KFCM算法的matlab程序（用FCM初始化聚类中心）

在“聚类——KFCM”这篇文章中已经介绍了KFCM算法，现在用matlab程序对iris数据库进行实现，用FCM初始化聚类中心，并求其准确度与运行时间。

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

1.iris数据

iris.data

```
5.1,3.5,1.4,0.2,1
4.9,3.0,1.4,0.2,1
4.7,3.2,1.3,0.2,1
4.6,3.1,1.5,0.2,1
5.0,3.6,1.4,0.2,1
5.4,3.9,1.7,0.4,1
4.6,3.4,1.4,0.3,1
5.0,3.4,1.5,0.2,1
4.4,2.9,1.4,0.2,1
4.9,3.1,1.5,0.1,1
5.4,3.7,1.5,0.2,1
4.8,3.4,1.6,0.2,1
4.8,3.0,1.4,0.1,1
4.3,3.0,1.1,0.1,1
5.8,4.0,1.2,0.2,1
5.7,4.4,1.5,0.4,1
5.4,3.9,1.3,0.4,1
5.1,3.5,1.4,0.3,1
5.7,3.8,1.7,0.3,1
5.1,3.8,1.5,0.3,1
5.4,3.4,1.7,0.2,1
5.1,3.7,1.5,0.4,1
4.6,3.6,1.0,0.2,1
5.1,3.3,1.7,0.5,1
4.8,3.4,1.9,0.2,1
5.0,3.0,1.6,0.2,1
5.0,3.4,1.6,0.4,1
5.2,3.5,1.5,0.2,1
5.2,3.4,1.4,0.2,1
4.7,3.2,1.6,0.2,1
```

4.8,3.1,1.6,0.2,1
5.4,3.4,1.5,0.4,1
5.2,4.1,1.5,0.1,1
5.5,4.2,1.4,0.2,1
4.9,3.1,1.5,0.1,1
5.0,3.2,1.2,0.2,1
5.5,3.5,1.3,0.2,1
4.9,3.1,1.5,0.1,1
4.4,3.0,1.3,0.2,1
5.1,3.4,1.5,0.2,1
5.0,3.5,1.3,0.3,1
4.5,2.3,1.3,0.3,1
4.4,3.2,1.3,0.2,1
5.0,3.5,1.6,0.6,1
5.1,3.8,1.9,0.4,1
4.8,3.0,1.4,0.3,1
5.1,3.8,1.6,0.2,1
4.6,3.2,1.4,0.2,1
5.3,3.7,1.5,0.2,1
5.0,3.3,1.4,0.2,1
7.0,3.2,4.7,1.4,2
6.4,3.2,4.5,1.5,2
6.9,3.1,4.9,1.5,2
5.5,2.3,4.0,1.3,2
6.5,2.8,4.6,1.5,2
5.7,2.8,4.5,1.3,2
6.3,3.3,4.7,1.6,2
4.9,2.4,3.3,1.0,2
6.6,2.9,4.6,1.3,2
5.2,2.7,3.9,1.4,2
5.0,2.0,3.5,1.0,2
5.9,3.0,4.2,1.5,2
6.0,2.2,4.0,1.0,2
6.1,2.9,4.7,1.4,2
5.6,2.9,3.6,1.3,2
6.7,3.1,4.4,1.4,2
5.6,3.0,4.5,1.5,2
5.8,2.7,4.1,1.0,2
6.2,2.2,4.5,1.5,2
5.6,2.5,3.9,1.1,2
5.9,3.2,4.8,1.8,2
6.1,2.8,4.0,1.3,2
6.3,2.5,4.9,1.5,2
6.1,2.8,4.7,1.2,2
6.4,2.9,4.3,1.3,2
6.6,3.0,4.4,1.4,2
6.8,2.8,4.8,1.4,2

6.7,3.0,5.0,1.7,2
6.0,2.9,4.5,1.5,2
5.7,2.6,3.5,1.0,2
5.5,2.4,3.8,1.1,2
5.5,2.4,3.7,1.0,2
5.8,2.7,3.9,1.2,2
6.0,2.7,5.1,1.6,2
5.4,3.0,4.5,1.5,2
6.0,3.4,4.5,1.6,2
6.7,3.1,4.7,1.5,2
6.3,2.3,4.4,1.3,2
5.6,3.0,4.1,1.3,2
5.5,2.5,4.0,1.3,2
5.5,2.6,4.4,1.2,2
6.1,3.0,4.6,1.4,2
5.8,2.6,4.0,1.2,2
5.0,2.3,3.3,1.0,2
5.6,2.7,4.2,1.3,2
5.7,3.0,4.2,1.2,2
5.7,2.9,4.2,1.3,2
6.2,2.9,4.3,1.3,2
5.1,2.5,3.0,1.1,2
5.7,2.8,4.1,1.3,2
6.3,3.3,6.0,2.5,3
5.8,2.7,5.1,1.9,3
7.1,3.0,5.9,2.1,3
6.3,2.9,5.6,1.8,3
6.5,3.0,5.8,2.2,3
7.6,3.0,6.6,2.1,3
4.9,2.5,4.5,1.7,3
7.3,2.9,6.3,1.8,3
6.7,2.5,5.8,1.8,3
7.2,3.6,6.1,2.5,3
6.5,3.2,5.1,2.0,3
6.4,2.7,5.3,1.9,3
6.8,3.0,5.5,2.1,3
5.7,2.5,5.0,2.0,3
5.8,2.8,5.1,2.4,3
6.4,3.2,5.3,2.3,3
6.5,3.0,5.5,1.8,3
7.7,3.8,6.7,2.2,3
7.7,2.6,6.9,2.3,3
6.0,2.2,5.0,1.5,3
6.9,3.2,5.7,2.3,3
5.6,2.8,4.9,2.0,3
7.7,2.8,6.7,2.0,3
6.3,2.7,4.9,1.8,3

```
6.7,3.3,5.7,2.1,3
7.2,3.2,6.0,1.8,3
6.2,2.8,4.8,1.8,3
6.1,3.0,4.9,1.8,3
6.4,2.8,5.6,2.1,3
7.2,3.0,5.8,1.6,3
7.4,2.8,6.1,1.9,3
7.9,3.8,6.4,2.0,3
6.4,2.8,5.6,2.2,3
6.3,2.8,5.1,1.5,3
6.1,2.6,5.6,1.4,3
7.7,3.0,6.1,2.3,3
6.3,3.4,5.6,2.4,3
6.4,3.1,5.5,1.8,3
6.0,3.0,4.8,1.8,3
6.9,3.1,5.4,2.1,3
6.7,3.1,5.6,2.4,3
6.9,3.1,5.1,2.3,3
5.8,2.7,5.1,1.9,3
6.8,3.2,5.9,2.3,3
6.7,3.3,5.7,2.5,3
6.7,3.0,5.2,2.3,3
6.3,2.5,5.0,1.9,3
6.5,3.0,5.2,2.0,3
6.2,3.4,5.4,2.3,3
5.9,3.0,5.1,1.8,3
```

2.源程序

Eg_KFCM.m

```
function [ave_acc_KFCM,max_acc_FCM,min_acc_FCM,run_time]=Eg_KFCM(data,real_label,K)
%输入K:聚的类，max_iter是最大迭代次数,T:遗传算法最大迭代次数，n:种群个数
%输出ave_acc_KFCM：迭代max_iter次之后的平均准确度,iter:实际KFCM迭代次数
% data_load=dlmread('E:\www.cnblogs.com\kailugaji\database\iris.data');
% data=data_load(:,1:4);
% real_label=data_load(:,5);
t0=cputime;
max_iter=20;
s=0;
accuracy=zeros(max_iter,1);
%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)));
for i=1:max_iter
```

```
    %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
%     para_miu=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
    [~,para_miu,iter_FCM]=My_FCM2(X,K);
    [label_1,iter_KFCM]=My_KFCM(X,K,para_miu);
    accuracy(i)=succeed(real_label,K,label_1);
    s=s+accuracy(i);
    fprintf('第 %2d 次，FCM的迭代次数为：%2d，KFCM的迭代次数为：%2d，准确度为：%.8f\n', i, iter_FCM, iter_KFCM, accuracy(i));
end
ave_acc_KFCM=s/max_iter;
max_acc_FCM=max(accuracy);
min_acc_FCM=min(accuracy);
run_time=cputime-t0;
```

My_FCM2.m

```
function [label_1,para_miu_new,iter]=My_FCM2(X,K)
%输入K：聚类数
%输出：label_1:聚的类, para_miu_new:模糊聚类中心μ，responsivity:模糊隶属度
format long
eps=1e-5;  %定义迭代终止条件的eps
alpha=2;  %模糊加权指数，[1,+无穷)
T=100;  %最大迭代次数
fitness=zeros(T,1);
[X_num,X_dim]=size(X);
%------------------------------------------------------------------------------------------------------
%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
para_miu=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
responsivity=zeros(X_num,K);
R_up=zeros(X_num,K);
% ------------------------------------------------------------------------------------------------------
% FCM算法
for t=1:T
    %欧氏距离，计算（X-para_miu）^2=X^2+para_miu^2-2*para_miu*X'，矩阵大小为X_num*K
    distant=(sum(X.*X,2))*ones(1,K)+ones(X_num,1)*(sum(para_miu.*para_miu,2))'-2*X*para_miu';
    %更新隶属度矩阵X_num*K
    for i=1:X_num
        for j=1:K
            if distant(i,j)==1
                responsivity(i,j)=0;
            elseif distant(i,j)==0
                responsivity(i,j)=1./sum(responsivity(i,:)==0);
            else
                R_up(i,j)=distant(i,j).^(-1/(alpha-1));  %隶属度矩阵的分子部分
                responsivity(i,j)= R_up(i,j)./sum( R_up(i,:),2);
```

```
            end
        end
    end
    %目标函数值
    fitness(t)=sum(sum(distant.*(responsivity.^(alpha))));
     %更新聚类中心K*X_dim
    miu_up=(responsivity'.^(alpha))*X;  %μ的分子部分
    para_miu=miu_up./((sum(responsivity.^(alpha)))'*ones(1,X_dim));
    if t>1
        if abs(fitness(t)-fitness(t-1))<eps
            break;
        end
    end
end
para_miu_new=para_miu;
iter=t;  %实际迭代次数
[~,label_1]=max(responsivity,[],2);
```

My_KFCM.m

```
function [label_1,iter,fitness_min]=My_KFCM(X,K,para_miu)
%输入K：聚类数
%输出：label_1:聚的类, para_miu_new:模糊聚类中心μ，responsivity:模糊隶属度
format long
eps=1e-5;  %定义迭代终止条件的eps
alpha=2;  %模糊加权指数，[1,+无穷)
T=100;  %最大迭代次数
sigma_1=150;  %高斯核函数的参数
[X_num,X_dim]=size(X);
fitness=zeros(X_num,1);
responsivity=zeros(X_num,K);
R_up=zeros(X_num,K);
% ---------------------------------------------------------------------------------------------------
% KFCM算法
for t=1:T
    %欧氏距离，计算（X-para_miu）^2=X^2+para_miu^2-2*para_miu*X'，矩阵大小为X_num*K
    distant=(sum(X.*X,2))*ones(1,K)+ones(X_num,1)*(sum(para_miu.*para_miu,2))'-2*X*para_miu';
    %高斯核函数，X_num*K的矩阵
    kernel_fun=exp((-distant)./(2*sigma_1*sigma_1));
    %更新隶属度矩阵X_num*K
    for i=1:X_num
        for j=1:K
            if kernel_fun(i,j)==1
                responsivity(i,j)=0;
            else
                R_up(i,j)=(1-kernel_fun(i,j)).^(-1/(alpha-1));  %隶属度矩阵的分子部分
```

```
                responsivity(i,j)= R_up(i,j)./sum( R_up(i,:),2);
            end
        end
    end
    %目标函数值
    fitness(t)=2*sum(sum((ones(X_num,K)-kernel_fun).*(responsivity.^(alpha))));
     %更新聚类中心K*X_dim
    miu_up=(kernel_fun.*(responsivity.^(alpha)))'*X;  %μ的分子部分
    para_miu=miu_up./(sum(kernel_fun.*(responsivity.^(alpha)))'*ones(1,X_dim));
    if t>1
        if abs(fitness(t)-fitness(t-1))<eps
        %if norm(responsivity(t)-responsivity(t-1))<=eps
            break;
        end
    end
end
iter=t;  %实际迭代次数
[~,label_1]=max(responsivity,[],2);
fitness_min=fitness(iter);
```

succeed.m

```
function accuracy=succeed(real_label,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
%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);  %iris数据库，1 2 3
            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

```
>>  data_load=dlmread('E:\www.cnblogs.com\kailugaji\database\iris.data');
>>  data=data_load(:,1:4);
>> real_label=data_load(:,5);
>> [ave_acc_KFCM,max_acc_FCM,min_acc_FCM,run_time]=Eg_KFCM(data,real_label,3)
第  1 次，FCM的迭代次数为：24，KFCM的迭代次数为：  7，准确度为：0.88000000
第  2 次，FCM的迭代次数为：29，KFCM的迭代次数为：  6，准确度为：0.90666667
第  3 次，FCM的迭代次数为：23，KFCM的迭代次数为：  5，准确度为：0.88666667
第  4 次，FCM的迭代次数为：22，KFCM的迭代次数为：  5，准确度为：0.90666667
第  5 次，FCM的迭代次数为：24，KFCM的迭代次数为：  5，准确度为：0.90666667
第  6 次，FCM的迭代次数为：21，KFCM的迭代次数为：  4，准确度为：0.90000000
第  7 次，FCM的迭代次数为：20，KFCM的迭代次数为：  5，准确度为：0.90666667
第  8 次，FCM的迭代次数为：23，KFCM的迭代次数为：  4，准确度为：0.90000000
第  9 次，FCM的迭代次数为：24，KFCM的迭代次数为：  4，准确度为：0.90000000
第 10 次，FCM的迭代次数为：19，KFCM的迭代次数为：  5，准确度为：0.88666667
第 11 次，FCM的迭代次数为：23，KFCM的迭代次数为：  5，准确度为：0.88666667
第 12 次，FCM的迭代次数为：30，KFCM的迭代次数为：  5，准确度为：0.89333333
第 13 次，FCM的迭代次数为：30，KFCM的迭代次数为：  7，准确度为：0.88000000
第 14 次，FCM的迭代次数为：22，KFCM的迭代次数为：  5，准确度为：0.90666667
第 15 次，FCM的迭代次数为：23，KFCM的迭代次数为：  5，准确度为：0.90666667
第 16 次，FCM的迭代次数为：25，KFCM的迭代次数为：  7，准确度为：0.88000000
第 17 次，FCM的迭代次数为：14，KFCM的迭代次数为：  7，准确度为：0.88000000
第 18 次，FCM的迭代次数为：16，KFCM的迭代次数为：  7，准确度为：0.88000000
第 19 次，FCM的迭代次数为：22，KFCM的迭代次数为：  6，准确度为：0.90666667
第 20 次，FCM的迭代次数为：25，KFCM的迭代次数为：  7，准确度为：0.88000000

ave_acc_KFCM =

   0.894000000000000


max_acc_FCM =

   0.906666666666667


min_acc_FCM =

   0.880000000000000


run_time =
```

2.015625000000000

4.注意

这篇文章介绍KFCM的实现过程，用FCM初始化聚类中心，而不是随机初始化，性能比FCM好一些。如有不对之处，望指正。