

基于 Spark 的高维数据相似性连接

成小海

(天津工业大学 计算机科学与软件学院, 天津 300387)

摘要:高维数据相似性连接(HDSJ)是指在给定的空间数据库中,频繁执行连接和距离计算操作找出向量空间满足给定条件的数据对。但是随着数据量和维数的增加,HDSJ 的计算成本将呈指数增加。针对 HDSJ 在处理海量数据时效率不佳的问题,利用 Spark 集群分布式和基于内存并行计算特性,提出了基于 Spark 框架的 HDSJ 改进方法。该方法主要借助 Spark 中高效的 RDD 算子,使用分段聚合近似(PAA)表示原始的高维向量,用符号聚合近似(SAX)将表示后的向量重新组织成组,这样可以避免大量不必要的计算。PAA 和 SAX 都是已有的降维技术,将二者结合使用可以很好地过滤掉大部分的干扰数据。实验结果证明,该方法在保证实验结果准确率的前提下提高了运算速率,比现有方法有更好的性能优势。

关键词:高维数据;相似性连接;Spark;分段聚合近似;符号聚合近似

中图分类号:TP311

文献标识码:A

文章编号:1673-629X(2018)08-0043-05

doi:10.3969/j.issn.1673-629X.2018.08.009

Similarity Joins of High-dimensional Data Based on Spark

CHENG Xiao-hai

(School of Computer Science and Software Engineering, Tianjin Polytechnic University,
Tianjin 300387, China)

Abstract:High-dimensional data similarity joins (HDSJ) is to find the data pairs of meeting the conditions by frequently using operations of the joins and distance calculation in a given spatial database. However, with the increasing of the data volume and the number of the dimensions, the computational cost of HDSJ will increase exponentially. In order to solve the problem of HDSJ of poor efficiency, we propose an improved method of HDSJ by using Spark cluster and memory parallel computing. This method mainly uses piecewise aggregate approximation (PAA) to represent the high-dimensional vectors and reorganize these vectors into groups based on their symbolic aggregate approximation (SAX) representations by using the efficient RDD operator in Spark cluster, which avoids many unnecessary calculations. PAA and SAX are existing dimensionality reduction techniques, the combination of the two can be used to filter out most of the interference data. Experiment shows that the proposed method can improve the operation rate while ensuring the accuracy rate, which has better performance than that of the existing method.

Key words:high-dimensional data; similarity joins; Spark; piecewise aggregate approximation; symbolic aggregate approximation

0 引言

高维数据相似性连接不仅可以用于分类,而且还可以用于预测,在文本分类、聚类分析、预测分析、模式识别、图像处理等领域应用广泛。但高维数据相似性连接仍是一个非常具有挑战性的工作,主要有以下两个原因。首先,数据集的规模非常大(数百万或数十亿的对象);其次,数据集的维数足够高(数千或数万)。因此不可能直接对数据集进行操作,必须借助有效的方法来降低数据集的维数,从而进行数据集间的运算^[1],其中用的最多的就是结合 Hadoop 框架^[2-9]

进行分布式运算。

近年来,有学者不断地对高维数据的相似性连接进行了研究和优化。例如,戴健等^[7]整合 MapReduce 框架,提出了分布式网格概略化 KNN joins(DSGMP-J)和基于 MapReduce 的 voronoi diagram 下的 KNN joins(VDMP-J);马友忠等^[8]结合 Hadoop 集群的分布式特性和 SAX 的降维技术,提出了向量随机生成 id 的方式进行 SAX 转换,虽然可以起到分布式计算的效果,但在运算过程中由于复制多份数据和相同 id 向量的重复计算导致内存消耗较大;刘雪莉等^[10]提出了实

收稿日期:2017-10-16

修回日期:2018-02-27

网络出版时间:2018-04-28

基金项目:国家自然科学基金(61402329)

作者简介:成小海(1993-),男,硕士研究生,CCF 会员,研究方向为大数据分析、数据挖掘。

网络出版地址: <http://www.cnki.net/kcms/detail/61.1450.TP.20180427.1626.020.html>

体数据库上相似性连接算法 ES-joins,用于解决字符串模糊匹配的相似性连接问题;刘艳等^[11]利用 Δ -tree 进行高维数据相似连接;周健雯等^[12]使用 R * 树的自相似性连接。

由于 Spark 具有运算速度快、易用性好和通用性强等优点,并且可以高效地处理大规模数据集,因此文中采用基于 Spark 的相似性连接,对原有基于 MapReduce 的方法进行改进,优化其中的计算步骤,同时结合 Spark 强大的 RDD 算子,以提高计算速度。

1 相关工作及背景知识

1.1 Spark 框架

Spark 诞生于加州大学伯克利分校 AMP 实验室,是一个基于大数据的分布式并行计算框架。由于其先进的设计理念,从 2013 年成为 Apache 孵化项目后,先后扩展了 Spark SQL、Spark Streaming、Mllib、GraphX 和 SparkR 等组件,逐渐成为大数据处理一站式解决平台。Spark 框架提供了多种资源调度管理,通过内存计算、有向无环图等机制保证分布式的快速迭代计算,同时引入了 RDD 的抽象保证数据的高容错性。

RDD 是一个分布式的内存抽象概念和只读分区记录的集合,这些集合是弹性的,如果数据集一部分丢失,则可以根据父子 RDD 中的继承关系重新进行计算。Spark 的创建、转换、控制和行为等操作都是来自对 RDD 的操作。

1.2 与 MapReduce 的比较

Spark 是对 MapReduce 分布式大数据处理的借鉴,并对其进行了内存框架优化,使其具有更强的数据处理能力。

(1)Spark 将中间过程放进内存,迭代效率明显提高。而 MapReduce 将中间计算结果保存在磁盘中,如果多个 Reduce 将会严重影响程序的运行速度。Spark 支持有向无环图的分布式计算,中间运算结果保存在内存中,使其整体效率明显提升。

(2)Spark 容错性高。Spark 使用了 RDD 操作,可以在计算时通过 CheckPoint 实现容错。

(3)Spark 在 Shuffle 时不是所有情况都进行排序,而 MapReduce 在此阶段需要花费大量时间来排序。

(4)Spark 更加通用。Spark 提供的数据操作类型有很多种,一般较为通用的是转换操作和行动操作两种类型。而 Hadoop 只提供 Map 和 Reduce 两种操作。

因此,为了让集群高效工作,可以尝试使用 Spark 框架来处理高维相似性连接问题。

结合 SAX 技术来降低向量的维数,并通过计算 SAX 表示之间的距离来提高过滤效果。这样可以减少候选对,节省计算成本。

2 问题定义

2.1 高维向量的 PAA 表示

PAA^[13]是一种维数降低技术,广泛应用于时间序列处理和轨迹相关问题研究。它就是将原始高维数据等间距分割为较低的维数,利用定义 2 给出的距离计算公式计算出原始向量的近似距离。文中使用的向量是序列无关向量,维度的顺序不会影响欧几里得距离的计算结果。在需要的时候,可以重新排列向量的维度顺序,然后用分段聚合近似表示高维向量。

2.2 高维向量的 SAX 表示

SAX^[8,14-16]也是一种维数降低技术,其主要是先将原数据规格化,使其服从高斯分布,然后离散化并用 PAA 表示,最后转化为字符串。该方法具体的变换步骤举例如下:

- (1)将原始时间序列 $R = \{R_1, R_2, \dots, R_n\}$ 规格化为均值为 0、标准差为 1 的标准序列;
- (2)将其离散化为 x 个相等大小的范围,每个范围内的数据都用定义 2 中的 PAA 表示;
- (3)通过查表 1 确定其所在的区间,并将每个区间用不同符号编号,表示形式为 $R = \{A_{1i}, A_{2i}, \dots, A_{xi}\}$ 。

表 1 从 3-8 等分区划分的高斯分布查找表

β	区间划分个数					
	3	4	5	6	7	8
β_1	-0.43	-0.67	-0.84	-0.97	-1.07	-1.15
β_2	0.43	0.00	-0.25	-0.43	-0.57	-0.67
β_3		0.67	0.25	0.00	-0.18	-0.32
β_4			0.84	0.43	0.18	0.00
β_5				0.97	0.57	0.32
β_6					1.07	0.67
β_7						1.15

图 1 为原始向量 R 用 SAX 表示后的模型。

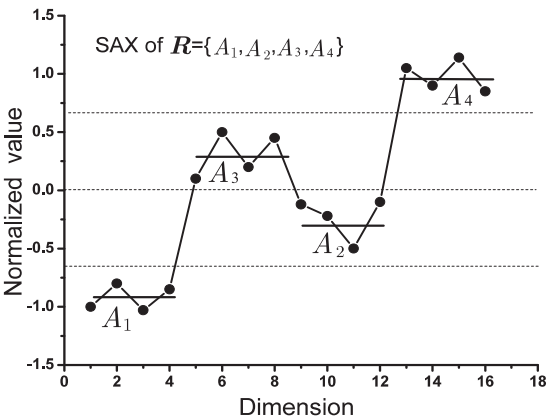


图 1 时间序列 R 的 SAX 表示

定义 1:高维相似性连接。给定两个 n 维数据集 R 和 S ,假设 R 和 S 中的所有点都在 n 维空间中,已知距离阈值为 ε ,则在 R 和 S 上的相似性度量是计算所有点对组成的集合 (R_i, S_i) 使得 $D_e(R_i, S_i) \leq \varepsilon$,其中 R_i

$\in R, S_i \in S$ 和 D_E 代表欧几里得距离。如果将 R 和 S 看作同一数据集,则进行自相似性连接。文中也主要使用自相似性连接进行实验。

定义 2:PAA 表示。给定一个 n 维向量 R ,将其维数进行等分,令 N 为等分后的维度, $R_{N1}, R_{N2}, \dots, R_{NN}$ 是 N 个维度表示,其中有关系 $R_n = R_{N1} \cup R_{N2} \cup \dots \cup R_{NN}$ 和 $R_i \cup R_j = \emptyset$ 。则向量 R 用 PAA 表示为 $R_n = (R_{N1}, R_{N2}, \dots, R_{NN})$ 。

定义 3:聚合度 λ 。假定 R 的维数为 n ,将其用 PAA 表示后的维度为 N ,则聚合度 λ 就定义为 $\lambda = n/N$ 。

文中首先将原始高维向量进行规格化。假设原始向量为 R ,其中均值为 μ ,标准差为 σ ,则规格化公式为:

$$R_i = (R_i - \mu) / \sigma$$

(1)

给定两个向量 R 和 S ,它们的欧几里得距离为:

$$D_E(R, S) = \sqrt{\sum_{i=1}^d (R_i - S_i)^2}$$

(2)

它们的 PAA 表示后的距离可以定义为:

$$D_p(R_p^A, S_p^A) = \sqrt{\lambda \sum_{i=1}^N (R_{Ni} - S_{Ni})^2}$$

(3)

在文献[15]中已经证明,PAA 表示 D_p 的距离是

原始欧几里得距离 D_E 的下限,也就是说:

$$D_p(R_p^A, S_p^A) \leq D_E(R, S)$$

(4)

给定两个向量 R 和 S 以及它们的 SAX 表示 R_s 和 S_s ,可以定义新距离:

$$\text{MINDIST}(R_s^A, S_s^A) = \sqrt{\lambda} \sqrt{\sum_{i=1}^N \text{dist}(\hat{R}_{Ni}, \hat{S}_{Ni})^2}$$

(5)

其中, $\text{dist}(\hat{R}_{Ni}, \hat{S}_{Ni})$ 是计算任意两个符号对之间距离的子函数,其中距离是通过查表得到。

容易证明, SAX 表示 (MINDIST) 之间的距离是 PAA 表示 D_p 之间的距离的下限,并且 D_p 是欧几里得距离 D_E 的下界;根据传递性,MINDIST 是欧氏距离的下边界近似:

$$\text{MINDIST}(R_s^A, S_s^A) \leq D_E(R, S)$$

(6)

3 高维相似性连接在 Spark 上的实现

高维相似性连接是文中讨论的主要问题。看似简单,但由于维数较大和数据量较多,导致计算成本呈指数倍增加。如何用较短的时间和较小的代价得到正确的结果是文中进行研究的目的。以下是高维相似性连接在 Spark 集群上实现的详细过程。

具体流程如图 2 所示。

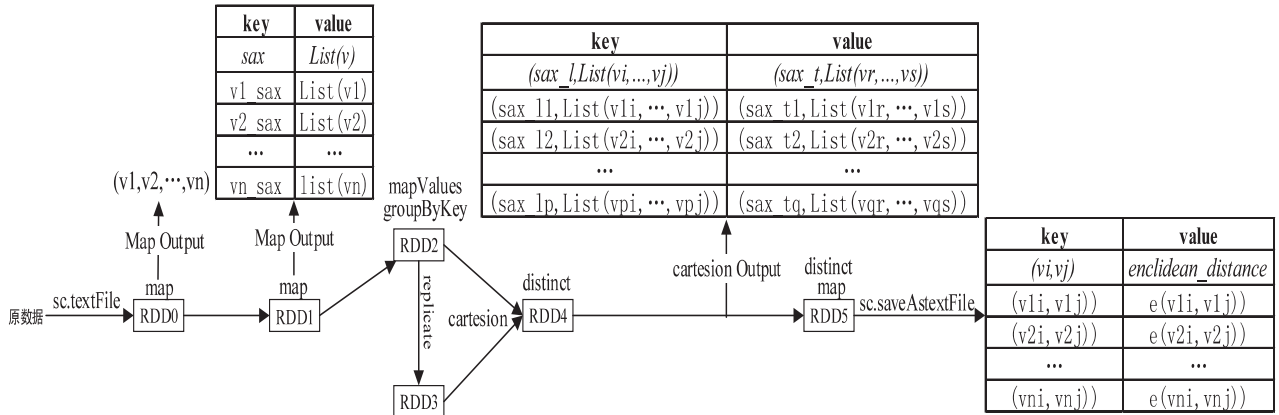


图 2 高维数据相似性连接在 Spark 上的流程

(1)数据预处理。主要通过 map 操作对数据进行规格化处理,使每一行数据都表示为均值为 0、标准差为 1 的标准序列。首先通过集群将分布式读取到的每一行数据计算其均值 μ 和标准差 σ ,然后根据规格化公式,使其标准化,最后返回标准化后数据。

(2)生成 SAX 键值对。将预处理中计算出的标准序列结合预先设定的聚合度 λ ,表示出相应的 PAA 集合。然后将生成的数据集用 SAX 表示,最后将生成的集合通过键值对返回。具体过程见算法 1。

(3)将具有相同 SAX 表示的键值对聚合。该步骤主要是通过 RDD 算子的 groupByKey 函数实现。

(4)进行数据卡方积运算。将上一步中 SAX 表示

后的 RDD 数据集通过 cartesian 函数实现自连接,返回通过 Key-value 表示后的 SAX 对。这个步骤保证数据集中每两个 SAX 表示后的数据集可以碰面一次。

(5)精化候选向量对。该步骤是计算出满足阈值的相似连接对。主要通过 map 函数比较上一步中返回的键值对中满足阈值的 key-value 对,然后计算 value 中符合条件的原始数据对之间的距离。

算法 1:Generate key-value pairs

```
line=>|
1. numbs<-line. substring(0, line. length-1). split(“\t”)
2. for i<- 0 until numbs. length do
3. numi <- numbs(i). toDouble
4. sum += numi
```

```

5. if  $k < \text{Util. } P$  then
6. if  $i < \text{numbs. length} - 1$  then
7.  $k = k + 1$ 
8. else
9. // Express by piecewise aggregate approximation
10.  $pp = \text{sum} / k$ 
11.  $\text{sax} += \text{Util. getSax}(pp)$ 
12. else
13.  $pp = \text{sum} / k$ 
14.  $\text{sax} += \text{Util. getSax}(pp)$ 
15.  $\text{sum} = 0; k = 1$ 
16. //return key-value pairs
17. ( $\text{sax}, f.\text{toList}$ )
}

```

4 实验

4.1 实验配置

该实验在 hadoop2. 6. 1 集群的基础上部署了 Spark1. 6. 1, 程序用 Scala 编写, 编译器为 IDEA2016。这个集群包含 10 个节点, 每个节点的配置如下: 核数为 4, 内存 6 GB, 磁盘 500 GB, 操作系统为 Linux CentOS release 6. 2(Final)。其中 1 个为 Master 节点, 其余 9 个为 Worker 节点, 实验中使用的数据是从 HDFS 中读取并将计算结果写入 HDFS 中。

使用的数据集来自文献[13], 选用了其中的部分数据。其中 256 维和 512 维数据集通过 960 维数据生成, 各维度向量均采用 20 万条数据进行测试。

分别在 Spark 和 Hadoop 平台上测试了相同数据集在不同条件约束下的运行时间。

4.2 实验结果分析

为了验证改进算法的有效性, 分别从三个角度进行了测试, 即维数改变对实验结果的影响、聚合度 λ 改变对实验结果的影响和阈值 ε 改变对实验结果的影响。所有实验都是基于相同硬件配置, 对原有 Hadoop 集群和优化后 Spark 集群的运行时间的比较。

首先, 比较 Hadoop 平台和 Spark 平台在设置默认聚合度 λ 和阈值 ε 的前提下, 只改变维度大小对所需时间的影响情况, 其中设置 λ 为 16, ε 为 0.1。实验结果如图 3 所示。

由图 3 可以看出, 两种平台对相同数据集维度的执行时间是不同的。基于 Spark 平台的运行时间更短, 与原有基于 Hadoop 平台的算法相比平均可缩短 10% ~ 20%。并且数据维度越高, 优势越明显。

其次, 分别改变聚合度 λ 和阈值 ε 的设定参数, 分别测试对实验运行时间的影响情况, 其中用到的数据集维度默认为 128 维。改变聚合度 λ 可以改变生成分段聚合近似数据表的维数, 从而影响符号聚合近似表

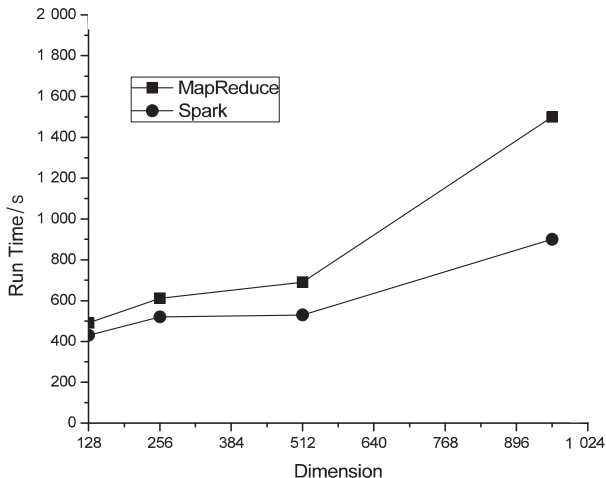


图 3 不同维度两种框架运行时间对比

示数据集的生成维数。改变阈值 ε 可以控制计算相似度的多少, 从而影响所需的计算时间。实验结果如图 4 和图 5 所示。

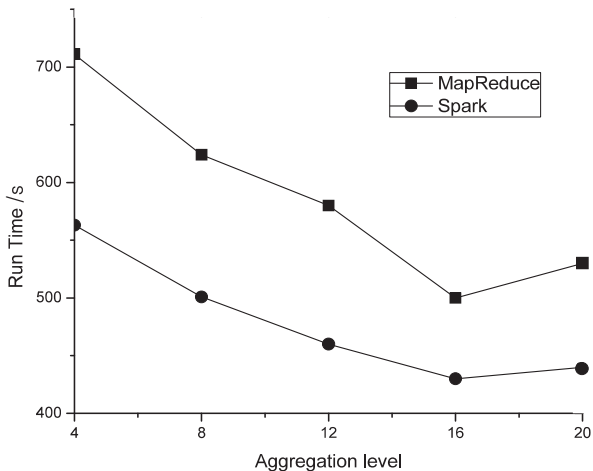


图 4 两种框架在不同聚合度 λ 下的运行时间对比

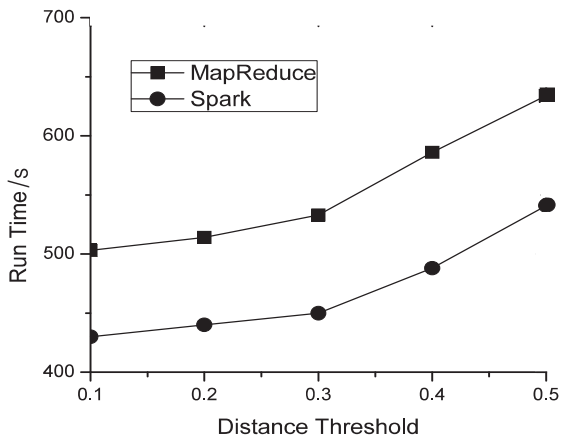


图 5 两种框架在不同阈值 ε 下的运行时间对比

由图 4 可以看出, 虽然聚合度越大, 运行时间越小, 但当聚合度增加到一定值的时候, 会看到运行时间反向增加。这是因为在进行实际向量间运算时, 由于内部向量维数较大, 增加了运算成本。但在同等条件下, Spark 计算平台的时间减小更多, 运行效率更高。

由图5可以看出,改变阈值 ε 对实验结果的运行时间有一定影响。阈值越大,花费的时间越多,但基于Spark的优化速度还是有明显的优势。

5 结束语

随着数据量规模的增加,相似性计算成本也将随之变大。单独使用一台机器已经无法满足相似性计算的性能、时间等各方面的要求。提高高维数据间的相似性连接效率是必须面对的课题。文中主要针对高维数据集相似性分类算法进行了优化和改进,使用SAX方法对数据集进行裁剪,该方法减少了数据之间不必要的距离计算时间。结合现有大数据框架分布式并行计算的特点,可以很好地提高高维数据相似性连接效率。通过实验发现,使用Spark框架的高维数据集相似性连接算法具有显著的性能优势。

参考文献:

- [1] 庞俊,于戈,许嘉,等.基于MapReduce框架的海量数据相似性连接研究进展[J]. 计算机科学,2015,42(1):1-5.
- [2] BARAGLIA R, MORALES G D F, LUCCESE C. Document similarity self-joins with MapReduce[C]//IEEE international conference on data mining. Sydney, NSW, Australia: IEEE, 2010: 731-736.
- [3] SILVA Y N, REED J M. Exploiting MapReduce-based similarity joins[C]//Proceedings of the 2012 ACM SIGMOD international conference on management of data. Scottsdale, Arizona, USA: ACM, 2012: 693-696.
- [4] KIM Y H, SHIM K. Parallel top-k similarity joins algorithms using MapReduce[C]//Proceedings of the 2012 ACM SIGMOD international conference on management of data. [s. l.]: IEEE, 2012: 510-521.
- [5] LU Wei, SHEN Yanyan, CHEN Su, et al. Efficient processing of K nearest neighbor joins using MapReduce[J]. Proceedings of the VLDB Endowment, 2012, 5(10): 1016-1027.
- [6] YOKOYAMA T, ISHIKAWA Y, YU S. Processing all k-nearest neighbor queries in Hadoop[C]//International conference on web-age information management. [s. l.]: [s. n.], 2012: 346-351.
- [7] 戴健,丁治明.基于MapReduce快速kNN Joins方法[J]. 计算机学报, 2015, 38(1): 99-108.
- [8] MA Youzhong, MENG Xiaofeng, WANG Shaoya. Parallel similarity joins on massive high-dimensional data using MapReduce[J]. Concurrency and Computation: Practice & Experience, 2016, 28(1): 166-183.
- [9] 徐媛媛,陈华辉.基于MapReduce的增量式数据集的相似性连接[J]. 计算机应用研究, 2014, 31(11): 3369-3374.
- [10] 刘雪莉,王宏志,李建中,等.基于实体的相似性连接算法[J]. 软件学报, 2015, 26(6): 1421-1437.
- [11] 刘艳,郝忠孝.基于 Δ -tree的高维数据相似连接算法[J]. 计算机科学, 2011, 38(10): 157-160.
- [12] 周健雯,李聪聪,熊赞,等.一种基于R*树的自相似性连接算法[J]. 计算机应用与软件, 2014, 31(8): 50-53.
- [13] LUO Wuman, TAN Haoyu, MAO Huajian, et al. Efficient similarity joins on massive high-dimensional datasets using MapReduce[C]//13th international conference on mobile data management. Bengaluru, Karnataka, India: IEEE, 2012: 1-10.
- [14] 李桂玲,王元珍,杨林权,等.基于SAX的时间序列相似性度量方法[J]. 计算机应用研究, 2012, 29(3): 893-896.
- [15] KEOGH E, CHAKRABARTI K, PAZZANI M, et al. Dimensionality reduction for fast similarity search in large time series databases[J]. Knowledge and Information Systems, 2001, 3(3): 263-286.
- [16] LIN J, KEOGH E, LONARDI S, et al. A symbolic representation of time series, with implications for streaming algorithms[C]//Proceedings of the 8th ACM SIGMOD workshop on research issues in data mining and knowledge discovery. San Diego, California: ACM, 2003: 2-11.