基于MapReduce自适应参数的粗糙K-modes算法研究

杨 阳[1] 张为群[1] 刘 枫[1,2] 黄仁杰[1]
(西南大学计算机与信息科学学院 重庆 400715)[1]
(重庆市智能仪表及控制装备工程技术研究中心 重庆 400715)[2]

摘 要 粗糙K-modes聚类算法需要根据经验为w_l、w_u和ε 3个参数设定其固定值,聚类效果不稳定,容易受到噪声干扰。提出一种基于MapReduce自适应参数的粗糙K-modes算法,它根据聚类不同阶段的特点自动调整参数值,优化聚类效果。在此基础上,对自适应参数的粗糙K-modes算法进行MapReduce并行化设计,以提高算法的运行效率。实验证明,提出的自适应参数的粗糙K-modes算法聚类效果稳定,通过对算法的并行设计提高了算法对大规模数据的聚类分析性能。

关键词 粗糙K-modes,自适应参数,MapReduce并行化
中图法分类号 TP391 **文献标识码** A

Research on Rough K-modes Algorithm with Adaptive Parameters Based on MapReduce

YANG Yang[1] ZHANG Wei-qun[1] LIU Feng[1,2] HUANG Ren-jie[1]
(Faculty of Computer and Information Science, Southwest University, Chongqing 400715, China)[1]
(Chongqing Engineering Research Center for Instrument and Control Equipment, Chongqing 400715, China)[2]

Abstract In the traditional rough K-modes algorithm, three important parameters, w_l, w_k and ε, are set fixedly, making the clustering result unstable and interfered easily by noise. We proposed a rough K-modes algorithm based on MapReduce adaptive parameters, which adjusts parameters depending on the characteristics of different clustering stages, optimizing the clustering result. In addition, we designed the rough K-modes algorithm with adaptive parameters that can be used in MapReduce to improve the efficiency of the algorithm and performance of clustering large-scale data. Finally, this algorithm's validity is proved by experiments.

Keywords Rough K-modes, Adaptive parameters, MapReduce validity

1 引言

聚类分析[1]是数据挖掘领域的一个重要分支,由于不需要数据集结构的任何先验知识,因此它在机器学习和智能技术等领域,通常被称为无监督的学习方法。聚类的目标是将一个数据集划分为若干个子类,使类内对象尽可能相似,而类间对象尽可能相异。

在众多聚类算法中,K-means算法[1]以高效而著称,然而由于其局限于处理数值型数据集,使得应用受到限制。K-modes算法[2]就是在字符型数据集范围内对K-modes算法进行扩展的一个有效聚类算法,它继承了K-means高效的特点,且算法简单、易实现,因此被广泛应用于各个领域。它采用简单匹配方法度量同一分类属性下两个属性值之间的距离,用Mode代替K-means聚类的means,通过基于频率的方法在聚类过程中不断更新mode。然而随着聚类任务复杂性的提高,K-modes聚类算法的缺点日益突出。近年来,国内外学者针对K-modes算法的不足进行了优化和改进。

在相异度度量方面,Ng等人[3]利用基于相对频率的相异度度量对传统的K-modes聚类算法进行了改进,有效地提高了聚类精度。

不管是传统的K-modes算法还是Ng改进的新K-modes算法[3],在计算类中心时都隐含假定类中各数据对象具有一样的重要性,没有考虑类中数据对象对类中心的不同影响[4]。针对这个问题,李仁侃等人[4]提出一种粗糙K-modes算法,利用粗糙集[5]的上、下近似思想划分数据对象,考虑类中各数据点的不同贡献。在粗糙K-modes中主要用到3个参数w_l,w_u和ε,这3个参数的取值对聚类结果有着重要的影响。然而在经典的粗糙K-modes算法中,这3个参数都是根据经验进行设定的,而且一旦设定便无法修改。这种静态设定方法使得聚类效果不稳定,容易受到噪声干扰。

针对这个问题,本文提出一种基于自适应参数的粗糙K-modes聚类算法,它根据聚类不同阶段的特点,自动调整参数

到稿日期:2012-01-09 返修日期:2012-05-05 本文受中央高校基本业务科研费(XDJK2011C075),重庆市信息产业发展资金项目(2010 16005),重庆市智能仪表及控制装备工程技术研究中心资助。

杨 阳(1982-),男,硕士生,主要研究方向为模式识别与机器学习、无线传感网等,E-mail:yycia@swu.edu.cn;张为群(1950-),男,教授,硕士生导师,主要研究方向为软件工程、形式语言、软件测试等,E-mail:zhangwq@swu.edu.cn;刘 枫(1950-),男,教授,硕士生导师,主要研究方向为计算机控制、控制网络和智能仪表等;黄仁杰(1979-),男,博士生,讲师,主要研究方向为计算机控制系统、模式识别与机器学习。

值的变化。实验表明，该算法收敛速度快、聚类性能高、聚类效果稳定。

粗糙 K-modes 聚类算法计算复杂，同时随着数据量的增大，单台服务器容易出现内存不足等性能瓶颈。为了解决该问题，本文对粗糙 K-modes 算法进行了基于 MapReduce 编程模型[6,7]的并行设计。实验表明，基于本文设计的并行粗糙 MapReduce 算法执行效率高、性能高，可扩展性好。

2 相关理论和技术

2.1 传统 K-modes 算法

K-modes 聚类算法是通过对 K-means 聚类算法进行扩展，使其应用于分类属性数据聚类。它采用简单匹配的方法度量同一分类属性下两个属性值之间的距离，用 Mode 代替 K-means 聚类的 means，通过基于频率的方法在聚类过程中不断更新 Mode。

K-modes 聚类算法的基本步骤如下：

Step1 从数据集中随机选择 k 个对象作为初始类中心，其中 k 表示聚类个数。

Step2 应用简单匹配方法计算对象与类中心之间的距离，并将每个对象分配到离它最近的类中。

Step3 基于频率方法重新计算各类的类中心。

Step4 重复上述 Step3，Step4，直到目标函数 F 不再发生变化为止。

2.2 Ng 的改进 K-modes 算法

在文献[3]中，Ng 等人提出了新 K-modes 聚类算法。为了挖掘数据对象与类中心具有相同分类属性时隐含的关系，新 K-modes 算法统计了 m 个属性的所有分类值在各类中的出现频率，各类分别用类中 m 个出现频率最高的属性分类值更新类中心，并记录它们的出现频率，以此作为启发式信息来定义数据点与类中心的新的相异度度量，即使出现相同的类中心，数据对象分配时也可以通过类中心各属性取值在类中的支配地位（相对频率）来选择不同的类。

定义 1 属性集 $A=\{a_1,a_2,\cdots,a_m\}$，类中心 $Z=\{Z_1,Z_2,\cdots,Z_l\}$，$X_i\in U(1\leqslant i\leqslant n)$ 和 $Z_l(1\leqslant l\leqslant k)$，新的相异度度量定义如下：

$$d_n(X_i,Z_l)=\sum_{j=1}^{m}\varphi(x_{ij},z_{lj}) \tag{1}$$

式中，$\varphi(x_{ij},z_{lj})=\begin{cases}1, z_{lj}\neq x_{ij}\\ 1-|c_{l,j,r}|/|c_l| \end{cases}$，$|c_l|$ 为第 l 类的对象个数，即：$|c_l|=|\{i|w_{i,l}=1\}|$；$|c_{l,j,r}|$ 为第 l 类中数据对象在第 j 个属性的取值为 $a_j^{(r)}$ 的个数，即：$|c_{l,j,r}|=|\{w_{s,l}|z_{l,j}=x_{s,j}=a_j^{(r)},w_{s,l}=1\}|$。$\frac{|c_{l,j,r}|}{|c_l|}$ 为第 j 个属性取值为 $a_j^{(r)}$ 在 l 类中的支配地位。

2.3 粗糙 K-modes 算法

经典 K-modes 算法[4]在类中心的计算时，假定所有数据点对类的贡献是一致的，忽略了现实中每个类存在一些边界点和核心点。粗糙集理论的核心思想是利用上、下近似来描述不确定的区域，李仁侃等人[4]将粗糙集理论引入 K-modes 算法，提出粗糙 K-modes 算法，把既可能属于一类也可能属于另一类的点归入所有可能所属类的上近似，以确定属于某一类的点归入所属类的下近似。当一个数据点与某一类的相异度接近于与所属类的相异度时，把该数据点划分为该类和所属类的上近似；如果不存在这样的类，则把数据点划分为所属类的下近似。通过上、下近似，可以找出该类的边界点。显然，在求解一个聚类的中心时，边界点的权重应该更小，使类中心更合理地表示该类的分布。因此引进下近似和边界点的权重 w_{lower} 和 $w_{boundry}$（$w_{lower}+w_{boundry}=1$）以及控制阈值 ε。

定义 2 属性集 $A=\{a_1,a_2,\cdots,a_m\}$，类中心 $Z=\{Z_1,Z_2,\cdots,Z_l\}$，$X_i\in U(1\leqslant i\leqslant n)$ 和 $Z_l(1\leqslant l\leqslant k)$，改进后的相异度度量如式(1)所示，$|c_l|$ 改为考虑了边界点后第 l 类内所有数据对象的加权比重总和，即：

$$|c_l|=w_{lower}\times\sum_{X_i\in C_l}^{i=1,2,\cdots,n}w_{i,l}+w_{boundry}\times\sum_{X_i\in C_l^B}^{i=1,2,\cdots,n}w_{i,l} \tag{2}$$

$|c_{l,j,r}|$ 改为考虑了边界点后的第 l 类中数据对象在第 j 个属性的取值为 $a_j^{(r)}$ 的加权比重总和，即：

$$|c_{l,j,r}|=w_{lower}\times\sum_{x_i\in C_l,x_{i,j}=a_j^{(r)}}^{i=1,2,\cdots,n}w_{i,l}+w_{boundry}\times\sum_{x_i\in C_l^B,x_{i,j}=a_j^{(r)}}^{i=1,2,\cdots,n}w_{i,l} \tag{3}$$

粗糙 K-modes 算法的最优化目标函数以及隶属度更新方法同经典 K-modes 算法，其中相异度量方法采用定义 2 的度量方式。

3 基于自适应参数的粗糙 K-modes 算法

3.1 自适应参数设定

在传统粗糙 K-modes 算法中主要用到 3 个参数 w_l，w_u 和 ε，这 3 个参数的取值对聚类结果有重要的影响[8]。参数 w_l 控制着下近似中的样本对聚类中心的影响程度，参数 w_u 控制着上近似中的样本对聚类中心的影响程度，参数 ε 的大小决定着样本隶属关系的清晰程度。在传统的粗糙 K-modes 算法中，这 3 个参数都是根据经验进行设定的，而且一旦设定便无法修改。聚类是一个动态变化的过程，这种静态设置参数的方法无法适应聚类前期和后期的特点，使得聚类收敛慢、聚类效果不理想。针对这个问题，本文根据粗糙 K-modes 聚类算法的特点，定义了 3 个参数变化曲线。

聚类初期，聚类中心在一个较大的幅度内调整，聚类中心没有明显显现出来，样本归属关系变动较大，因此大部分样本属于上近似，即聚类中心的调整应该主要依赖于上近似的样本。聚类后期，聚类中心基本确定，大部分样本的归属关系已明确，这时大部分样本属于相应的下近似中，即必然属于该类。因此聚类后期的聚类中心主要依赖于下近似样本。

由此可见，在整个聚类过程中，控制下近似对样本中心的影响程度的参数 w_l 应随着迭代次数的增加而由小变大，最后逐步趋于稳定；相应地，参数 w_u 随着迭代次数的增加而由大变小。因此构造 w_l 和 w_u 变化曲线，如式(4)和式(5)所示。首先为了实现 w_l 和 w_u 参数的调节，构造一个 Logistic 增长曲线，如式(4)和式(5)所示。

$$w_l=\frac{1}{k+ae^{-bt}} \tag{4}$$

$$w_u=1-\frac{1}{k+ae^{-bt}} \tag{5}$$

根据 ε 的意义可知，在聚类初期，样本的隶属关系不明确，ε 值应该大一些，即聚类中心的调整主要依赖于上近似的样本。随着迭代次数的增加，样本的隶属关系变得越清晰，ε 值应该随着迭代次数的增加变得越小，使得聚类后期聚类中

心的调整依赖于下近似中的样本。因此，构造 ε 变化曲线，如式(6)所示。

$$\varepsilon=\varepsilon-\frac{1}{t^2} \tag{6}$$

3.2 基于自适应参数的粗糙 K-modes 算法

基于自适应参数的粗糙 K-modes 聚类算法基本过程与传统的粗糙 K-modes 算法一致，具体步骤如下：

输入：聚类数 K，样本集 $X=\{X_1,X_2,\cdots,X_N\}$

输出：聚类结果 $M=\{m_1,m_2,\cdots,m_K\}$

Step1 初始化。随机选取 K 个样本作为初始聚类中心 Z^0。

Step2 根据用定义 1 和定义 2 更新隶属度矩阵获得 W^0，并使得 $X_i\in\overline{C_l}(w_{i,l}=1)$；如果 $\exists Z_j(1\leqslant j\leqslant k,j\neq l)$，使得 $d_n(X_i,Z_j)-d_n(X_i,Z_l)\leqslant\varepsilon$，则 $X_i\in\overline{C_j}$；否则 $X_i\in\underline{C_l}$。计算 $F(W^0,Z^0)$，设 $t=0$。

Step3 令 $W'=W^t$，计算上、下近似并更新各类属性分类值出现频率，使 $X_i\in\overline{C_l}(w_{i,l}=1)$。如果 $\exists Z_j(1\leqslant j\leqslant k,j\neq l)$，使得 $d_n(X_i,Z_j)-d_n(X_i,Z_l)\leqslant\varepsilon$ $(\varepsilon=\varepsilon-\frac{1}{t^2})$，则 $X_i\in\overline{C_j}$；否则 $X_i\in\underline{C_l}$，并更新聚类中心获得 $Z^{t+1}(C_l^\beta=\overline{C_l}-\underline{C_l})$，判断是否出现空类，若出现，则新的类中心用初始的类中心代替，如果 $F(W',Z^t)=F(W',Z^{t+1})$，输出 W'、Z^t 并停止；否则转到 Step4。

Step4 令 $Z'=Z^{t+1}$，更新隶属度矩阵，获得 W^{t+1}，如果 $F(W^t,Z')=F(W^{t+1},Z')$，输出 W^t、Z' 并停止；否则，令 $t=t+1$ 并转入 Step2。

4 基于自适应参数的粗糙 K-modes 算法的 MapReduce 并行化设计

从 K-modes 算法中可以看出，算法的主要工作包括两项：将每个样本分配到相应聚类的上、下近似中，统计各类属性分类值出现的频率。

4.1 分配样本

自适应参数的粗糙 K-modes 算法的第一个工作是分配不同样本到相应聚类的上、下近似且该操作是相互独立的，因此可以并行执行这一步，即启动一次 MapReduce 计算过程：完成数据对象到聚类中心的距离计算并将对象分配到相应聚类的上、下近似中。

Map 函数设计：Map 函数的任务是完成每个对象到聚类中心的计算并根据 ε 分配每个对象到相应聚类的上、下近似中。其输入为待聚类所有对象数据和上一轮聚类(或初始聚类)中心，输入数据对象〈*key*，*value*〉的形式为〈行号，记录行〉；每个 Map 函数都读入聚类中心描述文件，Map 函数对输入的每个对象 X_i 计算其与每个聚类中心 Z_j 的距离 $d(X_i,Z_j)$，将 X_i 划分到与其距离最小的聚类中心 Z_l 的上近似，即 $X_i\in\overline{C_l}$。如果 $\exists Z_j(1\leqslant j\leqslant k,j\neq l)$，使得 $d_n(X_i,Z_j)-d_n(X_i,Z_l)\leqslant\varepsilon(\varepsilon=\varepsilon+\frac{1}{t^2})$，则 $X_i\in\overline{C_j}$；否则 $X_i\in\underline{C_l}$。输出中间结果〈key'，$value'$〉，其中 key' 为聚类类别 ID，$value'$ 包括对象的属性向量和该对象属于该类上近似标记或者下近似标记。

Reduce 函数设计：Reduce 函数的任务是根据 Map 函数得到的中间结果，将具有相同 key' 的合并，将合并的中间结果按照分类属性类别重组为〈*key*，*value*〉，其中 *key* 为分类属性类别 ID，*value* 为相应属性的取值向量。该中间结果将作为下一个 MapReduce 计算过程的 Map 函数的输入。

4.2 更新聚类中心

实现自适应参数的粗糙 K-modes 聚类算法的第二个主要任务是统计各类属性分类值出现的频率。其中，统计各类属性分类值出现频率的操作相互独立，因此考虑将这一步并行执行。即启动第二次 MapReduce 计算过程：完成各类属性分类值出现频率的统计并更新聚类中心。

Map 函数设计：Map 函数的主要任务统计各类属性分类值出现的频率。其输入为上一个 MapReduce 计算过程 Reduce 函数的输出〈*key*，*value*〉；从 *value* 中解析出对应于 *key* 属性的各个属性值，并识别每个属性值所属的上、下近似集合种类。然后根据公式 $\frac{|c_{l,j,r}|}{|c_l|}$ 计算对应的属性值出现的频率。输出中间结果〈key'，$value'$〉，其中 $key'=key$，$value'$ 为出现频率最高的属性值。

Reduce 函数设计：Reduce 函数的主要任务是根据 Map 函数获得的中间结果更新聚类中心，判断聚类算法是否收敛。如果算法收敛，则终止算法，输出聚类结果；否则将新的聚类中心写入聚类中心描述文件，进入下一个循环。

5 实验及分析

5.1 基于自适应参数的粗糙 K-modes 聚类算法实验

本实验在实验室 PC 机上运行，PC 机配置如下：CPU 型号为 Intel Xeon X330；内存为 4GB；硬盘为 512GB SATA。实验采用 4 组经典 UCI 数据集，选用的实验数据集如表 1 所列。

表 1 实验数据集描述

数据集	样本数目	分类属性数目	聚类个数
Vote	435	16	8
SPECT	267	22	4
Mushroom	8124	22	12
Breat-cancer	286	9	3

为验证本文所提算法的聚类效果，分别将本文算法与李仁侃等人的粗糙 K-modes 聚类算法、梁吉业等人[5]的基于新的距离度量的 K-modes 聚类算法进行对比。在相同的实验数据集上分别运行 3 种算法各 100 次，通过计算平均聚类质量来验证算法的有效性。聚类质量通过聚类正确度 FM[9]进行衡量，将聚类结果与数据集内在分类结构进行对比，FM 值越大，说明聚类结果的结构与数据集内在结构越相似，换句话说，聚类精确度值 FM 越大，说明聚类效果越好。实验结果如表 2 所列。

表 2 聚类精度实验结果对照表

数据集	粗糙 K-modes 聚类算法	新的距离度量 K-modes 聚类算法	本文算法
Vote	0.7842	0.7139	0.8226
SPECT	0.7439	0.8152	0.8336
Mushroom	0.9081	0.8573	0.9179
Breat-cancer	0.6062	0.6993	0.7981

由表 2 可以看出，本文提出的自适应参数粗糙 K-modes 聚类算法对 3 个参数进行自适应设置，降低了噪声的干扰程度，且聚类结果的精确度有所提高。

为了比较本文算法与传统粗糙聚类算法对聚类效果的稳定性，在 Vote 数据集的 100 次实验中，任选 4 次实验结果进行统计和分析，如表 3 所列。

表 3　聚类稳定性实验结果对照表

Vote 实验次	粗糙 K-modes 聚类算法	新的距离度量 K-modes 聚类算法	本文算法
第 26 次	0.8129	0.6121	0.8179
第 59 次	0.6591	0.8166	0.8416
第 72 次	0.7377	0.7023	0.8183
第 100 次	0.8579	0.7278	0.8281

为了进一步验证本文算法对聚类结果的稳定性及精度的准确性，重新对实验数据集进行设计。从 Mushroom 数据集中分别随机选取 1000 个样本、4500 个样本及 8124 个样本作为数据集，重新运行算法 100 次，将平均聚类质量作为实验结果来验证算法的稳定性。实验结果如表 4 所列。

表 4　聚类实验结果对照表

数据集	粗糙 K-modes 聚类算法	新的距离度量 K-modes 聚类算法	本文算法
1000 个样本	0.7952	0.8991	0.9126
4500 个样本	0.9216	0.7676	0.9036
8124 个样本	0.9082	0.8568	0.9179

由表 3 和表 4 可以看出，由于传统粗糙 K-modes 聚类算法对 3 个参数进行固定设置，导致其对不同数据集的聚类效果不稳定，容易受到噪声的影响。本文对 3 个参数进行自适应设置，从而降低了噪声干扰的影响程度，聚类效果稳定性明显高于传统粗糙 K-modes 聚类算法。

5.2　基于自适应参数的粗糙 K-modes 聚类算法并行化实验

本实验验证串行自适应参数的粗糙 K-modes 聚类算法与并行自适应参数的粗糙 K-modes 聚类算法在执行效率方面的性能对比。实验是在实验室搭建的 Hadoop 平台上运行的。图 1 给出实验室搭建的云计算平台结构：1 台服务器作为 NameNode 和 JobTracker 节点，其它 10 台 PC 机分别作为 DataNode 和 TaskTracker 节点。

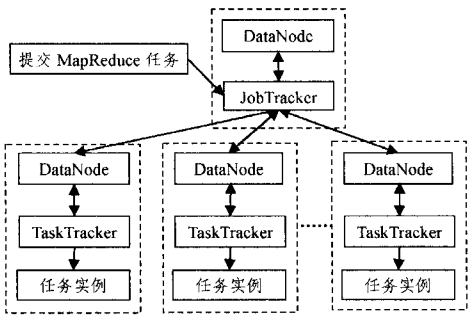

图 1　云计算平台结构示意图

5.2.1　单机实验对比

实验内容为比较 Hadoop 集群中的 1 个运算节点与自适应参数的粗糙 K-modes 聚类算法的串行实现软件在处理相同规模数据以及相同硬件配置环境下，从数据读入到完成聚类所需要的时间。在这两者的对比实验中，被处理数据从小规模逐步增加，实验情况如表 5 所列（t_1 是单价所用时间；t_2 是 1 个运算节点完成 K-modes 算法所用的时间）。

表 5　单机实验处理结果

实验次数	文件大小(kB)	t_1/s	t_2/s
1	3452	53	96
2	6684	118	134
3	18200	内存不足	207
4	37736	内存不足	384
5	82869	内存不足	652

上述实验表明：随着输入数据的增长，运行在单机上的内存资源消耗随之增大，导致机器性能下降，直至报告内存不足且不能完成计算任务，而 Hadoop 集群能完成计算任务，这初步体现在 Hadoop 集群上实现自适应参数的粗糙 K-modes 聚类算法具有处理大规模数据的能力；当输入数据很小时，Hadoop 集群上自适应参数的粗糙 K-modes 并行算法的处理效率低于单机上非并行化算法的处理效率，原因在于 Hadoop 集群上各个节点交互需要消耗一定的资源而增加处理延时。

5.2.2　加速比性能实验

加速比是衡量一个系统在扩展性方面优劣的主要指标，主要考察两个方面的性能，一是当计算硬件资源增加时，对于相同规模的作业，系统的处理能力；二是当计算资源和处理作业的规模保持相近比例增长时，系统的处理能力。

实验对 3 批实验数据进行测试，数据情况如表 6 所列。

表 6　实验数据集

实验数据集	文件大小/MB	数据块数	占用 HDFS 空间/MB
A	1784	78	15876
B	3978	158	29364
C	7915	306	59292

实验 1　分别选择 1,3,6,8,10 这几个 TaskTracker 节点参与运算，考察在逐渐增加节点的情况下，系统中完成任务的时间，实验结果如图 2 所示。从图 2 可以看出：每个作业运行的时间都随节点的增加而降低，增加的节点可以显著提高系统对同规模数据的处理能力，说明 MapReduce 在处理 K-modes 聚类算法上具有较好的加速比。

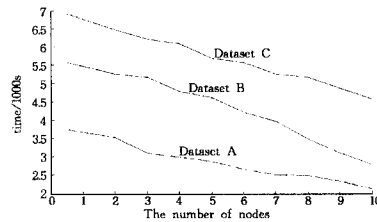

图 2　K-modes 聚类算法的 MapReduce 并行化实现加速比

实验 2　分别选择 3,6,9 这几个 TaskTracker 节点，对应选择 A,B 和 C 数据集进行计算，实验结果如图 3 所示。从图 3 可以看出：当节点数和处理数据的规模呈正比增长时，MapReduce 在处理数据上基本保持相同的水平，表现出了良好的扩展性。

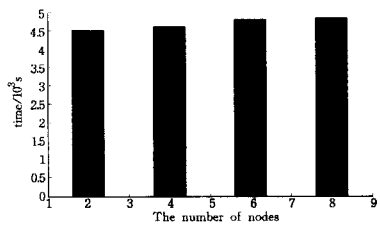

图 3　K-modes 聚类算法 MapReduce 实现方法的扩展性实验结果

结束语　粗糙 K-modes 算法对其涉及到的 3 个主要参数进行固定设置，导致算法聚类效果不稳定，容易受到数据集噪声的影响。本文在改进传统 K-modes 算法相异度度量的基础上，提出自适应参数的粗糙 K-modes 聚类算法。根据聚类前期和后期的不同特点，自动调整 3 个参数的值，使得粗糙 K-modes 算法适应性增强。实验证明，本文的算法聚类效果

（下转第 164 页）

表 1 （单位：s）

故障表示	故障个数					
	5	10	20	50	80	100
CTA	0	0	15	93	640	2174
RE(80)	30	80	161	280	440	625
RE(50)	15	40	72	115	195	295
RE(20)	15	43	70	83	98	150
RE(10)	15	31	48	68	100	135

在计算候选诊断集时，首先生成与当前观测一致的系统运行空间，删除不包含候选诊断的系统行为，然后根据系统的实际运行轨迹和故障模式空间计算可能发生的故障，得到候选诊断集。在系统的每次运行中与观测一致的行为模型远远小于系统的全局模型，显然这种分开处理的方法与文献[5]相比，可以降低原算法带来的高复杂性，得到的诊断系统具有更强的实用性。此外还可以通过添加时序信息缩小观测空间，或者在匹配过程中增加启发式函数以降低计算候选诊断集的时间复杂性。

结束语 本文提出了在基于正则表达式定义的故障下系统的可诊断性定义以及故障诊断过程，其克服了传统诊断方法的一些不足。在传统的故障诊断方法中将故障定义为一类特殊事件，规定当这些特殊事件发生时，系统发生故障。这种定义故障的方式存在如下局限，首先系统的可诊断性定义和诊断系统构建依赖于具体的系统模型，不能重复利用；其次，故障描述比较单一，没有将故障发生的因果关系和时序关系表示出来；最后，传统的故障诊断方法不能同时解决单故障、多故障、间歇性故障等不同的故障情形。

本文主要在以下几方面做了改进。首先与文献[4]比较，主要有(1)故障模式用正则表达式描述，代替了完备的自动机。这种故障定义方式使其可以在语义层上描述故障，也可以表示故障与系统之间的因果关系和时序关系，避免了文献[4]中故障定义完全与系统模型分离的不足。还可以描述故障的发生过程，认为故障是由一些类转移共同导致的；(2)通过构建故障模式空间描述多故障、间歇性故障等所有不同的故障情形，而不是通过穷举考虑每一种情况。其次在文献[5]的基础上，改进了构建故障模式空间的算法和计算候选诊断的算法，提出基于故障模式空间的可诊断性定义以及诊断系统。

正则表达式具有很强的表达能力，但毕竟不是专门为故障诊断设计的，在描述系统故障特征时可能会扩大问题的范围，从而造成匹配困难。作者进一步的研究是设计出更适合表示离散事件系统中系统故障的规则，提出高效的故障诊断方法。

参 考 文 献

[1] Sampath M, Sengupta R, Lafortune S, et al. Diagnosability of discrete-event systems[J]. IEEE Transactions on Automa-tic Control, 1995, 40(9): 1555-1575

[2] Baroni P, Lamperti G, Zanella M. Diagnosis of large active systems[J]. Artificial Intellig-ence, 1999, 110(1): 135-183

[3] Pencolé Y. Diagnosability analysis of distributed discrete event systems[A]// the16th European Conference on Artificial Inelligence, 2004[C]. Valencia, Spain, 2004: 43-47

[4] Jéron T, Marchand H. Supervision patterns indiscrete event systems diagnosis[A]// the 17th International Workshop on Principles of Diagnosis, 2006[C]. Peñaranda de Duero, Spain, 2006: 117-124

[5] Lamperti G, Zanella M. Injecting semantics in-to diagnosis of discrete-event systems[A]// the 21st International Workshop on Principles of Diagnosis, 2010[C]. Portland, America, 2010: 233-240

[6] Lamperti G, Zanella M. Context-sensitive diagn-osis of discrete-event systems[A]// Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence, 2011[C]. Catalonia, Spain, 2011: 969-975

[7] Jiang S, Kumar R. Failure diagnosis of discrete event systems with linear-time temporal lo-gic specifications[J]. IEEE Transactions on Automatic Control, 2004, 49(6): 128-133

[8] Jiang S, Huang Z. Apolynomial algorithm for te sting diagnosability of discrete event system[J]. IEEE Transactions on Automatic Control, 2001, 46(8): 1318-1321

[9] 刘磊. 编译原理及实现技术(第二版)[M]. 北京：机械工业出版社，2010

[10] Pencolé Y, Cordier M. A formal framework for the decentralized diagnosis of large scale d-iscrete event systems and its application to telecommunication networks[J]. Artificial In-telligence, 2005, 164(1/2): 121-170

[11] Briones L. Optimal observability for diagnos-eblity [A]// the 19st International Workshop on Principles of Diagnosis, 2008 [C]. Blue Mountains, NSW, Australia, 2008: 31-38

（上接第 152 页）

稳定、聚类精度高。另一方面，粗糙 K-modes 聚类算法计算复杂，同时随着数据量的增大，单台服务器容易出现内存不足等性能瓶颈。为了解决该问题，本文对粗糙 K-modes 算法进行了 MapReduce 并行设计，实验表明，算法在云计算平台上的性能得到提升，能够对海量数据进行有效聚类分析。

参 考 文 献

[1] Han Jia-wei, Kam B M. Data mining concepts and techniques [M]. San Francisco, USA: Morgan Kaufmann, 2001

[2] Huang Zhe-xue. Extensions to the k-means algorithm for clustering large data sets with categorical values[C]// Data Mining and Knowledge Discovery. Netherlands Kluwer Academic Publishers, 1998: 283-304

[3] Ng K N, Lim J, Huang J Z, et al. On the impact of dissimilarity measure in k-modes clustering algorithm [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2007, 29 (3): 503-507

[4] 李仁侃，叶东毅. 粗糙 K-Modes 聚类算法[J]. 计算机应用，2011, 31(1): 97-100

[5] 梁吉业，白亮，曹付元. 基于新的距离度量的 K-Modes 聚类算法[J]. 计算机研究与发展，2010, 47(10): 1749-1755

[6] 赵卫中，马慧芳，等. 基于云计算平台 Hadoop 的并行 k-means 聚类算法设计研究 [J]. 计算机科学，2011, 38(10): 166-176

[7] Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters[C]// Proceedings of Operating Systems Design and Implementation. San Francisco, CA, 2004: 137-150

[8] Huang Zhe-xue. Clustering large data sets with mixed numeric and categorical values[C]// Proc of PAKDD'97. Singapore: World Scientific, 1997: 21-35

[9] 赵恒，杨万海. 模糊 K-Modes 聚类精度分析[J]. 计算机工程，2003, 29(12): 27-28