Neuronske mreže - pogled u blackbox

Golobić, Nika

Sveučilište Josipa Jurja Strossmayera u Osijeku
Fakultet primijenjene matematike i informatike

Sveučilišni diplomski studij matematike
modul: financijska matematika i statistika

Neuronske mreže - pogled u blackbox

Diplomski rad

Mentor:
izv. prof. dr. sc. Nenad Šuvak

Student:
Nika Golobić

Osijek, 2024.

Sadržaj

1 | Uvod

Strojno učenje je grana umjetne inteligencije koja se bavi razvojem modela koji uče iz podataka kako bi donosili odluke ili predviđanja. Učenje u ovom kontekstu odnosi se na proces procjene parametara modela na temelju iskustva, odnosno podataka, kako bi model što točnije predvidio rezultate za nove, neviđene podatke.

Neuronske mreže su jedan od podskupova strojnog učenja i razvijene su kao pojednostavljeni modeli bioloških neuronskih sustava, s ciljem oponašanja načina na koji ljudski mozak obrađuje informacije. Sastoje se od slojeva neurona – ulaznog sloja, jednog ili više skrivenih slojeva te izlaznog sloja. Svaki čvor je povezan s ostalima i ima svoju pripadajuću težinu. Danas su neuronske mreže posebno uspješne u područjima kao što su prepoznavanje uzoraka, obrada slika i teksta te predviđanje financijskih kretanja.

Poglavlje 2 započinje motivacijom za korištenje neuronskih mreža, objašnjavajući njihovu sposobnost rješavanja složenih zadataka koji nadilaze klasične modele poput linearne regresije. Zatim se obrađuje arhitektura neuronskih mreža, s naglaskom na jednoslojni perceptron, koji se koristi za jednostavne probleme, te višeslojni perceptron, koji omogućuje rješavanje složenijih nelinearnih problema. Također, dan je primjer učenja XOR funkcije, čime se ilustrira snaga neuronskih mreža u prepoznavanju nelinearnih obrazaca. Nadalje, u ovom poglavlju razmatramo proces treniranja neuronskih mreža, pri čemu mreža prilagođava težine između neurona kako bi smanjila pogrešku predikcije. Objasnit ćemo ključne algoritme optimizacije, uključujući gradijentni spust, propagaciju unatrag i stohastičku gradijentnu metodu. Također ćemo obraditi tehnike za poboljšanje performansi modela, poput regularizacije i validacije modela, koje služe za poboljšanje svojstava pri primjeni modela na podacima na kojima nije treniran.

U Poglavlju 3 bavit ćemo se složenim neuronskim mreža, kao što su konvolucijske i rekurentne neuronske mreže, koje su ključne za analizu slika i vremenskih nizova. Objasnit ćemo načine na koje te mreže funkcioniraju i kako se nose s izazovima poput nestajanja gradijenata, uz poseban naglasak na LSTM mreže, koje su dizajnirane za dugoročno pamćenje u podacima.

Poglavlje 4 bavi se problemom interpretacije neuronskih mreža, često nazivanih modelima crne kutije (eng. *blackbox*). Iako su neuronske mreže izuzetno uspješne u rješavanju složenih problema, često je teško razumjeti kako točno

donose svoje odluke. U ovom poglavlju objašnjavamo zašto su neuronske mreže smatrane crnim kutijama te predstavljamo metode koje pomažu u interpretaciji njihovih predikcija. Poseban naglasak stavljen je na LIME (Local Interpretable Model-agnostic Explanations) metodu, koja omogućuje lokalnu interpretaciju rezultata složenih modela.

U Poglavlju 5 izradit ćemo model neuronske mreže za procjenu kreditnog rizika. Najprije ćemo prikazati logističku regresiju kao bazni model za usporedbu. Zatim ćemo izraditi model neuronske mreže prilagođen za ovaj problem, opisujući strukturu mreže, aktivacijske funkcije te optimizacijske metode. Na kraju ćemo koristiti LIME metodu za interpretaciju rezultata modela kako bismo bolje razumjeli kako neuronska mreža donosi svoje odluke i procijenili njezinu učinkovitost u procjeni kreditnog rizika.

2 | Teorijska osnova neuronskih mreža

2.1 Motivacija za korištenje neuronskih mreža

Neuronske mreže su postale iznimno popularne u modernom strojnom učenju zbog svoje sposobnosti prepoznavanja složenih uzoraka u podacima, a koriste se kako u nadziranom, tako i u nenadziranom učenju, ovisno o prirodi zadatka. Inspirirane načinom na koji ljudski mozak obrađuje informacije, neuronske mreže koriste neurone koji su međusobno povezani i surađuju kako bi naučili obrasce iz podataka.
Ključna prednost neuronskih mreža leži u njihovoj prilagodljivosti. Mogu se koristiti za razne zadatke poput prepoznavanja slika, klasifikacije, predikcije vremenskih nizova, prevođenja jezika i mnogih drugih. Jedan od poznatijih primjera korištenja neuronskih mreža je prepoznavanje rukom pisanih znamenki iz pikseliziranih slika.
Za razliku od tradicionalnih metoda poput linearne regresije, koje su učinkovite samo za modeliranje jednostavnih linearnih veza, neuronske mreže imaju sposobnost prepoznavanja skrivenih nelinearnih pravila i obrazaca u podacima, što omogućuje njihovu primjenu na probleme koje standardni statistički modeli ne mogu lako opisati. Upravo zbog te sposobnosti, neuronske mreže se koriste u situacijama gdje klasični pristupi nisu dovoljni. Sposobnost prepoznavanja složenih struktura čini ih ključnim alatom u mnogim područjima današnje tehnologije.

2.2 Ključni pojmovi i oznake

U ovom radu koristit ćemo sljedeće ključne pojmove i oznake:

- X: Prostor ulaznih podataka, značajki (eng. *features*) ili nezavisnih varijabli. Može biti prostor $\mathbb{R}$ (ako je X slučajna varijabla) ili $\mathbb{R}^n$ (ako je X slučajni vektor).
- **Ulazni sloj**: Prvi sloj neuronske mreže koji prima ulazne varijable X i predstavlja značajke modela.
- **Neuron**: Osnovna jedinica unutar slojeva neuronske mreže koja obrađuje ulaze, koristi težine i funkciju aktivacije kako bi generirala izlaz.

- **Skriveni slojevi**: Slojevi između ulaznog i izlaznog sloja, sastavljeni od neurona koji izvode složene transformacije ulaznih podataka i omogućuju modelu da uči složene obrasce.
- Y: Prostor izlaznih podataka, oznaka (eng. *label*) ili ovisnih varijabli.
- **Izlazni sloj**: Posljednji sloj neuronske mreže koji generira konačni izlaz modela $f(X)$ temeljen na aktivacijama iz prethodnog sloja.
- $P(X, Y)$: Zajednička distribucija ulaza X i izlaza Y, opisana funkcijom distribucije F ili vjerojatnosnom mjerom.
- $\mathbb{E}_F$: Očekivanje u odnosu na distribuciju zadanu s F.
- $\mathbb{P}_F$: Vjerojatnost zadana s F.
- $f(X)$: Predikcija modela na temelju ulaznih varijabli X.
- **Funkcija gubitka**: Mjeri razliku između stvarnih vrijednosti y i predikcija modela $f(X)$. U svrhu treniranja modela, minimiziramo funkciju gubitka kako bismo poboljšali točnost modela. Ova funkcija, kada se koristi za optimizaciju parametara modela tijekom učenja, naziva se **funkcija cilja** (eng. *cost function*).
- **Gradijent**: Vektor parcijalnih derivacija funkcije gubitka po težinama modela.
- **Brzina učenja (eng. *learning rate*)**: Hiperparametar koji određuje veličinu koraka pri ažuriranju težina modela tijekom treniranja.
- **Težine** ω_j: Parametri modela koji određuju važnost svake ulazne vrijednosti x_j.
- **Funkcija aktivacije**: Nelinearna funkcija koja se primjenjuje na ukupni ulaz modela (ponderirani zbroj ulaznih varijabli i pripadnih težina) s ciljem uvođenja nelinearnosti u model.
- **Aktivacije**: Vrijednosti dobivene primjenom funkcije aktivacije na ponderirani zbroj ulaznih varijabli i pripadnih težina. Aktivacije su izlazi funkcije aktivacije i koriste se kao ulazi za sljedeći sloj ili za izračun konačnog izlaza modela.
- **Pomak (eng. *bias*)** b: Parametar koji pomiče granicu aktivacije, osiguravajući da model može bolje generalizirati i učiti složenije obrasce u podacima, neovisno o vrijednostima ulaznih varijabli.
- **Prenaučenost (eng. *overfitting*)**: Prekomjerno prilagođavanje modela specifičnim značajkama trening podataka, što rezultira lošom sposobnošću modela da generalizira i primijeni naučeno na nove, neviđene podatke.

2.3 Klasifikacija i regresija

U strojnom učenju, problemi se često dijele na dva glavna tipa: klasifikacijske i regresijske probleme.
Klasifikacija je zadatak dodjeljivanja ulaznog podatka jednoj od unaprijed definiranih klasa. U najjednostavnijem slučaju, gdje je $X \subseteq \mathbb{R}^p$ i $Y = \{0, 1\}$, riječ je o binarnoj klasifikaciji. U tom slučaju, funkcija gubitka definirana je kao:

$$\ell(X, Y) = \begin{cases} 1, & \text{ako je } f(X) = Y, \\ 0, & \text{inače.} \end{cases}$$

Ova funkcija gubitka je indikator funkcija čija ne-nul vrijednost znači da se predikcija i stvarna vrijednost podudaraju. Rizik klasifikacije, odnosno vjerojatnost greške, dana je sljedećim izrazom:

$$R_F(f) = \mathbb{P}_F(f(X) \neq Y),$$

što predstavlja vjerojatnost da da se predikcija modela $f(X)$ ne podudara sa stvarnim izlazom Y.

Primjer 2.3.1. ***Klasifikacija tumora dojke***
U ovom klasifikacijskom problemu, cilj je odrediti je li tumor dojke maligan (zloćudan) ili benigni (dobroćudan) temeljem medicinskih podataka. Ulazni prostor X sadrži značajke koje predstavljaju različite biološke karakteristike tumora (npr. veličina, oblik, tekstura stanica). Izlazni prostor je $Y = \{0, 1\}$, gdje 0 označava benigni, a 1 maligan tumor.
Model koristi podatke o značajkama tumora za predviđanje njegove prirode, što omogućuje bržu i precizniju dijagnozu. Funkcija gubitka koristi se za minimiziranje greške modela u klasifikaciji tumora, a konačni cilj je postići što veću točnost uz nisku stopu pogrešne klasifikacije malignih tumora kao benignih.

S druge strane, regresija se odnosi na predviđanje varijable čiji je skup vrijednosti primjerice cijeli R ili neki interval realnih brojeva. Dakle, ako je $X \subseteq \mathbb{R}^p$ i $Y = \mathbb{R}$, tada je riječ o regresiji. Uobičajena funkcija gubitka za regresiju je kvadratna funkcija odstupanja $\ell(u, y) = (y - u)^2$, koja mjeri razliku između predviđene vrijednosti u i stvarne vrijednosti y. Rizik u ovom kontekstu predstavlja očekivano srednjekvadratno odstupanje, izraženo kao:

$$R_F(f) = \mathbb{E}_F[(Y - f(X))^2],$$

što predstavlja očekivano kvadratno odstupanje predviđenih od stvarnih vrijednosti izlazne varijable.

Primjer 2.3.2. ***Predviđanje cijene nekretnina***
Na temelju podataka koji uključuju različite značajke nekretnina, poput površine, broja soba i lokacije, potrebno je razviti model koji će predviđati cijenu nekretnina. Model će uzimati ove značajke kao ulazne varijable i predviđati cijenu, što može biti korisno na tržištu nekretnina za procjenu vrijednosti nekretnina i donošenje informiranih odluka o kupnji ili prodaji.

Razumijevanje razlike između klasifikacije i regresije ključno je za odabir ispravnog modela strojnog učenja koji će biti primijenjen na specifičan problem. Neuronske mreže mogu se koristiti i za klasifikacijske i za regresijske probleme, čineći ih svestranim alatom u statističkom učenju.

2.4 Arhitektura neuronskih mreža

Feedforward neuronske mreže, često nazivane višeslojnim perceptronima (MLP), temeljni su modeli u dubokom učenju. Cilj feedforward mreže je aproksimirati funkciju f^*, koja ulazu x pridružuje odgovarajući izlaz y. Mreža definira pridruživanje $y = f(x;\theta)$, pri čemu se parametri θ prilagođavaju kako bi se što bolje aproksimirala funkcija cilja. Ove mreže se nazivaju *feedforward* jer informacija teče u jednom smjeru, od ulaza prema izlazu, bez povratnih veza. Struktura feedforward mreža može se prikazati kao niz funkcija organiziranih u slojeve. Svaki sloj temelji se na rezultatu prethodnog, a mreža uključuje ulazni sloj, jedan ili više skrivenih slojeva, te izlazni sloj. Ulazni sloj sadrži onoliko neurona koliko imamo prediktora (ulaznih varijabli). Svaki neuron prima jednu ulaznu varijablu i prosljeđuje ju skrivenim slojevima. Skriveni slojevi sadrže neurone koji izračunavaju ponderirane zbrojeve ulaznih vrijednosti, na koje se zatim primjenjuje aktivacijska funkcija. U izlaznom sloju nalazi se jedan ili više neurona, ovisno o vrsti problema.

2.4.1 Jednoslojni perceptron i aktivacijske funkcije

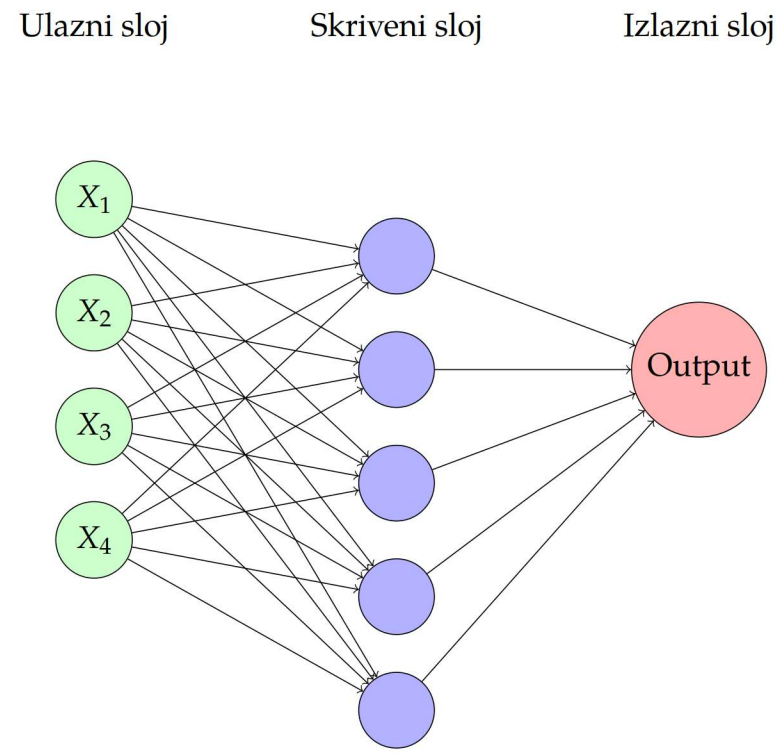
Neuronsku mrežu možemo prikazati grafom u kojem su čvorovi neuroni, a bridovi predstavljaju veze između njih. Svaka veza između neurona ponderira se težinom. Jednoslojni perceptron, osnovni model neuronske mreže koji ima samo jedan skriveni sloj, koristi se za binarnu klasifikaciju i regresiju. Funkcijski, takva se mreža može zapisati na sljedeći način:

$$f(X_1,\ldots,X_p) = \beta_0 + \sum_{i=1}^{n_1} \beta_i h_i(X_1,\ldots,X_p),$$

gdje su β_0 i β_i parametri modela, h_i funkcije koje transformiraju ulaze, a n_1 označava broj neurona u skrivenom sloju. Ovaj se izraz može proširiti tako da uključuje *aktivacijsku funkciju* σ, koja unosi nelinearnost u model. Bez aktivacijske funkcije, mreža ne bi mogla modelirati složene nelinearne odnose, već bi bila ograničena na linearnu kombinaciju ulaza, što bi ju činilo ekvivalentnom jednostavnom linearnom modelu. Izraz sada izgleda ovako:

$$f(X_1,\ldots,X_p) = \beta_0 + \sum_{i=1}^{n_1} \beta_i \sigma\left(\sum_{j=1}^{p} w_{ij}X_j + b_i\right),$$

gdje su w_{ij} težine koje povezuju ulaze X_j s neuronima u skrivenom sloju, a b_i je bias.
Primjer neuronske mreže s jednim skrivenim slojem prikazan je na slici 2.1.



Slika 2.1: Primjer neuronske mreže s jednim skrivenim slojem

Aktivacijske funkcije transformiraju linearne kombinacije ulaza $X_1, X_2, \ldots, X_p$ u nelinearne izlaze $\sigma(z_i)$, gdje je $z_i = \sum_{j=1}^{p} w_{ij} X_j + b_i$, a σ je funkcija aktivacije koja uvodi nelinearnost. Jedna od najčešće korištenih aktivacijskih funkcija je sigmoidna funkcija:

$$\sigma(v) = \frac{1}{1 + e^{-v}}.$$

Sigmoidna funkcija ograničava izlaz u interval [0,1], što je korisno za klasifikaciju jer omogućuje predviđanje vjerojatnosti za različite klase. Osim sigmoidne funkcije, moderni modeli koriste *ReLU* funkciju (Rectified Linear Unit):

$$\text{ReLU}(v) = \max(0, v).$$

ReLU funkcija je posebno pogodna za složene neuronske mreže koje sadrže više od jednog skrivenog sloja, jer smanjuje problem zasićenja gradijenata[1]. Kod sigmoidnih funkcija, gradijenti postaju vrlo mali za ekstremne vrijednosti ulaza, što usporava ili onemogućuje učenje u dubljim slojevima mreže.

Još jedna uobičajena aktivacijska funkcija je *tanh*, koja daje izlaze u intervalu [-1, 1]:

$$\tanh(v) = \frac{e^{v} - e^{-v}}{e^{v} + e^{-v}}.$$

[1]Zasićenje gradijenata se događa kada su izlazi aktivacijske funkcije, poput sigmoidne funkcije, blizu 0 ili 1, što rezultira vrlo malim gradijentima tijekom izračuna propagacije unatrag o kojoj će biti riječi u Poglavlju 2.5.2. To otežava učenje modela jer se težine ne ažuriraju značajno.

Tanh funkcija je korisna kada je potreban balans između pozitivnih i negativnih vrijednosti.
U slučaju klasifikacijskih problema s više kategorija, koristi se *softmax* funkcija:

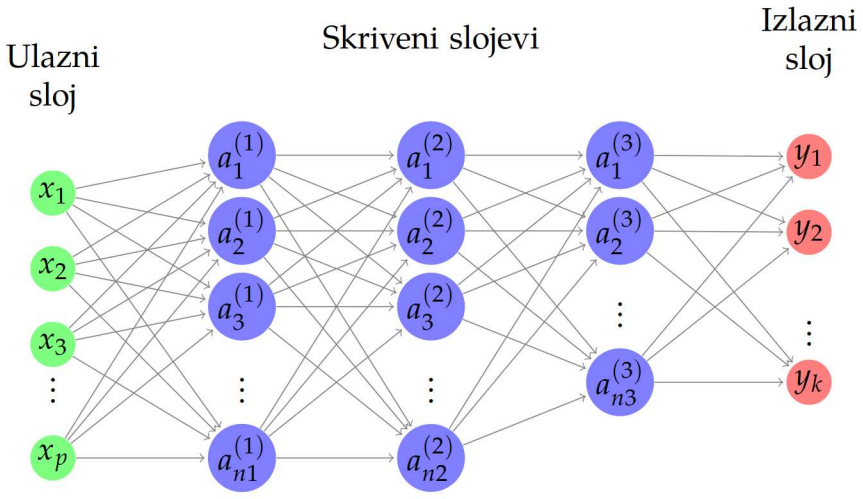
$$g(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}, \quad i = 1, \ldots, k.$$

Softmax funkcija omogućuje interpretaciju izlaznih vrijednosti kao vjerojatnosti pripadanja svakoj klasi.

Primjer 2.4.1. ***Treniranje neuronske mreže za predikciju plaća*** *(preuzeto iz [6])*
U ovom primjeru treniramo jednostavnu neuronsku mrežu za predikciju plaća koristeći podatke o igračima bejzbola. Podaci su podijeljeni na trening i testni skup. Ulazni podaci uključuju karakteristike igrača poput broja udaraca, godina iskustva i drugih statistika. Oni prolaze kroz skriveni sloj, koji sadrži 50 neurona. Svaki neuron u skrivenom sloju primjenjuje ReLU aktivacijsku funkciju na ulaze, što omogućava mreži da uči nelinearne odnose u podacima. "Dropout" sloj nasumično isključuje 40% neurona tijekom treniranja kako bi se smanjio overfitting (vidi 2.5.7). Mreža se trenira pomoću RMSprop optimizatora (vidi 2.5.4) kroz 1500 epoha[2], a uspjeh modela procjenjuje se na testnom skupu. Izlazni sloj ima jedan neuron, koji predviđa plaću igrača.

2.4.2 Neuronske mreže s više skrivenih slojeva

Neuronske mreže s više skrivenih slojeva, poznate pod nazivom višeslojni perceptron (eng. multilayer perceptron, MLP), predstavljaju jednu od najvažnijih struktura u svijetu dubokog učenja. MLP se sastoji od niza slojeva neurona, gdje svaki sloj transformira izlaz prethodnog sloja koristeći težine i aktivacijske funkcije. Primjer neuronske mreže s više skrivenih slojeva prikazan je na slici 2.2.



Slika 2.2: Primjer neuronske mreže s više skrivenih slojeva

Matematički, takvu mrežu možemo opisati na sljedeći način, koristeći oznake:

[2]Epoha je jedan puni prolaz kroz cijeli skup trening podataka tijekom treniranja modela.

- gornji indeks označava sloj,
- L predstavlja broj skrivenih slojeva,
- n_l predstavlja broj neurona u sloju $l = 0, 1, \ldots, L+1$, dok je n_0 dimenzija ulaznih podataka, a n_{L+1} dimenzija izlaza.

Neuronska mreža se može prikazati rekurzivnim odnosima između slojeva na sljedeći način:

$$z_i^{(1)}(x) = b_i^{(1)} + \sum_{j=1}^{n_0} w_{ij}^{(1)} x_j, \quad i = 1, \ldots, n_0,$$

$$z_i^{(l)}(x) = b_i^{(l)} + \sum_{j=1}^{n_{l-1}} w_{ij}^{(l)} \sigma_{l-1}\left(z_j^{(l-1)}(x)\right), \quad i = 1, \ldots, n_l, \quad l = 2, \ldots, L+1.$$

Vrijednosti $z_i^{(l)}(x)$ predstavljaju predaktivacije, dok su aktivacije definirane kao:

$$a_i^{(l)}(x) = \sigma_l\left(z_i^{(l)}(x)\right), \quad i = 1, \ldots, n_l, \quad l = 1, \ldots, L+1.$$

Težine neuronske mreže možemo prikazati matricama $W^{(l)} = [w_{ij}^{(l)}]$, gdje su $i = 1, \ldots, n_l$ i $j = 1, \ldots, n_{l-1}$, a vektore težinskih pomaka (bias) označavamo kao $b^{(l)} = [b_i^{(l)}]$. S ovim oznakama, predaktivacije možemo zapisati na sljedeći način:

$$z^{(1)}(x) = b^{(1)} + W^{(1)} x,$$

$$z^{(l)}(x) = b^{(l)} + W^{(l)} \sigma_{l-1}\left(z^{(l-1)}(x)\right) = b^{(l)} + W^{(l)} a^{(l-1)}(x), \quad l = 2, \ldots, L+1,$$

$$a^{(l)}(x) = \sigma_l\left(z^{(l)}(x)\right), \quad l = 1, \ldots, L+1.$$

Izlazni sloj daje vrijednosti $y_i = a_i^{(L+1)}(x), \quad i = 1, \ldots, n_{L+1}$, ili vektorski $y = f(x) = a^{(L+1)}(x)$. Ako je $n_{L+1} = 1$, tada je f realna funkcija.
Također, neuronska mreža se može zapisati kao funkcija:

$$f(x) = \sigma_{L+1}\left(W^{(L)} \sigma_L\left(W^{(L-1)} \sigma_{L-1}\left(\ldots \sigma_1\left(W^{(1)} x + b^{(1)}\right)\ldots\right) + b^{(L-1)}\right) + b^{(L)}\right).$$

Nepoznati parametri modela uključuju elemente matrica težina i vektora bias vrijednosti:

$$\theta = \left(W^{(1)}, \ldots, W^{(L+1)}, b^{(1)}, \ldots, b^{(L+1)}\right),$$

a njihov ukupni broj je:

$$\sum_{l=1}^{L+1} (n_l n_{l-1} + n_l).$$

U arhitekturi višeslojnog perceptrona, broj slojeva L naziva se *dubinom mreže*, dok $n_1, n_2, \ldots, n_L$ označavaju *širinu mreže*, odnosno broj neurona po sloju. Ovi parametri, poznati kao *hiperparametri*, definiraju strukturu mreže i igraju ključnu ulogu u njenoj sposobnosti da uči i generalizira na novim podacima.

Primjer 2.4.2. ***MLP za klasifikaciju MNIST podataka*** *(preuzeto iz [6])*
U ovom primjeru, koristimo MNIST skup podataka koji se sastoji od slika rukom pisanih znamenki (0-9). Svaka slika je dimenzija 28x28 piksela, što znači da je svaka slika pretvorena u vektor duljine 784 za treniranje modela.
Ulazni podaci normalizirani su na skalu od 0 do 1. Mreža se sastoji od tri sloja:

- *Prvi skriveni sloj ima 256 neurona s ReLU aktivacijskom funkcijom.*
- *Drugi skriveni sloj ima 128 neurona također s ReLU aktivacijskom funkcijom.*
- *Izlazni sloj koristi softmax funkciju kako bi mreža predvidjela vjerojatnosti pripadanja svakoj od 10 klasa.*

U prvom skrivenom sloju nasumično se isključuje 40% neurona tijekom svake iteracije treniranja, dok se u drugom sloju isključuje 30%. Ova tehnika pomaže spriječiti overfitting modela tako što mreža ne postaje previše ovisna o pojedinačnim neuronima, čime se poboljšava njezina sposobnost generalizacije na nove podatke.
Model se trenira pomoću RMSprop optimizatora i koristi kros-entropiju[3] kao funkciju gubitka za klasifikacijski problem. Trening se odvija kroz 30 epoha s veličinom uzorka od 128 uz podjelu za validaciju (vidi 2.5.7) od 20% podataka za trening. Nakon treniranja, model je postigao točnost od 98.1% na testnom skupu podataka.

2.4.3 Primjer učenja XOR funkcije

Jedan od klasičnih primjera koji ilustrira snagu neuronskih mreža je procjena XOR funkcije (engl. exclusive or). XOR funkcija je logička operacija na dvije binarne varijable x_1 i x_2, koja vraća 1 kada je točno jedna od tih vrijednosti jednaka 1, a vraća 0 u svim ostalim slučajevima. Naš cilj je odrediti funkciju $f(x;\theta)$ koja aproksimira XOR funkciju $f^*(x_1, x_2)$. Parametri modela θ prilagođavaju se kako bi funkcija $f(x;\theta)$ bila što sličnija $f^*(x_1, x_2)$.
Ovaj problem možemo promatrati kao problem minimizacije kvadratne greške. Empirijski rizik s kvadratnim gubitkom, koji je ekvivalentan srednje-kvadratnom odstupanju (MSE), zapisujemo kao:

$$R_{\text{emp}}(f) = \frac{1}{4}\sum_{i=1}^{4}\left(y_i - f(x_i)\right)^2,$$

gdje je $x_i = (x_{i1}, x_{i2})$ dan u Tablici 2.1.

[3]Funkcija gubitka za klasifikaciju poznata kao kros-entropija mjeri udaljenost između dvije distribucije. Njena empirijska forma je:

$$\ell(y, f(x;\theta)) = -\sum_{j=1}^{n_{L+1}} y_j \log(f(x;\theta))_j,$$

dok je funkcija cilja:

$$C(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n_{L+1}} y_{ij}\log(f(x_i;\theta))_j.$$

Ova funkcija gubitka omogućava korištenje gradijentnih metoda za minimizaciju zbog svoje diferencijabilnosti.

i	x_{i1}	x_{i2}	y_i
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	0

Tablica 2.1: Ulazni i izlazni podaci za XOR funkciju.

Pretpostavimo da je funkcija f linearna, što znači da je:

$$f(x_1, x_2) = b + w_1 x_1 + w_2 x_2 = b + w^\top x,$$

gdje su b i $w = (w_1, w_2)^\top$ nepoznati parametri.
Optimizacijom parametara linearne funkcije minimizacijom empirijskog rizika dobivamo $\hat{\omega} = 0$ i $b = \frac{1}{2}$. To znači da linearni model ne daje dobru procjenu XOR funkcije, jer uvijek daje izlaz 0.5 za sve ulazne vrijednosti.
Jedan od načina da riješimo ovaj problem je korištenje višeslojne feedforward mreže s jednim skrivenim slojem koji sadrži dva neurona.
Feedforward mreža može se opisati sljedećim izrazom:

$$a^{(1)} = \sigma(W^{(1)} x + b^{(1)}),$$

gdje je $W^{(1)}$ matrica težina, $b^{(1)}$ vektor bias vrijednosti, σ aktivacijska funkcija, a $a^{(1)}$ aktivacije skrivenog sloja. Izlazni sloj definiran je kao:

$$y = W^{(2)} a^{(1)} + b^{(2)}.$$

Za rješavanje XOR problema koristimo ReLU aktivacijsku funkciju $\sigma(z) = \max(0, z)$. Potpuni model tada postaje:

$$f(x) = W^{(2)} \max\{0, W^{(1)} x + b^{(1)}\} + b^{(2)}.$$

Minimizacijom empirijskog rizika dobivamo optimalne parametre:

$$\hat{W}^{(1)} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \quad \hat{b}^{(1)} = \begin{bmatrix} 0 \\ -1 \end{bmatrix},$$

Ulazni podaci su:

$$X = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}.$$

Prvi korak je izračunavanje predaktivacija u skrivenom sloju:

$$\hat{z}^{(1)} = \hat{W}^{(1)} X + \hat{b}^{(1)} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix}.$$

Primjenom ReLU aktivacijske funkcije na $z^{(1)}$, dobijemo aktivacije skrivenog sloja:

$$\hat{a}^{(1)} = \max\{0, \hat{z}^{(1)}\} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix}.$$

Konačno, izlaz mreže računa se množenjem aktivacija skrivenog sloja s težinama izlaznog sloja:

$$\hat{y} = \hat{W}^{(2)}\hat{a}^{(1)} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}.$$

Ovaj rezultat pokazuje da je mreža uspješno procijenila XOR funkciju te dala točne odgovore za svaki primjer u skupu podataka.
U praksi se koristi algoritam gradijentnog spusta kako bi se pronašli parametri koji minimiziraju grešku. Iako rješenje koje smo opisali predstavlja globalni minimum funkcije gubitka, postoje i druga ekvivalentna rješenja koja gradijentni spust može pronaći, ovisno o početnim vrijednostima parametara. U stvarnim primjenama, gradijentni spust ne pronalazi uvijek jednostavna rješenja s cjelobrojnim vrijednostima, ali postiže vrlo malu grešku. Gradijentni spust jedna je od metoda prilagođavanja parametara neuronske mreže u svrhu predviđanja izlaza. Taj proces prilagođavanja naziva se treniranje, koje ćemo detaljnije objasniti u sljedećem poglavlju.

2.5 Treniranje neuronskih mreža

Treniranje neuronskih mreža je proces u kojem se model neuronske mreže prilagođava na temelju dostupnih podataka kako bi mogao pravilno predviđati ili klasificirati nove ulaze. Ovaj proces uključuje prikupljanje podataka, inicijalizaciju težina, propagaciju unaprijed, izračunavanje gubitka, propagaciju unatrag te ažuriranje težina. Postoji mnogo različitih algoritama optimizacije koji se koriste za treniranje neuronskih mreža, a neki od najpoznatijih su gradijentni spust, propagacija unatrag i RMSprop.

2.5.1 Gradijentna metoda

Gradijentna metoda je osnovni algoritam za optimizaciju koji se koristi pri treniranju neuronskih mreža. Temelji se na traženju smjera u kojem funkcija najbrže opada, a taj smjer je određen gradijentom funkcije gubitka. Gradijent funkcije $f(x)$ definiran je kao vektor parcijalnih derivacija te pokazuje smjer najvećeg porasta funkcije. To je zato što parcijalne derivacije pokazuju kako se funkcija mijenja po svakoj varijabli, a gradijent kombinira te promjene u smjer u kojem funkcija najbrže raste. Kako bismo smanjili vrijednost funkcije gubitka, krećemo se u smjeru

negativnog gradijenta.
Jednostavan izraz za prilagodbu parametara u gradijentnom spustu je:

$$\theta_{t+1} = \theta_t - \epsilon \nabla_\theta \ell(\theta_t),$$

gdje je ϵ brzina učenja, a $\nabla_\theta \ell(\theta_t)$ gradijent funkcije gubitka $\ell(\theta)$ u trenutnom koraku t.
Važno je odabrati odgovarajuću brzinu učenja ϵ. Ako je ϵ prevelik, koraci će biti preveliki. Korak se odnosi na veličinu promjene u parametrima. Ako je ϵ prevelik, koraci će biti preveliki, što znači da će parametri preskočiti točke koje bi vodile do minimuma, uzrokujući nestabilnost i moguće divergiranje funkcije gubitka. S druge strane, premala brzina učenja može usporiti konvergenciju i onemogućiti modelu da brzo dosegne optimalno rješenje. Ponekad se koristi strategija *pretrage linije* (eng. *line search*), gdje se testiraju različite vrijednosti ϵ kako bi se odabrala ona koja najviše smanjuje funkciju gubitka.

2.5.2 Propagacija unatrag

Propagacija unatrag (eng. *backpropagation*) se temelji na efikasnom izračunu gradijenata funkcije gubitka s obzirom na težine mreže, omogućujući prilagodbu parametara mreže korištenjem gradijentnog spusta. Cilj ovog algoritma je prilagoditi težine neuronske mreže tako da minimiziraju funkciju gubitka, odnosno da izlaz neuronske mreže što bolje aproksimira stvarne vrijednosti.
Prilikom izračuna izlaznih vrijednosti, podaci se propagiraju unaprijed kroz mrežu, prolazeći kroz skrivene slojeve do izlaznog sloja. Zatim se u procesu propagacije unatrag izračunavaju gradijenti funkcije gubitka po svim težinama mreže, počevši od izlaznog sloja prema ulaznom. Na taj način algoritam prilagođava težine na temelju dobivenih gradijenata kako bi se smanjila greška modela.
Pretpostavimo da imamo kvadratnu funkciju gubitka definiranu kao:

$$\ell(\theta) = \frac{1}{2n} \sum_{i=1}^{n} \left(y_i - f(x_i; \theta)\right)^2,$$

gdje je y_i stvarna vrijednost, dok je $f(x_i; \theta)$ predikcija mreže za ulaz x_i i parametre θ, a n je broj uzoraka u skupu podataka. Faktor $\frac{1}{2}$ je dodan radi jednostavnosti prilikom deriviranja.
Za svaki podatak, zadatak je izračunati parcijalne derivacije funkcije gubitka u odnosu na sve parametre mreže $w_{ij}^{(l)}$ i $b_j^{(l)}$. Proces počinje računanjem gradijenata u izlaznom sloju, a zatim se gradijenti propagiraju unatrag kroz mrežu.
Koraci algoritma propagacije unatrag:

- **Propagacija unaprijed**: Izračunavaju se predikcije mreže $f(x_i; \theta)$ za svaki ulazni podatak, propagirajući ulaze kroz sve slojeve mreže do izlaza.
- **Izračun pogreške**: Na izlazu mreže, pogreška se računa kao razlika između stvarne vrijednosti y_i i predikcije mreže $a^{(L+1)}(x_i)$. Za kvadratnu funkciju

gubitka, parcijalne derivacije funkcije gubitka u odnosu na izlaznu aktivaciju su:

$$\delta^{(L+1)} = \left(a^{(L+1)}(x) - y\right) \cdot \sigma'\left(z^{(L+1)}(x)\right),$$

gdje je $\sigma'(z)$ derivacija aktivacijske funkcije.

- **Izračun gradijenata**: Gradijenti se izračunavaju počevši od izlaznog sloja prema ulaznom sloju. Za svaki sloj l, gradijenti se računaju kao:

$$\delta^{(l)} = \left(W^{(l+1)}\right)^{\top} \delta^{(l+1)} \cdot \sigma'\left(z^{(l)}(x)\right).$$

- **Prilagodba težina**: Nakon što su izračunati gradijenti, težine i bias vrijednosti se prilagođavaju korištenjem gradijentnog spusta. Parcijalne derivacije funkcije gubitka u odnosu na težine i bias vrijednosti su:

$$\frac{\partial C}{\partial w_{ij}^{(l)}} = \delta_i^{(l)} a_j^{(l-1)},$$

$$\frac{\partial C}{\partial b_i^{(l)}} = \delta_i^{(l)}.$$

Ove formule omogućuju iterativnu prilagodbu težina i bias vrijednosti u mreži tako da se funkcija gubitka minimizira.
Propagacija unatrag je učinkovita kada treniramo mrežu koristeći jedan uzorak $(n = 1)$ jer se svi potrebni gradijenti mogu izračunati u jednom prolazu unatrag kroz mrežu, čime je broj operacija jednak broju operacija za prolaz unaprijed. Međutim, u praksi se najčešće koristi cijeli skup podataka $(n > 1)$, pri čemu prolaz unatrag mora biti ponovljen za svaki uzorak ili za manji podskup podataka. To povećava računske zahtjeve, zbog čega se u praksi često primjenjuju metode poput stohastičkog gradijentnog spusta ili mini-batch gradijentnog spusta, kako bi se optimizirala prilagodba parametara temeljem manjih podskupova podataka.

2.5.3 Stohastička gradijentna metoda

Stohastički gradijentni spust (SGD) je proširenje klasičnog algoritma gradijentnog spusta i njegova glavna prednost je efikasnost pri radu s velikim skupovima podataka. Veliki skupovi podataka su potrebni za dobru generalizaciju modela, ali treniranje na takvim skupovima može biti računalno zahtjevno.
Umjesto korištenja cijelog skupa podataka za svaku iteraciju, SGD ažurira parametre na temelju manjih podskupova, poznatih kao *mini-batch*.
Neka je $I = \{i_1, \ldots, i_m\}$ mini-batch, odnosno manji skup podataka koji se bira slučajno iz cijelog skupa podataka, a funkcija gubitka za svaki podatak (x_i, y_i) definirana je kao:

$$\ell_i(\theta) = (y_i - f(x_i; \theta))^2.$$

Na temelju ovog skupa, iteracije gradijentne metode u mini-batch backpropagation algoritmu možemo definirati kao:

$$\theta_{k+1} = \theta_k - \alpha_k \frac{1}{m} \sum_{i \in I} \nabla \ell_i(\theta_k), \quad k = 0, 1, \ldots$$

Ovdje je α_k brzina učenja (learning rate), dok je m broj uzoraka u mini-batchu.

2.5.4 RMSProp

RMSProp algoritam, koji je predložio Hinton (2012), predstavlja poboljšanje AdaGrad[4] algoritma u nekonveksnim prostorima. Dok je AdaGrad efikasan u konveksnim problemima, kod nekonveksnih problema, poput treniranja neuronskih mreža, može doći do prebrzog smanjenja brzine učenja jer uzima u obzir cijelu povijest kvadrata gradijenata. To može rezultirati premalim koracima u kasnijim fazama treniranja, što usporava konvergenciju.
RMSProp uvodi eksponencijalno ponderirani pomični prosjek, čime se zaboravlja povijest prevelikih gradijenata te se omogućuje brža konvergencija u lokalnim konveksnim regijama. Algoritam uključuje novu hiperparametarsku vrijednost ρ, koja kontrolira duljinu tog pomičnog prosjeka. Standardni oblik RMSProp algoritma može se opisati sljedećim koracima:

$$r_t = \rho r_{t-1} + (1 - \rho) g_t^2,$$
$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{r_t + \epsilon}} g_t,$$

gdje je g_t gradijent funkcije cilja, r_t eksponencijalno ponderirani prosjek kvadrata gradijenata, η brzina učenja, a ϵ mala konstanta za numeričku stabilnost. Empirijski, RMSProp se pokazao kao jedan od najefikasnijih algoritama za treniranje dubokih neuronskih mreža.

2.5.5 Regularizacija

U strojnom učenju, jedan od glavnih izazova je osigurati da model ne samo da dobro radi na podacima za treniranje, već i na novim, neviđenim podacima. Ovaj proces poznat je kao generalizacija, dok regularizacija obuhvaća niz metoda koje su osmišljene kako bi se smanjila testna pogreška, često uz blago povećanje pogreške na trening skupu. Cilj regularizacije je spriječiti prekomjerno prilagođavanje (eng. *overfitting*), odnosno situaciju u kojoj model postaje previše složen i počinje "učiti napamet" trening podatke, umjesto da generalizira na općenite obrasce.
Postoje mnoge strategije regularizacije koje se koriste u dubokom učenju, uključujući L_2 regularizaciju, dropout, i early stopping. Svaka od ovih tehnika ima svoj način na koji smanjuje složenost modela i poboljšava njegovu sposobnost generalizacije, a u nastavku ćemo detaljnije objasniti neke od njih.

[4] AdaGrad algoritam prilagođava stope učenja svakog parametra modela individualno, skalirajući ih obrnuto proporcionalno kvadratnom korijenu zbroja svih njihovih povijesnih kvadriranih vrijednosti. [2]

2.5.6 L_2 Regularizacija Parametara (Weight Decay)

L_2 regularizacija, poznata i kao "weight decay" dodaje penalizaciju u funkciju cilja modela koja potiče model da zadrži male vrijednosti težina, odnosno da ih drži blizu nuli.
Kada primjenjujemo L_2 regularizaciju, mijenjamo funkciju cilja dodavanjem regularizacijskog člana:

$$\Omega(\theta) = \frac{1}{2}\|w\|_2^2.$$

Ovaj član penalizira velike vrijednosti težina i tjera ih prema nuli. U drugim područjima, L_2 regularizacija je poznata i kao *ridge* regresija ili Tihonovljeva regularizacija. Razmotrimo model s težinama w i funkciju cilja $C(w; X, y)$, gdje X predstavlja ulazne podatke, a y ciljne vrijednosti. Nakon dodavanja regularizacijskog člana, nova ciljna funkcija postaje:

$$\tilde{C}(w; X, y) = \frac{\alpha}{2} w^\top w + C(w; X, y).$$

Parametar α kontrolira intenzitet regularizacije. Što je α veći, to su veće težine više penalizirane, tj. gurnute bliže prema nuli. Glavni učinak L_2 regularizacije je smanjivanje veličine težina koje ne doprinose značajno poboljšanju performansi modela. U smjerovima gdje je funkcija cilja osjetljiva, L_2 regularizacija ima mali učinak, dok u smjerovima gdje promjene težina ne poboljšavaju model značajno, te težine se povlače prema nuli.

Dropout

Dropout metoda nasumično "isključuje" određeni postotak neurona u slojevima tijekom treniranja, čime se sprječava da mreža postane previše ovisna o pojedinim neuronima ili obrascima u podacima. Time se mreža prisiljava na učenje robusnijih značajki koje generaliziraju bolje na nove podatke.
Dropout trenira mrežu uzimajući podskupove neurona iz izvornog modela. Na taj način, dropout trenira sve moguće podmreže temeljne mreže, ali bez potrebe da eksplicitno trenira svaku podmrežu pojedinačno, što bi bilo računalno neizvedivo. Tijekom svake iteracije treniranja, za svaki neuron se nasumično odlučuje hoće li biti uključen u mrežu ili ne, prema unaprijed određenoj vjerojatnosti (obično 0.5 za skrivene slojeve i 0.8 za ulazne slojeve). Ovaj proces se ponavlja prilikom svakog prolaska kroz podatke.
Dropout se može smatrati aproksimacijom treniranja ansambla mreža, gdje svaka podmreža igra ulogu zasebnog modela u ansamblu. Na kraju treniranja, koristi se cijela mreža bez "isključivanja" neurona, a predikcije se dobivaju prosječnim doprinosom svih neurona, što se može smatrati približnom geometrijskom sredinom svih podmreža.

2.5.7 Validacija i testiranje modela

U treniranju neuronskih mreža, osim treniranja samih parametara modela, važnu ulogu igraju i hiperparametri, poput brzine učenja, broja slojeva ili regulariza-

cijskih parametara. Hiperparametri nisu direktno prilagođeni tijekom treniranja, već ih je potrebno postaviti ručno ili pomoću validacijskog skupa podataka.
Kako bi se izbjegao overfitting na trening skupu, model se validira na posebnom skupu podataka, poznatom kao *validacijski skup*. Validacijski skup se koristi za podešavanje hiperparametara bez utjecaja na parametre treniranja, dok se *testni skup* koristi za procjenu konačne generalizacijske sposobnosti modela na neviđenim podacima. Ova dva skupa se nikada ne smiju preklapati kako bi procjene generalizacijske sposobnosti bile što preciznije.
Nakon što su svi hiperparametri optimizirani, generalizacijska pogreška modela procjenjuje se na testnom skupu. Međutim, testni skup se ne smije koristiti tijekom treniranja modela, jer to može dovesti do optimističnih procjena performansi modela, poznato kao "curenje podataka" (eng. *data leakage*).
Kako bi se optimizacija hiperparametara učinila pouzdanijom, često se koristi tehnika kros-validacija (eng. *cross-validation*). Kros-validacija omogućuje iskorištavanje svih podataka za procjenu performansi modela. Najčešća varijanta ove metode je *k-fold cross-validation*, gdje se podaci dijele na k disjunktnih podskupova. Tijekom svake iteracije, jedan podskup se koristi kao validacijski skup, dok se preostali podaci koriste za treniranje. Na kraju se pogreška procjenjuje prosjekom rezultata svih k iteracija, što smanjuje varijabilnost u procjenama. Ova tehnika je osobito korisna kod manjih skupova podataka jer omogućuje pouzdaniju procjenu performansi modela.

3 | Složene neuronske mreže

U ovom poglavlju istražujemo naprednije arhitekture neuronskih mreža koje omogućuju rješavanje složenijih zadataka u području strojnog učenja. Fokusirat ćemo se na konvolucijske neuronske mreže (CNN) i rekurentne neuronske mreže (RNN) koje su se pokazale izuzetno uspješnima u obradi podataka s poznatom mrežastom strukturom, poput slikovnih i vremenskih nizova.

3.1 Konvolucijske neuronske mreže

Konvolucijske neuronske mreže (CNN) su vrsta neuronskih mreža razvijenih za klasifikaciju slika u velikim bazama podataka. CNN-ovi pokušavaju oponašati način na koji ljudi prepoznaju slike prepoznajući određene značajke ili obrasce na slikama koji su karakteristični za pojedine objekte. CNN najprije identificira osnovne značajke slike, poput rubova ili malih područja boje. Te osnovne značajke se zatim kombiniraju kako bi se prepoznale složenije značajke, poput dijelova uha ili oka. Na kraju, prisutnost tih složenih značajki doprinosi prepoznavanju objekta, odnosno klase kojoj slika pripada.

Slojevi CNN-a

CNN koristi dvije vrste slojeva:

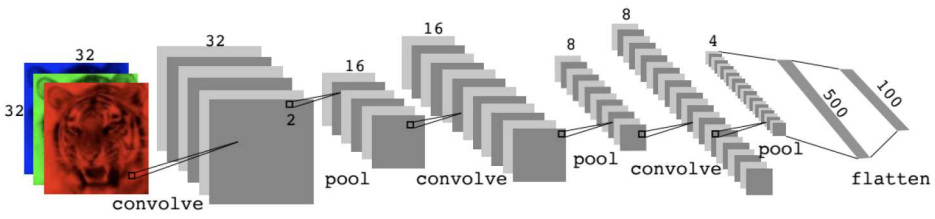
- **Konvolucijski slojevi**: Ovi slojevi koriste konvolucijske filtere koji pretražuju male uzorke na slici. Svaki filter proizvodi novu dvodimenzionalnu mapu značajki (eng. *feature map*), a broj filtera određuje broj kanala u toj mapi. Primjerice, ulazna slika od 32×32 piksela u boji predstavlja trodimenzionalnu mapu značajki, gdje svaki kanal odgovara jednoj boji (crvena, zelena, plava). Nakon konvolucijskog sloja s k filtera, imamo novu sliku s k kanala.
- **Slojevi za sažimanje (eng. *pooling*)**: Ovi slojevi smanjuju dimenzije slike, a najčešće korištena metoda je max pooling. Max pooling uzima najveću vrijednost unutar svakog nepokrivajućeg bloka od 2×2 piksela, smanjujući tako veličinu slike za faktor dva u svakom smjeru. Evo jednostavnog primjera max poolinga:

$$\text{Max pool} \begin{bmatrix} 7 & 4 & 9 & 2 \\ 3 & 1 & 8 & 5 \\ 6 & 2 & 4 & 7 \\ 3 & 9 & 1 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 7 & 9 \\ 9 & 7 \end{bmatrix}$$

Ovaj postupak konvolucije i sažimanja ponavlja se kroz više slojeva mreže. Kako se dimenzije slike smanjuju, obično se povećava broj konvolucijskih filtera u sljedećim slojevima kako bi se nadoknadio gubitak informacija. Proces se nastavlja sve dok slike ne postanu dovoljno male za potpuno povezane slojeve.

Arhitektura CNN-a

Tipična arhitektura CNN-a, kao što je prikazano na slici 3.1, uključuje izmjenične konvolucijske i pooling slojeve. Nakon nekoliko takvih slojeva, trodimenzionalna mapa značajki "spljoštava" se u vektor koji se zatim prosljeđuje u potpuno povezane slojeve. Konačni izlazni sloj koristi softmax funkciju za klasifikaciju u više klasa. Na primjeru iz paketa podataka CIFAR100, ulazna slika prolazi kroz nekoliko konvolucijskih i pooling slojeva prije nego što se klasificira u jednu od 100 klasa.



Slika 3.1: Arhitektura CNN-a za klasifikacijski problem CIFAR100. Konvolucijski slojevi se izmjenjuju sa slojevima maksimalnog grupiranja (max-pooling) veličine 2×2, koji prepolovljuju veličinu u obje dimenzije. [6]

Primjer 3.1.1. ***Primjer rada konvolucijske neuronske mreže*** *(preuzeto iz [7])*

Koraci u radu CNN-a:

1. ***Ulazni sloj:*** *CNN prima sliku dimenzija 224x224 piksela u RGB formatu. Prije obrade, od svake slike oduzima se prosječna RGB vrijednost iz skupa za treniranje radi normalizacije.*

2. ***Konvolucijski slojevi:***

- *Slika prolazi kroz niz konvolucijskih slojeva s malim filterima veličine 3×3 piksela. Ovi filteri klize preko slike i prepoznaju jednostavne značajke poput rubova, kutova i tekstura.*

- *U ranijim slojevima prepoznaju se osnovni uzorci (npr. rubovi), dok dublji slojevi prepoznaju složenije uzorke (npr. oblici i objekti).*

3. ***Aktivacijska funkcija (ReLU)***: *Nakon svake konvolucije primjenjuje se nelinearna funkcija ReLU (Rectified Linear Unit), koja omogućuje mreži učenje složenijih uzoraka.*

4. ***Pooling slojevi***: *Nakon određenih konvolucijskih slojeva primjenjuje se "max-pooling" koji smanjuje dimenzije slike uz očuvanje značajki. Pooling se obično izvodi na prozoru veličine* 2×2 *piksela.*

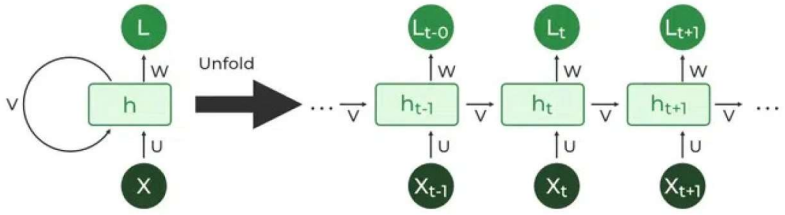
5. ***Potpuno povezani slojevi***: *Nakon prolaska kroz sve konvolucijske slojeve, slika se spljošti u vektor. Taj vektor ulazi u potpuno povezane slojeve koji kombiniraju značajke i donose odluku o klasifikaciji.*

6. ***Izlazni sloj***: *Posljednji sloj koristi softmax funkciju za klasifikaciju slike u jednu od mogućih klasa (npr. 1000 klasa).*

Zaključak: *Arhitektura CNN-a koristi male filtere od* 3×3 *piksela kako bi zadržala ključne značajke slike i smanjila broj parametara, čime je model učinkovitiji i precizniji.*

3.2 Rekurentne neuronske mreže

Rekurentne neuronske mreže (RNN) koriste se za obradu podataka koji dolaze u obliku niza, kao što su tekst, vremenski nizovi ili zvučni zapisi. Glavna prednost RNN-a je u tome što uzima u obzir prethodne elemente niza kako bi prilagodio trenutni element, omogućujući mreži da "pamti" informacije iz prošlosti.



Slika 3.2: Arhitektura rekurentne neuronske mreže (RNN). Ulazne vrijednosti X_t, X_{t-1}, itd. prolaze kroz sloj mreže koji ima povratnu petlju. Trenutno stanje mreže h_t se računa koristeći prethodno stanje h_{t-1} i trenutni ulaz X_t, dok se rezultat prosljeđuje dalje kroz mrežu. Težine U i W povezuju ulaz i povratne informacije sa stanjem mreže. [9]

Arhitektura RNN-a

Temeljna jedinica RNN-a uključuje skriveni sloj koji ima povratnu petlju. U standardnom feedforward modelu, informacije teku samo u jednom smjeru, od ulaza prema izlazu, dok kod RNN-a podaci također teku i unatrag putem povratne veze.

Ova povratna veza omogućuje RNN-u da „pamti“ informacije iz prethodnih ulaznih vrijednosti. To se postiže primjenom rekurzivne funkcije na svaki element niza podataka. Matematički, skriveni sloj se ažurira prema formuli:

$$h_t = f(W \cdot x_t + U \cdot h_{t-1} + b),$$

gdje je:

- x_t trenutni ulaz,
- h_{t-1} prethodno stanje skrivenog sloja,
- W i U su težinske matrice,
- b je pomak (bias),
- f je aktivacijska funkcija, obično *tanh* ili *ReLU*.

Na taj način, mreža koristi stanje h_{t-1} kako bi uključila kontekst prethodnih koraka u obradu trenutnog ulaza x_t.

Vrste rekurentnih neuronskih mreža (RNN)

Postoje četiri vrste RNN-ova, ovisno o broju ulaza i izlaza u mreži:

- **Jedan prema jedan (eng. *One-to-One*)**
  Ponaša se isto kao i obična neuronska mreža, ima samo jedan ulaz i jedan izlaz.
- **Jedan prema više (eng. *One-to-Many*)**
  U ovoj vrsti postoji jedan ulaz, ali više izlaza. Primjer korištenja je generiranje opisa slike (Image Captioning), gdje mreža dobiva sliku kao ulaz i generira cijelu rečenicu kao izlaz.
- **Više prema jedan (eng. *Many-to-One*)**
  Ovdje mreža prima više ulaza kroz vrijeme, ali daje samo jedan izlaz. Primjer je analiza sentimenta, gdje mreža prima više riječi i daje jedan izlaz – poput emocionalnog tona rečenice (pozitivan/negativan).
- **Više prema više (eng. *Many-to-Many*)**
  U ovoj vrsti mreže postoje više ulaza i više izlaza. Primjer je prevođenje jezika, gdje mreža prima više riječi na jednom jeziku i generira više riječi na drugom jeziku kao izlaz.

Problem eksplozije i nestajanja gradijenta

Jedan od glavnih izazova u treniranju RNN-a je problem eksplozije i nestajanja gradijenta, pri čemu gradijenti postaju vrlo mali, što dovodi do zanemarivih promjena u parametrima, zbog čega model ima poteškoća s učenjem dugoročnih ovisnosti. Tijekom propagacije unatrag, gradijenti postaju vrlo mali ili vrlo veliki, posebno kod dugih nizova. To može onemogućiti pravilno treniranje mreže i ažuriranje težina. Kako bi se ublažio ovaj problem, razvijene su varijante RNN-ova,

kao što su LSTM (Long Short-Term Memory) i GRU (Gated Recurrent Unit), koje bolje upravljaju prijenosom informacija kroz duže nizove.

3.2.1 LSTM (Long Short-Term Memory)

LSTM je vrsta rekurentne neuronske mreže koja je posebno dizajnirana za rješavanje problema s pamćenjem dugoročnih ovisnosti u podacima. Tradicionalne RNN mreže imaju problema s učenjem takvih dugoročnih veza zbog postupnog gubitka informacija kako podaci prolaze kroz mrežu. LSTM rješava taj problem uvodeći posebne jedinice zvane memorijske ćelije, koje mogu zadržati informacije kroz dulje vremensko razdoblje.
LSTM koristi tri ključna elementa nazvana "vrata" koja kontroliraju protok informacija:

- **Ulazna vrata** (eng. input gate) odlučuju koje nove informacije ulaze u memorijsku ćeliju. Memorijska ćelija je posebna jedinica dizajnirana da efikasnije upravlja dugoročnim ovisnostima, omogućujući mreži da zadrži važne informacije kroz niz vremenskih koraka.
- **Zaboravna vrata** (eng. forget gate) odlučuju koje informacije iz memorijske ćelije trebaju biti zaboravljene.
- **Izlazna vrata** (eng. output gate) kontroliraju koje informacije iz memorijske ćelije će biti poslane kao izlaz mreže.

Ova vrata omogućuju LSTM-u da učinkovito upravlja informacijama kroz vrijeme, zadržavajući relevantne podatke i eliminirajući one koje više nisu važne. Na taj način LSTM mreže mogu učiti dugoročne uzorke i veze u podacima.

LSTM arhitektura

Arhitektura LSTM-a se temelji na lancu jedinica koje sadrže ove memorijske ćelije i vrata. Svaka jedinica u lancu obrađuje podatke kroz vrijeme, čuvajući važne informacije i koristeći ih za donošenje odluka.

Primjer 3.2.1. ***Analiza mišljenja pomoću LSTM RNN-a na IMDb filmskim recenzijama*** *(preuzeto iz [6])*
U ovom primjeru treniramo rekurentnu neuronsku mrežu (LSTM) kako bismo klasificirali filmske recenzije na temelju njihove sentimentalne vrijednosti (pozitivne ili negativne). Podaci dolaze iz IMDb baze podataka filmskih recenzija.
Prvo se procjenjuje duljina svake recenzije. Više od 91% recenzija ima manje od 500 riječi, pa su sve recenzije ograničene na 500 riječi. Kraće recenzije popunjavaju se prazninama na početku. Svaka riječ iz recenzije pretvorena je u broj koji predstavlja njezinu poziciju u rječniku od 10.000 najčešćih riječi.
Prvi sloj modela je sloj za ugrađivanje riječi (eng. embedding) koji kodira svaku riječ u vektore dimenzije 32. Zatim dolazi LSTM sloj s 32 jedinice, koji modelira redoslijed riječi u recenzijama. Na kraju, izlazni sloj koristi sigmoidnu funkciju aktivacije za binarnu klasifikaciju (pozitivna ili negativna recenzija).

Model se trenira s optimizatorom rmsprop i funkcijom gubitka binary crossentropy. Tijekom treniranja prati se točnost modela na testnim podacima, a model postiže 87% točnosti na testnim podacima.

4 | Pogled u blackbox

Model crne kutije (blackbox) označava sustav koji ne otkriva svoje unutarnje mehanizme. U strojnom učenju, ovaj pojam se koristi za modele čiji su parametri i način donošenja odluka teško razumljivi.
Iz matematičke perspektive, neuronske mreže su često blackbox iz nekoliko razloga:

- **Složenost modela**: U neuronskoj mreži svaki neuron obrađuje linearne kombinacije ulaza i prolazi kroz nelinearne aktivacijske funkcije. Ova kombinacija linearnosti i nelinearnosti, multiplicirana kroz desetke ili stotine slojeva, čini vrlo teško praćenje kako pojedinačne promjene u ulaznim podacima utječu na konačni rezultat.
- **Veliki broj parametara**: Duboke neuronske mreže mogu imati milijune, pa čak i milijarde parametara (težina). Ovaj ogroman broj parametara znači da nije praktično niti jednostavno dešifrirati kako svaki od njih doprinosi konačnom izlazu, pa su veze između ulaza i izlaza izuzetno složene.
- **Nelinearnost**: Aktivacijske funkcije poput ReLU-a, sigmoidnih ili tanh funkcija uvode nelinearnost u sustav, čineći matematičku analizu mreže puno složenijom. Zbog ove nelinearnosti, mreže su osjetljive na male promjene u ulaznim podacima, a predviđanje njihovog ponašanja bez izravnog testiranja postaje gotovo nemoguće.
- **Trening modela**: U procesu treniranja, mreža koristi algoritme poput gradijentnog spusta da minimizira funkciju gubitka. Međutim, način na koji mreža "uči" nije nužno intuitivan ili transparentan – postoji mnogo lokalnih minimuma, a mreža može konvergirati u rješenja koja se teško interpretiraju.

Međutim, postoji niz alata i tehnika koji se koriste za interpretaciju dubokih neuronskih mreža i smanjenje problema crne kutije. Ovi alati pomažu u razumijevanju kako model dolazi do svojih odluka, bilo analizom značajki, težina, slojeva ili izlaza modela. U nastavku ćemo objasniti jedan od njih.

LIME (Local Interpretable Model-agnostic Explanations)

LIME je metoda koja objašnjava predikcije modela tako što generira lokalno razumljive, linearnije modele oko svake pojedinačne predikcije. Umjesto da pokušava objasniti cijeli model, LIME se fokusira na objašnjavanje pojedinačnih odluka

modela.
LIME radi na sljedeći način:

1. **Odabire se instanca:** Prvo se odabere primjer (instanca) čiju predikciju crne kutije želimo objasniti.

2. **Stvara se novi skup podataka:** Zatim se stvara skup podataka tako da se značajke (karakteristike) iz originalnog primjera malo promijene. To znači da se unose male varijacije u značajke kako bi se vidjelo kako te promjene utječu na predikciju modela.

3. **Dobivaju se predikcije crne kutije:** Model crne kutije daje predikcije za te nove, malo izmijenjene podatke.

4. **Ponderiraju se nove točke:** Te nove točke ponderiraju se (dodjeljuje im se težina) ovisno o tome koliko su blizu originalnom primjeru. Što su točke bliže originalnom primjeru, to im se daje veća težina.

5. **Treniranje jednostavnog modela:** Na tom skupu podataka, s težinama dodijeljenim prema udaljenosti, trenira se jednostavan model (npr. linearna regresija) kako bi se objasnilo kako su te promjene utjecale na predikciju.

LIME je koristan alat za objašnjavanje predikcija crnih kutija međutim, ima svojih nedostataka. Postoji mogućnosti da proizvede varijabilne rezultate, što može dovesti do različitih objašnjenja za vrlo slične primjere. Također, postoji rizik da se putem prilagodbe algoritma prikriju pristranosti modela. Ipak, metoda ima velik potencijal, ali zahtijeva daljnji razvoj kako bi bila pouzdana i primjenjiva u širokom spektru zadataka.

5 | Modeliranje neuronske mreže za procjenu kreditnog rizika

U ovom poglavlju izradit ćemo neuronsku mrežu za klasifikaciju kreditnog rizika koristeći skup podataka preuzet s platforme Kaggle [14]. Skup podataka sastoji se od 28.638 jedinki i 12 varijabli koje sadrže informacije kao što su dob klijenata, primanja, duljina zaposlenja, namjena kredita i druge relevantne značajke. Konkretno, cilj je predvidjeti hoće li klijentu biti odobren kredit ili neće. Klasifikacija se vrši pomoću binarne varijable `kredit_status`, pri čemu je vrijednost `1` dodijeljena klijentima koji su uspjeli otplatiti kredit, dok vrijednost `0` predstavlja klijente koji nisu uspjeli otplatiti kredit.

Prije modeliranja, obradili smo podatke na način da smo uklonili sve retke s nedostajućim vrijednostima. Zatim smo određene kategorijalne varijable, poput vlasništva nad nekretninom i namjene kredita, pretvorili u faktorske varijable kako bi ih model mogao ispravno interpretirati. Međutim, budući da modeli poput neuronskih mreža zahtijevaju numeričke ulaze, ove faktorske varijable dodatno smo pretvorili u *dummy* varijable. Ove *dummy* varijable pretvaraju svaku kategoriju u niz binarnih varijabli (0 ili 1) koje model može koristiti za učenje. Konkretno, varijabla vlasništvo nad nekretninom može imati vrijednosti "vlasnik", "stanar" i "ostalo". Za svaku od tih kategorija kreirana je nova dummy varijabla:

- `vlasnistvo_vlasnik`: 1 ako je osoba vlasnik, 0 inače,
- `vlasnistvo_stanar`: 1 ako je osoba stanar, 0 inače,
- `vlasnistvo_ostalo`: 1 ako osoba spada u kategoriju "ostalo", 0 inače.

Osim toga, numeričke varijable su normalizirane skaliranjem kako bi se osiguralo učinkovitije učenje modela.

Kako bismo trenirali neuronsku mrežu, podijelili smo podatke na trening i test skupove. Trening skup čini 80% ukupnih podataka, dok je preostalih 20% korišteno za testiranje modela.

5.1 Logistička regresija kao bazni model

Prije izrade neuronske mreže, kreirali smo model logističke regresije radi usporedbe rezultata. Model smo trenirali na istom skupu podataka, a rezultati

su evaluirani na testnom skupu podataka. Predikcije modela izražene su kao vjerojatnosti, a kako bi ih transformirali u konkretne klasifikacije, koristili smo prag od 0.5. To znači da smo vjerojatnosti veće od 0.5 klasificirali kao "odobreno" (1), a vjerojatnosti manje od 0.5 kao "neodobreno" (0).

Matrica konfuzije[1] prikazana u Tablici 5.1 omogućuje analizu točnosti klasifikacije modela:

	Stvarno: 0	**Stvarno: 1**
Predviđeno: 0	4300	523
Predviđeno: 1	211	694

Tablica 5.1: Matrica konfuzije: Predviđeni i stvarni ishodi odobrenja (1) i neodobrenja (0) kredita dobiveni modelom logističke regresije

Definira se na sljedeći način:

- **True Negative (TN)**: Model je ispravno predvidio da kredit neće biti odobren (0) u 4300 slučajeva.
- **False Negative (FN)**: Model je pogrešno predvidio da kredit neće biti odobren (0), iako je trebao biti odobren (1), u 523 slučajeva.
- **False Positive (FP)**: Model je pogrešno predvidio da će kredit biti odobren (1), ali nije trebao biti, u 211 slučajeva.
- **True Positive (TP)**: Model je ispravno predvidio da će kredit biti odobren (1) u 694 slučajeva.

Na temelju ove matrice, izračunata je ukupna točnost modela kao omjer točnih predikcija i ukupnog broja podataka:

$$\text{Točnost} = \frac{TN + TP}{TN + FN + FP + TP} = \frac{4300 + 694}{5728} = 87.19\%.$$

5.2 Kreiranje neuronske mreže

Za kreiranje neuronske mreže koristili smo paket `keras` u programskom jeziku R. Naš model sastoji se od tri sloja, pri čemu ulazni sloj sadrži 23 neurona, a skriveni sloj 16 neurona. Oba sloja koriste *ReLU* aktivacijsku funkciju. Izlazni sloj modela je sloj s jednim neuronom koji koristi *sigmoidnu* aktivacijsku funkciju.

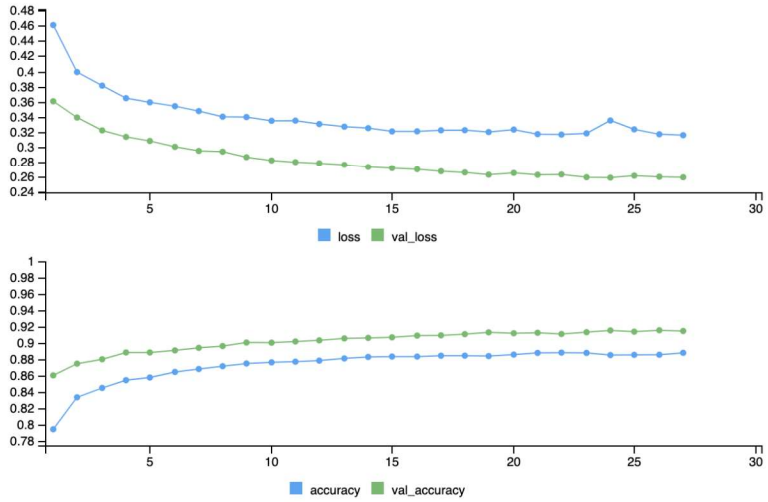
Kako bismo smanjili mogućnost overfittinga, u prvom i drugom sloju dodali smo *dropout* slojeve s omjerom izbacivanja od 50%. To znači da se nasumično izbacuje polovica neurona tijekom treniranja kako bi se poboljšala generalizacija modela.

[1] Matrica konfuzije je tablica koja prikazuje stvarne klase i predviđene klase, omogućujući uvid u performanse modela. Na primjer, u ovom slučaju, ona prikazuje koliko je kredita ispravno klasificirano kao "odobreno" ili "neodobreno", te koliko je klasifikacija bilo netočnih.

Model smo trenirali korištenjem *RMSprop* optimizatora, a za funkciju gubitka odabrali smo *binary cross-entropy*, što je standardna funkcija gubitka za binarne klasifikacijske zadatke.

Model je treniran kroz 30 epoha, pri čemu smo koristili tehniku *early stopping*, koja zaustavlja treniranje ako se gubitak na skupu za validaciju ne poboljša tijekom tri uzastopne epohe. 20% podataka iz trening skupa izdvojeno je za validaciju, dok je ostatak korišten za treniranje.

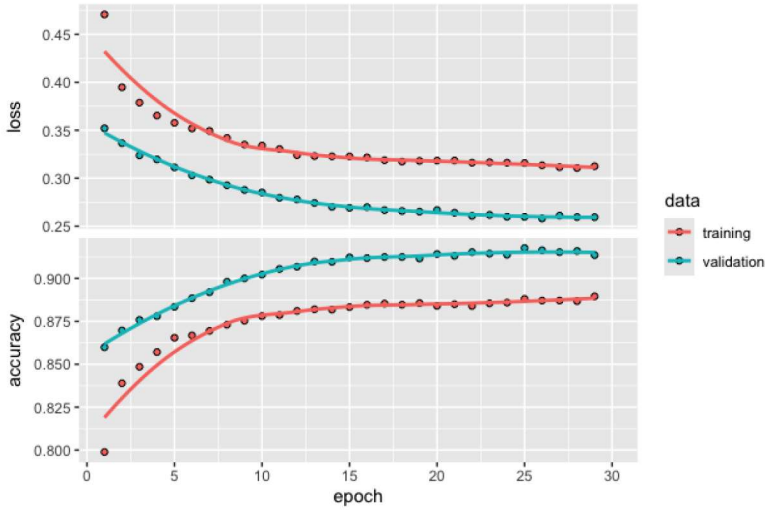
Na grafu 5.1 prikazana je točnost modela tijekom treninga i validacije. Rastuća linija točnosti na validacijskom skupu pokazuje da se model s vremenom poboljšava u prepoznavanju obrazaca na neviđenim podacima. Ukoliko bi se validacijska točnost značajno razlikovala od točnosti na trening skupu, to bi moglo ukazivati na problem overfittinga. Također, uočimo kako je model stao nakon 27. epohe zbog korištenja early stoppinga.



Slika 5.1: Graf prikazuje točnost (accuracy) tijekom treninga i validacije kroz 30 epoha.

Na grafu 5.2 prikazan je gubitak modela kroz epohe. Gubitak za trening i validacijski skup opada s brojem epoha, što je znak da se model poboljšava tijekom treniranja. U idealnom slučaju, gubitak bi trebao opadati u oba skupa podataka. Ako bi validacijski gubitak počeo rasti dok trening gubitak opada, to bi ukazivalo na overfitting.

Nakon treniranja, model smo evaluirali na testnom skupu podataka, pri čemu smo postigli točnost od 91.29%. Također, kreirali smo matricu konfuzije, Tablica 5.2 kako bismo dodatno procijenili uspješnost modela.



Slika 5.2: Graf prikazuje gubitak (loss) tijekom treninga i validacije kroz 30 epoha.

	Stvarno: 0	**Stvarno: 1**
Predviđeno: 0	4456	444
Predviđeno: 1	55	773

Tablica 5.2: Matrica konfuzije: Predviđeni i stvarni ishodi odobrenja (1) i neodobrenja (0) kredita dobiveni modelom neuronske mreže

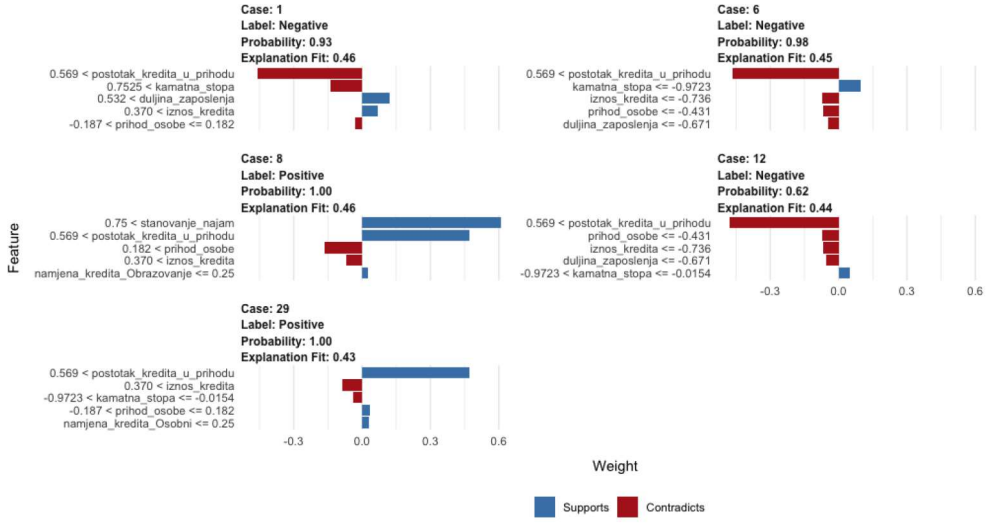
Ukoliko usporedimo model neuronske mreže s modelom logističke regresije, vidimo da neuronska mreža postiže veću točnost za 4.10%. Točnije, model neuronske mreže je ispravno klasificirao 235 klijenata više nego model logističke regresije, što pokazuje da je neuronska mreža preciznija u predviđanju odobrenja i neodobrenja kredita.

5.2.1 Korištenje LIME-a za interpretaciju rezultata

Kao što smo spomenuli u prethodnom poglavlju, problem kod neuronskih mreža može biti nedostatak interpretabilnosti. Iz tog razloga koristimo algoritam LIME, koji nam omogućava detaljniju analizu modela. Algoritam LIME stvara model za svakog pojedinog klijenta kako bi se objasnilo kako su pojedine varijable utjecale na odluku o odobrenju ili neodobrenju kredita. Na primjer, za svakog klijenta, identificira varijable poput prihoda, kamatne stope, namjene kredita i vlasništva nad nekretninom, te procjenjuje koliki je njihov doprinos konačnoj odluci.

Na slici 5.3 prikazana je važnost varijabli za različite slučajeve iz testnog skupa. Svaki panel predstavlja jedan slučaj te prikazuje varijable koje su imale najveći utjecaj na odluku modela. Sljedeće stavke opisuju informacije dobivene LIME metodom za svaki slučaj iz testnog skupa:

- **Case (Slučaj)**: Oznaka specifičnog slučaja iz testnog skupa.



Slika 5.3: Prikaz važnosti varijabli za različite slučajeve iz testnog skupa pomoću LIME-a.

- **Label (Oznaka)**: Stvarna klasa danog slučaja (ukoliko je kredit odobren label je Positive, u suprotnom Negative).
- **Probability (Vjerojatnost)**: Vjerojatnost koju model predviđa da slučaj pripada određenoj klasi. U nekim slučajevima, poput slučajeva 8 i 29, procjena vjerojatnosti je 1, što znači da je model potpuno siguran u svoju predikciju. Iako vjerojatnosti od 1 teoretski predstavljaju potpunu sigurnost, u stvarnom svijetu takve predikcije treba tumačiti oprezno zbog mogućnosti grešaka modela.
- **Explanation Fit (Prikladnost objašnjenja)**: Ovaj broj pokazuje koliko LIME-ova analiza dobro objašnjava odluku modela za dani slučaj, tj. koliko precizno LIME-ovo objašnjenje opisuje način na koji je model donio svoju odluku. Što je ovaj broj bliži 1, to je objašnjenje pouzdanije u smislu da bolje odražava zašto je model donio tu specifičnu odluku.
- **Supports/Contradicts (Podržava/Kontradiktira)**: Varijable označene plavom bojom pozitivno utječu na odluku modela (odnosno, povećavaju vjerojatnost da će model odobriti kredit), dok crveno označene varijable negativno utječu na odluku. Duljina traka prikazuje težinu utjecaja varijable na konačnu odluku.

Slučaj 1 - Pogrešna klasifikacija

Model je u slučaju 1 predvidio da klijent neće dobiti kredit s vjerojatnošću od 0.93. No, ovaj je klijent u stvarnosti dobio kredit, što znači da je model pogrešno klasificirao ovaj slučaj. Pomoću LIME metode možemo vidjeti kako su pojedine varijable utjecale na odluku modela:

- **postotak kredita u prihodu** ima značajan negativan utjecaj na odluku modela,
- **kamata stopa** također negativno utječe,
- **duljina zaposlenja** doprinosi odluci,
- **iznos kredita** je manja, ali relevantna varijabla koja također podržava odluku modela,
- **prihod osobe** imao je nešto manji negativan utjecaj.

Iako je model pokazao visoku razinu sigurnosti u svoju odluku (predviđena vjerojatnost od 0.93), LIME-ovo objašnjenje imalo je prikladnost (Explanation Fit) od 0.46, što ukazuje na slabije objašnjenje odluke za ovaj slučaj. Unatoč tome, došlo je do pogrešne klasifikacije jer je kredit bio odobren.

Slučaj 29 - Ispravna klasifikacija

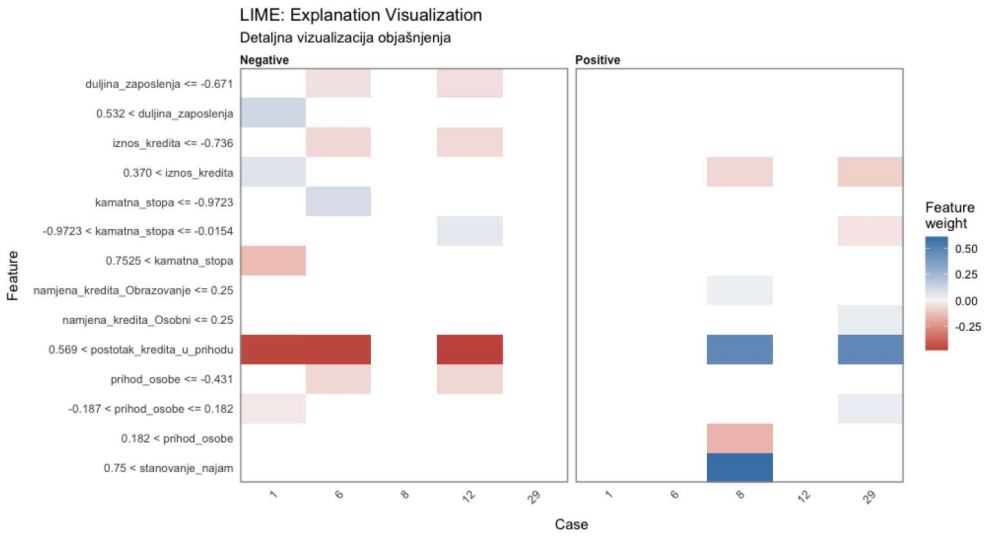
U ovom slučaju model je točno predvidio da će klijent dobiti kredit s vjerojatnošću od 1. LIME metoda pokazuje kako su određene varijable utjecale na odluku modela:

- **postotak kredita u prihodu** značajno je pridonio odluci da se kredit odobri,
- **iznos kredita** također pozitivno utječe na odluku,
- **kamata stopa** ima negativan utjecaj, ali nedovoljan da se odluka promijeni,
- **prihod osobe** ima manji negativan utjecaj,
- **namjena kredita** za "Osobni" ima pozitivan doprinos.

Iako postoje varijable koje negativno utječu na odluku, pozitivni utjecaji su prevladali, a model je donio točnu odluku. LIME-ovo objašnjenje imalo je prikladnost (Explanation Fit) od 0.43, što također ukazuje na slabije objašnjenje odluke modela za ovaj slučaj.

Slika 5.4 prikazuje detaljnu LIME vizualizaciju utjecaja varijabli na odluke modela za nekoliko slučajeva. Plave nijanse predstavljaju varijable koje podupiru odluku o odobravanju kredita (Positive), dok crvene nijanse označavaju varijable koje vode prema odbijanju kredita (Negative). Jačina boje pokazuje snagu utjecaja svake značajke na konačnu odluku.

Za primjer `Case 1`, koji smo već detaljno opisali, možemo uočiti da su varijable poput `postotak kredita u prihodu` i `kamata stopa` imale najjači negativan utjecaj na odluku modela da kredit ne bude odobren. Ostatak objašnjenja detaljno je prikazan ranije. Slika 5.4 dodatno potvrđuje kako su ove varijable doprinijele pogrešnoj klasifikaciji, gdje je model išao u prilog odluci koja nije bila točna.



Slika 5.4: Detaljna vizualizacija objašnjenja utjecaja varijabli korištenjem LIME-a.

Ova vizualizacija također pokazuje slične obrasce za ostale slučajeve. Na primjer, u `Case 29`, gdje je kredit bio ispravno odobren, vidimo da značajke poput `postotak kredita u prihodu` i `iznos kredita` imaju pozitivan utjecaj (plave nijanse), što je rezultiralo točnom odlukom modela.

Korištenjem LIME metode, dobili smo uvid u to kako pojedine varijable utječu na odluke modela o odobravanju ili neodobravanju kredita. Iako model često ispravno klasificira slučajeve, poput `Case 29`, uočili smo da postoje i pogrešne klasifikacije, kao što je slučaj `Case 1`. Vizualizacije omogućuju bolji uvid u način na koji model koristi različite varijable, ali također pokazuju da model ponekad može biti previše siguran u odluku, što rezultira pogrešnim ishodom.

Ovi primjeri naglašavaju važnost interpretabilnosti modela, pogotovo u važnim područjima poput procjene kreditnog rizika, gdje je ključno razumjeti razloge iza odluka modela kako bi se poboljšala njegova točnost i pouzdanost u praksi.

Literatura

[1] J. Duchi, E. Hazan, Y. Singer, *Adaptive Subgradient Methods for Online Learning and Stochastic Optimization*, Journal of Machine Learning Research, 12 (2011), pp. 2121-2159.

[2] I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning*, MIT Press, 2016.

[3] M. Anthony, P. L. Bartlett, *Neural Network Learning: Theoretical Foundations*, Cambridge University Press, 1999. Here's the citation formatted according to your needs:

[4] D. E. Rumelhart, G. E. Hinton, R. J. Williams, *Learning Representations by Back-Propagating Errors*, *Nature*, vol. 323, no. 6088, 1986, pp. 533-536.

[5] S. Haykin, *Neural Networks and Learning Machines*, 3rd ed., Prentice Hall, 2009.

[6] G. James, D. Witten, T. Hastie, R. Tibshirani, *An Introduction to Statistical Learning, with Applications in R (corrected June 2023)*, Springer, 2023.

[7] K. Simonyan, A. Zisserman, *Very Deep Convolutional Networks for Large-Scale Image Recognition*, Published as a conference paper at ICLR 2015, 2015.

[8] D. Grahovac, *Statističko učenje: Dijelovi predavanja*, Fakultet primijenjene matematike i informatike, Sveučilište Josipa Jurja Strossmayera u Osijeku, svibanj 2024., https://www.mathos.unios.hr/wp-content/uploads/2024/08/SU-handouts.pdf [Pristupljeno: 01. listopada 2024.].

[9] GeeksforGeeks, *Introduction to Recurrent Neural Network*, https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/, [Pristupljeno: 12. listopada 2024.].

[10] GeeksforGeeks, *Introduction to Long Short-Term Memory*, https://www.geeksforgeeks.org/deep-learning-introduction-to-long-short-term-memory/, [Pristupljeno: 12. listopada 2024.].

[11] Towards Data Science, *Why We Will Never Open Deep Learning's Black Box*, https://towardsdatascience.com/why-we-will-never-open-deep-learnings-black-box-4c27cd335118, [Pristupljeno: 13. listopada 2024.].

[12] Nerd For Tech, *Why is Deep Learning Called Black Box? Why is it a Problem?*, https://medium.com/nerd-for-tech/why-is-deep-learning-called-b

lack-box-why-is-it-a-problem-43ac3d3ec24, [Pristupljeno: 13. listopada 2024.].

[13] C. Molnar, *Interpretable Machine Learning - A Guide for Making Black Box Models Explainable*, https://christophm.github.io/interpretable-ml-book/lime.html, [Pristupljeno: 13. listopada 2024.].

[14] Laotse, *Credit Risk Dataset*, Kaggle, https://www.kaggle.com/datasets/laotse/credit-risk-dataset, (2021), [pristupljeno: 17. listopada 2024.]

Sažetak

Ovaj diplomski rad istražuje neuronske mreže i njihovu primjenu u procjeni kreditnog rizika. U radu se objašnjavaju osnovni pojmovi neuronskih mreža, počevši od arhitekture jednoslojnog i višeslojnog perceptrona, te procesa treniranja modela uz korištenje različitih optimizacijskih algoritama. Također se razmatraju složenije mreže, poput konvolucijskih i rekurentnih neuronskih mreža, kao i izazov interpretacije tih modela, poznatih kao blackbox. Rad ističe važnost tehnika interpretacije rezultata, s naglaskom na LIME metodu, koja omogućuje bolje razumijevanje odluka modela. Na kraju, izrađen je model neuronske mreže za procjenu kreditnog rizika, čija je učinkovitost analizirana i interpretirana primjenom LIME metode.

Ključne riječi

neuronske mreže, strojno učenje, perceptron, treniranje modela, optimizacija, kreditni rizik, LIME, interpretacija modela.

Neural Networks - insight into the blackbox

Summary

This thesis explores neural networks and their application in credit risk assessment. The work explains the basic concepts of neural networks, starting with the architecture of single-layer and multi-layer perceptrons, and the process of training models using various optimization algorithms. Furthermore, it also discusses more complex networks, such as convolutional and recurrent neural networks, as well as the challenge of interpreting these models, known as black box models. The thesis emphasizes the importance of result interpretation techniques, focusing on the LIME method, which allows for a better understanding of model decisions. Finally, a neural network model for credit risk assessment is developed, and its effectiveness is analyzed and interpreted using the LIME method.

Keywords

neural networks, machine learning, perceptron, model training, optimization, credit risk, LIME, model interpretation.

Životopis

Rođena sam 9. siječnja 2000. godine u Zagrebu, gdje sam završila osnovnu školu Braće Radić i XI. gimnaziju. Godine 2018. upisala sam Prirodoslovno-matematički fakultet Sveučilišta u Zagrebu, smjer Matematika - nastavnički. Nakon završetka preddiplomskog studija 2022. godine, odlučila sam upisati diplomski studij na tadašnjem Odjelu za matematiku Sveučilišta Josipa Jurja Strossmayera u Osijeku, smjer Financijska matematika i statistika (danas Fakultet primijenjene matematike i informatike).

Tijekom studija, 2023. godine, odradila sam praksu u trajanju od mjesec dana u sklopu kolegija na fakultetu, u Erste banci u sektoru financijskih tržišta. Praksa mi je omogućila stjecanje iskustva u financijskom sektoru, što je dodatno oblikovalo moj interes za primjenu matematike u financijama.