数字图像处理 人脸识别项目

17341190 叶盛源

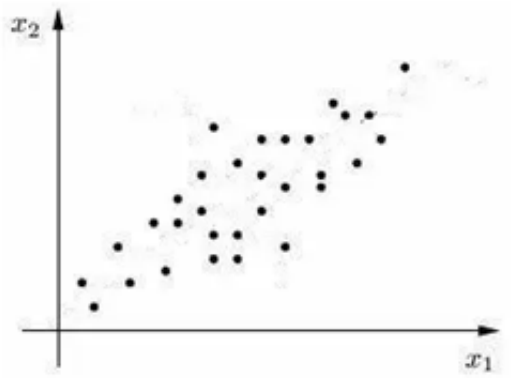
数据科学与计算机学院

PCA算法原理

PCA（Principal Component Analysis）是一种常用的数据分析方法。PCA通过线性变换将原始数据变换为一组各维度线性无关的表示，可用于提取数据的主要特征分量，常用于高维数据的降维。PCA的思想是将n维特征映射到k维空间上$k < n$，这k维特征是全新的正交特征，是通过PCA的算法重新构造出来的k维特征。

在线性代数中，矩阵乘法可以看作一个线性的变换，相当于对向量的维度进行变换。这是PCA降维的基础。

协方差



如上面的二维情况，如果我们必须使用一维来表示这些散点，目标就是找到一个矩阵，对这些向量线性变换改变维度后，将这些点投影到一条直线上。投影后如果想要尽可能区分这些不同的点，我们需要让投影后的点之间尽量分离，也就是方差要尽量大。可以用数学上的方差来表述。被形式化表述为：寻找一个一维基，使得所有数据变换为这个基上的坐标表示后，方差值最大。

对于上面的一维问题来说可以这样解决，如果是更高维度的情况，比如三维要降到一维，我们需要找到两个基方向。首先，我们还是先要找到方差最大的基方向，然后再重复一次上面的操作找第二个，不过如果两次都是找方差最大的基方向，两次找到的方向就会相同（这样得到的两个基就没有意义，因为重复表示了特征）。所以我们还需要增加一个条件，就是要求找到的基向量要线性无关，也就是找到的两个向量最好是垂直的，也就是最好代表不同特征的方向。

因此，我们得到了降维问题的优化目标：**将一组N维向量降为K维，其目标是选择K个单位正交基，使得原始数据变换到这组基上后，各个特征两两间协方差为0，而特征的方差则尽可能大，在正交的约束下，取最大的K个方差。**

直观的来看就是：从原始的空间中顺序地找一组相互正交的坐标轴，新的坐标轴的选择与数据本身是密切相关的。其中，第一个新坐标轴选择是原始数据中方差最大的方向，第二个新坐标轴选取是与第一个坐标轴正交的平面中使得方差最大的，第三个轴是与第1,2个轴正交的平面中方差最大的。依次类推，可以得到k个这样的坐标轴。

算法基本流程

假设一共有N个图像样本，各个图像样本是d维向量，我们目标是降为k维，流程如下：

1. 计算所有样本图像向量的均值：

$$\overline{x} = \frac{1}{N}\sum_{i=1}^{N} x_i, x_i \in R^d$$

2. 接着计算协方差矩阵，这里要使用无偏的估计方法：

$$C = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})(y_i - \overline{y})^T$$

3. 然后计算协方差矩阵的特征向量，然后选择特征值前k大的特征向量，组成投影后维度的向量基。将原始数据中的每一个点都通过矩阵乘法线性映射到k维的特征空间中，就可以得到映射后的样本向量。

人脸识别PCA算法

PCA训练

人脸识别是对图像进行操作，基本算法原理就是上面说的过程。因为图像是二维的向量，我们可以将图像每个像素点沿一维展平，变成一个一维向量，将每个像素点作为一个特征，就可以应用于PCA算法中。不过这样的样本向量特征维度非常大，计算代价很大，而且存在很多不必要的信息：可能某些特征是别的特征的线性组合，这些特征就可以被去掉，因为他们可以被其他更主要的特征替代。

k值确定

定义x_{approx}为映射后的值，x为样本点均值映射后的值。

选择不同的K值，然后用下面的式子不断计算，选取能够满足下列式子条件的最小K值即可。

$$\frac{\frac{1}{m}\sum_{i=1}^{m}||x^{(i)} - x_{approx}(i)||^2}{\frac{1}{m}\sum_{i=1}^{m}||x^{(i)}||} \leq t$$

其中t值可以由自己定，比如t值取0.01，则代表了该PCA算法保留了99%的主要信息

识别

我们通过计算协方差矩阵的特征向量，得到数据降维的变换矩阵，我们可以把高维的向量映射到k维的特征空间中。之后我们将数据库中的人脸图片用这种方法进行特征的提取，然后将待检测的图像也映射到k维空间上，最后我们分别将待检测图像和数据库中的图像进行比较，计算相似度或者平方误差等，最后将最接近数据库图像的作为待测图像的类别。

matlab实现

首先我们先观察数据集，数据集由40个不同人的脸图片组成，每个人有10张图片。对每个人的 10 张图像，随机选择 5 张作为训练数据，其余作为测试数据。我们首先划分数据集，从10个照片中，随机选出5张作为训练，剩下的5个作为测试，同时我们定义一个数组，记录每个训练样本的标签：

```
function [train_data,test_data,test_label] = divide_data(n)
    train_data = zeros(40,10304);    % 将图像的像素展平是10304维的向量
    test_data = [];
    test_label=[];

    % 遍历每个人物的每张照片
    for i = 1:40
        random_num = randperm(10, n);  % 从10张人脸照片中选出n张作为训练集
        for j = 1:10
            img =
reshape(double(imread(['./ORL_faces/s',num2str(i),'/',num2str(j),'.pgm'])),1,1
0304);
            if ismember(j, random_num)
                train_data(i,:) = train_data(i,:) + img;
            else
                test_data = [test_data; img];
                test_label = [test_label;i];    % 记录训练集的标签
            end
        end
    end
    train_data = train_data / n;        % 计算几张图片的平均值
end
```

接着我们编写人脸识别的测试函数，我们首先将划分出的训练集用matlab的自带函数`pca`计算出线性变换的矩阵，然后将每个训练集中的向量映射到新的特征空间中，训练出来40个人脸类别的特征向量，之后我们将测试集的向量也做一个映射，然后使用2范数找出最小距离的类别，作为测试图像的类别。

```
function accuracy = Identify(train_data, test_data,test_label)
    [coeff, ~, ~] = pca(train_data); % 使用pca训练数据
    train_pca = train_data * coeff;  % 把训练的数据映射到k维空间上
    % 进行图像识别
    [M, N] = size(train_data); % 读取数据库中数据集大小
    [m, n] = size(test_data); % 读取等待测试的图片数据集大小
    count = 0;  % 统计正确预测的图片数
    for i = 1:m
        similarity = [];
        test_pca = test_data(i,:) * coeff;
        for k = 1:M    % 遍历所有人物图像的特征向量 计算和待检测图片的距离
```

```
            similarity = [similarity,norm(train_pca(k,:) - test_pca, 2)];
        end
        [~, index] = min(similarity);    % 从所有可能种类中找出相似度最高的向量
        if index == test_label(i)
            count = count+1;
        end
    end
    accuracy = count / m;
end
```

实验结果

每个人有10张照片，因此每次训练的时候是从10张照片中随机选取5个照片，所以我们可以多次执行选取平均的准确度。

重复执行了100次后，我们绘制了100次的精确度曲线图可以看到最高的时候正确率达到了0.97多，而平均的正确率也有0.91左右。总体来说，模型的表现相对还是不错的，在数据量不多的情况下，用传统的机器学习方法得到一个比较优秀的实验结果。