NewsClassifier

January 18, 2021

[2]:
```
import keras
```

[3]:
```
from keras.datasets import reuters

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

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-
datasets/reuters.npz
2113536/2110848 [==============================] - 0s 0us/step
```

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

[4]:
```
8982
```

[5]:
```
len(test_data)
```

[5]:
```
2246
```

[6]:
```
train_data[10]
```

[6]:
```
[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]
```

[7]:
```
word_index = reuters.get_word_index()
reverse_word_index = dict([(value, key) for (key, value) in word_index.items()])
# Note that our indices were offset by 3
# because 0, 1 and 2 are reserved indices for "padding", "start of sequence",␣
 ↪and "unknown".
decoded_newswire = ' '.join([reverse_word_index.get(i - 3, '?') for i in␣
 ↪train_data[0]])
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-
datasets/reuters_word_index.json
557056/550378 [==============================] - 0s 0us/step
```

[8]:
```
decoded_newswire
```

[8]:
```
'? ? ? said as a result of its december acquisition of space co it expects
earnings per share in 1987 of 1 15 to 1 30 dlrs per share up from 70 cts in 1986
the company said pretax net should rise to nine to 10 mln dlrs from six mln dlrs
in 1986 and rental operation revenues to 19 to 22 mln dlrs from 12 5 mln dlrs it
said cash flow per share this year should be 2 50 to three dlrs reuter 3'
```

[9]:
```
train_labels[10]
```

[9]:
```
3
```

[10]:
```
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)
```

[11]:
```
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

# Our vectorized training labels
one_hot_train_labels = to_one_hot(train_labels)
# Our vectorized test labels
one_hot_test_labels = to_one_hot(test_labels)
```

[12]:
```
from keras.utils.np_utils import to_categorical

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

[13]:
```
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'))
```

[14]:
```
model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

[15]:
```
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:]
```

[16]:
```
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 36ms/step - loss: 2.5963 - accuracy:
```

```
0.5253 - val_loss: 1.7492 - val_accuracy: 0.6290
Epoch 2/20
16/16 [==============================] - 0s 20ms/step - loss: 1.4373 - accuracy:
0.6936 - val_loss: 1.3158 - val_accuracy: 0.7060
Epoch 3/20
16/16 [==============================] - 0s 18ms/step - loss: 1.0752 - accuracy:
0.7610 - val_loss: 1.1443 - val_accuracy: 0.7360
Epoch 4/20
16/16 [==============================] - 0s 15ms/step - loss: 0.8514 - accuracy:
0.8162 - val_loss: 1.0674 - val_accuracy: 0.7710
Epoch 5/20
16/16 [==============================] - 0s 17ms/step - loss: 0.6817 - accuracy:
0.8599 - val_loss: 0.9748 - val_accuracy: 0.8010
Epoch 6/20
16/16 [==============================] - 0s 23ms/step - loss: 0.5524 - accuracy:
0.8867 - val_loss: 0.9494 - val_accuracy: 0.8060
Epoch 7/20
16/16 [==============================] - 1s 43ms/step - loss: 0.4444 - accuracy:
0.9088 - val_loss: 0.9197 - val_accuracy: 0.8110
Epoch 8/20
16/16 [==============================] - 0s 15ms/step - loss: 0.3612 - accuracy:
0.9211 - val_loss: 0.8924 - val_accuracy: 0.8210
Epoch 9/20
16/16 [==============================] - 0s 14ms/step - loss: 0.2983 - accuracy:
0.9339 - val_loss: 0.9420 - val_accuracy: 0.8050
Epoch 10/20
16/16 [==============================] - 0s 20ms/step - loss: 0.2492 - accuracy:
0.9432 - val_loss: 0.9436 - val_accuracy: 0.8130
Epoch 11/20
16/16 [==============================] - 1s 34ms/step - loss: 0.2152 - accuracy:
0.9479 - val_loss: 0.9160 - val_accuracy: 0.8220
Epoch 12/20
16/16 [==============================] - 0s 28ms/step - loss: 0.1891 - accuracy:
0.9486 - val_loss: 0.9536 - val_accuracy: 0.8160
Epoch 13/20
16/16 [==============================] - 1s 67ms/step - loss: 0.1679 - accuracy:
0.9516 - val_loss: 1.0230 - val_accuracy: 0.8000
Epoch 14/20
16/16 [==============================] - 1s 52ms/step - loss: 0.1534 - accuracy:
0.9536 - val_loss: 1.0026 - val_accuracy: 0.8110
Epoch 15/20
16/16 [==============================] - 1s 71ms/step - loss: 0.1470 - accuracy:
0.9541 - val_loss: 0.9916 - val_accuracy: 0.8160
Epoch 16/20
16/16 [==============================] - 1s 39ms/step - loss: 0.1348 - accuracy:
0.9558 - val_loss: 1.0270 - val_accuracy: 0.8130
Epoch 17/20
16/16 [==============================] - 1s 45ms/step - loss: 0.1276 - accuracy:
```

```
0.9559 - val_loss: 1.0498 - val_accuracy: 0.8160
Epoch 18/20
16/16 [==============================] - 0s 16ms/step - loss: 0.1218 - accuracy:
0.9559 - val_loss: 1.0894 - val_accuracy: 0.8060
Epoch 19/20
16/16 [==============================] - 0s 16ms/step - loss: 0.1210 - accuracy:
0.9558 - val_loss: 1.0924 - val_accuracy: 0.8060
Epoch 20/20
16/16 [==============================] - 0s 16ms/step - loss: 0.1099 - accuracy:
0.9588 - val_loss: 1.0967 - val_accuracy: 0.8180
```

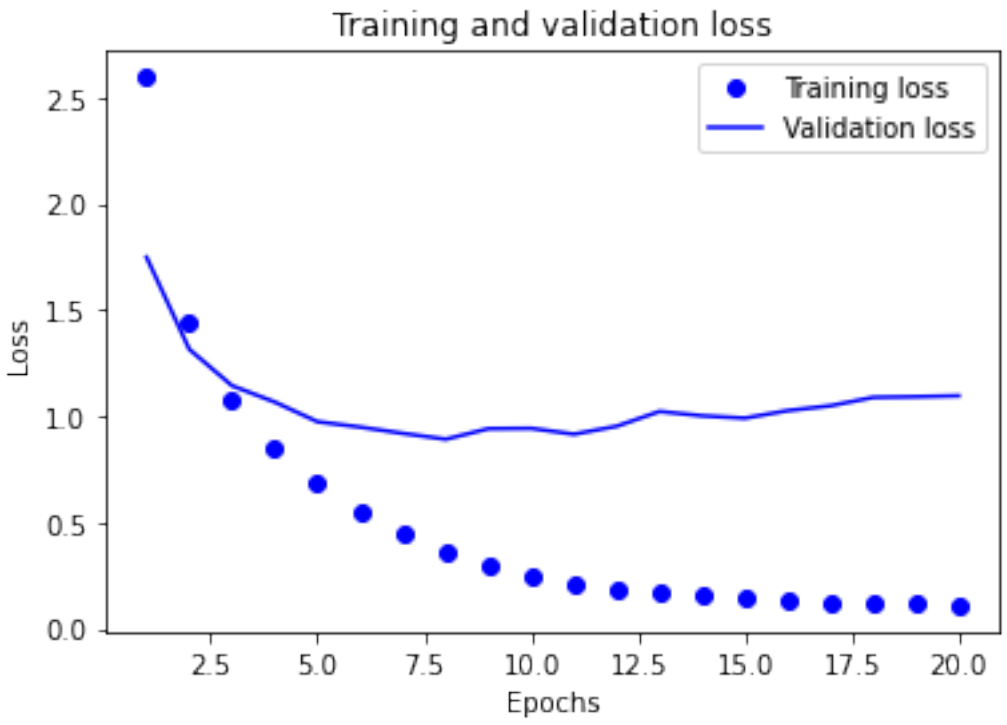
[17]:
```
import matplotlib.pyplot as plt

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()
```



[21]:
```
history.history.keys()
```

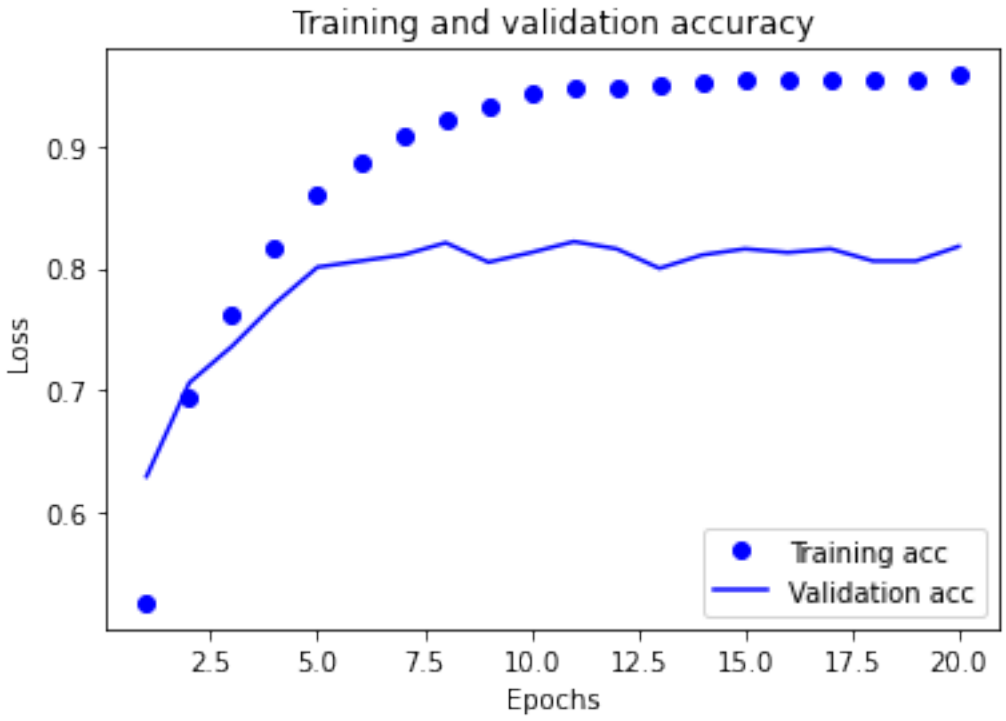
[21]:
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

[22]:
```
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()
```



[23]:
```
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=8,
          batch_size=512,
          validation_data=(x_val, y_val))
results = model.evaluate(x_test, one_hot_test_labels)
```

```
Epoch 1/8
16/16 [==============================] - 1s 80ms/step - loss: 2.7431 - accuracy:
0.4712 - val_loss: 1.8407 - val_accuracy: 0.6280
Epoch 2/8
16/16 [==============================] - 1s 54ms/step - loss: 1.5185 - accuracy:
0.6929 - val_loss: 1.3506 - val_accuracy: 0.7040
Epoch 3/8
16/16 [==============================] - 0s 19ms/step - loss: 1.1227 - accuracy:
0.7558 - val_loss: 1.1825 - val_accuracy: 0.7340
Epoch 4/8
16/16 [==============================] - 0s 30ms/step - loss: 0.8908 - accuracy:
0.8061 - val_loss: 1.0557 - val_accuracy: 0.7760
Epoch 5/8
16/16 [==============================] - 1s 52ms/step - loss: 0.7141 - accuracy:
0.8484 - val_loss: 0.9953 - val_accuracy: 0.7910
Epoch 6/8
16/16 [==============================] - 0s 28ms/step - loss: 0.5717 - accuracy:
0.8822 - val_loss: 0.9306 - val_accuracy: 0.8000
Epoch 7/8
16/16 [==============================] - 0s 19ms/step - loss: 0.4628 - accuracy:
0.9065 - val_loss: 0.9118 - val_accuracy: 0.7970
Epoch 8/8
16/16 [==============================] - 0s 27ms/step - loss: 0.3757 - accuracy:
0.9230 - val_loss: 0.8748 - val_accuracy: 0.8160
71/71 [==============================] - 0s 3ms/step - loss: 0.9902 - accuracy:
0.7845
```

[24]:
```
results
```

[24]:
```
[0.9902112483978271, 0.7845057845115662]
```

[25]:
```
import copy

test_labels_copy = copy.copy(test_labels)
np.random.shuffle(test_labels_copy)
```

```
float(np.sum(np.array(test_labels) == np.array(test_labels_copy))) /␣
↪len(test_labels)
```

[25]: 0.18432769367764915

[26]:
```
predictions = model.predict(x_test)
```

[27]:
```
np.sum(predictions[0])
```

[27]: 1.0

[28]:
```
np.argmax(predictions[0])
```

[28]: 3

[29]:
```
y_train = np.array(train_labels)
y_test = np.array(test_labels)
```

[30]:
```
model.compile(optimizer='rmsprop', loss='sparse_categorical_crossentropy',␣
↪metrics=['acc'])
```

[31]:
```
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 11ms/step - loss: 3.2206 - accuracy:
0.4178 - val_loss: 2.6998 - val_accuracy: 0.5030
Epoch 2/20
63/63 [==============================] - 0s 8ms/step - loss: 2.4876 - accuracy:
0.5200 - val_loss: 2.2924 - val_accuracy: 0.5390
Epoch 3/20
63/63 [==============================] - 1s 8ms/step - loss: 1.9753 - accuracy:
0.5773 - val_loss: 1.7615 - val_accuracy: 0.5830
Epoch 4/20
63/63 [==============================] - 0s 7ms/step - loss: 1.5207 - accuracy:
0.5951 - val_loss: 1.5400 - val_accuracy: 0.5890
```

```
Epoch 5/20
63/63 [==============================] - 1s 10ms/step - loss: 1.3423 - accuracy:
0.6364 - val_loss: 1.5152 - val_accuracy: 0.6130
Epoch 6/20
63/63 [==============================] - 0s 7ms/step - loss: 1.2318 - accuracy:
0.6622 - val_loss: 1.4717 - val_accuracy: 0.6290
Epoch 7/20
63/63 [==============================] - 0s 8ms/step - loss: 1.1440 - accuracy:
0.6745 - val_loss: 1.4340 - val_accuracy: 0.6280
Epoch 8/20
63/63 [==============================] - 0s 8ms/step - loss: 1.0734 - accuracy:
0.6897 - val_loss: 1.4369 - val_accuracy: 0.6480
Epoch 9/20
63/63 [==============================] - 0s 7ms/step - loss: 1.0158 - accuracy:
0.7046 - val_loss: 1.4404 - val_accuracy: 0.6490
Epoch 10/20
63/63 [==============================] - 1s 10ms/step - loss: 0.9629 - accuracy:
0.7211 - val_loss: 1.4694 - val_accuracy: 0.6600
Epoch 11/20
63/63 [==============================] - 1s 9ms/step - loss: 0.9170 - accuracy:
0.7404 - val_loss: 1.5044 - val_accuracy: 0.6680
Epoch 12/20
63/63 [==============================] - 1s 8ms/step - loss: 0.8723 - accuracy:
0.7519 - val_loss: 1.5138 - val_accuracy: 0.6730
Epoch 13/20
63/63 [==============================] - 0s 7ms/step - loss: 0.8215 - accuracy:
0.7657 - val_loss: 1.5430 - val_accuracy: 0.6990
Epoch 14/20
63/63 [==============================] - 1s 8ms/step - loss: 0.7699 - accuracy:
0.7925 - val_loss: 1.5150 - val_accuracy: 0.6940
Epoch 15/20
63/63 [==============================] - 0s 7ms/step - loss: 0.7146 - accuracy:
0.8056 - val_loss: 1.5263 - val_accuracy: 0.7070
Epoch 16/20
63/63 [==============================] - 1s 8ms/step - loss: 0.6642 - accuracy:
0.8235 - val_loss: 1.5727 - val_accuracy: 0.7120
Epoch 17/20
63/63 [==============================] - 0s 7ms/step - loss: 0.6171 - accuracy:
0.8381 - val_loss: 1.6788 - val_accuracy: 0.7060
Epoch 18/20
63/63 [==============================] - 1s 9ms/step - loss: 0.5767 - accuracy:
0.8489 - val_loss: 1.6295 - val_accuracy: 0.7140
Epoch 19/20
63/63 [==============================] - 1s 12ms/step - loss: 0.5422 - accuracy:
0.8571 - val_loss: 1.6671 - val_accuracy: 0.7210
Epoch 20/20
63/63 [==============================] - 0s 7ms/step - loss: 0.5102 - accuracy:
0.8649 - val_loss: 1.7075 - val_accuracy: 0.7130
```

[31]:
```
<tensorflow.python.keras.callbacks.History at 0x7f26b823f2b0>
```

[ ]:

[ ]: