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

学号：1170300418
姓名：于新蕊

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

- 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。
- 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

windows64，pycharm，python3.0，anaconda

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

PCA(主成分分析，Principal Component Analysis)是最常用的一种统计分析、简化数据集的方法。通过PCA可以使要分析的数据的维度降低，且这些维度还会包含原数据集的主要信息。关于PCA的推导有两种方式：最小投影距离和最大投影方差。

- 最小投影距离：样本点到这个超平面的距离都足够近
- 最大投影方差：样本点在这个超平面上的投影尽可能分开

1.1 中心化和坐标变化

假设m个n维数据$X=(x^{(1)},x^{(2)},...,x^{(m)})$，我们令$x^{(i)}=x^{(i)}-\mu$，其中$\mu=\frac{1}{m}\sum\limits_{i=1}^{m}x^{(i)}$。

这样我们会得到$\sum\limits_{i=1}^{m}x^{(i)}=0$，这就是中心化。

中心化的目的是方便求协方差矩阵：中心化后X^TX就是样本集的协方差矩阵。

我们知道$x^{(i)}$是一个n维向量，这个n维空间中的基底是$(1,0,\cdots,0)^T$，$(0,1,\cdots,0)^T$，$\cdots$，$(0,0,\cdots,1)^T$。如果我们换一组基底，$x^{(i)}$又是另一种表现形式。

现在假设我们n维空间中的基底时$W=(w^{(1)},w^{(2)},\cdots,w^{(n)})$,那么$Z=W^TX$。

假设我们只取W中的前d个向量，即$W=(w^{(1)},w^{(2)},\cdots,w^{(d)})$，其中$d<n$。那么$Z=W^TX$就是$d$维空间下的一个坐标，即**Z是X从n维空间到d维空间的一个投影。**

那么我们从d维空间下的坐标Z变化到n维空间，就是$X=ZW$。

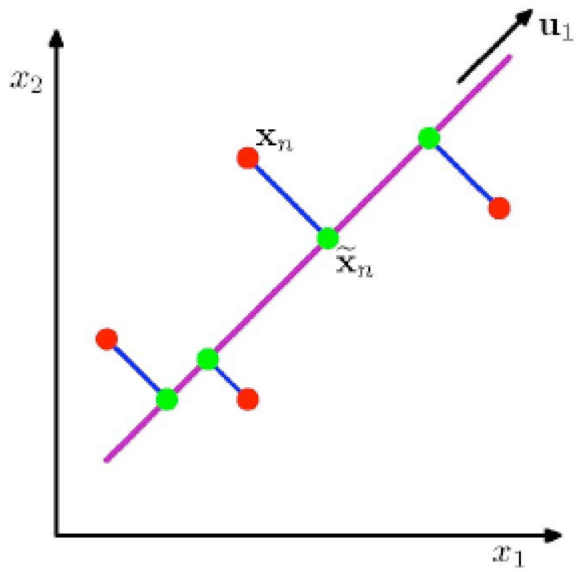
总结就是：

某种投影$W=(w^{(1)},w^{(2)},\cdots,w^{(d)})$

投影坐标：$Z=W^TX$

还原坐标：$X = ZW$

显然我们想要$x^{(i)}$这些点的投影越分散越好、距离平面越近越好，就像这样：



因此我们有两种策略：最小投影距离和最大投影方差。事实上二者相当于优化同一个函数。

1.2 PCA的推导:基于最小投影距离

现在我们考虑整个样本集，我们希望所有的样本到这个超平面的距离足够近，即最小化

$$\arg\min_{W} \sum_{i=1}^{m} ||\overline{x}^{(i)} - x^{(i)}||_2^2$$

将这个式子进行整理，可以得到：

$$\begin{aligned}
\sum_{i=1}^{m} ||\overline{x}^{(i)} - x^{(i)}||_2^2 &= \sum_{i=1}^{m} ||Wz^{(i)} - x^{(i)}||_2^2 \\
&= \sum_{i=1}^{m} (Wz^{(i)})^T (Wz^{(i)}) - 2\sum_{i=1}^{m} (Wz^{(i)})^T x^{(i)} + \sum_{i=1}^{m} x^{(i)T} x^{(i)} \\
&= \sum_{i=1}^{m} z^{(i)T} z^{(i)} - 2\sum_{i=1}^{m} z^{(i)T} W^T x^{(i)} + \sum_{i=1}^{m} x^{(i)T} x^{(i)} \\
&= \sum_{i=1}^{m} z^{(i)T} z^{(i)} - 2\sum_{i=1}^{m} z^{(i)T} z^{(i)} + \sum_{i=1}^{m} x^{(i)T} x^{(i)} \\
&= -\sum_{i=1}^{m} z^{(i)T} z^{(i)} + \sum_{i=1}^{m} x^{(i)T} x^{(i)} \\
&= -tr(W^T(\sum_{i=1}^{m} x^{(i)} x^{(i)T})W) + \sum_{i=1}^{m} x^{(i)T} x^{(i)} \\
&= -tr(W^T X X^T W) + \sum_{i=1}^{m} x^{(i)T} x^{(i)}
\end{aligned}$$

因此相当于

$$arg\max_{W}\ tr(W^TXX^TW)\ \ s.t.W^TW=I$$

1.3 PCA的推导:基于最大投影方差

对于任意一个样本$x^{(i)}$，在新的坐标系中的投影为$W^Tx^{(i)}$,在新坐标系中的投影方差为$W^Tx^{(i)}x^{(i)T}W$。要使所有的样本的投影方差和最大，也就是最大化$\sum\limits_{i=1}^{m}W^Tx^{(i)}x^{(i)T}W$，即

$$arg\max_{W}\ tr(W^TXX^TW)\ \ s.t.W^TW=I$$

1.4 参数求解

拉格朗日乘数法：

$$J(W)=tr(W^TXX^TW+\lambda(W^TW-I))$$
$$\frac{\partial J(W)}{W_i}=0$$
$$XX^TW=（-\lambda）W$$

可以看出W_i是特征向量。

$$tr(W^TXX^TW)=\sum_{i=1}^{d}\lambda_i$$

因此$W=(w^{(1)},w^{(2)},\cdots,w^{(d)})$，其中$w^{(i)}$是$XX^T$第$i$大的特征值对应的特征向量。

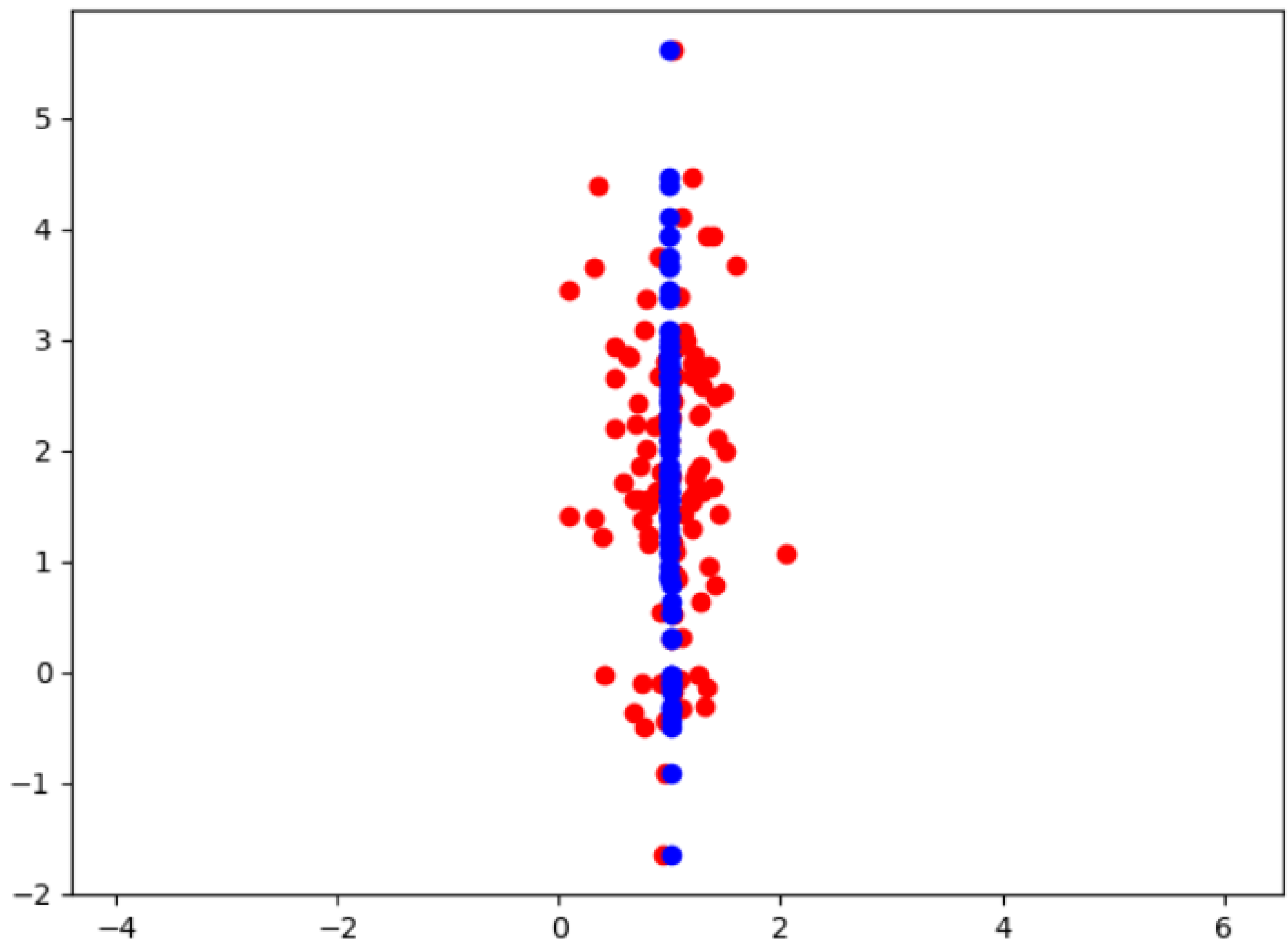
2.算法实现

给定样本集$X=x^{(1)},x^{(2)},...,x^{(m)}$和低维空间的维数$d$

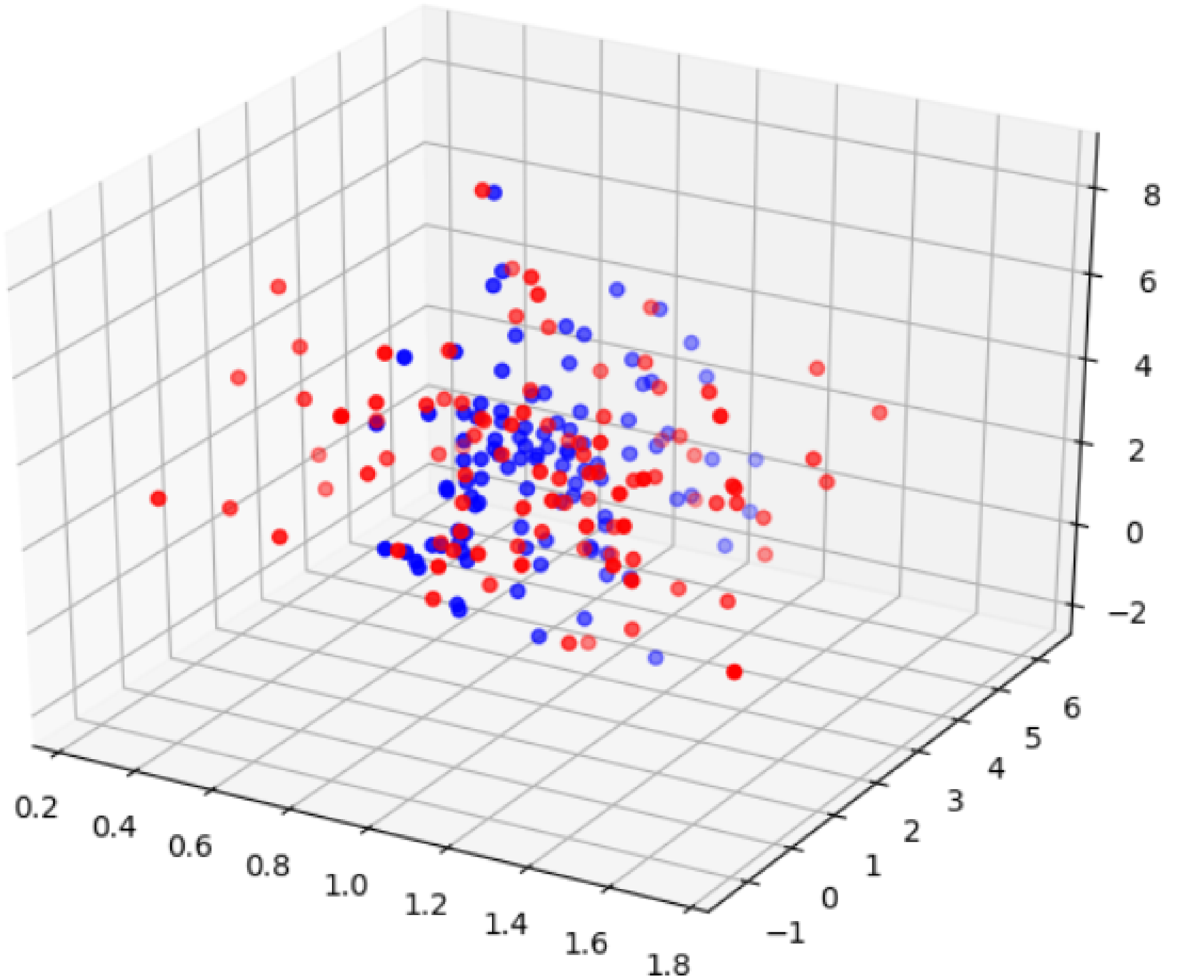
1. 对所有的样本进行中心化操作：
   1. 计算样本均值$\mu=\frac{1}{m}\sum\limits_{j=1}^{m}x^{(j)}$
   2. 所有样本减去均值$x^{(j)}=x^{(j)}-\mu,\ j\in 1,2,...,m$
2. 计算样本的协方差矩阵X^TX
3. 对协方差矩阵X^TX进行特征值分解
4. 取最大的d个特征值对应的单位特征向量$w_1,w_2,...,w_d$，构造投影矩阵$W=(w_1,w_2,...,w_d)$
5. 输出投影矩阵W

四、实验结果与分析

二维降一维



三维降二维



人脸数据

每一行单过一个样本，因此数据集大小是23×2500。从2500维降到10维的效果：

信噪比公式:

$$MSE = \frac{1}{MN}\sum_{i=0}^{M-1}\sum_{j=0}^{n-1}||I(i,j) - K(i,j)||^2$$

$$PSNR = 10\log_{10}(\frac{MAX_I^2}{MSE}) = 20\log_{10}(\frac{MAX_I^2}{\sqrt{MSE}})$$

人脸数据信噪比:

```
image 1: snr = 26.567129607343887
image 2: snr = 31.183839258090778
image 3: snr = 28.05962724136331
image 4: snr = 27.60549550427767
image 5: snr = 28.05209735007121
image 6: snr = 29.468379307458846
image 7: snr = 27.585224583184498
image 8: snr = 27.560216289273654
image 9: snr = 26.178151863193104
image 10: snr = 29.795966577101403
image 11: snr = 28.596744322666474
image 12: snr = 27.29037717122575
image 13: snr = 26.97213009186286
image 14: snr = 27.911257238116278
image 15: snr = 31.422632481966296
image 16: snr = 30.714372077135916
image 17: snr = 25.644284571409028
image 18: snr = 27.300237309623128
image 19: snr = 29.424072998446125
image 20: snr = 26.68660596532291
image 21: snr = 29.893610136085258
image 22: snr = 29.163073050010276
image 23: snr = 25.27842702193753
```

分析

三维数据降到二维时，降维后的数据分布在一个平面(2维)上，并且与方差最小的1维相垂直。二维数据同理。一个23×2500的数据，可以粗略用10维数据来表示，通过PCA降维的方法。

五、结论

PCA算法中舍弃了$n-d$个最小的特征值对应的特征向量，用交大的d特征值对应的特征向量来表示数据，称其为主成分。PCA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，如图片压缩。

六、参考文献

[吴恩达机器学习(Coursera)]

七、附录：源代码（带注释）

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from PIL import Image
import warnings
warnings.filterwarnings("ignore")


def generate_data(dim=3, size=100):
    if dim == 2:
        mean = [1, 2]
        cov = [[0.1, 0], [0, 2]]
    if dim == 3:
        mean = [1, 2, 3]
        cov = [[0.1, 0, 0], [0, 3, 0], [0, 0, 3]]
    data = []
    for index in range(size):
        data.append(np.random.multivariate_normal(mean, cov).tolist())
    data = np.array(data)
    data.reshape(size, dim)
    return data


def pca(data, reduced_dim=2):
    mean = np.mean(data, axis=0)
    data = data - mean
    cov = np.dot(data.T, data)
    eig_vals, eig_vecs = np.linalg.eig(cov)
    idx = np.argsort(-eig_vals, axis=0)[:reduced_dim:]
    # print(np.dot(np.dot(eig_vecs.T, cov), eig_vecs))
    eig_vecs = eig_vecs[:, idx]
    return mean, data, eig_vecs
```

```python
def recover_data(w, centered_data, mean):
    return np.dot(np.dot(centered_data, w), w.T) + mean


def plot(origin_data, pca_data):
    fig = plt.figure()
    dim = origin_data.shape[1]
    if dim == 3:
        ax = Axes3D(fig)
        ax.scatter(origin_data[:, 0], origin_data[:, 1], origin_data[:, 2], c='r',
label='Original Data')
        ax.scatter(pca_data[:, 0], pca_data[:, 1], pca_data[:, 2], c='b', label='PCA Data')
    if dim == 2:
        plt.axis('equal')
        plt.scatter(origin_data[:, 0], origin_data[:, 1], c='r', label='Original Data')
        plt.scatter(pca_data[:, 0], pca_data[:, 1], c='b', label='PCA Data')
    plt.show()


def calc_snr(source, target):
    diff = source - target
    diff = diff ** 2
    mse = np.sqrt(np.mean(diff))
    return 20 * np.log10(255.0 / mse)


def test(path='./Japanese'):
    data = []
    for i in range(1, 24):
        new_path = path + '/' + str(i) + '.tiff'
        img = np.array(Image.open(new_path).resize((50, 50)).convert('L'),
'f').astype(np.float).flatten()
        data.append(img)
        # w = np.array(w, dtype=np.uint32)
    data = np.array(data).reshape(23, -1)
    mean, centered_data, w = pca(data, 10)
    pca_data = recover_data(w, centered_data, mean)
    pca_data[pca_data < 0] = 0
    # print(pca_data)
    pca_data = pca_data.astype(np.uint8)
    for i in range(1, 24):
        new_data = pca_data[i - 1].reshape(50, 50)
        new_image = Image.fromarray(new_data, mode='L')
        new_image.save(path + '/' + str(i) + '_.tiff')
        print('image {}: snr = {}'.format(i, calc_snr(data[i - 1].flatten(),
new_data.flatten())))


if __name__ == '__main__':
    # data = generate_data(dim=3)
    # mean, centered_data, w = pca(data, reduced_dim=2)
    # pca_data = np.dot(np.dot(centered_data, w), w.T) + mean
```

```
# plot(data, pca_data)
test()
```