# GMM算法的matlab程序

在“GMM算法的matlab程序（初步）”这篇文章中已经用matlab程序对iris数据库进行简单的实现，下面的程序最终的目的是求准确度。

作者：凯鲁嘎吉 - 博客园 http://www.cnblogs.com/kailugaji/

## 1.采用iris数据库

### iris_data.txt

```
5.1    3.5    1.4    0.2
4.9    3    1.4    0.2
4.7    3.2    1.3    0.2
4.6    3.1    1.5    0.2
5    3.6    1.4    0.2
5.4    3.9    1.7    0.4
4.6    3.4    1.4    0.3
5    3.4    1.5    0.2
4.4    2.9    1.4    0.2
4.9    3.1    1.5    0.1
5.4    3.7    1.5    0.2
4.8    3.4    1.6    0.2
4.8    3    1.4    0.1
4.3    3    1.1    0.1
5.8    4    1.2    0.2
5.7    4.4    1.5    0.4
5.4    3.9    1.3    0.4
5.1    3.5    1.4    0.3
5.7    3.8    1.7    0.3
5.1    3.8    1.5    0.3
5.4    3.4    1.7    0.2
5.1    3.7    1.5    0.4
4.6    3.6    1    0.2
5.1    3.3    1.7    0.5
4.8    3.4    1.9    0.2
5    3    1.6    0.2
5    3.4    1.6    0.4
5.2    3.5    1.5    0.2
5.2    3.4    1.4    0.2
4.7    3.2    1.6    0.2
4.8    3.1    1.6    0.2
5.4    3.4    1.5    0.4
```

5.2 4.1 1.5 0.1
5.5 4.2 1.4 0.2
4.9 3.1 1.5 0.2
5 3.2 1.2 0.2
5.5 3.5 1.3 0.2
4.9 3.6 1.4 0.1
4.4 3 1.3 0.2
5.1 3.4 1.5 0.2
5 3.5 1.3 0.3
4.5 2.3 1.3 0.3
4.4 3.2 1.3 0.2
5 3.5 1.6 0.6
5.1 3.8 1.9 0.4
4.8 3 1.4 0.3
5.1 3.8 1.6 0.2
4.6 3.2 1.4 0.2
5.3 3.7 1.5 0.2
5 3.3 1.4 0.2
7 3.2 4.7 1.4
6.4 3.2 4.5 1.5
6.9 3.1 4.9 1.5
5.5 2.3 4 1.3
6.5 2.8 4.6 1.5
5.7 2.8 4.5 1.3
6.3 3.3 4.7 1.6
4.9 2.4 3.3 1
6.6 2.9 4.6 1.3
5.2 2.7 3.9 1.4
5 2 3.5 1
5.9 3 4.2 1.5
6 2.2 4 1
6.1 2.9 4.7 1.4
5.6 2.9 3.6 1.3
6.7 3.1 4.4 1.4
5.6 3 4.5 1.5
5.8 2.7 4.1 1
6.2 2.2 4.5 1.5
5.6 2.5 3.9 1.1
5.9 3.2 4.8 1.8
6.1 2.8 4 1.3
6.3 2.5 4.9 1.5
6.1 2.8 4.7 1.2
6.4 2.9 4.3 1.3
6.6 3 4.4 1.4
6.8 2.8 4.8 1.4
6.7 3 5 1.7
6 2.9 4.5 1.5
5.7 2.6 3.5 1
5.5 2.4 3.8 1.1
5.5 2.4 3.7 1

5.8 2.7 3.9 1.2
6 2.7 5.1 1.6
5.4 3 4.5 1.5
6 3.4 4.5 1.6
6.7 3.1 4.7 1.5
6.3 2.3 4.4 1.3
5.6 3 4.1 1.3
5.5 2.5 4 1.3
5.5 2.6 4.4 1.2
6.1 3 4.6 1.4
5.8 2.6 4 1.2
5 2.3 3.3 1
5.6 2.7 4.2 1.3
5.7 3 4.2 1.2
5.7 2.9 4.2 1.3
6.2 2.9 4.3 1.3
5.1 2.5 3 1.1
5.7 2.8 4.1 1.3
6.3 3.3 6 2.5
5.8 2.7 5.1 1.9
7.1 3 5.9 2.1
6.3 2.9 5.6 1.8
6.5 3 5.8 2.2
7.6 3 6.6 2.1
4.9 2.5 4.5 1.7
7.3 2.9 6.3 1.8
6.7 2.5 5.8 1.8
7.2 3.6 6.1 2.5
6.5 3.2 5.1 2
6.4 2.7 5.3 1.9
6.8 3 5.5 2.1
5.7 2.5 5 2
5.8 2.8 5.1 2.4
6.4 3.2 5.3 2.3
6.5 3 5.5 1.8
7.7 3.8 6.7 2.2
7.7 2.6 6.9 2.3
6 2.2 5 1.5
6.9 3.2 5.7 2.3
5.6 2.8 4.9 2
7.7 2.8 6.7 2
6.3 2.7 4.9 1.8
6.7 3.3 5.7 2.1
7.2 3.2 6 1.8
6.2 2.8 4.8 1.8
6.1 3 4.9 1.8
6.4 2.8 5.6 2.1
7.2 3 5.8 1.6
7.4 2.8 6.1 1.9
7.9 3.8 6.4 2

```
6.4	2.8	5.6	2.2
6.3	2.8	5.1	1.5
6.1	2.6	5.6	1.4
7.7	3	6.1	2.3
6.3	3.4	5.6	2.4
6.4	3.1	5.5	1.8
6	3	4.8	1.8
6.9	3.1	5.4	2.1
6.7	3.1	5.6	2.4
6.9	3.1	5.1	2.3
5.8	2.7	5.1	1.9
6.8	3.2	5.9	2.3
6.7	3.3	5.7	2.5
6.7	3	5.2	2.3
6.3	2.5	5	1.9
6.5	3	5.2	2
6.2	3.4	5.4	2.3
5.9	3	5.1	1.8
```

View Code

## iris_id.txt

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2

2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
View Code

# 2.matlab程序

## My_GMM.m

```
function label_2=My_GMM(K)
%输入K：聚类数，K个单高斯模型
%输出label_2:聚的类，para_pi:单高斯权重，para_miu_new:高斯分布参数μ，para_sigma:高斯分布参数sigma
format long
eps=1e-15;  %定义迭代终止条件的eps
data=dlmread('E:\www.cnblogs.comkailugaji\data\iris\iris_data.txt');
%-------------------------------------------------------------------------------------------------
%对data做最大-最小归一化处理
[data_num,~]=size(data);
X=(data-ones(data_num,1)*min(data))./(ones(data_num,1)*(max(data)-min(data)));
[X_num,X_dim]=size(X);
para_sigma=zeros(X_dim,X_dim,K);
%-------------------------------------------------------------------------------------------------
%随机初始化K个聚类中心
rand_array=randperm(X_num);  %产生1~X_num之间整数的随机排列
```

```
center=X(rand_array(1:K),:);  %随机排列取前K个数，在X矩阵中取这K行作为初始聚类中心
%根据上述聚类中心初始化参数
para_miu_new=center;  %初始化参数miu
para_pi=ones(1,K)./K;  %K类单高斯模型的权重
for k=1:K
    para_sigma(:,:,k)=eye(X_dim);  %K类单高斯模型的协方差矩阵,初始化为单位阵
end
%欧氏距离，计算（X-para_miu）^2=X^2+para_miu^2-2*X*para_miu'，矩阵大小为X_num*K
distant=repmat(sum(X.*X,2),1,K)+repmat(sum(para_miu_new.*para_miu_new,2)',X_num,1)-2*X*para_miu_new';
%返回distant每行最小值所在的下标
[~,label_1]=min(distant,[],2);
for k=1:K
    X_k=X(label_1==k,:);  %X_k是一个（X_num/K, X_dim）的矩阵,把X矩阵分为K类
    para_pi(k)=size(X_k,1)/X_num;  %将（每一类数据的个数/X_num）作为para_pi的初始值
    para_sigma(:,:,k)=cov(X_k);  %para_sigma是一个（X_dim, X_dim）的矩阵，cov(矩阵)求的是每一列之间的协方差
end
%-------------------------------------------------------------------------------------------------
%EM算法
N_pdf=zeros(X_num,K);
while true
    para_miu=para_miu_new;
    %---------------------------------------------------------------------------------------------
    %E步
    %单高斯分布的概率密度函数N_pdf
    for k=1:K
        X_miu=X-repmat(para_miu(k,:),X_num,1);  %X-miu,(X_num, X_dim)的矩阵
        sigma_inv=inv(para_sigma(:,:,k));  %sigma的逆矩阵,(X_dim, X_dim)的矩阵//很可能出现奇异矩阵
        exp_up=sum((X_miu*sigma_inv).*X_miu,2);  %指数的幂，(X-miu)'*sigma^(-1)*(X-miu)
        coefficient=(2*pi)^(-X_dim/2)*sqrt(det(sigma_inv));  %高斯分布的概率密度函数e左边的系数
        N_pdf(:,k)=coefficient*exp(-0.5*exp_up);
    end
%    N_pdf=guass_pdf(X,K,para_miu,para_sigma);
    responsivity=N_pdf.*repmat(para_pi,X_num,1);  %响应度responsivity的分子，（X_num,K）的矩阵
    responsivity=responsivity./repmat(sum(responsivity,2),1,K);  %responsivity:在当前模型下第n个观测数据来自第k个分模型的概率，即分模型k对观测数据Xn的响应度
    %---------------------------------------------------------------------------------------------
    %M步
    R_k=sum(responsivity,1);  %(1,K)的矩阵，把responsivity每一列求和
    %更新参数miu
    para_miu_new=diag(1./R_k)*responsivity'*X;
    %更新k个参数sigma
    for i=1:K
        X_miu=X-repmat(para_miu_new(i,:),X_num,1);
        para_sigma(:,:,i)=(X_miu'*(diag(responsivity(:,i))*X_miu))/R_k(i);
    end
    %更新参数pi
    para_pi=R_k/sum(R_k);
    %---------------------------------------------------------------------------------------------
    %迭代终止条件
    if norm(para_miu_new-para_miu)<=eps
        break;
```

```
    end
end
%-------------------------------------------------------------------------------------------------------
%聚类
[~,label_2]=max(responsivity,[],2);
```

### succeed.m

```
function accuracy=succeed(K,id)
%输入K：聚的类，id：训练后的聚类结果，N*1的矩阵
N=size(id,1);   %样本个数
p=perms(1:K);   %全排列矩阵
p_col=size(p,1);   %全排列的行数
new_label=zeros(N,p_col);   %聚类结果的所有可能取值，N*p_col
num=zeros(1,p_col);  %与真实聚类结果一样的个数
real_label=dlmread('E:\www.cnblogs.comkailugaji\data\iris\iris_id.txt');
%将训练结果全排列为N*p_col的矩阵，每一列为一种可能性
for i=1:N
    for j=1:p_col
        for k=1:K
            if id(i)==k
                new_label(i,j)=p(j,k)-1;
            end
        end
    end
end
%与真实结果比对，计算精确度
for j=1:p_col
    for i=1:N
        if new_label(i,j)==real_label(i)
                num(j)=num(j)+1;
        end
    end
end
accuracy=max(num)/N;
```

# 3.结果

```
>> label_1=My_GMM(3);
>> accuracy=succeed(3,label_1)

accuracy =

   0.966666666666667
```

# 4.注意

GMM算法我只进行了一次计算准确度，因为有可能会出现奇异矩阵的情况，导致算法出错，现在我还没有想出如何解决奇异矩阵的问题，因此只给出了一次循环。望指正。

2020.7.30 奇异问题已初步解决，见评论链接。

补充：GMM的Python代码：upload/GMM.py at master · wl-lei/upload · GitHub

GMM的MATLAB代码：https://github.com/kailugaji/Gaussian_Mixture_Model_for_Clustering （注意！！！：我完善了GMM程序，比这篇博客的代码更加完善，放到了GitHub里面，进一步了解GMM代码请移步GitHub）