Assignment_10.2

August 3, 2021

[1]:

```
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
import numpy as np
import matplotlib.pyplot as plt
from pathlib import Path
from keras.models import Sequential
from keras.layers import Embedding, Flatten, Dense
import os
from contextlib import redirect_stdout
import time
start_time = time.time()
```

[2]:

```
results_dir = Path('results').joinpath('model_1')
results_dir.mkdir(parents=True, exist_ok=True)
imdb_dir = Path('imdb/aclImdb/')
test_dir = os.path.join(imdb_dir, 'test')
train_dir = os.path.join(imdb_dir, 'train')
```

[3]:

```
training_samples = 200
maxlen = 100
max_words = 1000
embedding_dim = 100
training_samples = 200
validation_samples = 10000
```

[4]:

```
labels = []
texts = []
for label_type in ['neg', 'pos']:
    dir_name = os.path.join(test_dir, label_type)
    for fname in sorted(os.listdir(dir_name)):
        if fname[-4:] == '.txt':
            f = open(os.path.join(dir_name, fname), encoding="utf8")
            texts.append(f.read())
            f.close()
            if label_type == 'neg':
                labels.append(0)
            else:
                labels.append(1)
```

[5]:
```
tokenizer = Tokenizer(num_words=max_words)
tokenizer.fit_on_texts(texts)
sequences = tokenizer.texts_to_sequences(texts)
word_index = tokenizer.word_index
print('Found %s unique tokens.' % len(word_index))
data = pad_sequences(sequences, maxlen=maxlen)
labels = np.asarray(labels)
print('Shape of data tensor:', data.shape)
print('Shape of label tensor:', labels.shape)
```

```
Found 87393 unique tokens.
Shape of data tensor: (25000, 100)
Shape of label tensor: (25000,)
```

[6]:
```
indices = np.arange(data.shape[0])
np.random.shuffle(indices)
data = data[indices]
labels = labels[indices]
x_train = data[:training_samples]
y_train = labels[:training_samples]
x_val = data[training_samples: training_samples + validation_samples]
y_val = labels[training_samples: training_samples + validation_samples]
```

[7]:
```
model = Sequential()
model.add(Embedding(max_words, embedding_dim, input_length=maxlen))
model.add(Flatten())
model.add(Dense(32,activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

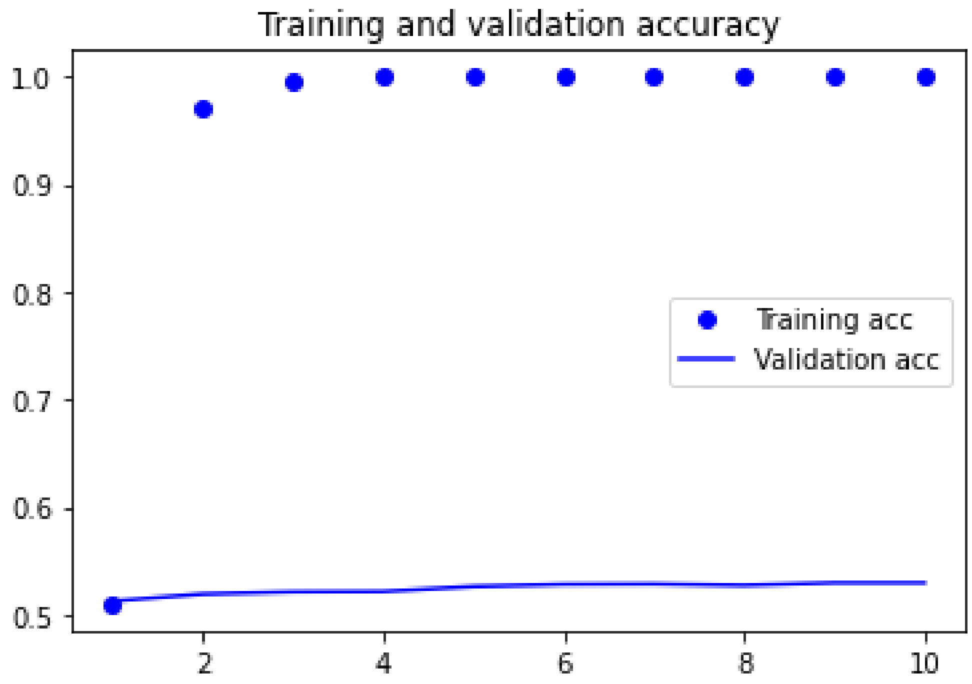
[9]:
```
# Save the summary to file
summary_file = results_dir.joinpath('Assignment_10.2_ModelSummary.txt')
with open(summary_file, 'w') as f:
    with redirect_stdout(f):
        model.summary()
model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['acc'])
history=model.fit(x_train, y_train, epochs=10,␣
 ↪batch_size=32,validation_data=(x_val, y_val))
result_model_file = results_dir.joinpath('pre_trained_glove_model.h5')
model.save_weights(result_model_file)
```

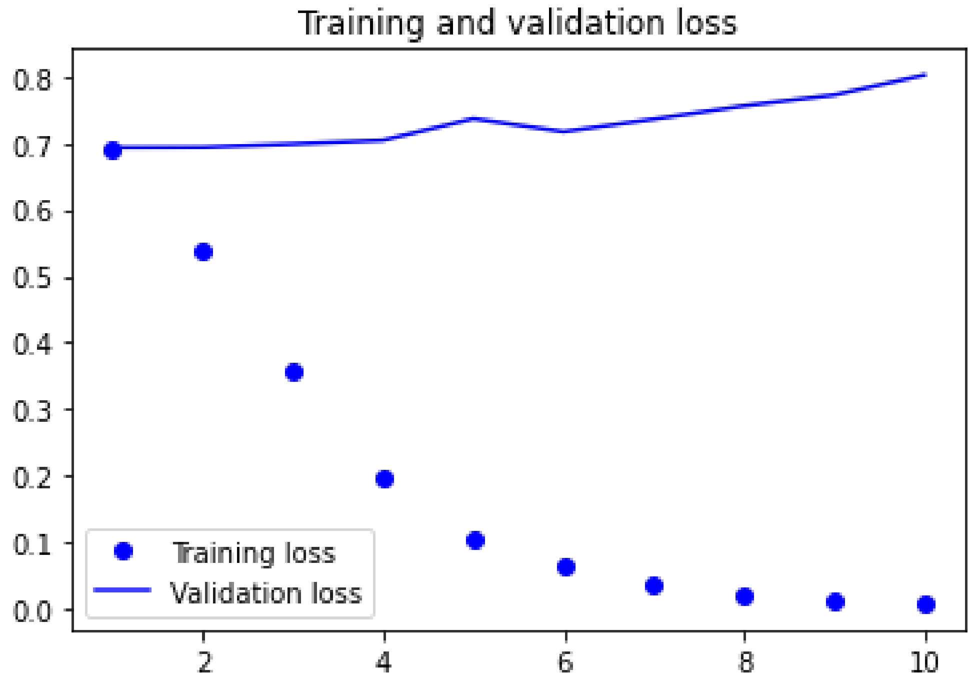
```
Epoch 1/10
7/7 [==============================] - 1s 158ms/step - loss: 0.6907 - acc:
0.5451 - val_loss: 0.6937 - val_acc: 0.5128
Epoch 2/10
7/7 [==============================] - 1s 98ms/step - loss: 0.5506 - acc: 0.9744
```

```
- val_loss: 0.6940 - val_acc: 0.5194
Epoch 3/10
7/7 [==============================] - 1s 100ms/step - loss: 0.3728 - acc:
0.9973 - val_loss: 0.6989 - val_acc: 0.5214
Epoch 4/10
7/7 [==============================] - 1s 97ms/step - loss: 0.2098 - acc: 1.0000
- val_loss: 0.7047 - val_acc: 0.5217
Epoch 5/10
7/7 [==============================] - 1s 104ms/step - loss: 0.1083 - acc:
1.0000 - val_loss: 0.7372 - val_acc: 0.5264
Epoch 6/10
7/7 [==============================] - 1s 107ms/step - loss: 0.0691 - acc:
1.0000 - val_loss: 0.7176 - val_acc: 0.5284
Epoch 7/10
7/7 [==============================] - 1s 100ms/step - loss: 0.0346 - acc:
1.0000 - val_loss: 0.7365 - val_acc: 0.5287
Epoch 8/10
7/7 [==============================] - 1s 101ms/step - loss: 0.0216 - acc:
1.0000 - val_loss: 0.7564 - val_acc: 0.5275
Epoch 9/10
7/7 [==============================] - 1s 100ms/step - loss: 0.0133 - acc:
1.0000 - val_loss: 0.7726 - val_acc: 0.5298
Epoch 10/10
7/7 [==============================] - 1s 92ms/step - loss: 0.0080 - acc: 1.0000
- val_loss: 0.8027 - val_acc: 0.5298
```

[10]:

```
 # Place plot here
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(acc) + 1)
plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()
plt.figure()
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()
img_file = results_dir.joinpath('Assignment_10.2_Model Accuracy Validation.png')
plt.savefig(img_file)
plt.show()
```





[11]:

```
labels=[]
texts=[]
```

```python
for label_type in ['neg', 'pos']:
    dir_name = os.path.join(test_dir, label_type)
    for fname in sorted(os.listdir(dir_name)):
        if fname[-4:] == '.txt':
            f = open(os.path.join(dir_name, fname), encoding="utf8")
            texts.append(f.read())
            f.close()
            if label_type == 'neg':
                labels.append(0)
            else:
                labels.append(1)

sequence = tokenizer.texts_to_sequences(texts)
x_test = pad_sequences(sequences, maxlen=maxlen)
y_test = np.asarray(labels)
model.load_weights(result_model_file)
eval = model.evaluate(x_test, y_test)
print("")
print(eval)
print("Complete: --- %s seconds has passed ---" % (time.time() - start_time))
```

```
782/782 [==============================] - 2s 2ms/step - loss: 0.8052 - acc:
0.5270

[0.8052107691764832, 0.5270400047302246]
Complete: --- 113.56341886520386 seconds has passed ---
```

[ ]:

```

```