# 人脸特征检测与识别

Eigenfaces就是特征脸的意思，是一种从主成分分析（Principal Component Analysis，PCA）中导出的人脸识别和描述技术。特征脸方法的主要思路就是将输入的人脸图像看作一个个矩阵，通过在人脸空间中一组正交向量，并选择最重要的正交向量，作为“主成分”来描述原来的人脸空间。为了更好地理解特征脸方法，需要先了解PCA的主要过程。

在很多应用中，需要对大量数据进行分析计算并寻找其内在的规律，但是数据量巨大造成了问题分析的复杂性，因此我们需要一些合理的方法来减少分析的数据和变量同时尽量不破坏数据之间的关联性。于是这就有了主成分分析方法，PCA作用：数据降维。减少变量个数；确保变量独立；提供一个合理的框架解释。去除噪声，发现数据背后的固有模式。

PCA的主要过程：

1.特征中心化：将每一维的数据（矩阵A）都减去该维的均值，使得变换后（矩阵B）每一维均值为0；

2.计算变换后矩阵B的协方差矩阵C；

3.计算协方差矩阵C的特征值和特征向量；

4.选取大的特征值对应的特征向量作为”主成分”，并构成新的数据集；

Pca缺点：PCA本质上是将方差最大的方向作为主要特征，并且在各个正交方向上将数据“离相关”，也就是让它们在不同正交方向上没有相关性。

PCA也存在一些限制，例如它可以很好的解除线性相关，但是对于高阶相关性就没有办法了，对于存在高阶相关性的数据，可以考虑Kernel PCA，通过Kernel函数将非线性相关转为线性相关，另外，PCA假设数据各主特征是分布在正交方向上，如果在非正交方向上存在几个方差较大的方向，PCA的效果就大打折扣了。最后需要说明的是，PCA是一种无参数技术，也就是说面对同样的数据，如果不考虑清洗，谁来做结果都一样，没有主观参数的介入，所以PCA便于通用实现，但是本身无法个性化的优化。

特征脸方法

特征脸方法就是将PCA方法应用到人脸识别中，将人脸图像看成是原始数据集，使用PCA方法对其进行处理和降维，得到“主成分”——即特征脸，然后每个人脸都可以用特征脸的组合进行表示。这种方法的核心思路是认为同一类事物必然存在相同特性（主成分），通过将同一目标（人脸图像）的特性寻在出来，就可以用来区分不同的事物了。人脸识别嘛，就是一个分类的问题，将不同的人脸区分开来。特征脸方法的过程（先计算特征脸，然后识别人脸）：

1.将训练集中的N个人脸拉成一列（reshape(1,1)），然后组合在一起形成一个大矩阵A。若人脸图像大小为m \* m，则矩阵A的维度是m \* m \* N；

2.将N个人脸在对应的维度求平均，得到一个“平均脸”；

3.将矩阵A中N个图像都减去“平均脸”，得到新矩阵B；

4.计算B的协方差矩阵；

5.计算协方差矩阵的特征值和特征向量（特征脸）；

6.将训练集图像和测试集图像都投影到特征向量空间中，再使用聚类方法（最近邻或k近邻等）得到里测试集中的每个图像最近的图像，进行分类即可。

在opencv中void EigenFaceRecognizer::train();就是对训练集进行处理最终得到特征向量和阈值的过程，其中的阈值用于后面对测试图像的识别。int EigenFaceRecognizer:: predict(InputArray src) const;是对测试图像进行识别。

特征脸识别的局限性

要让系统准确识别需要保证人脸图像满足：

待识别图像中人脸尺寸接近特征脸中人脸的尺寸；

待识别人脸图像必须为正面人脸图像。

若不满足此条件，识别错误率很高。从PCA方法的过程可以看出，特征脸识别的方法是以每张人脸的一个维度（可以看出是矩阵的一列）为单位进行处理的，求得的特征向量（特征脸）中包含训练集每个纬度的绝大部分信息。但是若测试集中人脸尺寸不同，那么与特征脸中维度的也就没法对应起来。

Fisherfaces

在将Fisherfaces之前需要对线性判别进行一定的说明。

Fisher线性判别分析（linear discriminant analysis，LDA）

线性判别分析是由Fisher提出的线性判别方法，可以用来处理两类的线性判别问题。两类的线性判别问题可以看做所有的样本投影到一个方向（或者说是一个维度空间中），然后再这个空间中确定一个分类的阈值。过这个阈值点且与投影方向垂直的超平面就是分类面。判别思路是选择投影方向，使得投影后两类相隔尽可能远，类内又尽可能聚集（类间方差最大，类内方差最小）。

它的过程分为：

确定最优的投影方向：

在这个方向上确定分类阈值；

Fisherfaces方法

Fisherfaces方法结合了PCA和LDA的优点，其具体过程如下：

PCA降维：对原始样本进行PCA处理，获取PCA处理之后的新样本；

LDA特征提取：对降维后的样本使用Fisher线性判别方法，确定一个最优的投影方向，构造一个一维的特征空间（这就被称为Fisherfaces），将多维的人脸图像投影到Fisherfaces特征空间，利用类内样本数据形成一组特征向量，这组特征向量就代表了人脸的特征。

在opencv中void FisherFaceRecognizer::train();就是对训练集进行处理最终得到特征向量和阈值的过程，其中的阈值用于后面对测试图像的识别。int FisherFaceRecognizer:: predict(InputArray src) const;是对测试图像进行识别

LBP-DBN-face-recognition

使用LBP特征提取算法提取人脸特征，DBN网络来实现人脸识别，测试数据库-ORL数据库，识别率可达90%以上

###工程使用方法：

clone下工程，要根据实际情况修改test\_example\_DBN.m中的文件路径

我们在该文件中，a=imread(strcat('ORL\ORL\s',num2str(i),'\_',num2str(j),'.bmp'));

###工程说明：

####英文简称注释：

DBN-深度信念网络、RBM-受限的玻尔兹曼机 、LBP-局部二值模式

####几点说明：

0-这个程序的功能，使用DBN算法来实现人脸识别，数据库使用ORL数据库，在迭代次数达到3000时，识别准确率98%左右

2-此程序还附带了画学习曲线的功能，画正则参数，隐层结点数，训练样本数的学习曲线

3-DBN是有几层RBM构成，我这个程序实现的是4层网络，输入层-隐层1-隐层2-输出层

4-DBN的训练基本上分为两步，先用RBM的训练方法训练网络得到初始值，来初始化整个网络，然后用BP反向传播算法来微调整个网络

5-神经网络参数的更新，使用matlab提供的fmincg函数，前提是我们要先得到网络的代价函数nnCostFunction

####各文件的功能说明：

test\_example\_DBN：主函数

dbnsetup：初始化DBN网络

dbntrain:训练DBN网络，DBN是由多层RBM组成，训练方法是逐层训练，先训练第一层网络，然后固定第一层网络的参数，将第一层网络的输出作为下一层网络的输入

dbnunfoldtonn：DBN训练得到的参数来初始化神经网络

fmincg：最优化函数，只要我们得到网络的代价函数，和反向传播算法，就可以用此函数求最优解

getmapping，lbp,lbptest:完成lbp算法

hidden\_node\_learn\_curve:关于隐层结点的学习曲线

learningCurve：关于训练样本数目的学习曲线

nnCostFunction：求神经网络的代价函数，和BP反向传播算法

predict：利用网络进行预测

randInitializeWeights：随机初始化网络参数

rbmdown，rbmup，sigm，sigmoid，sigmoidGradient，sigmrnd：训练过程中使用到的计算函数

train\_nn：训练神经网络

rbmtrain：训练一个RBM网络

validationCurve：关于正则参数的学习曲线

整个框架就是：

先用LBP算子对人脸图像进行特征提取，然后用RBM算法逐层训练RBM，训练得到的RBM参数来初始化化DBN网络，然后计算出整个网络的代价函数，并使用BP反向传播算法，谨记反向传播算法也是梯度下降算法，并且可以用梯度校验来验证。梯度下降算法，学习率是一个比较关键的参数，但是，在我的程序中，自己不设学习率，只要我们实现了反向传播算法和代价函数，我们使用fmincg函数去得到最优解。

###优点：

神经网络采用的是随机初始化，但容易收敛到局部最小值，用DBN训练得到网络的初始化参数，而不是随机初始化。DBN随机初始化之后，网络的训练方式就跟训练神经网络一样。