****

模式识别

聚类分析实验报告

|  |  |
| --- | --- |
| 姓名： | 黄勉 |
| 学号： | 57118304 |

东南大学网络空间安全学院

School of Cyber Science & Engineering

Southeast University

2020年12月

### 一．问题描述

试用K—均值法对如下模式分布进行聚类分析。编程实现，编程语言不限。

{X1(0,0), X2(3,8), X3(2,2), X4(1,1), X5(5,3),

X6(4,8), X7(6,3), X8(5,4), X9(6,4), X10(7,5)}

### 算法描述

首先从n个数据对象任意选择 k 个对象作为初始聚类中心；而对于所剩下其它对象，则根据它们与这些聚类中心的相似度（距离），分别将它们分配给与其最相似的（聚类中心所代表的）聚类；然后再计算每个所获新聚类的聚类中心（该聚类中所有对象的均值）；不断重复这一过程直到标准测度函数开始收敛为止。一般都采用均方差作为标准测度函数..k个聚类具有以下特点：各聚类本身尽可能的紧凑，而各聚类之间尽可能的分开。

### 计算过程

1. 首先用ClusterCenter（）函数选取3个聚类中心
2. 将各样本按最小距离原则分配到3类的某一类，在代码中用Distance（）函数比较样本点与每个聚类中心的距离大小
3. 用NewClusterCenter（）函数计算新的聚类中心
4. 用Iteration( )函数进行迭代，设置flag位对聚类中心是否与前一次一致进行判断。迭代至两次聚类中心不变为止。

### 代码实现

#include<iostream>

#include<cmath>

using namespace std;

#define M 10

//初始化模式样本

int X[10][2]= {{0,0},{3,8},{2,2},{1,1},{5,3},{4,8},{6,3},{5,4},{6,4},{7,5}};

float Z[3][2];//该数组用于存放聚类中心

float preZ1[M][2];//存储上一轮的聚类中心

float preZ2[M][2];

float preZ3[M][2];

int k;//求出与聚类中心的最小距离，将该样本分至相应聚类中心

//初始化三个聚类

int S1[M][2];int s1=0;

int S2[M][2];int s2=0;

int S3[M][2];int s3=0;

float x,y;//新聚类中心坐标

int turn=0;

//初始化聚类中心

void ClusterCenter()

{

for(int i=0;i<3;i++){

for(int j=0;j<2;j++){

Z[i][j]=X[i][j];

//cout<<Z[i][j];

}

}

}

//计算点到每个聚类中心的最大值

int Distance(int a[1][2],float b[3][2])

{

//float dmin;

float d[3];

for(k=0;k<3;k++){

d[k]=sqrt((a[0][0]-b[k][0])\*(a[0][0]-b[k][0])+(a[0][1]-b[k][1])\*(a[0][1]-b[k][1]));

}

if((d[0] <= d[1]) && (d[0] <= d[2]))

// dmin=d[0];

k=0;

else if ((d[1] <= d[0]) && (d[1] <= d[2]))

// dmin=d[1];

k=1;

else

// dmin=d[2];

k=2;

return k;

}

//计算新的聚类中心

void NewClusterCenter(int s,int a[M][2])

{

int sum1=0,sum2=0;

for(int i=0;i<s;i++){

sum1+=a[i][0];

sum2+=a[i][1];

//cout<<a[i][0]<<endl;

//cout<<a[i][1]<<endl;

}

//cout<<sum1<<endl;

//cout<<sum2<<endl;

cout<<"样本个数"<<s<<endl;

x=(float)sum1/s;

y=(float)sum2/s;

cout<<"新聚类中心坐标为"<<endl;

cout<<x<<endl;

cout<<y<<endl;

}

//迭代

void Iteration()

{

s1=0;

s2=0;

s3=0;

cout<<"这是第"<<turn+1<<"轮迭代"<<endl;

int flag=0;

for(int i=0;i<10;i++){

int temp[1][2];

for(int j=0;j<2;j++){

temp[0][j]=X[i][j];

// cout<<temp[0][j];

}

Distance(temp,Z);

cout<<"将样本（";

if(k==0){

for(int j=0;j<2;j++){

S1[s1][j]=X[i][j];

cout<<S1[s1][j]<<" ";

}

s1++;

}

else if(k==1){

for(int j=0;j<2;j++){

S2[s2][j]=X[i][j];

cout<<S2[s2][j]<<" ";

}

s2++;

}

else{

for(int j=0;j<2;j++){

S3[s3][j]=X[i][j];

cout<<S3[s3][j]<<" ";

}

s3++;

}

cout<<"）分至第"<<k+1<<"类"<<endl;

}

//cout<<s1<<" "<<s2<<" "<<s3<<endl;

/\* for(int i=0;i<s1;i++){

for(int j=0;j<2;j++){

cout<<S1[i][j];

}

cout<<endl;

}\*/

cout<<endl;

cout<<"第一类";

NewClusterCenter(s1,S1);

Z[0][0]=x;

Z[0][1]=y;

cout<<endl;

cout<<"第二类";

NewClusterCenter(s2,S2);

Z[1][0]=x;

Z[1][1]=y;

cout<<endl;

cout<<"第三类";

NewClusterCenter(s3,S3);

Z[2][0]=x;

Z[2][1]=y;

cout<<endl;

for(int j=0;j<2;j++){

preZ1[turn][j]=Z[0][j];

preZ2[turn][j]=Z[1][j];

preZ3[turn][j]=Z[2][j];

// cout<<preZ1[turn][j]<<endl;

// cout<<preZ2[turn][j]<<endl;

// cout<<preZ3[turn][j]<<endl;

}

turn++;

if(turn==1){

Iteration();

}

else{

if(preZ1[turn-1][0]!=preZ1[turn-2][0]||preZ1[turn-1][1]!=preZ1[turn-2][1])

{flag++;

//cout<<"flag1"<<flag<<endl;

}

else if(preZ2[turn-1][0]!=preZ2[turn-2][0]||preZ2[turn-1][1]!=preZ2[turn-2][1])

{flag++;

//cout<<"flag2"<<flag<<endl;

}

else if(preZ3[turn-1][0]!=preZ3[turn-2][0]||preZ3[turn-1][1]!=preZ3[turn-2][1])

{flag++;

//cout<<"flag3"<<flag<<endl;

}

else

flag=0;

//cout<<"flag"<<flag<<endl;

if(flag>0){

Iteration();

}

else{

cout<<"最终聚类中心分别为："<<endl;

for(int i=0;i<3;i++){

cout<<"Z"<<i+1<<":(";

for(int j=0;j<2;j++){

cout<<Z[i][j]<<" ";

}

cout<<")"<<endl;

}

cout<<"以下样本属于第一类";

for(int i=0;i<s1;i++){

cout<<"(";

for(int j=0;j<2;j++){

cout<<S1[i][j]<<" ";

}

cout<<")";

}

cout<<endl;

cout<<"以下样本属于第二类";

for(int i=0;i<s2;i++){

cout<<"(";

for(int j=0;j<2;j++){

cout<<S2[i][j]<<" ";

}

cout<<")";

}

cout<<endl;

cout<<"以下样本属于第三类";

for(int i=0;i<s3;i++){

cout<<"(";

for(int j=0;j<2;j++){

cout<<S3[i][j]<<" ";

}

cout<<")";

}

}

}

}

int main()

{

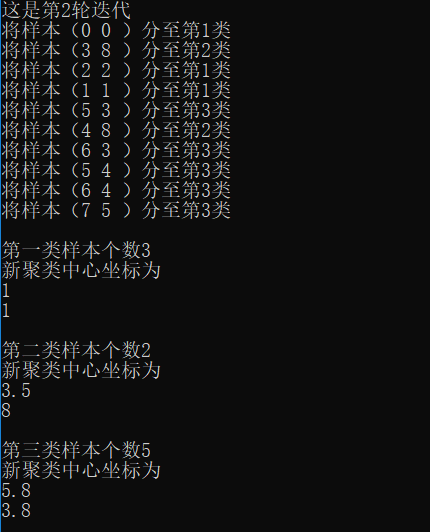
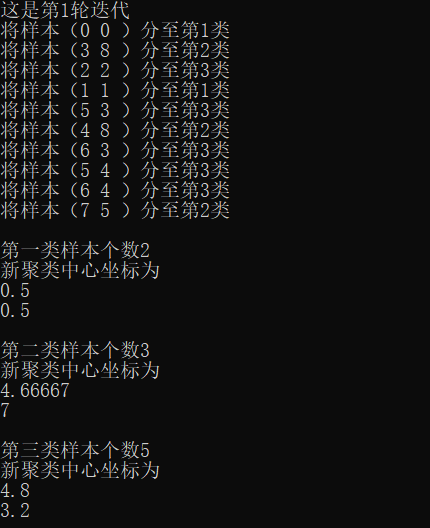
ClusterCenter();

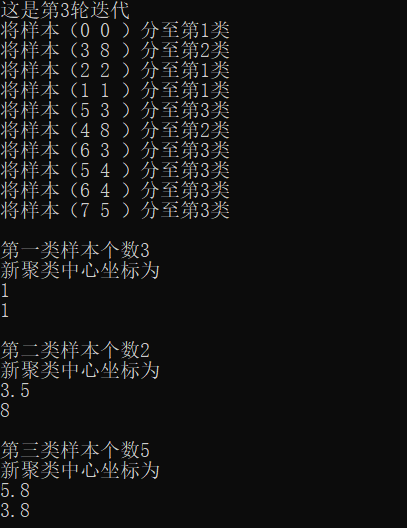
Iteration();

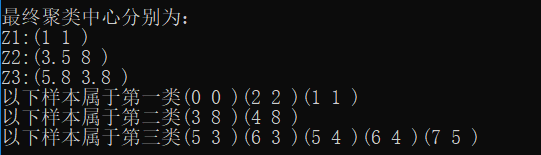
return 0;

}

### 结果分析







k-means算法是一种基于样本间相似性度量的间接聚类方法，属于非监督学习方法。实际应用中，需要不断试探K值与聚类中心的起始值。