哈尔滨工业大学计算机科学与技术学院

实验报告

课程名称：　机器学习

课程类型：　选修

实验题目：　PCA 模型实验

学号：1162100102
班级：1603104
姓名：　王晨懿

目录

一、实验目的

实现一个 PCA 模型，能够对给定数据进行降维（即找到其中的主成分），可以利用已有的矩阵特征向量提取方法。

二、实验要求及实验环境

实验要求：

测试：
（1）首先人工生成一些数据（如三维数据），让它们主要分布在低维空间中，如首先让某个维度的方差远小于其它维度，然后对这些数据旋转。生成这些数据后，用你的 PCA 方法进行主成分提取。
（2）利用手写体数字数据 mnist，用你实现 PCA 方法对该数据降维，找出一些主成分，然后用这些主成分对每一副图像进行重建，比较一些它们与原图像有多大差别（可以用信噪比衡量）。

实验环境：

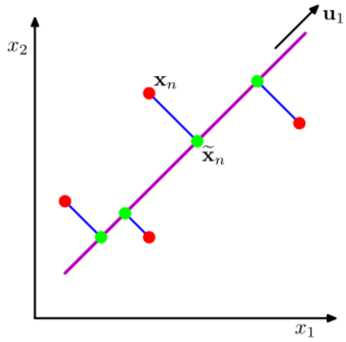
Window10 64 位操作系统
Pycharm

三、设计思想（本程序中的用到的主要算法及数据结构）

PCA 的原理

主成分分析寻找一个低维空间，被称为主子平面，用紫色的线表示，使得数据点（红点）在子空间上的正交投影能够最大化投影点（绿点）的方差。

PCA 的另一个定义基于的是投影误差的平方和的最小值，用蓝线表示。



考虑一组观测数据集 {xn }, n=1,2,⋯,N ，是 D 维空间中的变量，目标是投影到 M 维

最大方差形式

选择一个单位向量u_1 ，满足$\mathrm{u}_1^{\mathrm{T}}\mathrm{u}_1 = 1$

样本均值

$$\bar{\boldsymbol{x}} = \frac{1}{N}\sum_{n=1}^{N}\boldsymbol{x}_n$$

投影数据的方差为

$$\frac{1}{N}\sum_{n=1}^{N}\{\boldsymbol{u}_1^T\boldsymbol{x}_n - \boldsymbol{u}_1^T\bar{\boldsymbol{x}}\}^2 = \boldsymbol{u}_1^T\boldsymbol{S}\boldsymbol{u}_1$$

其中 S 是协方差矩阵

$$\boldsymbol{S} = \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{x}_n - \bar{\boldsymbol{x}})(\boldsymbol{x}_n - \bar{\boldsymbol{x}})^T$$

我们现在关于 u1 最大化投影方差$\mathrm{u}_1^{\mathrm{T}}\,\mathrm{S}\,\mathrm{u}_1$，最大化过程满足归一化条件$\mathrm{u}_1^{\mathrm{T}}\mathrm{u}_1 = 1$，引入拉格朗日乘数，记作$\lambda_1$ ，然后对下式进行最大化

$$\boldsymbol{u}_1^T\boldsymbol{S}\boldsymbol{u}_1 + \lambda_1(1 - \boldsymbol{u}_1^T\boldsymbol{u}_1)$$

令其关于 u1 导数等于 0，则有

$$Su_1 = \lambda_1 u_1$$

这表明 u1 一定是 S 的一个特征向量。

左乘u_1^T ，使用$u_1^T u_1 = 1$，我们看到方差为

$$u_1^T S u_1 = \lambda_1$$

因此当我们将 u 1 设置为与具有最大的特征值 λ 1 的特征向量相等时，方差会达到最大值。这个特征向量被称为第一主成分。

我们计划将我们的数据投影到前 M 个主成分中，那么我们只需寻找前 M 个特征值和特征向量。

最小误差形式

引入 D 维基向量的一个完整的单位正交集合 {u i } ，其中 i = 1,...,D ，且满足

$$u_i^T u_j = \delta_{ij}$$

由于基是完整的，因此每个数据点可以精确地表示为基向量的一个线性组合，即

$$x_n = \sum_{i=1}^{D} \alpha_{ni} u_i$$

系数 ani 对于不同的数据点不同。这对应于将坐标系旋转到了一个由{ui}定义的新坐标系，利用单位正交性质，我们有$a_{nj} = x_n^T u_j$ ，则有

$$x_n = \sum_{i=1}^{D} (x_n^T u_i) u_i$$

我们可以用下式来近似每个数据点 x n，通过前 M 个矢量表示

其中 z_{ni} 取决于特定数据点，b_i 是常量。

$$\tilde{x}_n = \sum_{i=1}^{M} z_{ni} u_i + \sum_{i=M+1}^{D} b_i u_i$$

目标是最小化

$$J = \frac{1}{N} \sum_{n=1}^{N} \|x_n - \tilde{x}_n\|^2$$

考虑关于 {z_{ni}} 的最小化。消去 x_n，令它关于 z_{nj} 的导数为零，然后使用单位正交的条件，我们有

$$z_{nj} = x_n^T u_j$$

类似地，令 J 关于 b i 的导数等于零，再次使用单位正交的关系，我们有

$$b_j = \bar{x}^T u_j$$

带入得

$$\boldsymbol{x}_n - \tilde{\boldsymbol{x}}_n = \sum_{i=M+1}^{D} \{(\boldsymbol{x}_n - \bar{\boldsymbol{x}})^T \boldsymbol{u}_i\} \boldsymbol{u}_i$$

于是，我们得到了失真度量 J 的表达式

$$J = \frac{1}{N} \sum_{n=1}^{N} \sum_{i=M+1}^{D} (\boldsymbol{x}_n^T \boldsymbol{u}_i - \bar{\boldsymbol{x}}^T \boldsymbol{u}_i)^2 = \sum_{i=M+1}^{D} \boldsymbol{u}_i^T \boldsymbol{S} \boldsymbol{u}_i$$

剩下的任务是关于 {u i} 对 J 进行最小化，这必须是具有限制条件的最小化，因为如果不这样，我们会得到 u i = 0 这一没有意义的结果。

例如在二维数据空间以及一维主子空间的情况。选择方向 u2 来最小化 J。限制条件 $u_2^T u_2 = 1$ 。使用拉格朗日乘数λ_2 来强制满足这个限制。

$$\tilde{J} = \boldsymbol{u}_2^T \boldsymbol{S} \boldsymbol{u}_2 + \lambda_2 (1 - \boldsymbol{u}_2^T \boldsymbol{u}_2)$$

令其关于 u2 的导数等于 0，我们有$Su_2 = \lambda_2 u_2$ 。从而 u2 是 S 的特征向量，特征值为 λ_2 。解出u_2 带回原式得到 $J = \lambda_2$ 。因此将u_2 选择为特征值较小的特征向量，可得到 J 的最小值。

因此我们应该讲主子空间与具有较大的特征值的特征向量对齐。对于任意的 D 和任意的 M<D，最小化 J 的一般解都可以通过将 ui 选择为协方差矩阵的特征向量的方式得到，即

$$\boldsymbol{S} \boldsymbol{u}_i = \lambda_i \boldsymbol{u}_i$$

失真度量的对应的值为

$$J = \sum_{i=M+1}^{D} \lambda_i$$

于是我们可以通过将这些特征向量选择成 D-M 个最小的特征值对应的特征向量，来得到 J 的最小值。因此定义了主子空间的特征向量是对应于 M 个最大特征值的特征向量。

PCA 的实现

1. 首先对样本进行中心化

```python
n, m = data.shape
# 中心化
mu = np.mean(data, axis=0)
X = np.array([x - mu for x in data])
```

2. 计算样本的协方差矩阵

这里需要注意的是，分母为 n-1；

或者调用 numpy.cov()函数，二者皆可。

```python
# 协方差矩阵
sigma = np.dot(X.T, X) / (n - 1)  # 无偏估计，除以n-1
# sigma = np.cov(X, rowvar=0)  # 求协方差矩阵,rowvar为0，一行代表一个样本
```

3. 获取最大的 k 个特征值对应的特征向量

```
# 获取前k个特征向量
eig_val, eig_vec = np.linalg.eig(sigma)  # 特征根和特征向量
eig_pairs = [(abs(eig_val[i]), eig_vec[:, i]) for i in range(m)]
eig_pairs.sort(reverse=True, key=lambda item: item[0])  # 根据特征值降序排列
reduce = np.array([eig_pairs[i][1] for i in range(k)]).T
print(reduce)
```

4. 最后将数据转移到新坐标系，并用主成分对样本进行近似

```
# 将数据转移到新坐标系
reduce_d_data = np.dot(X, reduce)
# 用主成分对样本进行近似
new_data = np.dot(reduce_d_data, reduce.T) + mu
```

四、实验结果与分析

手工生成的数据集

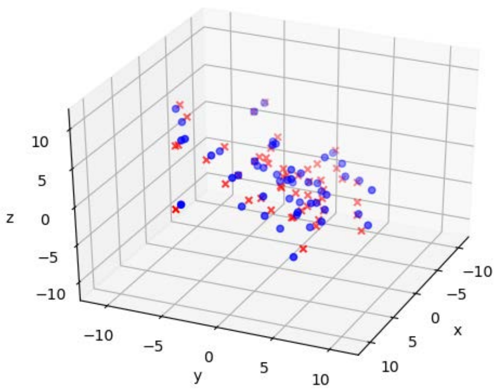
生成数据的函数如下：

```
def generate_data(n, angle):
    """
    生成数据
    :param n: 样本数
    :param angle: 绕x轴旋转的角度
    :return: 数据集
    """
    x, y, z = np.random.multivariate_normal([0, 0, 0], [[20, 0, 0], [0, 20, 0], [0, 0, 1]], n).T
    X = np.c_[np.array(x).reshape(n, ), np.array(y).reshape(n, ), np.array(z).reshape(n, )]
    X = rotate_mtx(X, angle)
    x, y, z = X[:, 0], X[:, 1], X[:, 2]
    ax.scatter(x, y, z, c='b')
    return X
```

设置的均值为[0, 0, 0] ，协方差矩阵为 $\begin{matrix} 20 & 0 & 0 \\ 0 & 20 & 0 \\ 0 & 0 & 1 \end{matrix}$

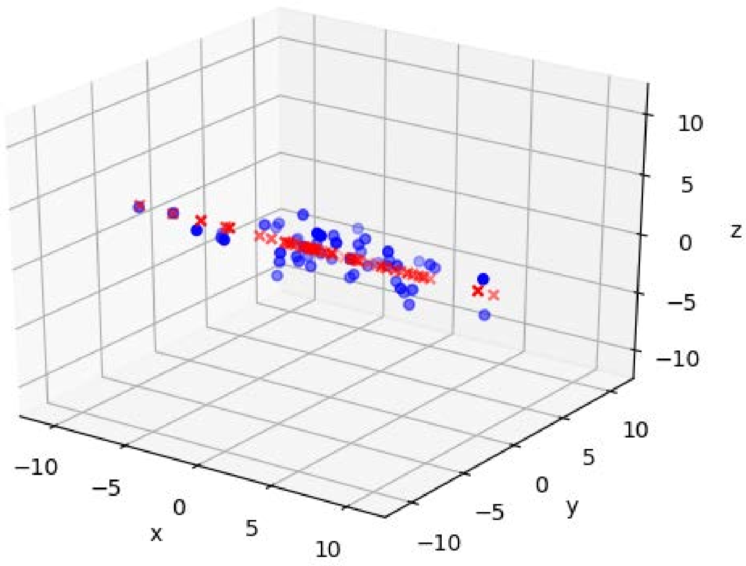
然后将其绕 x 轴旋转。

这里我选择生成 50 个样本，绕 x 轴旋转 30 度，生成数据如下图所示：



其中蓝色球形为样本点，红色×形为进行主成分分析后，对样本进行近所产生的点。

旋转视角，我们可以看到，经过 PCA 算法降维后的样本，明显地分布在了一个二维平面上，并且达到了预期：这个投影平面能够最大化投影点的方差。



mnist 数据集

首先读入 mnist 数据集

```
def read_mnist_dataset(path='MNIST_data'):
    labels_path = os.path.join(path, 'train-labels.idx1-ubyte')
    images_path = os.path.join(path, 'train-images.idx3-ubyte')
    with open(labels_path, 'rb') as f:
        struct.unpack('>II', f.read(8))
        labels = np.fromfile(f, dtype=np.uint8)
    with open(images_path, 'rb') as f:
        struct.unpack('>IIII', f.read(16))
        imgs = np.fromfile(f, dtype=np.uint8).reshape(len(labels), 784)
    return imgs
```

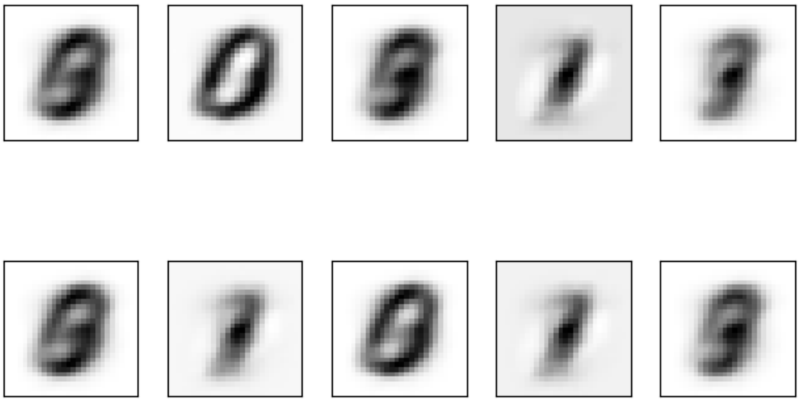
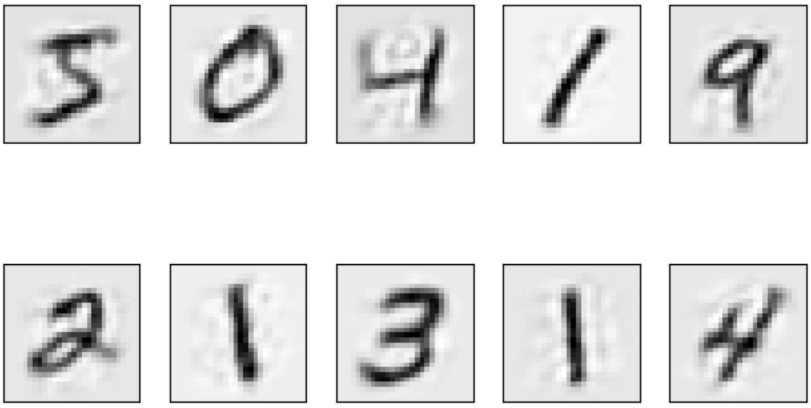
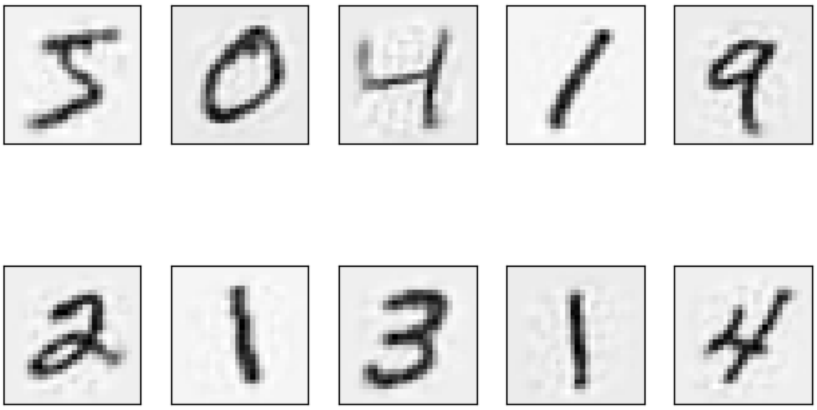
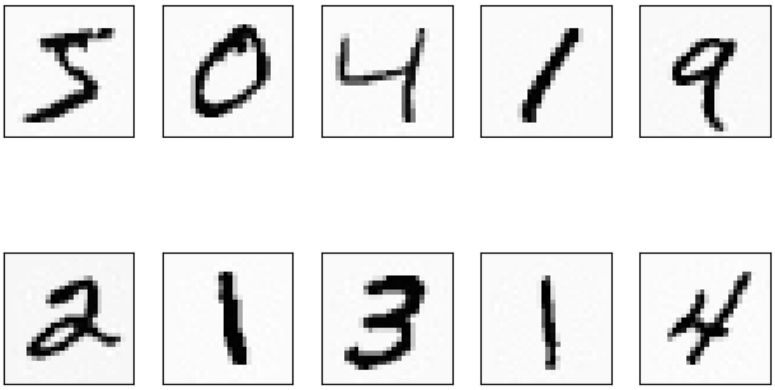
然后取前 1000 张图片进行降维

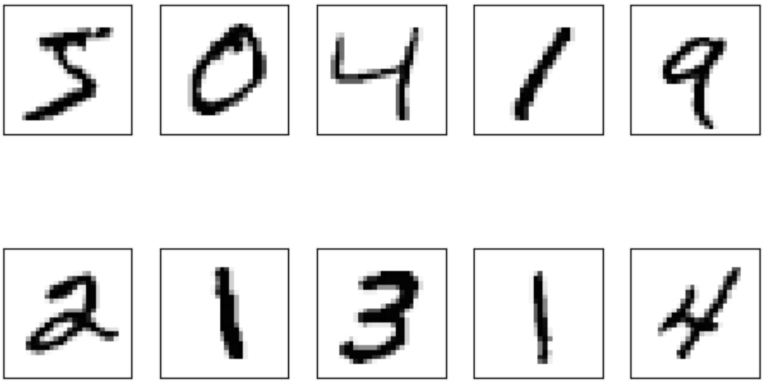
```
imgs = read_mnist_dataset()
origin_imgs = imgs[:1000]  # 取前1000张图片进行测试

reduce, new_data = pca(origin_imgs, dimension)
```

展示降维前后的图片，每次展示前 10 张，并输出信噪比。

降维前，前 10 张图片分别为：

选择多个维度降维，多次测试得到下表：

维度	降维后	信噪比
1		-0.7135
50		9.8377
200		13.5197
400		30.1101

700		256.9693

可以看出当将到 1 维时难以辨认出数字；当维度到达 50 维就大致可以分辨出数字了；当维度到达 400，几乎已经看不出差距了。

PCA 的这种方法经常应用在维度降低、有损数据压缩、特征抽取上，性能十分强大。

五、结论

PCA 是一种常用的降维方法。通过 PCA 能将样本投影至低维空间中，并最大化保留原始数据，从而可以延伸出一系列的应用。

显然低维空间和高维空间必有不同，因为较小的特征值对应的特征向量舍弃了，这是降维导致的结果。

舍弃的部分信息能使样本的采样密度增大，这正是降维的重要动机；另一方面，当数据受到噪声影响时，较小特征值所对应的特征向量往往与噪声有关，将他们舍弃还能在一定程度上起到去噪的效果。

六、参考文献

https://docs.scipy.org/doc/numpy/

https://matplotlib.org/api/_as_gen/mpl_toolkits.mplot3d.axes3d.Axes3D.html

https://en.wikipedia.org/wiki/Principal_component_analysis

七、附录：源代码（带注释）

见附件