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：主修

实验题目： PCA模型实验

学号：1161000309

姓名： 高靖龙

1. **实验目的**

对PCA算法进行实践，查考工程实现细节。

**二、实验要求及实验环境**

**实验要求：**

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），可以利用已有的矩阵特征向量提取方法。

**验证：**

（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）利用手写体数字数据mnist，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图像进行重建，比较一些它们与原图像有多大差别（可以用信噪比衡量）。

**实验环境：**

Manjaro linux

**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原理**

PCA（主成分分析）是 对于多维数据提取主要特征来提出冗余维度的降维过程。主要思想是对D维数据，找到一个M维子空间，使得D维样本到M维子空间投影点的留存信息量最大。可以从最大化方差、最小化误差两个不同的角度分析得出相同的结论。

**最大化方差**

最大化方差要求投影点之间方差最大，等价于对所有正交基，选取投影点在其上分量之间方差大小前M的M个向量作为主特征。。

考察某个单位正交基  
样本点在其上分量的方差为：

因此我们的目标函数是

使用拉格朗日乘子法形式为：

对上式求导：

代入得:因此从最大化方差考察，M个主成分即的特征值最大的前M个特征向量。

**最小化误差**最小化误差要求选取的M维子空间上D维样本的投影点与样本之间的距离最短。  
考察某个一组单位正交基

对M维子空间：

D维样本M维投影可表示为：  
D维样本的投影点与样本之间的距离:  
同理使用拉格朗日乘子法并求导：

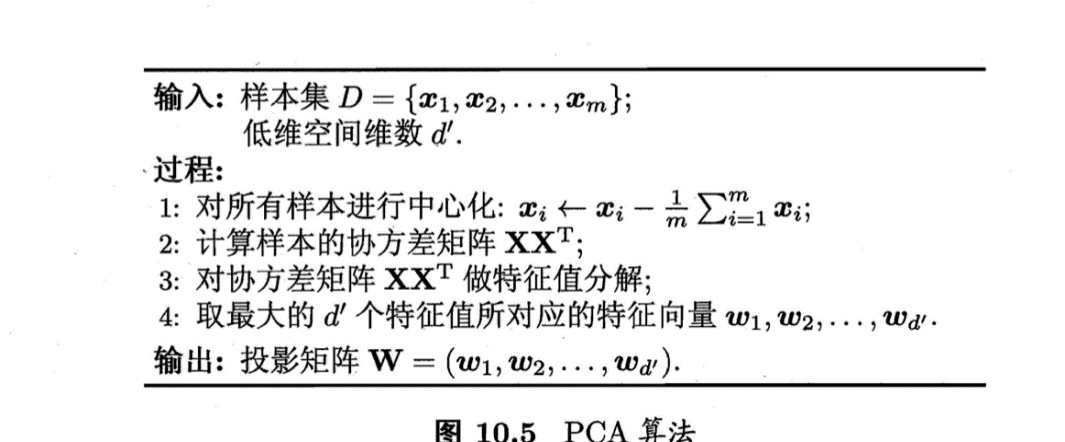
因此从最小化误差考察，投影空间应该是由的特征值最小的后D-M-1个特征向量定义，故主成分是的特征值最大的前M个特征向量。

**投影坐标与重建**

我们要得到在投影空间的坐标

我们还要解决如何将投影空间坐标还原到D维空间

通过投影空间坐标、投影向量、原样本均值我们就能将投影空间坐标映射回原空间。

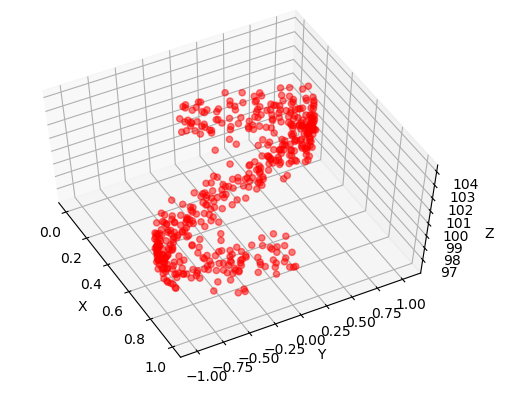
**算法的实现**

|  |
| --- |
| **def** pca(datamatrix, pcnum):  *"""  pca分析，输出低维坐标、由低维坐标还原的高维坐标、主向量* **:param** *datamatrix:* **:param** *pcnum: number of principal component* **:return***: lowD, newD,topdvects 低维坐标、由低维坐标还原的高维坐标、主向量  """* datamatrix = mat(datamatrix)  *# 中心化* meanmatrix = datamatrix.mean(axis=0)  remeaneddata = datamatrix - meanmatrix  *# 计算cov矩阵，不用除N，等价的，这样精度高* datacov = remeaneddata.T \* remeaneddata  ***# 特征分解,得到特征向量矩阵,eig\_vect是按列存储的* vals, vects = linalg.eig(datacov)  indexs = argsort(vals)  indexs = indexs[len(indexs) - pcnum:len(indexs)]  *# 列向量* topdvects = vects[:, indexs]**  *# 数据转到新空间* lowD = remeaneddata \* topdvects  newD = lowD \* topdvects.T + meanmatrix  **return** lowD.A, newD.A,topdvects.A |

实现中“*特征分解,得到特征向量矩阵*”这一步要求取出特征值最大的前M个特征向量，初版实现的很复杂，基本上就是按照循环来一个一个提取，之后对成熟代码进行参考，使用**argsort和numpu.array切片**进行了优化。另外在中心化和重建部分，由于GMM实验和该实验同步进行，GMM后来使用np.array进行优化，这里也顺便写了，原版是使用循环来对矩阵中的每个位置单独计算。这里提示我们应该熟悉所使用的库，对于矩阵操作尽量使用，不管在速度上还是代码整洁程度上都有帮助。

1. 实验结果与分析
2. **手工生成样本对PCA进行测试**

**样本生成**

这里使用之前实现过的正弦函数样本，并对原本的二维样本点增加一个符合一维高斯分布的维度并适当旋转。  


**针对新增维度不同的比重进行重建**

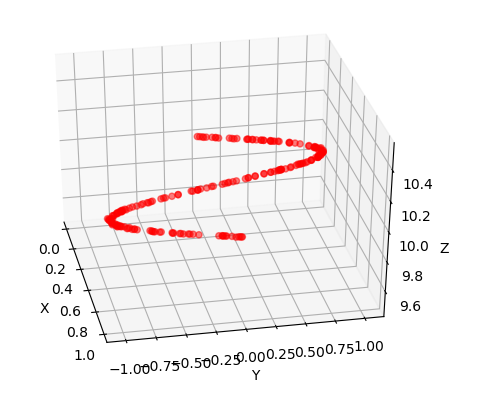
对3维数据进行2-PCA提取、重建。

调整新增维度高斯模型的方差，直观比较图像。

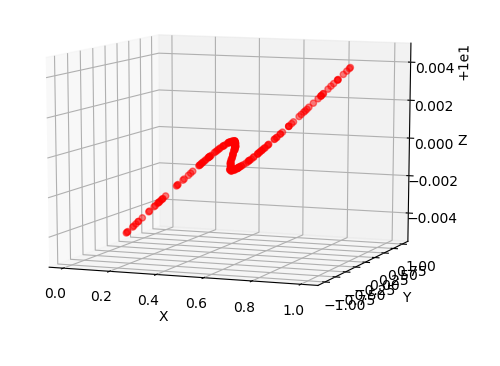
从图像可以看出随着var的增加，重建后的图像被扭曲的程度增大。当var足够大时，即新增维度权重更高时，可以发现另外两个维度退化成X-Y平面上的斜线。

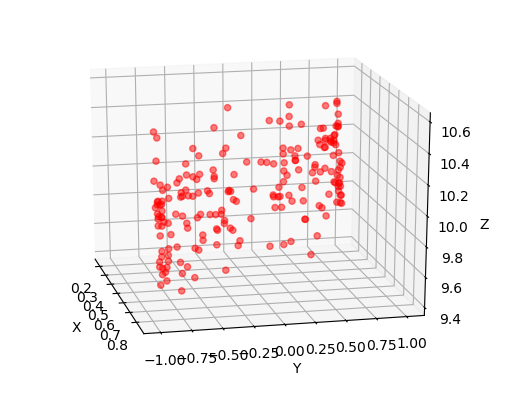
总体上PCA有效的实现了主成分提取，能够从重建图像中直观反映出原样本特征。

**Var=0**

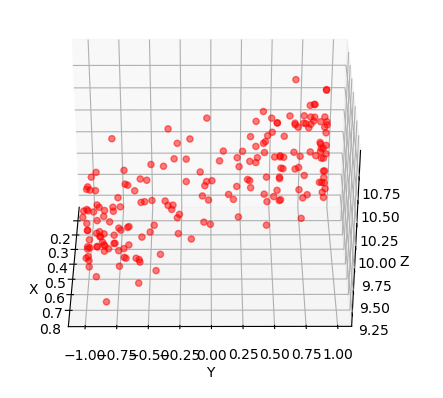


**Var=0.0125**

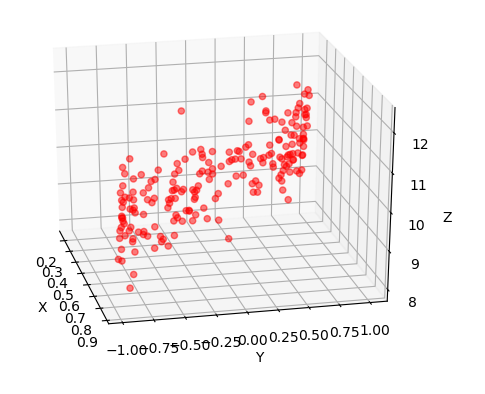


**Var=0.0.5**

**Var=0.1**



**Var=0.5（可以看到X-Y平面的正弦曲线退化成斜线）**



1. **对手写体数字数据mnist，进行降维和重建**

**手写体数据**28\*28的二维矩阵，每一个值都是0-255整数，表示该像素的灰度。

整体显示的是数字7的手写体图片。

**图像PCA与重建**调用实现的PCA函数对图像灰度矩阵进行主成分分析，每行28个像素当做一个28维样本，共28个样本，然后进行重建。

重建结果存储在本实验相应”data”文件夹中。以PCA提取的主成分个数进行命名。

**信噪比衡量**

下面给出不同PCA主成分个数下重建图像与原图像信噪比。

|  |  |  |  |
| --- | --- | --- | --- |
| **主成分数** | PSNR | **主成分数** | PSNR |
| **0** | 10.99018 |  |  |
| **1** | 13.0337 | **29** | 13.0337 |
| **2** | 15.73557 | **30** | 15.73557 |
| **3** | 17.89595 | **31** | 17.89595 |
| **4** | 19.88589 | **32** | 19.88589 |
| **5** | 22.57858 | **33** | 22.57858 |
| **6** | 24.89098 | **34** | 24.89098 |
| **7** | 27.33381 | **35** | 27.33381 |
| **8** | 30.29361 | **36** | 30.29361 |
| **9** | 32.30879 | **37** | 32.30879 |
| **10** | 34.45336 | **38** | 34.45336 |
| **11** | 36.25733 | **39** | 36.25733 |
| **12** | 38.30336 | **40** | 38.30336 |
| **13** | 41.89679 | **41** | 41.89679 |
| **14** | 44.24775 | **42** | 44.24775 |
| **15** | 47.8822 | **43** | 47.8822 |
| **16** | 51.70668 | **44** | 51.70668 |
| **17** | 59.93564 | **45** | 59.93564 |
| **18** | 74.1664 |  |  |
| **19** | 112.2991 |  |  |
| **20** | 299.2566 |  |  |
| **21** | 299.2632 |  |  |
| **22** | 299.2632 |  |  |
| **23** | 299.2632 |  |  |
| **24** | 299.2632 |  |  |
| **25** | 299.2634 |  |  |
| **26** | 299.2542 |  |  |
| **27** | 299.2543 |  |  |
| **28** | 299.2566 |  |  |

**分析**随着特征数提升信噪比慢慢增加，重建质量提升，当features=20，提升最大，之后变化不大。

1. **维度溢出**

当features=29时突降且和features=features-28效果相同。因为原数据维度28，最多只有28个维度可供进行主成分分析。当features>28，代码将其处理为features-28，这是编程实现产生的现象。

1. **性能激增**

再次强调在18-19-20处的性能激增，基本上是呈指数上涨，这提示我们，在实际使用中把features选取这种临近值达到边际收益最优。

**思考：在进行处理的时候，一定要按行分割成28个28维样本吗？**划分形式有很多，比如可以变成两行两行的，或者按列均可。但是效果有差别，譬如上述图例因为按行划分，所以可以看到重建的图片有行颜色分布的趋同性，模糊方向也是纵向的（划分的结果详见上交文件的data/mnistprocess文件夹，图片以pca维度数明命名）。

**思考：像素所处位置是否要标出呢？**

像素所处的位置加入到样本中，比如28个29维样本，新增一个维度是样本行所处的行序号。或者28\*28个3维样本，三个维度分别是列、行坐标、灰度，这样也可以进行PCA。但是由于这样的位置信息规律性强，因此基本上不会被PCA提取成主成分。意义不大。

五、结论

1. PCA主成分分析主要是将对数据影响大的特征向量提取出来对数据进行降维，有效对抗维度灾难。其原理是分别从最大化方差、最小化误差两个角度给出得到同样的结果。
2. 依据PCA主成分分析得到的特征向量以及原样本在新子空间中的坐标和在原空间中的均值能够对图像进行重建，重建的图像将丢失在PCA中抛弃的非主成分向量携带的信息。
3. PCA中主成分个数的选择中存在性能激增的现象，即对图像重建问题的性能，某个主成分个数远远高于较之小的主成分个数。我们在PCA实际使用中应该寻找这样的数以期获取最优边际收益。
4. 对于图像，选哪些数据当做一个样本效果不同。
5. 将样本的顺序或者说位置信息加入到样本中由于变化的规律性，通常不会被选为主成分。

六、参考文献

[1]NumPy for Matlab users — NumPy v1.15 Manual[EB/OL]. Docs.scipy.org, 2018. (2018)[2018 -11 -15]. https://docs.scipy.org/doc/numpy-1.15.1/user/numpy-for-matlab-users.html.

[2]ArrogantL/AiLearning[EB/OL]. GitHub, 2018. (2018)[2018 -11 -15]. https://github.com/ArrogantL/AiLearning.

[3]BISHOP C. PATTERN RECOGNITION AND MACHINE LEARNING[M]. [S.l.]: SPRINGER-VERLAG NEW YORK, 2016.

七、附录：源代码（带注释）

|  |
| --- |
| from numpy import \* |
|  | from numpy import linalg |
|  | import matplotlib.pyplot as plt |
|  | from mpl\_toolkits.mplot3d import Axes3D |
|  | from DataGenerator import generateData,generateThiDimensionalData |
|  |  |
|  |  |
|  | def pca(datamatrix, pcnum): |
|  | """ |
|  | pca分析，输出低维坐标、由低维坐标还原的高维坐标、主向量 |
|  | :param datamatrix: |
|  | :param pcnum: number of principal component |
|  | :return: lowD, newD,topdvects 低维坐标、由低维坐标还原的高维坐标、主向量 |
|  | """ |
|  | datamatrix = mat(datamatrix) |
|  | # 中心化 |
|  | meanmatrix = datamatrix.mean(axis=0) |
|  | remeaneddata = datamatrix - meanmatrix |
|  | # 计算cov矩阵 |
|  | datacov = remeaneddata.T \* remeaneddata |
|  | # 特征分解,得到特征向量矩阵,eig\_vect是按列存储的 |
|  | vals, vects = linalg.eig(datacov) |
|  | indexs = argsort(vals) |
|  | indexs = indexs[len(indexs) - pcnum:len(indexs)] |
|  | # 列向量 |
|  | topdvects = vects[:, indexs] |
|  | # 数据转到新空间 |
|  | lowD = remeaneddata \* topdvects |
|  | newD = lowD \* topdvects.T + meanmatrix |
|  | return lowD.A, newD.A,topdvects.A |
|  |  |
|  |  |
|  | if \_\_name\_\_ == '\_\_main\_\_': |
|  | # D = generateData(num=200) |
|  | D = generateThiDimensionalData(num=200) |
|  | lowD, newD,topdvects = pca(mat(D), 2) |
|  | A = newD[:,0] |
|  | B = newD[:,1] |
|  | C = newD[:, 2] |
|  | # plt.plot(A, B, 'r\*', linewidth=2) |
|  | # plt.show() |
|  | colors = ['blue', 'red', 'black', 'green'] |
|  | ax = plt.subplot(111, projection='3d') # 创建一个三维的绘图工程 |
|  | ax.scatter(newD[:, 0], newD[:, 1], newD[:, 2], s=75, c="blue", |
|  | alpha=0.5) # 绘制数据点 |
|  | ax.set\_zlabel('Z') # 坐标轴 |
|  | ax.set\_ylabel('Y') |
|  | ax.set\_xlabel('X') |
|  | plt.show() |

|  |
| --- |
| import struct |
|  |  |
|  | import matplotlib.pyplot as plt |
|  | import numpy as np |
|  | from numpy import \* |
|  |  |
|  | from PCA import pca |
|  |  |
|  |  |
|  | def readData(): |
|  | """ |
|  | 参考mnist官网以及github实现数据读取，读取结果是一个28\*28的list，取值为0-255整数 |
|  | :return: |
|  | """ |
|  | with open("data/train-images-idx3-ubyte", 'rb') as f1: |
|  | buf = f1.read() |
|  | temp = struct.unpack\_from('>784B', buf, struct.calcsize('>IIII')) |
|  | im = np.reshape(temp, (28, 28)) |
|  | return im |
|  |  |
|  |  |
|  | def mnistProcess(): |
|  | """ |
|  | 分析minist-5,目标图片是数字5的手写 |
|  | 将不同pcnum的结果保存到data/mnistprocess |
|  |  |
|  | 这里将每一行当作一个样本，即28个28维样本。 |
|  | 可选的其他样本划分方式： |
|  | 1. 加入行序号：由于行号规律强，PCA通常不会将其作为主成分，所以加不加影响不大。 |
|  | 2. 28\*28个三维样本，（行号，列号，灰度）：行列号对PCA影响不大，仅剩下一个维度无法PCA |
|  |  |
|  | """ |
|  | im = readData() |
|  | for i in range(50): |
|  | # i=17与i=18，人眼就非常难以分辨了。 |
|  | lowD, newD, topdvects= pca(im, i) |
|  | PSNR=analyzePSNR(im,newD) |
|  | print(i,PSNR) |
|  | plt.imshow(np.matrix.tolist(newD), cmap='gray') |
|  | plt.savefig("data/mnistprocess/" + str(i) + ".png") |
|  | def analyzePSNR(im,im2): |
|  | SUM=np.sum(np.power(im-im2,2)) |
|  | MSE = SUM / (im.shape[0]\*im.shape[1]) |
|  | PSNR = 10 \* math.log((255.0 \* 255.0 / (MSE)), 10) |
|  | return PSNR |
|  |  |
|  |  |
|  | if \_\_name\_\_ == '\_\_main\_\_': |
|  | mnistProcess() |

|  |
| --- |
| from math import sin, pi |
|  |  |
|  | import matplotlib.pyplot as plt |
|  | import numpy as np |
|  |  |
|  | from numpy.random.mtrand import multivariate\_normal |
|  |  |
|  |  |
|  | def generateData(num,mean=100,var=0): |
|  | X=np.random.rand(num) |
|  |  |
|  | S=np.random.randn(num) |
|  | tmp=[sin( 2\*pi \* m) for m in X] |
|  | T=[] |
|  | for i in range(num): |
|  | T.append(tmp[i]+S[i]/100) |
|  |  |
|  | assert len(X)==len(T) |
|  | R=[] |
|  | for i in range(len(X)): |
|  | R.append(multivariate\_normal([mean], [[var]])) |
|  | # R.append(0) |
|  | D=[] |
|  | for x,t,r in zip(X,T,R): |
|  | D.append((x,t,r)) |
|  | return D |
|  |  |
|  | def generateThiDimensionalData\ |
|  | (num,featurenum=3): |
|  | """ |
|  |  |
|  | :param vars: |
|  | :param means: |
|  | :param num:数据量 |
|  | :return: |
|  | """ |
|  | XX = [] |
|  |  |
|  | mean = [1, 0, 0] |
|  | vars = [[100, 0, 0], [0, 10, 0], [0, 0.1, 0]] |
|  | for i in range(num): |
|  | XX.append(multivariate\_normal(mean, vars)) |
|  |  |
|  |  |
|  |  |
|  |  |
|  | return XX |
|  | if \_\_name\_\_ == '\_\_main\_\_': |
|  | D, X, T, R = generateData(500) |
|  |  |
|  | plt.plot(X, T, 'r\*', linewidth=2) |
|  | plt.show() |