Deep Learning Using TensorFlow

Dr. Ash Pahwa

Lesson 8:

Recurrent Neural Network + Reinforcement Learning

Lesson 8.2: Implementing RNN in TensorFlow

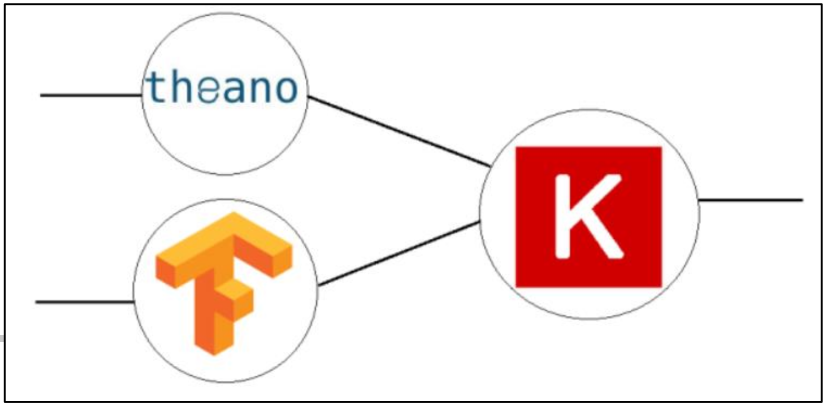
Outline

- RNN (LSTM) Implementation using Keras with TensorFlow background
  - With and without scaling

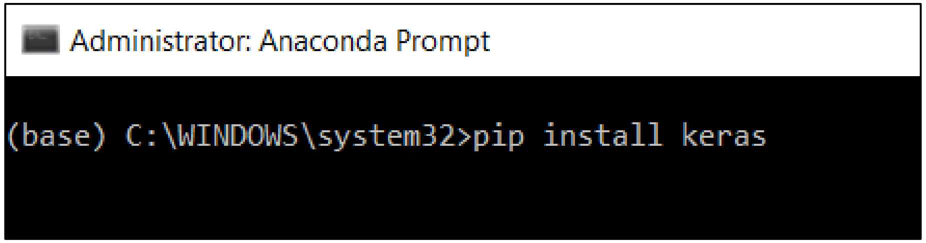
1. RNN (LSTM) Implementation

Using Keras with TensorFlow Backend

Installing Keras



- Start “Anaconda Prompt”
- (Right Click)
  - “More”
    - “Run as administrator”
- pip install keras



```
(base) C:\WINDOWS\system32>pip install keras
```

```
C:\Users\ash\.keras
File name ‘keras.json’
----------------------------------------------
{
    "floatx": "float32",
    "epsilon": 1e-07,
    "backend": "tensorflow",
    "image_data_format": "channels_last"
}
```

Goal:
Build an RNN + LSTM

- Input-1
  - 3,4,5,6,7
- Predicted Output-1
  - 8
- --
- Input-2
  - 25,26,27,28,29
- Predicted Output-2
  - 30

Load the Libraries

```
#################################################
# RNN Using Python and Keras
#
###############################################
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split

from keras.models import Sequential
#Using TensorFlow backend.
from keras.layers import LSTM
```

Dataset

- Input Data
  - 0,1,2,3,4
  - 1,2,3,4,5
  - 2,3,4,5,6
  - 3,4,5,6,7
  - …
- Output Data
  - 5
  - 6
  - 7
  - 8
  - …

```
###########################################
# Create dataset for RNN
#
DataList = [[ [i+j] for i in range(5)] for j in range (100)]
type(DataList)
Out[6]: list

DataList[0:5]
Out[7]:
[[[0], [1], [2], [3], [4]],
 [[1], [2], [3], [4], [5]],
 [[2], [3], [4], [5], [6]],
 [[3], [4], [5], [6], [7]],
 [[4], [5], [6], [7], [8]]]

DataList[95:100]
Out[8]:
[[[95], [96], [97], [98], [99]],
 [[96], [97], [98], [99], [100]],
 [[97], [98], [99], [100], [101]],
 [[98], [99], [100], [101], [102]],
 [[99], [100], [101], [102], [103]]]

TargetList = [(i+5) for i in range(100)]
type(TargetList)
Out[10]: list

TargetList[0:5]
Out[11]: [5, 6, 7, 8, 9]
TargetList[95:100]
Out[12]: [100, 101, 102, 103, 104]
```

Dataset

```
data = np.array(DataList, dtype = float)
data[0:5]
Out[14]:
array([[[ 0.],
        [ 1.],
        [ 2.],
        [ 3.],
        [ 4.]],

       [[ 1.],
        [ 2.],
        [ 3.],
        [ 4.],
        [ 5.]],

       [[ 2.],
        [ 3.],
        [ 4.],
        [ 5.],
        [ 6.]],

       [[ 3.],
        [ 4.],
        [ 5.],
        [ 6.],
        [ 7.]],

       [[ 4.],
        [ 5.],
        [ 6.],
        [ 7.],
        [ 8.]]])
```

```
data.shape
Out[15]: (100, 5, 1)

target = np.array(TargetList, dtype = float)

target[0:5]
Out[17]: array([ 5.,  6.,  7.,  8.,  9.])

target.shape
Out[18]: (100,)
```

Split Data into Train + Test

```
# Split data set
#
x_train, x_test, y_train, y_test =
train_test_split(data, target, test_size=0.2,
random_state=4)

x_train[0:2]
Out[28]:
array([[[ 80.],
        [ 81.],
        [ 82.],
        [ 83.],
        [ 84.]],

       [[  4.],
        [  5.],
        [  6.],
        [  7.],
        [  8.]]])

y_train[0:2]
Out[29]: array([ 85.,   9.])
```

```
x_test[0:2]
Out[30]:
array([[[ 20.],
        [ 21.],
        [ 22.],
        [ 23.],
        [ 24.]],

       [[ 10.],
        [ 11.],
        [ 12.],
        [ 13.],
        [ 14.]]])

y_test[0:2]
Out[31]: array([ 25.,  15.])
```

Specify the RNN Model and add LSTM Layer

```
##################################################
# RNN Model
#
model = Sequential()

# Add the LSTM
model.add(LSTM((1), batch_input_shape=(None,5,1), return_sequences=False))

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

model.summary()

_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_1 (LSTM)                (None, 1)                 12        
=================================================================
Total params: 12
Trainable params: 12
Non-trainable params: 0

_________________________________________________________________
```

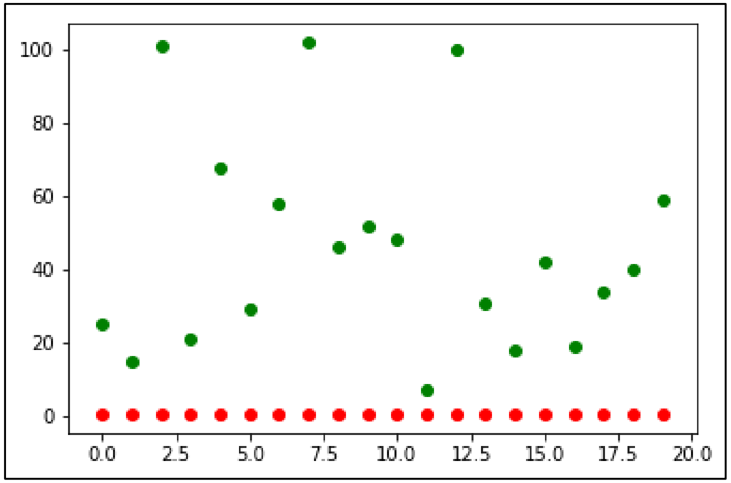
Fit the Train data into the LSTM Model

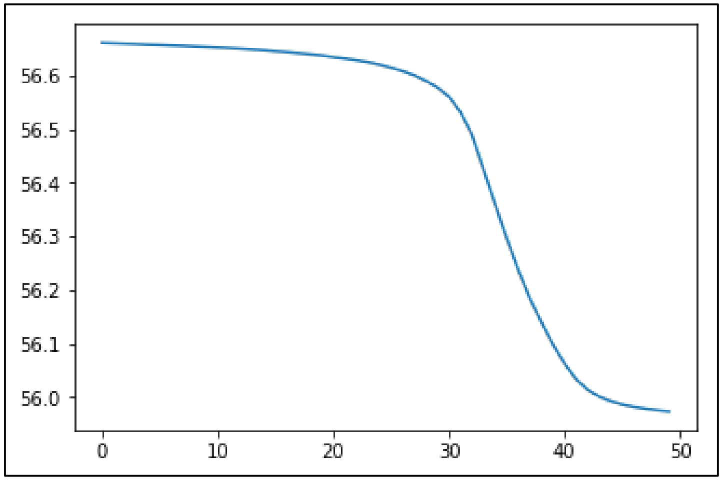
```
#################################################
# Fit the training data to the model
# Measure the accuracy with test data
#

history = model.fit(x_train, y_train, epochs=50, validation_data=(x_test,y_test))

Train on 80 samples, validate on 20 samples
Epoch 1/50
80/80 [==============================] - 0s 6ms/step - loss: 56.6611 - acc: 0.0000e+00 -
val_loss: 45.7316 - val_acc: 0.0000e+00
Epoch 2/50
80/80 [==============================] - 0s 112us/step - loss: 56.6604 - acc: 0.0000e+00 -
val_loss: 45.7305 - val_acc: 0.0000e+00
Epoch 3/50
80/80 [==============================] - 0s 112us/step - loss: 56.6596 - acc: 0.0000e+00 -
val_loss: 45.7294 - val_acc: 0.0000e+00
Epoch 4/50
80/80 [==============================] - 0s 112us/step - loss: 56.6588 - acc: 0.0000e+00 -
val_loss: 45.7281 - val_acc: 0.0000e+00
Epoch 5/50
80/80 [==============================] - 0s 137us/step - loss: 56.6578 - acc: 0.0000e+00 -
val_loss: 45.7266 - val_acc: 0.0000
```

Plot the Results

Very Poor: All Predictions are Zeros

```
####################################################
# Predict using Testing data
#
results = model.predict(x_test)

plt.scatter(range(20), results,c='r')
plt.scatter(range(20), y_test, c='g')
Out[52]: <matplotlib.collections.PathCollection at 0x2258e3a2550>
####################################################
# Plot the loss Function
#
plt.plot(history.history['loss'])
Out[56]: [<matplotlib.lines.Line2D at 0x2258e392978>]
```





Make 2 changes to the Model

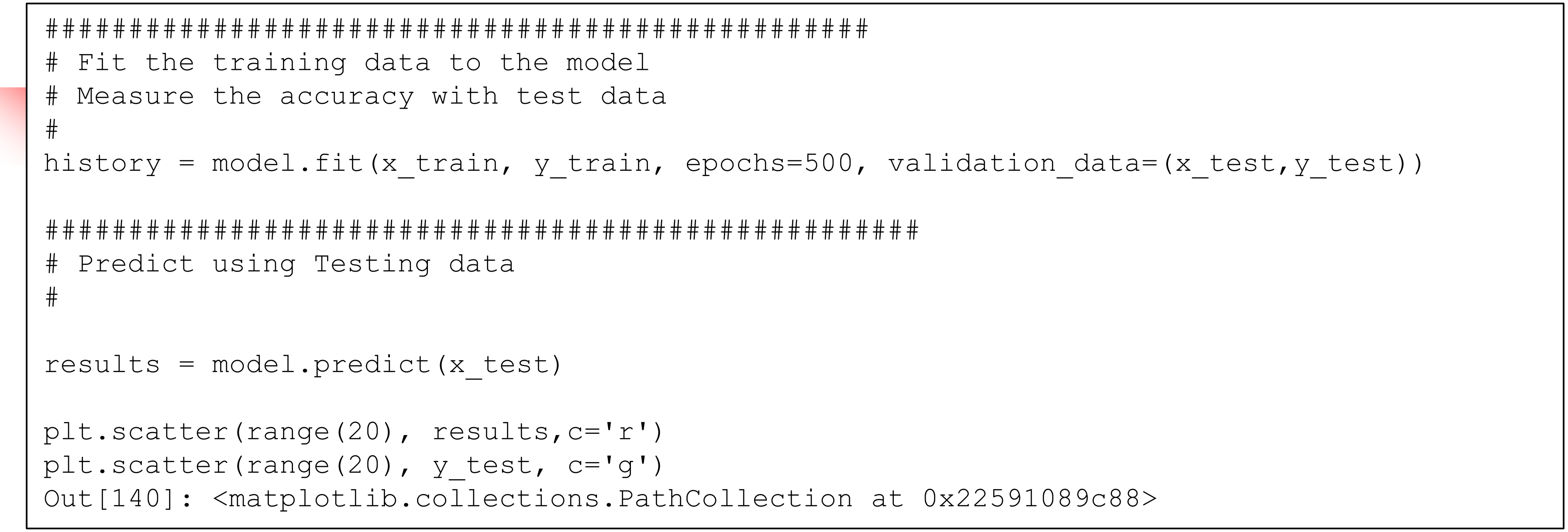
- Change
  - Scale the data between 0 and 1
  - Increase the epochs from 50 to 500

Scale the data between 0 and 1

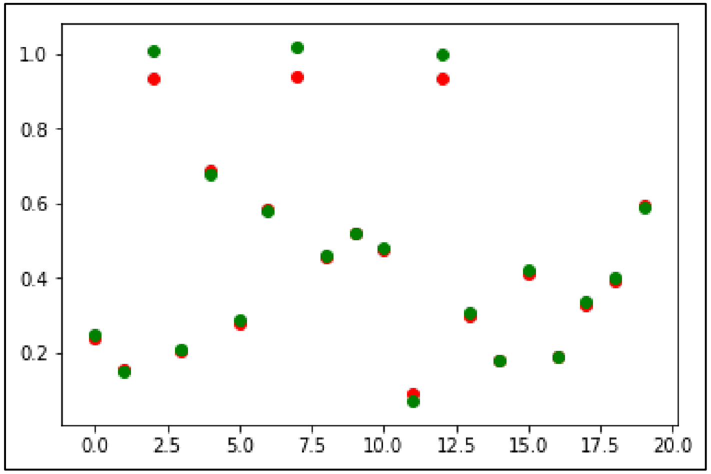
```
####################################################
# Scale the data between 0 and 1
dataScale = data/100
dataScale[0:2]
Out[112]:
array([[[ 0.  ],
        [ 0.01],
        [ 0.02],
        [ 0.03],
        [ 0.04]],

       [[ 0.01],
        [ 0.02],
        [ 0.03],
        [ 0.04],
        [ 0.05]]])

targetScale = target/100
targetScale[0:2]
Out[114]: array([ 0.05,  0.06])
###################################################
# Split data set
#x_train, x_test, y_train, y_test = train_test_split(data, target, test_size=0.2, random_state=4)

x_train, x_test, y_train, y_test = train_test_split(dataScale, targetScale, test_size=0.2, random_state=4)
```

$$y_i^j = \frac{x_i^j - min_j}{max_j - min_j}$$

Increase Epochs from 50 to 500



```
#################################################
# Fit the training data to the model
# Measure the accuracy with test data
#
history = model.fit(x_train, y_train, epochs=500, validation_data=(x_test,y_test))

####################################################
# Predict using Testing data
#

results = model.predict(x_test)

plt.scatter(range(20), results,c='r')
plt.scatter(range(20), y_test, c='g')
Out[140]: <matplotlib.collections.PathCollection at 0x22591089c88>
```

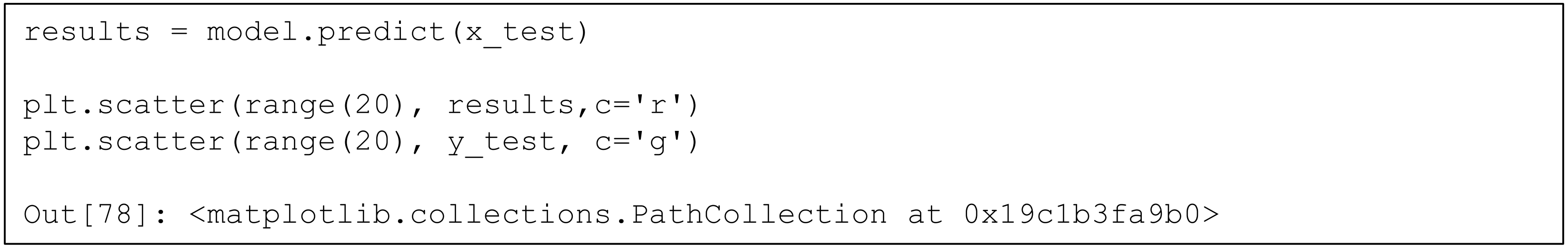


Copyright 2019 - Dr. Ash Pahwa

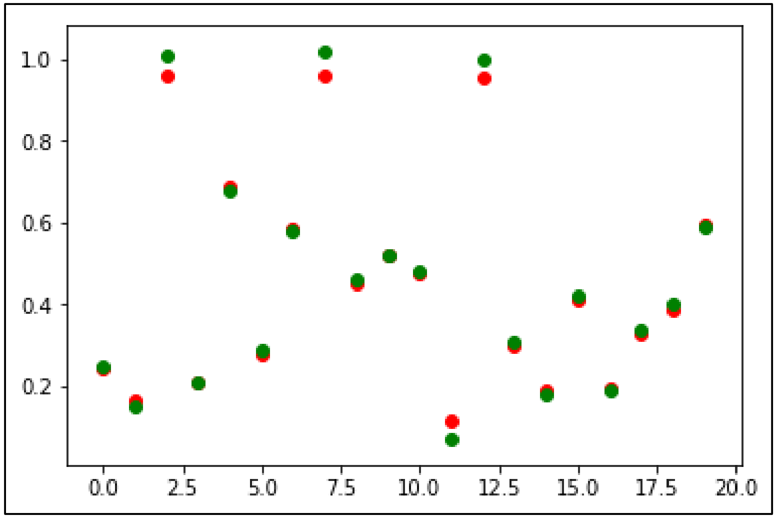
Add Another Layer

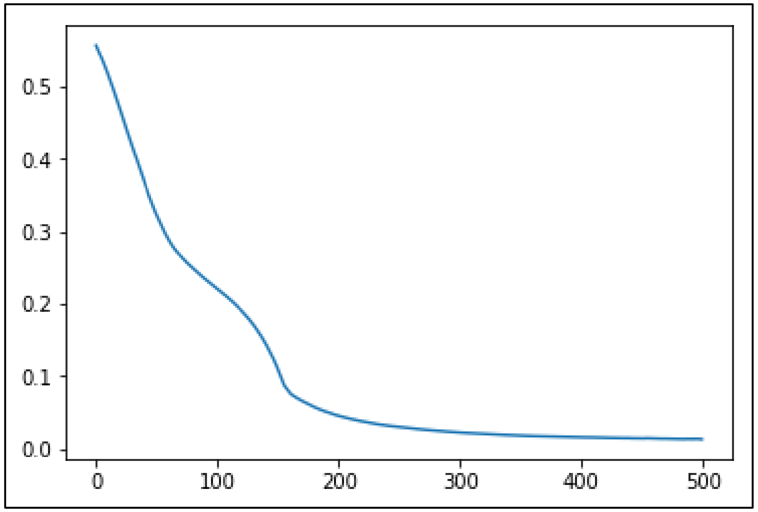
```
##################################################
# RNN Model
#
model = Sequential()

# Add the LSTM
model.add(LSTM((1), batch_input_shape=(None,5,1), return_sequences=True))
model.add(LSTM((1), return_sequences=False))

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

model.summary()
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_3 (LSTM)                (None, 5, 1)              12        
_________________________________________________________________
lstm_4 (LSTM)                (None, 1)                 12        
=================================================================
Total params: 24
Trainable params: 24
Non-trainable params: 0
_________________________________________________________________
```

Results



```
results = model.predict(x_test)

plt.scatter(range(20), results,c='r')
plt.scatter(range(20), y_test, c='g')

Out[78]: <matplotlib.collections.PathCollection at 0x19c1b3fa9b0>
```





Summary

- RNN (LSTM) Implementation using Keras with TensorFlow background
  - With and without scaling