pd.tvz.hr
POLYTECHNIC & DESIGN Vol. 7, No. 1, 2019.
DOI: 10.19279/TVZ.PD.2019-7-1-04

PREGLED TEHNOLOGIJA U NEURONSKIM MREŽAMA

OVERVIEW OF ARTIFICIAL NEURAL NETWORK TECHNOLOGIES

Tin Krambeger, Bojan Nožica, Ivica Dodig, Davor Cafuta

Tehničko veleučilište u Zagrebu, Vrbik 8, Zagreb, Hrvatska

Sažetak

Neuronske mreže u današnje vrijeme se sve više istražuju. Razlog tome je sklopovlje koje danas nudi mogućnost obrade velike količine podataka u stvarnom vremenu. Za uspješan rad i konstruiranje neuronske mreže od velikog su značaja aktivacijske funkcije. Njihovim kvalitetnim odabirom utječe se na brzinu i kvalitetu učenja same neuronske mreže. U radu su objašnjeni osnovni principi rada neuronske mreže nakon odabira kvalitetnih aktivacijskih funkcija. Nadalje su prikazani osnovni principi učenja neuronskih mreža s naglaskom na odabir optimizacijskog algoritma koji se koriste za učenje neuronske mreže.

Ključne riječi: *Neuronske mreže, NN, Pregled*

Abstract

Neural networks are being researched more and more today. The reason for research lies in the hardware that nowadays offers the ability to process large amounts of data in real time. For the successful operation and construction of the neural network, one of great importance is the activation function.

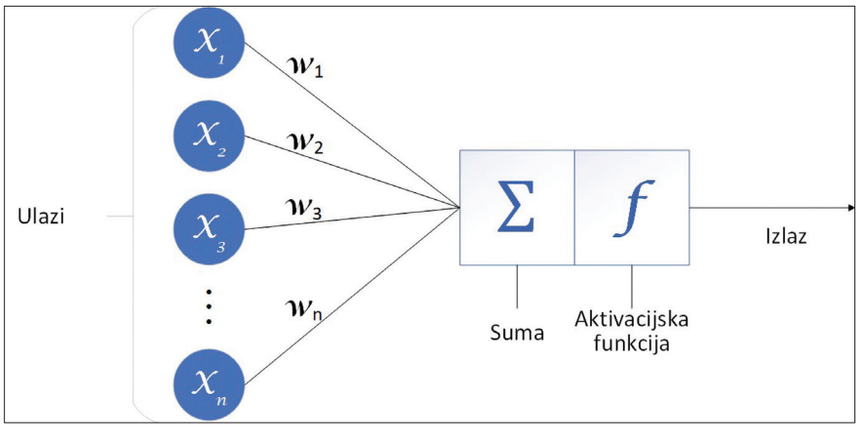
Activation function selection affects the speed and quality of training the neural network itself. The basic principles of the neural network after the selection of activation functions are explained in the paper. The basic principles of learning neural networks are outlined, focusing on selecting the optimization algorithm used to learn the neural network.

Keywords: *Neural networks, NN, Preview*

1. Uvod

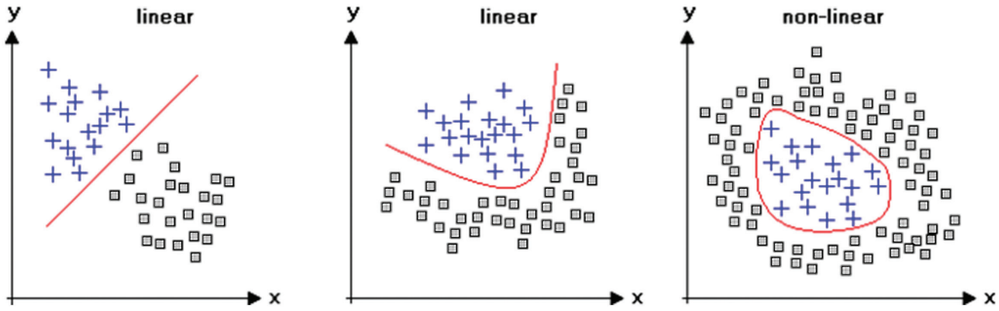
1. *Introduction*

Ideja neuronskih mreža izuzetno je stara i datira od pedesetih godina prošlog stoljeća, ali tek posljednjih godina neuronske mreže su područje računarstva koje se iznimno istražuje prvenstveno zbog sve bržeg računalnog sklopovlja u vidu računanja matematičkih operacija i interneta. Tome su najviše doprinijele grafičke kartice pomoću kojih je moguće računati kompleksne matrične račune koje se najviše koriste u neuronskim mrežama i dostupnosti velike količine podataka koji su potrebni za treniranje neuronskih mreža.



Slika 1 *Umjetni neuron*

Figure 1 *Artificial neuron*



Slika 2 *Linearni i nelinearni modeli*

Figure 2 *Linear and nonlinear models*

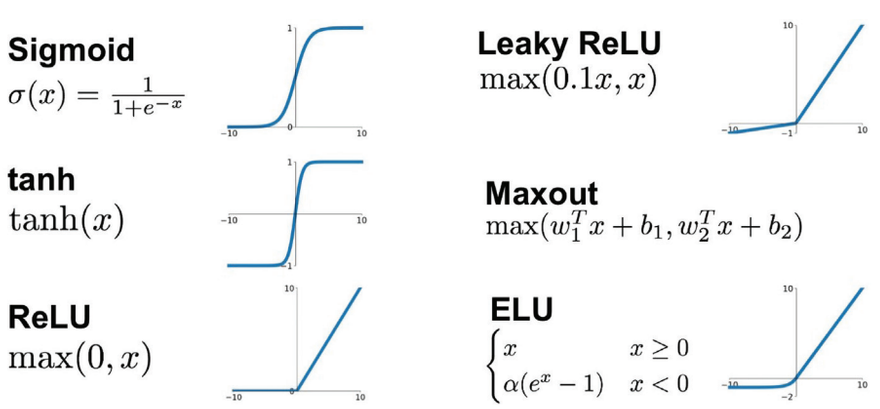
Neuronske mreže su nastale na neki način kao simulacija bioloških procesa učenja. Živčani sustav inteligentnijih živih bića sastoji se od stanica koje se zovu neuroni. Neuroni su međusobno spojeni pomoću aksona i dendrita. Na sličan način funkcionira i umjetni neuron. Dendrit bi bio ulaz u neuron do kojeg dolaze izlazi iz prethodnog sloja (X_1-X_n) nakon čega se računa suma ulaza nad kojom se poziva aktivacijska funkcija čija se vrijednost potom prosljeđuje dalje na sljedeći sloj. Aktivacijska funkcija služi umjetnoj neuronskoj mreži da joj se podari nelinearnost. Pojednostavljena shema umjetnog neurona nalazi se na slici 1.

U trenutku pisanja ovog rada nije moguće u potpunosti simulirati rad mozga iz nekoliko razloga. Samo za usporedbu, ljudski mozak ima oko 100 milijardi neurona i oko 100 trilijuna veza (sinapsi) i troši oko 20W energije. Za usporedbu je moguće uzeti najveću umjetnu neuronsku mrežu koja ima oko 10 milijuna neurona i jednu milijardu konekcija, a računalo koje je pokreće ima 16 tisuća CPU-a i troši oko 3 milijuna wata. Mozak ima 5 tipova senzora. Ne zna se točno kako mozak uči, ali vrlo vjerojatno ne uči računanjem parcijalnih derivacija i to je jedan od velikih nedostataka simulacije živčanog sustava.

2. Aktivacijske funkcije neuronske mreže
2. *Neural network activation functions*

Kao što je opisano na slici 1., nakon što se sumiraju ulazi u neuron, nad sumom je potrebno izvršiti funkciju aktivacije. Zapravo suma, tj. skalarni produkt vektora nije najpravilniji izraz jer neke aktivacijske funkcije ne koriste skalarni produkt ulaznih vektora već uzimaju maksimalnu vrijednost (ReLu). Aktivacijske funkcije koriste se kako bi neuronska mreža bila nelinearna. U slučaju kada bi postojao samo linearan izlaz iz svakog sloja neuronske mreže, sve slojeve bi se moglo zamijeniti jednim slojem zbog nedostatka nelinearnosti.[1] Primjer linearnosti i nelinearnosti statističkih modela moguće je vidjeti na slici 2. Bez nelinearnosti nije nikako moguće napraviti kompleksniji model klasifikacije, te bi se neuronska mreža zapravo mogla svesti na logističku regresiju.

Aktivacijske funkcije se razlikuju po matematičkim formulama, odnosno načinu na koji djeluju nad ulaznim setom podataka. Izuzetno su bitne prilikom konstruiranja neuronske mreže jer o odabiru aktivacijske funkcije ovisi brzina učenja i vrlo često preciznost i performanse same neuronske mreže.



Slika 3 *Aktivacijske funkcijei*

Figure 3 *Activation functions*

Osim što je bitna za nelinearnost mreže, nad aktivacijskom funkcijom potrebno je moći izračunati derivaciju.
Sigmoidna funkcija nije centrirana po nuli i ima računanje eksponenta što je izuzetno zahtjevno za računanje. Osim toga, najveći je problem zasićenja gradijenata zato što funkcija ima kodomenu između 0 i 1 i vrijednosti mogu ostati konstantne zbog čega će gradijent imati manje vrijednosti. Zbog toga, moguće je da se ne dogodi promjena kod gradijentnog pada.
Hiperbolički tangens je neparna, monotono rastuća funkcija. Kodomena vrijednosti joj je -1 i 1 i centrirana je po nuli. Funkcija se definira kao u nastavku:

$$tanh(x) = \frac{sinh(x)}{cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

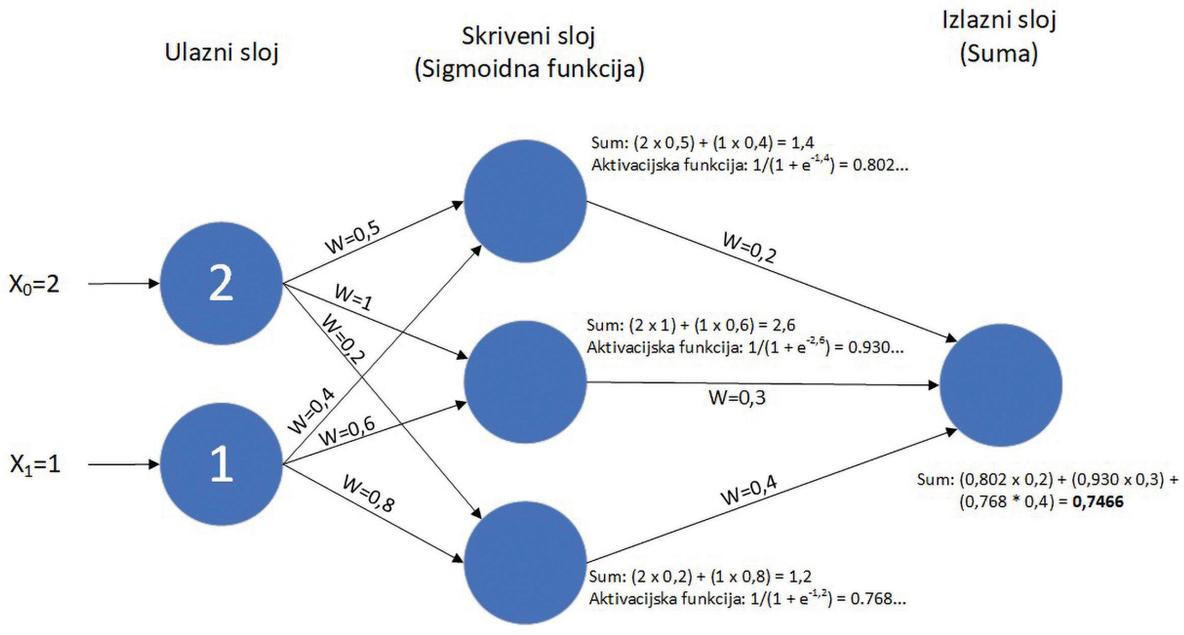
Kao i kod sigmoidne funkcije postoji problem zasićenja gradijenata. Kada je ulaz veći od nule gradijenti će biti pozitivni ili negativni što može dovesti do problema eksplozije ili nestajanja gradijenata.
Aktivacijska funkcija ispravljene linearne jedinice (engl. Rectified Linear Unit Activation Function) ili ReLU je najčešće korištena aktivacijska funkcija zbog jednostavnosti propagacije unazad i zato što nije zahtjevna za računanje. Najveći problem ReLU funkcije je mrtav ReLU što se dogodi u slučaju ako je ulaz manji od nule, funkcija će uvijek davati nulu. [2][3]
Propuštajući ReLU (engl. Leaky ReLU) rješava problem mrtvog ReLU-a na način da dodaje dodatnu konstantu nagiba grafa. Na taj način je riješen problem mrtvog ReLU-a i ubrzano je učenje zbog boljeg balansa.[4]
Maxout za aktivaciju koristi maksimalnu vrijednost unutar grupe linearnih dijelova. Uspoređuje vrijednosti nad grupama kandidata, a ReLU ih uspoređuje s nulom. Skupa je za izračun jer dodaje dodatni skriveni sloj i potrebno ju je koristiti uz dropout.[5]
Aktivacijska funkcija eksponencijalne linearne jedinice (engl. Exponential linear units) je također varijacija ReLU funkcije s boljom vrijednosti za x<0. Ima identična svojstva kao i ReLU, ali nema problema s mrtvim ReLU-ima.
Najveća manjkavost je što ima više računanja zbog eksponencijalne funkcije. [6]
Jezgreno bazirana ne parametarska aktivacijska funkcija (engl. kernel-based non-parametric activation function) je nastala 2017 kako bi se dodatno povećala fleksibilnost neuronskih mreža. KAF je ne parametrizirana aktivacijska funkcija definirana kao jednodimenzionalna jezgrena aproksimacija. [7] Softplus funkcija je uvedena 2001 godine i ima vrlo sličnu krivulju kao i ReLU, ali mekaniju krivulju oko nule i puno bolju derivaciju. Najveći je problem što slično kao kod ReLU funkcije može doći do mrtvog neurona.
Softmax funkcija se koristi za zadnji sloj neurona kod klasifikacije (primjerice slika). Funkcija prima nenormalizirani vektor i normalizira ga u vjerojatnosnu distribuciju. [8][9]

3. Princip rada neuronske mreže
3. *The working principles of neural networks*

Mrežu je moguće gledati kao kompleksni računski graf u kojem je jedinica računanja neuron. [1] Mreže je moguće klasificirati na nekoliko različitih modela, ali sve rade na sličnom principu. Imaju ulaze iz kojih kroz slojeve trebaju kreirati smisleni izlaz koji su naučile postupkom učenja. Slojevi se sastoje od neurona koji su simulirani pomoću aktivacijskih funkcija. Svaki neuron posjeduje vlastitu težinu (w) koja se propagira na sljedeći sloj kroz aktivacijsku funkciju i na taj način se dobivaju izlazi. U primjeru na slici 4. moguće je vidjeti primjer jednostavne mreže s nasumično postavljenim vrijednostima ulaza i težina kao i izračun vrijednosti izlaza iz neuronske mreže. Na slici je izuzetno pojednostavnjeni model koji ima samo jedan nelinearni sloj s sigmoidnom aktivacijskom funkcijom. Neuronska mreža će najčešće se sastojati od nekoliko nelinearnih skrivenih slojeva, a izlazni sloj će biti različit s obzirom na to što se od neuronske mreže očekuje da radi.[1], [8]

4. Učenje neuronske mreže
4. *Neural network training*

Učenje neuronske mreže radi se pomoću algoritma zvanog propagacija unazad. [10] Težine neuronske mreže je moguće naučiti pomoću mnoštva kvalitetnih primjera što je najbitniji uvjet za kvalitetno učenje.



Slika 4 *Jednostavna neuronska mreža s izračunima*

Figure 4 *Simple neural network with calculations*

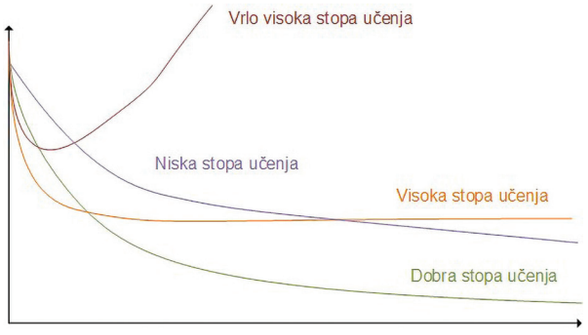
Za učenje neuronske mreže potrebno se odlučiti za jedan od nekoliko optimizacijskih algoritama za učenje poput stohastičkog gradijentnog pada, Momentuma, Adagrada, Nesterov, Adadelta, Adam, Nadam, RMSprop, AdaMax i sl. [11] Prilikom učenja neuronske mreže prvo je potrebno poznavati ono što se uči, odnosno ulazne podatke i očekivane izlazne podatke za što je potrebno poznavati dio matematike koja se bavi statističkim pogreškama. Funkcija koja računa pogrešku naziva se funkcija gubitka (Engl. Loss function). Postoji nekoliko varijanti kako se funkcija može izvesti i to je izuzetno ovisno o konkretnom problemu koji se pomoću neuronske mreže pokušava riješiti. Nakon računanja greške (gubitka) računa se gradijent kako bi se znalo u kojem smjeru treba mijenjati vrijednosti i propagirati ih po neuronskoj mreži pomoću algoritma propagacije unazad.
Postoji nekoliko varijanti funkcija gubitka i vrlo je bitno da se koriste ispravno u vidu problema koji se pokušava riješiti neuronskom mrežom. Funkcije gubitka izračunava grešku između željenih rezultata i dobivenih rezultata. Funkcije gubitka moguće je podijeliti u dvije skupine regresijskog i klasifikacijskog gubitka. Naravno, ovisno o tome radi li neuronska mreža regresiju (neko određeno predviđanje podataka) ili radi klasifikaciju (primjerice klasifikacija slika po nekom parametru). [12]–[21]

5. Gradijent i gradijentni pad

5. *Gradient and gradient descent*

Nakon što se izračuna funkcija gubitka, pomoću iste je potrebno izračunati gradijent. Gradijent je više varijabilna generalizacija derivata i jedan je od osnovnih koncepata u analizi vektora i teorije nelinearnih mapiranja. Derivaciju je moguće definirati nad funkcijama s jednom varijablom, a za funkciju s više varijabli ulogu derivacije preuzima gradijent. Kao i derivacija, gradijent određuje nagib tangente grafa funkcije, odnosno gradijent pokazuje smjer gdje je najveće povećanje funkcije. Na gradijentni pad je moguće figurativno gledati kao na brdo na čiji sam vrh se mora popeti slijep čovjek u što manje koraka. Ta procedura se opisuje matematički pomoću gradijenta i zove se gradijentni pad jer je vrh planine zapravo dno i gradijentni pad je minimizacijska funkcija. U neuronskim mrežama gradijent je potrebno izračunati kako bi se odredio smjer greške (smjer kretanja) i koje težinske vrijednosti je potrebno namjestiti kako bi se unaprijedilo rješenje mreže. Gradijent mjeri koliko se izlaz funkcije promijeni ako se malo promjene ulazi.
Gradijentni pad osim izračuna gradijenta treba i stopu učenja. To je jedan broj koji će odraditi veličinu koraka prema minimumu funkcije.

Taj parametar ne smije biti ni prevelik ni premalen. Ako je stopa učenja prevelika, algoritam će premašiti cilj, odnosno minimum, a ako je premala, proces učenja će trajati predugo. Prilikom odabira stope učenja najbolje je kreirati graf funkcije gubitka kojeg mogu kreirati svi današnji programski okviri za rad s neuronskim mrežama. Na grafu će biti odmah vidljivo radi li se o ispravnoj ili neispravnoj stopi učenja. [22]–[25]



Slika 5 *Funkcija gubitka s obzirom na stopu učenja*

Figure 5 *Loss function in correlation to learning rate*

6. Propagacija unazad
6. *Backpropagation*

Propagacija unazad je algoritam za učenje neuronskih mreža koji je kreirao David E. Rumelhart 1986. godine. Propagacija unazad je skraćeni oblik pisanja, dok je dulji oblik pisanja propagiranje greške unazad zato što se greška izračunava pomoću funkcije gubitka nakon čega se distribuira odnosno propagira kroz neuronsku mrežu. To je algoritam koji se koristi kako bi se izračunali gradijenti koji su potrebni kako bi se podesile težinske vrijednosti u neuronskoj mreži. [8] Algoritam propagacije unaprijed je zapravo generalizacija delta pravila na višeslojne unaprijedne mreže (Engl. feedforward network) koja je moguća zbog korištenja pravila lanca. Pravilo lanca u matematici je pravilo za računanje derivacije kompozitne funkcije (funkcija koja se sastoji od više funkcija). [22], [26]
Propagaciju unazad koriste optimizatori gradijentnog pada. Najčešći algoritmi koji se koriste u modernim neuronskim mrežama su Adadelta, Adagrad, Adam, Adamax, Nadam, RMSprop, Vanilla, stohastički gradijentni pad i mnogi drugi.

Različiti optimizatori postoje iz povijesnih razloga i razloga što neuronska mreža nije namijenjena rješavanju samo jednog problema, već može rješavati mnogo različitih problema iz različitih područja koji traže drugačiji pristup problemu. Razlikuju se također s obzirom na količinu podataka koje je potrebna, točnost i vrijeme izvođenja.
Vanilla gradijentni pad, odnosno *batch* gradijentni pad računa gradijent funkciju gubitka s obzirom na parametre θ za cijeli set podataka (Engl. dataset). Dakle, ova verzija gradijentnog pada će računati gradijent za cijeli set podataka kako bi napravio jedno ažuriranje (Engl. update). Zbog tog problema, Vanilla algoritam je izuzetno spor i zahtjeva ogromne količine radne memorije kako bi cijeli set podataka stao u nju prilikom računanja gradijentnog pada.
Stohastički gradijentni pad (u danjem tekstu SGD) je metoda koja računa parametre θ za svaki primjer treniranja iz seta podataka. Vanilla radi redundantne računske operacije za velike setove podataka jer računa gradijente za slične primjere prije svakog ažuriranja. SGD radi jedno od jednom jedno ažuriranje. Dok Vanilla jako često ostane u lokalnom minimumu, SGD zbog svoje fluktuacije može preskočiti lokalni minimum i pronaći globalni. Najveći je problem SGD-a konvergencija do apsolutnog minimuma funkcije jer se može dogoditi promašivanje. Postoje metode koje tijekom učenja smanjuju stopu učenja pa je taj problem pomoću njih znatno umanjen.
Gradijentni pad mini seta (Engl. Mini-batch gradient descent) koristi najbolje od oba svijeta kako bi napravio ažuriranje za svaki mini set iz seta podataka. Na taj način smanjuje varijaciju ažuriranja parametara i na taj način dolazi do stabilnije konvergencije i može koristiti optimiziranije računanje matricama (prednosti novih grafičkih kartica). Veličina seta ovisi o problemu koji se rješava, ali je negdje u rangu od 50 do 256. Kada se koristi SGD, najčešće se koristi i ovaj algoritam jer se već nalazi u programskim okvirima za treniranje neuronskih mreža.
Problem kod SGD algoritma je upravo šverdanje, odnosno krivudanje ukoliko krivulja nema dovoljno jasno izražen put prema globalnom optimumu. Naravno, krivudanje usporava učenje jer se nepotrebno gubi vrijeme. Kada bi se

povećala stopa učenja, stvari se mogu popraviti, ali i pogoršati zbog promašivanja globalnog optimuma. Momentum algoritam rješava taj problem pomoću dodavanja vrijednosti parametra izglađivanja (Engl. smoothing parameter) na vektor kretanja. Na neki način, kao da se lopta dodatno proguruje nizbrdo. Rezultat je brža konvergencija i reducirana oscilacija kretanja.
Nesterov momentum je nadogradnja klasičnog momentum algoritma. Naime, postoji problem kod momentuma da se „guranje lopte" ne uspori pri kraju, tako da se često događa da se optimum prvo promaši, pa se vraća nazad na isti. Taj problem rješava Nesterov momentum na način da prilikom ažuriranja gleda malo unaprijed kako bi prilagodio parametar izglađivanja. [29]
Adagrad algoritam koji optimizira stopu učenja s obzirom na parametre na način da provodi veće korake za rjeđe parametre i manje korake za češće parametre. Zbog toga je idealan algoritam za rad s raspršenim podacima. Adagrad povećava robusnost SGD-a i koristi se prilikom treniranja velikih neuronskih mreža. Eliminira potrebu za podešavanjem stope učenja.
Adadelta je dodatak na Adagrad algoritam koji ne pohranjuje prijašnje gradijente kao što radi Adagrad, već ih svede na fiksnu veličinu nakon čega se oni stariji gube. To dovodi do redukcije smanjenja stope učenja. [30]
RMSprop i Adadelta su bili razvijani neovisno jedan o drugome i izuzetno su slični kako bi riješili problem s Adagrad algoritmom. RMSprop je identičan prvom ažurirajućem vektoru Adadelte.
Predviđanje adaptivnog momenta (Engl. Adaptive Moment Estimation, Adam) je još jedan od algoritama koji računa vlastitu stopu učenja poput Adagrad i RMSprop algoritma, ali eksponencijalno zaglađuje gradijent prvog reda kako bi ukomponirao momentum u ažuriranje.
AdaMax algoritam je izuzetno sličan Adam algoritmu uz minimalne promjene oko stabilizacije vršnih vrijednosti. Autori Kingma i Ba izmjenjuju Adam algoritam i dokazuju da uz izmjene doprinose bržoj konvergenciji ka globalnom optimumu. [31]
Nadam algoritam (Engl. Nesterov-accelerated Adaptive Moment Estimation) kombinira algoritme Adam i Nesterov accelerated gradient. [32]

Ako su kod ulaza podataka raštrkani podaci najvjerojatnije će se najbolji rezultati postići s nekim od optimizatora koji koriste adaptivnu stopu učenja. SGD će naći minimum, izuzetno je robustan, ali će mu trebati neko dulje vrijeme za navedene radnje. Jako mnogo radova još uvijek koristi SGD zato što je dobra stara testirana metoda koja funkcionira. [11], [27]–[43]

7. Zaključak
7. *Conclusion*

Rad daje pregled područja umjetnih neuronskih mreža kao i neka najnovija saznanja u istima. Kao što je moguće vidjeti u radu, brojni znanstvenici intenzivno rade na različitim rješenjima za slične probleme prvenstveno zbog nedostatka znanja o biološkim procesima učenja i nedostatka računalne snage. Umjetne neuronske mreže još uvijek ne mogu premašiti ljudsku inteligenciju zbog nedostatka računalne snage i izuzetno kompleksne arhitekture mozga koju još uvijek nije moguće postići. Bez obzira na navedeno, umjetne neuronske mreže su mnogo bolje u nekim zadacima kao što su prepoznavanje slika, igranje šaha i sličnih zadataka koji su izuzetno specifični i precizni.

8. REFERENCE
8. *REFERENCES*

[1] C. C. Aggarwal, Neural Networks and Deep Learning. Springer, 2018.

[2] A. Maas, A. Hannun, and A. Ng, "Rectifier Nonlinearities Improve Neural Network Acoustic Models," 2013.

[3] A. F. Agarap, "Deep Learning using Rectified Linear Units (ReLU)," no. 1, 2018.

[4] S. Qiu, X. Xu, and B. Cai, "FReLU: Flexible Rectified Linear Units for Improving Convolutional Neural Networks," 2017.

[5] I. J. Goodfellow, D. Warde-farley, and A. Courville, "Maxout Networks," 2013.

[6] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)," Nov. 2015.

[7] S. Scardapane, S. Van Vaerenbergh, S. Totaro, and A. Uncini, "Kafnets: kernel-based non-parametric activation functions for neural networks," pp. 1–35, 2017.

[8] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, 2016.

[9] J. S. Bridle, "Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition," in Neurocomputing, Berlin, Heidelberg: Springer Berlin Heidelberg, 1990, pp. 227–236.

[10] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," Nature, vol. 323, no. 6088, pp. 533–536, 1986.

[11] S. Ruder, "An overview of gradient descent optimization algorithms," Sep. 2016.

[12] A. C. Rencher and W. F. Christensen, Methods of multivariate analysis, Third edition. 2012.

[13] "Max-Margin Classifications," in Machine Learning for Multimedia Content Analysis, Boston, MA: Springer US, 2007, pp. 235–266.

[14] C. Cortes and V. Vapnik, "SUPPORT-VECTOR NETWORKS," Mach. Learn., vol. 20, no. 973, pp. 273–297, 1995.

[15] A. Natekin, A. Knoll, and O. Michel, "Gradient boosting machines, a tutorial," 2013.

[16] "Max-Margin Classifications," in Machine Learning for Multimedia Content Analysis, Boston, MA: Springer US, pp. 235–266.

[17] P. Grover, "5 Regression Loss Functions All Machine Learners Should Know," 2018. [Online]. Available: https://heartbeat.fritz.ai/5-regression-loss-functions-all-machine-learners-should-know-4fb140e9d4b0.

[18] R. Parmar, "Common Loss functions in machine learning," 2018. [Online]. Available: https://towardsdatascience.com/common-loss-functions-in-machine-learning-46af0ffc4d23.

[19] K. Janocha and W. M. Czarnecki, "On Loss Functions for Deep Neural Networks in Classification," Schedae Informaticae, vol. 25, no. December, pp. 49–59, 2016.

[20] "Loss Functions and Optimization Algorithms. Demystified." [Online]. Available: https://medium.com/data-science-group-iitr/loss-functions-and-optimization-algorithms-demystified-bb92daff331c. [Accessed: 18-Feb-2019].

[21] "Losses - Keras Documentation." [Online]. Available: https://keras.io/losses/. [Accessed: 18-Feb-2019].

[22] M. Hazewinkel, Encyclopaedia of mathematics. Springer-Verlag, 2002.

[23] B. A. Dubrovin, A. T. Fomenko, and S. P. (Sergeĭ P. Novikov, Modern geometry--methods and applications. Springer-Verlag, 1992.

[24] J. Dean et al., "Large Scale Distributed Deep Networks," in Advances in Neural Information Processing Systems 25, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1223–1231.

[25] "Gradient Descent in a Nutshell – Towards Data Science." [Online]. Available: https://towardsdatascience.com/gradient-descent-in-a-nutshell-eaf8c18212f0. [Accessed: 19-Feb-2019].

[26] O. Hern and P. Rico, "A Semiotic Reflection on the Didactics of the Chain Rule," vol. 7, pp. 321–332, 2010.

[27] A. Karpathy, "Intro to neural nets for programmers." [Online]. Available: http://karpathy.github.io/neuralnets/.

[28] P. Jay, "Back-Propagation is very simple. Who made it Complicated?" [Online]. Available: https://medium.com/@14prakash/back-propagation-is-very-simple-who-made-it-complicated-97b794c97e5c.

[29] Nesterov Yurii E., "A method for solving the convex programming problem with convergence rate O (1/k^ 2)," Dokl. akad. Nauk Sssr, vol. 269, pp. 543–547, 1983.

[30] M. D. Zeiler, "ADADELTA: An Adaptive Learning Rate Method," Dec. 2012.

[31] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," Dec. 2014.

[32] T. Dozat, "INCORPORATING NESTEROV MOMENTUM INTO ADAM," ICLR 2016 Work., no. 1, pp. 2013–2016, 2016.

[33] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," J. Mach. Learn. Res., vol. 12, pp. 2121–2159, 2011.

[34] G. E. Hinton, "Neural networks for machine learning, Coursera Video." 2012.

[35] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09, 2009, pp. 1–8.

[36] J. Pennington, R. Socher, and C. Manning, "Glove: Global Vectors for Word Representation," in Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1532–1543.

[37] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient BackProp," 1998, pp. 9–50.

[38] C. Darken, J. Chang, and J. Moody, "Learning rate schedules for faster stochastic gradient search," in Neural Networks for Signal Processing II Proceedings of the 1992 IEEE Workshop, pp. 3–12.

[39] Y. Kawamoto, K. Chatzikokolakis, and C. Palamidessi, "On the Compositionality of Quantitative Information Flow," Nov. 2016.

[40] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," Feb. 2015.

[41] N. Qian, "On the momentum term in gradient descent learning algorithms," Neural Networks, vol. 12, no. 1, pp. 145–151, Jan. 1999.

[42] Y. Bengio, N. Boulanger-Lewandowski, and R. Pascanu, "Advances in optimizing recurrent networks," in 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, 2013, pp. 8624–8628.

[43] B. Yoshua, B.-L. Nicolas, and P. Razvan, "ADVANCES IN OPTIMIZING RECURRENT NETWORKS."

AUTORI · *AUTHORS*

Tin Krambeger - nepromjenjena biografija nalazi se u časopisu Polytechnic & Design Vol. 5, No. 1, 2017.

Korespodencija
tin.krambeger@tvz.hr

Bojan Nožica
Viši predavač u području tehničkih znanosti, polje računarstvo, grana informacijski sustavi. Na Tehničkom veleučilištu u Zagrebu djeluje od 2005. godine, najprije kao vanjski suradnik zadužen za uvođenje ISVU sustava, a zatim kao zaposlenik i nastavnik od 2009. godine. Autor je priznatog patenta te više znanstvenih i stručnih radova. Diplomirao je 1984. god. na Elektrotehničkom fakultetu Sveučilišta u Zagrebu, smjer Računarske tehnike i od tada stekao više od 25 godina stručnog iskustva u zemlji i inozemstvu (Sjedinjene Američke države, Savezna Republika Njemačka).

Korespodencija
bojan.nozica@tvz.hr

Ivica Dodig - nepromjenjena biografija nalazi se u časopisu Polytechnic & Design Vol. 5, No. 1, 2017.

Korespodencija
ivica.dodig@tvz.hr

Davor Cafuta - nepromjenjena biografija nalazi se u časopisu Polytechnic & Design Vol. 5, No. 1, 2017.

Korespodencija
davor.cafuta@tvz.hr