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： 主成分分析

学号：116030507
姓名： 聂晨晞

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：

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唯独，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。

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：

MacOS 10.14 + python 3.6

三、设计思想（本程序中的用到的主要算法及数据结构）

1．算法原理

在我们的实例中，使用的输入数据集表示为$\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$，维度n=2即$x^{(i)} \in \mathbf{R}^2$。假设我们想把数据从2维降到1维，PCA首先需要进行预处理，将所有的数据的每一维度的平均值都归一化到0附近，可以使用对每一个维度的每一个节点都减去一个平均值来做到，然后我们对该数据集求特征值和特征向量，我们可以通过numpy等软件进行求解，得到的特征向量表达如下所示：

$$U = \begin{bmatrix} | & | & & | \\ u_1 & u_2 & \cdots & u_n \\ | & | & & | \end{bmatrix}$$

此处，u1是主特征向量（对应最大的特征值），u2是次特征向量。以此类推，另记$\lambda_1, \lambda_2, \ldots, \lambda_n$为对应的特征值。

旋转数据
至此，我们把x用(u_1, u_2)基表达为：

$$x_{\mathrm{rot}} = U^T x = \begin{bmatrix} u_1^T x \\ u_2^T x \end{bmatrix}$$

数据降维

$$\tilde{x}^{(i)} = x_{\mathrm{rot},1}^{(i)} = u_1^T x^{(i)} \in \Re.$$

我们选择前i个使用如上的公式进行数据降维

数据还原

$$\tilde{x} = \begin{bmatrix} x_{\mathrm{rot},1} \\ \vdots \\ x_{\mathrm{rot},k} \\ 0 \\ \vdots \\ 0 \end{bmatrix} \approx \begin{bmatrix} x_{\mathrm{rot},1} \\ \vdots \\ x_{\mathrm{rot},k} \\ x_{\mathrm{rot},k+1} \\ \vdots \\ x_{\mathrm{rot},n} \end{bmatrix} = x_{\mathrm{rot}}$$

数据还原，就是将后面的多的纬度都用0进行填充。

选择主成分个数
我们对于每一个特征值，使用图下的公式来确定需要有几个纬度

$$\frac{\sum_{j=1}^{k} \lambda_j}{\sum_{j=1}^{n} \lambda_j} \geq 0.99.$$

2. 算法的实现

产生数据：

```
def data_generator():
    sampleNo = 10
    mux = 3
    sigmax = 1.5
    muy = 5
    sigmay = 1.5
    muz = 0
    sigmaz = 0.2
    x = np.random.normal(mux, sigmax, sampleNo)
    y = np.random.normal(muy, sigmay, sampleNo)
    z = np.random.normal(muz, sigmaz, sampleNo)
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    ax.scatter(x, y, z, c="red")
    ax.set_xlabel("x")
    ax.set_ylabel("y")
    ax.set_zlabel("z")
    plt.show()
    x = x.reshape(-1, 1)
    y = y.reshape(-1, 1)
    z = z.reshape(-1, 1)
    data = np.hstack((x, y))
    data = np.hstack((data, z))
    return data, x, y, z
```

使用如上图所示的代码，产生一个三维数据组，我们限定

第三个维度在0附近并且对第三个维度实现降维。

```
def zero_mean(dataMat):
    '''
    按列求解该数据矩阵的平均值，然后所有数据减去对应的平均值
    以达到将平均值达到0的要求
    :param dataMat: 数据矩阵，每一行表示一个数据，每一列代表一个纬度（特征）
    :return: 平均值归零后的数据
    '''
    data_mean = np.mean(dataMat, axis=0) # 按照列求平均值，即求各个纬度的平均值
    data_zero_mean = dataMat - data_mean
    return data_zero_mean, data_mean
```

使用上图所示的代码对数据进行归一化，使得他们的平均值都在零附近

```
def coverience_matrix(data):
    m = data.shape[1]
    cov = np.zeros((m, m))
    for i in range(len(data)):
        #print(data[i].transpose().dot(data[i]))
        cov += data[i].reshape(1,-1).transpose().dot(data[i].reshape(1, -1))
        #cov += data[i].transpose().dot(data[i])
    cov = cov / m
    return cov
```

按照上一节叙述的公式计算协方差矩阵

```
def decide_k(eigVals, m, percent = 0.99):
    '''
    按照http://deeplearning.stanford.edu/wiki/index.php/主成分分析 （翻墙访问）
    中指定的公式来决定降维之后的纬度
    :param eigVals:特征值
    :param eigVects:特征向量
    :param m:降维之前的纬度
    :return:降维之后的数据纬度
    '''
    sum1 = sum(eigVals)
    eigVals_sort = np.sort(eigVals)
    eigVals_sort = eigVals_sort[::-1]
    sum2 = 0
    for i in range(m):
        sum2 += eigVals_sort[i]
        if sum2 / sum1 > percent:
            break
    return (i + 1)
```

按照上一节所说的公式决定需要有多少个维度

```python
def rotate_data(data, eigVals, eigVects):
    '''
    根据特征向量和特征值选取k个纬度，然后进行数据旋转
    :param data:数据
    :param eigVals: 特征值
    :param eigVects: 特征向量
    :return: 旋转之后的数据
    '''
    def decide_k(eigVals, m, percent = 0.99):...
    k = decide_k(eigVals, data.shape[1])
    print("k = " + str(k))
    eigValIndice = np.argsort(eigVals) #对特征值从小到大排列
    n_eigValIndice = eigValIndice[-1:-(k + 1):-1] # 最大k个特征值的下标
    U = []
    for i in range(k - 1):
        if i == 0:
            U = np.vstack((eigVects[n_eigValIndice[i]], eigVects[n_eigValIndice[i + 1]]))
        else:
            U = np.vstack((U, eigVects[n_eigValIndice[i + 1]]))
    data_rot = np.array(U).dot(data.transpose())
    return data_rot.transpose(), U
```

按照上一节所说的过程实现数据旋转

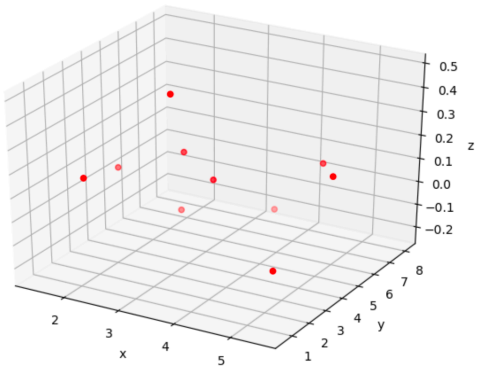
```python
def restore_data(data_rot, U, data_mean):
    #print(data_rot.transpose().shape)
    #print(U.transpose().shape)
    restored_data = U.transpose().dot(data_rot.transpose()) + data_mean.reshape(-1, 1)
    return np.array(restored_data)
```

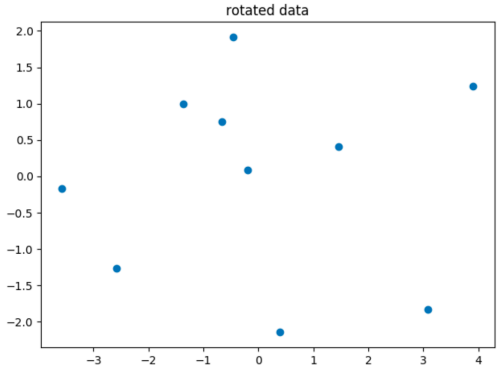
按照上一节所说的过程进行数据的恢复。

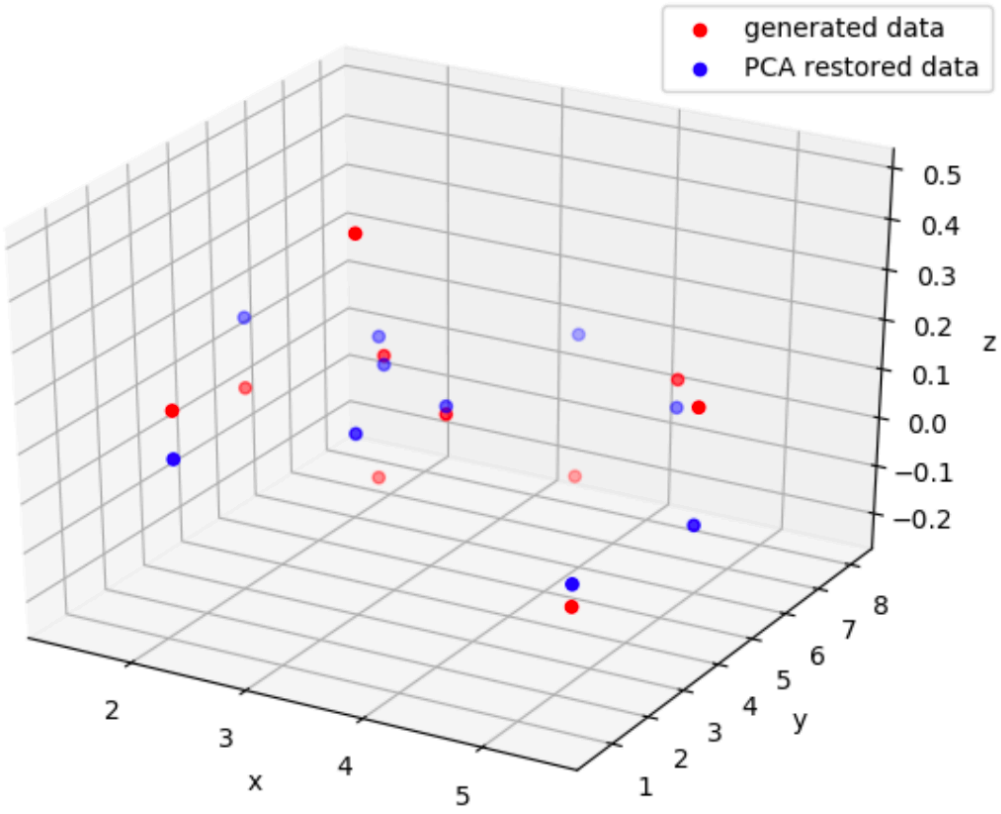
```python
def PCA(dataMat):
    data, data_mean = zero_mean(dataMat) #将所有数据的均值归一化到零附近
    cov = coverience_matrix(data) #计算协方差矩阵
    eigVals, eigVects = np.linalg.eig(np.mat(cov)) #求特征值和特征向量
    data_rot, U = rotate_data(data, eigVals, eigVects) # 旋转、降维
    restored_data = restore_data(data_rot, U, data_mean) # 恢复数据
    return data_rot, restored_data, U # 返回降维后的和恢复之后的数据以及旋转用的矩阵U以备用
```

合在一起，整合成一个函数。

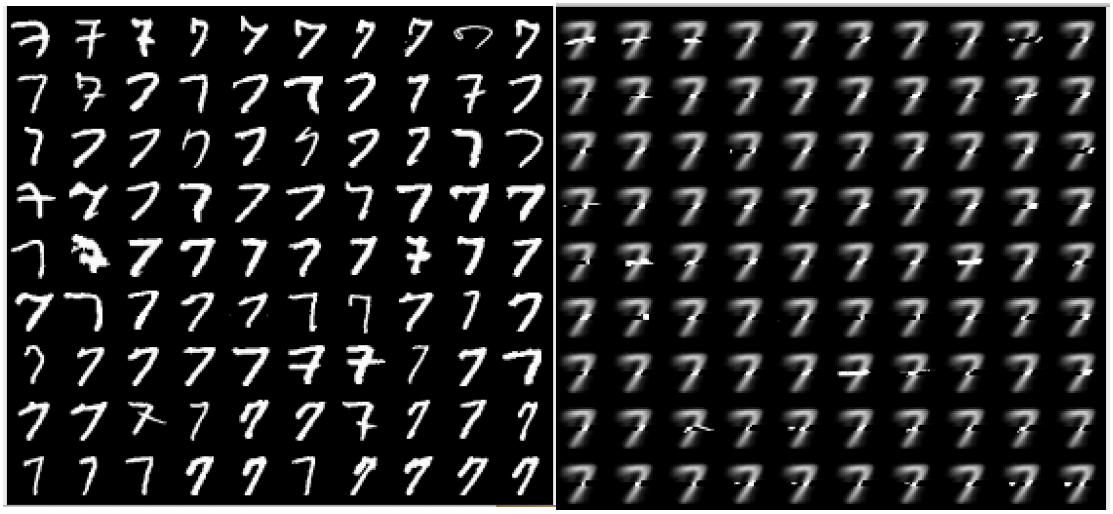
四、实验结果与分析







对于三维数据，我们得到了如上的结果，可以看到我们恢复的数据和原本产生的数据已经差不多了，只在第三个维度上有差距的



然后mnist手写数据数据集上我们有图上图所示的结果，可以看到左图是答案，右图是恢复的数据，恢复数据和答案

相差并不是很大。

五、结论

本文中实现了PCA，并且在生成数据和手写数字数据集上进行了测试，结果表明，PCA具有在保留大部分数据的情况下降维的能力。

六、参考文献

http://deeplearning.stanford.edu/wiki/index.php/主成分分析 斯坦福大学深度学习主成分分析博客。

七、附录：源代码（带注释）