Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 621.31
DOI: https://doi.org/10.24867/18BE13Bosic

REŠAVANJE IGRE SUDOKU POMOĆU VEŠTAČKIH NEURONSKIH MREŽA
SOLVING SUDOKU USING ARTIFICIAL NEURAL NETWORKS

Jovan Bosić, *Fakultet Tehničkih nauka, Novi Sad*

Oblast – **ELEKTROTEHNIKA I RAČUNARSTVO**

Kratak sadržaj – *U ovom radu dat je pregled različitih modela veštačkih i konvolucijskih neuronskih mreža i razmatrane su prednosti upotrebe optimizacionog algoritma Adam u odnosu na neke druge optimizacione algoritme. Pomenuti modeli veštačkih neuronskih mreža primenjeni su na problem rešavanja igre Sudoku. Data je detaljna analiza obrade podataka pre obuke modela, a potom je izvršeno poređenje dobijenih rezultata.*

Ključne reči: *Veštačke neuronske mreže, konvolucijske neuronske mrerže, Adam.*

Abstract - *This paper provides an overview of different models of artificial and convolutional neural networks and discusses the advantages of using the Adam optimizer over some other optimization algorithms. Proposed ANN models will be used for solving a Sudoku game. A detailed analysis of data processing before model training is given, and a comparison of the obtained results is made.*

Keywords: *Artificial neural networks, convolutional neural networks, Adam*

1. UVOD

Cilj ovog rada jeste prikazivanje efikasnosti veštačkih neuronskih mreža (*engl. Artificial neural network - ANN*) i konvolucijskih neuronskih mreža (*engl. Convolutional neural network - CNN*) u rešavanju igre Sudoku. Teorijske osnove biće prikazane u radu, a praktična primena veštačkih neuronskih mreža implementirana je na matematičkom modelu pomoću *Colaboratory*-ja ili skraćeno *Colab* koji je proizvod kompanije *Google Research*. *Colab* omogućava bilo kome da napiše i izvrši proizvoljan *Python* kod preko pretraživača, a posebno je pogodan za mašinsko učenje, analizu podataka i generalno obrazovanje, [1].

Rad će biti grupisan po celinama, gde će prvo biti opisana problematika pripreme podataka za formiranje modela neuronski mreža, potom će biti dat kratak uvod u veštačke neuronske i konvolucijske mreže, dok će poslednja celina biti posvećena primeni pomenutih modela veštačke inteligencije na rešavanje igre sudoku.

2. PRIPREMA PODATAKA

Mnogi faktori utiču na uspeh mašinskog učenja (*engl. Machine learning*). Zastupljenost i kvalitet podataka su prvi i najvažniji. Ako postoji mnogo nebitnih i suvišnih informacija, ili šuma i nepouzdanih podataka, otkrivanje karakteristika modela tokom faze obuke predstavlja težak zadatak. Približno, obrada podataka traje 70% ukupnog vremena projekta, što pokazuje koliko je faza obrade važna za bilo koji projekat mašinskog i dubokog učenja (*engl. Deep learning*). Obrada podataka je proces pretvaranja neobrađenih podataka u validan format potreban za obuku. Uopšteno, obrada se odnosi na transformacije primenjene na podatke pre unosa u algoritam. Pre same obrade neophodno je razumevanje opisa podataka gde možemo istražiti definicije i sadržaj svih promenljivih pre rada na algoritmu mašinskog učenja. Važno je znati da kvalitet ulaznih podataka odlučuje o kvalitetu izlaznih podataka. Stoga, da bi se izbegli naočigled tačni, ali zapravo pogrešni rezultati, podaci se moraju pregledati pre nego što se unesu kao ulaz u algoritam. Obrada podataka često može imati značajan uticaj na performanse generalizacije kod algoritama mašinskog učenja pod nadzorom, kao što je istaknuto u [2]. U stvarnom svetu, podaci sadrže šum, izostavljenje vrednosti i nedosledne podatke zbog velike veličine, a takođe se i podaci prikupljaju iz skupova podataka koji su prisutni u više oblika i potiču iz više izvora. Pored navedenih osobina, podaci mogu biti i redundantni, gde su određene karakteristike podataka povezane tako da nije potrebno sve njih uključiti u modeliranje, i međuzavisnosni, gde dve ili više osobina međusobno prenose važne informacije koje su nejasne ako se bilo koja od njih uključi sama, kao što je navedeno u [3].

3. VEŠTAČKE NEURONSKE MREŽE

U radu [6] napravljen je računarski model za neuronske mreže zasnovane na matematici i algoritmima koji koriste logiku praga (*eng. Threshold Logic*). Ovaj rad je uveo prvi matematički model neuronske mreže, koristeći elektronska kola. Jedinice ovog modela, jednostavni formalizovani neuroni, još uvek su standardna referenca u polju veštačkih neuronskih mreža. Neuron ovog modela naziva se Mekuloh-Pits (*McCulloch-Pitts*) neuron.

Veštačke neuronske mreže su vrsta mašinskog učenja koje se često povezuje sa dubokim učenjem. Karakteristika duboke neuronske mreže jeste da ima dva ili više skrivenih slojeva , gde su skriveni slojevi oni koje veštačka neuronska mreža sama kontroliše. S razlogom se može reći i da je većina neuronskih mreža u upotrebi zapravo oblik dubokog učenja.

Neuronske mreže koje imaju mogućnost da propagiraju podatke unazad (*engl. Backpropagation*) najčešće su mreže

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bila dr Mirna Kapetina

koje se sastoje od nekoliko slojeva, te se te višeslojne mreže upotrebljavaju u 80 posto slučajeva zbog svoje dokazane sposobnosti učenja i generalizacije, kao što je istaknuto u [8].

3.3. Aktivacione funkcije

Različite aktivacione funkcije koriste se za različite slučajeve, a razumevanje njihovog rada je neophodno da bi pravilno izabrali koja od njih je najbolja za određeni zadatak. Aktivaciona funkcija se primenjuje na izlaz neurona (ili sloj neurona) i transformiše ga. Takođe, aktivacione funkcije se koriste jer ako je sama funkcija aktivacije nelinearna, ona omogućava neuronskim mrežama sa obično dva ili više skrivenih slojeva da mapiraju nelinearne funkcije, kao što je navedeno u [8].

Najčešće primenjivani oblici aktivacionih funkcija su rampa funkcija (*engl.The Rectified Linear Activation Function - ReLU*) i Softmaks funkcija koja se koristi za probleme klasifikacije.

3.4. Računanje greške gubitka mreže

Glavni cilj kod rešavanja problema uz pomoć mašinskog ucenja je obučiti model tokom vremena da reši određeni zadatak. Da bi se model obučio, podešavaju se vrednosti težina i drugi parametara kako bi poboljšali tačnost i pouzdanost modela. Kako bi ovo podešavanje bilo tačno, neophodno je znati grešku modela. Funkcija gubitka (*engl. Loss*), koja se takođe naziva i funkcija troškova, je mera koji kvantifikuje koliko je model pogrešan u predviđanjima.

Model poseduje softmaks aktivacionu funkciju na izlaznom sloju, što znači da kao izlaznu vrednost iz funkcije model daje raspodelu verovatnoća. Kategorična unakrsna entropija, takođe nazivana i logaritamska, eksplicitno se koristi za poređenje verovatnoća, tj. y vektora ili "očekivane, željene vrednosti", i neke predviđene distribucije ($\hat{y}$ ili "predviđanja") tj. izlaza iz softmaks funkcije, pa ima smisla koristiti unakrsnu entropiju u ovom slučaju, kao što je istaknuto u [8]. U tom slučaju gubici se računaju prema

$$L_i = -\sum_j y_{i,j} * log(\hat{y}_{i,j}), \tag{1}$$

gde je L_i vrednost gubitka određenog uzorka, indeks i je i-ti uzorak iz skupa podataka, j je indeks kod izlaznog vektora, y je željena vrednost, a $\hat{y}$ je predviđena vrednost.

3.5. Optimizacija

Kada je neuronska mreža modelovana, sposobna za prenos podataka i sposobna da izračuna gubitak, sledeći korak je da se utvrdi kako prilagoditi težine i pristrasnosti kako bi se smanjio taj gubitak. Pronalaženje adekvatnog načina za prilagođavanje težina i pristrasnosti neurona kako bi se minimalizovao gubitak, jedan je od glavnih problema neuronskih mreža.

3.5.1. Propagiranje podataka unazad

Od objavljivanja PDP tomova 1986,[10] učenje putem propagacije unazad postalo je najpopularniji metod obuke neuronskih mreža. Razlog popularnosti jeste jednostavnost i relativna moć algoritma. Njegova moć proizilazi iz činjenice da se, za razliku od svojih prethodnika, pravila učenja perceptrona i Vidrov-Hoffovog pravila učenja, može koristiti za obuku nelinearnih mreža proizvoljne povezanosti. Pošto su takve mreže često potrebne za primene u stvarnom svetu, takva procedura učenja je od velikog značaja. Osnovna ideja je stara i jednostavna; naime, definiše se funkcija gubitka i koristi se algoritam gradijentnog spusta (*engl. Gradient Descent*) da bi pronašli skup vrednosti težina koje optimizuju performanse na određenom zadatku, kao što je navedeno u [9].

Da bi se primenio pomenuti algoritam, potrebno je izračunati gradijente funkcije gubitaka po svim parametrima, koji se dalje koriste za prilagođavanje težina i pristrasnosti kako bi se gubitak smanjio, kao što je navedeno u [8].Optimizacioni algoritam, koji je prema naučnoj literaturu najčešće primenljiv u ove svrhe je modifikovana verzija osnovnog gradijentnog algoritma i naziva se adaptivni gradijentni algoritam sa momentom (*eng. Adam - Adaptive Momentum*). Adam kao metod za efikasnu stohastičku optimizaciju zahteva samo gradijente prvog reda sa malim zahtevima za memorijom. Metod izračunava individualne stope adaptivnog učenja za različite parametre iz procena prvog i drugog momenta gradijenata. Ime Adam je izvedeno iz procene adaptivnog momenta. Metod je dizajniran da kombinuje prednosti dve veoma popularne metode: AdaGrad, koji dobro funkcioniše sa gradijentima koji su retki (mreža ne prima dovoljno jak signal za podešavanje težina i pristrasnosti neurona), i RMSProp, koji dobro funkcioniše u online i nestacionarnim postavkama. Neke od Adamovih prednosti su da su veličine ažuriranja parametara invarijantne u odnosu na reskaliranje gradijenta, njegov korak izračunavanja je približno ograničen hiperparametrom, i radi sa retkim gradijentima.

Adam je algoritam za optimizaciju stohastičkih funkcija koje je potrebno minimizovati - ciljne funkcije (*engl. Objective functions*) zasnovan na gradijentu prvog reda i na adaptivnim procenama momenata nižeg reda. U matematici momenti su funkcije kvantitativne mere povezane sa oblikom grafa funkcije. Ako funkcija predstavlja masu, tada je prvi momenat centar mase, a drugi momenat rotaciona inercija. Adam je jednostavan algoritam za implementaciju, računarski je efikasan, ima male potrebe za memorijom, invarijantan je na dijagonalno skaliranje gradijenata i pogodan je za probleme koji su veliki u smislu podataka i/ili parametara. Metoda je takođe prikladna za nestacionarne ciljne funkcije i probleme sa zašumljenim i/ili retkim odnosno oskudnim gradijentima (*engl. Sparse gradients*). Hiperparametri imaju intuitivne interpretacije i obično zahtevaju malo podešavanja, kao što je navedeno u [5]. Ciljne fukcije mogu imati razne izvore šuma pored šuma nastalog tokom uzorkovanja podataka, kao što je regularizacija isključivanja neurona naznačena u radu [7]. Za rešavanje ovakvih smetnji, potrebne su efikasne tehnike stohastičke optimizacije, poput Adam-a.

4. KONVOLUCIJSKE NEURONSKE MREŽE

Neuronske mreže topologije rešetke specijalizirane za obradu podataka nazivaju se "konvolucijske neuronske mreže"(*engl. Convolutional neural network – CNN*). Predstavljaju jednu od popularnijih kategorija neuronskih mreža, posebno za rad s višedimenzionalnim podacima. Funkcionišu poput standardnih veštačkih neuronskih mreža, uz razliku da je svaka jedinica pojedinog sloja konvolucij-

ske mreže (minimalno) dvodimenzionalni filter koji obavlja operaciju konvolucije. Ovo je presudno za prepoznavanje uzoraka kod ulaznog slikovnog ili video materijala, kao što je istaknuto u [4]. Sami filteri imaju sličan, ali manji prostorni oblik u odnosu na ulazni medij te koriste zajedničke parametre radi smanjenja broja promenljivih za učenje.

4.1. Slojevi mreže

Konvolucijska neuronska mreža sadrži nekoliko gradivnih elemenata koji se nazivaju konvolucijski slojevi. Na ulazu CNN nalazi se monohromatska slika ili slika u boji, nakon čega se izmenjuju konvolucijski slojevi i slojevi sažimanja. Zatim sledi nekoliko potpuno povezanih jednodimenzionalnih slojeva poput onih u ANN, a na kraju se nalazi izlazni sloj.

4.2. Inicijalizacija težinskih vrednosti

Ispravna inicijalizacija težinskih vrednosti ključna je kod učenja vrlo dubokih neuronskih mreža te bi loša inicijalizacija mogla rezultovati problemom "nestajućeg"ili pak gradijenta koji "eksplodira". Postoji nekoliko različitih pristupa inicijalizaciji, a svaki sadrži određene komparativne prednosti i mane. Dobre rezultate u dubokim neuronskim mrežama pokazuje pravougaona nasumična inicijalizacija za koju se raščlanjuje slučajna težina matrice, a potom se pravougaona matrica koristi za inicijalizaciju težinskih vrednosti konvolucijskih neuronskih slojeva. Kako bi se izbegli problemi s nestajanjem i eksplozijom gradijenata, pristupa se nenadziranom pred-učenju slojeva koji može biti popraćen fazom nadziranog podešavanja. Da bi se ublažio problem proporcionalnosti varijanse izlaznih i ulaznih veza, pristupa se inicijalizaciji težinskih vrednosti varijansom zavisnom o broju ulaznih i izlaznih veza neurona gde su w težinske vrednosti mreže, n_{f-in} broj veza koje ulaze u neuron, a n_{f-out} broj veza koje izlaze iz neurona:

$$Var(w) = -\frac{2}{n_{f-in} + n_{f-out}} \tag{2}$$

5. PRIMENA VEŠTAČKIH NEURONSKIH MREŽA ZA REŠAVANJE IGRE SUDOKU

Sudoku je postala veoma popularna u Velikoj Britaniji 2004. godine. *Sudoku* ili *Su Doku* je Japanska reč odnosno fraza koja znači *broj mesta*. Ideja igre je krajnje jednostavna, potrebno je popuniti mrežu dimenzija 9x9 koja je podeljena u blokove 3x3 brojevima od 1 do 9 po određenim pravilima: svaki blok mora da ima sve brojeve od 1 do 9 bez ponavljanja, ali takođe i za svaki red i svaku kolonu važi isto pravilo.

Postoji mnogo algoritama pogodnih za rešavanje igre Sudoku, ali u okviru ovog rada su se razmatrale dve metode, rešavanje uz pomoć veštačkih neuronskih mreža i rešavanje uz pomoć konvolucijskih neuronskih mreža. Za implementaciju ovih algoritama korišćen je programski jezik *Python*.

Pristup pripremanja podataka za obradu kod običnih i konvolucijskih veštačknih neuronskih mreža je dosta različit. Kako obična neuronska mreža predstavljena u radu može da primi samo jednodimenzione nizove kao ulaz, bilo je jednostavno pripremiti podatke jer su oni već bili u formi niza dimenzija 81x1. Podaci su preuzeti sa web stranice https://www.kaggle.com/bryanpark/sudoku , gde ima 1.000.000 primera rešenih i nerešenih Sudoku-a u dve kolone u formi csv fajla. Uz pomoć Python-ove biblioteke Pandas bilo je jednostavno pristupiti podacima.

Kada je reč o pripremi podataka za konvolucijske neuronske mreže, priča je dosta drugačija. Pošto CNN primaju podatke u matričnom obliku kao što su na primer slike, ovde je bilo potrebno predstaviti Sudoku kao matricu dimenzija 9x9x1. Poslednja dimezija označava da podaci imaju jedan kanal, kao što je to slučaj kod crno belih slika.

Takođe, potrebno je napomenuti kako je urađeno poređenje stvarnih i predviđenih vrednosti. Kod ANN predviđenje vrednosti su dimenzija 729x1 iz razloga što za svako polje u Sudoku-u kojih ima 81 mogu se naći brojevi od 1 do 9, to znači da treba ne samo predvideti tačan broj, već i tačnu poziciju. Iz ovog razloga je rešavanje igre Sudoku ne tako klasičan primer klasifikacije. Kako bi uspeli da uporedimo predviđene vrednosti sa stvarnim, koji su dimenzija 81x1, bilo je potrebno transformisati izlazni niz. To je urađeno tako što je svaka cifra u nizu predstavljena nulama i jedinicama na sledeći način: svaki od 81 brojeva je prestavljen sa 9 cifara od kojih je 8 nuli i jedna 1 koja se nalazi na poziciji koja odgovara određenom broju. Kako je za obuku CNN korišten Keras, nije bilo potrebe za ovim transformacijama na izlaznim podacima jer je bilo moguće uporediti ih u formi 81x1.

Za samu obuku i validaciju mreža korišćeno je 70% podataka odnosno 700 000, a za testiranje modela na novim podacima korišćeno je 30% odnosno 300 000 primera.

```
Predictions
[[4 4 6 1 1 7 4 4 9]
 [7 7 4 1 4 5 5 2 9]
 [5 6 9 4 4 4 4 6 7]
 [2 7 5 8 5 3 5 3 9]
 [3 4 8 2 5 4 9 1 4]
 [3 5 1 4 1 9 2 3 9]
 [7 7 5 7 6 5 5 4 6]
 [2 9 5 1 8 6 3 9 3]
 [3 5 7 6 2 6 4 6 4]]
Solved
[[6 2 4 5 1 3 7 9 8]
 [7 8 5 2 4 9 3 1 6]
 [9 1 3 7 8 6 4 5 2]
 [8 3 2 9 6 1 5 4 7]
 [5 7 9 3 2 4 8 6 1]
 [4 6 1 8 7 5 2 3 9]
 [1 4 7 6 3 2 9 8 5]
 [3 9 8 1 5 7 6 2 4]
 [2 5 6 4 9 8 1 7 3]]
Comparison
[[False False False False  True False False False False]
 [ True False False False  True False False False False]
 [False False False False False False  True False False]
 [False False False False False False  True False False]
 [False False False False False  True False False False]
 [False False  True False False False  True  True  True]
 [False False False False False False False False False]
 [False  True False  True False False False False False]
 [False  True False False False False False False False]]
```

Slika 1: *Primer rešenja igre Sudoku pomoću ANN.*

Kako obuka mreža zavisi mnogo kako od samih arhitektura, odnosno broja slojeva i broja neurona po slojevima, ali i od podešavanja hiperparametara kao što su parametri optimizatora, bilo je potrebno dosta testiranja i probavanja različitih pristupa. Kod ANN najbolje se pokazala arhitektura sa 3 skrivena sloja, gde prvi sloj sadrži 30 neurona, drugi 100, a treći 729. Ova arhitektura je dala tačnost od 10,6% i gubitkom od 4.359 sa trening podacima i tačnost od 10,6% i gubitkom od 1.870 sa test podacima odnosno

podacima koje ranije nije videla. Obuka je trajala 3 epohe, mreža je bila obučavana sa skupovima podataka od 64 elementa, a stopa učenja je bila 0.001. Tokom prve epohe gubitak modela je iznosio oko 80, a tokom druge epohe je krenuo da opada. Pokušano je obučavanje modela i kroz više epoha, ali nisu postignuti bolji rezultati. Na slici 1 je prikazan primer rešavanja igre Sudoku pomoću opisanog modela.

Što se tiče CNN, arhitektura je bila sledeća: korišćena su 3 konvoluciona sloja, prva dva sloja imala su 64 filtera dimenzija 3x3, dok je poslednja konvolucija imala 128 filtera dimenzija 1x1. Na kraju je dodat jedan skriveni sloj dimenzija 729x1 koji je pretvoren u matricu 81x9 radi računanja predviđenih vrednosti. Obuka je trajala 2 epohe sa skupovima podataka od po 32 jedinice. U prvoj epohi stopa učenja je bila 0.001 a u drugoj je redukovana na 0.0001. Rezultat koji je postignut je sledeći: gubitak koji je dobijen nakon testiranja je iznosio 0.3615, a tačnost je iznosila 90%. Ovo su bili ubedljivo najbolji postignuti rezultati. Primer rešene igre Sudoku pomoću ovog modela se može videti na slici 2.

```
game = '''
          0 8 0 0 3 2 0 0 1
          7 0 3 0 8 0 0 0 2
          5 0 0 0 0 7 0 3 0
          0 5 0 0 0 1 9 7 0
          6 0 0 7 0 9 0 0 8
          0 4 7 2 0 0 0 5 0
          0 2 0 6 0 0 0 0 9
          8 0 0 0 9 0 3 0 5
          3 0 0 8 2 0 0 1 0
      '''

game = solve_sudoku(game)

print('solved puzzle:\n')
print(game)
```

```
solved puzzle:

[[4 8 9 5 3 2 7 6 1]
 [7 1 3 4 8 6 5 9 2]
 [5 6 2 9 1 7 8 3 4]
 [2 5 8 3 4 1 9 7 6]
 [6 3 1 7 5 9 2 4 8]
 [9 4 7 2 6 8 1 5 3]
 [1 2 5 6 7 3 4 8 9]
 [8 7 6 1 9 4 3 2 5]
 [3 9 4 8 2 5 6 1 7]]
```

Slika 2: *Primer rešenja igre Sudoku pomoću CNN.*

6. ZAKLJUČAK

Rezultati obuke i testiranja pokazuju da uz dobro podešene hiperparametre mreže, CNN pokazuje znatno bolje rezultate u odnosu na ANN kada je u pitanju rešavanje igre Sudoku. Iako su CNN kompleksnije, optimizacija i fleksibilnost koje one pružaju su obećavajući kada govorimo o rešavanju problema kao sto je rešavanje igre Sudoku.

Interesantno bi bilo razmotriti i druge pristupe rešavanja ovoga problema uz pomoć veštačkih i konvolucijskih neuronskih mreža. Možda bi ljudski način razmišljanja i rešavanja igre, gde se rešava broj po broj, a ne cela matrica odjednom, doveo do još boljih rezultata.

7. LITERATURA

[1] https://research.google.com/colaboratory/faq.html

[2] C. M. Teng., ”Correcting noisy data”, *16th International Conf. on Machine Learning* 1999.

[3] Isabelle Guyon and André Elisseeff, ”An Introduction to Variable and Feature Selection”, *JMLR Special Issue on Variable and Feature Selection* 2003.

[4] Khan, Rahmani et al, ”A Guide to Convolutional Neural Networks for Computer Vision”, 2018.

[5] Diederik P. Kingma and Jimmy Lei Ba, ”Adam: A method for stochastic optimization”, *Published as a conference paper at ICLR* 2015.

[6] Walter Pitts and Warren McCulloch, ”A Logical Calculus of Ideas Immanent in Nervous Activity”, 1943.

[7] Alex Krizhevsky and Ilya Sutskever and Geoffrey E. Hinton, ”ImageNet Classification with Deep Convolutional Neural Networks”, 2012.

[8] Harrison Kinsley and Daniel Kukieła, ”Neural Networks from Scratch in Python”, *Kinsley Enterprises Inc.* 2020.

[9] Yves Chauvin and David E. Rumelhart, ”Backpropagation:Theory, Architectures, and Applications”, *Lawrence Erlbaum Associates* 1995.

[10] Rumelhart and McClelland and the PDP Research Group, ”Parallel distributed processing: Explorations in the microstructure of cognition.”, 1986.

Kratka biografija:

Jovan Bosić rođen je u Kninu 1995. god. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Duboko učenje odbranio je 2021. godine.
Kontakt: jovanbosic95@gmail.com