Predviđanje vremenskih prilika upotrebom neuronske mreže

Seminarski rad iz Računarske Inteligencije
Matematički fakultet,
Univerzitet u Beogradu

Matija Kovačević 270/2015
Nenad Milovanović 177/2015

3. februar 2019

Sadržaj

1 Uvod

Problem kojim smo se bavili u ovom radu je predviđanje vremenskih serija upotrebom neuronske mreže, konkretno vremenska prognoza. Pod vremenskom serijom se podrazumeva uređen niz opservacija, gde se uređenje vrši u odnosu na vreme, najčešće u jednakim vremenskim intervalima.
Koristili smo bazu podataka koja sadrzi podatke (temperatura, vlažnost vazduha, vazdušni pritisak itd.) za otprilike 6800 uzastopnih dana (https://www.kaggle.com/juliansimon/weather_madrid_lemd_1997_2015.csv).
Ideja je da se mreža istrenira tako da za zadatih 10 dana predvidi vremenske prilike za 11. dan.
U literaturi je navedeno nekoliko radova drugih istraživača koji su nam bili od pomoći pri rešavanju ovog problema.

2 Opis rešenja

Projekat je napisan na programskom jeziku Python 3 uz upotrebu biblioteka Numpy (za rad sa nizovima i matricama) i Matplotlib (za crtanje grafika). Koristili smo **Keras** – API za neuronske mreže koji radi na TensorFlow biblioteci.

Iz gore navedene baze podataka smo koristili 6 parametara:

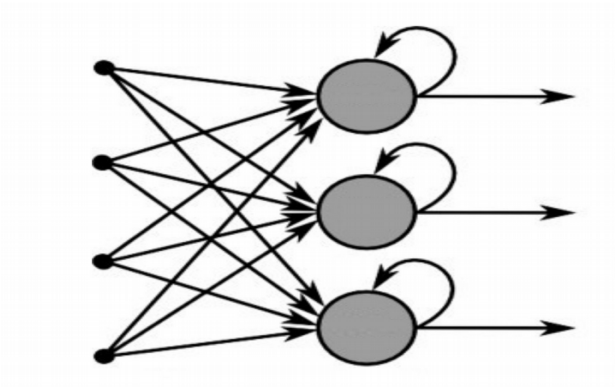
- broj meseca u godini
- maksimalnu temperaturu
- minimalnu temperaturu
- vlažnost vazduha
- vazdušni pritisak
- brzinu vetra

Za treniranje je korišćeno 5000 podataka, a za testiranje preostalih 1810.
Podatke smo delili tako da svakih 10 uzastopnih dana predstavlja ulaz, a prvi sledeći dan izlaz mreže.

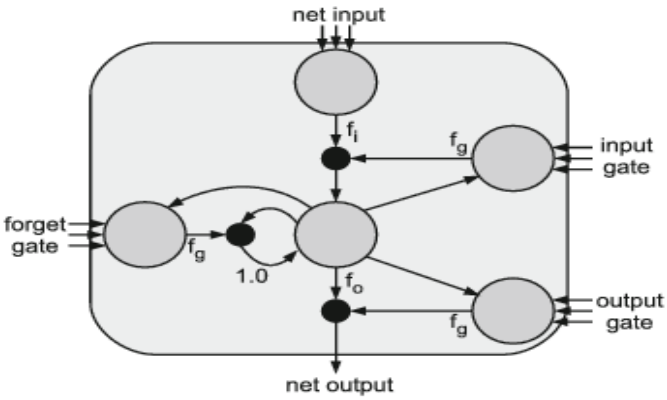
Ulaz Izlaz
1, 2, 3, 4, 5, 6, 7, 8, 9, 10 → 11
2, 3, 4, 5, 6, 7, 8, 9, 10, 11 → 12
…
6800, 6801, 6802, 6803, 6804, 6805, 6806, 6807, 6808, 6809 → 6810

Za rešavanje ovog problemo smo pravili rekurentnu neuronsku mrežu **LSTM** (*Long Short Term Memory*).
RNN pamte prethodna stanja i uzimaju ih u obzir za generisanje novih izlaza, što doprinosi povećanju preciznosti. Zbog ove osobine rekurentne mreže su pogodne sa obradu podataka kao što su vremenske serije.

Kod običnih neuronskih mreža može se javiti problem nestajućih gradijenata. LSTM otklanja ovaj problem i omogućuje duže “pamćenje”. LSTM jednica se sastoji od ćelije i tri vrste kapija: ulazne, izlaze i kapije za zaboravljanje. Ćelija je odgovorna za praćenje zavisnosti između elemenata ulazne sekvence. Ulazna kapija kontroliše stepen do kojeg nova vrednost ulazi u ćeliju, kapija za zaboravljanje kontroliše do koje mere vrednost ostaje u ćeliji i izlazna kapija kontroliše stepen do kojeg se vrednost u ćeliji koristi za izračunavanje izlaza.



Rekurentna neuronska mreža



LSTM

U prvom delu programa smo učitali podatke iz csv fajla, obradili ih i podelili na trening i test skup. Sve podatke smo skalirali na interval [0, 1]. Zatim smo napravili model mreže (LSTM mrežu sa ulaznim, jednim skrivenim i izlaznim slojem). Prilikom treniranjanja mreze 10% trening skupa je odvojeno kao skup za validaciju. U terminalu se prilikom treniranja ispisuje progres kao na sledećoj slici:

```
Epoch 14/20
4500/4500 [==============================] - 8s 2ms/step - loss: 0.0066 - acc: 0.7818 - val_loss: 0.0095 - val_acc: 0.8380
Epoch 15/20
4500/4500 [==============================] - 9s 2ms/step - loss: 0.0065 - acc: 0.7809 - val_loss: 0.0087 - val_acc: 0.8380
Epoch 16/20
4500/4500 [==============================] - 9s 2ms/step - loss: 0.0064 - acc: 0.7896 - val_loss: 0.0084 - val_acc: 0.8400
Epoch 17/20
4500/4500 [==============================] - 8s 2ms/step - loss: 0.0063 - acc: 0.7913 - val_loss: 0.0081 - val_acc: 0.8400
Epoch 18/20
4500/4500 [==============================] - 8s 2ms/step - loss: 0.0063 - acc: 0.7949 - val_loss: 0.0080 - val_acc: 0.8360
Epoch 19/20
4500/4500 [==============================] - 7s 2ms/step - loss: 0.0062 - acc: 0.7922 - val_loss: 0.0082 - val_acc: 0.8340
Epoch 20/20
4500/4500 [==============================] - 7s 2ms/step - loss: 0.0062 - acc: 0.7920 - val_loss: 0.0081 - val_acc: 0.8380
1800/1800 [==============================] - 2s 904us/step
Test score: [0.006380708428720633, 0.8011111111111111]
```

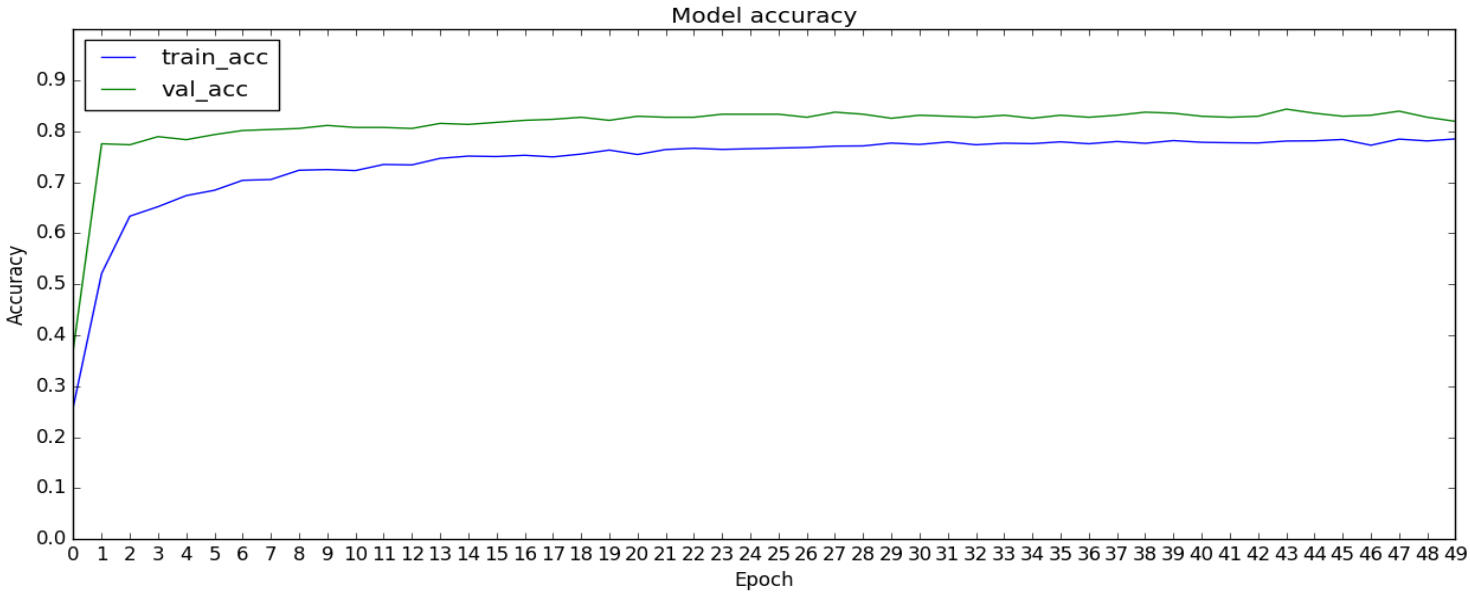
Model je zatim sačuvan u fajlu (ekstenzija .h5) kako ne bismo morali svaki put da ponavljamo treniranje mreže.
Na kraju se vrši evaluacija, tj. računanje funkcije gubitka i tačnosti za test podatke.

3 Rezultati

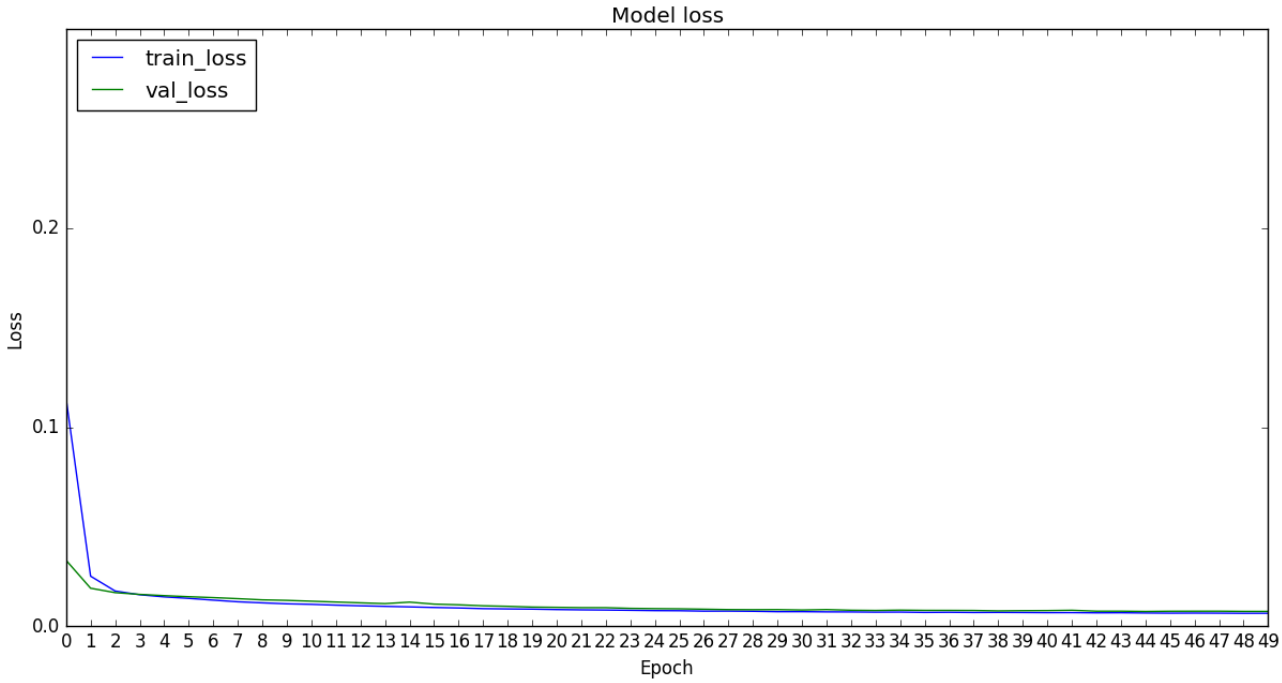
Prilikom testiranja programa menjali smo broj slojeva u mreži, broj neurona, funkcije aktivacije i pratili vrednosti funkcije gubitka i tačnosti na trening skupu, skupu za validaciju kao i na test skupu. Testirali smo mreže sa 2, 3 i 4 LSTM sloja pri tom menjajuci i brojeve neurona u slojevima, i uvideli da nema nikakve znacajne promene u rezultatima. Koristili smo dropout kako bi izbegli preprilagodjavanje. Za funkciju aktivacije je korišćena funkcija ReLU (Rectified Linear Unit). Korišćenjem drugih funkcija aktivacije (sigmoid, tanh, softmax itd.) dobijali smo znatno losije rezultate.

Primetili smo da se već nakon 10 do 15 epoha dostiže maksimalna moguća tačnost, a treniranjem u 50 epoha se tačnost popravlja za najviše 0.5% do 1%. Najbolji rezultat je dobijen korišćenjem LSTM mreže sa 3 sloja: ulazni (80 neurona), skriveni sloj (40 neurona) i izlazni sloj.

Maksimalna dostignuta tacnost na test skupu je ~80%. Na slikama ispod se nalaze grafikoni funkcije gubitka i tacnosti za trening skup i skup za validaciju dobijenih u 50 epoha.

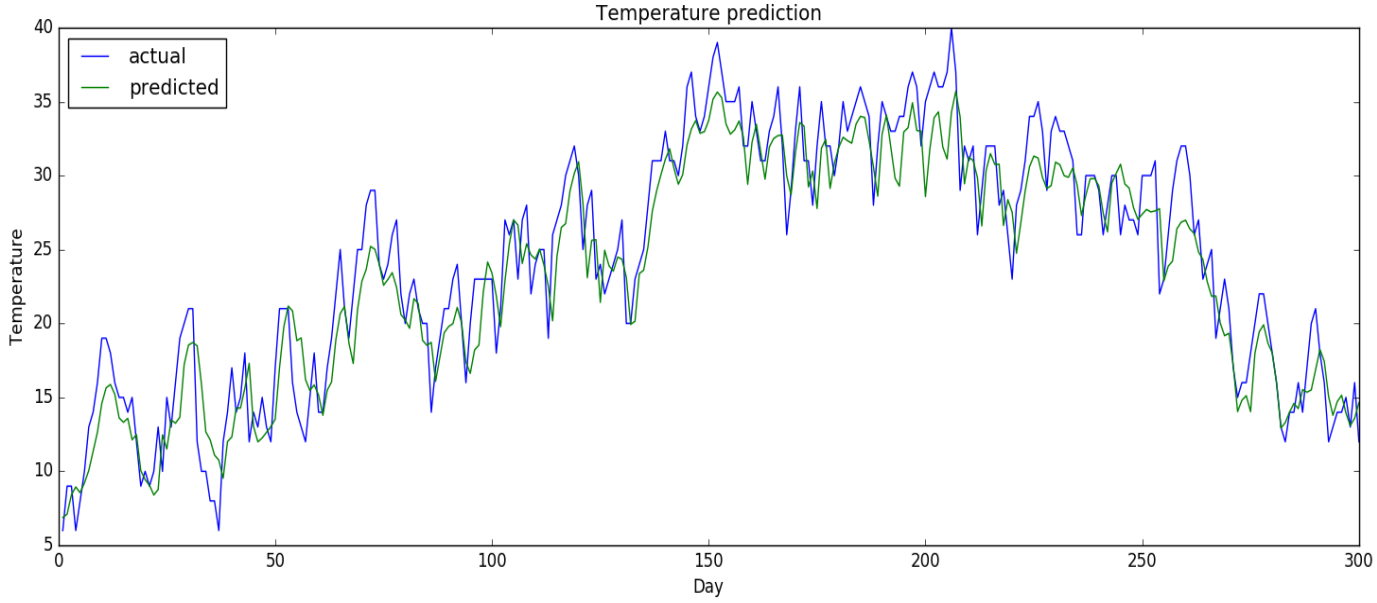


Tacnost nad trening skupom i skupom za validaciju



Funkcija gubitka na trening skupu i skupu za validaciju

Sledeća slika predstavlja odnos tačne i predviđene temperature za 300 dana.



4 Zaključak

- Tačnost predvidjanja od ~80% nije potpuno zadovoljavajuća, ali smatramo da je uzrok nepostizanja boljeg rezultata to što je skup podataka koji smo koristili sadržao dnevne vrednosti za svaki od parametara, koji se mogu neočekivano menjati iz dana u dan.
- Da bi se popravila tačnost potrebno je koristiti skup podataka sa što češćim merenjima i sa više parametara. Ali to implicira povećanje vremenske slozenosti programa.
- Upotreba rekurentnih neuronskih mreža daje najbolje rezultate za ovakav tip problema.

5 Literatura

- Chollet, F. (2017). Deep learning with python. Manning Publications Co..
- Hall, T., Brooks, H. E., & Doswell III, C. A. (1999). Precipitation forecasting using a neural network. Weather and forecasting, 14(3), 338-345.
- Giles, C. L., Lawrence, S., & Tsoi, A. C. (2001). Noisy time series prediction using recurrent neural networks and grammatical inference. Machine learning, 44(1-2), 161-183.
- Baboo, S. S., & Shereef, I. K. (2010). An efficient weather forecasting system using artificial neural network. International journal of environmental science and development, 1(4), 321.
- https://www.keras.io
- https://www.machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting