# 人工智能实验：Topic 4——降维技术

## Part 1: 主成分分析

**3190102060　黄嘉欣**

### 一、PCA 概述

主成分分析是迄今为止最流行的降维方法，其通过识别靠近数据的超平面，然后将数据投影到其上。一般而言，PCA 中的第一个主成分就是从数据差异性最大（即方差最大）的方向提取出来的，第二个主成分则来自于数据差异性次大的方向，并且该方向与第一个主成分方向正交。通过数据集的协方差矩阵及其特征值分析，我们就可以求得这些主成分的值。一旦得到了协方差矩阵的特征向量，我们就可以保留其中最大的 N 个值，这些特征向量也给出了 N 个最重要特征的真实结构。通过将数据乘上这 N 个特征向量，我们可以将它转换到新的空间。总的来说，PCA 的实现主要需要以下几个步骤：① 去除平均值；② 计算协方差矩阵；③ 计算协方差矩阵的特征值和特征向量；④ 将特征值从大到小排序；⑤ 保留最上面的 N 个特征向量；⑥ 将数据转换到上述 N 个特征向量构建的新空间中。

### 二、实验 4-1

① 实验题目

利用 PCA 函数，对“testSet.txt”中的数据进行降维分析：

(1) 可视化 topNfeat 分别等于 1、2 时，PCA 的输出数据；

(2) 通过与原始数据对比，讨论 topNfeat 分别等于 1、2 时的降维效果；

② 实验结果与分析

如图 2.1，由于原始数据为 2 维，当 topNfeat=1 时，将会舍弃掉一个维度，得到的降维数据维度为 1 维（为了可视化展示，在一维 x 的基础上添加了第二维 y=0）；重构得到的数据与原始数据相比差异明显，其为通过原始数据中心的一条直线，降维效果明显；当 topNfeat=2 时，此时将保留两个维度的特征向量，因此不会产生维度的降低，重构得到的数据应该与原始数据完全相同，如图 2.2 所示。

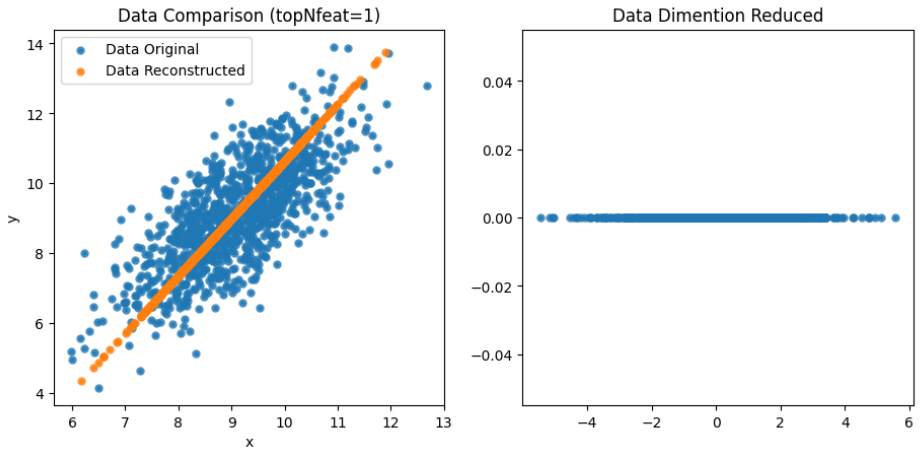


图 2.1 topNfeat=1 时降维效果

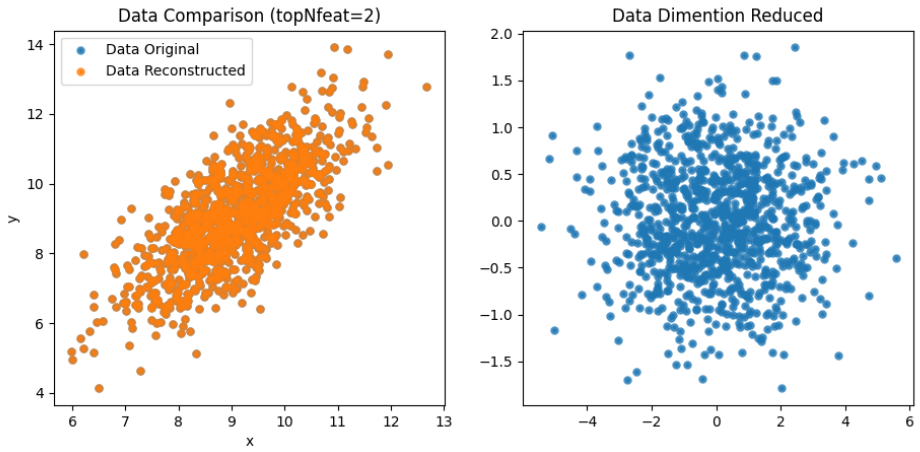


图 2.2 topNfeat=2 时降维效果

## 三、实验 4-2

① 实验题目

利用 PCA，对“secom.data”中的数据进行降维，原始数据拥有 590 个特征，且包含很多的缺失值 NaN，需要将其替换为平均值。

(1) 讨论 topNfeat 的取值对降维数据的影响；

(2) 找到降维后恢复的数据与原始数据相对误差小于 9%的 topNfeat；

② 实验结果与分析

如图 3.1，使 topNfeat 从 0 逐渐增大，得到降维后重构得到的数据与原始数据之间的相对误差变化的平滑曲线。可以发现，随着 topNfeat 不断变大，两者之间的相对误差也在不断的减小。这是因为当 topNfeat 越大时，最上面的特征向量保存的个数越多，维

度减少越小，重构后数据的准确度也就越高，与理论相吻合。

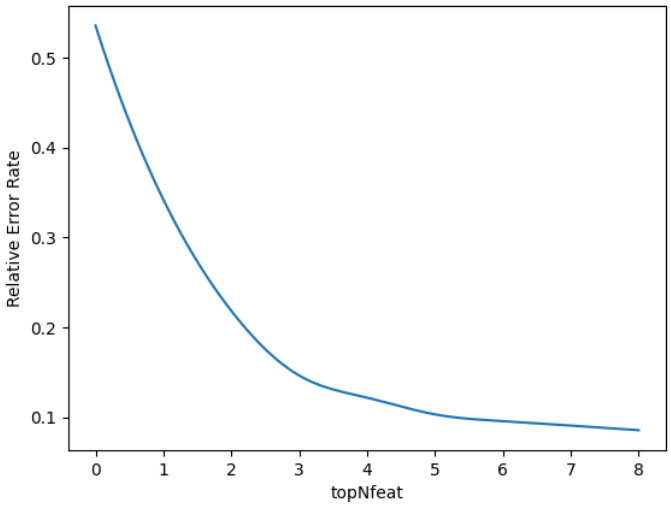


图 3.1 重构数据与原始数据的相对误差

如下，为 topNfeat 从 0 增大到 8 时的相对误差输出。当 topNfeat=8 时，降维后恢复的数据与原始数据之间的相对误差小于 9%。

```
topNfeat: 0, Re_err: 0.535965
topNfeat: 1, Re_err: 0.342120
topNfeat: 2, Re_err: 0.218514
topNfeat: 3, Re_err: 0.146507
topNfeat: 4, Re_err: 0.121884
topNfeat: 5, Re_err: 0.103267
topNfeat: 6, Re_err: 0.095779
topNfeat: 7, Re_err: 0.090812
topNfeat: 8, Re_err: 0.085691
The topNfeat is 8 when the relative error rate is 0.085691, which is less than 0.09.
```

**四、附录**：实验 Python 代码——*pca.py*

```python
from numpy import *
import matplotlib.pyplot as plt
from scipy.interpolate import make_interp_spline

def getDataSet(fileName, type):
    """从文件中读取数据集
    Args:
      fileName: 文件路径
      type: 文件类型, 0: txt, 1: data
    Returns:
      dataMat: 数据矩阵
    """
    with open(fileName,"r") as f:
        flist = f.readlines()           # 以 list 的形式返回
```

```
        flines = len(flist)             # 读取文件的行数
        if type is 0:
            dataMat = zeros((flines,2))  # flines*2 的矩阵
            # print(flist[0].strip())
            for i in range(0,flines):
                line = flist[i].strip()  # 读取每一行的内容(去除头尾空白符)
                lineData = line.split("\t")  # 按 TAB 分割每行内容
                dataMat[i,:] = lineData[0:2] # 样本矩阵的第 i 行
        else:
            dataMat = zeros((flines,590))    # flines*590 的矩阵
            # print(flist[0].strip())
            for i in range(0,flines):
                line = flist[i].strip()  # 读取每一行的内容(去除头尾空白符)
                lineData = line.split(" ")     # 按空格分割每行内容
                dataMat[i,:] = lineData[0:590] # 样本矩阵的第 i 行
    return dataMat

def replaceNaNWithMean(dataMat):
    """将 NaN 替换为均值
    Args:
      dataMat: 原始数据
      reconsMat: 重构后的数据
    Returns:
    """
    noNaNDataMat = dataMat
    numFeat = noNaNDataMat.shape[1]         # 特征数
    for i in range(numFeat):
    # 求该行的均值
        meanVal = mean(dataMat[nonzero(~isnan(dataMat[:,i]))[0],i])
        # 将 NaN 替换为均值
        noNaNDataMat[nonzero(isnan(noNaNDataMat[:,i]))[0],i] = meanVal
    return noNaNDataMat

def show(dataMat, lowDDataMat, reconsMat, topNfeat):
    """可视化
    Args:
      dataMat: 原始数据
      lowDDataMat: 降维后的数据
      reconsMat: 重构后的数据
      topNfeat: 用于区分图像
    Returns:
    """
    if topNfeat is 1:
        plt.figure(1)
```

```
    else:
        plt.figure(2)
    plt.subplot(1,2,1)
    if topNfeat is 1:
        plt.title("Data Comparison (topNfeat=1)")
    else:
        plt.title("Data Comparison (topNfeat=2)")
    l1 = plt.scatter(dataMat[:,0].tolist(),dataMat[:,1].tolist(),
                     marker='.',cmap='Blues',alpha=0.8,
                     linewidths=3)      # 原始数据
    l2 = plt.scatter(reconsMat[:,0].tolist(),reconsMat[:,1].tolist(),
                     marker='.',cmap='Blues',alpha=0.8,
                     linewidths=3)      # 重构后的数据
    plt.legend(handles=[l1,l2],labels=["Data Original",
                "Data Reconstructed"],loc='best')
    plt.xlabel("x")
    plt.ylabel("y")
    plt.subplot(1,2,2)
    if topNfeat is 1:
        y = zeros((lowDDataMat.shape[0],1))
        plt.scatter(lowDDataMat[:,0].tolist(),y,marker='.',cmap='Blues',
                     alpha=0.8,linewidths=3)     # 降维后的数据
    else:
        plt.scatter(lowDDataMat[:,0].tolist(),lowDDataMat[:,1].tolist(),
                     marker='.',cmap='Blues',alpha=0.8,
                     linewidths=3)                # 降维后的数据
    plt.title("Data Dimention Reduced")
    plt.draw()

def pca(dataMat, topNfeat=99999):
    """主成分分析
    Args:
      dataMat: 需要处理的数据矩阵
      topNfeat: 需要保留的特征向量个数
    Returns:
      lowDDataMat: 降维后的数据矩阵
      reconsMat: 重构后的数据矩阵
    """
    dataMean = mean(dataMat, axis=0)             # 求各个特征的均值
    new_dataMat = dataMat - dataMean             # 零均值化
    # 协方差矩阵，new_dataMat 每一行代替一个样本
    dataCov = cov(new_dataMat, rowvar=0)
    # 特征值与特征矩阵，eigenVecs 每一列代表一个特征向量
    eigenVals, eigenVecs = linalg.eig(mat(dataCov))
```

```
    # 对特征值从小到大排序
    sortedEigenVals = argsort(eigenVals)
    # 保留前 topNfeat 个特征值
    remEigenVals = sortedEigenVals[:-(topNfeat+1):-1]
    # 前 topNfeat 个特征值对应的特征向量
    remEigenVecs = eigenVecs[:,remEigenVals]
    lowDDataMat = new_dataMat * remEigenVecs                # 降维后的数据
    reconsMat = (lowDDataMat * remEigenVecs.T) + dataMean   # 重构数据
    return lowDDataMat, reconsMat

def main():
    # lab 4-1
    dataMat = getDataSet("testSet.txt",0)        # 获取数据矩阵
    lowDDataMat, reconsMat = pca(dataMat,1)      # 主成分分析, topNfeat=1
    show(dataMat, lowDDataMat, reconsMat, 1)     # 可视化
    lowDDataMat, reconsMat = pca(dataMat,2)      # 主成分分析, topNfeat=2
    show(dataMat, lowDDataMat, reconsMat, 2)     # 可视化

    # lab 4-2
    dataMat = getDataSet("secom.data",1)         # 获取数据矩阵
    noNaNDataMat = replaceNaNWithMean(dataMat)
    topNfeats = []                               # 可视化
    Re_errs = []
    for i in range(590):
        lowDDataMat, reconsMat = pca(noNaNDataMat,i)   # 主成分分析
        Re_err = linalg.norm(noNaNDataMat-
reconsMat)/linalg.norm(noNaNDataMat)             # 相对误差
        topNfeats.append(i)
        Re_errs.append(Re_err)
        print("topNfeat: %d, Re_err: %f" % (i,Re_err))
        if Re_err < 0.09:
            break
    print("The topNfeat is %d when the relative error rate is %f, which
is less than 0.09." % (i,Re_err))
    model = make_interp_spline(topNfeats, Re_errs)    # 绘制平滑曲线
    xs = linspace(0,i,500)
    ys = model(xs)
    plt.figure(3)                                      # 可视化
    plt.plot(xs, ys)
    plt.xlabel("topNfeat")
    plt.ylabel("Relative Error Rate")
    plt.show()

if __name__ == '__main__':
```

```
    main()
```