CS5590/490 - Python and Deep Learning Programming
In Class Programming Report - 10
Class ID 24 - Anurag Thantharate
Date Submitted: 04/12/2020

---

1. In the code provided there are three mistake which stops the code from running successfully; find those mistakes and explain why they need to be corrected to be able to get the code run.

   I found below errors while running the provided code:
   1) Input dimension is not defined
   2) Number of words was hard coded, made it dynamic with max length review
   3) SoftMax gives better accuracy, so I changed the activation function in code.
   4) Number of neurons is also correct = 3, pos, neg and unsup

```
C:\ProgramData\Anaconda3\python.exe "C:/Users/ld630534/Documents/Anurag Projects/Spring 2020/Python with DL/DLICP3/sentiment_analysis.py"
Using TensorFlow backend.
   Unnamed: 0  type  ... label        file
0           0  test  ...   neg     0_2.txt
1           1  test  ...   neg  10000_4.txt
2           2  test  ...   neg  10001_1.txt
3           3  test  ...   neg  10002_3.txt
4           4  test  ...   neg  10003_3.txt

[5 rows x 5 columns]
Traceback (most recent call last):
  File "C:/Users/ld630534/Documents/Anurag Projects/Spring 2020/Python with DL/DLICP3/sentiment_analysis.py", line 54, in <module>
    model.add(layers.Dense(300,input_dim=input_dim, activation='relu'))
NameError: name 'input_dim' is not defined

Process finished with exit code 1
```

```python
#Import libraries
from keras.models import Sequential
from keras import layers
from keras.preprocessing.text import Tokenizer
import pandas as pd
from sklearn import preprocessing
from sklearn.model_selection import train_test_split

#Read the data
df = pd.read_csv('imdb_master.csv',encoding='latin-1')
print(df.head())
sentences = df['review'].values
y = df['label'].values
define input_dim
# tokenizing data
tokenizer = Tokenizer(num_words=2000)
tokenizer.fit_on_texts(sentences)

# getting the vocabulary of data
sentences = tokenizer.texts_to_matrix(sentences)

le = preprocessing.LabelEncoder()
y = le.fit_transform(y)
X_train, X_test, y_train, y_test = train_test_split(sentences, y, test_size=0.25, random_state=1000)

model = Sequential()
model.add(layers.Dense(300, input_dim=input_dim, activation='relu'))
model.add(layers.Dense(3, activation='sigmoid'))
model.compile(loss='sparse_categorical_crossentropy',optimizer='adam',metrics=['acc'])

history=model.fit(X_train,y_train, epochs=5, verbose=True, validation_data=(X_test,y_test), batch_size=256)
```

```python
#Import libraries
from keras.models import Sequential
from keras import layers
from keras.preprocessing.text import Tokenizer
import pandas as pd
from sklearn import preprocessing
from sklearn.model_selection import train_test_split

#Read the data
df = pd.read_csv('imdb_master.csv',encoding='latin-1')
print(df.head())
sentences = df['review'].values
y = df['label'].values

# Identifying the number of input dimensions
max_len_review = max([len(s.split()) for s in sentences])
print('Maximum length of review is', max_len_review)

# tokenizing data
tokenizer = Tokenizer(num_words=max_len_review)
tokenizer.fit_on_texts(sentences)

# getting the vocabulary of data
sentences = tokenizer.texts_to_matrix(sentences)

le = preprocessing.LabelEncoder()
y = le.fit_transform(y)
X_train, X_test, y_train, y_test = train_test_split(sentences, y, test_size=0.25, random_state=1000)

model = Sequential()
model.add(layers.Dense(300, input_dim=max_len_review, activation='relu'))
model.add(layers.Dense(3, activation='sigmoid'))
model.compile(loss='sparse_categorical_crossentropy',optimizer='adam',metrics=['acc'])

history=model.fit(X_train,y_train, epochs=5, verbose=True, validation_data=(X_test,y_test), batch_size=256)
```

```
72192/75000 [===========================>..] - ETA: 0s - loss: 0.4766 - acc: 0.7852
72704/75000 [===========================>.] - ETA: 0s - loss: 0.4764 - acc: 0.7854
73216/75000 [===========================>.] - ETA: 0s - loss: 0.4763 - acc: 0.7855
73728/75000 [===========================>.] - ETA: 0s - loss: 0.4766 - acc: 0.7854
74240/75000 [===========================>.] - ETA: 0s - loss: 0.4768 - acc: 0.7852
74752/75000 [===========================>.] - ETA: 0s - loss: 0.4772 - acc: 0.7849
75000/75000 [==============================] - 13s 176us/step - loss: 0.4773 - acc: 0.7848 - val_loss: 0.9917 - val_acc: 0.5110
```

with Sigmoid

```
71936/75000 [===========================>..] - ETA: 0s - loss: 0.2243 - acc: 0.9522
72448/75000 [===========================>..] - ETA: 0s - loss: 0.2243 - acc: 0.9521
72960/75000 [===========================>.] - ETA: 0s - loss: 0.2244 - acc: 0.9520
73472/75000 [===========================>.] - ETA: 0s - loss: 0.2246 - acc: 0.9519
73984/75000 [===========================>.] - ETA: 0s - loss: 0.2248 - acc: 0.9517
74496/75000 [===========================>.] - ETA: 0s - loss: 0.2250 - acc: 0.9517
75000/75000 [==============================] - 13s 167us/step - loss: 0.2250 - acc: 0.9516 - val_loss: 1.1315 - val_acc: 0.5150

Process finished with exit code 0
```

With Softmax

2. Add embedding layer to the model, did you experience any improvement?

```
# Importing libraries
from keras.layers import Embedding
from keras.layers import Flatten
from keras.models import Sequential
from keras import layers
from keras.preprocessing.text import Tokenizer
import pandas as pd
from keras_preprocessing.sequence import pad_sequences
from sklearn import preprocessing
from sklearn.model_selection import train_test_split

# Reading input data of IMBD reviews
df = pd.read_csv('imdb_master.csv', encoding='latin-1')
print(df.head())
sentences = df['review'].values
y = df['label'].values

# Identifying the number of input dimensions present within sentences to build the network model
max_review_len = max([len(s.split()) for s in sentences])
print('Maximum length of review is', max_review_len)

# Tokenizing data
tokenizer = Tokenizer(num_words=max_review_len)
tokenizer.fit_on_texts(sentences)

# Preparation of data for embedding layer
vocab_size = len(tokenizer.word_index)+1
sentences = tokenizer.texts_to_matrix(sentences)
padded_docs = pad_sequences(sentences, maxlen=max_review_len)
```

```
# Encoding target associated with IMDB data of reviews
le = preprocessing.LabelEncoder()
y = le.fit_transform(y)
X_train, X_test, y_train, y_test = train_test_split(padded_docs, y, test_size=0.25, random_state=1000)

# Model defined and fit
model = Sequential()

# Adding the embedding layer to the model defined
model.add(Embedding(vocab_size, 50, input_length=max_review_len))
model.add(Flatten())
# Input layer of model defined
model.add(layers.Dense(300, input_dim=max_review_len, activation='relu'))
model.add(layers.Dense(3, activation='softmax'))
# Output layer of model defined
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['acc'])

# Model fit using training data set
history = model.fit(X_train, y_train, epochs=3, verbose=True, validation_data=(X_test, y_test), batch_size=256)

# Evaluation of the performance of the model fit
[test_loss, test_acc] = model.evaluate(X_test, y_test)
print("Evaluation result on Test Data : Loss = {}, accuracy = {}".format(test_loss, test_acc))
```

```
73984/75000 [============================>.] - ETA: 7s - loss: 0.8073 - acc: 0.5504
74240/75000 [============================>.] - ETA: 5s - loss: 0.8073 - acc: 0.5504
74496/75000 [============================>.] - ETA: 3s - loss: 0.8073 - acc: 0.5503
74752/75000 [============================>.] - ETA: 1s - loss: 0.8073 - acc: 0.5502
75000/75000 [==============================] - 596s 8ms/step - loss: 0.8073 - acc: 0.5502 - val_loss: 0.8315 - val_acc: 0.5171
```

Train

```
24832/25000 [============================>.] - ETA: 0s
24864/25000 [============================>.] - ETA: 0s
24896/25000 [============================>.] - ETA: 0s
24928/25000 [============================>.] - ETA: 0s
24960/25000 [============================>.] - ETA: 0s
24992/25000 [============================>.] - ETA: 0s
25000/25000 [==============================] - 89s 4ms/step
Evaluation result on Test Data : Loss = 0.8314772221183777, accuracy = 0.5171200037002563
```

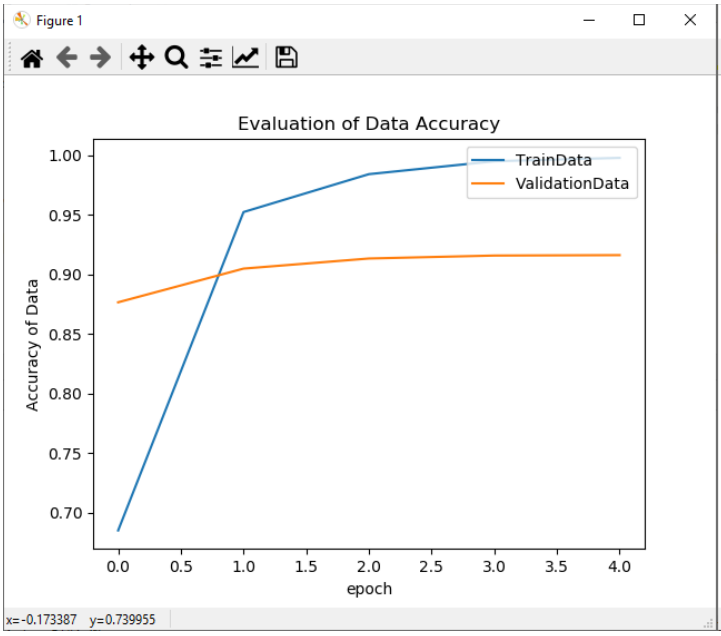
Test

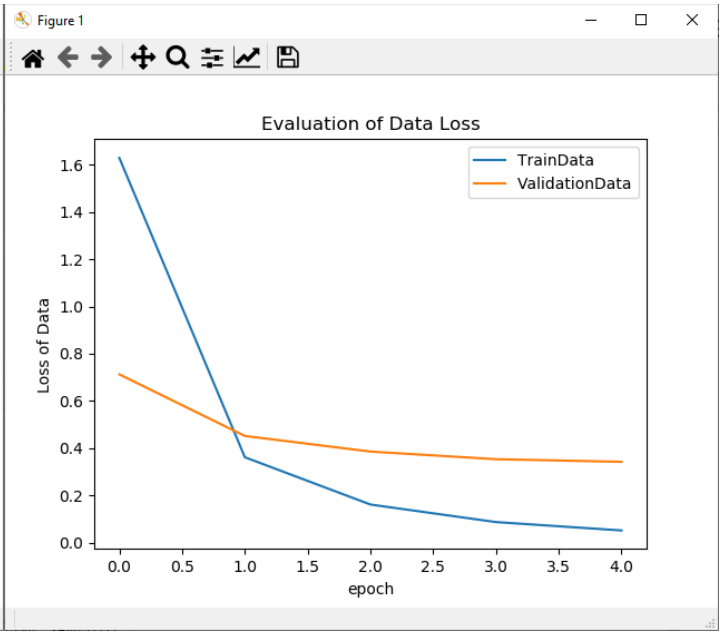
3. Apply the code on 20_newsgroup data set we worked in the previous classes
   from sklearn.datasets import fetch_20newsgroups
   newsgroups_train = fetch_20newsgroups(subset='train', shuffle=True, categories=categories,)

```python
# Importing libraries
from sklearn.datasets import fetch_20newsgroups
from keras.models import Sequential
from keras import layers
from keras.preprocessing.text import Tokenizer
import pandas as pd
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from keras.callbacks import TensorBoard

# Reading input data of IMBD reviews
newsgroups_train = fetch_20newsgroups(subset='train', shuffle=True)
sentences = newsgroups_train.data
y = newsgroups_train.target

# Identifying the number of input dimensions present within sentences to build the network model
max_review_len = max([len(s.split()) for s in sentences])
print('Maximum length of review is', max_review_len)

# Tokenizing data
tokenizer = Tokenizer(num_words=max_review_len)
tokenizer.fit_on_texts(sentences)
```

```
le = preprocessing.LabelEncoder()
y = le.fit_transform(y)
X_train, X_test, y_train, y_test = train_test_split(sentences, y, test_size=0.25, random_state=1000)

# Model defined and fit
model = Sequential()

# Input layer of model defined
model.add(layers.Dense(300, input_dim=max_review_len, activation='relu'))
# Hidden layer of model defined
model.add(layers.Dense(20, activation='softmax'))
# Output layer of model defined
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['acc'])

# Model fit using training data set
history = model.fit(X_train, y_train, epochs=5, verbose=True, validation_data=(X_test, y_test), batch_size=256)

# Evaluation of the performance of the model fit
[test_loss, test_acc] = model.evaluate(X_test, y_test)
print("Evaluation result on Test Data : Loss = {}, accuracy = {}".format(test_loss, test_acc))

1984/2829 [====================>.........] - ETA: 0s
2144/2829 [=====================>........] - ETA: 0s
2272/2829 [======================>......] - ETA: 0s
2432/2829 [=======================>.....] - ETA: 0s
2592/2829 [========================>...] - ETA: 0s
2752/2829 [=========================>.] - ETA: 0s
2829/2829 [==============================] - 1s 350us/step
Evaluation result on Test Data : Loss = 0.34434431613271166, accuracy = 0.9126899838447571
```

4. Plot the loss and accuracy using history object





*Bonus question

1. Predict over one sample of data and check what will be the prediction for that.

```
predictedmodel = model.predict(X_test[10].reshape(1, 885))
print("Actual value=" + str(y_test[10]) + "    Predicted value=" + str(predictedmodel.argmax()))
```

2. Plot loss and accuracy in Tensorboard.

```
#Import libraries
from keras.models import Sequential
from keras import layers
from keras.preprocessing.text import Tokenizer
import pandas as pd
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from keras.callbacks import TensorBoard

#Read the data
df = pd.read_csv('imdb_master.csv',encoding='latin-1')
print(df.head())
sentences = df['review'].values
y = df['label'].values

# Identifying the number of input dimensions
```

```
# Tensor board analysis
tensorAnalysis = TensorBoard(log_dir="logs", histogram_freq=1, write_graph=True, write_images=False)
history = model.fit(X_train, y_train, verbose=1, validation_data=(X_test, y_test), callbacks=[tensorAnalysis])

# Evaluation of the performance of the model fit using Tensorflow
[test_loss, test_acc] = model.evaluate(X_test, y_test)
print("Evaluation result using Tensorflow on Test Data : Loss = {}, accuracy = {}".format(test_loss, test_acc))

# Graphical evaluation of accuracy associated with training and validation data
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('Evaluation of Data Accuracy using Tensorflow')
plt.xlabel('epoch')
plt.ylabel('Accuracy of Data')
plt.legend(['TrainData', 'ValidationData'], loc='upper right')
plt.show()

# Graphical evaluation of loss associated with training and validation data
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.xlabel('epoch')
plt.ylabel('Loss of Data')
plt.title('Evaluation of Data Loss using Tensorflow')
plt.legend(['TrainData', 'ValidationData'], loc='upper right')
plt.show()
```

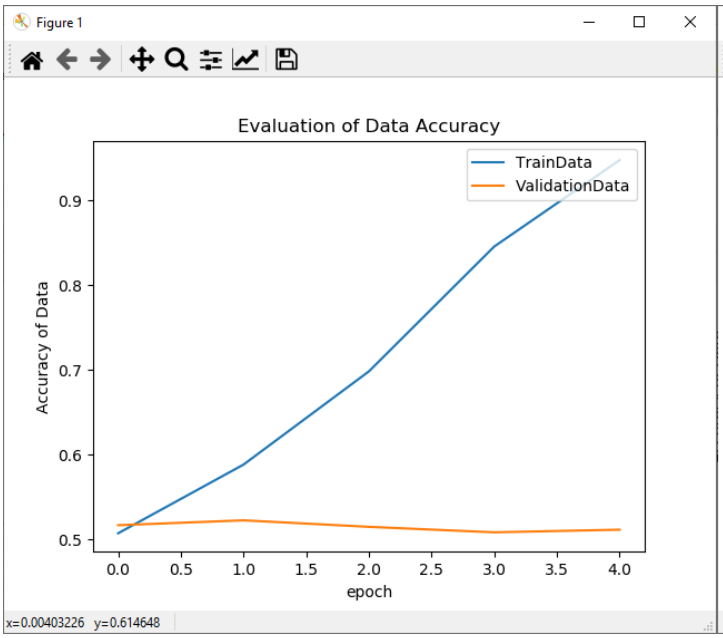


Figure 1
Evaluation of Data Accuracy
TrainData
ValidationData
Accuracy of Data
epoch
0.5
0.6
0.7
0.8
0.9
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
x=0.00403226 y=0.614648

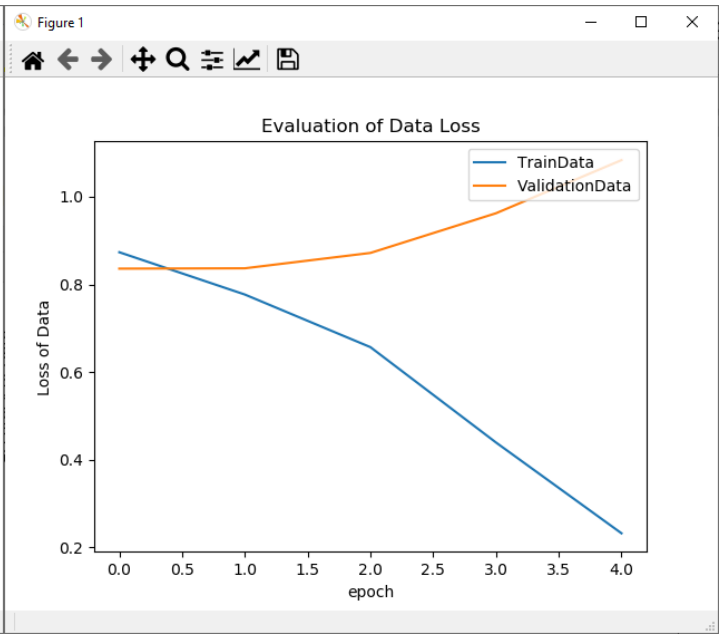


Figure 1
Evaluation of Data Loss
TrainData
ValidationData
Loss of Data
epoch
0.2
0.4
0.6
0.8
1.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
