第35卷 第17期 计 算 机 工 程 2009年9月
Vol.35 No.17 Computer Engineering September 2009
·软件技术与数据库· 文章编号：1000—3428(2009)17—0043—03 文献标识码：A 中图分类号：TP301.6

基于最大频繁项集的聚类算法

刘美玲
(广西民族大学数学与计算机科学学院，南宁 530006)

摘 要：介绍频繁项集的概念及其性质，把最大频繁项集作为聚类的依据，提出一种基于最大频繁项集的聚类算法，将关联分析与聚类分析相结合，在聚类中充分利用数据项间的关联性，无须输入聚类个数，并在多个数据集上进行实验。实验结果表明，与传统的基于距离的聚类算法 K-Means 相比，该算法减少计算数据对象间距离的时间花销，提高算法的效率，具有较高的聚类精度，聚类结果的可解释性也较强。

关键词：聚类分析；最大频繁项集；Apriori 性质

Clustering Algorithm Based on Maximal Frequent Itemsets

LIU Mei-ling
(College of Mathematics & Computer Science, Guangxi University for Nationalities, Nanning 530006)

【Abstract】 The concept and properties of frequent itemsets are introduced. The maximal frequent itemsets is made as clustering basis. A new Clustering Algorithm Based on Maximal Frequent Itemsets(CABMFI) is proposed, which integrates the association analysis and clustering analysis. The relevance between data items is fully used, and need not input the clustering number. It is tested with several datasets. Experimental results show that, compared with traditional distance-based K-Means clustering algorithm, this algorithm can reduce the time cost of computing the distance of objects, improve the efficiency, and has better accuracy. The interpretability of clustering results is also well.

【Key words】 clustering analysis; maximal frequent itemsets; Apriori properties

1 概述

聚类分析是数据挖掘中一个重要研究领域，它的目标是将数据集划分为若干类，使同一个类内的数据对象具有较高的相似度，而不同类中的数据对象不相似。关联规则挖掘是从大量的数据中找出隐含的、有价值的知识[1]。

目前的聚类算法有很多，如 K-Means、BIRCH、利用模拟退火算法的聚类方法[2]以及基于遗传算法的聚类方法，这些算法都有一定的理论基础并得到广泛应用。现有的一些算法在聚类前没有分析数据项间可能隐含的关系，并利用这些关系进行聚类，在一定程度上可能降低聚类的精确度。关联规则挖掘产生的频繁集与关联规则在一定程度上反映了数据之间的关联。

本文在分析关联规则中频繁项集的概念与性质的基础上，将关联规则挖掘的思想扩展到聚类分析中，以最大频繁项集作为聚类的依据，提出一种简单有效的聚类算法——基于最大频繁项集的聚类算法(Clustering Algorithm Based on Maximal Frequent Itemsets, CABMFI)。

2 算法的基本思想

2.1 关联规则挖掘

关联规则挖掘[3]的任务是从给定的数据集中发现数据项之间存在的有价值的联系。挖掘关联规则有 2 个步骤：(1)挖掘所有的频繁项集；(2)根据所得的频繁项集，产生相应的强关联规则。频繁项集具有以下 2 个性质，称为 Apriori 性质[3]：(1)一个频繁项集的任一子集也是频繁项集；(2)若一个项集是非频繁的，则该项集的所有超集都是非频繁的。

2.2 CABMFI 的基本思想

频繁项集表示常常伴随出现的若干数据项，它在一定程度上反映了数据集的特点以及数据项之间隐含的关系。因此，根据频繁项集的概念与性质，利用数据项的关联性在数据对象聚类前作预处理，然后根据数据项集合对数据对象进行聚类。算法的具体思想描述如下：

(1)关联规则挖掘过程中挖掘出来的频繁项集数量通常很大，而且项集之间可能存在着包含关系，为减少聚类过程中频繁项集的数量，首先利用 Apriori 性质在挖掘出的所有频繁项集中删除具有被包含关系的频繁子项目集，剩下所有的最大频繁项集，以最大频繁项集作为初始的聚类，此时的聚类是若干数据项的集合(若文中无特别说明，则聚类指的是数据项的集合)。

(2)一个数据集的最大频繁项集个数可能较多，且考虑到一个聚类内数据项之间的关联度可能较小而该聚类的数据项与另一聚类的数据项之间的相似度却较大。因此，根据聚类的目标，定义聚类的类内关联度与类间相似度的度量方法。根据该方法计算初始聚类的类内关联度与类间相似度，若某 2 个类的类间相似度大于相应类的类内关联度，则合并这 2 个类；重新计算新类的类内关联度以及与其他类的类间相似度，若有满足合并条件的类则将它们进行合并，重复该过程直到没有任何 2 个聚类间的相似度大于相应类内的关联度为止，这就得到优化后的聚类。

(3)将每个数据对象的属性值与各聚类的数据项进行比

基金项目：广西民族大学青年科研基金资助项目(0509QN32)
作者简介：刘美玲(1979－)，女，讲师，主研方向：数据库，数据挖掘，软件技术
收稿日期：2009-03-02 **E-mail**：lml97@126.com

较，若一个数据对象与某个聚类具有的相同特征最多，则将此对象划分到该聚类中，表明该对象更多地具有这个数据项聚类所表现的特征。

3 相关定义与算法的主要过程

在一个数据集中，对于具有相同支持度的 2 个频繁项集来说，可能一个频繁项集中数据项与数据项之间具有较高的关联度，而另一个项集内部具有较低关联度但与其他项集却具有较高的相似度，因为这些数据项之间的信任度可能不一样，所以使用支持度作为一个聚类的内部关联度无法区别这个不同点。因此，文中以信任度作为依据计算一个聚类的类内关联度与聚类间的相似度，然后对初始产生的所有聚类进行优化，将具有强关联的 2 个类合并为 1 个类。

3.1 类内关联度的度量

定义 1 假设 $I=\{i_1, i_2, \cdots, i_m\}$是数据库 D 中的所有数据项的集合，对于 $t_i, t_j \in i$，t_i 对 t_j 的依赖性定义为

$$depend(t_i \mid t_j) = p(t_i \mid t_j) = \frac{p(t_i \cup t_j)}{p(t_j)} \tag{1}$$

其中，$p(t)$表示项集 t 的支持度。

在关联规则挖掘中，$depend(t_i|t_j)$表示规则 $t_j \Rightarrow t_i$ 的信任度，即 $depend(t_i|t_j)=conf(t_j \Rightarrow t_i)$。项集$\{t_i, t_j\}$的关联度定义为

$$Adegree(t_i t_j) = \frac{depend(t_i \mid t_j) + depend(t_j \mid t_i)}{2} = \frac{conf(t_j \Rightarrow t_i) + conf(t_i \Rightarrow t_j)}{2} \tag{2}$$

一般地，对于一个项集 $t=\{t_1,t_2,\cdots,t_n\} \subseteq I$，该项集有很多子集，如$\{t_1, t_2\}$, $\{t_1, t_2, t_3\}$与$\{t_1, t_2, \cdots, t_k\}(3<k<n)$，如果把该项集的所有子集的关联度作为度量整个项集的关联度，则有很多子集被重复计算，因此，为准确地把握类内关联度，计算中只考虑所有的 2-项集，项集$\{t_1,t_2,\cdots,t_n\}$的关联度可以定义为所有项两两之间关联度的均值：

$$Adegree(t_1 t_2 \cdots t_n) = \frac{\sum_{i=1,j=1,i\neq j}^{n} Adegree(t_i t_j)}{C_n^2} = \frac{\sum_{i=1,j=1,i\neq j}^{n} depend(t_i \mid t_j)}{n(n-1)} = \frac{\sum_{i=1,j=1,i\neq j}^{n} conf(t_j \Rightarrow t_i)}{n(n-1)} \tag{3}$$

特别地，当 $n=1$ 时，$Adegree(t_1)=1$。对于一个项目集来说，若它的类内关联度越大，则它的内聚性越强。

3.2 类间相似度的度量

定义 2 给定 2 个聚类 C_p 与 C_q，它们的相似度采用平均距离表示，即等于 2 个聚类中的所有数据项对(t_i, t_j)之间相似度的均值，其中，$t_i \in C_p$, $t_j \in C_q$，而 2 个数据项 t_i 与 t_j 之间的相似度根据式(4)计算，即等于规则 $t_i \Rightarrow t_j$ 与规则 $t_j \Rightarrow t_i$ 的信任度均值。因此，类间相似度为

$$SMIC(C_p, C_q) = \frac{1}{n_p n_q} \sum_{t_i \in C_p, t_j \in C_q} Sim(t_i, t_j) = \frac{1}{n_p n_q} \sum_{t_i \in C_p, t_j \in C_q} \frac{conf(t_i \Rightarrow t_j) + conf(t_j \Rightarrow t_i)}{2} \tag{4}$$

其中，n_p, n_q 分别表示聚类 C_p 与 C_q 中数据项的个数。

对于 2 个聚类来说，它们的相似度越大，则这 2 个类的相关性越强，合并的可能性就越大。

初始的数据项聚类按照以上公式计算类内关联度与类间相似度后，若 2 个聚类的类间相似度大于各自的类内关联度，则合并这 2 个类，经过重复计算直到没有满足此条件的 2 个类为止。最终得到经过优化后的数据项聚类，根据这些数据项聚类将数据对象归类。为了将某个数据对象划分到适当的类中，定义以下的关联度评分函数，该函数用于确定一个数据对象应该属于哪个聚类。

定义 3 对于数据集的任一数据对象 O_i，该数据对象关于某个聚类 C_j 的关联度分值计算如下：

$$Score(O_i, C_j) = \frac{|O_i \cap C_j|}{|O_i|} \tag{5}$$

其中，$|O_i \cap C_j|$表示数据对象 O_i 与聚类 C_j 具有相同数据项的个数；$|O_i|$表示数据对象 O_i 中的数据项个数。通过该式将一个数据对象划分到关于它具有最高 *Score* 的聚类中，若一个数据对象关于多个聚类具有相同的 *Score* 值，则可以将其划分到其中的任一个类中。

3.3 CABMFI 算法过程

随着聚类的合并，聚类个数在减少，根据以上的分析，一个好的聚类应该具备以下条件：

(1)聚类内的数据项之间具有较强的关联度；

(2)该聚类与其他聚类具有较弱的相似度；

(3)该聚类应该是频繁的或者它的支持度与最小支持度的差值小于某个给定的阈值。

第(3)个条件显示了当具有较大相似度的聚类进行合并后，其支持度可能比最小支持度小，因此，指定一个阈值用于确定最终的聚类。综合以上分析，可以构造一个用于发现数据库中一组好的聚类的算法，步骤如下：

输入 数据库 D，最小支持度阈值(*minsup*)以及阈值 δ

输出 一些数据项的聚类以及数据对象的聚类

(1)挖掘出所有的频繁项集，删除有被包含关系的子集后得到所有的最大频繁项集。把每个最大频繁项集看成一个初始聚类。

(2)根据式(3)、式(4)计算各聚类的类内关联度 *Adegree* 与类间相似度 *SMIC*。

(3)优化聚类，该步骤的目的是合并相似度较大的聚类。如果 2 个聚类的类间相似度大于相应类的类内关联度，则将这些类作为候选要合并的类。选择相似度最大的 2 个类，如果合并后的类满足一个好的聚类应具备的第(3)个条件，则合并它们，并重新计算新类的类内关联度。若合并后的类包含原有的类，则删除原有的类。重复执行第(2)步与第(3)步，直到聚类的结果中同一个聚类内的数据项具有较高的关联度，而聚类间具有较低的相似度或者没有任何 2 个类间的相似度大于相应类内的关联度为止。

(4)对于每个数据对象，根据式(5)计算它的 *Score* 值，然后将其划分到关于它具有最高分值的聚类中，若一个数据对象关于多个聚类具有相同的 *Score* 值，则可以将其划分到其中的任一个类中。

4 实验

为验证 CABMFI 算法的有效性与正确性，将其与典型的 K-Means 算法进行实验对比，实验数据集来自 UCI 的机器学习数据库，从中选择 Zoo, Mushroom, Car Evaluation 3 个分类数据集，其中，Mushroom 数据集有 2 480 个关于蘑菇根茎的空属性值，实验前首先对这些空属性值按照出现最多的属性值进行填补。

算法的实现采用 Java 语言，运行环境为 Borland JBuilder 2006。实验在 Intel Pentium 4，256 MB 内存，运行 Windows

2000 Professional 的 PC 机上进行。

实验前把类别属性去掉，去掉类别属性的原因是假设事先不知道每个数据对象所属的类别，实验后以聚类的结果与原数据集给出的类别之间的差异作为聚类精确度的计算，实验中设定 δ=0.1。对于每个数据集，利用 CABMFI 在不同的最小支持度下分别做实验，以得到的聚类个数作为 K-Means 的输入，2 种算法的聚类精确度比较结果如表 1 所示。

表 1　CABMFI 与 K-Means 聚类精确度的比较

数据集	记录数	属性数	类别数	聚类个数	K-Means 精确度/(%)	CABMFI 精确度/(%)	CABMFI 最小上确界
Zoo	101	18	7	7	93.07	96.04	0.50
				5	90.10	94.06	0.60
				4	87.13	92.08	0.65
Mushroom	8 124	22	2	8	97.34	98.20	0.40
				5	93.33	94.88	0.50
				2	97.21	98.66	0.65
Car	1 728	6	4	5	91.78	92.25	0.20
				4	95.37	96.00	0.30

从表 1 可以看出，对于任一个数据集，随着最小支持度的增大，聚类的个数在减少，这是因为最小支持度越大，满足条件的频繁项集个数越小，最大频繁项集的数量也随之变少，最终使得聚类的个数也少。

另外，CABMFI 在不同聚类个数下的精确度都比 K-Means 高。实验结果表明，同一个聚类中的数据对象具有与该聚类相对应的数据项聚类的特征，聚类的精确度越高说明聚类的可解释性越强。

为验证算法的运行效率，对 2 个算法的运行时间进行比较，如图 1~图 3 所示，可以看出，CABMFI 算法的运行速度比 K-Means 快。

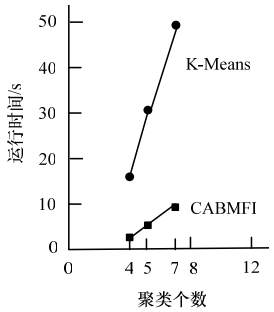

图 1　在 Zoo 上的运行时间比较

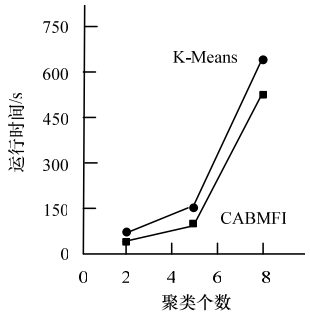

图 2　在 Mushroom 上的运行时间比较

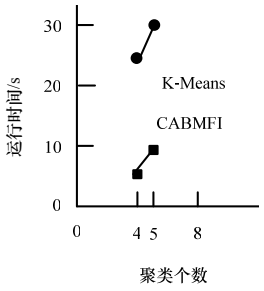

图 3　在 Car 上的运行时间比较

5　结束语

本文提出的基于最大频繁项集的聚类算法不需要输入聚类个数，利用数据项间隐含的关联进行聚类，适用于交易数据库与关系数据库。实验结果表明，与 K-Means 算法相比，该算法具有较高的运行效率与聚类精确度。

但存在的问题是在挖掘频繁项集时最小支持度如何确定，如果在给定值下不能挖掘到频繁项集，就无法进行聚类，因此，只能通过多次尝试。下一步工作将研究如何发现不同聚类中有趣的关联模式。

参考文献

[1] Ramakrishnan M. An Efficient Data Clustering Method for Very Large Databases[C]//Proc. of the ACM Int'l Conf. on Management of Data. [S. l.]: ACM Press, 1996.

[2] Brown C. A Practical Application of Simulated Annealing to Clustering[J]. Pattern Recognition, 1992, 25(5): 401-412.

[3] 朱　明. 数据挖掘[M]. 合肥: 中国科学技术大学出版社, 2002.

编辑　陈　文

（上接第 42 页）

树的分析中演化而来，它考虑了逻辑语句的复杂关系，所生成的测试用例集完全符合 MC/DC 覆盖的要求。该算法的应用可以帮助提高测试工具生成测试用例的针对性，不仅简化了测试用例的生成过程，而且提高了测试用例的覆盖率和效率，从而提高了软件测试的质量。

参考文献

[1] 张义德, 王国庆, 汤幼宁. 更改的判定条件覆盖测试技术研究[J]. 计算机工程与设计, 2003, 24(5): 19-26.

[2] NASA. A Practical Tutorial on Modified Condition/Decision Coverage[Z]. Washington D. C., USA: NASA, 2001: 7-9, 73-76.

[3] RTCA, Inc.. Software Considerations in Airborne Systems and Equipment Certification/DO-178B[Z]. Washington, USA: RTAC, 1992.

[4] Chilenski J J. Miller S P. Applicability of Modified Condition/ Decision Coverage to Software Testing[J]. Software Engineering Journal, 1994, 9(5): 193-200.

[5] Chilenski J J. An Investigation of Three Forms of the Modified Condition Decision Coverage(MCDC) Criterion[J]. Computer Programming and Software, 2001, 18(4): 214-219.

[6] 朱晓波, 杨伟民, 叶　芯. 更改条件/判定最小真值表生成算法及其应用[J]. 上海理工大学学报, 2007, 29(1): 84-88.

[7] 陈　鑫, 杨　平. 应用 MC/DC 准则时需考虑的问题及其改进方法[J]. 计算机工程与设计, 2004, 25(3): 406-410.

编辑　张正兴