

## Assignment 10-2

May 20, 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'))

```

```

[8]: # 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)
```

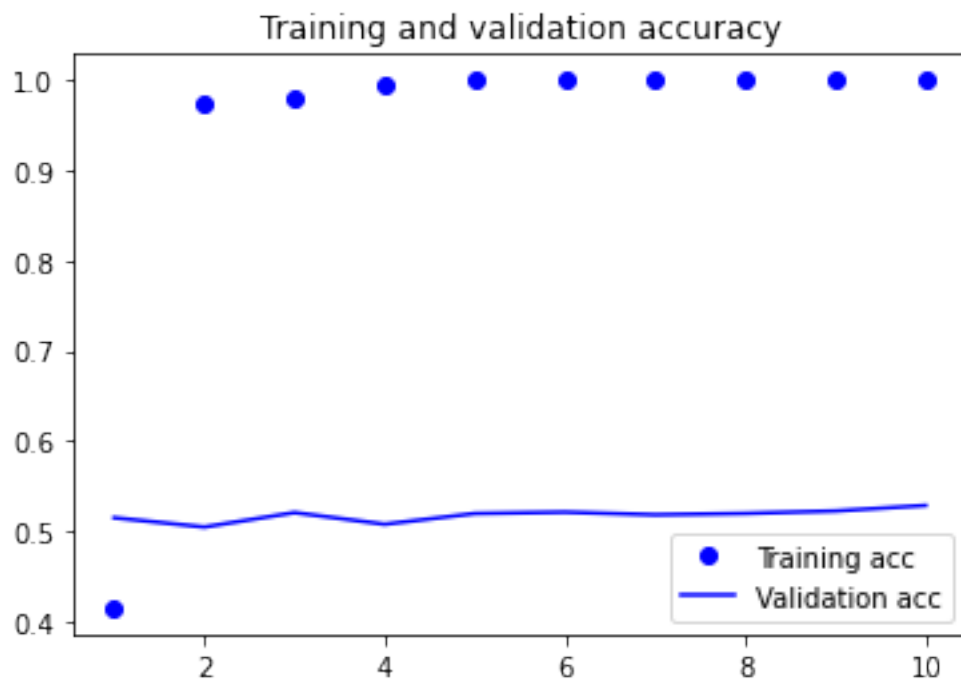
```
Epoch 1/10
7/7 [=====] - 1s 146ms/step - loss: 0.6959 - acc:
0.4105 - val_loss: 0.6926 - val_acc: 0.5150
Epoch 2/10
7/7 [=====] - 1s 99ms/step - loss: 0.5569 - acc: 0.9851
- val_loss: 0.6962 - val_acc: 0.5047
Epoch 3/10
7/7 [=====] - 1s 98ms/step - loss: 0.3725 - acc: 0.9806
- val_loss: 0.6995 - val_acc: 0.5207
Epoch 4/10
7/7 [=====] - 1s 93ms/step - loss: 0.1941 - acc: 0.9963
- val_loss: 0.7312 - val_acc: 0.5078
Epoch 5/10
7/7 [=====] - 1s 103ms/step - loss: 0.1104 - acc:
1.0000 - val_loss: 0.7111 - val_acc: 0.5197
Epoch 6/10
7/7 [=====] - 1s 101ms/step - loss: 0.0627 - acc:
1.0000 - val_loss: 0.7288 - val_acc: 0.5212
Epoch 7/10
7/7 [=====] - 1s 102ms/step - loss: 0.0324 - acc:
1.0000 - val_loss: 0.7288 - val_acc: 0.5183
Epoch 8/10
7/7 [=====] - 1s 101ms/step - loss: 0.0194 - acc:
1.0000 - val_loss: 0.7391 - val_acc: 0.5198
Epoch 9/10
7/7 [=====] - 1s 102ms/step - loss: 0.0119 - acc:
1.0000 - val_loss: 0.7491 - val_acc: 0.5225
Epoch 10/10
7/7 [=====] - 1s 99ms/step - loss: 0.0072 - acc: 1.0000
- val_loss: 0.7554 - val_acc: 0.5286
```

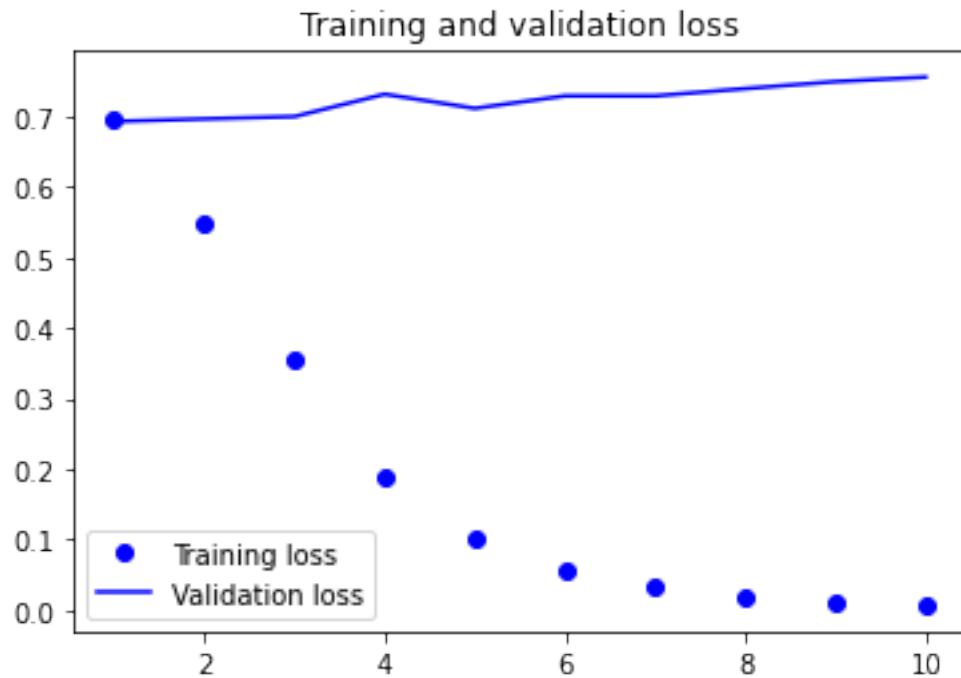
```
[9]: # Place plot here
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']

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()
```





```
[10]: 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)

      sequence = tokenizer.texts_to_sequences(texts)
      x_test = pad_sequences(sequence, 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: 0.7503 - acc:  
0.5324

[0.750349760055542, 0.5324000120162964]

Complete: --- 171.33618259429932 seconds has passed ---

[ ]: