Assignment 5.1

January 6, 2021

0.1 File information

File: Assignment_5.1.ipynb

Name: Amie Davis

Date: 1/5/2021

Course: DSC650 - Big Data

Assignment Number: 5.1

Purpose: Implement the movie review classifier found in section 3.4 of Deep Learning with Python.

1 Classifying movie reviews: a binary classification example

1.1 This file contains code from Deep Learning with Python

www.manning.com/books/deep-learning-with-python

1.2 Data Source: The IMDB dataset - comes packaged with Keras.

[2]:
```
import keras
keras.__version__
```

[2]:
```
'2.3.1'
```

1.3 Load the data

[3]:
```
# Use Keras imdb dataset for movie reviews
# Data is labeled 0 for negative reviews and 1 for positive reviews
# Split data into training & test datasets
# Keep the top 10,000 most frequently occurring words
from keras.datasets import imdb

(train_data, train_labels), (test_data, test_labels) = imdb.
 ↪load_data(num_words=10000)
```

[4]:
```
# Review Data
train_data[0]
```

[4]:
```
[1,
 14,
 22,
 16,
 43,
 530,
 973,
 1622,
 1385,
 65,
 458,
 4468,
 66,
 3941,
 4,
 173,
 36,
 256,
 5,
 25,
 100,
 43,
 838,
 112,
 50,
 670,
 2,
 9,
 35,
 480,
 284,
 5,
 150,
 4,
 172,
 112,
 167,
 2,
 336,
 385,
 39,
 4,
 172,
 4536,
```

```
1111,
17,
546,
38,
13,
447,
4,
192,
50,
16,
6,
147,
2025,
19,
14,
22,
4,
1920,
4613,
469,
4,
22,
71,
87,
12,
16,
43,
530,
38,
76,
15,
13,
1247,
4,
22,
17,
515,
17,
12,
16,
626,
18,
2,
5,
62,
386,
12,
```

```
8,
316,
8,
106,
5,
4,
2223,
5244,
16,
480,
66,
3785,
33,
4,
130,
12,
16,
38,
619,
5,
25,
124,
51,
36,
135,
48,
25,
1415,
33,
6,
22,
12,
215,
28,
77,
52,
5,
14,
407,
16,
82,
2,
8,
4,
107,
117,
5952,
```

```
15,
256,
4,
2,
7,
3766,
5,
723,
36,
71,
43,
530,
476,
26,
400,
317,
46,
7,
4,
2,
1029,
13,
104,
88,
4,
381,
15,
297,
98,
32,
2071,
56,
26,
141,
6,
194,
7486,
18,
4,
226,
22,
21,
134,
476,
26,
480,
5,
```

```
 144,
 30,
 5535,
 18,
 51,
 36,
 28,
 224,
 92,
 25,
 104,
 4,
 226,
 65,
 16,
 38,
 1334,
 88,
 12,
 16,
 283,
 5,
 16,
 4472,
 113,
 103,
 32,
 15,
 16,
 5345,
 19,
 178,
 32]
```

[5]:
```
# Review Labels
train_labels[0]
```

[5]: 1

1.4 Prepare the data

[6]:
```
# Use one-hot-encoding to turn data into vectors of 0s and 1s
import numpy as np

def vectorize_sequences(sequences, dimension=10000):

    # Create an all-zero matrix of shape (len(sequences), dimension)
```

```
    results = np.zeros((len(sequences), dimension))

    for i, sequence in enumerate(sequences):
        results[i, sequence] = 1.  # set specific indices of results[i] to 1s
    return results

# Vectorize training data
x_train = vectorize_sequences(train_data)

# Vectorize test data
x_test = vectorize_sequences(test_data)
```

[7]:
```
# Review Vectorized Data
x_train[0]
```

[7]:
```
array([0., 1., 1., …, 0., 0., 0.])
```

[8]:
```
# Vectorize labels
y_train = np.asarray(train_labels).astype('float32')
y_test = np.asarray(test_labels).astype('float32')
```

1.5 Build Keras Neural Network Model

[9]:
```
# Define the model
from keras import models
from keras import layers

# input_shape is size of data vector
# 16 hidden layers
# Use sigmoid for output fxn since binary classification problem
model = models.Sequential()
model.add(layers.Dense(16, activation='relu', input_shape=(10000,)))
model.add(layers.Dense(16, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
```

[10]:
```
# Compile Model

# Use binary_crossentropy for binary classification problem
model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

```
WARNING:tensorflow:From C:\Users\amomu\Anaconda3\lib\site-
packages\tensorflow\python\ops\nn_impl.py:180:
add_dispatch_support.<locals>.wrapper (from tensorflow.python.ops.array_ops) is
deprecated and will be removed in a future version.
Instructions for updating:
```

```
Use tf.where in 2.0, which has the same broadcast rule as np.where
```

1.6 Validate Model

[11]:
```
# Create Validation Set
x_val = x_train[:10000]
partial_x_train = x_train[10000:]

y_val = y_train[:10000]
partial_y_train = y_train[10000:]
```

[12]:
```
# Train Model
# Collect measurement logs
history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))
```

```
WARNING:tensorflow:From C:\Users\amomu\Anaconda3\lib\site-
packages\keras\backend\tensorflow_backend.py:422: The name tf.global_variables
is deprecated. Please use tf.compat.v1.global_variables instead.

Train on 15000 samples, validate on 10000 samples
Epoch 1/20
15000/15000 [==============================] - 2s 111us/step - loss: 0.4974 -
accuracy: 0.7881 - val_loss: 0.3757 - val_accuracy: 0.8659
Epoch 2/20
15000/15000 [==============================] - 1s 91us/step - loss: 0.2911 -
accuracy: 0.9089 - val_loss: 0.3138 - val_accuracy: 0.8773
Epoch 3/20
15000/15000 [==============================] - 1s 92us/step - loss: 0.2179 -
accuracy: 0.9269 - val_loss: 0.2860 - val_accuracy: 0.8869
Epoch 4/20
15000/15000 [==============================] - 1s 92us/step - loss: 0.1745 -
accuracy: 0.9431 - val_loss: 0.2853 - val_accuracy: 0.8871
Epoch 5/20
15000/15000 [==============================] - 1s 93us/step - loss: 0.1398 -
accuracy: 0.9557 - val_loss: 0.2879 - val_accuracy: 0.8867
Epoch 6/20
15000/15000 [==============================] - 1s 92us/step - loss: 0.1177 -
accuracy: 0.9645 - val_loss: 0.2979 - val_accuracy: 0.8830
Epoch 7/20
15000/15000 [==============================] - 1s 94us/step - loss: 0.1019 -
accuracy: 0.9677 - val_loss: 0.3312 - val_accuracy: 0.8767
Epoch 8/20
15000/15000 [==============================] - 1s 91us/step - loss: 0.0794 -
accuracy: 0.9773 - val_loss: 0.3494 - val_accuracy: 0.8760
```

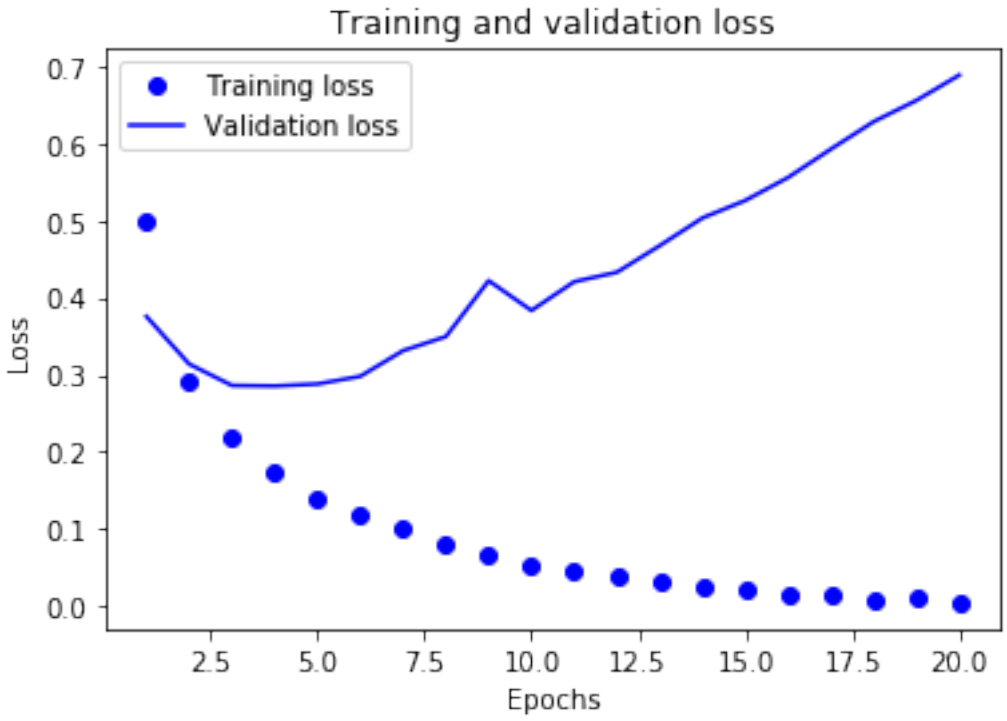
```
Epoch 9/20
15000/15000 [==============================] - 1s 90us/step - loss: 0.0654 -
accuracy: 0.9843 - val_loss: 0.4220 - val_accuracy: 0.8645
Epoch 10/20
15000/15000 [==============================] - 1s 90us/step - loss: 0.0541 -
accuracy: 0.9873 - val_loss: 0.3833 - val_accuracy: 0.8765
Epoch 11/20
15000/15000 [==============================] - 1s 89us/step - loss: 0.0463 -
accuracy: 0.9890 - val_loss: 0.4206 - val_accuracy: 0.8698
Epoch 12/20
15000/15000 [==============================] - 1s 93us/step - loss: 0.0369 -
accuracy: 0.9920 - val_loss: 0.4329 - val_accuracy: 0.8738
Epoch 13/20
15000/15000 [==============================] - 1s 94us/step - loss: 0.0306 -
accuracy: 0.9932 - val_loss: 0.4675 - val_accuracy: 0.8716
Epoch 14/20
15000/15000 [==============================] - 1s 95us/step - loss: 0.0231 -
accuracy: 0.9963 - val_loss: 0.5034 - val_accuracy: 0.8698
Epoch 15/20
15000/15000 [==============================] - 2s 102us/step - loss: 0.0222 -
accuracy: 0.9957 - val_loss: 0.5261 - val_accuracy: 0.8687
Epoch 16/20
15000/15000 [==============================] - 1s 97us/step - loss: 0.0152 -
accuracy: 0.9975 - val_loss: 0.5559 - val_accuracy: 0.8679
Epoch 17/20
15000/15000 [==============================] - 1s 93us/step - loss: 0.0144 -
accuracy: 0.9976 - val_loss: 0.5927 - val_accuracy: 0.8683
Epoch 18/20
15000/15000 [==============================] - 1s 94us/step - loss: 0.0074 -
accuracy: 0.9997 - val_loss: 0.6283 - val_accuracy: 0.8656
Epoch 19/20
15000/15000 [==============================] - 1s 93us/step - loss: 0.0114 -
accuracy: 0.9974 - val_loss: 0.6559 - val_accuracy: 0.8663
Epoch 20/20
15000/15000 [==============================] - 1s 90us/step - loss: 0.0043 -
accuracy: 0.9999 - val_loss: 0.6887 - val_accuracy: 0.8648
```

[13]:
```
# Set measurement history logs
history_dict = history.history
history_dict.keys()
```

[13]:
```
dict_keys(['val_loss', 'val_accuracy', 'loss', 'accuracy'])
```

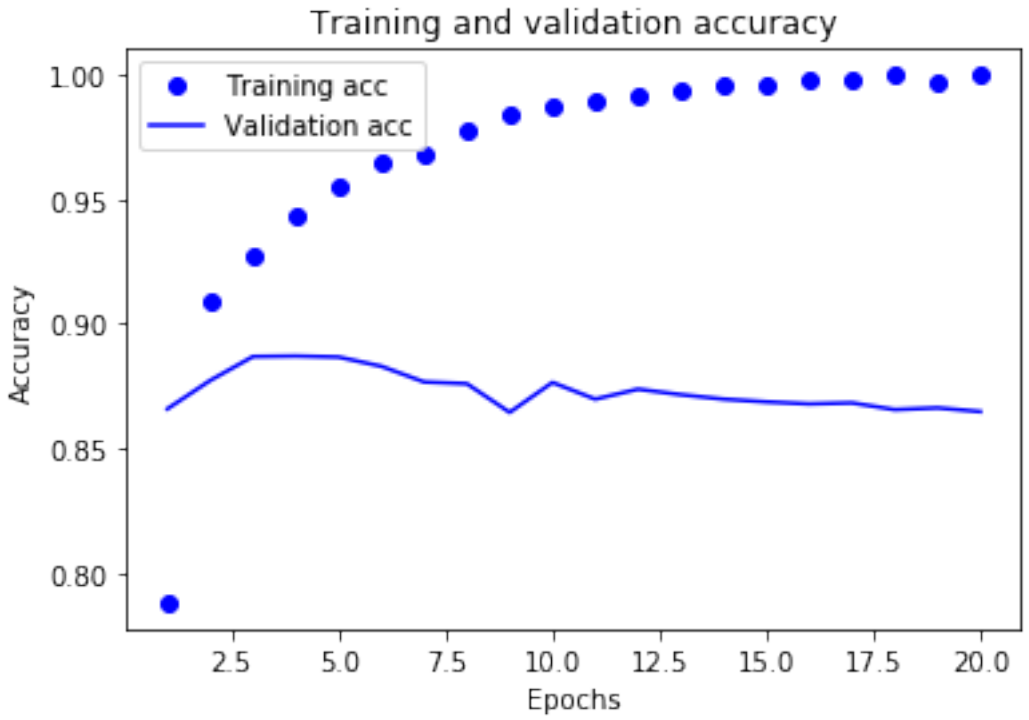
[15]:
```
# Plot the training and validation loss
import matplotlib.pyplot as plt

acc = history.history['accuracy']
```

```
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1, len(acc) + 1)

plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
```



[16]:
```
# Plot the training and validation accuracy

plt.clf()   # clear figure
acc_values = history_dict['accuracy']
val_acc_values = history_dict['val_accuracy']

plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
```

```
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.show()
```



1.7 Re-Train & Evaluate Model

[17]:
```
# Re-train model based on plots
# Accuracy gets lower at 5 epochs
# Loss increases around 4.5
# Re-train with 5 epochs

model = models.Sequential()
model.add(layers.Dense(16, activation='relu', input_shape=(10000,)))
model.add(layers.Dense(16, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

```
model.fit(x_train, y_train, epochs=5, batch_size=512)
results = model.evaluate(x_test, y_test)
```

```
Epoch 1/5
25000/25000 [==============================] - 2s 69us/step - loss: 0.4344 -
accuracy: 0.8267
Epoch 2/5
25000/25000 [==============================] - 2s 61us/step - loss: 0.2475 -
accuracy: 0.9108
Epoch 3/5
25000/25000 [==============================] - 2s 60us/step - loss: 0.1961 -
accuracy: 0.9282
Epoch 4/5
25000/25000 [==============================] - 2s 61us/step - loss: 0.1654 -
accuracy: 0.9417
Epoch 5/5
25000/25000 [==============================] - 2s 62us/step - loss: 0.1426 -
accuracy: 0.9493
25000/25000 [==============================] - 2s 92us/step
```

[18]:
```
# Show Evaluation results
results
```

[18]:
```
[0.32810947559833525, 0.8750399947166443]
```

Achieves an accuracy of 87.5%

1.8 Use Model to Generate predictions

[19]:
```
# Show predictions
# Likely Positive - Close to 1
# Likely Negative - Close to 0
# Indeterminate - Close to 0.5

model.predict(x_test)
```

[19]:
```
array([[0.12863287],
       [0.99971986],
       [0.6582345 ],
       …,
       [0.05825239],
       [0.03601018],
       [0.5615906 ]], dtype=float32)
```