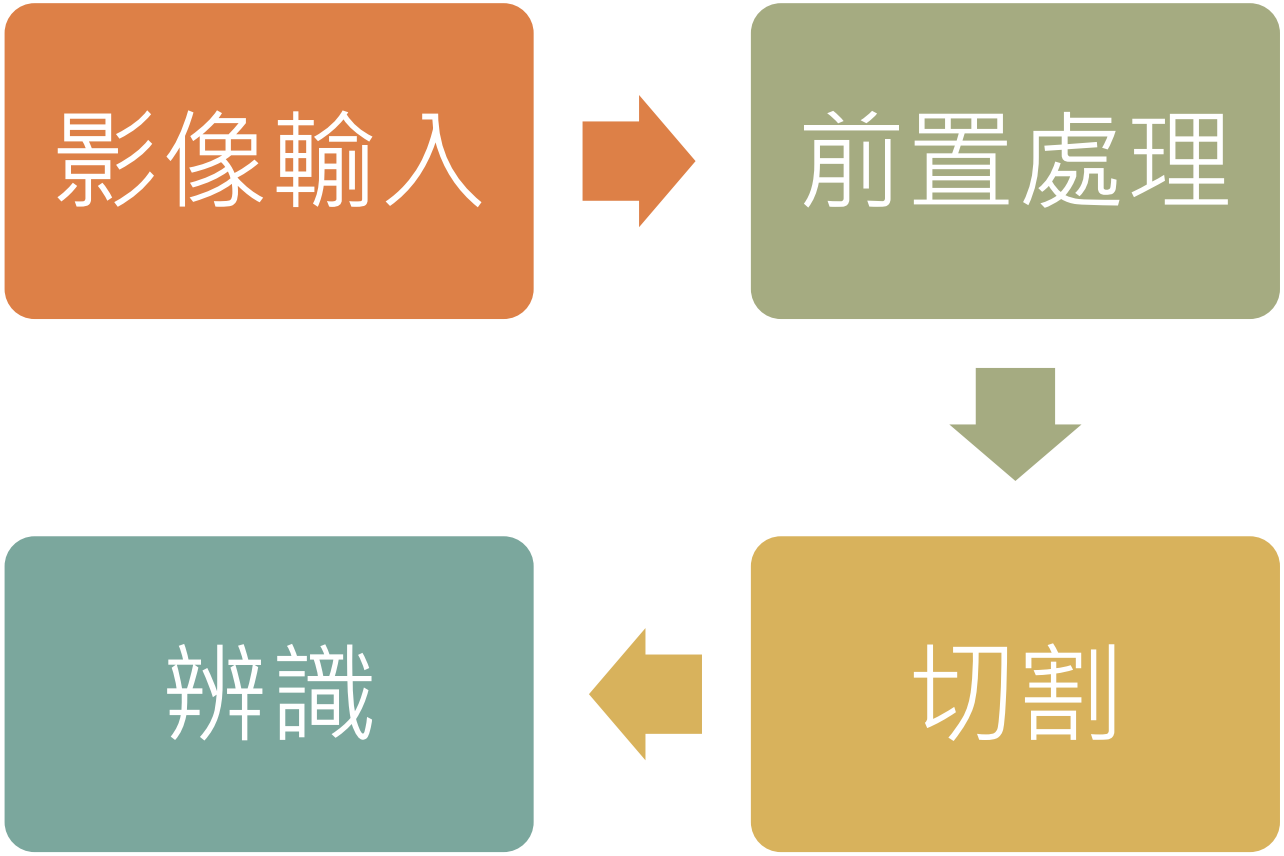
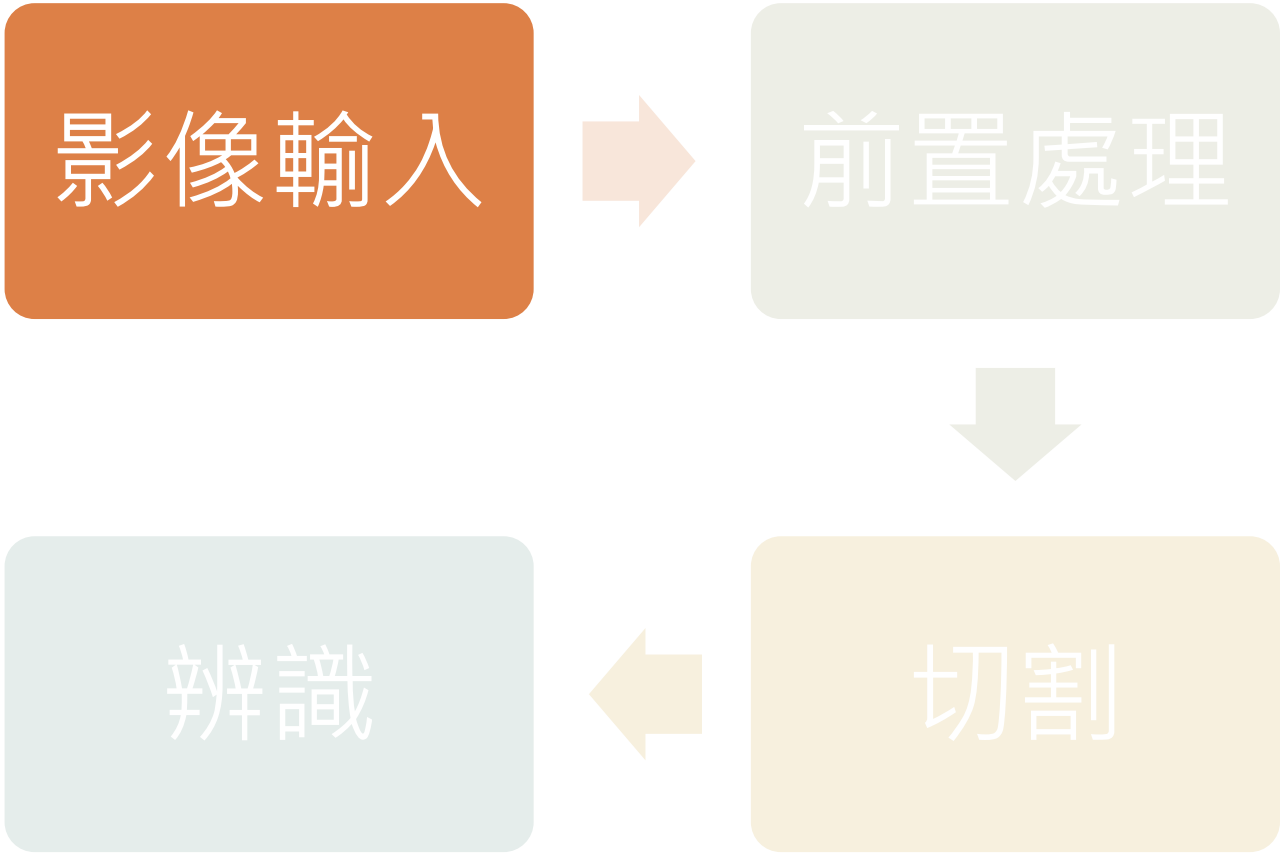
影像辨識
--以Python為例

106.3.1 古佳怡

流程



流程

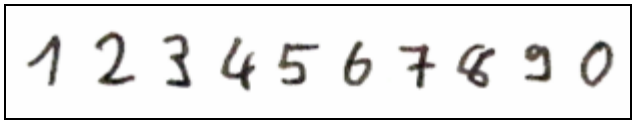


影像輸入

觀察影像特性，選擇適合的處理方法

有雜訊的 ➔ 前置處理

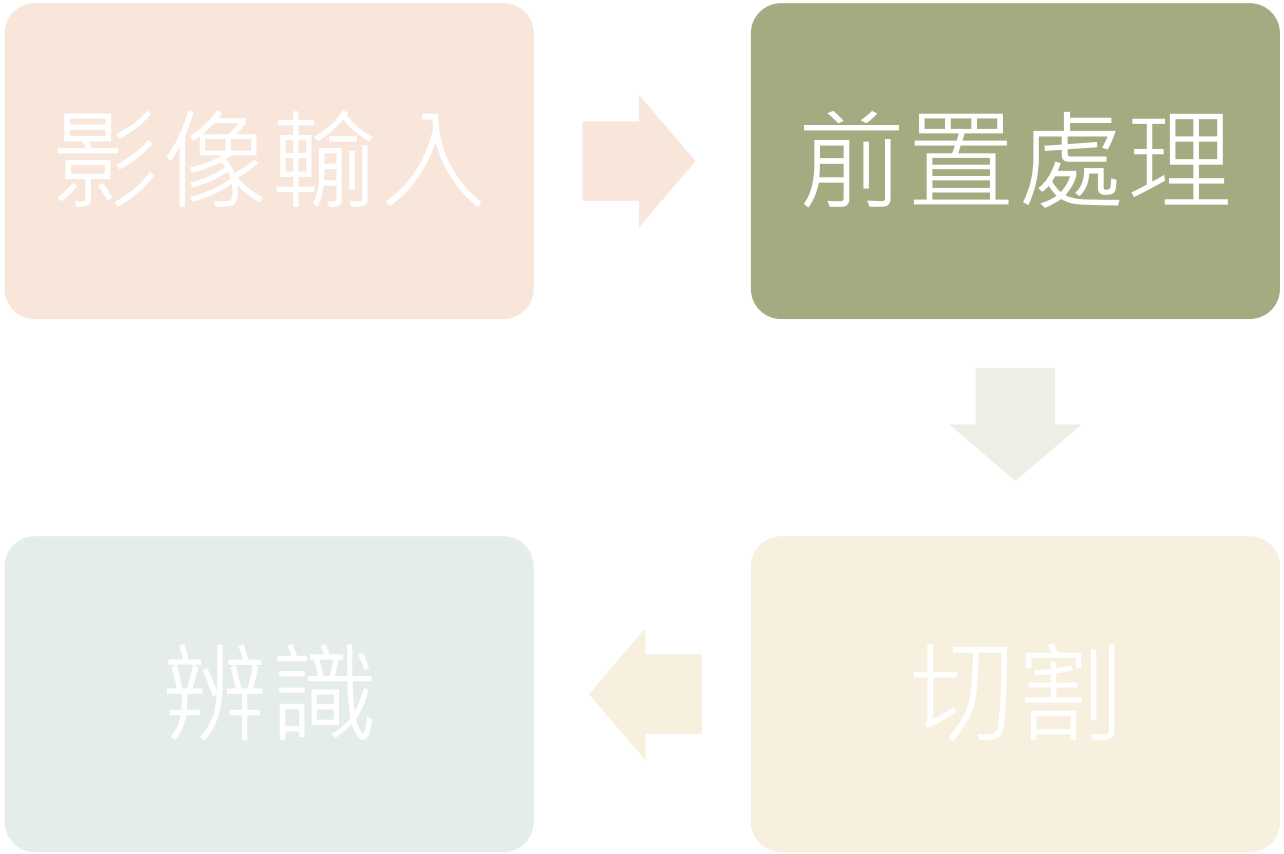
有傾斜角度的 ➔ 切割後轉正



字體特別的 ➔ 適合的dataset

當然還有更多.....

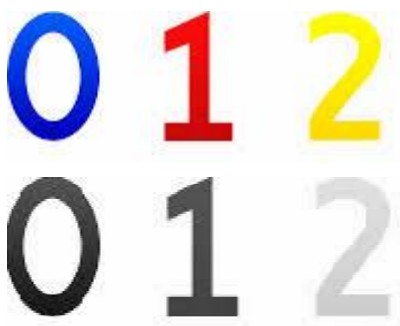
流程



前置處理

只對文字內容感興趣，
色彩不應影響辨識結果

灰階化



一、調高亮度與對比
二、設立門檻值

➔去除黯淡者

去雜訊

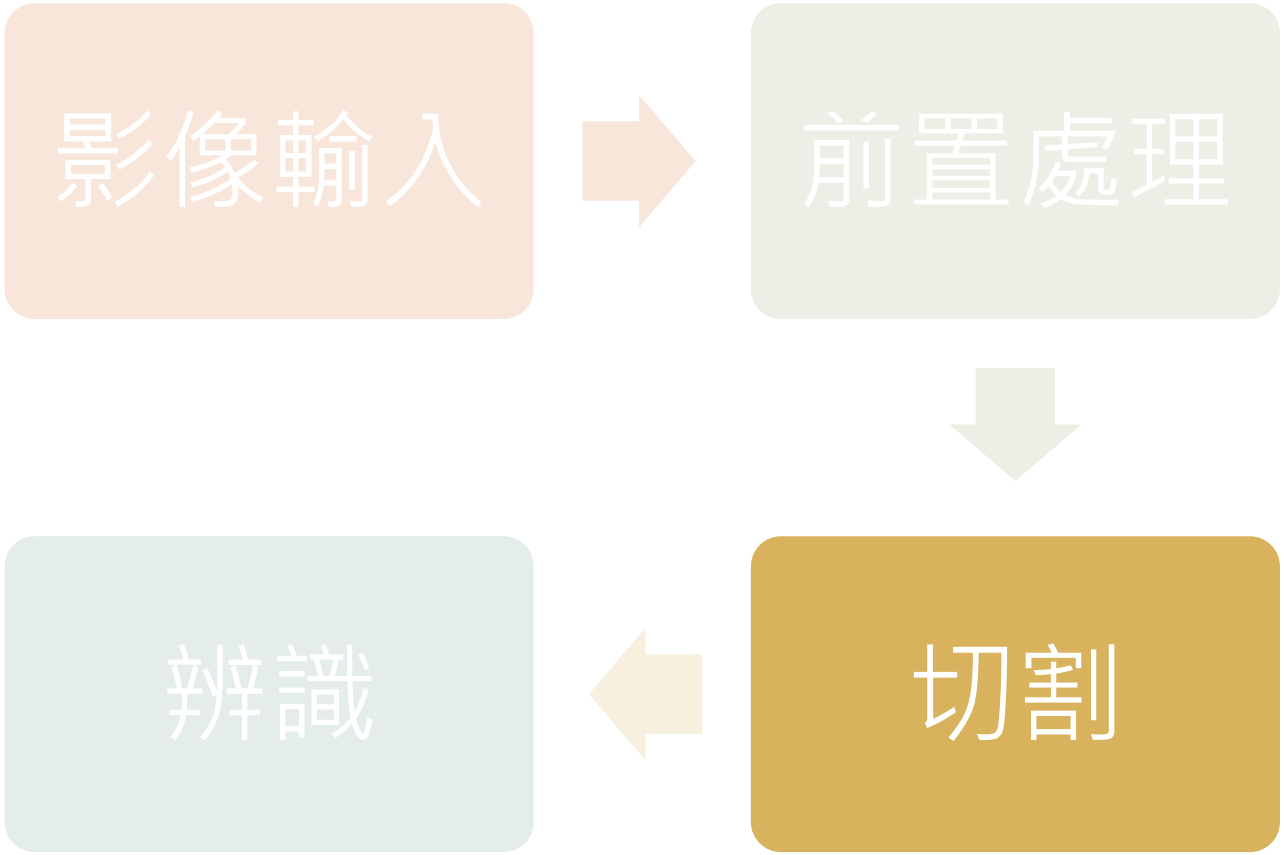
三、濾波器
四、侵蝕(就像是橡皮擦)

➔去除小噪點

補點

膨脹，以補滿或連接
去雜訊所導致的缺口

流程

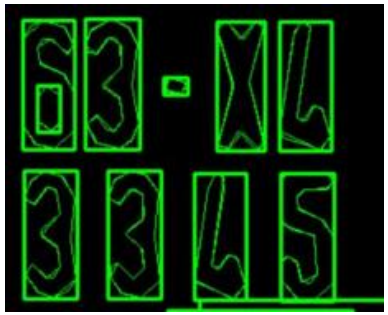


切割

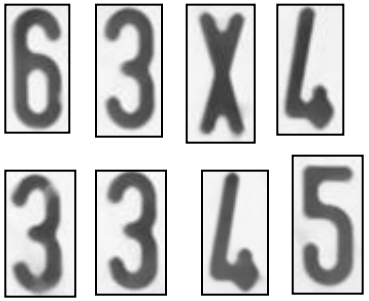
也可以找輪廓時，
直接找四點構成的輪廓

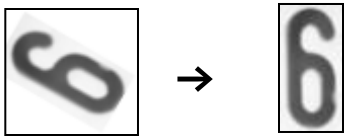
用長方形包裹輪廓

找輪廓 → 包裹 → 擷取 → 正規化



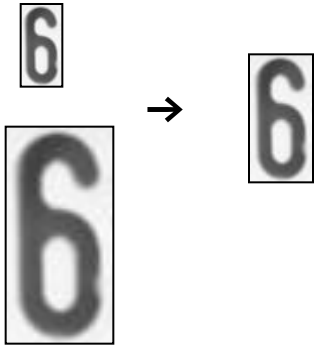
設立大小門檻值，
以擷取適當長方形



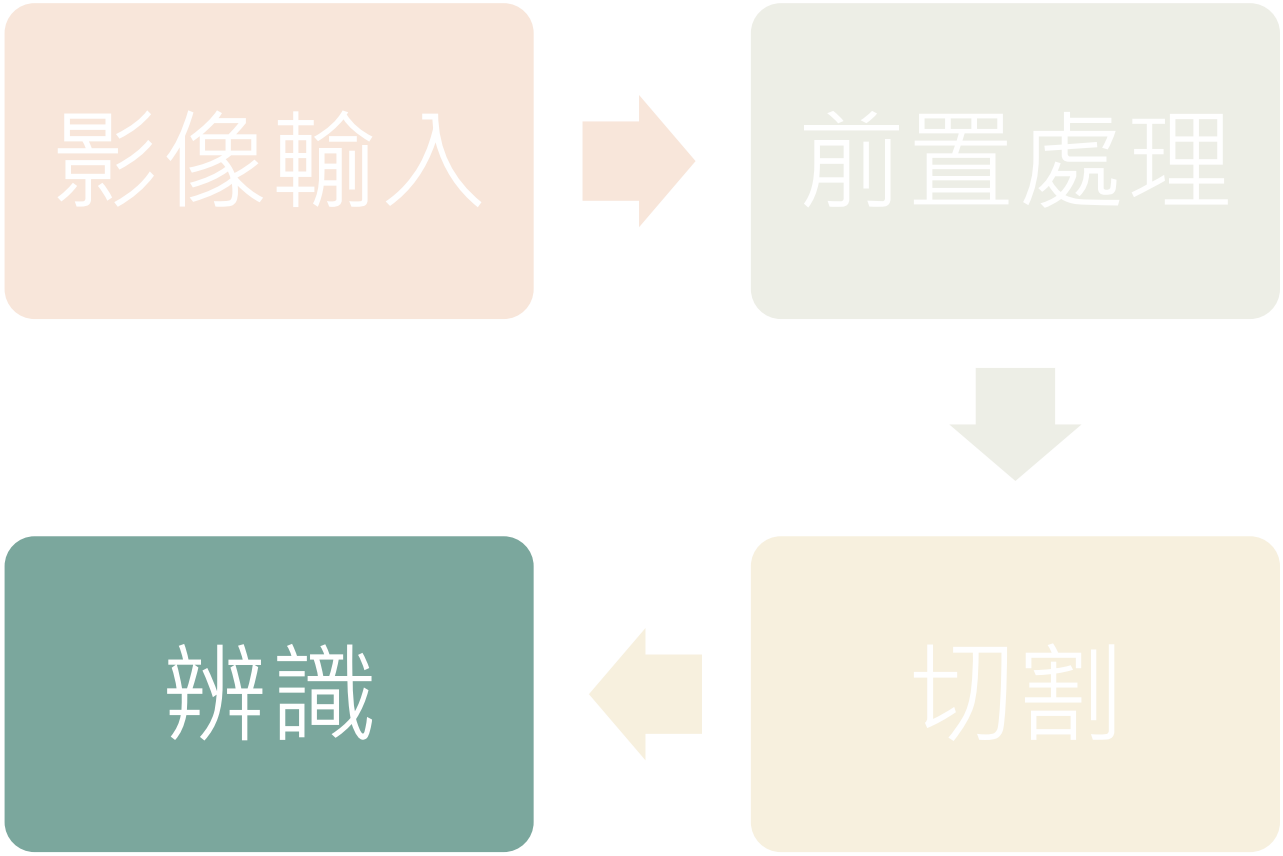


傾斜轉正
即找旋轉後
最窄的長方形
(最左和最右點
距離最短者)

大小統一
因辨識時點對點



流程



DataSet(1)

手寫體，共 5000 張

註：是黑底白字!!!

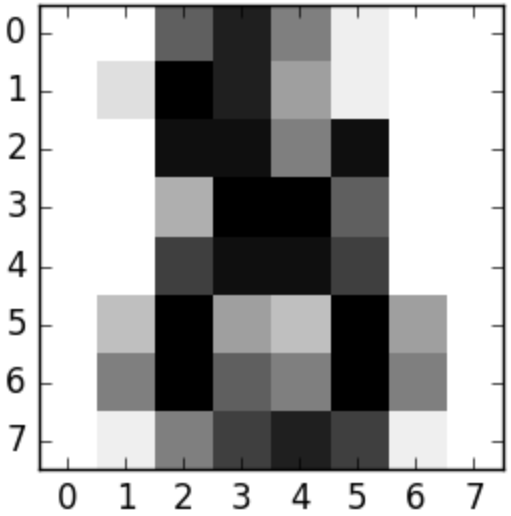
OpenCV資料夾所附的digits.png

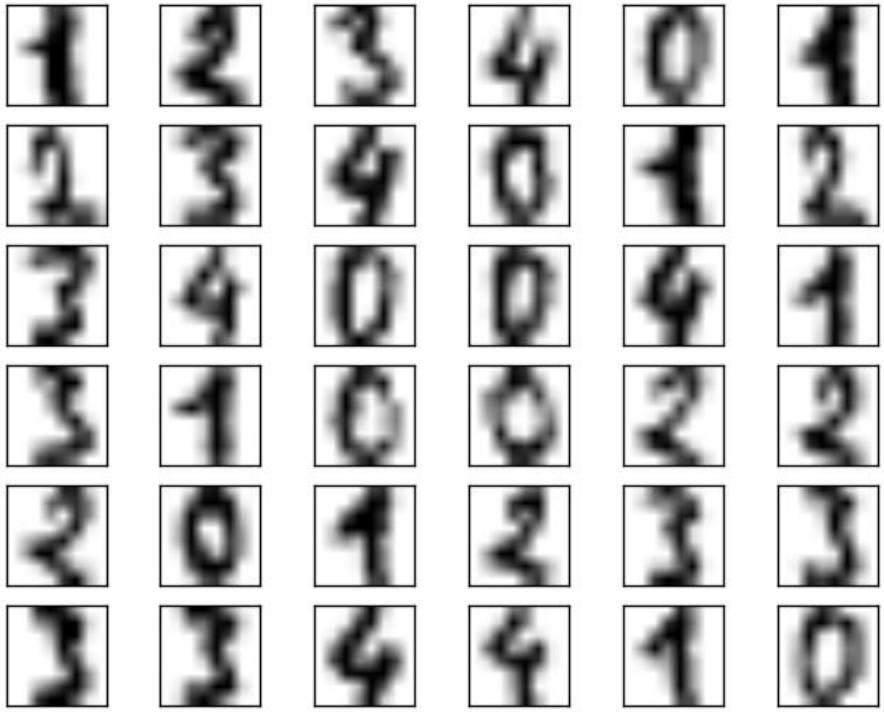
20 x 20 pixels

1000 x 2000 pixels

DataSet(2)

sklearn的**手寫數字資料庫**

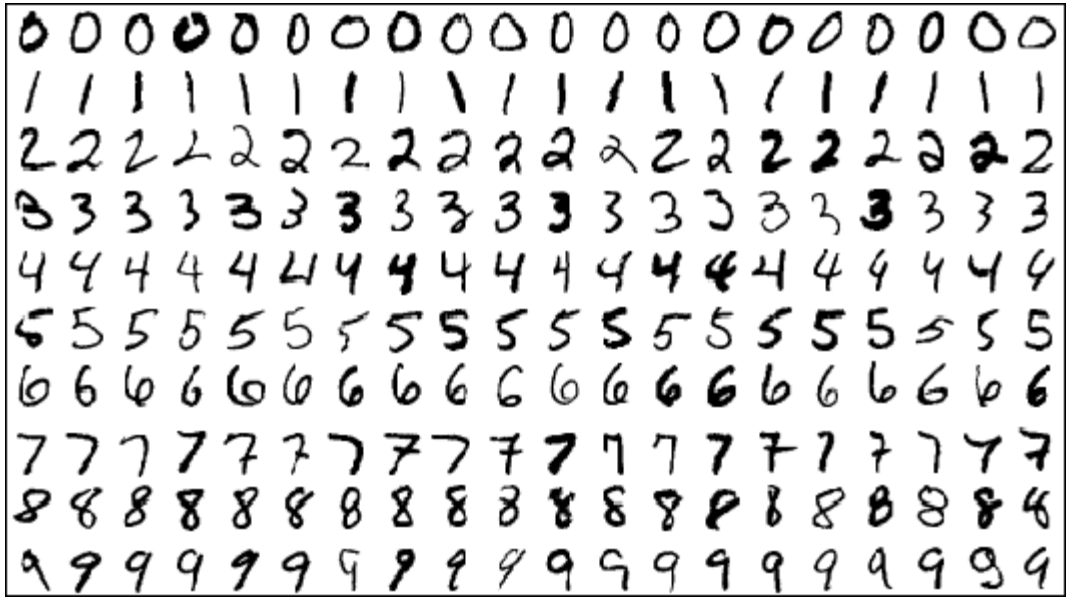




手寫體，1998年建立，筆跡來自43人，**共 1797 張**
一開始為 32x32 pixels，後來經運算為 **8x8 pixels**，
灰階記錄的範圍為 **0~16的整數**。

DataSet(3)

MNIST的**手寫數字資料庫**



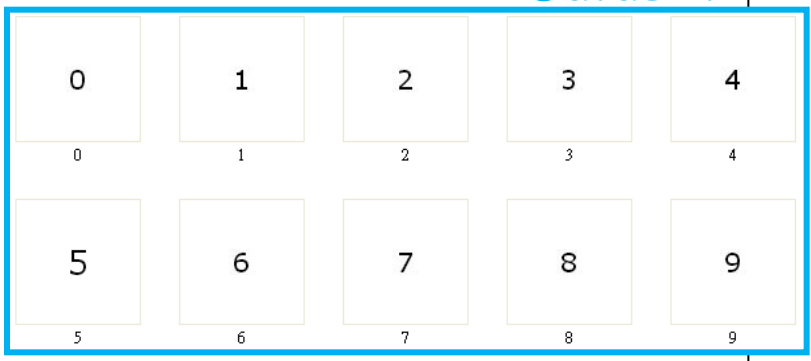
手寫體，
共 60000 張訓練圖片、10000 張測試圖片
28x28 pixels

方法一、找出最像的(極簡版)

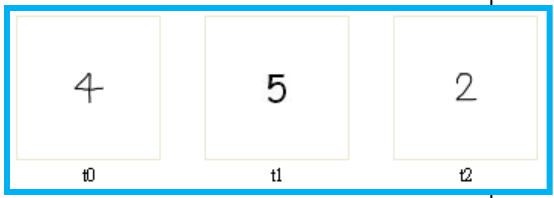

```
# coding=UTF-8
import cv2
import numpy as np
import os

def mse(imageA, imageB):
        err = np.sum((imageA.astype("float") - imageB.astype("float")) ** 2)
        err /= float(imageA.shape[0] * imageA.shape[1])
        return err

def getNumber(pic):
        min_a = 9999999999
        min_png = None
        for png in os.listdir("data"):
                ref = cv2.imread("data/"+png)
                if mse(ref, pic) < min_a:
                        min_a = mse(ref, pic)
                        min_png = png
        return min_png, min_a

pic0 = cv2.imread("t0.jpg")
print getNumber(pic0)
```

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2$$

DataSet



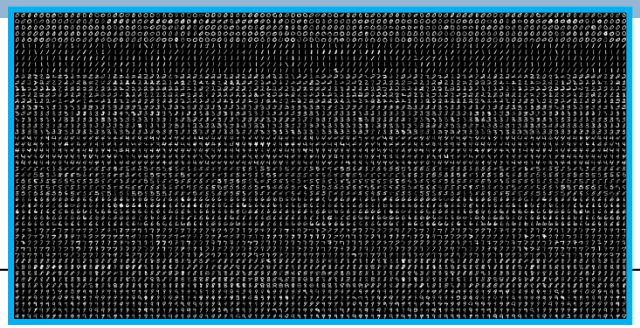
TestData



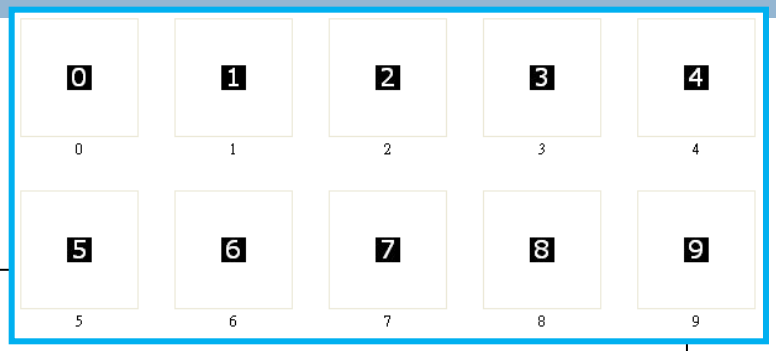

```
('3.jpg', 26959.695)
('5.jpg', 15617.047500000001)
('2.jpg', 20768.189999999999)
```

方法一、找出最像的(進階版)

DataSet

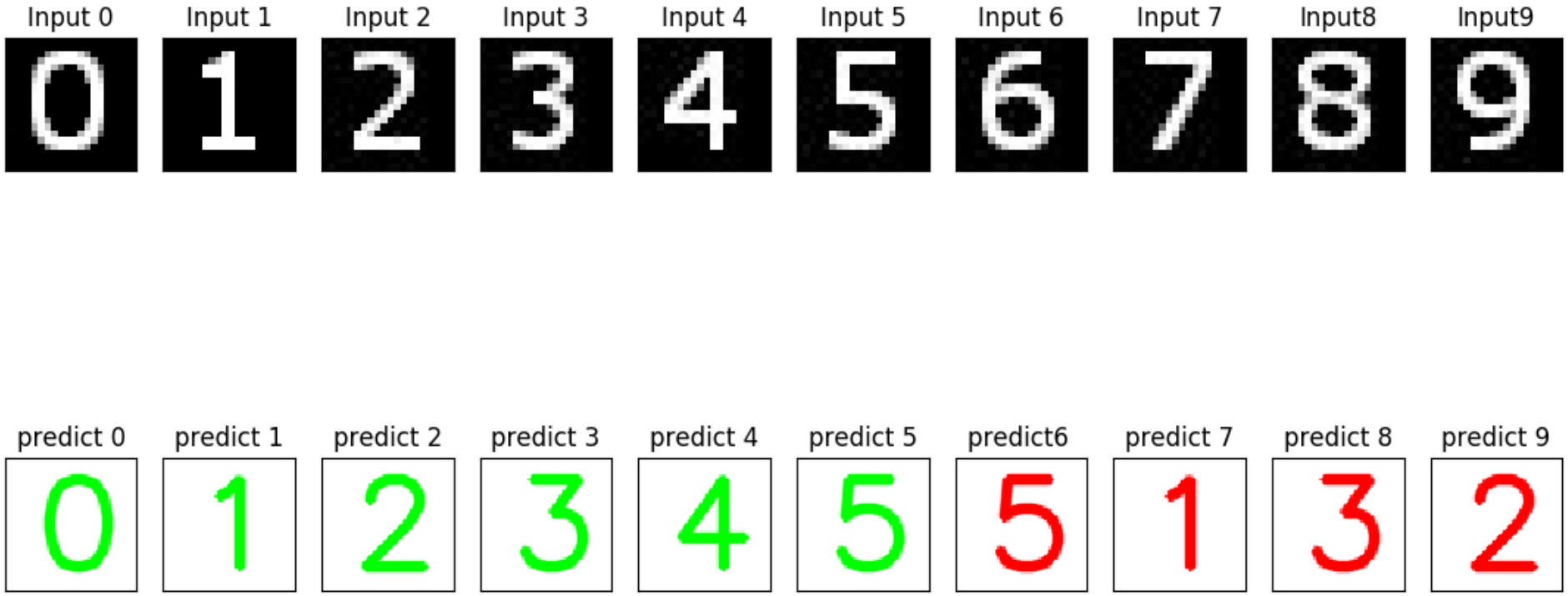


TestData

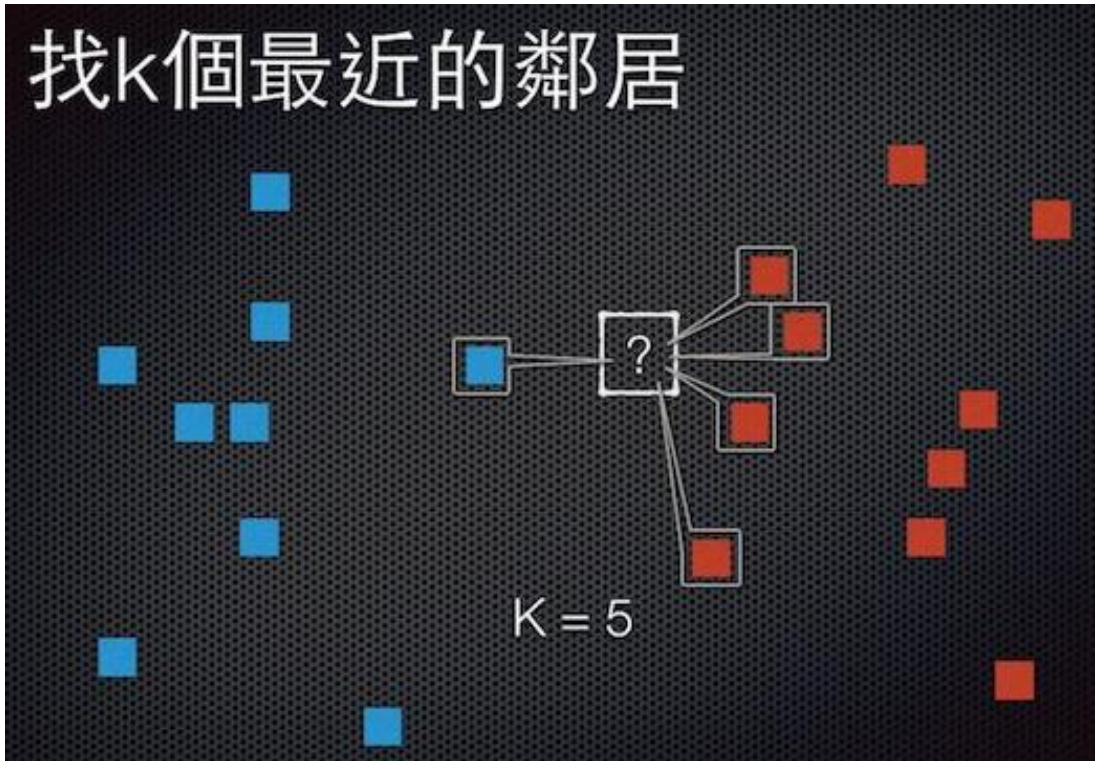



```
# 節錄部份
def getNumber(pic):
        min_a = 9999999999
        min_png = None
        gray = cv2.imread('digits.png',0)
        x = [np.hsplit(row,100) for row in np.vsplit(gray,50)] # 切割
        label = 0
        for row in x:
                for ref in row:
                        if mse(ref, pic) < min_a:
                                min_a = mse(ref, pic)
                                min_png = int(label/5)
                label = label + 1
        return min_png, min_a
```

方法一、找出最像的(進階版)



方法二、K-Nearest Neighbors(KNN)



如果最近的是藍色，但次四近的是紅色?

➔少數服從多數，避免異常值(outlier)

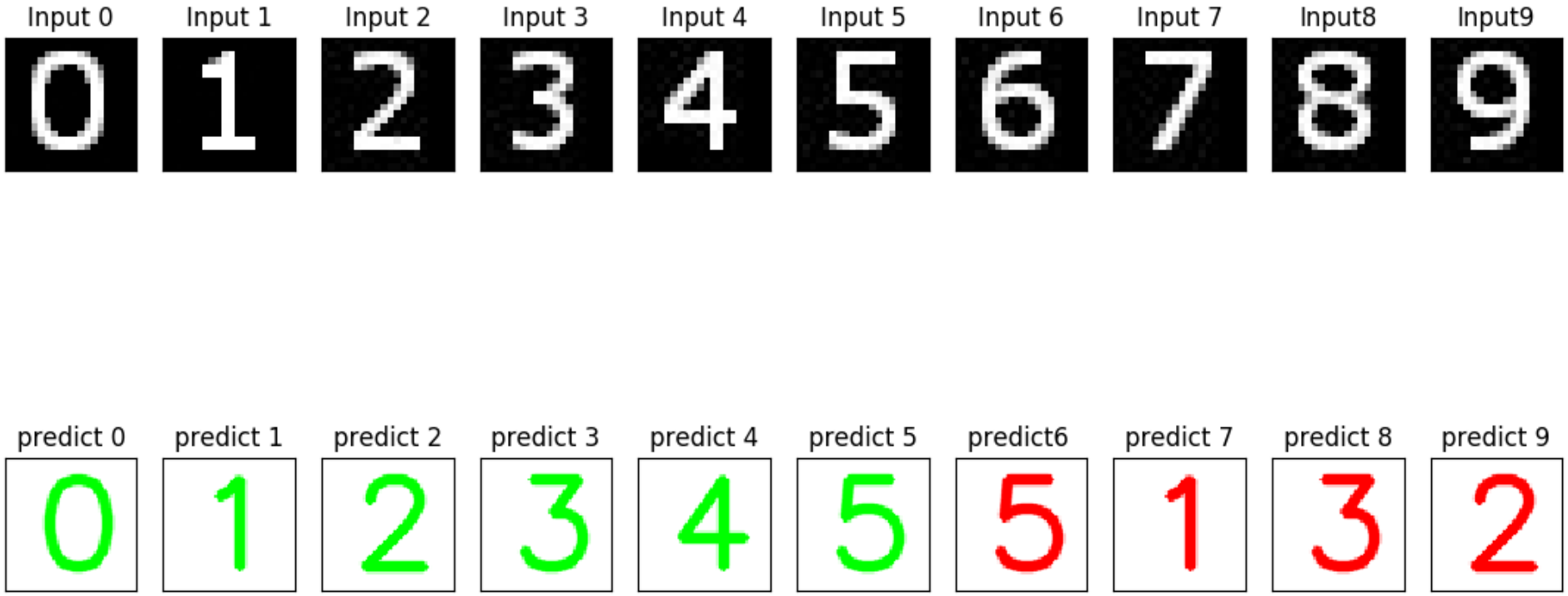
方法二、K-Nearest Neighbors(KNN)


```
# 節錄部份
gray = cv2.imread('digits.png',0)
x = [np.hsplit(row,100) for row in np.vsplit(gray,50)]
x = np.array(x)

train = x[:,:50].reshape(-1,400).astype(np.float32) # Size = (2500,400)
test = x[:,50:100].reshape(-1,400).astype(np.float32) # Size = (2500,400)

# Create labels for train and test data
k = np.arange(10)
train_labels = np.repeat(k,250)[:,np.newaxis]
test_labels = train_labels.copy()

# Initiate kNN, train the data, then test it with test data for k=7
knn = cv2.ml.KNearest_create()
knn.train(train,cv2.ml.ROW_SAMPLE,train_labels)
ret, result, neighbours, dist = knn.findNearest(test, k=7)

# Now we check the accuracy of classification
matches = result==test_labels
correct = np.count_nonzero(matches)
accuracy = correct*100.0/result.size
print (accuracy)
```

K值設多少?

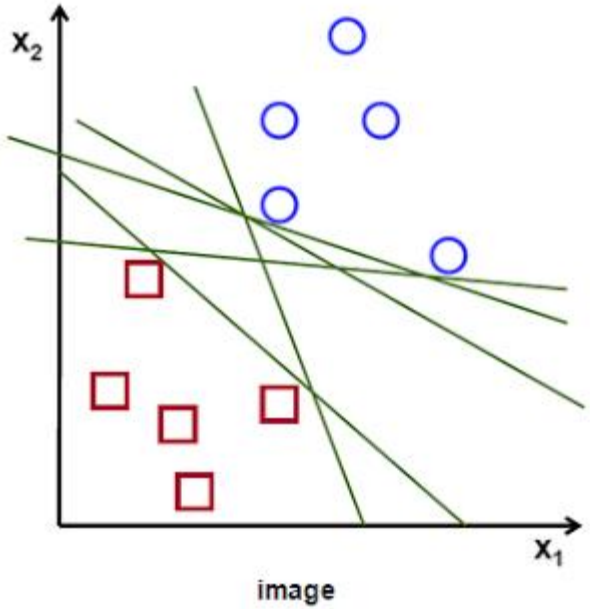
91.44

方法二、K-Nearest Neighbors(KNN)



方法三、Support Vector Machine(SVM)

KNN每筆測資都要重新計算和所有圖片的距離，太耗時...



image

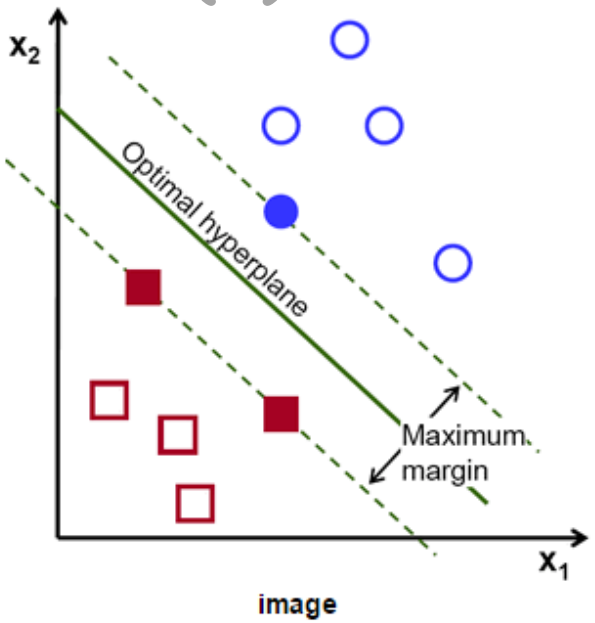
找出區分不同類別的

$f(x) = ax + b$

➔當有測資時，就可以直接用它

方法三、Support Vector Machine(SVM)

但是要找出f(x)，好像還是有點麻煩?



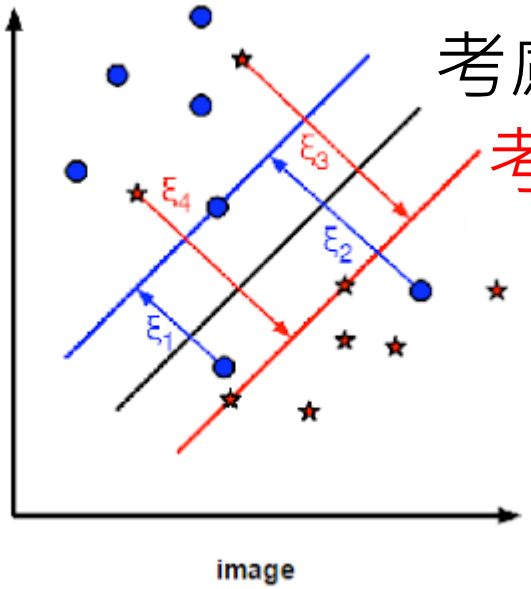
image

實際上只要考慮彼此決定性的點(靠近邊界的)就好

➔找出距離這些點最遠的f(x)，以劃分鑑別度

方法三、Support Vector Machine(SVM)

f(x)距離最遠就足夠了嗎? 如果有錯誤發生?



f(x)不只考慮距離，也需要考慮是錯的最少的

➔用參數C (0≤x≤1)作為後者比重，在兩者取捨

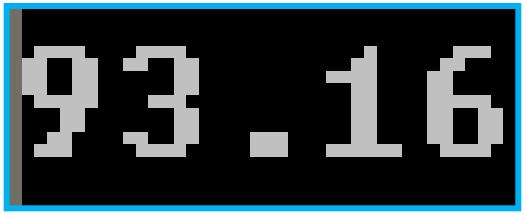
前者適合雜訊大、錯誤本來就多、錯誤代價低時

後者適合雜訊少、錯誤不應該多、錯誤代價高時

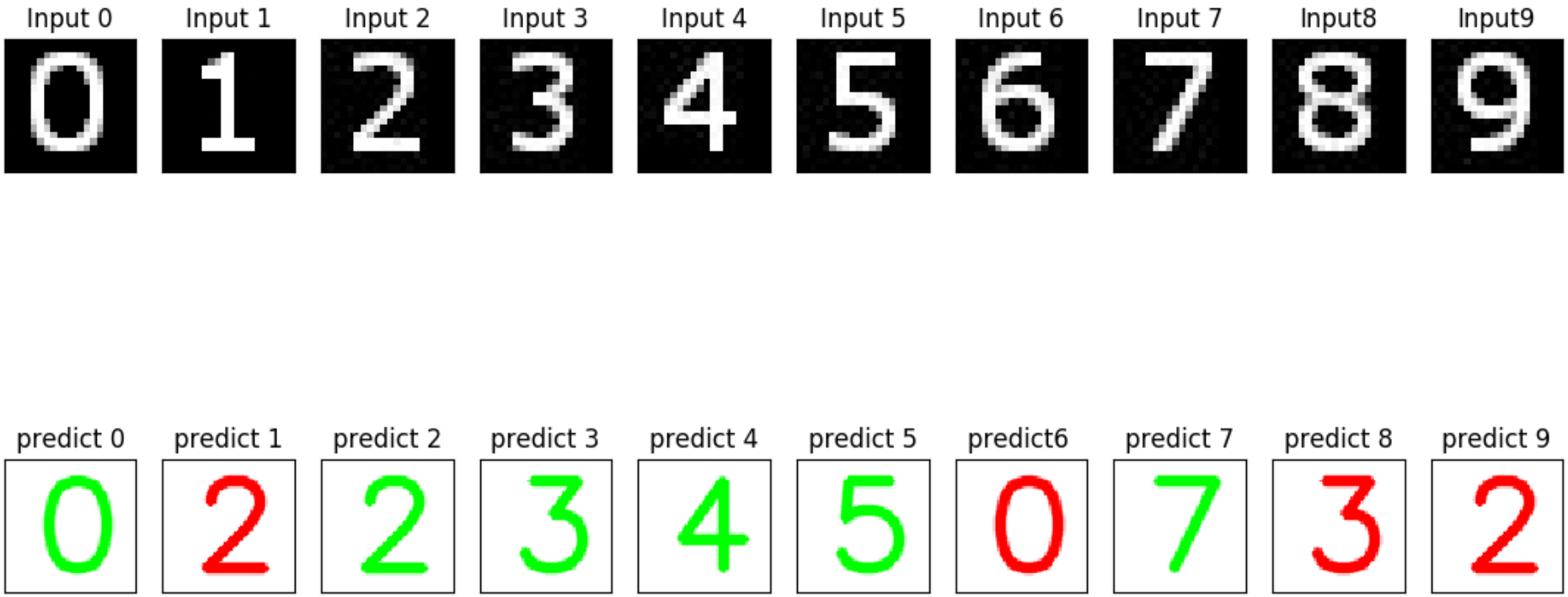
方法三、Support Vector Machine(SVM)


```
# 節錄部份
SVM = cv2.ml.SVM_create()
# SVM.setC(0)
SVM.setType(cv2.ml.SVM_C_SVC)
SVM.setKernel(cv2.ml.SVM_LINEAR)
SVM.setTermCriteria((cv2.TERM_CRITERIA_COUNT, 100, 1.e-06))
SVM.train(traindata, cv2.ml.ROW_SAMPLE, responses)

# 省略
result = SVM.predict(testData)
# Now we check the accuracy of classification
matches = result[1].astype(np.int32)==test_labels
correct = np.count_nonzero(matches)
accuracy = correct*100.0/len(result[1])
print (accuracy)
```

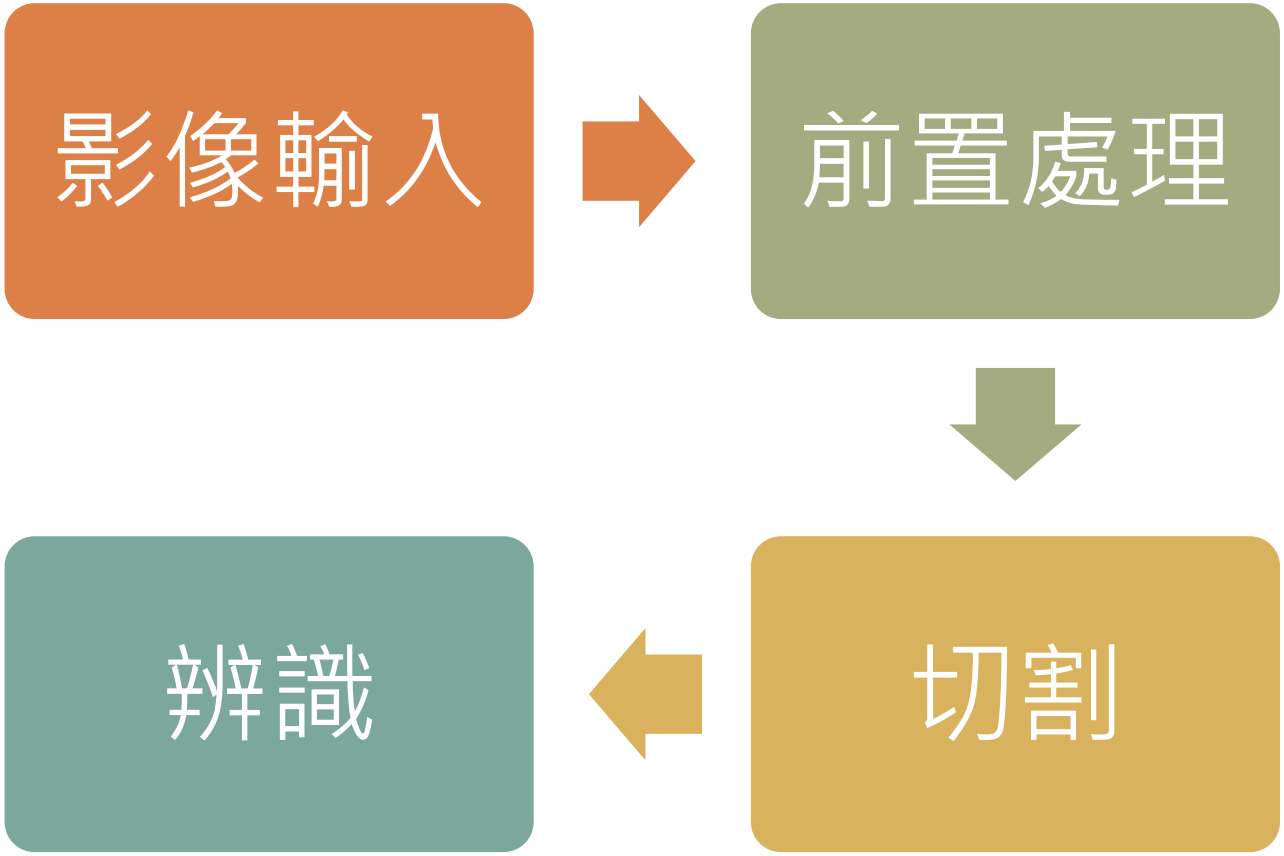
C值設多少?
Kernel(f(x))的型態?



方法三、Support Vector Machine(SVM)

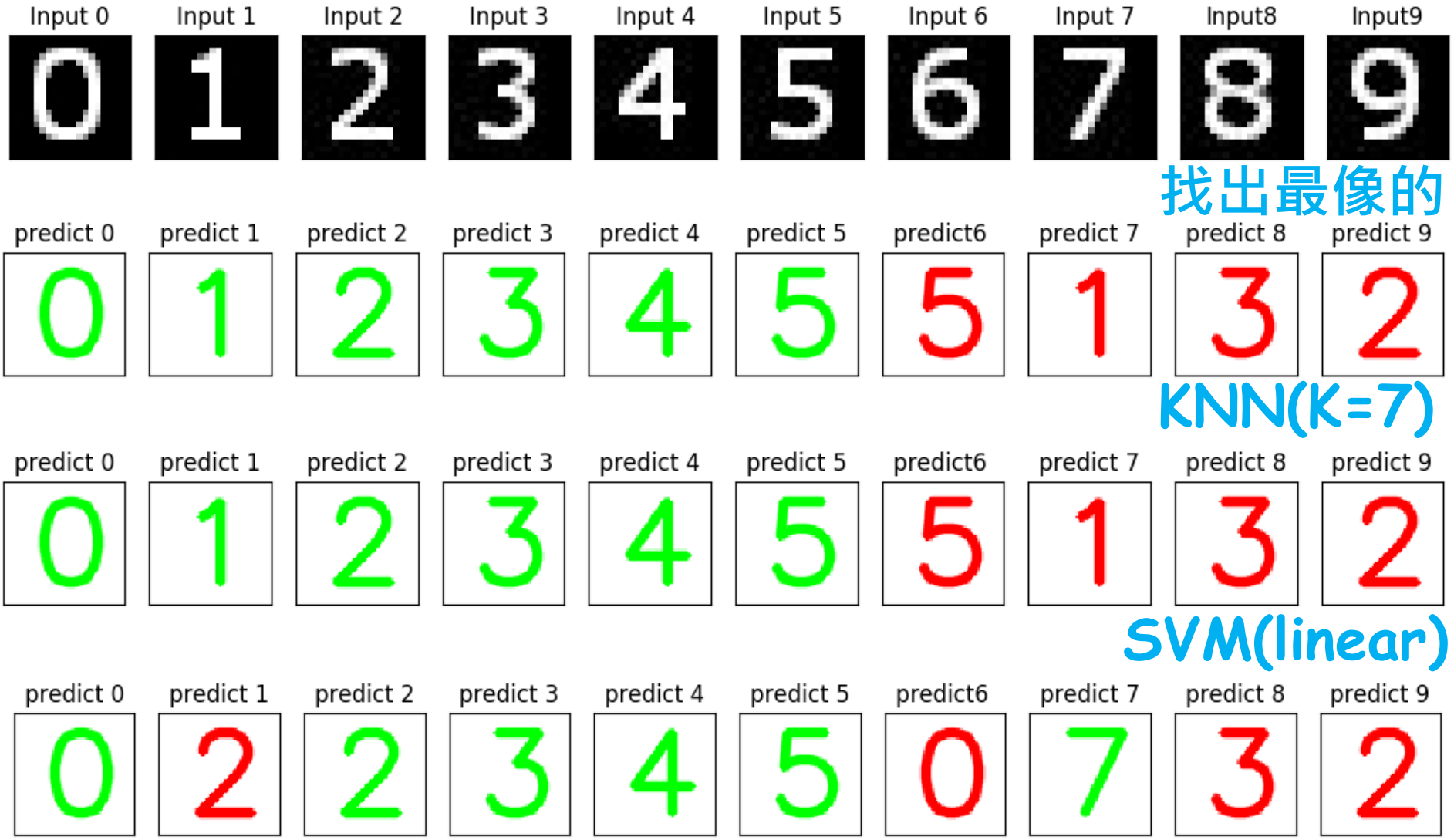


流程

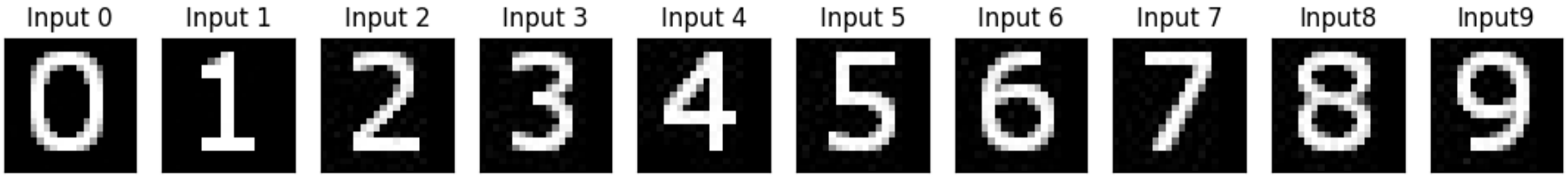


沒了?

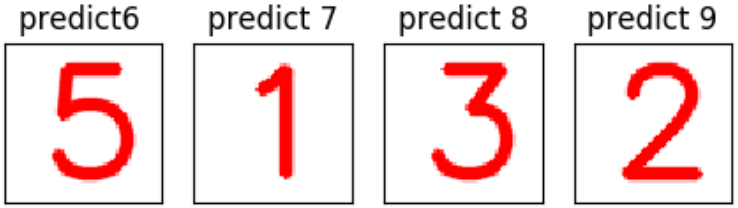
分析比較


Input 0
Input 1
Input 2
Input 3
Input 4
Input 5
Input 6
Input 7
Input8
Input9
找出最像的
predict 0
predict 1
predict 2
predict 3
predict 4
predict 5
predict6
predict 7
predict 8
predict 9
0 1 2 3 4 5 5 1 3 2
KNN(K=7)
predict 0
predict 1
predict 2
predict 3
predict 4
predict 5
predict6
predict 7
predict 8
predict 9
0 1 2 3 4 5 5 1 3 2
SVM(linear)
predict 0
predict 1
predict 2
predict 3
predict 4
predict 5
predict6
predict 7
predict 8
predict 9
0 2 2 3 4 5 0 7 3 2

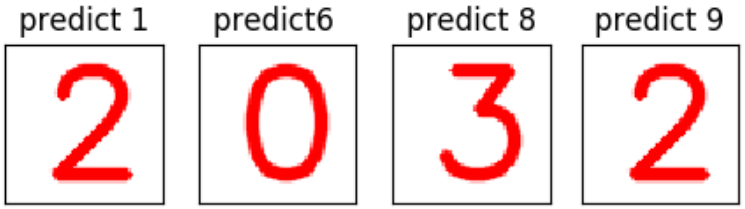
觀察DataSet



KNN(K=7)



SVM(linear)



字體問題，好像情有可原...

這次是真的沒了

參考資料

前置處理、切割

CAPTCHA (驗證碼) OCR 前置處理 - 去噪、補點 (Python)
http://blog.steven5538.tw/2014/06/22/captcha-ocr-preprocess-python/
CAPATHA(驗證碼) OCR (Python)
http://blog.steven5538.tw/2015/06/02/CAPATHA-ocr-using-python-opencv/
使用opencv進行數字識別
http://icodeit.org/2013/01/basic-digits-recognization/
如何透過OpenCV破解台灣證券交易所買賣日報表的驗證碼(Captcha) (Part 1)?[影片]
http://course.largitdata.com/course/37/

參考資料

辨識

機器學習(1)--使用OPENCV KNN實作手寫辨識
http://arbu00.blogspot.tw/2016/11/1-opencv-knn.html
機器學習(2)--使用OPENCV SVM實作手寫辨識
http://arbu00.blogspot.tw/2016/11/2-opencv-svm.html
如何透過OpenCV破解台灣證券交易所買賣日報表的驗證碼(Captcha) (Part 2)?[影片]
http://course.largitdata.com/course/38/
Digit Recognition using OpenCV, sklearn and Python
http://hanzratech.in/2015/02/24/handwritten-digit-recognition-using-opencv-sklearn-and-python.html

DataSet

MINST手寫數字資料庫
https://keras-cn.readthedocs.io/en/latest/other/datasets/
sklearn手寫數字資料庫
https://machine-learning-python.kspax.io/Datasets/ex1_the_digits_dataset.html