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：1173710204

姓名： 陈东鑫

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）

二、实验要求及实验环境

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）。

实验环境：

操作系统：Windows10

语言：python3

编程环境：jupyter notebook

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即主成分分析（Principal Component Analysis），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。

从两个方向入手，进行PCA

特征值分解：

对数据D进行去平均值

求得D的协方差矩阵

将矩阵cov进行特征值分解：

其中，Q是矩阵A的特征向量组成的矩阵，则是一个对角阵，对角线上的元素就是特征值。将特征值由大到小排列的，这些特征值所对应的特征向量就是描述这个矩阵变换方向（从主要的变化到次要的变化排列）。

对于输入数据集

需要降到k维。

1) 去平均值(即去中心化)，即每一位特征减去各自的平均值。、

2) 计算协方差矩阵

3) 用特征值分解方法求协方差矩阵的特征值与特征向量。

4) 对特征值从大到小排序，选择其中最大的k个。然后将其对应的k个特征向量分别作为行向量组成特征向量矩阵P。

5) 将数据转换到k个特征向量构建的新空间中，即Y=PX

奇异值分解：

对数据D进行去平均值

求得D的协方差矩阵

将矩阵cov进行特征值分解：

U是一个m\*m的方阵，U里面的正交向量被称为左奇异向量。Σ是一个m\*n的矩阵，Σ除了对角线其它元素都为0，对角线上的元素称为奇异值。是v的转置矩阵，是一个n\*n的矩阵，它里面的正交向量被称为右奇异值向量。而且Σ上的值按从大到小的顺序排列。

对于输入数据集

需要降到k维。

1) 去平均值(即去中心化)，即每一位特征减去各自的平均值。

2) 计算协方差矩阵

3) 通过SVD计算协方差矩阵的特征值与特征向量。

4) 对特征值从大到小排序，选择其中最大的k个。然后将其对应的k个特征向量分别作为列向量组成特征向量矩阵。

5) 将数据转换到k个特征向量构建的新空间中。

2. 算法的实现

1）生成数据

使用np.random.multivariate\_normal(mean, cov, size)函数生成三维高斯分布数据集。mean代表均值，cov代表协方差矩阵，size代表样本点个数。训练集中，均值为[0, 0, 0]，协方差矩阵为[[1, 0, 0], [0, 10, 0], [0, 0, 10]]，样本点个数为500。

将样本点进行旋转，得到旋转后的样本点集

如下图所示：

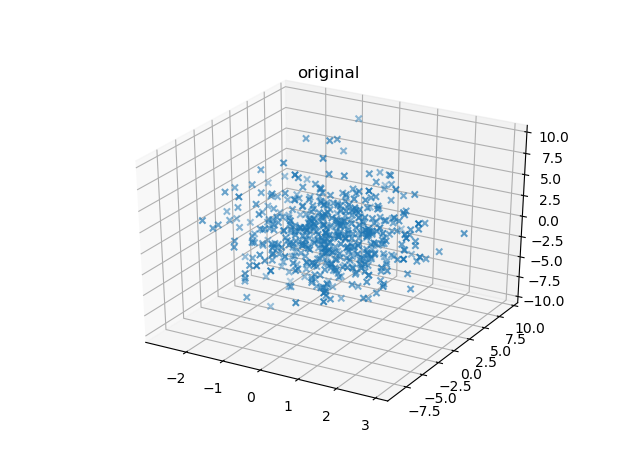


图 1初始数据集

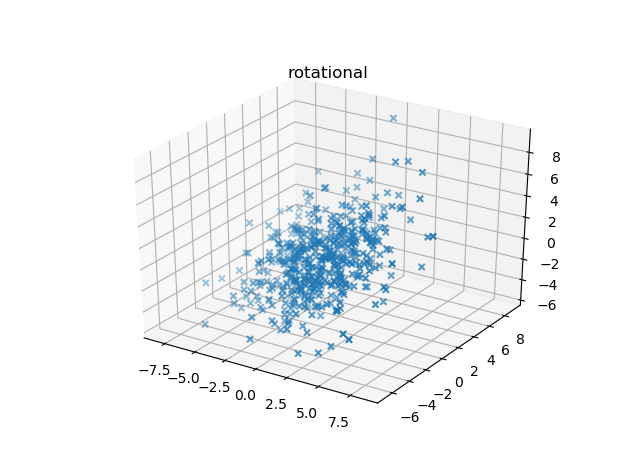


图 2旋转后数据集

生成数据所需函数的python代码如下

1. **def** genData(mean, cov, size, alpha, beta, gamma):
2. X = np.random.multivariate\_normal(mean, cov, size)
3. X = np.c\_[X, np.ones((X.shape[0], 1))]
5. alp = np.mat([[1, 0, 0, 0],
6. [0, math.cos(alpha), math.sin(alpha), 0],
7. [0, -math.sin(alpha), math.cos(alpha), 0],
8. [0, 0, 0, 1]])
9. bet = np.mat([[math.cos(beta), 0, -math.sin(beta), 0],
10. [0, 1, 0, 0],
11. [math.sin(beta), 0, math.cos(beta), 0],
12. [0, 0, 0, 1]])
13. gam = np.mat([[math.cos(gamma), math.sin(gamma), 0, 0],
14. [-math.sin(gamma), math.cos(gamma), 0, 0],
15. [0, 0, 1, 0],
16. [0, 0, 0, 1]])
18. alphaX = np.dot(X, alp)
19. betaX = np.dot(alphaX, bet)
20. gammaX = np.dot(betaX, gam)
21. data = np.mat(gammaX[:, 0:3]).T
22. ori = np.mat(X[:, 0:3]).T
23. **return** data, ori

生成数据的代码如下：

1. mean = [0, 0, 0]
2. cov = [[1, 0, 0], [0, 8, 0], [0, 0, 10]]
3. size = 500
4. k = 2
5. n = 3
6. alpha = 0.15 \* math.pi
7. beta = 0.25 \* math.pi
8. gamma = 0.35 \* math.pi
9. data, ori = genData(mean, cov, size, alpha, beta, gamma)

2）训练模型

由算法原理处的论述，算法实现所需的函数代码如下

1. **def** PCA(data, k):
2. data = data.astype('int64')
3. average = np.average(data,axis=1)
4. #     cov = np.cov(data)
5. #     data = data - average
6. cov = np.dot(data,data.T)/data.shape[0]
7. U, Sigma, VT = np.linalg.svd(cov)
8. #     print(cov)
9. #     print(U)
10. #     print(Sigma)
11. #     print(VT)
12. U = U[:, :k]
13. #     data = data + average
14. #     print(U.shape)
15. newData = np.dot(U.T, data)
16. #     newData = newData + average
17. oldData = np.dot(U, newData).T
18. #     oldData = oldData + average
19. **return** U, newData, oldData.T

四、实验结果与分析

实验结果如下

对于手动生成的数据集，进行PCA之后的输出如下：

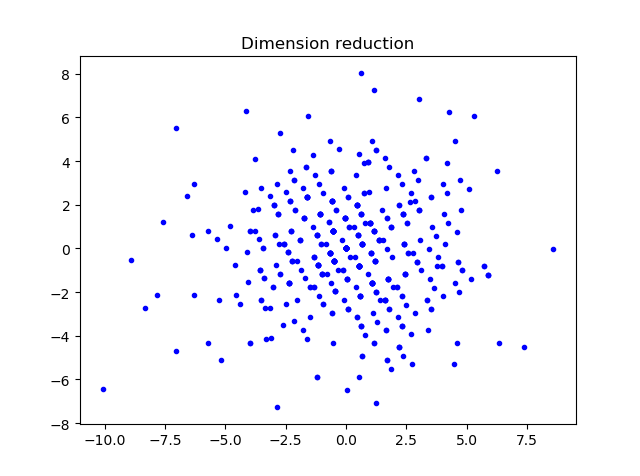


图 3 PCA之后在新的基底下的数据

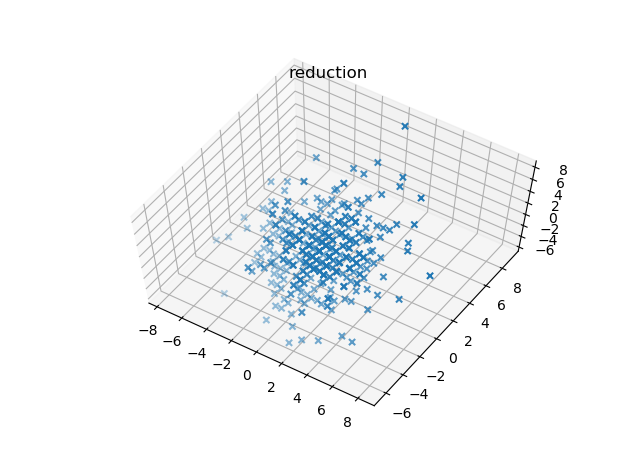


图 4 PCA之后在原始基底下的数据

对于图片的分析，此处以一张图片为例子，原图为256X256大小的图片

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 降维力度 | 原图（256） | 64 | 16 | 8 |
| 图片效果 | C:\Users\yi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\36.jpg |  |  |  |
| 信噪比 |  | 40.15653344499146 | 29.35381703262431 | 26.212077921915473 |

由表可见，保留的特征越多，图片越清晰，信噪比越高

五、结论

PCA可对数据提取主成分，进行降维处理。可用于图片压缩等方面。

六、附录：源代码（带注释）

1. **import** numpy as np
2. **import** math
3. **import** matplotlib.pyplot as plt
4. **from** mpl\_toolkits.mplot3d **import** Axes3D
5. **from** PIL **import** Image
7. **def** genData(mean, cov, size, alpha, beta, gamma):
8. X = np.random.multivariate\_normal(mean, cov, size)
9. X = np.c\_[X, np.ones((X.shape[0], 1))]
11. alp = np.mat([[1, 0, 0, 0],
12. [0, math.cos(alpha), math.sin(alpha), 0],
13. [0, -math.sin(alpha), math.cos(alpha), 0],
14. [0, 0, 0, 1]])
15. bet = np.mat([[math.cos(beta), 0, -math.sin(beta), 0],
16. [0, 1, 0, 0],
17. [math.sin(beta), 0, math.cos(beta), 0],
18. [0, 0, 0, 1]])
19. gam = np.mat([[math.cos(gamma), math.sin(gamma), 0, 0],
20. [-math.sin(gamma), math.cos(gamma), 0, 0],
21. [0, 0, 1, 0],
22. [0, 0, 0, 1]])
24. alphaX = np.dot(X, alp)
25. betaX = np.dot(alphaX, bet)
26. gammaX = np.dot(betaX, gam)
27. data = np.mat(gammaX[:, 0:3]).T
28. ori = np.mat(X[:, 0:3]).T
29. **return** data, ori
31. **def** PCA(data, k):
32. data = data.astype('int64')
33. average = np.average(data,axis=1)
34. #     cov = np.cov(data)
35. #     data = data - average
36. cov = np.dot(data,data.T)/data.shape[0]
37. U, Sigma, VT = np.linalg.svd(cov)
38. #     print(cov)
39. #     print(U)
40. #     print(Sigma)
41. #     print(VT)
42. U = U[:, :k]
43. #     data = data + average
44. #     print(U.shape)
45. newData = np.dot(U.T, data)
46. #     newData = newData + average
47. oldData = np.dot(U, newData).T
48. #     oldData = oldData + average
49. **return** U, newData, oldData.T

52. **def** PSNR(I, K):
53. m, n = I.shape
54. MAX = 255
55. MSE = np.average(np.power(I - K,2))
56. PSNR = 10 \* math.log10(MAX\*\*2 / MSE)
57. **return** PSNR
59. mean = [0, 0, 0]
60. cov = [[1, 0, 0], [0, 8, 0], [0, 0, 10]]
61. size = 500
62. k = 2
63. n = 3
64. alpha = 0.15 \* math.pi
65. beta = 0.25 \* math.pi
66. gamma = 0.35 \* math.pi
67. data, ori = genData(mean, cov, size, alpha, beta, gamma)
68. U, newData, oldData = PCA(data, k)
70. %matplotlib qt5
72. A = ori.T
73. B = data.T
74. C = newData
75. D = oldData
77. fig = plt.figure()
78. ax = plt.subplot(111, projection='3d')  # 创建一个三维的绘图工程
79. ax.scatter(A[:, 0].tolist(), A[:, 1].tolist(), A[:, 2].tolist(),
80. marker='x')  # 绘制数据点
81. plt.title('original')
82. plt.show()
84. fig = plt.figure()
85. ax = plt.subplot(111, projection='3d')  # 创建一个三维的绘图工程
86. ax.scatter(B[:, 0].tolist(), B[:, 1].tolist(), B[:, 2].tolist(),
87. marker='x')  # 绘制数据点
88. plt.title('rotational')
89. plt.show()
91. plt.figure()
92. plt.plot(C[0, :], C[1, :], '.', color='b')
93. plt.title('Dimension reduction')
94. plt.show()
96. fig = plt.figure()
97. ax = plt.subplot(111, projection='3d')  # 创建一个三维的绘图工程
98. ax.scatter(D[0, :].tolist(), D[1, :].tolist(), D[2, :].tolist(),
99. marker='x')  # 绘制数据点
100. plt.title('reduction')
101. plt.show()
103. k = 8
104. **for** i **in** range(60):
105. infile = "GSI/" + str(i) + ".jpg"
106. outfile = "generalization/" + str(i) + ".jpg"
107. inimg = Image.open(infile)
108. data = np.mat(inimg).astype('int64')
109. #     averagee = np.average(data)
110. #     data = data - average
111. U, newData, oldData = PCA(data, k)
112. #     pcaData = pcaData + average
113. outimg = Image.fromarray(oldData.astype('uint8'))
114. outimg.save(outfile)
115. **print**(i,"PSNR:",PSNR(data,oldData))
116. #     if oldData.any()<0 or oldData.any()>255:
117. #         print(False)
118. #     if oldData.any()<=1:
119. #         print(i)