ML HW8 report

資科工碩一

0756138

黃伯凱

Symmetric SNE

一般來說，將 data 分成多個 class 的時候會利用 data 之間的距離來區分，而 SNE 則是將距離轉換成機率來表達點跟點之間的 similarity。P 代表高維時的 similarity，Q 則代表低維時的 similarity，使用的 gaussian 來計算 similarity。

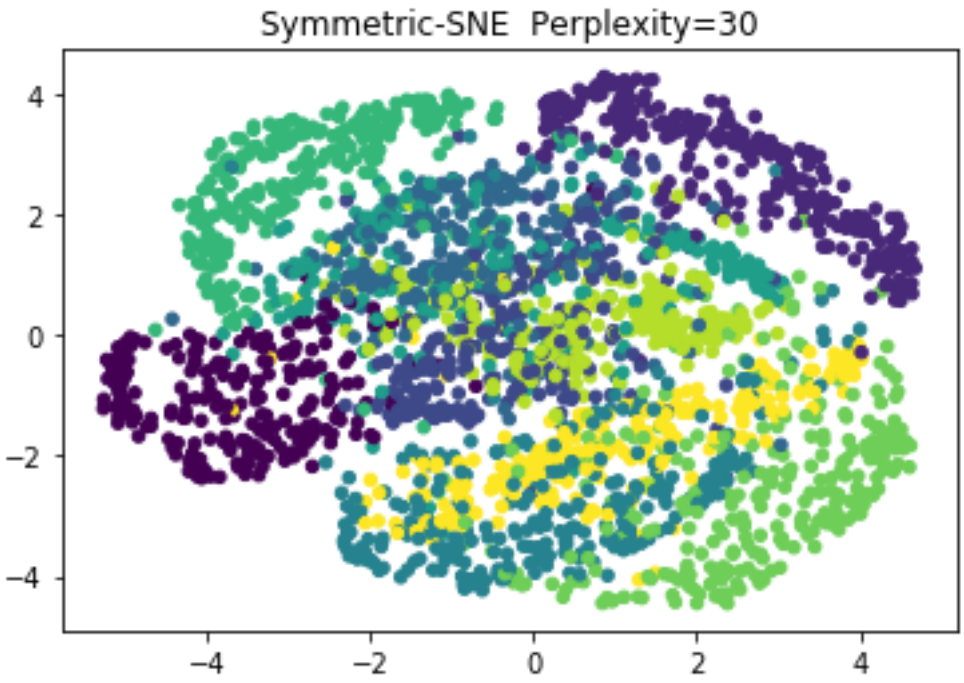
$$p_{ij} = \frac{\exp(-\,||\,x_i - x_j\,||^2\,/2\sigma^2)}{\sum_{k \neq l} \exp(-\,||\,x_k - x_l\,||^2\,/2\sigma^2)} \qquad q_{ij} = \frac{\exp(-\,||\,y_i - y_j\,||^2)}{\sum_{k \neq l} \exp(-\,||\,y_k - y_l\,||^2)}$$

目標函數則是兩個維度之間的差距，由於上述是計算出各自的分佈，所以差距就要用 KL divergence 來計算。

$$C = \sum_i KL(P_i\,||\,Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$

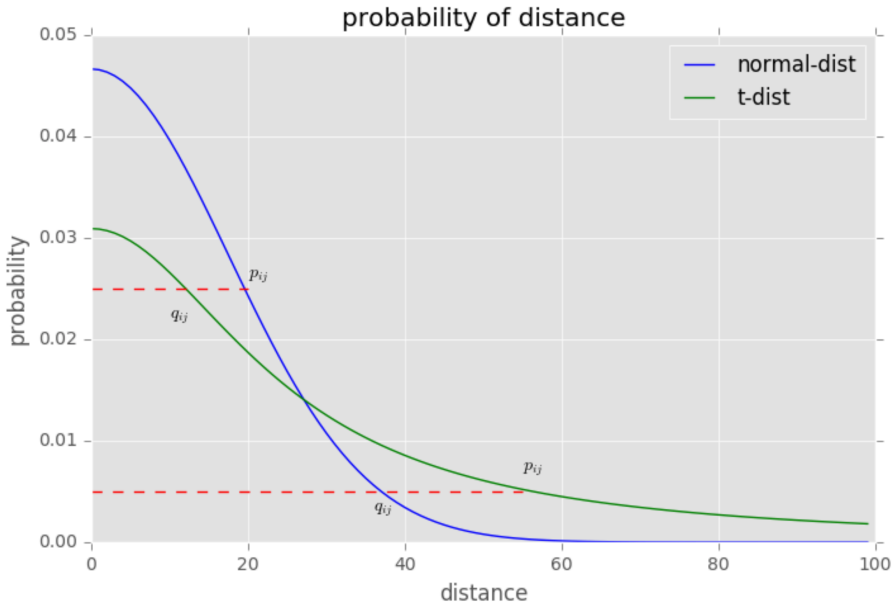
Gradient 的算法則如下：

$$\frac{\delta C}{\delta y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)$$



T – SNE

如 Symmetric-SNE 的結果，即使能夠成功分辨出不同的 class，在視覺化上我們卻難以把不同 class 的 data 明顯區分開來。所以 t – SNE 改成使用 t – distribution 來計算 similarity：



如上圖所示，藍色的線是 Symmetric – SNE 使用的 Gaussian，綠色的線是 t – SNE 使用的 t – distribution。可以得到兩個差異：

1. 藍線轉換成距離之後，幾乎都是在 60 以內；綠線轉換成距離之後，距離甚至可以超過 100。從此現象可以知道 t – distribution 可以把 data 之間的距離拉開。
2. t – distribution 在短距內的 data 數量並沒有像 Gaussian 那樣那麼多，所以在 gaussian 低維靠很近的 data，t – distribution 可以有效把那們分開。

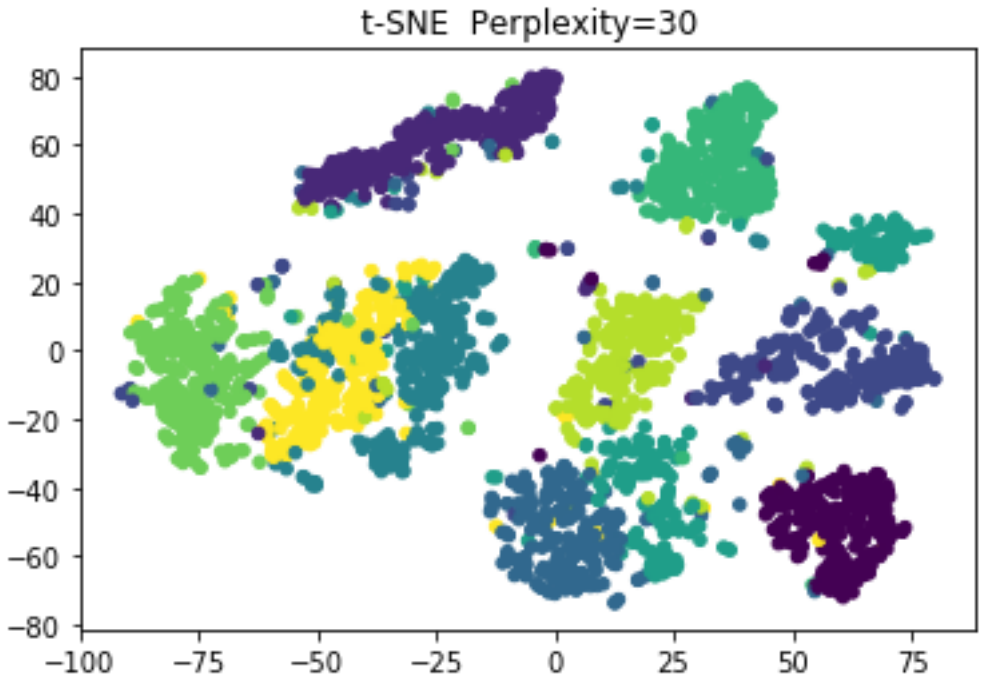
根據以上現象，可以確定 t – distribution 可以將不同 class 的 data 在低維時確實地分開。以上圖兩個紅色虛線為例，p=0.025 時 t – distribution 得到的距離 < Gaussian；p=0.005 時 t – distribution 得到的距離 > Gaussian。再次驗證 t – distribution 可以把近距 data 拉更近，遠距 data 拉更遠。

由於利用 t – distribution 來算 similarity，P 和 Q 變成如下：

$$p_{j|i} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2/2\sigma_i^2)} \qquad q_{ij} = \frac{(1+\|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_{k\neq l}(1+\|\mathbf{y}_k - \mathbf{y}_l\|^2)^{-1}}$$

Gradient 也要跟著改變：

$$\frac{\delta C}{\delta y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)(1+ \mid\mid y_i - y_j \mid\mid^2)^{-1}$$



Code Difference

Symmetric – SNE

根據上面的式子，要改 Q 的算式，所以要改這邊的 num

```
# Compute pairwise affinities
sum_Y = np.sum(np.square(Y), 1)
num = -2. * np.dot(Y, Y.T)
num = np.exp(-1 * np.add(np.add(num, sum_Y).T, sum_Y))
num[range(n), range(n)] = 0.
Q = num / np.sum(num)
Q = np.maximum(Q, 1e-12)
```

Gradient 則要從這邊改

```
# Compute gradient
PQ = P - Q
for i in range(n):
    dY[i, :] = np.sum(np.tile(PQ[:, i], (no_dims, 1)).T * (Y[i, :] - Y), 0)
```

T – SNE

根據上面的式子，要改 Q 的算式，所以要改這邊的 num

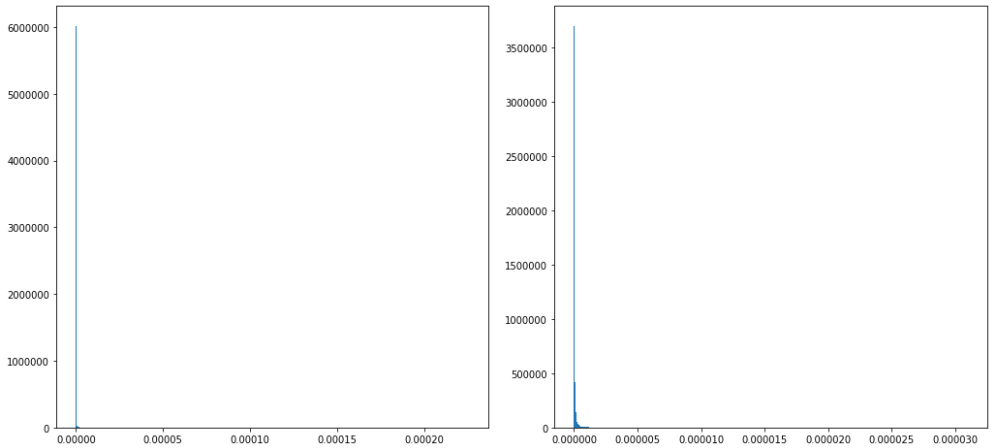
```
# Compute pairwise affinities
sum_Y = np.sum(np.square(Y), 1)
num = -2. * np.dot(Y, Y.T)
num = 1. / (1. + np.add(np.add(num, sum_Y).T, sum_Y))
num[range(n), range(n)] = 0.
Q = num / np.sum(num)
Q = np.maximum(Q, 1e-12)
```

Gradient 則要從這邊改

```
# Compute gradient
PQ = P - Q
for i in range(n):
    dY[i, :] = np.sum(np.tile(PQ[:, i] * num[:, i], (no_dims, 1)).T * (Y[i, :] - Y), 0)
```

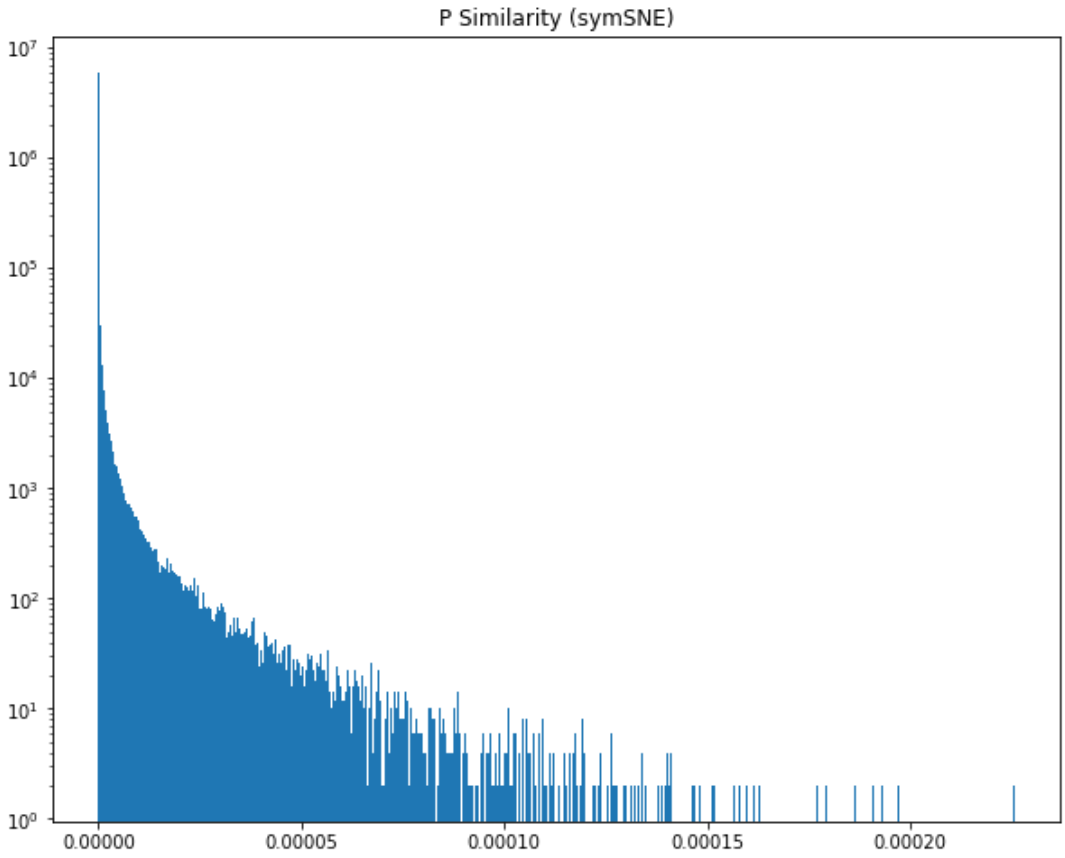
Distribution of Pairwise Similarity

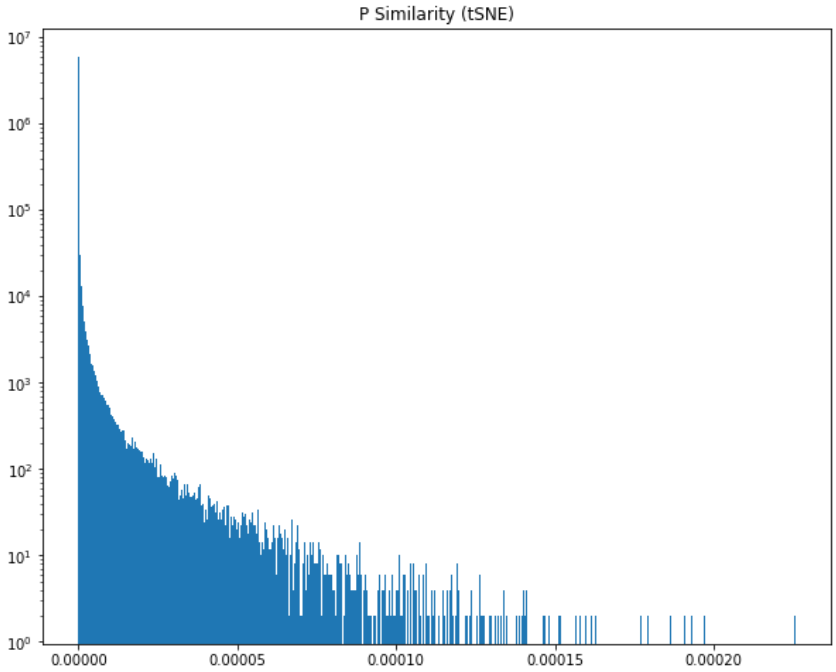
由於大部分的距離都非常小，會長的像下面這種分布：

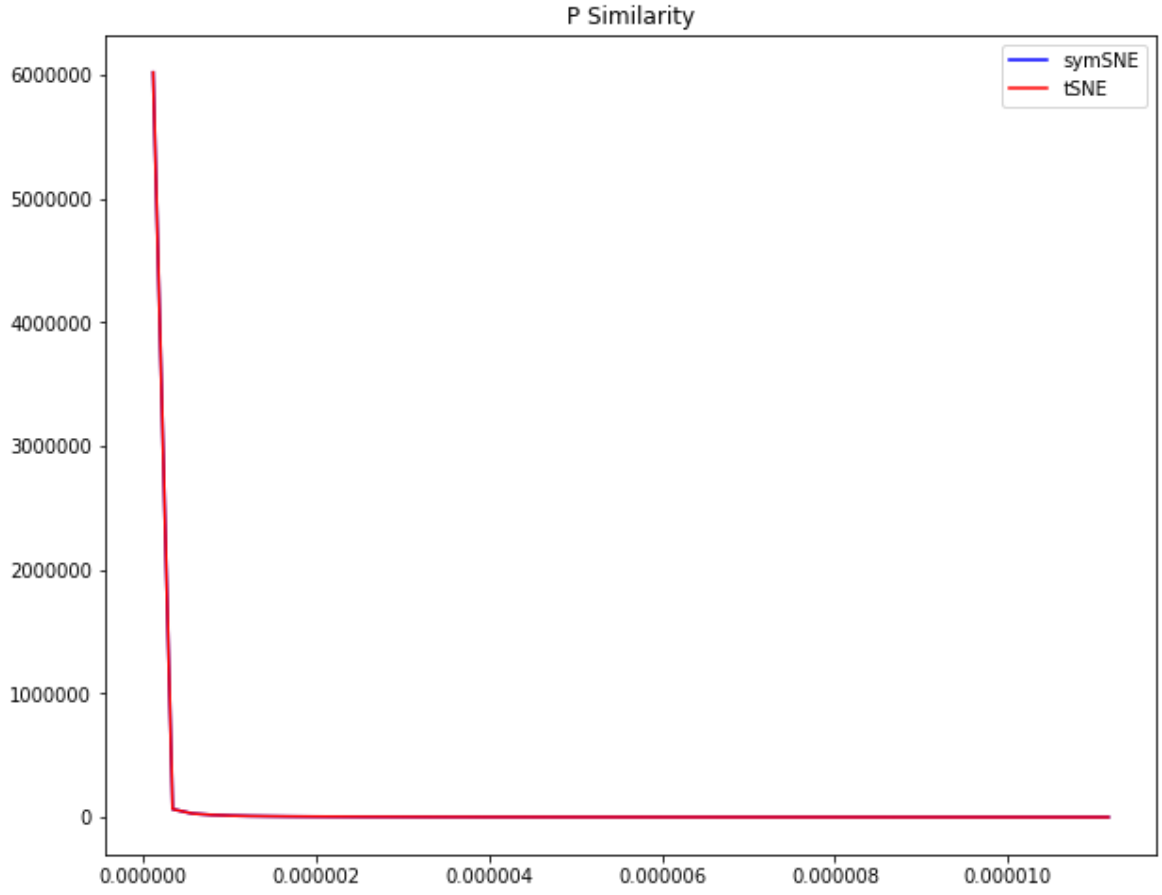


所以以下的分佈都有先取 log 再畫出來

P Similarity

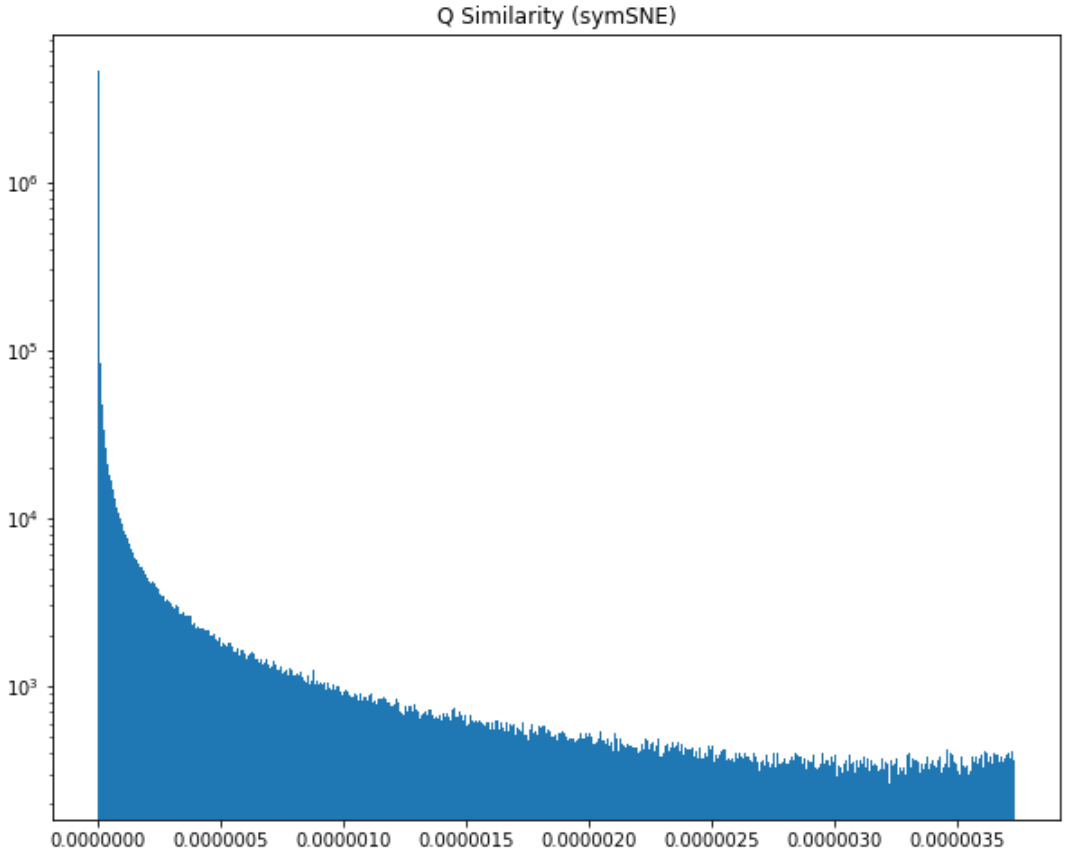


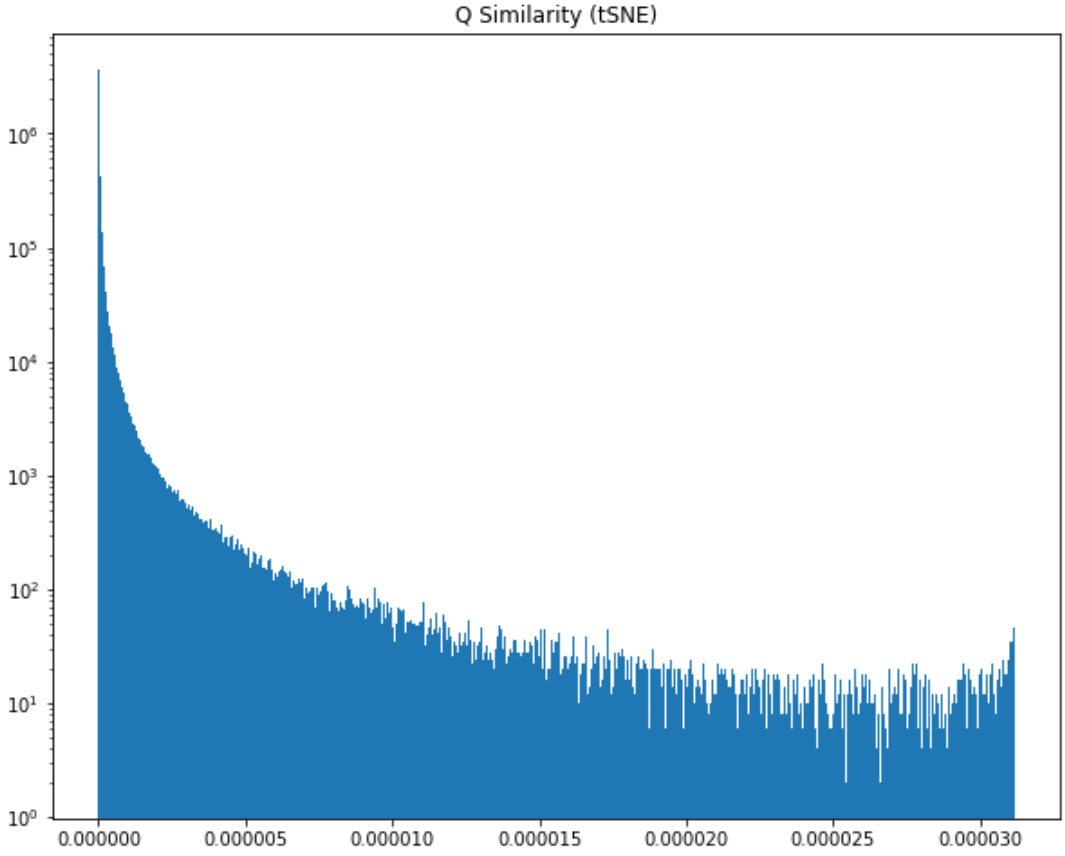


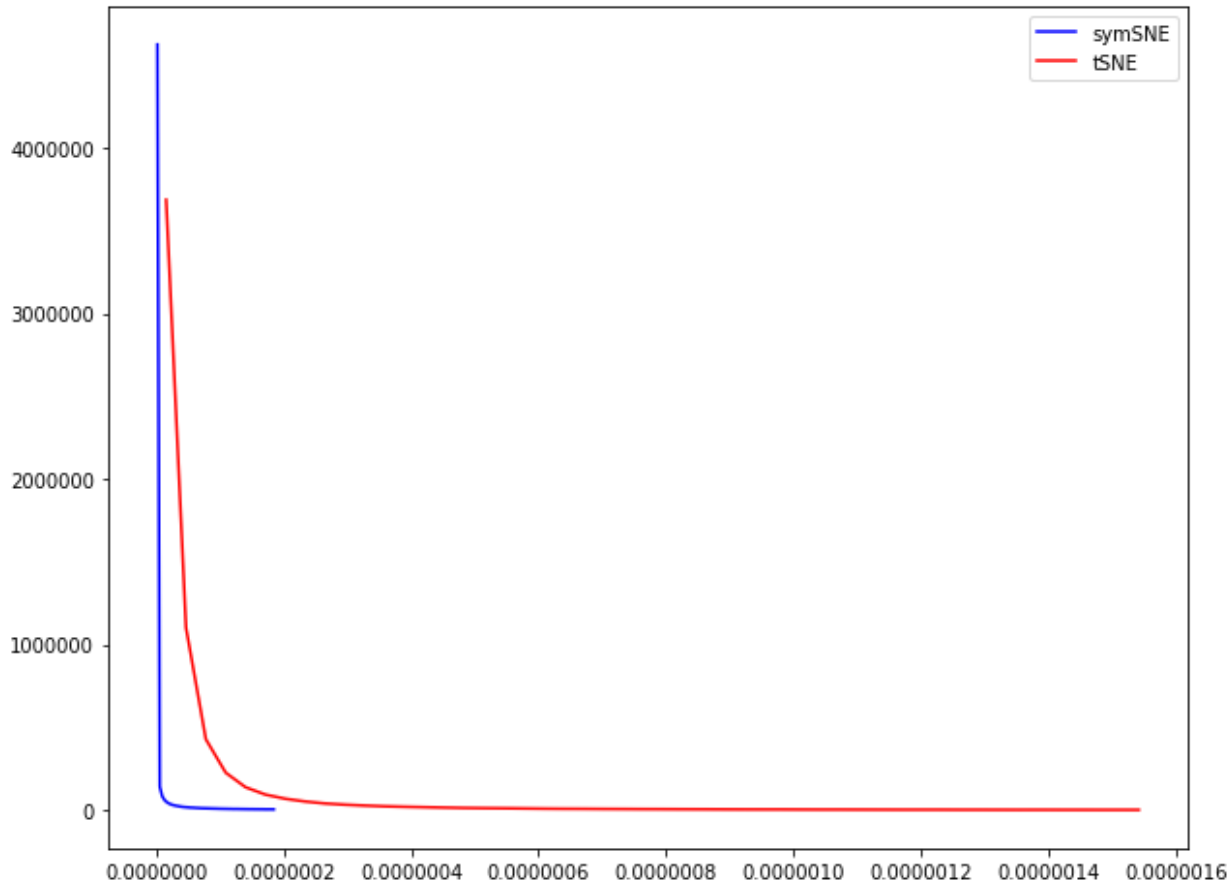


因為 P Similarity 的計算方式沒有改變，所以兩種方法的 P Similarity 會長一模一樣。

Q Similarity





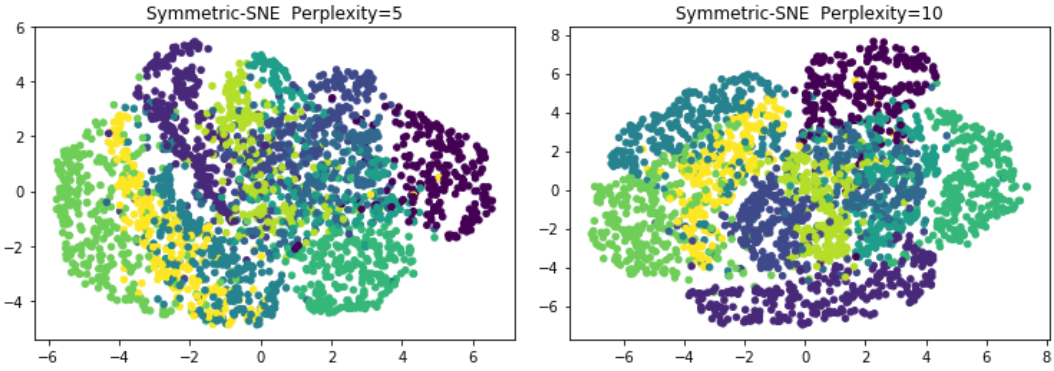


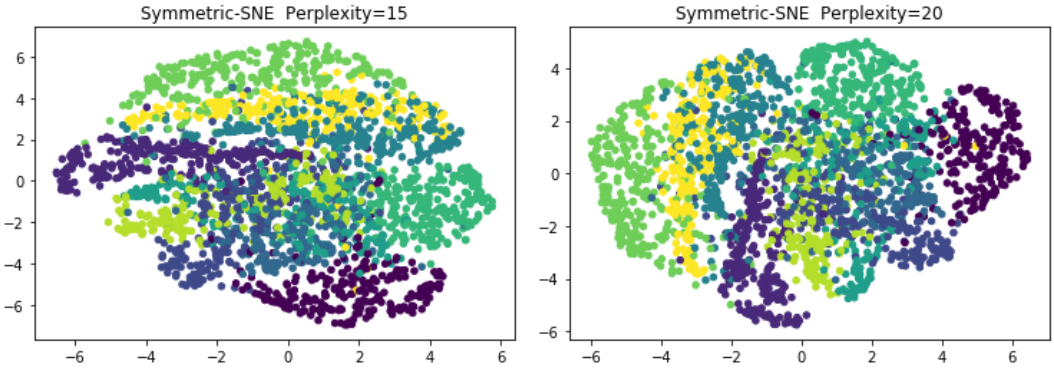
(x 軸為 Similarity，y 軸為 data 數量)

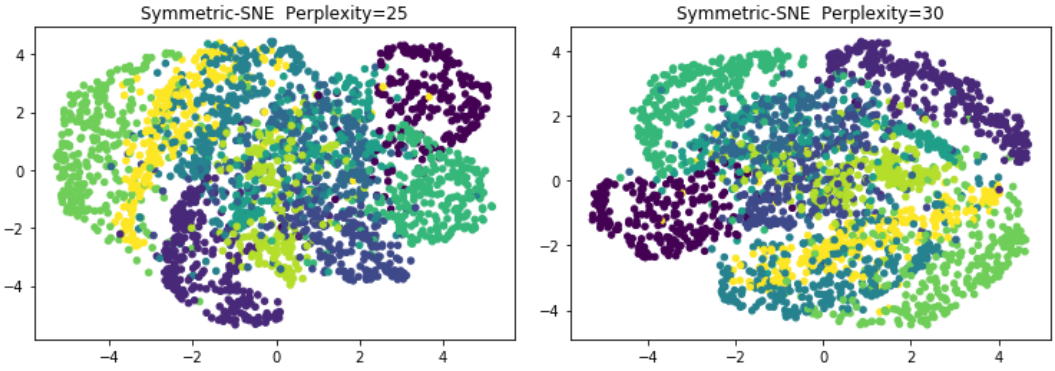
可以發現 Symmetric – SNE 的結果大多距離非常遠，因為大部分的 data Similarity 都非常接近 0。反觀 t – SNE，Similarity 為 0.0000002 的 data 數量其實還是蠻多的，代表距離近的 data 比 Symmetric – SNE 還多。

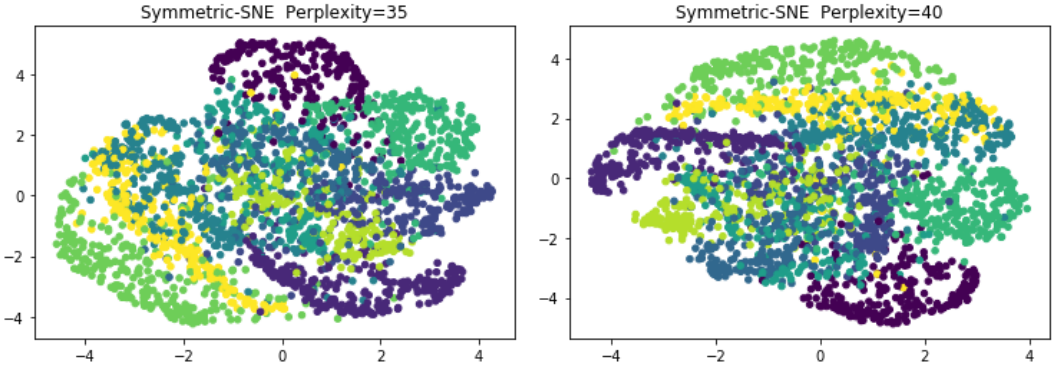
Different Perplexity

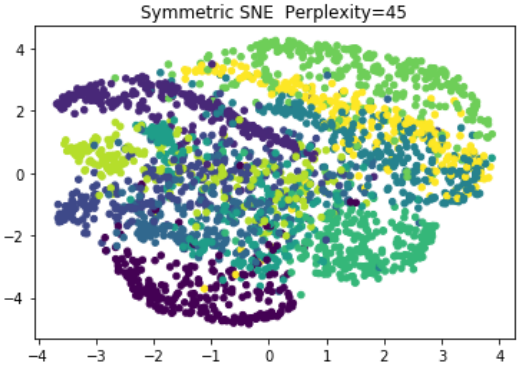
Symmetric – SNE

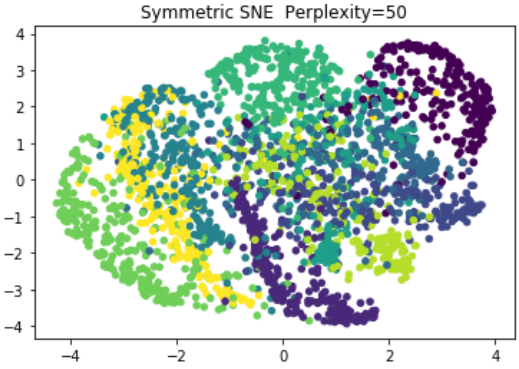


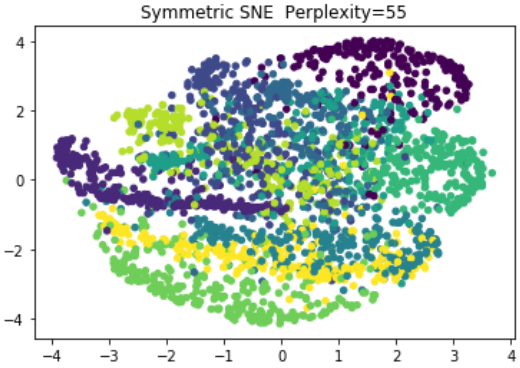


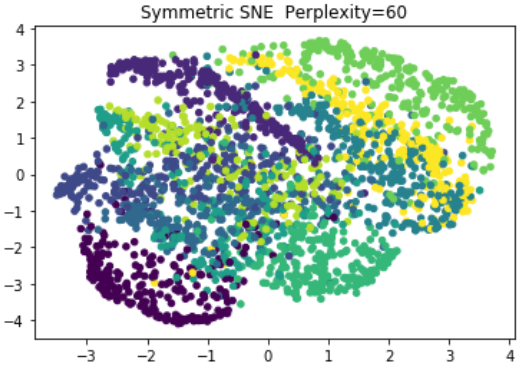


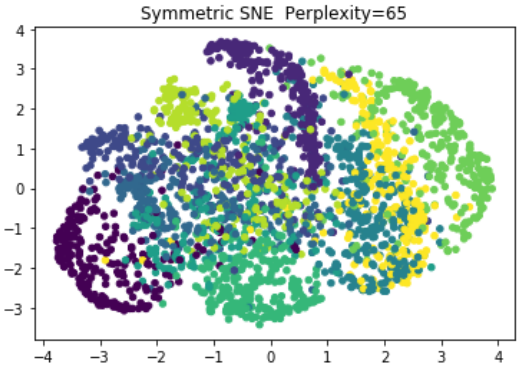


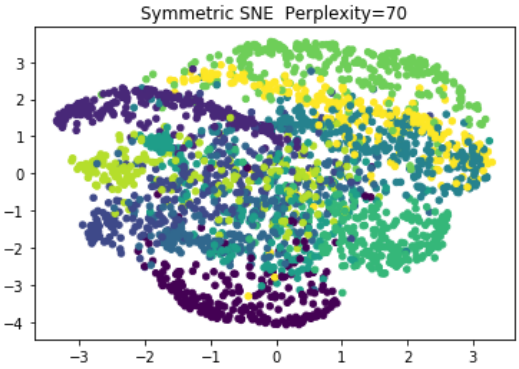

Symmetric SNE Perplexity=45

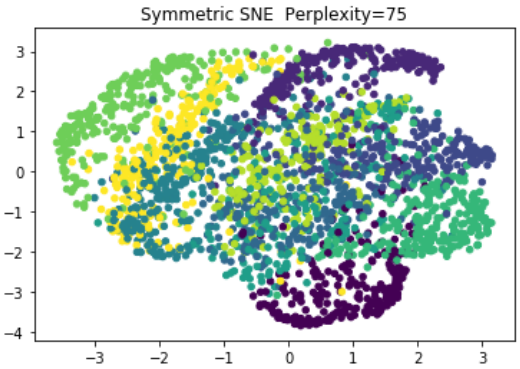

Symmetric SNE Perplexity=50

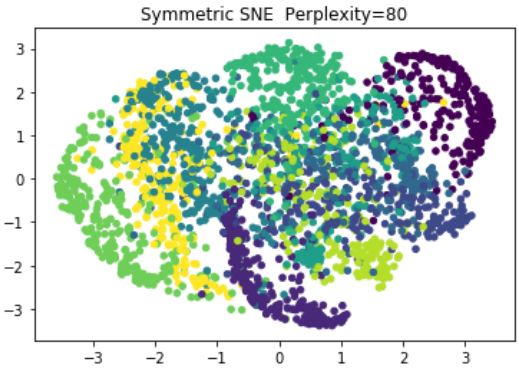

Symmetric SNE Perplexity=55

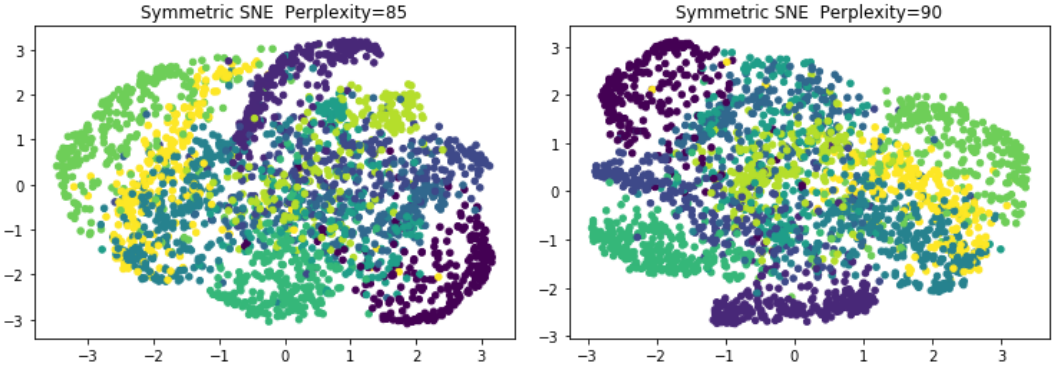

Symmetric SNE Perplexity=60


Symmetric SNE Perplexity=65

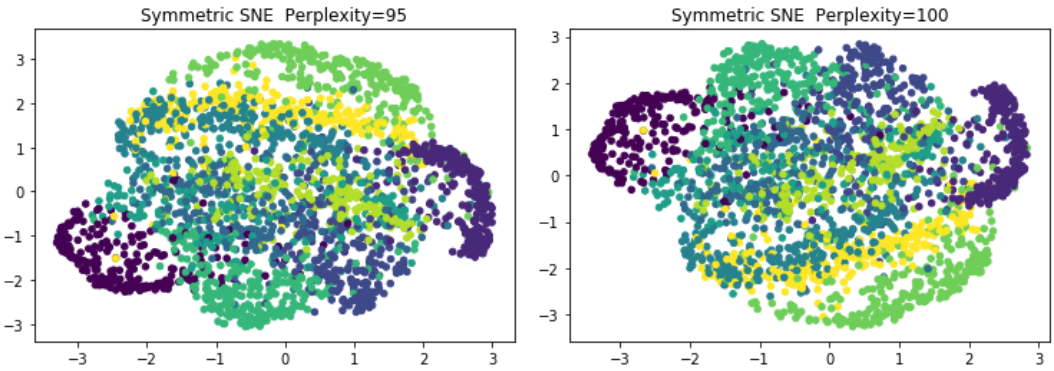

Symmetric SNE Perplexity=70


Symmetric SNE Perplexity=75


Symmetric SNE Perplexity=80

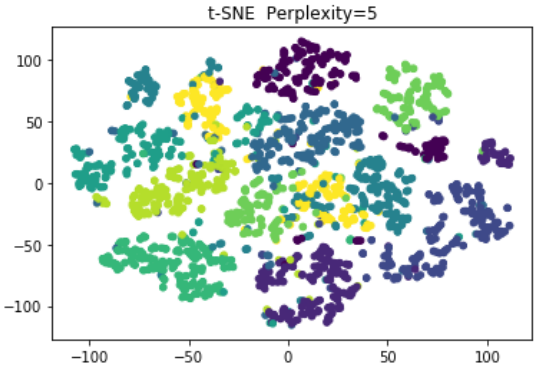

Symmetric SNE Perplexity=85

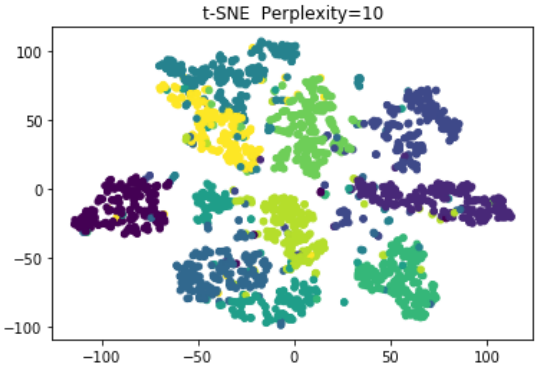
Symmetric SNE Perplexity=90

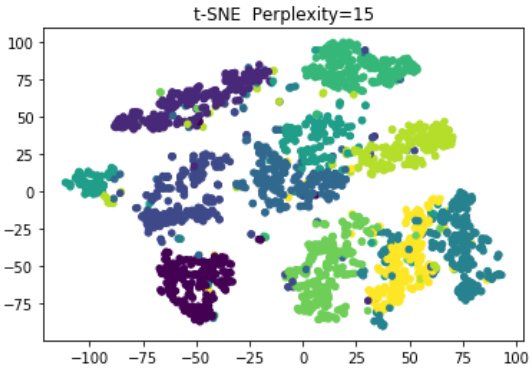

Symmetric SNE Perplexity=95

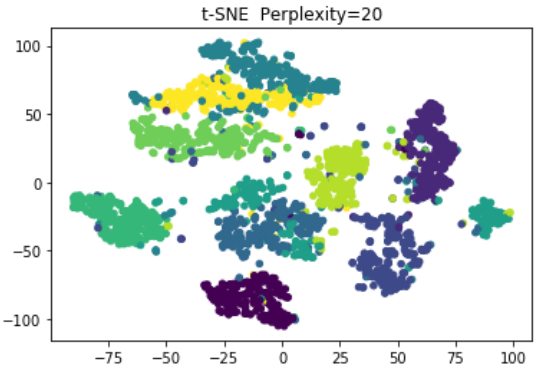
Symmetric SNE Perplexity=100

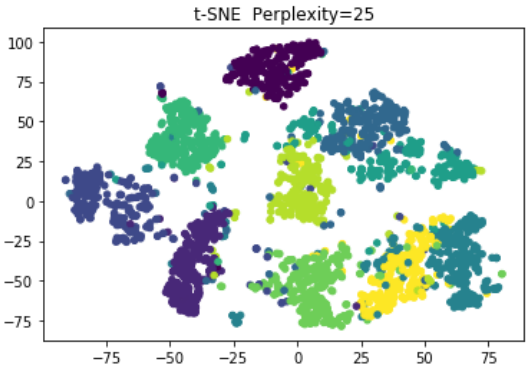
T - SNE

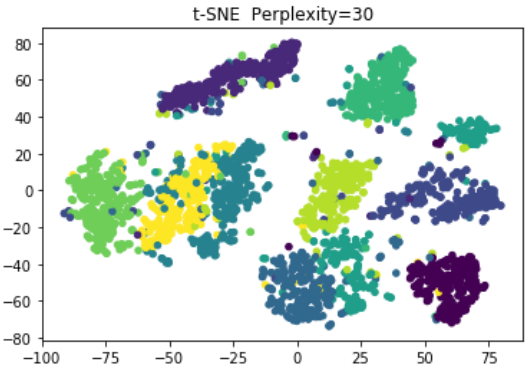

t-SNE Perplexity=5

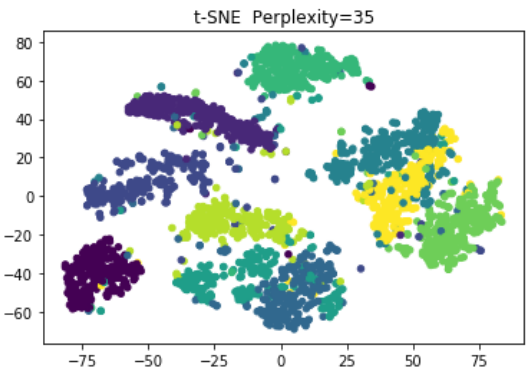

t-SNE Perplexity=10

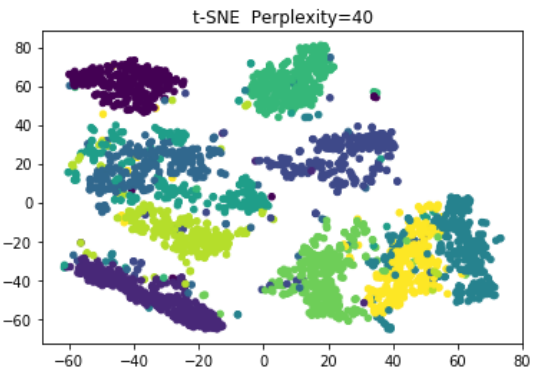

t-SNE Perplexity=15

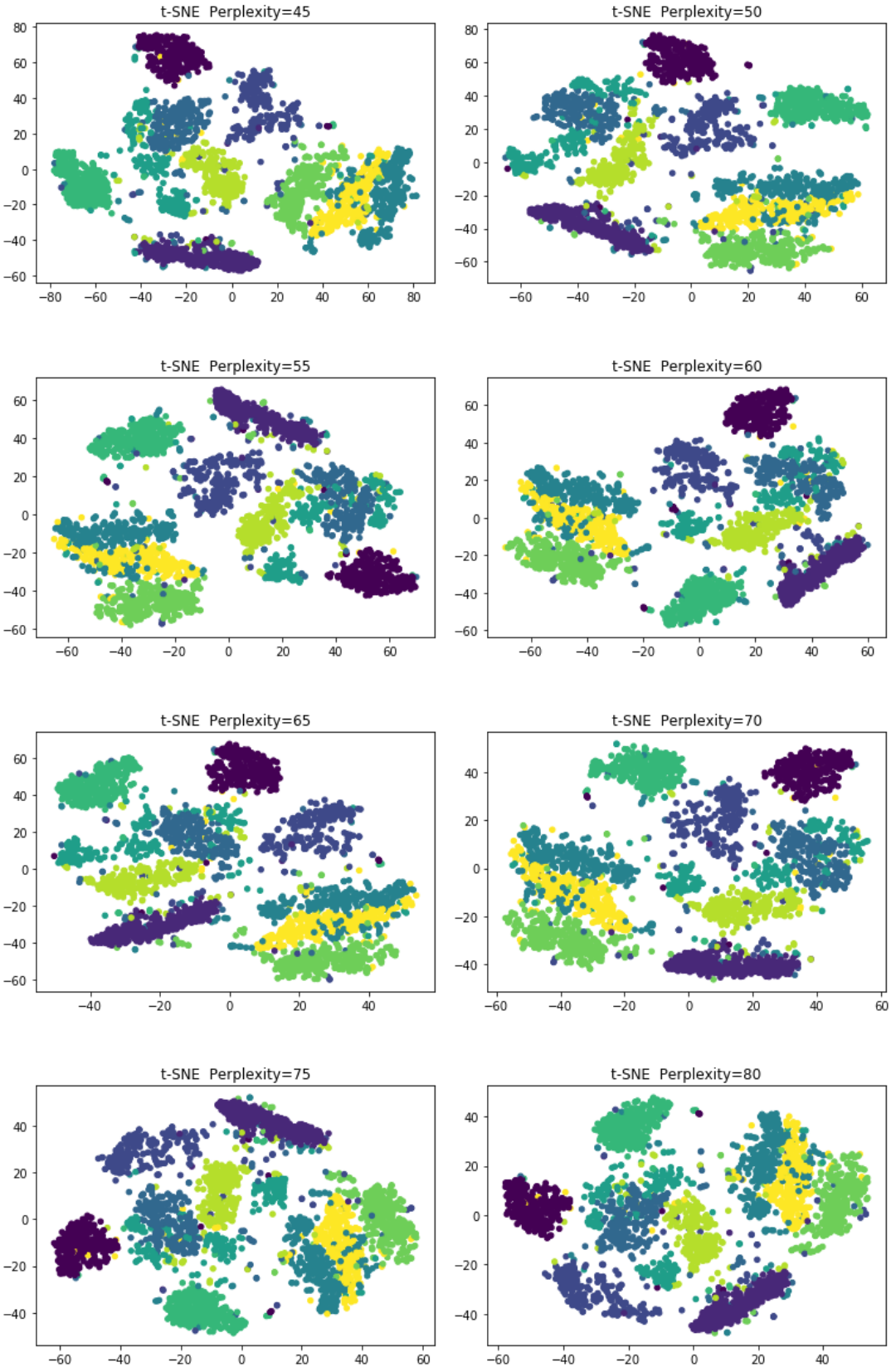

t-SNE Perplexity=20

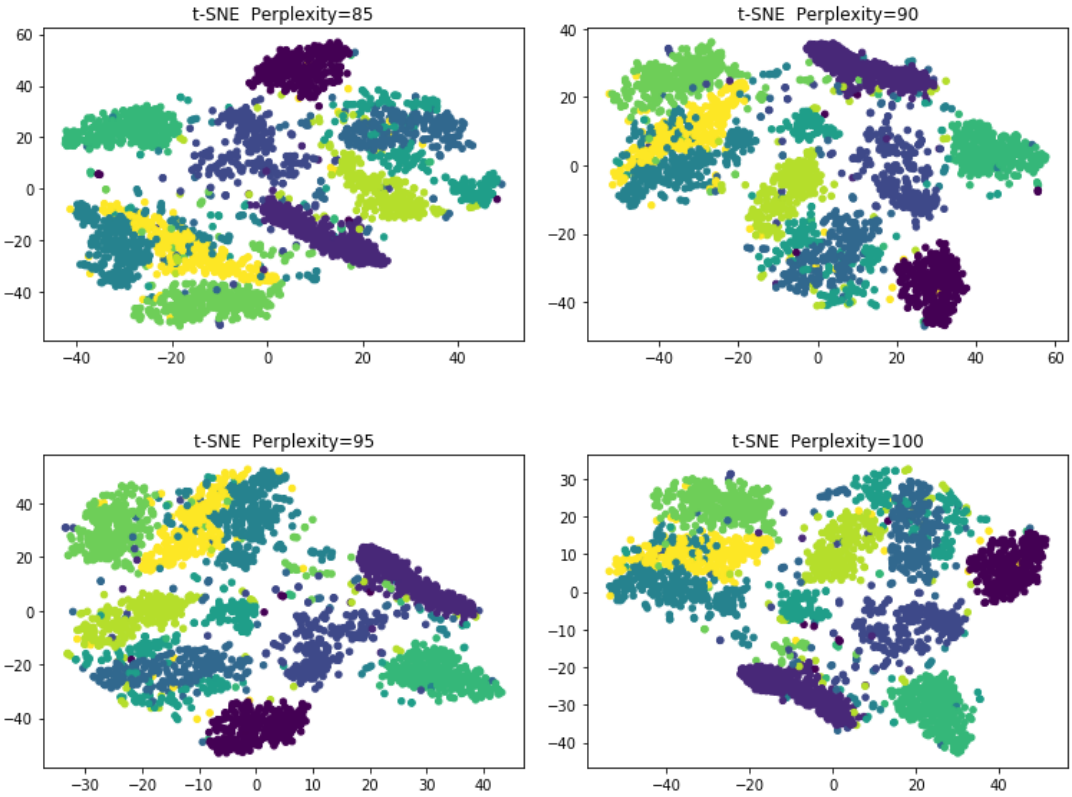

t-SNE Perplexity=25


t-SNE Perplexity=30


t-SNE Perplexity=35


t-SNE Perplexity=40


t-SNE Perplexity=45
t-SNE Perplexity=50
t-SNE Perplexity=55
t-SNE Perplexity=60
t-SNE Perplexity=65
t-SNE Perplexity=70
t-SNE Perplexity=75
t-SNE Perplexity=80



Symmetric – SNE 的結果不論 perplexity 調多大或多小基本上不太會有什麼差異，因為 Symmetric – SNE 本來就不會把每個 class 之間的距離拉大，所以 perplexity 從 5~100 看起來都沒什麼差異。

而 t – SNE 就可以蠻明顯的看得出來大概在 perplexity 為 15~40 時可以把每個 class 切割得很清楚，perplexity 太大或者太小的時候不同 class 混在一起的機率就比較高。但是 perplexity 在 5~100 的區間裡時，t – SNE 的結果確實是比 Symmetric – SNE 好的。