SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 689

Predviđanje kretanja tržišta dionica neuronskim mrežama

Dario Pavlović

Zagreb, lipanj 2022.

Ova stranica treba biti:

ILI prazna stranica

ILI stranica sa zahvalom (po želji studenta; zahvala nije obvezna).

Sadržaj

Uvod

U današnje vrijeme dosta se priča o razvoju umjetne inteligencije, no što je umjetna inteligencija točno? Sljedeći citat nam može pobliže objasniti šta bi trebala biti umjetna inteligencija.

„Umjetna inteligencija – naziv za znanstvenu disciplinu koja se bavi izgradnjom računalnih sustava čije se ponašanje može tumačiti kao inteligentno“ - John McCarthy, (1956.). Strojno učenje možemo nazvati primjenom umjetne inteligencije. Cilj nekog modela strojnog učenja je da uči i poboljšava se iz prethodno stečenih znanja i iskustava. Kao što smo strojno učenje nazvali primjenom umjetne inteligencije, tako možemo duboko učenje nazvati primjenom strojnog učenja. Duboko učenje pokušava imitirati način na koji ljudska inteligencija funkcionira da bi stekao određena znanja. Napretkom metoda strojnog učenja, u slučaju ovog završnog rada metoda dubokog učenja, pronalazimo njihove razne primjene u životu, primjerice kod autonomnih automobila, prepoznavanja objekata, prepoznavanja izraza lica, traženje uzorka korisnikovog ponašanja u svrhu plasiranja ciljanih reklami, i tako dalje. Tako će svrha i cilj ovog rada biti traženje uzorka kretanja i pokušaj predviđanja tržišta dionica koristeći model dubokog učenja, preciznije rekurentnu neuronsku mrežu.

1. Strojno učenje

1.1. Motivacija i definicija

Strojno učenje definiramo kao primjenu umjetne inteligencije, to jest strojnim učenjem želimo postići da naš sustav preko prethodno stečenih znanja i iskustva odlučuje. Sljedećim citatom možemo dobro predstaviti strojno učenje.

„Ono što je zajedničko mnogim definicijama jest spoznaja da danas živimo u svijetu u kojem smo okruženi obiljem podataka, te da nam je interesantno razvijati programske sustave koji su sposobni iskorištavati te podatke, učiti iz njih i na temelju toga nuditi korisna ponašanja.“ – [1]

Kako je i u citatu navedeno, za provedbu odlučivanja i stjecanja znanja trebamo imati podatke ili iz samog kompleta podataka želimo uočiti uzorak i saznati nešto više o tim podacima. Raspoloživi podaci mogu biti numerički ili kategorički. Primjer numeričkih podataka su godine starosti čovjeka, cijene, broj ljudi na koncertu, dok su primjeri kategoričkih podataka brand odjeće, marka automobila, boja očiju, itd.

1.2. Vrste strojnog učenja

Strojno učenje dijelimo na tri glavna područja, a to su:

- Nadzirano učenje
- Nenadzirano učenje
- Podržano učenje

1.2.1. Nadzirano učenje

Nadzirano učenje se definira setom ulaznih podataka x koji se preslikavaju na izlaz y, to jest naš skup za učenje je oblika $D = \{(x_i, y_i)\}_{i=1}^{N}$, gdje x_i predstavlja podatke o primjerku, a y_i predstavlja ciljnu vrijednost koju pridružuje tom primjerku (engl. *target value*). Zapravo tražimo presliku $\hat{y} = f(x)$, gdje ako je y_i diskretna/nebrojčana vrijednost radimo klasifikaciju, a ako je y_i kontinuirana/brojčana vrijednost radimo regresiju. Umjetne neuronske mreže generalno spadaju u ovu vrstu strojnog učenja, iako postoje i nenadzirane vrste neuronskih mreža.

1.2.2. Nenadzirano učenje

Nenadzirano učenje definiramo samo setom ulaznih podataka, skup učenja je onda oblika $D = \{(x_i)\}_{i=1}^{N}$, vidimo da naš skup onda ne prima y_i, odnosno ciljne vrijednosti uz ulazne. U postupke nenadziranog učenja spadaju postupci grupiranja (engl. *clustering*), postupci smanjena dimenzionalnost (engl. *dimensionality reduction*), postupak otkrivanja stršećih/novih vrijednosti (engl. *outlier/novelty detection*).

1.2.3. Podržano učenje

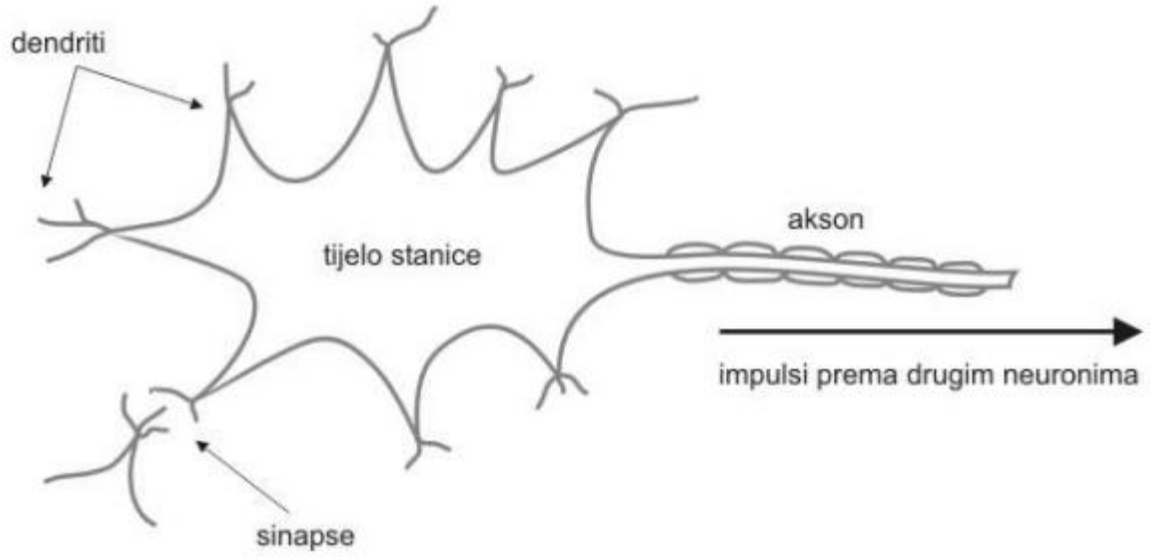
Podržano učenje se bavi optimizacijom ponašanja. Odnosno zadaća podržanog učenja je razviti optimalnu strategiju ponašanja na temelju pokušaja s odgođenom nagradom.

2. Umjetne neuronske mreže

2.1. Motivacija i definicija

Umjetne neuronske mreže ili skraćeno neuronske mreže, su model dubokog učenja inspiriran biološkim neuronima koji su sastavni dio mozga i živčanog sustava. U sljedećoj usporedbi možemo vidjeti biološku inspiraciju za model umjetnog neurona.

2.1.1. Biološki neuron

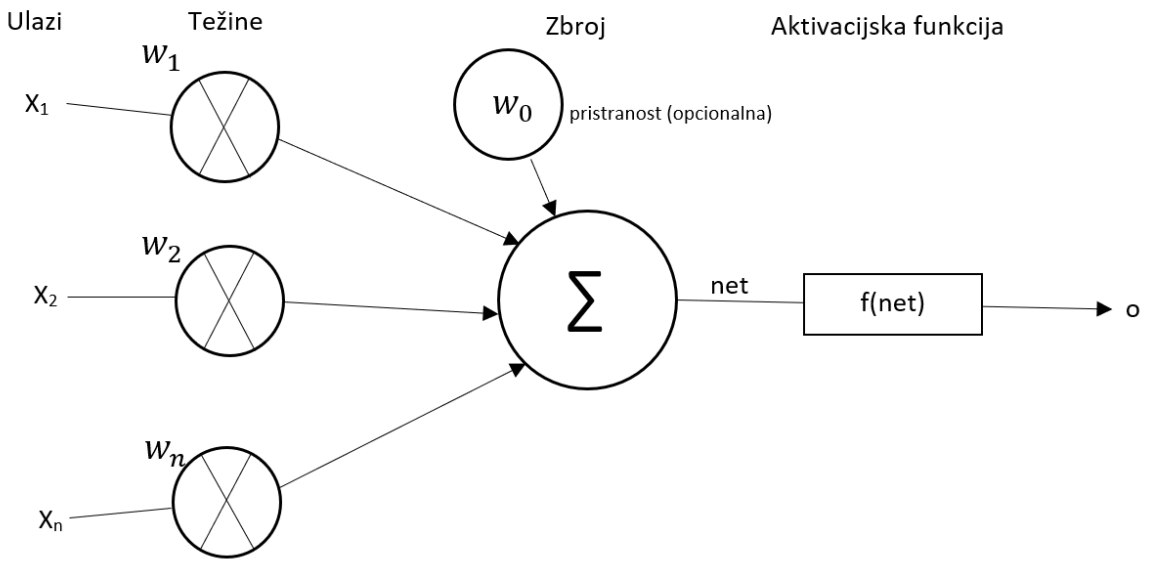


Slika 2.1 Biološki neuron [6]

Na slici 2.1 se nalazi biološki neuron, koji se sastoji od tijela u kojem se u središtu nalazi jezgra, a na rubovima tijela se nalaze dendriti koji primaju impulse od susjednih neurona. Prema desno se širi akson, na kraju kojeg se nalaze teledendroni sa završnim kvržicama koji šalju impulse dalje u susjedne neurone.

2.2. Umjetni neuron

Inspirirani biološkim modelom, 1943. godine Warren McCulloh i Walter Pitts definiraju matematički model umjetnog neurona prikazanog na slici 2.2.



Slika 2.2 Model umjetnog neurona (engl. perceptron)

Ulazi x_1, x_2, …, x_n predstavljaju dendrite biološkog neurona, te oni sadrže određene vrijednosti pobude prethodnog neurona. Težine w_1, w_2, …, w_n predstavljaju u kojoj jačini signal primljen kroz ulaz x_i utječe na neuron. Uz to postoji i težina koju predstavlja konstantan pomak (engl. *bias*) w_0.

$$net = \sum_{i=1}^{n} w_i x_i + w_0 \tag{2.1}$$

Tijelo stanice umjetnog neurona prema svojem izlazu generira sumu označenu sa *net* , te se ona računa prema izrazu 2.1. Konačni izlaz *o* ovisi o aktivacijskoj funkciji *o* = *f* (*net*).

2.2.1. Funkcije aktivacije

Prilikom konstrukcije modela neuronske mreže možemo koristiti različite aktivacijske funkcije kao što su to funkcija skoka, funkcija identiteta, te različite sigmoidalne funkcije. Kao primjer aktivacijske funkcije možemo uzeti *funkciju skoka* , neuron koji onda dobivamo poznat je kao *TLU-perceptron* , te funkciju onda definiramo kao

$$step(net) = \begin{cases} 0, & net < 0 \\ 1, & net \geq 0 \end{cases} \tag{2.2}$$

Aktivacijska funkcija korištena u rekurentnim neuronskim mrežama je hiperbolička tangens funkcija (*tanh*):

$$tanh(net) = \frac{e^{net} - e^{-net}}{e^{net} + e^{-net}} \tag{2.3}$$

, *tanh* aktivacijska funkcija vraća vrijednosti između -1 i 1. Osim *tanh* funkcije, LSTM model interno koristi i *sigmoidnu* funkciju oblika:

$$\sigma(net) = \frac{1}{1 - e^{-net}} \tag{2.4}$$

, σ funkcija vraća vrijednosti između 0 i 1. Aktivacijske funkcije *tanh* i σ će nam biti bitne kod modela povratnih neuronskih mreža.

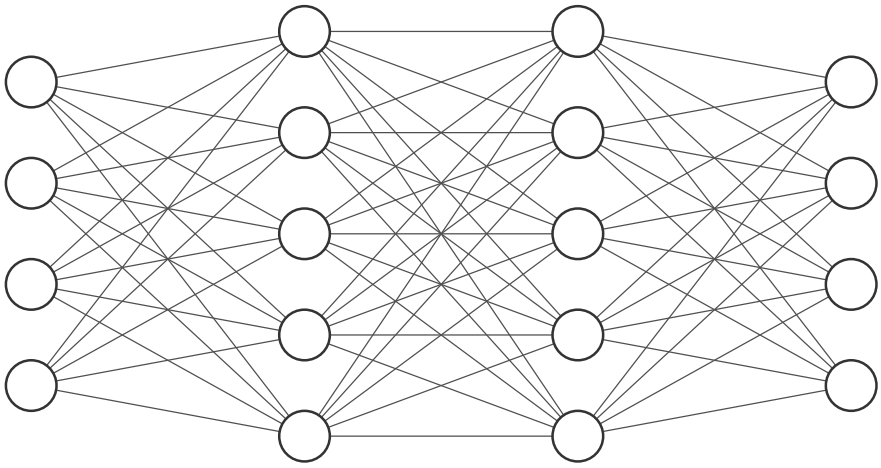
2.2.2. Vrste neuronskih mreža

Iako neuronske mreže možemo podijelite na dosta načina, ugrubo ih dijelimo na:

- Jednoslojne mreže bez povratnih veza (engl. *single-layer feedforward networks*)
- Višeslojne mreže bez povratnih veza (engl. *multi-layer feedforward networks*)
- Rekurentne neuronske mreže (engl. *recurrent neural networks - RNN*)
- Konvolucijske neuronske mreže (engl. *convolutional neural networks - CNN*)

2.3. Višeslojne mreže bez povratnih veza

Nakon definicije umjetnog neurona, samim time i jednoslojne mreže bez povratnih veza, možemo pričati o višeslojnom modelu umjetnih neuronskih mreža bez povratnih veza čija su oni glavna sastavnica.



Slika 2.3 Višeslojna neuronska mreža

Na slici 2.3 se nalazi jedna neuronska mreža sa četiri sloja. Ulazni sloj čine četiri neurona i kroz njih mreži predajemo ulazne podatke. Zatim ga slijede dva skrivena sloja, svaki po pet neurona. Te kao zadnji sloj imamo izlazni sloj sa četiri neurona iz kojih izvlačimo rezultate.

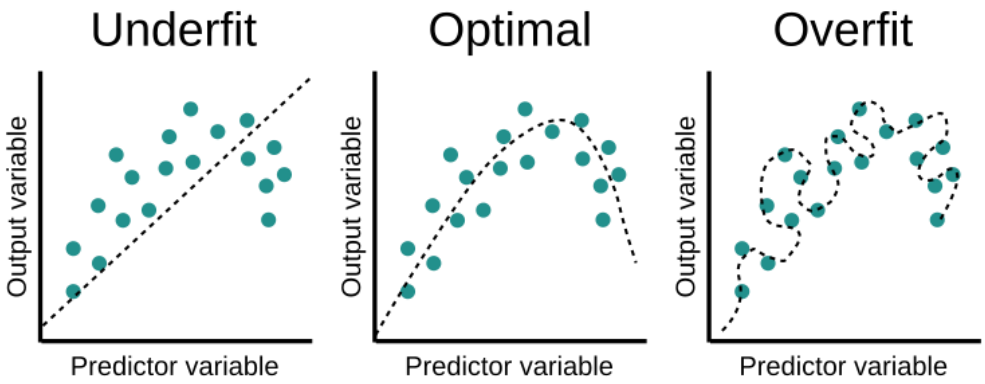
Generalno rad s umjetnim neuronskih mrežama dijelimo u dvije faze: fazu učenja i fazu iskorištavanja. U fazi učenja neuronskoj mreži predočavamo uzorke iz skupa za učenje, gdje je jedna *iteracija* predočavanje jednog uzorka skupa, a jedna *epoha* predočavanje čitavog skupa uzoraka. Uslijed učenja dolazi do promjena u jakosti veza između neurona, time se mreža onda prilagođava viđenim podacima, zatim ju u fazi iskorištavanja koristimo.

2.4. Učenje

Cilj faze učenja neuronske mreže je postići najbolju raspodjelu težina između neurona. Kao što je i navedeno, postoji više vrsti učenja neuronske mreže, dok ćemo u implementaciji koristiti nadzirano učenje. Ulaz nadziranog učenja smo prethodno definirali kao par (x_i, y_i), gdje x_i predstavlja podatke o primjerku, a y_i predstavlja ciljnu vrijednost koju pridružuje tom primjerku.

2.4.1. Unakrsna provjera

Unakrsna provjera (engl. *cross-validation*) je način pristupa učenju modela strojnog učenja. Prilikom učenja modelu predajemo skup podataka koji nazivamo skupom za učenje (engl. *train set*), te pomoću tog skupa, sa funkcijom gubitka i optimizacijom neuronske mreže prilagođavamo težine između čvorova neuronske mreže prema zadanom skupu. Osim skupa za učenje, imamo skup za provjeru (engl. *test set*) koji nam služi za provjeru koliko dobro naš model radi. Za veći set podatak generalno uzimamo omjer 80% podataka za skup za učenje, a 20% podataka kao skup za provjeru. Za manji set podataka bolja podjela bi bila 70/30 ili 60/40. Važnost unakrsne provjere je ta da izbjegnemo prekomjerno prilagođavanje modela (engl. *overfitting*). Ako dođe do prekomjernog prilagođavanja našeg modela onda on loše generalizira, to jest na još neviđenim podacima će predviđati loše.



Slika 2.4 Nedovoljno, optimalno i prekomjerno prilagođavanja regresijske funkcije [2]

Osim problema prekomjernog prilagođavanja možemo i susresti problem prevelike generalizacije modela. (TODO objasnit preveliku generalizaciju ako potrebno)

2.4.2. Funkcija gubitka

Prilikom učenja neuronske mreže, u svakoj iteraciji izlaz uspoređujemo sa ciljnim izlazom, te kako bismo „ocijenili“ koliko dobro mreža procjenjuje ciljnu vrijednost *y*, koristimo funkciju gubitka (engl. *loss function*) označenu sa *L*. Za odabir funkcije gubitka imamo više različitih opcija poput:

- Regresijske funkcije gubitka
  - Srednja kvadratna pogreška (engl. *Mean squared error loss*)
  - Srednja kvadratna logaritamska pogreška (engl. *Mean squared logarithmic error loss*)
  - Srednja apsolutna pogreška (engl. *Mean absolute error loss*)
- Binarne klasifikacijske funkcije gubitka
  - Gubitak zglobnice (engl. *Hinge loss*)
  - Kvadrirani gubitak zglobnice (engl. *Squared hinge loss*)
  - Binarna unakrsna entropija (engl. *Binary cross-entropy*)
- Višeklasne klasifikacije funkcije gubitka

Pošto se u ovom radu implementirani model bavi regresijom i kontinuiranim vrijednostima cijena dionica, koristimo funkciju srednje kvadratne pogreške (engl. *mean squared error - MSE*).

$$L = \frac{1}{N}\sum_{1}^{N}(y_i - \hat{y}_i)^2 \tag{2.5}$$

Formula (2.5) prikazuje izračun standardne pogreške kvadrata, gdje su $\hat{y}_i$ vrijednosti koje nam daje neuronska mreža, a y_i ciljne vrijednosti.

Nakon izračuna pogreške u svakoj iteraciji, cilj nam je prilagoditi neuronsku mrežu tako što mijenjamo težine veza između umjetnih neurona. Ovaj postupak nazivamo algoritmom propagacije pogreške unatrag (engl. *backpropagation algorithm*), odnosno postupak optimizacije neuronske mreže tako da minimiziramo funkciju gubitka.

2.4.3. Optimizacija neuronske mreže

Prilikom optimizacije mreže, odnosno smanjenja funkcije gubitka, imamo dvije mogućnosti optimizacije:

- Računamo gradijent na temelju svih primjera i zatim korigiramo težine veza
- Za određeni broj primjera (engl. *batch size*) računamo gradijent i odmah korigiramo težine veza

U implementaciji našega modela koristimo drugu navedenu metodu, odnosno za određeni broj primjera računamo prilagođeni gradijent i odmah korigiramo.

Za optimizaciju mreže postoje različiti algoritmi kao što su:

- Standardni gradijentni spust (engl. *gradient descent*)
- Stohastički gradijentni spust (engl. *stohastic gradient descent*)
- Adaptivni gradijent (engl. *adaptive gradient – AdaGrad*)
- AdaDelta
- RMSprop
- Adam

Prema provedenim eksperimentima [3], *Adam* nadmašuje ostale algoritme. Iako postoje argumenti da standardni gradijentni spust bolje generalizira [4], generalno najbolji za našu implementaciju je adaptivni optimizacijski algoritam *Adam* zbog najbrže konvergencije, te je memorijski efektivan za veliki set podataka, zbog toga ćemo i prilikom implementacije modela koristiti njega. Izgled formule za standardni gradijentni spust je sljedeći:

$$\theta = \theta - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \tag{2.6}$$

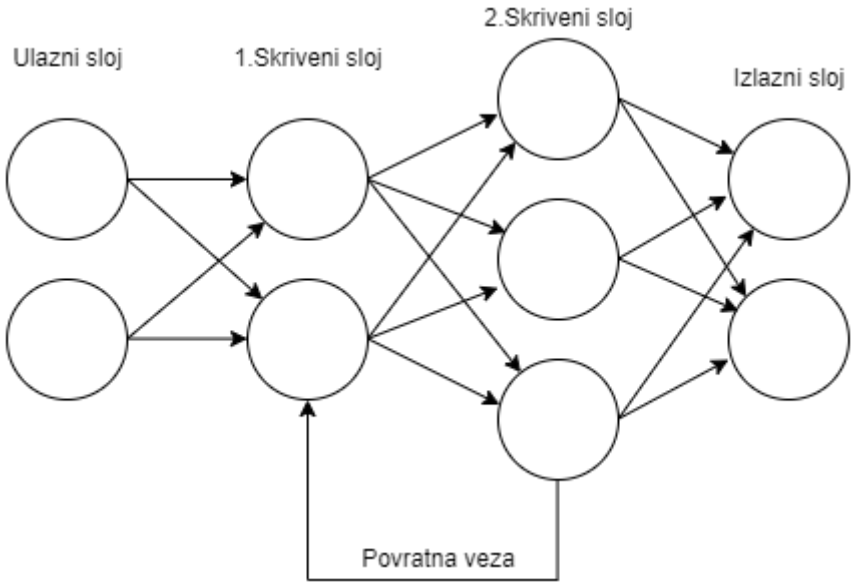
, gdje α predstavlja *stopu učenja*, a $\frac{\partial}{\partial \theta_j} J(\theta)$ predstavlja gradijent funkcije u θ_j.

Iako ga koristimo kasnije u samoj implementaciji, matematičku pozadinu *Adam* optimizacijskog algoritma nećemo navoditi u ovom radu zbog njene kompleksnosti. Za više o *Adam* optimizacijskom algoritmu i njegovoj matematičkoj pozadini pogledati [3].

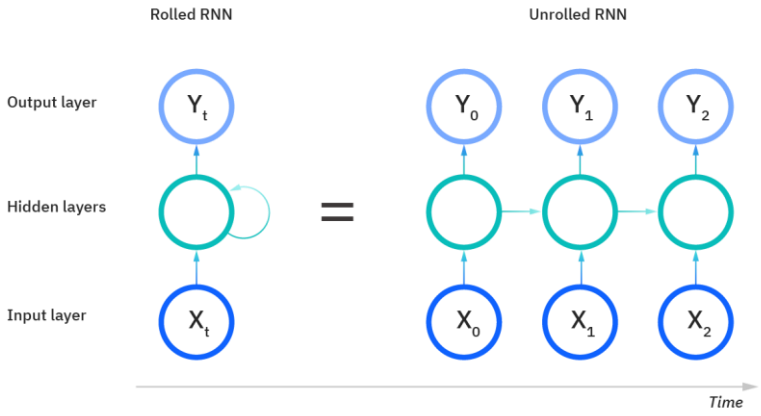
2.5. Povratne neuronske mreže

2.5.1. Definicija

Povratne ili rekurentne neuronske mreže su vrsta umjetnih neuronskih mreža koje sadrže povratne veze između neurona, što im omogućava privremeno dinamičko ponašanje. Povratne neuronske mreže inspiraciju dobivaju iz dinamičkih sustava. Koristimo ih jer su idealne kod modeliranja sekvencijskih podataka, kao što su to naprimjer tekst, zvuk, cijene dionica itd.



Slika 2.5 Rekurentna neuronska mreža



Slika 2.6 „Spakirana“ i „razmotana“ rekurentna neuronska mreža [5]

Osim slike 2.5, rekurentne neuronske mreže možemo prikazati kao i na slici 2.6, u njihovom „spakiranom“ stanju i „razmotanom“ stanju koji prikazuje neuronsku mrežu kroz vremenske korake *t*. Na slici 2.6 plavim čvorovima su označeni ulazi x_t, zelenim čvorovima skriveni slojevi h_t, te svjetloplave boje izlazi y_t.

Definirajmo h_t kao vektor skrivenog sloja u vremenskom koraku (engl. *timestep*) *t*, h_t vrijednosti onda računamo idućom formulom:

$$h_t = f(x_t, h_{t-1}) \tag{2.7}$$

Iz formule vidimo da stanje h_t ovisi o vrijednost stanja u prethodnom koraku h_{t-1} i o trenutačnoj vrijednosti vektora ulaza x_t. U jednostavnoj povratnoj neuronskoj mreži funkcija *f* je jednoslojna neuronska mreža.

$$net = W_h h_{t-1} + W_x x_t + b_h \tag{2.8}$$

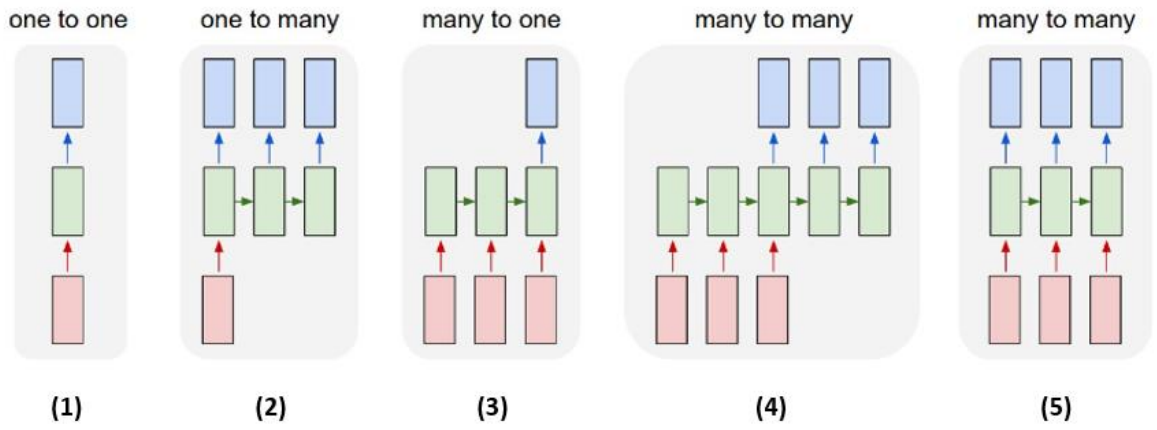
$$h_t = g(net) \tag{2.9}$$

Vrijednost *net* sloja h_t računamo prema formuli 2.8, gdje su W_h i W_x pripadajuće matrice težina između neurona, a b_h je vektor pristranosti, odnosno konstantni pomak. Za računanje projekcije u izlazni sloj koristimo sljedeću formulu:

$$y_t = g(W_y h_t + b_y) \tag{2.10}$$

Kod povratnih mreža za funkciju g uzimamo aktivacijske funkcije koje su prethodno definirane u poglavlju *2.2.1. Funkcije aktivacije*, a to su *tanh* (2.3), σ (2.4) i ReLU koju jednostavno definiramo kao $ReLU(net) = \max(0, net)$.

2.5.2. Vrste povratnih neuronskih mreža



Slika 2.7 Vrste povratnih mreža [7]

Na slici 2.7 pravokutnici predstavljaju vektore, a strelice predstavljaju funkcije (npr. matrično množenje). Ulazni vektori su crvene boje, izlazni plave boje, a zeleni predstavljaju RNN stanja. Slika označena sa **(1)** nam predstavlja fiksiran ulaz i izlaz bez povratnih veza, kao primjer problema koji u to spada je klasifikacija slika. Nadalje slika **(2)** prikazuje sekvencijski izlaz, u što spada opis slike riječima. Brojem **(3)** označavamo sekvencijski ulaz i fiksiran izlaz, npr. riječ moramo klasificirat kao „pozitivna“ ili „negativna“. Na slici **(4)** imamo sekvencijski ulaz i izlaz, npr. prijevod rečenice iz jednog jezika u drugi. I na zadnjoj slici **(5)** imamo usklađeni sekvencijski ulaz i izlaz, npr. klasifikacija svake sličice u videu. U svakom od navedenih slučaja uviđamo da nema ograničenja na količinu zelenih pravokutnika koji predstavljaju povratnu transformaciju. Detaljnije u [7].

2.5.3. Problemi običnih rekurentnih neuronskih mreža

U nedostatke i probleme povratnih neuronskih mreža spada zahtjevno i sporo računanje, učenje može biti kompleksno, mreža ne uzima buduće ulaze u svrhu odlučivanja, itd. No glavni problem običnih povratnih mreža je da pati od problema kratke memorije, ako je sekvenca ulaza duga, onda će mreža imati problem „prisjećanja“ ranijih ulaznih podataka. Konkretnije, za vrijeme propagacije pogreške unazad , dolazi do problema eksplodirajućeg i nestajućeg gradijent.

$$\left\| \frac{\partial h_i}{\partial h_{i-1}} \right\|_2 \ll 1 \tag{2.11}$$

$$\left\| \frac{\partial h_i}{\partial h_{i-1}} \right\|_2 \gg 1 \tag{2.12}$$

Problem nestajućeg gradijenta (2.11) je taj da kada želimo optimizirati mrežu nakon nekog ulaza i izračuna pogreške, gradijent između dva uzastopna skrivena stanje se eksponencijalno brzo smanji u vrijednosti i tako ne prilagodi ostatak mreže, tako onda kod tog dijela mreže ne dođe do prilagođavanja težina. Kod problema eksplodirajućeg gradijenta (2.12) naš gradijent eksponencijalno postane ogroman ili čak toliko velik da se ne može zapisati u memoriji računala (*NaN*). Više o problemima povratnih mreža i o matematičkoj pozadini problema nestajućeg i eksplodirajućeg gradijenta u [8].

Kao rješenje problema s kratkom memorijom i nestajućim/eksplodirajućim gradijentom koristimo podvrste povratnih mreža s propusnicama (engl. *gated recurrent neural networks*), kao što su *Long Short-Term Memory* (LSTM) i *Gated Recurrent Units* (GRU). Pošto u implementaciji koristimo LSTM model, on će biti detaljnije objašnjen u nastavku. Za detaljnije o povratnim mrežama s propusnicama pogledati [9].

2.5.4. LSTM

3. Izgradnja LSTM modela u svrhu predviđanja kretanja tržišta dionica

3.1. Opis problema predviđanja cijena dionica

4. Rezultati izgrađenog modela

Zaključak

Literatura

[1] Čupić, *Uvod u strojno učenje*, Zagreb, 2020

[2] Poveznica: https://www.educative.io/edpresso/overfitting-and-underfitting; pristupano 15.5.2022

[3] Kingma, D. P., Ba, J., *Adam: A method for stochastic optimization*., 2014

[4] Hardt, M., Recht, B., Singer, Y., *Train faster, generalize better: Stability of stochastic gradient descent*, *International Conference on Machine Learning*, 2016

[5] Poveznica: https://www.ibm.com/cloud/learn/recurrent-neural-networks; pristupano 20.5.2022

[6] Čupić, *Umjetne neuronske mreže*, Zagreb, 2016

[7] Andrej Karptahy, poveznica: http://karpathy.github.io/2015/05/21/rnn-effectiveness/; pristupano 22.5.2022

[8] Razvan Pascanu, Tomas Mikolov, Yoshua Bengio, *On the difficulty of training Recurrent Neural Networks*, 2012

[9] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, Yoshua Bengio, *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*, 2014

Sažetak

Naslov, sažetak, ključne riječi (na hrvatskom jeziku)

Sažetak opisuje sadržaj rada, prepričan u stotinjak riječi.

Summary

Title, summary, keywords (na engleskom jeziku)

Skraćenice

LSTM	*Long short-term memory*	naziv podvrste povratnih mreža
RNN	*Recurrent neural networks*	povratne neuronske mreže
CNN	*Convolutional neural networks*	konvolucijske neuronske mreže

Privitak