t-SNE, symmetric-SNE

HW# 3

0556623
Chieh Yu Chen
陳傑宇

1. Try to modify the code a little bit and make it back to symmetric SNE.

I copy the code "***tsne.py***", then rewrite the function ***tsne()***, with flag ***is_symmetric_sne*** to control whether the symmetric SNE or not (tSNE).

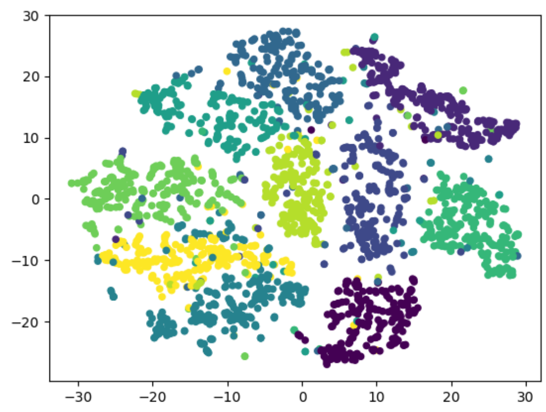
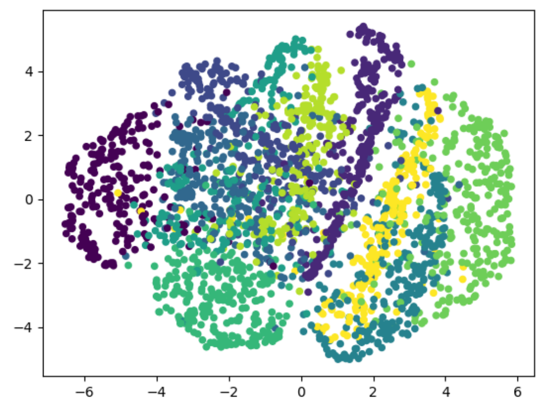
If the flag ***is_symmetric_sne*** enabled. According to the definition of symmetric SNE, I modify the definition of pairwise affinity (use Gaussian instead of t-Distribution) and the gradient computation part.

Reference of Symmetric SNE:

$$q_{ij} = \frac{\exp(- \mid\mid y_i - y_j \mid\mid^2)}{\sum_{k \neq l} \exp(- \mid\mid y_l - y_k \mid\mid^2)}$$

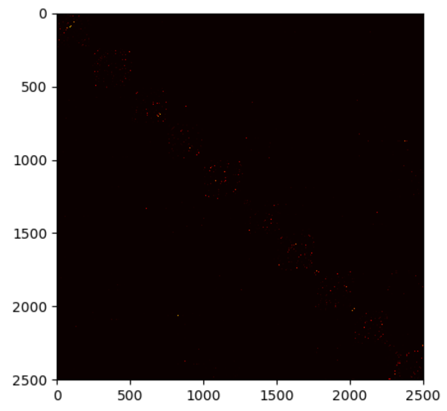
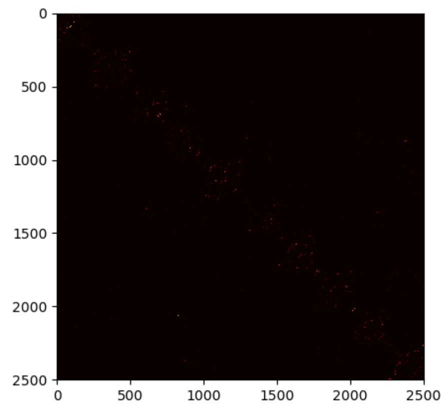
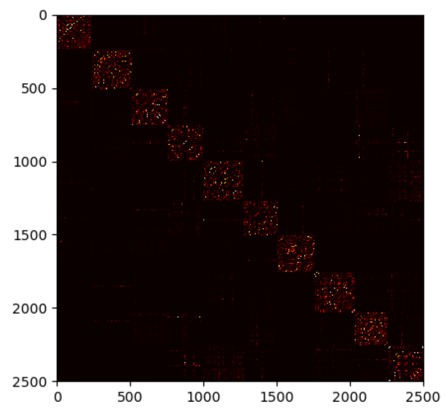
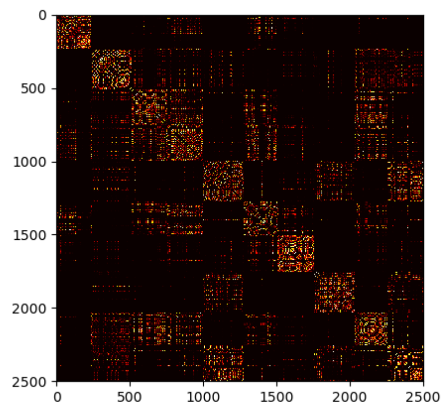
$$\frac{\partial C}{\partial y_i} = 2 \sum_j (p_{ij} - q_{ij})(y_i - y_j)$$

2. Try to visualize the embedding of both t-SNE and symmetric SNE and discuss their differences.

t-SNE	**symmetric SNE**
	

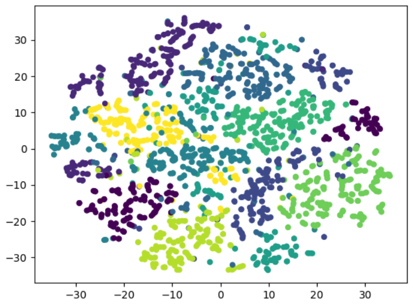
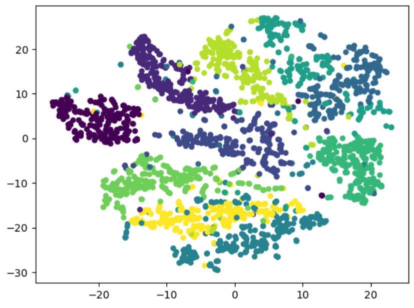
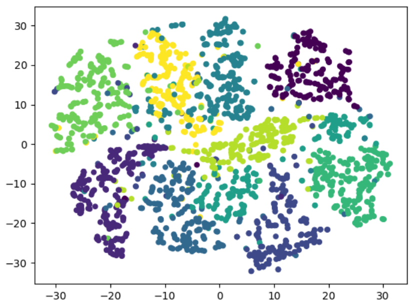
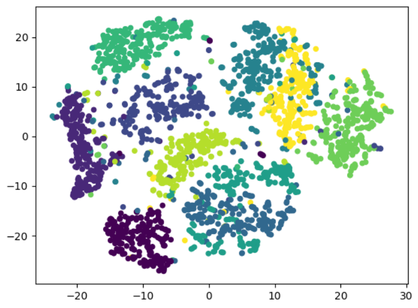
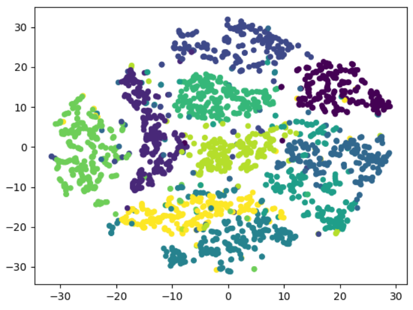
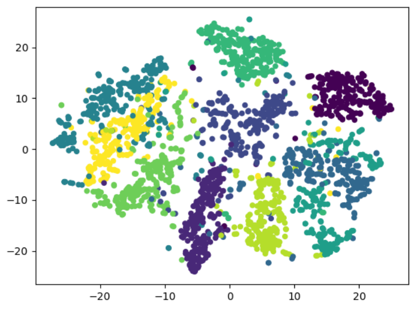
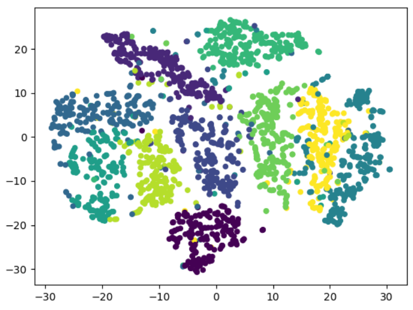
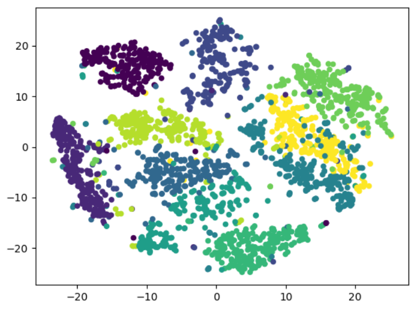
Both of embedding are with the setting (max_iter=200, perplexity=20, initial_dim=50). We can see very clearly the embedding of symmetric SNE has the crowding problems, which resulting in the occlusion problem, and the scale of output are also different, tSNE ranges from -30 to 30; symmetric SNE ranges from -5 to 5. In handwriting digits dataset, the embedding space of tSNE is much larger than symmetric SNE, which is also make sense to property of t-distribution and Gaussian distribution.

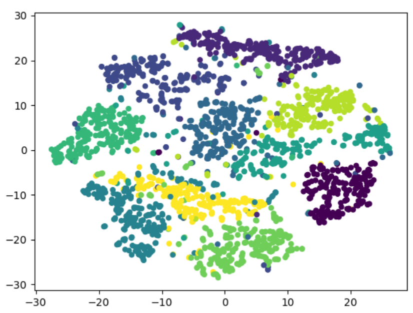
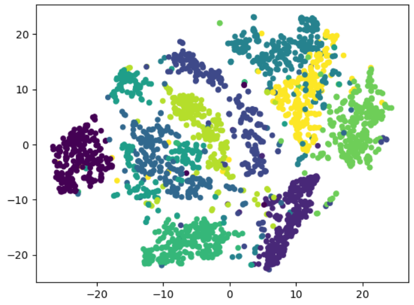
3. Try to visualize the distribution of pairwise similarities in both high-dimensional space and low-dimensional space, based on both t-SNE and symmetric SNE.

	t-SNE	**Symmetric-SNE**
High dim		
Low dim		

To better visualize the pairwise similarity, I reorder the data according to the labels data. So ideally we should see ten square clusters on the similarity matrix alike the low dimension representation of t-SNE. As for the low dimension representation of symmetric-SNE, besides the diagonal part we can see lots of relation between different digits, which will result in the occlusion in the 2D space. Without the visual result in 2D embedding, we can judge the quality of clustering base on the visual result of similarity (affinity) matrix.

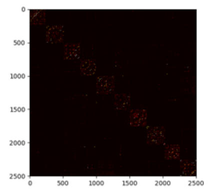
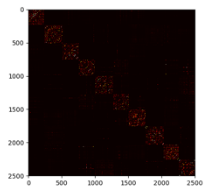
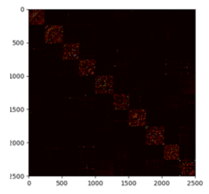
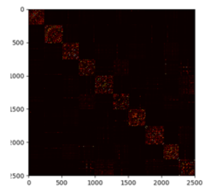
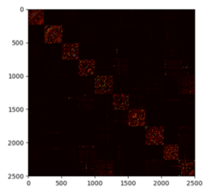
4. Try to play with different settings of perplexity, and see if there is any change in visualization.

Perp.	Result	Perp.	Result
5		30	
10		35	
15		40	
20		45	

25		50	

According to QA part on author's website, he suggests that typical value for perplexity should range from 5 to 50, so I try 10 experiment to analysis their difference.

By comparing all results collected, I found that the larger value of perplexity is used, the more compact result of cluster is shown. However, the tSNE algorithm is quite robust, only one result with perplexity 5 get bad result, others are good visualizations.

5	15	25	35	45
				

By comparing the low dimension embeddings, I found that the larger value of perplexity, the more occlusions happened (the non-diagonal area has higher value). In my opinions, the reason of it is because the perplexity is related to number of effective nearest neighbors, the more effective neighbors we compute, the more global visual result we get. (more like Euclidean measurement on space instead of geodesic measurement on manifold)

5. Implementation Details Explanations

In "***tSNE.py***", I modify the function ***tsne()*** with more parameters as below:

- **is_symmetric_sne** : boolean, whether to cluster data by symmetricSNE or tSNE.
- **labels** : data's label, used to visualize embedding colors and reorder the affinity matrix for the purpose of better visualization.
- **filename** : string, the name of figure of embedding result.
- **is_vis_by_iter** : boolean, whether to record the training process of SNE algorithm for every fixed amount of iterations.