Assignment 4

107062208 邱靖豪

(a) How to use the function “scipy.sparse.linalg.eigs.”?

這個 function 的功能主要是用來算放進去的矩陣的 eigenvectors 跟對應的 eigenvalues，以我的 code 當範例，spla 是先從 scipy.sparse.linalg 引入的，接著，就使用.eigs 的功能，L 在這裡是那個稀疏矩陣，k = m，k 是希望得到的 eigenvectors 數，而 which = ‘SR’ 指的是找到前 k 小，同時也是實數的 eigenvectors。而等號左邊的 vals, vecs，指的是得到的 eigenvalues 跟 eigenvectors。

```
import scipy.sparse.linalg as spla
vals, vecs = spla.eigs(L, k=m, which='SR')
```

(b) How K-means algorithm works?

K-means algorithm 是一種演算法，目標是將 n 個待觀察物分類成 k 個群集，K-Means 是聚類演算法中的最常用的一種，演算法最大的特點是簡單，好理解，運算速度快。他分成三個步驟，首先，**(1) Initialisation**，先隨機舉出 k 個初始群集中心 (means 或 centroids)。接著是 **(2)Assignment**，以「連結各個觀察物最近的 centroids」，創造出 k 個 clusters，方法是計算每筆資料到 k 個群心的歐式距離(歐基李德距離 Euclidean distance)。再來是**(3)Update**，將每筆資料分類給距離最近的那

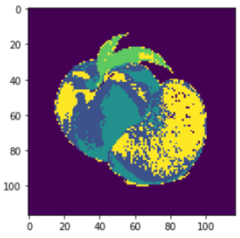
個群心，接著，每個群心內都會有被分類過來的資料，用原本的資料加上被分類過來的資料，更新一次新的群心。反覆以上步驟，直到群心沒有太大變動，演算即結束。

(c) Original images and the figures after running two segmentation methods.

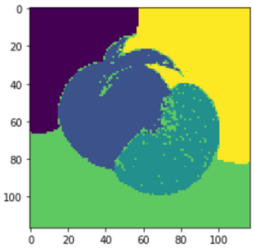
Original image:

Original:

RGB K-Means:



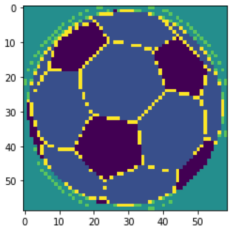
Spectral Clustering:



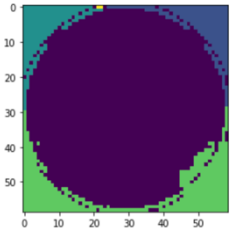
My image1:

Original:

RGB K-Means:



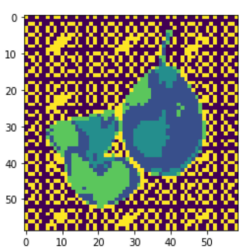
Spectral Clustering:



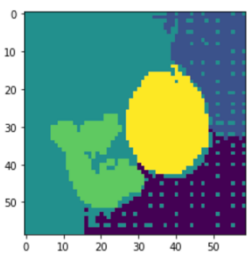
My image2:

Original:

RGB K-Means:



Spectral Clustering:



(d)Discussion about what kinds of images prefer which segmentation methods

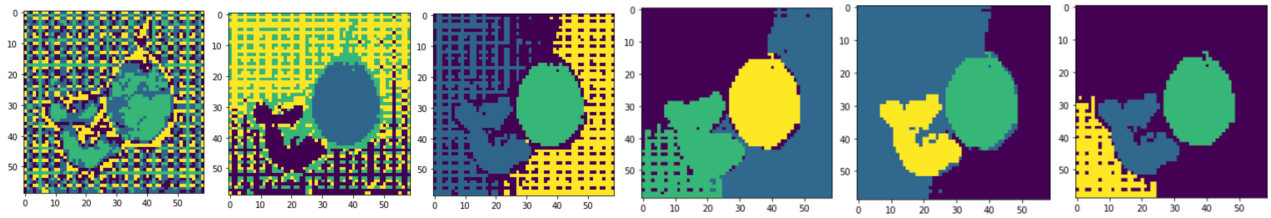
從我自己挑的兩張圖看來，只要是圖形顏色單調，或者是形狀沒那麼複雜的，RGB K-Means 的效果較好，反之，比較複雜的圖案，跟顏色比較有陰影對比的圖片，用 Spectral Clustering 會比較好。

(e) Figures of segmentation results for different number of eigenvectors and clusters

Different numbers of eigenvectors

Cluster = 4

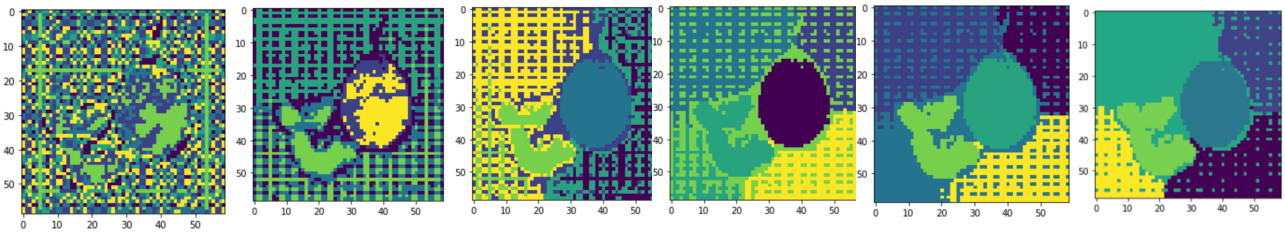
1 2 3 4 5 6



Different numbers of eigenvectors

Cluster = 6

1 2 3 4 5 6



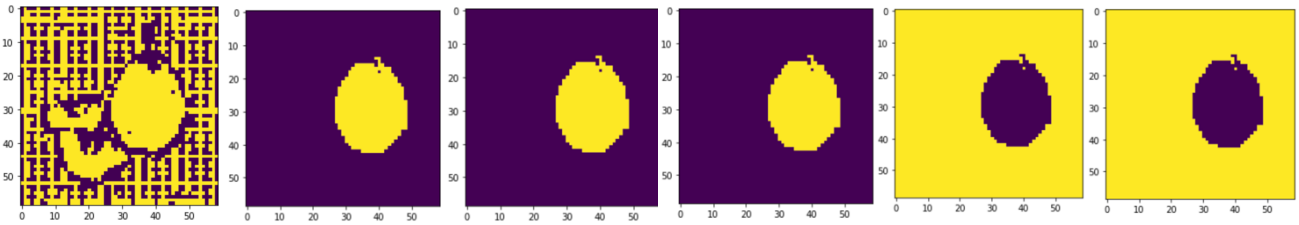
(f) Discussion about the influence of different number of eigenvectors and clusters

Clusters 在上面提過，在這個演算法裡的功能是代表分出類群，因此，cluster=n 代表把圖片分成 n 個色塊，由上題的 figures 可以發現，同一張圖片當 clusters 的數值較大時，需要較多的 eigenvectors 才能清楚的顯示圖片，以 eigenvectors=4 作為對照，cluster=4 時能比較好看出輪廓了。但假如 cluster 數過小，分成的色塊也就比較少，假使 eigenvectors 數多，還是沒辦法讓輪廓較清楚。以 cluster=2 為例，甚至當 eigenvector 變大時，沒辦法將完整的輪廓顯示出。

Different numbers of eigenvectors

Cluster = 2

1 **2** **3** **4** **5** **6**



(g)Discussion about the COO format

它是一種儲存 sparse matrix 的方式，因為在 sparse matrix 中 value 為 0 的數值很多，用一般的二維矩陣來記錄很佔空間，因為只會紀錄很分散的非零值，所以，COO format 就是用來優化此問題的方法。考慮原本的 A 是

N element 大小的 sparse matrix，但經過 COO format 後，matrix 的大小就會變為 sparse matrix 中不為 0 的 element 個數(k)*3(row index、column index、value)，分別用來記錄那些非零元素得 row 值、column 值以及數值大小，這樣子存取空間是 3k，考慮到 k<<N，所以 3k<<N，存取空間來說，應該是縮小不少。但是，COO 不支援儲存和刪減 element，除非轉換為其它格式的矩陣，否則很難對其做操作或是矩陣運算，因此在計算 eigenvalues 上，所需的時間比較長，因為還要轉換成可運算的矩陣，這是 coo format 較不方便的地方。

(h) Result and discussion on the bonus question

Affinity measure	Expression
Random	$W_r = X_r X_r^T$, where X_r is a $n \times d$ random matrix
Trivial	$W_t = X\ X^T$
Exponential	$w_{ij} = e^{-d^2(x_i, x_j)}$ con $w_{ii} = cte$
Scaled exponential	$w_{ij} = e^{-d^2(x_i, x_j)/(\sigma_i \sigma_j)}$, with $w_{ii} = 0$, where $\sigma_i = d(x_i, x_N)$ and x_N is the N -th nearest neighbor

Fig 2. Weighted graph with three nodes

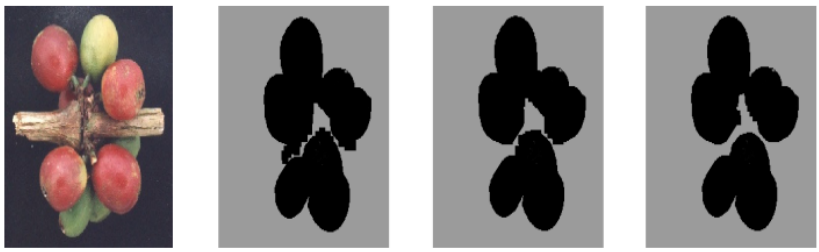
這是在網路上搜尋到的論文，所做的比較。根據它的公式去解讀的話，我們使用 scaled exponential affinity 的話，切割出來的照片會比較少鋸齒，物件跟背景的切割會比較圓滑。我們能了解到，pixel 之間的 weight 如果是用作業中的公式，就會發現到如果兩個顏色差異是一樣的，他們的 weight 就

會是一樣的，比如說一組是紅（1,0,0）黑（0,0,0），另一組是藍（0,1,0）跟黑，量來說差異是一樣的，然而卻有不同的顏色。因此，如果使用 scaled exponential affinity，每個 weight 在運算時，他的 pixel 都有不同的σ值，所以即使他們在顏色編碼上的差異一樣，他們的 weight 會不一樣，這樣就能分辨出顏色差距一樣，但實際顏色不一樣的組合，另外，這篇論文的算法是用距離，但是，我覺得因為 pixel 利用到的編碼概念，跟算距離的概念是可以相通的，所以這個方法在此也適用。

以下是 example，(a)為原圖，(b)是 trivial affinity，(c)是 exponential affinity (d)是 scaled exponential affinity，可以看出 scaled exponential affinity 的確比較光滑，符合原圖輪廓一些。

參考資料及例子來源：

http://diegopeluffo.com/publicaciones/STSIVA2010.pdf



(a) (b) (c) (d)