

改进的快速 DBSCAN 算法

王桂芝,王广亮

(河南商业高等专科学校 计算机应用系,郑州 450044)

(wgz123@yahoo.cn)

摘要:针对 DBSCAN 算法时间性能低效的问题,分析快速聚类过程中丢失对象的原因,提出一种新的改进算法 IF-DBSCAN。该算法在不丢失对象的基础上,通过选取核心对象邻域中的代表对象来扩展类,从而减少邻域查询次数,提高了算法的时间性能。实验结果表明,IF-DBSCAN 算法是正确和高效的。

关键词:聚类;DBSCAN 算法;邻域;核心对象

中图分类号: TP301 **文献标志码:** A

Improved fast DBSCAN algorithm

WANG Gui-zhi, WANG Guang-liang

(Department of Computer Application, Henan Business College, Zhengzhou Henan 450044, China)

Abstract: The time performance of Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is inefficient. Concerning this problem, the authors analyzed the reasons of losing object in the process of fast clustering, and proposed a new Improved Fast DBSCAN (IF-DBSCAN) algorithm. On the basis of not losing data object, this algorithm expanded a category by selecting representative objects from the neighborhood of core data object, so that it reduced the number of regional inquiries and improved the algorithm's time performance. The experimental results show that IF-DBSCAN algorithm is correct and efficient.

Key words: clustering; DBSCAN algorithm; neighborhood; core object

0 引言

聚类(Clustering)是数据挖掘中的重要组成部分。所谓聚类,就是将数据对象分组成多个类或簇(Cluster),在同一个簇中的对象之间具有较高的相似度,而不同簇中的对象差别较大^[1]。通过聚类,人们能够识别密集的和稀疏的区域,发现全局的分布模式和数据属性之间有趣的相互关系。在数据挖掘中,聚类分析能作为一个独立的工具来获得数据分布的情况,观察每个簇的特点,集中对特定的某些簇做进一步分析。此外,聚类分析还可以作为其他算法(如特征和分类等)的预处理步骤,这些算法再在生成的簇上进行处理。

迄今为止,人们已经提出了许多聚类算法,如 K-MEANS^[2]、CLARANS^[3]、DBSCAN^[4]、CURE^[5]、CLIQUE^[6]和 C2P^[7]等算法,其中 DBSCAN 算法是一种基于密度的聚类算法。该算法将具有一定密度的区域划分为簇,可以在含有“噪声”的空间数据集中发现任意形状的聚类。但其时间性能是低效的。本文提出了一个新的 DBSCAN 改进算法 IF-DBSCAN。该算法在时间和空间性能上都比传统的 DBSCAN 算法有较大提高。

1 基于密度的聚类算法 DBSCAN

1.1 基本概念

基于密度的聚类算法的核心思想是:对于构成簇的每个对象,其 ε 邻域包含的对象个数,必须不小于一个给定值($MinPts$),也就是说其邻域的密度必须不小于某个阈值。下面给出基于密度聚类算法分析中的一些定义^[4]。

定义 1 ε -邻域。给定对象半径 ε 内的区域称为该对象的

ε -邻域。

定义 2 核心对象。给定 $\varepsilon, MinPts$,若对象 p 的 ε 邻域 $N_\varepsilon(p)$ 包含的对象个数 $|N_\varepsilon(p)| \geq MinPts$,则称 p 为核心对象。

定义 3 直接密度可达。给定 $\varepsilon, MinPts$,当:

- 1) $p \in N_\varepsilon(q)$ 且
- 2) $|N_\varepsilon(q)| \geq MinPts$

则称对象 p 是从对象 q 出发直接密度可达的。

定义 4 密度可达。给定对象集合 D ,当存在一个对象链 $p_1, p_2, \dots, p_n, p_1 = q, p_n = p$,对 $p_i \in D, p_{i+1}$ 是从 p_i 关于 ε 和 $MinPts$ 直接密度可达的,则称对象 p 从对象 q 关于 ε 和 $MinPts$ 密度可达(非对称)。

定义 5 密度相连。如果对象集合 D 中存在一个对象 o ,使得对象 p 和 q 是从 o 关于 ε 和 $MinPts$ 密度可达的,那么对象 p 和 q 关于 ε 和 $MinPts$ 密度相连(对称)。

定义 6 簇和噪声。基于密度可达性的最大的密度相连对象的集合称为簇,不在任何簇中的对象被认为是“噪声”。

1.2 DBSCAN 算法

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) 是一个基于密度的聚类算法。该算法采用迭代查找的方法,通过迭代地查找所有直接密度可达的对象,找到各个簇所包含的所有密度可达的对象^[4]。具体方法如下:

- 1) 检查数据库中尚未检查过的对象 p ,如果 p 未被处理(归入某个簇或标记为噪声),则检查其 ε 邻域 $N_\varepsilon(p)$,若 $N_\varepsilon(p)$ 包含的对象数不小于 $MinPts$,建立新簇 C ,将 $N_\varepsilon(p)$ 中所有点加入 C ;

- 2) 对 C 中所有尚未被处理的对象 q ,检查其 ε 邻域 $N_\varepsilon(q)$,若 $N_\varepsilon(q)$ 包含至少 $MinPts$ 个对象,则将 $N_\varepsilon(q)$ 中未

收稿日期:2009-03-23;修回日期:2009-05-10。

作者简介:王桂芝(1970-),女,河南郑州人,副教授,硕士,主要研究方向:数据挖掘、聚类分析;王广亮(1970-),男,河南郑州人,副教授,主要研究方向:聚类分析。

归入任何一个簇的对象加入 C ;

3) 重复步骤 2), 继续检查 C 中未处理对象, 直到没有新的对象加入当前簇 C ;

4) 重复步骤 1) ~ 3), 直到所有对象都归入了某个簇或标记为噪声。

DBSCAN 算法可以在有噪声的数据中发现任意形状的聚类, 但该算法也具有明显的局限性。DBSCAN 的时间复杂度为 $O(n^2)$ 。另外, DBSCAN 算法要对每个数据对象进行邻域查询, 其时间性能低效。

2 IF-DBSCAN 算法

基于 DBSCAN 算法时间性能问题, 周水庚等人^[8]提出了一种快速的聚类算法 FDBSCAN, 本文的 IF-DBSCAN 算法正是在此基础上提出的。

2.1 IF-DBSCAN 算法基础

FDBSCAN 算法通过选用核心对象附近区域包含的所有对象的代表对象作为种子对象来扩展类, 减少了区域查询的次数, 从而减低了聚类时间和 I/O 开销。该算法提出, 在 n 维空间中, 选择 $2n$ 个代表对象。也就是说, 在每一维空间上, 选择两个对象作为代表对象用于类的扩展^[8]。但是, 如果某些对象唯一地通过被忽略的核心对象 p 密度可达, 则当 p 所在的类 C 扩展完成后, 这些对象将未被包含在类 C 中, 此时称之为丢失对象。FDBSCAN 算法在进行选取代表对象进行快速聚类后, 还必须对丢失对象进行处理。

我们用 C 语言编程实现了 DBSCAN 算法和未处理丢失对象的 FDBSCAN 算法, 在 VC++ 6.0 环境下调试运行, 采用二维数据集进行验证。实现结果表明, 在未对丢失对象进行处理时, FDBSCAN 算法比 DBSCAN 算法的速度要快数倍, 甚至十倍以上。但此时, FDBSCAN 算法不能进行有效聚类——本来存在几个类的数据集却被分成几十个类。显然, FDBSCAN 算法必须对丢失对象进行处理, 特别是对丢失对象所引起的许多小类的合并问题必须解决。但是, 要进行类的合并必然要对已聚类后的数据库再次进行扫描, 并对某些数据对象进行区域查询。这必将降低算法的时间性能。为此提出一种新的快速聚类算法——IF-DBSCAN, 可以避免后期对丢失对象的处理过程。

2.2 IF-DBSCAN 算法思想

下面通过深入分析快速聚类中丢失对象的原因而提出 IF-DBSCAN 算法的基本思想。

2.2.1 丢失对象的原因分析

在二维空间中, 对一个核心对象的邻域选择 4 个代表对象作为种子对象, 并通过核心的种子对象再次选择其邻域的 4 个代表对象来扩展类, 用这种方法聚类会有两个问题。

1) 当用 4 个代表对象作为种子对象来进行扩展类时, 如果某一个种子对象不是核心对象将不再进行类的扩展, 同时也放弃它邻域中的所有对象。这样将会导致大量的对象丢失, 从而导致一些本应属于此类的数据对象被排除在外。这是因为, 代表对象的邻域并不仅仅是这一数据对象的邻域, 它同时代表了这一种子对象附近的许多对象的邻域, 甚至于一个类的扩展方向。如果把这一邻域放弃了, 等于放弃了这一方向 (即四分之一) 的类的扩展。

2) 一个核心对象的邻域可以近似地被 4 个分散较好的代表对象的具有相同半径的邻域所覆盖, 但是, 当用这 4 个种子对象进行类扩展时, 如果再次选择它们邻域的 4 个代表对象继续扩展时, 同样会丢失对象。这是因为, 当第一轮选择的 4

个种子对象再次选择它们各自邻域的 4 个代表对象时, 必定都有一个代表对象已归过类 (在初始的核心点邻域内), 如果一个种子对象邻域的另外三个代表对象中恰巧有两个代表对象也在初始核心点的邻域内, 那么这一种子对象则仅依靠离初始核心对象最远的那个代表对象来扩展类。如图 1 所示, p_1, p_2, p_3, p_4 是第一轮选择的 4 个代表对象作为种子对象, 其中 p_1 为核心对象, 其邻域的代表对象 q_1, q_2, q_3, q_4 , 由图可以看出 q_2, q_3, q_4 均在初始核心对象的邻域内, 即已标记了类, 此时仅对 q_1 计算邻域进行类的扩展, 这样显然会造成大量数据对象的丢失。

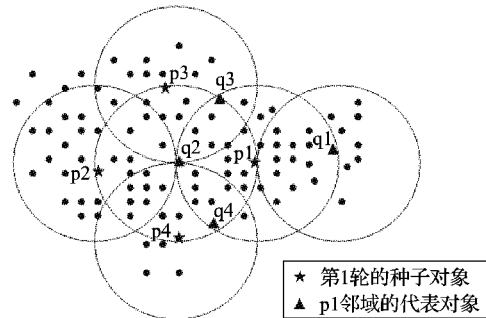


图 1 二维空间中的邻域及其代表对象

2.2.2 IF-DBSCAN 算法的基本思想

IF-DBSCAN 算法针对上述两个问题而提出, 算法的基本思想是在保证不丢失对象的基础上, 选择合适的代表对象进行类的扩展, 从而提高算法的时间性能。

首先讨论第二个问题。在一个核心对象的邻域内选择 4 个代表对象作为种子对象是不存在丢失对象的问题。但是, 当用这 4 个种子对象进行类扩展时, 如果再次选择它们邻域的 4 个代表对象继续扩展, 将会丢失对象, 此时应当选择足够多的代表对象。显然, 在核心对象的邻域中, 外围数据点的邻域可以完全覆盖内部数据点的邻域, 因此, 可以选择距核心对象较远的一圈数据作为代表对象进行全方位的类扩展。至于这一圈数据的距离应当尽量的远 (即接近 ϵ), 这样才能发挥这一算法的效率, 同时应当保证在各个方位都能选中代表对象。于是, IF-DBSCAN 选择半径在 0.9ϵ 之外的数据作为代表对象。

下面讨论第一个问题。在对 4 个种子对象进行处理时, 如果某一个种子对象不是核心对象将不再进行类的扩展, 同时也放弃它邻域中的所有对象。这将导致本应属于该类的数据对象而被未归为此类, 从而导致许多小类的生成。其实, 当某一个种子对象 A 不是核心对象, 但如果其邻域中对象个数接近于 $MinPts$, 则很大程度上意味着: 在它的附近 (更接近原始核心对象的方向) 存在一个核心对象 B, 显然这一核心对象中的所有数据对象都应归为此类。很容易理解, 这个核心对象 B 的邻域与非核心对象 A 的邻域几乎完全重合。于是, 可以这样处理, 当某一子对象的邻域中的对象个数接近于 $MinPts$ 时, 将其邻域中所有数据对象归为此类。这样做, 可能导致本应属于噪声的个别数据进行了归类。通过实验, 我们采用 $0.8MinPts$ 这一参数来作为处理条件, 这个参数可以很好地避免数据对象的丢失, 又可以从很大程度上区分出噪声数据。

综上所述, IF-DBSCAN 算法的具体作法为: 在一个核心对象的邻域中选与核心对象最远的 4 个代表对象作为种子对象进行类的扩展, 在对 4 个种子对象进行类扩展时, 把其邻

域内距离大于或等于 0.9ϵ 的未归类的数据对象作为代表对象进行类扩展;另外,如果种子对象不是核心对象,但其邻域中对象个数大于或等于 $0.8MinPts$,则将其邻域中未归类的数据对象归为此类,不再进行类的扩展。

2.3 IF-DBSCAN 算法框架

IF-DBSCAN 算法是 DBSCAN 算法的另一个快速版本。与 FDBSCAN 相比,主程序 IF-DBSCAN 不需要调用丢失点处理过程,ExpandCluster 过程也有所改变。其算法框架描述如下。

```

IF-DBSCAN( SetofPoints, Eps, MinPts, Representative-MinPts)
{ //SetofPoints 中的所有点被初始化为 UNCLASSIFIED
  ClusterId = nextId( NOISE);
  for( i = 1; i < SetofPoints.size; i++)
  {
    Point = SetofPoints.get( i)
    if( Point.CId == UNCLASSIFIED)
    {
      if ( ExpandCluster( SetofPoints, Point, ClusterId, Eps, MinPts,
        Rep-representative-MinPts))
        ClusterId = nextId( ClusterId)
    }
  }
}
ExpandCluster( SetofPoints, Point, ClusterId, Eps, MinPts, Rep
representative-MinPts)
{
  Candidate-seeds = SetofPoints.regionquery( Point, Eps);
  if ( candidate-seeds.size < MinPts)
  { //Point 为一边界点
    SetofPoint.changeCId( Point, NOISE);
    return 0;
  }
  else
  { //Point 为一核心点
    SetofPoints.changeCIds( candidate-seeds, CId);
    Representative-Seeds-Select( candidate-seeds,
      representative-seeds, Representative-MinPts, Point);
    while( representative-seeds!= Empty)
    {
      currentP = representative-seeds.first();
      result = SetofPoints.regionquery( currentP, Eps);
      if( result.size >= MinPts)
      { //currentP 为核心点
        for each point P in representative-resultP
          if( P.CId == UNCLASSIFIED)
            representative-seeds.append( P);
        for each point p in result
        {
          if ( p.CId == UNCLASSIFIED or NOISE)
            SetofPoints.changeCId( p, CId);
          if ( distance( P, currentP) >= 0.9Eps)
            //P 距离 currentP 在 0.9Eps 之外
            Rrepresentative-Seeds-Select( result, representative-resu ltP,
              Representative-MinPts, P);
        }
      }
    }
  }
  else
  { //currentP 不是核心点
    if( result.size >= 0.8MinPts)
    for each point p in result
      if ( p.CId == UNCLASSIFIED or NOISE)

```

```

      SetofPoints.changeCId( p, CId);
    }
    Representative-seeds.delete( currentP);
  }
  return 1;
}

```

3 相关实验

本文所涉及的算法均用 C 语言编程实现,并在 Windows XP,VC++ 6.0 环境下调运行,所有测试在 1 台 PC 机 (PIII800 CPU,256 MB 内存,20 GB 硬盘)进行。

3.1 IF-DBSCAN 算法的正确性对比实验

为了验证 IF-DBSCAN 算法的正确性,采用 5 个二维数据集进行实验,如图 2 所示。

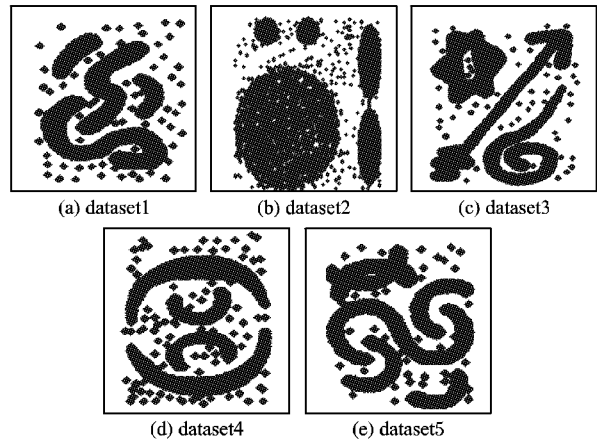


图2 原始数据集

其中,dataset1 有 9993 个点,数据自然聚为 4 类;dataset2 有 5034 个点,数据自然聚为 5 类;dataset3 有 12917 个点,数据自然聚为 3 类;dataset4 有 8487 个点,数据自然聚为 4 类;dataset5 有 11183 个点,数据自然聚为 5 类。

本文采用相同的参数验证 DBSCAN 算法和 IF-DBSCAN 算法,所得到的聚类结果完全相同,如图 3 所示。其中,dataset1 所采用参数为: $\epsilon = 4, MinPts = 12$; dataset2、dataset3、dataset4 所采用参数均为: $\epsilon = 3, MinPts = 15$; dataset5 所采用参数为: $\epsilon = 4, MinPts = 15$ 。

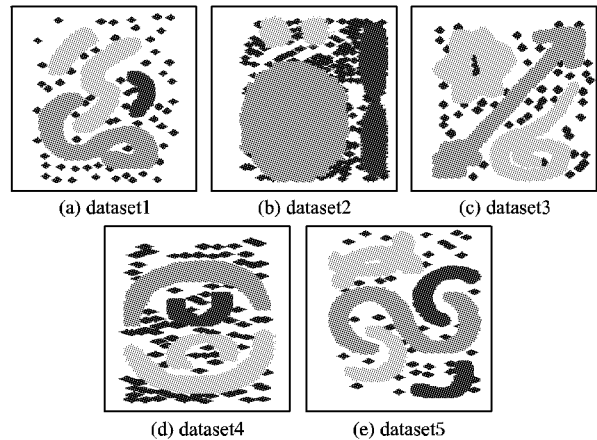


图3 DBSCAN 和 IF-DBSCAN 算法执行结果

通过对实验结果数据分析发现:数据集 dataset1、dataset3、dataset4 和 dataset5 在 DBSCAN 算法和 IF-DBSCAN 算法中所执行的结果完全相同,每一个点的聚类结果都一样,只有在数据集 dataset2 中,有个别数据点的聚类结果有差异。由此可见,IF-DBSCAN 算法是正确的,它可以与 DBSCAN 算

法具有相同的聚类功效。

3.2 IF-DBSCAN 算法的执行时间对比实验

在测试 IF-DBSCAN 算法与 DBSCAN 算法的执行时间时, 仍然采用上述 5 个数据集, 其中时间均以秒为单位。其执行时间对比如表 1 所示。

表 1 两种算法的执行时间对比

算法	数据集				
	dataset1	dataset2	dataset3	dataset4	dataset5
DBSCAN 算法	26	7	46	17	33
IF-DBSCAN 算法	19	3	21	8	23

从表 1 中数据可以看出, IF-DBSCAN 算法总是比 DBSCAN 算法的速度快, 尤其是对于数据集 dataset2, IF-DBSCAN 算法的执行时间只是 DBSCAN 算法执行时间的 42%。可见, IF-DBSCAN 算法与 DBSCAN 算法相比, 其时间性能是高效的。

为了验证算法执行时间与数据规模的关系, 采用数据集 dataset4 (参数: $\varepsilon = s$, $MinPts = 15$) 进行实验。本次实验分别选取 dataset4 中的 3000 个、5000 个、7000 个、9000 个、11000 个及全部数据点, 其实验结果图 4 所示。

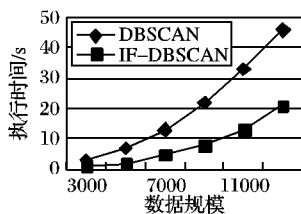


图 4 数据规模对执行时间的影响

由图 4 可以看出, 两个算法的执行时间近似抛物线, 这表明这两个算法的时间复杂度与数据的规模成平方的关系。从图中直观的可以发现, IF-DBSCAN 算法的执行效率明显比 DBSCAN 算法高, 而且随着数据规模的增长仍能保持良好的状态, 说明 IF-DBSCAN 算法具有良好的可扩展性。

4 结语

IF-DBSCAN 算法与 DBSCAN 算法相比, 它们具有相同的时间复杂度和空间复杂度。但是, DBSCAN 算法对每一个数

据点都计算其 ε 邻域, 建立其邻域链表, 而 IF-DBSCAN 算法尽管没有降低时间复杂度, 但却大大减少了计算 ε 邻域的数据点, 从而减少了算法的执行时间, 提高了其时间性能。另一方面, 由于 IF-DBSCAN 算法中存在大量的数据点并不进行邻域查询, 所以在该算法中数据点后的存在许多空的链表, 从而也减少了空间的使用, 提高了空间性能。

参考文献:

- [1] CHEN M S, HAN J H, YU P S. Data mining: An overview from a database perspective [J]. IEEE Transactions on Knowledge and Data Engineering, 1996, 8(6): 866-883.
- [2] KAUFAN L, RPUSSEUW P J. Finding groups in data: An introduction to cluster analysis [M]. New York: John Wiley & Sons, 1990.
- [3] ESTER M, KRIEGEL H P, XU X W. Knowledge discovery in large SPATIAL database: Focusing techniques for efficient class identification [C]// Proceedings of the 4th International Symposium on Advances in Spatial Databases, LNCS 951. London: Springer-Verlag, 1995: 67-82.
- [4] ESTER M, KRIEGEL H P, SANDER J, et al. A density-based algorithm for discovering clusters in large spatial database with noise [C]// KDD-96: Proceedings of the 2nd International Conference on Knowledge Discovering and Data Mining. Portland, Oregon: [s. n.], 1996: 226-231.
- [5] GUHA S, RASTOGI R, SHIM K. CURE: An efficient clustering algorithm for large databases [C]// Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data. New York: ACM Press, 1998: 73-84.
- [6] AGRAWAL R, GEHRKE J, GUNOPOLOS D, et al. Automatic subspace clustering of high dimensional data for data mining application [C]// Proceedings of the ACM SIGMOD International Conference on Very Large Data Bases. Roma: Morgan Kaufmann Publishers, 2001: 331-340.
- [7] ALEXANDROS N, YANNIS T, YANNIS M. C2P: Clustering based on closest pairs [C]// Proceedings of the 27th International Conference on Very Large Databases. Roma: Morgan Kaufmann Publishers, 2001: 331-340.
- [8] 周水庚, 周傲英, 曹晶, 等. 一种基于密度的快速聚类算法[J]. 计算机研究与发展, 2000, 37(11): 1287-1292.

(上接第 2493 页)

- [2] HECKERMAN D, GEIGER D, CHICKERING D. Learning Bayesian networks: The combination of knowledge and statistical data [J]. Machine Learning, 1995, 20(3): 197-243.
- [3] GEIGER D, HECKERMAN D. Learning Gaussian networks [C]// Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence. San Francisco, CA: [s. n.], 1994: 235-243.
- [4] 王飞, 刘大有, 薛万欣. 基于遗传算法的 Bayesian 网中连续变量离散化的研究[J]. 计算机科学, 2002, 25(8): 794-800.
- [5] FAYYAD U, IRANI K. Multi-interval discretization of continuous-valued attributes for classification learning [C]// IJCAI'93: Proceedings of 1993 International Joint Conference on Artificial Intelligence. San Francisco, CA: Morgan Kaufmann, 1993: 1022-1027.
- [6] PFAHRINGER B. Compression-based discretization of continuous variables [C]// Proceedings of the 12th International Conference on Machine Learning. San Francisco, CA: Morgan Kaufmann, 1995: 456-463.
- [7] FRIEDMAN N, GOLDSZMIDT M. Discretization of continuous attributes while learning Bayesian networks [C]// ISML'96: Proceedings of the 13th International Conference on Machine Learning. 1996: 157-165.
- [8] 王国胤. Rough 集理论与知识获取[M]. 西安: 西安交通大学出版社, 2001.
- [9] 潘巍, 李晋川, 王阳生, 等. 基于决策的剥离式连续属性离散化算法[J]. 计算机科学, 2007, 34(8): 208-210.
- [10] NGUYEN H S, SKOWRON A. Quantization of real values attributes rough sets and Boolean reasoning approaches [C]// Proceedings of the 2nd Joint Annual Conference on Information Science. Wrightsville Beach, NC: [s. n.], 1995: 34-37.
- [11] NGUYEN H S. Discretization problem for rough sets methods [C]// RSCTC'98: Proceedings of the 1st International Conference on Rough Sets and Current Trends in Computing. Warsaw, Poland: [s. n.], 1998: 545-552.
- [12] 赵军, 王国胤, 吴中福, 等. 基于粗集理论的数据离散化方法[J]. 小型微型计算机系统, 2004, 25(1): 60-64.
- [13] 彭佳文. 一种改进的启发式离散化算法及应用[J]. 计算机与现代化, 2008(9): 51-53.