

基于K-L变换的人脸识别算法

实验人：李东嵘 16342080

实验平台：Window10, MATLAB R2016a

实验目的：基于K-L变换，设计人脸识别算法，并在Yale数据集上测试效果。

1.引言

K-L变换在人脸识别领域有着一定的应用价值，本报告将按照如下顺序展开：首先，我们将讨论K-L变换的原理并明确在人脸识别中的具体算法。然后，我们将设计算法，在Yale数据集上测试算法效果，并汇报最终结果。

2.K-L变换的原理

设对样本矩阵 $X = (X_1, \dots, X_s)$ ，我们想将这些样本投影到一个低维空间，不妨考虑将样本维数降到

k 维，考虑其协方差矩阵 $\Sigma = (X - \bar{X})(X - \bar{X})^T$ ，我们取其前 k 大特征值对应的 k 个特征向量，记为 $\phi = (v_1, v_2, \dots, v_k)$ ，然后，接下来我们将通过这些特征向量将样本降维。

具体而言，我们知道 Σ 为一正定矩阵，因此 v_1, \dots, v_k 构成了一个 R^n 的 k 维子空间的标准正交基。我们将 v_1, \dots, v_k 称为这些数据的一组特征脸。我们可以利用这一组正交基将每个样本投影到这个低维子空间

上。事实上，对每个 X_i ，我们可以将其分解为 $X_i = \sum_{j=1}^k a_j v_j + \psi$ ，其中 $\psi \in \text{span}\{v_1, \dots, v_k\}^\perp$ 。于是，我们考虑 $u_i = \phi^T X_i$ ，即可得到每个样本在 $\text{span}\{v_1, \dots, v_k\}$ 上对应的投影，即 $(u_1, \dots, u_s) = \phi^T X$ 。从几何上来说，这相当于将每个人脸投影到了由 k 个特征脸向量张成的子空间上。

我们把 u_1, \dots, u_s 称为训练集在特征空间上的坐标。这样，对于一个新的测试样本 Y ，我们通过 $u = \phi^T Y$ 得到 Y 在特征空间的投影 u ，并寻找与 u 距离最近的向量 v ，并将 v 所对应样本的标签作为 u 的标签，就实现了分类问题。

具体而言，K-L变换算法步骤如下：

K-L变换进行人脸识别

输入：训练集 T ，测试集 F

输出：该算法在测试机上的正确率

1. $\Sigma = (T - \bar{T})(T - \bar{T})^T$
2. 对训练集T, 计算训练集的协方差矩阵
计算 Σ 的特征值, 并保留其前K个特征值, 记为 v_1, \dots, v_k
3. 计算 $(u_1, \dots, u_s) = \phi^T X$, 记为样本在低维空间上的坐标
4. for i in F

用 $w_i = \phi^T Y_i$ 计算每个测试样本在子空间上的投影, 用最近邻匹配法得到该数据的标签

5. 统计测试集的正确率。

3.Yale数据集介绍

我们将用著名人脸数据集Yale作为样本考察K-L算法的效果。

Yale数据集包含165张人脸图片, 被分为15类, 分属15个不同的人, 每张图片形态各异, 是测试人脸识别算法的优秀素材之一。

下图是Yale数据集的冰山一角。



4.k-L变换的具体实现

现在我们将具体实现K-L变换。出于可扩展性与灵活性考虑, 程序将被分为四个独立模块, 分别是:

- KLTransoform.m 模块接收训练集, 返回每个样本被降维后的坐标
- Classification.m 模块接收测试集, 并将测试集分类
- Accuracy.m 模块统计测试集上的识别正确率
- Recognize.m模块调用前两个模块, 用于进行任意图片的识别归类

这些模块的具体实现可以在附录中找到。

我们在导入数据后，随机将全体样本按照**135:30**的比例将其划分为训练集与测试集。然后，我们用训练集生成低维坐标(**Coordinates**)与投影矩阵(**Projector**)，并保留前**30**个特征值，其**MATLAB**命令如下：

```
[Projector,Coordinates]=KLTransform(Train,30); %30表明保留前30个特征值
```

调用**subplot()**命令可以观察到前**10**张特征脸如下：



接下来，我们用**Classification()**函数对测试集进行分类：

```
Labeled=Classification(Test,Train_Label,Projector,Coordinates);
```

其中，**Labeled**变量为一个列表，其元素为测试集相应样本被打上的标记。

最后，我们用**Accuracy()**函数测试正确率：

```
Accuracy(Labeled,Test_Label)
```

最终可以得到算法在测试集的正确率为**0.8333**，还算是一个不错的表现。

至此，用**K-L**变换进行人脸识别的任务已经完成，我们可以看到该变换在测试机上取得了不俗的成绩，但依然不够高。

APPENDIX

1.ReadYale.m

改程序用于从计算机本地读取Yale数据集，并均匀随机划分训练集与数据集。

```
%function [train,label,test]=ReadYale()
People_Num=15;
People_Type=11;
%15people included, and each has 11 images

%initialize Face,Train,Test,Train_Label,Test_Label
Face=zeros(128,128,People_Num*People_Type);
Train=zeros(128*128,People_Num*People_Type);
Test=zeros(128*128,30);
Train_Label=zeros(People_Num*People_Type,1);
Test_Label=zeros(30,1);

%Read the faces, and also record their labels
for i=1:People_Num
    for j=1:11
        if i<10
            if j<10
                char=['C:\Users\A\Desktop\yale\00',int2str(i),'0',int2str(j),'.bmp'];
            else
                char=['C:\users\A\Desktop\yale\00',int2str(i),int2str(j),'.bmp'];
            end
        else
            if j<10
                char=['C:\Users\A\Desktop\yale\0',int2str(i),'0',int2str(j),'.bmp'];
            else
                caher=['c:\users\A\Desktop\yale\0',int2str(i),int2str(j),'.bmp'];
            end
        end
        temp=imread(char);
        Face(:,:,People_Type*(i-1)+j)=temp;
        Train_Label(People_Type*(i-1)+j)=i;
    end
end

Face=im2double(Face);

%vectorize the faces, firstly put all vectors into the Training set
for i=1:People_Num*People_Type
    tmp=Face(:,:,i);
    Train(:,i)=tmp(:);
end

Cancellation=zeros(30,1);

%Randomly delete some columns of the training set and put the
%deleted data into the test set
for i=1:15
    random_1=unidrnd(11);
    random_2=unidrnd(11);
```

```

while random_1==random_2
    random_1=unidrnd(11);
    random_2=unidrnd(11);
end
Test(:,2*(i-1)+1)=Train(:,11*(i-1)+random_1);
Test(:,2*(i-1)+2)=Train(:,11*(i-1)+random_2);
Test_Label(2*(i-1)+1)=Train_Label(11*(i-1)+random_1);
Test_Label(2*(i-1)+2)=Train_Label(11*(i-1)+random_2);
Cancellation(2*(i-1)+1)=11*(i-1)+random_1;
Cancellation(2*(i-1)+2)=11*(i-1)+random_2;

end

Train(:,Cancellation)=[];
Train_Label(Cancellation)=[];

```

2.KLTransform.m

该程序接收训练集数据，并返回投影矩阵**Projector**与样本在低维空间的坐标**Coordinates**

```

%function [Projector,Coordinates]=KLTransform(Train,k,People_Num)
%Return the projection matrix and coordinates of samples after projecting
%Train is the training set
%k is the number of eigen vectors we would like to preserve

if nargin<3
    People_Num=9;
end

%count the sample mean
t=Train';
Avg=mean(t);
Avg=Avg';

s=size(Train);
col=s(2);
row=s(1);

Avg=repmat(Avg,1,col);

%Calculate the covariance Matrix
Sigma=Train*(Train');
Conver=((Train-Avg)')*(Train-Avg);

%eigen vectors and eigen values
[vectors,values]=eig(Conver);

vectors=Train*vectors;
values=diag(values);

%initialize the projector,put the k largest eigen vectors into the
%Projection matrix
Projector=zeros(row,k);
for i=1:k
    lambda=max(values);
    index=find(values==lambda);

```

```

    Projector(:,i)=vectors(:,index);
    values(index)=0;
    Projector(:,i)=Projector(:,i)/norm(Projector(:,i));
end

%Return the Coordinates after projection into low-dimension spaces
Coordinates=(Projector)'*Train;

%end

```

3.Classification.m

该函数接收测试集与Projector, Coordinates, 然后对测试集进行标注。

```

%function Labeled=Classification(Test,Train_Label,Projector,Coordinates,People_Num)
%label the test set with the training label, projector and coordinates

if nargin<5
    People_Num=9;
end

s=size(Test);
T_row=s(1);
T_col=s(2);

s1=size(Projector);
Pro_dim=s1(2);
Pro_Num=s1(1);

%Calculate the projected vectors of the test set
Projected=(Projector')*Test;

%initialize the labels
Labels=zeros(T_col,1);

%use the nearest neighbourhood method to label the test set
for i=1:T_col
    d=pdist2(Coordinates',Projected(:,i)');

    m=min(d);

    index=find(d==m);
    index;
    Labels(i)=Train_Label(index(1));
end

Labeled=Labels;

%end

```

4.Recognize.m

用于实现任意图像的识别分类

```
%function lab=Recognize(image,Train,Train_Label,k)
image=im2double(image);
image=image(:);

[P,C]=KLTransform(Train,k);
lab=Classification(image,Train_Label,P,C);

%end
```