

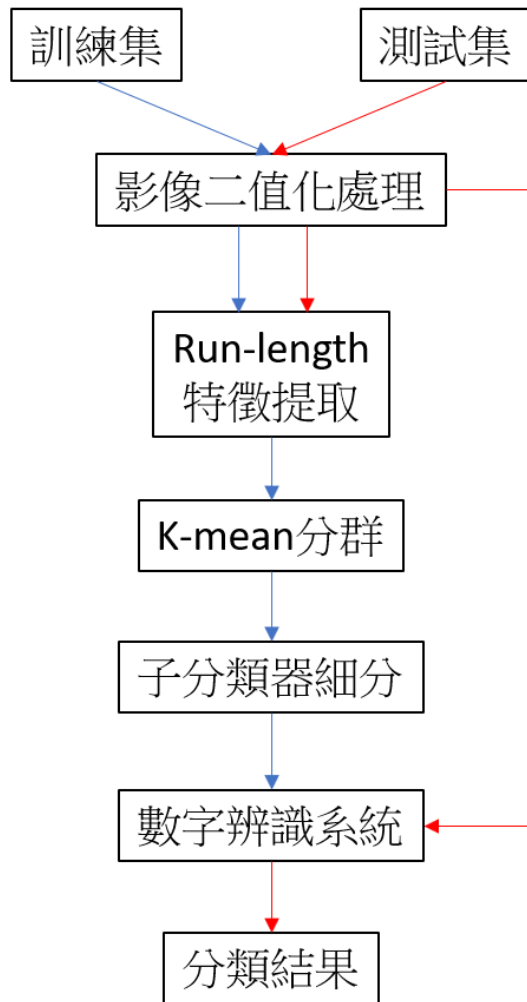
影像處理第11組

以**Run-length**為基礎之阿拉伯數字辨識

40740126S 盧育霆、40711017E 魏世洋、40923129L 湯可伊

摘要

本研究以阿拉伯數字影像為資料集，進行識別訓練和測試。首先進行影像二值化並用Otsu 演算法決定門檻值，以進行影像前處理，接著使用 run-length 演算法找出不同數字的形狀特徵，再利用K-means 方法，將每個特徵類別分成多群，並將容易相互辨識錯誤的類別進行細部分類。最後使用交叉驗證檢驗模型結果。



Otsu's thresholding

- Find t such that

$$\min_{0 \leq t < H} \{ \sigma_{\text{within}}^2(t) \}$$

- where $\sigma_{\text{within}}^2(t) = P_{C_1}(t) \frac{\sum_{i=0}^{t-1} N_i (i - M_{C_1}(t))^2}{N_{C_1}} + P_{C_2}(t) \frac{\sum_{i=t}^{L-1} N_i (i - M_{C_2}(t))^2}{N_{C_2}}$
- $P_{C_1}(t) = \sum_{i=0}^{t-1} p_i$, $P_{C_2}(t) = \sum_{i=t}^{L-1} p_i$,
- $M_{C_1}(t) = \frac{\sum_{i=0}^{t-1} i \cdot p_i}{P_{C_1}(t)}$, $M_{C_2}(t) = \frac{\sum_{i=t}^{L-1} i \cdot p_i}{P_{C_2}(t)}$,
- N_i = 灰階值 i 的 pixel 數量

Run-length

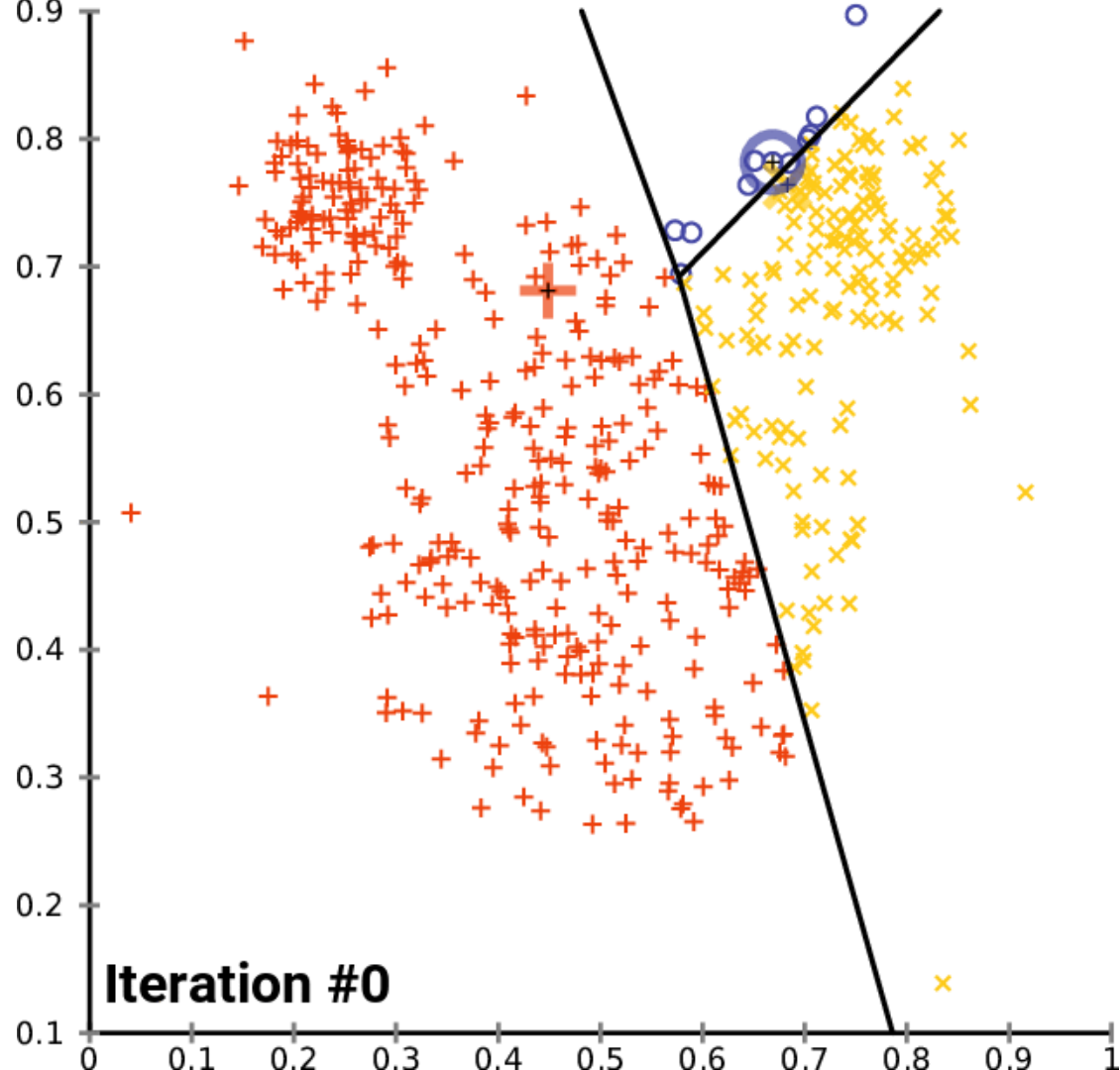
- Run-length 透過紀錄連續出現的事件來有效的地描述影像中的顏色、形狀、紋路等特性。
 - Ex: AAAAABBCCCCDDD → 5A2B4C3D
- 透過Run-length提取黑白兩色在4個不同角度(0° 、 45° 、 90° 、 135°)連續出現特徵值。

Example

- 計算0度的特徵值：
- For every pixel , check
 - initialize : $run = 1$ ◦
 - When current pixel and right pixel are same color : $run += 1$ ◦
 - Otherwise, $RL[color]. 0^{\circ} += run * run$, $run = 1$ ◦

K-means

- 可以將資料集根據其特徵分成K個群。
- 選K個資料當初始群中心。
- 接著計算其他資料與K個群中心的距離，加入離最近的群。
- 計算新的群中心。
- 再根據新的群中心，將所有資料再次計算與群中心的距離，加入離最近的群。
- 直到群中心不再變化。



其他技巧(1)

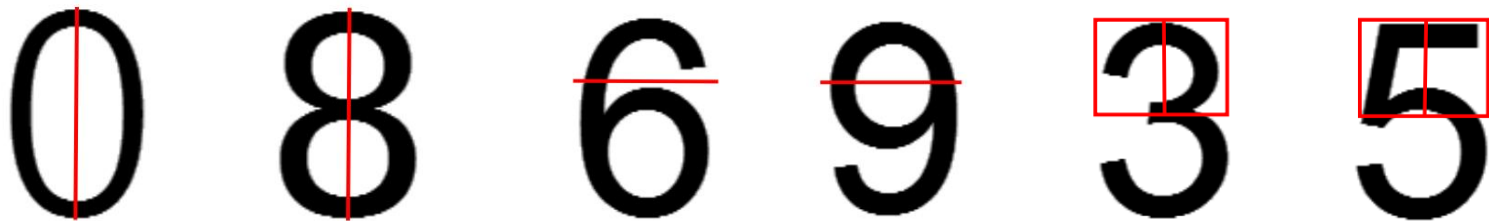
- 旋轉不變性：
 - 利用相同數字某個角度的特徵值相近，將不同角度的相同數字進行旋轉。



其他技巧(2)

- 子分類器：

- 利用黑白兩色的切換次數、顏色面積來判斷易分類錯誤的數字。



DEMO 影像資料

每個數字40張，共 $9 \times 40 = 360$ 張



DEMO 影像前處理

```
#-----#  
  
def binarize_image(image_path):  
    # 讀取影像  
    image = cv2.imread(image_path, 0) # 以灰階模式讀取影像  
    # 使用Otsu方法找到最佳門檻值  
    _, binary_image = cv2.threshold(image, 0, 255,  
                                     cv2.THRESH_BINARY + cv2.THRESH_OTSU)  
    return binary_image  
  
#-----#
```

DEMO 影像前處理

```
def process_images_in_folder(folder_path):  
    runLengthResult = []  
  
    # 處理資料夾內的影像  
    for filename in os.listdir(folder_path):  
        print(filename)  
        if filename.endswith(".jpg") or filename.endswith(".png"):  
            # [b0,b45,b90,b135,w0,w45,w90,w135]  
            RL = [0,0,0,0,0,0,0,0]  
  
            image_path = os.path.join(folder_path, filename)  
            # 進行影像二值化處理  
            image = binarize_image(image_path)  
            height, width = image.shape
```

DEMO run-length特徵值提取

```
## 00000
for x in range(1,height):
    runs=1
    for y in range(1,width):
        if(image[x][y-1] == image[x][y]):
            runs += 1
        else:
            runs *= runs
            if(image[x][y]==0):
                RL[0] = RL[0]+runs
            else:
                RL[4] = RL[4]+runs
            runs = 1

    if(y==width-1):
        runs *= runs
        if(image[x][y]==0):
            RL[0] = RL[0]+runs
        else:
            RL[4] = RL[4]+runs
```

```
## 909090
for y in range(1,width):
    runs = 1
    for x in range(1,height):
        if(image[height-x-1][y] == image[height-x-2][y]):
            runs += 1
        else:
            runs *= runs
            if(image[height-x-2][y]==0):
                RL[2] = RL[2]+runs
            else:
                RL[6] = RL[6]+runs
            runs = 1

    if(x==height-2):
        runs = runs*runs
        if(image[height-x-2][y]==0):
            RL[2] = RL[2]+runs
        else:
            RL[6] = RL[6]+runs
```

DEMO run-length特徵值提取

```
## 454545
for x in range(1,height):
    runs=1
    steps = x # max = height-1
    try:
        for y in range(1,steps):
            if(image[x-y][y] == image[x-y-1][y+1]):
                runs += 1
            else:
                runs *= runs
                if(image[x-y][y]==0):
                    RL[1] = RL[1]+runs
                else:
                    RL[5] = RL[5]+runs
                runs = 1

        if(y==steps-1):
            runs *= runs
            if(image[x-y][y]==0):
                RL[1] = RL[1]+runs
            else:
                RL[1] = RL[5]+runs
```

```
for y in range(2,width-2):
    runs = 1
    steps = height-y
    try:
        for x in range(1,steps):
            if(image[height-x][y+x-1] == image[height-x-1][y+x]):
                runs += 1
            else:
                runs *= runs
                if(image[height-x-1][y+x]==0):
                    RL[1] = RL[1]+runs
                else:
                    RL[5] = RL[5]+runs
                runs = 1

        if(x==steps-1):
            runs *= runs
            if(image[height-x-1][y+x]==0):
                RL[1] = RL[1]+runs
            else:
                RL[5] = RL[5]+runs
            runs = 1
```

DEMO run-length特徵值提取

```
## 135135
for x in range(1,width):
    runs = 1
    steps = x
    w = width
    for y in range(1,steps):
        w = w-1
        try:
            if(image[x-y][w] == image[x-y][w-1]):
                runs += 1
            else:
                runs *= runs
                if(image[x-y][w]==0):
                    RL[3] = RL[3]+runs
                else:
                    RL[7] = RL[7]+runs
                runs = 1
        if (y==steps-1):
            runs *= runs
            if(image[x-y][w]==0):
                RL[3] = RL[3]+runs
            else:
                RL[7] = RL[7]+runs
```

```
for y in range(1,width-1):
    runs = 1
    steps = height - y
    w = width - y
    for x in range(1,steps):
        try:
            w = w-1
            if(image[height-x][w] == image[height-x][w-1]):
                runs += 1
            else:
                runs *= runs
                if(image[height-x][w]==0):
                    RL[3] = RL[3]+runs
                else:
                    RL[7] = RL[7]+runs
                runs = 1
        if (x == steps-1):
            runs *= runs
            if(image[height-x][w]==0):
                RL[3] = RL[3]+runs
            else:
                RL[7] = RL[7]+runs
```


DEMO K-means分群

```
def run_Kmeans(X):  
  
    # k-fold交叉驗證的摺疊數  
    k_fold = 30  
    kf = KFold(n_splits=k_fold)  
    # 儲存每個k值的交叉驗證評估結果  
    evaluation_results = {}  
  
    for k in range(10):  
        # 儲存當前k值的所有評估指標  
        k_evaluation = []  
        for train_index, test_index in kf.split(X):  
            print("...")  
            # 取得訓練集和測試集  
            X_train=[]  
            X_test=[]  
            for i in train_index:  
                X_train.append(X[i])  
            for i in test_index:  
                X_test.append(X[i])
```


```
        # 建立並擬合k-means模型  
        kmeans = KMeans(n_clusters=10)  
        kmeans.fit(X_train)  
        # 在測試集上進行預測  
        y_pred = kmeans.predict(X_test)  
        # 計算評估指標(以輪廓係數為例)  
        score = silhouette_score(X_test, y_pred)  
        k_evaluation.append(score)  
  
        # 計算當前k值的平均評估指標  
        average_score = np.mean(k_evaluation)  
        # 將結果存到字典  
        evaluation_results[k] = average_score
```


DEMO 結果

```
-----  
Score=0.5534737733132984  
Score=0.5749197919965792  
Score=0.5773060592108745  
Score=0.5432559742488379  
Score=0.5593328788238187  
Score=0.5833050913694684  
Score=0.5635107433923211  
Score=0.5521566826655534  
Score=0.5689243428768166  
Score=0.5841266291251074  
-----
```

群間離散率約0.56

DEMO 結果

 : 標籤和數字的對應

 : 累積最多次的標籤

數字

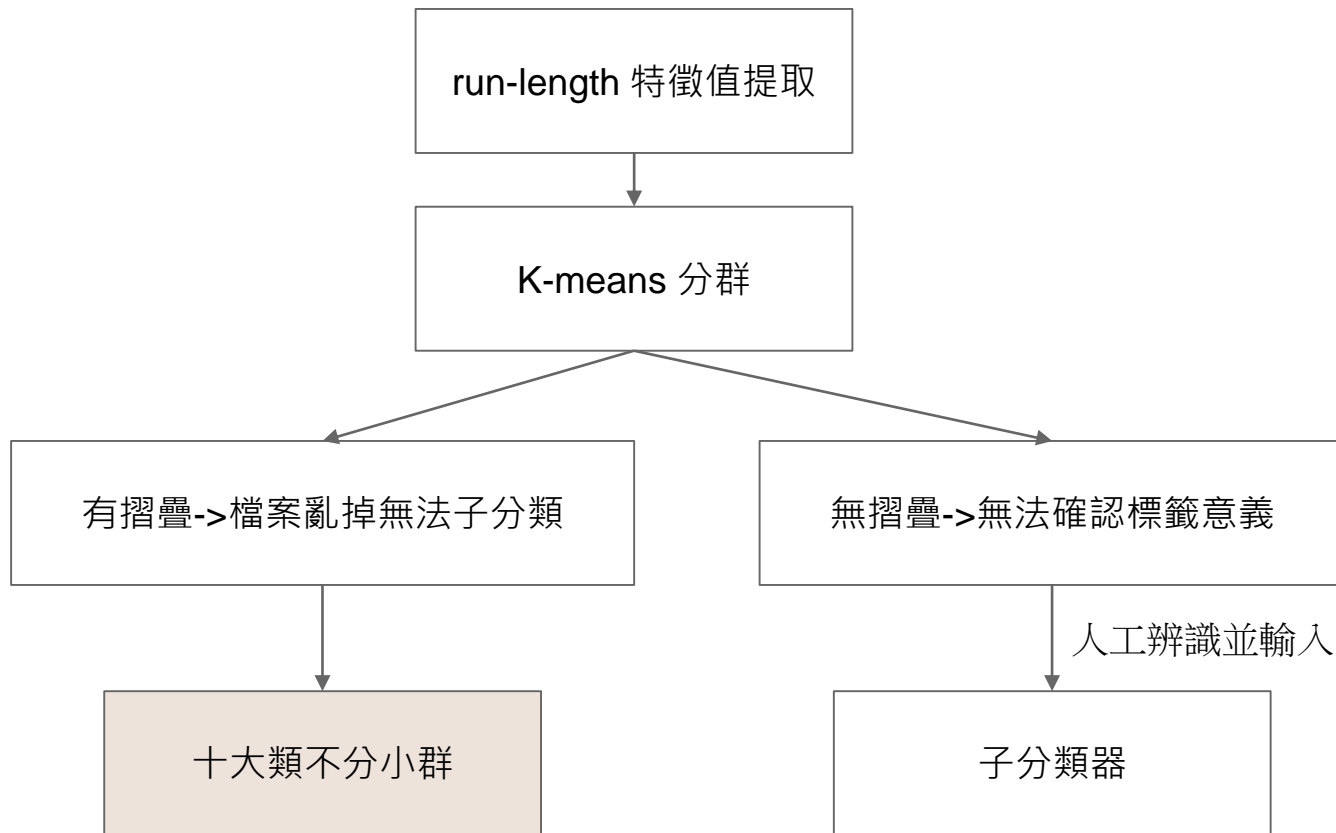
標籤

	0	1	2	3	4	5	6	7	8	9
0	5	0	0	10	0	4	1	13	1	6
1	0	0	0	4	16	13	0	5	2	0
2	4	0	0	20	0	4	2	10	0	0
3	4	2	0	18	2	1	4	7	2	0
4	4	0	1	12	0	9	2	10	2	0
5	10	0	0	15	0	5	0	8	2	0
6	0	0	0	18	2	12	0	2	6	0
7	7	0	0	18	0	2	2	11	0	0
8	2	0	0	18	0	2	0	12	6	0
9	8	0	0	16	0	2	6	0	8	0

表 4-1. K-means 分一群辨識結果

	0	1	2	3	4	5	6	7	8	9
0	60	0	0	0	0	0	0	0	7	19
1	0	85	0	0	0	0	0	0	0	0
2	0	0	30	28	0	19	0	10	3	0
3	0	0	0	49	0	11	0	8	0	13
4	3	0	2	0	59	0	2	5	3	22
5	0	0	0	18	0	73	0	0	1	1
6	3	0	3	0	3	0	6	0	2	43
7	0	0	6	1	0	0	0	81	0	0
8	57	0	0	0	0	28	0	0	0	10
9	2	0	1	0	1	0	0	0	6	47

DEMO 遇到的問題



DEMO 實做展示

<https://gitlab.com/ToshaET/imageprocessinggroup11>



謝謝大家