje nekoliko slojeva modela linearne regresije i nelinearnih aktivacijskih funkcija. Model umjetne neuronske mreže sastoji se od umjetnih neurona. Umjetni neuron možemo definirati kao „jedinicu za obradu podataka“. Vrijednosti s ulaza x_i množimo s težinom tog ulaza w_i, ukupnoj sumi dodaje se pomak w_0. Te se tako definira vrijednost net.



Slika 2.1. Umjetni neuron

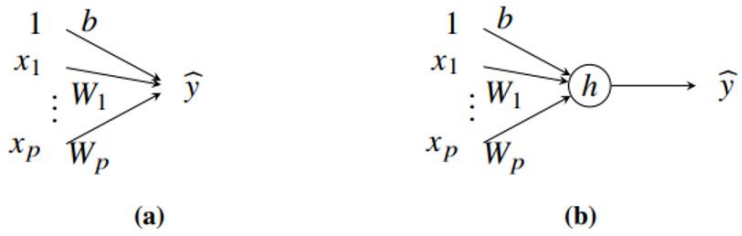
$$net = \sum_{i=1}^{n}(x_i \cdot w_i) + w_0 \tag{2.2}$$

2.1.1. Generalizirana linearna regresija

Opis neuronske mreže započinjemo linearnim regresijskim modelom.

$$y = W_1 x_1 + W_2 x_2 + \cdots + W_p x_p + b \tag{2.3}$$

Parametre težina označavamo s $W_1, \ldots W_p$ te termin za pomak s b. Varijable $x_1, \ldots, x_p$ su ulazne varijable . Grafički takav model možemo prikazati kao:



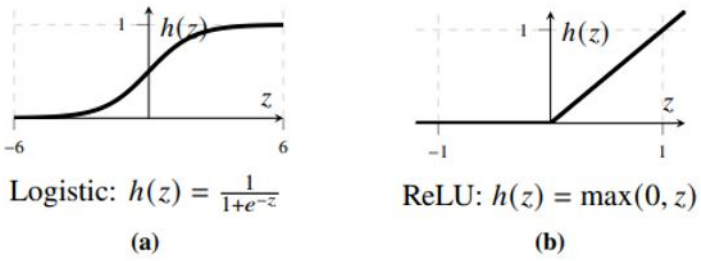
Slika 2.2. Generalizacija linearne regresije

Svaka ulazna varijabla x_i predstavljena je čvorom, a svaki parametar W_i vezom. Nadalje izlaz y je opisan kao zbroj svih članova $W_i x_i$. Za ulaznu varijablu pomaka b koristimo konstantu vrijednosti 1. Za opisivanje nelinearnih odnosa između ulaza i izlaza, uvodimo nelinearnu skalarnu funkciju koja se naziva aktivacijska funkcija. Aktivacijsku funkciju označit ćemo s $h : \mathbf{R} \to \mathbf{R}$. Model linearne regresija sada je modificiran u model generalizirane linearne regresije, gdje je linearna kombinacija ulaza transformirana pomoću aktivacijske funkcije.

$$y = h(W_1 x_1 + W_2 x_2 + \cdots + W_p x_p + b) \tag{2.4}$$

Ovakvo proširenje na linearnu regresiju vizualizirano je prikazano na slici 2.1.b. Uobičajeni izbor funkcije aktivacije su logistička funkcija ili funkcija ispravljanja (ReLU)

Na Slici 2.3. ilustrirane su logistička i ReLu aktivacija funkcija. Logistička funkcija je linearna oko područja gdje je z blizu ili jednak nuli, dok poprima vrijednost 0 ili 1 kako z raste ili otpada. ReLu funkcija je još jednostavnija. Funkcija je samo jednaka nuli ako je ulaz z negativan, a za pozitivne brojeve poprima vrijednost samog broja z. Dugi



Slika 2.3. Funkcije aktivacije

niz godina se za izbor aktivacijske funkcije u neuronskim mrežama koristila logistička funkcija, dok je sada standardni izbor u većini modela neuronskim mreža ReLU aktivacijska funkcija. Model generalizirane linearne regresije vrlo je jednostavan i sam po sebi nije sposoban opisivati vrlo kompliciranе odnose između ulaza i izlaza. Kako bismo povećali općenitost modela koriste se sljedeća proširenja: koristimo nekoliko paralelnih generaliziranih linearnih modela regresije za izgradnji sloja (to će dovesti do višeslojne neuronske mreže) te složiti slojeve u sekvencijski konstrukciju (to će dovesti do dubokih neuronskih mreža).

2.1.2. Dvoslojna neuronska mreža

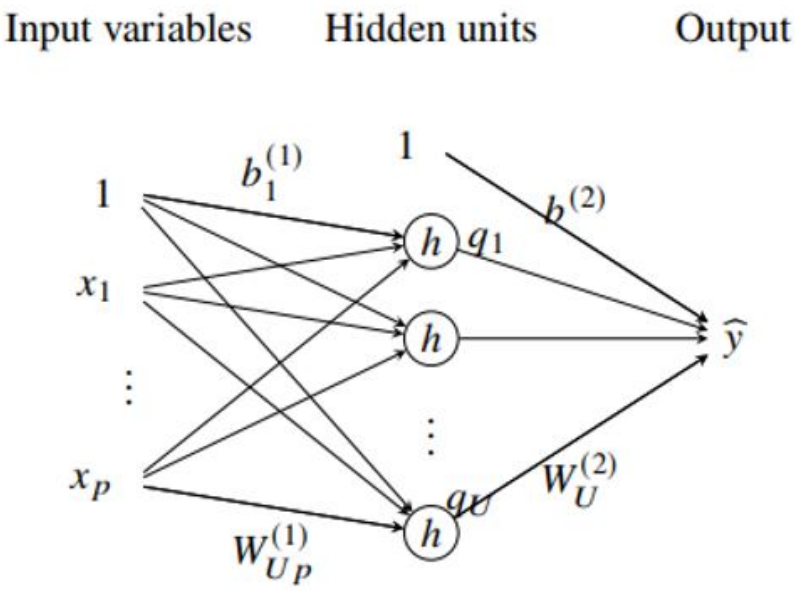
U formuli 1.1. izlaz y konstruiran je pomoću skalarnog regresijskog modela. Da bi se povećala njegova fleksibilnost i da ga pretvori u dvoslojnu neuronsku mrežu, njegov izlaz bit će zboj takvih generaliziranih modela linearne regresije , od kojih svaki ima svoj skup parametara. K-ti regresijski model opisat ćemo s b_k, W_{k1},...,W_{kp} , te izlaz označimo s q te dobivamo:

$$q_k = h(W_{k1}x_1 + W_{k2}x_2 + \cdots + W_{kp}x_{kp} + b), k = 1, \ldots U \tag{2.5}$$

Svaki q_k su takozvane skrivene jedinice, budući da nisu izlaz cijelog modela. U različite skrivene jedinice q_k, mogu djelovati kao ulazne varijable u dodatni model linearne regresije.

$$q_k = h(W_1q_1 + W_2q_2 + \cdots + W_Uq_U + b) \tag{2.6}$$

Ako proširimo grafički prikaz, model dvoslojne neuronske mreže možemo prikazati kao grafikon s dva sloja veza. Svaka veza ima parametar povezan s njom. Pomak se računa ne samo u ulaznom sloju nego i u skrivenom.



Slika 2.4. Dvoslojna neuronska mreža

2.1.3. Vektorizacija preko jednica

Model dvoslojne neuronske mreže također se može napisati pomoću matrične notacije, gdje su parametri u svakom sloju složeni u težinu matrice W i vektora pomaka b, te se cijeli prikaz može prikazati kao:

$$q = h(\mathbf{W}^1 x + b^1), \quad y = \mathbf{W}^2 q + b^2 \quad \mathbf{W}, x \in \mathbb{R} \tag{2.7}$$

U ovoj formuli smo komponente x i q složili kao vektore. Ovakva formula prikazuje nelinearni regresijski model. Aktivacijska funkcija h djeluje elementarno na ulaznom vektoru i rezultira izlaznim vektorom jednake dimenzije. Dvije težinske matrice i dva vektora pomaka su parametri modela te gdje operator već uzima sve elemente u matrici i stavlja ih u vektor.

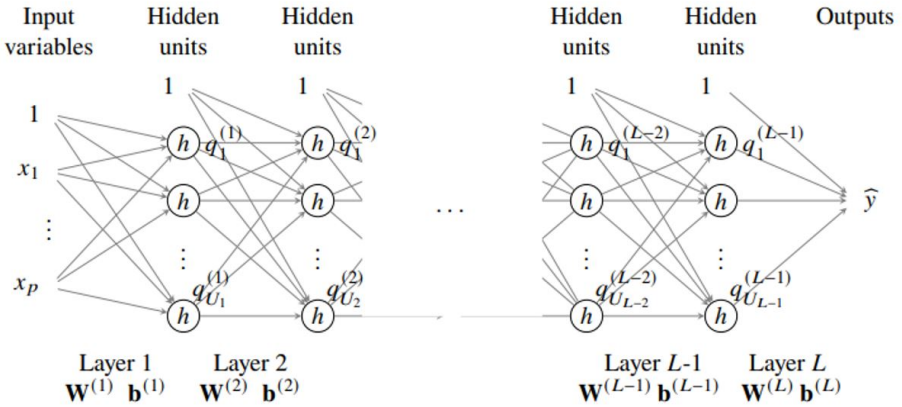
2.1.4. Duboke neuronske mreže

Dvoslojni neuronski mrežni model sam po sebi je koristan model, te je provedeno mnogo

istraživanja i analiza o njem. Međutim, prava opisana moć neuronske mreže ostvaraje se kada stavimo više takvih slojeva generaliziranih modela linearne regresije i time postižemo duboku neuronsku mrežu. Duboke neuronske mreže mogu modelirati složene odnose (popust onog između slike i njezine klase), te su jedna od najnaprednijih metoda u strojnom učenju današnjice. Slojeve numeriramo indexom l iz skupa $\{1, 2, \ldots, L\}$, gdje je L broj slojeva. Svaki sloj parametriziran je matricom težine $W^{(l)}$ i vektora pomaka $b^{(l)}$. Primjerice, $W^{(1)}$ i $b^{(1)}$ pripadaju sloju l=1. Također imamo više slojeva skrivenih jedinica označenih s $q^{(l)}$.

$$q^l = h(W^l q^{l-1} + b^l) \tag{2.8}$$

Svaki sloj preslikava skriveni sloj q^{l-1} do sljedećeg sloja q^l. To znači da su slojevi složeni tako da je izlaz prvog sloja skrivene jedinice q^1 ulaz u drugi sloj, itd. Slaganjem višestrukih slojeva konstruirali smo dubuku neuronsku mrežu. Grafički prikz duboke neuronske mreže:



Slika 2.5. Duboka neuronska mreža

2.1.5. Vektorizacija točkastih podataka

Tijekom treniranja modela neuronske mreže, mreža se koristi za izračun predviđenog ulaza, ne samo za jedna ulaz, nego za više ulaznih podataka. Kao što smo ranije objasnili vektorizaciju preko jedinica, također želimo vektorizirati ove jednadžbe preko podataka kako bismo omogućili učinkovitu izračun modela. Uz ovu notaciju, možemo, slično kao u modelu linearne regresije, slagati sve podatke u matrice, gdje svaki podatak predstavlja jedan redak.

$$\mathbf{Q} = h(\mathbf{X}\mathbf{W}^{(1)T} + b^{(1)T}), \quad y = \mathbf{Q}\mathbf{W}^{(2)T+b^{(2)T}} \tag{2.9}$$

Svaki sloj parametriziran je matricom težine $\mathbf{W}^{(l)}$ i vektora pomaka $b^{(l)}$, gdje l označava broj sloja. Ovo omogućuje da koristimo linearne algebarske operacije (množenje i zbrajanje) za izračun modela za sve podatke odjednom, čime se postiže učinkovitost i ubrzava proces treniranja i testiranja modela. Vektorizacija podataka pomaže u optimiziranju rada s velikim skupovima podataka i korištenju alata za ubrzavanje obrade. Ovakav pristup pomaže optimizirati izvedbu modela jer omogućuje rad s vektoriziranim podacima na učinkovit način, koristeći operacije na matricama. Time se smanjuje potreba za dodatnim transponiranjem i unaprjeđuje brzina treniranja i testiranja modela.

2.1.6. Neuronske mreže za klasifikaciju

Neuronske mreže se također mogu koristiti za klasifikaciju. Model je osmišljen tako da je izlaz vektor M. Ovdje se koristi dodatna funkcija aktivacije softmax koja djeluje na završni sloj neuronske mreže.

$$softmax(z) = \frac{1}{\sum_{j=1}^{M} e^{z_j}} \begin{bmatrix} e^{z_1} \\ e^{z_2} \\ \dots \\ e^{z_M} \end{bmatrix} \tag{2.10}$$

Funkcija softmax preslikava izlaz posljednjeg sloja $z = [z_1, ..., z_M]^T$ u $g = [g_1, ..., g_M]^T$ gdje je g_M model vjerojatnosti klase. Ulazne varijable z1,…,zM na softmax funkciju nazivaju se logiti. Funkcija softmax djeluje kao transformacija izlaza u modeliranje vjerojatnosti klase. Model vjerojatnosti klase koristi se za predviđanje vjerojatnosti različitih klasa na temelju ulaznih logitsa, što omogućuje donošenje odluka u višeklasnim klasifikacijskim problemima.

2.2. Obučavanje neuronske mreže

Neuronska mreža je parametarski model, a njene parametre nalazimo pomoću raznih tehnika. Svi parametri u modelu su težine matrice i svih pomaknutih vektora te ih oz-

načujemo s θ. Da bismo pronašli odgovarajuće vrijednosti za parametare θ moramo ih korisiti optimizacije. Funkcijski oblik funkcije gubitka ovisi o trenutačnom problemu, a uglavnom je to problem regresije ili klasifikacije. Za probleme regresije obično koristimo gubitak kvadrata pogreške.

$$L(x, y, \theta) = (y - f(x; \theta))^2 \tag{2.11}$$

Ovdje je $f(x; \theta)$ izlaz neuronske mreže. Također se i linearna regresija i logistička regresija mogu promatrati kao posebni slučajevi modela neuronske mreže gdje imamo samo jedan sloj u mreži. Isto tako nismo ograničeni na dvije funkcije gubitaka. Mogli bismo koristiti i neke druge funkcije gubitaka koje odgovaraju našim problemima. Ovakvi optimizacijski problemi ne mogu se riješiti u „zatvorenoj formi“ ,tako da treba koristiti numeričku optimizaciju. U svim algoritmima numeričke optimizacije, parametri se ažuriraju na iterativni način, dok u dubokom učenju obično koristimo različite verzije pretraživanja temeljno na gradijentima.

2.2.1. Propagacija unazad

Algoritam povratne propagacije važan je čimbenik u gotovo svim postupcima treniranja neuronskih mreža. Povratna propagacija je algoritam koji učinkovito izračunava funkciju troška i njezin gradijent u odnosu na sve parametre u neuronskoj mreži. Funkcija troška i njezin gradijent zatim se koriste u algoritmima stohastičke gradijentne pretrage. Parametri u ovom modelu su sve matrice i svi pomaci vektora. Stoga u svakoj iteraciji algoritma pretražujemo na temelju gradijenta, također moramo pronaći gradijent funkcije troška u odnosu na sve elemente u tim matricama i vektorima. Da bi se svi gradijenti izračunali na učinkovit način, povratna propagacija koristi strukturu modela umjesto da naivno izračunava derivacije u odnosu na svaki pojedini parametar posebno. Ovaj algoritam se sastoji od dva koraka, propagacije unaprijed i propagacije unatrag. U propagaciji unaprijed jednostavno procjenjujemo funkciju troška pomoću modela neuronske mreže. Budući da promatramo samo jednu podatkovnu točku, funkcija troška će imati uključen samo jedan termin gubitka. Kada je u pitanju povratna propagacija izračunavamo gradijente za sve slojeve i to rekurzivno počevši od zadnjeg sloja pa sve do prvog sloja. Da bismo mogli izračunati sve ove rekurzije, prvo moramo izračunati gradijent od funkcije

troška s obzirom na skrivene jedinice u zadnjem skrivenom sloju. Ovo ovisi o izboru funkcije gubitka.

2.2.2. Inicijalizacija

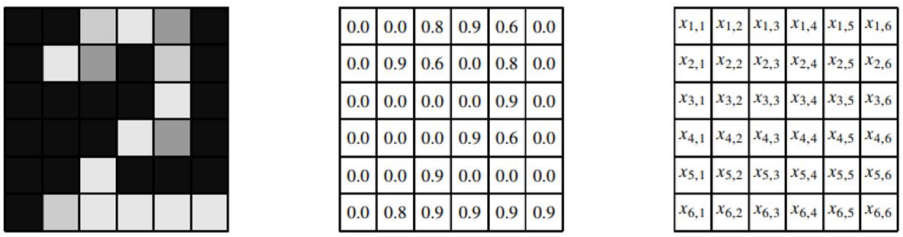
Većina problema na koje smo naišli bili su konveksni, a to znači da možemo jamčiti globalnu konvergenciju bez obzira na inicijalizaciju koju koristimo. Najčešće se kositi inicijalizacija za sve parametre mali nasumični broj kako bi omogućili različite skrivene jedinice za kodiranje različitih aspekata podataka. Ako se koristi ReLU aktivacijska funkcija, elementi pomaka b, obično se inicijaliziraju na malu pozitivnu vrijednost tako da oni rade u ne negativnom rasponu ReLU.

2.3. Konvolucijske neuronske mreže

Konvencijske neuronske mreže (CNN) posebna su vrsta neuronske mreže koja je izvorno skrojena za probleme u kojima ulazni podaci imaju mrežnu strukturu. Slike su također najčešći tip ulaznih podataka u aplikacijama u kojima se primjenjuju CNN. U slikama se pikseli nalaze u dvodimenzijskoj mreži. CNN se mogu koristiti za bilo koje ulazne podatke na „rešetki“ u jednoj ili više dimenzija.

2.3.1. Predstavljanje podataka sa slike

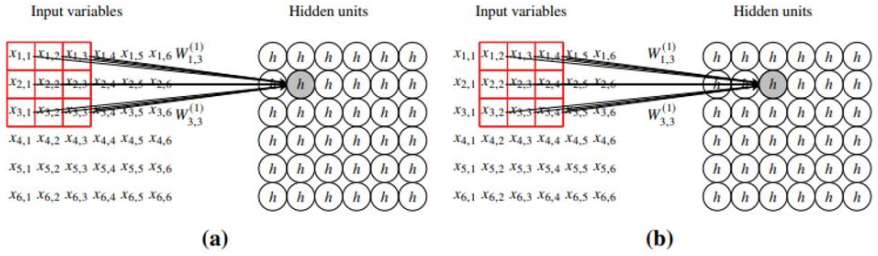
Digitalne slike u sivim tonovima sastoje se od piksela poredanih u matricu. Svaki piksel može se predstaviti kao raspon od 0 (crno) do 1 (bijelo), dok su vrijednosti između 0 i 1 nijanse sive boje. Prikaz podataka slike dan je na slici 2.5., svaki piksel je predstavljen brojem koji kodira intenzitet piksela. Vrijednosti piksela pohranjuju se u matricu.



Slika 2.6. Prikaz podataka na slici

2.3.2. Konvolucijski sloj

Nakon ulaznog sloja koristimo skriveni sloj s onoliko skrivenih jedinica koliko ima ulaznih varijabli. Za sliku od 6x6 piksela imamo 36 skrivenih jedinica. Konvolucijski sloj iskorištava strukturu prisutnu na slikama kako bi pronašao učinkovitije parametrizirani model. Za razliku od gustog sloja (gdje je svaka ulazna varijable povezana s svim skrivenim jedinicama), konvolucijski sloj koristi rijetke interakcije i dijeljenje parametara za postizanje parametrizacije.



Slika 2.7. Ilustracija interakcija u konvolucijskom sloju

Ilustracija interakcija u konvolucijskom sloju prikazana je na slici 2.6. gdje svaka skrivena jedinica ovisi samo o pikselima u malom području slike. Znaci ako se pomaknemo do skrivene jedinice jedan korak udesno, odgovarajuća regija na slici se također miče jedan korak udesno. Ovdje je prikazao za 3x3 veličinu piksela.

2.3.3. Rijetke interakcije

Pod rijetkim interakcijama misli se da je većina parametara u odgovarajućem gustom sloju prisiljena biti jednaka nuli, tj. skrivena jedinica u konvolucijskom sloju ovisi samo o pikselima u malom području slike, a ne o svim pikselima. Položaj područja je povezan s položajem skrivene jedinice u matričnom prikazu. Ako se pomaknemo jedan korak udesno, odgovarajuća regija na slici također se pomakne jedan korak udesno (kao što je ilustrirano slikama 2.7. a i b). Za skrivene jedinice na granici, odgovarajuće područje djelomično se nalazi izvan slike. Za ove granične slučajeve obično se koristi ispuna od nula, gdje se pikseli koji nedostaju jednostavno zamjene nulama.

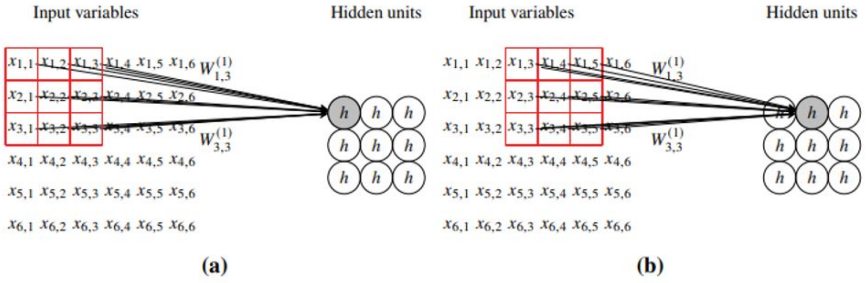
2.3.4. Dijeljenje parametara

U gustom sloju, svaka veza između ulazne varijable i skrivene jedinice ima svoj jedinstveni parametar. Uz dijeljenje parametara, dopušta se da isti parametar bude prisutan

na više mjesta u mreži. U konvolucijskom sloju skup parametara za različite skrivene jedinice je isti. Na primjer na slici 2.7. a koristimo isti skup parametara za preslikavanje regije 3x3 piksela u skrivenu jedincu. Umjesto da uočavamo zasebne skupove parametara za svaku poziciju, uočimo samo jedan skup od nekoliko parametara i koristio ga za sve veze između ulaznog sloja i skrivenih jedinica. Takav skupa parametara naziva se „filter“. Preslikavanje između ulaznih varijabli i skrivenih jedinica može se protumačiti kao konvolucija između ulaznih varijabli i filtera. Dijeljenje parametara i rijetke interakcije u konvolucijskom sloju čine CNN relativno nepromjenjivu u odnosu na „prijevode“ objekata u slici. Konvolucijski sloj koristi znatno manje parametara u usporedbi s odgovarajućim gustim slojem te s istom količinom parametara, konvolucijski sloj može kodirati više svojstava slike nego gusti sloj.

2.3.5. Konvolucijski sloj s koracima

U konvolucijskom sloju imamo onoliko skrivenih jedinica koliko imamo piksela na slici. Međutim kada dodamo više slojeva, želimo smanjiti broj skrivenih jedinica i pohraniti sam najvažnije informacije izračunate u prethodnim slojevima. Jedan od načina je da se filter ne primjenjuje na svaki piksel nego na svaka dva. Ako filter primijenimo na svaka dva piksela, i po stupcima i po redcima, skrivene jedinice će imati upola manje redaka i stupaca. Za sliku veličine 6x6 dobivao 3x3 skrivenih jedinica. Ovaj koncept prikazan je na slici 2.8. gdje je korak s=2 budući da se pomiče za 2 piksela u retku i stupcu.



Slika 2.8. Konvloucijski sloj s korakom s=2

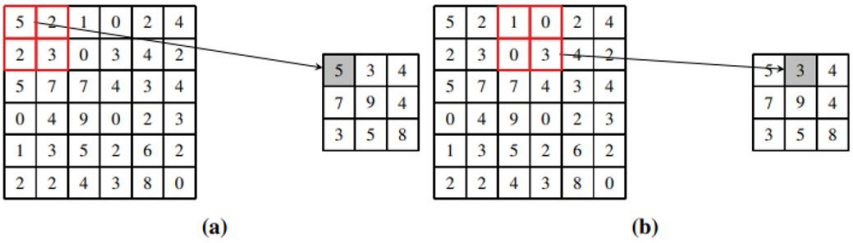
2.3.6. Sloj udruživanja

Drugi način sažimanja informacija u prethodnim slojevima postiže se korištenjem udruživanja. Sloj za udruživanje djeluje kao dodatni sloj nakon konvolucijskog sloja. Slično kao kod konvolucijskog ovisi samo o području piksela. Međutim za razliku od konvo-

lucijskog sloja, sloj za udruživanje ne dolazi ni s kakvim dodatnim parametrima koji se mogu trenirati. U udruživanju slojeva, također koristimo korake, što znači da je regija pomaknuta za s>1 piksela. Postoje dvije uobičajene verzije udruživanja: prosječno udruživanje i maksimalno udruživanje. U udruživanju prosjeka izračunava se prosjek jedinica u odgovarajućoj regiji. Matematički bi se to moglo zapisati kao :

$$q_{ij} = \frac{1}{F^2} \sum_{k=1}^{F} \sum_{l=1}^{F} q_s(i-1)+k, s(j-1)+l \tag{2.12}$$

Gdje je qi,j ulaz u sloj udruživanja, qij izlaz, F je veličina udruživanja, a s je korak koji se koristi u sloju udruživanja. U maksimalnom udruživanju, umjesto toga uzimamo maksimum piksela:



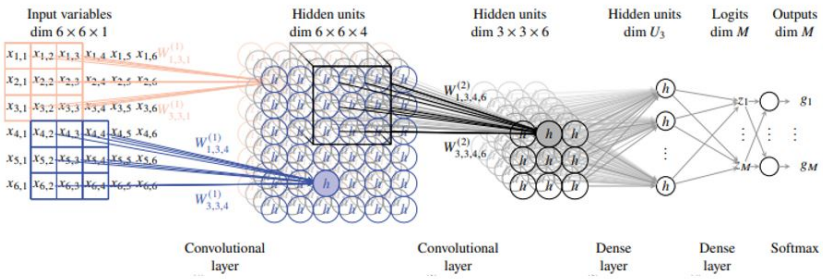
Slika 2.9. Maksimalno udruživanje

2.3.7. Više kanala

Mreže sa slika 2.7. i 2.8. imaju samo po deset parametara. Jedan filter nije dovoljan za kodiranje svih zanimljivih svojstava slika u skupu podataka. Stoga kako bismo proširili mrežu dodajemo više filtera, svaki sa svojim skupom parametara. Sada svaki filter proizvodi vlastiti skup skrivenih jedinica, takozvani kanal. Svaki sloj skrivenih jedinica u CNN-u organiziran je u takozvani „tenzor“ s dimenzijama (redovi stupci x kanali). Kada nastavimo slagati konvolucijske slojeve, svaki filter ne ovisi samo o jednom kanalu već o svim kanalima u prethodnom sloju. Kako posljedica toga, svaki filter je tenzor dimenzije (filterski redovi x filterski stupci x ulazni kanali).

2.3.8. Cijela CNN arhitektura

Potpuna CNN arhitektura sastoji se od višestrukih konvolucijskih slojeva. Za predikatne zadatke, smanjuje se broj stupaca i redaka u skrivenim jedinicama kako se ide kroz



Slika 2.10. Više kanala

mrežu, no umjesto toga povećava se broj kanala kako bi se mreži omogućilo kodiranje značajki više razine. Nakon nekoliko konvolucijskih slojeva obično se mreža završava s jedni ili više gustih slojeva. Ako se uzme u obzir problem klasifikacije slike, postavljamo softmax sloj na sam kraj kako bismo dobili rezultate u rasponu [0,1].

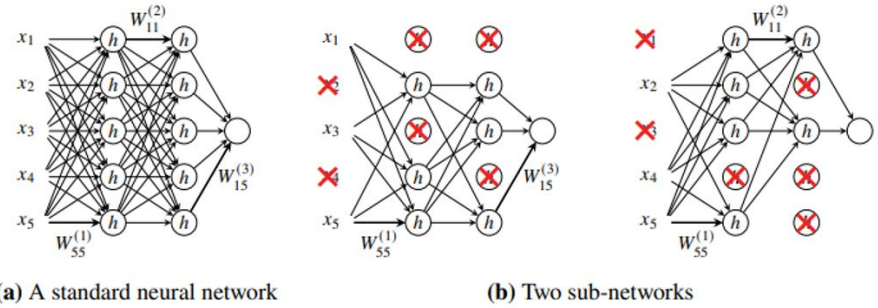
2.4. Dropout

Kao i svi modeli, tako i modeli neuronskih mreža mogu patiti od pretjeranog prilagođavanja ako imamo previše fleksibilan model u odnosu na složenost podataka. Jedan od načina za smanjivanje varijance, a samim time i rizik od pretjeranog opremanja, jest treniranje ne samo jednog nego više modela i izračunavanje prosjeka njihovih predviđanja (ovo je glavna ideja „bagging-a“). Kažemo da obučavamo skupinu modela, a svaki model nazivamo članom skupine. Bagging je također primjenjiv na neuronske mreže, međutim dolazi s nekim problemima. Velikom modelu neuronske mreže obično je potrebno dosta vremena da se uvježba, a ima mnogo parametara za pohranjivanje. Osposobljavanje ne samo jedne nego čitave skupine mnogih velikih neuronskih mreža bilo bi vrlo skupo, i u smislu vremena rada i memorije. Dropout je tehnika koja nam omogućuje kombiniranje mnogih neuronskih mreža bez potrebe da ih zasebno treniramo. Trik je u tome da se različitim modelima omogući međusobno dijeljenje parametara, što smanjuje računalne troškove i zahtjeve za memorijom.

2.4.1. Ansambl podmreža

Razmotamo gustu neuronsku mrežu poput one na slici 2.11. Kod dropout-a konstruiramo ekvivalent članu ansambla nasumičnim uklanjanjem nekih od skrivenih jedinica. Kada se jedna jedinica ukloni, uklanjamo i sve njezine dolazne i odlazne veze. Ovim pos-

tupkom dobivamo podmrežu koja sadrži samo podskup jedinica i parametara prisutnih u izvornoj mreži. Dvije takve podmreže pokazane su na istoj slici.



(a) A standard neural network **(b)** Two sub-networks

Slika 2.11. Neuronska mreža s dva skrivena soja i dvije podmreže s ispuštenim jedinicama

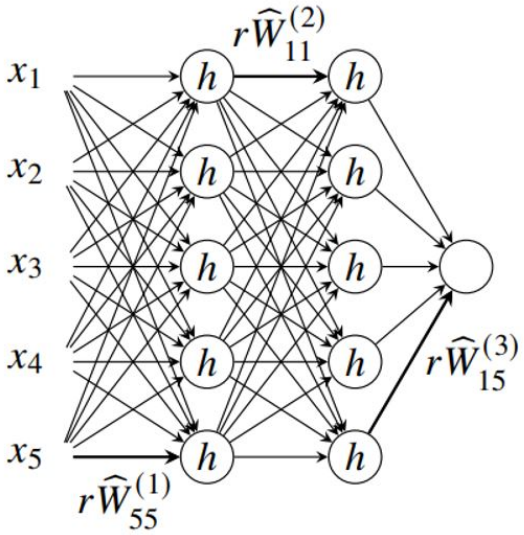
2.4.2. Trening s dropout-om

Za trening s dropout-om koristimo stohastičko gradijentno spuštanje. U svakom koraku gradijenta koristi se mini serija podataka za izračunavanje aproksimacije gradijenta. Međutim umjesto izračunavanja gradijenta za punu mrežu, generiramo slučajnu podmrežu nasumičnim dropout-om jedinica. Izračunavamo gradijent za tu podmrežu i zatim radimo korak gradijenta. Ovaj korak gradijenta ažurira samo parametre prisutne u podmreži. Parametri koji su povezani s odbačenim jedinicama ne utječu na izlaz te podmreže i ostaju netaknuti. U sljedećem koraku gradijenta koristimo drugu mini seriju podataka, uklanjamo drugu nasumično odabranu zbirku jedinica i ažuriramo parametre prisutne u toj podmreži. I tako se nastavlja sve dok se ne ispuni završni uvjet.

2.4.3. Predviđanje u vrijeme testiranja

Nakon što smo uvježbali podmreže, želimo napraviti predviđanje na temelju nevidljive ulazne podatkovne točke. Budući da svaka jedinica može biti ili unutarnja ili vanjska, postoji 2^U takvih podmreža, gdje je U ukupan broj jedinica u mreži. Dakle zbog ovakvog broja vrednovanje svih njih bilo bi neizvedivo. Međutim postoji jednostavan trik za približno postizanje istog rezultata. Umjesto da procijenimo sve moguće podmreže, jednostavno procijenimo mrežu koja sadrži sve parametre. Kako bismo kompenzirali činjenicu da je model treniran s dropout-om, množimo svaki procijenjeni parametar s vjerojatnošću da se ta jedinica zadrži tijekom obuke. Ovo osigurava da je očekivana vrijednost ulaza za sljedeću jedinicu ista tijekom obuke i testiranja. Slika 2.12. prikazuje kako smo

zadržali jedinicu s vjerojatnošću r u svim slojevima tijekom obuke. Sve su jedinice prisute bez ispadanja, ali se težine koje izlaze iz određene jedinice množe s vjerojatnošću da će ta jedinica biti uključena tijekom treninga.



Slika 2.12. Mreža korištena za predviđanje nakon uvježbavanja s ispadanjem

2.4.4. Dropout vs bagging

Postoji nekoliko važnih razlika između ove dvije metode:

1. U bagging-u su svi modeli neovisni u smislu da imaju svoje parametre. U slučaju dropout-a, različiti modeli dijele parametre

2. U bagging-u, svaki model se trenira do konvergencije. Kod dorpout-a, većina 2 na U podmreža uopće nije trenirana, a one koje su bile trenirane najvjerojatnije su bile trenirane samo za jedan korak gradijenta, međutim budući da dijele parametre svi će odjeli biti također ažurirani kada se druge podmreže obuče

3. Slično bagging-u, dropaout obučavamo svaki model na skupu podataka koji je nasumično odabran iz naših podataka o obuci. Međutim kod bagging-a to obično radimo na početnoj verziji cijelog skupa podataka, dok se kod dropout-a svaki model obučava na nasumično odabranoj mini seriji podataka

2.4.5. Dropout kao metoda regularizacije

Kao način da se smanji varijanca i izbjegne prekomjerno opremanje, dropout se može

promatrati kao metoda regularizacije. Postoji mnogo drugih metoda regularizacije za neuronsku mrežu, uključujući eksplicitnu regularizaciju, rano zaustavljanje i druge. Od svih njih dropout je postao jedna od najpopularnijih tehnika regulacije zbog svoje jednostavnosti i činjenice da je računski jeftin. Dobra praksa dizajniranja neuronske mreže često je proširiti mrežu dok se ne uklopi, zatim je još malo proširiti i na kraju dodati regularizaciju kao što je dropout kako bi se izbjeglo pretjerano uklapanje.

2.5. Primjena

Algoritam učenja neuronske mreže i struktura modela izražena u matematici trebaju biti realizirani računalnim programima za stvarnu upotrebu, stoga je izumljen niz biblioteka otvorenog koda za strojno učenje za posebno duboko učenje koje koristi duboku neuronsku mrežu s ogromnim brojem skrivenih slojeva.

2.5.1. Primjeni programskih biblioteka za duboko učenje

„Torch“ je najstarija biblioteka strojnog učenja za duboke neuronske mreže kreirana od strane Idiap Research Institute na EPFL-u. Može implementirati niz naprednih algoritama u jednom integriranom okviru. Uključuje četiri osnovne klase: skup podataka, trener, stroj i mjera. Trener šalje ulaz koji proizvode podaci na stoj koji može proizvesti izlaz koji se koristi za modificiranje sloja. Tijekom procesa obuke, mjerenja mogu procijeniti izvedbu modela. Najčešće se koristi u Pythonu.
„Theano“ je posebno dizajniran za velike proračune potrebne za duboke neuronske mreže. Dizajniran je na sveučilištu u Montrealu od strane instituta za učenje algoritama (MILA). „Theano“ je matematički kompilator napisan u biblioteci Pythona. Definira matematički izraz kao simbolički prikaz, poput računalnog grafa, koji dopušta simboličko razlikovanje zamršenih izraza. Izrazi su optimizirani i provedeni u C++.
"Caffe" je okvir otvorenog koda za algoritam dubokog učenja. Dizajniran je na temelju mrežnih slojeva u Kaliforniji od Yangqing Jia. "Caffe" se primjenjuje na obuku i implementaciju konvolucijskih neuronski mreža i široko se koristi za prepoznavanje slika.

2.5.2. Model TensorFlow

TensorFlow se koristi za definiranje, obuku i implementaciju duboke neuronske mreže.

U ovom odjeljku predstavit ćemo osnovnu strukturu TensorFlowa i objasnit ćemo način na koji se algoritam strojnog učenja izražava s pomoću računalnog grafa u TensorFlowu. Zatim ćemo raspravljati o proširenju osnovnog programskog modela kako realizirati računalni graf na osnovnim procesnim uređajima. Nakon toga istražujemo nekoliko optimizacija algoritama ugrađenih u TesorFlow prema hardveru i softveru. Poslije toga raspravljat ćemo o proširenju osnovnog programskog modela. Te na kraju se predstavlja programsko sučelje i alat za vizualizaciju u TensorFlowu.

2.5.3. Struktura računalnog grafa

TensorFlow je programski sustav koji koristi računski ili podatkovni graf za predstavljanje računalnog zadatka (algoritmi učenja). Računalni graf sastoji se od čvorova i usmjerenih bridova koji se povezuju čvorovima. Čvorovi predstavljaju operacije, a rubovi tijek podataka između operacija. U nastavku su pokazane glavne komponente računalnog grafa, odnosno operacije, tenzori, varijable i sesije.

1. Operacije - U TensoFlowu čvorovi predstavljaju operacije. Točnije, čvorovi opisuju kako ulazni podaci teku kroz njih u usmjerenom grafu. Operacija može biti s nula ili mnogo ulaza, a zatim proizvest mnogo izlaza. Takva operacija može biti matematička jednadžba, konstanta ili varijabla. Konstanta se dobiva operacijom koja je uzima na ulazu i daje izlaz isti kao odgovarajuća konstanta. Slično tome, varijabla je operacija koja je uzima bez ulaza i provodi trenutnu vrijednost te varijable. Svaku operaciju treba implementirati njegova jezgra koja se može izvršiti na hardverskom uređaju kao što je CPU.

2. Tensori - U TensorFlowu, podaci se predstavljaju tenzorima koji teku između čvorova u računalnom grafu. Tenzor je višedimenzionalni niz statičke vrste i dinamičkih dimenzija. Broj tenzora opisuje broj komponenti u svakoj dimenziji. U računskom grafu, prilikom kreiranja operacije, vrača se tenzor koji će usmjereni brid postaviti kao ulaz u povezanu operaciju.

3. Varijable - Prilikom izvođenja stohastičkog gradijentnog spuštanja, računalni graf neuronske mreže izvodi se iterativno za jedan eksperiment. Kroz evaluacije treninga, većina tenzora ne preživi, dok se stanje modela kao što su težine i pristranost moraju održavati. Stoga se varijable dodaju računskom grafu kao posebne

operacije. Tenzori za spremanje varijabli trajno se pohranjuju u međuspremnik u memoriji. Vrijednost varijabli mogu se učitati prilikom obuke i evaluacije modela. Prilikom kreiranja varijable, treba joj dati dodati tenzor kao početnu vrijednost prilikom izvršenja. Oblik i tip podataka tenzora tog tenzora automatski postaju oblik i tip varijable. Inicijalizacija varijabli mora se izvršiti prije obuke. To se može učiniti dodavanjem operacije za inicijalizaciju svih varijabli i njezinim izvršenjem prije osposobljavanja mreže.

4. Sesije - Izvršenje operacija i procjena tenzora izvode se u kontekstu sesije. Sesija koristi „pokretanje“ kao unos za izvođenje računalnog grafa. Uz pozivanje Run rutine, ulaz se uzima u računalni proces cijelog grafa kako bi se vratio izlaz prema definiciji grafa. Sesija distribuira operacije grafikona na uređaju kao što su CPU na različitim strojevima prema TensorFlowu algoritmu za postavljanje. Osim toga, redoslijed izvršavanja čvorova je eksplicitno definira. Ocjenjivanje modela osigurava održavanje ovisnosti kontrole.

2.5.4. Model izvršenja

Zadatak izvršenja računskog grafa podijeljen je u četiri skupine: klijent, master, radnici i skup uređaja. Klijent šalje zahtjev za izvršavanje grafa putem run rutine glavnom procesu koji je odgovaran za dodjelu zadatka skupu radnika. Svaki radnik odgovoran je za naziranje jednog ili više uređaja koji su fizički entiteti za implementaciju jezgre operacije. Na temelju broja strojeva, postoje dvije verzije TensorFlowa, naime lokalna i distribuirana implementacija. Lokalni sustav je implementacija čvorova na mnogo strojeva s mnogo uređaja.

Uređaji su osnovne i najmanje fizičke jedinice u TensorFlowu za izvršenje operacija. Centralni čvorovi bit će dodijeljeni dostupnim uređajima kao što su CPU koji će se izvršiti. Nadalje, TensorFlow omogućuje nove fizičke implementacije koje registriraju korisnici. Za paćenje operacija na uređaju, svaki radni proces odgovoran je za jedan ili više uređaja na jednom stroju. Naziv uređaj je određen njegovom vrstom i indeksom radne grupe u kojoj se nalazi.

Algoritam postavljanja određuje koji čvorovi će biti dodijeljeni kojem uređaju. Algoritam simulira izvođenje grafa od ulaznog tenzora do izlaznog tenzora. Algoritam usvaja troškovni model $C(d)$ kako bi odredio na kojem uređaju $D = \{d_1, \dots, d_n\}$ treba izvršiti ope-

raciju. Optimalan uređaj određen je modelom minimalnih troškova $d = \text{argmin}_{d \in D} C(d)$ za postavljanje određenog čvora tijekom obuke.
Izvršenje na više uređaja. TensorFlow obično dodjeljuje čvorove različitim uređajima sve dok korisnikov sustav nudi više uređaja. Ovaj proces klasificira čvorove, a zatim dodjeljuje čvorove koji pripadaju istoj klasi jednom uređaju. Stoga je potrebno pozabaviti se problemom ovisnosti čvorova dodijeljenih uređajima.

2.5.5. Optimizacije

Kako bi se osigurala učinkovitost i izvedba izvedbenog modela TensorFlow, neke optimizacije se konstruiraju u knjižnici. Uobičajena eliminacija podgrafa koji se izvodi u TensorFlowu je kanonizirati istu vrstu operacija na identičan ulazni tenzor jednoj operaciji pri obilasku računalnog grafa. Izlaz iz tenzora zatim se prijenosi na sve ovisne čvorove. Još jedna optimizacija, odnosno raspoređivanje, je da se čvorovi izvrše što je kasnije moguće kako bi se osiguralo da rezultat operacija ostane minimalno vrijeme u memoriji. Ovakav način smanjuje memoriju i poboljšava performanse modela. Optimizacija kompresije gubitaka odnosi se na dodjeljivanje čvorova konverzije na grafikon. Optimizirani robusni model ne mijenja izlazni odgovor zbog varijanca u signalima šuma. Stoga je zahtjev za preciznošću aritmetike algoritama smanjen.

2.5.6. Izračun gradijenta

U ovom odjeljku opisujemo jednu naprednu značajku koja se proteže do osnovnog TenskorFlow modela programiranja. Algoritmi strojnog učenja zahtijevaju izračunavanje gradijenta određenih čvorova u odnosu na jedan ili više čvorova u računalnom grafu. U neuronskoj mreži, gradijent troška s obzirom na težine trebao bi se uračunati s obzirom na primjer obuke koji se unosi u mrežu. Algoritam povratnog širenja koristi se za izračunavanja gradijenta.
Postoje dvije metode za izračunavanje gradijenta povratnog širenja kroz računalni graf. Prvi se odnosi na diferencijalu simbol-broj. Podaci se prenose unaprijed kroz grafikon kako bi se izračunala funkcija troška, a zatim se gradijent eksplicitno izračunava putem lančanog pravila u suprotnom smjeru kroz grafikon. Još jedna metoda koja je sve više prihvaćenija u TensorFlowu je derivacija od simbola do simbola koja automatski izračunava gradijent umjesto eksplicitne primjene algoritma povratne propagacije. Na taj se

način računskom grafu dodaju posebni čvorovi za izračunavanje gradijenta svake operacije uključene u lanac. Da bi se realizirao algoritam povratne propagacije , izvođenje gradijentnih čvorova je kao i kod drugih čvorova pokretanjem mehanizama za procjenu grafa. Ova metoda pruža rukovanje simbolom za izračunavanje izvedenica umjesto izračunavanja izvedenica kao numeričke vrijednosti. Može se konkretno objasniti na sljedeći način:

Gradijent određenog čvora v u odnosu na drugi tenzor α izračunava se unatrag od v do α kroz graf. Stvarajući izlazni tenzor. Stoga TensorFlow dodaje čvor gradijenta za svaku takvu operaciju množenjem derivacije svoje vanjske funkcije s vlastitom derivacijom. Procedura se obrnuto izračunava do kraja čvora koji proizvod simboličko rukovanje željenim gradijentom koji implicitno izvodi metodu povratnog širenja. Razlikovanje od simbola do simbola samo je još jedna operacija.

Simbol-simbol diferencijala može proizvesti značajne troškove računanja, kao i povećanje opterećanja memorije. Razlog je taj što postoje dvije različite jednađbe za primejrnu lančanog pravila. Prva jednađba ponovno koristi predhodbna izračunavanja što zahtjeva dulje vrijeme za pohranu nego što je potrebno za širenje naprijed. Primjenjuje se sljedeće lančano pravilo:

$$\frac{df}{dw} = f'(y) \cdot g'(x) \cdot h'(w) \tag{2.13}$$

Gdje je y=g(x),a x=h(w). Svaka funkcija treba izračunati svoje argumente i pozive svake unutarnje funkcije o kojoj ovisi. S obzirom na lanac koji ima tisuće čvorova, ponovno izračunavanje najunutarnjije funkcije za gotovo svaku operaciju u vezi ne čini se razumnim.

2.5.7. Programsko sučelje

Nakon rasprave o modelu TensorFlowa, fokusiramo se na praktičnije programsko sučelje. Raspravljat ćemo o dostupnom jezičnom sučelju i sažeti apstrakciju visoke razine TensorFlowa API-ja i kako brzo kreirati prototipove strojnog učenja.

TensorFlow podržava C++ i Python programske jezike koje korisnicima omogućuju pozivanje pozadinske funkcije. Trenutačno je programsko sučelje TensorFlow Python jednostavnije za korištenje, budući da pruža niz funkcija za pojednostavljenje i dovršetak konstrukcije i izvođenja računalnog grafa koji nisu podržani u C++.

Program TensorFlow sastoji se od dvije različite faze : faze izgradnje i faze izvođenja. U fazi izgradnje kreiraju se operacije u računskom grafu koji predstavljaju strukturu i algoritme učenja neuronske mreže. Zatim se u fazi izvođenja operacije u grafu ponavljaju kako bi se uvježbala mreža.

Za duboku neuronsku mrežu s ogromnim brojem skrivenih slojeva, koraci stvaranja težina i pristranosti, izračunavanje množenjem i zbrajanjem matrica, primjena nelinearne aktivacijske funkcije nisu učinkoviti. Stoga se predlaže niz knjižnica otvorenog koda za apstrahiranje ovih koraka, kao i za izgradnju blokova visoke razine kao što su cijeli slojevi odjednom. Jedna biblioteka apstrakcija je PrettyTensor koja može pružiti sučelje visoke razine za TensorFlow API. Omogućuje korisniku da omota TensorFlow operacije i tenzore u čistu verziju, a zatim se brzo poveže s bilo kojim slojevima u nizu. TFLearn je još jedna biblioteka apstrakcija izgrđena na temelju TensorFlowa, koja se može koristiti pomiješana s TensorFlow kodom.

2.5.8. TensorBoard

Duboko učenje obično koristi složene mreže. Kako bi se primijenila i otklonila pogreška tako kompliciranе mreže, potreban je jak alat z avizualizaciju. TensorBoard je web sučelje koje vizualizira i manipulira računalnom grafikom ugrađenom u TensorFlow. Glavna značajka TensorBorda je konstruirati jasno ili organizirano vizualno sučelje za računski graf s kompliciranom strukturom i mnogo slojeva kako bi se lakše razumjelo kako podaci teku kroz graf. Jedna važna metoda vizualne klasifikacije u TensorFlow su opsezi imena. Može grupirati operacije, ulaz, izlaz i odnose koji pripadaju jednom opseg imena u jedan blok, kao što je jedan mrežni sloj, koji se može interaktivno proširiti kako bi se prikazali detalji bloka.

Nadalje, TensoBoard može pratiti promjene jednog tenzora tijekom treninga. Operacije sažetka dodaju se čvorovima računskog grafa kako bi se proizvelo izvješće o vrijednostima tenzora. TensorBoard web sučelje može biti u interakciji na način da se, nakon

što s računalni graf učita, može promatrati model kao i nadzirati operacije.

2.5.9. Eksperiment s kreditnim karticama

U ovo sekciji provodimo eksperiment za otkrivanje prijevara s kreditnim karticama i predstavljamo rezultate. Skup podataka o transakcijama kreditnim karticama su napravili europski vlasnici u dva dana u rujnu 2013. te taj skup sadrži 284.807 transakcija od kojih su 192 prijevare (0,172%). Skup podataka za testiranje dan je na stanici kaggle kreiran od autora Currie32. Taj skup podataka sastoji se od trideset značajki koje su numeričke vrijednosti već transformirane analizom glavnih komponenti kako bi se smanjila dimenzija prostora značajki. Radi informacijske sigurnosti, izvorne značajke potrošača nisu dane.

Značajka „Vrijeme“ predstavlja vrijeme proteklo između svake transakcije i prve transakcije u skupu podataka. Značajka “Iznos“ je iznos transakcije. Oznaka „klasa“ je ciljna oznaka s vrijednošću 1 koja predstavlja slučaj prijevare i 0 koja predstavlja uobičajeni slučaj.

Standardizacija prostora značajki uobičajeni je zahtjev za učinkovito treniranje mreže budući da neuronska mreža osjetljiva na način na koji se skaliraju ulazni vektori. U mnogim skupovima podataka značajke se različito skalirane i imaju različit raspon vrijednosti što rezultira duljim vremenom obuke za konvergenciju mreže. Stoga je korisno standardizirati sve značajke na način da svaka značajka transformira u standardnu normalnu distribuciju pomicanjem srednje vrijednosti svake značajke, zatim skaliranje dijeljenjem nekonstantnim značajki njihovim standardnim odstupanjima.

U skupu podataka prijevarnu transakciju klasificiramo kao pozitivna klasa a normalnu negativna. Postoje četiri predikatna ishoda koja se mogu formulirati u 2 x 2 matricu zabune kao što je prikazano u tablici 2.1. Stupac predstavlja predviđenu oznaku, a red predstavlja pravu oznaku. TN je broj normalnih transakcija koje su ispravno klasificirane, FP je broj normalnih transakcija koja su pogrešno klasificirane, FN je broj transakcija prijevare koja su pogrešno klasificirane kao normalne, a TP je broj prevarenih transakcija ispravno klasificiranih.

Učinkovitost algoritma učenja neuronske mreže obično se precjenjuje korištenjem

	Predviđeno negativna	Predviđeno pozitivna
Zapravo negativna	TN	FP
Zapravo pozitivna	FN	TP

Tablica 2.1. Matrica zabune

predikatne točnosti, koja se definira kao Točnost=(TP+TN)/(TP+FP+TN+FN). Nažalost, takva jednostavna predikatna točnost nije prikladna mjera jer je distribucija normalnih i lažnih transakcija u skupu podataka izuzeto neuravnotežena. Jednostavna zadana strategija klasificiranja transakcije kao normalne transakcije daje vrlo visoku točnost. Unatoč visokoj točnosti, modelu nedostaje mogućnost otkrivanja bilo kakvih transakcija prijevare unutar svih transakcija. Naš cilj osposobljavanje mreže je postići prilično visoku stopu ispravnog otkrivanja prijevarnih transakcija što omogućuje nižu stopu pogreške u normalnim transakcijama. Stoga jednostavna predikatna točnost očito nije učinkovita metrika u takvom slučaju

ROC- karakteristika rada prijamnika opisuje performanse modela, što predstavljano odnosom između prave pozitivne stope i lažno pozitivne stope preko raspona pragova odlučivanja. U ROC prostoru X os je osjetljivost a Y os je specifičnost. Najbolje predviđanje je u točki (0,1), koja predstavlja sve pozitivne klase klasificirane ispravno i nijedna negativna klasa nije klasificirana kao pozitivna klasa.

Postoje i druge metrike procjene izvedene na temelju matrice zabune prikazane na tablici 2.1. Još ćemo spomenuti jednu posebnu metriku koja se zove odziv te se definira kao TP=TP/(TP+FN), to je postotak stvarno pozitivnih klasa koje je model isprano otkrio među ukupnim pozitivnim klasama. Koristi se za usporedbu različitih modela kada se preferira izvedba detekcije pozitivne klase.

```
[[54925  1936]
 [    4    97]]
Accuracy: 96.59%
Recall: 96.04%
```

Slika 2.13. Primjer matrice zabune, njezine točnosti i odziv

Logistička regresija odabrana je kao referentni model, koji se može smatrati najjed-

nostavnijom neuronskom mrežom ako se koristi sigmoidna aktivacija funkcija. Eksperiment je implementiran u TensorFlowu.

2.5.10. Logistički regresijski model

Logistička regresija je regresijska metoda za predviđanje binarne zavisne varijable koja ima vrijednosti 0 ili 1. Uzimajući u obzir ulazni vektor s n neovisnih varijabli x = {x1,. . . ,xn}, uvjetna vjerojatnost da zavisna varijabla bude klasa definirana je kao:

$$P(y = 1|x) = \frac{1}{1 + e^{-g(x)}} \tag{2.14}$$

gdje je $\frac{1}{1+e^{-g(x)}}$ logistička ili sigmoidna funkcija, $g(x) = w_0 + w_1x_1 + ... + w_nx_n$.

2.5.11. Implementacija u TensorFlowu

Skup podataka dijeli se na skup za obuku, skup za provjeru i skup za testiranje. Neka je struktura mreže je 4 skrivena sloja sa 20 čvorova u svakom sloju. Odabiremo parametre učenja testiranjem točnosti valjanosti. Otkrili smo da mreža konvergira na epohi =200 i stopi učenja 0,005 s obzirom na točnost provjeru valjanosti.

Dobije se da je prosječna točnost obuke = 0,99765, prosječna točnost validacije je = 0,996695, prosječna cijena obuke je= 18226.50195, prosječni trošak valjanosti 85031555. Zatim s na testnom skupu dobili i pamćenje od 86,89%.

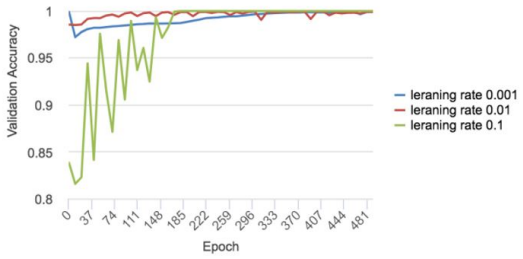
Zatim isprobavamo različite stope učenja kako bismo istražili njihov utjecaj na rezultate eksperimenta s obzirom na epohu=500 tijekom eksperimenta.

Stopa učenja	0.1	0.01	0.001
Prosječna validacija točnosti	0.96276	0.995911	0.99128

Tablica 2.2. Prosječna validacija točnosti kroz različite stope učenja

Kao što se vidi na slici, veća stopa učenja može ubrzati konvergenciju; no lako je prekoračiti lokalni minimum funkcije gubitka što dovodi do osciliranja. Ova nestalnost mreže imat će negativan utjecaj na performanse točnosti. U praksi pri približavanju optimalnom rješenju možemo koristiti male stope učenja budući da čini konvergentni put glatkijim kako bi se osigurala konvergencija na lokalni minimum. Zatim isprobavamo

različite inicijalizacije kako bismo istražili kako to utječe na rezultate. Težine bi trebale biti nasumične koje su dovoljno male blizu nule, ali ne identične nuli.



Slika 2.14. Validacija točnosti za različit broj epoda

3. Zaključak

U ovom radu prezentirana je osnovna teorija neuronskih mreža. Dan je jedan pregled o modelu neuronskih mreža, njegovoj generalizaciji i strukturi, raspravljalo koje aktivacijske funkcije je najbolje uzeti, dvoslojnim i dubokim neuronskim mreža, treningu, propagaciji unazad, konvolucijskim neuronskim mrežama i njezinoj strukturi. Neuronske mreže mogu značajno pridonijeti kvalitetnijoj analizi podataka. Budući da ne postoji standardiziran pristup za rješavanje problema pomoću neuronskih mreža, potrebno je testirati više različitih mreža za svaki problem kako bi se pronašla najkvalitetnija mreža s najvećom pouzdanošću. Prednost neuronskih mreža je sposobnost generalizacije i klasifikacije, čak i kada ulazne vrijednosti nisu poznate.

Dan je pregled rada kroz softver za strojno učenje TensorFlow, njegov osnovni model, programsko sučelje, optimizacije i način rada. Te na kraju jedan eksperiment na skupu podataka o prevarma s kreditnim karticama. Skup podataka napravljen je od strane europskih korisnika kartica i sastoji se od 284,807 kartičnih transakcija, od koji je 492 prevarena transakcija. Kroz taj eksperiment definirali smo matricu zabune i logistički regresijski model, koji je jedan od referentnih modela, ali s visokom razinom pamćenja.

Literatura

Yifai Lu: Deep neutral networks and fraud detecetion , Department of Mathematics Uppsala University, 2014

Lindhom A., Wahlström N., Lindsten F.: Schöm T. B. : Machine Learnin, Cambridge University Press, 2022

Dalbelo Bašić B., Čupić M., Šnajder J.: Umjetne neuronske mreže

Čupić M., Šnajder J.: Uvod u umjetnu inteligenciju

Kreja: Uvod u neuronske mreže, Umjetna inteligencija i strojno učenje URL `https://www.hashdork.com/hr/neuronska-mre%C5%BEa/`

TensorFlow: TensorFlow bascs URL `https://tensorflow.google.cn/guide/basics`

Hajba M., Pejović G., Špoljarić M.: Neuronske mreže - aproksimatori funkcija URL `http://e.math.hr/broj44/Hajba_etal`

Currie32: Predicting fraud with TensorFlow, 2017 URL `https://www.kaggle.com/code/currie32/predicting-fraud-with-tensorflow`

Dumančić S.: Neuronske mreže URL `https://www.mathos.unios.hr/~mdjumic/uploads/diplomski/dum05.pdf`

Sažetak

Primjena neuronskih mreža na probleme regresije i klasifikacije u strojnom učenju

Diego Mišetić

Brojni problemi povezani su s umjetnim neuronskim mrežama. Njihov cilj je brzo i jednostavno dolaženje do rjesenja kroz razne skrivene slojeve. Kroz rad dan je pregled o umjetnim neuronskim mrežama na problemima regresije i klasifikacije u strojnom učenju te opisan jedan eksperiment s istim kroz softver za strojno učenje TensorFlow. Dan je pregleda kroz razna poglavlja o samoj strukturi umjetnih neuronskih mreža, generalizaciji, primjeni kondolacijskih neuronskim mreža i dvoslojnih mreža, druoputu. Detaljno je opisno sučelje i programiranje u Tenseroflowu. Dan je primjer s eksperimentom o prevarama s transakcijama kreditnih kartica njegov opis i razrada eksperimenta kroz tablice i grafove.

Ključne riječi: Model neuronske mreže; obučavanje neuronskih mreža; konvolucijske neuronske mreže; dropout; TensorFlow

Abstract

Neural networks application to regression and classification problems in machine learning

Diego Mišetić

Numerous problems associated with artificial neural networks. Their goal is to quickly and easily find a solution through various hidden layers. The paper provides an overview of artificial neural networks on regression and classification problems in machine learning, and describes an experiment with the same using TensorFlow machine learning software. Through various chapters, an overview of the very structure of artificial neural networks, generalization, application of convolutional neural networks and two-layer networks, etc. There is a detailed description of the interface and programming in Tenseroflow. An example of a fraud experiment with credit card transactions is given, as well as a description of the experiment through tables and graphs.

Keywords: Neural network model; training of neural networks; convolutional neural networks; dropout; TensorFlow