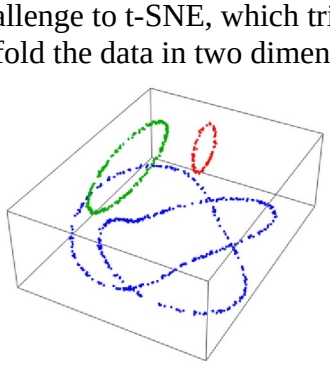
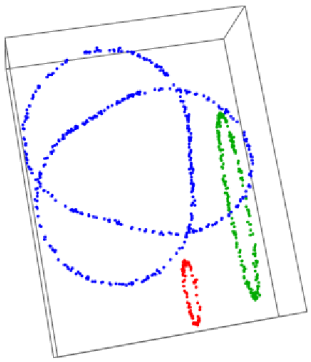
LCPB 20-21 exercise 4 (data visualization with t-SNE and clustering with DBSCAN)

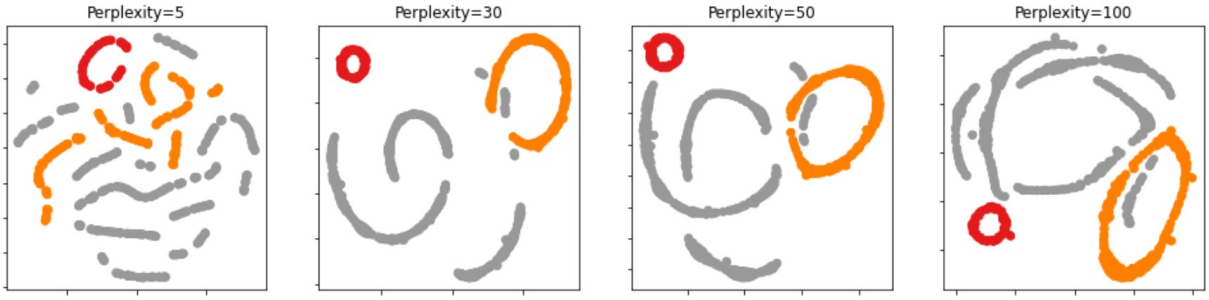
PART 1

1. Read data files as “data_t-SNE_310101_d5_R100_e1_N800.dat” provided in the google folder, which contain high dimensional data (d=5 in this case, with columns separated by the tab “\t”) with embedded manifolds as those in the figure, which represent three clusters with a linear closed structure. Given N data points, the first 10% belongs to cluster “0” (red), the next 30% to cluster “1” (green) and the last 60% to cluster “2” (blue). The green cluster is linked to the blue one, in the 3d representation. This is expected to challenge the convergence of t-SNE. The knotted structure of the blue cluster should introduce a similar challenge to t-SNE, which tries to unfold the data in two dimensions.

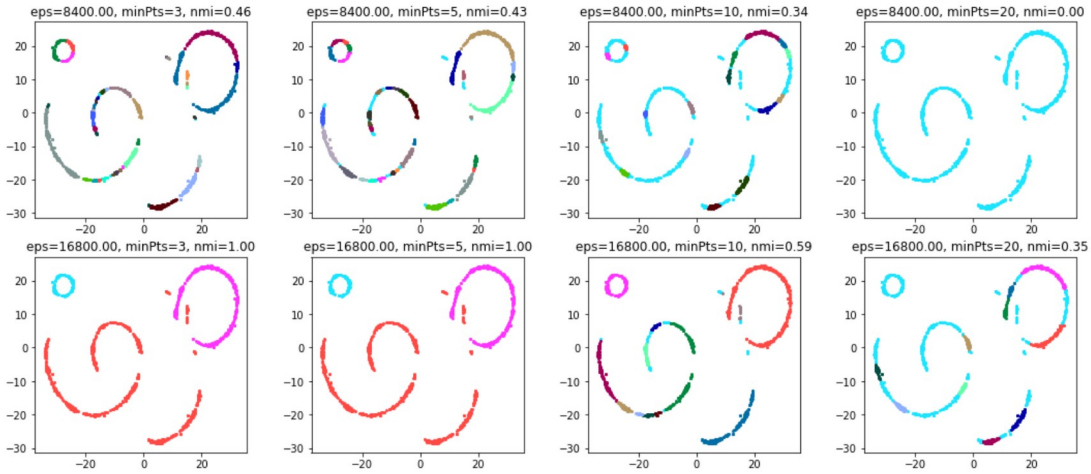




2. Apply t-SNE with 4 different perplexities to the data. The method is one out of many methods available at scikit website:
https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html
In t-SNE one may initialize the algorithm by using principal component analysis (PCA). Also a 3D version is available in the packages.
The result for the 2D case should look like



3. From NB15 extract the part concerning the DBSCAN algorithm for clustering and embed it in the notebook to analyze the clustering in d=5 dimensions to generate predicted labels, then plot the results. Something like the next figure should arise. It includes a grid with several values of “eps” and “minPts”
To understand a good range for the cutoffs “eps” in DBSCAN, sort all minimum distances to first neighbors and plot them.



The figure specifies, for each panel, the corresponding value of “nmi” defined in the notebook by Mehta et al, which quantifies the difference between true labels and predicted ones.

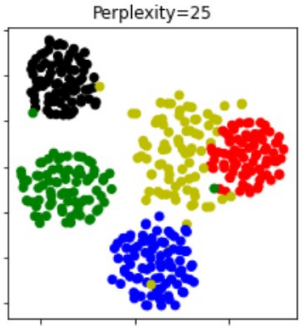
One needs to take care of introducing meaningful values of “eps” in DBSCAN: this parameter is the radius of the spheres around points, and one needs spheres to be reasonably full. In your notebook, find the typical scale separating points in d=5 from their first neighbors, and use it as a reference for “eps”, trying some multiples of that value.

PART 2

Repeat the t-SNE and DBSCAN analysis for the data generated with the notebook "exe04_1-generate.ipynb". These data look like

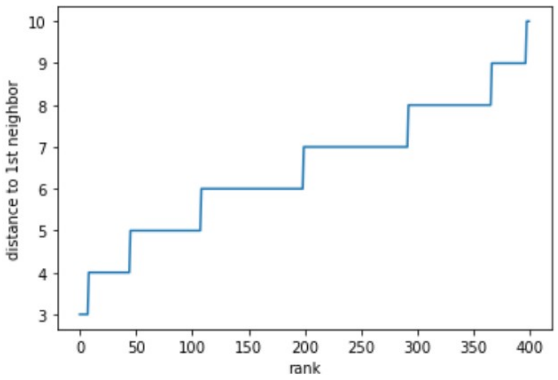
```
[0 1 1 1 1 1 1 0 1 0 1 0 0 1 1 0 1 1 1 0 0 0 1 0 0 1 1 0 0 1 1 1 1 0 0 1] 1
[1 1 1 1 1 1 0 0 1 0 0 1 1 1 1 0 1 1 1 1 0 1 1 0 0 0 0 1 1 1 0 0 1 0 1 0] 3
[0 0 1 1 0 1 0 0 0 1 1 0 0 0 1 1 0 1 1 0 1 1 1 0 0 0 1 0 0 1 0 1 0 1 0 0] 2
[0 1 1 1 0 1 0 0 0 0 1 1 0 1 1 0 1 1 0 0 1 1 1 1 0 0 0 1 0 1 0 0 1 1 0 1] 3
[0 1 0 1 1 1 0 1 0 1 1 1 1 0 0 1 0 0 0 0 1 1 0 0 1 0 1 0 1 1 0 1 1 0 0 1] 2
[0 1 1 0 1 1 0 0 1 1 1 1 0 1 1 0 1 1 1 0 0 0 0 0 1 1 1 0 0 1 1 1 0 1 1 0] 4
[0 0 1 1 0 1 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 1 0 0 0 1 1 1 1 0 0 0 1 0] 3
[0 0 1 1 1 1 1 1 0 1 1 0 0 1 0 0 1 0 1 1 1 0 1 0 0 1 1 0 0 0 1 1 0 0 1 0] 1
[0 0 1 0 1 1 1 1 1 1 1 1 0 1 1 0 0 1 0 0 0 0 0 1 1 1 0 1 0 0 1 0 1 0 1 0] 4
[1 1 1 0 0 1 0 1 1 1 0 1 1 0 0 1 0 0 1 0 0 1 0 1 0 0 1 0 1 0 1 1 0 0 0 0] 0
[1 1 1 1 0 1 0 0 1 1 1 1 1 1 1 0 1 0 0 1 1 1 0 0 0 0 0 1 1 1 1 1 0 0 0 1] 3
[0 0 0 0 1 1 0 1 0 0 1 1 0 0 1 0 1 1 1 1 1 1 0 0 0 0 1 1 0 0 1 1 1 0 1 1] 4
```

and should be read with option delimiter=' '.



The database contains samples with M=5 different labels y=0,1,2,3,4 (which in normal life would not be available, but here they are printed in the file "y_…" for checking the results). Each label corresponds to a specific enforced subsequence. One can check that y=0 is the best distinguishable one and y=4 is the most difficult to distinguish from the others.

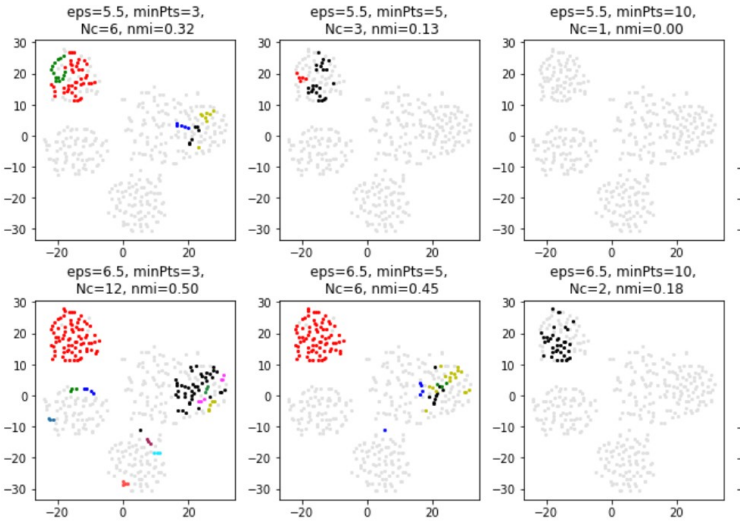
Why are points for y=1 (red) close to those with y=4 (yellow) in t-SNE?



To quantify the distance between two sequences, use the number of different bits. In DBSCAN this is obtained, for instance, using the option metric='l1' or metric='manhattan'. Again, to understand which is a good cutoffs "eps" for DBSCAN, sort all minimum distances to first neighbors and plot them as in the figure on the right.

In this second example you may find that t-SNE does perform well in grouping the clusters while DBSCAN does not. What is your explanation?

It is a situation in which DBSCAN applied to the data projected to 2D by t-SNE would work better than normal DBSCAN applied the 36-dimensional bit data, as one can check.



Here also the number Nc of clusters recognized by DBSCAN has been shown in the title of each panel (counting also label -1 for noise, light gray points).

Example of custom color map (to be used in t-SNE):

```
from matplotlib.colors import ListedColormap
mycol = ["k", "r", "g", "b","y"]
cmap = ListedColormap(mycol)
```