HW3-Q1: PCA & T-SNE

HW3-Q1 is required to finish the following the tasks with this incomplete Notebook file. Before you start, you need to install the MulticoreTSNE and download the test set of Fashion-MNIST dataset. Some instructions are given as follows.

- Install MulticoreTSNE by `pip install MulticoreTSNE` , and you may need `pip install cmake` first; Otherwise, you could build it from scratch.
- Download the testset files, `t10k-images-idx3-ubyte.gz and t10k-labels-idx1-ubyte.gz` , and save them to your own data path.

Please follow the instructions from HW2 to get the PDF submission with this notebook file.

In [ ]:

```
# !pip install cmake
# !pip install MulticoreTSNE
```

HW3-Q1a: Load data and get the PCA embeddings (20 pts)

This part only requires you to set up the environment (installing dependency and download data) and run the following codes.

In [12]:

```
%matplotlib inline

import time
import numpy as np
from sklearn import linear_model
from sklearn.datasets import load_digits
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler

from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
from MulticoreTSNE import MulticoreTSNE

def load_mnist(path, kind='train'):
    import os
    import gzip
    """Load MNIST data from `path`"""
    labels_path = os.path.join(path,
                               '%s-labels-idx1-ubyte.gz'
                               % kind)
    images_path = os.path.join(path,
                               '%s-images-idx3-ubyte.gz'
                               % kind)
    with gzip.open(labels_path, 'rb') as lbpath:
        labels = np.frombuffer(lbpath.read(), dtype=np.uint8,
                               offset=8)
    with gzip.open(images_path, 'rb') as imgpath:
        images = np.frombuffer(imgpath.read(), dtype=np.uint8,
                               offset=16).reshape(len(labels), 784)
    return images, labels
```

```
# Download the fashion-mnist dataset into your own File Path
datapath = '/Users/lin/Desktop' #'./dataset' # Change this path to your own file
# We only use the testing set of Fashion-Mnist
data, gnd = load_mnist(datapath, kind='t10k')
K = len(set(gnd))
# We simply made labels from 0-K
semantic_labels = list(range(K))
# normalzied features to range [0,1]
data = data / 255.
# Do some preprocessing for the data
scaler = StandardScaler().fit(data)
X = scaler.transform(data)
N, D = X.shape
print('Dataset is with size={} and {}-dimension features of {} classes'.format(N
print(semantic_labels)
# We set M=2 as the target reduced dimension size, for a straightforward visuali
M = 2
# Instantiate PCA model fist
pca = PCA(n_components=M)
# Train the pca model by taking the fist M eigenvectors
t0 = time.time()
X_pca = pca.fit_transform(X)
print('PCA training and inference time = {:.4f}'.format(time.time()-t0))
print(X_pca.shape)
```

```
Dataset is with size=10000 and 784-dimension features of 10 classes
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
PCA training and inference time = 0.2254
(10000, 2)
```

HW3-Q1b: T-SNE Embeddings (20 pts)

Obtain the T-SEN embeddings with the given Sklearn model and MulticoreTSNE model, respectively. Compare their total running time (just print their running time accordingly).

In [13]:

```
# Instantiate TSEN model with Sk-learn implmentation
tsne1 = TSNE(n_components=M)
# TODO
t0 = time.time()
X_tsne1 = tsne1.fit_transform(X)
print('Sklearn T-SNE training and inference time = {:.4f}'.format(time.time()-t0

# Instantiate TSEN model with MulticoreTSNE implementation
tsne2 = MulticoreTSNE(n_components=M, n_jobs=8)
# TODO
t0 = time.time()
X_tsne2 = tsne2.fit_transform(X)
print('Multicore T-SNE training and inference time = {:.4f}'.format(time.time()-
```

```
Sklearn T-SNE training and inference time = 180.3398
Multicore T-SNE training and inference time = 78.3464
```

HW3-Q1c: T-SNE Embeddings visualization (20 pts)

Plot the T-SNE Embeddings and compare it with the PCA Embeddings.

In [16]:

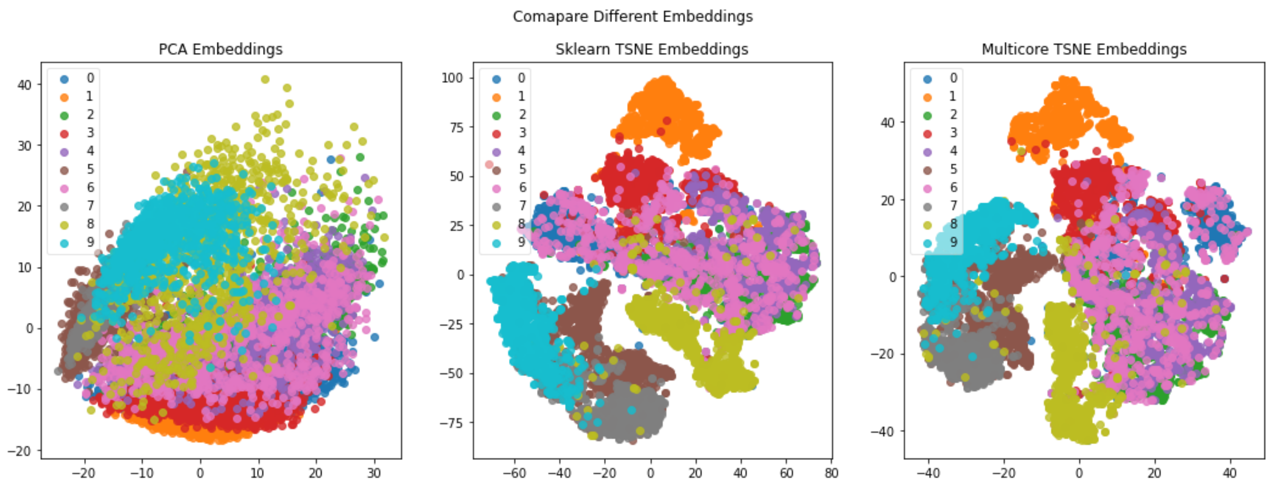
```
import seaborn as sns
# sns.set_theme()
```

```
fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(18,6))
fig.suptitle('Comapare Different Embeddings')
ax1.set_title('PCA Embeddings')
ax2.set_title('Sklearn TSNE Embeddings')
ax3.set_title('Multicore TSNE Embeddings')

for i in range(K):
    ax1.scatter(X_pca[gnd == i, 0], X_pca[gnd == i, 1], alpha=.8, label=semantic
    ax2.scatter(X_tsne1[gnd == i, 0], X_tsne1[gnd == i, 1], alpha=.8, label=sema
    ax3.scatter(X_tsne2[gnd == i, 0], X_tsne2[gnd == i, 1], alpha=.8, label=sema
ax1.legend(fancybox=True, framealpha=0.5)

# TODO: add legends for ax2 and ax3
ax2.legend(fancybox=True, framealpha=0.5)
ax3.legend(fancybox=True, framealpha=0.5)

plt.show()
```



Answer the following questions:

- Why T-SEN could lead to a better visualization than PCA?

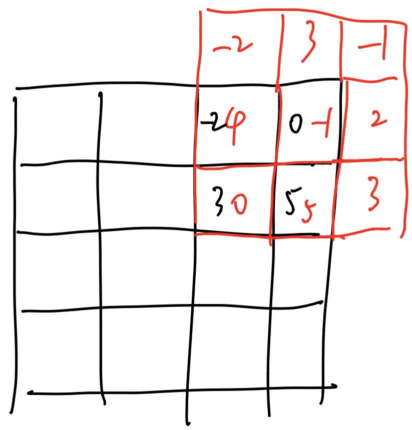
Hint: Recall the purpose of using Student's t-distribution.

Please directly answer the above question with Markdown inside this notebook file.

T-SNE is a non-linear data visualizer.it doesn't form a linear line to separate the classes or to calculate the variance and it doesn't use any norm or distance metric to calculate the distance between points.

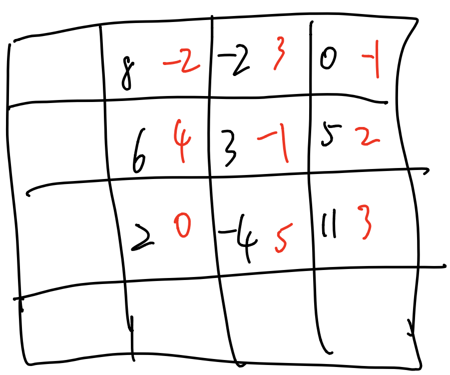
Q_2:

A:



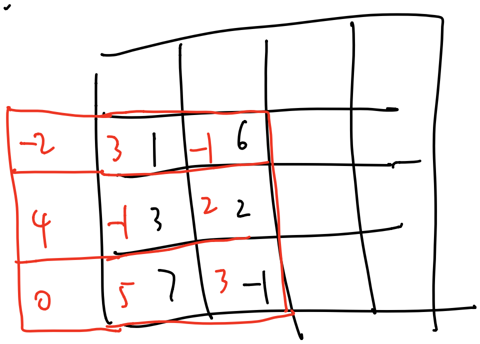
$$A = 4\cdot(-2) + 0\cdot(-1) + 0\cdot 3 + 5\cdot 5$$
$$+ 2\cdot 0 + (-1)\cdot 0 + 3\cdot 0 + (-2)\cdot(-2) + 3\cdot 5$$
$$= 36$$

B:



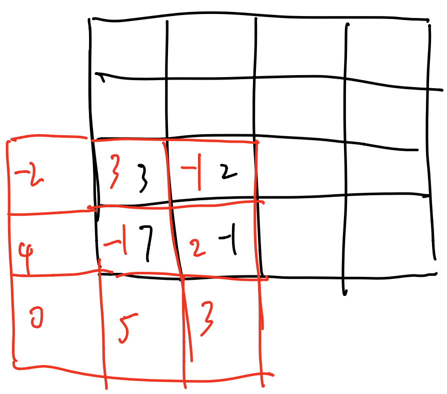
$$B = (-2)\cdot 8 + (-2)\cdot 3 + 0\cdot(-1)$$
$$+ 6\cdot 4 + 3\cdot(-1) + 5\cdot 2$$
$$+ 2\cdot 0 + (-4)\cdot 5 + 11\cdot 3 \quad = 22$$

C:



$$C = 3\cdot 1 + (-1)\cdot 6 + (-1)\cdot 3 + 2\cdot 2 + 5\cdot 7 + 3\cdot(-1)$$
$$+ (-2)\cdot 1 + 4\cdot 3 + 0\cdot 7$$
$$= 40$$

D:



$$D = 3\cdot 3 + (-1)\cdot 2 + (-1)\cdot 7 + 2\cdot(-1)$$
$$+ (-2)\cdot 3 + 4\cdot 7 + 0\cdot 7 + 5\cdot 7 + 3\cdot(-1)$$
$$= 52$$

Bonus Question:

(a) proof Bp1: $\delta_j^L = \frac{\partial C}{\partial z_j^L}$

$$\delta_j^L = \sum_k \frac{\partial c}{\partial a^2_k} \frac{\partial a^2_k}{\partial z_j^L}$$

$$\delta_j^L = \frac{\partial c}{\partial a_j^L} \frac{\partial a_j^L}{\partial z_j^L}$$

$$\delta_j^L = \frac{\partial c}{\partial a_j^L} \sigma'(z_j^L)$$

proof Bp2:

$$\delta_j^L = \frac{\partial c}{\partial z_j^L} = \sum_k \frac{\partial c}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial z_j^L} = \sum_k \frac{\partial z_k^{L+1}}{\partial z_j^L} \delta_k^{L+1}$$

$$z_k^{L+1} = \sum_j W_{kj}^{L+1} a_j^L + b_k^{L+1} = \sum_j W_{kj}^{L+1} \sigma(z_j^L) + b_k^{L+1}$$

$$\frac{\partial z_k^{L+1}}{\partial z_j^L} = W_{kj}^{L+1} \sigma'(z_j^L)$$

$$\delta_j^L = \sum_k W_{kj}^{L+1} \delta_k^{L+1} \sigma'(z_j^L)$$

proof Bp3: $\frac{\partial c}{\partial b_j^L} = \sum_k \left(\frac{\partial c}{\partial z_k^L} \frac{\partial z_k^L}{\partial b_j^L}\right) = \frac{\partial c}{\partial z_j^L} \frac{\partial z_j^L}{\partial b_j^L}$

$$= \delta_j^L \frac{\partial (\Sigma_k (w_{jk} a_k^{L-1} + b_j^L)}{\partial b_j}$$

$$= \delta_j^L$$

proof Bp4 : $\frac{\partial c}{\partial w_{jk}^L} = \sum_m \frac{\partial c}{\partial z_m^L} \frac{\partial z_m^L}{\partial w_{jk}^L} = \frac{\partial c}{\partial z_j^L} \frac{\partial z_j^L}{\partial w_{jk}}$

$$= \delta_j^L \frac{\partial}{\partial w_{jk}} \left(\sum_k w_{jk}^L a_k^{L-1} + b_j^L \right)$$

$$= a_k^{L-1} \delta_j^L$$

(b) $\frac{\partial C}{\partial b^L} = \begin{bmatrix} \frac{\partial c}{\partial b_1^L} \\ \vdots \\ \vdots \\ \frac{\partial C}{\partial b_{d_1}^L} \end{bmatrix}$

From Bp3, we know that $\frac{\partial c}{\partial b_j^L} = \delta_j^L$

$$\frac{\partial C}{\partial b^L} = \begin{bmatrix} \delta_1^L \\ \vdots \\ \delta_d^L \end{bmatrix} = \delta^L$$

$$\frac{\partial C}{\partial w^L} = \begin{bmatrix} \frac{\partial c}{\partial w_{11}^L} & \cdots & \frac{\partial c}{\partial w_{1 d_{L-1}}^L} \\ \vdots & \ddots & \vdots \\ \frac{\partial c}{\partial w_{d_L 1}^L} & \cdots & \frac{\partial c}{\partial w_{d_L d_{L-1}}^L} \end{bmatrix}$$

from Bp 4 we know $\frac{\partial L}{\partial w_{jk}^L} = a_k^{L-1} \delta_j^L$

$$= \begin{bmatrix} a_1^{L-1} \delta_1^L & \cdots & a_{d_{L-1}}^{L-1} \delta_1^L \\ \vdots & \ddots & \vdots \\ a_1^{L-1} \delta_{d_1}^L & \cdots & a_{d_{L-1}}^{L-1} \delta_{d_L}^L \end{bmatrix} = \delta^L \cdot (a^{L-1})^T$$

(c) Two assumptions: $C = \frac{1}{2n} \sum_x \|y(x) - a^L(x)\|^2$

$$C = \frac{1}{2}\|y - a^L\|^2 = \frac{1}{2}\sum_j (y_j - a_j^L)^2$$

$$\delta^2 = \nabla_{\hat{y}_1} L \odot \sigma'(z^L) = \frac{\partial}{\partial \hat{y}_1} \frac{1}{2}\|y_1 - \hat{y}_1\|_2^2 \odot \sigma'(z^L)$$

$$\delta^2 = (\hat{y}_1 - y_1) \odot \sigma'(z^3)$$

$$\delta^2 = ((w^3)^T \cdot \delta^3) \odot \sigma'(z^2)$$

$$\delta^1 ((w^2)^T \delta^2) \odot \sigma'(z^1)$$

BP(3-4) gives

$$\frac{\partial L}{\partial b^1} = \delta^1$$

$$\frac{\partial L}{\partial w^1} = \delta^1 (a^0)^T = \delta^1 \cdot x_1^T$$