In [1]:
```
#加载IMDB数据集
from keras.datasets import imdb
(train_data, train_labels), (test_data, test_labels) = imdb.load_data(num_words=10000)
```

```
Using TensorFlow backend.
```

In [2]:
```
train_data[0]
```

Out[2]:
```
[1,
 14,
 22,
 16,
 43,
 530,
 973,
 1622,
 1385,
 65,
 458,
 4468,
 66,
 3941,
 4,
 173,
 36,
 256,
 5,
```

In [3]:
```
train_labels
```

Out[3]:
```
array([1, 0, 0, ..., 0, 1, 0])
```

In [4]:
```
max([max(sequence) for sequence in train_data])
```

Out[4]:
```
9999
```

In [5]:
```
#索引解码成单词
word_index = imdb.get_word_index()
reverse_word_index = dict(
    [(value, key) for (key, value) in word_index.items()])
decoded_review = ' '.join(
    [reverse_word_index.get(i - 3, '?') for i in train_data[0]])  #第一个评论
```

In [6]:
```
decoded_review
```

Out[6]:
```
"? this film was just brilliant casting location scenery story direction everyone's really s
uited the part they played and you could just imagine being there robert ? is an amazi
ng actor and now the same being director ? father came from the same scottish islan
d as myself so i loved the fact there was a real connection with this film the witty rem
arks throughout the film were great it was just brilliant so much that i bought the film
as soon as it was released for ? and would recommend it to everyone to watch and th
e fly fishing was amazing really cried at the end it was so sad and you know what they
say if you cry at a film it must have been good and this definitely was also ? to the tw
o little boy's that played the ? of norman and paul they were just brilliant children are
often left out of the ? list i think because the stars that play them all grown up are suc
h a big profile for the whole film but these children are amazing and should be praised
 for what they have done don't you think the whole story was so lovely because it was
true and was someone's life after all that was shared with us all"
```

In [7]:
```
#数据向量化
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)
```

In [8]:
```
x_train[0]
```

Out[8]:
```
array([0., 1., 1., ..., 0., 0., 0.])
```

In [9]:
```
#标签向量化
y_train = np.asarray(train_labels).astype('float32')
y_test = np.asarray(test_labels).astype('float32')
```

In [10]:
```
#构建网络 模型定义
#激活函数relu 所有负值归0
#任意值“压缩”到[0, 1]区间内
from keras import models
from keras import layers
model = models.Sequential()
model.add(layers.Dense(16, activation='relu', input_shape=(10000,)))
model.add(layers.Dense(16, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
#relu 负值归零 sigmoid 任意值压缩到[0, 1]区间内
```

In [11]:
```
#损失函数 优化器 指标 编译模型
#
model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['accuracy'])
```

In [12]:
```
#从训练数据中留出验证集
x_val = x_train[:10000]
partial_x_train = x_train[10000:]

y_val = y_train[:10000]
partial_y_train = y_train[10000:]
```

In [13]:
```
#训练模型
model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['acc'])
history = model.fit(partial_x_train, partial_y_train, epochs=20, batch_size=512, validatior
```

```
Train on 15000 samples, validate on 10000 samples
Epoch 1/20
15000/15000 [==============================] - 6s 425us/step - loss: 0.5086
- acc: 0.7815 - val_loss: 0.3793 - val_acc: 0.8688
Epoch 2/20
15000/15000 [==============================] - 5s 356us/step - loss: 0.3004
- acc: 0.9049 - val_loss: 0.3002 - val_acc: 0.8897
Epoch 3/20
15000/15000 [==============================] - 6s 369us/step - loss: 0.2179
- acc: 0.9283 - val_loss: 0.3083 - val_acc: 0.8715
Epoch 4/20
15000/15000 [==============================] - 5s 348us/step - loss: 0.1751
- acc: 0.9437 - val_loss: 0.2839 - val_acc: 0.8838
Epoch 5/20
15000/15000 [==============================] - 4s 233us/step - loss: 0.1425
- acc: 0.9542 - val_loss: 0.2843 - val_acc: 0.8869
Epoch 6/20
15000/15000 [==============================] - 3s 229us/step - loss: 0.1150
- acc: 0.9651 - val_loss: 0.3157 - val_acc: 0.8768
Epoch 7/20
15000/15000 [==============================] - 4s 248us/step - loss: 0.0979
- acc: 0.9707 - val_loss: 0.3129 - val_acc: 0.8844
Epoch 8/20
15000/15000 [==============================] - 3s 222us/step - loss: 0.0806
- acc: 0.9765 - val_loss: 0.3859 - val_acc: 0.8655
Epoch 9/20
15000/15000 [==============================] - 4s 246us/step - loss: 0.0659
- acc: 0.9821 - val_loss: 0.3635 - val_acc: 0.8781
Epoch 10/20
15000/15000 [==============================] - 4s 234us/step - loss: 0.0555
- acc: 0.9853 - val_loss: 0.3844 - val_acc: 0.8791
Epoch 11/20
15000/15000 [==============================] - 3s 215us/step - loss: 0.0450
- acc: 0.9889 - val_loss: 0.4167 - val_acc: 0.8767
Epoch 12/20
15000/15000 [==============================] - 4s 256us/step - loss: 0.0384
- acc: 0.9913 - val_loss: 0.4505 - val_acc: 0.8697
Epoch 13/20
15000/15000 [==============================] - 4s 270us/step - loss: 0.0297
- acc: 0.9929 - val_loss: 0.4701 - val_acc: 0.8731
Epoch 14/20
```

```
15000/15000 [==============================] - 5s 305us/step - loss: 0.0243
- acc: 0.9949 - val_loss: 0.5029 - val_acc: 0.8717
Epoch 15/20
15000/15000 [==============================] - 5s 333us/step - loss: 0.0176
- acc: 0.9979 - val_loss: 0.5375 - val_acc: 0.8691
Epoch 16/20
15000/15000 [==============================] - 5s 313us/step - loss: 0.0170
- acc: 0.9968 - val_loss: 0.5730 - val_acc: 0.8697
Epoch 17/20
15000/15000 [==============================] - 4s 247us/step - loss: 0.0093
- acc: 0.9995 - val_loss: 0.6158 - val_acc: 0.8651
Epoch 18/20
15000/15000 [==============================] - 3s 228us/step - loss: 0.0117
- acc: 0.9975 - val_loss: 0.6384 - val_acc: 0.8667
Epoch 19/20
15000/15000 [==============================] - 4s 300us/step - loss: 0.0061
- acc: 0.9996 - val_loss: 0.7757 - val_acc: 0.8517
Epoch 20/20
15000/15000 [==============================] - 5s 353us/step - loss: 0.0045
- acc: 0.9999 - val_loss: 0.7039 - val_acc: 0.8652
```

In [14]:

```python
#绘制训练损失和验证损失
import matplotlib.pyplot as plt

history_dict = history.history
loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']

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

plt.plot(epochs, loss_values, 'bo', label='Training loss')
plt.plot(epochs, val_loss_values, 'b', label='Valifation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
```

```
<Figure size 640x480 with 1 Axes>
```

In [15]:

```python
history_dict.keys()
```

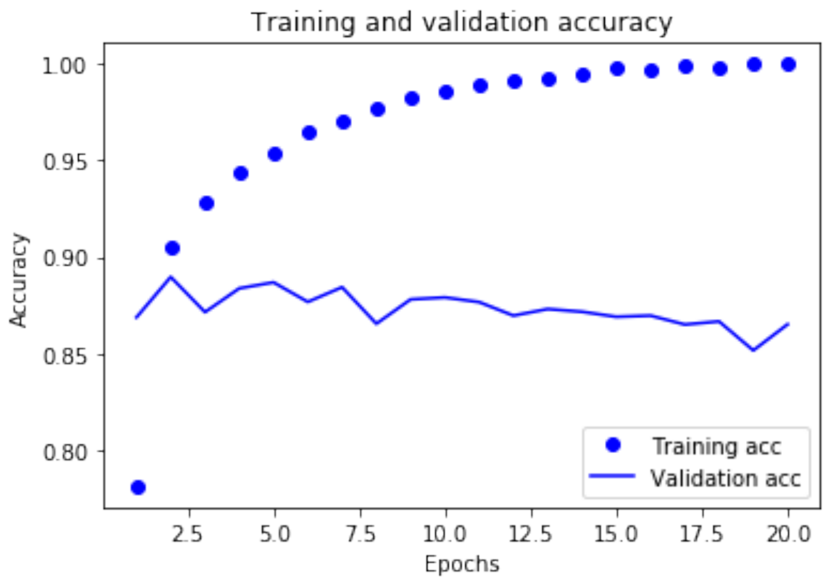
Out[15]:

```
dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
```

In [16]:
```
#绘制训练精度和验证精度
plt.clf()
acc = history_dict['acc']
val_acc = history_dict['val_acc']

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()
```



In [17]:
```
results = model.evaluate(x_test, y_test)
results
```

```
25000/25000 [==============================] - 7s 271us/step
```

Out[17]:
```
[0.7766584446382523, 0.85068]
```

In [18]:
```
#预测评价正面的可能性
model.predict(x_test)
```

Out[18]:
```
array([[0.00921098],
       [0.9999999 ],
       [0.95997983],
       ...,
       [0.001115  ],
       [0.00480793],
       [0.6486915 ]], dtype=float32)
```