**实验四 K-means算法和KNN算法**

**一、实验目的**

1. 掌握K-means算法的基本原理和步骤。
2. 熟练使用MATLAB软件编写K-means算法程序。
3. 利用K-means算法解决数据聚类问题。
4. 利用KNN算法解决数据分类问题

**二、实验原理**

K-means是最为常用的聚类算法，该算法是将相似的样本归置在一起的一种无监督算法。采用距离作为相似性的评价指标，即认为两个对象的距离越近，其相似度就越大。

算法主要步骤可描述如下：

1. 随机产生K个初始聚类中心。
2. 计算测试点到聚类中心的距离，选择距离最近的聚类中心将测试点归类。
3. 更新每类的聚类中心。
4. 重复步骤2、3迭代更新，直至聚类中心不再改变，或者新的聚类中心与前一步聚类中心的距离小于某个值。

KNN算法是最简单的分类算法，如果一个样本在特征空间中的k个最相似(即特征空间中最邻近)的样本中的大多数属于某一个类别，则该样本也属于这个类别。算法主要步骤可描述如下：

1、计算已知类别数据集中的点与当前点之间的距离；

2、按照距离递增依次排序；

3、选取与当前点距离最小的k个点

4、确定k个点在所在类别的出现频率

5、返回k个点出现频率最高的类别作为当前点的预测分类

**三、实验内容**

一）、聚类

1. 数据自行产生和采用给定数据。

2. 根据需要进行数据的归一化。

3. 确定初始K个聚类中心。

1. 编写程序，完成聚类和聚类中心更新。
2. 确定算法终止条件。
3. 达到迭代要求，显示聚类中心和每个样本的聚类结果。

二）、分类

1 、读取分类数据。

1. 数据归一化。
2. 计算测试样本距离训练样本的距离。
3. 从小到大排序。
4. 取出前K个；
5. 统计前K个最邻近样本的分类号；
6. 把分类号最多的赋给测试样本；
7. 统计正确率。

**四、预习要求**

1. 认真阅读教材中K-means和KNN算法的基本原理与步骤。
2. 根据要求完成数据预处理和初始化。
3. 编写算法迭代选择程序。

**五、实验要求**

1. K值、初始聚类中心、逼近误差、距离测度自选，绘制流程图。

2. 上机编写程序并调试运行程序。

3. 对比K值对算法结果的影响。

4. 对比初始聚类中心对算法结果的影响。

5. 绘制聚类结束后的聚类中心。

6、统计分类结束的正确率。

6. 根据实验结果，撰写实验报告。

**六、实验报告**

1. 实验报告包括实验目的、实验原理、实验内容、实验程序、实验结果与实验分析、实验心得体会。

2. 给出完整的源程序，并对源程序中的关键内容或语句给出必要的文字注释。

3. 给出实验结果（实验结果有图的话可截图打印）。

4. 利用实验数据，分析并解答以下问题，

* 1. K值对聚类结果的影响。
  2. 初始聚类中心对聚类结果的影响。
  3. K值对分类结果的影响。

5. 分析K-means和KNN算法的优缺点，并提出改进意见。

6. 本实验的心得体会。

附1： K-means算法例程

clear all;close all;clc;

% 第一组数据

mu1=[0 0 ]; %均值

S1=[.1 0 ;0 .1]; %协方差

data1=mvnrnd(mu1,S1,100); %产生高斯分布数据

%第二组数据

mu2=[1.25 1.25 ];

S2=[.1 0 ;0 .1];

data2=mvnrnd(mu2,S2,100);

% 第三组数据

mu3=[-1.25 1.25 ];

S3=[.1 0 ;0 .1];

data3=mvnrnd(mu3,S3,100);

% 显示数据

plot(data1(:,1),data1(:,2),'b+');

hold on;

plot(data2(:,1),data2(:,2),'r+');

plot(data3(:,1),data3(:,2),'g+');

grid on;

%三类数据合成一个不带标号的数据类

data=[data1;data2;data3];

N=3;%设置聚类数目

[m,n]=size(data);

pattern=zeros(m,n+1);

center=zeros(N,n);%初始化聚类中心

pattern(:,1:n)=data(:,:);

for x=1:N

center(x,:)=data( randi(300,1),:);%第一次随机产生聚类中心

end

while 1

distance=zeros(1,N);

num=zeros(1,N);

new\_center=zeros(N,n);

for x=1:m

for y=1:N

distance(y)=norm(data(x,:)-center(y,:));%计算到每个类的距离

end

[~, temp]=min(distance);

pattern(x,n+1)=temp;

end

k=0;

for y=1:N

for x=1:m

if pattern(x,n+1)==y

new\_center(y,:)=new\_center(y,:)+pattern(x,1:n);

num(y)=num(y)+1;

end

end

new\_center(y,:)=new\_center(y,:)/num(y);

if norm(new\_center(y,:)-center(y,:))<0.1

k=k+1;

end

end

if k==N

break;

else

center=new\_center;

end

end

[m, n]=size(pattern);

%显示聚类后的数据

figure;

hold on;

for i=1:m

if pattern(i,n)==1

plot(pattern(i,1),pattern(i,2),'r\*');

plot(center(1,1),center(1,2),'ko');

elseif pattern(i,n)==2

plot(pattern(i,1),pattern(i,2),'g\*');

plot(center(2,1),center(2,2),'ko');

elseif pattern(i,n)==3

plot(pattern(i,1),pattern(i,2),'b\*');

plot(center(3,1),center(3,2),'ko');

end

end

grid on;

附3： KNN算法例程

clear all;

close all;

clc;

load data%读取数据

Data=data;

for i=1:100

for j=1:3

Data(i,j)=(data(i,j)-min(data(:,j)))/(max(data(:,j))-min(data(:,j)));

end

end

D1=Data(1:80,:);

D2=Data(1:20,:);

k=20;

for i=1:20

temp=D2(i,1:3);

for j=1:80

distance(j)=norm(temp-D1(j,1:3));

end

[distance1,index]=sort(distance);%升序排列

In=index(1:k);

I1=length(find(D1(In,4)==1));

I2=length(find(D1(In,4)==2));

I3=length(find(D1(In,4)==3));

[maxI,class(i)]=max([I1,I2,I3]);

end

ratio=length(find((class'-D2(:,4))==0))/20