文章编号:1001 - 9081(2007)07 - 1695 - 04

一种改进的密度偏差抽样算法

张建锦[1],吴 渝[1],刘小霞[2]
(1. 重庆邮电大学 计算机科学与技术研究所,重庆 400065;
2. 北京邮电大学 计算机科学与技术学院,北京 100876)
(zhangjj616@ gmail. com)

摘 要:随机抽样技术已经广泛应用于数据挖掘的各类算法中,它在处理分布均匀的数据集时非常有效,但在处理分布比较倾斜的数据集时容易丢失小的聚类。为此提出基于网格的密度偏差抽样算法,仅需要扫描一遍数据集就可以得到近似的密度偏差抽样。经实验测试分析表明,该算法不仅提高了聚类的正确性,而且抗噪声能力强、效率高,是解决海量数据挖掘的一种有效途径。

关键词:数据挖掘;偏差抽样;聚类;数据约简;海量数据
中图分类号: TP274 **文献标志码**:A

Improved density biased sampling algorithm

ZHANG Jian-jin[1], WU Yu[1], LIU Xiao-xia[2]
(1. *Institute of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing* 400065, *China*;
2. *School of Computer Science and Technology, Beijing University of Posts and Telecommunications, Beijing* 100876, *China*)

Abstract: Uniform random sampling is widely applied to many kinds of algorithms in data mining. It processes uniform distribution data set extremely effectively, but easily loses slight cluster and consequently decreases clustering accuracy, when the processing data set is skew distribution. A grid-based density biased sampling algorithm (G_DBS) was proposed. It got approximate density biased samples through scanning data only one time. Our experimental evaluation shows that G_DBS algorithm not only improves the accuracy of clustering, but also is insensitive to noise and has high efficiency. It is one of the effective solutions to mass data mining.

Key words: data mining; biased sampling; clustering; data reduction; mass data

0 引言

数据挖掘研究自兴起以来一直是学术研究的热点问题,现在已经形成大量的经典算法。随着网络技术的飞速发展,以及数据库技术的进步,如:多媒体数据库、空间信息系统,传感技术以及空间数据库,使得数据挖掘需要处理的数据规模越来越大,动辄以 GB,甚至 TB 计。经典数据挖掘算法处理海量数据时,需要消耗大量的时间和空间资源,并且挖掘效果也不理想,甚至内存消耗殆尽,导致算法崩溃,无法进行挖掘处理。

在面向海量数据集的处理时,现在采用的主要方法有数据约简、分布式并行处理、批处理、量式处理等。基于统计学的抽样方法是对部分数据进行分析,然后推测原始总体数据集的相关信息,无疑是实现数据约简的成功方法之一。随机抽样的方法已经广泛应用于数据挖掘的各种算法中,该方法在抽取数据时对每个数据以相等的概率进行抽取。当数据服从均匀分布时,随机抽样不仅能获得高质量样本,而且实现简单、运行效率高。但是,大量自然现象服从基夫(Zipf)分布,例如人口的分布:大城市的人口分布不仅数量多且密度大,而偏远小城镇的人口分布数量少且密度小;如果采用随机抽样,每个数据点都以相同的概率被包含在样本中的话,那么密度小而数量少的偏远地区抽取的数据点非常少,甚至丢失这些区域。在这种情况下,若采用密度偏差抽样(Density Biased Sampling,DBS)策略[1],它根据密度进行偏差抽样,可以增加偏远地区的数据点的抽取概率,则获得的样本保持了原始数据集的分布特征,提高了样本的质量,得以在对样本的聚类操作中找到偏远地区的稀疏类,使聚类结果更加优化。

鉴于 DBS 策略采用哈希表存储数据,容易产生哈希冲突,并且对噪声数据敏感,本文提出了基于网格的密度偏差抽样算法。本算法将数据划分为网格,对每个单元格进行样本抽样,并采用缓冲技术存储,有效地避免了哈希冲突。实验表明,该算法可以获得理想的样本,提高聚类的正确性,算法鲁棒性好,是解决海量数据挖掘的一种有效途径。

1 密度偏差抽样

随机抽样策略是指以相同的概率抽取数据集中的每一个数据,因此有可能丢失某些重要的记录,导致抽样样本不能保持原始数据集的特征。此外,随机抽样还对噪声敏感。在基于随机抽样的聚类中,特别是数据集偏斜、含有噪声的情况

收稿日期:2007 - 01 - 03;**修回日期**:2007 - 03 - 05。 **基金项目**:重庆市自然科学基金资助项目(2005BB2063);重庆市自然科学基金重点项目(2005BA2003);重庆市教委科学技术研究项目(050509)。

作者简介:张建锦(1983 -),男,山西孝义人,硕士研究生,主要研究方向:数据挖掘、人工智能; 吴渝(1970 -),女,重庆人,教授,博士,主要研究方向:数据挖掘、小波分析、多媒体技术; 刘小霞(1982 -),女,山西忻州人,硕士研究生,主要研究方向:数据挖掘、信息安全。

下,有可能丢失小的聚类(如图1),影响聚类的效果。

密度偏差抽样是数据挖掘中一种新的抽样策略[1],每个数据点根据其局部密度来确定抽样概率,通过数据分布密度情况来生成样本。该样本保持了原始数据集的分布情况及数据特征,抗噪声的能力强。在基于密度偏差抽样的聚类中,不丢失小的聚类,能够正确生成聚类。

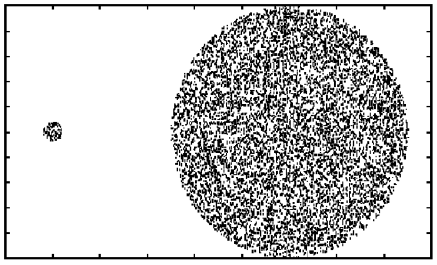

图1 偏斜数据集

如图1中所示两个簇,右边较大的簇由50000个点组成,左边较小的簇由1000个点组成。但是左边簇的密度是右边簇密度的两倍。假设要随机抽取的样本是1%,也就是选取510个点,那么在左边的簇中点被抽取的概率为1000/51000,仅仅9个点可能被选中。在这个样本上聚类,则将这9个点忽略或者视为孤立点。而用DBS来抽样,左边簇中点的抽取概率将大大提升,其中87个点能被抽取,进而形成正确的样本,最终得到理想的聚类结果。

2 基于网格的密度偏差抽样策略

密度偏差抽样方法的关键就是密度的获取。本文提出了基于网格的密度偏差抽样方法(Grid DBS,简称 G_DBS),可以获得理想的样本,有效实现数据约简。G_DBS 实现方法是:首先根据数据的属性进行划分,构造网格,然后采用缓冲管理技术,在内存中开辟大小为原始数据集20%的空间,对划分的网格,用数组实现映射存储。在完成一个网格抽样操作之后,及时地输出样本结果,进而释放占用的内存空间,实现内存空间的动态分配。这样,与文献[1]中的哈希表存储方式相比,避免了哈希冲突,提高了对噪声数据的鲁棒性。

2.1 网格概念

一个 d- 维数据集,其属性$(A_1, A_2, \cdots, A_d)$都是有界的,设第 i 维上的值在区间$[l_i, h_i]$中,$i = 1, 2, \cdots, d$,则 $S = [l_1, h_1] \times [l_2, h_2] \times \cdots \times [l_d, h_d]$就是 d- 维数据空间。

定义1 网格单元。将每一维分成 k 个长度相等、不相交的区间的段,每个区间都是左闭右开的等长的区间,这样将数据空间被分割成 k^d 个网格单元。网格单元在第 i 维上的长度为 $\theta_i = (h_i - l_i)/k$,第 i 维上的第 j 个区间段可由 $I_{ij} = (l_i + (j-1) * \theta_i, l_j + j * \theta_i)$, $j = l, 2, \cdots, k$ 得出。

定义2 网格相邻单元。一个网格单元的相邻单元是那些与该单元有相邻边界的单元或有相邻点的单元。每个单元的邻居数为 2^{d+1}(处于数据空间边界的单元除外,特例 $d = 1$ 时,邻居数为2)。

2.2 G_DBS 的实现

在实现 G_DBS 算法时,为了获得数据分布密度,首先需要完整扫描一遍总体数据集。给定 d 维数据集,数据量 N,抽样后的数据量为 n ($N \geqslant n$)。将数据空间量化为有限数目的单元格,构造形成 G 个单元格的网格。在构造网格的过程中,记录在每个单元格中数据的统计信息,将每个单元格内的数据的个数作为数据分布的密度,记为单元格〈grid_id, density〉。

设第 i 个单元格 G_i 的大小为 n_i,依据每个单元格的大小定义与其相应的权重, G_i 中$\{x_1, x_2, \cdots, x_{n_i}\}$,带权重 w_j 的点 x_j 包含在样本中的概率为 $P(x_j)$,则:

$$\sum_{j=1}^{n_i} w_j P(x_j) = kn_i \tag{1}$$

其中 k 是常量。这样就保持了数据的分布密度情况。如果 $k = p$,$p = n/N$, n 为样本数量,则得到随机抽样。

每一个单元格内点 x 的抽样概率是相等的,我们定义 $P(x \mid x \in G_i) = f(n_i)$,在同一个单元格内的点带有相等的权重: $w(n_i) = 1/f(n_i)$。那么,在单元格 G_i 中:

$$\sum_{x}^{n_i} P(X_i) * w(n_i) = \sum_{x}^{n_i} f(n_i) * 1/f(n_i) = n_i \tag{2}$$

基于以上条件,定义每个单元格中点的抽样概率函数:

$$f(n_i) = a/n_i^e (0 \leqslant e \leqslant 1) \tag{3}$$

其中 e 为常量, $0 \leqslant e \leqslant 1$。当 $e = 0$,偏差抽样就降为随机抽样;当 $e = 1$,在每个单元格内将抽取相同数量的数据点。

样本量的期望就是每个单元格抽取样本期望的总和,因此样本量:

$$n = \sum_{i=1}^{G} n_i f(n_i) = \sum_{i=1}^{G} n_i a/n_i^e \tag{4}$$

a 决定生成样本数量为 n,,从而推出:

$$a = n/\sum_{i=1}^{G} n_i^{1-e} \tag{5}$$

则每个单元格中点的抽样概率为:

$$f(n_x) = n/\left(n_x^e \sum_{i=1}^{G} n_i^{1-e}\right) \tag{6}$$

如图2所示,将数据空间划分为等分的网格,通过开辟内存临时存储空间保存对应数据。当数据动态更新,由于式(4)是一个减函数,如果单元格内的数据有所增加,则抽样的概率减小而抽样的权值将增大,于是用新的更小概率替换原始数据,新数据抽样概率增大,从而保证了样本质量。

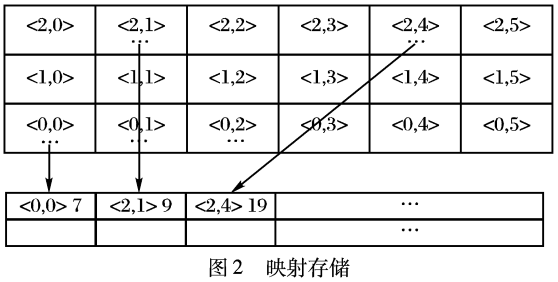

图2 映射存储

3 基于网格的密度偏差抽样算法及分析

算法 G_DBS 描述如下:

输入 初始数据集 N,抽样样本量 n,参数 e,属性区间 M;

输出 获得数据子集 S

1) 初始化数据缓冲区存储空间,设置大小约为数据集的20%;

2) 扫描数据集,将其根据属性区间量化为空间网格,数

量为 G,记录(grid_id, n_i);

3) 根据式(4),得到 a;

4) 考查单元格 G_i,如果 $i \leqslant G$,执行 5);否则转 7);

5) 如果 $n_i \neq 0$,计算式(5),执行6);否则 i ++,返回执行 4);

6) 对单元格 G_i 执行随机抽样;获得 $n_i \cdot f(n_i)$ 个样本点,释放内存空间;i ++,返回执行 4);

7) 输出结果数据。

G_DBS 初始阶段对数据集构建网格结构,如果属性区间过小,处理的代价会明显增加;如果属性区间过大,将会降低偏差抽样的效果,影响样本质量。上文指出常量参数 e, $0 \leqslant e \leqslant 1$,当 $e = 0$,偏差抽样就降为随机抽样;当 $e = 1$,在每个单元格内将抽取相同数量的数据点。在实验过程中,我们对 e 取值 0.5,进行结果比较分析。

4　实验测试

我们使用环境为 VC6.0 在 Windows XP 的服务器(CPU: PIV 2.3G,Memory:512M)上构建算法测试平台。为了测试算法的正确性,更好地控制数据,选择人工数据集 DataSet[7] 和 UCI 数据库中的真实数据集 Forest CoverType[8] 对本文提出的算法的正确性和执行时间进行测试。其中,DataSet 采用文献[7]中的数据集,如图 3 中(a)所示,包含 100000 个数据,5 个类,一个大的球形类,两个小的球形类,两个椭圆形类,类的大小不均衡,并且含有 20% 噪声数据。

实验进行三方面比较,一是 G_DBS 与 DBS[1]、RS (Random Sampling)抽样效果的比较,即样本质量对比;二是在各种抽样策略样本上进行的正确聚类效果的比较;三是在 G_DBS 样本上的聚类与在原始数据集上直接聚类两者所需时间的比较。

4.1　抽样样本质量分析

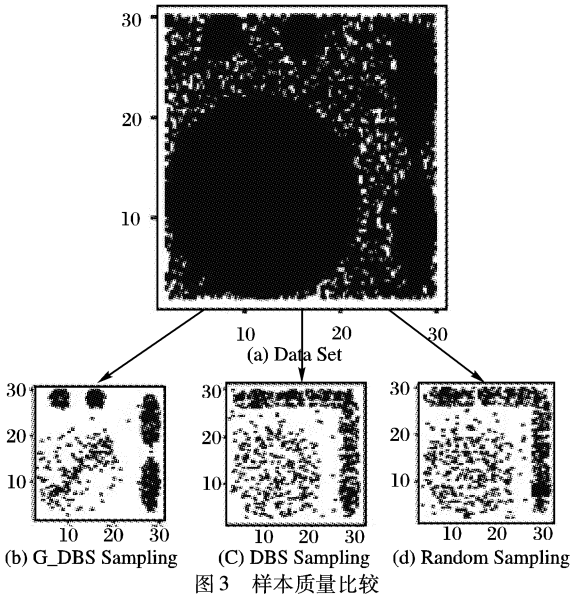

图 3　样本质量比较

对数据集 DataSet 进行抽样 1% 样本的结果如图 3 所示,(b)显示结果为采用 G_DBS 算法抽样,显然样本数据保持了原始数据集的分布特征;(c)为采用 DBS 算法抽样后的结果;(d)为 RS 抽样的结果。不难发现,在密度分布不均匀、含有噪声的数据集中,采用 G_DBS 抽样后的结果,样本质量高,抗噪声;DBS 受噪声数据影响很大,并且在数据量较大的情况下,由于哈希冲突,造成样本质量低下;RS 将 4 个较小的类混合为 1 个类,可见 RS 在数据分布不均匀、含噪声的情况下,不能获得质量满意的样本。

4.2　基于抽样样本的聚类效果分析

在对聚类的效果分析中,我们主要测试识别聚类的正确个数 NC(the number of correctly found clusters)。在这里我们采用分层聚类算法。分别对每个数据集进行不同比例的抽样,在样本上进行聚类 NC 测试,对数据集 DataSet 的测试如表 1 所示。

表 1　对 DataSet 聚类 NC 测试结果

Sampling Size(%)	正确结果	G_DBS	DBS	RS
0.5	5	4	2	3
1	5	5	2	4
1.5	5	5	3	4
2	5	5	4	4
3	5	5	5	4
4	5	5	5	5

实验数据表明:G_DBS 与 DBS 和 RS 相比,只需要少量的样本就可以得到正确的聚类,当样本为 1% 时,G_DBS 就可以得到 100% 正确聚类,而 DBS 需要 3%,RS 需要 4%。在样本量小于 2% 时,DBS 的聚类效果不及 RS,原因是在原始数据集中包含 20% 的噪声数据,DBS 不能有效区分噪声数据和聚类数据,这是导致 DBS 性能低下的主要原因。

数据集 Forest CoverType 中含有 581 012 条数据,7 类树木,类别以及相应的类的数据量是 Spruce-Fir: 211 840, Lodgepole Pine: 283 301, Ponderosa Pine: 35 754, Cottonwood/Willow: 2 747, Aspen: 9 493, Douglas-fir: 17 367, Krummholz: 20510。数据集中不同类间数据偏斜很大。例如:Lodgepole Pine 有 283 301 个数据点,Cottonwood/Willow 只有 2 747 个数据点。对数据集 Forest CoverType 的测试 NC 如表 2 所示。

表 2　对 Forest CoverType 聚类 NC 测试结果

Sampling Size(%)	正确结果	G_DBS	DBS	RS
0.5	7	5	4	4
1	7	6	5	4
1.5	7	7	6	4
2	7	7	7	5
3	7	7	6	6
4	7	7	7	7

从实验中可以看出,在分布不均匀、偏斜很大的大规模数据集中,G_DBS 依然有高效的正确聚类。当样本比例为 1.5% 时,就可以得到正确结果,比 DBS、RS 获得更好的识别结果。图 4 给出了数据集 Forest CoverType 的 NC 测试结果。

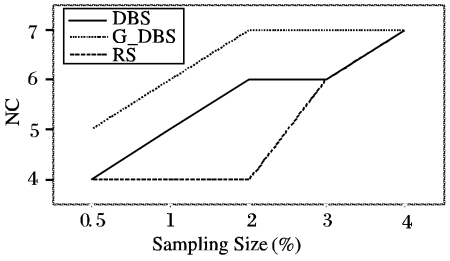

图 4　三种策略 NC 测试结果

4.3 G_DBS 与 RS、DBS 算法执行时间分析

实验用三维的人工数据集,数据量逐渐递增。测试 G_DBS 与 RS、DBS 算法,抽取 1.5% 的样本所需要的运行时间如图 5 所示。

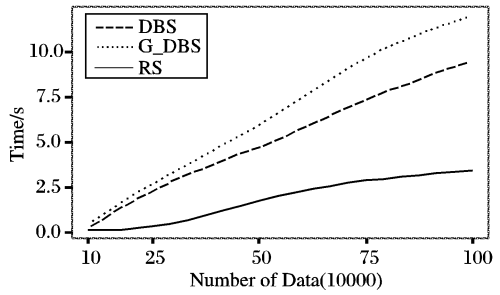

图 5 时间测试

实验表明:在同一数据集上,获取等量样本,RS 执行时间最少,G_DBS 执行时间比 DBS 略长,但基于 G_DBS 聚类的正确性,远远超过基于 DBS 或 RS 的聚类效果,充分体现了本文方法的高效性。

5 结语

本文提出了基于网格的密度偏差抽样算法,该方法适用于海量数据的聚类应用。首先介绍了密度偏差抽样策略的基本理论,然后经实验将 G_DBS 与 C. Palmer 提出的 DBS 以及随机抽样的效果进行对比分析,并在相应的样本上进行聚类的正确性测试。实验表明,我们提出的 G_DBS 在分布不均匀并且在含有噪声的海量数据挖掘中,具有鲁棒性,抽样样本质量高。不难发现,在海量数据挖掘的数据约简策略上,G_DBS 有良好的表现,并且可以应用于数据挖掘的不同领域,例如关联规则、分类等。

参考文献:

[1] PALMER C , FALOUTSOS C . Density biased sampling : an improved method for data mining and clustering[C]// Proceedings of 2000 ACM SIGMOD International Conference on Management of Data. Dallas, USA: ACM Press, 2000: 82 - 92.

[2] KERDPRASOP K, KERDPRASOP N, SATTAYATHAM P. Density biased clustering based on reservoir sampling[C]// Proceedings of the Sixteenth International Workshop on Database and Expert Systems Applications. Copenhagen, Denmark: [s. n.]. 2005: 1122 - 1126.

[3] TOIVONEN H. Sampling large databases for association rules[C]// Proceedings of the 22th International Conference on Very Large Databases (VLDB'96). Bombay, India: Morgan Kaufmann, 1996: 134 - 145.

[4] CHEN B, HAAS P, SCHEUERMANN P. A new two-phase sampling based algorithm for discovering association rules[C]// Proceedings of 2002 ACM SIGKDD international conference on knowledge discovery and data mining. Edmonton, Alberta, Canada: ACM Press, 2002: 462 - 468.

[5] HAN J, KAMBER M. 数据挖掘概念与技术[M]. 范明, 孟小峰, 等译. 北京: 机械工业出版社, 2001: 8.

[6] KOLLIOS G, GUMOPULOS D, KOUDAS N, *et al.* Efficient biased sampling for approximate clustering and outlier detection in large datasets[J]. IEEE Transactions on Knowledge and Data Engineering, 2003, 15(5): 1170 - 1187.

[7] GUHA S, RASTOGI R, SHIM K. CURE: An efficient clustering algorithm for large databases[C]// Proceedings of 1998 ACM SIGMOD International Conference on Management of Data. Seattle, USA: ACM Press, 1998: 73 - 84.

[8] [EB/OL]. [2006 - 12 - 31] http://kdd. ics. uci. edu/databases/covertype/covertype. html.

(上接第 1694 页)

的实用性。2)合并邻近簇采用统一的计算口径,取消基于聚类特征的数据点个数判别来分类(支)执行合并簇的方式,因此提高了算法的执行效率。

HCAP 算法增加了步骤三的凝聚层次聚类,并用相似度阈值 f 的聚类约束倍数 t 来约束数据空间的聚类,对步骤二生成的子簇进一步合并求精,通过这一步,能得到较高的聚类质量并最大限度发现各种复杂数据特性的簇。

综合四个阶段,HCAP 算法总的时间复杂度为 $O(n^2)$,这已包含数据预处理及相异度矩阵的计算时间,这比凝聚层次聚类算法的 $O(n^3)$ 效率要高,但比 K 均值算法的 $O(n)$ 更花费时间。由于有更高的聚类质量,因而还是值得的。

6 结语

实验表明,HCAP 算法是合理有效、可靠和快速的。从算法设计角度分析,该算法充分利用了 K 均值算法和层次凝聚算法各自的优点,并在一定程度上避免了这两种算法的缺陷。与 K 均值算法和层次凝聚算法比较,HCAP 算法有如下优点:1)解决了 K 均值算法中初始簇数 k 的选择难题;2)K 均值算法中初始聚点的选择也是一个问题,原来的随机选择会造成每次聚类结果的不一致并降低了聚类质量,现在采用自动生成初始聚点的方法解决了该问题;3)凝聚层次聚类有"一旦合并则不能撤销"的缺点,通过第二步的 K 均值可以重新更改点的聚类簇位置,即允许重新分配数据点,基本解决了局部最优变成全局最优的问题,这在一定程度上提高了聚类质量;4)根据相似度阈值 f 和相应的聚类约束倍数 t 的选择并约束聚类空间,能得到较高的聚类质量,并能最大限度发现各种复杂数据特性的簇,且时间性能优于层次凝聚算法;5)基本能发现大多数形状的子簇。

参考文献:

[1] MURTY N M, KRISHNA G. A hybrid clustering procedure for concentric and chain-like clusters[J]. Interantional Journal of Computer and Information Sciences, 1981, 10(6): 397 - 412.

[2] KARPIS G, HAN E-H, KUMAR V. CHAMELEON: A hierarchical clustering algorithm using dynamic modeling[J]. Computer, 1999, 32: 68 - 75.

[3] ZHANG T, RAMAKRISHNAN R, LIVNY M. BIRCH: An efficient data clustering method for very large databases[C]// Proceedings of 1996 ACM-SIGMOD International Conference on Management of Data (SIGMOD'96). Montreal, Canada: ACM Press, 1996: 103 - 114.

[4] HAN J, KAMBER M. 数据挖掘概念与技术[M]. 范明, 孟小峰, 译. 北京: 机械工业出版社, 2001.

[5] TAN P-N, STEINBACH M, KUMAR V. 数据挖掘导论[M]. 范明, 范宏建, 译. 北京: 人民邮电出版社, 2006.

[6] 张儒良, 王翰虎. 一种有效的聚类分析算法的研究[J]. 计算机时代, 2004(9): 34 - 35.

[7] LIN C-R, CHEN M-S. Combining partitional and hierarchical algorithms for robust and efficient data clustering with cohesion self-merging[J]. IEEE Transactions on Knowledge and Data Engineering, 2005, 17(2): 145 - 159.