《人工智能与模式识别》

实 验 报 告 书

班级： 计科 2002

学号： 202801108

姓名： 李蔚

指导教师： 翟婷婷

2022-2023 学年 第 二 学期

**实验名称： 聚类算法应用实验**

**实验时间： 2023 年 5 月 5日 第 12 周 星期 五**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 题目 | 二 | 三 | | | | 总分 |
| 1(a) | 1(b) | 1(c) | 2 |
| 得分 |  |  |  |  |  |  |

一、实验目的

1．熟悉各种类型的聚类方法及其优缺点。

2. 掌握k-means聚类和k-medoids聚类算法的步骤，并能在Matlab中编程实现。

3．熟悉聚类算法在图像分割中的运用。

二、实验预习（预备知识的问题及回答）

1. 分析聚类与分类的区别是什么？（1分）

**解答：**聚类是无监督学习的一种，旨在将相似的数据点分组到一起，形成聚类簇，而无需先定义类别。分类则是有监督学习的一种，通过从已知类别的样本中学习模式和规律，将新数据点分配到预定义的类别中。简而言之，聚类是基于相似性将数据点分组，而分类是基于已知类别的模式和规律将数据点分配到类别。

三、实验内容（包含实验所用命令或相关程序源代码）

1. 使用matlab完成以下实验：

(a) 使用mvnrnd函数从多变量正态分布中提取3个样本集合：set1，set2，set3，每个集合中包含1000个样本：

第一个集合所使用的正态分布参数：均值向量为[0 0]，协方差矩阵为[0.5 0; 0 0.5]；

第二个集合所使用的正态分布参数：均值向量为[2 3]，协方差矩阵为[0.6 0.1; 0.1 0.6]；

第三个集合所使用的正态分布参数：均值向量为[-2 3]，协方差矩阵为[0.5 -0.1; -0.1 0.5]；

使用scatter函数将这3个集合中的点显示在一幅图中，要求同一个集合中的点的颜色相同，不同集合中的点颜色不同。这相当于生成了3个样本簇。(注意使用figure; hold on 命令保证先画的图不会被后画的图覆盖)

为了运行结果的可再现性，请在程序的开头加上：

clear %清除工作区中所有变量

rng default %为了运行结果的可再现性 for reproducibility

**代码及运行截图：** （1分）

clear % 清除工作区中所有变量

rng default % 设置随机种子为默认，以保证可再现性

% 生成样本集合

set1 = mvnrnd([0 0], [0.5 0; 0 0.5], 1000);

set2 = mvnrnd([2 3], [0.6 0.1; 0.1 0.6], 1000);

set3 = mvnrnd([-2 3], [0.5 -0.1; -0.1 0.5], 1000);

% 绘制图像

figure;

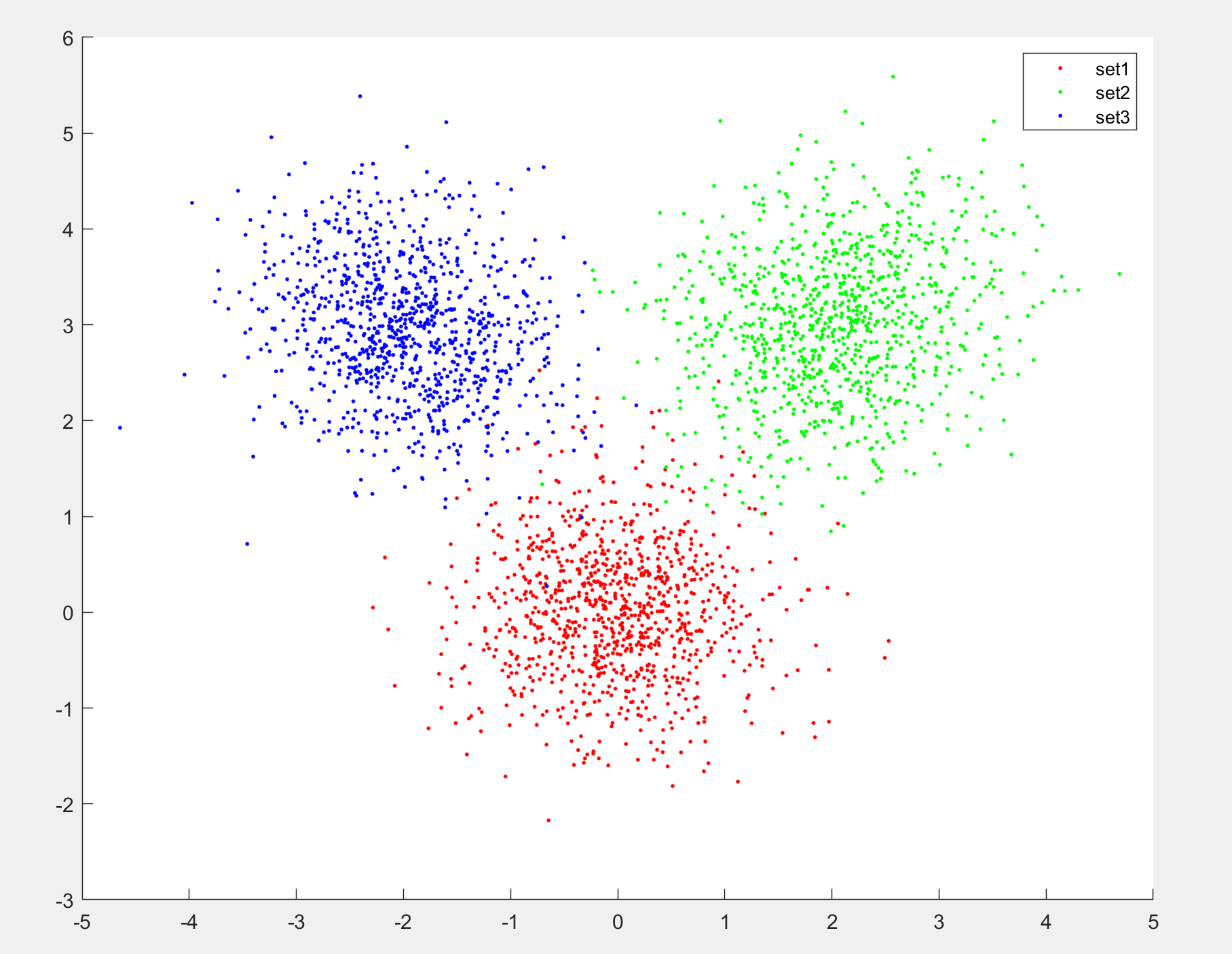
hold on;

scatter(set1(:, 1), set1(:, 2), '.', 'r');

scatter(set2(:, 1), set2(:, 2), '.', 'g');

scatter(set3(:, 1), set3(:, 2), '.', 'b');

legend('set1', 'set2', 'set3');



(b) 编写my\_kmeans函数文件，实现k-means算法的聚类功能，要求函数的输入为待聚类的数据data和聚成的类数k，输出为data中每一个样本所在的簇的编号。将(a)中的3个样本集**合并**为样本集set，然后调用my\_kmeans(set, 3)将样本集set中的样本点聚成3类，根据函数返回值画出聚类结果，保证属于同一个簇的样本点的颜色相同，属于不同簇的样本点的颜色不同，用scatter函数画。

**要求：**聚类算法要求采用**平方欧式距离**来度量样本之间的不相似性，k-means的初始聚类中心使用随机选择k个样本点作为初始聚类中心。此处不允许使用matlab自带的kmeans函数直接得到结果。提示：两个向量的平方欧式距离等于。

**代码和聚类后的结果图：（4分）**

% 将set1、set2、set3合并为样本集set

set = [set1; set2; set3];

% 调用my\_kmeans函数进行聚类

[cluster\_labels, cluster\_centers] = my\_kmeans(set, 3);

% 绘制聚类结果

figure;

scatter(set(cluster\_labels == 1, 1), set(cluster\_labels == 1, 2), 'r');

hold on;

scatter(set(cluster\_labels == 2, 1), set(cluster\_labels == 2, 2), 'g');

scatter(set(cluster\_labels == 3, 1), set(cluster\_labels == 3, 2), 'b');

scatter(cluster\_centers(:, 1), cluster\_centers(:, 2), 'k', 'filled');

legend('Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster Centers');

% my\_kmeans函数实现

function [cluster\_labels, cluster\_centers] = my\_kmeans(data, k)

% 初始化聚类中心

indices = randperm(size(data, 1));

cluster\_centers = data(indices(1:k), :);

% 初始化聚类标签

cluster\_labels = zeros(size(data, 1), 1);

% 迭代更新聚类中心和聚类标签

converged = false;

while ~converged

% 计算样本点到聚类中心的距离

distances = pdist2(data, cluster\_centers, 'squaredeuclidean');

% 将样本点分配到距离最近的聚类中心

[~, new\_labels] = min(distances, [], 2);

% 更新聚类中心

for i = 1:k

cluster\_centers(i, :) = mean(data(new\_labels == i, :));

end

% 检查是否收敛

if isequal(new\_labels, cluster\_labels)

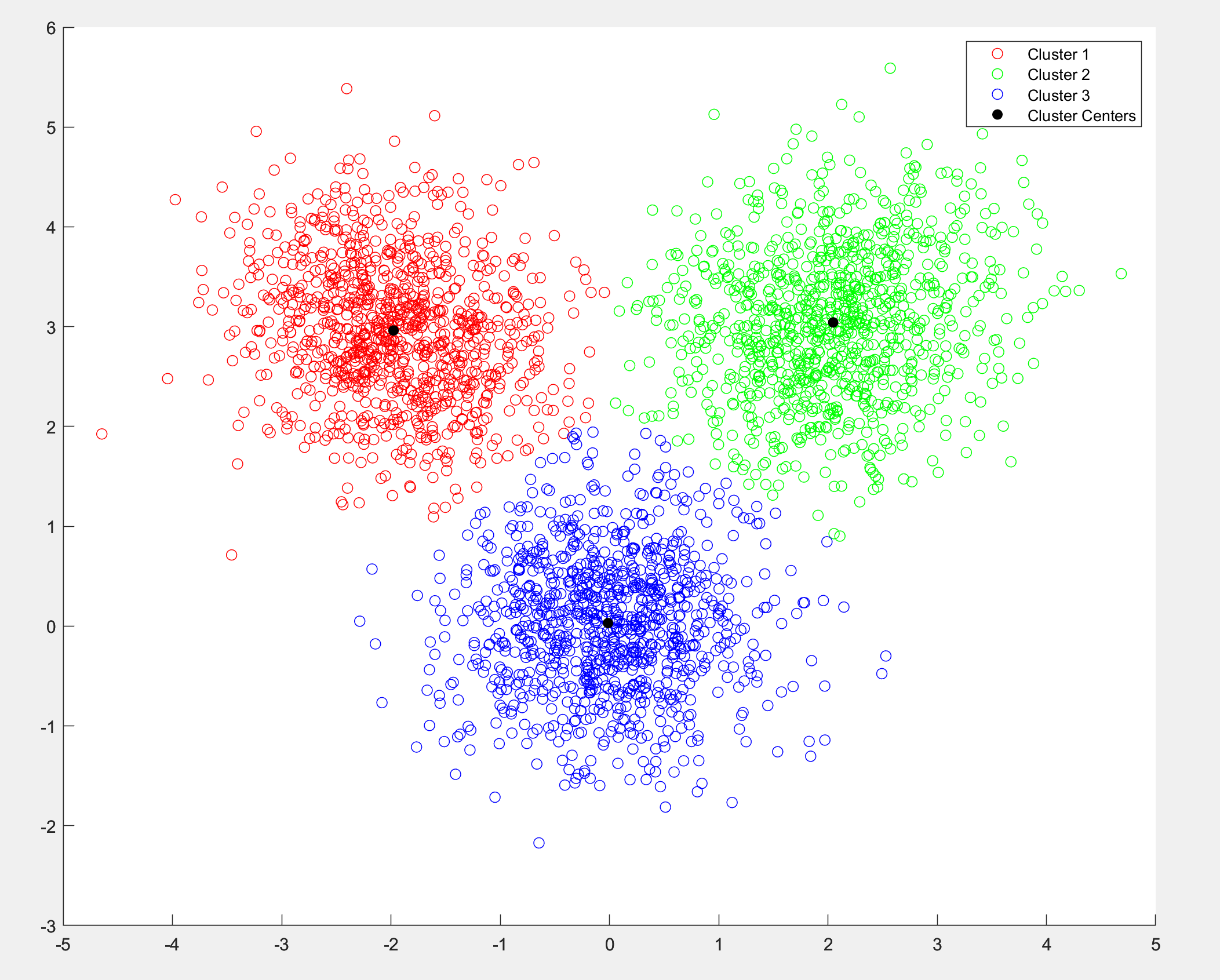
converged = true;

end

cluster\_labels = new\_labels;

end

end

****

(c) 按照与(b)相同的要求编写my\_kmedoids函数文件，实现k-medoids算法的聚类功能，并调用该函数对set中样本点聚成3类，画出聚类结果。

**代码和聚类后的结果图：（4分）**

% 调用my\_kmedoids函数进行聚类

[cluster\_labels, cluster\_centers] = my\_kmedoids(set, 3);

% 绘制聚类结果

figure;

scatter(set(cluster\_labels == 1, 1), set(cluster\_labels == 1, 2), 'r');

hold on;

scatter(set(cluster\_labels == 2, 1), set(cluster\_labels == 2, 2), 'g');

scatter(set(cluster\_labels == 3, 1), set(cluster\_labels == 3, 2), 'b');

scatter(cluster\_centers(:, 1), cluster\_centers(:, 2), 'k', 'filled');

legend('Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster Centers');

% my\_kmedoids函数实现

function [cluster\_labels, cluster\_centers] = my\_kmedoids(data, k)

% 初始化聚类中心（选择随机的k个样本点作为初始聚类中心）

indices = randperm(size(data, 1));

cluster\_centers = data(indices(1:k), :);

% 初始化聚类标签

cluster\_labels = zeros(size(data, 1), 1);

% 迭代更新聚类中心和聚类标签

converged = false;

while ~converged

% 计算样本点与聚类中心之间的距离

distances = pdist2(data, cluster\_centers, 'squaredeuclidean');

% 将样本点分配到距离最近的聚类中心

[~, new\_labels] = min(distances, [], 2);

% 更新聚类中心

for i = 1:k

cluster\_indices = find(new\_labels == i);

cluster\_distances = pdist2(data(cluster\_indices, :), data(cluster\_indices, :), 'squaredeuclidean');

[~, min\_index] = min(sum(cluster\_distances, 2));

cluster\_centers(i, :) = data(cluster\_indices(min\_index), :);

end

% 检查是否收敛

if isequal(new\_labels, cluster\_labels)

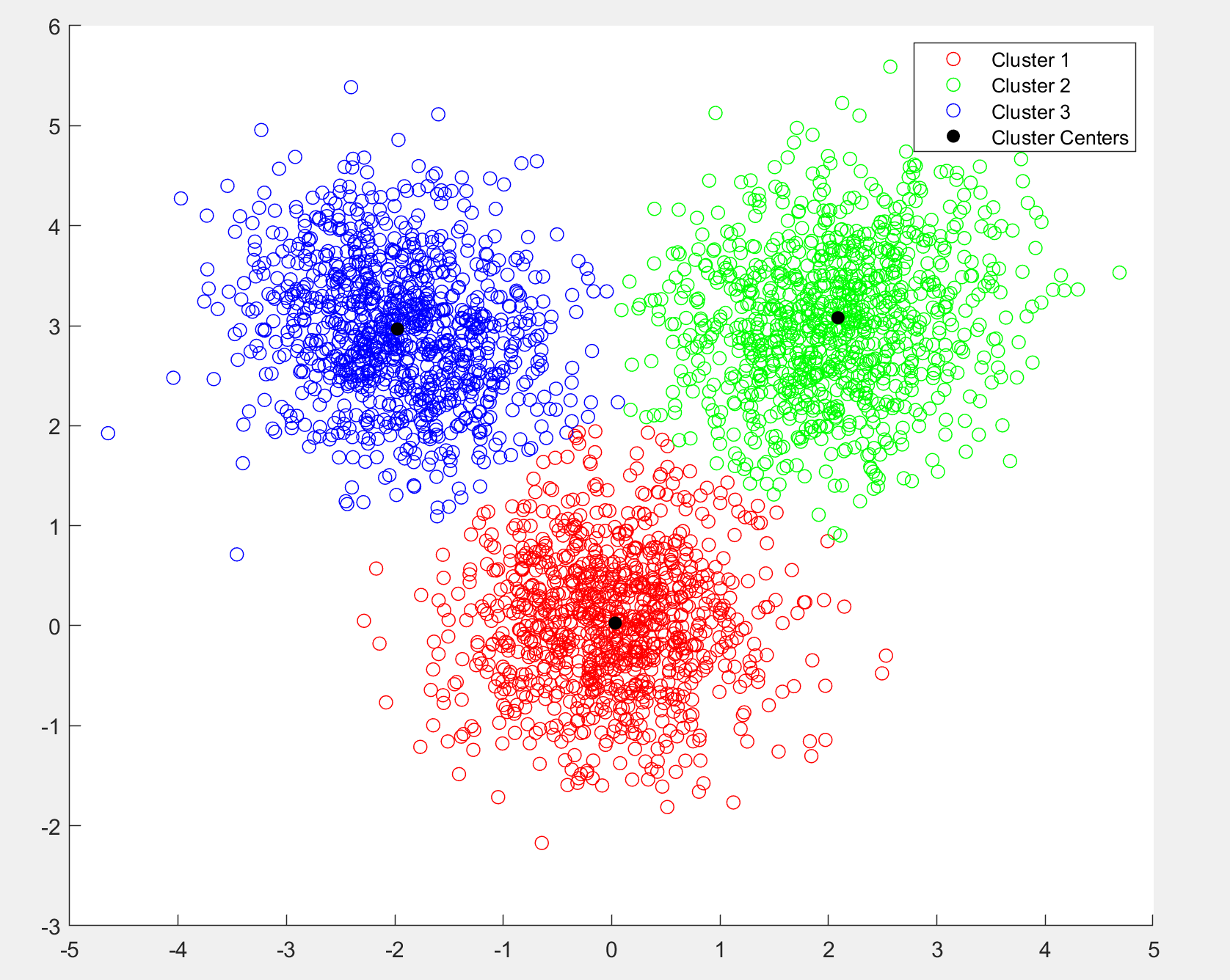
converged = true;

end

cluster\_labels = new\_labels;

end

end



2. 图像分割(image segmentation)是将图像划分成几个有意义的部分/区域的过程。这种划分通常是基于图像中像素的特征，例如，根据颜色值将图像分成区域。**现在，利用k-means算法对cherry.jpg和dog.jpg两图RGB模式的图像进行区域分割，分别考虑聚成k=2 和k=3类两种情况，并用imshow(Img, [])函数显示分割结果。**此处可以使用matlab自带的kmeans函数。

**补充知识：**RGB模式的图像读入后为m\*n\*3的数组，记这个数组为a，则a(:, :, 1)为颜色通道R上的取值矩阵，a(:, :, 2)为颜色通道G上的取值矩阵，a(:, :, 3)为颜色通道B上取值矩阵。图像中位于第i行第j列的像素点的颜色为a(i, j, :)，它包括a(i, j, 1)，a(i, j, 2)，a(i, j, 3)，分别是在3个颜色通道上的颜色取值，这些取值共同决定一个像素点显示为什么颜色。白色的RGB值为[1, 1, 1]，黑色的RGB值为[0,0,0]，红色的RGB值为[1 0 0]。

**利用k-means进行分割的思路：**对所有像素点按照**颜色**进行聚类，把颜色差异不大的像素点聚成一类，颜色差异较大的聚成不同的类，k的取值决定了聚成几类。

**matlab代码：(3分)**

% 读取图像

cherryImg = imread('cherry.jpg');

dogImg = imread('dog.jpg');

% 将图像转换为 double 类型

cherryImg = im2double(cherryImg);

dogImg = im2double(dogImg);

% 获取图像的尺寸

[cherryRows, cherryCols, ~] = size(cherryImg);

[dogRows, dogCols, ~] = size(dogImg);

% 将图像转换为矩阵形式，每一行代表一个像素的 RGB 值

cherryData = reshape(cherryImg, cherryRows \* cherryCols, 3);

dogData = reshape(dogImg, dogRows \* dogCols, 3);

% 使用 k-means 算法进行图像分割

% k=2, cherry.jpg

k = 2;

[cherryIdx2, cherryCentroids2] = kmeans(cherryData, k);

% k=2, dog.jpg

[k2, dogCentroids2] = kmeans(dogData, k);

dogIdx2 = k2;

% k=3, cherry.jpg

k = 3;

[cherryIdx3, cherryCentroids3] = kmeans(cherryData, k);

% k=3, dog.jpg

[k3, dogCentroids3] = kmeans(dogData, k);

dogIdx3 = k3;

% 将聚类结果映射回图像

cherrySeg2 = reshape(cherryIdx2, cherryRows, cherryCols);

dogSeg2 = reshape(dogIdx2, dogRows, dogCols);

cherrySeg3 = reshape(cherryIdx3, cherryRows, cherryCols);

dogSeg3 = reshape(dogIdx3, dogRows, dogCols);

% 显示分割结果

figure;

subplot(2, 2, 1);

imshow(cherryImg);

title('Original Cherry Image');

subplot(2, 2, 2);

imshow(cherrySeg2, []);

title('Segmented Cherry Image (k = 2)');

figure;

subplot(2, 2, 1);

imshow(dogImg);

title('Original Dog Image');

subplot(2, 2, 2);

imshow(dogSeg2, []);

title('Segmented Dog Image (k = 2)');

figure;

subplot(2, 2, 1);

imshow(cherryImg);

title('Original Cherry Image');

subplot(2, 2, 2);

imshow(cherrySeg3, []);

title('Segmented Cherry Image (k = 3)');

figure;

subplot(2, 2, 1);

imshow(dogImg);

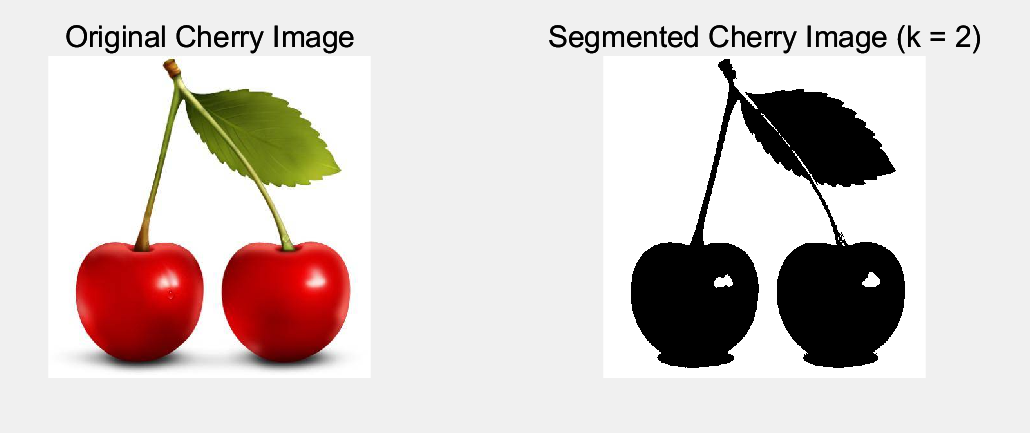
title('Original Dog Image');

subplot(2, 2, 2);

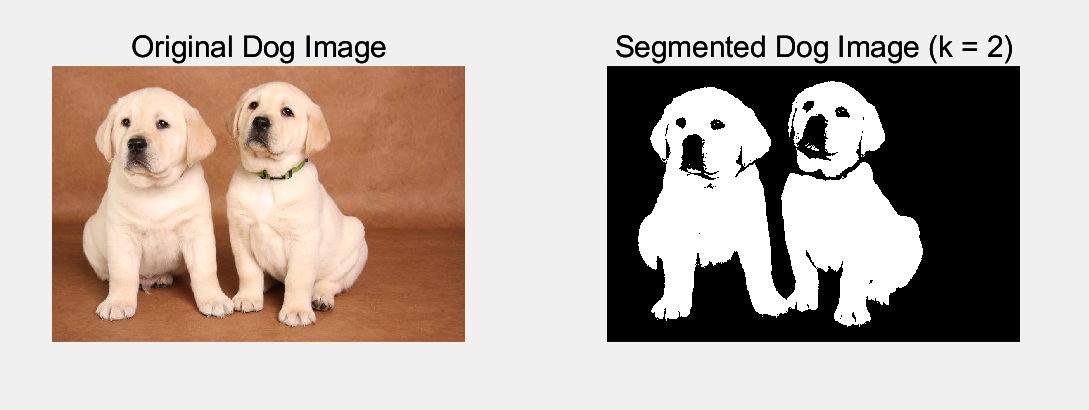
imshow(dogSeg3, []);

title('Segmented Dog Image (k = 3)');

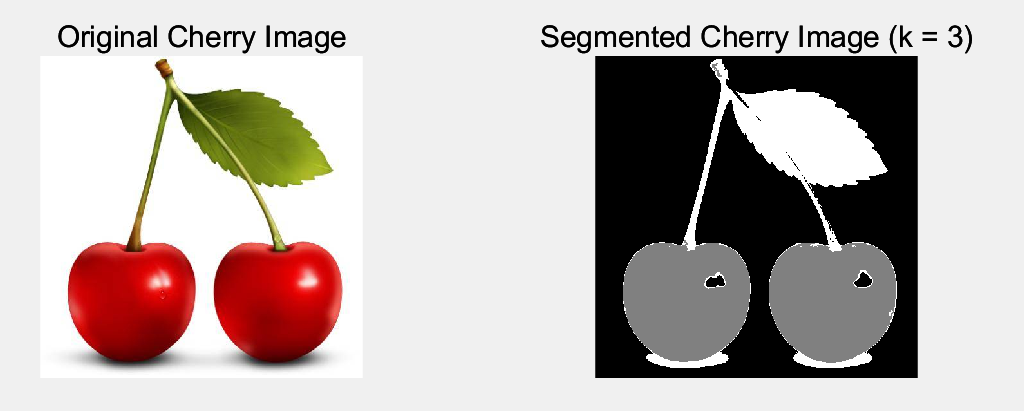
**k=2时，cherry.jpg聚类前后的结果图：（0.5分）**

****

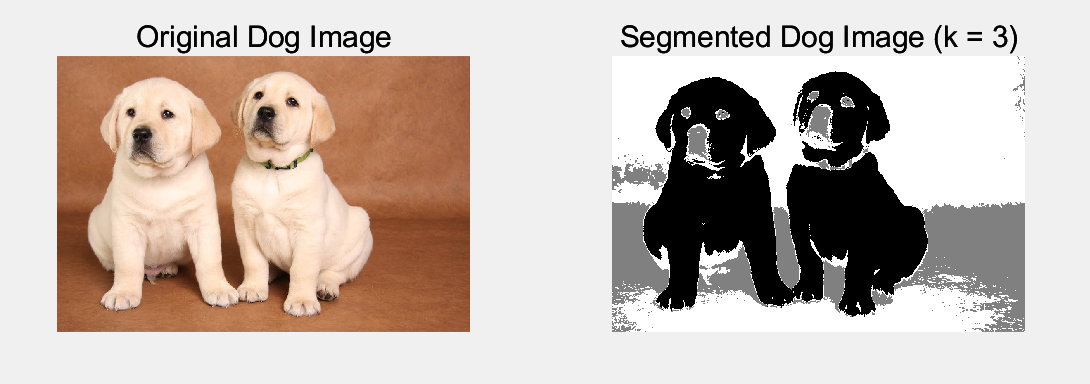
**k=2时，dog.jpg聚类前后的结果图：（0.5分）**

****

**k=3时，cherry.jpg聚类前后的结果图：（0.5分）**

****

**k=3时，dog.jpg聚类前后的结果图：（0.5分）**

****