PCA算法报告

1.基本信息和运行方法

数据集：课件提供的1000个MNIST数据

编程环境：Windows 10，python 3.8.8

依赖库：

- matplotlib==3.3.4
- numpy==1.20.1
- Pillow==8.4.0
- scikit_learn==1.0.1（仅仅用于比较）

运行方法：

首先安装依赖

```
cd src
pip install -r requirements.txt
```

运行PCA算法

```
python PCA.py
```

运行交互界面

```
python interaction.py
```

在交互界面中，可以用鼠标左键点击散点，点击后可以查看散点的id，分类信息，主成分结果和原始图片

2.基本原理

对于m个n维数据$x_1, x_2, ...x_m$，我们希望能找到$k << n$个线性无关的基向量来表示每个数据x_i，并且让表达尽量接近原始形式。因此要最小化如下方程：$min_{V^TV=I}\sum_{i=1}^{m}||x_i - VV^Tx_i||_2^2$，其中$V \in R^k$

可以证明得到，V为数据矩阵协方差XX^T的最大k个特征值对应的单位特征向量。

因此，PCA的算法流程如下：

1.将每一维度数据减去其均值

2.计算数据协方差矩阵XX^T

3.对XX^T进行奇异值分解，使得$XXT = U\Sigma V^T$, 其中U和V为单位正交矩阵，Σ为对角阵

4.把Σ的特征值按照由大到小排序，取得最大的k个特征值和对应的特征向量，特征向量按行组成主成分矩阵P，$Y = PX$为主成分分析结果。

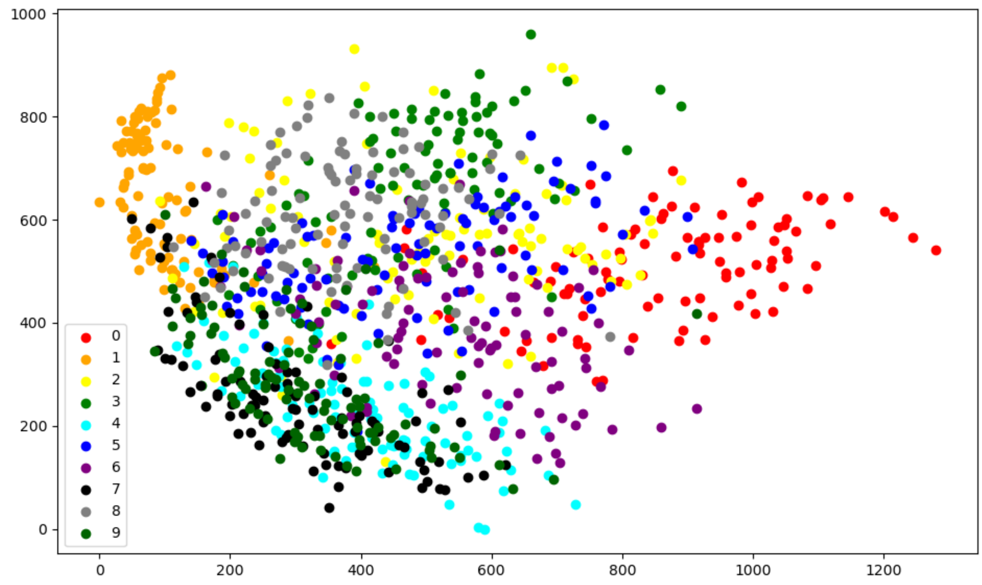
3.实现细节

我使用python的numpy库实现上述算法，因为要可视化，所以取k=2，具体代码如下：

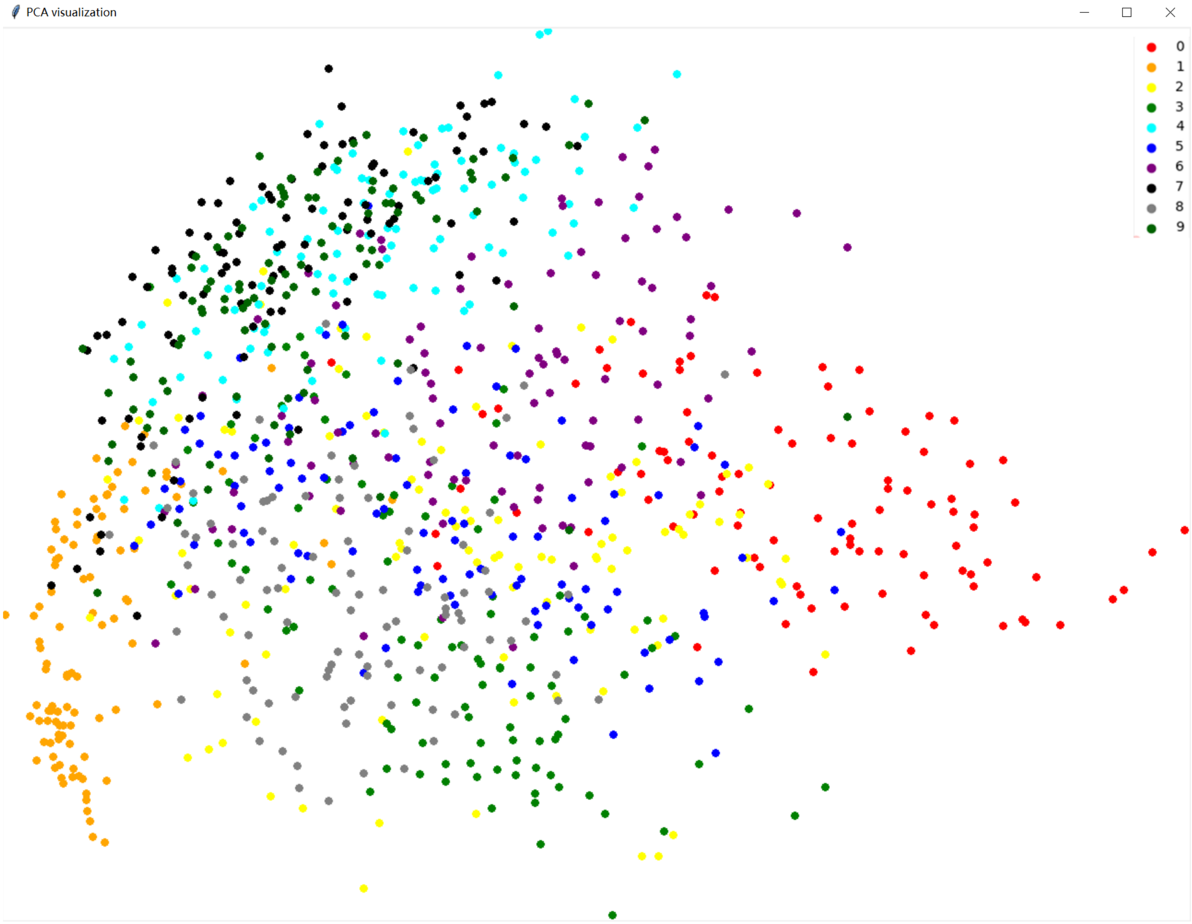
```
self.input = self.data.reshape(self.N, -1).T #784 * self.N
mean = np.mean(self.input, axis=1, keepdims=True).repeat(self.N, axis=1) #784 *
self.N
self.input = self.input - mean #784 * self.N #数据减去均值
self.covariance = np.matmul(self.input, self.input.T) / self.N #784 * 784 #求协方
差
U, sigma, VT = np.linalg.svd(self.covariance) #奇异值分解
P = U[:, 0:2].T #2 * 784 #求主成分矩阵P
self.result = np.matmul(P, self.input) #2 * self.N #求主成分分析结果
```

之后，我使用两套方案进行可视化：一套是使用matplotlib的散点图绘制方法，一套是使用python自带的tkinter图形界面进行可交互的可视化。两种方案我都将主成分分析结果每个维度归一化到[0,1]之间，然后映射到对应的x，y坐标。

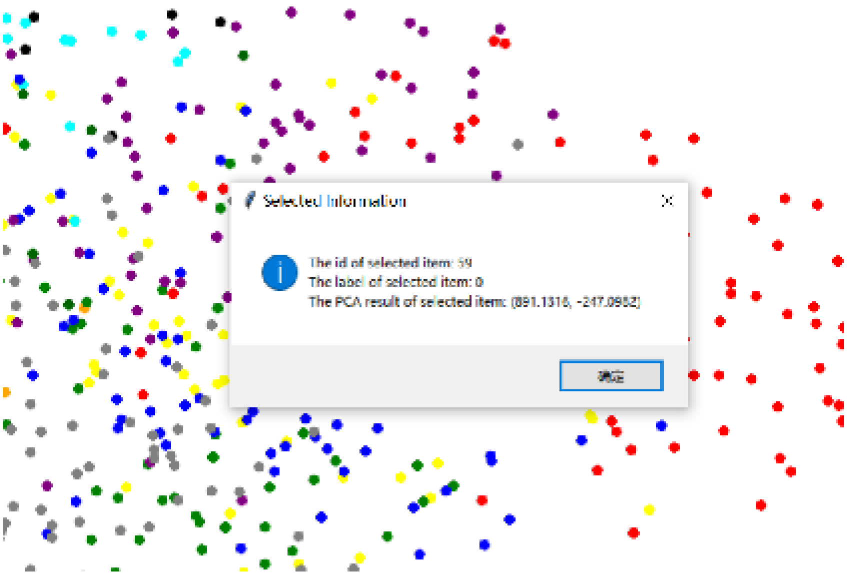
对于方案1，我使用plt.scatter函数绘制散点图，我预先定义了若干颜色对应不同的标签，以将不同的标签可视化，结果如下：

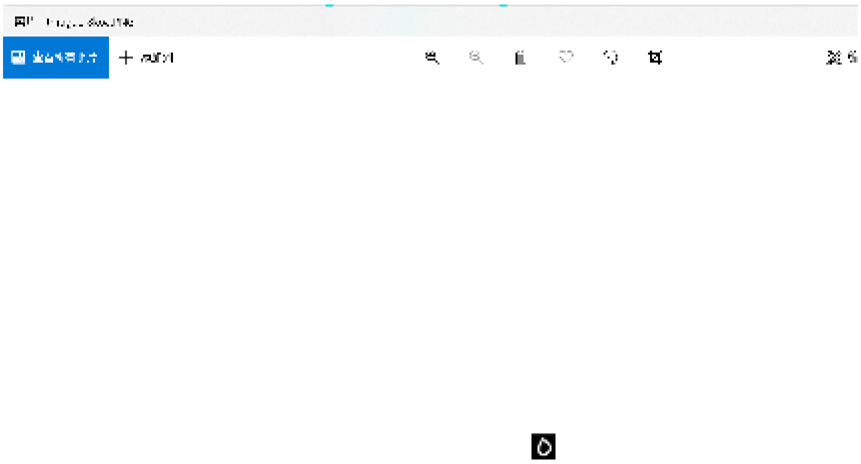


对于方案2，我用tkinter的canvas把每个数据点绘制成圆形，并且按照同样的方法设置了每个点的颜色，显示效果是matplotlib方法将y轴翻转过来，因为二者的y坐标定义相反。



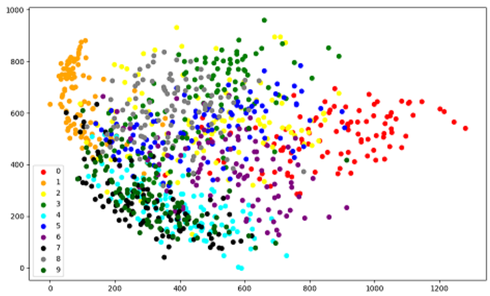
同时，我通过绑定回调函数的方法捕捉了鼠标左键点击的位置，进而判定被点击的点。点击鼠标后，能够显示该点的详细信息和原始图片，具体如下：



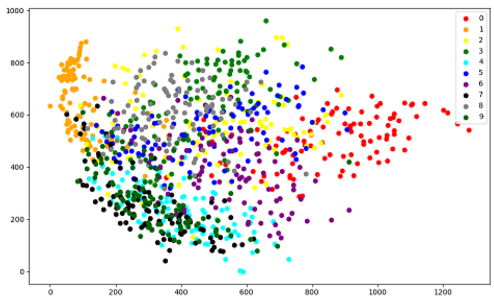


4.结果分析

首先，我们同样调用了sklearn的PCA模块，比较二者结果相同，证明我们的实现正确：



我的实现结果



sklearn的实现结果

其次，观察可视化结果，可以看出不同标签的图片并没有被分开的非常好。

我们查看了被选择的两个维数的方差值，分别为0.0979，0.0743，总共只能表示17.12%的主成分信息。查看前十大的特征值，分别为

	1	2	3	4	5	6	7	8	9	10
特征值	**325072**	**249608**	232372	182435	163797	153275	117191	101736	93975	72223

可以看出，第3到第10大的特征值并没有明显比前两大特征值要小许多，他们对应的主成分也占据很大的比重，而且整个数据有784维，维数很多，舍去这些信息会让散点映射结果不佳。

综上，虽然我们的PCA实现完全正确，但是使用PCA将这个高维数据可视化到2维空间中并不是一个好的选择。