Machine Learning HW7

0556562 陳鴻君

Overview

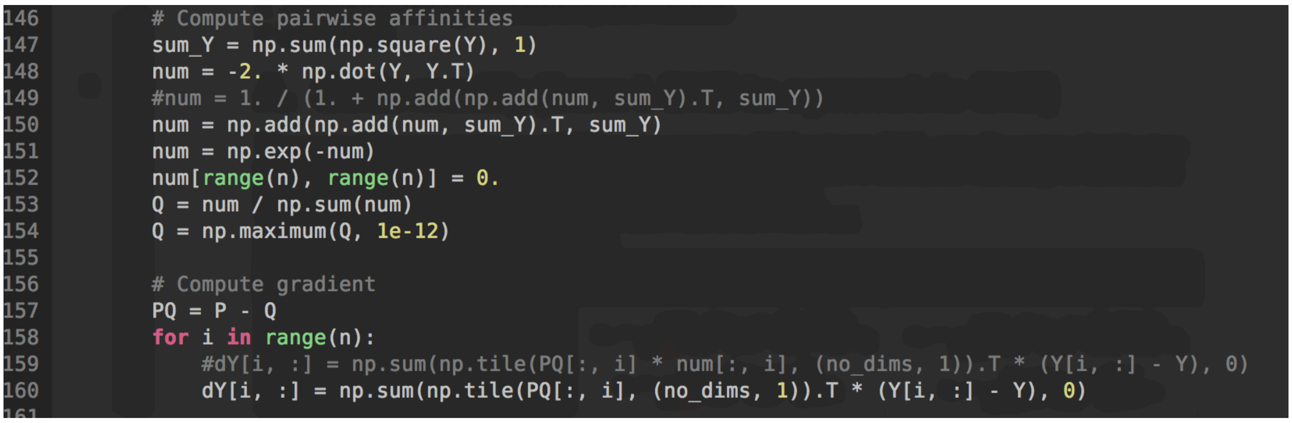
- Code modify (from t-SNE to symmetric SNE)
- Visualize and discuss their difference
- Try different settings of perplexity and visualize
- Conclusion

A. Code modify (from t-SNE to symmetric SNE)

The main difference between t-SNE and symmetric SNE is that t-SNE uses Gaussian distribution in high dimension and Student t distribution in low dimension. In the other hand, symmetric SNE uses Gaussian distribution in both high and low dimension.
Therefore, the calculation in low dimension has to be modified.
And also the gradient need to modify.

	symmetric SNE	**t-SNE**
pairwise similarities	$p_{ij} = \frac{\exp(-\parallel x_i - x_j \parallel^2 / (2\sigma^2))}{\sum_{k \neq l} \exp(-\parallel x_l - x_k \parallel^2 / (2\sigma^2))}$ $q_{ij} = \frac{\exp(-\parallel y_i - y_j \parallel^2)}{\sum_{k \neq l} \exp(-\parallel y_l - y_k \parallel^2)}$	$p_{ij} = \frac{\exp(-\parallel x_i - x_j \parallel^2 / (2\sigma^2))}{\sum_{k \neq l} \exp(-\parallel x_l - x_k \parallel^2 / (2\sigma^2))}$ $q_{ij} = \frac{(1 + \parallel y_i - y_j \parallel^2)^{-1}}{\sum_{k \neq l} (1 + \parallel y_i - y_j \parallel^2)^{-1}}$
gradient	$\frac{\partial C}{\partial y_i} = 2 \sum_j (p_{ij} - q_{ij})(y_i - y_j)$	$\frac{\delta C}{\delta y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)(1 + \parallel y_i - y_j \parallel^2)^{-1}$

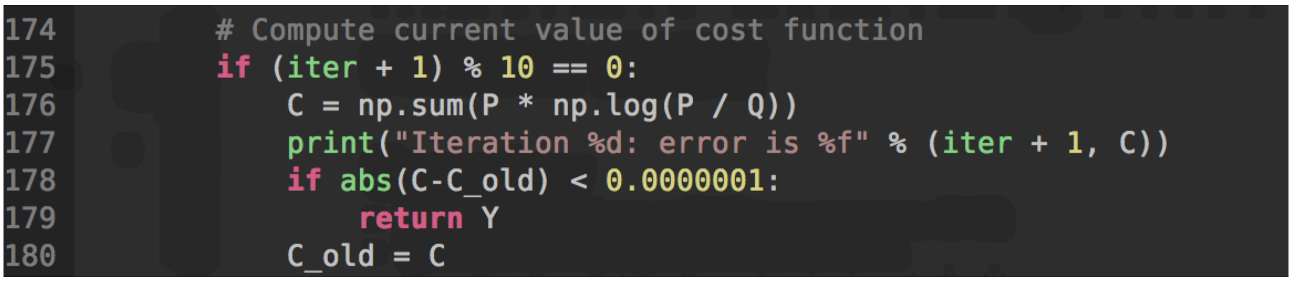
Implement in Python:
(line 149, 159 are the t-SNE version; line 150, 160 are symmetric SNE version)



```
146         # Compute pairwise affinities
147         sum_Y = np.sum(np.square(Y), 1)
148         num = -2. * np.dot(Y, Y.T)
149         #num = 1. / (1. + np.add(np.add(num, sum_Y).T, sum_Y))
150         num = np.add(np.add(num, sum_Y).T, sum_Y)
151         num = np.exp(-num)
152         num[range(n), range(n)] = 0.
153         Q = num / np.sum(num)
154         Q = np.maximum(Q, 1e-12)
155 
156         # Compute gradient
157         PQ = P - Q
158         for i in range(n):
159             #dY[i, :] = np.sum(np.tile(PQ[:, i] * num[:, i], (no_dims, 1)).T * (Y[i, :] - Y), 0)
160             dY[i, :] = np.sum(np.tile(PQ[:, i], (no_dims, 1)).T * (Y[i, :] - Y), 0)
```

In addition, I also add a converge condition beyond 1000 iterations which will return if the gain of C is lower than 0.0000001. (C_old is the cost in last round.)

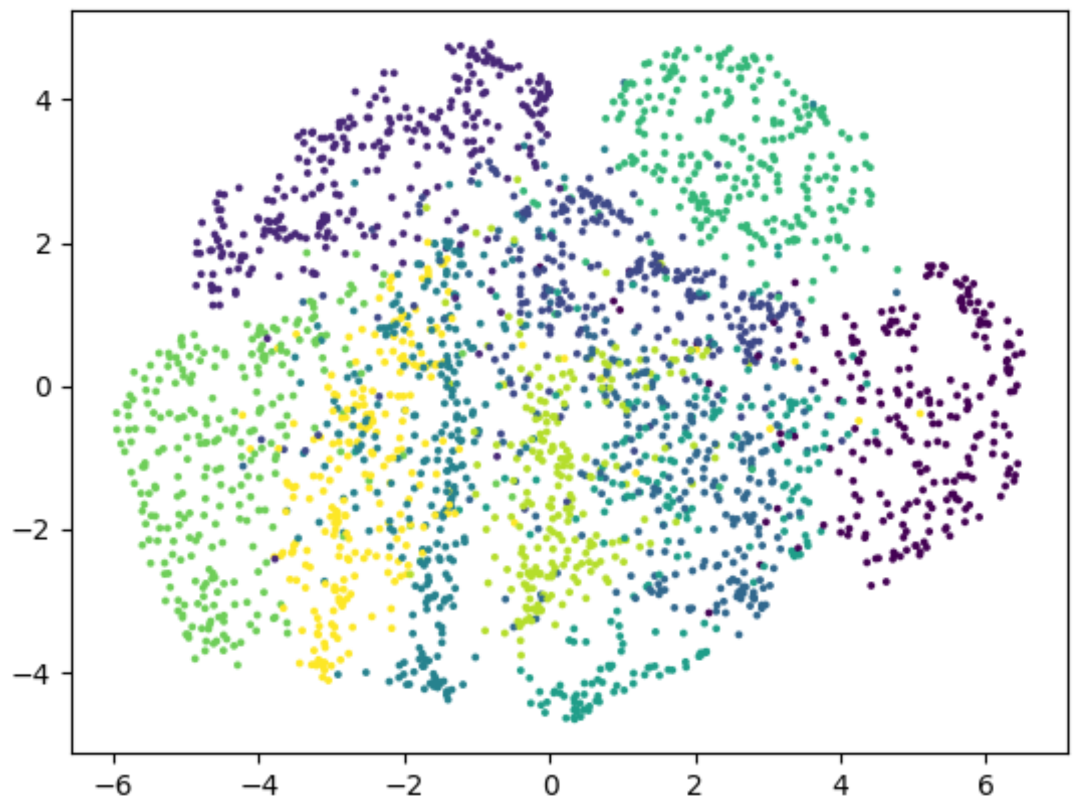
The reason why I add this condition is that I discover that symmetric SNE doesn't need 1000 iterations to converge.



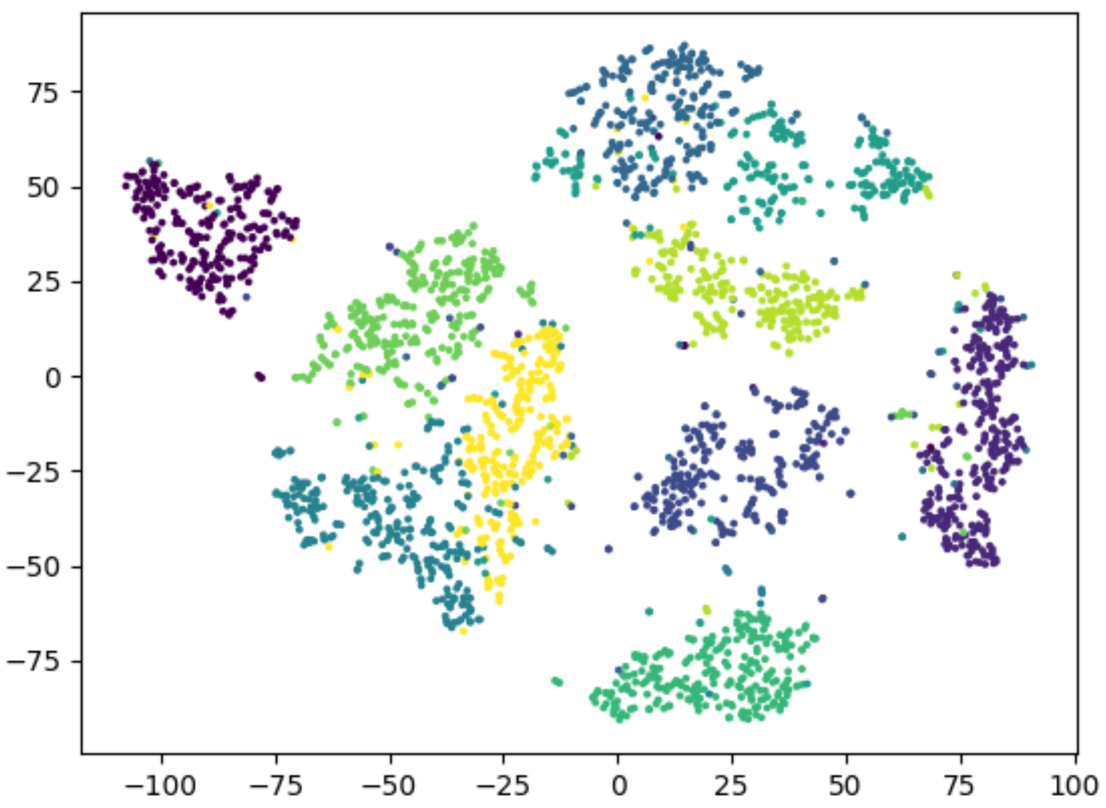
```python
        # Compute current value of cost function
        if (iter + 1) % 10 == 0:
            C = np.sum(P * np.log(P / Q))
            print("Iteration %d: error is %f" % (iter + 1, C))
            if abs(C-C_old) < 0.0000001:
                return Y
            C_old = C
```

B. Visualize and discuss their difference

Visualization of symmetric SNE:

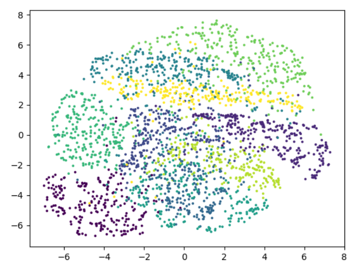
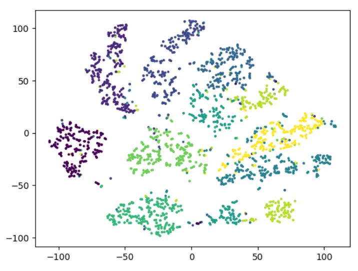
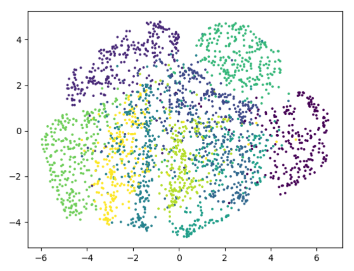
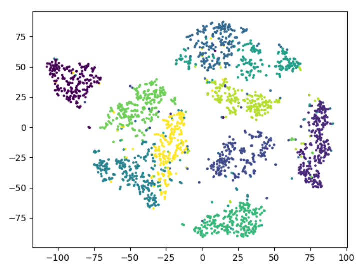
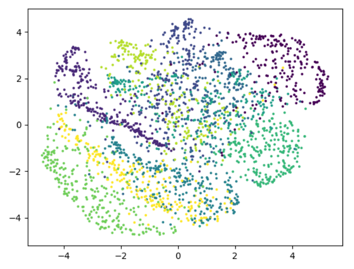
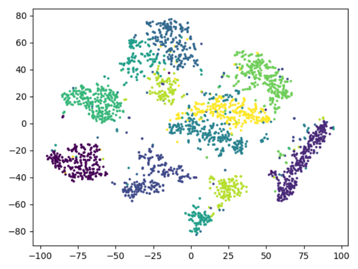
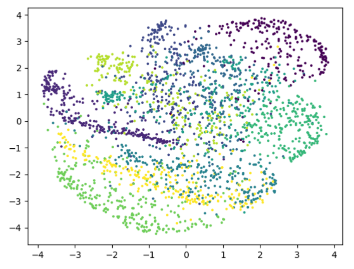
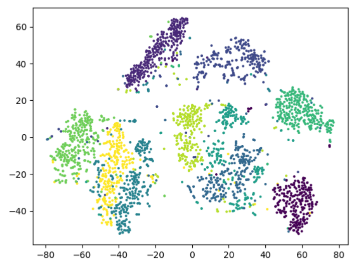
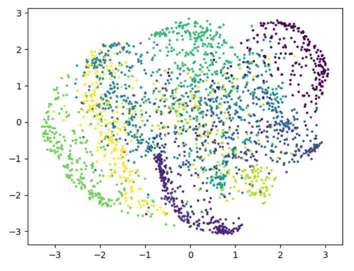
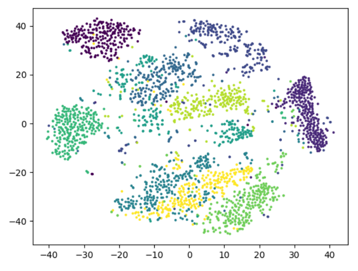


Visualization of t-SNE:



Discussion:
As the visualization results, the distance between groups is more separated in t-SNE. In other words, the distance between groups is more crowded in symmetric SNE.
Actually this is the major idea that t-SNE is designed. The crowding problem is pointed out in symmetric SNE when the output dimensionality is smaller than the effective dimensionality of the data on the input, the neighborhoods are mismatched. (ref: page 180 of course materials by Prof.Chiu)
So t-SNE uses Student-t distribution in low dimension to preserve local similarity structure of data.

C. Try different settings of perplexity and visualize

	symmetric SNE	t-SNE
Perplexity = 10		
Perplexity = 20		
Perplexity = 30		
Perplexity = 50		
Perplexity = 100		

Discussion:
No matter symmetric SNE or t-SNE perform different shape with different perplexity values.
In symmetric SNE, the mnist2500 seems more scattered and in block shape in low perplexity, and more crowded and in narrow stripe in high perplexity.
In t-SNE, the mnist2500 seems more scatter and bend in low perplexity, and in piece shape in high perplexity.
There is a good discussion and visualization on https://distill.pub/2016/misread-tsne/. It shows the incredible flexibility of t-SNE. The same data set perform totally different shapes with vary perplexity values. The user can adjust perplexity to tidy up the visualization.

D. Conclusion

Through implement, I learned the difference between t-SNE and symmetric SNE. Not only the theory but how it change via visualization. I felt the power of preserving local similarity by t-SNE and saw the flexibility with different perplexity values. I could find meaningful topology and visualize complex models such as ANN.
Thanks teachers gave us chance to learn and implement multiple models and theories in this course this semester. I definitely will recommend to others.