import numpy

import pandas

import tensorflow as tf

from pandas import DataFrame

from keras.models import Sequential

from keras.layers import Dense , Dropout, Activation

from keras.models import model\_from\_json

from keras.models import load\_model

**Numpy**: Numpy là một thư viện lõi phục vụ cho khoa học máy tính của Python, hỗ trợ cho việc tính toán các mảng nhiều chiều, có kích thước lớn với các hàm đã được tối ưu áp dụng lên các mảng nhiều chiều đó

**Pandas**: là một thư viện mã nguồn mở, hỗ trợ trong thao tác dữ liệu. Trong bài này, Pandas dùng để lấy dữ liệu từ file csv

**Tensorflow**:  là thư viện mã nguồn mở dùng cho tính toán số học sử dụng đồ thị luồng dữ liệu, có tích hợp sẵn rất nhiều các thư viện machine learning

**Keras** : là một open source cho neural network được viết bởi ngôn ngữ python. keras là một API bậc cao có thể sử dụng chung với các thư viện deep learning nổi tiếng như tensorflow(được phát triển bởi gg), CNTK(được phát triển bởi microsoft),theano(người phát triển chính Yoshua Bengio).

Tham khảo :

Sử dụng thư viện pandas để lấy dữ liệu từ file csv

path\_train = 'D:/data1/NN\_data/export\_data\_x\_start\_test\_40.csv'

path\_test  = 'D:/data1/NN\_data/export\_data\_x\_end\_test\_40.csv'

input\_train = pandas.read\_csv(path\_train)

output\_test = pandas.read\_csv(path\_test)

# split into input (X) and output (Y) variables

X\_train = input\_train.iloc[:,0:40]

Y\_train = input\_train.iloc[:,40]

X\_test  = output\_test.iloc[100:200,0:40]

Y\_test  = output\_test.iloc[100:200,40]

Link tập dữ liệu : <https://drive.google.com/drive/u/0/folders/13n9F2Gkv-_8ijylI-PpBDBXMFqg_WVa1?fbclid=IwAR3gwpiR7FCumpzndC2kCWdFhzniLTxw2fjz6qmRzo1mWh9Ux5W8twKgVrw>

path\_train , path\_test : địa chỉ đường dẫn đến file csv cần đọc

pandas.read\_csv : đọc file csv

.iloc[địa chỉ hàng , địa chỉ cột]: dùng để lấy dữ liệu.

‘:’ : lấy toàn bộ hàng/cột

‘100 : 200’: lấy từ hàng/cột 100 đến 200

Tham khảo : <https://github.com/nguyenvanhieuvn/nguyenvanhieu.vn/blob/master/python-tutorial/pandas_tutorial.ipynb>

# create modelz

model = Sequential()

model.add(Dense(50, input\_dim=40, activation='relu'))

model.add(Dense(20, activation='relu'))

model.add(Dense(1, activation='linear'))

Tạo model:

**Model = Sequential()** :Khởi tạo model

The Sequential model is a linear stack of layers.

**Model.add** : thêm layer vào model

**Dense** thể hiện một fully connected layer, tức toàn bộ các unit của layer trước đó được nối với toàn bộ các unit của layer hiện tại

Dense(40, input\_dim=40): ‘input\_dim = 40’ : có 40 input

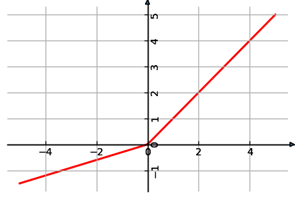
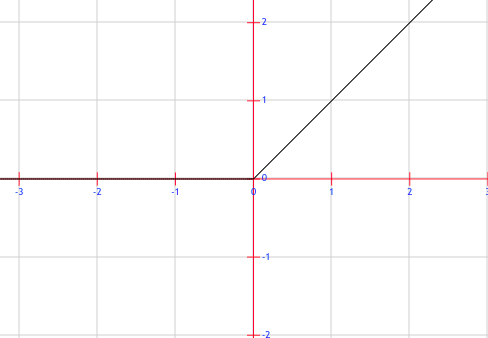
‘50’: có 50 noron trong layer

Dense(20,) : ’20’ : có 20 noron trong layer

**activation** : với một đầu vào hay tập đầu vào (input), activation sẽ cho ra đầu ra của một nút bất kỳ nào đó

VD: model.add(Dense(50, input\_dim=40, activation='relu')):

input 40 – output 50 .

(INPUT)->RELU-> (OUTPUT)

Relu : f(x)=max(0,x) có thể hiểu ReLU đơn giản lọc các giá trị < 0

\*) Các activation function cơ bản :

**Relu** : f(x)=max(0,x)

**Linear**: ‘a linear function is a function whose graph is a line in the place’ – là một đường thẳng dạng y=ax+b (đây là một dạng đơn giản nên hàm activation function thường phải là hàm non-linear)

Tham khảo : <https://blog.vietanhdev.com/posts/2019-09-23-cac-ham-kich-hoat-activation-function-trong-neural-networks/>

# Deploying model

model.compile(loss='mse', optimizer='adam', metrics=['accuracy'])

# Fit the model

model.fit(X\_train, Y\_train, epochs=6, batch\_size=2)

**model.compile** : ‘Before training a model, you need to configure the learning process, which is done via the compile method’

loss: một hàm để đánh giá độ tốt của model

optimizer:

metrics : For any classification problem you will want to set this to metrics=['accuracy']

**model.fit** : dùng để training model

một “batch” gồm nhiều mẫu để cập nhật trọng số của mô hình.

batch size thể hiện số lượng mẫu mà Mini-batch GD sử dụng cho một lần cập nhật trọng số

một epoch là một lần “duyệt” qua hết số lượng mẫu trong tập huấn luyện

iteration là số lượng batch mà thuật toán phải “duyệt” trong 1 epoch

VD: tập huấn luyện gồm 300000 dòng chọn batch\_size = 2 tức là mỗi lần cập nhật trọng số sử dụng 2 dòng . Khi đó, ta mất 300000/2=150000 iteration để duyệt qua tập huấn luyện (hoàn thành 1 epoch)

# serialize model to JSON

model\_json = model.to\_json()

with open("model09.json", "w") as json\_file:

    json\_file.write(model\_json)

# serialize weights to HDF5

model.save\_weights("model09.h5")

print("Saved model to disk")

Lưu model vào file JSON và lưu trọng số vào file .h5

\*) Chú ý : Trong trường hợp nào cũng có thể dùng

from keras.models import load\_model

model.save('my\_model.h5') # creates a HDF5 file 'my\_model.h5'

del model # deletes the existing model

# returns a compiled model

# identical to the previous one

model = load\_model('my\_model.h5')

# calculate predictions

predictions = model.predict(X\_test)

rounded = [round(x[0]) for x in predictions]

print(rounded)

for x in range(99):

    print(rounded[x],'====',Y\_test[x+100])

# print(Y\_test[1])

**model.predict**: Sử dụng model cho tập test

path\_train = 'D:/MITECH/2020/DATA/NN\_data2/train/export\_data\_x\_center\_train\_20.csv'

data\_train = read\_csv(path\_train)

data\_train = data\_train.to\_numpy()

shape\_0 = data\_train.shape[0]

shape\_1 = data\_train.shape[1]

# normalize the data\_train

data\_train = data\_train.reshape(shape\_0\*shape\_1, 1)

scaler\_train = MinMaxScaler(feature\_range=(0, 1))

data\_train = scaler\_train.fit\_transform(data\_train)

data\_train = data\_train.reshape(shape\_0,shape\_1)

X\_train = data\_train[:,0:shape\_1-1]

Y\_train = data\_train[:,(shape\_1-1)]

Một kiểu train model khác ở đây là chuyển dữ liệu về từ 0 đến 1 rồi train.

Chuyển dữ liệu kiểu này cho kết quả tốt hơn một chút so với không chuyển

Đánh giá : Model CNN dùng activation function là linear cũng khá là ổn. Và rõ ràng với bài toán regression thì nên sử dụng hàm linear.

Tuy nhiên so với LSTM03 thì vẫn còn kém một chút

Thời gian: với epochs = 10, batchsize = 1 thì khoảng 1.5 tiếng cho 1 lần train (bằng 1/5 so với LSTM03)

