**哈尔滨工业大学计算机科学与技术学院**

机器学习实验报告

|  |  |
| --- | --- |
| **姓名** | 徐亚楠 |
| **学号** | 1190201224 |
| **实验题目** | PCA |
| **报告时间** | 2021.11.20 |

1. **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）

1. **实验要求及实验环境**

**2.1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。

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:** Pycharm 2021 windows10 python3.8

**三、设计思想（本程序中的用到的主要算法及数据结构）**

PCA(主成分分析，Principal Component Analysis)是最常⽤的⼀种降维⽅法。

PCA的主要思想是将 D维特征通过⼀组投影向量映射到K维上，这K维是全新的正交特征，称之为主成分，采⽤主成分作为数 据的代表，有效地降低了数据维度，且保留了最多的信息。关于PCA的推导有两种⽅式：最⼤投影⽅差 和最小投影距离。

最⼤投影⽅差：样本点在这个超平⾯上的投影尽可能分开

最小投影距离：样本点到这个超平⾯的距离都⾜够近

**3.1中心化**

在开始PCA之前需要对数据进行预处理，即对数据中心化。设数据集 ，其中，即 X 是一个 的矩阵。则此数据集的中心向量（均值向量）为：

对数据集每个样本均进行操作：，就得到了中心化后的数据，此时有。

中心化可以给后面的计算带来极大的便利，因为中心化之后的常规线性变换就是绕原点的旋转变化，也就是坐标变换。此时，

协方差为

设使用的投影坐标系的一组**标准正交基**为

，故有 ，使用这组基变换中心化矩阵 X，得降维压缩后的矩阵 ，重建得到

。

## 3.2 最大投影方差

对于任意一个样本 ，在新的坐标系中的投影为 ，在新坐标系中的投影方差为 。要使所有的样本的投影方差和最大，也就是求 ，即

求解：在 方向投影后的方差

因为 是投影方向，且已经假设它是单位向量，即 $u\_1^Tu\_1=1$，用拉格朗日乘子法最大化目标函数：

对 求导，令导数等于0，解得 ，显然，和 是一组对应的 S 的特征向量和特征值，所以有 $，结合在 方向投影后的方差式，可得求得最大化方差，等价于求最大的特征值。

要将 d 维的数据降维到 k 维，只需计算前 k个最大的特征值，将其对应的特征向量（ 的）转为行向量（ 的）组合成特征向量矩阵 ，则降维压缩后的矩阵为 。

## 3.3 最小投影距离

现在考虑整个样本集，希望所有的样本到这个超平面的距离足够近，也就是得到 Y 后，与 X 的距离最小。即求：

可以看到，最小投影距离与我们在最大投影方差中得到的结果是一致的，只是两种不同的思考角度。

1. **实验结果与分析**

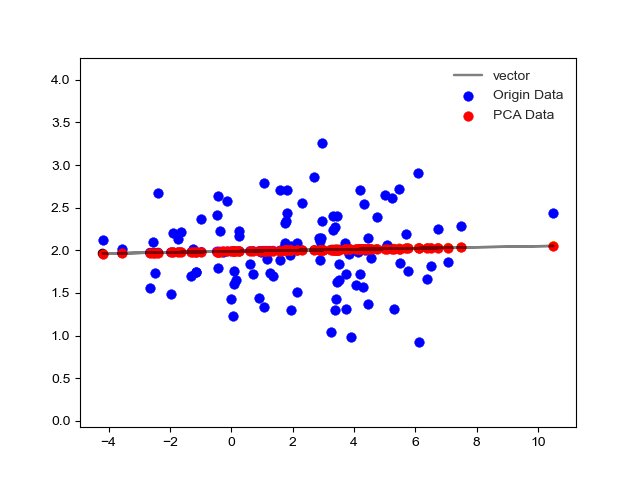
## 4.1生成数据测试

我们通过可视化二维和三维数据的降维来观察PCA的效果。

### 4.1.1 二维降到一维

生成高斯分布数据的参数：

可以看到第2维的⽅差远⼩于第1维的⽅差，所以第2维包含了更多的信息，进⾏PCA，得到的结果如下：

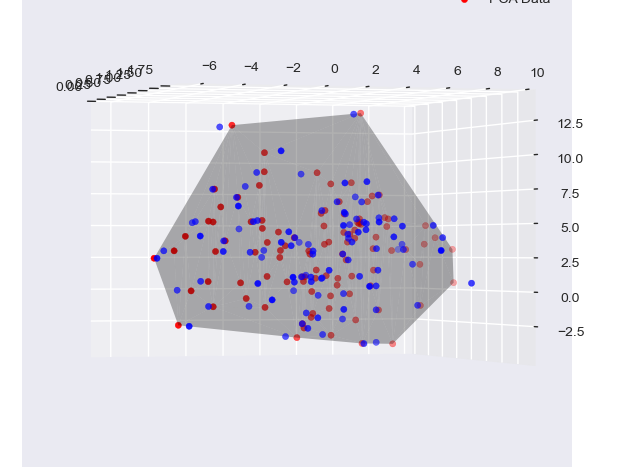
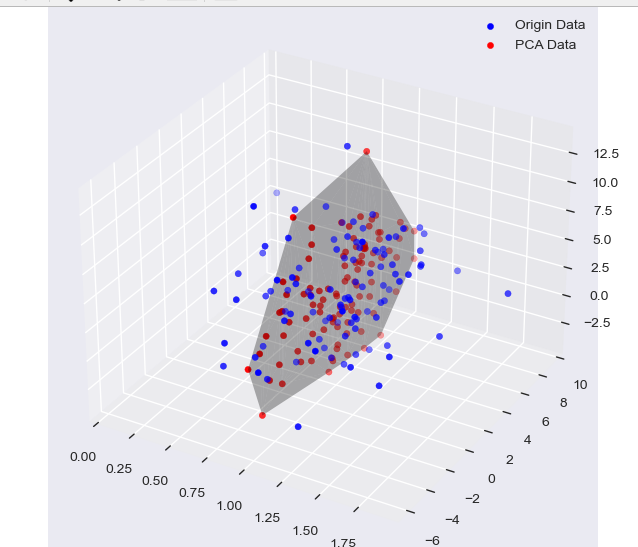


可以看到在PCA之后的数据分布在直线(1维)上，另外其在横轴上的⽅差更⼤，纵轴上的⽅差更⼩，所以在进⾏PCA之后得到的直线与横轴接近。

### 4.1.2 三维降到二维

生成高斯分布数据的参数：

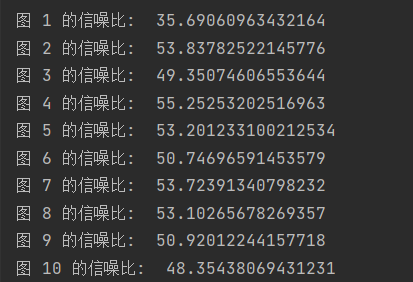
第1维的⽅差是远⼩于其余两个维度的，所以在第1维相较于其他两维信息更少，如下是PCA得到的结果：



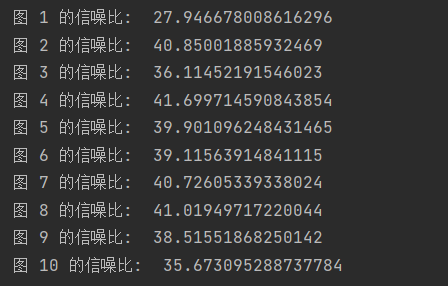
## 4.2 人脸数据测试

## 信噪比计算公式

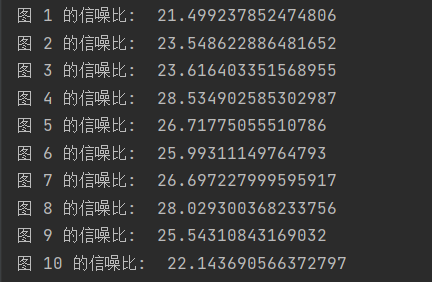
**100维**



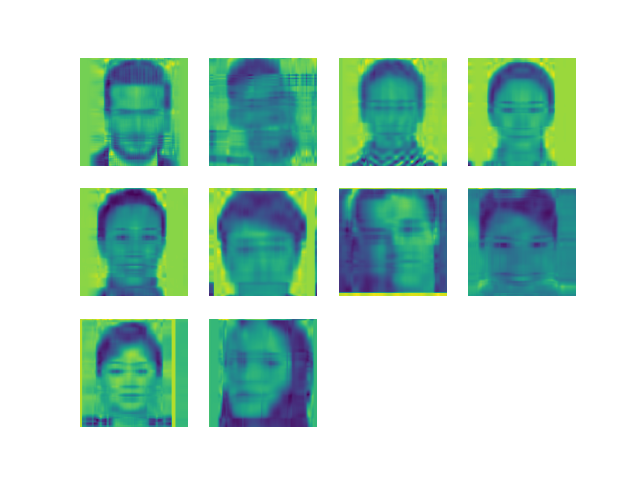
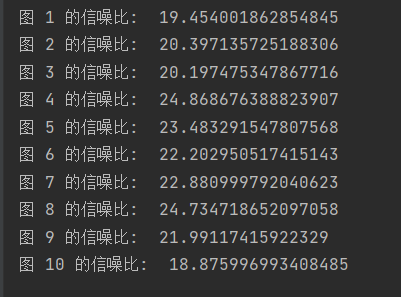
**50维**



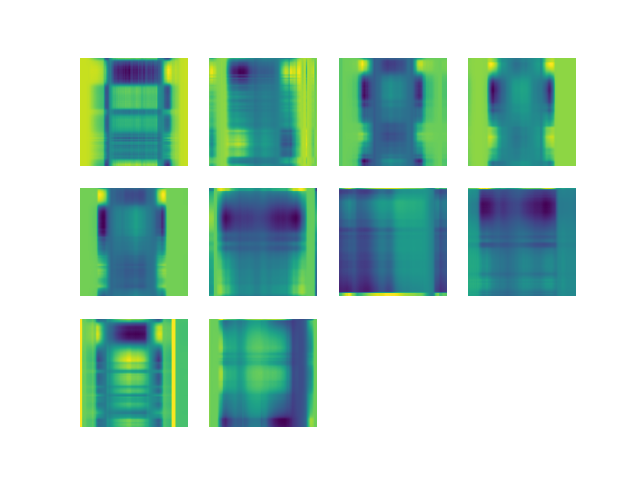
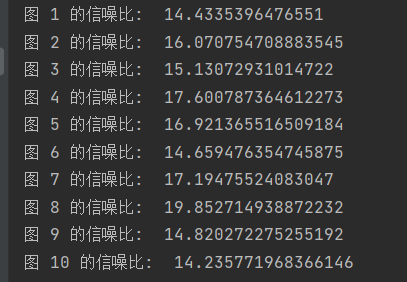
**10维**



**5维**



**1维**



十张图片原为250 \* 250像素大小。通过结果可以看出，随着维度的降低，信噪比也越来越低。 k=50和100 时，都能较好的保留图片特征，人眼几乎无法分辨损失，降到 k=10 时，图片有了较为明显的损失，k=5时，能勉强辨别出图片，但在降到 k=1 时，图片已完全模糊，失真。

这十张图片中，有相似也有差别较大的，可以看到，从降10维降到5维时，图一图二已经没有什么辨识度。而第8张图，直到降至3维，仍保留着主要特征，说明它和其他图的差别较大，从信噪比也可以看出。图8始终保持着极高的信噪比。在实验的过程中，还会遇到求解特征向量时，特征向量出现虚部的问题，但是发现只有在降维程度较高的时候（10维以上），特征向量矩阵的后面才会出现虚部。在10维以内的特征向量矩阵求解结果，全部都是实特征向量。

1. **结论**
2. PCA降低了训练数据的维度的同时保留了主要信息，但在训练集上保留下来的特征未必是模型真正所需特征，被舍弃掉的特征未必⽆⽤，只是在训练数据上没有表现，因此PCA有可能加重了过拟合。
3. PCA算法中舍弃了 d-k 个最⼩的特征值对应的特征向量，⼀定会导致低维空间与⾼维空间不同，但是通过这种⽅式有效提⾼了样本的采样密度；并且由于较⼩特征值对应的往往与噪声相关，通过PCA在⼀定程度上起到了降噪的效果。
4. PCA用于图片的降维可以极大地缓解存储压力，尤其是在如今像素越来越高的情况下。使用PCA降维我们只需要存储三个比较小的矩阵，能够较大地节省存储空间。
5. **参考文献**

[1] [周志华](http://cs.nju.edu.cn/zhouzh/) 著.机器学习, 北京: 清华大学出版社, 2016年1月

**七、附录：源代码（带注释）**

1. **import** DataMaker
3. **if** \_\_name\_\_ == '\_\_main\_\_':
4. size = 100
5. mu = [1, 2, 3]
6. sigma = [[0.1, 0, 0], [0, 10, 0], [0, 0, 10]]
8. train\_x=DataMaker.generate\_data(mu,sigma,size)
9. DataMaker.show\_3D(train\_x)
11. # mu = [2, 2]
12. # sigma = [[10, 0], [0, 0.2]]
13. # train\_x = DataMaker.generate\_data(mu, sigma, size)
14. # DataMaker.show\_2D(train\_x)
15. # , 10, 5, 3, 1
16. # k\_list = [100,50, 10, 5, 3, 1]
17. # DataMaker.face\_process('photo',k\_list)
18. **import** math
20. **import** numpy as np
21. **import** matplotlib.pyplot as plt
22. **from** mpl\_toolkits.mplot3d **import** Axes3D
23. **import** os
24. **from** PIL **import** Image
25. # size = (50, 50) # 由于较大的数据在求解特征值和特征向量时很慢，故统一压缩图像为size大小
26. **import** cv2
28. **def** pca(x,k):
29. n = x.shape[0]
30. mu = np.sum(x, axis=0) / n
31. x\_center = x - mu
32. cov = (x\_center.T @ x\_center) / n
34. values, vectors = np.linalg.eig(cov)
35. index = np.argsort(values)  # 从小到大排序后的下标序列
36. vectors = vectors[:, index[:-(k + 1):-1]].T  # 把序列逆向排列然后取前k个，转为行向量
37. vectors=np.real(vectors)#为了防止当维度过高时产生复数
38. **return** x\_center, mu, vectors

41. #生成高斯分布数据
43. **def** generate\_data(mu,sigma,size):
44. """
45. 生成高斯分布数据
46. @:param mu
47. @:param sigma
48. """
49. x = np.random.multivariate\_normal(mu, sigma, size)
50. **return** x
51. #二维降到一维
52. **def** show\_2D(x):
53. x\_center, mu, vectors = pca(x, 1)
55. x\_pca = x\_center @ vectors.T @ vectors + mu
56. plt.scatter(x\_pca[:, 0].tolist(), x\_pca[:, 1].tolist(), c=x\_pca[:, 0].tolist(), cmap=plt.cm.gnuplot)
58. plt.style.use('seaborn')
59. plt.scatter(x[:, 0], x[:, 1], c="b", label="Origin Data")
60. plt.scatter(x\_pca[:, 0], x\_pca[:, 1], c='r', label='PCA Data')
61. plt.plot(x\_pca[:, 0], x\_pca[:, 1], c='k', label='vector', alpha=0.5)
62. plt.legend(loc='best')
63. plt.ylim(np.min(x[:, 1]) - 1, np.max(x[:, 1]) + 1)
64. plt.show()
65. # plt.style.use('default')
66. #将三维降到二维
67. **def** show\_3D(x):
68. x\_center, mu, vectors = pca(x, 2)
69. x\_pca = x\_center @ vectors.T @ vectors + mu
70. # show
71. plt.style.use('seaborn')
72. fig = plt.figure()
73. ax = Axes3D(fig)
74. ax.scatter(x[:, 0], x[:, 1], x[:, 2], c="b", label='Origin Data')
75. ax.scatter(x\_pca[:, 0], x\_pca[:, 1], x\_pca[:, 2], c='r', label='PCA Data')
76. ax.plot\_trisurf(x\_pca[:, 0], x\_pca[:, 1], x\_pca[:, 2], color='k', alpha=0.3)
77. ax.legend(loc='best')
78. plt.show()
79. plt.style.use('default')
80. #读取指定目录下的照片
81. **def** face\_process(path,k\_list):
82. file\_list = os.listdir(path)
83. x\_list = []
84. **for** file **in** file\_list:
85. file\_path = os.path.join(path, file)
86. img = plt.imread(file\_path)
87. plt.imshow(img)
88. plt.axis('off')
89. # plt.show()
90. pic = Image.open(file\_path).convert('L')  # 读入图片，并将三通道转换为灰度图
92. x\_list.append(np.asarray(pic))
94. # n\_samples, n\_features = x\_list.shape
95. # print(data)
96. **for** k **in** k\_list:
98. x\_pca\_list = []
99. x\_psnr\_list = []
100. **for** x **in** x\_list:
101. x\_centerlized, mu, vectors = pca(x, k)  # PCA降维
102. x\_pca = x\_centerlized @ vectors.T @ vectors + mu  # 重建数据
103. x\_pca\_list.append(x\_pca)
104. x\_psnr\_list.append(psnr(x, x\_pca))
105. **print**(len(x\_pca\_list))
106. show\_pic(k, x\_pca\_list, x\_list, x\_psnr\_list)
107. #计算信噪比
108. **def** psnr(source, target):
109. """
110. 计算峰值信噪比
111. """
112. rmse = np.sqrt(np.mean((source - target) \*\* 2))
113. **return** 20 \* np.log10(255.0 / rmse)
114. **def** show\_pic(k, x\_pca\_list, x\_list, x\_psnr\_list):
115. #plt.figure(figsize=(8,5), frameon=False)
116. size = math.ceil((len(x\_list) + 1) / 2)
117. # print(size)
118. plt.subplot(2, size, 1)
119. plt.title('Real Image')
120. #print(x)
121. # plt.imshow(x)
122. # plt.axis('off')  # 去掉坐标轴
123. #print(x\_pca\_list[1])
124. # print(len(x\_list))
125. # print(len(x\_pca\_list))
126. **for** i **in** range(len(x\_list)):
127. plt.subplot(3,4,i+1)
128. # print(x\_pca\_list[i])
129. # plt.title('k = ' + str(x\_pca\_list[i]) + ', PSNR = ' + '{:.2f}'.format(x\_psnr\_list[i]))
130. # print(i)
131. plt.imshow(x\_pca\_list[i])
132. **print**('图', i+1, '的信噪比: ', x\_psnr\_list[i])
133. plt.axis('off')
134. plt.show()