3. Use T-SNE and symmetric SNE to project all your data mnist_X.csv onto 2D space and mark the data points into different colors respectively. The color of the data points depends on which cluster it belongs to (mnist_label.csv). The T-SNE code is provided. You need to modify it to symmetric SNE and discuss their differences. You also have to visualize the distribution of pairwise similarities in both high-dimensional space and low-dimensional space, based on both T-SNE and symmetric SNE.

Explanation of code.

The teacher provided us the code 'tsne.py'. We just need to modify this code to symmetric SNE. According to the definition of symmetric SNE:

$$q_{ij} = \frac{\exp(-\| y_i - y_j \|^2)}{\sum_{k \neq l} \exp(-\| y_l - y_k \|^2)} \quad \text{and} \quad \frac{\partial C}{\partial y_i} = 2 \sum_j (p_{ij} - q_{ij})(y_i - y_j)$$

I added flag *Symmetric* and made it possible to switch from t-sne to s-sne. You must set this value to *True* if you want to work with s-sne. I also implemented PCA by using code from previous task. In 'tsne.pe' I modified function 'tsne' and changed the way to compute Q (the pairwise affinity) and Gradient.

1. Computing Q for symmetric SNE:

```
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,
num[range(n), range(n)] = 0.
Q = num / np.sum(num)
Q = np.maximum(Q, 1e-12)
```

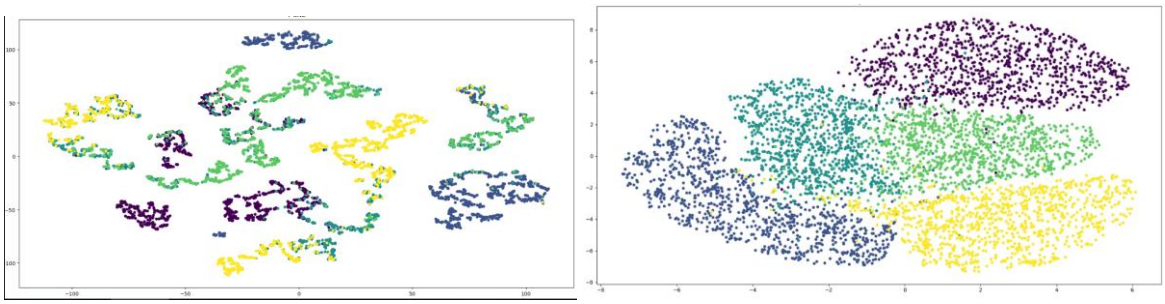
2. Computing Gradient for symmetric SNE:

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

Results.

```
pylab.scatter(Y[:, 0], Y[:, 1], 20, mnist_label)
pylab.show()
```

T-SNE **S-SNE**



We can see very clearly the embedding of S-SNE has the crowding problems, which resulting in the occlusion problem, and the scale of output are also different, t-SNE ranges from -150 to 150; symmetric SNE ranges from -8 to 8. 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.

T-SNE **S-SNE**

pylab.title('high-dimensional space')

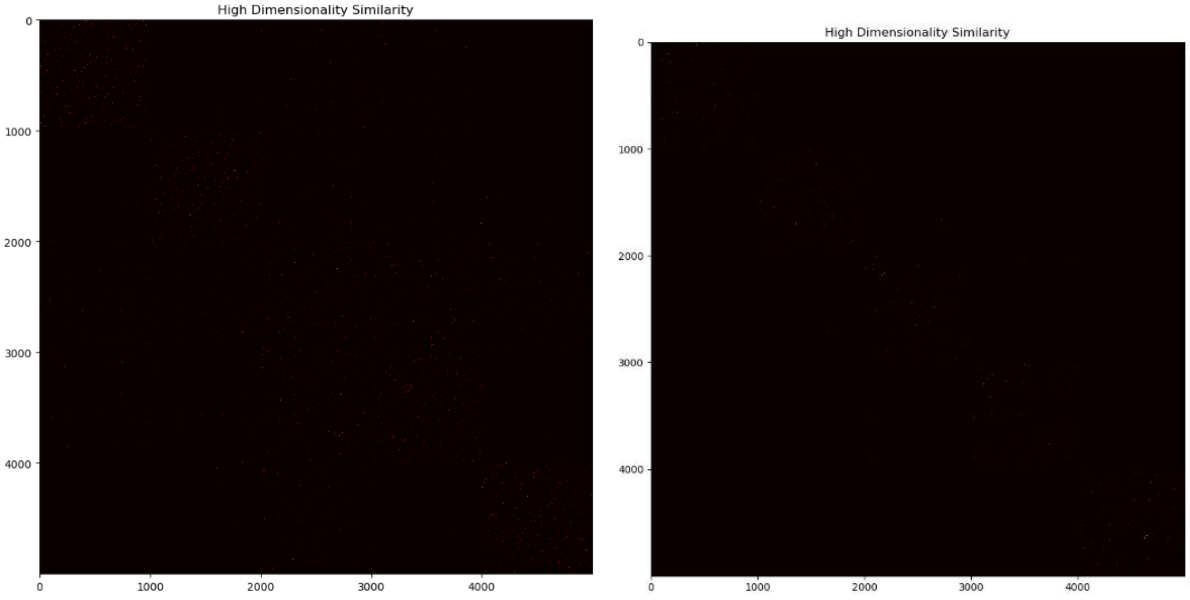
pylab.imshow(P, cmap='hot', interpolation='nearest')

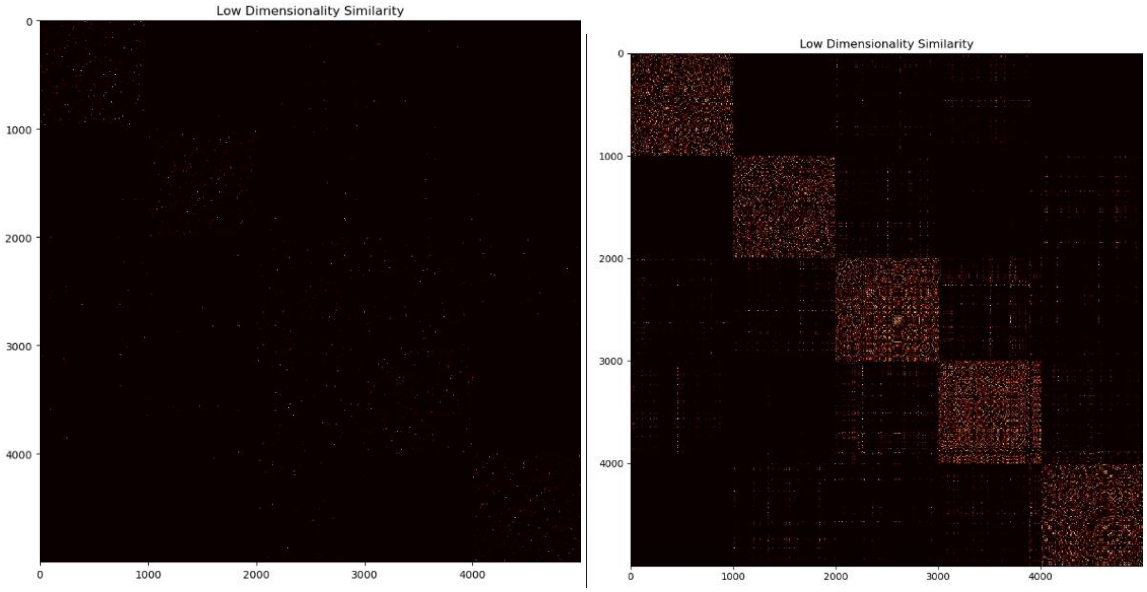
pylab.show()

pylab.title('low-dimensional space')

pylab.imshow(Q, cmap='hot', interpolation='nearest')

pylab.show()





For the low dimension representation of S-SNE, besides the diagonal part we can see a lot of relations (compare to T-SNE) between different digits, which will result in the occlusion in the 2D space.