# 一种新的基于粗糙集的 leader 聚类算法*)

张　琼　张　莹　白清源　谢丽聪　谢伙生
(福州大学数学与计算机科学学院　福州 350002)

**摘　要**　传统聚类方法将对象严格地划分到某一类,但很多时候边界对象不能被严格地划分。粗糙集用上近似集和下近似集表示一个类,对这种边界不确定的处理非常有效,典型算法有基于粗糙集的 k-means 聚类算法和基于粗糙集的 leader 聚类算法。本文针对 RFA(Rough Fuzzy Approach)算法存在的不足,提出了一种新的基于粗糙集的 leader 聚类算法(NRL,Novel Rough-based Leader)。其基本思想是首先数据项由于与其最近类中心的距离不同,分别被划分到 leader 集或者 supporting leader 集,然后对 leader 集和 supporting leader 集进行标号,得到聚类结果。实验结果表明 NRL 算法非常有效。

**关键词**　聚类,粗糙集,k-means 算法,leader 算法

## A Novel Rough-based Leader Clustering Algorithm

ZHANG Qiong　ZHANG Ying　BAI Qing-Yuan　XIE Li-Cong　XIE Huo-Sheng
(College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350002)

**Abstract**　Objects are partitioned into clusters with crisp boundaries in the conventional algorithms. However, clusters do not necessarily have crisp boundaries. Rough set is represented with lower-bound and upper-bound, and is good for the case. At present, there have been some typical algorithms, such as the rough-based k-means clustering algorithm and the rough-based leader clustering algorithm. In this paper, a novel rough-based leader clustering algorithm is proposed, since there are some disadvantages in the RFA algorithm. At first, data are partitioned into the set of leaders or the set of supporting leaders according to the difference of the distance of data and the nearest leader. And then, it labels the set of leaders and the set of supporting leaders in order to find the clustering result. We present results to demonstrate its validity.

**Keywords**　Clustering, Rough set, K-means algorithm, Leader algorithm

## 1　引言

聚类是一个将数据集划分成若干类的过程,使得同一类的数据对象相似性较大,不同类的数据对象相似性较小。许多领域,包括数据挖掘、知识发现、数据压缩、模式识别、统计学等等,都有聚类的相关研究及应用。

聚类方法通常可以分为以下几大类:划分方法、层次方法、基于密度方法、基于网格方法、基于模型方法[1]。传统聚类方法将数据集划分成 k 类,每类至少包含一个对象,每个对象只能属于其中某一类。也就是说,类与类之间有明显的分界线。但是,Web 数据与传统的结构化数据不一样:Web 数据往往是无结构、半结构的,是不完整、不精确的,还包含很多噪声数据。因此,有些 Web 数据不能被严格地划分到某一类,换句话说,类与类之间不能被严格地划分开来[2]。粗糙集用上近似集和下近似集表示一个类,处理这种不完整、不确定的数据非常有效。

## 2　相关工作

目前研究比较多的有基于粗糙集的 k-means 聚类算法和基于粗糙集的 leader 聚类算法。文[3]提出了基于粗糙集 k-means 算法。这个算法比传统 k-means 算法改进的地方在于:(1)用上近似集和下近似集表示一个类;一个数据项要么被划分到最近的类中心的下近似集,要么被划分到最近的类中心的上近似集。(2)计算类中心时,下近似集的数据和上近似集的数据权值不同。基于粗糙集 k-means 算法的不足之处与传统 k-means 算法一样,需要重复扫描数据集,多次迭代;如果存在噪声数据,对聚类结果影响很大。而 leader 算法最大的优点就是只要一遍扫描数据集,噪声数据对聚类结果没有影响。文[4]提出了 ARFL(Adaptive Rough Fuzzy Leader)算法,是一种基于粗糙集的 leader 算法,也只需要一遍扫描数据集就可以得到全部的 leader(即类中心)集。但是 ARFL 算法选择不同的数据项作为起始的 leader,得到的 leader 集不一样;而且设置不同的阈值,得到的 leader 集差别也很大;最主要的不足是得到的 leader(即类中心)个数太多,类分得太细。文[5]提出的 RFA 算法(Rough Fuzzy Approach),对 ARFL 算法进行了改进。但是文[5]的 RFA 算法也有一些局限性。我们针对这些局限性,对 RFA 算法进行了改进,提出了一种新的基于粗糙集的 leader 聚类算法(NRL, Novel Rough-based Leader)。

文章第 3 部分简单介绍粗糙集的相关理论以及 RFA 算法,第 4 部分详细介绍本文提出的 NRL 算法,第 5 部分给出实验结果,最后是结论。

*)福州大学科技发展基金资助项目(2005-XQ-13、2006-XQ-22,XRC-0511),福建省教育厅资助项目(JB06023)。张　琼　硕士研究生,主要研究方向:数据挖掘;张　莹　硕士生导师,主要研究方向:数据库技术、数据挖掘。

## 3 粗糙集的相关理论以及 RFA 算法

### 3.1 粗糙集的相关理论

上世纪 80 年代初，波兰的 Pawlak 提出了粗糙集，用下近似集和上近似集表示一个类[6,7]。他把那些无法确认的对象都归属于边界区域，而这种边界区域被定义为上近似集和下近似集之差集。下近似集的对象能够被准确地划分到某个类，而上近似集的对象是不能够被准确地划分到某个类，只能说可能归属于某个类。粗糙集的性质很多，我们这里用到的有：

(1) 一个对象至多只能归属于一个类的下近似集。

(2) 如果一个对象归属于一个类的下近似集，那么它也归属于这个类的上近似集。

(3) 如果一个对象不属于任何一个类的下近似集，那么它肯定属于两个或者两个以上类的上近似集。

### 3.2 RFA 算法

RFA 算法定义了两个数据结构：{leader}、{supporting leader}。{leader}是所有 leader 的集合，即各个类中心的集合。而{supporting leader}是所有 supporting leader 的集合，即那些不能被准确地划分到某个 leader 的数据集合。同时，设置了 LT、UT、OT 等三个需要用户输入的阈值参数。其基本思想：首先数据项由于与其最近类中心的距离不同，分别被划分到 leader 集或者 supporting leader 集；然后 supporting leader 集中的 supporting leader 要么被划分到某个 leader 的下近似集，要么作为一个新的 leader；最后对 leader 集和 supporting leader 集进行标号，得到聚类结果。

## 4 NRL 算法

### 4.1 NRL 算法的总体思想

RFA 算法的不足之处主要在于：(1)数据集中的数据项只跟{leader}中当前已经存在的 leader 比较，没有考虑到后面生成的 leader 可能对这个数据项的归类产生影响。(2)把所有不能被准确地划分的 supporting leader 都作为一个新的 leader，添加到{leader}。针对这些局限性，我们提出了 NRL 算法。

NRL 算法由两个子算法组成，分别是 ERClustering 子算法和 ELabel 子算法。首先对数据集用 ERClustering 子算法得到 leader 集和 supporting leader 集，然后用 ELabel 子算法对 leader 集和 supporting leader 集进行标号，最后得到聚类结果。

### 4.2 ERClustering 子算法

ERClustering 子算法根据输入的数据集以及设置的 LT、UT、OT，把能够被准确地划分的数据项划分到相应 leader 的下近似集，不能够被准确地划分的数据项作为 supporting leader，添加到{supporting leader}。算法中 LT、UT、OT、{leader}、{supporting leader}的定义与 RFA 算法中的定义是一样的。算法具体描述如下：

```
输入：数据集、LT、UT、OT
输出：{leader}、{supporting leader}
方法：
从数据集中取出任意一个数据项作为起始的 leader，添加到{leader}；
for 数据集中的其他数据项 do
{
  从当前{leader}中，找到与这个数据项最近的 leader，计算这个数据项与最近 leader 的距离，记为 distance()；
  if(distance()<LT)
    这个数据项被划分到其最近 leader 的下近似集；
  else if(distance()>UT)
  {
    这个数据项作为新的 leader，添加到{leader}；
    new()；//改进的地方
  }
  else
  {
    在当前{leader}中，计算满足 LT<distance()<UT 的 leader 个数，记为 overlap；
    If(overlap<OT)
      这个数据项作为新的 supporting leader，添加到{supporting leader}
    else
    {
      这个数据项作为新的 leader，添加到{leader}，同时修改满足 LT<distance()<UT 的 leader 的 UT；
      new()；//改进的地方
    }
  }
}
```

ERClustering 子算法改进的地方有：

(1) RFA 算法是生成所有的{leader}以后，再对{supporting leader}进行操作。而 ERClustering 子算法是每生成一个新的 leader 时，都执行 new()操作。因为{supporting leader}中的部分 supporting leader 有可能在生成某个新的 leader 时，满足 supporting leader 与这个新的 leader 的距离小于 LT，这时 supporting leader 就可以直接被划分到这个新的 leader 的下近似集。

(2) RFA 算法是把{supporting leader}中不能被准确地划分的 supporting leader 都作为新的 leader，添加到{leader}。而 ERClustering 子算法是把{supporting leader}中不能被准确地划分的 supporting leader 继续保留在{supporting leader}。因此，ERClustering 子算法输出的是{leader}和{supporting leader}。其中，new()函数具体如下：

```
Void new( )
{
  For {supporting leader}中的每个 supporting leader do
  {
    If (supporting leader 与这个新的 leader 的距离<LT)
      supporting leader 被划分到这个新的 leader 的下近似集；
  }
}
```

### 4.3 ELabel 子算法

ELabel 子算法的思想是：{leader}中，只要某个 leader 的上近似集与另一个 leader 的上近似集相同数据项的个数达到 wrap 以上，则这两个 leader 的标签设置一样。也就是说，{supporting leader}中有 wrap 以上的 supporting leader，同时都属于某些 leader 的上近似集，这些 leader 都设置相同的标签。具体算法描述如下：

```
输入：{leader}、{supporting leader}、wrap
输出：各个类
方法：
(1) 初始化{leader}中所有的 leader，标签设置为空。
(2) 从{leader}中取出任意一个标签为空的 leader，记为 leader[i]，给它设置一个新的标签；
(3) 找到所有这样的 leader[j]，leader[j]的上近似集与 leader[i]的上近似集相同数据项的个数达到 wrap 以上，给这样的 leader[j]设置与 leader[i]相同的标签；
(4) 如果{leader}还有标签为空的 leader，重复(2)(3)。否则，算法结束。
```

### 4.4 举例说明

下面我们举个简单的例子。我们设数据集={1,1.4,1.8,1.7,2.4,4}。先运行 NRL 算法中的 ERClustering 子算法，设置阈值 LT=0.5、UT=1、OT=3，得到{leader}={1,2.4,4}，{supporting leader}={1.8,1.7}，数据项{1.4}被划分到 leader{1}的下近似集。然后运行 ELabel 子算法，设置阈值 wrap=2。因为{1.8,1.7}同时属于 leader{1}和 leader{2.4}的上近似集，所以 leader{1}和 leader{2.4}设置相同的标签，记为标签 1。leader{4}设置另一个新的标签，记为标签 2。

因此，上面这个例子用 NRL 算法得到的最后聚类结果是{1，1.4，1.8，1.7，2.4}、{4}。

## 5 实验

下面通过实验分析 NRL 算法的性能，文中所有算法由 C＋＋编程实现。

算法使用的数据是 UCI 数据集中的 Iris，是模式识别中一个经典的数据集，记录的是一些植物的特征数据。共有 5 个属性，其中包括 4 个数值型条件属性、1 个分类型决策属性，共有 150 条数据和 3 种决策值，这些数据简单记为 0，1，2，…，149。其中 0～49、50～99、100～149 分别对应一种决策。去掉 Iris 的决策属性，对数据集分别用 RFA 算法、NRL 算法和 K-means 算法进行聚类。

由于 RFA 算法、NRL 算法和 K-means 算法都涉及到参数，因此参数设置不同，最后得到的聚类结果也不一样。为了比较三种聚类算法的聚类效果，我们设置了合适的参数，使得三种算法得到的聚类结果都是 3 类，而且聚类效果比较好。然后聚类结果分别与数据集的决策属性比较，分析聚类结果的正确率。

下面分别给出三种算法得到的聚类结果。表 1 给出 RFA 算法在如下阈值条件下得到的聚类结果。先设置 LT＝0.5，UT＝1，OT＝3，起始 leader 选择第一个数据项。再设置 LT＝1.5，UT＝2，OT＝3，起始 leader 选择第一个数据项。

表 1 RFA 算法的聚类结果

| 第 1 类 | 第 2 类 | 第 3 类 |
|---|---|---|
| 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49 (50 个数据项) | 50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,91,92,94,95,96,97,98,99,101,102,103,104,106,110,111,112,113,115,116,119,120,121,123,124,125,126,127,128,130,131,132,133,134,137,138,139,140,141,142,143,144,145,146,147,148,149 (86 个数据项) | 72,93,100,105,107,108,109,114,117,118,122,129,135,136,(14 个数据项) |

表 2 给出 NRL 算法在如下阈值条件下得到的聚类结果：LT＝0.5，UT＝1，OT＝2，wrap＝2。

表 2 NRL 算法的聚类结果

| 第 1 类 | 第 2 类 | 第 3 类 |
|---|---|---|
| 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49 (50 个数据项) | 50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,73,74,75,76,78,79,80,81,82,84,85,86,87,88,89,90,91,92,93,94,95,96,97,98,99,127,138,149 (50 个数据项) | 72,83,77,100,101,102,103,104,105,106,107,108,109,110,111,112,113,114,115,116,117,118,119,120,121,122,123,124,125,126,128,129,130,131,132,133,134,135,136,137,139,140,141,142,143,144,145,146,147,148 (50 个数据项) |

表 3 给出 K-means 算法中初始数据项选择 0、1、2 的情况下得到的聚类结果。

图 1 给出 RFA 算法、NRL 算法和 K-means 算法得到的聚类结果分别与数据集的决策属性比较后得到的正确率。

表 3 k-means 算法的聚类结果

| 第 1 类 | 第 2 类 | 第 3 类 |
|---|---|---|
| 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49 (50 个数据项) | 51,53,54,55,57,58,59,60,61,62,63,64,65,66,67,68,69,71,73,74,75,78,79,80,81,82,84,85,87,88,89,90,91,92,93,94,95,96,97,98,99,106,121,(43 个数据项) | 50,52,56,70,72,76,77,83,86,100,101,102,103,104,105,107,108,109,110,111,112,113,114,115,116,117,118,119,120,122,123,124,125,126,127,128,129,130,131,132,133,134,135,136,137,138,139,140,141,142,143,144,145,146,147,148,149,(57 个数据项) |

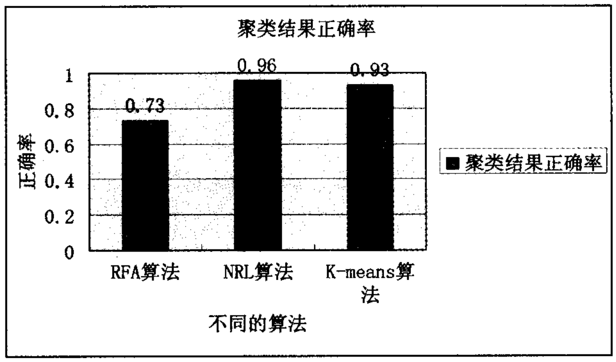


图 1 三种聚类算法正确率比较

从实验可以看出，NRL 算法的聚类效果比 RFA 算法的聚类效果要好，比 K-means 算法的聚类效果也会好一点。我们还用不同的数据集测试三种算法的性能，发现 NRL 算法比 RFA 算法不管聚类效果还是运行的时间都会好。K-means 算法需要重复迭代，花的时间比较多，而且容易陷入局部最优。

总的来讲，三种算法都涉及到阈值的设置，设置不同的阈值，得到的聚类结果差别很大。为了得到比较好的聚类结果，需要反复测试，选择合适的阈值。

**结论** k-means 算法和 leader 算法都是经典的聚类方法，但是这些传统的聚类方法往往是把对象严格地划分到不同的类，不能够处理不完整、不精确的数据。基于粗糙集的聚类方法正好可以解决这个问题。NRL 算法有如下几个优点：

(1) 只需要一遍扫描数据集就可以得到所有的 leader 集。

(2) 有很好的扩展性。增加一个新的数据项，对前面的数据项聚类没有影响，整个数据集不需要重新再聚类。

(3) 噪声数据对聚类结果没有影响。

正是由于这些优点，基于粗糙集的 leader 聚类算法得到广泛地应用。Web 数据是无结构的、半结构的，而且存在很多不完整数据、噪声数据，因此传统的聚类方法都不适合这些数据的聚类。接下来我们要做的工作是基于粗糙集改进传统的聚类方法，用于挖掘 Web 数据。

## 参 考 文 献


1 朱明著. 数据挖掘. 中国科学技术大学出版社，2002
2 Lingras P. Rough set clustering for Web mining. IEEE, 2002. 1039～1044
3 Lingras P. Interval Set Clustering of Web Users with Rough K-Means. Journal of Intelligent Information System, 2004, 23(1): 5～16
4 Asharaf S. An adaptive rough fuzzy single pass algorithm for clustering large data sets. Pattern Recognition, 2003, 36: 3015～3018
5 Asharaf S. A Rough Fuzzy Approach to Web Usage Categorization. Fuzzy Sets and Systems, 2004, 148(1): 119～129
6 Pawlak Z. Rough Sets. International Journal of Computer and Information Sciences, 1982 (11): 341～356
7 Pawlak Z. rough classification. international journal of man-machine studies, 1984, 20: 469～483