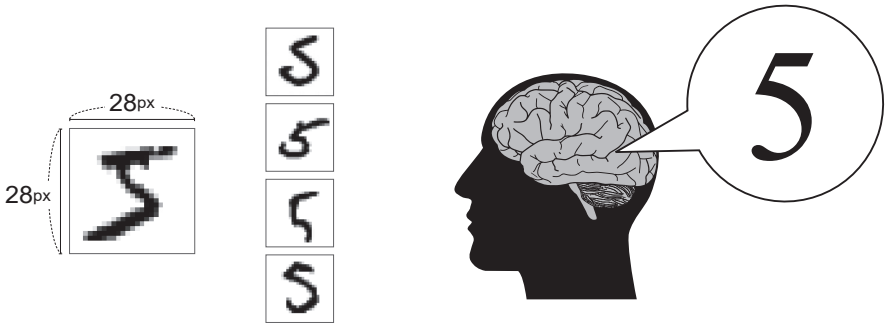
2.1 認識多層感知器 (MLP)

2.1.1 認識神經網路

這是一個長寬各 28 像素的手寫數字圖片，但是人類的大腦很神奇，可以很輕鬆就辨識出這個數字。而且當我們換上代表相同數字的其他手寫圖片，即使內容前後組成有所差異，也都能被輕易地辨識出來。

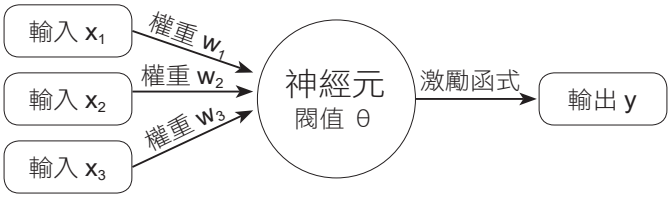


不過當我們希望寫個程式來重現這樣的能力，瞬間就會變成非常困難的任務！神經網路的加入是讓機器學習解決這個問題的很好途徑，但你了解它是怎麼運作的嗎？這裡我們將用簡單的方式說明神經網路，了解它完整的樣貌。

神經元的運作

神經網路 (Neural Network) 一如其名，是由人類的大腦神經結構的運作借鏡而來，在機器學習的世界中，神經元就像是大腦的神經細胞，是神經網路最基礎的結構，在它們相互結合下，建構整個龐大的運作網路，實現學習、處理及預測等功能。

神經元是彼此相連的，下圖是單獨取出單一神經元的運作模型，每個神經元中都有一個 **閥值**，它的功能是設下一個門檻，如果所接收的訊號值運算後大於這個門檻，神經元就會被觸發，將接收的值經由 **激勵函式** 轉換，輸出到下一個神經元。

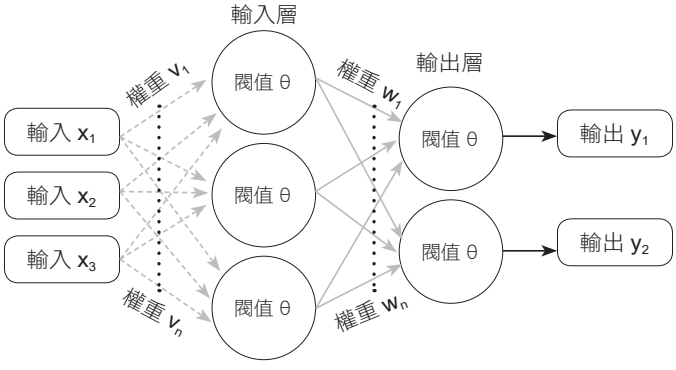


▲ 單一神經元模型：單層感知器

其中接收的訊號值就是由其他神經元傳遞過來的多個 **輸入值 (x)** 乘上相關的 **權重 (w)** 再與 **閥值 (θ)** 比較的動作，是很重要的關鍵，**機器學習就是在調整每個輸入值與所配置的權重**。訊號值越大越容易觸發神經元 對於神經網路運作的影響也越大。反之，訊號值越小影響就越小，而太小的訊號甚至可以忽略以節省運算的資源，讓輸出值的誤差降到最小，這個調整轉換輸出值的方式就是 **激勵函式**。

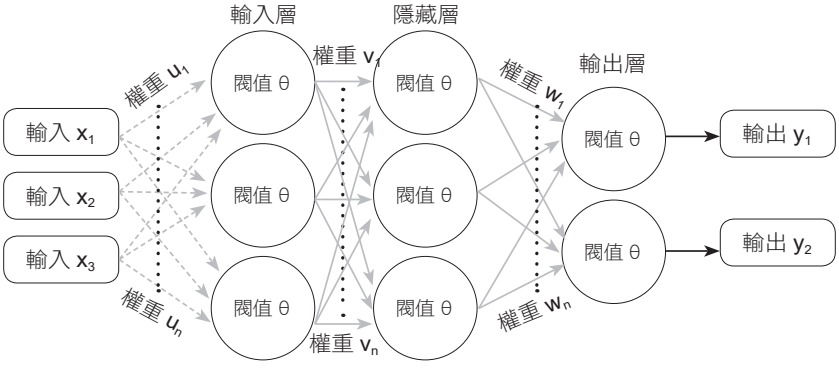
感知器的模型

感知器 (Perceptron) 就是模仿人類大腦皮層中神經網路模型進行學習的機制，所以傳遞訊號的神經元都是按層排列。單一神經元模型就是最單純的 **單層感知器**。為了解決更複雜的問題，於是發展出由接收輸入訊號的 **輸入層** 與產生輸出信號的 **輸出層** 所建構的 **2 層感知器**。



▲ 2 層感知器

為了提高學習的準確率，神經網路更發展到有一個 **輸入層**、一個或多個 **隱藏層** 及一個 **輸出層** 的 **多層感知器** (MLP，Multilayer Perceptron)。



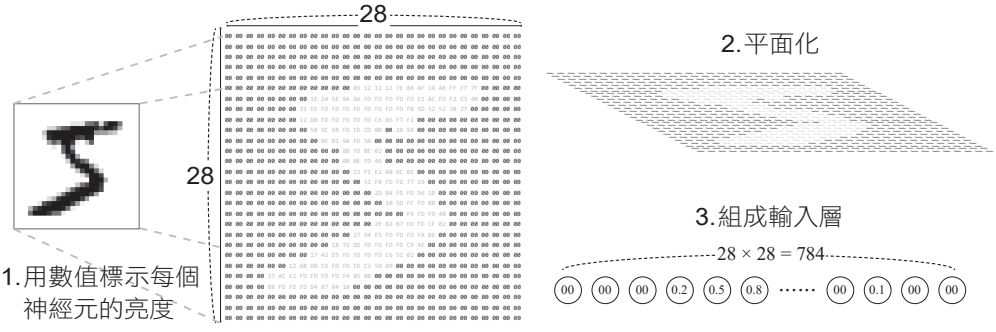
▲ 多層感知器

2.1.2 多層感知器的運作

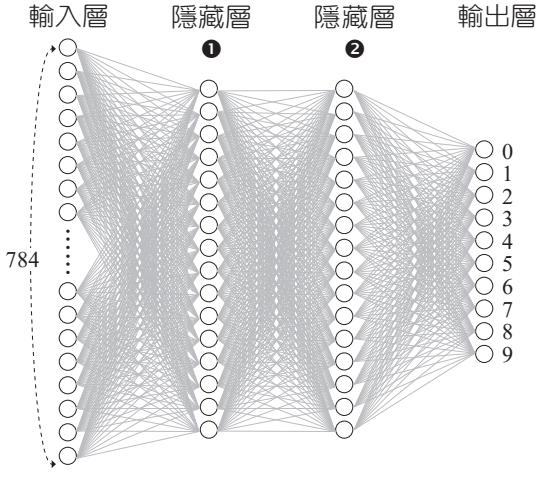
介紹了神經元、感知器後，接著我們就利用手寫數字辨別來說明多層感知器大致的運作方式。

多層感知器的模型

神經元在接收輸入訊號後可以想像它是儲存了一個數字的容器，其值介於 0 到 1 之間。以 28 * 28 像素的手寫辨識圖片來說，每個像素就是一個神經元，也就是一張圖片在 **輸入層** 總共有 784 個神經元，每個神經元都儲存了一個數字來代表對應像素的灰階值，數值的範圍介於 0 跟 1 之間。而灰階值 0 代表黑色，1 代表白色，這些數字我們稱為 **激勵值**，數值越大則該神經元就越亮。在輸入時要將矩陣平面化 (將 28 列前後相接成一列)，也就是這 784 個神經元組成了神經網路的第一層。



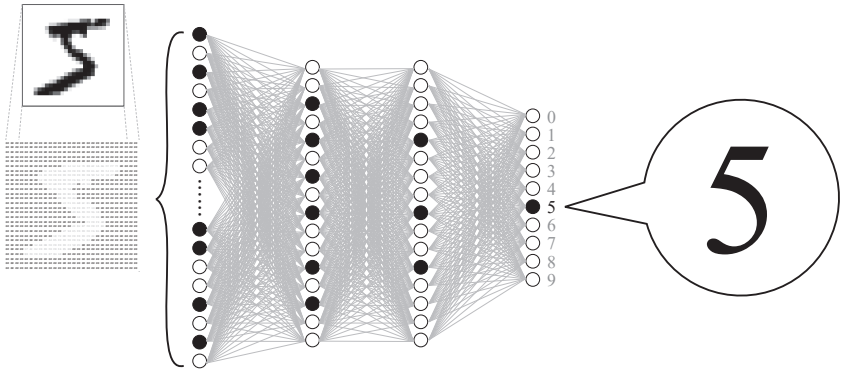
完成了輸入層，先不管其他層的內容，我們來看看它最右方的 **輸出層**，也就是最後判斷的結果，其中有 10 個神經元，各代表了數字 0 到 9，其中也有代表的激勵值。



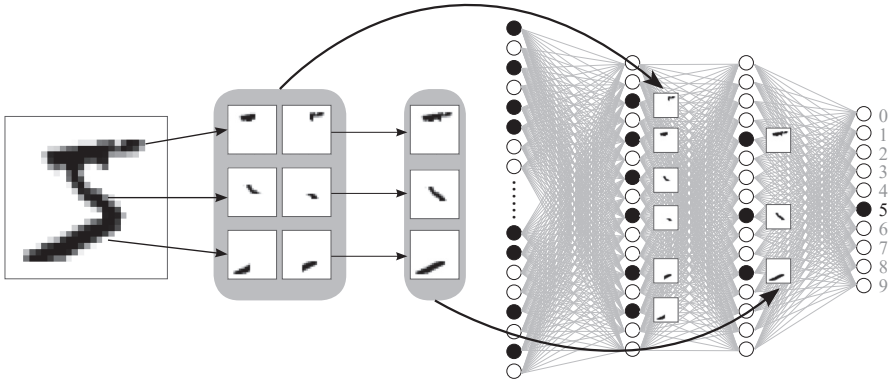
為了方便說明，在這裡我們設計了二個 **隱藏層**，每層有 16 個神經元。在真實的案例中可依據需求來設置調整隱藏層與神經元的數量。

多層感知器的流程

不同以往的資料處理技術，在神經網路中每一層神經元中激勵值操作的結果會影響下一層的激勵值，一層一層之間激勵值的傳遞最後輸出判斷的結果，它的本質是在模仿人類大腦細胞被激發，引發其他神經細胞的連鎖反應。



而激勵值是如何在各層之間傳遞的呢？而隱藏層又是如何運作的呢？再回到剛才的問題，在辨識手寫數字圖片時，可以將文字拆解成各個筆劃較好處理。

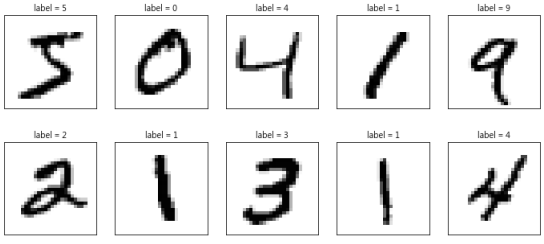


在理想的狀態下我們會希望在第二層的神經元能辨識各別不同的筆劃，也就是對應到某個筆劃的神經元激勵值趨近於 1 而被點亮，再到下一層時能連接其他的部份再將合併後的對應神經元點亮，最終到輸出層時就能將代表結果數字的神經元點亮，得到最後的答案。

2.2 認識 Mnist 資料集

在電腦螢幕上顯示「Hello World」，是許多程式初學者所學習的第一個範例。在機器學習的領域中，Mnist 資料集也擁有一樣的地位，當你開始接觸相關的資料，無論是學習或是教學，都一定不會陌生。

Mnist 資料集 (Modified National Institute of Standards and Technology database)，是由紐約大學 Yann LeCun 教授蒐集整理許多人 0 到 9 的手寫數字圖片所形成的資料集，其中包含了 60000 筆的訓練資料，10000 筆的測試資料。在 Mnist 資料集中，每一筆資料都是由 images (數字圖片) 和 labels (真實數字) 組成的單色圖片資料，很適合機器學習的初學者，練習建立模型、訓練和預測。



Mnist 資料集的應用範圍很廣，除了進行機器學習的練習，還可以真正使用在生活中，例如用來辨識支票的手寫金額、電話號碼、車牌號碼，甚至改考卷呢！

2.2.1 下載與讀取 Mnist 資料集

在接下來的幾章，我們都將利用 Mnist 資料集進行不同的機器學習與深度學習，省去收集、整理、格式化資料等繁瑣的工作，將注意力專注在學習上，你也可以藉由這個成熟的資料集磨鍊自己的開發技巧。

下載 Mnist 資料集

在 Python 中透過 Keras 就可以下載 Mnist 資料集，請先匯入 mnist 模組，再利用 mnist 模組的 load_data 方法，即可載入資料，語法如下：

```
from keras.datasets import mnist
(train_feature, train_label), \
(test_feature, test_label) = mnist.load_data()
```

2.4 多層感知器實戰：Mnist 手寫數字圖片辨識

本章將建立多層感知器模型，並以 Mnist 手寫數字圖片資料集，訓練模型、評估準確率並儲存，然後利用訓練的模型，辨識 Mnist 手寫數字圖片。

2.4.1 多層感知器訓練和預測

多層感知器的重點在於訓練與預測，Mnist 資料集在這二個階段的重要工作如下：

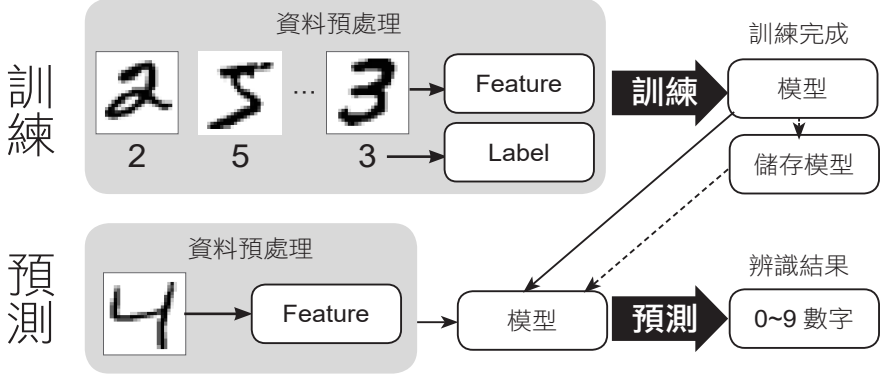
訓練 (Train)

Mnist 資料集共有 60000 筆訓練資料，將訓練資料的 Feature(數字圖片特徵值) 和 Label(數字真值實) 都先經過預處理，作為多層感知器的輸入、輸出，然後進行模型訓練。

預測 (Predict)

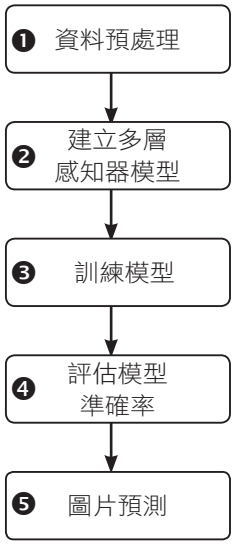
模型訓練完成以後就可以用來作預測，將要預測的數字圖片，先經過預處理變成 Feature(數字圖片特徵值)，就可送給模型作預測，得到 0~9 數字的預測結果。

也可以將訓練好的模型儲存起來，以後就可以不再重複訓練，如果要在其他程式中使用，只要載入儲存的模型就可以進行預測。



2.4.2 多層感知器手寫數字圖片辨識流程

以多層感知器進行 Mnist 手寫數字圖片訓練和預測的步驟如下：



將 Feature 特徵值換為 784 個 float 數字的 1 維向量，並將 float 數字標準化，將 Label 轉換為 One-Hot Encoding 編碼。

建立含有輸入、隱藏、輸出層的模型。

以訓練資料的 Feature 和 Label，執行指定次數的訓練。

使用測試資料，評估模型準確率。

以訓練完成的模型，對想要測試的數字圖片進行預測。

2.4.3 資料預處理

載入資料

匯入 mnist 模組，以 mnist 模組的 load_data 方法，載入資料。

```
from keras.datasets import mnist
(train_feature, train_label),\
(test_feature, test_label) = mnist.load_data()
```

Feature 特徵值轉換

將 Feature 特徵值轉換為 784 個 float 數字的 1 維向量。

```
train_feature_vector = train_feature.reshape(len(train_feature), 784)
                       .astype('float32')
test_feature_vector = test_feature.reshape(len( test_feature), 784)
                       .astype('float32')
```

Feature 特徵值標準化

將 0~255 的數字，除以 255 得到 0~1 之間浮點數，稱為標準化 (Normalize)，以提高模型預測的準確度。

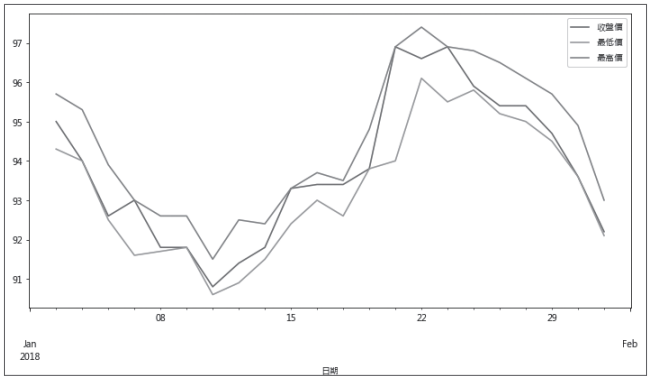
```
train_feature_normalize = train_feature_vector/255
test_feature_normalize = test_feature_vector/255
```

9.1 專題方向

股票是許多人投資的重要標的，且台灣股民開戶數也已突破千萬，取得正確的股票投資訊息是許多人關心的大事。就讓機器學習幫我們分析個股走勢吧！

專題檢視

本專題將先分析台灣證券交易所日成交資料網頁，擷取個股單月每日各項資料，再以單月資料繪製個股統計圖。



有了單月個股資料，可以使用迴圈集合全年 12 個月資料繪製全年統計圖。本專題使用 plotly 模組繪製互動統計圖，不但可以動態顯示單日股價資料，若是對某範圍資料感興趣，可以選取該範圍做細部觀察。

有了股票相關資訊後，就可以利用機器學習對股票進行預測，我們以鴻海 2018 年 1~12 月的收盤價作為訓練資料。

9.3.3 以 plotly 繪製全年個股統計圖

當股價範圍較大，會使得全年個股統計圖形太小不易觀察。若是以 plotly 模組繪製統計圖，不但可用文字方式動態顯示日期及股價，也可以局部放大需要詳細觀察的區塊。

以系統管理員身分執行 conda 即可安裝 plotly 模組，安裝需要一段時間，請耐心等候。

```
conda install -c plotly plotly spyder
```

使用 plotly 模組必須先含入模組程式庫，語法為：

```
import plotly
from plotly.graph_objs import Scatter, Layout
from plotly.offline import plot
```

以 plotly 模組繪圖，只需修改下列程式碼。

程式碼：ch09-7.py

```
…略
#conda install -c plotly plotly spyder
from plotly.graph_objs import Scatter, Layout
from plotly.offline import plot
…略
data = [
    Scatter(x=pdstock['日期'], y=pdstock['收盤價'], name='收盤價'),
    Scatter(x=pdstock['日期'], y=pdstock['最低價'], name='最低價'),
    Scatter(x=pdstock['日期'], y=pdstock['最高價'], name='最高價')
]
plot({"data": data, "layout": Layout(title='2018年個股統計圖')},
     auto_open=True)
```

程式說明

- 5-6　含入 plotly 程式庫。
- 34-37　設定圖形的 x 軸及 y 軸資料來源。每一個「Scatter(x=……」會繪製一條曲線。
- 35-37　「name」設定圖示說明中的文字。
- 39　以 plotly.offline 繪圖產生暫存的 .html 檔案，並以瀏覽器顯示。

plotly 模組會將繪製的圖形以離線方式顯示在暫存的 .html 檔案中，將滑鼠移到圖形中，就會動態顯示該日的日期及各種股價資訊：

Plotly 圖形右上方的工具列提供許多圖形操作功能。

- 📷：將圖形下到本機，圖形格式為「png」。
- 🔍：拖曳滑鼠設定顯示圖形範圍，此功能可將局部圖形放大觀察。
- ✣：使用滑鼠拖曳移動圖形。 ➕：放大圖形。 ➖：縮小圖形。
- ⛶：讓系統自動判斷繪圖座標範圍。 🏠：使圖形回復到最初繪製狀態。

按 🔍 後拖曳滑鼠選取部分區塊即可將該區塊圖形放大，仔細觀察該區塊資訊，放開滑鼠就會放大選擇的區塊圖。

9.4 股票預測

有了股票相關資訊後，就可以利用機器學習，對股票進行預測，由於股票的變化和前期資訊有很大的關聯，運用 RNN (循環神經網路) 的 LSTM (長短期記憶) 建立訓練模型最適合。我們以鴻海 2018 年 1~12 月的收盤價作為訓練資料。

程式碼：mystock_rnn.py

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import LSTM, Dense
from plotly.graph_objs import Scatter, Layout
from plotly.offline import plot

def load_data(df, sequence_length=10, split=0.8):
    data_all = np.array(df).astype(float)     # 轉為浮點型別矩陣
    #print(data_all.shape) # (241,1)
    scaler = MinMaxScaler()
    data_all = scaler.fit_transform(data_all)  # 將數據縮放為 0~1 之間
    data = []
    # data 資料共有 (241-10-1)=230 筆
    for i in range(len(data_all) - sequence_length - 1):
        # 每筆 data 資料有 11 欄
        data.append(data_all[i: i + sequence_length + 1])
    reshaped_data = np.array(data).astype('float64')
```

11.1 專題方向

Haar 特徵分類器可以在圖片中偵測某特定物件是否存在，並可得知該物件的座標位置。這個特定物件可以是人臉、交通標誌、貓、狗等，依據使用的 Haar 特徵模型檔而異。

本專題以偵測「車牌號碼」位置說明如何建立 Haar 特徵分類器模型。

專題檢視

首先蒐集「正樣本圖片」，就是包含要偵測物件的圖片，本專題就是含有車牌號碼的圖片，正樣本圖片格式必須為 BMP，所以將圖片格式轉換為 BMP，同時將圖形大小轉換為 300x225 像素。再蒐集「負樣本圖片」，負樣本圖片就是不包含要偵測物件的圖片，轉換負樣本圖片格式為灰階，圖形大小為 500x375 像素。

正樣本圖片中必須將要偵測的物件框選出來，系統才知道要偵測的物件是什麼，本專題要偵測的是車牌號碼，所以必須框選所有正樣本圖片中的車牌號碼。使用 Haar 特徵分類器模型進行物件偵測時，會根據訓練時的寬高比來框選物件。新式車牌的寬約為高的 3.8 倍，撰寫程式 (modMark.py) 將框選區域寬高比小於 3.8 的圖片，調整寬高比為 3.8。

正樣本圖片的數量越多，建立的 Haar 特徵分類器模型效果越好，撰寫程式 (make4Pic.py) 增加正樣本圖片，本專題使用的方法是移除邊緣長寬各 10% 圖形來產生新圖形，如此每張圖片可新增 4 張新圖片。最後進行訓練建立模型。

建立完成 Haar 特徵分類器模型後，可使用 Opencv 讀入模型，對未知的車牌進行偵測，觀察偵測車牌號碼的正確率如何。



11.2 準備訓練 Haar 特徵分類器資料

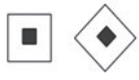
Opencv 最為人稱道的就是「人臉偵測」。使用 Opencv 提供的 Haar 特徵分類器人臉模型，即可輕鬆偵測人臉位置。是否也可以偵測其他物件呢？若要偵測指定物件，就要建立該物件的 Haar 特徵分類器模型，再使用自行建立的 Haar 特徵分類器模型偵測物件。

目前許多停車場已使用自動車牌辨識系統經營以節省人力成本，本章及下一章將模擬建立停車場自動車牌辨識系統，本章以偵測「車牌號碼」位置說明如何建立 Haar 特徵分類器模型，偵測的車牌將在下一章以機器學習辨識出車牌號碼。

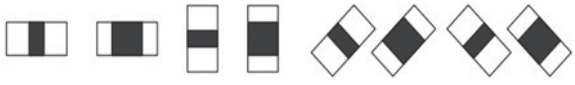
11.2.1 認識 Haar 特徵分類器

Haar 特徵是用來描繪一張圖片。Haar 特徵是一個矩形區域，此矩形可進行旋轉、平移、縮放等，共計有 15 個類型：

▲ 邊緣特徵



▲ 中心特徵



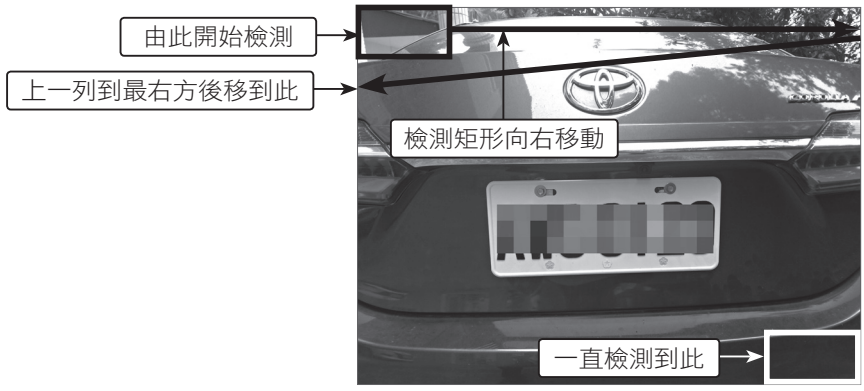
▲ 線性特徵

▲ 對角線特徵

Haar 特徵值反映了圖像的灰度變化情況。例如：臉部的一些特徵能由矩形特徵簡單的描述：眼睛要比臉頰顏色深，鼻樑兩側要比鼻樑顏色深，嘴巴顏色要比周圍顏色深等。Haar 特徵值越大，表示此區域越符合指定的 Haar 特徵。

訓練時，需框選出要偵測的圖形區域，系統就依照框選區域的 Haar 特徵值來建立 Haar 特徵分類器模型，提供的圖形數量越多且形態越多元，建立的模型準確度就會越高。

使用 Haar 特徵分類器模型時，系統會在要偵測的圖片左上角產生一個檢測矩形，檢查此矩形內的圖形是否符合 Haar 特徵分類器模型特徵。接著將此矩形向右移動檢測，到最右方時移到左側下方檢測，直到圖片右下角為止，這樣就可以檢測整張圖片。



11.2.2 處理正樣本及實測圖片

訓練 Haar 特徵分類器模型需要正樣本圖片及負樣本圖片，訓練完成後會產生模型，可用實測圖片 (非訓練圖片) 來測試模型的偵測效果。

「正樣本圖片」及「實測圖片」都是包含要偵測圖形的圖片，以本章要建立的偵測車牌號碼 Haar 特徵分類器模型為例，正樣本及實測圖片就是含有車牌號碼的圖片，例如：



拍攝車牌注意事項

停車場自動車牌辨識系統使用時，業者可以控制攝影機或相機的拍攝條件，如攝影機裝設位置、停車場光線等，如此可大幅提高辨識率。本章車牌圖片是以手機拍攝，拍攝時請注意下列事項以增加訓練效果及辨識率：

- **新舊車牌數量均衡**：目前車牌有兩種型式，舊式車牌為六碼，新式車牌七碼，兩者的長寬比例不同，文字字型也不同，訓練圖片時要包含這兩種車牌。
- **拍攝時手機和車牌平行 (高度相同)**：由於車牌的位置較低，拍攝時手機位置較高時，車牌會呈現梯形，盡量將手機位置和車牌位置等高，如此拍攝的車牌才會呈現矩形。

- **車牌大小適中**：手機距離車牌遠近會影響圖片中車牌的大小，太大或太小常會辨識困難，大小適中即可 (可參考上圖)。
- **充足的自然光線較佳**：拍攝時最好選在晴天的白天戶外場合，避免拍攝室內停車場的車牌。

讀者如果要實作本章範例，可複製書附光碟本章範例 < 原始檔 > 資料夾到硬碟中實作，其中 <carPlate_sr> 資料夾包含 73 張圖片是做為訓練用，<realPlate_sr> 資料夾包含 8 張圖片是做為實測用。

轉換圖片尺寸

手機拍攝圖片的解析度非常大，不適合做為訓練圖片，必須轉換為較小圖片才能進行訓練。下面程式會將圖片轉換為 300x225 像素大小，因為有 2 個資料夾檔案要轉換 (正樣本及實測圖片)，所以將轉換程式寫成函式。

程式碼：resize.py

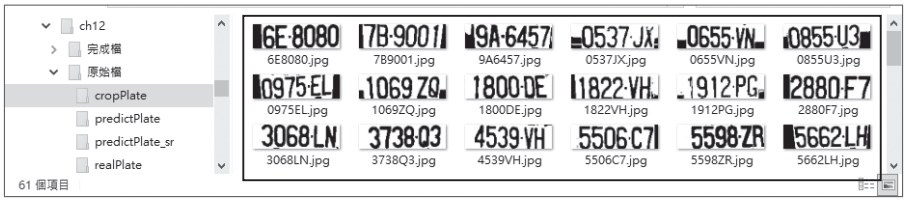
```
def emptydir(dirname):  # 清空資料夾
    if os.path.isdir(dirname):  # 資料夾存在就刪除
        shutil.rmtree(dirname)
        sleep(2)  # 需延遲，否則會出錯
    os.mkdir(dirname)  # 建立資料夾

def dirResize(src, dst):
    myfiles = glob.glob(src + '/*.JPG')  # 讀取資料夾全部 jpg 檔案
    emptydir(dst)
    print(src + ' 資料夾：')
    print('開始轉換圖形尺寸！')
    for i, f in enumerate(myfiles):
        img = Image.open(f)
        img_new = img.resize((300, 225), PIL.Image.ANTIALIAS) # 尺寸 300x225
        outname = str("resizejpg") + str('{:0>3d}').format(i+1) + '.jpg'
        img_new.save(dst + '/' + outname)
    print('轉換圖形尺寸完成！\n')

import PIL
from PIL import Image
import glob
import shutil, os
from time import sleep

```

12.1 專題方向

本專題由於無法取得數以萬計的車牌再進行機器學習的訓練，因此不是對整個車牌進行辨識，而是以前一章建立的車牌號碼偵測模型取得到圖片中車牌號碼的位置後，再將車牌號碼擷取下來，進而分割車牌號碼取得車牌中每一個文字，使用這些車牌號碼文字進行機器學習訓練，建立車牌號碼辨識模型，最後再使用這個模型進行車牌辨識。

專題檢視

首先將車牌圖片的檔案名稱更改為車牌號碼，如此在擷取車牌號碼文字後，可使用檔案名稱文字做為車牌號碼文字的標記。接著撰寫程式 (cropPlate.py) 將車牌號碼圖形擷取下來，車牌號碼圖形檔仍使用車牌號碼做為檔案名稱。



使用 opencv 的輪廓偵測功能可取得車牌號碼中文字輪廓的位置，再分別將文字擷取出來。車牌號碼文字為大寫英文字母及數字，新式車牌英文字母沒有「O」及「I」，因此有 24 個字母及 10 個數字共計 34 個文字，也就是機器學習時分為 34 類。分別以這 34 個文字建立資料夾，將擷取的車牌號碼文字圖形存入對應的資料夾中。

目前的資料量太少，無法進行機器學習訓練，所以要撰寫程式 (makedata.py) 增加資料數量：複製原始圖片，然後在圖片上隨機加入一些雜點，就能產生不同圖片。本專題將各分類圖片擴增到 500 筆左右，全體資料數量有一萬七千多筆。

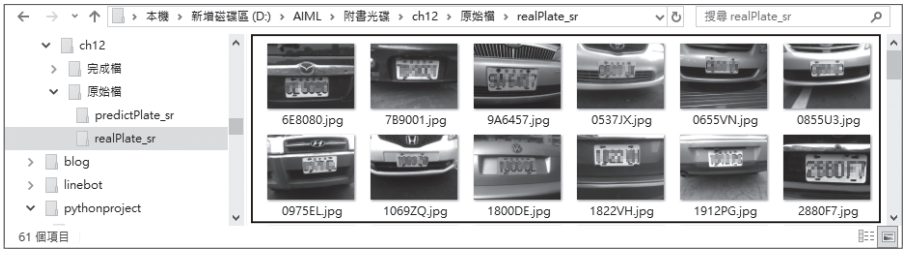
資料準備齊全後就進行機器學習訓練建立模型，然後就可用模型來辨識未知車牌的號碼了！

12.2 車牌號碼機器學習訓練資料

前一章在圖片中偵測車牌號碼位置後，可將車牌號碼擷取下來，進而分別取得車牌號碼每一個文字 (英文字母或數字)，可以使用這些車牌號碼文字建立機器學習訓練資料，以便進行機器學習訓練建立車牌號碼辨識模型。

12.2.1 原始圖片轉換尺寸

請複製書附光碟本章範例到硬碟中進行後續操作：<realPlate_sr> 資料夾含有 61 張數位相機拍攝的相片，用於建立機器學習訓練資料；<predictPlate_sr> 資料夾含有 5 張數位相機拍攝的相片，是讓機器學習模型實際測試的預測車牌。為了方便後續判斷車牌號碼辨識是否正確，所有圖片的檔案名稱已更改為車牌號碼，如此只要看檔案名稱即可得知該圖片的車牌號碼。



<haar_carplate.xml> 是前一章建立的車牌號碼 Haar 特徵分類器模型，可偵測圖片中車牌位置。

首先將所有數位相機拍攝的相片尺寸轉換為 300x225 像素圖形，以便讓 <haar_carplate.xml> 模型偵測。轉換尺寸的程式為：

程式碼：resize.py

```
def emptydir(dirname):  # 清空資料夾
    if os.path.isdir(dirname):  # 資料夾存在就刪除
        shutil.rmtree(dirname)
        sleep(2)  # 需延遲，否則會出錯
    os.mkdir(dirname)  # 建立資料夾

def dirResize(src, dst):
    myfiles = glob.glob(src + '/*.JPG')  # 讀取資料夾全部 jpg 檔案
    emptydir(dst)
    print(src + ' 資料夾：')
```

```
    print('開始轉換圖形尺寸！')
    for f in myfiles:
        fname = f.split("\\")[-1]
        img = Image.open(f)
        img_new = img.resize((300, 225), PIL.Image.ANTIALIAS) #尺寸300x225
        img_new.save(dst + '/' + fname)
    print('轉換圖形尺寸完成！\n')

import PIL
from PIL import Image
import glob
import shutil, os
from time import sleep

dirResize('realPlate_sr', 'realPlate')
dirResize('predictPlate_sr', 'predictPlate')
```

程式說明

- 1-5　emptydir 函式的功能是建立空的資料夾。若資料夾已存在，就先刪除再建立新資料夾。
- 7-17　轉換圖片尺寸函式。
- 8　讀取來源資料夾中所有 jpg 圖片檔。
- 9　建立目的資料夾。
- 12-16　逐一將圖片檔案轉換尺寸。
- 25　轉換訓練圖片，轉換後存於 <realPlate> 資料夾。
- 26　轉換預測用圖片，轉換後存於 <predictPlate> 資料夾。

執行後產生的 <realPlate> 及 <predictPlate> 資料夾中圖片檔案名稱和原來的 <realPlate_sr> 及 <predictPlate_sr> 資料夾完全相同，只是所有圖片的尺寸皆轉換為 300x225 像素。

12.2.2 擷取車牌號碼圖形

利用前一章建立的車牌號碼 Haar 特徵分類器模型 (haar_carplate.xml)，就可框選出車牌號碼，進而將車牌號碼圖形擷取下來。擷取車牌號碼圖形的程式為：

程式碼：cropPlate.py

```
 1 def emptydir(dirname):  #清空資料夾
略……
```