網路爬蟲 BeautifulSoup4

15 CHAPTER

15.1 網路爬蟲－取得網路文章內容

如果對抓取網頁上面的資料感興趣的話，可以透過 python 的 BeautifulSoup4 工具，只要是能在瀏覽器看到的文字，幾乎都可以透過 BeautifulSoup4 取得，並且應用在程式中。

Python 需要第三方的函式庫 BeautifulSoup4，可透過以下的方法安裝：

```
pip install requests
pip install BeautifulSoup4
```

接下來建立第一個爬蟲的程式。請使用以下的程式，將柯老師的網站內容下載並抓取，並且將網站內容的標題和連接抓取並顯示出來。

範例程式：01-BeautifulSoup-getTitle.py

```
import requests                                  # 匯入網路函式庫
from bs4 import BeautifulSoup                    # 匯入 BeautifulSoup 函式庫
req=requests.get('http://www.powenko.com/wordpress/')  # 取得網頁資料
soup=BeautifulSoup(req.text.encode('utf-8'), "html.parser")  # 轉換 UTF-8
print(soup.title)                                # 取得網頁標題
print(soup.title.string)                         # 取得網頁標題內的文字
print(soup.p)                                    # 取得第一個 HTML 中 p 的內容
print(soup.a)                                    # 取得第一個 HTML 中 a 的內容
print(soup.find_all('a'))                        # 取得所有 HTML 的 a
```

局部執行結果

```
powens-MacBook-Air:ch15 powenko$ python 01-BeautifulSoup-getTitle.py
</title>柯博文老師  - LoopTek.com 錄克軟體 CT0</title>
柯博文老師 – LoopTek.com 錄克軟體 CT0
<p>課程報名和相關資料請洽：</p>
<a href="http://www.powenko.com/wordpress" title="柯博文老師">
<img alt="柯博文老師" class="normal_logo" src="http://www.powenko.
com/wordpress/wp-content/uploads/2018/03/logo.png" title="柯博文老
師"/>
<img alt="柯博文老師" class="retina_logo" src="http://www.powenko.
com/wordpress/wp-content/uploads/2018/03/logo.png" style="width:
98px; height:99px;" title="柯博文老師"/>
</a>
[<a href="http://www.powenko.com/wordpress"
title="\u67ef\u535a\u6587\u8001\u5e2b">\n<img alt="\
u67ef\u535a\u6587\u8001\u5e2b" ....省略
```

教學影片 01-BeautifulSoup-getTitle.mp4

15.2 BeautifulSoup 的函數和屬性

在剛剛的程式中，是透過屬性將網站的內容加以取得和分析，而以下的是 BeautifulSoup 常用的屬性與函數：

title

取得網站的抬頭，包含 HTML 標籤。使用範例：

```
soup.title
```

輸出為 <title>柯博文老師 – LoopTek.com 錄克軟體 </title>。

name

取得 HTML 標籤，輸出為 u'title'。使用範例：

```
soup.title.name
```

輸出為 u'title'。

string

取得的網站的內容。使用範例：

```
soup.title.string
```

輸出為 u '柯博文老師 – LoopTek.com 錄克軟體'。

parent

取得該節點 HTML 的父類。使用範例：

```
soup.title.parent.name
```

輸出為 u'head'。

這裡的 u 的意思是 Unicode 的意思。

p

取得該網站的第一個 HTML 標籤的 p。使用範例：

```
soup.p
```

輸出為 The test 。

p['class']

取得該網站的第一個 HTML 標籤的 p（跳行）之 class，使用範例：

```
soup.p['class']
```

如果第一個 p 的內容為 The test ，則輸出為 [u'title']。

a

取得該網站的第一個 HTML 標籤的 a（連結）。使用範例：

```
soup.a
```

輸出為 test1，

find_all

取得該網站所有的 HTML 標籤。

範例一：如果要找所有 a（連結）的使用範例：

```
soup.find_all('a')
```

輸出的 List 陣列為：

```
[<a class="redcolor" href="http://powenko.com/1.html" id="link1">test1</a>,
<a class="bluecolor" href="http://powenko.com/2.html" id="link2">test2</a>,
<a class="redcolor" id="link3" href="http://powenko.com/3.html" id="link3">test3</a>]
```

get

取得該網站所有的 HTML 標籤內的屬性。

範例：如果要找所有 a （連結）的 URL 使用範例：

```
for link in soup.find_all('a'):
     print(link.get('href'))
```

輸出的 List 陣列的內容為：

```
http://powenko.com/1.html
http://powenko.com/2.html
http://powenko.com/3.html
```

select

取得該網站所有的 HTML 標籤、class 和 id。

範例一：如果要找所有 a （連結）的使用範例：

```
soup.select('a')
```

輸出的 List 陣列為：

```
[<a class="redcolor" href="http://powenko.com/1.html" id="link1">test1</a>,
<a class="bluecolor" href="http://powenko.com/2.html" id="link2">test2</a>,
<a class="redcolor" id="link3" href="http://powenko.com/3.html" id="link3">test3</a>]
```

範例二：如果要找取得該網站所有 class="redcolor" 的使用範例：

```
soup.select('.redcolor')
```

輸出的 List 陣列為：

```
[<a class="redcolor" href="http://powenko.com/1.html" id="link1">test1</a>,
<a class="redcolor" id="link3" href="http://powenko.com/3.html" id="link3">test3</a>]
```

範例三：如果要找取得該網站所有 id="link3" 的使用範例：

```
soup.select('#link3')
```

輸出的 List 陣列為：

```
[<a class="redcolor" id="link3" href="http://powenko.com/3.html" id="link3">test3</a>]
```

範例四：如果要找所有 a （連結）的文字使用範例：

```
for link in soup.select('a'):
        print(link.string)
```

輸出的 List 陣列為：

```
test1
test2
test3
```

contents

取得該內容中，依照 HTML 標籤所切割的內容。

範例：如果 HTML 內容如下：

```
<a href="..."> 文字一 <br>文字二</a>
```

如果取得「文字二」的文字資料，則可以使用範例：

```
soup.select('a')[0].contents[1]
```

接下來，透過以下的範例程式來練習 BeautifulSoup 的函數和屬性之使用方法。

範例程式：02-BeautifulSoup-API.py

```
import requests                              # 匯入網路函式庫
from bs4 import BeautifulSoup                # 匯入 BeautifulSoup 函式庫
text1="""                                    # 範例 HTML 內容
<head>
    <title>柯博文老師</title>
</head>
<body>
    <p class="title"><b>The test</b></p>
    <a class="redcolor" href="http://powenko.com/1.html" id="link1">test1
       </a>
    <a class="bluecolor" href="http://powenko.com/2.html" id="link2">test2
       </a>
    <a class="redcolor" id="link3" href="http://powenko.com/3.html" id=
       "link3">test3</a>
</body>
"""
soup=BeautifulSoup(text1, "html.parser")  # 轉換
print(soup.title)                         # 取得網頁標題
print(soup.title.name)
print(soup.title.string)                  # 取得網頁標題內的文字
```

```
18. print(soup.title.parent.name)
19. print(soup.title.parent.name)
20. print(soup.p)                          # 取得第一個 HTML 中 p 的內容
21. print(soup.p['class'])
22. print(soup.a)                          # 取得第一個 HTML 中 a 的內容
23. print(soup.find_all('a'))              # 取得所有 HTML 的 a
24. for link in soup.find_all('a'):        # 所有 a（連結）的 URL 網址
25.     print(link.get('href'))
26. print(soup.select('a'))                # 所有 a（連結）
27. print(soup.select('.redcolor'))        # 所有 class="redcolor"
28. print(soup.select('#link3'))           # 所有 id="link3"
29. for link in soup.select('a'):          # 所有 a（連結）的文字
30.     print(link.string)
```

執行結果

```
powens-MacBook-Air:ch15 powenko$ python 02-BeautifulSoup-API.py
<title>柯博文老師</title>
title
柯博文老師
head
<p class="title"><b>The test</b></p>
[u'title']
<a class"redcolor" href="http://powenko.com/1.html" id="link1">test1</a>
[<a class="redcolor" href="http://powenko.com/1.html" id="link1">
test1</a>, <a class="bluecolor" href="http://powenko.com/2.html"
id="link2">test2</a>, <a class="redcolor" href="http://powenko.
com/3.html" id="link3">test3</a>]
http://powenko.com/1.html
http://powenko.com/2.html
http://powenko.com/3.html
[<a class="redcolor" href="http://powenko.com/1.html" id="link1">
test1</a>, <a class="bluecolor" href="http://powenko.com/2.html"
id="link2">test2</a>, <a class="redcolor" href="http://powenko.
com/3.html" id="link3">test3</a>]
[<a class="redcolor" href="http://powenko.com/1.html" id="link1">
test1</a>, <a class="redcolor" href="http://powenko.com/3.html"
id="link3">test3</a>]
[<a class="redcolor" href="http://powenko.com/3.html" id="link3">
test3</a>]
test1
test2
test2
```

教學影片 02-BeautifulSoup-API.mp4

15.3 實戰案例－抓取作者的部落格文章

在這裡將使用柯老師自己的網站做個實際範例，介紹如何讀取該網站的內容標題。讓讀者瞭解相關技巧，再運用到其他的商業網站，如新聞網站和購物網站等。因為網站調整在所難免，如果該網站調整版面後，這個 Python 的爬蟲程式也就需要跟著調整，還請理解。

STEP 1 google chrome 工具

分析網站的技巧，第一個步驟是挑選一個你熟悉或者好用的網頁工具來觀看網站的原始程式和 HTML 結構。在這裡我使用的是 google chrome 這個軟體，請先下載和打開這個軟體，並且輸入 http://www.powenko.com/wordpress 來觀看。

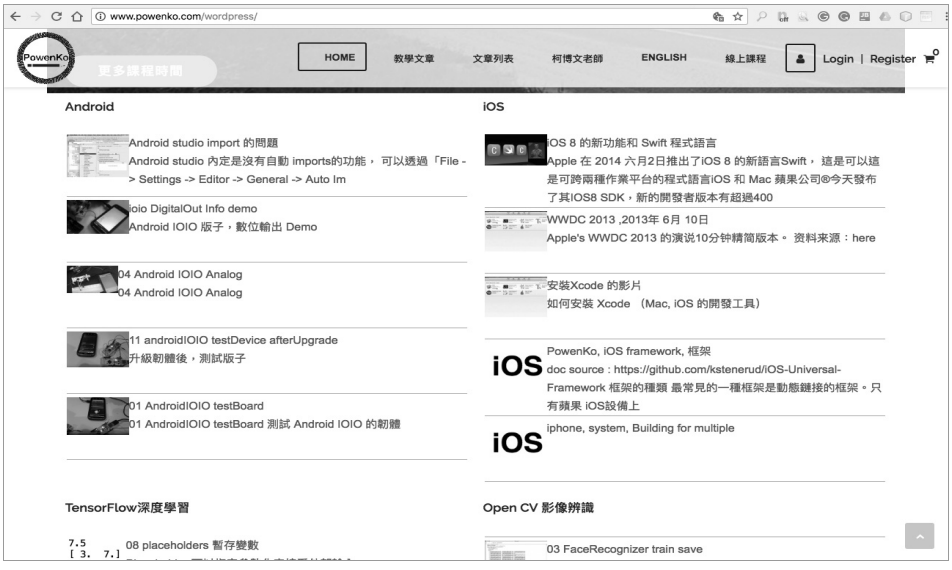

圖 15-1 柯博文老師的網頁

STEP 2 打開 google chrome 開發者工具

接下來請點選 Settings\More Tools\Developer Tools 來打開程式開發者工具。

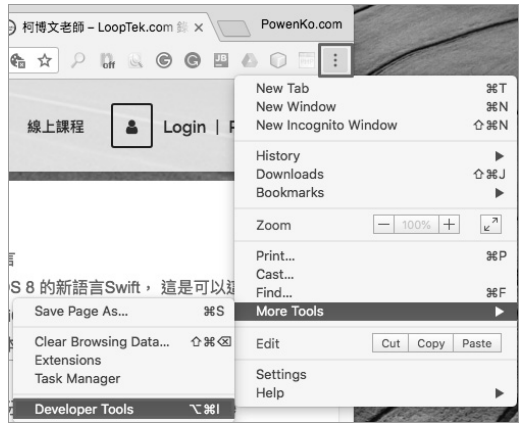

圖 15-2 開啓程式開發者工具

STEP 3 取得要抓取的文章原始程式

1. 接下來請點選【Elements 元素】和【方塊】。
2. 透過滑鼠的移動，就能在網頁上面看到所對應的原始程式碼。這邊請你先把滑鼠中移動到 Android 標題下面的文章，就會看到該 HTML 的原始程式碼與彼此標題之間的關係。

圖 15-3 開啓程式開發者工具

STEP 4 分析網頁

因為等一下要透過程式來抓取網站的 Android 新聞標題，所以必須要了解網站的架構，以便等一下撰寫程式時，可以精準的取得關鍵字。所以請用上一步的技巧來觀察標題，看看是否有名稱較為獨特的 class 、id 或 HTML，可以讓程式很快抓到的標題。

圖 15-4 分析原始網頁程式

柯博文老師的網站使用了很多 div 這樣子的標籤，而如果要抓取 Android 新聞文章的標題，文字放在 area 下的 a 內容，而這 Android 區域是放在 largefeaturepowenA2 的 div 中，其關係如下：

```
<div class="largefeaturepowenA2" …..>
     <div class="area" ...>
               <div style="...">
                     <a ….></a>
               </div>
               <a href="...">
                       Android studio import 的問題<br>
                     <font>Android studio 內定是沒有自動 imports 的功能，可以
                     透過</font>
```

```
                </a>
        </div>
        <hr style=...">
        <div class="area" …>
                <div style="...">
                        <a ….></a>
                </div>
                <a href="...">
                        ioio DigitalOut Info demo<br>....
                         <font> ...</font>
                </a>
         </div>
         <hr style=...">
          ……
</div>
```

STEP 5 瞭解網頁架構

所以了解網站和要抓取的目標之後，在程式中抓取首先必須：

1. 找到 class="largefeaturepowenA2" 這一個 div 標籤。
2. 因為該標籤在網站中還用在其他的地方，以 Android 標題來看，是第一個使用該標籤。
3. 然後，在裡面篩選和抓取所有的 class="area" 就可以取得您要的區域了。
4. 但是，它裡面還有一些圖片，所以需要再設定標籤 a（連結）中的第二個。
5. 然而，該區域中還包含大標題和內容，因為只要大標題，所以透過 contents[0] 來抓取該區域第一個 HTML 的段落。

完整程式如下。

範例程式：03-powenko.py

```
1.   import requests                                  # 匯入網路函式庫
2.   from bs4 import BeautifulSoup                    # 匯入爬蟲函式庫
3.   req=requests.get("http://www.powenko.com/wordpress")  # 取得網頁資料
```

```
4.   soup=BeautifulSoup(req.text.encode('utf-8'), "html.parser")  # 轉換 UTF-8
5.   largefeaturepowenA2=soup.select('.largefeaturepowenA2')
                                                  # 所有 class="largefea.."
6.   largefeature0=largefeaturepowenA2[0]          # 取得第一個
7.   for area in largefeature0.select('.area'):    # 所有 class="area"
8.     print(area.select('a')[1].contents[0])      # 標籤 a 第二個中的第一個段落
```

執行結果

```
powens-MacBook-Air:ch15 powenko$ python 03-powenko.py
                                Android studio import 的問題
                                ioio DigitalOut Info demo
                                04 Android IOIO Analog
                                11 androidIOIO testDevice afterUpgrade
                                01 AndridIOIO testBoard
```

教學影片 03-powenko.py.mp4

15.4 實戰練習

網頁爬文的技巧會因為不同的網頁內容格式而影響到程式的撰寫方法，所以最好的方法就多找幾個網站，實際多練習。透過實戰的方式，這樣才會提升網頁爬文的技巧。

另外，為了讓各位多練習，請示試試看以下的網站，而實際的案例請參考範例程式：

- ✓ 百度科技新聞標題。範例程式請參考 04-Baidu.py。
- ✓ Feebee 比價網站針對樹莓派硬體，顯示出每一個網站的售價。範例程式請參考 05-feebee.py。
- ✓ 中國時報新聞標題。範例程式請參考 06-chinesetimes.py

提醒一下，範例程式會因為網站的內容變化，會需要自行調整。

教學影片 04-powenko-Baidu.mp4

TensorFlow 改善類神經模型 MLP 結果

27 CHAPTER

27.1 模型 model 不同的寫法

而 Tensorflow 的 MLP 模組有不同的寫法，但是答案都一樣，緊接上一章原本的建立模型程式如下。

```
# 建立模型和訓練
model = tf.keras.models.Sequential([              # 建立類神經的模型
     tf.keras.layers.Dense(units=10, activation=tf.nn.relu, input_dim=1),
     # 10 個神經點，輸入一維陣列
     tf.keras.layers.Dense(units=10, activation=tf.nn.relu),  # 10 個神經元，使用 relu
     tf.keras.layers.Dense(units=2, activation=tf.nn.softmax) # 2 種答案使用 softmax
])
```

同樣的程式，也可以寫成如下所示。

部分範例程式：01_MLPpredict_TensorFlow_ModelAdd.py

```
# 建立模型
model = tf.keras.models.Sequential()            # 加入順序層
model.add(tf.keras.layers.Dense(units=10,       # 加入 10 個神經元
       activation=tf.nn.relu,
       input_dim=1   ))                         # 一維陣列的因
model.add(tf.keras.layers.Dense(units=10,       # 加入 10 個神經元
```

```
        activation=tf.nn.relu))
model.add(tf.keras.layers.Dense(units=2,        # 加入 2 個神經元輸出的答案
        activation=tf.nn.softmax   ))
```

執行結果

```
Epoch 20/20
 128/1000 [==>.................] - ETA: 0s - loss: 0.6519 - acc: 0.8984
1000/1000 [====================] - 0s 31us/step-loss: 0.6504 - acc: 0.8970
4/4 [======================] - 0s 55ms/step
score: [0.6432062387466431, 1.0]
predict: [[0.52375585 0.47624412]
 [0.5254014 0.47459856]
 [0.47642604 0.52357394]
 [0.47029015 9.52970994]]
predict: 0 0 1 1
y_test [0 0 1 1]
```

結果都是一樣，只是 TensorFlow 官方網站和相關資料的寫法二者都會有，所以這二種程式的寫法都要學會。本書會用後者的寫法，原因是這樣的寫法比較像正統的 Tensorflow.keras 的官方範例。

教學影片 01_MLPpredict_TensorFlow_ModelAdd.mp4

27.2 TensorFlow 與 Keras 函式庫的關係和差異

其實很多人在學習類神經會先學 Keras，然後再學習 TensorFlow。在早期的 TensorFlow 撰寫類神經其實很麻煩，程式碼相當複雜和冗長，但是自從 TensorFlow1.9 版加入 tensorflow.keras 的模組後，TensorFlow 開發類神經就容易多了。本章的範例刻意用 Keras 把剛剛的 TensorFlow 程式翻寫如下，各位可以看到其實沒有多大的差異，只是相關的函數因為類別的關係，繼承關係有所不同。

範例程式：02_MLPpredict_Keras.py

```
1.   import keras                              # 匯入 keras 函數
2.   from keras.models import Sequential       # 匯入 keras 相關函數
3.   from keras.layers import Dense            # 匯入 keras 相關函數
```

```
import numpy as np                          # 匯入 Numpy 函數

# 建立訓練資料
x1=np.random.random((500,1))
x2=np.random.random((500,1))+1
x_train=np.concatenate((x1, x2))            # 產生訓練 1000 個資料的因
y1=np.zeros((500,), dtype=int)
y2=np.ones((500,), dtype=int)
y_train=np.concatenate((y1, y2))            # 產生訓練 1000 個資料的果

# 建立模型和訓練
model = Sequential()
model.add(Dense(units=10, activation='relu', input_dim=1))
                                    # 訓練輸入為一維，每一筆只有一個 feature 特徵值
model.add(Dense(units=10, activation='relu'))   # 10 個神經點，使用 relu
model.add(Dense(units=2, activation='softmax')) # 2 種答案使用 softmax
model.compile(optimizer='adam',             # 編譯使用 adam 最佳化
    loss='sparse_categorical_crossentropy',     # 損失率使用稀疏分類交叉熵
    metrics=['accuracy'])                   # 設定編譯處理使用 adam 最佳化
model.fit(x_train, y_train,                 # 進行訓練的因和果資料
    epochs=20,                              # 設定訓練的次數
    batch_size=128)                         # 設定每次訓練的筆數

# 建立測試資料
x_test=np.array([[0.22],[0.31],[1.22],[1.33]])  # 自訂測試資料的因
y_test=np.array([0,0,1,1])                  # 自訂測試資料的果
score = model.evaluate(x_test, y_test, batch_size=128)  # 計算測試正確率
print("score:",score)                       # 輸出測試正確率

predict = model.predict(x_test)             # 預測
print("predict:",predict)                   # 輸出每一個預測的結果
print("Ans:",np.argmax(predict[0]),np.argmax(predict[1]),np.argmax
(predict[2]),np.argmax(predict[3]))         # 取得預測答案 1
predict2 = model.predict_classes(x_test)    # 取得預測答案 2
print("predict_classes:",predict2)          # 輸出預測答案 2
print("y_test",y_test[:])                   # 實際測試的果
```

執行結果

```
Epoch 20/20

 128/1000 [==>...............] - ETA: 0s - loss: 0.3858 - acc: 0.9531
1000/1000 [===================] - 0s 15us/step - loss: 0.3725 - acc: 0.9400

4/4 [======================]- 0s 22ms/step
score: [0.3468254506587982, 1.0]
predict: [[0.7083343 0.29166564]
 [0.7134547 0.2865453 ]
 [0.32057315 0.67942685]
 [0.27263054.0.7273695 ]]
```

```
Ans: 0.0.1.1
predict_classes: [0 0 1 1]
y_test. [00 1.1]
```

程式的撰寫方法和預測的答案幾乎一樣，唯一在匯入 Keras 時會出現 Using TensorFlow backend，意思是 Keras 實際的計算引擎還是使用 TensorFlow ，所以在 TensorFlow1.9 版之後，建議可以直接學習 Tensorflow，並用另外學習 Keras，目前網路上還有不少很棒的 Keras 範例程式，各位也要瞭解如何在 TensorFlow 和 Keras 轉換和使用。

教學影片 02_MLPpredict_Keras.mp4

27.3 One-hot Encoding 單熱編碼

什麼是分類的答案（果）？許多機器學習算法不能直接對文字結果進行運算，它們要求所有輸入和輸出資料都是數字。本書的經典案例包括：

- ✓ 水果的分類：檸檬為 1，柳丁為 2
- ✓ 鳶尾花的分類：setosa 為 1， virginica 為 2， versicolor 為 3

一般來說會有二種編碼答案：

- ✓ 整數編碼 Integer Encoding
- ✓ 單熱編碼 One-Hot Encoding

整數編碼 Integer Encoding：在此之前都是使用這種的方法分類的答案。

單熱編碼 One-Hot Encoding：對於不存在這種序數關係的分類變量，整數編碼 Integer Encoding 的自然排序可能導致性能不佳或意外結果，例如答案在預測各佔一半的情況發生。在類神經數據會推薦使用這個方法來處理，於速度上有比較好效果，而大多數的 MLP 類神經程式都是使用單熱編碼 One-Hot Encoding 來進行訓練。

在這種情況下，可以將整數編碼 Integer Encoding 轉換為單熱編碼 One-Hot Encoding，也就是將每個答案，給予唯一的二進制變量。

例如水果的分類：整數編碼 Integer Encoding：柳丁是 1 檸檬是 2

轉換為單熱編碼 One-Hot Encoding：柳丁是[1,0] 檸檬是[0,1]

例如鳶尾花的分類： 整數編碼 Integer Encoding：柔滑鳶尾花是 1 弗吉尼亞鳶尾花是 2. 弗吉尼亞鳶尾花是 3

轉換為單熱編碼 One-Hot Encoding：柔滑鳶尾花是[1,0,0] 弗吉尼亞鳶尾花是[0,1,0] 弗吉尼亞鳶尾花是[0,0,1]

而在 Tensorflow 透過 to_categorical 函數，可以將 label 標籤資料轉成單熱編碼 One-Hot Encoding。

```
tf.keras.utils.to_categorical(y,num_classes=None)
```

將整數轉換為二進制類陣列。

參數

- ✓ y：要轉換為矩陣的類向量（從 0 到 num_classes 的整數）。
- ✓ num_classes：類的總數。

範例程式：03_MLPOne-hotEncoding.py

```
1.   import tensorflow as tf                    # 匯入 TensorFlow 函數
1.   import numpy as np                         # 匯入 Numpy 函數
2.   #建立訓練資料
3.   x1=np.random.random((500,1))               # 產生 500 個亂數在 0~1 之間
4.   x2=np.random.random((500,1))+1             # 產生 500 個亂數在 1~2 之間
5.   x_train=np.concatenate((x1, x2))           # 產生訓練 1000 個資料的因
6.   y1=np.zeros((500,), dtype=int)             # 產生 500 個 0
7.   y2=np.ones((500,), dtype=int)              # 產生 500 個 1
8.   y_train=np.concatenate((y1, y2))           # 產生訓練 1000 個資料的果
9.
10.  # 將訓練結果轉為 One-Hot Encoding 單熱編碼
11.  y_train2 = tf.keras.utils.to_categorical(y_train,  num_classes=2)
12.
13.  # 建立模型和訓練
14.  model = tf.keras.models.Sequential([       # 建立類神經的模型
15.       tf.keras.layers.Dense(units=10, activation=tf.nn.relu, input_dim=1),
                                     # 訓練輸入為一維，每一筆只有一個 feature 特徵值
16.       tf.keras.layers.Dense(units=10, activation=tf.nn.relu),   #10 個神經元
17.       tf.keras.layers.Dense(units=2, activation=tf.nn.softmax) # 2 種答案
18.  ])
```

```
19.  model.compile(optimizer='adam',            # 設定編譯處理使用 adam 最佳化
20.      loss=tf.keras.losses.categorical_crossentropy,
                                                # 損失率使用分類交叉熵
21.      metrics=['accuracy'])                  # 設定編譯處理使用 adam 最佳化
22.  model.fit(x_train, y_train2,               # 進行訓練的因和單熱編碼的資料
23.            epochs=20,                       # 設定訓練的次數，也就是機器學習的次數
24.            batch_size=128)                  # 設定每次訓練的筆數
25.
26.  # 建立測試資料
27.  x_test=np.array([[0.22],[0.31],[1.22],[1.33]])  # 自訂測試資料的因
28.  y_test=np.array([0,0,1,1])                 # 自訂測試資料的果
29.
30.  # 將測試結果轉為 One-Hot Encoding 單熱編碼
31.  y_test2 = tf.keras.utils.to_categorical(y_test,  num_classes=2)
32.
33.  score = model.evaluate(x_test, y_test, batch_size=128)  # 計算測試正確率
34.  print("score:",score)                      # 輸出測試正確率
35.
36.  predict = model.predict(x_test)            # 預測
37.  print("predict:",predict)                  # 輸出每一個結果的機率
38.  print("Ans:",np.argmax(predict[0]),np.argmax(predict[1]),np.argmax
     (predict[2]),np.argmax(predict[3]))        # 顯示預測答案 1
39.  predict2 = model.predict_classes(x_test)   # 取得預測答案 2
40.  print("predict_classes:",predict2)         # 輸出預測答案 2
41.  print("y_test",y_test[:])                  # 實際測試的結果
```

執行結果

```
Epoch 20/20
 128/1000 [==>.............] - ETA: Os - loss: 0.5986 - acc: 0.6484
1000/1000 [================] - Os 22us/step - loss: 0.5948 - acc: 0.6550
4/4 [====================] - Os 14ms/step
score: [0.5742355585098267, 1.0]
predict: [[0.5163605  0.4836395 ]
 [0.504104   0.495896 ]
 [0.38526908 0.61473095]
 [0.37151873 0.6284812 ]]
Ans: 0.0.1.1
predict_classes: [0 0 1 1]
y_test. [0.0.1.1]
```

TIPS 這裡如果訓練的結果使用 One-Hot Encoding 單熱編碼，則測試的結果也要用 One-Hot Encoding 單熱編碼，訓練和測試的標籤答案 Label 方法需要一致。

另外，使用 One-Hot Encoding 單熱編碼編譯程式的 loss 參數會有所不同，由原本的

```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

換成

```
model.compile(optimizer='adam',
              loss=tf.keras.losses.categorical_crossentropy,
              metrics=['accuracy'])
```

教學影片 03_MLPOne-hotEncoding.mp4

27.4 處理多個特徵值

接下來的章節將透過程式，把類神經 MLP 的預測結果調整到更好，並改善其正確性。為了完成這件事情，首先，先寫一個 CreateDatasets 函數來產生較為複雜的訓練和測試的資料，其程式如下：

```
1.  def CreateDatasets(high,iNum,iArraySize):        # 數據產生器
2.      x_train = np.random.random((iNum, iArraySize)) * float(high)
        # 訓練數據的特徵
3.      y_train = ((x_train[:iNum,0]+x_train[:iNum,1])/2).astype(int)
        # 取整數為訓練數據的標籤答案
4.      return x_train, y_train,tf.keras.utils.to_categorical(y_train,
        num_classes=(high))
5.  category=4
6.  dim=2
7.  x_train,y_train,y_train2=CreateDatasets(category,1000,dim)
    # 建立全部 1000 個二維的訓練數據的特徵，而訓練數據的標籤答案有 4 種答案
```

執行結果

```
category = {int} 4
dim = {int} 2
x_train = {ndarray} [[2.57060333 1.89721391]\n [2.61870255 0.56253016]\n [3.96898919 2.75601175]\n ...
y_train = {ndarray} [2 1 3 2 2 0 2 2 1 2 2 1 2 3 0 2 3 1 2 1 2 1 1 1 1 2 1 1 2 2 1 1 1 2 0 2 1\n 0 2 2 3 0 1 1 3 1 1
y_train2 = {ndarray} [[0. 0. 1. 0.]\n [0. 1. 0. 0.]\n [0. 0. 0. 1.]\n ...\n [0. 0. 1. 0.]\n [0. 0. 0. 1.]\n [0. 1. 0. 0.]]
```

本函數將會產生數據集，而程式的內容就是產生 1000 個訓練值，而訓練數據的特徵因為透過亂數產生的 1000 筆資料。每個資料有 2 個，其數值會在 0~9.999... 之間，而訓練結果 Y，也就是數學公式 y= (x[0]+x[1])/2 的整數值，所以答案分別會是 0 到 9 的整數。

- ✓ x_train：建立全部 1000 個二維的訓練數據的特徵
- ✓ y_train：訓練數據的標籤答案有 10 種答案
- ✓ y_train2：訓練數據的標籤答案的單熱編碼 One-Hot Encoding

範例程式：04_MLPOne-20InputOutput.py

```
import tensorflow as tf
import numpy as np
def CreateDatasets(high,iNum,iArraySize): # 數據產生器
        x_train = np.random.random((iNum, iArraySize)) * float(high)
                                          # 特徵值 X 因
        y_train = ((x_train[:iNum,0]+x_train[:iNum,1])/2).astype(int)
                                          # 取整數為結果 y
return x_train, y_train,tf.keras.utils.to_categorical(y_train,
num_classes=(high))
category=10                               # 幾種答案
dim=2                                     # 每筆資料有 2 個特徵值 X 因
x_train,y_train,y_train2=CreateDatasets(category,1000,dim)
# 建立全部 1000 個二維的訓練特徵值 X 因，而訓練結果 Y 有 10 種答案
# 建立模型
model = tf.keras.models.Sequential()      # 建立類神經的模型
model.add(tf.keras.layers.Dense(units=10, # 10 個神經元
    activation=tf.nn.relu,                # 使用 relu 激活函數
    input_dim=dim))                       # 輸入的陣列大小
model.add(tf.keras.layers.Dense(units=10, # 10 個神經元
    activation=tf.nn.relu ))              # 使用 relu 激活函數
model.add(tf.keras.layers.Dense(units=category,  # 結果的種類
    activation=tf.nn.softmax ))           # 使用 tf.nn.softmax

# 編譯和訓練
model.compile(optimizer='adam',           # 設定編譯處理使用 adam 最佳化
    loss=tf.keras.losses.categorical_crossentropy,
                                          # 損失率使用分類交叉熵
```

```
23.     metrics=['accuracy'])                    # 以正確率為主
24. model.fit(x_train, y_train2,
25.           epochs=20,
26.           batch_size=128)
27.
28. # 建立測試資料
29. x_test,y_test,y_test2=CreateDatasets(category,10,dim)
    # 建立全部 10 個二維的測試特徵值 X 因，而測試結果 Y 有 10 種答案
30. # 測試
31. …                                            # 同上一個範例省略
```

執行結果

```
Epoch 20/20
 128/1000 [==>...............] - ETA: 0s. - loss: 2.1602 - acc: 0.1875.
1000/1000 [===================] - 0s 24us/step - loss: 2.1484 - acc: 0.1880
10/10 [===================] - 0s 7ms/step
score: [2.246499538421631, 0.10000000149011612]
Ans: 6 5 6 4
predict_classes: [6 5 6 4 6 6 6 6 4 6]
y_test  [8 4 8 2 5 4 4 6 1 8]
```

因為這次的訓練資料是 1000 筆，每筆有 2 個的訓練特徵值 X 因，而訓練結果 Y 有 10 種答案，所以看到訓練 20 次時，其訓練數據的正確率也只有到 0.1880，而用來預測測試數據也只有 10% 的資料正確。所以需要將 MLP 的類神經做更好的調整。

教學影片 04_MLPOne-20InputOutput.mp4

27.5 改善預測結果 ─深度學習訓練次數 epochs

MLP 的類神經有幾個方法可以調整參數，本章節範例延續上一個範例，只調整訓練次數 epochs 調整 10 倍來觀察其變化程度。

部分範例程式：05_MLPOne-20InputOutput_epochs.py

```
model.fit(x_train, y_train,
          epochs=20*10,              # 調整深度學習訓練次數 epochs 為 200 次
          batch_size=128)
```

執行結果

```
Epoch 200/200
 128/1000 [==>...............] - ETA: 0 -loss: 0.9316 - acc: 0.7500
1000/1000 [==================] - Os 10us/step - loss: 0.9487 - acc: 0.7140
10/10 [==================] - Os 5ms/step
score: [0.9293791651725769, 0.699999988079071]
Ans: 4 4 6 8
predict_classes: [4 4 6 8 3 4 6 4 1 4]
y_test [4 3 6 7 3 3 6 4 1 4]
```

雖然訓練時達到 0.7140 的正確率，但評估為 score: [0.9293, 0.6999]的正確率只有 69% 的正確率，並且還把[4 4 6 8 3 4 6 4 1 4] 預測為 [4 3 6 7 3 3 6 4 1 4]。另外也建議調整此函數的 batch_size，各位可以試著調整每次訓練筆數的效果，會發現 batch_size 調整較小效果會更好，這是因為學習的次數變多。

測試數據正確率	每一層的神經元	MLP 幾層	訓練次數 epochs	訓練資料筆數
10%	10 個	3 層	20 次	1000 筆
71%	10 個	3 層	200 次	1000 筆

教學影片 05_MLPOne-20InputOutput_epochs.mp4

27.6 改善預測結果－增加神經元和隱藏層

另外，MLP 可以增加神經元和中間隱藏層的數量，以下是修改的部分程式。

部分範例程式：06_MLPOne-20InputOutput_AddDenses.py

```
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Dense(units=10*10,      # 增加神經元的數量
    activation=tf.nn.relu,
    input_dim=dim))
model.add(tf.keras.layers.Dense(units=10*10,      # 增加神經元的數量
    activation=tf.nn.relu ))
model.add(tf.keras.layers.Dense(units=10*10,      # 新增中間隱藏層的數量
    activation=tf.nn.relu ))
model.add(tf.keras.layers.Dense(units=category,   # 輸出層，答案種類
    activation=tf.nn.softmax ))
```

執行結果

```
Epoch 20/20
 128/1000 [==>...............] - ETA: 0s.- loss: 1.2554 - acc: 0.4844
1000/1000 [===================] - Os 20us/step - loss: 1.2357 - acc: 0.4630
10/10 [===================] - Os 6ms/step
score: [1.0962533950805664, 0.5]
Ans: 5 8 8 3
predict_classes: [5 8 8 3 5 3 8 4 5 8]
y_test [4 7 8 2 5 3 7 4 4 8]
```

為了比較其差異性，所有參數調回原本的次數，只有多加上一個新的中間層，以及增加神經元 10 倍的數量。

雖然訓練時降到 0.463 的正確率，但是評估 score: [1.0962533950805664, 0.5] 正確率只有 0.5 的正確率。

教學影片 06_MLPOne-20InputOutput_AddDenses.py.mp4

27.7 改善訓練結果－增加訓練數據集

在本章節將測試，如果增加訓練數據集的資料筆數，是否會有更好的效果？為了比較其差異性，之前所有參數調回原本的參數，只有多加 10 倍的訓練數據集。

部分範例程式：07_MLPOne-20InputOutput_MoreDatasets.py

```
x_train,y_train,y_train2=CreateDatasets(category,1000*10,dim)
```

執行結果

```
Epoch 20/20
  128/10000 [..............] - ETA: 0s. - loss: 0.7909 - acc: 0.7891
 4096/10000 [======>.......] - ETA: 05- loss: 0.7787 - acc: 0.8157
 7680/10000 [========>.....] - ETA: 0s.- loss: 0.7709 - acc: 0.8215
10000/10000 [==============] - 0s 13us/step - loss: 0.7661 - acc: 0.8238
10/10 [==============] - 0s 4ms/step
score: [0.8177118301391602, 0.800000011920929]
Ans: 8 4 6 8
predict_classes: [8 4 6 8 4 7 3 2 6 5]
y_test  [8 4 5 9 4 7 3 2 6 5]
```

訓練時，正確率只有 0.8238，而評估分數為 score: [0.8177118301391602, 0.800000011920929] 的正確率為 80%。

教學影片 07_MLPOne-20InputOutput_MoreDatasets.mp4

27.8 如何達到預測 100% 正確？

接下來要請讀者思考和練習，如何修改 MLP 的類神經程式使其預測值 100% 正確？請先自行研究思考和測試程式，再參考筆者這邊的程式。

部分範例程式：10_MLPOne-20InputOutput-99.9.py

```
1.   x_train,y_train,y_train2=CreateDatasets(category,1000*10,dim) # 10 倍數據
2.
3.   # 建立模型
```

```
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Dense(units=10*100,   #  100 倍神經元
    activation=tf.nn.relu,
    input_dim=dim))
model.add(tf.keras.layers.Dense(units=10*100,   # 100 倍神經元
    activation=tf.nn.relu ))
model.add(tf.keras.layers.Dense(units=10*100,   # 增加中間隱藏層
    activation=tf.nn.relu ))
model.add(tf.keras.layers.Dense(units=10*100,   # 增加中間隱藏層
    activation=tf.nn.relu ))
model.add(tf.keras.layers.Dense(units=category,
    activation=tf.nn.softmax ))
model.compile(optimizer='adam',
    loss=tf.keras.losses.categorical_crossentropy,
    metrics=['accuracy'])
model.fit(x_train, y_train2,
          epochs=20*100,                  # 增加 100 倍深度學習訓練次數
          batch_size=64)                  # 減少比數，增加訓練次數
```

執行結果

```
Epoch 2000/2000
10000/10000 [================] - 2s. 245us/step - loss: 0.0367 - acc: 0.9969
10/10 [==================] - Os 8ms/step
score: [0.006179372314363718, 1.0]
Ans: 4 7 6 2
predict_classes: [4 7 6 2 1 9 2 7 6 6]
y_test [4 7 6 2 1 9 2 7 6 6]
```

其實就是全部都派上用場，將訓練的資料筆數增加為 100 倍，並增加神經元數量，增加中間層，增加 fit 的訓練 epochs 次數與 batch_size 減少，這樣就能夠有很好的效果。訓練時有 0.9969 的正確率，使用測試數據時預測為 score: [0.006179372314363718, 1.0] 的預測正確率為 100%的正確率。

教學影片 08_MLPOne-20InputOutput-99.9.mp4