计算机科学与技术学院可视化技术实验报告

实验题目：mnist 数据分析不同参数对 t-sne 结果的影响		学号：201900150221
日期：10.11	班级： 19 智能	姓名： 张进华
Email：zjh15117117428@163.com		

实验目的：

1. 使用 mnist 数据分析不同参数对 t-sne 结果的影响

实验软件和硬件环境：

Visual studio Code python 3.9.7

Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz 2.59 GHz

实验原理：

SNE 即 stochastic neighbor embedding，其基本思想为在高维空间相似的数据点，映射到低维空间距离也是相似的，SNE 把这种距离关系转换为一种条件概率来表示相似性，计算两点的概率采用高斯分布，而衡量两点的分布使用 KL 散度，通过随机梯度下降最小化损失函数。但是 SNE 倾向于保留局部特征，将高维离得近的点在低维时尽可能地聚在一起，但是不考虑不同类的间隔问题，整个降维后的图像会显得很拥挤。而 t_SNE 使用对称 SNE，简化梯度公式，同时在低维空间使用 t 分布取代高斯分布。

实验步骤

步骤一 导入数据集

导入的数据为 MNIST 中抽取的 1617 条数据，每一条数据是 64 维(8*8)，为了降低 TSNE 执行的复杂度，在进行 TSNE 之前首先通过 PCA 对数据进行降维，减少参数量，简化计算。

```
digits = datasets.load_digits(n_class=9)   #取前10种数字图片，0-10
data = digits.data   #data.shape=[1617,64]，表示1617张图片，每个图片8*8但是将图片表示为一个行向量
label = digits.target   #表示取出的图片对应的数字
```

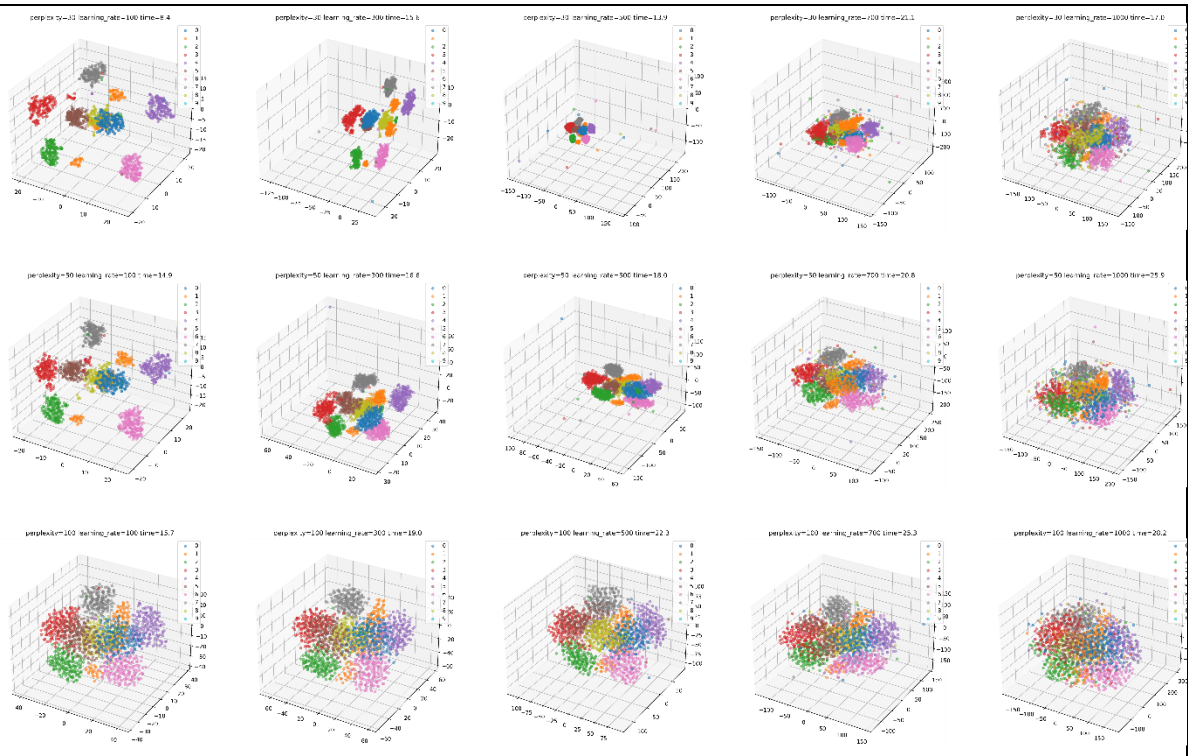
步骤二 比较 learning_rate 与 perplexity 对数据降维可视化的影响

参数设置如下：

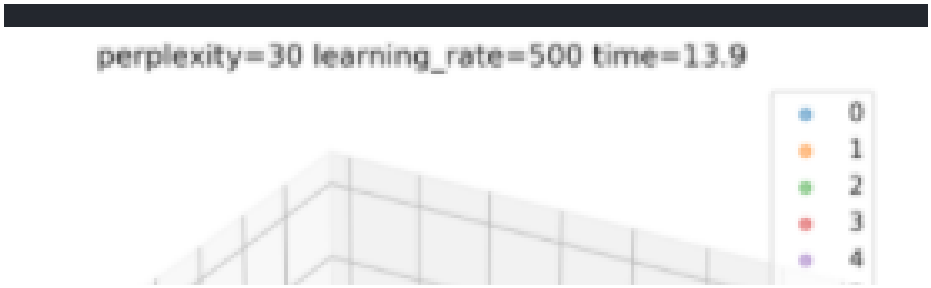
```
#绘制三维分类图
Perplexity = [30, 50, 100]
learningRate = [100,300,500,700,1000]
fig = plt.figure()
fig.set_size_inches(40,25)
fig.set_dpi(300)

for i in range(len(Perplexity)):
    for j in range(len(learningRate)):
        t0 = time()   #记录开始时间
        reduced_x = TSNE(n_components=3,perplexity=Perplexity[i],learning_rate=learningRate[j],init="pca").fit_transform(data)
        index = i * 5 + j
```

实验过程中将 learning_rate 与 perplexity 分别设置不同值，绘制不同参数下可视化的效果并记录这种参数下可视化所用的时间，如下图所示：



由于图片比较大，放在这里显得不太清楚，图片上的标签如下，是 learning_rate、perplexity、time



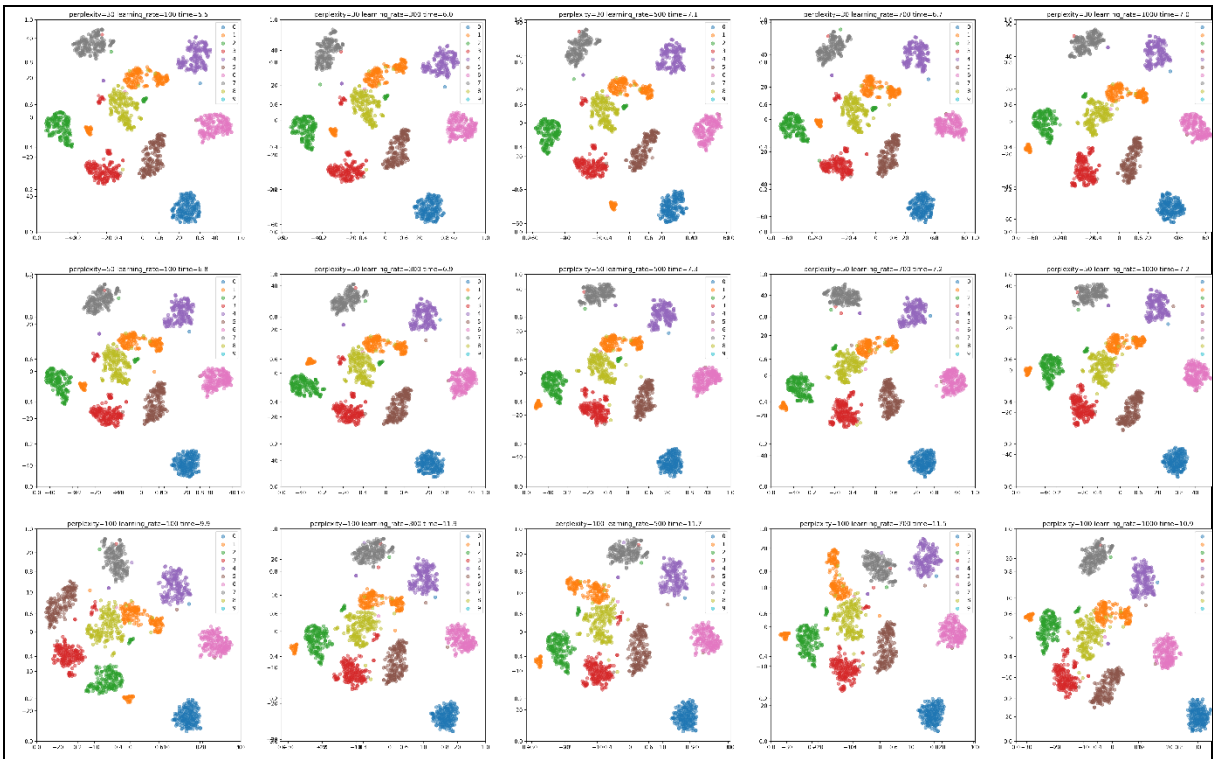
由上图可以看到：

1. 在 learning_rate 相同的情况下，不断增大 perplexity(纵向比较)，可以发现聚集的效果会变差，在第二行，perplexity = 50 的情况下分类的效果相对较好

2. 在 perplexity 相同的情况下，不断增大 learning_rate(横向比较)，可以发现当 learning_rate 取过大或者过小时效果都不太好，在第二列，learning_rate = 300 的情况下分类的效果相对较好。

3. 可以看到当 perplexity 增大时运行时间明显增加，同样，保持 perplexity 相同的情况下增大 learning_rate，运行时间增大。

可以看到将数据降维到 3 维后可视化的效果不太明显，这里尝试将数据降到 2 维并进行可视化，实现效果如下：



各个参数的意义还是和上面一样，但是更加直观，可以看到在 learning_rate = 300, perplexity = 50 的情况下分类效果最好

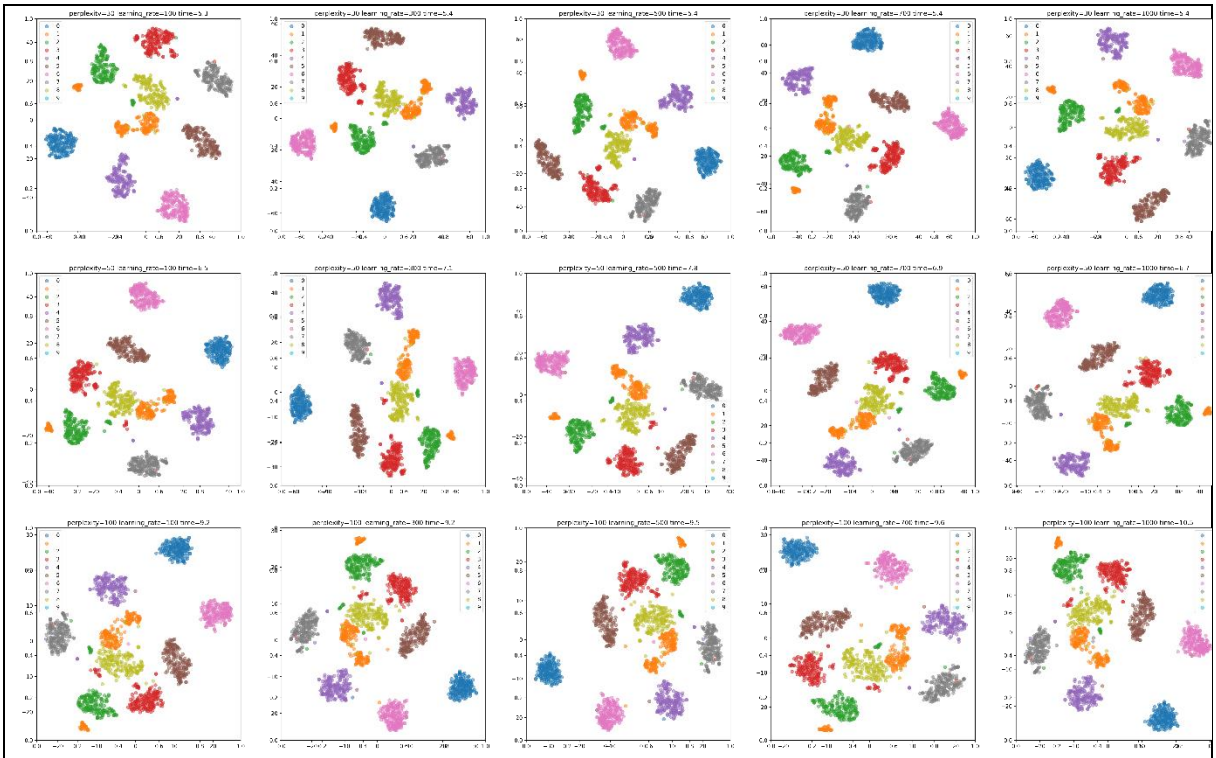
步骤三 比较使用 PCA 和 Random 初始化过程对数据可视化的影响

参数设置如下：

```
#绘制二维分类图
Perplexity = [30, 50, 100]
learningRate = [100,300,500,700,1000]
fig,ax=plt.subplots(3,5)
fig.set_size_inches(40,25)
fig.set_dpi(300)

for i in range(len(Perplexity)):
    for j in range(len(learningRate)):
        t0 = time()    #记录开始时间
        reduced_x = TSNE(n_components=2,perplexity=Perplexity[i],learning_rate=learningRate[j],init="random").fit_transform(data)
        index = i * 5 + j
```

前面一直采用先将数据用 PCA 降维后再用 TSNE, 下面采用直接对数据进行 TENE, 也就是修改参数 init = random, 其他参数跟上面的相同，比较数据集降到二维后二者的效果，采用 init = randow 的实现效果如下：



由图可以大概看出在 init = random 的情况下实现的效果比 PCA 要好一点。

结论分析与体会：

TSNE 中 perplexity 参数表示高维空间当前点的附近有效近邻点的个数，在设置时不宜过大，过大时会由于选取的点过多，导致计算量增大，同时在低维数据显得分散。

而 learning_rate 参数一方面随着其增大，运算时间增长，同时当其过大时，分类的效果会极具膨胀，变得很分散。

Init 参数选取 pca 和 random, 在我的实验中其实看不出二者有多大的区别，可能当数据的维度很大时采用 pca 能够有效的减少运算时间。