《人工智能与模式识别》

实 验 报 告 书

班级： 计科1801

学号： 181604124

姓名： 汤礽禾

指导教师： 翟婷婷

2020-2021 学年 第 二 学期

**实验名称： 聚类算法编程实现**

**实验时间： 2021 年 4 月 28日 第 9 周 星期 三**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 题目 | 二 | 三 | | | | | 总分 |
| 1(a) | 1(b) | 2(a)(b) | 3 |  |
| 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？（0.5分）

**解答：**

**分类：给定已标记的训练集，构造分类决策边界，对特征空间进行划分。**

**聚类：给定一组数据，发现数据中潜在的结构或分组。**

1. 分析k-medoids聚类算法相比于k-means聚类算法的优点是什么，缺点是什么？（0.5分）

**解答：**

**优点：**

**当样本中存在噪声/离群点时，K-medoid对噪声的鲁棒性比较好。簇的medoid可以削弱离群点的影响。**

**缺点：**

1. **k-medoids算法的时间复杂度更高：Ο(tkn^2)，其中,n是样本数，k是簇的数目，t是算法迭代次数。因为每次迭代中计算簇的medoid的代价比计算簇的均值的代价大的多。**
2. **两个算法都会收敛但是都容易陷入局部最优。**
3. **最终的聚类都会受初始簇中心选择的影响。**
4. **每次迭代搜会降低误差平方准则函数的值。**

3. 简述k-means聚类算法存在哪些缺点。（1分）

**解答：**

1. **簇数k的值很难确定。**
2. **仅适用于数值型数据，对分类型数据不适用，因为对于分类型数据，均值向量没有定义。**
3. **对于非凸/非球形的簇，聚类效果不佳。**
4. **对于各簇中样本规模差异太大、不同密度的簇，聚类效果不佳。**
5. **最终的聚类结果和初始簇中心的选择有关，容易陷入局部最优。**
6. **对噪声/离群点很敏感。离群点是离其它数据点很远的数据点，离群点可能是由于数据记录的错误造成的。在利用k-means算法执行聚类分析前，要去除离群点。**
7. **每次迭代搜会降低误差平方准则函数的值。**
8. **最终的聚类都会受初始簇中心选择的影响。**

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。

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

**matlab代码：** （1分）

clear;clc;close all;rng default;

%产生多维正态随机数，mu为期望向量，sigma为协方差矩阵，n为规模。

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)

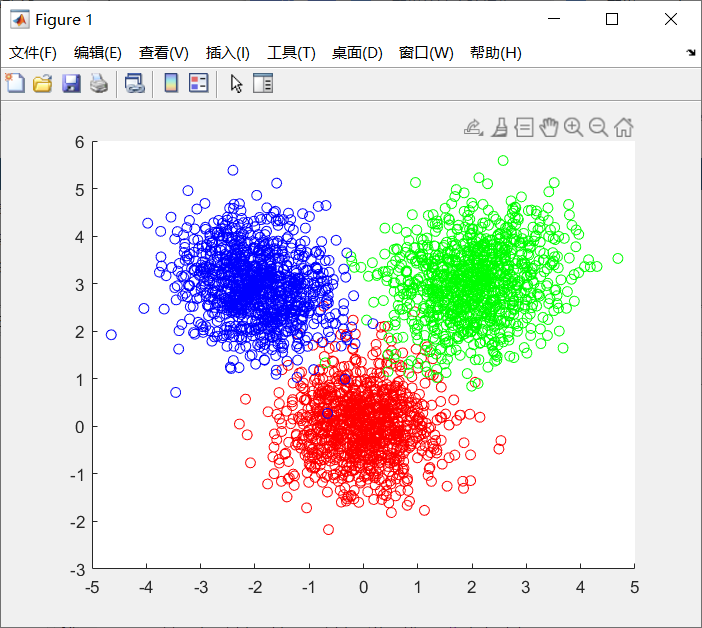
figure(1);

scatter(set1(:,1), set1(:,2), 36, 'R');hold on;

scatter(set2(:,1), set2(:,2), 36, 'G');hold on;

scatter(set3(:,1), set3(:,2), 36, 'B');hold on;

**scatter图截图：**（1分）



(b) 将上述3个样本集合并为样本集set中；然后编程实现k-means和k-medoids算法**，**对样本集set中的样本点进行聚类，最后显示聚类结果：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，同样用scatter函数画出。

**要求：**两个聚类算法都要求采用**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和kmedoids函数直接得到结果。提示：两个向量的平方欧式距离等于。

**k-means算法的matlab代码：（4分）**

%% k\_means

function [new\_center, lable] = k\_means(k, set)

% 1

[m, n] = size(set);% m=行数=数据个数，n=列数

center = zeros(k, n); % 初始化簇中心

for j = 1:n % 随机簇中心（2列随机数）

minJ = min(set(:,j));

rangeJ = max(set(:,j))-min(set(:,j));

center(:,j) = minJ+rand(k,1)\*rangeJ; %产生区间上的随机数

end

lable = zeros(m , 1); % label记录坐标点属于哪个簇。

% 2

while 1

distance = zeros(1, k); % 存放第i个样本点到第j个聚类中心的距离

new\_center = zeros(k, n);

for i = 1 : m % 样本点的个数

for j = 1 : k % 聚类中心数目，即确定是哪个簇

distance(:, j) = (set(i, :) - center(j, :)) \* (set(i, :) - center(j, :))'; % distance(j)：第j 个坐标 与 簇中心的 欧式距离

end

lable(i, 1) = find(distance == min(distance), 1); % 返回最小距离的下表

end

% 新的簇中心

for i=1 : k

new\_center(i,:) = mean(set(lable(:, 1)==i,:), 1);% 每一列均值的列向量。

end

% 若簇中心不改变，停止

if new\_center == center

break;

else

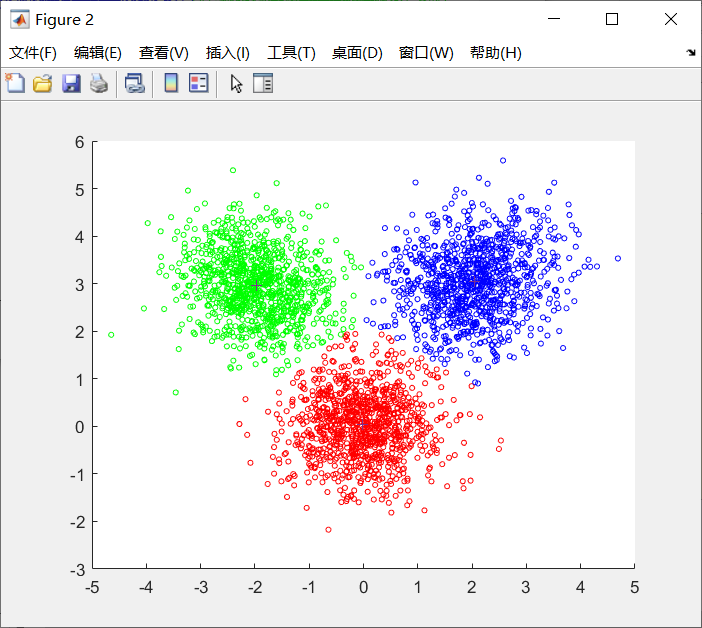
center = new\_center;

end

end

end

**k-means聚类后的结果图：（1分）**



**k-medoids算法的matlab代码：（4分）**

%% k-medoids

function [lable, new\_center] = k\_medoids(k, set)

[m, n] = size(set); % m=行数=数据个数，n=列数

center\_random = round(rand(k,1)\*m); % 随机3个坐标作为簇中心，center\_random = 坐标索引

center = set(center\_random, :);

lable = zeros(m , 1);

while 1

distance = zeros(1, k); %存放第i个样本点到第j个聚类中心的距离

new\_center = zeros(k, n); %新的簇中心

% 聚类

for i = 1 : m % 样本数量

for j = 1 : k % 聚类数目

distance(:, j) = (set(i, :) - center(j, :)) \* (set(i, :) - center(j, :))';% 欧氏距离

end

lable(i, 1) = find(distance == min(distance),1);

end

for i=1:k

cluster = set(lable==i,:); %将第i个簇提取到cluster里面

num = size(cluster,1); %获得第i个簇的样本点的数量=行数

sum\_dis = zeros(num,1); % 用于存放该簇中对应的每个样本点对应的距离平方的总和

% 第j个坐标点到其他坐标点的距离，存sum中

for j = 1 : num

for x = 1 : num

% if j ~= x

dis = (cluster(j, :) - cluster(x, :)) \* (cluster(j, :) - cluster(x, :))';

sum\_dis(j,1) = sum\_dis(j,1) + dis;

% end

end

end

new\_center(i,:) = cluster(sum\_dis==min(sum\_dis),:); % new\_center存到每个坐标点最小距离的坐标

end

% 若簇中心全都不改变，算法停止

if new\_center == center

break;

else

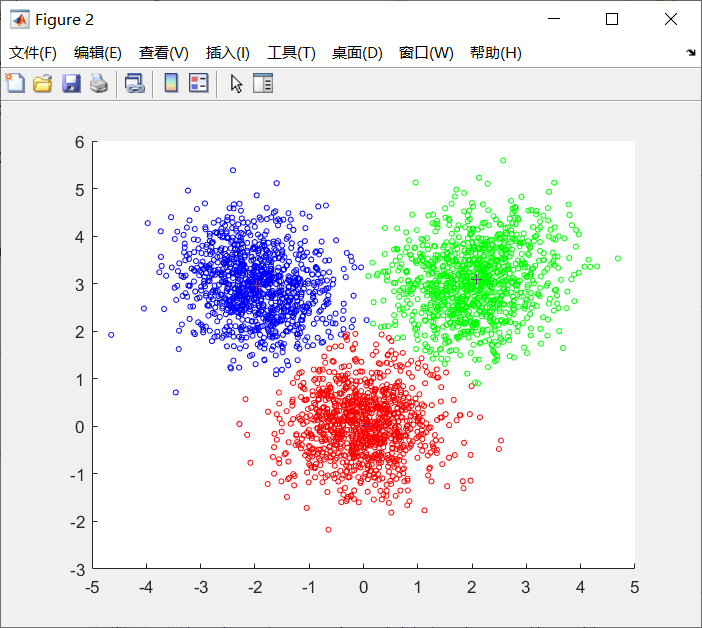
center = new\_center;

end

end

End

**k-medoids聚类后的结果图：（1分）**



1. 按照与第1题相同的步骤，完成对下列数据集set的聚类操作：

set1 = mvnrnd([0 0],[0.1 0; 0 0.1],2000);

set2 = mvnrnd([1.5 1],[0.07 0; 0 1],2000);

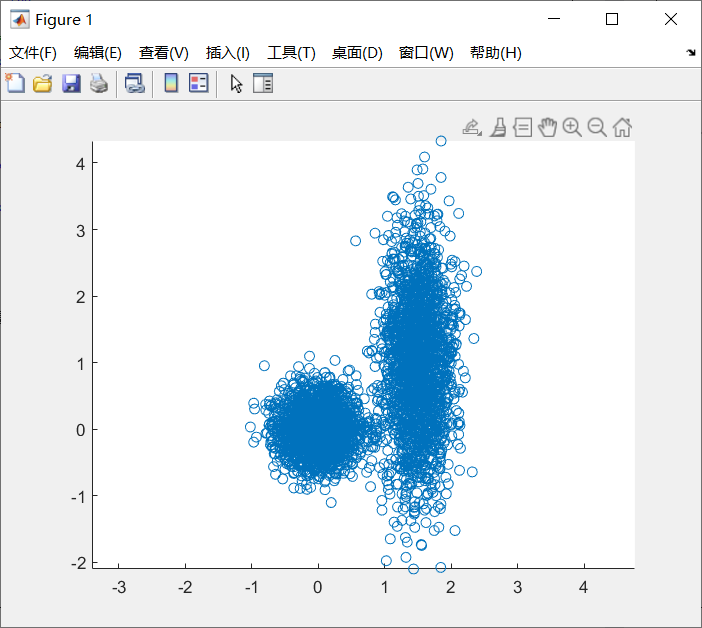
set = [set1; set2];

(a) 用你编写的k-means和k-medoids算法分别对样本集set进行聚类，并将聚类前的样本点的散点图和聚类后的样本散点图画出，两幅图中都要保证同一个簇的样本点的颜色相同，不同簇的样本点颜色不同。

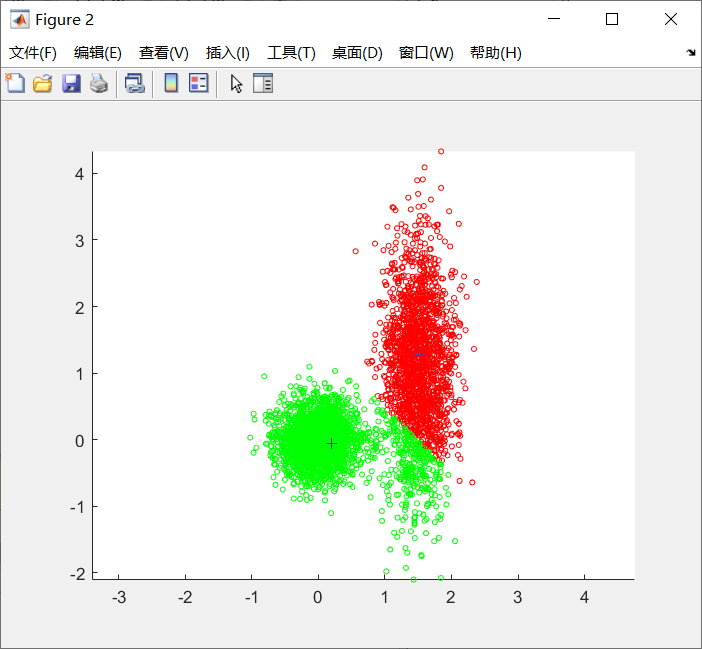
**提示**：在显示图时，使用axis equal命令，保证图中两个坐标轴上的刻度增量相同。

k-means**聚类前和聚类后的结果图：**(0.5分)

**聚类前**

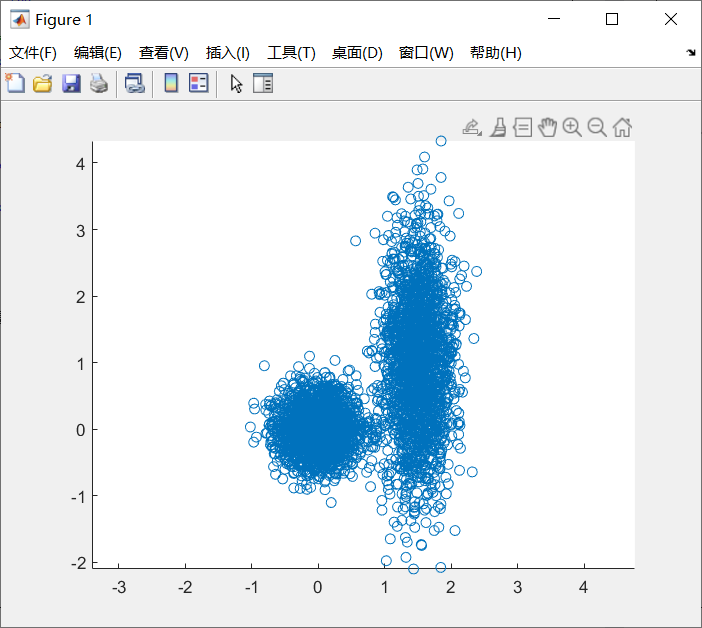
****

**聚类后**

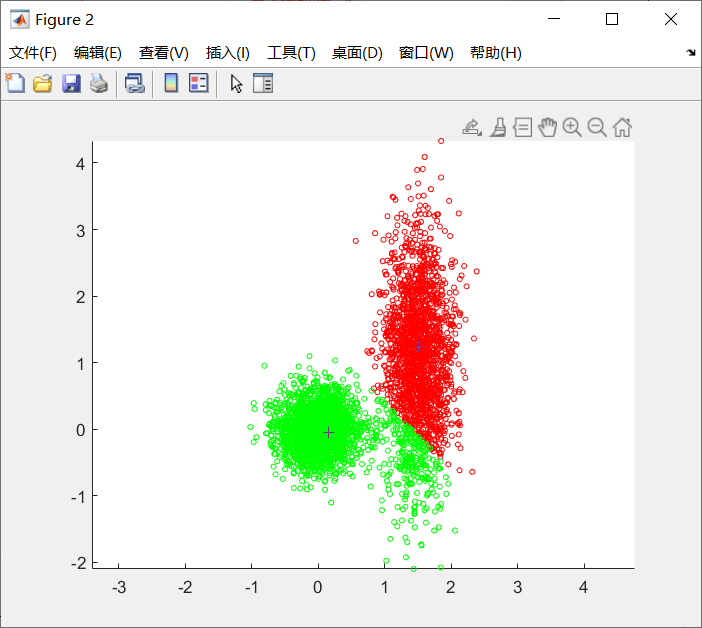
****

k-medoids**聚类前和聚类后的结果图：**(0.5分)

**聚类前**

****

**聚类后**

****

(b) 在set样本集上，你的聚类算法能找到准确的簇结构吗？如果不能，请分析原因。(1分)

**解：**

**不能。因为set样本有很多离群点聚集在右下方，使计算所得的均值向量偏向右下方，故不能找到准确的簇结构。**

3. 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k=2 和k=3两种情况，并用imshow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：(5分)**

clear;clc;close all;rng default;

%%

k = input('聚类数目：');

%%

set = imread('dog.jpg');

figure(1);

imshow(set);

set = im2double(set);

[row, col, c] = size(set);

c1 = set(:, :, 1);

c2 = set(:, :, 2);

c3 = set(:, :, 3);

data = [c1(:), c2(:), c3(:)];

new\_pic = kmeans(data, k);

new\_pic = reshape(new\_pic,row,col);

imshow(new\_pic,[]); % 缩放图像

%%

set2 = imread('cherry.jpg');

figure(2); imshow(set2);

set2 = im2double(set2);

[row, col, c] = size(set2);

c1 = set2(:, :, 1);

c2 = set2(:, :, 2);

c3 = set2(:, :, 3);

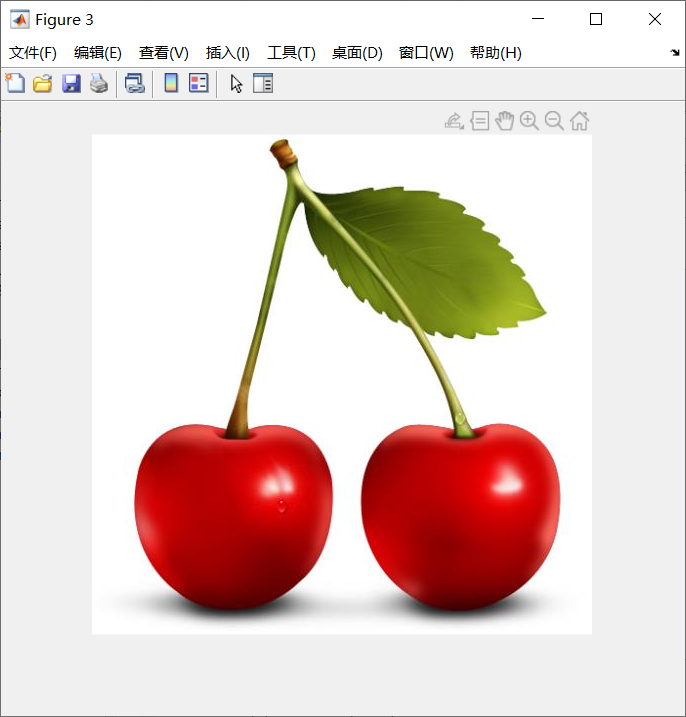
data = [c1(:), c2(:), c3(:)];

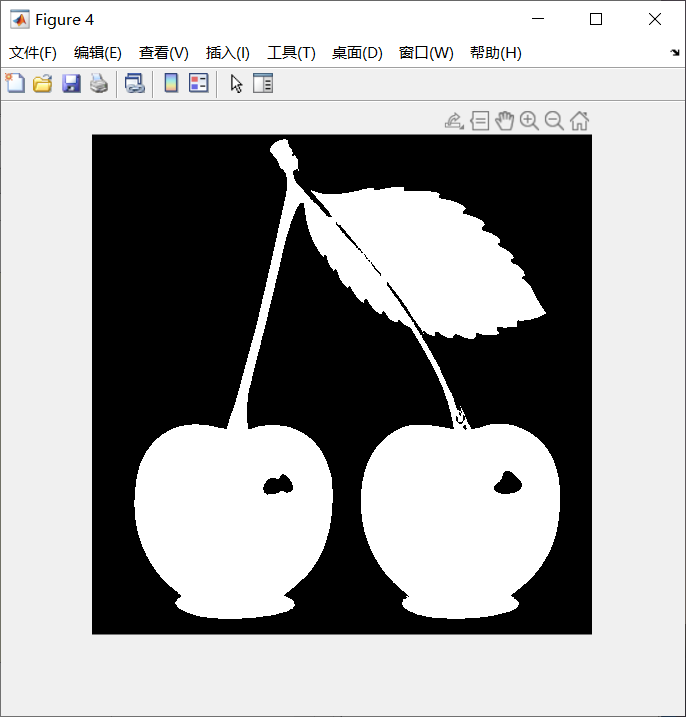
new\_pic = kmeans(data, k);

new\_pic = reshape(new\_pic,row,col);

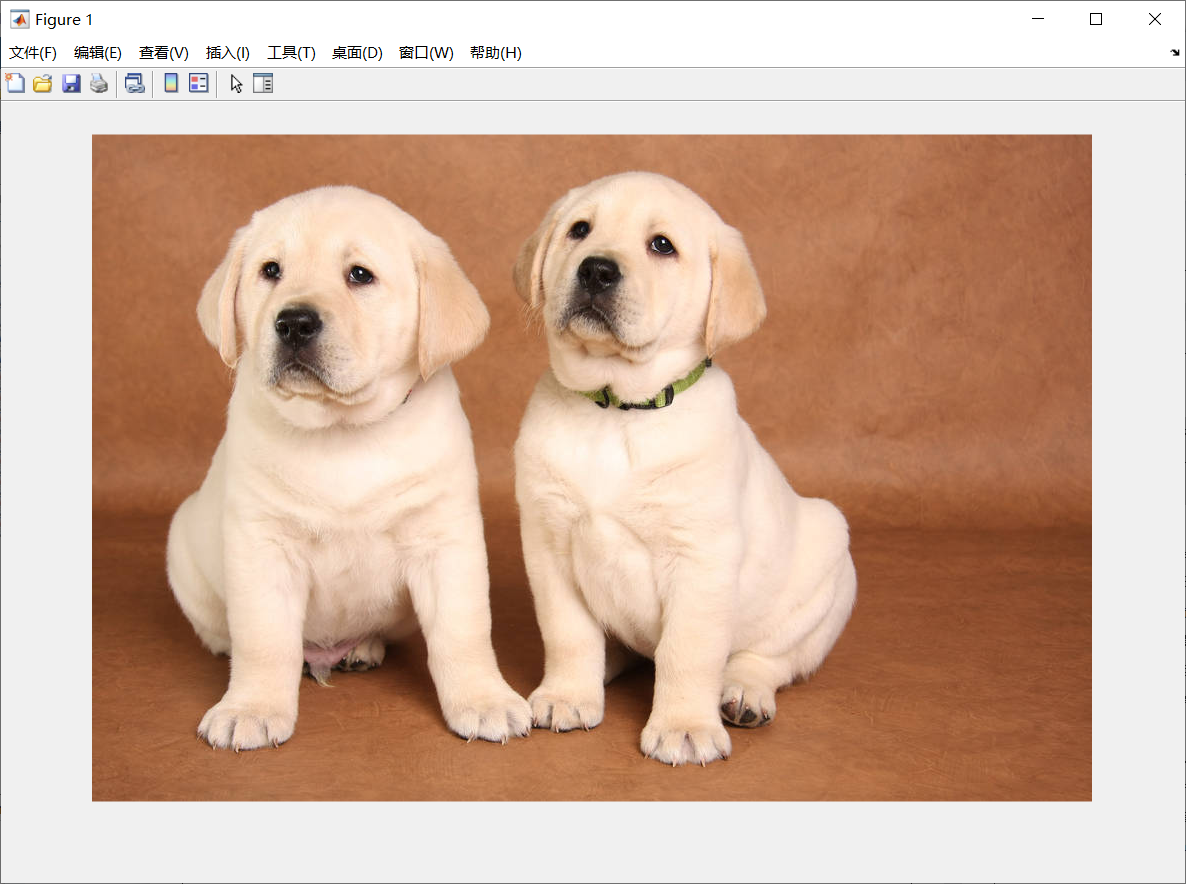
imshow(new\_pic,[]);

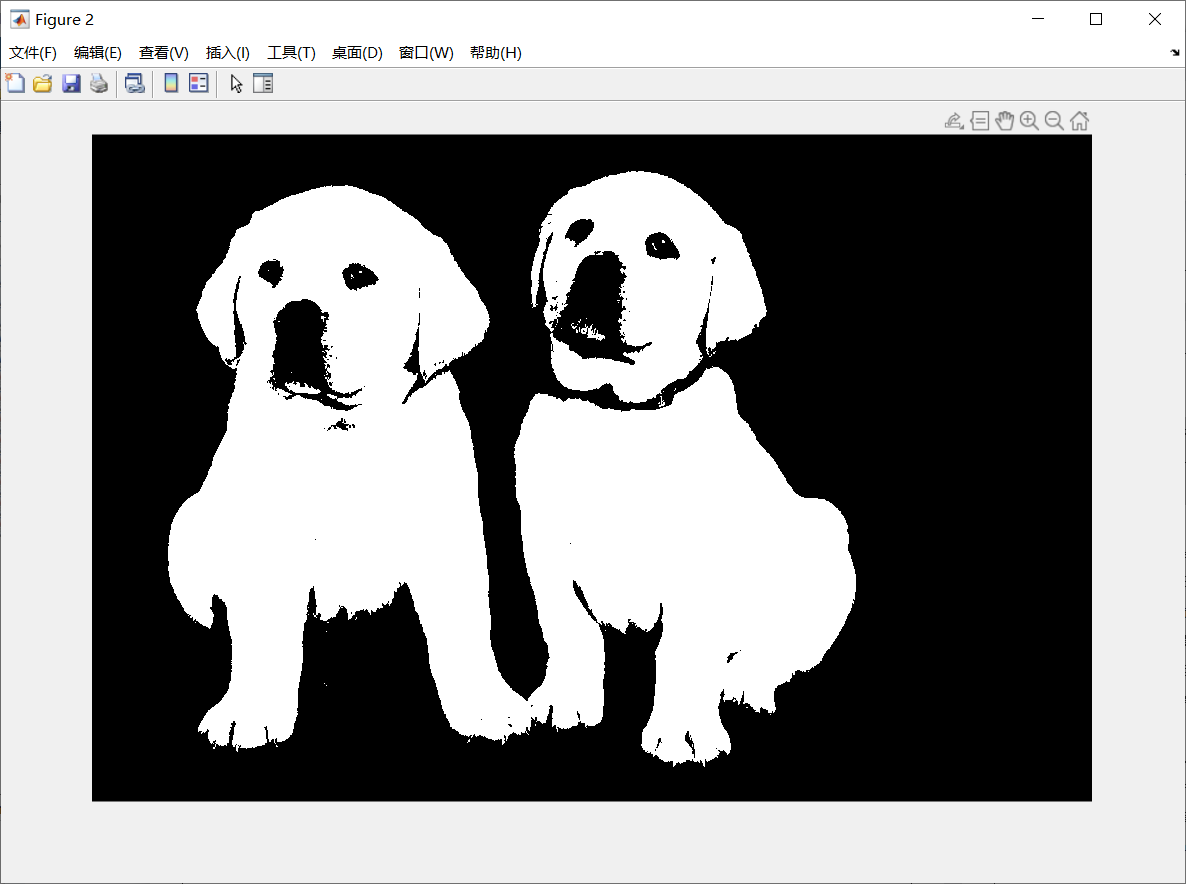
**k=2时，cherry.jpg分割前和分割后的结果图：（1分）**

****

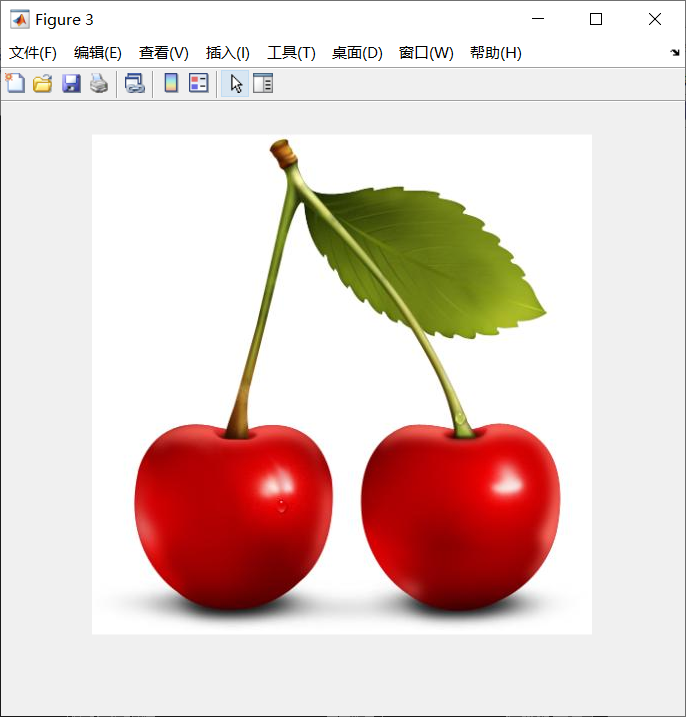
****

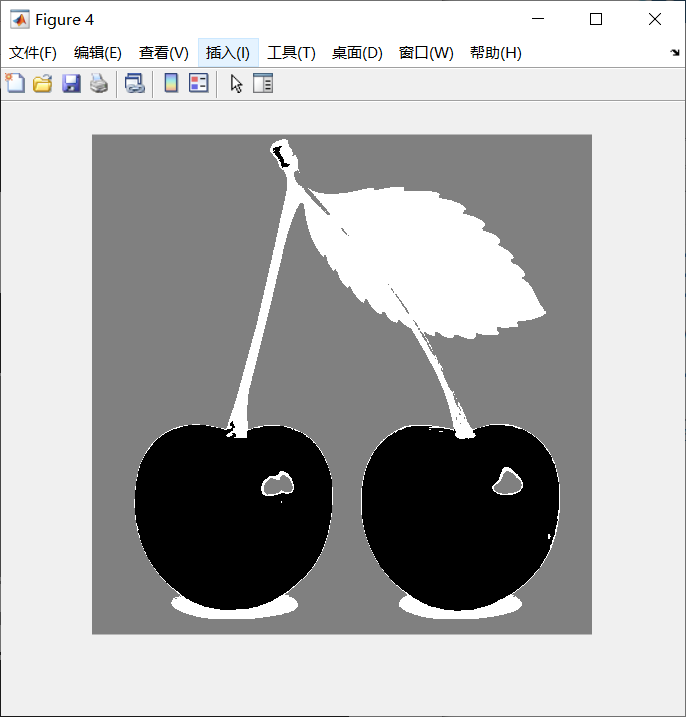
**k=2时，dog.jpg分割前和分割后的结果图：（1分）**

****

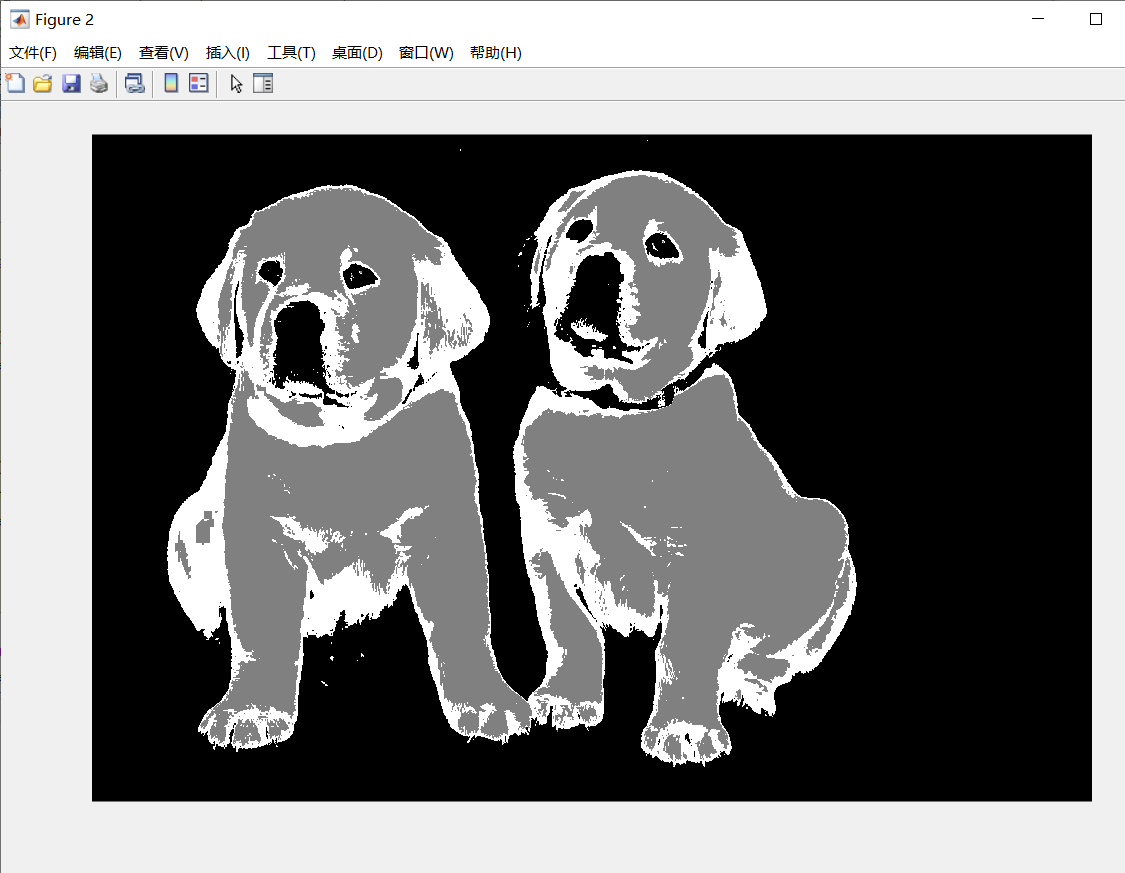
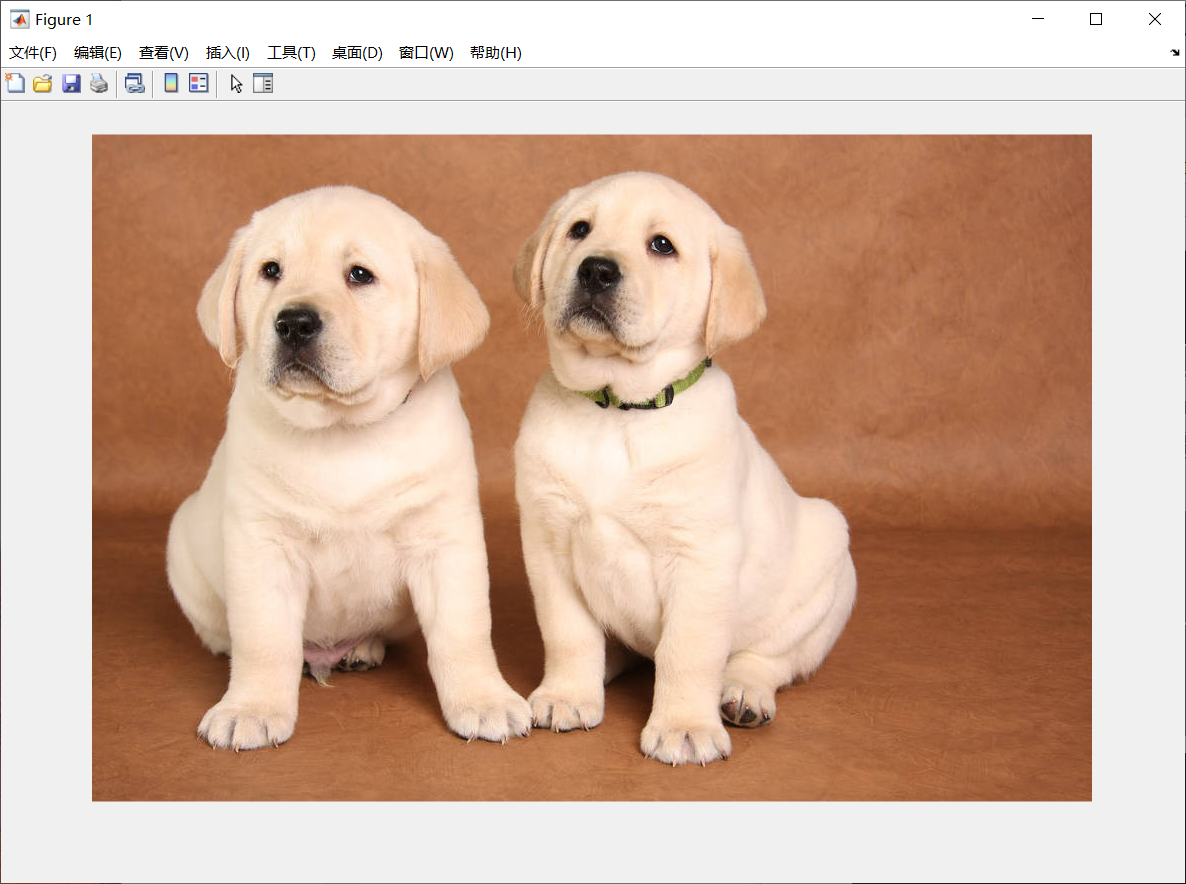
****

**k=3时，cherry.jpg分割前和分割后的结果图：（1分）**

****

****

**k=3时，dog.jpg分割前和分割后的结果图：（1分）**

****