浙江大学

本科实验报告

特征人脸识别

课程名称: 人工智能

姓名: 刘星烨

学院: 计算机科学与技术

专业: 计算机科学与技术

学号: 3180105954

指导教师: 吴飞

浙江大学实验报告

课程名称: 人工智能 实验类型:

实验项目名称: 特征人脸识别

姓名: 刘星烨 专业: 计算机科学与技术 学号: 3180105954

同组学生姓名: 指导教师: 吴飞

实验地点: 实验日期:

1 问题重述

本实验采用特征脸（Eigenface）算法进行人脸识别。特征脸（eigenface）是第一种有效的人脸识别方法，通过在一大组描述不同人脸的图像上进行主成分分析（PCA）获得。在模型训练过程中，首先要根据测试数据求出平均脸，然后将前 K 个特征脸保存下来，利用这 K 个特征脸对测试人脸进行识别，此外对于任意给定的一张人脸图像，可以使用这 K 个特征脸对原图进行重建。

2 设计思想

2.1 介绍

主成分分析（Principal Component Analysis，PCA），是一种统计方法。通过正交变换将一组可能存在相关性的变量转换为一组线性不相关的变量，转换后的这组变量叫主成分。在用统计分析方法研究多变量的课题时，变量个数太多就会增加课题的复杂性。人们自然希望变量个数较少而得到的信息较多。在很多情形，变量之间是有一定的相关关系的，当两个变量之间有一定相关关系时，可以解释为这两个变量反映此课题的信息有一定的重叠。主成分分析是对于原先提出的所有变量，将重复的变量（关系紧密的变量）删去多余，建立尽可能少的新变量，使得这些新变量是两两不相关的，而且这些新变量在反映课题的信息方面尽可能保持原有的信息。

2.2 基本思想

主成分分析是设法将原来众多具有一定相关性（比如 P 个指标），重新组合成一组新的互相无关的综合指标来代替原来的指标。主成分分析，是考察多个变量间相关性一种多元统计方法，研究如何通过少数几个主成分来揭示多个变量间的内部结构，即从原始变量中导出少数几个主成分，使它们尽可能多地保留原始变量的信息，且彼此间互不相关. 通常数学上的处理就是将原来 P 个指标作线性组合，作为新的综合指标。最经典的做法就是用 F1（选取的第一个线性组合，即第一个综合指标）的方差来表达，即 Var(F1) 越大，表示 F1 包含的信息越多。因此在所有的线性组合中选取的 F1 应该是方差最大的，故称 F1 为第一主成分。如果第一主成分不足以代表原来 P 个指标的信息，再考虑选取 F2 即选第二个线性组合，为了有效地反映原来信息，F1 已有的信息就不需要再出现在 F2 中，用数学语言表达就是要求 Cov(F1, F2)=0，则称 F2 为第二主成分，依此类推可以构造出第三、第四，……，第 P 个主成分。步骤

$$F_p = a_{1i} * Z_{X1} + a_{2i} * Z_{X2} + \ldots\ldots + a_{pi} * Z_{Xp}$$

其中

$$a_{1i}, a_{2i}, \ldots\ldots, a_{pi} (i = 1, \ldots\ldots, m)$$

为 X 的协方差阵 Σ 的特征值所对应的特征向量，

$$Z_{X1}, Z_{X2}, \ldots\ldots, Z_{Xp}$$

是原始变量经过标准化处理的值，因为在实际应用中，往往存在指标的量纲不同，所以在计算之前须先消除量纲的影响，而将原始数据标准化，本文所采用的数据就存在量纲影响 [注：本文指的数据标准化是指 Z 标准化。

$$A = (a_{ij})p \times m = (a_1, a_2, \ldots a_m)\ R_{ai} = {}_i a_i$$

，R 为相关系数矩阵，

$$_i\ a_i$$

是相应的特征值和单位特征向量，

$$_1\ _2\ \cdots\ _p\ 0$$

。进行主成分分析主要步骤如下：

- 指标数据标准化（SPSS 软件自动执行）；
- 指标之间的相关性判定；
- 确定主成分个数 m；
- 主成分 Fi 表达式；
- 主成分 Fi 命名。

3 代码内容

3.1 训练特征脸

根据数据进行计算，取平均值作为平均脸，将训练集减去平均值作为中心化脸，再计算协方差矩阵，根据特征向量与特征值大小排序得到特征脸向量。

```
    def eigen_train(trainset, k=20):
    """
    训练特征脸（eigenface）算法的实现

    :param trainset: 使用 get_images 函数得到的处理好的人脸数据训练集
    :param K: 希望提取的主特征数
    :return: 训练数据的平均脸，特征脸向量，中心化训练数据
    """

    #
        ##############################################################################

    ####                        训练特征脸（eigenface）算法的实现
                              ####
    ####                            请勿修改该函数的输入输出
                                  ####
    #
        ##############################################################################

    #


        #
    mean_face = np.zeros((1,len(trainset[0])))

    for i in trainset:
         mean_face = np.add(mean_face,i)

    mean_face = np.divide(mean_face,k).flatten()
    avg_img = mean_face
    norm_img = trainset-mean_face
    cov_matrix = np.cov(norm_img.T)

    cov_matrix = np.divide(cov_matrix, len(trainset))
    eigenvalues, eigenvectors = np.linalg.eig(cov_matrix)
    eig_pairs = [(eigenvalues[index], eigenvectors[:,index]) for
        index in range(len(eigenvalues))]
```

```
    # Sort the eigen pairs in descending order:
    eig_pairs.sort(reverse=True)
    eigvalues_sort  = [eig_pairs[index][0] for index in range(len(
        eigenvalues))]
    eigvectors_sort = [eig_pairs[index][1] for index in range(len(
        eigenvalues))]
    eigvectors_sort = np.array(eigvectors_sort)
    print(eigvectors_sort.shape[0])
    print(eigvectors_sort.shape[1])
    eigvectors_sort = eigvectors_sort.T
    for i in range(eigvectors_sort.shape[0]):
        eigvectors_sort[i]=eigvectors_sort[i]/np.sqrt(sum(np.square(
            eigvectors_sort[i])))
    feature = eigvectors_sort
    # 返回: 平均人脸、特征人脸、中心化人脸
    return avg_img, feature, norm_img
```

3.2 特征脸投影映射

使用特征向量与输入图像进行点乘，获得相应特征表示的数据。

```
    def rep_face(image, avg_img, eigenface_vects, numComponents = 0):
    """
    用特征脸（eigenface）算法对输入数据进行投影映射，得到使用特征脸向
        量表示的数据

    :param image: 输入数据
    :param avg_img: 训练集的平均人脸数据
    :param eigenface_vects: 特征脸向量
    :param numComponents: 选用的特征脸数量
    :return: 输入数据的特征向量表示，最终使用的特征脸数量
    """

    #
        ##############################################################################

    ####  用特征脸（eigenface）算法对输入数据进行投影映射，得到使用特
        征脸向量表示的数据  ####
    ####                              请勿修改该函数的输入输出
                                     ####
    #
        ##############################################################################

    #


        #
    image=image-avg_img
    describe=np.zeros(image.shape[0])
    for i in range(numComponents):
         describe[i]=np.dot(image,eigenface_vects[i])
    num=0
    for i in range(describe.shape[0]):
         if(abs(describe[i])<1e-8):
              describe[i]=0.0
         else:
              num=num+1
    #
```

```
    numEigenFaces=numComponents
    representation=describe
    # 返回: 输入数据的特征向量表示, 特征脸使用数量
    return representation, numEigenFaces
```

3.3 人脸恢复

```
    def recFace(representations, avg_img, eigenVectors, numComponents
        , sz=(112,92)):
    """
    利用特征人脸重建原始人脸

    :param representations: 表征数据
    :param avg_img: 训练集的平均人脸数据
    :param eigenface_vects: 特征脸向量
    :param numComponents: 选用的特征脸数量
    :param sz: 原始图片大小
    :return: 重建人脸, str 使用的特征人脸数量
    """

    #
        ##############################################################################

    ####                              利用特征人脸重建原始人脸
                                    ####
    ####                              请勿修改该函数的输入输出
                                    ####
    #
        ##############################################################################

    #


        #
    a=avg_img.copy()
    for i in range(representations.shape[0]):
         if(abs(representations[i])>1e-8):
             a=a+representations[i]*eigenVectors[i]
    face=a
    # 返回: 重建人脸, str 使用的特征人脸数量
    return face, 'numEigenFaces_{}'.format(numComponents)
```

4 实验结果

通过网站测试点测试。