自适应数据库中基于特征向量的聚类算法

强 彦,陈俊杰,高燕飞
QIANG Yan,CHEN Jun-jie,GAO Yan-fei

太原理工大学 计算机与软件学院,太原 030024
College of Computer Engineering and Software,Taiyuan University of Technology,Taiyuan 030024,China

QIANG Yan,CHEN Jun-jie,GAO Yan-fei.Research of cluster based on feature vectors in autonomic database.Computer Engineering and Applications,2008,44(27):162-164.

Abstract: In autonomic database system,workload characterization is a key part.In workload characterization,workload should be classified,then anticipate workload performance.Workload classification uses cluster algorithm.And in cluster algorithm,the typical is the *K*-means cluster algorithm.But in the *K*-means cluster algorithm,*k* should be defined and not changed.This paper makes an improvement in the algorithm,so the *k* is changed if needed.The result of the experiment shows that the veracity using Cluster algorithm Based on Feature Vectors(CFV) making of forecasting workload runtime is improved.

Key words: feature vectors;cluster;Cluster Based on Feature Vectors(CFV)

摘 要: 负载自适应数据库系统中,负载特征化部件要实时对各种数据库的访问负载分类,根据分类的情况预测负载对数据库资源需求。是对常规聚类算法的一个改进,提出基于特征向量的聚类算法和基于特征向量的增量聚类算法。使用该算法后负载分类速度和准确性有明显提高。

关键词: 特征向量;聚类算法;基于特征向量的聚类算法
DOI: 10.3778/j.issn.1002-8331.2008.27.052 **文章编号:** 1002-8331(2008)27-0162-03 **文献标识码:** A **中图分类号:** TP311.13

1 引言

数据库的性能调整和优化一直是数据库系统解决不断增长的数据量和访问量问题的办法,通过增加硬件设备和调整数据的存储和内存分配来满足数据库的访问需求。自适应数据库系统可以从新的角度来解决这一问题。在数据库资源条件一定的条件下,通过区分不同的访问需求,数据库系统智能决定先响应哪些访问,再响应哪些访问,动态合理地规划系统资源来更合理地满足对数据库系统的访问需求。这是数据库管理系统向自我管理方向的发展[1]。在这样的思路下,可以采用数据库系统负载自适应的基本框架模型[2]。模型如图 1 所示。

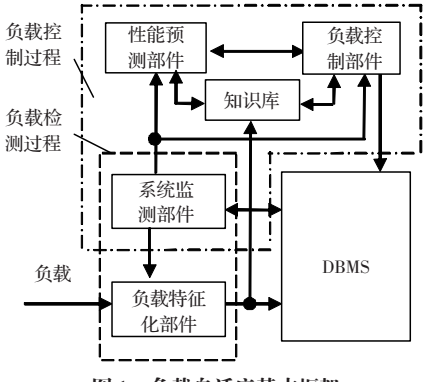

图 1 负载自适应基本框架

在该模型中,系统能够根据访问负载的一些特性自动的完成负载的性能预测、控制等功能。其中,负载特征化部件根据负载的特征向量分类;性能预测部件通过分类的结果预测负载的资源需求(包括 CPU 消耗,I/O 消耗等等);负载控制部件确定负载控制方案以满足负载的性能要求;系统监测部件监视系统的运行,获取负载运行的性能参数和系统资源的使用情况。在负载特征化部件中,客户端输入的负载根据负载的特征向量分类,然后输入到性能预测部件和知识库中,性能预测部件根据分类的情况和知识库预测负载的资源需求,知识库中保存分类的结果并记录每条负载的运行情况,当系统闲时,可以根据知识库的记录重新分类。负载特征化部件可采用聚类算法,聚类就是把一个没有类别标记的样本集按某种规则划分成若干类[3],使类内样本的相似性尽可能大,而类间样本相似性尽量小,是一种无监督的分类方法。常用聚类算法有[4]:*K*-means 算法[5]、STING 算法、CLIQUE 算法和 CURE 算法等等。其中的 *K*-means 算法,对处理大数据集,具有良好的可伸缩性和高效率,时间复杂度与对象的数目有线性关系;其理论上可靠、算法简单、收敛速度快、能有效地处理大数据集而被广泛使用[6]。但 *K*-means 算法的缺点是在算法运行前就必须确定 *K* 值,而在现实情况下,在数据库对方问负载进行分类前,无法确定数据库的访问负载可以分为几类,所以需要改进算法。

基金项目: 国家自然科学基金(the National Natural Science Foundation of China under Grant No.60773004);山西省自然科学基金(the Natural Science Foundation of Shanxi Province of China under Grant No.2007011050)。

作者简介: 强彦(1969-),男,博士,主要研究方向:自适应数据库。

收稿日期: 2007-11-08 **修回日期:** 2008-02-27

2　基于特征向量的聚类算法

2.1　选取特征向量

数据库的特征向量是数据库负载的显式特征，这些特征可能与负载的运行时间有关，包括用户名、表名、索引、SQL 语句中的命令等等。基于特征向量的聚类算法 CFV(Cluster based on Feature Vectors)要从这些特征向量中动态选取最具性能影响的特征向量组合，并且得出按照这些组合能把数据库的实时访问负载归类。

首先选取一组特征向量，为阐述方便，此处假设根据特征向量把负载分为 4 类，记为 S_1、S_2、S_3、S_4。

2.2　运行基本聚类算法

把这 4 类 S_1、S_2、S_3、S_4 当作是算法初始的分类；然后运行基本聚类算法，如 K-means 算法，最后还是得到 4 类，记为 L_1、L_2、L_3、L_4，但这 4 类中所包含的负载已经发生变化，算法描述如下：

(1)初始的分类记为 S_1、S_2、S_3、S_4，K=4、I=1；　// I 表示算法中迭代的次数

(2)计算初始聚类的中心 $Z_m(I)$，m=1，2，3，4；//在这里计算每一类负载的平均运行时间

$$Z_m(I)=\frac{1}{n_m}\sum_{j=1}^{n_m}Sm_j \quad j=1,2,\cdots,n_m \tag{1}$$

其中，n_m 表示第 m 类中包含负载的个数；

(3)分别计算每条负载与这 4 个聚类中心的距离(即：每个负载运行时间与每类平均运行时间的差值的绝对值)$D(X_i, Z_m(I))$，把负载归到离聚类中心距离最近的类中；

$$D(X_i,Z_m)=|X_i-Z_m| \quad i=1,2,\cdots,n,m=1,2,3,4 \tag{2}$$

(4)重复执行第(2)和第(3)步，直到聚类中心不再改变，即 $Z_m(I+1)=Z_m(I)$，把新的分类记为 L_1、L_2、L_3、L_4。

2.3　计算概率值

初始的分类数和引入的特征向量是需要调整的，通过计算重新聚类后负载在新类中的概率分布，可以作为调整聚类个数的依据。算法如下：根据分类的个数，建立一个 $K\times K$ 的矩阵，在这里是建立一个 4×4 的矩阵，如下式所示：

$$\begin{bmatrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{21} & P_{22} & P_{23} & P_{24} \\ P_{31} & P_{32} & P_{33} & P_{34} \\ P_{41} & P_{42} & P_{43} & P_{44} \end{bmatrix}$$

$$P_{ij}=\frac{S_i\text{中负载在}L_i\text{类里的个数}}{S_i\text{中负载的个数}} \tag{3}$$

P_{ij} 表示根据特征向量分类预测负载运行时间的准确性；P_{ij} 越大，说明预测出的准确性越大。所以，选取每一行最大的点 P_{ij}，如果 $P_{ij}\geqslant t$(t 为收敛半径，可作为算法结束的条件)，且所有选取的 P_{ij} 中 i 值不等，说明根据特征向量分类预测负载运行时间是准确的，分类完成。如果上述条件不满足，则说明根据特征向量分类预测负载运行时间的准确性不够，也就是说特征向量的选取是不合适的，就需要增加特征向量。例如假设，现有 100 个负载作为初始数据，根据负载特征分为 4 类，取 t>60%，运行基本算法后，建立矩阵，如下式：

$$\begin{bmatrix} 100\% & 0\% & 0\% & 0\% \\ 17\% & 50\% & 33\% & 0\% \\ 0\% & 45\% & 55\% & 0\% \\ 25\% & 0\% & 0\% & 75\% \end{bmatrix}$$

从上式中可以看出，第 2 行和第 3 行的最大值都没有大于 t，所以要增加特征向量，在这里只需要把第 2 行和第 3 行对应的最初 S_2 和 S_3 类加入新的特征向量作为分类组合就可以了；第 1 行和第 4 行的概率最大值都大于 t 值说明 S_1 和 S_4 分类已经达到要求，保持原有的 S_1 和 S_4 类。

2.4　增加特征向量

从待选的特征向量中，按顺序选取新特征向量，调整 K 值。接着运行基本聚类算法，计算概率值。选取每一行概率值最大的点 P_{ij}，如果 $P_{ij}\geqslant t$，且所有选取的 P_{ij} 中 i 值不等，说明根据特征向量分类预测 SQL 语句运行时间是比较准确，分类完成。如果上述条件不满足，就需要增加特征向量，重复上述算法。

2.5　减少特征向量

当分类完成后，要优化类的个数，也就是要计算每一个类中心(Z_m)与其它类中心的距离。如果距离小于 W(W 为收敛值，作为合并门限条件)，则合并这两个类；重复计算，直到类的个数不变，输出分类结果。

3　基于特征向量的增量聚类算法

CFV 算法基本完成了初始负载的分类，其优点是分类后使类内数据的聚合度比较高，缺点是该算法时间消耗比较大。如果能在前期聚类的基础上，仅对新增负载进行增量式聚类，则是非常有意义的。在自适应数据库系统运行的开始，采用 CFV 算法作为前期积累，随后运行基于特征向量的增量聚类算法 ICFV(Incremental Cluster based on Feature Vectors)。

具体步骤如下：

(1)提取新输入负载的特征向量。

(2)根据提取的特征向量预测新增负载应该归属于 CFV 算法结果中的哪一类，提取对应类的中心点，即类内负载平均运行时间，把该时间作为新增负载的运行时间，类的中心计算公式如下所示：

$$Z_m=\frac{1}{n_m}\sum_{j=1}^{n_m}Sm_j,\ m=1,2,3,4,j=1,2,\cdots,n_m \tag{4}$$

其中，Z_m 表示第 m 类的中心点；n_m 表示第 m 类中包含负载的个数。

(3)把该时间输入给性能预测部件并让新增负载运行，当负载运行完后，通过系统监视部件反馈回来的新增负载运行的实际时间，计算新增负载实际运行时间与每一类中心点的距离 $D(X_i,Z_m)$：

$$D(X_i,Z_m)=|X_i-Z_m|,i=1,2,\cdots,n;m=1,2,3,4 \tag{5}$$

(4)把新增负载归入到离中心点距离最近的类里，并重新计算该类的中心点。

(5)如果新增负载归入的类和在第(2)步中预测的类不同，则让计数器 K 加 1。

(6)如果 $K\geqslant X$(X 为重新启动 CFV 的门限值)时，要重新运行 CFV 算法；否则，算法结束。

4 实验结果和分析

根据上述算法,编制了一个自适应数据库中基于特征向量的聚类算法和增量聚类算法程序,并设计 100 个数据库负载进行实验。实验结果如表 1 所示。

表 1 运行结果表

按特征向量的分类	特征向量	每行最大的 P_{ij} 值/(%)	CFV 分类	特征向量	每行最大的 P_{ij} 值/(%)
S_1	Select	100	L_1	Select,表 1 或表 2	75
S_2	Insert	50	L_2	Insert,表 1	67
S_3	Update	55	L_3	Insert,表 2	100
S_4	Delete	75	L_4	Delete,表 1 或表 2	75
			L_5	Update,表 1	67
			L_6	Update,表 2	67

通过表 1 可以看出,初始的 100 负载根据 SQL 语句数据操纵语句关键词为特征向量分组,运行 CFV 算法后,每行最大的 P_{ij} 值达不到设定的值,经过 CFV 算法调整特征向量 K 值后,每行最大的 P_{ij} 值达到设定的值,即表示负载的聚类准确度达到了预定值,说明负载分类明显,特征向量选取准确。

从表 2 可以看出,在 CFV 算法初始聚类的基础上,ICFV 算法的准确率在稍高一些的同时,最主要的是 ICFV 算法比 CFV 算法的运行时间减少很多,从而,使得在自适应数据库中可以在实时运行的条件下,达到对数据库负载分类的时间要求。

5 结论

本文提出自适应数据库中负载的分类算法,在提高聚类准确度的同时,采用增量聚类算法减少聚类的运行时间,达到实时聚类的要求。通过实验验证,使用上述基于特征向量的聚类和增量聚类方法分类后,类内数据的聚合度比较高,通过聚类的结果预测数据库系统负载的运行时间比较准确,ICFV 算法时间消耗明显减少,使整个自适应数据库系统负载特征化部件满足负载实时分类要求,从而使性能预测部件能通过分类的结果预测负载的资源需求,如 CPU 消耗、运行时间和 I/O 消耗等等,最终通过负载控制部件动态合理地规划系统资源来更合理地使数据库系统满足外部访问需求。

表 2 算法对比表

选取的算法	准确率/(%)	运行时间/s
CFV 算法	90	530
ICFV 算法	93	83

参考文献:

[1] 胡天磊.自治数据库系统的理论与方法研究[D].杭州:浙江大学,2006.

[2] Niu B,Martin P,Powley W,et al.Workload adaptation in autonomic DBMSs[C]//Proceedings of CASCON 2006,Toronto,Canada,2006:161-173.

[3] 栾丽华.聚类算法研究[D].南京:南京师范大学,2004.

[4] 刘振岩.数据挖掘分类算法的研究与应用[D].北京:首都师范大学,2003.

[5] 张建萍,刘希玉.基于聚类分析的 K-means 算法研究及应用[J].计算机应用研究,2007,24(5).

[6] 袁方,周志勇,宋鑫.初始聚类中心优化的 K-means 算法[J].计算机工程,2007,33(3).

(上接 158 页)

设 dictionary 是所有训练样本的不同单词的集合,对于某一文本 d,对 d 中的每个在 dictionary 中出现的单词 w,分别计算:

$$p(w|E)=\frac{n_w+1}{n+|dictionary|}$$

$$p(w|\overline{E})=\frac{1}{|dictionary|}$$

其中,E 表示的是训练样本类,n 是所有训练样本的总词数(包括重复的),n_w 是同一词 w 在所有训练样本 E 中出现的总次数,然后同一测试样本计算 $p(d|E)=\prod p(w|E)$,$p(d|\overline{E})=\prod p(w|\overline{E})$,最后通过两个值的大小来判断所在的类别。

所有需要的中文专业词汇都在测试前手工加入了字典,选定关键词个数 m=25,两种算法都只经过分词而没有进行处理停词。

从上述的实验和分析得到的结论有:

(1)ILODC 拥有同大部分已有算法相似的分类效果,甚至在中文的实验的反例测试中 ILODC 要远优于 NB。可能的原因是 NB 算法对停词有较强的依赖性,而改进算法对停词的依赖小很多。

(2)ILODC 和已有算法相比,更适用于增量学习。

5 结语

本文提出了一个支持增量学习的文本单类别分类算法 ILODC。由分析和实验的结果表明,与已有算法相比,在不降低分类效果的同时,ILODC 算法有效地降低了增量学习所需时的运算量,比较适合用于需要增量学习的环境。

ILODC 的关键是训练样本集合中不能有噪声数据,去噪以及对关键词集合 S_{key} 的大小 m 的选择仍需要进一步研究。

参考文献:

[1] Manevitz L,Yousef M.One-class SVM for document classification[J].Journal of Machine Learning Research,2001.

[2] Manevitz L,Yousef M.One-class document classification via neural networks[J].Neurocomputing,2007.

[3] Wang Ke,Stolfo S J.One-class training for masquerade detection[C]//3rd IEEE International Conference on Data Mining,2003.

[4] Onoda T,Murata H,Yamada S.One class classification methods based non-relevance feedback document retrieval[C]//International Conference on Web Intelligence and Interlligent Agent Technology,2006.

[5] Wang D F,Yeung D S,Tsang E C C.Structured one-class classification[J].Systems,Man and Cybernetics,2006.

[6] Reuters-21578 test collections[S/OL].(1997).http://www.daviddlewis.com/resources/testcollections/reuters21578.