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： 1170300520

姓名： 郭子阳

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）

二、实验要求及实验环境

实验要求

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. 找一个人脸数据（小点样本量），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人脸图像进行重建，比较一些它们与原图像有多大差别（用信噪比衡量）。

实验环境

- Python 3.7.0
- JupyterLab 1.1.3

三、设计思想（本程序中的用到的主要算法及数据结构）

算法原理

主成分分析（Principal components analysis，即PCA）是最重要的降维方法之一。在数据压缩消除冗余和数据噪音消除等领域都有广泛的应用。

PCA的主要思想是将n维特征映射到k维上，这k维是全新的正交特征也被称为主成分，是在原有n维特征的基础上重新构造出来的k维特征。PCA的工作就是从原始的空间中顺序地找一组相互正交的坐标轴，新的坐标轴的选择与数据本身是密切相关的。其中，第一个新坐标轴选择是原始数据中方差最大的方向，第二个新坐标轴选取是与第一个坐标轴正交的平面中使得方差最大的，第三个轴是与第1,2个轴正交的平面中方差最大的。依次类推，可以得到n个这样的坐标轴。通过这种方式获得的新的坐标轴，我们发现，大部分方差都包含在前面k个坐标轴中，后面的坐标轴所含的方差几乎为0。于是，我们可以忽略余下的坐标轴，只保留前面k个含有绝大部分方差的坐标轴。事实上，这相当于只保留包含绝大部分方差的维度特征，而忽略包含方差几乎为0的特征维度，实现对数据特征的降维处理。

通过计算数据矩阵的协方差矩阵，然后得到协方差矩阵的特征值特征向量，选择特征值最大(即方差最大)的k个特征所对应的特征向量组成的矩阵。这样就可以将数据矩阵转换到新的空间当中，实现数据特征的降维。

样本X和样本Y的协方差：

$$Cov(X,Y) = E[(X - E(X))(Y - E(Y))] = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$$

当样本是n维数据时，它们的协方差实际上是协方差矩阵

$$Cov(X,Y,Z) = \begin{bmatrix} Cov(x,x) & Cov(x,y) & Cov(x,z) \\ Cov(y,x) & Cov(y,y) & Cov(y,z) \\ Cov(z,x) & Cov(z,y) & Cov(z,z) \end{bmatrix}$$

算法实现

计算所给数据的协方差矩阵：

```
average = np.mean(matrix, axis = 0)
# 每个数据减去均值
matrix = matrix - average
cov = np.dot(matrix.T, matrix) / (n-1)
cov = np.array(cov)
```

pca实现：

```
def pca(matrix, d):
    n, p = matrix.shape
    print('维度：' + str(p))
    # 计算协方差矩阵
    average = np.mean(matrix, axis = 0)
    # 每个数据减去均值
    matrix = matrix - average
    cov = np.dot(matrix.T, matrix) / (n-1)
    cov = np.array(cov)
    print('协方差矩阵为：\n', cov)
    # 计算特征根与特征向量
    val, vec = np.linalg.eig(cov)

    # 选择最大的特征向量
    pairs = [(np.abs(val[i]), vec[:,i]) for i in range(len(val))]
    pairs.sort(reverse=True)
    feature = []
    for i in range(d):
        feature.append(pairs[i][1])
    feature = np.mat(feature)

    print(feature)

    new_data = np.dot(feature,data.T).T
    return new_data
```

图片压缩：

将图像转换为灰度图：

```python
def changeImage(path):
    img = Image.open(path)
    # 将图像转换成灰度图
    img = img.convert("L")
    width = img.size[0]
    height = img.size[1]
    data = img.getdata()
    # 为了避免溢出，对数据进行一个缩放，缩小100倍
    data = np.array(data).reshape(height,width)/100
    return data
```

使用pca压缩图像：

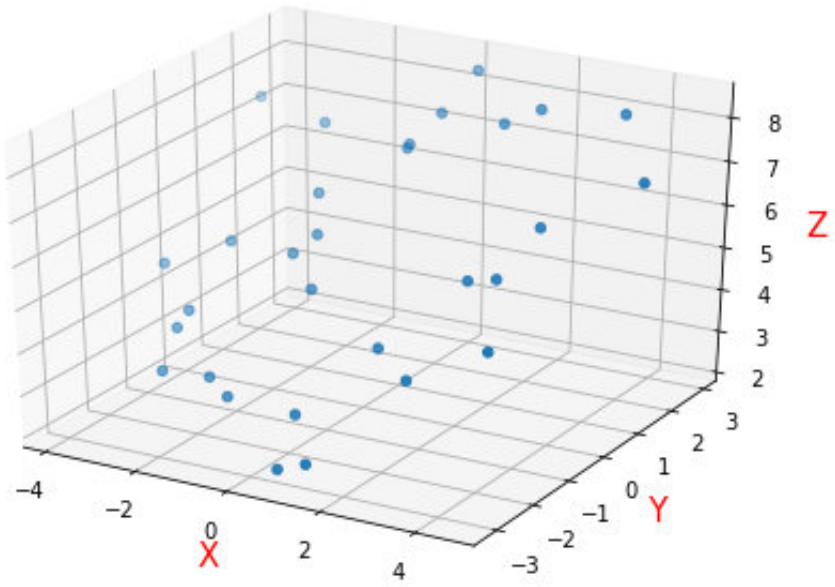
```python
def pca_pic(data,k):
    sample,featureNum = data.shape
    mean = np.array([np.mean(data[:,i]) for i in range(featureNum)]) # 求均值
    normal_data = data - mean  # 去中心化
    # 得到协方差矩阵
    matri = np.dot(np.transpose(normal_data),normal_data)
    val,vec = np.linalg.eig(matri)
    # 求前k个向量
    index = np.argsort(val)
    vecIndex = index[:-(k+1):-1]
    feature = vec[:,vecIndex]
    #降维后的数据
    new_data = np.dot(normal_data,feature)
    # 图片显示需要还原到原空间而不是直接返回降维后的数据
    # 将降维后的数据映射回原空间
    rec_data = np.dot(new_data,np.transpose(feature))+ mean
    return rec_data
```

计算新图像与原图像的信噪比：

```python
def error(data,recdata):
    sum1 = 0
    sum2 = 0
    D_value = data - recdata
    # 计算两幅图像之间的误差率，即信息丢失率
    for i in range(data.shape[0]):
        sum1 += np.dot(data[i],data[i])
        sum2 += np.dot(D_value[i], D_value[i])
    error = sum2/sum1
    return error
```

四、实验结果与分析

初始化三维散点：



计算协方差矩阵：

```
协方差矩阵为：
 [[5.44627237 0.7092216  0.7092216 ]
 [0.7092216  3.39077506 3.39077506]
 [0.7092216  3.39077506 3.39077506]]
```

降为二维后的数据为：

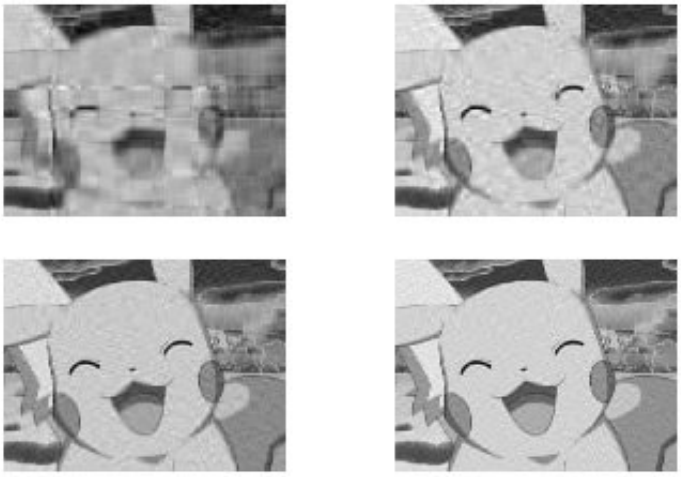
```
[[-0.75417425  4.76022182]
 [-5.32887178  3.24052393]
 [-1.88222938  4.14376336]
 [-0.1788507  -1.27782037]
 [-4.04689275  6.55616267]
 [-8.35947794  0.4766504 ]
 [-4.3696443  -0.27682909]
 [-3.58729537  3.70799127]
 [-5.2548065   3.19847544]
 [-5.91282967 -0.37125898]
 [-3.76696386 -1.90572745]
 [-2.32084896  1.94932773]
 [-7.39350472  1.86476459]
 [-6.76133672  2.1617696 ]
 [-2.41533529 -1.21983522]
 [-2.450428   -0.19257886]
 [-7.10791613  3.61348132]
 [ 0.5533455   2.67551991]
 [-6.12109216  3.34889353]
 [ 0.23738401 -0.95091214]
 [-0.62656102 -0.19331218]
 [ 0.02103932  1.17531925]
 [-3.05911366  2.9170246 ]
```

```
[-4.74450194 -0.67957644]
[-2.51964139  2.92955116]
[ 0.01882549  1.83018349]
[-0.50815069  3.44146096]
[-7.7240846  -1.09079871]
[-4.53333035  5.00258588]
[-0.1484229   3.2888653 ]]
```

图片压缩时，读入的图片：

将图片转换为灰度图：

分别取前10、20、30、40个特征压缩图片的结果：



分别计算信噪比：

```
(0.011047401427167642+0j)
(0.005509545911150714+0j)
(0.0034291469792751437+0j)
(0.0023158294904146557+0j)
```

五、结论

PCA是数据分析时通用的数据降维方法，可以快速提取出数据中较为主要的几个特征，在降低维度的同时，最大质量保证数据的完整性。

六、参考文献

[1] Principal component analysis - Wikipedia, https://en.wikipedia.org/wiki/Principal_component_analysis