模式识别 K-means、Fuzzy C-means

一、动态聚类算法介绍

1.1 介绍

聚类就是按照某个特定标准(如距离准则)把一个数据集分割成不同的类或簇，使得同一个簇内的数据对象的相似性尽可能大，同时不在同一个簇中的数据对象的差异性也尽可能地大。即聚类后同一类的数据尽可能聚集到一起，不同数据尽量分离。这种基于相似度度量的聚类方法也是实际中更常用的方法，其中，根据算法设计的不同又可分为动态聚类法和分级聚类法等。

动态聚类方法是一种普遍采用的方法，它具有以下 3 个要点：

① 选定某种距离度量作为样本间的相似度度量。

② 确定某个评价聚类结果质量的准则函数。

③ 给定某个初始分类，然后用迭代算法找出使准则函数取极值的最好聚类结果。

1.2 K 均值算法（K-means)

（1）算法原理

K 均值(K-means)算法是一种很常用的聚类算法，其基本思想是，通过迭代寻找 k 个聚类的一种划分方案，使得用这 k 个聚类的均值来代表相应各类样本时所得到的总体误差最小。K 均值法在向量量化(例如对音频信号)和图像分隔等领域有广泛的应用，也有时被称作广义 Llogd 算法(GLA)。

算法原理：

K 均值算法的基础是最小误差平方和准则。

将 N 个样本$\{\boldsymbol{y_1}, \boldsymbol{y_2}, \cdots, \boldsymbol{y_N}\}$划分到 k 个类$\{\Gamma_1, \Gamma_2, \cdots, \Gamma_k\}$中，k 为一正整数，目标：使得各个数据与其对应聚类中心点的误差平方和最小：

$$J_e = \sum_{i=1}^{k} J_i = \sum_{i=1}^{k} \sum_{y \in \Gamma_i} ||\boldsymbol{y} - \boldsymbol{m_i}||^2$$

- J_e是误差平方和聚类准则。
- k 为聚类个数。
- $\boldsymbol{y}$是划分到类Γ_i的样本。
- $\boldsymbol{m_1}, \boldsymbol{m_2}, \cdots, \boldsymbol{m_k}$类$\Gamma_1, \Gamma_2, \cdots, \Gamma_k$的质心（均值向量）：

$$\boldsymbol{m_i} = \frac{1}{N_i} \sum_{y \in \Gamma_i} \boldsymbol{y}$$

- N_i是第i聚类Γ_i中的样本数目。

误差平方和无法用解析的方法最小化，只能用迭代的方法，通过不断调整样本的类别归属来求解。

（2）K-means 算法流程

① 初始化：随机选择 k 个样本点，并将其视为各聚类的初始中心$\boldsymbol{m_1}, \boldsymbol{m_2}, \cdots, \boldsymbol{m_k}$；

② 按照最小距离法则逐个将样本$\boldsymbol{y}$划分到以聚类中心$\boldsymbol{m_1}, \boldsymbol{m_2}, \cdots, \boldsymbol{m_k}$为代表的 k 个类$\Gamma_1, \Gamma_2, \cdots, \Gamma_k$中；

③ 计算聚类准则函数J_e，重新计算 k 个类的聚类中心$\boldsymbol{m_1}, \boldsymbol{m_2}, \cdots, \boldsymbol{m_k}$；

④ 重复 2 和 3 到聚类中心$\boldsymbol{m_1}, \boldsymbol{m_2}, \cdots, \boldsymbol{m_k}$无改变或目标函数$J_e$不减小。

（3）算法优缺点

1、优点：

① 简单、快速。

② 对处理大数据集，该算法是相对可伸缩和高效率的。

③ 当类密集，且类与类之间区别明显（比如球型聚集） 时，聚类效果很好；

2、缺点：

① 结果与初始聚类中心有关；

② 必须预先给出聚类的类别数 k；

③ 对“噪声”和孤立点数据敏感，少量的这些数据对平 均值产生较大的影响；

④ 不适合发现非凸面形状的聚类

1.2 模糊 C 均值算法（Fuzzy C-means）

（1）模糊分类

在普通模式识别中，分类就是把样本空间(或样本集) 分成若干个子集。在模糊模式识别中，用模糊子集代替确定子集，从而得到模糊的分类结果，即分类结果的模糊化，其中，一个样本以不同的程度属于各个类别，而不再属于某个确定的类别。

模糊分类与确定的分类结果相比，模糊化的分类结果主要有两个显著的优点:

① 可以反映出分类过程中的不确定性，有利于用户根据结果进行决策；

② 模糊化的分类结果比明确的分类结果中包含更多的信息，有利于进一步决策。

（2）FCM 算法原理

与 K 均值相比，模糊 C 均值聚类最大的特点在于：在 C 均值算法中，把硬划分变为模糊划分。

设$\mu_j(x_i)$是第i个样本x_i属于第j类G_j隶属度，利用隶属度定义的聚类准则函数为：

$$J_f = \sum_{j=1}^{C}\sum_{i=1}^{N}[\mu_j(x_i)]^b||\boldsymbol{x_i} - \boldsymbol{m_j}||^2 \quad (1)$$

其中，b>1 是一个可以控制聚类结果的模糊程度的常数。 约束条件为一个样本属于各个聚类的隶属度之和为 1,即:

$$\sum_{j=1}^{C}\mu_j(x_i) = 1 \quad (i = 1,2,\cdots,N) \quad (2)$$

利用拉格朗日乘数法来求解在条件（2）约束下，式（1）的极小值。

令优化的目标函数为：

$$L = \sum_{j=1}^{C}\sum_{i=1}^{N}[\mu_j(x_i)]^b||\boldsymbol{x_i} - \boldsymbol{m_j}||^2 + \sum_{i=1}^{N}\lambda_i[\sum_{j=1}^{C}\mu_j(\boldsymbol{x_i}) - 1] \quad (3)$$

分别求 L 对$\boldsymbol{m_j}$、λ_i和$\mu_j(x_i)$的梯度 (或偏导)，并置为 0，可得必要条件：

$$\boldsymbol{m_j} = \frac{\sum_{i=1}^{N}[\mu_j(\boldsymbol{x_i})]^b\boldsymbol{x_i}}{\sum_{i=1}^{N}[\mu_j(\boldsymbol{x_i})]^b} \quad (j = 1,2,\cdots,C) \quad (4)$$

$$\mu_j(\boldsymbol{x_i}) = \frac{(1/||\boldsymbol{x_i} - \boldsymbol{m_j}||^2)^{1/(b-1)}}{\sum_{l=1}^{C}(1/||\boldsymbol{x_i} - \boldsymbol{m_l}||^2)^{1/(b-1)}} \quad (5)$$

（3）FCM 算法流程

① 设定聚类数目 C、参数 b 和一个适当的小数 ε>0，通常取 1<b≤5。

② 初始化各个聚类中心m^s，迭代次数 s=0。

③ 根据式（5）更新隶属度函数：$U^{(s)}=\{\mu_{ij}^{(s)}\}$

$$\mu_{ij}^{(s)}=\mu_j^{(s)}(\boldsymbol{x_i})=\frac{(1/||\boldsymbol{x_i}-\boldsymbol{m_j}||^2)^{1/(b-1)}}{\sum_{l=1}^{C}(1/||\boldsymbol{x_i}-\boldsymbol{m_l}||^2)^{1/(b-1)}}$$

④ 根据式（4）更新聚类中心$m^{(s+1)}$:

$$\boldsymbol{m_j}=\frac{\sum_{i=1}^{N}[\mu_j(\boldsymbol{x_i})]^b\boldsymbol{x_i}}{\sum_{i=1}^{N}[\mu_j(\boldsymbol{x_i})]^b}$$

⑤ 如果$\|m(s)-m(s+1)\|<\varepsilon$,则停止迭代,输出聚类中心和隶属度函数,否则$s=s+1$, 返回步骤（3）。

⑥ 输出：将样本点划分为隶属度最大的那一类。

二、实验数据集介绍

2.1 Iris 数据集介绍

Iris 数据集			
类别	3		
维度	4		
数据长度	150	50	Iris-setosa
		50	Iris-versicolor
		50	Iris-setosa

2.2 Sonar 数据集介绍

Sonar 数据集			
类别	2		
维度	60		
数据长度	208	97	R
		111	M

三、实验设置

对两类数据集，均分别采用 K-means、FCM 两种方法进行聚类，显示最终的聚类结果。

两类数据集处理方法具有相似性，因此实验设置基本一致，具体如下：

3.1 K-means

1、读取数据信息。首先从数据集文件（iris.data、sonar.all-data）中读取数据，由于是聚类的缘故，将所有数据均放在训练集中

2、随机选取 K 个聚类中心,并初始化聚类名称与聚类均值。在样本中随机选取 K 个标号作为初始聚类中心和聚类均值。

3、以准则函数作为迭代结束依据，进行迭代。

（1）按最小距离准则划分样本

（2）计算均值，更新聚类中心，计算准则函数值

4、迭代结束，按照数据标签，统计分类结果标签并计算准确率。

5、重复 1-4 步 10 次，计算平均准确率，绘图显示准确率、准则函数变化情况、聚类结果。

3.2 FCM

1、读取数据信息。首先从数据集文件（iris.data、sonar.all-data）中读取数据，由于是聚类的缘故，将所有数据均放在训练集中

2、随机选取 C 个聚类中心,初始化聚类中心(均值向量)、b 和终止条件。在样本中随机选取 C 个标号作为初始聚类中心和聚类均值。b 默认为 2

3、以$\|m(s) - m(s+1)\| < \varepsilon$**作为迭代结束依据，进行迭代。**

（1）按照公式计算，更新隶属度函数

（2）按照公式计算，更新聚类中心

4、迭代结束，根据隶属度函数划分样本。

5、划分结束，统计分类结果标签数目并计算准确率。

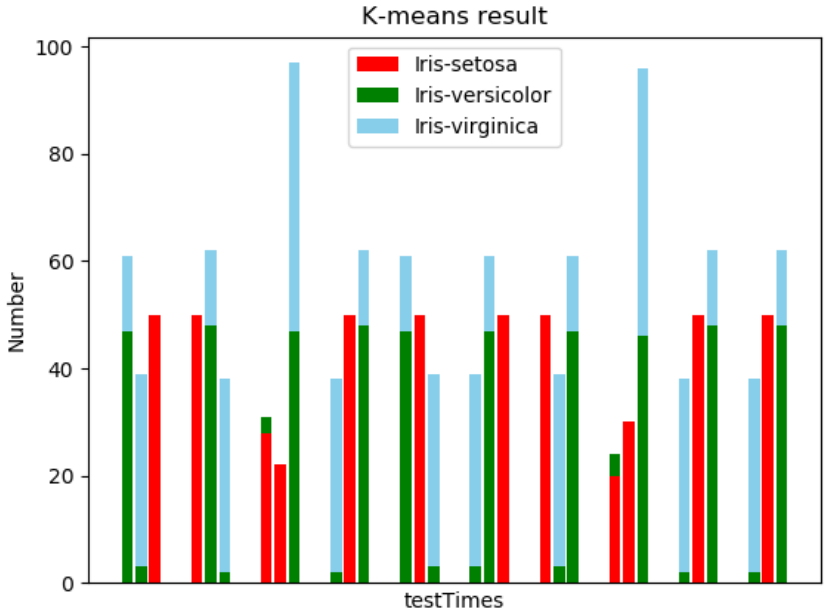
6、重复 1-5 步 10 次，计算平均准确率，绘图显示准确率变化趋势，绘图显示 10 次聚类结果。

四、实验结果展示与分析

4.1 K-means 聚类结果与分析

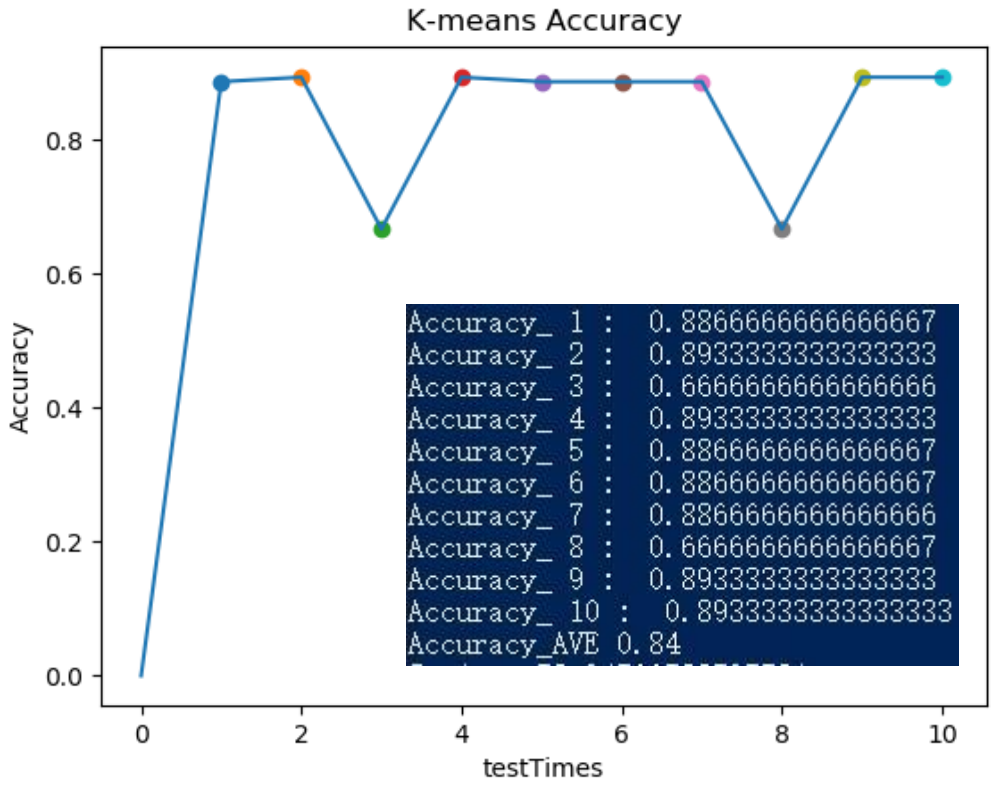
10 次聚类结果绘图及最后一次聚类数据展示如下，可以看出因初始聚类中心的选取不同，每次聚类的结果并不相同。对于 iris 数据集，可以得到不错的聚类结果，但 sonar 数据集却不能很好分开：

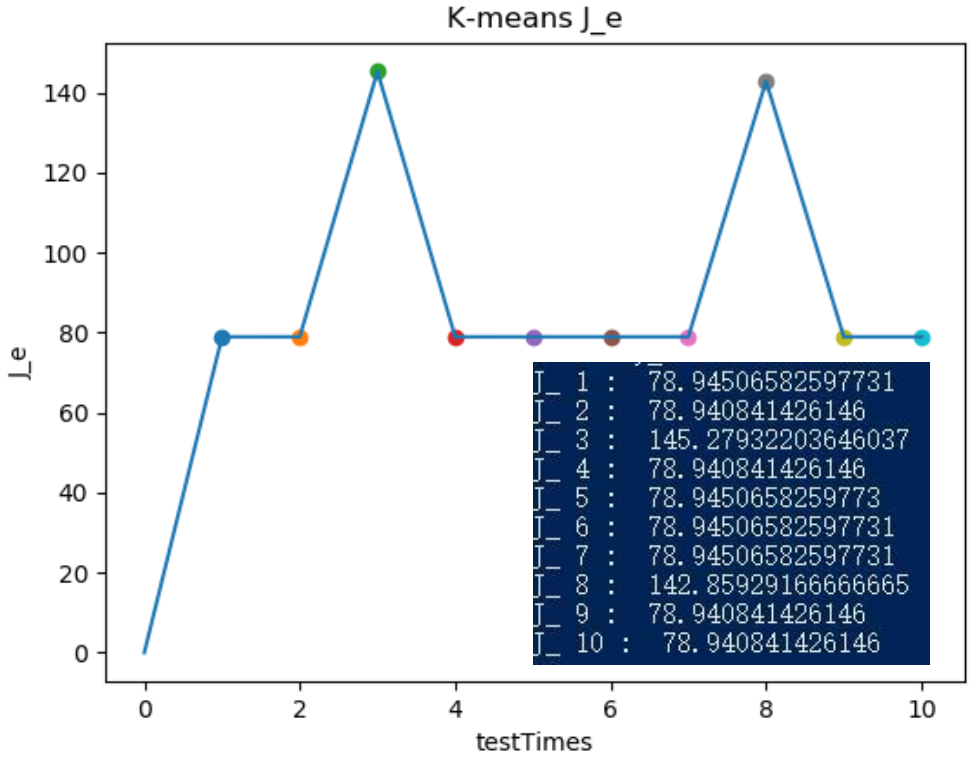
Iris 数据集 10 次 K 均值聚类结果绘图、最后一次聚类数据展示如下：



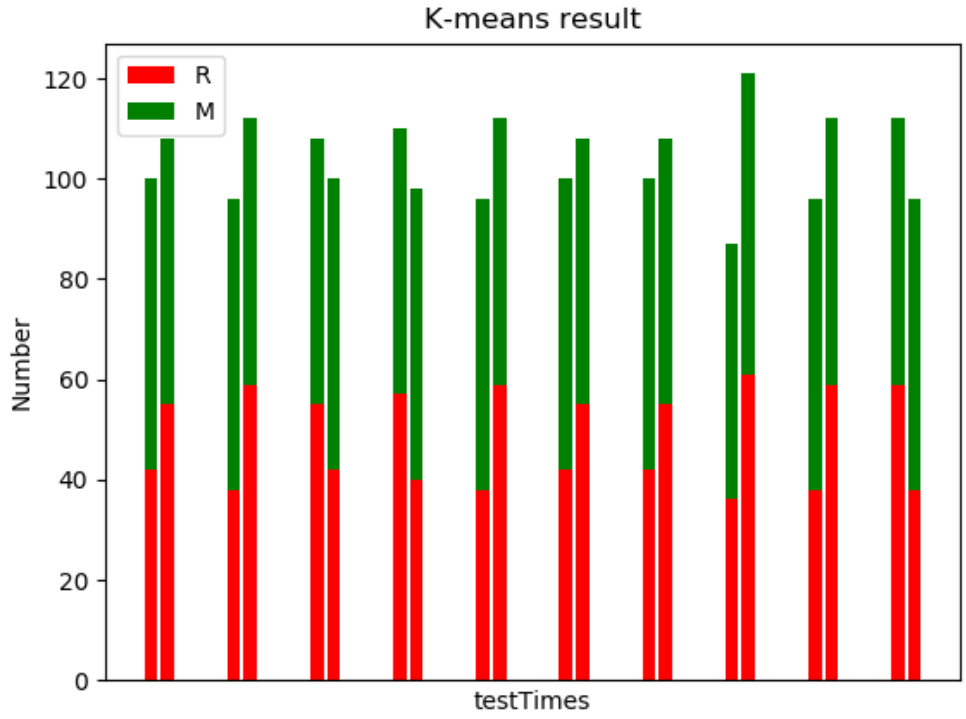

```
*****************************    category_0    *************************
clusteringCenter:  [6.85       3.07368421 5.74210526 2.07105263]
clusteringError:  23.879473684210517
dataSize:  38
Iris-setosa:     0.0
Iris-versicolor: 2.0
Iris-virginica:  36.0


*****************************    category_1    *************************
clusteringCenter:  [5.006 3.418 1.464 0.244]
clusteringError:  15.240400000000003
dataSize:  50
Iris-setosa:     50.0
Iris-versicolor: 0.0
Iris-virginica:  0.0


*****************************    category_2    *************************
clusteringCenter:  [5.9016129  2.7483871  4.39354839 1.43387097]
clusteringError:  39.820967741935476
dataSize:  62
Iris-setosa:     0.0
Iris-versicolor: 48.0
Iris-virginica:  14.0
```

Iris 数据集 10 次 K 均值聚类准确率、最终准则函数展示如下：



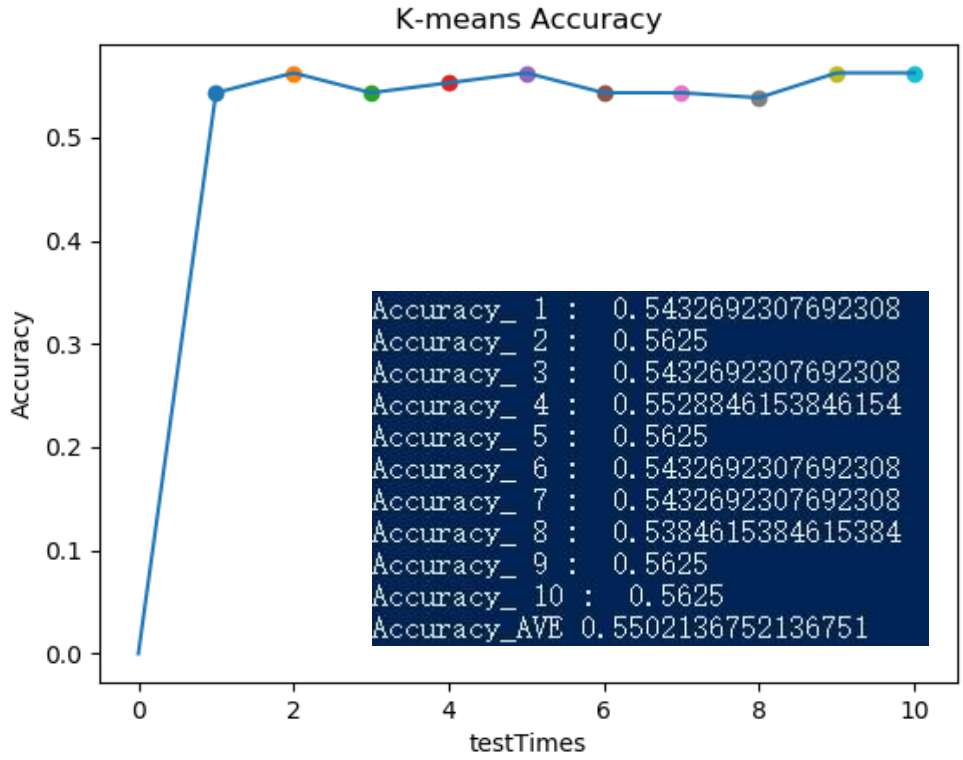


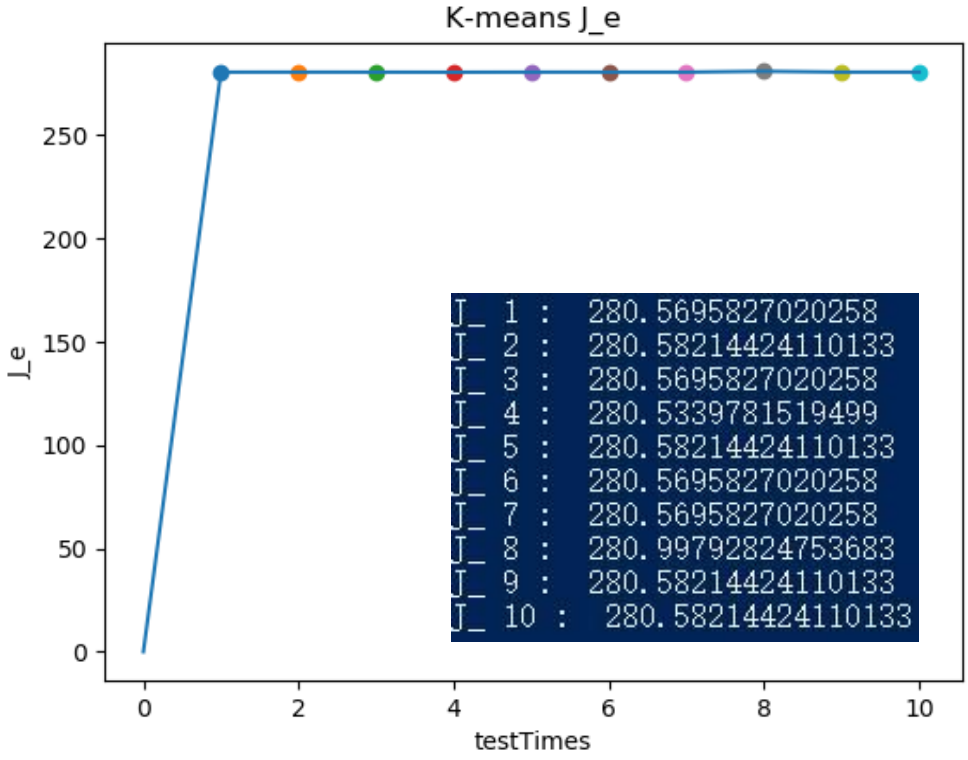
sonar 数据集 10 次 K 均值聚类结果绘图、最后一次聚类数据展示如下:




```
*************************    category_0    ***********************
clusteringCenter:  [0.02809375 0.03710446 0.04063036 0.04789732 0.06322321 0.09584196
 0.11261607 0.12531518 0.16587321 0.18323393 0.20047679 0.21399911
 0.22199464 0.21493839 0.21180804 0.24291071 0.24894107 0.274125
 0.32571161 0.40181429 0.45706875 0.47655982 0.53693036 0.60590625
 0.64839554 0.70831875 0.73542679 0.7733625  0.75377679 0.69890536
 0.61225893 0.536175   0.51434286 0.50071161 0.48729018 0.47442054
 0.43283214 0.40232143 0.37589286 0.34644821 0.33530714 0.32382768
 0.28100982 0.239825   0.2294625  0.19203839 0.13647232 0.1012
 0.05463482 0.02093571 0.01619554 0.01363661 0.01059018 0.01045089
 0.00903036 0.00774464 0.00756964 0.00746429 0.00807857 0.00670804]
clusteringError:  162.67965261651793
dataSize:  112
R: 59.0
M: 53.0


*************************    category_1    ***********************
clusteringCenter:  [0.0304125  0.03999063 0.04756771 0.06088646 0.08917812 0.11475313
 0.13239896 0.14586354 0.19215417 0.23745521 0.27747188 0.29248021
 0.33316771 0.39180312 0.44666042 0.53665833 0.61086563 0.66021042
 0.71376146 0.75115104 0.78638333 0.79660938 0.77536042 0.75052708
 0.70695625 0.69000521 0.6633375  0.6014625  0.51175417 0.4432875
 0.37872813 0.32571667 0.30390937 0.28950833 0.28206563 0.28034583
 0.28327708 0.26654896 0.26735729 0.27009167 0.23552083 0.22516979
 0.20632917 0.18403333 0.15963021 0.12398958 0.10609688 0.08001875
 0.04877187 0.01982708 0.01592083 0.01316771 0.01084792 0.0115125
 0.00959375 0.00877812 0.0081125  0.00851458 0.00778125 0.00627292]
clusteringError:  117.90249162458338
dataSize:  96
R: 38.0
M: 58.0
```

10 次 K 均值聚类准确率、最终准则函数展示如下：

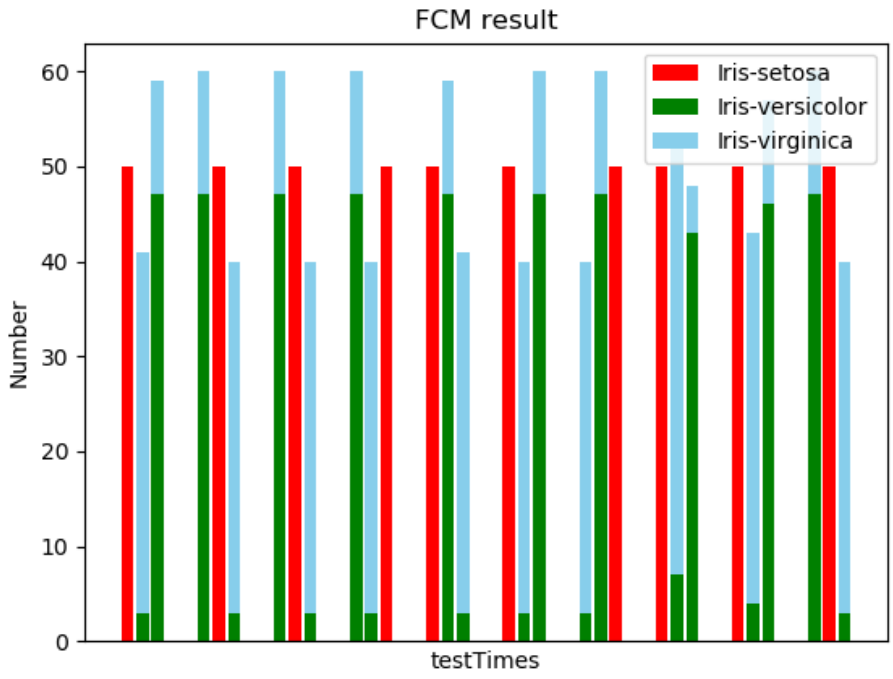




4.2 FCM 聚类结果与分析

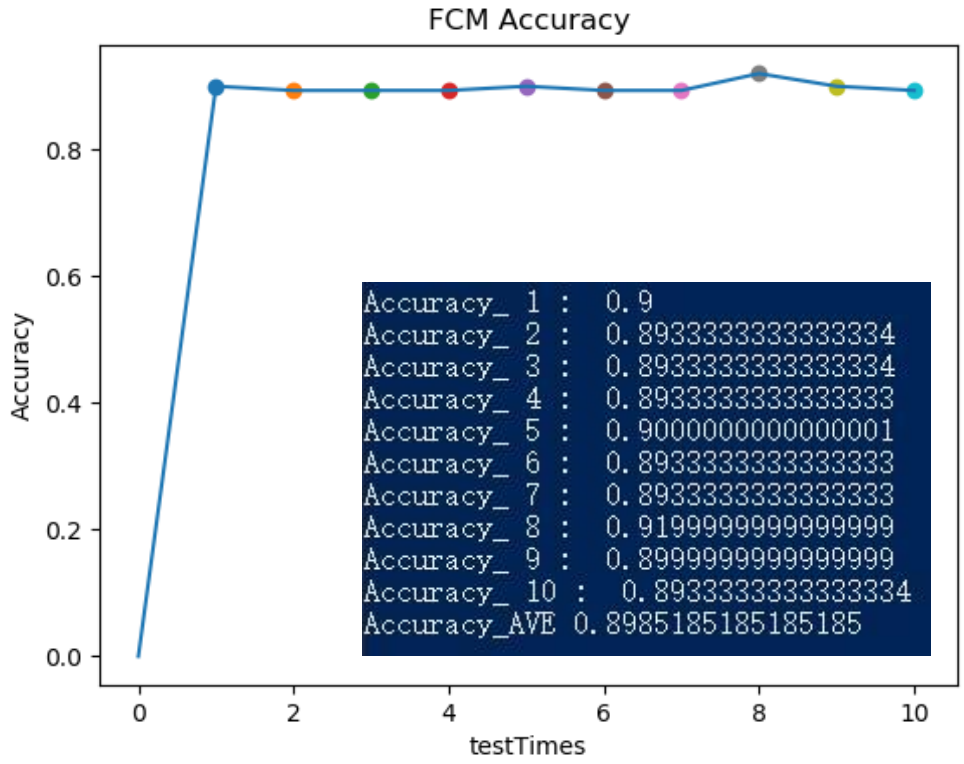
10 次模糊 C 均值聚类结果绘图及最后一次聚类数据展示如下，可以看出因初始聚类中心的选取不同，每次聚类的结果并不相同。相比于 K 均值聚类，聚类结果和准确率有所改善，但和 K 均值聚类相似，对于 iris 数据集，可以得到不错的聚类结果，但 sonar 数据集却不能很好分开：

Iris 数据集 10 次模糊 C 均值聚类结果绘图、最后一次聚类数据展示如下：

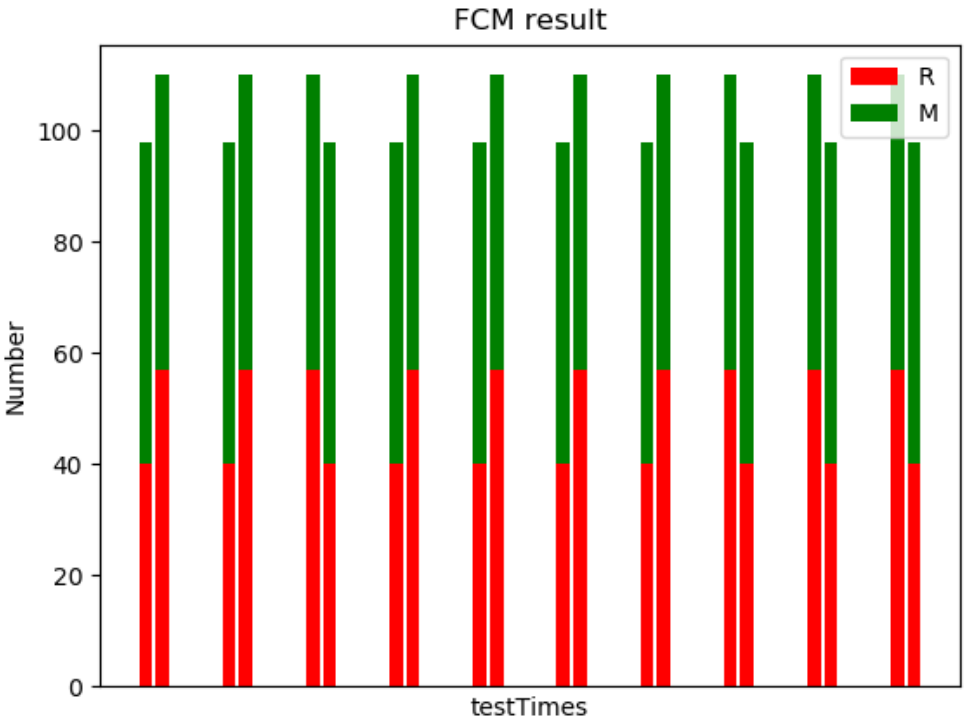



```
*****************************    category_0    *************************
clusteringMeans:  [5.88837358 2.76093727 4.36304809 1.39682455]
dataSize:  60
Iris-setosa:     0
Iris-versicolor: 47
Iris-virginica:  13


*****************************    category_1    *************************
clusteringMeans:  [5.00355891 3.40307513 1.48493467 0.25151093]
dataSize:  50
Iris-setosa:     50
Iris-versicolor: 0
Iris-virginica:  0


*****************************    category_2    *************************
clusteringMeans:  [6.77412101 3.05214004 5.64566987 2.05312743]
dataSize:  40
Iris-setosa:     0
Iris-versicolor: 3
Iris-virginica:  37
```

10 次模糊 C 均值聚类准确率展示如下：

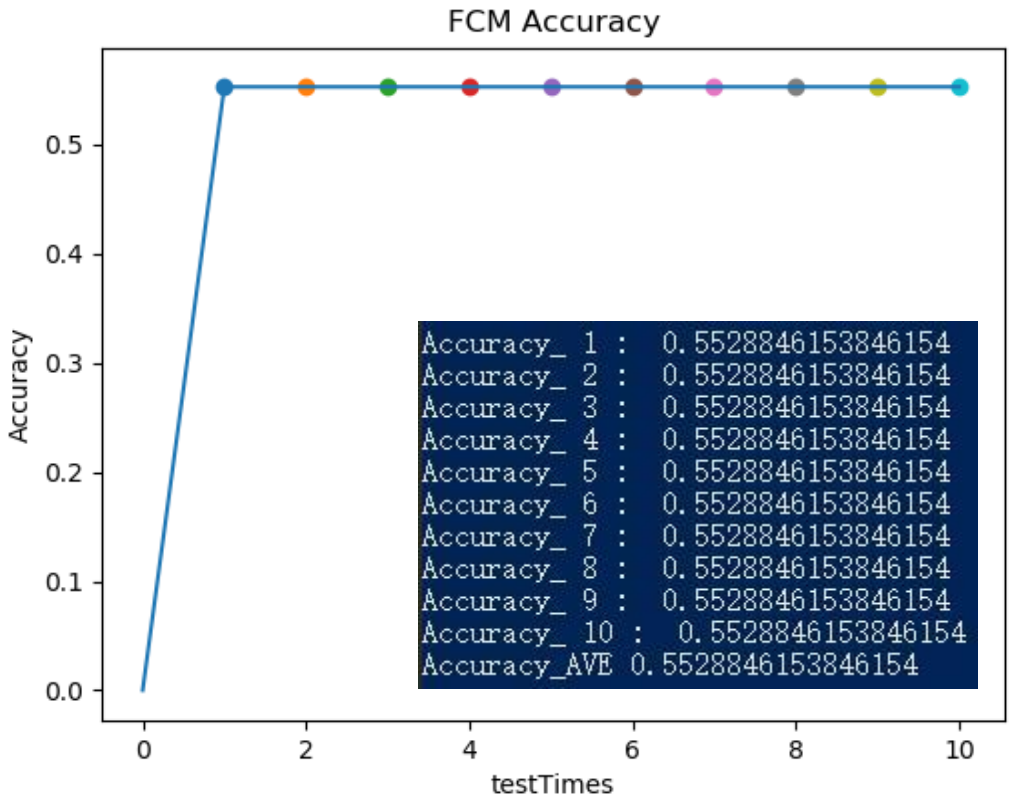


sonar 数据集 10 次模糊 C 均值聚类结果绘图、最后一次聚类数据展示如下：



```
*************************    category_0    **********************
clusteringMeans:  [0.02782464 0.03661911 0.04098896 0.04928634 0.06717555 0.09840742
 0.11522182 0.12788938 0.17084081 0.1981941  0.22077089 0.23488011
 0.24909407 0.25649363 0.26335712 0.30599975 0.32904948 0.35991024
 0.40966472 0.47229021 0.52279884 0.54652411 0.59357058 0.6418337
 0.66353083 0.70498212 0.72479317 0.74270143 0.70477094 0.64300093
 0.56582113 0.49572273 0.47125834 0.45535739 0.43946781 0.42482183
 0.39148828 0.36160037 0.34279022 0.32528034 0.30810309 0.29724867
 0.26395464 0.22766567 0.210917   0.17393987 0.1291704  0.09622162
 0.05317781 0.02053581 0.01614333 0.01342518 0.0104924  0.01036255
 0.00900302 0.00790569 0.00767362 0.00757948 0.00782385 0.00643801]
dataSize:  110
R: 57
M: 53


*************************    category_1    **********************
clusteringMeans:  [0.03013525 0.03959529 0.04612594 0.0582233  0.08373122 0.11107193
 0.12815011 0.14027266 0.1826362  0.21509815 0.24839086 0.26369203
 0.29682299 0.33857746 0.38126337 0.45582725 0.50995962 0.55263082
 0.60712607 0.65767099 0.69824445 0.70694608 0.70531101 0.70701232
 0.68955695 0.69605309 0.68007728 0.6439246  0.58019686 0.52008953
 0.44281881 0.37994398 0.36157398 0.34985022 0.34334523 0.34087533
 0.33247148 0.31353274 0.30440009 0.2946801  0.26947091 0.25799356
 0.22722914 0.19874047 0.18080366 0.14485317 0.11474012 0.08574569
 0.05015092 0.02011055 0.01588994 0.01331455 0.01086957 0.01152124
 0.0094866  0.00853607 0.00793727 0.00823504 0.0080432  0.00654346]
dataSize:  98
R: 40
M: 58
```

10 次模糊 C 均值聚类准确率展示如下：



五、Python 代码

5.1 iris 数据集

```
import os
import sys
import numpy as np
from numpy import *
import random
import matplotlib.pyplot as plt

def readData(DATA_PATH,DATA_SIZE,DATA_DIMENSION,test_rate):
    test_Labels = random.sample(range(0,DATA_SIZE-
1),int(DATA_SIZE*test_rate))
    test_Data = [[0 for i in range(DATA_DIMENSION+1)] for j in range(le
n(test_Labels))]
    train_Data = [[0 for i in range(DATA_DIMENSION+1)] for j in range(D
ATA_SIZE-len(test_Labels))]
    #随机提取训练、测试数据
    f = open(DATA_PATH)
    line = f.readline()
    i = 0
    train_Label = 0
    test_Label = 0
    while line:
        temp = line.split(",",DATA_DIMENSION)
        if i in test_Labels:
            test_Data[test_Label][0:DATA_DIMENSION] = list(map(float, t
emp[0:DATA_DIMENSION]))
            test_Data[test_Label][DATA_DIMENSION] = temp[DATA_DIMENSION
]
            test_Label = test_Label+1
        else:
            train_Data[train_Label][0:DATA_DIMENSION] = list(map(float,
 temp[0:DATA_DIMENSION]))
            train_Data[train_Label][DATA_DIMENSION] = temp[DATA_DIMENSI
ON]
            train_Label = train_Label+1
        i = i+1
        line = f.readline()
    #将要训练的数据分类保存
    _1_Num = _2_Num = _3_Num = 0
    for i in range(len(train_Data)):
        if train_Data[i][DATA_DIMENSION] == 'Iris-setosa\n':
```

```
            _1_Num = _1_Num+1
        if train_Data[i][DATA_DIMENSION] == 'Iris-versicolor\n':
            _2_Num = _2_Num+1
        if train_Data[i][DATA_DIMENSION] == 'Iris-virginica\n':
            _3_Num = _3_Num+1
    _1_Data = mat(np.zeros((_1_Num,DATA_DIMENSION)))
    _2_Data = mat(np.zeros((_2_Num,DATA_DIMENSION)))
    _3_Data = mat(np.zeros((_3_Num,DATA_DIMENSION)))
    _1_Num = _2_Num = _3_Num = 0
    for i in range(len(train_Data)):
        if train_Data[i][DATA_DIMENSION] == 'Iris-setosa\n':
            _1_Data[_1_Num] = train_Data[i][0:DATA_DIMENSION]
            _1_Num = _1_Num+1
        if train_Data[i][DATA_DIMENSION] == 'Iris-versicolor\n':
            _2_Data[_2_Num] = train_Data[i][0:DATA_DIMENSION]
            _2_Num = _2_Num+1
        if train_Data[i][DATA_DIMENSION] == 'Iris-virginica\n':
            _3_Data[_3_Num] = train_Data[i][0:DATA_DIMENSION]
            _3_Num = _3_Num+1
    f.close()
    return train_Data,test_Data,_1_Data.T,_2_Data.T,_3_Data.T

#计算欧氏距离
def calcu_Eucli_Dis(x,y):
    dis = np.sqrt(np.sum(np.square(np.array(x)-np.array(y))))
    return dis

#计算样本到均值向量的距离平方和
def calcu_error(samples, mean):
    sum=0
    for i in range(len(samples)):
        e = np.sum(np.square(np.array(samples[i])-np.array(mean)))
        sum += e

    return sum


#*************************************************************K-
means*************************************************************
**
#K-means 类结构
class Data_K_means:
    def __init__(self):
        self.name = ''
```

```
        self.clusteringCenter = []  #聚类中心
        self.clusteringData = []    #包含的样本数据
        self.clusteringMeans = 0    #均值
        self.clusteringError = 0    #每一类中的最小误差平方和

        self.Iris_setosa = 0
        self.Iris_versicolor = 0
        self.Iris_virginica = 0

    def print(self):
        print("*************************   ",self.name,"   ***********
************")

        print("clusteringCenter: ",self.clusteringCenter)
        print("clusteringError: "  ,self.clusteringError)
        # print("clusteringData: \n"  ,np.array(self.clusteringData))
        print("dataSize: "  ,len(self.clusteringData))

#按最小距离准则划分样本到 N 类
def divide_Sample(train_Data,category,dataDimension,K):
    for i in range(K):
        category[i].clusteringData = []
    for i in range(len(train_Data)):
        min_dis = 1024
        min_dis_label = -1
        for j in range(K):
            dis = calcu_Eucli_Dis(train_Data[i][:dataDimension],categor
y[j].clusteringCenter)
            if dis<min_dis:
                min_dis = dis
                min_dis_label = j
        # print("min_dis:",min_dis,"label:",min_dis_label)
        category[min_dis_label].clusteringData.append(train_Data[i])

#更新参数：计算均值与准则函数，更新聚类中心,返回准则函数
def update_Cluster_Center(category,dataDimension,K):
    #计算每一类的均值、误差 J_i,计算出准则函数
    J = 0
    for i in range(K):
        dataArray = np.array(np.array(category[i].clusteringData)[:,0:d
ataDimension],dtype=np.float64)
        #计算均值
        category[i].clusteringMeans = mean(dataArray,0)
```

```
        #更新聚类中心
        category[i].clusteringCenter = category[i].clusteringMeans
        #计算每一类内距离差
        error = calcu_error(dataArray,category[i].clusteringMeans)
        category[i].clusteringError = error
        J += error
    return J

def k_Means(dataPath,dataSize,dataDimension,K,testRate=0):
    #提取数据
    train_Data,test_Data,group1,group2,group3 = readData(dataPath,dataS
ize,dataDimension,testRate)
    print("train_Data SIZE:",len(train_Data))

    #随机选取 K 个聚类中心,并初始化聚类名称与聚类均值
    clustering_center_label = random.sample(range(0,len(train_Data)-
1),K)
    print("clustering_center_label:",clustering_center_label)
    category = [Data_K_means() for i in range(K)]
    for i in range(K):
        category[i].clusteringCenter = train_Data[clustering_center_lab
el[i]][:dataDimension]
        category[i].clusteringMeans = category[i].clusteringCenter
        category[i].name = "category_"+str(i)
        # category[i].print()

    J = 1024.0
    J_last = -1024.0
    repeat_time = 0
    #开始迭代
    while J!=J_last:    #以准则函数作为迭代结束依据
        J_last = J
        repeat_time += 1

        #按最小距离准则划分样本
        divide_Sample(train_Data,category,dataDimension,K)

        #计算均值，更新聚类中心，计算准则函数值
        J = update_Cluster_Center(category,dataDimension,K)

        print("-----------------------------------------------------
------------------>Num: ",repeat_time)
        # for i in range(K):
        #     category[i].print()
```

```
        print("J = ",J)
        print("J_last = ",J_last,"\n\n")

    #迭代结束，统计分类结果标签数目 并 计算准确率
    accuracy = 0
    for k in range(K):
        _1_Num = _2_Num = _3_Num = 0.0
        for i in range(len(category[k].clusteringData)):
            if category[k].clusteringData[i][dataDimension] == 'Iris-
setosa\n':
                _1_Num += 1
            if category[k].clusteringData[i][dataDimension] == 'Iris-
versicolor\n':
                _2_Num += 1
            if category[k].clusteringData[i][dataDimension] == 'Iris-
virginica\n':
                _3_Num += 1
        category[k].print()
        print("Iris-setosa:    ",_1_Num)
        print("Iris-versicolor:",_2_Num)
        print("Iris-virginica: ",_3_Num,"\n\n")
        category[k].Iris_setosa = _1_Num
        category[k].Iris_versicolor = _2_Num
        category[k].Iris_virginica = _3_Num

        if _1_Num>=_2_Num and _1_Num>=_3_Num:
            accuracy += _1_Num/len(category[k].clusteringData) * len(ca
tegory[k].clusteringData)/len(train_Data)
        elif _2_Num>=_1_Num and _2_Num>=_3_Num:
            accuracy += _2_Num/len(category[k].clusteringData) * len(ca
tegory[k].clusteringData)/len(train_Data)
        else:
            accuracy += _3_Num/len(category[k].clusteringData) * len(ca
tegory[k].clusteringData)/len(train_Data)

    return category,J,accuracy


def test_K_Means():
    J = [0]
    accuracy = [0]
    testTimes = 10

    #绘制聚类结果
```

```
    plt.figure()
    for i in range(testTimes):
        category,J_temp,accuracy_temp = k_Means("iris.data",150,4,K=3)
   #进行 K 均值聚类,返回聚类结果、误差平方和、精准度
        J.append(J_temp)
        accuracy.append(accuracy_temp)

        for k in range(3):
            p1 = plt.bar(x=k+(i*5), height=category[k].Iris_setosa, lab
el='Iris-setosa', color="red")
            p2 = plt.bar(x=k+(i*5), height=category[k].Iris_versicolor,
 label='Iris-versicolor', color="green",bottom=category[k].Iris_setosa)
            p3 = plt.bar(x=k+(i*5), height=category[k].Iris_virginica,
label='Iris-
virginica', color="skyblue",bottom=category[k].Iris_setosa+category[k].
Iris_versicolor)
        plt.legend((p1[0], p2[0], p3[0]), ('Iris-setosa', 'Iris-
versicolor', 'Iris-virginica'))
    plt.ylabel('Number')
    plt.xlabel("testTimes")
    plt.xticks([])
    plt.title('K-means result')
    plt.show(block = False)

    #绘制聚类精确度
    plt.figure()
    for i in range(1,testTimes+1):
        plt.scatter(i,accuracy[i])
        print("Accuracy_",i,": ",accuracy[i])
    print("Accuracy_AVE",mean(accuracy[1:testTimes]))
    plt.title("K-means Accuracy")
    plt.xlabel("testTimes")
    plt.ylabel("Accuracy")
    plt.plot(accuracy)
    plt.show(block = False)

    #绘制聚类最终准则函数值（#最小误差平方和）
    plt.figure()
    for i in range(1,testTimes+1):
        plt.scatter(i,J[i])
        print("J_",i,": ",J[i])
    plt.title("K-means J_e")
    plt.xlabel("testTimes")
    plt.ylabel("J_e")
```

```
    plt.plot(J)
    plt.show()

#*************************************************************FCM*******
************************************************************
class Data_FCM:
    def __init__(self):
        self.name = ''

        self.Data = []          #最终包含的样本数据
        self.Membership = []    #隶属度
        self.Means = 0          #均值/聚类中心

        self.Iris_setosa = 0
        self.Iris_versicolor = 0
        self.Iris_virginica = 0

    def print(self):
        print("*************************   ",self.name,"   **********
************")

        print("clusteringMeans: ",self.Means)
        # print("clusteringData: \n"  ,np.array(self.clusteringData))
        print("dataSize: "  ,len(self.Data))

#计算 更新隶属度函数
def update_Membership(train_Data,category,dataDimension,C,b):
    trainDataArray = np.array(np.array(train_Data)[:,0:dataDimension],d
type=np.float64)
    for j in range(C):  #u_j(x_i)
        category[j].Membership = []     #清空隶属度向量
        for i in range(len(train_Data)):
            #计算分母、分子
            denom = 0.0
            for l in range(C):
                denom += pow((1/np.square(np.linalg.norm(np.array(train
DataArray[i])-np.array(category[l].Means)))),1/(b-1))
            numer = pow((1/np.square(np.linalg.norm(np.array(trainDataA
rray[i])-np.array(category[j].Means)))),1/(b-1))
            #更新隶属度
            if numer == inf:    #解决自己与自己隶属度计算出错问题
                category[j].Membership.append(1.0)
            else:
                category[j].Membership.append(numer/denom)
```

```
        # print(j,np.array(category[j].Membership))

#计算 更新聚类中心
def update_Cluster_Center_FCM(train_Data,category,dataDimension,C,b):
    trainDataArray = np.array(np.array(train_Data)[:,0:dataDimension],d
type=np.float64)
    for j in range(C):
        #计算分母、分子
        denom = 0.0
        numer = 0.0
        for i in range(len(train_Data)):
            denom += pow(category[j].Membership[i],b)
            numer += pow(category[j].Membership[i],b) * trainDataArray[
i]
        # print(j,":",numer,denom,numer/denom)
        category[j].Means = numer/denom

#根据隶属度函数划分样本
def divide_Sample_FCM(train_Data,category,C):
    for i in range(len(train_Data)):
        membership_max = -1
        label = -1
        for j in range(C):
            if category[j].Membership[i] > membership_max:
                membership_max = category[j].Membership[i]
                label = j
        category[label].Data.append(train_Data[i])

def FCM(dataPath,dataSize,dataDimension,C,b=2,end_condi=0.001,testRate=
0):
    #提取数据
    train_Data,test_Data,group1,group2,group3 = readData(dataPath,dataS
ize,dataDimension,testRate)
    print("train_Data SIZE:",len(train_Data))

    #随机选取 C 个聚类中心,初始化聚类中心(均值向量)
    clustering_center_label = random.sample(range(0,len(train_Data)-
1),C)
    print("clustering_center_label:",clustering_center_label)
    category = [Data_FCM() for i in range(C)]
    for i in range(C):
        category[i].Means = train_Data[clustering_center_label[i]][:dat
aDimension]
        category[i].name = "category_"+str(i)
```

```
        category[i].print()

    m = [1024 for i in range(dataDimension)]
    m_last = [-1024 for i in range(dataDimension)]
    repeat_time = 0
    #开始迭代
    while calcu_Eucli_Dis(m,m_last) > end_condi:
        repeat_time += 1
        m_last = m

        #计算，更新隶属度
        update_Membership(train_Data,category,dataDimension,C,b)

        #计算均值，更新聚类中心
        update_Cluster_Center_FCM(train_Data,category,dataDimension,C,b
)

        m = category[0].Means

        print("------------------------------------------------------------
--------------------->Num: ",repeat_time)
        print("m=",m)
        print("m_last = ",m_last)
    #迭代结束，划分样本
    divide_Sample_FCM(train_Data,category,C)

    #划分结束，统计分类结果标签数目 并 计算准确率
    accuracy = 0
    for k in range(C):
        _1_Num = _2_Num = _3_Num = 0
        for i in range(len(category[k].Data)):
            if category[k].Data[i][dataDimension] == 'Iris-setosa\n':
                _1_Num += 1
            if category[k].Data[i][dataDimension] == 'Iris-
versicolor\n':
                _2_Num += 1
            if category[k].Data[i][dataDimension] == 'Iris-
virginica\n':
                _3_Num += 1
        category[k].print()
        print("Iris-setosa:    ",_1_Num)
        print("Iris-versicolor:",_2_Num)
        print("Iris-virginica: ",_3_Num,"\n\n")
        category[k].Iris_setosa = _1_Num
```

```
        category[k].Iris_versicolor = _2_Num
        category[k].Iris_virginica = _3_Num

        if _1_Num>=_2_Num and _1_Num>=_3_Num:
            accuracy += _1_Num/len(category[k].Data) * len(category[k].
Data)/len(train_Data)
        elif _2_Num>=_1_Num and _2_Num>=_3_Num:
            accuracy += _2_Num/len(category[k].Data) * len(category[k].
Data)/len(train_Data)
        else:
            accuracy += _3_Num/len(category[k].Data) * len(category[k].
Data)/len(train_Data)

    return category,accuracy

def test_FCM():
    accuracy = [0]
    testTimes = 10

    #绘制聚类结果
    plt.figure()
    for i in range(testTimes):
        category,accuracy_temp = FCM("iris.data",150,4,C=3,b=2,end_cond
i=0.001)    #进行 K 均值聚类,返回聚类结果、误差平方和、精准度
        accuracy.append(accuracy_temp)

        for k in range(3):
            p1 = plt.bar(x=k+(i*5), height=category[k].Iris_setosa, lab
el='Iris-setosa', color="red")
            p2 = plt.bar(x=k+(i*5), height=category[k].Iris_versicolor,
 label='Iris-versicolor', color="green",bottom=category[k].Iris_setosa)
            p3 = plt.bar(x=k+(i*5), height=category[k].Iris_virginica,
label='Iris-
virginica', color="skyblue",bottom=category[k].Iris_setosa+category[k].
Iris_versicolor)
        plt.legend((p1[0], p2[0], p3[0]), ('Iris-setosa', 'Iris-
versicolor', 'Iris-virginica'))
    plt.ylabel('Number')
    plt.xlabel("testTimes")
    plt.xticks([])
    plt.title('FCM result')
    plt.show(block = False)

    #绘制聚类精确度
```

```
    plt.figure()
    for i in range(1,testTimes+1):
        plt.scatter(i,accuracy[i])
        print("Accuracy_",i,": ",accuracy[i])
    print("Accuracy_AVE",mean(accuracy[1:testTimes]))
    plt.title("FCM Accuracy")
    plt.xlabel("testTimes")
    plt.ylabel("Accuracy")
    plt.plot(accuracy)
    plt.show()


if __name__ == '__main__':
    # test_K_Means()
    test_FCM()
```

5.2 sonar 数据集（与 iris 相差无几）

```
import os
import sys
import numpy as np
from numpy import *
import random
import matplotlib.pyplot as plt

def readData(DATA_PATH,DATA_SIZE,DATA_DIMENSION,test_rate):
    test_Labels = random.sample(range(0,DATA_SIZE-
1),int(DATA_SIZE*test_rate))
    test_Data = [[0 for i in range(DATA_DIMENSION+1)] for j in range(le
n(test_Labels))]
    train_Data = [[0 for i in range(DATA_DIMENSION+1)] for j in range(D
ATA_SIZE-len(test_Labels))]
    #随机提取训练、测试数据
    f = open(DATA_PATH)
    line = f.readline()
    i = 0
    train_Label = 0
    test_Label = 0
    while line:
        temp = line.split(",",DATA_DIMENSION)
        if i in test_Labels:
            test_Data[test_Label][0:DATA_DIMENSION] = list(map(float, t
emp[0:DATA_DIMENSION]))
```

```
            test_Data[test_Label][DATA_DIMENSION] = temp[DATA_DIMENSION
]
            test_Label = test_Label+1
        else:
            train_Data[train_Label][0:DATA_DIMENSION] = list(map(float,
 temp[0:DATA_DIMENSION]))
            train_Data[train_Label][DATA_DIMENSION] = temp[DATA_DIMENSI
ON]
            train_Label = train_Label+1
        i = i+1
        line = f.readline()
    #将要训练的数据分类保存
    _1_Num = _2_Num = _3_Num = 0
    for i in range(len(train_Data)):
        if train_Data[i][DATA_DIMENSION] == 'R\n':
            _1_Num = _1_Num+1
        if train_Data[i][DATA_DIMENSION] == 'M\n':
            _2_Num = _2_Num+1
        if train_Data[i][DATA_DIMENSION] == 'Iris-virginica\n':
            _3_Num = _3_Num+1
    _1_Data = mat(np.zeros((_1_Num,DATA_DIMENSION)))
    _2_Data = mat(np.zeros((_2_Num,DATA_DIMENSION)))
    _3_Data = mat(np.zeros((_3_Num,DATA_DIMENSION)))
    _1_Num = _2_Num = _3_Num = 0
    for i in range(len(train_Data)):
        if train_Data[i][DATA_DIMENSION] == 'R\n':
            _1_Data[_1_Num] = train_Data[i][0:DATA_DIMENSION]
            _1_Num = _1_Num+1
        if train_Data[i][DATA_DIMENSION] == 'M\n':
            _2_Data[_2_Num] = train_Data[i][0:DATA_DIMENSION]
            _2_Num = _2_Num+1
        if train_Data[i][DATA_DIMENSION] == 'Iris-virginica\n':
            _3_Data[_3_Num] = train_Data[i][0:DATA_DIMENSION]
            _3_Num = _3_Num+1
    f.close()
    return train_Data,test_Data,_1_Data.T,_2_Data.T,_3_Data.T

#计算欧氏距离
def calcu_Eucli_Dis(x,y):
    dis = np.sqrt(np.sum(np.square(np.array(x)-np.array(y))))
    return dis

#计算样本到均值向量的距离平方和
def calcu_error(samples, mean):
```

```
    sum=0
    for i in range(len(samples)):
        e = np.sum(np.square(np.array(samples[i])-np.array(mean)))
        sum += e

    return sum


#*************************************************************K-
means*************************************************************
**
#K-means 类结构
class Data_K_means:
    def __init__(self):
        self.name = ''

        self.clusteringCenter = []  #聚类中心
        self.clusteringData = []    #包含的样本数据
        self.clusteringMeans = 0    #均值
        self.clusteringError = 0    #每一类中的最小误差平方和

        self.R = 0
        self.M = 0

    def print(self):
        print("*************************   ",self.name,"   **********
***********")

        print("clusteringCenter: ",self.clusteringCenter)
        print("clusteringError: "  ,self.clusteringError)
        # print("clusteringData: \n"  ,np.array(self.clusteringData))
        print("dataSize: "  ,len(self.clusteringData))

#按最小距离准则划分样本到 N 类
def divide_Sample(train_Data,category,dataDimension,K):
    for i in range(K):
        category[i].clusteringData = []
    for i in range(len(train_Data)):
        min_dis = 1024
        min_dis_label = -1
        for j in range(K):
            dis = calcu_Eucli_Dis(train_Data[i][:dataDimension],categor
y[j].clusteringCenter)
            if dis<min_dis:
```

```
                min_dis = dis
                min_dis_label = j
        # print("min_dis:",min_dis,"label:",min_dis_label)
        category[min_dis_label].clusteringData.append(train_Data[i])

#更新参数：计算均值与准则函数，更新聚类中心,返回准则函数
def update_Cluster_Center(category,dataDimension,K):
    #计算每一类的均值、误差 J_i,计算出准则函数
    J = 0
    for i in range(K):
        dataArray = np.array(np.array(category[i].clusteringData)[:,0:d
ataDimension],dtype=np.float64)
        #计算均值
        category[i].clusteringMeans = mean(dataArray,0)
        #更新聚类中心
        category[i].clusteringCenter = category[i].clusteringMeans
        #计算每一类内距离差
        error = calcu_error(dataArray,category[i].clusteringMeans)
        category[i].clusteringError = error
        J += error
    return J

def k_Means(dataPath,dataSize,dataDimension,K,testRate=0):
    #提取数据
    train_Data,test_Data,group1,group2,group3 = readData(dataPath,dataS
ize,dataDimension,testRate)
    print("train_Data SIZE:",len(train_Data))

    #随机选取 K 个聚类中心,并初始化聚类名称与聚类均值
    clustering_center_label = random.sample(range(0,len(train_Data)-
1),K)
    print("clustering_center_label:",clustering_center_label)
    category = [Data_K_means() for i in range(K)]
    for i in range(K):
        category[i].clusteringCenter = train_Data[clustering_center_lab
el[i]][:dataDimension]
        category[i].clusteringMeans = category[i].clusteringCenter
        category[i].name = "category_"+str(i)
        # category[i].print()

    J = 1024.0
    J_last = -1024.0
    repeat_time = 0
    #开始迭代
```

```
    while J!=J_last:    #以准则函数作为迭代结束依据
        J_last = J
        repeat_time += 1

        #按最小距离准则划分样本
        divide_Sample(train_Data,category,dataDimension,K)

        #计算均值，更新聚类中心，计算准则函数值
        J = update_Cluster_Center(category,dataDimension,K)

        print("------------------------------------------------------
-------------------->Num: ",repeat_time)
        # for i in range(K):
        #     category[i].print()
        print("J = ",J)
        print("J_last = ",J_last,"\n\n")

    #迭代结束，统计分类结果标签数目 并 计算准确率
    accuracy = 0
    for k in range(K):
        _1_Num = _2_Num = 0.0
        for i in range(len(category[k].clusteringData)):
            if category[k].clusteringData[i][dataDimension] == 'R\n':
                _1_Num += 1
            if category[k].clusteringData[i][dataDimension] == 'M\n':
                _2_Num += 1
        category[k].print()
        print("R:",_1_Num)
        print("M:",_2_Num,"\n\n")

        category[k].R = _1_Num
        category[k].M = _2_Num

        if _1_Num>=_2_Num:
            accuracy += _1_Num/len(category[k].clusteringData) * len(ca
tegory[k].clusteringData)/len(train_Data)
        else:
            accuracy += _2_Num/len(category[k].clusteringData) * len(ca
tegory[k].clusteringData)/len(train_Data)


    return category,J,accuracy


def test_K_Means():
```

```
    J = [0]
    accuracy = [0]
    testTimes = 10

    #绘制聚类结果
    plt.figure()
    for i in range(testTimes):
        category,J_temp,accuracy_temp = k_Means("sonar.all-
data",208,60,K=2)    #进行 K 均值聚类,返回聚类结果、误差平方和、精准度
        J.append(J_temp)
        accuracy.append(accuracy_temp)

        for k in range(2):
            p1 = plt.bar(x=k+(i*5), height=category[k].R, label='R', co
lor="red")
            p2 = plt.bar(x=k+(i*5), height=category[k].M, label='M', co
lor="green",bottom=category[k].R)
        plt.legend((p1[0], p2[0]), ('R', 'M'))
    plt.ylabel('Number')
    plt.xlabel("testTimes")
    plt.xticks([])
    plt.title('K-means result')
    plt.show(block = False)

    #绘制聚类精确度
    plt.figure()
    for i in range(1,testTimes+1):
        plt.scatter(i,accuracy[i])
        print("Accuracy_",i,": ",accuracy[i])
    print("Accuracy_AVE",mean(accuracy[1:testTimes]))
    plt.title("K-means Accuracy")
    plt.xlabel("testTimes")
    plt.ylabel("Accuracy")
    plt.plot(accuracy)
    plt.show(block = False)

    #绘制聚类最终准则函数值（#最小误差平方和）
    plt.figure()
    for i in range(1,testTimes+1):
        plt.scatter(i,J[i])
        print("J_",i,": ",J[i])
    plt.title("K-means J_e")
    plt.xlabel("testTimes")
    plt.ylabel("J_e")
```

```
    plt.plot(J)
    plt.show()

#*************************************************************FCM*******
************************************************************
class Data_FCM:
    def __init__(self):
        self.name = ''

        self.Data = []          #最终包含的样本数据
        self.Membership = []    #隶属度
        self.Means = 0          #均值/聚类中心

        self.R = 0
        self.M = 0
        self.Iris_virginica = 0

    def print(self):
        print("*************************   ",self.name,"   **********
************")

        print("clusteringMeans: ",self.Means)
        # print("clusteringData: \n"  ,np.array(self.clusteringData))
        print("dataSize: "  ,len(self.Data))

#计算 更新隶属度函数
def update_Membership(train_Data,category,dataDimension,C,b):
    trainDataArray = np.array(np.array(train_Data)[:,0:dataDimension],d
type=np.float64)
    for j in range(C):  #u_j(x_i)
        category[j].Membership = []     #清空隶属度向量
        for i in range(len(train_Data)):
            #计算分母、分子
            denom = 0.0
            for l in range(C):
                denom += pow((1/np.square(np.linalg.norm(np.array(train
DataArray[i])-np.array(category[l].Means)))),1/(b-1))
            numer = pow((1/np.square(np.linalg.norm(np.array(trainDataA
rray[i])-np.array(category[j].Means)))),1/(b-1))
            #更新隶属度
            if numer == inf:    #解决自己与自己隶属度计算出错问题
                category[j].Membership.append(1.0)
            else:
                category[j].Membership.append(numer/denom)
```

```
        # print(j,np.array(category[j].Membership))

#计算 更新聚类中心
def update_Cluster_Center_FCM(train_Data,category,dataDimension,C,b):
    trainDataArray = np.array(np.array(train_Data)[:,0:dataDimension],d
type=np.float64)
    for j in range(C):
        #计算分母、分子
        denom = 0.0
        numer = 0.0
        for i in range(len(train_Data)):
            denom += pow(category[j].Membership[i],b)
            numer += pow(category[j].Membership[i],b) * trainDataArray[
i]
        # print(j,":",numer,denom,numer/denom)
        category[j].Means = numer/denom

#根据隶属度函数划分样本
def divide_Sample_FCM(train_Data,category,C):
    for i in range(len(train_Data)):
        membership_max = -1
        label = -1
        for j in range(C):
            if category[j].Membership[i] > membership_max:
                membership_max = category[j].Membership[i]
                label = j
        category[label].Data.append(train_Data[i])

def FCM(dataPath,dataSize,dataDimension,C,b=2,end_condi=0.001,testRate=
0):
    #提取数据
    train_Data,test_Data,group1,group2,group3 = readData(dataPath,dataS
ize,dataDimension,testRate)
    print("train_Data SIZE:",len(train_Data))

    #随机选取 C 个聚类中心,初始化聚类中心(均值向量)
    clustering_center_label = random.sample(range(0,len(train_Data)-
1),C)
    print("clustering_center_label:",clustering_center_label)
    category = [Data_FCM() for i in range(C)]
    for i in range(C):
        category[i].Means = train_Data[clustering_center_label[i]][:dat
aDimension]
        category[i].name = "category_"+str(i)
```

```
        category[i].print()

    m = [1024 for i in range(dataDimension)]
    m_last = [-1024 for i in range(dataDimension)]
    repeat_time = 0
    #开始迭代
    while calcu_Eucli_Dis(m,m_last) > end_condi:
        repeat_time += 1
        m_last = m

        #计算，更新隶属度
        update_Membership(train_Data,category,dataDimension,C,b)

        #计算均值，更新聚类中心
        update_Cluster_Center_FCM(train_Data,category,dataDimension,C,b
)

        m = category[0].Means

        print("------------------------------------------------------------
--------------------->Num: ",repeat_time)
        print("m=",m)
        print("m_last = ",m_last)
    #迭代结束，划分样本
    divide_Sample_FCM(train_Data,category,C)

    #划分结束，统计分类结果标签数目 并 计算准确率
    accuracy = 0
    for k in range(C):
        _1_Num = _2_Num = 0
        for i in range(len(category[k].Data)):
            if category[k].Data[i][dataDimension] == 'R\n':
                _1_Num += 1
            if category[k].Data[i][dataDimension] == 'M\n':
                _2_Num += 1
        category[k].print()
        print("R:",_1_Num)
        print("M:",_2_Num,"\n\n")
        category[k].R = _1_Num
        category[k].M = _2_Num

        if _1_Num>=_2_Num:
            accuracy += _1_Num/len(category[k].Data) * len(category[k].
Data)/len(train_Data)
```

```
        else:
            accuracy += _2_Num/len(category[k].Data) * len(category[k].
Data)/len(train_Data)

    return category,accuracy

def test_FCM():
    accuracy = [0]
    testTimes = 10

    #绘制聚类结果
    plt.figure()
    for i in range(testTimes):
        category,accuracy_temp = FCM("sonar.all-
data",208,60,C=2,b=2,end_condi=0.001)    #进行 K 均值聚类,返回聚类结果、误
差平方和、精准度
        accuracy.append(accuracy_temp)

        for k in range(2):
            p1 = plt.bar(x=k+(i*5), height=category[k].R, label='R', co
lor="red")
            p2 = plt.bar(x=k+(i*5), height=category[k].M, label='M', co
lor="green",bottom=category[k].R)
        plt.legend((p1[0], p2[0]), ('R', 'M'))
    plt.ylabel('Number')
    plt.xlabel("testTimes")
    plt.xticks([])
    plt.title('FCM result')
    plt.show(block = False)

    #绘制聚类精确度
    plt.figure()
    for i in range(1,testTimes+1):
        plt.scatter(i,accuracy[i])
        print("Accuracy_",i,": ",accuracy[i])
    print("Accuracy_AVE",mean(accuracy[1:testTimes]))
    plt.title("FCM Accuracy")
    plt.xlabel("testTimes")
    plt.ylabel("Accuracy")
    plt.plot(accuracy)
    plt.show()


if __name__ == '__main__':
```

```
# test_K_Means()
test_FCM()
```