assignment5.2

December 17, 2021

[1]:
```
# A multi-class classification
import keras
keras.__version__
```

```
Using TensorFlow backend.
```

[1]:
```
'2.3.1'
```

[2]:
```
from keras.datasets import reuters

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

We have 8982 training data and 2246 test data

[3]:
```
print('train_data', len(train_data))
print('test_data', len(test_data))
```

```
train_data 8982
test_data 2246
```

[4]:
```
# Preparing the data
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

# Our vectorized training data
x_train = vectorize_sequences(train_data)
# Our vectorized test data
x_test = vectorize_sequences(test_data)
```

[5]:
```
from keras.utils.np_utils import to_categorical

one_hot_train_labels = to_categorical(train_labels)
one_hot_test_labels = to_categorical(test_labels)
```

[6]:
```python
# Building the network
from keras import models
from keras import layers

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'))
```

[7]:
```python
model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

[8]:
```python
# Validating our approach
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:]
```

[9]:
```python
# train the network for 20 epochs
history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))
```

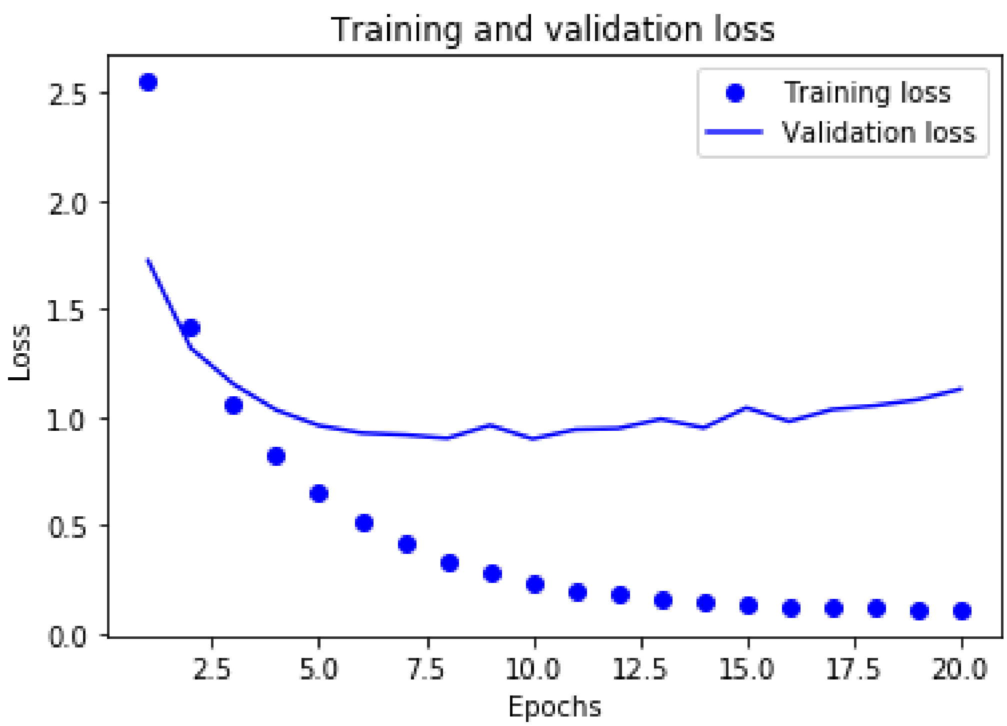
```
Train on 7982 samples, validate on 1000 samples
Epoch 1/20
7982/7982 [==============================] - 1s 124us/step - loss: 2.5492 -
accuracy: 0.5175 - val_loss: 1.7229 - val_accuracy: 0.6350
Epoch 2/20
7982/7982 [==============================] - 1s 91us/step - loss: 1.4146 -
accuracy: 0.7090 - val_loss: 1.3197 - val_accuracy: 0.7100
Epoch 3/20
7982/7982 [==============================] - 1s 94us/step - loss: 1.0551 -
accuracy: 0.7764 - val_loss: 1.1530 - val_accuracy: 0.7530
Epoch 4/20
7982/7982 [==============================] - 1s 96us/step - loss: 0.8284 -
accuracy: 0.8217 - val_loss: 1.0334 - val_accuracy: 0.7870
Epoch 5/20
7982/7982 [==============================] - 1s 93us/step - loss: 0.6554 -
accuracy: 0.8629 - val_loss: 0.9619 - val_accuracy: 0.8080
Epoch 6/20
7982/7982 [==============================] - 1s 90us/step - loss: 0.5230 -
accuracy: 0.8909 - val_loss: 0.9274 - val_accuracy: 0.8080
Epoch 7/20
7982/7982 [==============================] - 1s 90us/step - loss: 0.4182 -
```

```
accuracy: 0.9108 - val_loss: 0.9188 - val_accuracy: 0.8100
Epoch 8/20
7982/7982 [==============================] - 1s 101us/step - loss: 0.3364 -
accuracy: 0.9315 - val_loss: 0.9031 - val_accuracy: 0.8160
Epoch 9/20
7982/7982 [==============================] - 1s 104us/step - loss: 0.2790 -
accuracy: 0.9386 - val_loss: 0.9630 - val_accuracy: 0.8020
Epoch 10/20
7982/7982 [==============================] - 1s 103us/step - loss: 0.2402 -
accuracy: 0.9440 - val_loss: 0.9003 - val_accuracy: 0.8280
Epoch 11/20
7982/7982 [==============================] - 1s 95us/step - loss: 0.2009 -
accuracy: 0.9470 - val_loss: 0.9436 - val_accuracy: 0.8160
Epoch 12/20
7982/7982 [==============================] - 1s 96us/step - loss: 0.1798 -
accuracy: 0.9510 - val_loss: 0.9486 - val_accuracy: 0.8220
Epoch 13/20
7982/7982 [==============================] - 1s 96us/step - loss: 0.1626 -
accuracy: 0.9526 - val_loss: 0.9917 - val_accuracy: 0.8140
Epoch 14/20
7982/7982 [==============================] - 1s 112us/step - loss: 0.1503 -
accuracy: 0.9534 - val_loss: 0.9516 - val_accuracy: 0.8340
Epoch 15/20
7982/7982 [==============================] - 1s 179us/step - loss: 0.1379 -
accuracy: 0.9544 - val_loss: 1.0441 - val_accuracy: 0.8090
Epoch 16/20
7982/7982 [==============================] - 1s 142us/step - loss: 0.1301 -
accuracy: 0.9582 - val_loss: 0.9809 - val_accuracy: 0.8150
Epoch 17/20
7982/7982 [==============================] - 1s 149us/step - loss: 0.1231 -
accuracy: 0.9579 - val_loss: 1.0358 - val_accuracy: 0.8160
Epoch 18/20
7982/7982 [==============================] - 1s 94us/step - loss: 0.1178 -
accuracy: 0.9588 - val_loss: 1.0535 - val_accuracy: 0.8190
Epoch 19/20
7982/7982 [==============================] - 1s 96us/step - loss: 0.1127 -
accuracy: 0.9579 - val_loss: 1.0812 - val_accuracy: 0.8070
Epoch 20/20
7982/7982 [==============================] - 1s 98us/step - loss: 0.1129 -
accuracy: 0.9570 - val_loss: 1.1299 - val_accuracy: 0.8030
```

[10]:

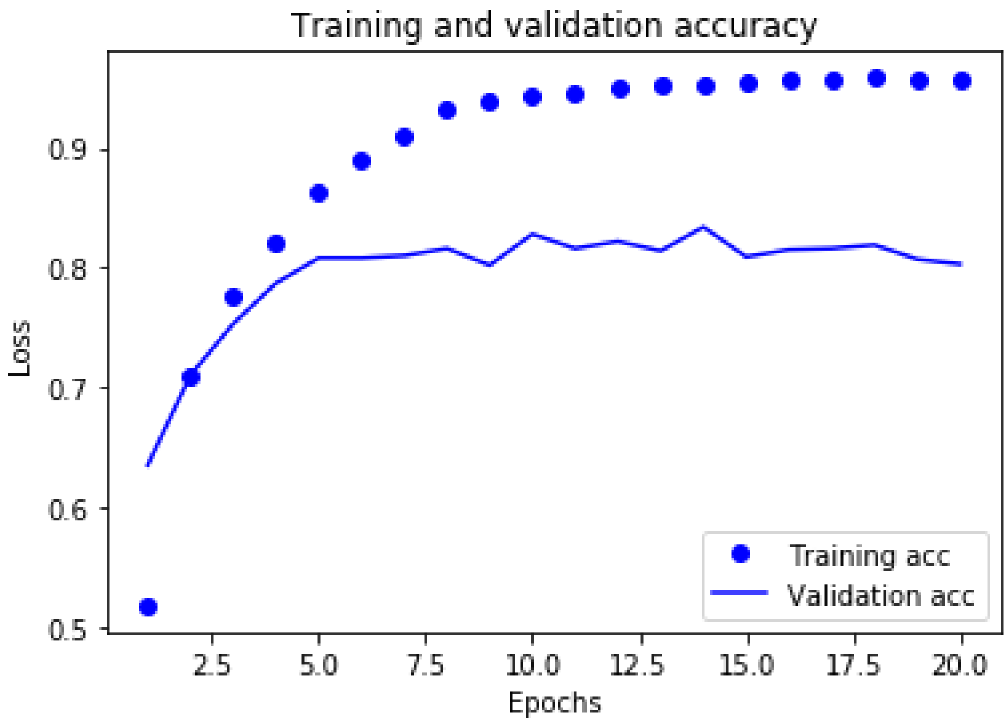
```python
# loss and accuracy curves

import matplotlib.pyplot as plt

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()
```



[11]:
```
plt.clf()   # clear figure

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('Loss')
plt.legend()
```

```
plt.show()
```



[13]:
```
results = model.evaluate(x_test, one_hot_test_labels)
results
```

```
2246/2246 [==============================] - 0s 108us/step
```

[13]: [1.256862762031657, 0.7831701040267944]

0.0.1 accuracy reached is ~78%

0.0.2 Generating predictions on new data

[16]:
```
predictions = model.predict(x_test)
predictions[0].shape
```

[16]: (46,)