ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.18, No.4, April 2007, pp.884−892
DOI: 10.1360/jos180884

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

一种有效的挖掘数据流近似频繁项算法*

王伟平[1+]，李建中[1,2]，张冬冬[1]，郭龙江[1,2]

[1](哈尔滨工业大学 计算机科学与技术学院,黑龙江 哈尔滨 150001)

[2](黑龙江大学 计算机科学与技术学院,黑龙江 哈尔滨 150080)

An Efficient Algorithm for Mining Approximate Frequent Item over Data Streams

WANG Wei-Ping[1+], LI Jian-Zhong[1,2], ZHANG Dong-Dong[1], GUO Long-Jiang[1,2]

[1](School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China)

[2](School of Computer Science and Technology, Heilongjiang University, Harbin 150080, China)

+ Corresponding author: Phn: +86-451-6415872, E-mail: wpwang@hit.edu.cn, http://www.hit.edu.cn

Wang WP, Li JZ, Zhang DD, Guo LJ. An efficient algorithm for mining approximate frequent item over data streams. *Journal of Software*, 2007,18(4):884−892. http://www.jos.org.cn/1000-9825/18/884.htm

Abstract: A frequent item of a data stream is a data point whose occurrence frequency is above a given threshold. Finding frequent item of data stream has wide applications in various fields, such as network traffic monitor, data stream OLAP and data stream mining, etc. In data stream model, the algorithm can only scan the data in one pass and the available memory space is very limited relative to the volume of a data stream, therefore it is usually unable to find all the accurate frequent items of a data stream. This paper proposes a novel algorithm to find ε-approximate frequent items of a data stream, its space complexity is $O(\varepsilon^{-1})$ and the processing time for each item is $O(1)$ in average. Moreover, the frequency error bound of the results returned by the proposed algorithm is $\varepsilon(1-s+\varepsilon)N$. Among all the existed approaches, this method is the best.

Key words: data stream; data mining; frequent item; ε-approximate

摘 要: 数据流频繁项是指在数据流中出现频率超出指定阈值的数据项.查找数据流频繁项在网络故障监测、流数据分析以及流数据挖掘等多个领域有着广泛的应用.在数据流模型下,算法只能一遍扫描数据,并且可用的存储空间远远小于数据流的规模,因此,挖掘出所有准确的数据流频繁项通常是不可能的.提出一种新的挖掘数据流近似频繁项的算法.该算法的空间复杂性为 $O(\varepsilon^{-1})$,每个数据项的平均处理时间为 $O(1)$,输出结果的频率误差界限为 $\varepsilon(1-s+\varepsilon)N$,在目前已有的同类算法中均为最优.

关键词: 数据流;数据挖掘;频繁项;ε-近似

中图法分类号: TP311 文献标识码: A

* Supported by the Key Program of the National Natural Science Foundation of China under Grant No.60533110 (国家自然科学基金重点项目); the National Natural Science Foundation of China under Grant No.60473075 (国家自然科学基金); the Key Program of Natural Science Foundation of Heilongjiang Province of China under Grant No.zjg03-05 (黑龙江省自然科学基金); the Program for New Century Excellent Talents in University of China under Grant No.NCET-05-0333 (新世纪优秀人才支持计划)

Received 2006-04-14; Accepted 2006-05-16

数据流是一个按照时间递增顺序排列的无穷序列.近年来,很多应用都涉及到数据流.例如,传感器网络中的监测信号、互联网中传递的 IP 数据包、Web 服务器上的用户点击记录、电信公司的通话记录等等.与传统的数据库不同,数据流中的数据是无限的,无法全部保存下来,并且数据流上的查询具有很强的实时性要求.因此,数据流查询处理算法都是基于数据流一遍扫描(one pass)的主存算法.这些算法在内存中动态地维护数据流的概要数据(synopsis),利用概要数据来满足用户的查询.查询结果通常是真实查询结果的近似[1].

数据集合的频繁项是指在数据集合中出现频率超过一定阈值的数据项.假设一个数据集合中数据项的个数为 N,给定支持度 $s\in(0,1)$,则所有出现次数超过 sN 的数据项被称为频繁项.挖掘数据集合的频繁项有着广泛的应用,下面是几个取自数据库、数据挖掘和网络等领域的具体应用实例.

Iceberg 查询[2].Iceberg 查询是一种带有 HAVIGN 条件的聚集查询,其中,HAVIGN 条件是用户指定的阈值.例如:给定关系 $R=(c_1,c_2,c_3)$,下面的查询即为一个 Iceberg 查询,其中,θ为用户指定的阈值.

SELECT c_1,c_2,COUNT(c_3)

FROM R

GROUP BY c_1,c_2

HAVING COUNT(c_3)>=θ

挖掘关联规则[3].在数据集合上挖掘关联规则时,首先需要计算数据集合上的频繁项集.

网络故障监测[4].在网络故障监测的应用中,需要对流过的 IP 数据包进行监视.如果某一类 IP 数据包大量出现,则可能意味着网络出现了异常.例如,当网络中目的地址相同的 IP 数据包大量出现时,则可能发生了拒绝服务攻击(DOS 攻击).

挖掘数据流的频繁项是一个具有挑战性的问题.由于数据流具有无限性的特点,算法只能一遍扫描数据.通过一遍扫描数据流来计算精确的频繁项,至少需要 $O(M)$的存储空间,M 为数据流中出现的所有不同值的集合,可能是无穷的.于是,准确挖掘数据流中的频繁项通常是不可能的.因此,人们主要关注挖掘数据流频繁项近似算法[5−8].挖掘数据流频繁项近似算法的特点是一遍扫描,只使用很少的存储空间,查询结果虽然是真实结果的近似,但算法可以保证近似查询结果与真实结果之间的误差不超过用户指定的区间范围.下面,我们给出挖掘数据流中频繁项近似算法的相关概念.

设 $s\in(0,1)$和$\varepsilon\in(0,1)$分别是用户指定的支持度和误差度(ε远小于 s),N 是当前数据流已经到来的数据项个数.在任意时刻,用户发出频繁项查询,如果算法保证其输出结果满足:

(1) 所有真实频率超过 sN 的数据项均被输出;

(2) 所有真实频率低于$(s-\varepsilon)N$ 的数据项均不输出;

(3) 算法估计的频率值与真实频率的误差小于εN,

则称算法输出结果满足ε-近似要求.如果算法以概率 1 保证结果满足ε-近似要求,则称该算法为确定的ε-近似算法.如果算法以概率 $1-\delta$来保证查询结果满足ε-近似的要求(其中,$\delta\in(0,1)$),则称该算法为(ε,δ)随机近似算法.在评价ε-近似算法的优劣时,通常使用两个度量:算法的空间复杂度和数据流每个数据项的平均处理时间.具有最少空间需求、处理速度最快的ε-近似算法是研究人员追求的目标.本文提出了一种确定的ε-近似算法来挖掘数据流上的频繁项.在任意时刻,这个算法的输出结果均满足ε-近似要求.与已有的研究工作相比,本文的贡献为:

(1) 本文算法的空间复杂度为 $O(\varepsilon^{-1})$.数据流每到一个数据项,算法的平均处理时间均为 $O(1)$.在已有的同类算法中,本文算法的空间复杂度和时间复杂度最低;

(2) 本文提出的算法无须预知数据流的值域范围,并且不受数据流数据分布的影响.算法易于实现;

(3) 对于用户指定的误差度ε和支持度 s,本文算法在理论上可以保证算法估计的频繁项频率与真实的频繁项频率之间的误差不超过$\varepsilon(1-s+\varepsilon)N$,而其他算法所保证的误差界限为$\varepsilon N$.本文算法结果的精度更高.

本文第 1 节介绍相关工作.第 2 节给出本文算法的详细描述,证明算法的正确性,分析算法空间复杂性和时间复杂性,并给出结果误差分析.第 3 节将本文算法与其他算法进行对比,从理论和实验结果两个方面证明本文

算法的性能.第 4 节总结本文的工作.

1 相关工作

挖掘静态数据集合的频繁项已有很长历史,在已经取得的研究结果中[2,3,9],最好的算法[9]需要两遍扫描数据集合,只使用 $O(s^{-1})$的存储空间.近年来,在数据流上挖掘频繁项成为一个热点的研究问题.由于多遍扫描的精确频繁项挖掘算法不适用于数据流模型,因此,人们开始研究基于一遍扫描的频繁项挖掘近似算法,并取得一些研究成果[5−8,10−12].这些研究成果按照所采用的技术来划分,主要有两类:Hash 和抽样,下面分别加以介绍.

一般地,基于 Hash 的方法将数据流的值域映射到主存能够保存的 Hash 表中,Hash 表的每个桶对应一个计数器,具有相同 Hash 值的数据项共用一个计数器.这些具有相同 Hash 值的数据项的频率使用同一计数器的值进行估计,其估计的频率值与真实频率值之间会有一定的误差,基于 Hash 的方法可以保证其估计的频率值误差不超过一定的界限.在基于 Hash 的方法中,Charikar 等人提出了 Count Sketch 方法[5],需要 $O(k/\varepsilon^2\log N/\delta)$的存储空间,以 $1-\delta$的概率输出所有出现频率超过 $1/(k+1)$的数据项;Cormode 等人提出了 groupTest 算法[6],算法的空间复杂性降为 $O(k(\log k+\log 1/\delta)\log M)$;Jin 等人提出了 hCount 算法[7],hCount 算法使用 k 个相互独立的 Hash 函数,每个 Hash 函数均将数据流的值域$[0,M-1]$映射到 m 个桶中,每个桶中保存一个计数器.当数据流到来一个数据项,则用 k 个 Hash 函数对其哈希,更新相应的 k 个计数器,然后使用 k 个计数器中最小的值作为其频率的估计结果.对于用户指定误差度ε和概率δ,hCount 算法使用 $O(\varepsilon^{-1}\log(-M/\log\delta))$的空间以 $1-\delta$的概率满足查询结果ε-近似的要求.基于 Hash 的方法均是(ε,δ)随机近似算法,其中多数算法需要知道数据流的值域范围,而数据流的值域通常是未知的.

抽样是另一种挖掘数据流频繁项的常用技术.Manku 和 Motwani 提出了一种确定的ε-近似算法——Lossy Counting 算法[8].Lossy Counting 算法的基本思想是:在主存中维护数据流的一个样本集合,每当数据流到来一个数据项,若其值已经出现在样本集合中,则将相应的计数器加 1;否则,将新到的数据项以及该数据项此前在数据流中出现频率的上界(估计值)加入到样本集合中.数据流每到来 $1/\varepsilon$个数据项,Lossy Counting 算法对样本集合进行一次扫描,删除其中频率低于εN 的样本.Lossy Counting 算法的空间复杂度为 $O(1/\varepsilon\log\varepsilon N)$.在同一篇文章中,Manku 和 Motwani 还提出了一种(ε,δ)随机近似算法——Sticky 算法,该算法在 $O(1/\varepsilon\log s^{-1}\delta^{-1})$的空间内以 $1-\delta$的概率满足查询结果ε-近似的要求.Demaine 等人使用 k 个计数器,输出了出现频率超过 $1/(k+1)$的频繁项.但在数据流数据分布未知的情况下,他们提出的算法不能给出输出结果频率值的误差范围[10].而本文提出的算法能够保证输出结果的频率误差不超过$(1-s+\varepsilon)N$.在文献[11]中,Graham 等人应用 Lossy Counting 算法解决了层次数据流上的频繁项计算问题.Metwally 等人解决了同时处理数据流频繁项查询和 Top-k 查询的问题[12].我们在表 1 中列出了目前主要的研究结果及其特点.

Table 1 Algorithms for mining approximate frequent items of data stream

表 1 挖掘数据流近似频繁项的算法

Algorithm	Frequency error bound	Randomize or deterministic	Space requirement	Time for peritem	Reference
Lossy counting	εN	D	$O(\varepsilon^{-1}\log\varepsilon n)$	$O(\log\varepsilon n)$	[8]
Sticky sampling	εN	R	$O(\varepsilon^{-1}\log s^{-1}\delta^{-1})$	$O(1)$	[8]
Count sketch	εN	R	$O(k/\varepsilon^2\log n)$	$O(k/\varepsilon^2\log n)$	[5]
groupTest	εN	R	$O(k(\log k+\log 1/\delta)\log M)$	$O(\log k)$	[6]
hCount	εN	R	$O(\varepsilon^{-1}\log(-M/\log\delta))$	$O(\log(-M/\log\delta))$	[7]

本文提出了一种确定的ε-近似算法来挖掘数据流上的频繁项,算法的空间复杂度为 $O(\varepsilon^{-1})$,数据流每个数据项的平均处理时间为 $O(1)$.算法输出结果的频率误差界限为$\varepsilon(1-s+\varepsilon)N$.

2 算法描述及分析

文献[9,10]提出的方法基本一致,使用 $1/s$ 个样本计算数据流中的频繁项.但是,他们的方法无法给出输出结

果的频率误差,不能保证算法结果满足ε-近似的要求.本文受文献[9]算法思想的启发,提出一种计算确定ε-近似频繁项的算法(以下记为 EC(efficient count)算法).EC 算法需要用户预先指定误差度ε,而支持度 s 由用户在发出查询时给出,s 可以是$(\varepsilon,1)$区间内的任意数值.算法固定保存数据流的 $1/\varepsilon$个样本,并随着数据流数据的不断到来,以一定的原则动态地维护样本集合.在任意时刻,当用户查询支持度超过 s 的频繁项时,EC 算法利用样本集合给出近似的查询结果.可以证明,EC 算法满足查询结果ε-近似的要求.下面给出 EC 算法的具体定义.

2.1 算法描述

EC 算法使用样本集合 D 保存 $1/\varepsilon$个样本,每个样本为一个 4 元组$\langle e,f,N_e,df\rangle$,其中:$e$ 是数据流中的一个数据项;f 和 df 为两个计数器,具体含义下文中将作介绍;N_e 为这个 4 元组加入到样本集合 D 时,数据流到来的数据项个数.在任意时刻,令 N 表示当前数据流中到来的数据项个数.

EC 算法由 ECOUNT 和 FQuery 两部分组成.当数据流到来新的数据项时,ECOUNT 算法负责维护样本集合 D;当用户发出频繁项查询时,FQuery 算法负责输出查询结果.算法的详细描述如下所示:

算法 **1**. ECOUNT.

输入:样本集合 D,新到元组 e.

输出:样本集合 D.

```
1.  N++;
2.  IF e 在 D 中
3.      e 对应的计数器 f++;
4.  ELSE
5.      IF D 不满
6.          将⟨e,1,N,0⟩加到 D 中
7.      ELSE
8.          WHILE D 中没有 f==0 的 4 元组 DO
9.              FOR D 中的每个 4 元组 DO
10.                 f−−;df++;
11.             ENDFOR
12.         ENDWHILE
13.             删除 D 中所有 f==0 的 4 元组,加入新的⟨e,1,N,0⟩
14.     ENDIF
15. ENDIF
```

算法 **2**. FQuery.

输入:样本集合 D,支持度 s.

输出:频繁项.

```
1.  FOR D 中每个 4 元组⟨e,f,N_e,df⟩ DO
2.      IF f+df>(s−ε)N
3.          输出⟨e,f,N_e,df⟩
4.      ENDIF
5.  ENDFOR
```

ECOUNT 算法如下工作:初始时,样本集合 D 为空.当数据流到来一个数据项 e 时,若 e 已经在 D 中,则将 e 对应的 f 计数器加 1;若 e 不在 D 中,但是 D 不满,则在 D 中加入一个新的 4 元组$\langle e,1,N,0\rangle$;若 e 不在 D 中,并且样本集合 D 已满,则需要删除 D 中的一个(或几个)4 元组,ECOUNT 不断地将样本集合 D 中每个 4 元组的 f 计数器同时减 1,与此同时,将每个 4 元组的 df 计数器加 1,直到集合 D 中有 4 元组的 f 计数器等于 0 为止,ECOUNT 算法删除 D 中所有 f 计数器为 0 的 4 元组,然后将新的 4 元组$\langle e,1,N,0\rangle$加入到样本集合 D 中.在任意时刻,当用

户查询支持度超过 s 的频繁项时,FQuery 算法输出样本集合 D 中所有 $f+df>(s-\varepsilon)N$ 的 4 元组,其中,$f+df$ 的值为算法给出的数据项 e 的出现频率.在下一节中,我们将证明 EC 算法的正确性,并分析算法的空间和时间复杂性.

2.2 算法正确性分析

对于数据项 e,令£表示当前 e 在数据流中出现的真实频率.令 $S[i,j]$表示数据流 S 中第 i 个与第 j 个数据项之间的所有数据项组成的数据集.令$£_{e\in S[i,j]}$表示数据项 e 在 $S[i,j]$中出现的真实频率.下面证明 EC 算法的输出结果满足ε-近似的要求,其中,引理 1 的证明引自文献[9].

引理 **1**. 在当前时刻,数据流 S 中没有出现在集合 D 中的数据项 e,其真实频率$£<\varepsilon N$.

证明:$\forall e\in S-D$,对于 e 在 S 中的每次出现,若当时 $e\in D$,则 ECOUNT 算法将 e 的 f 计数器加 1;若当时 $e\notin D$,则 ECOUNT 算法将 e 加入到 D 中,并将其 f 计数器设为 1.因此,e 在 S 中出现了£次,则 ECOUNT 算法对其 f 计数器执行了£次加 1 操作.在当前时刻,$e\notin D$ 则说明 ECOUNT 算法对 e 的 f 计数器执行了£次减 1 操作(否则, $e\in D$).由于 ECOUNT 算法只有在集合 D 满的情况下才会对 f 计数器执行减 1 操作,因而在 e 执行减 1 操作的同时,样本集合 D 中其他 $1/\varepsilon-1$ 个数据项 f 计数器也执行了减 1 操作.故 ECOUNT 算法至少执行了$£\times 1/\varepsilon$次减 1 操作,则可知$£\times 1/\varepsilon<N$,$£<\varepsilon N$. □

引理 **2**. 对于 D 中的每个 4 元组$\langle e,f,N_e,df\rangle$,$f+df$ 的值为数据项 e 在 $S[N_e,N]$中出现的真实频率,即

$$f+df=£_{e\in S[N_e,N]}.$$

证明:由 ECOUNT 算法的定义可知,4 元组$\langle e,f,N_e,df\rangle$在 N_e 时刻加入到集合 D 中.对于数据项 e 在 $S[N_e,N]$中的每次出现,ECOUNT 算法都将 e 的 f 计数器加 1.而在此期间,ECOUNT 算法对 e 的 f 计数器执行减 1 操作的次数为 e 的 df 值.所以,$f+df=£_{e\in S[N_e,N]}$. □

引理 **3**. 对于 D 中的每个 4 元组$\langle e,f,N_e,df\rangle$,其 $f+df<=£$.

证明:由引理 2 可知. □

引理 **4**. 对于 D 中的每个 4 元组$\langle e,f,N_e,df\rangle$,其 $f+df>£-\varepsilon N$.

证明:任给 D 中的一个 4 元组$\langle e,f,N_e,df\rangle$,由 ECOUNT 算法的定义可知:$N_e$ 为这个 4 元组加入到 D 时,数据流到来的数据项个数.数据项 e 在 S 中出现的真实频率等于 e 在 $S[1,N_e]$中出现的真实频率加上 e 在 $S[N_e,N]$中出现的真实频率,即$£=£_{e\in S[1,N_e]}+£_{e\in S[N_e,N]}$.由引理 2 可知,$f+df=£_{e\in S[N_e,N]}$.在 N_e 时刻,集合 D 中没有 e 出现,由引理 1 可知,$£_{e\in S[1,N_e]}<\varepsilon N_e$.所以,$£=£_{e\in S[1,N_e]}+£_{e\in S[N_e,N]}<f+df+\varepsilon N_e$.又由 $N_e<=N$ 可知,$£<f+df+\varepsilon N$,即 $f+df>£-\varepsilon N$. □

定理 **1**. EC 算法的输出结果满足ε-近似的要求.

证明:下面证明 EC 算法输出结果满足ε-近似的 3 点要求:

(1) 所有真实频率超过 sN 的数据项均输出.

由引理 1 可知:所有真实频率超过εN 的数据项均在 D 中,则所有真实频率超过 sN 的数据项一定在 D 中.对于每个真实频率超过 sN 的数据项 e,$£>sN$,则$£-\varepsilon N>(s-\varepsilon)N$.由引理 4 可知,$f+df>£-\varepsilon N$,故 $f+df>(s-\varepsilon)N$.FQuery 算法输出了所有 $f+df>(s-\varepsilon)N$ 的数据项,所以,真实频率超过 sN 的数据项均被输出.

(2) 所有真实频率低于$(s-\varepsilon)N$ 的数据项均不输出.

对于 D 中每个真实频率低于$(s-\varepsilon)N$ 的数据项 e,即$£<(s-\varepsilon)N$,由引理 3 可知,$f+df<£$,故 $f+df<(s-\varepsilon)N$,由 FQuery 算法的定义可知,这样的数据项没有输出.

(3) 算法估计的频率值与真实频率的误差小于εN.

由引理 3 和引理 4 可知:$£-\varepsilon N<f+df<=£$,其中,$f+df$ 为算法估计的频率值.

综上,EC 算法的输出结果满足ε-近似的要求. □

2.3 算法复杂性分析

下面分析 EC 算法的空间和时间复杂性.

定理 **2**. EC 算法的空间复杂度为 $O(\varepsilon^{-1})$,数据流每个数据项的平均处理时间为 $O(1)$.

证明:显然,EC 算法的存储开销为 $O(\varepsilon^{-1})$.下面分析数据流每个数据项的平均处理时间.我们使用 Hash 方法

来存储样本集合 D,可以证明数据流每个数据项的平均处理时间为 $O(1)$.当数据流每到来一个数据项 e 时,EC 算法执行的操作有两种:

(1) 插入操作

插入操作又有两种可能.一种是 e 已经在样本集合 D 中,此时,需要将 e 所对应的计数器 f 加 1,则该操作的处理时间为 $O(1)$;另外一种可能是 e 不在样本集合 D 中,此时,需要在样本集合 D 中加入 e,并将其计数器 f 的值置为 1,该操作的处理时间也为 $O(1)$.

(2) 删除操作

并不是每个数据项的到来 EC 算法均需执行删除操作,只有在新来的数据项不在样本集合 D 中并且 D 是满的情况下,EC 算法才需要执行删除操作.这里,我们应用平摊分析的思想,先分析到目前为止,EC 算法执行删除操作总的时间开销,然后分析平摊到每个数据项上的删除操作的时间开销.设当前数据流到来的数据项总的个数为 N,由于每个新数据项 e 的到来均会令 EC 算法执行一次 $e.f$++操作*,则 EC 算法执行总的 f++操作的次数为 N.由 EC 算法的定义可知,EC 算法执行总的 f−−操作次数一定小于等于其执行总的 f++操作的次数.因此,EC 算法执行 f−−和 df++操作的总次数之和小于等于 $2N$.执行一次 f−−(或者 df++)操作的时间开销为 $O(1)$,故 EC 算法执行删除操作总的时间开销为 $O(2N)$,即 $O(N)$.平摊到数据流 N 个数据项上,则每个数据项平均处理删除操作的时间开销为 $O(1)$.

由上述(1),(2)可知,数据流每个数据项的平均处理时间为 $O(1)$. □

2.4 算法误差分析

在引理 4 中,我们证明了对于 D 中的每个样本$\langle e,f,N_e,df\rangle$均有$£-(f+df)<\varepsilon N_e$,$f+df$ 为 EC 算法输出的 e 在数据流中的出现频率.因此,对于 FQuery 输出的每个数据项 e,其估计频率与真实频率的误差小于εN_e,如果$\varepsilon N_e<1$,则可断定 EC 算法给出的频率就是 e 在数据流中出现的真实频率.

定理 **3**. 给定支持度 s,在 EC 算法的输出结果中,算法给出的每个数据项 e 的频率与其真实频率的误差小于 $\varepsilon(1-s+\varepsilon)N$.

证明:由 FQuery 的定义可知,对于 FQuery 输出的每个数据项 e,均有 $f+df>(s-\varepsilon)N$.而 $f+df$ 为 e 在 $S[N_e,N]$中出现的真实频率,则必有 $N-N_e>=f+df$,因而 $N-N_e>(s-\varepsilon)N$,即 $N_e<(1-s+\varepsilon)N$.由引理 4 的证明可知,$£-(f+df)<\varepsilon N_e$.因此,$£-(f+df)<\varepsilon(1-s+\varepsilon)N$. □

由于 s 远大于ε,因此,$\varepsilon(1-s+\varepsilon)N<\varepsilon N$.目前,在已有的挖掘数据流频繁项的近似算法中,对于给定的ε,其输出结果的频率误差范围为εN.而定理 3 说明,EC 算法输出结果的频率误差范围为$\varepsilon(1-s+\varepsilon)N$,算法输出结果频率值的精度更高.

3 实验结果及分析

我们使用模拟数据集和真实数据集对 EC 算法的性能进行了测试.在已有的研究工作中,Lossy Counting 算法是确定的ε-近似算法,hCount 算法在几种(ε,δ)随机近似算法中性能是最好的.因此,我们也实现了 Lossy Counting 算法[8]和 hCount 算法[7]与本文提出的 EC 算法进行比较.算法使用 C 语言实现,实验在 PC 机上运行,CPU 主频为 3GHz,内存为 512MB.

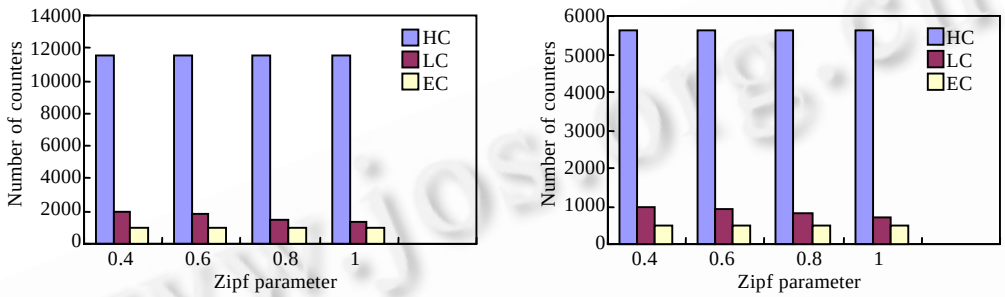
3.1 模拟数据集实验

本文提出的 EC 算法不受数据分布的影响,算法的空间开销只与误差度ε有关,而 Lossy Counting 算法的空间开销与误差度ε和数据分布均有关.因此在实验中,我们测试了在误差度和数据分布变化情况下 3 种算法的性能.实验使用 3 个指标来衡量算法的性能:算法内存开销、算法平均处理时间和算法输出结果频率的绝对误差.

* 若 e 不在样本集合 D 中,则 EC 算法将其加入到样本集合 D 中,并将其频率值 f 置为 1,仍可看作 f 的值由 0 变为 1,执行了一次 f++操作.

在模拟数据实验中,我们生成了 4 组不同参数的 zipf 分布的数据集作为实验数据.每个数据集的值域均为[1...10000],数据集大小为 1 000 000 个数据项.由于 hCount 算法是(ε,δ)随机近似算法,实验中δ取值为 0.01.

在实验 1 中,我们考察了在数据分布变化的情况下 3 种算法的存储开销.实验结果如图 1 所示.实验中,误差度ε分别取 0.1%和 0.2%.实验结果说明,本文提出的 EC 算法内存开销最小,而 Lossy Counting 算法的空间需求小于 hCount 算法.hCount 算法和 EC 算法的内存开销只与误差度ε有关,与数据分布无关.在误差度ε取值相同的情况下,hCount 算法的内存开销远大于 EC 算法.在最坏情况下,Lossy Counting 算法空间复杂度为$\varepsilon^{-1}\log\varepsilon n$,但是,这种情况并不经常出现.实验结果表明,当数据集服从 zipf 分布时,Lossy Counting 算法的内存开销要低于 hCount 算法.图 1 的实验结果同时说明,当误差度ε取值越小时,3 种算法需要的内存空间就越大.

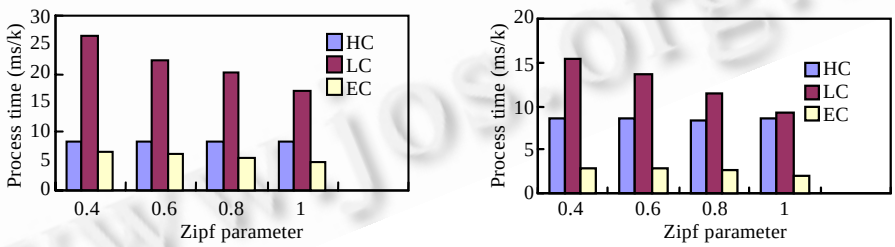


(a) ε=0.1% (b) ε=0.2%

Fig.1 The effect of varying data distribution on the algorithms' space requirement

图 1 变化的数据分布对算法存储开销的影响

在实验 2 中,我们考察了在数据分布变化情况下 3 种算法的平均处理时间.实验结果如图 2 所示.实验中,误差度ε分别取 0.1%和 0.2%.实验结果表明,每个数据项的平均处理时间也是 EC 算法最低,而 hCount 算法要好于 Lossy Counting 算法.hCount 算法的处理时间与数据分布无关,而随着 Zipf 参数的增大,Lossy Counting 算法和 EC 算法的处理时间均呈下降趋势.这是因为 Zipf 参数越大,数据集的偏斜度就越大,Lossy Counting 算法和 EC 算法执行删除操作的时间开销变小,因而平均处理时间就会降低.3 种算法的处理时间均与误差度ε有关,ε取值越小,算法的平均处理时间越大.

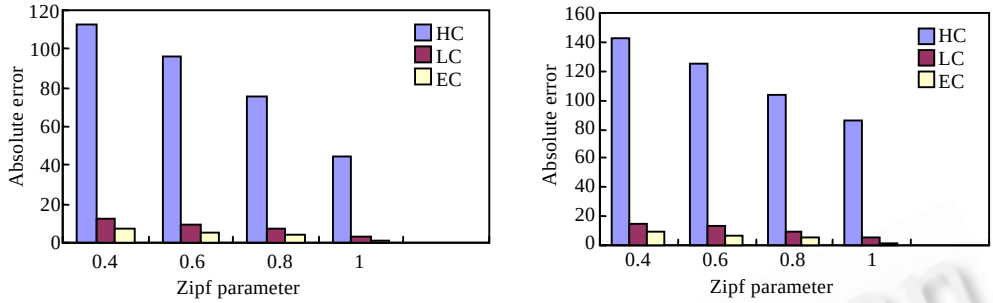


(a) ε=0.1% (b) ε=0.2%

Fig.2 The effect of varying data distribution on the algorithms' processing time

图 2 变化的数据分布对算法处理时间的影响

在实验 3 中,我们测试了 3 种算法输出结果频率的绝对误差.图 3 给出的是每种算法输出结果中最大的频率误差.Lossy Counting 算法和 EC 算法输出结果频率的绝对误差很小,远远低于 hCount 算法;而 Lossy Counting 算法的绝对误差略高于 EC 算法.随着 Zipf 参数的增大,数据集的偏斜度也增大,因而 3 种算法的绝对误差变小.实验结果表明:误差度ε取值越小,算法输出的频率值误差就越小.这与理论分析的结果是一致的.



(a) ε=0.1% (b) ε=0.2%

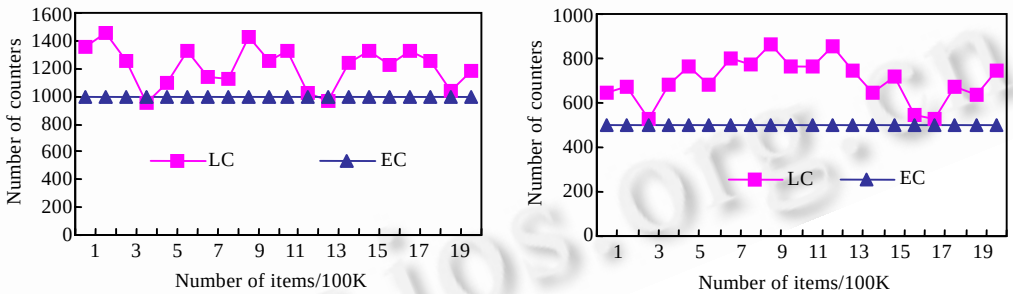
Fig.3 The effect of varying data distribution on the results' frequency error

图 3 变化的数据分布对算法输出结果频率误差的影响

3.2 真实数据集实验

在真实数据实验中,我们使用了 1998 年世界杯官方网站的访问日志作为实验数据[13].这份日志记录了在 1998 年世界杯比赛期间,世界杯官方网站的所有访问请求(共计 1 352 804 107 个).每个访问请求包括访问时间、源 IP 地址、访问页面的 ID 等 8 个属性,我们提取了其中的访问页面 ID 作为实验数据.通过挖掘访问页面的频繁项,可以了解世界杯期间哪些页面是最热的.

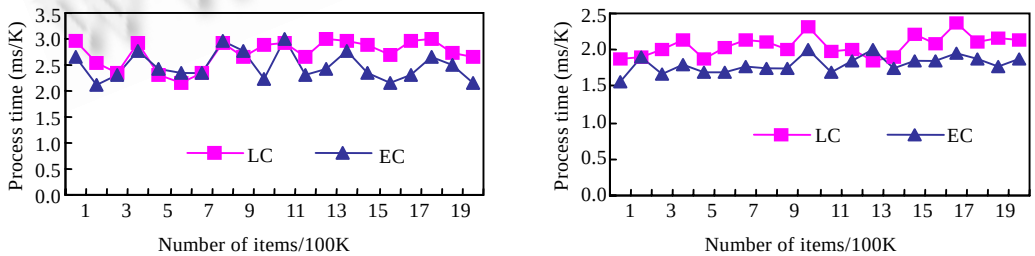
由于 hCount 算法的空间开销和频率误差远大于 Lossy Counting 算法和 EC 算法,因此在真实数据实验中,我们只对 Lossy Counting 算法和 EC 算法进行了测试.我们提取了 200 万个访问页面 ID 作为实验数据.真实数据集的实验结果说明:EC 算法的内存需求低于 Lossy Counting 算法;EC 算法的平均处理时间略低于 Lossy Counting 算法;EC 算法输出结果的频率误差也要低于 Lossy Counting 算法.整体而言,EC 算法的性能优于 Lossy Counting 算法(如图 4、图 5 所示).



(a) ε=0.1% (b) ε=0.2%

Fig.4 Algorithms' space requirement

图 4 算法的空间开销



(a) ε=0.1% (b) ε=0.2%

Fig.5 Algorithms' processing time

图 5 算法的处理时间

4 总 结

挖掘数据流的频繁项是一个具有挑战性的问题.在数据流处理模型下,算法只能一遍扫描数据,并且可用的存储空间非常有限.本文提出了一种确定的ε-近似算法——EC 算法.算法的空间开销为 $O(\varepsilon^{-1})$,数据流每个数据项的平均处理时间为 $O(1)$.在目前已有的研究结果中,本文算法的空间复杂度和时间复杂度均为最低.理论分析和实验结果表明,本文算法在输出结果的精度上也优于同类算法.

References:

[1] Babcock AK, Babu S, Datar M. Model and issues in data stream systems. In: Popa L, ed. Proc. of the 21st ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems. Madison: ACM, 2002. 1−16.

[2] Fang M, Shivakumar N, Garcia-Molina H, Motwani R, Ullman J. Computing iceberg queries eefficiently. In: Gupta A, Shmueli O, Widom J, eds. Proc. of the 24th Int'l Conf. on Very Large Data Bases. New York: Morgan Kaufmann Publishers, 1998. 299−310.

[3] Agrawal R, Srikant R. Fast algorithms for mining association rules. In: Bocca JB, Jarke M, Zaniolo C, eds. Proc. of the 20th Int'l Conf. on Very Large Data Bases. Santiago: Morgan Kaufmann Publishers, 1994. 487−499.

[4] Estan C, Verghese G. New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice. ACM Trans. on Computer Systems, 2003,21(3):270−313.

[5] Charikar M, Chen K, Farach-Colton M. Finding frequent items in data streams. In: Widmayer P, Ruiz FT, Bueno RM, Hennessy M, Eidenbenz S, Conejo R, eds. Proc. of the Int'l Colloquium on Automata, Languages and Programming. Malaga: Springer-Verlag, 2002. 693−703.

[6] Cormode G, Muthukrishnan S. What's hot and what's not: Tracking most frequent items dynamically. In: Halevy AY, Ives ZG, Doan AH, eds. Proc. of the 22nd ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems. San Diego: ACM Press, 2003. 296−306.

[7] Jin C, Qian W, Sha C, Yu JX, Zhou A. Dynamically maintaining frequent items over a data stream. In: Carbonell J, ed. Proc. of the 2003 ACM CIKM Int'l Conf. on Information and Knowledge Management. New Orleans: ACM Press, 2003. 287−294.

[8] Manku GS, Motwani R. Approximate frequency counts over data streams. In: Bernstein P, Ioannidis Y, Ramakrishnan R, eds. Proc. of the 28th Int'l Conf. on Very Large Data Bases. Hong Kong: Morgan Kaufmann Publishers, 2002. 346−357.

[9] Karp R, Papadimitriou C, Shenker S. A simple algorithm for finding frequent elements in sets and bags. Trans. on Database Systems, 2003,28(1):51−55.

[10] Demaine E, López-Ortiz A, Munro JI. Frequency estimation of Internet packet streams with limited space. In: Möhring RH, Raman R, eds. Algorithms. ESA 2002, Proc. of the 10th Annual European Symp. Rome: Springer-Verlag, 2002. 348−360.

[11] Graham C, Flip K, Muthukrishnan S, Divesh S. Finding hierarchical heavy hitters over data sreams. In: Freytag JC, Lockemann PC, Abiteboul S, Carey MJ, Selinger PG, Heuer A, eds. Proc. of the 29th Int'l Conf. on Very Large Data Bases. Berlin: Morgan Kaufmann Publishers, 2003. 464−475.

[12] Metwally A, Agrawal D, Abbadi AE. Efficient computation of frequent and top-k elements in data streams. In: Eiter T, Libkin L, eds. Proc. of the Int'l Conf. on Data Theory. Edinburgh: Springer-Verlag, 2005. 398−412.

[13] http://ita.ee.lbl.gov/html/contrib/WorldCup.html. 2004.

王伟平(1975－),男,吉林舒兰人,博士生,主要研究领域为数据流查询处理.

张冬冬(1976－),男,博士,主要研究领域为分布式数据流查询处理.

李建中(1950－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为数据库,并行计算.

郭龙江(1973－),男,博士生,主要研究领域为数据流挖掘.