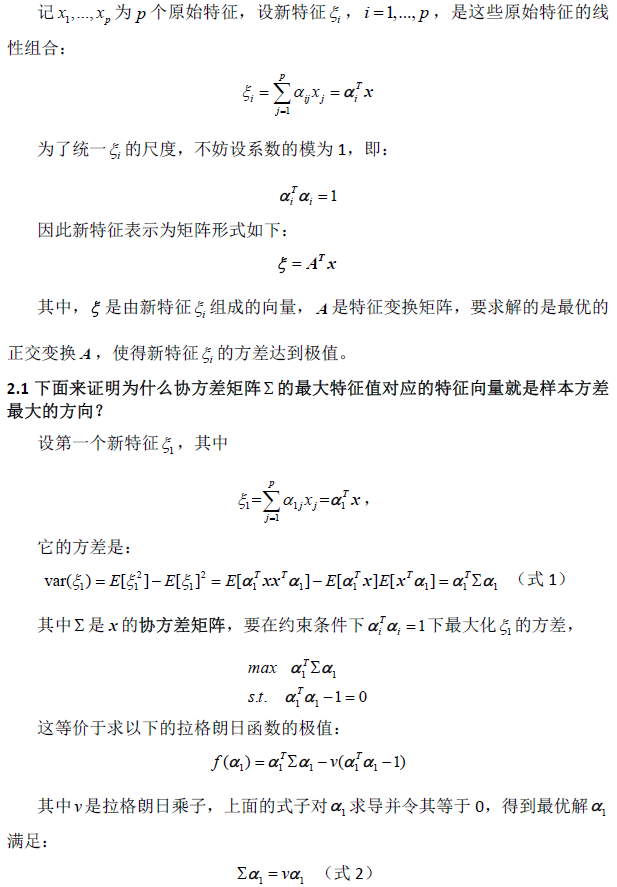
**实验八 主成分分析降维**

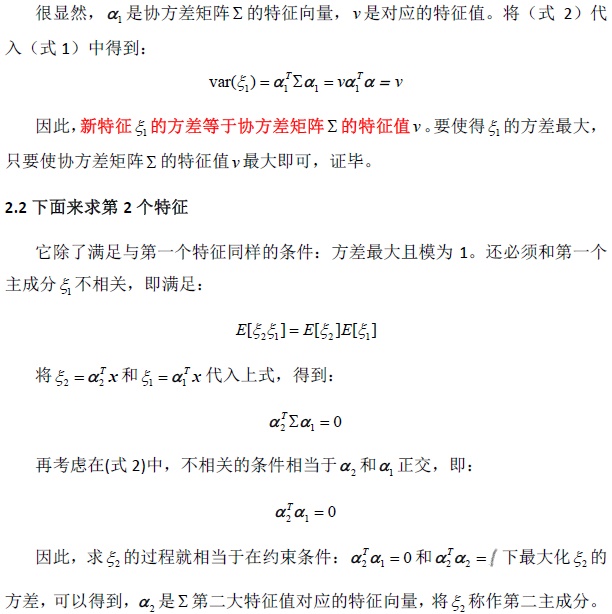
主成分分析(Principal Component Analysis, PCA)是一种目前常见的数据降维方法，其目的是在“信息”损失较小的前提下，将高维的数据转换到低维。在低维的情况下，数据更容易进行处理，从而减小计算量，其相关特征可能在数据中明确地显示出来。

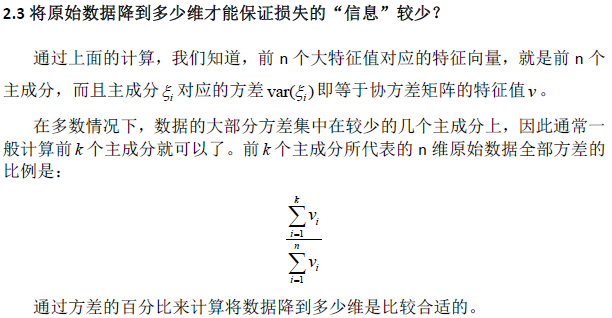
在信号处理中认为，信号具有最大的方差，而噪声有较小的方差，信噪比就是信号与噪声的方差比，越大越好。在PCA中，数据从原来的坐标系转换到新的坐标系，新坐标系的选择是由数据本身决定的。第一个新坐标轴选择的是原始数据中方差最大的方向，第二个新坐标轴的选择和第一个坐标轴正交且具有最大方差的方向。该过程一直重复，重复次数为原始数据中特征的数目。我们会发现，大部分方差都包含在最前面的几个新坐标轴中，因此可以忽略余下的坐标轴，即对数据进行了降维处理。

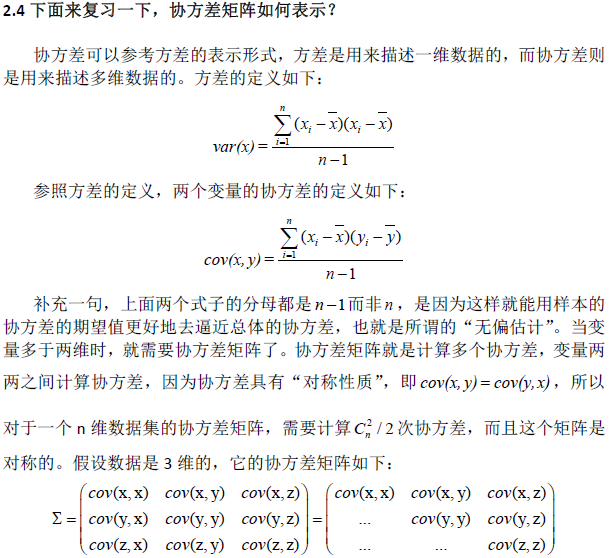
要实现PCA算法，需要知道：**原数据协方差矩阵Σ的最大特征值对应的特征向量就是方差最大的方向。**

大家可以看一下下面这几张图：









另外大家可以参考一下[这篇博客](http://www.cnblogs.com/jerrylead/archive/2011/04/18/2020209.html)。

PCA的伪代码大致如下：

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

根据特征值确定主成分的个数k

对特征值排序，获取最大的k个特征值及其对应的特征向量

将数据转换到由这些特征向量构建的新空间中

**实验要求：**

通过Numpy实现PCA。

**实验步骤：**

新建pca.py文件，导入numpy库及画图库:

*import numpy as np*

*import matplotlib.pyplot as plt*

①编写数据读取函数。

*def loadDataSet(filename):*

*fr = open(fileName)*

*stringArr = [line.strip().split(‘\t’) for line in\*

*fr.readlines()]*

*dataArr = [list(map(float,line)) for line in stringArr]*

*fr.close()*

*return np.mat(dataArr)*

②编写函数，根据特征值和比例确定主成分个数。

*def getNumOfFeat(eigVals,percentage):*

# 使用Numpy中的sort函数对特征值进行排序

# sort()是升序排序，需要倒过来

# 可以使用python的切片完成

# 【提示】a[-1::-1]即为a的倒序

# 将排好序的特征值保存到变量sortArray中

**#【代码待补全】**

# 使用Numpy中的sum函数对特征值求和

# 保存到变量arraySum中

**#【代码待补全】**

*tempSum = 0* #用以保存主成分的方差之和

*numOfFeat = 0* #用以保存主成分个数

*for i in sortArray:*

*tempSum += i*

*numOfFeat += 1*

*if tempSum >= arraySum\*percentage:*

*return numOfFeat*

③编写pca函数。

*def pca(dataMat, percentage=0.9):*

# 按列计算均值，保存到meanVals

# 使用numpy的mean函数

# 注意参数axis的取值

**#【代码待补全】**

# 去平均值

*meanRemoved = dataMat – meanVals*

# 计算协方差矩阵

# rowvar=0表示数据集中一行代表一条数据

*covMat = np.cov(meanRemoved, rowvar=0)*

# 计算协方差矩阵的特征值和特征向量

# 使用numpy中linalg模块的eig()函数

# 该函数有两个返回值，第一个是特征值，第二个是特征向量

# 注意先用numpy的mat函数变换一下协方差矩阵

# 要求特征值用变量eigVals，特征向量用变量eigVects

**#【代码待补全】**

# 调用上一步的函数获取主成分个数，保存到k

**#【代码待补全】**

# 对特征值排序，获取最大的k个特征值的下标

*eigValInd = np.argsort(eigVals)*

*eigValInd = eigValInd[:-(k+1):-1]*

# 根据k个特征值的下标获取对应的特征向量

*redEigVects = eigVects[:,eigValInd]*

# 根据k个特征向量将取均值化的数据转换到新空间

# 直接相乘就好，保存到lowDDataMat

#**【代码待补全】**

# 根据降维后的数据重构回原来的数据空间中

*reconMat = (lowDDataMat\*redEigVects.T)+meanVals*

*return lowDDataMat, reconMat*

④测试函数。

*def test(percentage=0.9):*

*dataMat = loadDataSet('testSet.txt')*

*lowDMat, reconMat = pca(dataMat, percentage)*

*fig = plt.figure()*

*ax = fig.add\_subplot(111)*

*ax.scatter(dataMat[:,0].flatten().A[0],\*

*dataMat[:,1].flatten().A[0],marker='^',s=90)*

*ax.scatter(reconMat[:,0].flatten().A[0],\*

*reconMat[:,1].flatten().A[0],marker='o',s=50,c='red')*

*plt.show()*

*print('Low dimension shape is',np.shape(lowDMat),\*

*'original shape is',np.shape(reconMat))*

直接在python命令行中使用如下代码查看结果：

*import pca*

*pca.test(0.5)*

其中test中的参数可以随便设置成(0,1)中的数，大家可以试试设成不同的值结果有什么不一样。

**\*\*\*拓展延伸\*\*\***

大家可以参照pca()函数的步骤，写一个接收参数k作为主成分个数的版本。