Razvoj LSTM neuronske mreže i primena nad problemom sekvencijalnog učenja

Seminarski rad u okviru kursa
Računarska inteligencija
Matematički fakultet

Nevena Soldat, Milena Kurtić
nevenasoldat@gmail.com, mimikurtic67@gmail.com

7. april 2020.

Sažetak

U okviru ovog rada predstavljen je primer implementacije LSTM neuronske mreže u programskom jeziku Python, uz korišćenje biblioteka za rad kao što su pandas, numpy, keras i druge. Problem koji rešavamo jeste predviđanje broja obolelih osoba, kao i broja smrtnih slučajeva od virusa koji je zahvatio čitav svet. Videćemo koje su prednosti navedene vrste neuronskih mreža, kao i koja su njena ograničenja. Takođe ćemo se poslužiti jednom od dobro poznatih statističkih metoda koje rade sa ovom vrstom podataka i uporedićemo dobijene rezultate oba modela.

Sadržaj

1 Uvod

LSTM (eng. Long Short-Term Memory) je podvrsta rekurentne neuronske mreže. Rekurente neuronske mreže (eng. Recurrent Neural Networks - RNN) su specijalan tip neuronskih mreža koje se koriste za sekvencijalne probleme učenja. Eksperimenti su pokazali da ih je veoma teško trenirati efikasno. Naime, prilikom ažuriranja težina, može doći do toga da njihova promena bude toliko mala da nema efekta (dovodi do problema koji se zove vanishing gradient), odnosno toliko velika da su promene prevelike (dovodi do problema koji je poznat kao exploding gradient). LSTM mreže prevazilaze probleme klasičnih rekurentnih mreža.

1.1 Primena na vremenske serije

Predviđanje vremenskih serija se može opisati kao proces koji izvlači korisne informacije iz vrednosti koje su se realizovale u nekom prethodnom trenutku, i na osnovu njih predviđa buduće vrednosti. Nailazimo na veliku primenu ove tehnike u oblastima poput vremenske prognoze, planiranja transporta, odnosno regulisanja saobraćaja. Metode predviđanja koje su bazirane na neuronskim mrežama stiču veliku popularnost jer je dokazano da mogu biti podjednako dobre kao klasične statističke metode [1]. Analiza vremenskih serija i predviđanje su predmet intenzivnog proučavanja poslednjih 40 godina. Statistički model ARIMA (eng. Autoregressive Integrated Moving Average) je zbog svoje uspešnosti često korišćen za analizu procesa koji su vremenski zavisni. Međutim, glavno ograničenje ovog modela je to što ima tendenciju da teži ka srednjim vrednostima.

2 LSTM neuronske mreže

3 Opis problema

Cilj ovog rada je demonstriranje upotrebe LSTM neuronske mreže na problem predviđanja vremenskih serija. Kako je predviđanje toka pandemije virusa u trenutku pisanja ovog rada jedna od najektuelnijih tema, podaci koje koristimo predstavljaju broj obolelih, kao i broj žrtava zaraze ovim virusom. Zadatak je predvideti kako će se ovi brojevi menjati kroz dane koji slede. Kako bismo pokazali uspešnost ove metode, uporedićemo je sa ranije pomenutom metodom ARIMA.

3.1 Opis baze

Za potrebe ovog projekta korišćena je baza podataka COVID19 Global Forecasting koja se u trenutku razvijanja ovog projekta svakodnevno ažurira (može se pronaći na sledećoj adresi: https://www.kaggle.com/c/covid19-global-forecasting-week-3). Baza sadrži podatke o broju osoba koje su potvrđeno zaražene virusom COVID-19, broj osoba koje su preminule, datume po danima počevši od 22.1.2020, kao i pokrajine i države na koje se date brojke odnose. Podaci su smešteni u fajl pod nazivom train.csv.

4 Rešenje problema

Problem predviđanja toka pandemije virusa rešavamo pomoću LSTM neuronske mreže. Program se sastoji iz sledećih delova:

- **Preprocesiranje**
- **Učenje modela nad trening podacima**
- **Testiranje modela**

4.1 Preprocesiranje

Kako bismo primenili LSTM neuronske mreže moramo da sredimo podatke koje imamo. To, u ovom slučaju, podrazumeva popunjavanje nedostajućih vrednost kao i normalizaciju podataka.

Nedostajuće vrednosti imamo u koloni Province_State. U svakom takvom slučaju u kom je vrednost iste kolone null, tu vrednost zamenjujemo sa vrednostima u koloni Country_Region. 1

```
# where Province_State is null we fill it with Country_Region
def fillState(state, country):
    if state == "empty":
        return country
    return state

dataframe['Province_State'].fillna("empty", inplace = True)
dataframe['Province_State'] = dataframe.loc[:,['Province_State', 'Country_Region']].apply(lambd
a x: fillState(x['Province_State'], x['Country_Region']), axis = 1)

test['Province_State'].fillna("empty", inplace = True)
test['Province_State'] = test.loc[:,['Province_State', 'Country_Region']].apply(lambda x: fillS
tate(x['Province_State'], x['Country_Region']), axis = 1)
```

Slika 1: Popunjavanje nedostajućih vrednosti

Kada mreža koristi podatke koji su u velikom rasponu vrednosti, za velike ulaze može doći do velikog usporavanja učenja kao i konvergencije, stoga je potrebno skalirati date podatke. Postoje dva nacina za skaliranje vrednosti:

- **Normalizacija**: Ponovno skaliranje vrednosti podataka tako da su svi u opsegu od 0 do 1.
- **Standardizacija**: Reskaliranje vrednosti podataka tako da je srednja vrednost 0 a standardna devijacija 1.

Za potrebe ovog projekta korisćena je normalizacija (Standardizaciju ima smisla koristiti kada bi imali Gausovu raspodelu). Normalizaciju datog skupa podataka vršimo pomoću biblioteke scikit-learn i objekta MinMaxScaler.2

```
scaler = MinMaxScaler(feature_range=(0,1))
dataset = scaler.fit_transform(dataset)
```

Slika 2: Normalizacija podataka

4.2 Učenje modela nad trening podacima

4.3 Testiranje modela

5 Zaključak

Literatura

[1] Sima Siami-Namini and Akbar Siami Namin. Forecasting economics and financial time series: Arima vs. lstm. *arXiv preprint arXiv:1803.06386*, 2018.

A Dodatak