

# Kaggle – Digit Recognizer

關薦勇

29/11/2018

|  |   |
|--|---|
| Part 1. Introduction.....                                  | 1 |
| Part 2. Data preparation.....                              | 1 |
| 2.1 資料處理 .....   | 1 |
| 2.2 標準化與資料形態轉換 .....                                       | 2 |
| 2.3 切割為訓練集 train 與驗證集 val .....                            | 2 |
| 2.4 資料集視覺化與預覽.....   | 2 |
| Part 3. CNN engineering .....                              | 3 |
| 3.1 建立模型 .....   | 3 |
| 3.2 建立 optimizer 與 annealer.....                           | 4 |
| 3.3 數據擴充 .....   | 4 |
| 3.4 擬合模型 .....   | 5 |
| Part 4. Evaluate the mode.....                             | 5 |
| 4.1 訓練集與驗證集曲線 Curve of Training set & Validation set ..... | 5 |
| 4.2 混淆矩陣 Confusion Matrix.....                             | 6 |
| 4.3 上傳 Kaggle.....   | 7 |
| 4.4 結論與改善 .....  | 7 |
| Part 5. Predict different pixel photos .....               | 8 |
| 5.1 圖像讀取預處理 .....  | 8 |
| 5.2 範例.....  | 8 |

## Part 1. Introduction

資料量：訓練集 42000 x 785, 變量 “label” 為因變量 y

測試集 28000 x 784

資料描述：每個圖像的尺寸為 28x28，總共為 784 個像素。

每個像素具有與其相關聯的單個像素值，指示該像素的亮度或暗度，較高的數值意味著較暗。該像素值是 0 到 255 之間的整數，包括 0 和 255。

缺失值：無

資料構成圖像的排列如下圖

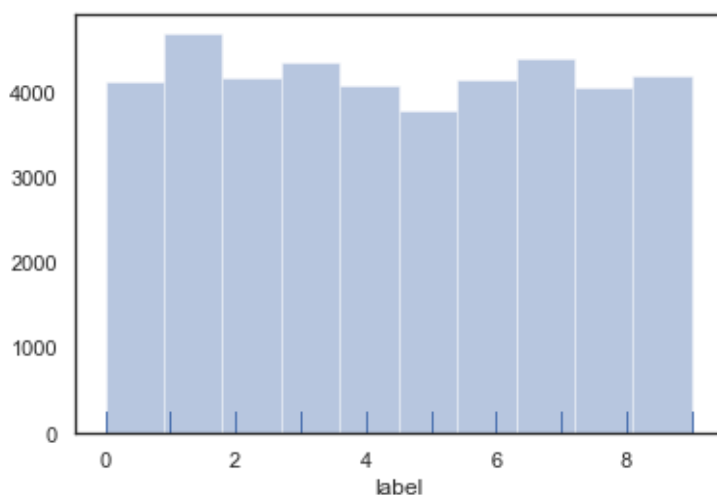
```
000 001 002 003 ... 026 027
028 029 030 031 ... 054 055
056 057 058 059 ... 082 083
|   |   |   | ... |   |
728 729 730 731 ... 754 755
756 757 758 759 ... 782 783
```

(圖像取自 kaggle)

## Part 2. Data preparation

### 2.1 資料處理

- 從訓練集裡提取 y 變量 label
- 觀察訓練集 y 變量的分佈是否均勻，假設為不均勻分佈可能會影響模型的適配性能



| label | count |
|-------|-------|
| 0     | 4132  |
| 1     | 4684  |
| 2     | 4177  |
| 3     | 4351  |
| 4     | 4072  |
| 5     | 3795  |
| 6     | 4137  |
| 7     | 4401  |
| 8     | 4063  |
| 9     | 4188  |

變量 y：0~9 的數值，上圖為每個數值的出現總次數

使用卡方配適度檢定(Pearson's Chi-square Test)檢定變量  $y$  是否分佈均勻

使用方法為 `scipy.stats` 中的 `chisquare` 檢定

```
Power_divergenceResult(statistic=120.45285714285716, pvalue=1.0793226805531035e-21)
```

Statistic 卡方檢定統計量 = 120.4529

p-value = 1.079 > 0.05

不拒絕虛無假設，沒有顯著證據顯示變量  $y$  的分佈不均勻

## 2.2 標準化與資料形態轉換

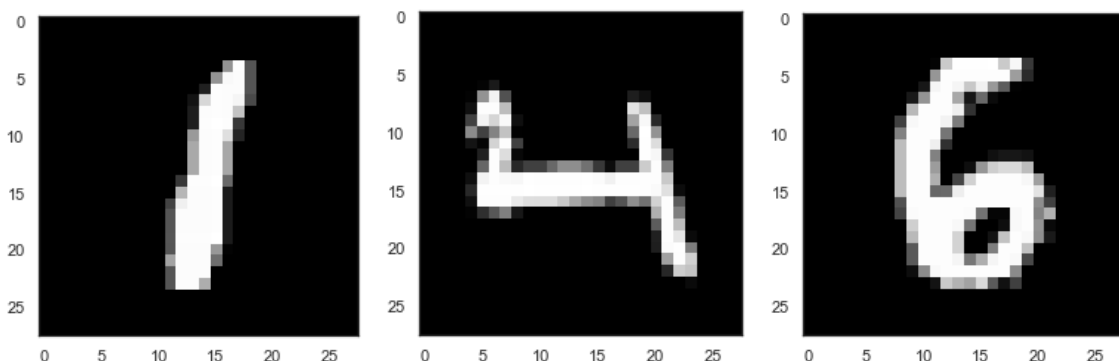
- 為了使得梯度收斂的更快，將訓練集(不包含因變量)與測試集進行標準化  
使用方法為除以 255，使得所有的特征值介於 0 至 1 之間。
- 由於訓練集與測試集中每一個觀察值為 784x1 的矩陣，但照片應為 28x28 的像素排列，故將之轉換為 28x28x1 矩陣
- 訓練集變量  $y$  轉換為虛擬變量  
(ex: 2 -> [0,0,1,0,0,0,0,0,0])

## 2.3 切割為訓練集 train 與驗證集 val

將目前的測試集進一步切割為訓練集與驗證集

( $x_{train}$ ,  $y_{train}$ ) 與 ( $x_{val}$ ,  $y_{val}$ ) 比例為 9 : 1

## 2.4 資料集視覺化與預覽



上述圖像為當前訓練集的前三個觀察值的圖像呈現形態

## Part 3. CNN engineering

### 3.1 建立模型

初始化 CNN 模型

```
classifier=Sequential()
```

- 。選擇建立的架構依序為 3 層特征探測器為 32 的卷積層，3 層特征探測器為 64 的卷積層以及 1 層特征探測器為 128 的卷積層。
- 。第一層卷積層需要額外設立讀取圖像的大小尺寸轉化為為 (28 x 28 x 1) 的像素大小，把載入的圖像像素都統一為相同大小的像素尺寸進行建立模型
- 。並且每一層都會設有一層的 Batch Normalization。
- 。每三層的結束都會添加一層(2x2)最大池化 MaxPool 層以及 Dropout 層(需要丟棄的輸入比例為 0.4)，以避免過度擬合
- 。這些卷積層所使用的激活函數皆為 relu (Rectified Linear Unit)
- 。卷積層建立完成後對模型進行 Flatten 扁平化處理
- 。再來建立一層 Dense 隱藏層，由於要變量 y 的值區間為 0 到 9 (10 個值)，故隱藏層的數量設為 10，激活函數選擇為 softmax

CNN 建構步驟依序(由上至下順序建構)

a.

|          |            |               |           |
|----------|------------|---------------|-----------|
| 卷積層      | 特征探測器：32   | 特征探測器大小：(3,3) | 激活函數：relu |
| 批標準化     |            |               |           |
| 卷積層      | 特征探測器：32   | 特征探測器大小：(3,3) | 激活函數：relu |
| 批標準化     |            |               |           |
| 卷積層      | 特征探測器：32   | 特征探測器大小：(3,3) | 激活函數：relu |
| 批標準化     |            |               |           |
| 最大池化     | 池化尺寸：(2,2) |               |           |
| Droupout | 丟棄比例：0.4   |               |           |

b.

|      |          |               |           |
|------|----------|---------------|-----------|
| 卷積層  | 特征探測器：64 | 特征探測器大小：(3,3) | 激活函數：relu |
| 批標準化 |          |               |           |
| 卷積層  | 特征探測器：64 | 特征探測器大小：(3,3) | 激活函數：relu |

|         |            |                    |           |
|---------|------------|--------------------|-----------|
| 批標準化    |            |                    |           |
| 卷積層     | 特征探測器：64   | 特征探測器大小：(3,3)      | 激活函數：relu |
| 批標準化    |            |                    |           |
| 最大池化    | 池化尺寸：(2,2) | strides 卷積步長：(2,2) |           |
| Drouput | 丟棄比例：0.4   |                    |           |

c.

|      |           |               |           |
|------|-----------|---------------|-----------|
| 卷積層  | 特征探測器：128 | 特征探測器大小：(4,4) | 激活函數：relu |
| 批標準化 |           |               |           |

d.

|         |          |              |  |
|---------|----------|--------------|--|
| 扁平層     |          |              |  |
| Drouput | 丟棄比例：0.4 |              |  |
| 隱藏層     | 數量：10    | 激活函數：softmax |  |

## 3.2 建立 optimizer 與 annealer

e. 優化器：RMSprop，參數為 keras.optimizers.RMSprop 默認參數

參數為：lr 學習率：0.001

rho RMSProp 梯度平方的移动均值的衰减率：0.9

epsilon 模糊因子：None

decay 每次参数更新后学习率衰减值：0.0

f. 編譯模型：loss = "categorical\_crossentropy"

metrics=["accuracy"]

h. 学习速率定时器

```
annealer = LearningRateScheduler(lambda x: 1e-3 * 0.95 ** x)
```

## 3.3 數據擴充

g. 通過實時數據增強生成張量圖像數據批次：

```
keras.preprocessing.image.ImageDataGenerator()
```

|        |                    |      |
|--------|--------------------|------|
| 隨機縮放範圍 | zoom_range         | 0.15 |
| 上下位置平移 | height_shift_range | 0.1  |
| 水平位置平移 | width_shift_range  | 0.1  |

|           |                 |       |
|-----------|-----------------|-------|
| 隨機旋轉的度數範圍 | rotation_range  | 15    |
| 隨機水平翻轉    | horizontal_flip | False |
| 隨機垂直翻轉    | vertical_flip   | False |

### 3.4 擬合模型

`classifier.fit_generator()`

|                          |                 |                |
|--------------------------|-----------------|----------------|
| 指定進行梯度下降時每個 batch 包含的樣本數 | batch_size      | 64             |
| 訓練終止時的 epoch 值           | epochs          | 60             |
| 指定驗證集                    | validation_data | (x 驗證集, y 驗證集) |
| 一個 epoch 包含的步數           | steps_per_epoch |                |
| 日志顯示                     | verbose         | 2              |
| 回調函數                     | callbacks       | [annealer]     |

## Part 4. Evaluate the mode

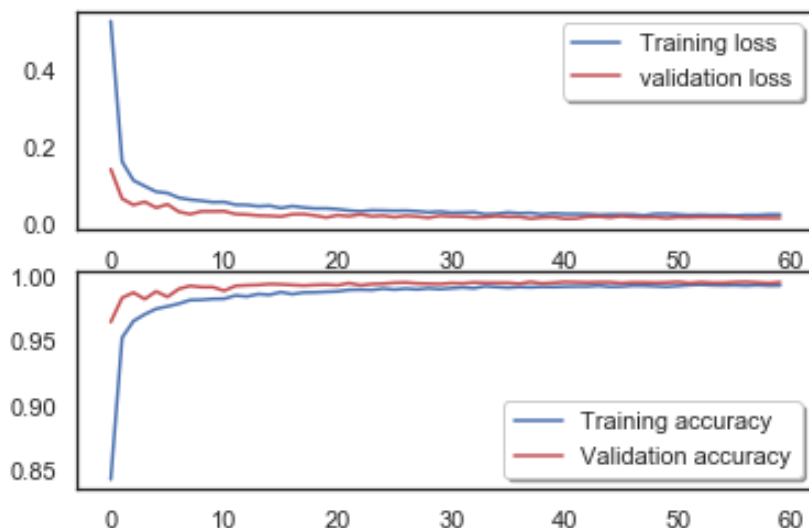
### 4.1 訓練集與驗證集曲線 Curve of Training set & Validation set

\*\*下列為第 55 至 60 Epoch 的訓練結果\*\*

```
Epoch 55/60
- 453s - loss: 0.0221 - acc: 0.9934 - val_loss: 0.0167 - val_acc: 0.9955
Epoch 56/60
- 426s - loss: 0.0208 - acc: 0.9936 - val_loss: 0.0165 - val_acc: 0.9960
Epoch 57/60
- 433s - loss: 0.0222 - acc: 0.9933 - val_loss: 0.0143 - val_acc: 0.9962
Epoch 58/60
- 478s - loss: 0.0220 - acc: 0.9937 - val_loss: 0.0148 - val_acc: 0.9957
Epoch 59/60
- 452s - loss: 0.0237 - acc: 0.9932 - val_loss: 0.0145 - val_acc: 0.9952
Epoch 60/60
- 459s - loss: 0.0236 - acc: 0.9934 - val_loss: 0.0142 - val_acc: 0.9960
```

在 Epoch = 60 之後，模型在驗證集上達到 99.6%的準確率

訓練時 Validation set 的準確度都一直保持著大於 Training set 的準確度，說明著這個模型不會有過度訓練 Training set 的情況



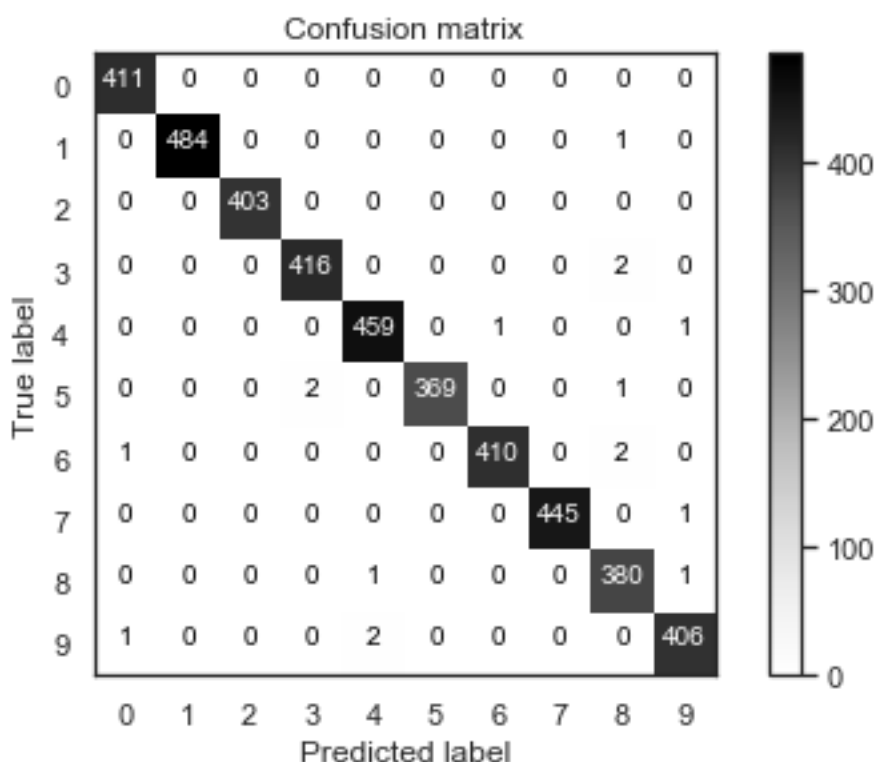
上圖為 loss 誤差曲線  
(y 軸的值越低越好)

下圖為  
accuracy 準確率曲線  
(y 軸的值越高越好)

在 Epoch = 1 開始, Validation set 的訓練曲線就已經優於 Training set

在 Epoch = 10 開始, Validation set 的訓練曲線走向已經非常趨近於 Training set 的訓練曲線

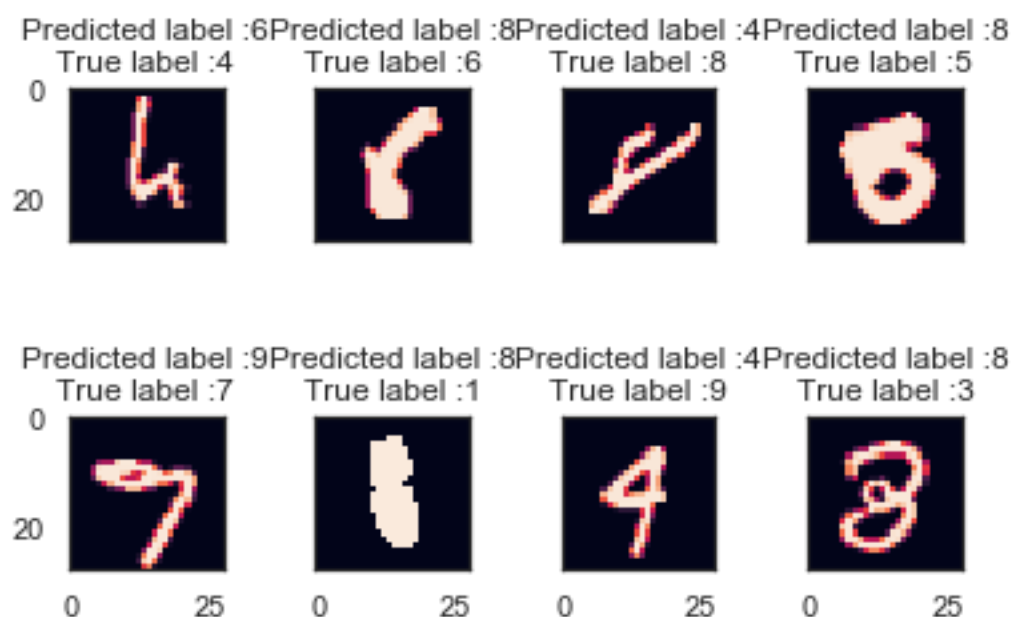
## 4.2 混淆矩阵 Confusion Matrix



從混淆矩阵圖中可以觀察出數字 0 與數字 2 的預測率在驗證集上準確率達到了 100%

其中 8 被預測錯誤的次數最多, 有 2 次被誤認為 6 與有 2 次被誤認為 3

下圖為一些預測錯誤的值



### 4.3 上傳 Kaggle

評分方法：根據預測的分類準確性（您獲得的圖像百分比）進行評估

獲得分數為：0.99671

排名：239/2705 (Top 9%)

時間：2018/12/2 03:25

### 4.4 結論與改善

從混淆矩陣可看出這個模型容易把 3 或者 6 誤認為 8，所以本人初步認為可以通過調整權重的方式讓被識別為 8 的門檻高一些來改善模型的預測能力。但凡事都有兩面性，預測判斷為 8 的門檻高了有可能會導致一些真實值為 8 但由於調整了權重問題導致這些值被誤認為其他值。有可能模型的預測能力反而降低。



## Part 5. Predict different pixel photos

### 5.1 圖像讀取預處理

Step1. 讀取圖像

Step2. 將圖像轉換為灰色圖像

(0 表示黑，255 表示白，其他数字表示不同的灰度)

Step3. 將圖像轉換為 28x28 像素尺寸的圖像

Step4. 並對圖像依照 2.2 標準化與資料形態轉換 進行轉換

Step5. 進行預測

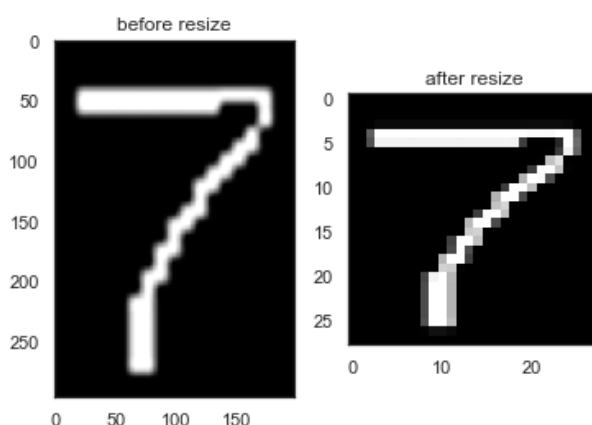
### 5.2 範例

使用的 package

```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
```



左圖為原始圖像  
圖像數值：7  
圖像類型：RGBA  
像素：296x198x4 pixel



Predict result: 7

模型預測為 7，預測值與真實值相同

程式碼會附在 [github](#) 裡

注意：這個模型由於訓練的所有圖像皆為背景為黑色，字體顏色為灰白色。

所以只能用來預測相同色彩構架的圖像，如白底黑體的圖像會預測錯誤