Deep Learning Final Project Increment 2

The dataset consists of 5 attributes + 7 attributes added since last increment:

* Province\_State – Province/State within a country or region (some as null).
* Country\_Region – Country/Region.
* Date – Date in the format of yyyy-mm-dd.
* ConfirmedCases – Integer value of confirmed cases at the date in the location of that row.
* Fatalities – Integer value of fatalities at the data in the location of that row.
* HumanDevIndex - human development index (which is a way of showing how developed a country is)
* GovtType – government type
* HospitalBeds - number of hospital beds for 1000 people
* Continent
* PopDensity - population density (population/km)
* AvgMarchTemp - average temperature in March (Fahrenheit)
* LockdownDate - and the date that at least part of the country started a lockdown

We are using sequential and non-sequential data for 70 dates starting in March and in all different countries and provinces/states with them to predict future Covid-19 confirmed cases and fatalities throughout April.

One simple way to predict future disease spread is to simple use similar past data of other disease spread in the past. The problem is that Covid-19 is a new disease, one of much larger viral proportional than seen from past diseases and due to the uniqueness and traits of the disease a lot of factors about the areas that it is spreading affect how is spreads and affects a population. So with the use of a machine learning algorithm, it can take in account of patterns of other countries with similar traits. Comparisons and connections which could get complex to properly use for predictions.

As RNN’s go for predicting this problems sequences, it lacks as non-sequential data goes it to using it properly. So far as of this increment, the use of an LSTM model is being used. Non-sequential attribute data is not being implemented as of now. Future solutions could be using 2 models, one for sequential data (LSTM), and another for non-seqential, then merging the 2 to create a full model for predictions. Further research is needed.

For now the model is soley being trained on ConfirmedCases and Fatalities attributes, both sequential. An example of train all data on the model then predicting for Colorado, US.

EDA on train2.csv and test2.csv:

# Get numerical values for the dates  
all\_dates = df\_train.Date.append(df\_test.Date)  
all\_dates = all\_dates.astype('category')  
date\_dict\_num\_key = dict(enumerate(all\_dates.cat.categories))  
date\_dict = {y: x for x, y in date\_dict\_num\_key.items()}  
  
  
 # EDA on train data  
# Date attribute is changed to numerical value.  
num\_date\_col = []  
for date in df\_train["Date"]:  
 num\_date\_col.append(date\_dict[date])  
  
df\_train["Date"] = num\_date\_col  
  
# Province\_State attributes null values are changed to None  
df\_train = df\_train.fillna({"Province\_State": "None"})  
  
# Country\_Region and Province\_State attributes are changed to category object,  
# then are changed to numerical values. A dictionary is made to be able to  
# get the label for the numerical value in the future.  
# First for Country\_Region  
df\_train["Country\_Region"] = df\_train["Country\_Region"].astype('category')  
country\_region\_dict\_num\_key = dict(enumerate(df\_train["Country\_Region"].cat.categories))  
country\_region\_dict = {y: x for x, y in country\_region\_dict\_num\_key.items()}  
df\_train["Country\_Region"] = df\_train["Country\_Region"].cat.codes  
# Now for Province\_State  
df\_train["Province\_State"] = df\_train["Province\_State"].astype('category')  
province\_state\_dict\_num\_key = dict(enumerate(df\_train["Province\_State"].cat.categories))  
province\_state\_dict = {y: x for x, y in province\_state\_dict\_num\_key.items()}  
df\_train["Province\_State"] = df\_train["Province\_State"].cat.codes  
  
df\_train = df\_train.fillna({"HumanDevIndex": 7})  
df\_train = df\_train.fillna({"GovtType": "None"})  
df\_train["GovtType"] = df\_train["GovtType"].astype('category')  
df\_train = df\_train.fillna({"HospitalBeds": 2})  
df\_train = df\_train.fillna({"Continent": "None"})  
df\_train["Continent"] = df\_train["Continent"].astype('category')  
df\_train = df\_train.fillna({"PopDensity": 50})  
df\_train = df\_train.fillna({"AvgMarchTemp": 40})  
# df\_train["LockdownDate"] = pd.to\_datetime(df\_train.LockdownDate) # need to update this attribute's values

The model that we would use would be simple, an RNN. RNN as the choice because we are predicting future data based on patterns. The model as is would be extremely inaccurate because of the little data given in the initial dataset so data must be added. Data we could add to the training dataset include density of the area, 1st or 3rd world , when the country goes on lockdown, hospital beds per capita, and type of government, etc.

The current problem with the model as seen too low of values and a plateauing of the data when it should still be increasing. Possible solutions are stateful LSTM which will be played with in the future.

Current model:

varis = 2 # amount of output sequences  
  
  
def build\_model():  
 model = Sequential()  
 model.add(LSTM(units=4, return\_sequences=True, activation='tanh', input\_shape=(None, varis)))  
 #model.add(Dense(2))  
 model.add(LSTM(units=varis, return\_sequences=True))  
 model.compile(loss='mean\_squared\_error', optimizer='adam', metrics=['accuracy'])  
 return model

model.fit(trainx, trainy, epochs=20, batch\_size=10, shuffle=False)

Using low number of units for layers, tanh is the default of LSTM layers and most viable out of sigmoid, tanh, and relu. Using higher numbers of units for layers, batch sizes, or epochs saw no improvement and more overfitting.

This was simply run in Pycharm environment and TensorFlow as backend. Simply load the saved model .h5 file and main.py and can make predictions with method:

model\_predict(model, t\_len, sample\_country, sample\_province)