

Modified code:

x2p function 沒有做任何更改，因為 symmetric SNE 和 t-SNE 對 P 的算法都相同，另外就是加了一個坐 symmetric SNE 的 function:

```
def sne(X=np.array([]), no_dims=2, initial_dims=50, perplexity=40.0):
    # Check inputs
    if isinstance(no_dims, float):
        print("Error: array X should have type float.")
        return -1
    if round(no_dims) != no_dims:
        print("Error: number of dimensions should be an integer.")
        return -1

    # Initialize variables
    X = pca(X, initial_dims).real

    (n, d) = X.shape
    max_iter = 500
    momentum=0.5
    eta = 500
    Y = np.random.randn(n, no_dims)
    dY = np.zeros((n, no_dims))
    # Compute P-values
    P = x2p(X, 1e-5, perplexity)
    P = P + np.transpose(P)
    P = P / np.sum(P)
    P = P * 4. # early exaggeration
    P = np.maximum(P, 1e-12)
    # Run iterations

    Y_m2 = Y.copy()
    Y_m1 = Y.copy()
    for iter in range(max_iter):
        #print(Y)
        # Compute pairwise affinities
        sum_Y = np.sum(np.square(Y), 1)
        num = -2. * np.dot(Y, Y.T)
        num = np.add(np.add(num, sum_Y).T, sum_Y)
        num[range(n), range(n)] = 0.
        Q = np.exp(-num) / np.sum(np.exp(-num))

        Q = np.maximum(Q, 1e-12)
        # 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)
        dY=2.*dY

        Y = Y - eta* dY
        if momentum: # Add momentum
            Y += momentum * (Y_m1 - Y_m2)
            Y_m2 = Y_m1.copy()
            Y_m1 = Y.copy()
        if (iter + 1) % 10 == 0:
            C = np.sum(P * np.log(P / Q))
            print("Iteration %d: error is %f" % (iter + 1, C))

        # Stop lying about P-values
        if iter == 100:
            P = P / 4.

    # Return solution
    return Y
```

內容跟 sample code 裡的 t-SNE 有些地方重複，一樣的都先將 X 坐 PCA 再用 x2p 算出 P，但計算 Q 和 Y 的方法並不相同

```

Y_m2 = Y.copy()
Y_m1 = Y.copy()
for iter in range(max_iter):
    #print(Y)
    # Compute pairwise affinities
    sum_Y = np.sum(np.square(Y), 1)
    num = -2. * np.dot(Y, Y.T)
    num = np.add(np.add(num, sum_Y).T, sum_Y)
    num[range(n), range(n)] = 0.
    Q = np.exp(-num) / np.sum(np.exp(-num))

    Q = np.maximum(Q, 1e-12)
    # 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)
    dY=2.*dY
    Y = Y - eta* dY
    if momentum: # Add momentum
        Y += momentum * (Y_m1 - Y_m2)
        Y_m2 = Y_m1.copy()
        Y_m1 = Y.copy()

```

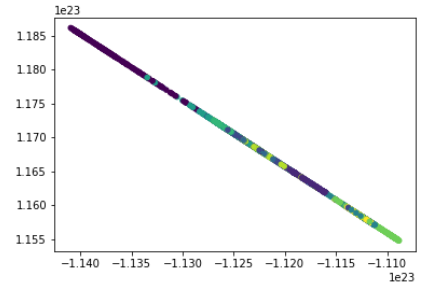
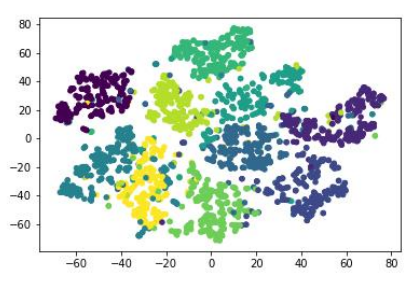
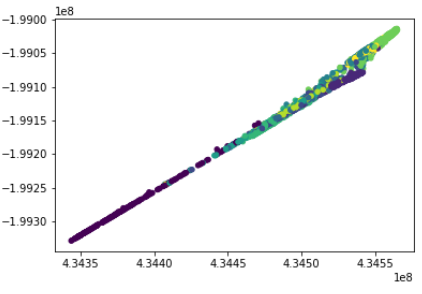
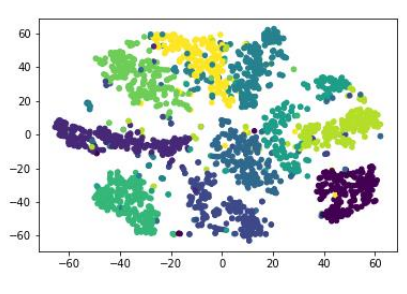
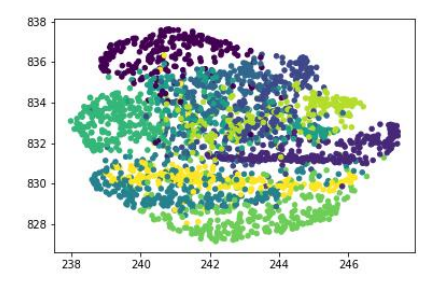
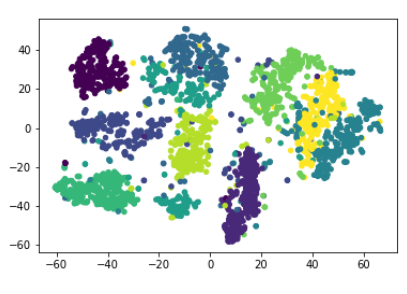
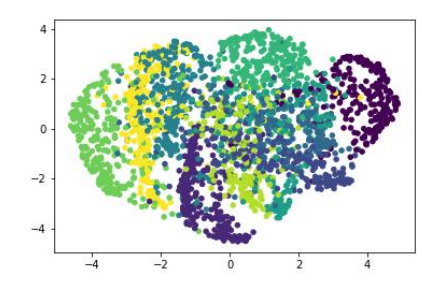
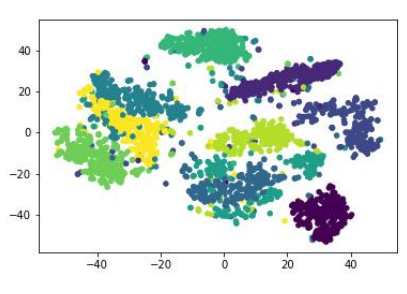
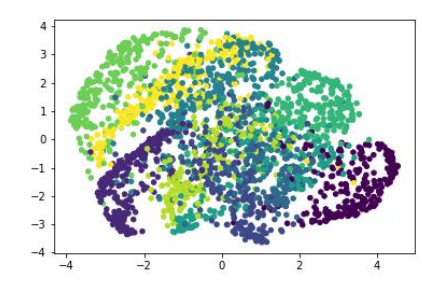
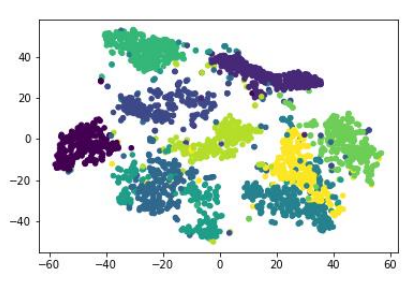
Y\_m2 和 Y\_m1 是到時候做 gradient descent 需要的參數

計算 Q 的 euclidean distance 也是用 sample code 給的方法，但這邊只要把算出來的 num 取 exponential 就好，並不需要乘上 beta

Gradient 的部分也跟 sample code 類似，只是不需要乘上  $(1 + \|y_i - y_j\|^2)^{-1}$

Y 的 update 方法我用  $Y^{(t)} = Y^{(t-1)} + \eta \frac{\delta C}{\delta Y} + \alpha(t)(Y^{(t-1)} - Y^{(t-2)})$

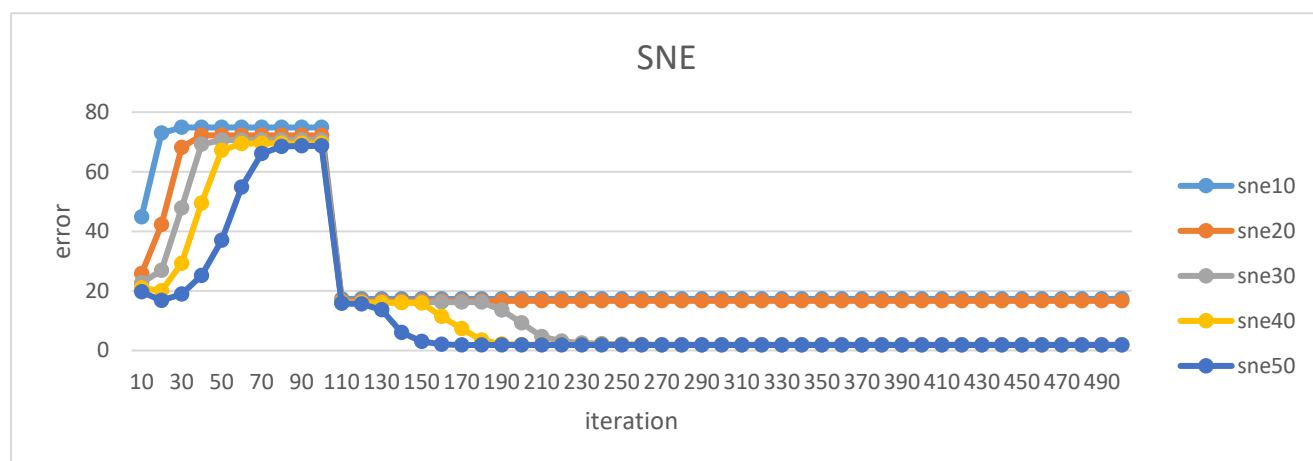
Performance:

	SNE	t-SNE
10		
20		
30		
40		
50		

表一、symmetric SNE 和 t-SNE ,perplexity=10-50 的 visualization

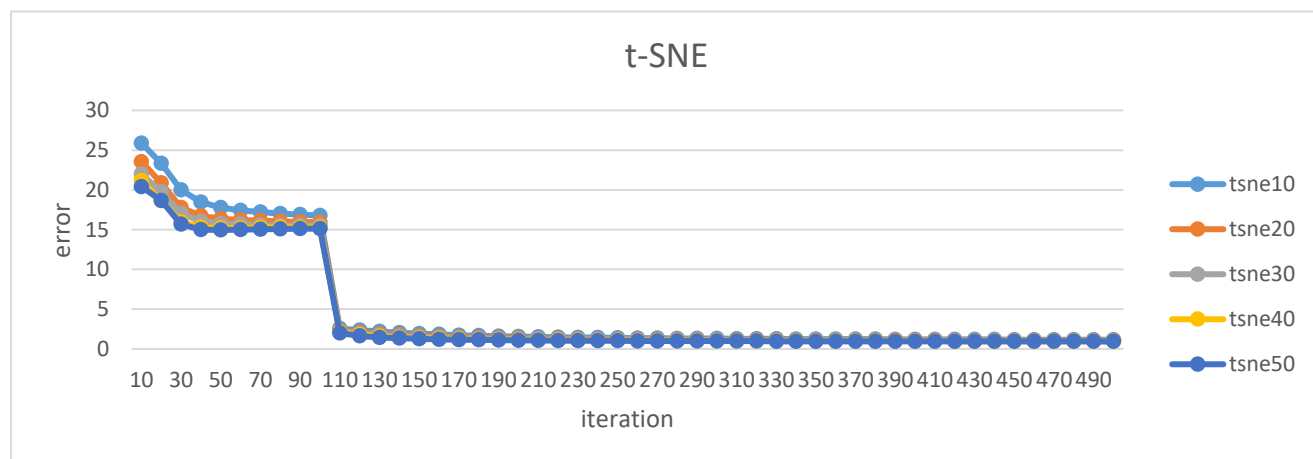
不管是 t-SNE 或 symmetric SNE 都可以做到將同 class 的 data 都分到同一區，但是 SNE 的 class 與 class 之間很難分開，不像 t-SNE 這樣分較開，因此有 crowding problem。

Perplexity 在這個範圍對 t-SNE 感覺沒有太大的影響，但是 symmetric SNE 的 perplexity 太低(=10,20)時，會接近成一條斜線，導致無法有效分類。



圖一、Symmetric SNE perplexity=10-50,iteration=500 的 cost function

10-100 次 iteration 的 error 會逐漸上升，越接近 100 斜率越小，100 次後會突然急速下降，應該是因為 sample code 的寫法當 iteraion=10 時  $P=P/4$  的關係，減少 P 的影響，讓 error 變小。



圖二、t-SNE perplexity=10-50,iteration=500 的 cost function

t-SNE iteration=10-100 時 error 就開始下降，越接近 100 越慢，一樣在 iteration=100 後快速下降到 2.多，因此後續的下降的速度會越來越慢。

	SNE p=10	SNE p=20	SNE p=30	SNE p=40	SNE p=50	t-SNE p=10	t-SNE p=20	t-SNE p=30	SNE p=10
10	44.78	25.80	22.70	20.97	19.65	25.89	23.54	21.99	21.17

	779	54	82	272	617	339	522	663	037
20	73.02	42.20	26.90	19.99	16.79	23.33	20.89	19.74	18.62
	304	472	408	286	016	279	636	181	819
30	74.85	68.16	47.80	29.22	18.90	19.97	17.78	17.05	15.98
	994	102	41	184	068	774	42	557	265
40	74.85	72.19	69.25	49.48	25.15	18.45	16.73	16.04	15.31
	994	868	403	036	505	285	898	274	069
50	74.85	72.19	70.62	67.14	37.00	17.77	16.35	15.77	15.20
	994	868	825	418	4	829	676	213	323
60	74.85	72.19	70.62	69.44	54.85	17.43	16.21	15.67	15.22
	994	868	879	491	283	998	589	17	026
70	74.85	72.19	70.62	69.50	66.10	17.20	16.12	15.58	15.27
	994	868	879	957	005	544	288	526	325
80	74.85	72.19	70.62	69.51	68.43	17.03	16.03	15.53	15.30
	994	868	879	072	587	779	743	826	836
90	74.85	72.19	70.62	69.51	68.62	16.90	15.97	15.51	15.29
	994	868	879	072	888	23	413	911	175
100	74.85	72.19	70.62	69.51	68.63	16.79	15.92	15.50	15.26
	994	868	879	072	935	005	902	836	903
110	17.32	16.66	16.27	15.99	15.76	2.564	2.319	2.186	2.115
	869	338	09	139	882	395	46	745	809
120	17.32	16.66	16.27	15.99	15.62	2.374	2.081	1.905	1.781
	869	338	09	139	659	713	059	317	619
130	17.32	16.66	16.27	15.99	13.66	2.203	1.894	1.719	1.580
	869	338	09	139	385	673	665	158	218
140	17.32	16.66	16.27	15.98	6.062	2.053	1.753	1.589	1.451
	869	338	09	486	552	857	892	145	489
150	17.32	16.66	16.27	15.88	3.058	1.927	1.645	1.494	1.361
	869	338	09	317	229	428	8	013	343
160	17.32	16.66	16.27	11.36	2.108	1.822	1.561	1.421	1.295
	869	338	09	178	623	447	709	823	262
170	17.32	16.66	16.26	7.323	1.864	1.735	1.494	1.365	1.245
	869	338	525	785	685	286	438	118	118
180	17.32	16.66	16.21	3.479	1.833	1.662	1.439	1.319	1.205
	869	338	456	313	667	718	422	761	309
190	17.32	16.66	13.54	2.222	1.826	1.602	1.393	1.282	1.172
	869	338	081	921	626	532	761	412	888
20	17.32	16.66	9.215	2.006	1.821	1.552	1.355	1.251	1.146

0	869	338	746	609	833	815	373	171	068
21	17.32	16.66	4.610	1.948	1.817	1.511	1.322	1.224	1.123
0	869	338	503	728	962	293	802	733	559
22	17.32	16.66	3.103	1.930	1.814	1.475	1.294	1.202	1.104
0	869	338	158	881	889	528	906	026	405
23	17.32	16.66	2.501	1.919	1.812	1.443	1.270	1.182	1.087
0	869	338	01	382	882	876	81	457	912
24	17.32	16.66	2.236	1.912	1.811	1.415	1.249	1.165	1.073
0	869	338	053	727	756	813	89	519	629
25	17.32	16.66	2.082	1.909	1.811	1.390	1.231	1.150	1.061
0	869	338	767	007	162	917	555	657	2
26	17.32	16.66	2.005	1.904	1.810	1.368	1.215	1.137	1.050
0	869	338	335	597	83	766	321	489	258
27	17.32	16.66	1.978	1.895	1.810	1.348	1.200	1.125	1.040
0	869	338	505	085	62	98	827	742	653
28	17.32	16.66	1.972	1.883	1.810	1.331	1.187	1.115	1.032
0	869	338	052	019	47	084	786	349	125
29	17.32	16.66	1.970	1.879	1.810	1.314	1.176	1.106	1.024
0	869	338	558	145	352	768	026	021	473
30	17.32	16.66	1.969	1.877	1.810	1.299	1.165	1.097	1.017
0	869	338	785	36	256	859	419	56	584
31	17.32	16.66	1.969	1.876	1.810	1.286	1.155	1.089	1.011
0	869	338	194	219	175	232	795	741	388
32	17.32	16.66	1.968	1.875	1.810	1.273	1.147	1.082	1.005
0	869	338	746	417	103	803	082	769	802
33	17.32	16.66	1.968	1.874	1.810	1.262	1.139	1.076	1.000
0	869	338	372	825	04	298	103	438	716
34	17.32	16.66	1.968	1.874	1.809	1.251	1.131	1.070	0.996
0	869	338	052	375	982	572	772	594	056
35	17.32	16.66	1.967	1.874	1.809	1.241	1.125	1.065	0.991
0	869	338	789	024	926	669	035	299	82
36	17.32	16.66	1.967	1.873	1.809	1.232	1.118	1.060	0.987
0	869	338	575	748	87	483	785	437	845
37	17.32	16.66	1.967	1.873	1.809	1.223	1.112	1.055	0.984
0	869	338	402	526	811	904	993	93	311
38	17.32	16.66	1.967	1.873	1.809	1.215	1.107	1.051	0.981
0	869	338	265	344	745	896	62	733	121
39	17.32	16.66	1.967	1.873	1.809	1.208	1.102	1.047	0.978

0	869	338	155	193	669	391	62	841	177
40	17.32	16.66	1.967	1.873	1.809	1.201	1.097	1.044	0.975
0	869	338	066	062	575	342	96	223	434
41	17.32	16.66	1.966	1.872	1.809	1.194	1.093	1.040	0.972
0	869	338	995	945	447	696	614	841	635
42	17.32	16.66	1.966	1.872	1.809	1.188	1.089	1.037	0.970
0	869	338	937	837	262	421	565	676	021
43	17.32	16.66	1.966	1.872	1.808	1.182	1.085	1.034	0.967
0	869	338	888	732	96	485	771	723	827
44	17.32	16.66	1.966	1.872	1.808	1.176	1.082	1.031	0.965
0	869	338	848	625	383	867	197	968	876
45	17.32	16.66	1.966	1.872	1.807	1.171	1.078	1.029	0.964
0	869	338	815	513	071	54	83	383	084
46	17.32	16.66	1.966	1.872	1.804	1.166	1.075	1.026	0.962
0	869	331	787	391	337	499	657	954	39
47	17.32	16.66	1.966	1.872	1.801	1.161	1.072	1.024	0.960
0	869	338	762	258	956	732	668	66	768
48	17.32	16.66	1.966	1.872	1.800	1.157	1.069	1.022	0.959
0	869	338	742	104	822	2	843	485	238
49	17.32	16.66	1.966	1.871	1.800	1.152	1.067	1.020	0.957
0	869	338	724	932	229	897	177	422	79
50	17.32	16.66	1.966	1.871	1.799	1.148	1.064	1.018	0.956
0	869	338	708	745	875	79	659	476	418

表二、symmetric SNE 和 t-SNE ,perplexity=10-50 iteration=0-500 次的 cost function