Razvoj rekurentne neuronske mreže i primena na analizi vremenskih serija

Seminarski rad u okviru kursa
Računarska inteligencija
Matematički fakultet

Kristina Pantelić, 91/2016, kristinapantelic@gmail.com
Nevena Mesar, 107/2015, mi15107@alas.matf.bg.ac.rs

17. jun 2020.

Sažetak

Danas je upotreba neuronskih mreža za rešavanje diverziteta računarskih problema u širokoj upotrebi. Za različite probleme koriste se različite vrste neuronskih mreža. U ovom radu čitalac će se upoznati sa terminom neurona, neuronske mreže, rekurentne neuronske mreže i primenom rekurentne neuronske mreže na analizi vremenskih serija.

Sadržaj

1 Uvod **2**

2 Rekurenta neuronska mreža **2**
2.1 Model rekurentne neuronske mreže 2
2.2 Algoritam rekurentne neuronske mreže 4

3 Implementacija algoritma **4**
3.1 Potrebne biblioteke 5
3.2 Podaci i predprocesiranje 5
3.3 Treniranje mreže 7
3.3.1 Inicijalizacija 7
3.3.2 Koraci algoritma 7
3.4 Testiranje 11
3.5 Primeri rezultata 11
3.5.1 Model `n = 54, m = 270, alpha = 0.5, eta = 0.3` 12
3.5.2 Model `n = 54, m = 270, alpha = 0.2, eta = 0.3` 13
3.5.3 Model `n = 54, m = 108, alpha = 0.2, eta = 0.3` 14
3.5.4 Model `n = 54, m = 108, alpha = 0.9, eta = 0.5` 15
3.5.5 Model `n = 54, m = 216, alpha = 0.2, eta = 0.3` 16

4 Zaključak **17**

Literatura **17**

1 Uvod

Algoritmi neuronskih mreža su nastali kao pokušaj da se oponaša način na koji ljudski mozak uči tj. način na koji ljudski mozak klasifikuje i spoznaje stvari. Neuronske mreže su razvijene tako da simuliraju neurone tj. mreže neurona mozga[4]. U tradicionalnoj neuronskoj mreži svi ulazi i izlazi su nezavisni jedni od drugih, ali u slučajevima kada se, na primer, zahteva predviđanje sledeće reči u rečenici, zahteva se da imamo informaciju o prethodnoj reči i da je na neki način zapamtimo[2]. Ovaj problem je bio motivacija za pojavljivanje i razvijanje rekurentne neuronske mreže. Rekurentna neuronska mreža (RNN) ima "memoriju" koja pamti sve informacije koje je prethodno izračunala tj. naučila. Treba imati na umu da tradicionalna neuronska mreža takođe pamti informacije, ali samo one koje je naučila tokom treninga. Na primer, ako klasifikator nauči zavisnosti ulaznih od izlaznih podataka tokom treninga, koristiće to znanje da klasifikuje test instance. Dok RNN takođe uči prilikom treninga, dodatno, ona pamti informacije naučene iz prethodnih trening instanci da bi generisala izlazne podatke[3]. Ovo svojstvo pamćenja informacija kroz vreme omogućava RNN mreži predikciju u oblasti vremenskih serija[2], kojima u ovom radu posvećujemo pažnju u kontekstu primene RNN.

Na samom početku ovog rada upoznaćemo se sa modelom naše rekurentne neuronske mreže, a u daljem tekstu ćemo se upoznati sa korišćenim skupom podataka vremenskih serija i primenom RNN na skup.

2 Rekurenta neuronska mreža

Postoji više vrsta neuronskih mreža koje, svaka zbog svojih specifičnih karakteristika, daju dobre rezultate na specifičnim poljima istraživanja podataka. Vrsta mreže koja je tema našeg rada je rekurentna neuronska mreža, a naša koncentracija je na razvijanju Jordanove strukturne rekurentne neuronske mreže (eng. *Jordan SRNN*).

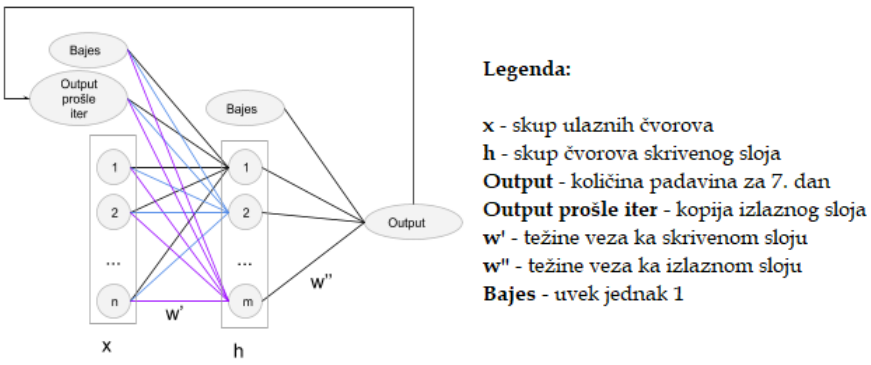
Jordanova rekurentna neuronska mreža je zasnovana na konceptu koji podrazumeva ponovno korišćenje dobijenih izlaznih vrednosti mreže kao dodatnih informacija za usmeravanje pretrage i izračunavanje novih vrednosti rezultata. Pored ulaznih instanci, dodaju se novih k vrednosti koje predstavljaju vrednosti k rezultata. Dakle, kopija izlaznog sloja se sprovodi na ulaz.

Naš cilj je konstruisati rekurentu neuronsku mrežu sa jednim skrivenim slojem koja će za 6 unetih uzastopnih dana prognozirati količinu padavina za 7. dan. U nastavu ćemo predstaviti model ove mreže u skladu sa brojem naših izlaznih vrednosti (jedan izlazni podatak) i odgovarajuće oznake korišćene u kôdu za implementaciju mreže.

2.1 Model rekurentne neuronske mreže

Neuronska mreža se sastoji od 3 sloja koju čine ulazni, skriveni i izlazni sloj. Svaki od slojeva ima odgovarajući broj neurona tj. promenljivih. Ulazni sloj se sastoji od $n+p$ ulaznih promenljivih $x_1^{(l)}$, $x_2^{(l)}$, ..., $x_n^{(l)}$,$o_1^{(l-1)}$, $o_2^{(l-1)}$, ..., $o_p^{(l-1)}$, uz dodatnu promeljivu $x_0^{(l)}$, koja se naziva bajes i čija je vrednost uvek jednaka 1, za svako l. Promenljiva l u superskriptu promenljive x_i predstavlja redni broj instance u skupu odabranih instanci za treniranje mreže. Radi jednostavnosti sve ulazne promenljive označavamo sa $x_i \in [0, n+p]$. Skriveni sloj čine m neurona $h_1^{(l)}$, $h_2^{(l)}$, ..., $h_m^{(l)}$ uz dodatnu promenljivu $h_0^{(l)} = 1$, za svako l, koja predstavlja bajes za skriveni sloj mreže. Izlazni sloj čine p neurona, čiji su izlazi $o_1^{(l)}$, $o_2^{(l)}$, ..., $o_p^{(l)}$. Svi neuroni ulaznog sloja su spojeni sa svim neuronima skrivenog sloja, osim bajesa, vezama čije su težine w'_{ij}, $0 \leq i \leq n+p$, $0 \leq j \leq m$. Svi neuroni skrivenog sloja su povezani sa svim neuronima izlaznog sloja vezama sa težinama w''_{jk}, $0 \leq j \leq m$, $1 \leq k \leq p$.

Nakon završavanja algoritma, cilj je da izlazne vrednosti budu što bliže izlaznim vrednostima y_1, y_2,...,y_p.



Slika 1: Jordanova RNN sa jednim skrivenim slojem

Proces učenja se sastoji od više iteracija, gde se svaka iteracija izvršava u nekoliko koraka. Najpre se, za sve j, $1 \leq j \leq m$, izračunava vrednost:

$$u'_j = \sum_{i=0}^{n+p} x_i w'_{ij} \tag{1}$$

i $h_j = f(u'_j)$, gde je $f(x) = (1 + e^{-x})^{-1}$ sigmoidna funkcija. Sigmoidna funkcija predstavlja aktivacionu funkciju neurona skrivenog i izlaznog sloja. Analogno kao kod skrivenog sloja, određuju se vrednosti izlaza iz neurona izlaznog sloja. Tako se dobija

$$u''_k = \sum_{j=0}^{m} h_j w''_{jk} \tag{2}$$

i $o_k = f(u''_k)$. Nakon odgovarajućeg broja iteracija, očekujemo da će vrednosti o_k biti približno jednake željenim izlaznim vrednostima y_k. Greška izlaznog sloja neurona k je definisana:

$$E_k = \frac{1}{2}(y_k - o_k)^2. \tag{3}$$

Pri ažuriranju vrednosti w''_{jk}, važi $w''_{jk} = w''_{jk} + \Delta w''_{jk}$, gde je

$$\Delta w''_{jk} = -\eta \frac{\partial E_k}{\partial w''_{jk}} + \alpha \Delta w''_{jk}.$$

Parametri η i α, respektivno, mere uticaj parcijalnog izvoda greške E_k po w''_{jk} i prethodne vrednosti Δw"$_{jk}$. Kako je E_k funkcija od o_k, o_k funkcija od u''_k, a u''_k funckija čiji je jedan od argumenata w''_{jk}, prema pravilu o izvodu složene funkcije važi

$$\frac{\partial E_k}{\partial w''_{jk}} = \frac{\partial E_k}{\partial o_k} \frac{\partial o_k}{\partial u''_k} \frac{\partial u''_k}{\partial w''_{jk}} = -(y_k - o_k) f'(u''_k) h_j = -(y_k - o_k) o_k (1 - o_k) h_j$$

Iz poslednjeg reda sledi da je

$$\Delta w''_{jk} = \eta (y_k - o_k) o_k (1 - o_k) h_j + \alpha w''_{jk}$$

Greška neurona skrivenog sloja predstavlja sumu svih odgovarajućih grešaka neurona izlaznog sloja i određena je sa

$$E_j = \frac{1}{2}\sum_{k=1}^{p}(y_k - o_k)^2$$

.

Pri ažuriranju vrednosti w'_{ij}, važi $w'_{ij} = w'_{ij} + \Delta w'_{ij}$, gde je

$$\Delta w'_{ij} = -\eta\frac{\partial E_j}{\partial w'_{ij}} + \alpha\Delta w'_{ij}$$

.

Vrednost $\frac{\partial E_j}{\partial w'_{ij}}$ se može izračunati prema pravilu o izvodu složene funkcije na sledeći način:

$$\frac{\partial E_j}{\partial w'_{ij}} = \frac{\partial E_j}{\partial o_k}\frac{\partial o_k}{\partial u''_k}\frac{\partial u''_k}{\partial h_j}\frac{\partial h_j}{\partial u'_j}\frac{\partial u'_j}{\partial w'_{ij}}$$

$$= -\sum_{k=1}^{p}(y_k - o_k)f'(u''_k)w''_{jk}f'(u'_j)x_i$$

$$= -\sum_{k=1}^{p}(y_k - o_k)o_k(1-o_k)w''_{jk}h_j(1-h_j)x_i$$

Sledi da je

$$\Delta w'_{ij} = \eta x_i h_j(1-h_j)\sum_{k=1}^{p}(y_k - o_k)o_k(1-o_k)w''_{jk} + \alpha\Delta w'_{ij}[1]$$

2.2 Algoritam rekurentne neuronske mreže

Na osnovu prethodnog opisa, algoritam učenja se izvršava u sledećih osam koraka[1]:

1. Ulazni podaci se sastoje od parova vektora ($x_1^{(l)}$, $x_2^{(l)}$,..., $x_{n+p}^{(l)}$) i (y_1, y_2,..., y_p), gde x_i predstavljaju ulaze uključujući kopiju rezultata iz prethodne iteracije algoritma, a y_k željene izlaze za svaku ulaznu instancu. Skup ulaznih podataka se proširuje bajesom $x_0^{(l)} = 1, \forall l$ iz skupa podataka.
2. Inicijalizovati parametre η, α i odrediti kriterijum zaustavljanja. Inicijalizovati početne vrednosti za w'_{ij} i w''_{jk} i postaviti $\Delta w'_{ij} = \Delta w''_{jk} = 0$. Izabrati aktivacionu funkciju f (ovde je korišćena sigmoidna funkcija).
3. Izabrati novi par ulaznog i izlaznog vektora (novi test primer iz skupa za učenje).
4. Odrediti vrednosti u'_j i h_j. Postaviti $h_0^{(l)} = 1, \forall l$ iz skupa podataka.
5. Odrediti vrednosti u''_k i o_k. Ukoliko je ispunjen kriterijum zaustavljanja, prekinuti izvršavanje.
6. Odrediti $\Delta w''_{jk}$ i ažurirati vrednosti w''_{jk}.
7. Odrediti $\Delta w'_{ij}$ i ažurirati vrednosti w'_{ij}.
8. Preći na korak 3.

3 Implementacija algoritma

U primerima metoda izostavljene su metode korišćenje za logovanje.

3.1 Potrebne biblioteke

Za implementaciju algoritma oslonili smo se na neke od funkcionalnosti postojećih biblioteka:

- `numpy` – kako bismo optimizovali matrične operacije
- `pandas` – za učitavanje i manipulisanje skupom podataka
- `sklearn` – za podelu podataka na test i trening skup, predprocesiranje podataka i ocenu greški
- `matplotlib` – za iscrtavanje grafika

```
from sklearn.metrics import mean_absolute_error, mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from pandas import DataFrame, Series, read_csv, errors, concat, Grouper
from numpy import zeros, array, append, concatenate, multiply, vstack, matrix, around
from numpy.random import rand, seed
from matplotlib import pyplot as plt
```

3.2 Podaci i predprocesiranje

Podaci su preuzeti sa Meteo blue, za Švajcarski grad Basel. Skup podataka je namerno izabran tako da su nema nedostajućih podataka i da su svi dani u kontinuitetu. U realnosti to ne bi bio slučaj. Ovaj rad ne obuhvata obradu takvih slučajeva, ali je jedan od mogućih koraka za unapređenje projekta.
U pitanju je period počev od datuma 31.12.1990. do 31.12.2019, ukupno 11326 uzastopnih dana. Skup svih atributa koji su dostupni u originalnom skupu podataka se može videti u tabeli 1. Uklonjeni su atributi koji su osenčeni sivom pozadinom. Odlučili smo da ih uklonimo radi brzine izračunavanja, moguće ih je uključivati u cilju eksperimentisanja. `Timestamp` smo izbacili jer su podaci učitavani hronološki, pa je samim tim indeks u skupu podataka dovoljan.

Atributi su preimenovani tako da im je dodeljen kraći naziv. Ciljni atribut je `precipitation`, dok su ostali korišćeni za predviđanje.

```
colnames = {
    "Basel Temperature [2 m elevation corrected]": "tempMin",
    "Basel Temperature [2 m elevation corrected].1": "tempMax",
    "Basel Temperature [2 m elevation corrected].2": "tempMean",
    # "Basel Relative Humidity [2 m]": "rHumidMin",
    # "Basel Relative Humidity [2 m].1": "rHumidMax",
    "Basel Relative Humidity [2 m].2": "rHumidMean",
    "Basel Precipitation Total": "precipitation",
    "Basel Cloud Cover Total": "cloudCoverage",
    "Basel Evapotranspiration": "evapor",
    # "Basel Wind Speed [10 m]": "windSpeedMin",
    # "Basel Wind Speed [10 m].1": "windSpeedMax",
    "Basel Wind Speed [10 m].2": "windSpeedMean",
    # "Basel Soil Temperature [0-10 cm down]": "soilTempMin",
    # "Basel Soil Temperature [0-10 cm down].1": "soilTempMax",
    "Basel Soil Temperature [0-10 cm down].2": "soilTempMean",
    # "Basel Soil Moisture [0-10 cm down]": "soilMoistMin",
    # "Basel Soil Moisture [0-10 cm down].1": "soilMoistMax"
    "Basel Soil Moisture [0-10 cm down].2": "soilMoistMean"
}
```

Timestamp				
Temperature	℃	2 m elevation corrected	daily	Minimum
Temperature	℃	2 m elevation corrected	daily	Maximum
Temperature	℃	2 m elevation corrected	daily	Mean
Relative Humidity	%	2 m	daily	Minimum
Relative Humidity	%	2 m	daily	Maximum
Relative Humidity	%	2 m	daily	Mean
Precipitation Total	mm	sfc	daily	Summation
Cloud Cover Total	%	sfc	daily	Mean
Evapotranspiration	mm	sfc	daily	Summation
Wind Speed	km/h	10 m	daily	Minimum
Wind Speed	km/h	10 m	daily	Maximum
Wind Speed	km/h	10 m	daily	Mean
Wind Direction Dominant	°	10 m	daily	None
Temperature	℃	sfc	daily	Minimum
Temperature	℃	sfc	daily	Maximum
Temperature	℃	sfc	daily	Mean
Soil Temperature	℃	0-10 cm down	daily	Minimum
Soil Temperature	℃	0-10 cm down	daily	Maximum
Soil Temperature	℃	0-10 cm down	daily	Mean
Soil Moisture	fraction	0-10 cm down	daily	Minimum
Soil Moisture	fraction	0-10 cm down	daily	Maximum
Soil Moisture	fraction	0-10 cm down	daily	Mean
Vapor Pressure Deficit	hPa	2 m	daily	Minimum
Vapor Pressure Deficit	hPa	2 m	daily	Maximum
Vapor Pressure Deficit	hPa	2 m	daily	Mean

Tabela 1: Dostupni atributi, osenčeni se ne koriste

Podaci su podeljeni na test i trening skup u razmeri $3:7$, skup za test ima 3398 redova, dok je za trening korišćeno 7928 redova. Za podelu je korišćena `test_train_split` metoda iz `sklearn.model_selection`.

```
def test_train_split_mine():
    global df, x_train, y_train, x_test, y_test, test_size
    global g_y_colname, n_x_test, n_x_train
    Y = df[g_y_colname]
    X = df.drop(columns=[g_y_colname])
    x_train, x_test, y_train, y_test = train_test_split(
        X, Y, shuffle=False, test_size=test_size)
    n_x_train = x_train.shape[0]
    n_x_test = x_test.shape[0]
    x_test.reset_index(inplace=True, drop=True)
    y_test.reset_index(inplace=True, drop=True)
```

Izvršeno je skaliranje podataka pomoću `MinMaxScaler` iz `sklearn.preprocessing` modula tako da sve vrednosti upadnu u rang $[0.1, 0.9]$. Skaliranje testnog skupa vršilo se na osnovu skupa za trening. Ciljni atribut je takođe skaliran na taj interval sa intervala $[0, 100]$.

```
def scale_attributes():
    global x_train, x_test, y_train, y_test
```

```
    scaler = MinMaxScaler(feature_range=(0.1, 0.9))
    scaler.fit(x_train, y_train)
    x_train = scaler.transform(x_train)
    x_test = scaler.transform(x_test)

    y_train = f_scale_y(y_train)
    y_test = f_scale_y(y_test)
```

3.3 Treniranje mreže

Kao aktivacionu funkciju koristimo sigmoidnu funkciju.

```
def activation_f(u):
    return 1.0 / (1.0 + exp(-u))
```

3.3.1 Inicijalizacija

Za početak je potrebno inicijalizovati matrice težina i njihove derivate. Težinske matrice su popunjene proizvoljnim vrednostima. Naglasiti da je prva iteracija zbog dodavanja izlaznog čvora, i definisati kroz koliko paterna treba da prođemo.

```
init_iter = True
n = N * n_attrs + 1     # broj ulaznih cvorova + 1 za bias
n_x = len(x_train)
num_of_patterns = n_x - N
m += 1                  # broj cvorova skrivenog sloja + bias
w_ = rand(n, m) - 0.5
dw_ = zeros((n, m))
w__ = rand(m) - 0.5     # samo jedan output -> obican niz
dw__ = zeros(m)
o = zeros(num_of_patterns)
output_arr = array([])
day = 0
end_of_epoch = n_x - N - 1
```

Treniranje je realizovano kroz epohe. Jedna epoha predstavlja jedan prolazak kroz celokupan skup za trening. Na kraju svake epohe računaju se greške pomoću `mean_absolute_error`, `mean_squared_error` iz `sklearn.metrics`, kao i `root squared error` dobijena od `mse` i prikazuju na izlazu. Trenira se dok se ne dostigne maksimalan broj epoha koji je 20. Takođe, u prvoj epohi, središnjoj i poslednjoj pravi se grafik koji prikazuje prave vrednosti i predviđene za poslednjih 365 dana epohe. Za sve predviđene vrednosti koje su skalirane na pravu vrednost dale negativan broj smatraćemo da su 0.

Podrazumevane vrednosti parametara su $\eta = 0.3$ i $\alpha = 0.2$. Prilikom pokretanja programa moguće je postaviti ih na druge vrednosti.

Za broj čvorova skrivenog sloja testirano je par varijanti. Ostavljeno je da se računa po formuli:

```
m = (int)((N * n_attrs) * 4)
```

3.3.2 Koraci algoritma

Cilj je konstruisati neuronsku mrežu sa jednim skrivenim slojem koja će za 6 unetih uzastopnih dana prognozirati količinu padavina za 7. dan. Ideja korišćena kako bi se podaci maksimalno iskoristili je prikazana na slici 2.

Ulazni čvorovi su svi atributi 6 uzastopnih dana, tako da su dani jedan za drugim u nizu i poštuje se redosled atributa. Na prvom mestu je uvek bias, dok se na kraj dodavao

1	2	3	4	5	6	→	7
2	3	4	5	6	7	→	8
... ...							
7923	7924	7925	7926	7927	7928	→	7929

Slika 2: Paterni za treniranje mreže

izlaz prethodne iteracije. U prvoj, inicijalnoj iteraciji, izlazni čvor ne postoji pa se dodavala prazna lista.

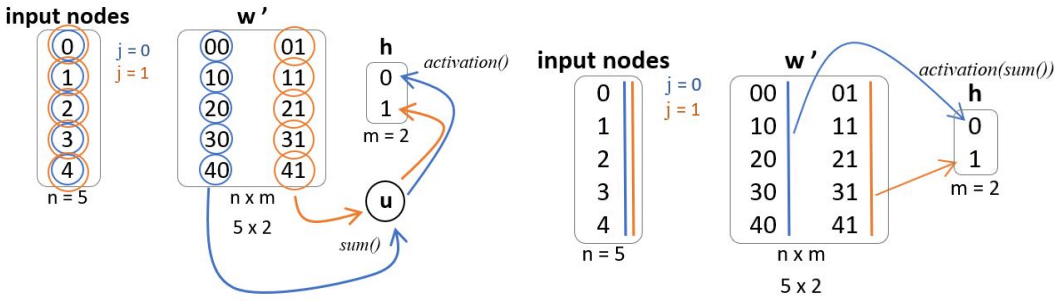
```
input_nodes = concatenate((bias,
                           (x_train[day:(day+N)]).reshape(-1),
                           output_arr))
```

U nastavku biće prikazana naivna implementacija i koraci kojima se došlo do sažimanja operacija i njihovog prevođenja u matrične i vektorske operacije biblioteke `numpy`. Odlučili smo da iskoristimo `numpy` biblioteku radi optimizacije vremenske složenosti osnovnih operacija algoritma.

Izracunavanje h1,..., hm cvorova skrivenog sloja

```
# Naivna implementacija
h[0] = 1.0
for j in range(1, m+1):
    u = 0
    for i in range(0, n+1):
        u += input_nodes[i] * w_[i][j]
    h[j] = activation_f(u)
```

Proces dolaska do poboljšane implementacije se svodi na sledeće interpretiranje naivne: *Za svaku kolonu težinske matrice pokoordinatno pomnoži kolonu sa ulaznim čvorovima, dobijene vrednosti sumiraj i provuci kroz aktivacionu funkciju.* Umesto da se fokusiramo na pojedinačne elemente matrice uopštili smo postupak koristeći osobine matrica i vektora. Grafički to možemo predstaviti na primeru manje dimenzije prikazanom na slici 3.



Slika 3: Primeri izračunavanja vrednosti čvorova skrivenog sloja. Naivni pristup sa leve i pristup koji koristi vektorske operacije sa desne strane.

`numpy` funkcija `multiply` radi pokoordinatno množenje koje nama u ovom slučaju treba. Takođe, bilo je potrebno transponovati težinsku matricu kako bi se dimenzije poklopile, i atributi množili svojim težinama. Prednost petlje u `python`-u jeste to što iterira po redovima, pa samim tim možemo automatski proći kroz težinsku matricu. Konačno dobijamo poboljšanu verziju ovog koraka:

```
# Poboljsana implementacija
u = []
for i in w_.T:
    u += [activation_f(sum(multiply(i, input_nodes)))]
h = array(u)
h[0] = 1  # bias ostaje nepromenjen
```

Izracunavanje o1,...,op

```
# Naivna implementacija
for k in range(1, p+1):
    u = 0
    for j in range(0, m+1):
        u += h[j] * w__[j][k]
    o[day] = activation_f(u)
```

U prvobitnoj implementaciji w'' težinsku matricu smo posmatrali kao matricu, kasnije smo sveli na niz pošto imamo samo jedan izlazni čvor, pa prema tome nema potrebe praviti matricu. Koristi se isti princip kao u prethodnom koraku, pokoordinatno se pomnože skriveni čvorovi sa težinskim vektorom u ovom slučaju, sumiraju se dobijeni rezultati i provuku kroz aktivacionu funkciju.

```
# Poboljsana implementacija
o[day] = activation_f(sum(multiply(h, w__)))
output_arr = array([o[day]])
```

Greška Greška je broj koji se pojavljuje u izračunavanju na više mesta: prilikom računanja `dh` i ažuriranja `dw__`. Zato je pogodno izdvojiti je u posebnu promenljivu i nadalje je koristiti.

```
error = (y_train[day+N] - o[day]) * o[day] * (1.0 - o[day])
```

Izracunavanje deltaH(j)

```
# Naivna implementacija
for j in range(1, m+1):
    dh[j] = 0.0
    for k in range(1, p+1):
        dh[j] += w__[j][k] * error
```

Umesto resetovanja vrednosti i njihovog ponovnog izračunavanja u postojećoj `dh`, možemo direktno iskoristiti vrednosti w'' težinskog vektora u našem slučaju i osobine da se prilikom množenja konstantom, svaki element vektora množi tom konstatom.

```
# Poboljsana implementacija
dh = w__ * error
```

Azuriranje W'(ij) i deltaW'(ij) Ovde smo imali više modifikacija u procesu poboljšanja koda. Počeli smo od bukvalne interpretacije koraka algoritma i započeli njegovo parčanje.

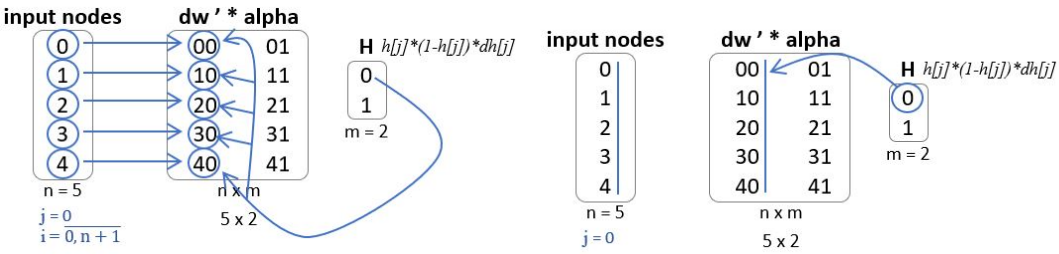
```
# Bukvalna implementacija
for j in range(1,m+1):
    for i in range(0,n+1):
        dw_[i][j] = eta * input_nodes[i] * h[j] * (1 - h[j]) * dh[j]
                    + alpha * dw_[i][j]
        w_[i][j] += dw_[i][j]
```

Ono što prvo možemo da primetimo jeste da je kolona fiksirana sa `j` što znaci da se svaki red mnozi sa samo jednom vrednosti iz kolone, tako da izracunavanje koje je vezano za `j` možemo izdvojiti. Takođe, u to izračunavanje uključujemo i `eta` pošto je to konstantna vrednost.

```
# Naivna implementacija
for j in range(1, m+1):
    H = h[j] * (1 - h[j]) * dh[j]
    for i in range(0, n+1):
        dw_[i][j] = eta * input_nodes[i] * H + alpha * dw_[i][j]
        w_[i][j] += dw_[i][j]
```

Sledeće što primećujemo jeste da je možemo ovaj korak svesti samo na jednu petlju ukoliko `H` posmatramo kao vektor kome se vrednosti ne menjaju u zavisnosti od redova težinske matrice. Pokoordinatno će se pomnožiti vrednosti u vektoru i odatle je dovoljno da izdvajamo vrednosti na odgovarajućim indeksima.
Što se ažuriranja `dw_` tiče, zaključujemo da se svaki element u redu množi sa `alpha` i nakon toga mu se dodaje vrednost proizvod `eta` sa ulaznim čvorom za taj red kao i jedna od vrednosti iz `H`. Ova operacija se može svesti prvo na množenje matrice konstantom `alpha` i potom prolazak petljom kroz njene kolone kako bi se sabrali odgovarajući elementi. Primer se može videti na manjoj dimenziji na slici 4.
Na kraju imamo sabiranje matrica kako bismo ažurirali i `w_`.

```
# Poboljsana implementacija
H = h * (1-h) * dh * eta
dw_ *= alpha
for j in range(0, m):
    dw_.T[j] += (input_nodes * H[j])
w_ += dw_
```



Slika 4: Ažuriranje `dw_`. Naivni pristup sa leve strane i poboljšani sa desne.

Azuriranje W"(jk) i deltaW"(jk)

```
# Naivna implementacija
for k in range(1, p):
    for j in range(0, m):
        dw__[j][k] = eta * h[j] * error + alpha * dw__[j][k]
        w__[j][k] += dw__[j][k]
```

`dw__` i `w_` su u našem slučaju vektori pošto imamo samo jedan izlaz. Samim tim je dovoljno primeniti pokoordinatne operacije nad vektorima.

```
# Poboljsana implementacija
dw__ *= alpha
dw__ += (eta * h * error)
w__ += dw__
```

Ukoliko smo prošli prvu iteraciju dodajemo još jedan red u težinsku matricu `w_` pošto nam se povećao broj ulaznih čvorova za 1.

Implementirano je čuvanje modela koji je imao najmanju `mse` tako što se upišu težinske matrice u tekstualni fajl koji je prilikom ponovnog pokretanja programa moguće učitati.

3.4 Testiranje

Nakon treniranja mreže ili učitavanja postojećeg modela vrši se evaluacija modela. Prolazi se kroz ceo test skup po sličnom principu po kome smo trenirali mrežu.

```
input_nodes = concatenate(
            (bias, (x_test[day:(day+N)]).reshape(-1), output_arr))
day += 1
# Izracunavanje h1,..., hm cvorova skrivenog sloja
# U prvoj iteraciji nemamo output cvor tako da
# ne koristimo celu w' matricu
u = []
if init_iter:
    for i in best_w_[:-1].T:
        u += [activation_f(multiply(i, input_nodes).sum())]
    init_iter = False
else:
    for i in best_w_.T:
        u += [activation_f(multiply(i, input_nodes).sum())]
h = array(u)
h[0] = 1  # bias ostaje nepromenjen

# Izracunavanje o1,...,op
o[day] = activation_f(sum(multiply(h, best_w__)))
output_arr = array([o[day]])
```

Računa se `mean_sqared_error`, `mean_absolute_error` i prikazuje na izlazu. Generiše se grafik na kome su prikazane prave i predviđene vrednosti.

3.5 Primeri rezultata

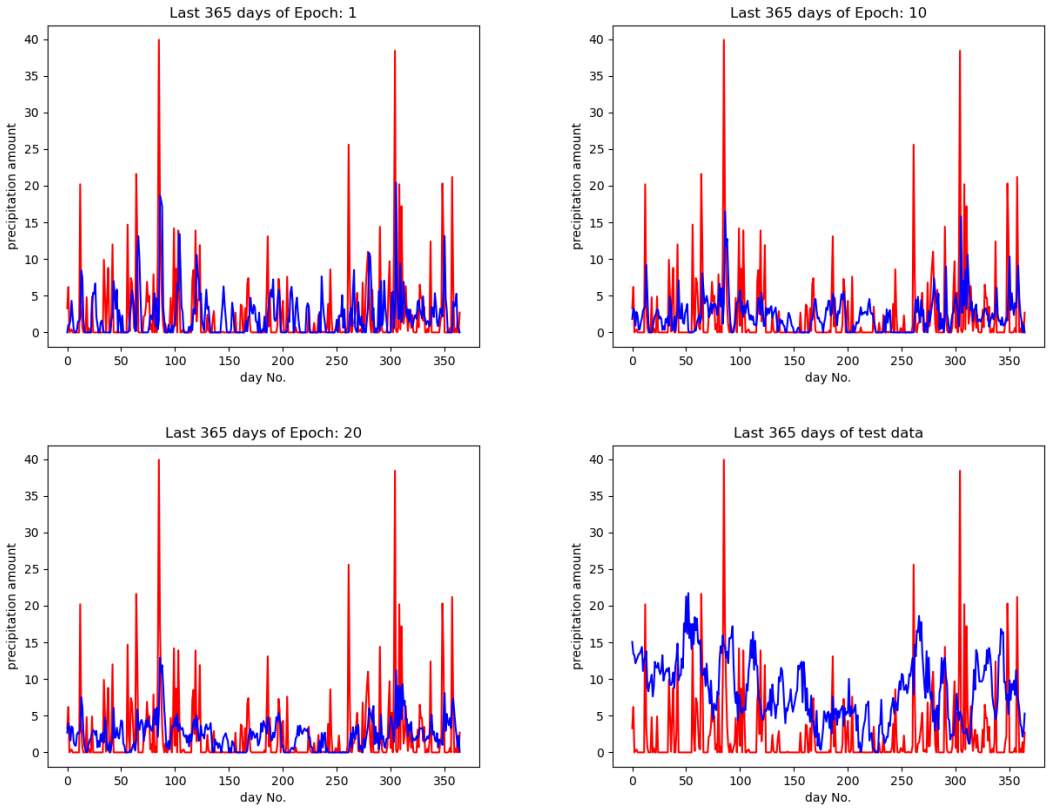
Prilikom testiranja imali smo razne kombinacije, od kojih će biti prikazano par. U daljem tekstu imaćemo sledeće oznake: `n` – broj ulaznih čvorova, `m` – broj čvorova skrivenog sloja. Na svakoj strani je prikazan izlaz programa za različite kombinacije parametara. Svaki grafik epohe sadrži poslednjih 365 dana u toj epohi. Isto važi i za prikaz grafika test skupa.

Na graficima je crvenom bojom predstavljen skup tačnih vrednosti, a plavom su označene vrednosti predviđane programom.

3.5.1 Model `n = 54, m = 270, alpha = 0.5, eta = 0.3`

```
broj dana za predikciju= 6; broj ulaznih cvorova= 54; broj cvorova skriveni sloj= 270
alpha = 0.5 eta = 0.3
epoch:   0/20   mse: 0.00230558769024192      mae: 0.030745924289642186      rmse: 0.048016535591834615
epoch:   1/20   mse: 0.0020691819015563008    mae: 0.02903043113699024       rmse: 0.04548826114017001
epoch:   2/20   mse: 0.001938548994169053     mae: 0.027713894391377336      rmse: 0.04402895631478281
epoch:   3/20   mse: 0.0018782308177129618    mae: 0.02718125800641104       rmse: 0.04333856040194416
epoch:   4/20   mse: 0.0018541484309476885    mae: 0.02706152932361909       rmse: 0.04305982386108527
epoch:   5/20   mse: 0.0018472157254949184    mae: 0.02711965092474976       rmse: 0.04297924761434195
epoch:   6/20   mse: 0.001847096882086492     mae: 0.027240328981882785      rmse: 0.04297786502476004
epoch:   7/20   mse: 0.0018484149606793445    mae: 0.02736813761528103       rmse: 0.04299319667900195
epoch:   8/20   mse: 0.0018486031378729 86    mae: 0.027468230387172166      rmse: 0.04299538507645891
epoch:   9/20   mse: 0.0018466459437720528    mae: 0.027535487234559175      rmse: 0.04297261853520277
epoch:  10/20   mse: 0.0018423689617472       mae: 0.027561658799244805      rmse: 0.04292282564961445
epoch:  11/20   mse: 0.0018360429200479063    mae: 0.027550815717757516      rmse: 0.042849071402399214
epoch:  12/20   mse: 0.0018281416207006791    mae: 0.027509304146725484      rmse: 0.0427567728050268
epoch:  13/20   mse: 0.0018191847531010947    mae: 0.02745052855455493       rmse: 0.042651902104139446
epoch:  14/20   mse: 0.0018096434011468487    mae: 0.02737767645017048       rmse: 0.04253990363349274
epoch:  15/20   mse: 0.0017998957765097856    mae: 0.0272990199920374        rmse: 0.042425178567800814
epoch:  16/20   mse: 0.0017902188217654774    mae: 0.027220601868053973      rmse: 0.04231097755624984
epoch:  17/20   mse: 0.0017808003071023023    mae: 0.027141672344482493      rmse: 0.042199529702382964
epoch:  18/20   mse: 0.0017717585383886071    mae: 0.02706356552362999       rmse: 0.042092262215146
epoch:  19/20   mse: 0.0017631615172861157    mae: 0.02698921856530739       rmse: 0.041990016876468576
epoch:  20/20   mse: 0.001755042129148845     mae: 0.026918010213622875      rmse: 0.041893222950124584
Cuvanje modela: model10_34_23.txt
Uspesno cuvanje modela...
Da li zelite da ucitate neki od postojecih modela? Inace ce se raditi sa trenutnim. (Y/N)
n
MSE - test: 0.007809762722696304      MAE - test: 0.06803416142550417
```

Slika 5: Output programa za parametre `m = 270, alpha = 0.5, eta = 0.3`

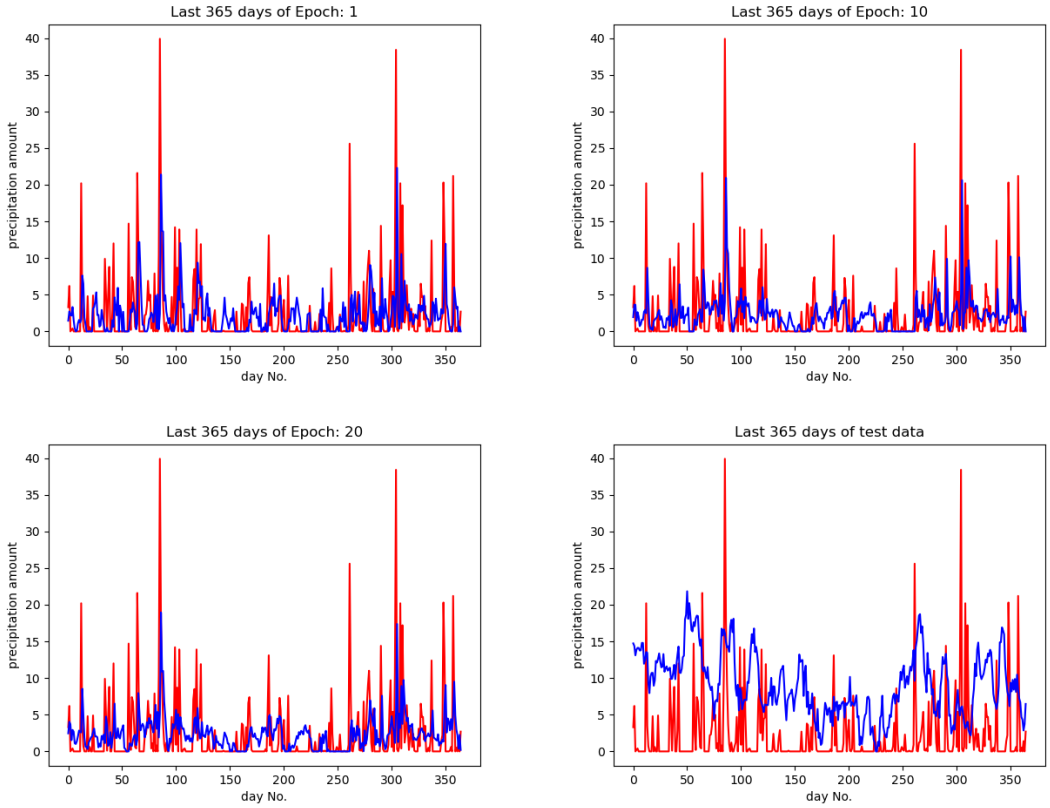


Slika 6: Grafici za parametre `m = 270, alpha = 0.5, eta = 0.3`

3.5.2 Model `n = 54, m = 270, alpha = 0.2, eta = 0.3`

```
broj dana za predikciju= 6; broj ulaznih cvorova= 54; broj cvorova skriveni sloj= 270
alpha = 0.2 eta = 0.3
epoch:   0/20   mse: 0.0020269406544871807     mae: 0.028004053018290072      rmse: 0.045021557663936736
epoch:   1/20   mse: 0.001890504081243114      mae: 0.026942564149147627      rmse: 0.04347992733714161
epoch:   2/20   mse: 0.0018196302392409823     mae: 0.026221842554555657      rmse: 0.04265712413232967
epoch:   3/20   mse: 0.0017822232490548485     mae: 0.025900933803923828      rmse: 0.04221638602550967
epoch:   4/20   mse: 0.0017627793303375927     mae: 0.025788196477321963      rmse: 0.041985465703474015
epoch:   5/20   mse: 0.0017530898416978208     mae: 0.025774947636076753      rmse: 0.04186991571161591
epoch:   6/20   mse: 0.0017484690989736473     mae: 0.025805534939315693      rmse: 0.041814699556180564
epoch:   7/20   mse: 0.0017462094607348239     mae: 0.0258565543824413        rmse: 0.041787671157110726
epoch:   8/20   mse: 0.0017447740486098808     mae: 0.02591221237660181       rmse: 0.04177049255886122
epoch:   9/20   mse: 0.0017433371103788757     mae: 0.02596884453143699       rmse: 0.04175328861753138
epoch:  10/20   mse: 0.0017415055255598313     mae: 0.02601497775557538       rmse: 0.041731349433727055
epoch:  11/20   mse: 0.0017391416649945803     mae: 0.02604763713138203       rmse: 0.041703017456709054
epoch:  12/20   mse: 0.001736249819904523      mae: 0.02606718475596866       rmse: 0.04166833113894199
epoch:  13/20   mse: 0.0017329051905294111     mae: 0.02607708128901555       rmse: 0.04162817784301171
epoch:  14/20   mse: 0.0017292112152060485     mae: 0.02608050394965929       rmse: 0.041583785484321276
epoch:  15/20   mse: 0.0017252751760931007     mae: 0.026076631202983366      rmse: 0.041536431913358915
epoch:  16/20   mse: 0.0017211952721594226     mae: 0.026067270878615822      rmse: 0.041487290489491145
epoch:  17/20   mse: 0.001717054744560637      mae: 0.026055096018513333      rmse: 0.041437359285560624
epoch:  18/20   mse: 0.0017129202224039387     mae: 0.026042036435217755      rmse: 0.04138744039444743
epoch:  19/20   mse: 0.0017088424555715819     mae: 0.026028215969460994      rmse: 0.041338147703683846
epoch:  20/20   mse: 0.0017048582457663906     mae: 0.026013956321552966      rmse: 0.04128992910827519
Cuvanje modela: model11_20_22.txt
Uspesno cuvanje modela...
Da li zelite da ucitate neki od postojecih modela? Inace ce se raditi sa trenutnim. (Y/N)
n
MSE - test: 0.008091158261669746      MAE - test: 0.07069133506849452
```

Slika 7: Output programa za parametre `m = 270, alpha = 0.2, eta = 0.3`

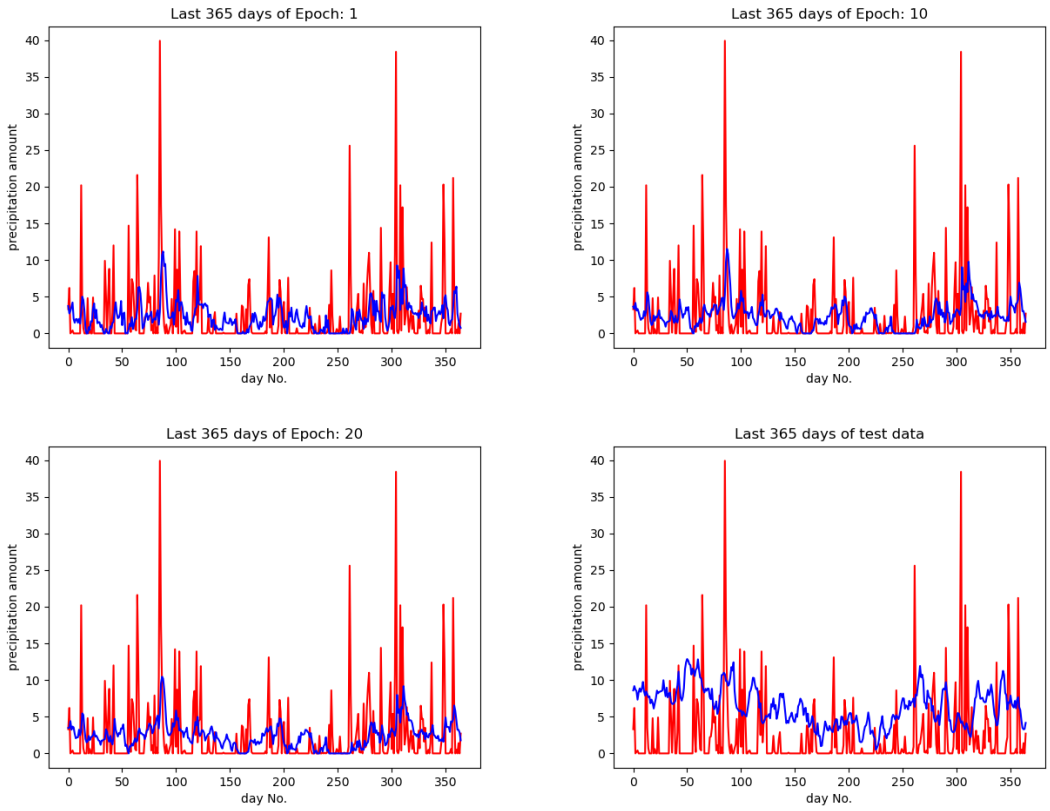


Slika 8: Grafici za parametre `m = 270, alpha = 0.2, eta = 0.3`

3.5.3 Model `n = 54, m = 108, alpha = 0.2, eta = 0.3`

```
broj dana za predikciju= 6; broj ulaznih cvorova= 54; broj cvorova skriveni sloj= 108
alpha = 0.2 eta = 0.3
Ucitati model? (Y/N)
n
Zapoceto treniranje...
epoch:   0/20   mse: 0.0018333969136350868     mae: 0.026933730168636608      rmse: 0.0428181843804135
epoch:   1/20   mse: 0.0016754780687101315     mae: 0.025783809351965038      rmse: 0.04093260398154668
epoch:   2/20   mse: 0.0016590487256534369     mae: 0.02568695221407384       rmse: 0.040731421846695176
epoch:   3/20   mse: 0.0016544457597040723     mae: 0.025708908399360842      rmse: 0.04067487873004752
epoch:   4/20   mse: 0.0016543808344862826     mae: 0.02576578925655835       rmse: 0.040674080622508026
epoch:   5/20   mse: 0.001655983693567097      mae: 0.025823072251350093      rmse: 0.04069377954389463
epoch:   6/20   mse: 0.001657923819381214      mae: 0.02587356628509388       rmse: 0.040717610678688086
epoch:   7/20   mse: 0.001659587624372455      mae: 0.025915958927055307      rmse: 0.04073803657974271
epoch:   8/20   mse: 0.0016607316316972973     mae: 0.025948983750714708      rmse: 0.0407520751827106
epoch:   9/20   mse: 0.001661298631963185      mae: 0.025973995917623664      rmse: 0.04075903129323837
epoch:  10/20   mse: 0.0016613203279609002     mae: 0.025990148196568922      rmse: 0.04075929744194446
epoch:  11/20   mse: 0.0016608669737958744     mae: 0.02599940101815533       rmse: 0.04075373570356311
epoch:  12/20   mse: 0.0016600215298744568     mae: 0.026002650164185535      rmse: 0.04074336178906273
epoch:  13/20   mse: 0.0016588664753508535     mae: 0.026000842992122776      rmse: 0.040729184565258036
epoch:  14/20   mse: 0.0016574772979276228     mae: 0.025995562091115195      rmse: 0.04071212716043738
epoch:  15/20   mse: 0.0016559196408595365     mae: 0.025987549434830667      rmse: 0.04069299252770109
epoch:  16/20   mse: 0.0016542485113777163     mae: 0.02597756267861766       rmse: 0.040672453963065916
epoch:  17/20   mse: 0.0016525086452444057     mae: 0.02596627872154472       rmse: 0.040651059583292605
epoch:  18/20   mse: 0.0016507354798691062     mae: 0.025954102295667428      rmse: 0.04062924414592408
epoch:  19/20   mse: 0.0016489563946454236     mae: 0.025941314477399502      rmse: 0.04060734409740957
epoch:  20/20   mse: 0.001647192008298528      mae: 0.02592846928388935       rmse: 0.040585613316771844
Cuvanje modela: model14_14_49.txt
Uspesno cuvanje modela...
MSE - test: 0.003506690181926135      MAE - test: 0.04771320812085592
```

Slika 9: Output programa za parametre `n = 54, m = 108, alpha = 0.2, eta = 0.3`

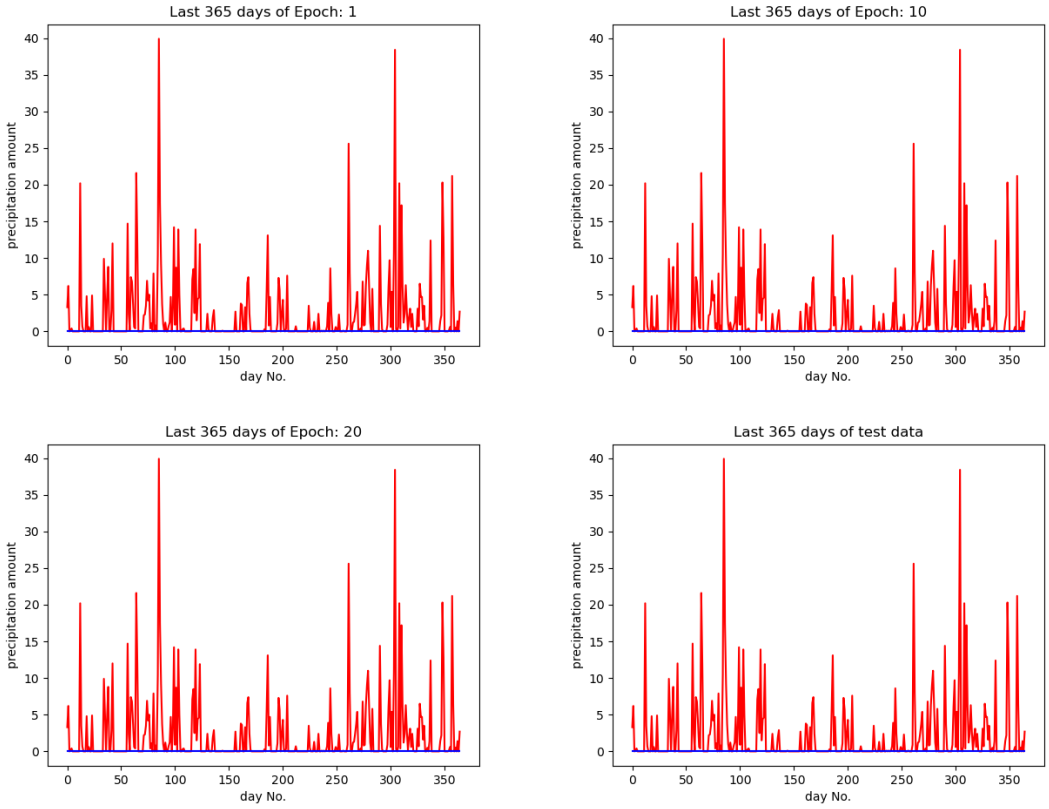


Slika 10: Grafici za parametre `n = 54, m = 108, alpha = 0.2, eta = 0.3`

3.5.4 Model `n = 54, m = 108, alpha = 0.9, eta = 0.5`

```
broj dana za predikciju= 6; broj ulaznih cvorova= 54; broj cvorova skriveni sloj= 108
alpha = 0.9 eta = 0.5
Ucitati model? (Y/N)
n
Zapoceto treniranje...
epoch:   0/20   mse: 0.016278322555309498      mae: 0.12094758357319745      rmse: 0.1275865296781345
epoch:   1/20   mse: 0.01623426739833596       mae: 0.12088270642205709      rmse: 0.12741376455601633
epoch:   2/20   mse: 0.01623426739833596       mae: 0.12088270642205709      rmse: 0.12741376455601633
epoch:   3/20   mse: 0.01623426739833596       mae: 0.12088270642205709      rmse: 0.12741376455601633
epoch:   4/20   mse: 0.01623426739833596       mae: 0.12088270642205709      rmse: 0.12741376455601633
epoch:   5/20   mse: 0.01623426739833596       mae: 0.12088270642205709      rmse: 0.12741376455601633
epoch:   6/20   mse: 0.01623426739833596       mae: 0.12088270642205709      rmse: 0.12741376455601633
epoch:   7/20   mse: 0.01623426739833596       mae: 0.12088270642205709      rmse: 0.12741376455601633
epoch:   8/20   mse: 0.01623426739833596       mae: 0.12088270642205709      rmse: 0.12741376455601633
epoch:   9/20   mse: 0.01623426739833596       mae: 0.12088270642205709      rmse: 0.12741376455601633
epoch:  10/20   mse: 0.01623426739833596       mae: 0.12088270642205709      rmse: 0.12741376455601633
epoch:  11/20   mse: 0.01623426739833596       mae: 0.12088270642205709      rmse: 0.12741376455601633
epoch:  12/20   mse: 0.01623426739833596       mae: 0.12088270642205709      rmse: 0.12741376455601633
epoch:  13/20   mse: 0.01623426739833596       mae: 0.12088270642205709      rmse: 0.12741376455601633
epoch:  14/20   mse: 0.01623426739833596       mae: 0.12088270642205709      rmse: 0.12741376455601633
epoch:  15/20   mse: 0.01623426739833596       mae: 0.12088270642205709      rmse: 0.12741376455601633
epoch:  16/20   mse: 0.01623426739833596       mae: 0.12088270642205709      rmse: 0.12741376455601633
epoch:  17/20   mse: 0.01623426739833596       mae: 0.12088270642205709      rmse: 0.12741376455601633
epoch:  18/20   mse: 0.01623426739833596       mae: 0.12088270642205709      rmse: 0.12741376455601633
epoch:  19/20   mse: 0.01623426739833596       mae: 0.12088270642205709      rmse: 0.12741376455601633
epoch:  20/20   mse: 0.01623426739833596       mae: 0.12088270642205709      rmse: 0.12741376455601633
Cuvanje modela: model14_26_29.txt
Uspesno cuvanje modela...
MSE - test: 0.016415549855460915      MAE - test: 0.12109056609589977
```

Slika 11: Output programa za parametre `m = 108, alpha = 0.9, eta = 0.5`

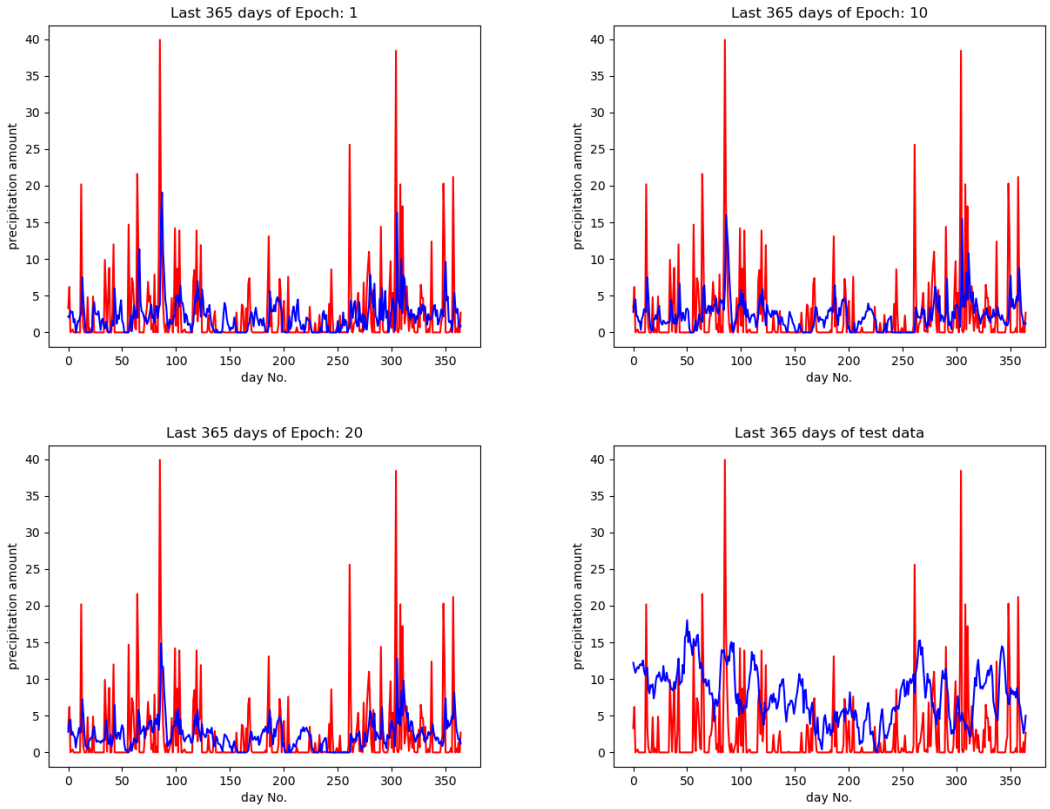


Slika 12: Grafici za parametre `m = 108, alpha = 0.9, eta = 0.5`

3.5.5 Model `n = 54, m = 216, alpha = 0.2, eta = 0.3`

```
broj dana za predikciju= 6; broj ulaznih cvorova= 54; broj cvorova skriveni sloj= 216
alpha = 0.2 eta = 0.3
Ucitati model? (Y/N)
n
Zapoceto treniranje...
epoch:   0/20   mse: 0.0021575027163934207     mae: 0.02652431359427752       rmse: 0.046448925890631966
epoch:   1/20   mse: 0.0017313560050236467     mae: 0.02581874779568006       rmse: 0.04160956626815097
epoch:   2/20   mse: 0.0016980976950275693     mae: 0.02556963193327894       rmse: 0.041207980962764595
epoch:   3/20   mse: 0.0016852775891562715     mae: 0.025536138780498437      rmse: 0.04105213257744683
epoch:   4/20   mse: 0.0016834115936480711     mae: 0.025610076577534314      rmse: 0.041029399138277314
epoch:   5/20   mse: 0.0016864911263050693     mae: 0.025723452596539614      rmse: 0.041066910357428515
epoch:   6/20   mse: 0.0016908278668649722     mae: 0.025836977136316         rmse: 0.04111967736820137
epoch:   7/20   mse: 0.0016944480036495896     mae: 0.02593223751307915       rmse: 0.04116367334980674
epoch:   8/20   mse: 0.0016965760280770999     mae: 0.026000441221906062      rmse: 0.041189513569318825
epoch:   9/20   mse: 0.0016971200202701494     mae: 0.026043706717404472      rmse: 0.041196116567828935
epoch:  10/20   mse: 0.0016963014217072916     mae: 0.026067881692874308      rmse: 0.04118617998439879
epoch:  11/20   mse: 0.0016944406496029875     mae: 0.026076416048813448      rmse: 0.04116358402281059
epoch:  12/20   mse: 0.001691851396522331      mae: 0.026073081517814237      rmse: 0.041132121225659284
epoch:  13/20   mse: 0.0016887977037938406     mae: 0.02606213812282413       rmse: 0.04109498392497363
epoch:  14/20   mse: 0.0016854834230127954     mae: 0.026046374292068134      rmse: 0.04105463948219245
epoch:  15/20   mse: 0.001682056578389527      mae: 0.026028069265398295      rmse: 0.04101288307824173
epoch:  16/20   mse: 0.0016786192645684816     mae: 0.026008373142093383      rmse: 0.04097095635408675
epoch:  17/20   mse: 0.0016752384087034553     mae: 0.02598816921477992       rmse: 0.04092967638161161
epoch:  18/20   mse: 0.0016719553351191461     mae: 0.025967897216678644      rmse: 0.040889550439191014
epoch:  19/20   mse: 0.0016687934545317527     mae: 0.025948199550662257      rmse: 0.040850868467289074
epoch:  20/20   mse: 0.0016657640735455374     mae: 0.025929346769125082      rmse: 0.04081377308636801
Cuvanje modela: model14_34_23.txt
Uspesno cuvanje modela...
MSE - test: 0.005802880771351822      MAE - test: 0.06029731664924966
```

Slika 13: Output programa za parametre `m = 216, alpha = 0.2, eta = 0.3`



Slika 14: Grafici za parametre `m = 216, alpha = 0.2, eta = 0.3`

4 Zaključak

Prema prikazanim modelima primećujemo da iako su greške modela sa različitim parametrima približne, maltene iste, nakon skaliranja podataka u njihov originalni rang vrednosti vidi se koliko je zapravo loše predviđano. Za veći broj čvorova u skrivenom sloju model uspeva da se donekle približi ekstremima, dok se za manji broj to ne postiže. Za velike vrednosti α prebrzo ode u minimum i ne može da se povrati iz njega. U odnosu na test skup svi modeli su se podjenako loše pokazali. Na nepoznatim podacima nije efikasan, ali za uzastopne podatke se može reći da donekle uspeva da predvidi naredne vrednosti.

Model treniran ovakvim algoritmom koji uzima u obzir samo jedan skriveni sloj je efikasniji za jednostavnije probleme predviđanja, tako da bi naredni korak u poboljšanju algoritma bio dodavanje još jednog skrivenog sloja i više eksperimentisanja sa parametrima.

Literatura

[1] Model rekurentne neuronske mreže. on-line at: http://poincare.matf.bg.ac.rs/~stefan/ri/index.htm.

[2] Uvod u RNN. on-line at: https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/.

[3] Uvod u RNN i poređenje sa tradicionalnom NN. on-line at: https://towardsdatascience.com/recurrent-neural-networks-d4642c9bc7ce.

[4] Sara Gavran. Veštačke neuronske mreže u istraživanju podataka: pregled i primena. Master's thesis, Univerzitet u Beogradu, Matematički fakultet, 2016.