# Reuters

May 7, 2020

## 0.1 导入路透社的数据集

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

加载数据的时候就限定了单词的索引不超过10000，返回的数据就仅包含这10000个单词的了。

[9]:
```
len(train_data)
```

[9]:
```
8982
```

如何解码单词：

[11]:
```
word_index = reuters.get_word_index()
reverse_word_index = dict([(value, key) for (key, value) in word_index.items()])
decode_newswire = ' '.join([reverse_word_index.get(i-3, '?') for i in␣
 ↪train_data[0]])
decode_newswire
```

[11]:
```
'? ? ? 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'
```

## 0.2 1. 数据向量化

[12]:
```
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

x_train = vectorize_sequences(train_data)
```

```
x_test = vectorize_sequences(test_data)

#
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

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

由于要分类的最终数据有46个分类，因此中间Dense层16维则太小，修改为64维。

[15]:

```
from keras import layers
from keras import models

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'))
```

```
WARNING:tensorflow:From /usr/local/lib/python3.7/site-
packages/keras/backend/tensorflow_backend.py:63: The name tf.get_default_graph
is deprecated. Please use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From /usr/local/lib/python3.7/site-
packages/keras/backend/tensorflow_backend.py:488: The name tf.placeholder is
deprecated. Please use tf.compat.v1.placeholder instead.

WARNING:tensorflow:From /usr/local/lib/python3.7/site-
packages/keras/backend/tensorflow_backend.py:3626: The name tf.random_uniform is
deprecated. Please use tf.random.uniform instead.
```

softmax：输出各个位置上的概率的激活函数，所有概率之和 = 1

[20]:

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

编译了模型，categorical_crossentropy用于评估2个概率之间的差值。

[18]:

```
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:]
```

[21]:

```
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 164us/step - loss: 2.5366 - acc:
0.5018 - val_loss: 1.7197 - val_acc: 0.6060
Epoch 2/20
7982/7982 [==============================] - 1s 112us/step - loss: 1.4424 - acc:
0.6888 - val_loss: 1.3531 - val_acc: 0.6960
Epoch 3/20
7982/7982 [==============================] - 1s 109us/step - loss: 1.1020 - acc:
0.7578 - val_loss: 1.1873 - val_acc: 0.7340
Epoch 4/20
7982/7982 [==============================] - 1s 109us/step - loss: 0.8757 - acc:
0.8115 - val_loss: 1.0790 - val_acc: 0.7660
Epoch 5/20
7982/7982 [==============================] - 1s 111us/step - loss: 0.7033 - acc:
0.8524 - val_loss: 1.0480 - val_acc: 0.7630
Epoch 6/20
7982/7982 [==============================] - 1s 114us/step - loss: 0.5656 - acc:
0.8809 - val_loss: 0.9276 - val_acc: 0.8040
Epoch 7/20
7982/7982 [==============================] - 1s 123us/step - loss: 0.4518 - acc:
0.9077 - val_loss: 0.9061 - val_acc: 0.8060
Epoch 8/20
7982/7982 [==============================] - 1s 113us/step - loss: 0.3676 - acc:
0.9233 - val_loss: 0.8956 - val_acc: 0.8110
Epoch 9/20
7982/7982 [==============================] - 1s 123us/step - loss: 0.2998 - acc:
0.9340 - val_loss: 0.9197 - val_acc: 0.7970
Epoch 10/20
7982/7982 [==============================] - 1s 117us/step - loss: 0.2533 - acc:
0.9425 - val_loss: 0.9241 - val_acc: 0.8030
Epoch 11/20
7982/7982 [==============================] - 1s 113us/step - loss: 0.2158 - acc:
0.9464 - val_loss: 0.8988 - val_acc: 0.8170
Epoch 12/20
7982/7982 [==============================] - 1s 111us/step - loss: 0.1912 - acc:
0.9480 - val_loss: 0.9210 - val_acc: 0.8080
Epoch 13/20
7982/7982 [==============================] - 1s 110us/step - loss: 0.1692 - acc:
0.9530 - val_loss: 0.9590 - val_acc: 0.8000
Epoch 14/20
```

```
7982/7982 [==============================] - 1s 111us/step - loss: 0.1558 - acc:
0.9533 - val_loss: 0.9618 - val_acc: 0.8030
Epoch 15/20
7982/7982 [==============================] - 1s 112us/step - loss: 0.1398 - acc:
0.9550 - val_loss: 1.0184 - val_acc: 0.8030
Epoch 16/20
7982/7982 [==============================] - 1s 111us/step - loss: 0.1349 - acc:
0.9565 - val_loss: 1.0351 - val_acc: 0.8010
Epoch 17/20
7982/7982 [==============================] - 1s 131us/step - loss: 0.1264 - acc:
0.9585 - val_loss: 0.9971 - val_acc: 0.8030
Epoch 18/20
7982/7982 [==============================] - 1s 117us/step - loss: 0.1208 - acc:
0.9570 - val_loss: 1.0390 - val_acc: 0.8060
Epoch 19/20
7982/7982 [==============================] - 1s 110us/step - loss: 0.1166 - acc:
0.9557 - val_loss: 1.0511 - val_acc: 0.8050
Epoch 20/20
7982/7982 [==============================] - 1s 129us/step - loss: 0.1086 - acc:
0.9578 - val_loss: 1.0309 - val_acc: 0.8080
```

[22]:

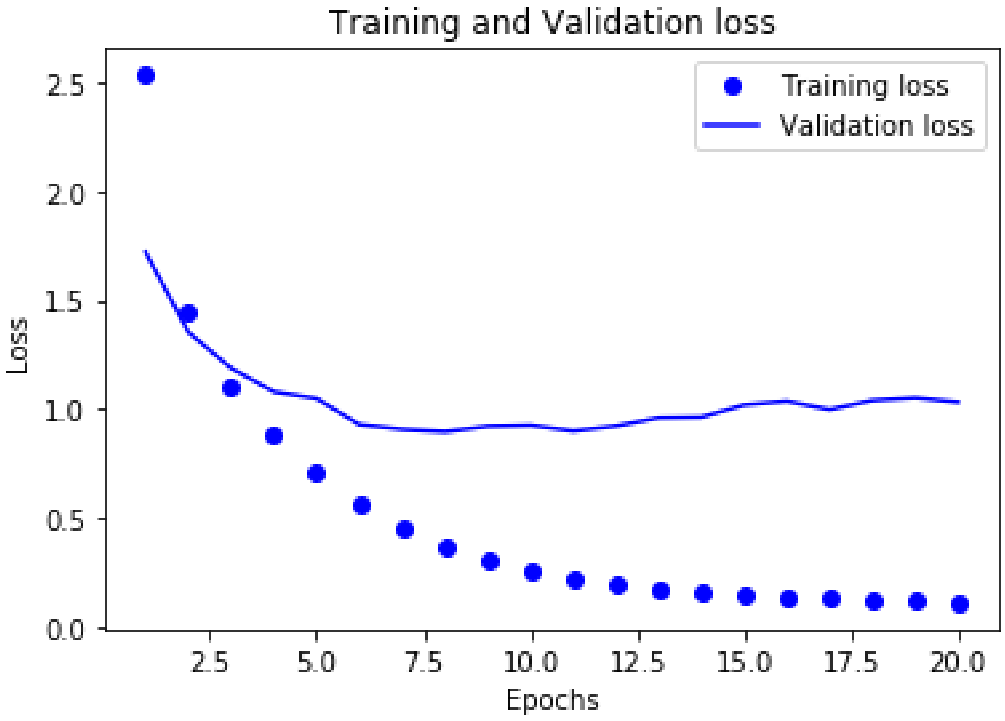
```python
import matplotlib.pyplot as plt

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()
```

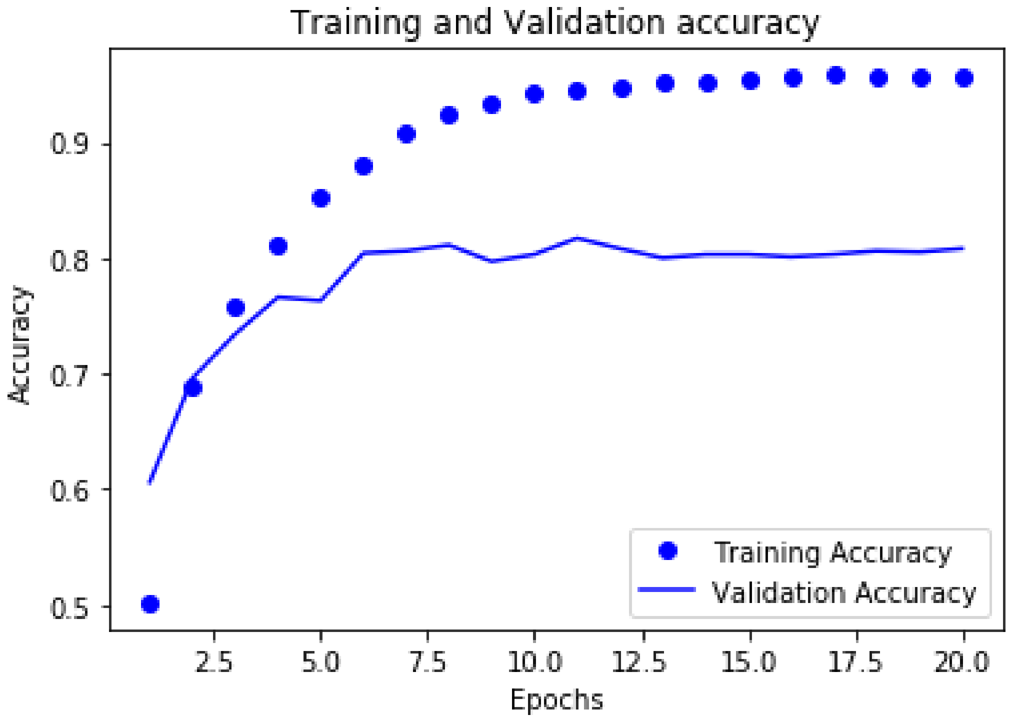


[23]:
```
acc = history.history['acc']
val_acc = history.history['val_acc']

epochs = range(1, len(acc) + 1)

plt.plot(epochs, acc, 'bo', label='Training Accuracy')
plt.plot(epochs, val_acc, 'b', label="Validation Accuracy")
plt.title('Training and Validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.show()
```



从图中可以看出，从第九轮过后开始过拟合了。重新一次9轮的训练：

[24]:

```
history = 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)
```

```
Train on 7982 samples, validate on 1000 samples
Epoch 1/9
7982/7982 [==============================] - 1s 123us/step - loss: 0.1066 - acc:
0.9589 - val_loss: 1.0775 - val_acc: 0.8110
Epoch 2/9
7982/7982 [==============================] - 1s 116us/step - loss: 0.1045 - acc:
0.9573 - val_loss: 1.0975 - val_acc: 0.8040
Epoch 3/9
7982/7982 [==============================] - 1s 116us/step - loss: 0.1020 - acc:
0.9585 - val_loss: 1.1302 - val_acc: 0.7950
Epoch 4/9
7982/7982 [==============================] - 1s 124us/step - loss: 0.0986 - acc:
0.9594 - val_loss: 1.1511 - val_acc: 0.7960
Epoch 5/9
7982/7982 [==============================] - 1s 121us/step - loss: 0.1000 - acc:
```

```
0.9583 - val_loss: 1.1416 - val_acc: 0.7980
Epoch 6/9
7982/7982 [==============================] - 1s 116us/step - loss: 0.0962 - acc:
0.9580 - val_loss: 1.1762 - val_acc: 0.7960
Epoch 7/9
7982/7982 [==============================] - 1s 112us/step - loss: 0.0985 - acc:
0.9573 - val_loss: 1.1604 - val_acc: 0.8000
Epoch 8/9
7982/7982 [==============================] - 1s 111us/step - loss: 0.0939 - acc:
0.9590 - val_loss: 1.2325 - val_acc: 0.7850
Epoch 9/9
7982/7982 [==============================] - 1s 117us/step - loss: 0.0936 - acc:
0.9585 - val_loss: 1.2479 - val_acc: 0.7810
2246/2246 [==============================] - 0s 132us/step
```

[25]:

```
results
```

[25]:

```
[1.3896791956197547, 0.7693677649684815]
```

9轮的训练结果，在测试集上得到了76%的精确度，总体上尚可。

- Bones：完全随机的分类算法代码：

[28]:

```
import copy
test_labels_copy=copy.copy(test_labels)
np.random.shuffle(test_labels_copy)
hits_array = np.array(test_labels) == np.array(test_labels_copy)
float(np.sum(hits_array)) / len(test_labels)
```

[28]:

```
0.188780053428317
```

完全随机的算法得到的正确率大概是18%。