

统计分析综合作业

16337183 孟衍璋

实验要求

ORL Database是一个有名的人脸数据库。里面有40个ID的人脸，每个ID有10张图。本次作业是一个综合的project. 要求自己写PCA,马氏距离，k-means代码，不调用相关的库。

数据集的文件组织：有40个文件夹(s_1, s_2, \dots, s_{40})，每个文件夹是一个ID，每个文件夹(ID)有10张人脸(1.pgm, 2.pgm, ..., 10.pgm)。

训练和测试集划分：训练集由每个文件夹中的1.pgm, 2.pgm, ..., 6.pgm组成，共240张图。测试集由剩下160张图组成。说明：这里的训练集也同时作为gallery，测试集是query。意思是，1) 我们会用训练集来计算pca，同时在评测阶段，我们把训练集的图片作为检索库。2) 举个例子来说，我们会用160张图的每一张作为query，依次去对照这240张图(gallery)，看看哪张图最像，然后返回这张最像的图。

问题1：PCA降维。

每张图的大小是 $112 * 92$ ，我们把它拉成一个向量 $112 * 92 = 10304$ 维度，240张图就组成 $240 * 10304$ 的矩阵。我们降维到 $240 * 40$ 维度的矩阵，也就是特征10304维度降到了40维度。同时保存PCA的投影矩阵和均值。我们利用训练集计算的投影矩阵和均值，对测试集的每一张图(10304维度)降维到40维度。得到 $160 * 40$ 矩阵。160是图像的个数，40是维度。

问题2：人脸识别和匹配。

我们根据计算得到的训练集($240 * 40$)和测试集($160 * 40$)进行匹配。注意到训练集在这里将作为gallery库。

- 用欧式距离匹配：**我们用每个测试集图(query)的40维特征去对比gallery库中的40张图。计算哪张图欧氏距离最近，返回哪一张图。计算返回的图和该张图的ID是否一样。最终，我们计算160张图中，正确检索的图的个数(collect_num)。计算准确率 $\text{collect_num}/160$ 。
- 用马氏距离匹配：**类似1)，这里我们使用马氏距离，对训练每个ID求均值和协方差，假设每个ID的协方差都不一样。利用马氏距离归类于某个ID。

问题3：聚类分析。

聚类分析，我们利用k-means。在这一问，我们不分训练和测试，直接利用400张图进行聚类分析。我们分别利用降维前和降维后的特征进行聚类，并利用第四次作业的聚类评测标准进行评测。

实验原理

总结出的各操作的步骤：

pca步骤：

1. 原矩阵 $n * p$ ，计算它的协方差矩阵为 $p * p$ 。
2. 计算协方差矩阵的特征值特征向量，其中特征值是 $p * 1$ ，特征向量是 $p * p$ ，且每一列对应一个特征值。
3. 特征值中选最大的前 k 个，再选择对应的 k 个特征向量就是投影矩阵，大小为 $p * k$ 。
4. 原矩阵 $n * p$ 乘以投影矩阵 $p * k$ ，得到 $n * k$ 。

聚类分析步骤：

1. 选取初始类，此时均值为初始类中唯一元素的值。
2. 将所有图片分配到与之距离最接近的一类中，计算新类均值。
3. 直到本次迭代与上次迭代均值不变的时候，就停止。

实验步骤

PCA

将训练数据存储在矩阵中，存储之前需要用 `reshape` 方法将其变为行向量。

```
% 将训练数据存储在矩阵train_data中，有240行，代表240张图片，每行即每张图片表示成一个112*92维向量。
train_data = zeros(240,112*92);
att_faces = dir('./att_faces');
count = 1;
for i=4:43
    file = dir(['./att_faces/', att_faces(i).name]);
    for j=[3,5,6,7,8,9]
```

```

        temp = imread(['./att_faces/', att_faces(i).name, '/' ,
file(j).name]);
        train_data(count,:) = reshape(temp, [1,112*92]);
        count = count + 1;
    end
end

% 将测试数据存储在矩阵test_data中，有160行，代表160张图片，每行即每张图片表示成一个112*92维向量。
test_data = zeros(160,112*92);
att_faces = dir('./att_faces');
count = 1;
for i=4:43
    file = dir(['./att_faces/', att_faces(i).name]);
    for j=[10,11,12,4]
        temp = imread(['./att_faces/', att_faces(i).name, '/' ,
file(j).name]);
        test_data(count,:) = reshape(temp, [1,112*92]);
        count = count + 1;
    end
end
end

```

再做主成分分析，这里使用 `pca` 函数。返回值的含义如下注释所示：

```

% 主成分分析
% coeff为原矩阵的协方差矩阵的特征向量，特征值中选最大的k(=40)个，再选择对应的k列特征向量。
% 投影矩阵为coeff前k列。
% score为进行pca压缩后的数据。
% latent为从大到小排序的特征值。
% explained为每个主成分对应的贡献比例。
% mu为原矩阵每列的均值，取消中心化操作之后mu为0。
% [coeff,score,latent,tsquared,explained,mu] = pca(train_data,
'Centered', false);
[coeff,score,latent,tsquared,explained,mu] = pca(train_data);

% 投影矩阵
projection_matrix = coeff(:,1:40);
% 根据投影矩阵计算pca压缩后的结果
train_data_pca = train_data * projection_matrix;

```

% 利用训练集计算的投影矩阵和均值，对测试集的每一张图（10304维度）降维到40维度。得到160*40矩阵。160是图像的个数，40是维度。

```
test_data_pca = test_data * projection_matrix;
```

这里使用命令 `save projection_matrix` 和 `save mu` 保存PCA的投影矩阵和均值。

用欧式距离匹配

% 人脸识别与匹配

% 欧式距离

```
euclidean_distance = dist(train_data_pca, test_data_pca');
```

```
[~, euclidean_index] = min(euclidean_distance);
```

```
euclidean_collect_num = 0;
```

```
for i=1:160
```

```
    if strcmp(att_faces(ceil(euclidean_index(i)/6)+3).name,  
att_faces(ceil(i/4)+3).name)
```

```
        euclidean_collect_num = euclidean_collect_num + 1;
```

```
    end
```

```
end
```

```
euclidean_accuracy = euclidean_collect_num / 160;
```

```
disp(['The accuracy obtained by Euclidean distance matching method is  
, num2str(euclidean_accuracy*100), '%']]);
```

用马氏距离匹配

在这里需要注意，使用 `mahal` 函数时值 X 不能其列数不能超过行数，所以这里的PCA操作将10304维降成了40维。

% 人脸识别与匹配

% 马氏距离

% 把每个ID分为一类

```
train_data_mahal = zeros(6,4,40);
```

```
for i=1:40
```

```
    train_data_mahal(:, :, i) = train_data_pca((i-1)*6+1:i*6, :);
```

```
%    train_data_mahal(:, :, i) = train_data_mahal(:, :, i)';
```

```
end
```

% 计算每个ID的均值

```
mu_mahal = zeros(40,4);
```

```

for i=1:40
    mu_mahal(i,:) = mean(train_data_mahal(:,:,i));
end
% 计算马氏距离并分类
mahal_distance = zeros(160,40);
for i = 1:40
    mahal_distance(:,i) = mahal(test_data_pca,
train_data_mahal(:,:,i));
end
class_mahal = zeros(1,160);
for i=1:160
    [~,mahal_index] = min(mahal_distance(i,:));
    class_mahal(i) = mahal_index;
end
% 计算准确率
mahal_collect_num = 0;
for i=1:160
    if strcmp(att_faces(class_mahal(i)+3).name,
att_faces(ceil(i/4)+3).name)
        mahal_collect_num = mahal_collect_num + 1;
    end
end
mahal_accuracy = mahal_collect_num / 160;
disp(['The accuracy obtained by mahal distance matching method is ',
num2str(mahal_accuracy*100), '%']);

```

聚类分析

将所有图片存储在矩阵中，每行代表一张图片：

% 将所有图片存储在矩阵data中，有400行，代表400张图片，每行即每张图片表示成一个112*92维向量。

```
data = zeros(400,112*92);
att_faces = dir('./att_faces');
count = 1;
for i=4:43
    file = dir(['./att_faces/', att_faces(i).name]);
    for j=3:12
        temp = imread(['./att_faces/', att_faces(i).name, '/',
file(j).name]);
        data(count,:) = reshape(temp, [1,112*92]);
        count = count + 1;
    end
end
```

利用降维后的特征进行聚类：

```
% 主成分分析
% coeff为原矩阵的协方差矩阵的特征向量，特征值中选最大的k(=40)个，再选择对应的k列特征向量。
% 投影矩阵为coeff前k列。
% score为进行pca压缩后的数据。
% latent为从大到小排序的特征值。
% explained为每个主成分对应的贡献比例。
% mu为原矩阵每列的均值，取消中心化操作之后mu为0。
% [coeff,score,latent,tsquared,explained,mu] = pca(data, 'Centered', false);
[coeff,score,latent,tsquared,explained,mu] = pca(data);

% 投影矩阵
projection_matrix = coeff(:,1:40);
% 根据投影矩阵计算pca压缩后的结果
data_pca = data * projection_matrix;

%k-means(利用降维后的特征进行聚类)
disp('Clustering by Reduced Dimension Features:');
% 每张图的标签
labels = zeros(1,400);
for i = 1:400
    labels(i) = ceil(i/10);
end
```

```

% 选取初始类
cluster = zeros(40,40); % 每一行代表一个类
for i = 1:40
    cluster(i,:) = data_pca(10*(i-1)+1,:);
end
number_of_iterations = 0;
% 开始迭代
while 1
    % 计算每张图应该归于哪个类
    distance = zeros(40,400); % distance的每一列代表某张图与40个类之间的欧式距离
    for i = 1:40
        distance(i,:) = dist(cluster(i,:), data_pca');
    end
    cluster_belong_to = zeros(1,400);
    for i = 1:400
        [~,kmeans_index] = min(distance(:,i));
        cluster_belong_to(i) = kmeans_index;
    end
    % 算出新的类均值
    new_cluster = zeros(40,40);
    for i = 1:40
        temp = find(cluster_belong_to == i);
        for j = 1:size(temp, 2)
            new_cluster(i,:) = new_cluster(i,:) + data_pca(temp(j),:);
        end
        new_cluster(i,:) = new_cluster(i,:) / size(temp, 2);
    end
    % 给每个类标上标签
    labels_of_cluster = zeros(1,40);
    for i = 1:40
        G = zeros(1,40); % 记录每一个类中各个标签出现的次数
        temp = find(cluster_belong_to == i);
        for j = 1:size(temp, 2)
            G(labels(temp(j))) = G(labels(temp(j))) + 1;
        end
        [~,t] = max(G);
        labels_of_cluster(i) = t;
    end
    % 如果新算出的类均值和之前的不相等
    if ~isequal(cluster, new_cluster)
        cluster = new_cluster;
    end
end

```

```

        number_of_iterations = number_of_iterations + 1;
        disp(['Now the number of iterations is ',
num2str(number_of_iterations)]);
    % 如果新算出的类均值和之前的相等
    else
        % 计算准确率
        kmeans_collect_num_pca = 0;
        for i = 1:400
            n = cluster_belong_to(i);
            if labels_of_cluster(n) == labels(i)
                kmeans_collect_num_pca = kmeans_collect_num_pca + 1;
            end
        end
        kmeans_accuracy_pca = kmeans_collect_num_pca / 400;
        disp(['Iterative completed. The accuracy obtained by kmeans is
', num2str(kmeans_accuracy_pca*100), '%']);
        break;
    end
end
end

```

利用图片所有特征进行聚类：

```

%k-means(利用图片所有特征进行聚类)
disp('Clustering Using All Characters of Pictures:');
% 每张图的标签
labels = zeros(1,400);
for i = 1:400
    labels(i) = ceil(i/10);
end
% 选取初始类
cluster = zeros(40,112*92); % 每一行代表一个类
for i = 1:40
    cluster(i,:) = data(10*(i-1)+1,:);
end
number_of_iterations = 0;
% 开始迭代
while 1
    % 计算每张图应该归于哪个类
    distance = zeros(40,400); % distance的每一列代表某张图与40个类之间的欧
    氏距离
    for i = 1:40
        distance(i,:) = dist(cluster(i,:), data');
    end
end

```



```

end
cluster_belong_to = zeros(1,400);
for i = 1:400
    [~,kmeans_index] = min(distance(:,i));
    cluster_belong_to(i) = kmeans_index;
end
% 算出新的类均值
new_cluster = zeros(40,112*92);
for i = 1:40
    temp = find(cluster_belong_to == i);
    for j = 1:size(temp, 2)
        new_cluster(i,:) = new_cluster(i,:) + data(temp(j),:);
    end
    new_cluster(i,:) = new_cluster(i,:) / size(temp, 2);
end
% 给每个类标上标签
labels_of_cluster = zeros(1,40);
for i = 1:40
    G = zeros(1,40); % 记录每一个类中各个标签出现的次数
    temp = find(cluster_belong_to == i);
    for j = 1:size(temp, 2)
        G(labels(temp(j))) = G(labels(temp(j))) + 1;
    end
    [~,t] = max(G);
    labels_of_cluster(i) = t;
end
% 如果新算出的类均值和之前的不相等
if ~isequal(cluster, new_cluster)
    cluster = new_cluster;
    number_of_iterations = number_of_iterations + 1;
    disp(['Now the number of iterations is ',
num2str(number_of_iterations)]);
% 如果新算出的类均值和之前的相等
else
    % 计算准确率
    kmeans_collect_num = 0;
    for i = 1:400
        n = cluster_belong_to(i);
        if labels_of_cluster(n) == labels(i)
            kmeans_collect_num = kmeans_collect_num + 1;
        end
    end
end
end

```



```

        kmeans_accuracy = kmeans_collect_num / 400;
        disp(['Iterative completed. The accuracy obtained by kmeans is
', num2str(kmeans_accuracy*100), '%']);
        break;
    end
end

```

实验结果

PCA的投影矩阵和均值

 mu.mat	2019/1/17 21:37	Microsoft Acces...	26,165 KB
 projection_matrix.mat	2019/1/17 21:37	Microsoft Acces...	26,165 KB

用欧式距离匹配

```

>> face_euclidean_distance
The accuracy obtained by Euclidean distance matching method is 95.625%

```

用马氏距离匹配

```

>> face_mahal_distance
The accuracy obtained by mahal distance matching method is 56.875%

```

聚类分析

```

>> face_kmeans
Clustering by Reduced Dimension Features:
Now the number of iterations is 1
Now the number of iterations is 2
Now the number of iterations is 3
Now the number of iterations is 4
Now the number of iterations is 5
Iterative completed. The accuracy obtained by kmeans is 80.5%
Clustering Using All Characters of Pictures:
Now the number of iterations is 1
Now the number of iterations is 2
Now the number of iterations is 3
Now the number of iterations is 4
Iterative completed. The accuracy obtained by kmeans is 77.25%

```

参考文献

- <https://ww2.mathworks.cn/help/stats/pdist2.html?#d120e570044> pdist2
- <https://ww2.mathworks.cn/help/stats/mahal.html> mahal
- <https://ww2.mathworks.cn/help/stats/pca.html?requestedDomain=zh> pca
- https://blog.csdn.net/qq_29007291/article/details/54425356 matlab pca函数的使用方法
- <https://blog.csdn.net/VictoriaW/article/details/78399623> 主成分分析PCA
- <https://blog.csdn.net/carrie8899/article/details/8500088> matlab 找矩阵中每行或每列的最大值