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：

姓名：

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

2.1 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）。

2.2 实验环境

Python3.7+PyCharm+Jupyter notebook

三、设计思想（本程序中的用到的主要算法及数据结构）

PCA 是一种常见的数据分析方式，常用于高维数据的降维，可用于提取数据的主要特征分量。PCA 的数学推导可以从最大可分性和最近重构性两方面入手，这二者可以得到等价推导。前者的优化条件是划分之后方差最大，而后者的优化条件就是原始数据点到划分平面的距离最小。本实验证明将从最大可分性入手证明。

3.1 基

要解决降维问题，事实上就是找到一组基，使用原始数据集上的点在这一组基下的投影结果即可。在线性代数，基(basis)（也称为基底）是描述、刻画的向量空间的基本工具。向量空间的基是它的一个特殊的子集，基的元素称为基向量。PCA 的目的就是找到一组基，在这组基构建的空间中表示原始的数据。

可以写出基变换的简单矩阵表示形式：

$$(p_1, p_2, .., p_R)^T(a_1, a_2, .. a_M) = \begin{pmatrix} p_1a_1 & p_2a_2 \dots p_1a_M \\ \dots & \\ p_Ra_1 & p_Ra_2 \dots p_Ra_M \end{pmatrix}$$

其中p_i表示一个行向量，表示第 i 个基，a_j表示一个列向量，也就是发生映射之前的原始数据的值。而上述的矩阵相乘事实上就是一种线性空间的变换，将原始数据映射到了由p_i表示的向量空间中。

3.2 最大可分性推导

最大可分性的优化角度就是当原始数据从一组基表示转化到另外一组基的表示时，希望映射之后的点之间的距离越远越好，事实上也就是要求映射之后的点组成的数据集的方差越大越好。从熵的角度来理解，尽管我们对数据集进行了降维，但是我们依然希望降维之后的数据能

携带较多的信息，而如果映射之后的数据的距离太近那么将会导致熵相对较小，携带的信息也就相对较少。

接下来给出方差和协方差的表示：

$$Var(a) = \frac{1}{m}\sum_{i=1}^{m}(a_i - \mu)^2$$

又因为在预处理数据的时候将数据零均值化，那么方差简化如下：

$$Var(a) = \frac{1}{m}\sum_{i=1}^{m}a_i^2$$

同理，为了方便计算，我们将协方差的系数进行了一定的简化，简化后的协方差计算公式如下：

$$Cov(a,b) = \frac{1}{m}\sum a_i b_i$$

方差刻画了数据的离散程度，而协方差则刻画了两个变量之间的线性相关关系，如果协方差为 0 则表示两个变量线性不相关。

因此，优化目标表示如下：将一组 N 维的向量降为 K 维向量，其目标是选择 K 个单位正交基，使得原始数据变换到这组基上之后，各变量两两协方差为 0，方差尽可能大。

为了表示方便，假设我们现在拥有 m 个 n 维数据，将其排列为数据矩阵X_{n*m},定义$C_{n*n} = \frac{1}{m}XX^T$，C 是一个对称矩阵，其中对角线上的元素就是每个变量的方差，其余位置的元素表示变量之间两两之间的协方差，因此优化的目标就是使得矩阵 C 成为一个对角矩阵，且对角线上的元素从上到下的大小顺序是从大到小。

设原始数据矩阵 X 对应的协方差矩阵为 C，而 P 是一组基按行组成的矩阵，设 Y=PX，则 Y 为 X 对 P 做基变换后的数据。设 Y 的协方差矩阵为 D，我们推导一下 D 与 C 的关系：

$$\begin{aligned} D &= \frac{1}{m}YY^T \\ &= \frac{1}{m}PXX^TP^T \\ &= PCP^T \end{aligned}$$

因此优化目标转换为寻找矩阵 P，使得PCP^T是一个对角矩阵且对角线上元素从大到小排列。那么 P 的前 K 行就是我们要找的基。

又因为协方差矩阵 C 是实对称矩阵，因此有如下性质：

实对称矩阵不同特征值对应的特征向量必然正交。

设特征向量λ 重数为 r，则必然存在 r 个线性无关的特征向量对应于 λ，因此可以将这 r 个特征向量单位正交化。

也就是说对于 C 来说可以找到 n 个正交的特征向量，设为$e_1, e_2 \ldots e_n$，将特征向量组成矩阵$E = (e_1, e_2, .. e_n)$，则对于协方差矩阵，有：

$$E^T CE = \begin{pmatrix} \lambda_1 & & \\ & \lambda_2 & \\ \cdots & & \\ & & \lambda_n \end{pmatrix}$$

其中λ_i就是 C 对应的特征值，因此我们找到了需要的矩阵 P：$P = E^T$。

接下来，如果我们想把数据降到 K 维，只需要取 P 的前 K 行作为基，Y=PX 就是降维之后的数据矩阵。

3.3 信噪比

峰值信噪比 PSNR 经常用作图像压缩等领域中信号重建质量的测量方法，它常简单地通过均方差（MSE）进行定义。两个 m×n 单色图像 I 和 K，如果一个为另外一个的噪声近似，那么它们的的均方差定义为：

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

而信噪比可以定义为：

$$PSNR = 10 * log(\frac{MAX_I^2}{MSE})$$

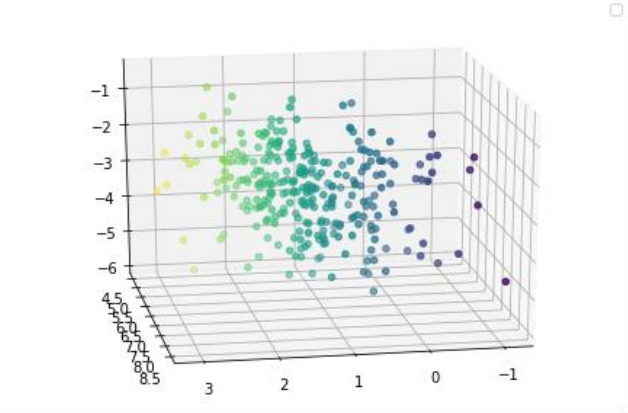
信噪比可以用来衡量两个信号之间的相似程度，信噪比越大表示两个信号越相似。

四．实验结果与分析

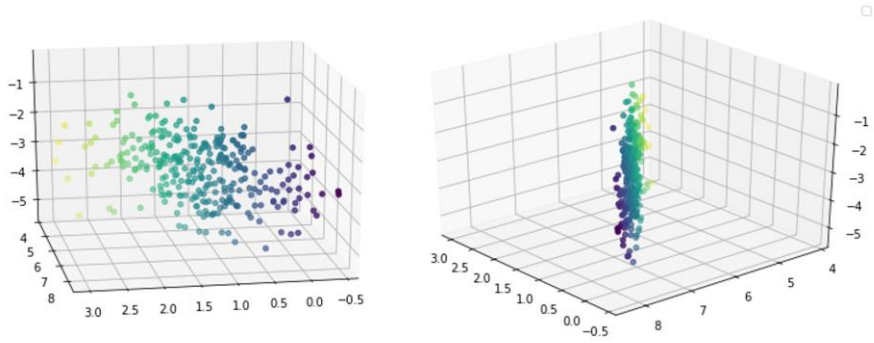
4.1 手工数据集

测试 PCA 性能时使用手工生成的三维高斯分布数据和三维瑞士卷数据集来测试 PCA 的结果。

首先考虑三维高斯分布数据，生成过程中 x 轴的方差明显小于其余两个轴，原始数据分布情况如下：

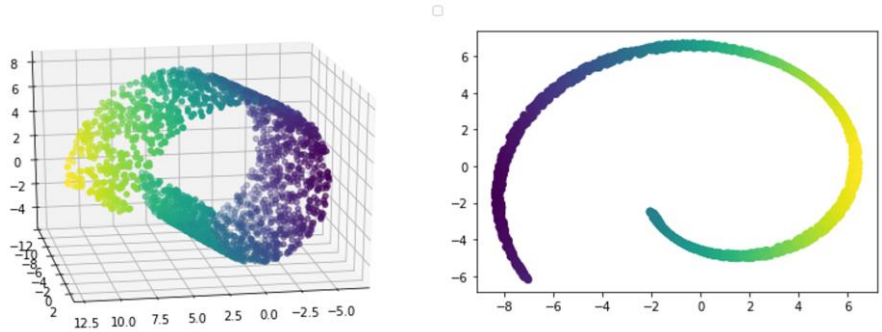


PCA 降维之后重构之后的数据表示如下（为了方便观察更换了角度）：

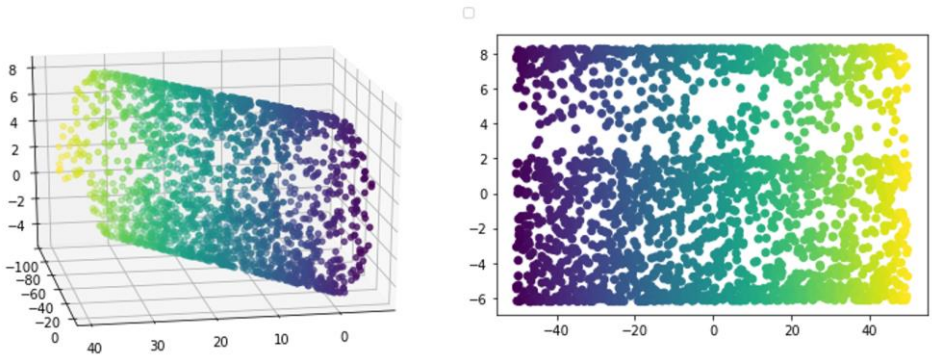


可以发现 PCA 确实将数据投影到了一个二维的平面上，满足我们的需求。由于高斯分布选择的投影平面特征并不是很明显，以下使用瑞士卷数据集可以明显发现 PCA 选择的投影平面的逻辑。

以下数据集为厚度为 10 的瑞士卷数据的原始数据与降维后数据示意图：



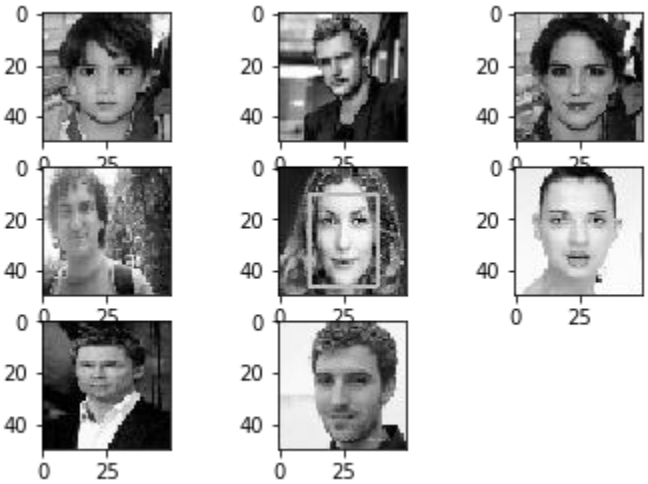
以下数据集为厚度为 100 的瑞士卷数据的原始数据与降维后数据示意图：



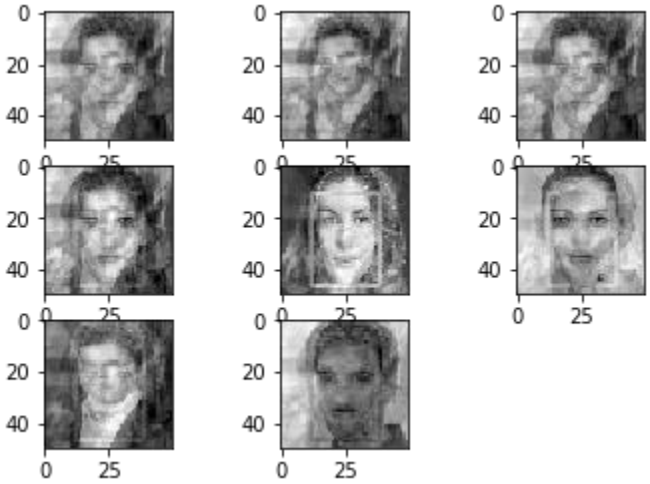
可以发现当瑞士卷的厚度比较小时候，也就是在 Z 轴方向上点比较稀疏的时候，PCA 降维之后的结果比较明显是正投影的漩涡形状；而当瑞士卷的厚度比较大的时候，也就是在 Z 轴方向点比较密集的时候，降维的结果就不是漩涡形状了。这也和我们对于 PCA 的分析相一致。

4.1 人脸数据集

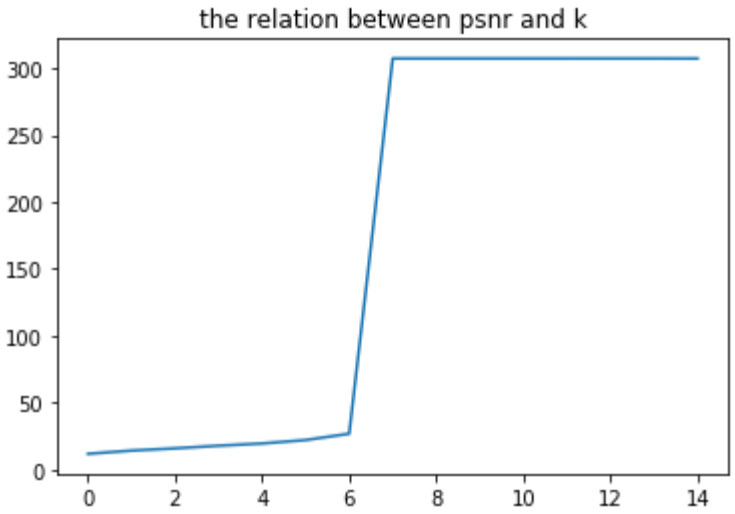
选择了 8 张人脸图片，由于部分图片大小太大，因此做了统一大小处理之后转换为灰度图之后结果如下：



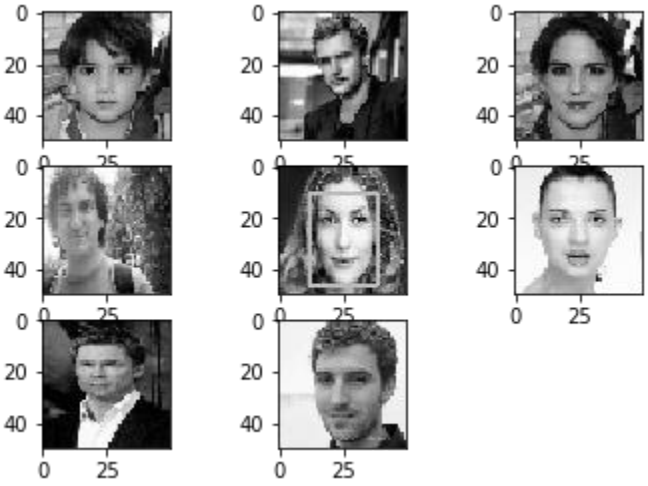
使用 PCA 降维到 2 维之后重构结果如下：



可以发现实现降维之后重构的图片只存在基本轮廓，一些细节和原图的差距相对来说较大。以下使用信噪比来衡量降维之后重构图片的效果好坏。以下为信噪比与降维维数的关系折线图，可以发现当取 7 个特征值对应的特征向量作为基之后基本上能带来的信噪比已经达到最大，后续继续增加基的数量基本没有效果。也就是说图像降为 7 维之后基本已经可以保证大部分信息都能够得到保留，由此可见 PCA 是相当有用的，极大压缩了数据量。



可以查看降维到 7 维之后的重构图片如下：



可以发现降维到 7 维之后重构的图片和原图在肉眼看来基本一致，和 PSNR 结果显示的基本一致。

五．结论

1.PCA 通过寻找协方差矩阵的最大的 K 个特征值对应的向量作为一组基来表示源数据达到降维的效果，通过实验发现这种手段是相当可靠的。

2.经过实验发现，实验采用的是 50*50 的人脸图片，也就是每一张图片的特征是 2500 维的向量，而使用 PCA 之后发现使用 7 维的基就可以较好表示图片，如果使用于图片压缩将会大规模减小图片的大小。

3.同时需要注意的是 PCA 算法提高了样本的采样密度，并且由于较小特征值对应的特征向量往往容易受到噪声的影响，PCA 算法舍弃了这部分“次要成分”，一定程度上起到了降噪的效果。

六．参考文献

[1]周志华,《机器学习》,清华大学出版社,2016

七、附录：源代码（带注释）

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import os
import cv2
import math
def rotate(X,theta):
    """
    旋转数据
    :param X:数据集，格式为 3*n
    :param theta:需要旋转的角度
    :return:旋转后的数据，格式为 3*n
    """
    rotate = [[np.cos(theta), -
np.sin(theta), 0], [np.sin(theta), np.cos(theta), 0], [0, 0, 1]]
    return np.dot(rotate,X)
def generate_data(num=100,y_scale=100):
    """
    生成瑞士卷数据
    :param num: 生成的数据点的数量
    :param y_scale:生成的瑞士卷的厚度
    :return: 数据集，格式为 n*3
    """
    t=np.pi*(1+2*np.random.rand(1,num))
    x=t*np.cos(t)
    y=y_scale*np.random.rand(1,num)
    z=t*np.sin(t)
    X=np.concatenate((x,y,z))
    X=rotate(X,60)
    X=X.T
    return X
def generate_guass_data(num):
    """
    生成三维高斯分布数据，并旋转
    :param num:需要的数据点个数
    :return:数据集，格式为 n*3
    """
    mean=[5,4,-3]
    cov=[[0.1,0,0],[0,1,0],[0,0,1]]
    X=np.random.multivariate_normal(mean, cov, size=num).T#3*num
    X=rotate(X,40 * np.pi / 180).T
    return X
def show_3D(X):
    fig = plt.figure()
    ax = Axes3D(fig)
    ax.view_init(elev=20, azim=80)
    ax.scatter(X[:, 0], X[:, 1], X[:, 2], c=X[:, 0])
    ax.legend(loc='best')
    plt.show()
def show_3D_change_view(X):
    fig = plt.figure()
```

```
    ax = Axes3D(fig)
    ax.view_init(elev=30, azim=140)
    ax.scatter(X[:,0], X[:,1], X[:,2], c=X[:,0])
    ax.legend(loc='best')
    plt.show()
def PCA(X,k):
    """
    PCA 降维
    :param X:数据，格式为 n*d,n 表示数据集中数据个数，d 表示维度
    :param k: 需要降到的维度数量，满足 k<=d
    :return:降维后的结果
    """
    n=X.shape[0]
    d=X.shape[1]
    assert d>=k
    #零均值化
    X_mean=np.sum(X,0)/n
    X=X-X_mean
    ##求协方差矩阵
    Cov=np.dot(X.T,X)#d*d
    ##求特征值和特征向量
    eigValues,featureVectors=np.linalg.eig(Cov)
    ##特征值排序
    eigSort=np.argsort(eigValues)
    ##选前 k 个特征值对应的特征向量
    eigRes=featureVectors[:,eigSort[:-(k+1):-1]]#d*k
    ##计算 PCA 结果
    pca_res=np.dot(X,eigRes)##n*k
    return X,eigRes,X_mean
def show(X):
    plt.scatter(X[:, 0].tolist(), X[:, 1].tolist(), c=X[:, 0].tolist())
    plt.show()
def self_data_run():
    X=generate_data(2000)
    X,eigRes,X_mean=PCA(X,2)
    show(np.dot(X,eigRes))
def read_data(path,compress_size):
    """
    读取人脸数据集
    :param path: 人脸数据集地址
    :param compress_size: 图片压缩比例，格式为(,)
    :return: 训练数据集
    """
    file_list=os.listdir(path)
    X_list=[]
    i=1
    for file in file_list:
        plt.subplot(3,3,i)
        file_path=os.path.join(path,file)
                img=cv2.resize(cv2.imread(file_path),compress_size)
                img_grey=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
                plt.imshow(img_grey,cmap="gray")
                img_temp=img_grey.reshape(-1)#展开为一维数据
                X_list.append(img_temp)
                i+=1
           plt.show()
           return np.asarray(X_list)

      def face_data_run():
```

```
110.             compress_size=(50,50)
111.             X=read_data('./data',compress_size)
112.             n,d=X.shape
113.             X,eigRes,X_mean=PCA(X,2)
114.             #降维
115.             eigRes=np.real(eigRes)
116.             pca=np.dot(X,eigRes)##n*k
117.             ##重构
118.             recons=np.dot(pca,eigRes.T)+X_mean
119.             for i in range(n):
120.                 plt.subplot(3,3,i+1)
121.                 plt.imshow(recons[i].reshape(compress_size),cmap="gray")
122.             plt.show()
123.         def PSNR(X_real,X_recon):
124.             """
125.             计算信噪比
126.             :param X_real: 未降维的图片
127.             :param X_recon: 重构图片
128.             :return: 信噪比值
129.             """
130.             mse=np.mean((X_real/255.-X_recon/255.)**2)
131.             MAX=1
132.             return 20*math.log10(MAX/math.sqrt(mse))
133.         if __name__=="__main__":
134.             read_data('./data',(50,50))
```