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

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb.npz (https://storage.go
ogleapis.com/tensorflow/tf-keras-datasets/imdb.npz)
17464789/17464789 [==============================] - 5s 0us/step
```

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[0]
```

Out[3]:
```
1
```

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

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

In [5]:
```
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]])
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb_word_index.json (https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb_word_index.json)
1641221/1641221 [==============================] - 0s 0us/step
```

In [6]:
```
import numpy as np
```

In [7]:
```
def vectorize_sequences(sequences, dimension=10000):
    results = np.zeros((len(sequences), dimension))
    for i, sequence in enumerate(sequences):
        results[i, sequence] = 1.
    return results
```

In [8]:
```
x_train = vectorize_sequences(train_data)
x_test = vectorize_sequences(test_data)
```

In [9]:
```
x_train[0]
```

Out[9]:
```
array([0., 1., 1., ..., 0., 0., 0.])
```

In [10]:
```
y_train = np.asarray(train_labels).astype('float32')
y_test = np.asarray(test_labels).astype('float32')
```

In [11]:
```
from keras import models
from keras import layers
```

In [12]:
```
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'))
```

In [13]:
```
model.compile(optimizer='rmsprop',
    loss='binary_crossentropy',
    metrics=['accuracy'])
```

In [14]:
```
from keras import optimizers
```

In [15]:
```
model.compile(optimizer=optimizers.RMSprop(lr=0.001),
    loss='binary_crossentropy',
    metrics=['accuracy'])
```

```
C:\Users\theoj\AppData\Roaming\Python\Python39\site-packages\keras\optimizers\optimizer_v2\rmsprop.py:140: Use
rWarning: The `lr` argument is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)
```

In [16]:
```
from keras import losses
from keras import metrics
```

In [17]:
```
model.compile(optimizer=optimizers.RMSprop(lr=0.001),
    loss=losses.binary_crossentropy,
    metrics=[metrics.binary_accuracy])
```

In [18]:
```
x_val = x_train[:10000]
partial_x_train = x_train[10000:]
```

In [19]:
```
y_val = y_train[:10000]
partial_y_train = y_train[10000:]
```

In [20]:
```
model.compile(optimizer='rmsprop',
    loss='binary_crossentropy',
    metrics=['acc'])
```

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

```
Epoch 1/20
30/30 [==============================] - 7s 61ms/step - loss: 0.5170 - acc: 0.7824 - val_loss: 0.3844 - val_ac
c: 0.8736
Epoch 2/20
30/30 [==============================] - 1s 19ms/step - loss: 0.3058 - acc: 0.9041 - val_loss: 0.3102 - val_ac
c: 0.8844
Epoch 3/20
30/30 [==============================] - 1s 17ms/step - loss: 0.2211 - acc: 0.9308 - val_loss: 0.2873 - val_ac
c: 0.8860
Epoch 4/20
30/30 [==============================] - 1s 19ms/step - loss: 0.1768 - acc: 0.9421 - val_loss: 0.2900 - val_ac
c: 0.8844
Epoch 5/20
30/30 [==============================] - 0s 16ms/step - loss: 0.1424 - acc: 0.9544 - val_loss: 0.2829 - val_ac
c: 0.8849
Epoch 6/20
30/30 [==============================] - 0s 16ms/step - loss: 0.1179 - acc: 0.9645 - val_loss: 0.2954 - val_ac
c: 0.8856
Epoch 7/20
30/30 [==============================] - 0s 16ms/step - loss: 0.0964 - acc: 0.9716 - val_loss: 0.3130 - val_ac
c: 0.8842
Epoch 8/20
30/30 [==============================] - 0s 16ms/step - loss: 0.0819 - acc: 0.9753 - val_loss: 0.3871 - val_ac
c: 0.8705
Epoch 9/20
30/30 [==============================] - 0s 16ms/step - loss: 0.0651 - acc: 0.9841 - val_loss: 0.3614 - val_ac
c: 0.8743
Epoch 10/20
30/30 [==============================] - 0s 16ms/step - loss: 0.0540 - acc: 0.9866 - val_loss: 0.3782 - val_ac
c: 0.8780
Epoch 11/20
30/30 [==============================] - 0s 16ms/step - loss: 0.0408 - acc: 0.9907 - val_loss: 0.4646 - val_ac
c: 0.8601
Epoch 12/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0341 - acc: 0.9929 - val_loss: 0.4516 - val_ac
c: 0.8755
```

```
Epoch 13/20
30/30 [==============================] - 0s 16ms/step - loss: 0.0277 - acc: 0.9946 - val_loss: 0.4781 - val_ac
c: 0.8700
Epoch 14/20
30/30 [==============================] - 0s 16ms/step - loss: 0.0201 - acc: 0.9970 - val_loss: 0.5099 - val_ac
c: 0.8699
Epoch 15/20
30/30 [==============================] - 0s 16ms/step - loss: 0.0190 - acc: 0.9962 - val_loss: 0.5499 - val_ac
c: 0.8720
Epoch 16/20
30/30 [==============================] - 0s 16ms/step - loss: 0.0115 - acc: 0.9989 - val_loss: 0.5835 - val_ac
c: 0.8693
Epoch 17/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0106 - acc: 0.9986 - val_loss: 0.6158 - val_ac
c: 0.8647
Epoch 18/20
30/30 [==============================] - 0s 16ms/step - loss: 0.0090 - acc: 0.9986 - val_loss: 0.6483 - val_ac
c: 0.8685
Epoch 19/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0042 - acc: 0.9999 - val_loss: 0.6772 - val_ac
c: 0.8672
Epoch 20/20
30/30 [==============================] - 0s 14ms/step - loss: 0.0082 - acc: 0.9981 - val_loss: 0.7165 - val_ac
c: 0.8647
```

In [22]:
```
history_dict = history.history
```

In [23]:
```
 history_dict.keys()
```

Out[23]:
```
dict_keys(['loss', 'acc', 'val_loss', 'val_acc'])
```

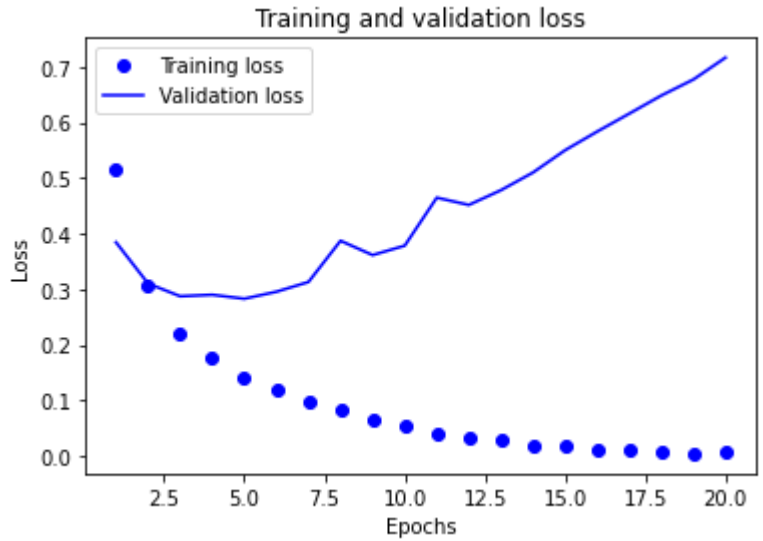
In [24]:
```
import matplotlib.pyplot as plt
```

In [25]:
```
history_dict = history.history
loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']
acc_values = history_dict['acc']
```

In [26]:
```
epochs = range(1, len(acc_values) + 1)
```

In [27]:
```
plt.plot(epochs, loss_values, 'bo', label='Training loss')
plt.plot(epochs, val_loss_values, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
```

Out[27]:
```
<matplotlib.legend.Legend at 0x1fb2531a520>
```



In [28]:
```
plt.show()
```

In [29]:
```
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'))
```

In [30]:
```
model.compile(optimizer='rmsprop',
    loss='binary_crossentropy',
    metrics=['accuracy'])
model.fit(x_train, y_train, epochs=4, batch_size=512)
results = model.evaluate(x_test, y_test)
```

```
Epoch 1/4
49/49 [==============================] - 1s 10ms/step - loss: 0.4514 - accuracy: 0.8227
Epoch 2/4
49/49 [==============================] - 0s 10ms/step - loss: 0.2616 - accuracy: 0.9081
Epoch 3/4
49/49 [==============================] - 0s 10ms/step - loss: 0.2031 - accuracy: 0.9272
Epoch 4/4
49/49 [==============================] - 0s 10ms/step - loss: 0.1662 - accuracy: 0.9428
782/782 [==============================] - 2s 3ms/step - loss: 0.2921 - accuracy: 0.8848
```

In [31]:
```
results
```

Out[31]:
```
[0.2920589745044708, 0.8848000168800354]
```

In [32]:
```
model.predict(x_test)
```

```
782/782 [==============================] - 1s 2ms/step
```

Out[32]:
```
array([[0.18512133],
       [0.99988806],
       [0.9159456 ],
       ...,
       [0.08447732],
       [0.09145257],
       [0.5668796 ]], dtype=float32)
```

In [ ]:
```

```