**模式识别实验**

基于IRIS数据集的Kmeans和ISODATA聚类

3115315017

高榕

# 实验目的

1. 了解模式识别和数据挖掘中聚类的基本理论和知识；
2. 掌握Kmeans算法，并使用Kmeans对IRIS数据集进行聚类分类；
3. 掌握ISODATA算法，并使用ISODATA对IRIS数据集进行聚类分析；

# 实验原理

## Kmeans算法

1. 基本原理

该算法需要人工设置参数，即需要人工指定聚类的类别数C和初始的聚类中心K，然后按照最小距离原则将每个点分配到距离它最近的聚类中心Ki，不断计算各个聚类中心的平均半径，更新新的聚类中心K’，当迭代过程中聚类中心基本不再发生变化时，停止迭代，当前的聚类中心即为当前条件下的最佳聚类中心。

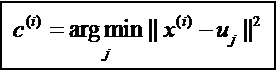
1. 基本输入

分类的类数C，初始的聚类中心K；

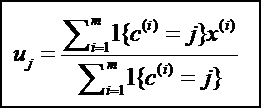
1. 基本输出

C个聚类重心K’；

1. 算法步骤
2. 随机选择C个点，作为初始的聚类中心；
3. 将每个点按照最小距离的原则划分到距离它最近的聚类中心Ki；
4. 计算每个聚类中心与各个点距离的平均值，更新新的聚类中心K’；
5. 重复2，3步骤，直到聚类中心基本不再发生变化时，这里的不再变化需要人工设置阈值；
6. 迭代结束，输出C个聚类中心K’；
7. 算法公式



求出所有数据和初始化随即数据的距离，然后找出距离最近的初始化随即数据



求出所有和这个初始数据距离最近的原始数据的均值

## ISODATA算法

1. 基本原理

ISODATA算法，即迭代自组织分析，通过设定初始参数而引入人机对话环节，并使用合并与分裂的机制，当某两类聚类中心的距离小于某一阈值时将它们合并为一类；当某一聚类的标准差大于某一设定阈值或其样本数目超过某一阈值时，将其分裂为两类；当某一类聚类的样本数目少于某一阈值时，将其取消。如此，根据初始聚类中心和设定的类别数目等参数进行迭代，最终得到一个比较理想的聚类分类结果。

1. 与Kmeans算法的不同点
2. ISODATA不是每调整一个样本的类别就重新计算一次各类样本的均值，而是在每次把全部样本都分类完毕之后再重新计算一次样本的均值，前者一般称为逐个样本修正法，后者成为成批样本修正法；
3. ISODATA算法不仅能够通过调整样本所属类别完成聚类分析，而且还能够自动的进行类别的“合并”和“分裂”，从而得到类数较为合理，分类较为明显的各个聚类结果。
4. 算法步骤

ISODATA算法的具体步骤如下：

首先介绍一下需要预选初值的各个参数的定义。

1. 步骤1(确定控制参数及设置初始聚类中心)

需确定的控制参数为，聚类期望数，一个聚类中的最少样本数，标准偏差控制参数，用于控制分裂，类间距离控制参数，用于控制合并，每次迭代允许合并的最大聚类对数，允许迭代的次数。设初始聚类数为及聚类中心。



1. 步骤2(分类)

对所有样本，按给定的个聚类中心，以最小距离进行分类，即若



Untitled%20Document.files/5_3006.gif

1. 步骤3(撤消类内样本数过小类别)

若有任何一个类，其样本数，则舍去，令，将 原样本分配至其它类。



1. 步骤4(更新均值向量)

按现有样本分类结果，调整均值参数

Untitled%20Document.files/5_3009.gif

1. 步骤5(计算类内平均距离)

每类中各样本离开均值的平均距离

Untitled%20Document.files/5_3010.gif

1. 步骤6(计算整个样本集偏离均值的平均距离)

Untitled%20Document.files/5_3011.gif

1. 步骤7(入口选择)

如这是最后一次迭代(取决于迭代上限)，则转步骤11，并设置，防止合并发生。



1. 如果，则转向步骤8，执行分裂步骤；



2. 如果，则转向步骤11，执行合并步骤；



3. 如果，当迭代次数是奇数时转至步骤8，迭代次数是偶数时转至步骤11。



1. 步骤8(求各类内各分类标准偏差)

对每个聚类，求其标准偏差



Untitled%20Document.files/5_3013.gif

Untitled%20Document.files/5_3014.gif

式中是类中第个样本的第分量，是的第个分量，是第个聚类第个分量的标准偏差，D是样本特征维数。



1. 步骤9(求每类具有最大标准偏差的分量)

Untitled%20Document.files/5_3018.gif

指每类具有最大标准偏差的分量。

1. 步骤10(分裂计算步骤)

若任一个有，并且有(a) 且，或有(b) ，则把分裂成两个聚类，其中心相应为与，把原来的取消，且令，由于与值设置不当将会导致影响到其它类别，因此与可按以下步骤计算：给定一值，



Untitled%20Document.files/5_3025.gif

Untitled%20Document.files/5_3026.gif

其中值应使中的样本到与的距离不同，但又应使中的样本仍然在分裂后的新样本类中。



1. 步骤11(计算类间聚类中心距离)

类与类的类间距离



Untitled%20Document.files/5_3027.gif

1. 步骤12(列出类间距离过近者)

比较与并将小于的按上升次序排列



Untitled%20Document.files/5_3029.gif

该队列最大个数是控制合并对数的参数



1. 步骤13(执行合并)

从类间距离最大的两类开始执行合并过程，此时需将与合并，得



Untitled%20Document.files/5_3032.gif

且，从第二个开始，则要检查其涉及类别是否已在前面合并过程中被合并，如两者并未被合并，则执行合并过程。



1. 步骤14(结束步骤)

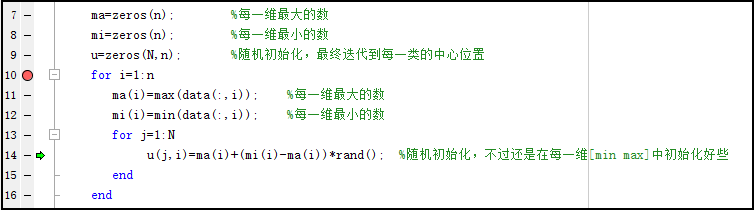
如是最后一次迭代则终止，否则可根据需要转步骤1或步骤2，转步骤1是为了更改控制数。迭代计数要加1。

# 实验过程及关键代码

## Kmeans算法

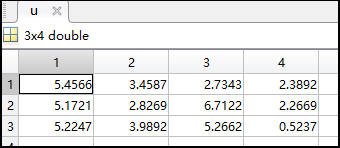
1. 算法输入
2. Kmeans的输入为所有IRIS数据集，共有150个数据样本，每一个样本是4维数据，原始数据中1-50号样本为第一类数据，51-100为第二类数据，101-150为第三类数据；
3. 聚类中心的数目C设为3；
4. 初始的聚类中心K，我们采用随机赋值的方法，在每一维数据的最大最小值间随机选取一个值，组合成为初始的聚类中心，并初始化3个初始聚类中心；

聚类中心初始化代码如下：



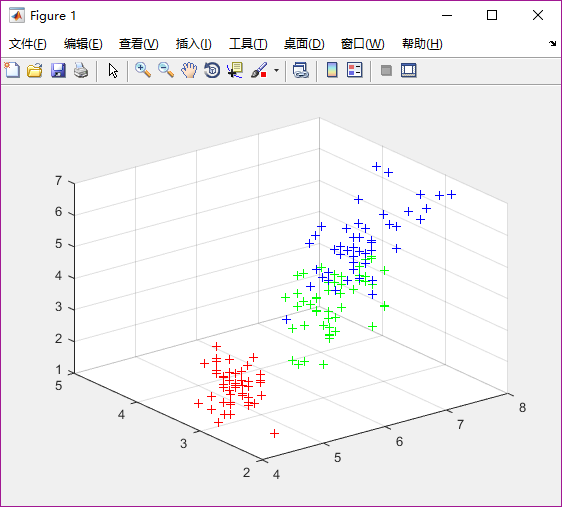
图示10-15行为初始随机聚类中心的过程，首先获取当前维度最大最小值，然后在最大最小值之间随机获取一个数据作为聚类中心分量，使用两个循环得到3个初始聚类中心，每个聚类中心4个维度

1. 算法结果
2. 第一次聚类：
3. 初始聚类中心K，我们通过在程序中设置断点的监督初始聚类中心的随机生成值；



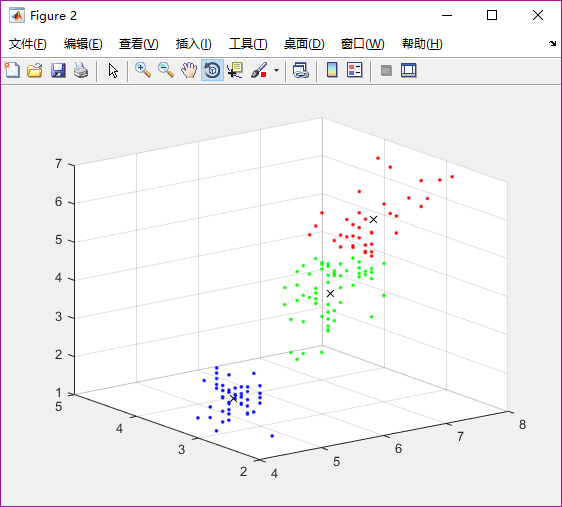
行数为聚类中心的数目，列数为每一个聚类中心的维度值

1. 初始数据三维图，因为IRIS数据是四维的，但是为了显示方便，我们选择数据中的前三个维度作为坐标生成初始数据图示；



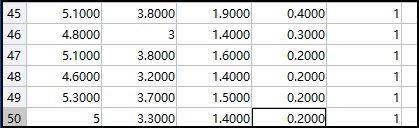
原始IRIS数据图示，由图可见第一类数据分别和第二类、第三类数据线性可分，第二类和第三类数据线性不可分。

1. 分类结果三维图，显示最终聚类分类结果；

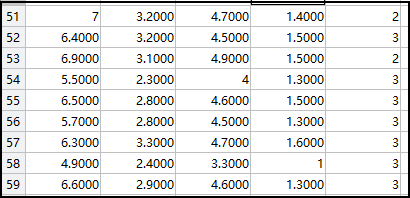


分类结果，图中叉号为最终聚类中心位置，与原始IRIS数据图进行比较可知，第二类和第三类中部分数据被分类错误，原本属于第三类的被划分到第二类中，同样的原本属于第二类的被划分到第三类中。

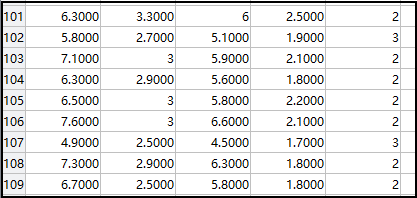
1. 量化的聚类结果；



第一类聚类结果完全正确，1-50号数据全部归类正确

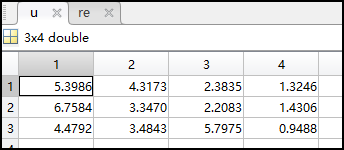


第三类数据聚类结果与第二类数据聚类结果有重叠，重叠部分是因为这两类数据是非线性的，聚类过程中发生了错误。



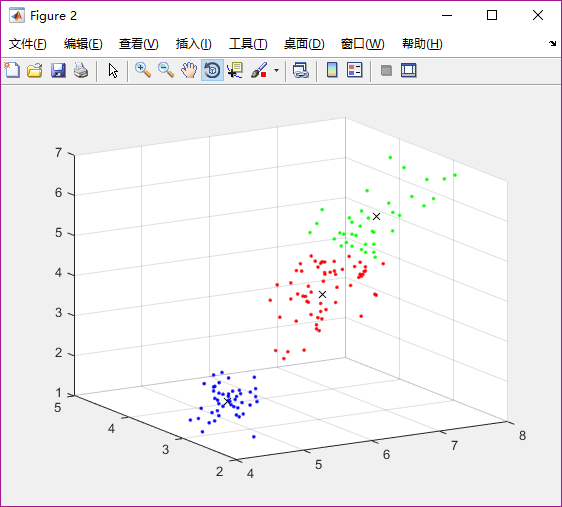
第二类数据聚类结果，显而易见也与第三类数据发生了误认。

1. 第二次聚类：
2. 聚类中心位置



三个聚类中心

1. 聚类结果



聚类结果

1. 实验结果分析

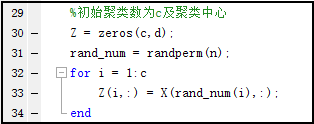
在利用Kmeans实现聚类分类过程中，以样本与聚类中心的的距离作为依据，寻找距离样本最近的聚类中心，并把样本归为这个聚类中心的元素，并重新计算当前每个聚类中心各自的所有的样本的距离均值，然后更新聚类中心坐标，直到聚类中心几乎不再发生改变。

根据Kmeans的算法过程可知，Kmeans算法本身对样本点的特征并没有进行区分，只是简单的利用它与聚类中心的距离，按照聚类半径归类的原则进行分类，因此，Kmeans只能将本身线性可分的数据区分，如果不同类数据之间有混淆，即非线性可分，该方法就不能很好的对数据进行聚类，错误率会较高，效果较差，所以提出了Kmeans的改进算法ISODATA。

## ISODATA算法

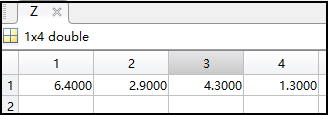
1. 算法输入
   1. ISODATA的输入为所有IRIS数据集，共有150个数据样本，每一个样本是4维数据，原始数据中1-50号样本为第一类数据，51-100为第二类数据，101-150为第三类数据；
   2. 聚类中心的数目C设为1，在程序实际执行过程中，聚类会根据实际情况“分裂”和“合并”，聚类中心的数目C会发生变化；
   3. 初始的聚类中心K，我们采用随机赋值的方法，在每一维数据的最大最小值间随机选取一个值，组合成为初始的聚类中心，并初始化3个初始聚类中心；

聚类中心初始化代码如下：



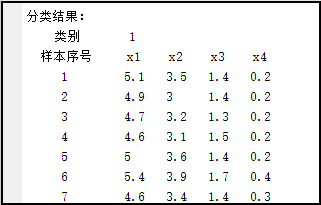
初始聚类中心我们从IRIS数据中抽取，在1-150范围内随机一个数字，然后就以该数行号的数据作为聚类中心

1. 算法结果
2. 第一次聚类
3. 聚类中心位置

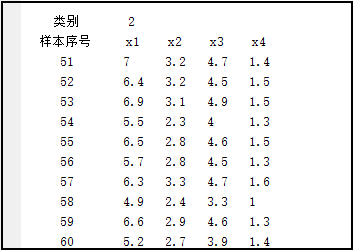
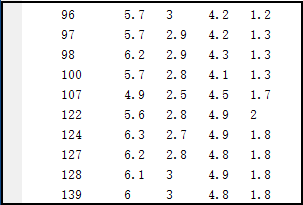


第一次聚类过程的聚类中心，程序随机选取的是第75个IRIS数据作为聚类中心

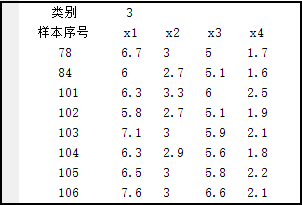
1. 聚类结果



第一类数据的聚类结果，样本序号为IRIS数据的序号，最终第一类数据全部分类正确。

第二类数据的聚类结果，有图可知在100序号之后的IRIS数据应该为第三类，但是被算法错误的归类为第二类数据，因此ISODATA也是线性聚类算法，并不能处理非线性数据

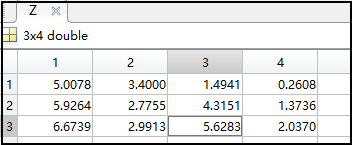


第三类数据的聚类结果，由图可知，有部分第二类数据错误的被归类为第三类数据，而错误的被归类到第二类数据中的第三类数据无法再被归类为第三类数据



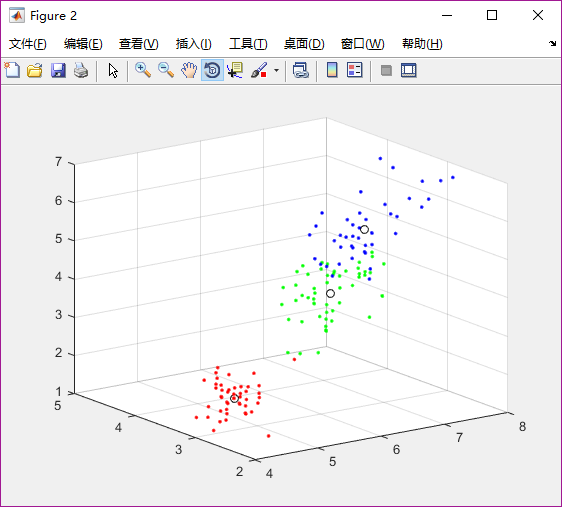
最终聚类分类错误率

1. 聚类中心最终值



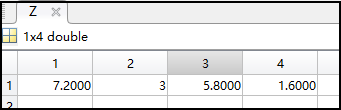
经过有限次迭代之后，程序最终自动分裂并找到的三个聚类中心

1. 聚类结果图示



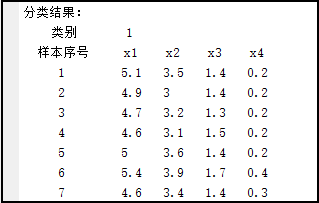
图示为最终聚类分类结果，图中黑色空心圆圈为聚类中心位置

1. 第二次聚类
2. 聚类中心位置

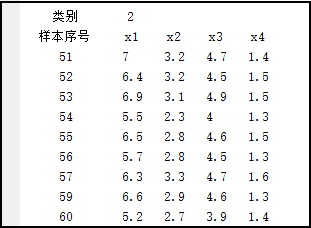
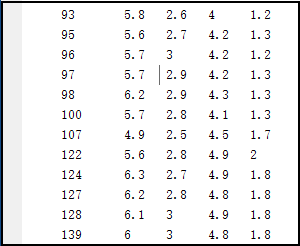


聚类中心位置见上图，程序随机选取的IRIS数据集中第130号数据

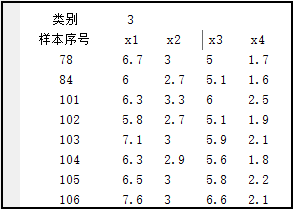
1. 聚类结果



第一类数据的聚类结果，样本序号为IRIS数据的序号，最终第一类数据全部分类正确。

第二类数据的聚类结果

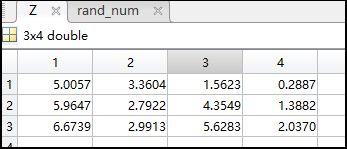


第三类数据的聚类结果



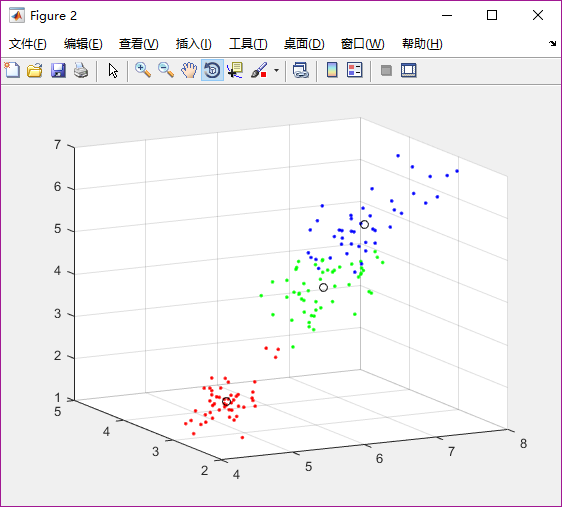
最终聚类分类错误率

1. 聚类中心最终值



经过有限次迭代之后，程序最终自动分裂并找到的三个聚类中心

1. 聚类结果图示



图示为最终聚类分类结果，图中黑色空心圆圈为聚类中心位置

1. 实验结果分析

ISODATA算法能够实现对数据类别的自动“分裂”和“合并”，根据程序事先设定好的阈值，作为“分裂合并”的判据，但最核心的聚类算法，仍然是采用最短距离原则，将以聚类中心为核（数据维数不同表现形式就不同），一定半径范围内的样本归为同一类，并不能解决非线性不可分的问题，因此第二类和第三类数据有错误分类的情况。

1. 参数分析
2. 初始聚类中心的不同，对聚类结果的影响

改变初始聚类中心的初始选取，是否导致收敛速度变慢。在原先的试验中，初始聚类中心的选择是随机的。比如设置初始有10类数据，那么初始的10个聚类中心可能是150个数据的任何一个。对照试验为，初始数据类数C为10不变，将初始聚类中心设为150个数据中的前10个数据。试验发现，改变初始聚类中心对试验的结果影响也不大。

故C与m{j}的初始条件对实验无影响。我想原因在于由于算法中存在数据类别的分裂与合并过程，无论初始条件如何，同类数据的性质不会改变，最后总会聚集在某一个中心附近。

1. 合并参数与标准偏差参数选取的不同，对对聚类结果的影响

合并参数OC与L影响的是聚类合并时的阈值。OC与L越大，在一次迭代中可能被合并的聚类数就越多。故选取必须十分的小心，过大的OC与L会导致在一次迭代中，有多个类被合并，会导致收敛过程变慢，最终聚类数的震荡，无法稳定在目标聚类数K。

标准偏差参数OS影响的是分裂操作，OS取的越大，分裂的阈值越高，也就有越少的聚类被分裂。如果OS取的过大或过小，都会有问题，过大引起聚类数目无法达到要求类数，过小会引起震荡。

但是一般来说，只使用一套参数对于分类问题是不够的。因为在聚类数目，聚类中心不断改变的过程中，最大标准差，聚类间距都是不断改变的。如果只使用一套OC，OS参数，那么最终会引起震荡问题。如下图所示：

图1：C=10，OS=0.5，OC=4 图2：C=10，OS=0.8，OC=4

由图可以看出最终聚类结果出现震荡现象，迭代次数均达到了设置的迭代上限。控制参数OS选择不同，且随机产生初始聚类中心，导致最终聚类数目不同。其原因由于在某本应该停止合并的一步迭代中，将某两类数据进行了合并（OC过大），而在下一步中，又将某一类进行分裂（OS过小），导致震荡。但是在面对一些未知类别数据的时候，一开始我们不可能取得合适的参数，这就需要等待程序运行若干次迭代后，观察程序的输出，并再对初始参数进行调整。

经过多次实验的参数调整，发现C=10，OS=1，OC=1.5这组参数可以得到很好的聚类数目收敛效果。如下图：

图3：C=10，OS=1，OC=1.5 图4：C=10，OS=1，OC=1.5

由图可以看出，由于初始的随机选取，导致每次收敛的次数不一样。但大多数迭代次数达到4时，聚类数目就开始收敛到3了，也就达到了本实验设定的期望类别数。聚类结果如下：

c =3

accuracy1 =

1 0 0

2 0 0

3 50 100

accuracy2 =

1 2 4

2 48 96

图5：C=150，OS=1，OC=1.5

accuracy3 =

1 36 72

2 14 28

第一类能100%的聚类到一块，而第二、三类由于数据特征相似程度较高，所以出现了聚类错误的情况，但大部分数据都能正确分类。

1. 每次允许合并的最大聚类对数

每次允许合并的最大聚类数L显然影响的是聚类收敛的快慢。如果取的小，则需要更多的迭代次数来达到目标聚类数。

下图为L=1与L=5的对比(纵坐标为c)



图6：C=10，OS=1，OC=1.5，L=1 图7：C=10，OS=1，OC=1.5，L=5

由图可知，当L=1时收敛需要的迭代次数大于L=5时的迭代次数，也就是说每次允许合并的最大聚类数L越大，聚类数目收敛速度越快。两组实验的最终聚类数目和三类数据的聚类正确率差别不大。

1. 初始聚类中心数目不同对收敛速度的影响

初始聚类中心数目的取定对最后类别数目的收敛速度有影响，当初始数目较多时，收敛速度会变慢。但收敛代数最多也不会超过60代。我认为，这与ISODATA方法本身的特性有关，这是一种动态聚类方法，而我们所用来测试的IRIS数据，原本就非常明显的分为3个类别，这样，在刚开始迭代次数不多时，各类数据已经聚集在属于自己类别的聚类中心了

以合适的参数（OS=1，OC=1.5），初始聚类数分别为C=10，C=50，C=150的结果分别如下：



图8：C=10，迭代次数为4 图9：C=50，迭代次数为14



图10：C=100，迭代次数为26 图11：C=150，迭代次数为34

# 实验代码

## Kmeans代码

1. //Kmeans\_main
2. clear all;
3. close all;
4. clc;
6. data = load('iris\_data.txt');
7. iris\_1 = data(1:50,:);
8. iris\_2 = data(51:100,:);
9. iris\_3 = data(101:150,:);
11. plot3(iris\_1(:,1),iris\_1(:,2),iris\_1(:,3),'r+');
12. hold on;
13. plot3(iris\_2(:,1),iris\_2(:,2),iris\_2(:,3),'g+');
14. plot3(iris\_3(:,1),iris\_3(:,2),iris\_3(:,3),'b+');
15. grid on;
17. [u, re] = KMeans(data,3);
18. [m, n] = size(re);
20. figure;
21. hold on;
22. for i = 1: m
23. if re(i,5) == 1
24. plot3(re(i,1),re(i,2),re(i,3),'b.');
25. elseif re(i,5) == 2
26. plot3(re(i,1),re(i,2),re(i,3),'r.');
27. else
28. plot3(re(i,1),re(i,2),re(i,3),'g.');
29. end
30. end
31. for i = 1: 3
32. plot3(u(i,1),u(i,2),u(i,3),'kx');
33. end
34. hold off;
35. grid on;
36. //Kmeans.function
37. function [u, re]=KMeans(data,N)
38. [m,n]=size(data);
39. ma=zeros(n);
40. mi=zeros(n);
41. u=zeros(N,n);
42. for i=1:n
43. ma(i)=max(data(:,i));
44. mi(i)=min(data(:,i));
45. for j=1:N
46. u(j,i)=ma(i)+(mi(i)-ma(i))\*rand();
47. end
48. end
50. while 1
51. pre\_u=u;
52. for i=1:N
53. tmp{i}=[];
54. for j=1:m
55. tmp{i}=[tmp{i};data(j,:)-u(i,:)];
56. end
57. end
59. quan=zeros(m,N);
60. for i=1:m
61. c=[];
62. for j=1:N
63. c=[c norm(tmp{j}(i,:))];
64. end
65. [junk index]=min(c);
66. quan(i,index)=norm(tmp{index}(i,:));
67. end
69. for i=1:N
70. for j=1:n
71. u(i,j)=sum(quan(:,i).\*data(:,j))/sum(quan(:,i));
72. end
73. end
75. if norm(pre\_u-u)<0.1
76. break;
77. end
78. end
80. re=[];
81. for i=1:m
82. tmp=[];
83. for j=1:N
84. tmp=[tmp norm(data(i,:)-u(j,:))];
85. end
86. [junk index]=min(tmp);
87. re=[re;data(i,:) index];
88. end
90. end

## ISODATA代码

1. clear all;
2. close all;
3. clc;
5. X = load('iris\_data.txt');
6. iris\_1 = X(1:50,:);
7. iris\_2 = X(51:100,:);
8. iris\_3 = X(101:150,:);
9. X=X(:,1:4);
10. [n,d]=size(X);
12. plot3(iris\_1(:,1),iris\_1(:,2),iris\_1(:,3),'r+');
13. hold on;
14. plot3(iris\_2(:,1),iris\_2(:,2),iris\_2(:,3),'g+');
15. plot3(iris\_3(:,1),iris\_3(:,2),iris\_3(:,3),'b+');
16. grid on;
18. c = 1;
19. K = 3;
20. Thita\_n = 20;
21. Thita\_s = 0.5;
22. Thita\_c = 2;
23. L = 0;
24. I = 10;
25. J = 1;
27. Z = zeros(c,d);
28. rand\_num = randperm(n);
29. for i = 1:c
30. Z(i,:) = X(rand\_num(i),:);
31. end
33. w=zeros(n,2);
34. for i=1:n
35. w(i,1)=i;
36. end
38. step2 = true;
40. while 1
41. while step2
42. N = zeros(c,2);
43. for i = 1:c
44. N(i,1) = i;
45. end
47. for i = 1:n
48. x = X(i,:);
49. for j = 1:c
50. y = Z(j,:);
51. dist=sqrt(sum((x-y).^2));
52. if j == 1
53. mini = dist;
54. min\_id = 1;
55. else
56. if dist < mini
57. mini = dist;
58. min\_id = j;
59. end
60. end
61. end
62. w(i,2) = min\_id;
63. N(min\_id,2) = N(min\_id,2) + 1;
64. end
65. k = c;
66. step2 = false;
67. for i = 1:c
68. if N(i,2) < Thita\_n
69. k = k - 1;
70. step2 = true;
71. end
72. end
73. c = k;
74. end
76. Z = zeros(c,d);
77. for i = 1:n
78. id = w(i,2);
79. Z(id,:) = Z(id,:) + X(i,:);
80. end
82. for i = 1:c
83. if N(i,2)~= 0
84. Z(i,:) = Z(i,:) / N(i,2);
85. end
86. end
88. D = zeros(c,2);
89. for i = 1:c
90. D(i,1) = i;
91. end
92. for i = 1:n
93. id = w(i,2);
94. x = X(i,:);
95. y = Z(id,:);
96. dist = sqrt(sum((x-y).^2));
97. D(id,2) = D(id,2) + dist;
98. end
100. for i = 1:c
101. if N(i,2)~= 0
102. D(i,2) = D(i,2) / N(i,2);
103. end
104. end
106. D1 = 0;
107. for i = 1:c
108. D1 = D1 + N(i,2) \* D(i,2);
109. end
110. D1 = D1 / n;
112. step8 = false;
113. step11 = false;
114. if J < I
115. if c <= K/2
116. step8 = true;
117. else
118. if mod(J,2) == 0 || c >= 2\*K
119. step11 = true;
120. else
121. step8 = true;
122. end
123. end
124. else
125. Thita\_c = 0;
126. step11 = true;
127. end
129. if step8
130. xgm=zeros(c,d);
131. for i = 1:n
132. id = w(i,2);
133. x = X(i,:);
134. y = Z(id,:);
135. xgm(id,:) = xgm(id,:)+(x-y).^2;
136. end
137. for i = 1:c
138. if N(i,2) ~= 0
139. xgm(i,:) = sqrt(xgm(i,:) / N(i,2));
140. end
141. end
143. xgm\_max = zeros(c,1);
144. max\_id = zeros(c,1);
145. for i = 1:c
146. [xgm\_max(i),max\_id(i)] = max(xgm(i,:));
147. end
149. for i = 1:c
150. if xgm\_max(i) > Thita\_s
151. if(D(i,2)>D1 && N(i,2)>2\*(Thita\_n+1)) || c<=K/2
152. rand\_k = rand;
153. if rand\_k == 0
154. rand\_k = 0.95;
155. end
156. r = zeros(1,d);
157. r(max\_id(i)) = rand\_k \* xgm\_max(i);
158. ZZ = Z;
159. Z = zeros(c+1,d);
160. for j = 1:c+1
161. if j < i
162. Z(j,:) = ZZ(j,:);
163. else
164. if j == i
165. Z(j,:) = ZZ(j,:) - r;
166. Z(j+1,:) = ZZ(j,:) + r;
167. else
168. if j >= i + 2
169. Z(j,:) = ZZ(j-1,:);
170. end
171. end
172. end
173. end
174. c = c + 1;
175. J = J + 1;
176. step2 = true;
177. step11 = false;
178. else
179. step11 = true;
180. end
181. else
182. step11 = true;
183. end
184. end
186. end
188. if step11
189. D = zeros(c,c);
190. add = true;
191. i = 1;
192. while i <= c
193. for j = i+1:c
194. x=Z(i,:);
195. y=Z(j,:);
196. D(i,j) = sqrt(sum((x-y).^2));
197. if D(i,j) < Thita\_c
198. ZZ = Z;
199. Z = zeros(c-1,d);
200. for id = 1:c-1
201. if id < min(i,j)
202. Z(id,:) = ZZ(id,:);
203. end
204. if id == min(i,j)
205. Z(id,:) = (N(i,2)\*ZZ(i,:)+N(j,2)\*ZZ(j,:)) /(N(i,2)+N(j,2));
206. end
207. if id > min(i,j) && id < max(i,j)
208. Z(id,:) = ZZ(id,:);
209. end
210. if id >= max(i,j)
211. Z(id,:) = ZZ(id+1,:);
212. end
213. end
214. c = c - 1;
215. add = false;
216. break;
217. end
218. end
219. if add
220. i = i + 1;
221. else
222. add = true;
223. i = 1;
224. end
225. end
227. if J < I
228. J = J + 1;
229. step2 = true;
230. else
231. break;
232. end
233. end
235. end
237. figure;
238. hold on;
239. grid on;
240. disp('·ÖÀà½á¹û£º');
241. dsp1='';
242. for i=1:c
243. dsp2='';
244. shown=false;
245. for j=1:d
246. if j >= d/2 && shown == false
247. shown=true;
248. dsp1=[' ',num2str(i),' '];
249. else
250. dsp1=[dsp1,' '];
251. end
253. dsp2=[dsp2,'x',num2str(j),' '];
254. end
255. disp([' Àà±ð ',dsp1]);
256. disp([' Ñù±¾ÐòºÅ ',dsp2]);
257. for idx=1:n
258. if i==w(idx,2)
259. dsp3='';
260. for idx2=1:d
261. temp=X(idx,idx2);
262. numspace=6-length(num2str(temp));
263. spaces='';
264. for j=1:numspace
265. spaces=[spaces,' '];
266. end
267. dsp3=[dsp3,num2str(X(idx,idx2)),spaces];
268. end
269. numspace=9-length(num2str(idx));
270. spaces='';
271. for j=1:numspace
272. spaces=[spaces,' '];
273. end
274. disp([' ',num2str(idx),spaces,dsp3]);
275. if i == 1
276. plot3(X(idx,1),X(idx,2),X(idx,3),'r.');
277. elseif i == 2
278. plot3(X(idx,1),X(idx,2),X(idx,3),'g.');
279. elseif i == 3
280. plot3(X(idx,1),X(idx,2),X(idx,3),'b.');
281. end
282. end
283. end
284. disp(' ');
285. end
287. sum = 0;
288. for i = 1:n
289. if i <= 50
290. if w(i,2) ~= 1
291. sum = sum + 1;
292. end
293. else
294. if i >50 && i <=100
295. if w(i,2) ~= 2
296. sum = sum + 1;
297. end
298. else
299. if w(i,2) ~= 3
300. sum = sum + 1;
301. end
302. end
303. end
304. end
306. for i = 1: c
307. plot3(Z(i,1),Z(i,2),Z(i,3),'ko');
308. end
309. error = sum / n