

**计算智能导论**

**（聚类实现图像分割）**

授课老师：张梦璇

学院：人工智能学院

班级：1820013

专业：智能科学与技术

姓名：刘继垚

学号：18200100176

## 1背景介绍

**1.1 图像分割概述**

图像分割就是把图像细分为构成它的对象或子区域，这些区域是互不相交的，每个区域都满足特定区域的一致性。分割的程度主要取决于人们想要解决的问题，当感兴趣的区域或对象已经被区分出来，分割就算完成。图像分割是图像处理中的重要问题，也是计算机视觉研究中的一个经典难题。计算机视觉中的图像理解包括目标检测、特征提取和目标识别等，都依赖于分割的质量。

目前，图像分割算法一般是围绕亮度值的两个基本特性设计的：不连续性和相似性。亮度值的不连续性的应用途径主要是基于像素点特性（如灰度值）的不连续变化分割图像，如最常用的边缘检测。而利用亮度值的相似性可以形成一套机制，即依据事先指定的准则将图像分割为相似的区域。一些实例包括门限处理、区域分离、区域生长和聚类等。而采用模糊C均值聚类进行图像分割的好处是避免了阈值的设定问题，聚类的过程不需要人工干预，只需输入预想的分类数目即可实现自动化的图像分割。

**1.2 模糊聚类在图像处理中的应用**

模糊聚类广泛应用在控制、模式识别、信号处理、人工智能、决策等领域。本文使用FCM算法实现图像分割，用于实现火星图像场景分割。

## 2．实验原理

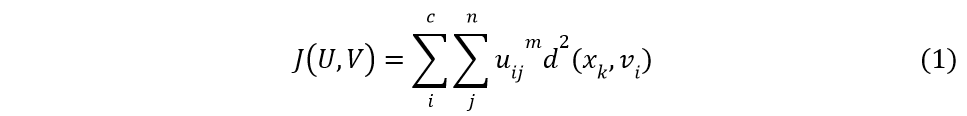
**2.1 模糊聚类理论**

基于模糊集合的特点，模糊聚类方法应运而生。聚类，就是将一组给定的未知类标号的样本分成内在的多个类别，使得同一类中的样本具有较高的相似度，而不同的类中样本差别大。聚类分析的目的是揭示和刻画数据的内在结构，其内容涉及统计学、生物学、以及机器学习等研究领域，并在模式识别、数据分析和挖掘、图像处理等领域获得了广泛的应用。

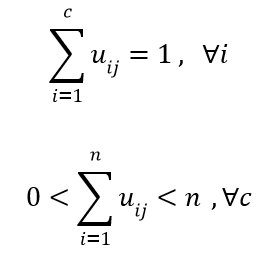
在众多模糊聚类算法中，FCM算法应用最广泛且较成功，它通过优化目标函数得到每个样本点对所有类中心的隶属度，从而决定样本点的类属以达到自动对样本数据进行分类的目的。

* 1. **模糊C-均值聚类算法**

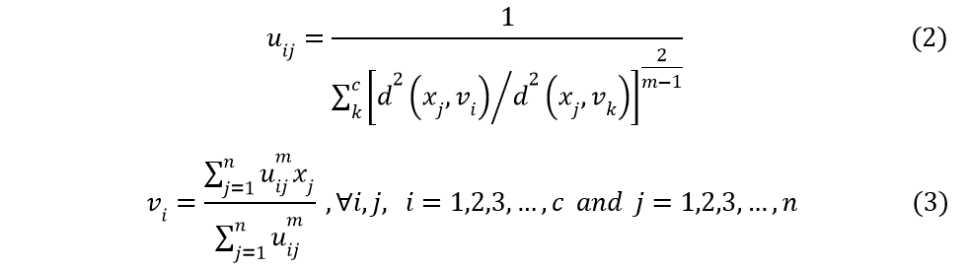
根据聚类的数目C和一组包含n个L维向量的数据,用算法输出元素的隶属度,它代表着数据属于第i个类的概率，可以通过求式(1)目标函数的最小值得到，通常取m=2。



其中，式(1)的约束条件为：



  在(1)式的约束条件下，可以求得(1)中目标函数取最小值时相应的隶属度矩阵和聚类中心。通常，该最小值用极小值代替，因此分别对各变量求偏导，并令偏导数为0，联立并解出更新后的模糊隶属度和聚类中心，如下公式(2)(3)。



**2.3 算法流程**

根据FCM的基本原理，总结出该算法的步骤如下：

  1)设置目标函数的精度e，模糊指数m（m通常取2）和算法最大迭代次数；

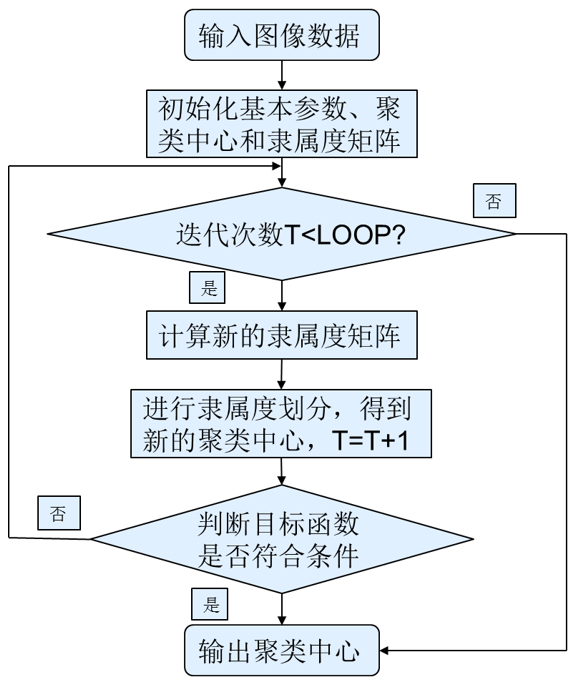
  2)初始化隶属度矩阵或聚类中心；

  3)由式(2)(3)更新模糊划分矩阵和聚类中心；

  4)若目标函数则迭代结束；否则，跳转执行第三步；

  5)根据所得到的隶属度矩阵，取样本隶属度最大值所对应类作为样本聚类的结果，聚类结束。

  给出FCM的算法流程图。



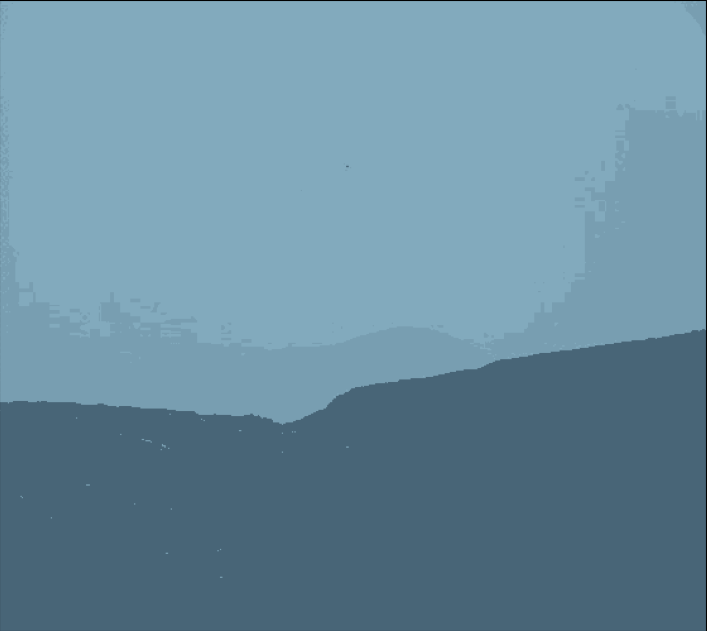
## 实验过程及结果及分析

本次实验使用FCM算法，用于实现像素级别的火星图像场景分割。参数设置如下：

|  |  |
| --- | --- |
| 分类数 | 3 |
| 模糊指数m | 2 |
| 终止迭代精度 | 0.001 |

**3.1 实验结果**

原图 分割结果

**四、实验总结与结果分析**

本实验通过FCM实现了火星图像的分割。

我们将图片按像素分为三类，可以看出，FCM分割的图像效果是显著的，能够准确地按照像素值的大小进行聚类，实现基于像素值的分割。

FCM算法有几个超参数，其中m值和迭代停止阈e对算法影响最大，考虑到e可以设定一极小量，即满足条件就行;而m值研究过取2时最合适。

那么，如何使分类最合理，使目标函数值最小。这也直接决定了分割质量的好坏。

在FCM算法中，一开始都是随机初始一个隶属矩阵，然后通过不停的迭代计算，在达到迭代停止阈值时完成分类，分类的好坏直接受制于最开始的随机隶属矩阵。

## 五、实验代码（python）

**import cv2**

**import numpy as np**

**import matplotlib.pyplot as plt**

**np.random.seed(1234)#Random seeds, random to the same array at a time**

**def fcm(X,k,b):**

**iter = 0**

**N,p = X.shape**

**P = np.random.randn(N,k)#Generating Random Numbers with Normal Distribution**

**P = P / np.dot(np.sum(P,1).reshape(N,1),np.ones((1,k)))**

**J\_prev = np.inf**

**J = []**

**while True:**

**t = pow(P,b)**

**#Here's a concrete iteration algorithm**

**C = np.dot(X.T, t).T / (sum(t, 0).reshape(k, 1) \* np.ones((1, p)))**

**dist = np.dot(np.sum(C\*C,1).reshape(k,1),np.ones((1,N))).T+\**

**np.sum(X\*X,1).reshape(N,1)\*np.ones((1, k)) - \**

**2\*np.dot(X,C.T)**

**t2 = pow(1.0/dist,1.0/(b-1))**

**P = t2/(np.sum(t2,1).reshape(N,1)\*np.ones((1,k)))**

**J\_cur = sum(sum((pow(P,b))\*dist,0),0)/ N**

**J.append(J\_cur)**

**print(iter, J\_cur)**

**if abs(J\_cur - J\_prev) < 0.001:**

**#Stop iteration when the error after iteration is less than 0.001**

**break**

**iter += 1**

**J\_prev = J\_cur**

**return C,dist,J**

**img = cv2.imread(r'data\_seg\1P262601208EFF89AHP2440R2M2.tif')**

**m,n,p = img.shape**

**img = img.astype(np.double)#This step is critical for converting to double type**

**img = img.reshape(m\*n,p)**

**C,dist,J = fcm(img,3,2)**

**#J is the error value of each iteration, which can be displayed by plot.**

**label = dist.argmin(axis=1)**

**print(label.shape)**

**img\_1 = C[label,:]**

**img\_2 = img\_1.reshape(m,n,p)**

**# print(type(img\_2))**

**plt.imshow(img\_2/255,cmap='gray')**

**plt.show()**