浙江大学

本科实验报告

Principal Component Analysis

课程名称：	人工智能实验
姓　　名：	姚桂涛
学　　院：	信息与电子工程学院
专　　业：	信息工程
学　　号：	**3190105597**
指导老师：	胡浩基、魏准

2022 年 3 月 25 日

一、 实验题目

1. 实验 4-1

利用 PCA 函数，对 testSet.txt 中数据做降维分析：（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 4-2

利用 PCA，对 secom.data 数据降维，（1）讨论 topNfeat 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

二、 实验代码

1. pca.py

pca.py

```python
from numpy import *
'''
老师，这个函数的map函数是用的python2标准，python3无法正常使用。
我在这个文章找到了解决方法。
https://blog.csdn.net/qq_35056459/article/details/108277208
'''
def loadDataSet(fileName, delim='\t'):
    fr = open(fileName)
    stringArr = [line.strip().split(delim) for line in fr.readlines()]
    datArr=[]
    for line in stringArr:
        data=[]
        for j in line:
            data.append(float(j))
        datArr.append(data)
    return mat(datArr)


def pca(dataMat, topNfeat=9999999):
    meanVals = mean(dataMat, axis=0)
    meanRemoved = dataMat - meanVals        #remove mean
    covMat = cov(meanRemoved, rowvar=0)
    eigVals,eigVects = linalg.eig(mat(covMat))
    eigValInd = argsort(eigVals)            #sort, sort goes smallest to largest
    eigValInd = eigValInd[:-(topNfeat+1):-1] #cut off unwanted dimensions
    redEigVects = eigVects[:,eigValInd]     #reorganize eig vects largest to smallest
    lowDDataMat = meanRemoved * redEigVects #transform data into new dimensions
    reconMat = (lowDDataMat * redEigVects.T) + meanVals
    return lowDDataMat, reconMat

def replaceNanWithMean():
```

```
    datMat = loadDataSet('secom.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
```

2. 实验 4-1

实验 4-1

```
from numpy import *
import pca
import matplotlib.pyplot as plt

dataMat = pca.loadDataSet('testSet.txt')
lowDMat, reconMat = pca.pca(dataMat, 2)
shape(lowDMat)
fig = plt.figure(1)
ax = fig.add_subplot(1,1,1)
ax.scatter(dataMat[:,0].tolist(), dataMat[:,1].tolist(), marker = '^', s = 90)
ax.scatter(reconMat[:,0].tolist(), reconMat[:,1].tolist(), marker = 'o', s = 50, c = 'red')
plt.savefig("4-1-topNfeat(2).png", dpi = 400)
plt.show()
```

3. 实验 4-2

实验 4-2

```
import numpy as np
import pca
import matplotlib.pyplot as plt

dataMat = pca.replaceNanWithMean()
meanVals = np.mean(dataMat, axis = 0)
meanRemoved = dataMat - meanVals
covMat = np.cov(meanRemoved, rowvar = 0)
eigVals, eigVects = np.linalg.eig(np.mat(covMat))

# 计算降维后恢复的数据与原始数据相对误差小于9%的topNfeat
topNfeat = 0
Re_err = 0
for i in range(999):
    lowDMat, reconMat = pca.pca(dataMat, i)
    Err = np.linalg.norm(dataMat - reconMat) / np.linalg.norm(dataMat)
    if Err < 0.09:
        topNfeat = i
        Re_err = Err
```

```
20          break
21
22  print('降维后恢复的数据与原始数据相对误差小于9%的topNfeat为:',topNfeat)
23  print('此时相对误差为：%0.5f%%' % (Re_err*100) )
```

三、 实验结果

1. 实验 4-1

topNfeat = 1:

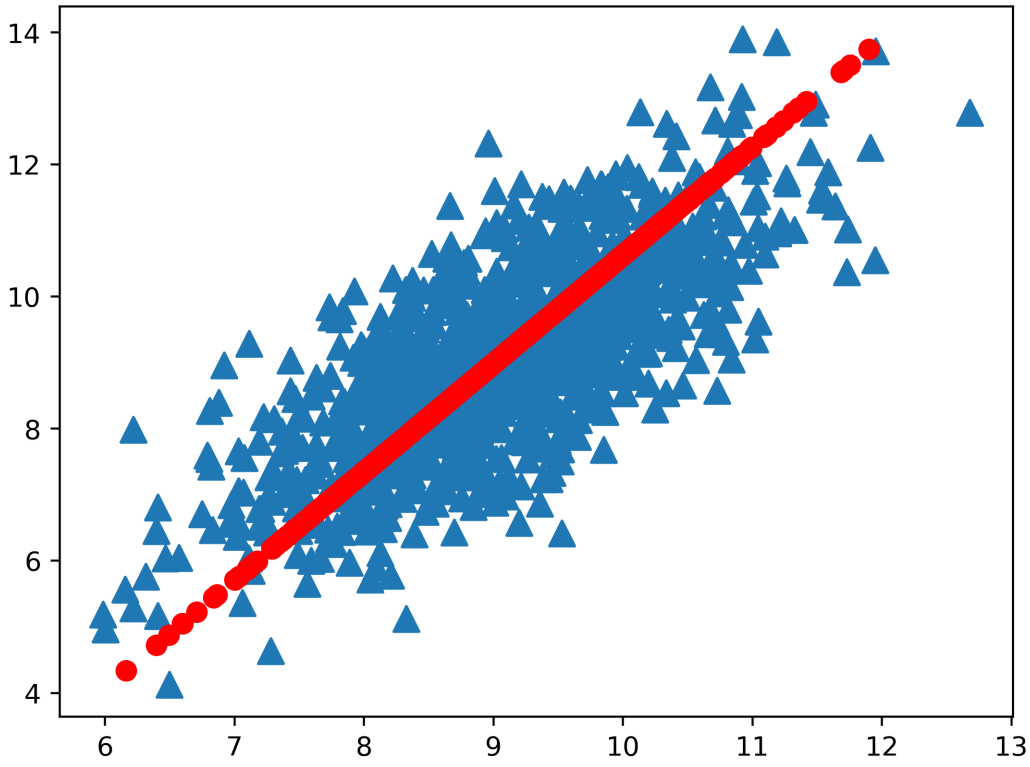

图 1: topNfeat = 1 可视化

topNfeat = 2:

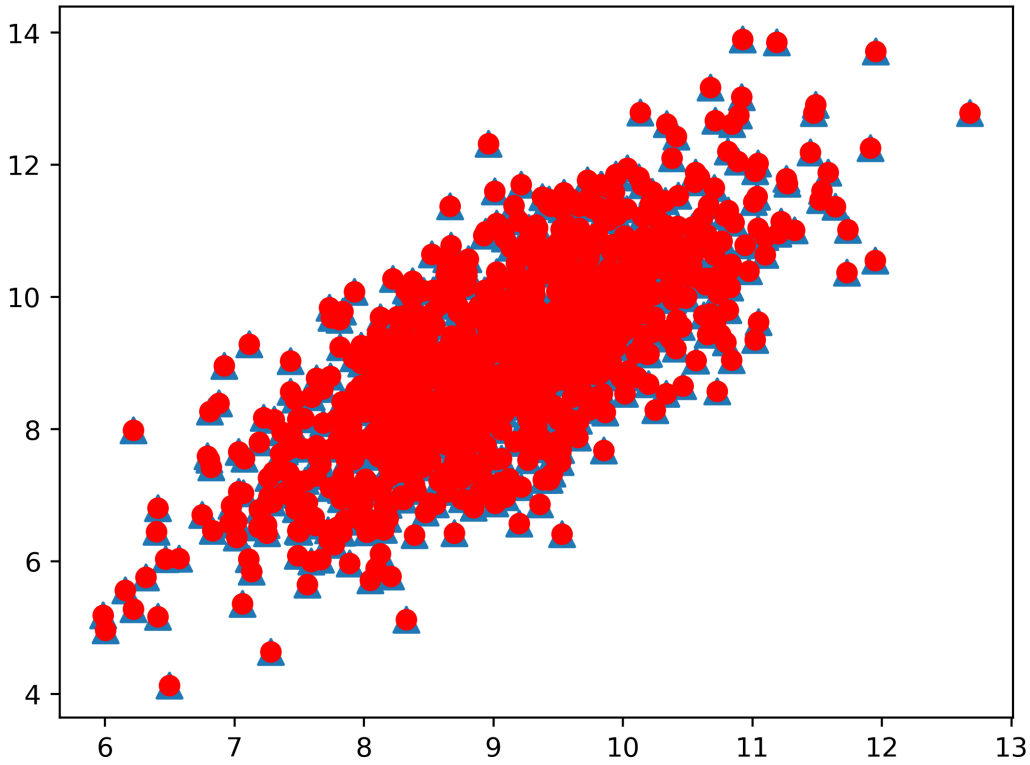

图 2: topNfeat = 2 可视化

结果可视化如下图：

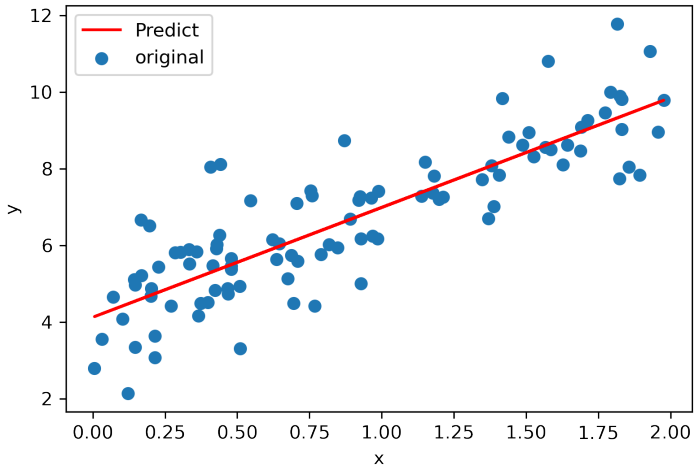

图 3: 线性回归结果可视化

2. 多项式回归

结果如下：
系数 = [0.94986314, 0.47823706]
截距 = [2.03896946]
模型为：

$$y = 0.47823706x^2 + 0.94986314x + 2.03896946$$

结果可视化如下图：

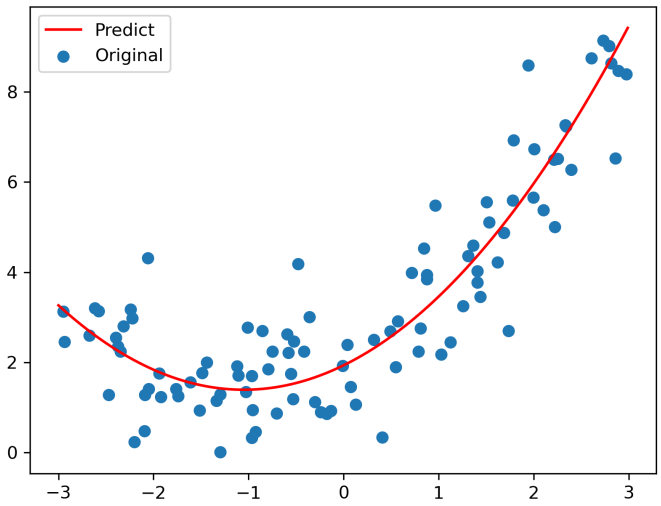

图 4: 多项式回归结果可视化

3. 逻辑回归

结果可视化如下图：

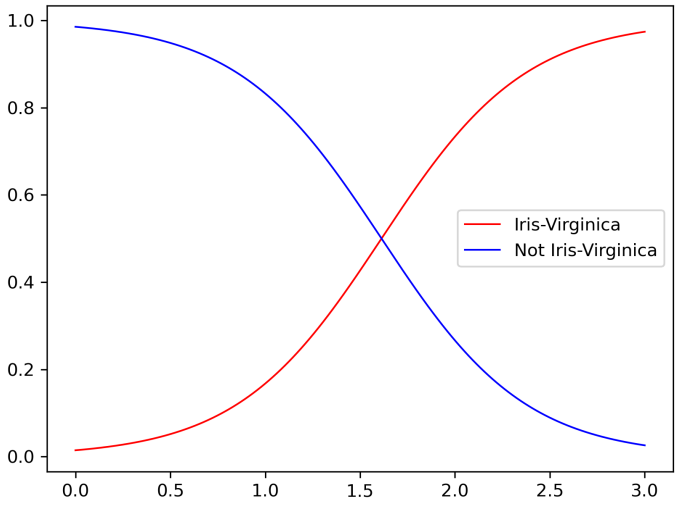

图 5: 逻辑回归结果可视化