姓名：向建宇

学号：1801210687

1.用PCA对数据集进行降维：

**算法原理：**

降维是将高维度的数据保留下最重要的一些特征，去除噪声和不重要的特征，从而实现提升数据处理速度的目的。

PCA(principal Component Analysis)，即主成分分析方法，是一种使用最广泛的数据压缩算法。在PCA中，数据从原来的坐标系转换到新的坐标系，由数据本身决定。转换坐标系时，以方差最大的方向作为坐标轴方向，因为数据的最大方差给出了数据的最重要的信息。第一个新坐标轴选择的是原始数据中方差最大的方法，第二个新坐标轴选择的是与第一个新坐标轴正交且方差次大的方向。重复该过程，重复次数为原始数据的特征维数。

**思路分析：（具体步骤解释请看代码注释）**

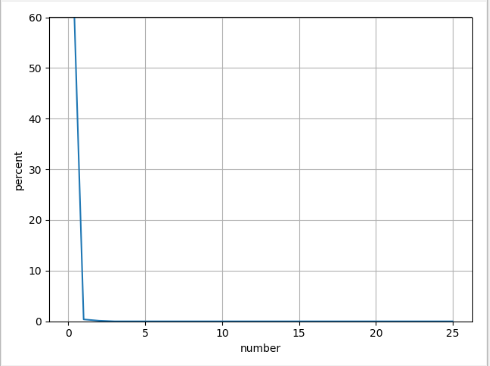
其实最主要的就是根据数学推导，通过计算数据矩阵的协方差矩阵，然后得到协方差矩阵的特征值及特征向量，选择特征值最大（也即包含方差最大）的N个特征所对应的特征向量组成的矩阵，我们就可以将数据矩阵转换到新的空间当中，实现数据特征的降维（N维）。

其主要步骤如下：

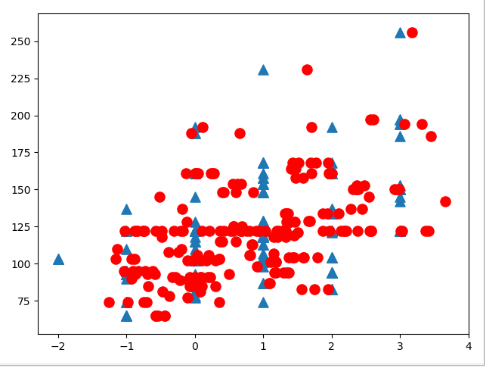
|  |
| --- |
| 去除平均值  计算协方差矩阵  计算协方差矩阵的特征值和特征向量  将特征值排序  保留前N个最大的特征值对应的特征向量  将数据转换到上面得到的N个特征向量构建的新空间中（实现了特征压缩） |

**结果截图：**

我根据特征值绘出了前15个主成分的占比：



可以看到几乎前三个就占了97%以上的百分比，所以后面的主成分的意义并不大了



上图是将原数据矩阵降维后重新重构，再取其前两维与原数据矩阵的前两维比较得到如图。

（三角形表示原数据点，圆形表示重构的数据点）

完整代码见附录。

附录：

|  |
| --- |
| **from** numpy **import** \* **import** matplotlib **import** matplotlib.pyplot **as** plt *#导入numpy库 #解析文本数据函数 #@filename 文件名txt #@delim 每一行不同特征数据之间的分隔方式，默认是tab键'\t'* **def** loadDataSet(filename,delim=**','**):*#打开文本文件* fr=open(filename)  *#对文本中每一行的特征分隔开来，存入列表中，作为列表的某一行  #行中的每一列对应各个分隔开的特征* stringArr=[line.strip().split(delim) **for** line **in** fr.readlines()]  **for** i **in** range(len(stringArr)):  **for** j **in** range(len(stringArr[0])):  **if** stringArr[i][j]==**'alfa-romero'**:  stringArr[i][j]=**'0'  elif** stringArr[i][j]==**'audi'**:  stringArr[i][j]=**'1'  elif** stringArr[i][j]==**'bmw'**:  stringArr[i][j]=**'2'  elif** stringArr[i][j]==**'chevrolet'**:  stringArr[i][j]=**'3'  elif** stringArr[i][j]==**'dodge'**:  stringArr[i][j]=**'4'  elif** stringArr[i][j]==**'honda'**:  stringArr[i][j]=**'5'  elif** stringArr[i][j]==**'isuzu'**:  stringArr[i][j]=**'6'  elif** stringArr[i][j]==**'jaguar'**:  stringArr[i][j]=**'7'  elif** stringArr[i][j]==**'mazda'**:  stringArr[i][j]=**'8'  elif** stringArr[i][j]==**'mercedes-benz'**:  stringArr[i][j]=**'9'  elif** stringArr[i][j]==**'mercury'**:  stringArr[i][j]=**'10'  elif** stringArr[i][j]==**'mitsubishi'**:  stringArr[i][j]=**'11'  elif** stringArr[i][j]==**'nissan'**:  stringArr[i][j]=**'12'  elif** stringArr[i][j]==**'peugot'**:  stringArr[i][j]=**'13'  elif** stringArr[i][j]==**'plymouth'**:  stringArr[i][j]=**'14'  elif** stringArr[i][j]==**'porsche'**:  stringArr[i][j]=**'15'  elif** stringArr[i][j]==**'renault'**:  stringArr[i][j]=**'16'  elif** stringArr[i][j]==**'saab'**:  stringArr[i][j]=**'17'  elif** stringArr[i][j]==**'subaru'**:  stringArr[i][j]=**'18'  elif** stringArr[i][j]==**'toyota'**:  stringArr[i][j]=**'19'  elif** stringArr[i][j]==**'volkswagen'**:  stringArr[i][j]=**'20'  elif** stringArr[i][j]==**'volvo'**:  stringArr[i][j]=**'21'  elif** stringArr[i][j]==**'diesel'**:  stringArr[i][j]=**'0'  elif** stringArr[i][j]==**'gas'**:  stringArr[i][j]=**'1'  elif** stringArr[i][j]==**'std'**:  stringArr[i][j]=**'0'  elif** stringArr[i][j]==**'turbo'**:  stringArr[i][j]=**'1'  elif** stringArr[i][j]==**'four'**:  stringArr[i][j]=**'4'  elif** stringArr[i][j]==**'two'**:  stringArr[i][j]=**'2'  elif** stringArr[i][j]==**'hardtop'**:  stringArr[i][j]=**'0'  elif** stringArr[i][j]==**'wagon'**:  stringArr[i][j]=**'1'  elif** stringArr[i][j]==**'sedan'**:  stringArr[i][j]=**'2'  elif** stringArr[i][j]==**'hatchback'**:  stringArr[i][j]=**'3'  elif** stringArr[i][j]==**'convertible'**:  stringArr[i][j]=**'4'  elif** stringArr[i][j]==**'4wd'**:  stringArr[i][j]=**'0'  elif** stringArr[i][j]==**'fwd'**:  stringArr[i][j]=**'1'  elif** stringArr[i][j]==**'rwd'**:  stringArr[i][j]=**'2'  elif** stringArr[i][j]==**'front'**:  stringArr[i][j]=**'0'  elif** stringArr[i][j]==**'rear'**:  stringArr[i][j]=**'1'  elif** stringArr[i][j]==**'dohc'**:  stringArr[i][j]=**'0'  elif** stringArr[i][j]==**'dohcv'**:  stringArr[i][j]=**'1'  elif** stringArr[i][j]==**'l'**:  stringArr[i][j]=**'2'  elif** stringArr[i][j]==**'ohc'**:  stringArr[i][j]=**'3'  elif** stringArr[i][j]==**'ohcf'**:  stringArr[i][j]=**'4'  elif** stringArr[i][j]==**'ohcv'**:  stringArr[i][j]=**'5'  elif** stringArr[i][j]==**'rotor'**:  stringArr[i][j]=**'6'  elif** stringArr[i][j]==**'eight'**:  stringArr[i][j]=**'8'  elif** stringArr[i][j]==**'five'**:  stringArr[i][j]=**'5'  elif** stringArr[i][j]==**'six'**:  stringArr[i][j]=**'6'  elif** stringArr[i][j]==**'three'**:  stringArr[i][j]=**'3'  elif** stringArr[i][j]==**'twelve'**:  stringArr[i][j]=**'12'  elif** stringArr[i][j]==**'1bbl'**:  stringArr[i][j]=**'0'  elif** stringArr[i][j]==**'2bbl'**:  stringArr[i][j]=**'1'  elif** stringArr[i][j]==**'4bbl'**:  stringArr[i][j]=**'2'  elif** stringArr[i][j]==**'idi'**:  stringArr[i][j]=**'3'  elif** stringArr[i][j]==**'mfi'**:  stringArr[i][j]=**'4'  elif** stringArr[i][j]==**'mpfi'**:  stringArr[i][j]=**'5'  elif** stringArr[i][j]==**'spdi'**:  stringArr[i][j]=**'6'  elif** stringArr[i][j]==**'spfi'**:  stringArr[i][j]=**'7'  elif** stringArr[i][j]==**'?'**:  stringArr[i][j]=**'NaN'** *#利用map()函数，将列表中每一行的数据值映射为float型* datArr = [list(map(float,line)) **for** line **in** stringArr]  **return** mat(datArr) *#pca特征维度压缩函数 #@dataMat 数据集矩阵 #@topNfeat 需要保留的特征维度，即要压缩成的维度数，默认4096* **def** pca(dataMat, topNfeat=15):  meanVals = mean(dataMat, axis=0)  meanRemoved = dataMat - meanVals *#去除平均值* covMat = cov(meanRemoved, rowvar=0)  eigVals,eigVects = linalg.eig(mat(covMat))*#计算协方差矩阵的特征值及对应的特征向量  #均保存在相应的矩阵中* eigValInd = argsort(eigVals) *#从小到大排序对N个值* eigValInd = eigValInd[:-(topNfeat+1):-1] *#cut off unwanted dimensions消除不要的维度* redEigVects = eigVects[:,eigValInd] *#reorganize eig vects largest to smallest重组* lowDDataMat = meanRemoved \* redEigVects*#transform data into new dimensions数据转到新的维度空间* reconMat = (lowDDataMat \* redEigVects.T) + meanVals  **return** lowDDataMat, reconMat  *#缺失值处理函数* **def** replaceNanWithMean():  datMat = loadDataSet(**'imports-85.data'**, **','**)  numFeat = shape(datMat)[1]  **for** i **in** range(numFeat):  meanVal = mean(datMat[nonzero(~isnan(datMat[:,i].A))[0],i]) *#values that are not NaN (a number)* datMat[nonzero(isnan(datMat[:,i].A))[0],i] = meanVal *#set NaN values to mean* **return** datMat  *#绘制主成分占比* **def** show\_contenpt(datMat):  meanVals = mean(datMat, axis=0)  meanRemoved = datMat - meanVals *# 去除平均值* covMat = cov(meanRemoved, rowvar=0)  eigVals, eigVects = linalg.eig(mat(covMat))  y = 100 \* (eigVals / sum(eigVals))  plt.plot(y)  plt.grid(**True**) *##增加格点* plt.ylim(0, 60)  plt.xlabel(**'number'**)  plt.ylabel(**"percent"**)  plt.show()  *#前两列在重构后与原数据的变化比较* **def** show\_convert(dataMat,reconMat):  fig = plt.figure()  ax = fig.add\_subplot(111)  *# 三角形表示原始数据点* ax.scatter(dataMat[:, 0].flatten().A[0], dataMat[:, 1].flatten().A[0], \  marker=**'^'**, s=90)  *# 圆形点表示第一主成分点，点颜色为红色* ax.scatter(reconMat[:, 0].flatten().A[0], reconMat[:, 1].flatten().A[0]\  , marker = **'o'**, s = 90, c = **'red'**)  fig.show()  **if** \_\_name\_\_ == **'\_\_main\_\_'**:  datMat=replaceNanWithMean()  show\_contenpt(datMat)  lowDDataMat, reconMat = pca(datMat)  show\_convert(datMat,reconMat) |