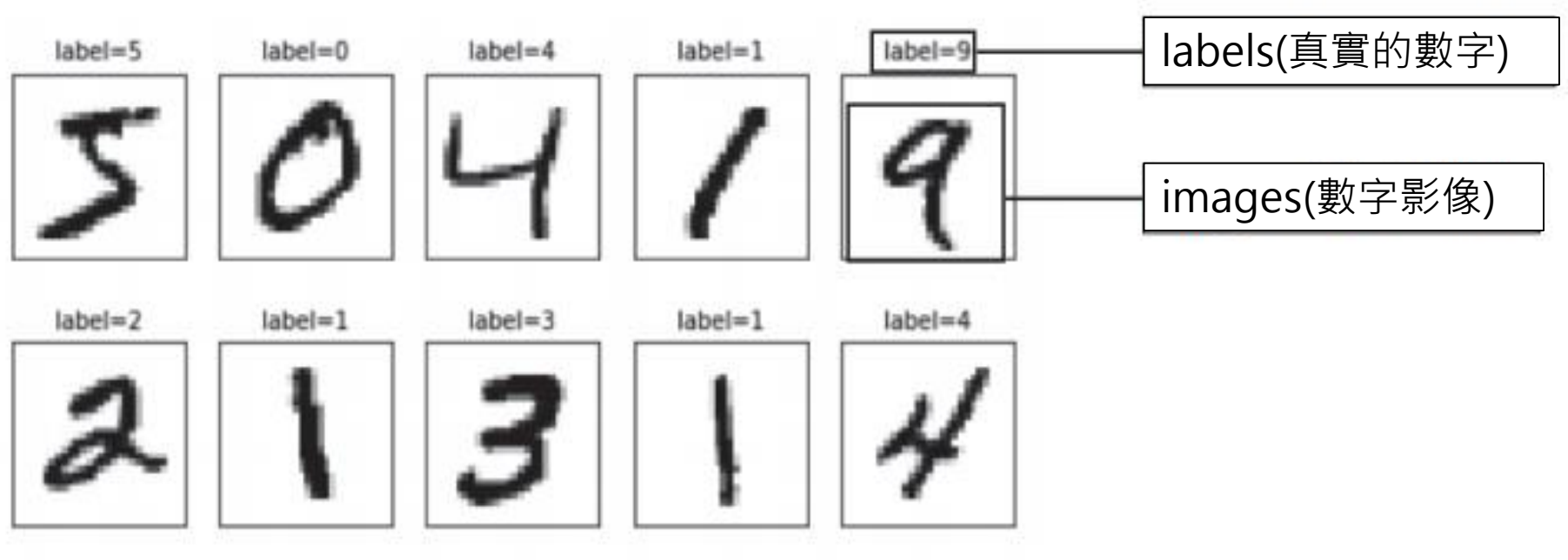
ANN

CH06
Keras MNIST手寫數字辨識資料集介紹

本章我們將介紹MNIST手寫數字辨識資料集,是由Yann LeCun所蒐集,他也是Convulution Nueral Networks的創始人MNIST數字文字辨識資料集,因為資料量不會太多,而且是單色的影像比較簡單,很適合作為深度學習的初學者,練習建立模型、訓練、預測。

MNIST基本介紹

MNIST資料集共有訓練資料60000筆、測試資料10000筆，MNIST資料集每一筆資料都由images(數字影像，28*28)與labels(真實的數字)所組成，如下圖：



6.1 下載MNIST資料

Step1.匯入Keras及相關模組

首先匯入Keras及相關模組。

```
In [1]: import numpy as np
        import pandas as pd
        from keras.utils import np_utils
        np.random.seed(10)

        Using TensorFlow backend.
```

keras自動以Tensorflow作為Backend

程式碼	說明
from keras.utils import np_ utils	匯入keras.utils因為後續要將label標籤轉換為One-hot encoding
import numpy as np	匯入numpy模組，NumPy是Python語言的擴充程式庫。支援維度陣列與矩陣運算。
np.random.seed(10)	設定seed可以讓每次需要隨機產生的資料,都有相同的輸出

Step2. 匯入Keras模組

Keras已經提供了現成模組，可以幫您下載並讀取mnist資料，所以先匯入mnist模組。

```
from tensorflow.keras.datasets import mnist
```

Step3. 第一次執行下載mnist資料

第一次執行mnist.load_data()方法，程式會檢查使用者目錄下，是否已經有mist資料集檔案，如果還沒有，就會下載檔案。以下是第一次執行下載檔案的畫面。因為必須要下載檔案所以執行時間會比較久。

```
In [3]: (X_train_image, y_train_label), \
        (X_test_image, y_test_label) = mnist.load_data()

        Downloading data from https://s3.amazonaws.com/img-datasets/mnist
        .pkl.gz
```

Step4. 查看下載的mnist資料檔案

```
[ ]    !ls ~/.keras/datasets
```

1.使用ls 指令：查看已下載的檔案

```
mnist.npz
```

2. 執行後顯示：已經下載的檔案mnist.npz

Step5. 讀取mnist資料

當您下次再次執行mnist.load_data()時，由於之前已經下載檔案，所以就不需要再執行下載，只需要讀取檔案，執行時間就會快很多。

```
In [3]: (X_train_image, y_train_label), \
        (X_test_image, y_test_label) = mnist.load_data()
```

Step6. 查看mnist資料

下載後，你可以使用函數 len()，查看mnist資料集筆數。

```
In [4]: print('train data=',len(x_train_image))
        print(' test data=',len(x_test_image))
        train data= 60000
         test data= 10000
```

以上執行結果，你可以看到資料可分為2部分：

- train 訓練資料60000筆。
- test測試資料10000筆。

Step6. 查看mnist資料

可以使用函數 type() ，查看mnist資料型態。

```
[ ]    type(x_train_image)
```

1.使用type()函數：查看訓練資料數字影像的資料型態

```
numpy.ndarray
```

2.執行後顯示：資料型態是numpy的ndarray

```
[ ]    type(y_train_label)
```

3.使用type()函數：查看訓練資料標籤的資料型態

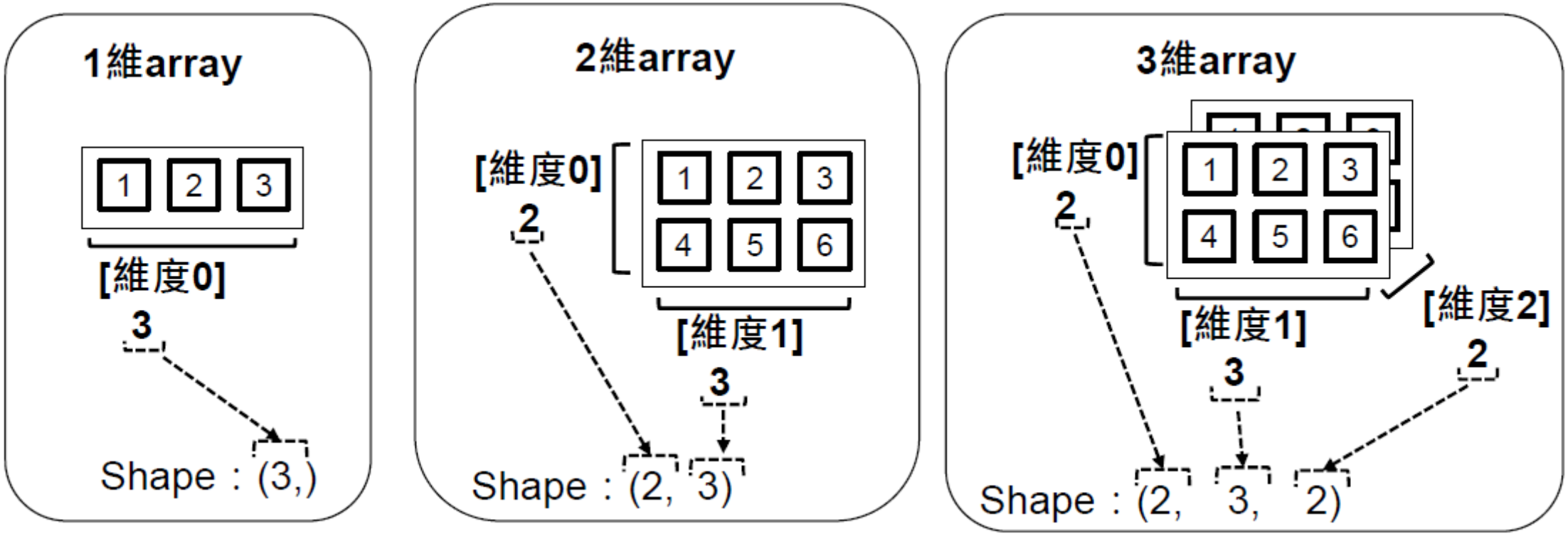
```
numpy.ndarray
```

4.執行後顯示：資料型態是numpy的ndarray

6.2 查看訓練資料

查看mnist訓練與測試資料形狀shape

訓練與測試資料都是 numpy array 資料，具有形狀(shape)屬性，也就是每一個維度的長度，舉例說明如下：



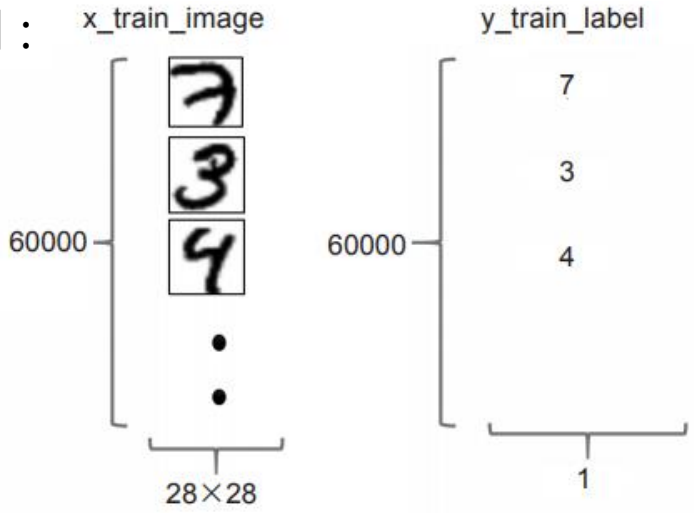
6.2 查看訓練資料

Step1.訓練資料是由images與labels所組成

```
In [5]: print ('x_train_image:',x_train_image.shape)
        print ('y_train_label:',y_train_label.shape)

        x_train_image: (60000, 28, 28)
        y_train_label: (60000,)
```

訓練資料是由images與labels所組成共60000筆，images是單色的數字影像，labels是數字影像的真實值，如下圖：



Step2. 定義plot_image函數顯示數字影像

為了能夠顯示images數字影像，我們建立下列plot_image函數。

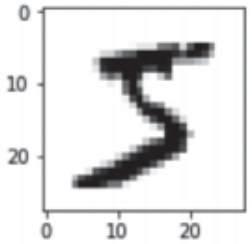
```
In [7]: import matplotlib.pyplot as plt
        def plot_image(image):
            fig = plt.gcf()
            fig.set_size_inches(2, 2)
            plt.imshow(image, cmap='binary')
            plt.show()
```

程式碼	說明
import matplotlib.pyplot as plt	首先匯入 matplotlib.pyplot 模組
def plot_image(image):	定義 plot_image 函數，傳入 image 作為參數
fig = plt.gcf() fig.set_size_inches(2, 2)	設定顯示圖形的大小
plt.imshow(image, cmap='binary')	使用 plt.imshow 顯示圖形，傳入參數 image 是 28×28 的圖形，cmap參數設定為 binary 以黑白灰階顯示
plt.show()	開始繪圖

Step3. 執行 plot_image 函數查看第 0 筆數字影像

以下程式呼叫 plot_image 函數，傳入 mnist.train.images[0]
也就是訓練資料集的第 0 筆資料，結果可看到數字 5 的圖形

```
In [8]: plot_image(x_train_image[0])
```



Step4. 查看第0筆labels資料

```
In [8]: y_train_label[0]
Out[8]: 5
```

第0筆labels資料，是第0筆數字影像的真實值，所以是5。

6.3 查看多筆訓練資料 images 與 label

Step1. 建立 plot_images_labels_prediction () 函數

方便查看數字圖形、真實數字結果與預測結果

```
def plot_images_labels_prediction(
    images,labels,prediction,idx=0,num=10):
    fig = plt.gcf().set_size_inches(12, 12)
    if num>25: print("num must be 1~25");return;
    for i in range(0, num)
        ax=plt.subplot(5,5, i+1)
        ax.set_xticks([]);ax.set_yticks([])
        ax.imshow(images[idx], cmap='binary')
        title= "label=" +str(labels[idx])
        if len(prediction)>0:
          title+=",predict="+str(prediction[idx])
        ax.set_title(title,fontsize=10)
        idx+=1
    plt.show()
```

1.定義：函數名稱與參數

2.設定：顯示圖形大小與數量

3.for迴圈：i由0至num次，讀取資料並顯示num個子圖形

4.設定子圖形：x,y無刻度

5.設定：子圖形影像

6.設定：子圖形標題

7.設定idx加1：回到for迴圈讀取下一筆資料

8. 開始顯示圖形

import matplotlib.pyplot as plt

- 匯入 matplotlib.pyplot 模組，後續會使用 plt 引用

def plot_images_labels_prediction(images, labels, prediction, idx, num=10):

- 定義 plot_images_labels_prediction() 函數，傳入下列參數:
  - Images: 數字影像、labels: 真實值、prediction: 預測值
  - Idx: 開始顯示的資料index，預設是0，也就是讀取第0筆資料
  - num: 要顯示的資料筆數，預設10筆，不超過25筆

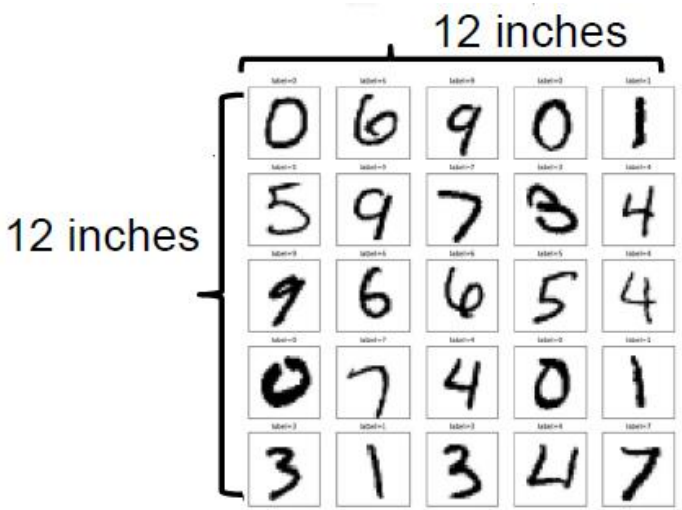
fig = plt.gcf().fig.set_size_inches(12, 12)

- 設定顯示圖形的大小，12×12 inches

If num>25: print(“num must be 1~25”); return;

- 使用if判斷如果顯示筆數大於25，則顯示錯誤訊息，並使用return結束函數執行

plt.show() #開始繪圖



程式碼	說明
for i in range(0, num):	開始 For 迴圈，執行區塊內的程式碼，劃出 num 個數字圖形
ax=plt.subplot(5,5, 1+i)	建立 subgraph 子圖形為5行5列
ax.imshow(images[idx], cmap='binary')	畫出 subgraph 子圖形
title= "label=" +str(labels[idx])	設定子圖形標題 title，顯示標籤欄位
if len(prediction)>0:	如果有傳入預測結果
title+=",predict="+str(prediction[idx])	標題 title 加入預測結果
ax.set_title(title,fontsize=10)	設定子圖形標題 title 與大小
ax.set_xticks([]);ax.set_yticks([])	設定不顯示刻度
idx+1	讀取下一筆

執行 plot_images_labels_prediction 顯示1筆測試資料影像與標籤

```
[ ]plot_images_labels_prediction(
        images=x_test_image,
        labels=y_test_label,
        prediction=[],
        idx=10,
        num=1)
```

1.執行函數

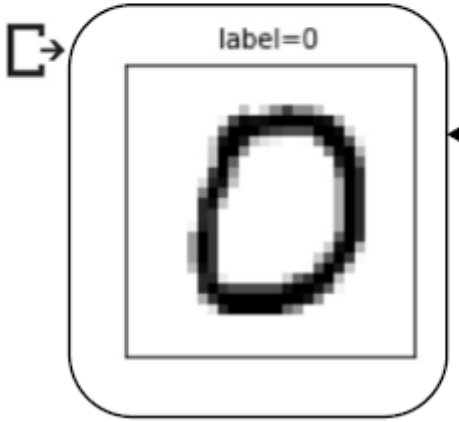
參數images:輸入測試資料影像

參數labels:輸入測試資料的label

參數prediction:因無預測結果，所以傳入空list

參數idx:輸入從第幾筆資料開始顯示，例如：設定為第10筆

參數num: 輸入要顯示筆數(預設顯示10筆，可設定為1~25筆資料)，例如：設定為1筆

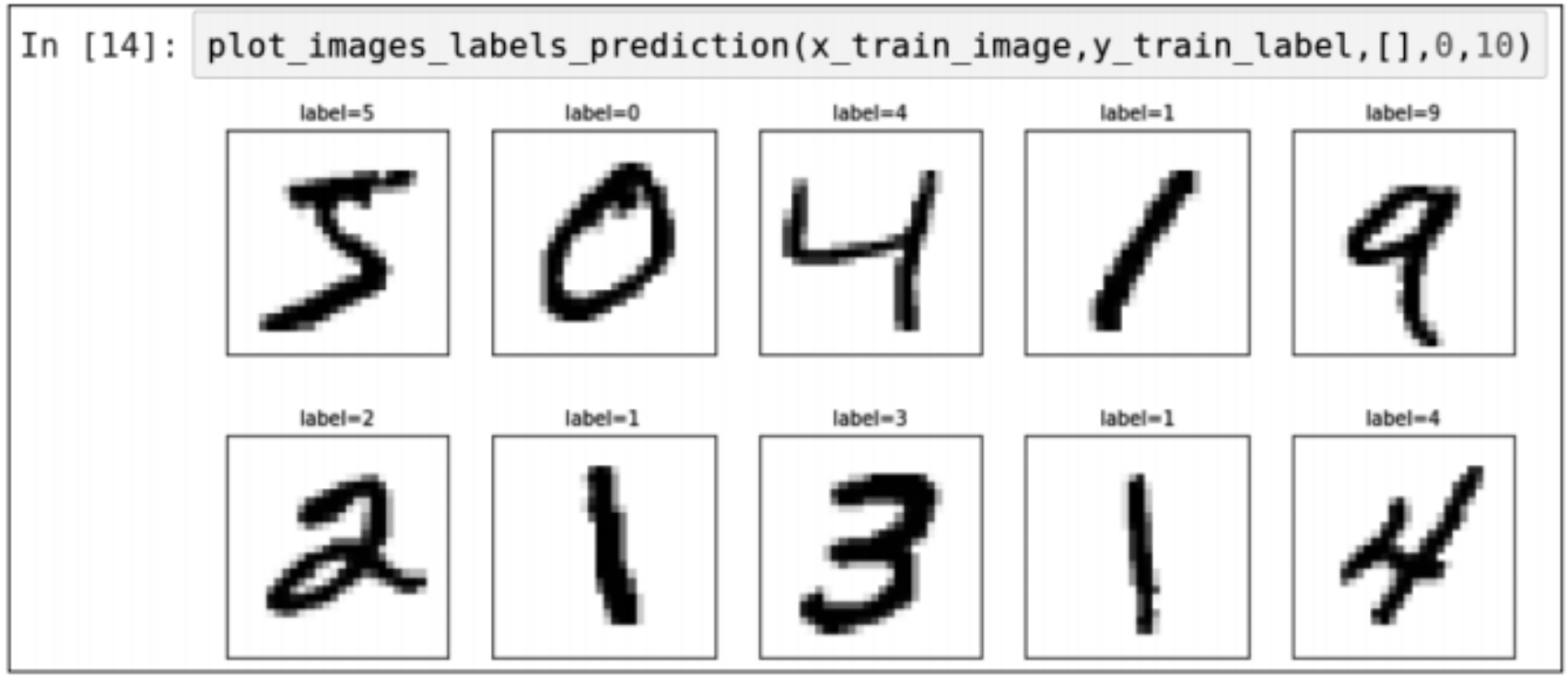


2.執行後顯示：
測試資料由第10筆開始，顯示1筆影像與標籤

Step2. 查看訓練資料前10筆資料

執行plot_images_labels_prediction()函數,顯示訓練資料前10筆資料。輸入x_test_image，y_test_label，不過目前還没有預測結果prediction所以傳入空list[]，由第0筆開始顯示至第9筆。

```
In [14]: plot_images_labels_prediction(x_train_image,y_train_label,[],0,10)
```



Step3. 查看test測試資料

查看test測試資料筆數，你可以看到共計10000筆資料。

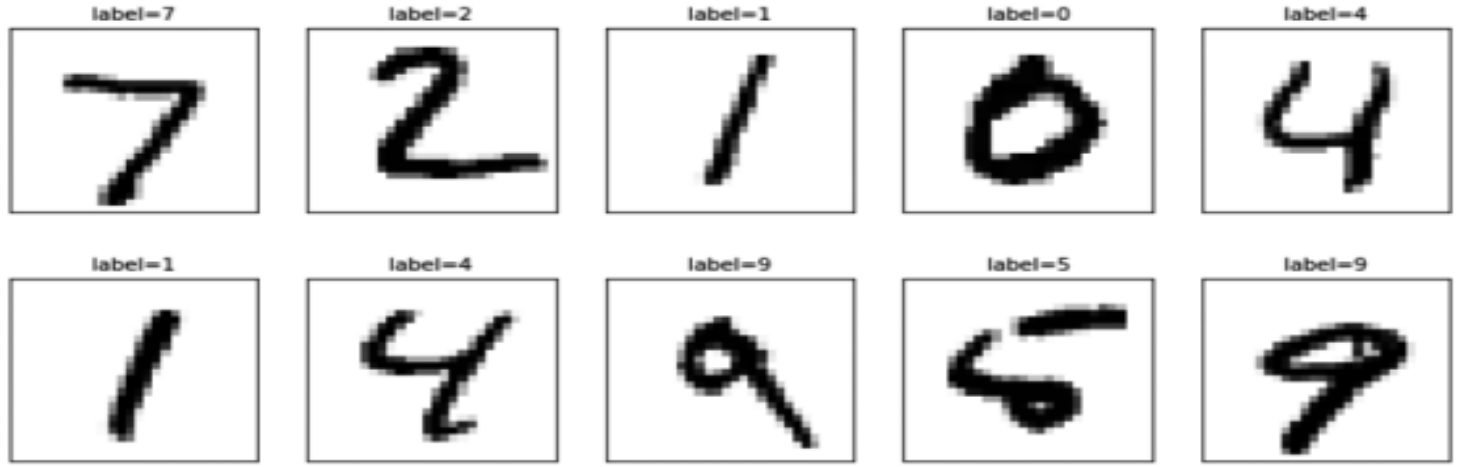
```
In [12]: print ('x_test_image:',x_test_image.shape)
         print ('y_test_label:',y_test_label.shape)

         x_test_image: (10000, 28, 28)
         y_test_label: (10000,)
```

Step4. 查看test測試資料

執行 show_images_labels()顯示測試資料前10筆資料。

```
In [15]: plot_images_labels_prediction(x_test_image,y_test_label,[],0,10)
```



6.4 多層感知器模型資料預處理

- 在建立多層感知器模型(Multilayer Perceptron)之前，需先將 images 與 labels 進行預處理，才能開始進行訓練與預測

- 資料預處理分為兩部份
  - Features: 數字影像的特徵值
  - Labels: 數字影像的真實值

  - 範例程式：TF_Keras_Mnist_Preprocess.ipynb

6.5 Features(數字影像的特徵值)資料預處理

1. 將原本 28×28 數字影像，以 reshape 轉換為1維向量，長度為784，格式為 float
2. 數字影像 image 的一維向量數字標準化

Step1. 查看 image 的 shape

- 根據以下程式，可知每一個數字影像的 shape 都是 28×28

```
In [14]: print ('x_train_image:',x_train_image.shape)
         print ('y_train_label:',y_train_label.shape)

         x_train_image: (60000, 28, 28)
         y_train_label: (60000,)
```

Step2. 將 image 以 reshape 轉換

- 將 28×28 的2維數字影像，以 reshape 轉換為1維向量
- 再以 astype 轉換格式為 float

```
In [15]: x_Train =x_train_image.reshape(60000, 784).astype('float32')
         x_Test = x_test_image.reshape(10000, 784).astype('float32')
```

reshape為2維張量：
- 第0個維度：代表筆數，因筆數不確定所以設定為-1
- 第1個維度：將原本28x28轉換為784個float數值

Step3. 查看轉為1維向量的 shape

- 每個影像共有 784 個 float 數字

```
In [16]: print ('x_train:',x_Train.shape)
         print ('x_test:',x_Test.shape)

         x_train: (60000, 784)
         x_test: (10000, 784)
```

Step4. 查看 image 影像內容

- 查看第 0 筆影像內容
- 內容大部份是0，少部分是0-255的數字，代表圖形每一個點的灰階顏色深淺

```
In [17]: x_train_image[0]

Out[17]: array([[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
                [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
                [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
                [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
                [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
                [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   3,
                 18,  18,  18, 126, 136, 175,  26, 166, 255, 247, 127,   0,   0,
                  0,   0],
                [  0,   0,   0,   0,   0,   0,   0,   0,  30,  36,  94, 154, 170,
                253, 253, 253, 253, 253, 225, 172, 253, 242, 195,  64,   0,   0,
```

Step5. 將數字影像 image 的數字標準化

- 數字標準化可以提高後續訓練模型的準確率
- 因為 image 數字介於0-255，最簡單的標準化方式是除以255

```
In [18]: x_Train_normalize = x_Train/ 255
         x_Test_normalize = x_Test/ 255
```

Step6. 查看數字標準化後的結果

- 標準化後，所有數字都介於0與1之間

```
In [19]: x_Train_normalize[0]
```

```
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.01176471, 0.07058824, 0.07058824,
0.07058824, 0.49411765, 0.53333336, 0.6862745 , 0.10196079,
0.6509804 , 1.        , 0.96862745, 0.49803922, 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.11764706, 0.14117648, 0.36862746, 0.6039216 ,
0.6666667 , 0.99215686, 0.99215686, 0.99215686, 0.99215686,
0.99215686, 0.88235295, 0.6745098 , 0.99215686, 0.9490196 ,
0.7647059 , 0.2509804 , 0.        , 0.        , 0.        ,
```

查看影像特徵標準化前後差異

1. 查看標準化前：訓練資料第0筆

2.選取第200至220數字：因為影像的像素共有784個點，大部分都是0，所以我們顯示第200至220數字非0的點查看

```
[ ]   print(x_train[0][200:220] )
```

3.執行後顯示標準化前：是0~255的數字

```
[  0.   0.   0.  49. 238. 253. 253. 253. 253. 253. 253. 253. 253. 251.
  93.  82.  82.  56.  39.   0.]
```

4. 查看標準化後：訓練資料第0筆

5.選取第200至220數字

```
[ ]   print(x_train_normalize[0][200:220])
```

6.執行後顯示標準化後：是0至1之間的數字

```
[0.         0.         0.         0.19215687 0.93333334 0.99215686
 0.99215686 0.99215686 0.99215686 0.99215686 0.99215686 0.99215686
 0.99215686 0.9843137  0.3647059  0.32156864 0.32156864 0.21960784
 0.15294118 0.        ]
```

6.6 Labels(數字影像的真實值)資料預處理

- Labels 標籤欄位原本是0-9的數字，可用 One-hot encoding 轉換為10個0或1的1維向量
- 例如7轉換為 [0000000100]

Step1. 匯入 np_utils 模組

```
[ ]    import tensorflow.keras.utils as np_utils
```

1. 匯入 np_utils模組

Step2. Label 標籤欄位執行 One-hot encoding 轉換

- 使用 np_utils.to_categorical 分別傳入參數 y_train_label(訓練資料)與 y_test_label(測試資料)的label標籤欄位，執行轉換

```
In [21]: y_TrainOneHot = np_utils.to_categorical(y_train_label)
         y_TestOneHot = np_utils.to_categorical(y_test_label)
```

Step3. 查看執行 One-hot encoding 轉換之後的 label 標籤欄位

- 第一筆資料真實值是 5，轉換後只有第 6 個數字(由 0 算起)是 1

```
In [22]: y_TrainOneHot[:5]

Out[22]: array([[0., 0., 0., 0., 0., 1., 0., 0., 0., 0.],
                [1., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
                [0., 0., 0., 0., 1., 0., 0., 0., 0., 0.],
                [0., 1., 0., 0., 0., 0., 0., 0., 0., 0.],
                [0., 0., 0., 0., 0., 0., 0., 0., 0., 1.]], dtype=float32)
```

結論

- 本章介紹如何使用 Keras 下載並且讀取 Mnist 資料集
- 介紹 Mnist 資料集的特色，並完成資料的預處理
- 後續就可以使用 Keras 建立多層感知器模型進行訓練
- 並使用模型進行預測