ML HW08

Used python 3.5.3

● Symmetric SNE

程式碼主要的部份可以寫成

計算 high dimension similarity P

For (iter)

　　計算 low dimension similarity Q

　　計算更新 gradient

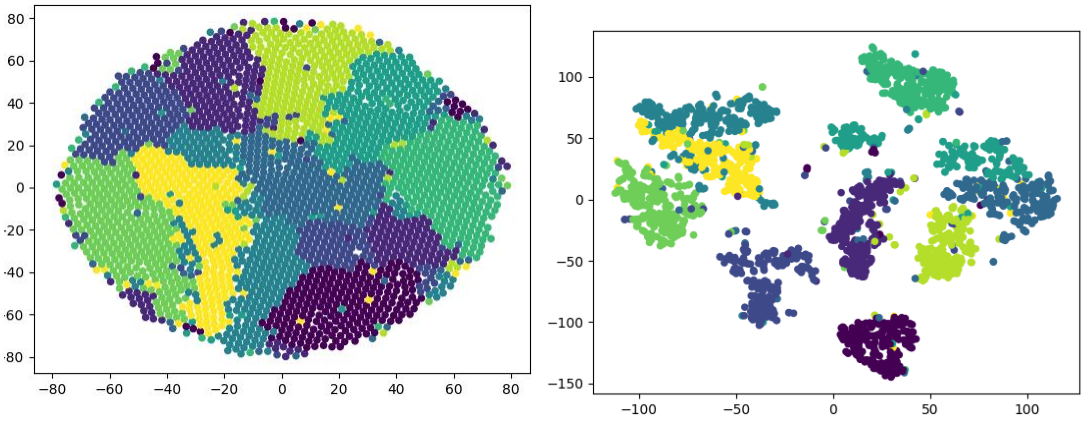
　　更新 low dimension 值 y

t-SNE 和 symmetric SNE 的差別在於 t-SNE 的 low dimension 採用 student-t，symmetric SNE 仍用 Gaussian。

```
106 def lowDimensionDistribution(Y,n,mode=0):
107     #mode 0 = student-t model= gasssian
108     # Compute pairwise affinities
109     if mode==0:
110         sum_Y = np.sum(np.square(Y), 1)
111         # num is similarity numerator matrix, with diaganal=0 (1+|y-yi|^2)^-1
112         num = -2. * np.dot(Y, Y.T)
113         num = 1. / (1. + np.add(np.add(num, sum_Y).T, sum_Y))
114         num[range(n), range(n)] = 0.
115         Q = num / np.sum(num)
116         Q = np.maximum(Q, 1e-12)
117     elif mode==1:
118         sum_Y = np.sum(np.square(Y), 1)
119         # num is similarity numerator matrix, with diaganal=0 exp(-|y-yi|^2)
120         num = -2. * np.dot(Y, Y.T)
121         num = np.exp(-1.*(np.add(np.add(num, sum_Y).T, sum_Y)))
122         num[range(n), range(n)] = 0.
123         Q = num / np.sum(num)
124         Q = np.maximum(Q, 1e-12)
125     else:
126         print("Unknown mode for low Dimension")
127         return -1
128     return Q,num
```

Mode 0 對應的是 tSNE，mode 1 對應 symmetric SNE。他在計算時將||yi-yj|^2 拆成 yi^2-2*yiyj+yj^2，np.maximum()則是為了避免除以 0 的狀況發生。而上下的實際差別只在 $(1+ \| y_i - y_j \|^2)^{-1}$ 改成 $\exp(- \| y_i - y_j \|^2)$ 。

● Embedding Results

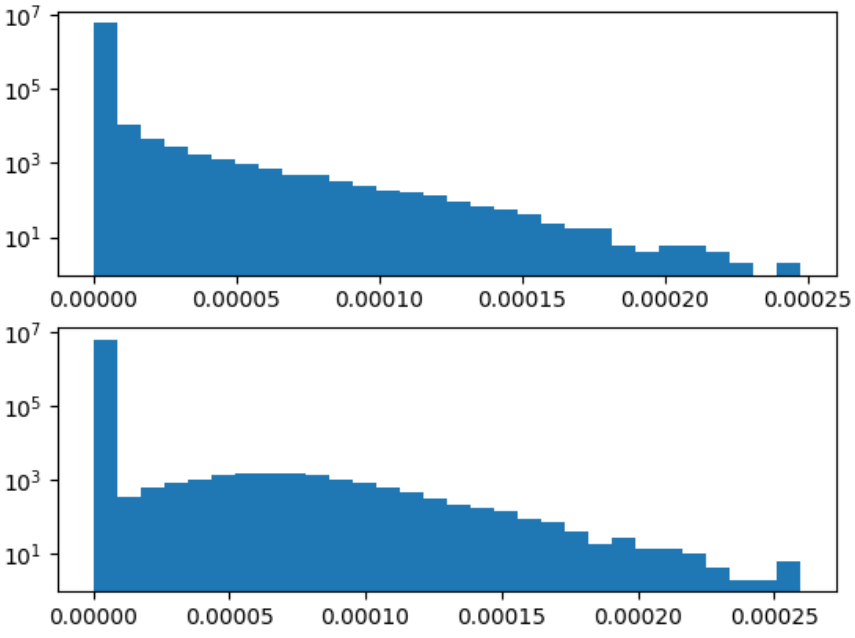


▲左側是 Symmetric SNE，右側是 tSNE

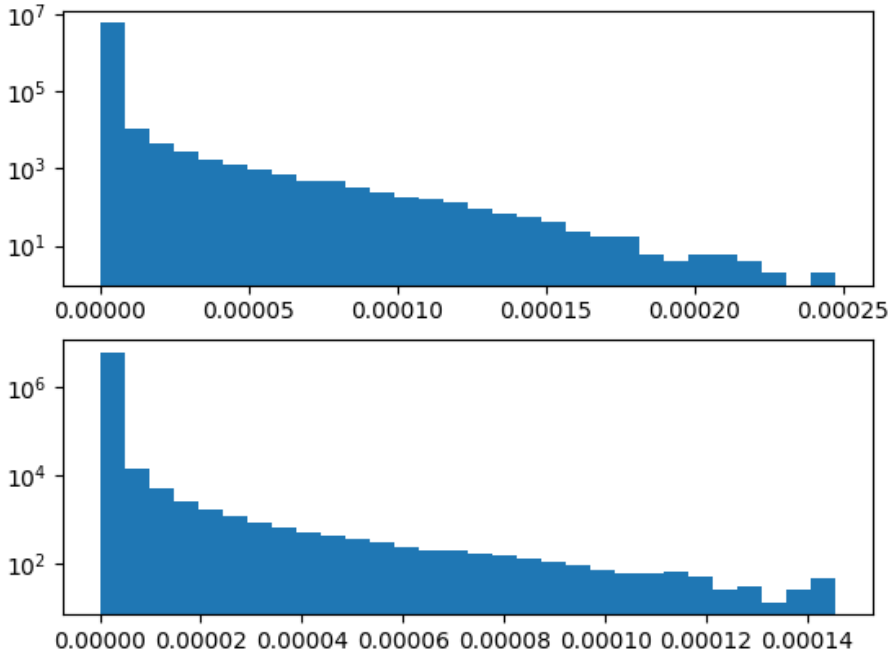
雖然左右都有維持住各自在 high dimension 的 similarity，但可以很明顯地觀察到左側的 Symmetric SNE 有 Crowding problem，區塊間相接在一起會產生 neighbor mismatch 的問題。右側用 tSNE 則是順利的解決這個問題，區塊間分隔很明顯，值的分布從標準的高斯分布 ([-80,80]) 變成較為散落的分布 ([-100,100])。

左側的圖可以觀察到 SNE 的性質，就是維持住 local structure 是最重要的，而將原先距離遠的點拉近則沒有那麼大的懲罰。

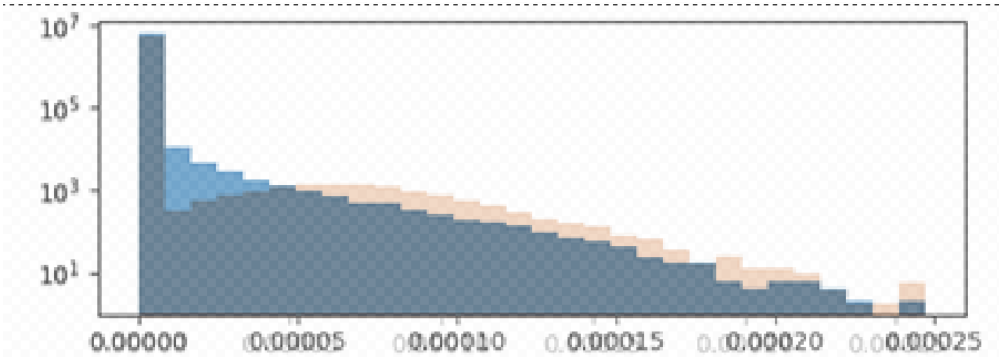
● Pairwise Similarity



Symmetric SNE



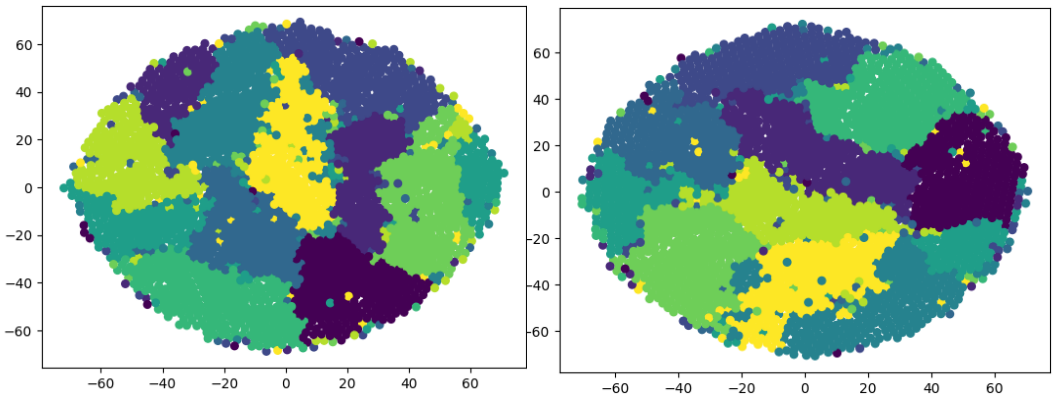
t-SNE



這是 symmetric SNE 的堆疊圖，橙色為 low dimension，可以觀察到大部分的 similarity 都上升了。符合保持高 similarity 的特色，但同時也代表著在 low dimension 大家比較近，容易出現 Crowding problem.

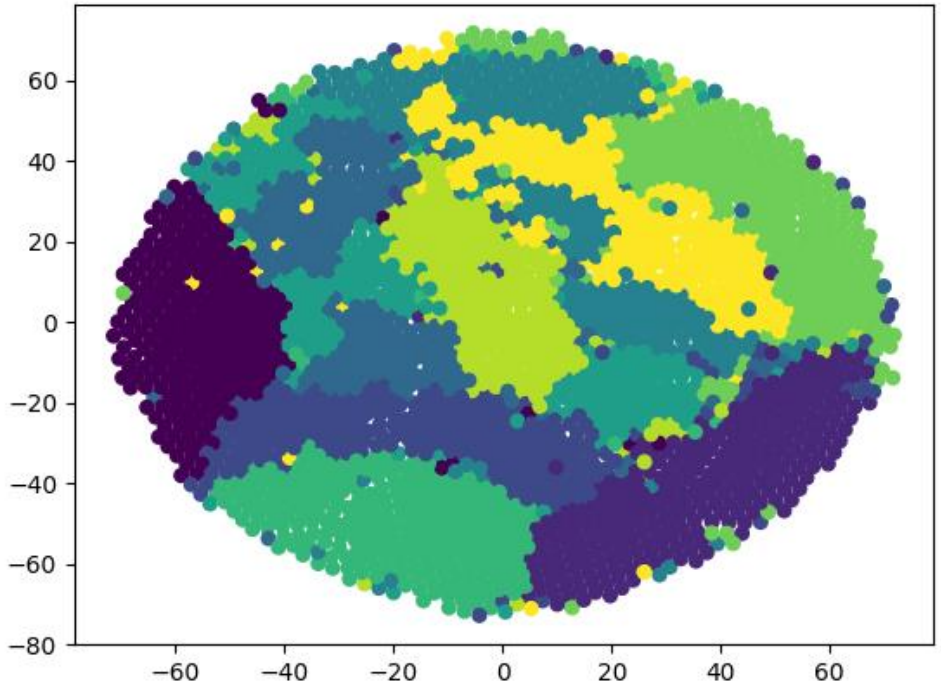
至於另一張圖的 t-SNE 曲線比較平滑，符合 student-t 的樣子。

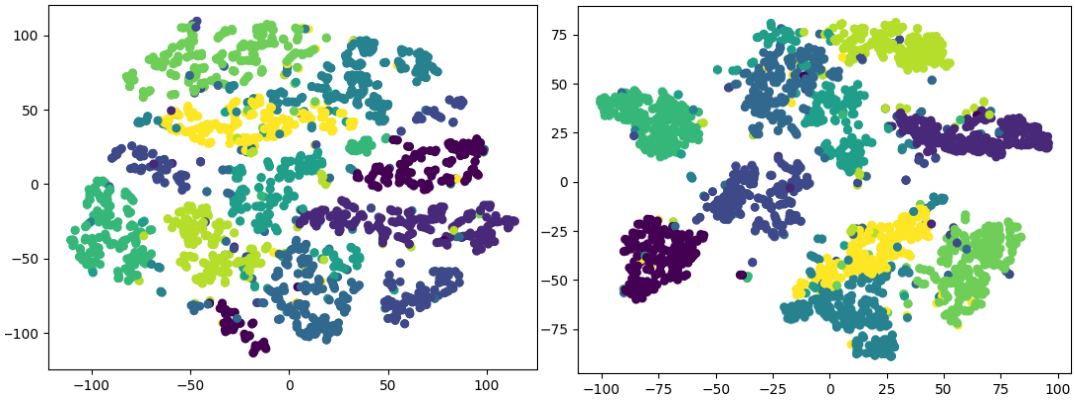
- Perplexity and Visualization



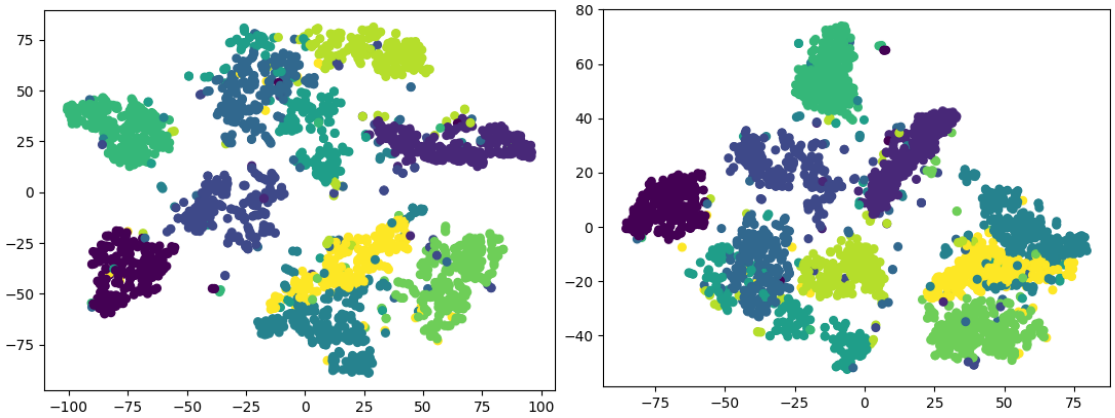
此為 symmetric SNE 左側為 perplexity 20，右側是 perplexity 50，因為已經有 Crowding problem 所以即使調整 perplexity 也應該不會有多大差別，頂多位置調換。

以下是 perplexity 200，並無差別。





這個是 t-SNE，左側為 perplexity 5 (低於正常值)，右側是 perplexity 20。可以觀察到左側的結構較為鬆散甚至於要平均分布了。這是因為考慮的 neighbor 太少，使得 local structure 沒有保留下來。



同為 t-SNE，左側為 perplexity 20，右為 perplexity 50，可以看到同一個群內的距離拉的更近了，同時群跟群的距離也拉近了。當考慮的 neighbor 變多時，local structure 會拉得更近，而為了保持 local structure 則會拉近一些非自己一群的作為平衡。