山东大学 计算机科学与技术 学院

机器学习 课程实验报告

学号：201618130133	姓名：代荣森	班级：2016.4
实验题目：PCA		
实验学时：4	实验日期：2018.12.25	

实验目的：
学习使用 PCA 进行人脸识别

硬件环境：
i5-6200U 8G RAM HD Graphics520

软件环境：
Visual Studio Code + Python

实验步骤与内容：
首先读取数据，从 40 个人中每个人的 10 张照片里随机取 6 张作为训练集，另外 4 张作为测试集，文件夹号作为 label（每人一个文件夹）。使用 PCA 进行降维提取特征值，再通过欧式距离法或者 SVM 分类器进行人脸识别。对于不同的 k 值，即特征值的维度，得到以下正确率：

k = 1,15.00%
k = 3,66.25%
k = 5,85.62%
k = 7,91.88%
k = 9,91.25%
k = 11,91.25%
k = 13,91.88%
k = 15,91.88%
k = 17,91.25%
k = 19,91.88%
k = 21,91.25%
k = 23,91.25%
k = 25,91.88%
k = 27,91.88%
k = 29,91.88%
k = 31,91.88%
k = 33,91.88%

由此可见，当 k 值从很小逐渐增大的时候，正确率增大很快，但增大的速率逐渐减慢

结论分析与体会：
最近在学习 python，所以使用 python 来写了这个实验，由于一些规则用法不一样，所以有些地方需要特别注意
这个实现稍微比 K-Means 难一些，但有了之前的基础，问题也不是很大。
在测试的时候刚开始正确率只有 2.5%，各种 debug 找不到错误，浪费了很长时间，后来发现是训练集把一个文件夹读了 40 遍……所以写程序细心真的很重要
当选取的特征值很少时，运算速度会很快，但正确率会很低，随着特征的增加，正确率提高很快，运算速度逐渐减慢，当特征继续增加，正确率趋于不变，但运算速度依然减慢，所以选择一个合适的特征数量 k 很重要。

附录：程序源代码

PCA.py

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.sparse import linalg
import matplotlib.image as img
import random

def loadImageSet():
    trainData = []; testData = []; trainLabel = []; testLabel = []
    path = 'D:\\GitHub\\Machine-Learning\\PCA in Face Detection\\orl_faces'
    for i in range(40):
        count = random.sample(range(10),6)#无重复随机 6 个 0~9 的数
        data = [img.imread(path + '\\s%d\\%d.pgm' %(i+1,j+1)) for j in range(10)]#
每次循环读入一个文件夹里的 10 张人脸
        trainData.extend(data[j].ravel() for j in count)#随机的 6 张作为训练集
        testData.extend(data[j].ravel() for j in range(10) if j not in count)#剩下的
4 张作为测试集
        trainLabel.extend([i] * 6)#读入 6 个当前 i 作为训练集 label
        testLabel.extend([i] * 4)#读入 4 个当前 i 作为测试集 label
    return np.array(trainData), np.array(testData), np.array(trainLabel),
np.array(testLabel)

def pca(data,k):
    m = np.size(data)
    trainMean = np.mean(data,0)
    data = data - trainMean#数据归一化，即减去平均值
    data = np.mat(data)#转为矩阵
    S = data.T * data / m#计算协方差矩阵
    U = linalg.eigs(S,k)#计算特征值
    Z = data * U[1]#得到特征矩阵
    return np.array(Z),trainMean,U

def main():
    trainD, testD, trainL, testL = loadImageSet()#读入图片集

    k = 10#取特征的数量，即降维
    Z,trainMean,U = pca(trainD,k)
    testD = testD - trainMean#对测试集进行归一化
    testD = np.mat(testD)
    Z1 = np.array(testD * U[1])#得到测试集的特征矩阵

    testP = [trainL[np.sum((Z-d)**2,1).argmin()]for d in Z1]#欧式距离法得到测试集的预
测值 label
    print("%.2f%%" %((testP == testL).mean()*100))
    # 改变 k
```

```
    # predict = {}
    # for k in range(1,50,2):
    #   Z,trainMean,U = pca(trainD,k)
    #   testD1 = testD - trainMean#对测试集进行归一化
    #   testD1 = np.mat(testD1)
    #   Z1 = np.array(testD1 * U[1])#得到测试集的特征矩阵
    #   testP = [trainL[np.sum((Z-d)**2,1).argmin()]for d in Z1]#欧式距离法得到测试集的预测值 label
    #   predict[k] = (testP == testL).mean()
    #   print("k = %d,%.2f%%" %(k,predict[k]*100))
    # plt.plot(predict)
    # plt.show()

    # testP = []
    # o_length = np.zeros(240)
    # for d in range(len(Z1)):
    #   for j in range(240):
    #       o_length[j] = (np.sum((Z[j]-Z1[d])**2))**0.5
    #   b = o_length.tolist().index(min(o_length))
    #   a = trainL[b]
    #   testP.append(a)
    # print((testP == testL).mean())

if __name__ == '__main__':
    main()
```