Assignment 5.2

October 5, 2020

1 Assignment 5.2

Brandon Sams

01Oct2020

1.1 Loading the Reuters dataset

[1]:
```
from keras.datasets import reuters
```

[2]:
```
(train_data, train_labels), (test_data, test_labels) = reuters.
↪load_data(num_words=10000)
```

[3]:
```
len(train_data)
```

[3]: 8982

[4]:
```
len(test_data)
```

[4]: 2246

[5]:
```
train_data[10]
```

[5]:
```
[1,
 245,
 273,
 207,
 156,
 53,
 74,
 160,
 26,
 14,
 46,
 296,
 26,
 39,
 74,
```

```
2979,
3554,
14,
46,
4689,
4329,
86,
61,
3499,
4795,
14,
61,
451,
4329,
17,
12]
```

1.2 Decoding newswires back to text

[6]:
```python
word_index = reuters.get_word_index()
reverse_word_index = dict([(value, key) for (key, value) in word_index.items()])
decoded_newswire = ' '.join([reverse_word_index.get(i - 3, '?') for i in␣
↪train_data[0]])
```

[7]:
```python
train_labels[10]
```

[7]:
```
3
```

1.3 Encoding the data

[8]:
```python
import numpy as np

def vectorize_sequences(sequences, dimension=10000):
    results = np.zeros((len(sequences), dimension))
    for i, sequence in enumerate(sequences):
        results[i, sequence] = 1.
    return results
```

[9]:
```python
x_train = vectorize_sequences(train_data)
```

```
The history saving thread hit an unexpected error (OperationalError('database is
locked')).History will not be written to the database.
```

[10]:
```python
x_test = vectorize_sequences(test_data)
```

[11]:
```python
def to_one_hot(labels, dimension=46):
    results = np.zeros((len(labels), dimension))
    for i, label in enumerate(labels):
        results[i, label] = 1.
    return results
```

[12]:
```python
one_hot_train_labels = to_one_hot(train_labels)
one_hot_test_labels = to_one_hot(test_labels)
```

[13]:
```python
from keras.utils.np_utils import to_categorical
```

[14]:
```python
one_hot_train_labels2 = to_categorical(train_labels)
one_hot_test_labels2 = to_categorical(test_labels)
```

[15]:
```python
one_hot_train_labels == one_hot_train_labels2
```

[15]:
```
array([[ True,  True,  True, …,  True,  True,  True],
       [ True,  True,  True, …,  True,  True,  True],
       [ True,  True,  True, …,  True,  True,  True],
       …,
       [ True,  True,  True, …,  True,  True,  True],
       [ True,  True,  True, …,  True,  True,  True],
       [ True,  True,  True, …,  True,  True,  True]])
```

[16]:
```python
one_hot_test_labels == one_hot_test_labels2
```

[16]:
```
array([[ True,  True,  True, …,  True,  True,  True],
       [ True,  True,  True, …,  True,  True,  True],
       [ True,  True,  True, …,  True,  True,  True],
       …,
       [ True,  True,  True, …,  True,  True,  True],
       [ True,  True,  True, …,  True,  True,  True],
       [ True,  True,  True, …,  True,  True,  True]])
```

1.4 Model definition

[17]:
```python
from keras import models
from keras import layers
```

[18]:
```python
model = models.Sequential()
model.add(layers.Dense(64, activation='relu', input_shape=(10000,)))
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(46, activation='softmax'))
```

1.5 Compiling the model

[19]:
```
model.
↪compile(optimizer='rmsprop',loss='categorical_crossentropy',metrics=['accuracy'])
```

1.6 Setting aside a validation set

[20]:
```
x_val = x_train[:1000]
partial_x_train = x_train[1000:]

y_val = one_hot_train_labels[:1000]
partial_y_train = one_hot_train_labels[1000:]
```

1.7 Training the model

[21]:
```
history = model.fit(partial_x_train,
         partial_y_train,
         epochs=20,
         batch_size=512,
         validation_data=(x_val,y_val))
```

```
Epoch 1/20
16/16 [==============================] - 1s 46ms/step - loss: 2.6325 - accuracy:
0.5510 - val_loss: 1.6695 - val_accuracy: 0.6560
Epoch 2/20
16/16 [==============================] - 0s 19ms/step - loss: 1.3669 - accuracy:
0.7095 - val_loss: 1.2558 - val_accuracy: 0.7200
Epoch 3/20
16/16 [==============================] - 0s 19ms/step - loss: 0.9998 - accuracy:
0.7851 - val_loss: 1.0821 - val_accuracy: 0.7580
Epoch 4/20
16/16 [==============================] - 0s 16ms/step - loss: 0.7834 - accuracy:
0.8346 - val_loss: 1.0037 - val_accuracy: 0.7760
Epoch 5/20
16/16 [==============================] - 0s 18ms/step - loss: 0.6236 - accuracy:
0.8696 - val_loss: 0.9357 - val_accuracy: 0.8060
Epoch 6/20
16/16 [==============================] - 0s 16ms/step - loss: 0.4985 - accuracy:
0.8959 - val_loss: 0.8973 - val_accuracy: 0.8140
Epoch 7/20
16/16 [==============================] - 0s 17ms/step - loss: 0.4002 - accuracy:
0.9173 - val_loss: 0.9002 - val_accuracy: 0.8110
Epoch 8/20
16/16 [==============================] - 0s 15ms/step - loss: 0.3270 - accuracy:
0.9313 - val_loss: 0.8824 - val_accuracy: 0.8220
Epoch 9/20
16/16 [==============================] - 0s 14ms/step - loss: 0.2714 - accuracy:
```

```
0.9404 - val_loss: 0.8749 - val_accuracy: 0.8270
Epoch 10/20
16/16 [==============================] - 0s 15ms/step - loss: 0.2292 - accuracy:
0.9455 - val_loss: 0.9406 - val_accuracy: 0.8070
Epoch 11/20
16/16 [==============================] - 0s 15ms/step - loss: 0.2007 - accuracy:
0.9493 - val_loss: 0.9464 - val_accuracy: 0.8100
Epoch 12/20
16/16 [==============================] - 0s 15ms/step - loss: 0.1768 - accuracy:
0.9545 - val_loss: 0.9317 - val_accuracy: 0.8210
Epoch 13/20
16/16 [==============================] - 0s 17ms/step - loss: 0.1599 - accuracy:
0.9538 - val_loss: 1.0298 - val_accuracy: 0.7940
Epoch 14/20
16/16 [==============================] - 0s 17ms/step - loss: 0.1492 - accuracy:
0.9546 - val_loss: 0.9749 - val_accuracy: 0.8180
Epoch 15/20
16/16 [==============================] - 0s 16ms/step - loss: 0.1435 - accuracy:
0.9551 - val_loss: 1.0550 - val_accuracy: 0.8050
Epoch 16/20
16/16 [==============================] - 0s 18ms/step - loss: 0.1312 - accuracy:
0.9560 - val_loss: 1.0047 - val_accuracy: 0.8160
Epoch 17/20
16/16 [==============================] - 0s 15ms/step - loss: 0.1255 - accuracy:
0.9577 - val_loss: 1.0657 - val_accuracy: 0.8040
Epoch 18/20
16/16 [==============================] - 0s 14ms/step - loss: 0.1212 - accuracy:
0.9578 - val_loss: 1.0608 - val_accuracy: 0.8130
Epoch 19/20
16/16 [==============================] - 0s 14ms/step - loss: 0.1146 - accuracy:
0.9577 - val_loss: 1.0782 - val_accuracy: 0.8030
Epoch 20/20
16/16 [==============================] - 0s 14ms/step - loss: 0.1115 - accuracy:
0.9585 - val_loss: 1.1148 - val_accuracy: 0.7980
```

1.8 Plotting the training and validation loss

[22]:
```python
import matplotlib.pyplot as plt
```

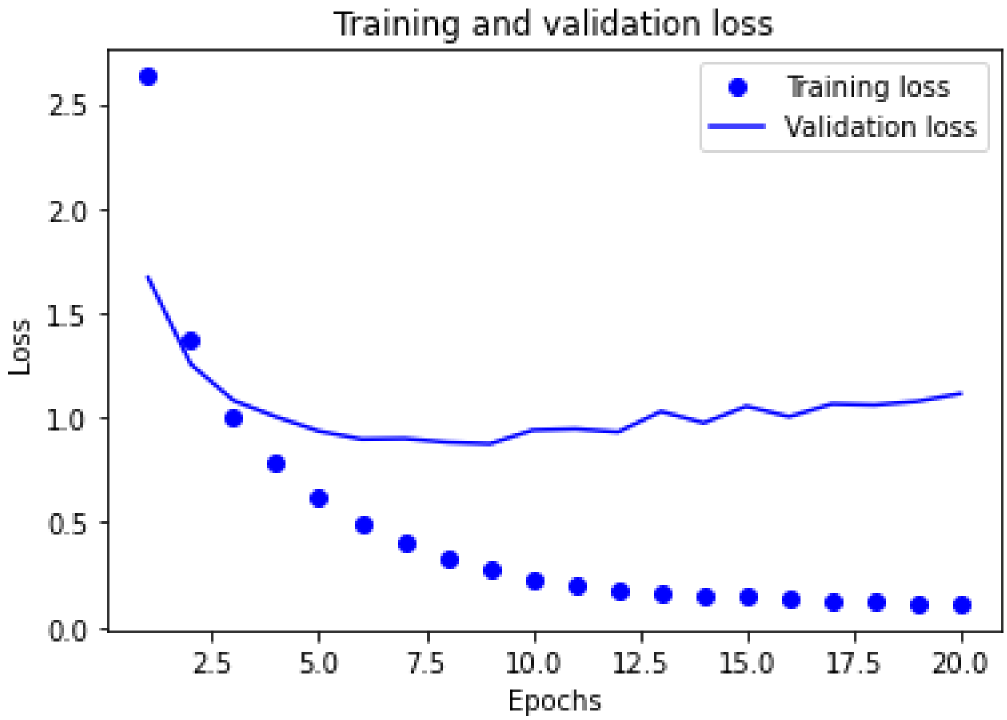
[23]:
```python
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1, len(loss) + 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()
```



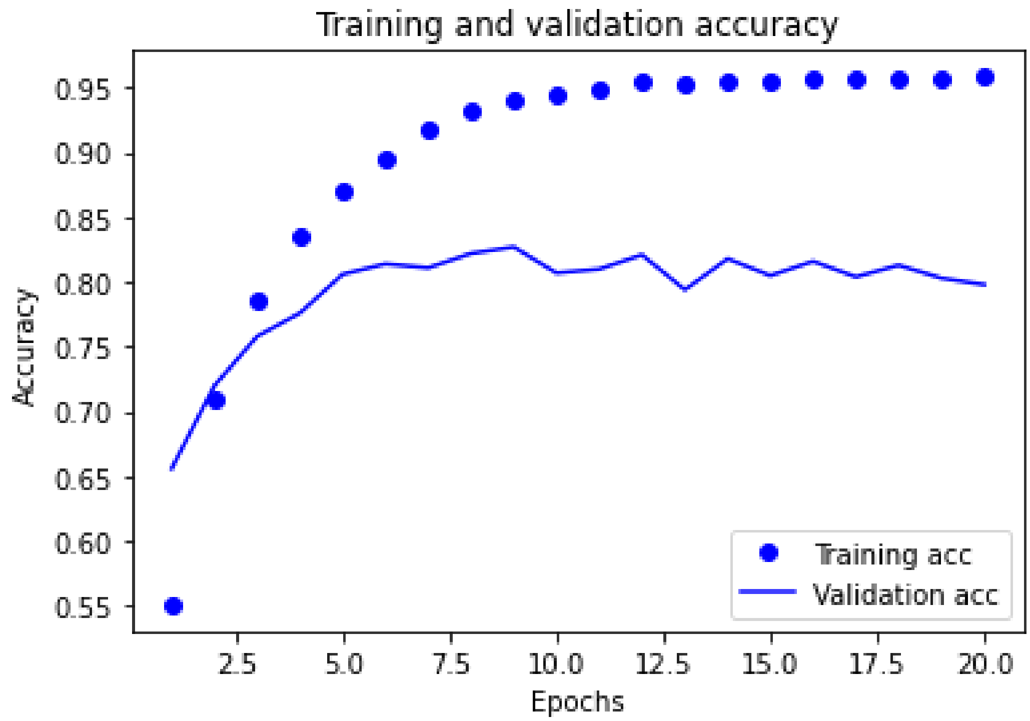
1.9 Plotting the training and validation accuracy

[24]:
```
plt.clf()

acc = history.history['accuracy']
val_acc = history.history['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()
```



1.10 Retraining a model from scratch

[25]:
```
model = models.Sequential()
model.add(layers.Dense(64, activation='relu', input_shape=(10000,)))
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(46, activation='softmax'))

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

model.fit(partial_x_train,
          partial_y_train,
          epochs=9,
          batch_size=512,
          validation_data=(x_val, y_val))
results = model.evaluate(x_test, one_hot_test_labels)
```

```
Epoch 1/9
16/16 [==============================] - 0s 19ms/step - loss: 2.6176 - accuracy:
0.5497 - val_loss: 1.7611 - val_accuracy: 0.6490
Epoch 2/9
16/16 [==============================] - 0s 15ms/step - loss: 1.4284 - accuracy:
```

```
0.7015 - val_loss: 1.3209 - val_accuracy: 0.7210
Epoch 3/9
16/16 [==============================] - 0s 16ms/step - loss: 1.0630 - accuracy:
0.7739 - val_loss: 1.1333 - val_accuracy: 0.7560
Epoch 4/9
16/16 [==============================] - 0s 15ms/step - loss: 0.8338 - accuracy:
0.8195 - val_loss: 1.0294 - val_accuracy: 0.7800
Epoch 5/9
16/16 [==============================] - 0s 17ms/step - loss: 0.6608 - accuracy:
0.8596 - val_loss: 0.9784 - val_accuracy: 0.7980
Epoch 6/9
16/16 [==============================] - 0s 16ms/step - loss: 0.5323 - accuracy:
0.8876 - val_loss: 0.9356 - val_accuracy: 0.7960
Epoch 7/9
16/16 [==============================] - 0s 18ms/step - loss: 0.4263 - accuracy:
0.9116 - val_loss: 0.8940 - val_accuracy: 0.8050
Epoch 8/9
16/16 [==============================] - 0s 15ms/step - loss: 0.3428 - accuracy:
0.9287 - val_loss: 0.9095 - val_accuracy: 0.8080
Epoch 9/9
16/16 [==============================] - 0s 14ms/step - loss: 0.2858 - accuracy:
0.9379 - val_loss: 0.9206 - val_accuracy: 0.8040
71/71 [==============================] - 0s 2ms/step - loss: 0.9912 - accuracy:
0.7872
```

[26]:
```
results
```

[26]: [0.9912168979644775, 0.7871772050857544]

1.11 Generating predictions on new data

[27]:
```
predictions = model.predict(x_test)
```

[28]:
```
predictions[0].shape
```

[28]: (46,)

[29]:
```
np.sum(predictions[0])
```

[29]: 0.99999994

[30]:
```
np.argmax(predictions[0])
```

[30]: 3

[31]:
```
y_train = np.array(train_labels)
y_test = np.array(test_labels)
```

[32]:
```
model.compile(optimizer='rmsprop',
             loss='sparse_categorical_crossentropy',
             metrics=['acc'])
```

1.12 A model with an information bottleneck

[33]:
```
model = models.Sequential()
model.add(layers.Dense(64, activation='relu', input_shape=(10000,)))
model.add(layers.Dense(4, activation='relu'))
model.add(layers.Dense(46, activation='softmax'))

model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
model.fit(partial_x_train,
          partial_y_train,
          epochs=20,
          batch_size=128,
          validation_data=(x_val, y_val))
```

```
Epoch 1/20
63/63 [==============================] - 1s 10ms/step - loss: 3.3417 - accuracy:
0.4009 - val_loss: 3.0469 - val_accuracy: 0.4880
Epoch 2/20
63/63 [==============================] - 1s 11ms/step - loss: 2.8512 - accuracy:
0.3934 - val_loss: 2.7813 - val_accuracy: 0.2810
Epoch 3/20
63/63 [==============================] - 1s 11ms/step - loss: 2.5179 - accuracy:
0.2772 - val_loss: 2.4739 - val_accuracy: 0.2830
Epoch 4/20
63/63 [==============================] - 1s 10ms/step - loss: 1.9670 - accuracy:
0.4124 - val_loss: 1.8134 - val_accuracy: 0.5960
Epoch 5/20
63/63 [==============================] - 1s 10ms/step - loss: 1.3790 - accuracy:
0.6342 - val_loss: 1.5331 - val_accuracy: 0.6070
Epoch 6/20
63/63 [==============================] - 1s 9ms/step - loss: 1.2074 - accuracy:
0.6566 - val_loss: 1.5395 - val_accuracy: 0.6150
Epoch 7/20
63/63 [==============================] - 1s 9ms/step - loss: 1.1132 - accuracy:
0.6853 - val_loss: 1.5188 - val_accuracy: 0.6500
Epoch 8/20
63/63 [==============================] - 1s 8ms/step - loss: 1.0381 - accuracy:
0.7288 - val_loss: 1.5277 - val_accuracy: 0.6610
Epoch 9/20
63/63 [==============================] - 1s 9ms/step - loss: 0.9731 - accuracy:
0.7512 - val_loss: 1.5512 - val_accuracy: 0.6640
```

```
Epoch 10/20
63/63 [==============================] - 0s 7ms/step - loss: 0.9186 - accuracy:
0.7627 - val_loss: 1.5547 - val_accuracy: 0.6680
Epoch 11/20
63/63 [==============================] - 1s 8ms/step - loss: 0.8700 - accuracy:
0.7744 - val_loss: 1.5797 - val_accuracy: 0.6620
Epoch 12/20
63/63 [==============================] - 1s 8ms/step - loss: 0.8269 - accuracy:
0.7805 - val_loss: 1.6208 - val_accuracy: 0.6700
Epoch 13/20
63/63 [==============================] - 1s 9ms/step - loss: 0.7928 - accuracy:
0.7889 - val_loss: 1.7076 - val_accuracy: 0.6710
Epoch 14/20
63/63 [==============================] - 1s 8ms/step - loss: 0.7614 - accuracy:
0.7939 - val_loss: 1.6779 - val_accuracy: 0.6700
Epoch 15/20
63/63 [==============================] - 1s 9ms/step - loss: 0.7376 - accuracy:
0.7954 - val_loss: 1.7956 - val_accuracy: 0.6700
Epoch 16/20
63/63 [==============================] - 1s 9ms/step - loss: 0.7143 - accuracy:
0.7997 - val_loss: 1.7683 - val_accuracy: 0.6710
Epoch 17/20
63/63 [==============================] - 0s 8ms/step - loss: 0.6947 - accuracy:
0.8004 - val_loss: 1.8461 - val_accuracy: 0.6660
Epoch 18/20
63/63 [==============================] - 1s 13ms/step - loss: 0.6769 - accuracy:
0.8028 - val_loss: 1.9517 - val_accuracy: 0.6600
Epoch 19/20
63/63 [==============================] - 0s 8ms/step - loss: 0.6595 - accuracy:
0.8067 - val_loss: 2.0261 - val_accuracy: 0.6710
Epoch 20/20
63/63 [==============================] - 0s 8ms/step - loss: 0.6478 - accuracy:
0.8071 - val_loss: 2.0127 - val_accuracy: 0.6610
```

```
[33]: <tensorflow.python.keras.callbacks.History at 0x7fb3282d49d0>
```

```
[ ]:
```