·先进计算· 文章编号：1000-3428(2016)01-0018-07 文献标识码：A 中图分类号：TP391

基于索引的内存相似性连接算法

董明秀[a,b]，王 鹏[a,b]，汪 洋[a,b]，李秋虹[a,b]，汪 卫[a,b]

（复旦大学 a. 计算机科学技术学院；b. 上海市数据科学重点实验室，上海 201203）

摘 要： 在传统的相似性连接算法中，精确计算和分区阶段互相独立，精确计算时需要对每个分区中的所有数据进行两两比较，计算量较大。针对该问题，设计一种新的内存索引——距离树，并在其基础上提出两结构内存相似性连接算法。根据数据的潜在分布将其分发到不同的分区中，保证具有一定相似度的数据对分配在同个或相邻的分区内，同时通过树节点之间的位置信息保存分区阶段的计算结果，使精确计算阶段仅需对每个分区中相邻的叶节点数据进行比较计算。实验结果表明，与 TOUCH 算法相比，基于距离树的算法可使运行速度提高 2 倍～3 倍，并具有更好的可扩展性。

关键词： 相似性连接；磁盘；查询；内存；索引；分区

中文引用格式： 董明秀，王 鹏，汪 洋，等. 基于索引的内存相似性连接算法[J]. 计算机工程，2016，42(1)：18-24，30.

英文引用格式： Dong Mingxiu，Wang Peng，Wang Yang，et al. Memory Similarity Join Algorithm Based on Index[J]. Computer Engineering，2016，42(1)：18-24，30.

Memory Similarity Join Algorithm Based on Index

DONG Mingxiu[a,b]，WANG Peng[a,b]，WANG Yang[a,b]，LI Qiuhong[a,b]，WANG Wei[a,b]

（a. School of Computer Science；b. Shanghai Key Laboratory of Data Science，
Fudan University，Shanghai 201203，China）

【Abstract】 In traditional similarity join algorithms，data partition and refined calculation are isolated. During the refined calculation phase，all pairs of data in the same partition need to be compared with each other which leads to a large number of comparison computations. In order to solve this problem，this paper designs a new memory index：DistanceTree，and proposes an in-memory similarity join algorithm based on it. This algorithm distributes data into different partitions according to the potential distribution of data，ensures the data with same similarity to the same or adjacent partitions，and saves the calculation results of partition phase through the tree node location information. By leveraging the calculation result，only pairs of data in the same or adjacent leaf nodes need to be compared. Experimental results show that similarity join algorithm based on DistanceTree is 2 times ～ 3 times more efficient than TOUCH algorithm and also is more scalable.

【Key words】 similarity join；disk；query；memory；index；partition

DOI： 10.3969/j.issn.1000-3428.2016.01.004

1 概述

相似性连接是指在给定的数据集中找到指定相似性函数度量下所有相似值大于用户给定阈值的数据对。相似性连接问题在很多领域的应用中都是基本问题。在地理学应用中，相似性连接可用于检测地理特征的碰撞或临近度[1]，如地标、房屋、道路等；在医学影像应用中，相似性连接可用于检测哪些癌细胞的直接距离小于某个阈值[2]；在网页去重[3]和剽窃检测[4]等应用中，相似性连接也有广泛的应用。由于上述应用都需要处理持续增长的海量数据，因此相似性连接的可扩展性变得越来越重要。

但是，针对无序且没有索引的数据集，相似性连接的计算成本非常大，由于目前数据规模增长非常快，因此在很多科学应用中，相似性连接是一个瓶颈，这将阻止它们发展到更大的规模。除了数据量

基金项目： 国家自然科学基金资助项目(61103009)；上海市科委大数据专项基金资助项目(13511504800)。

作者简介： 董明秀(1988－)，女，硕士研究生，主研方向为时间序列相关性查询、分布式计算；王 鹏，副教授、博士研究生；汪 洋、李秋虹，博士研究生；汪 卫，教授、博士研究生。

收稿日期： 2014-12-19 **修回日期：** 2015-03-10 **E-mail：** mdong12@fudan.edu.cn

的快速增长,实际应用中的数据模型日益复杂和密集,以及数据维度的提高也增加了计算复杂度。

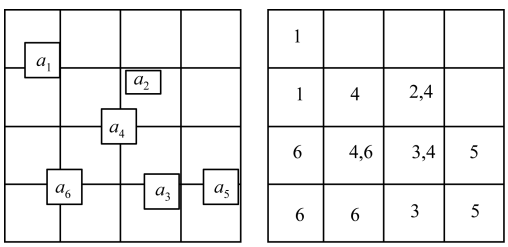
现有研究主要关注基于磁盘的相似性连接算法,在内存连接计算方面缺乏有效性和可扩展性。暴力算法,也就是对数据集中任意 2 条数据记录进行比较,计算成本会随着数据个数呈指数性增长,其对于现实数据不可行。研究表明,在相似性连接过程中可采用剪枝策略,其中的数据分区-精算是一种典型的两阶段剪枝计算模型。目前已有的相关方法包括 TOUCH[5]、PBSM[6]以及文献[7-8]算法等,但其中数据分区和数据精确计算都是 2 个独立的阶段,在精算阶段仍然会产生很多不必要的计算。

针对上述问题,本文设计一种新型索引结构——距离树,并给出基于该结构的两阶段内存相似性连接算法。根据数据的潜在分布对数据进行分区,只有同一或相邻分区内的数据之间才有可能存在相似性。在同一分区内部,通过树节点之间的位置信息保存分区阶段的计算结果。在精确计算阶段,距离树不需要对分区中所有的数据进行两两比较,只有相邻的节点之间需要进行比较计算,从而减少了计算量。

2　相关工作

相似性连接问题具有非常广泛的应用,但目前的相关技术只有少数是基于内存设计的,研究者更关注基于磁盘的相似性连接算法。然而,随着内存的不断扩大,基于磁盘的算法也可以应用到内存中。下面讨论的本文的 2 个主要对比算法都是可以基于内存实现的。

目前很多的相似性连接技术通常会先把 2 个数据集各自分成几个分区,然后在内存中对相邻的2 个分区进行相似连接计算。PBSM 就是其中一种高效的算法。如图 1 所示,PBSM 首先把整个数据空间划分成均匀的单元格,对于数据集 A 中的每条数据,需要把它分发到和它有交叠的所有单元格 c_A 中,数据集 B 中的数据按照相同方式分发到单元格 c_B 中。当分发结束之后,所有在相同位置的单元格 c_A 和 c_B 需要进行两两比较。



(a)分区　　(b)分发

图 1　PBSM 算法的分区与分发

由图 1 可见,当把数据分发到不同分区中时,很多数据需要同时分发到几个分区中,因此,这些数据需要被计算多次而且会产生很多重复的结果。

TOUCH 算法也是分成 2 个阶段:分区阶段和精算阶段。该算法主要关注的是分区阶段,在这个阶段,首先根据第一个输入数据集 A 构建一棵 R 树[9] R_A,对于数据集 B 中的数据并不构建索引,遍历 B 中的每条数据 x_{b_i}并把它插入 R_A 中,根据 x_{b_i}与 R_A 每一层交叠的最小边界矩形个数来确定 x_{b_i}的最终位置。TOUCH 算法只对一个数据集构建索引,不仅有效地减少了计算量和内存使用量,而且还避免了 PBSM 算法中产生的重复计算。然而 TOUCH 算法使用了 R 树作为索引结构,一方面 R 树维度灾难的性质使算法不能处理维度很高的数据,另一方面 R 树中的最小边界矩形是相交的,因此,在精算阶段会有大量的计算产生。最重要的是,TOUCH 算法并没有对精算阶段做任何优化,在分区阶段结束之后,分到同一个分区内部所有数据的地位是相同的,也就是分区阶段的计算仅仅用于分区,分区结束之后,之前的计算对于同一个分区的数据已经没有作用,同一个分区内的任意 2 条数据需要进行两两比较计算,这会产生大量的额外计算。

本文提出的基于距离树的算法也是由分区阶段和精算阶段组成的。但该算法同时关注了 2 个阶段,在分区阶段结束之后,分区阶段的计算会同时保留在每个分区内部,也就是在每个分区内部数据的地位是不同的,利用保存下来的计算,精算阶段只需要对每个分区内部的小部分数据进行两两比较计算,大幅减少了计算量。

3　预备知识

3.1　度量距离

定义 1(度量距离)[10]　一个数据集 D 上的度量是一个函数 $d:D\times D\rightarrow\mathbf{R}$($\mathbf{R}$ 表示实数集),对于 D 中的任意数据 x,y,z,这个函数需要满足以下条件。

(1)非负性:$d(x,y)\geqslant 0$;

(2)巧合定律:如果 $x=y$,$d(x,y)=0$;

(3)对称性:$d(x,y)=d(y,x)$;

(4)三角不等式:$d(x,z)\leqslant d(x,y)+d(y,z)$。

本文算法可以使用任意度量距离作为度量函数,度量距离就是相似性度量函数。根据度量距离的性质,本文主要利用三角不等式性质来避免大量不必要的计算。

3.2　问题定义

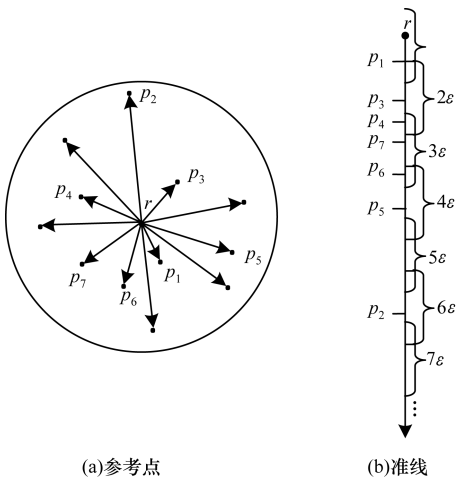
定义 2(相似性连接)　假设 D 包含了所要研究的所有数据,给定一个距离度量函数 $d:D\times D\rightarrow\mathbf{R}$,一个距离阈值 ε,本文研究的问题就是找到所有的数

据对$(x_i,x_j)\in D\times D$满足$d(x_i,x_j)\leqslant\varepsilon,i\neq j$。

相似性连接技术可以应用到不同的数据集和度量函数中。本文使用了一个高维数据集D和欧式距离来说明相似性连接问题以及本文算法的有效性。欧式距离度量函数用d表示。本文主要讨论基于距离树的自相似连接应用,但距离树也可以应用到2个不同数据集的相似连接应用。

4 距离树

本文提出一种适用于高维数据相似性连接的新型树形索引结构。距离树的基本原理是基于参考点的特性。给定数据集$X=\{x_1,x_2,\cdots,x_n\}$和参考点r,计算X中的每条数据和r的欧式距离,并且把这些距离按照升序排序。在几何上,可以看作把X中的每条数据绕着r旋转然后映射到一条准线上,如图2所示。对于数据p_i和p_j,如果它们的映射距离超过了ε,那么它们之间真实的欧式距离$d(p_i,p_j)>\varepsilon$。因此,这种映射方式可以精简许多非必要的比较计算。



(a)参考点 (b)准线

图2 参考点和准线

本文扩展了参考点机制:(1)使用多个参考点,距离树的每一层使用一个参考点;(2)把每条准线分裂成不相交的ε-长度分段;(3)把通过不同参考点生成的片段组织成一颗树结构。

4.1 距离树结构

距离树在构造之前已知3个参数:

(1)一组中心点$R=\{r_1,r_2,\cdots,r_n\}$,$n>Height_{\text{Tree}}$,距离树从根节点开始每增加一层高度需要从R中取出一个点当作该层的中心点。

(2)叶节点能容纳的最大数据量ψ,如果叶节点中插入的数据个数大于等于ψ,该叶节点会变成内部节点同时会分裂成多个叶节点,只有叶节点能存储数据。

(3)用户指定的相似度阈值ε。

距离树构造完成后如图3所示。

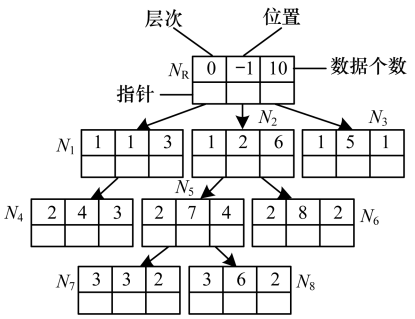


图3 距离树

在图3中中包含2种节点:内部节点和叶节点。每个节点包含如下信息:

(1)节点在树中的层次L。根节点的层次是0,其他节点的层次是它的高度。

(2)以当前节点为根节点的数据记录个数C。

(3)距离树中叶节点的最大数据容量ψ,一个叶节点最多包含ψ个指向记录的指针。每个内部节点包含一个或多个指向它的子节点的指针。

(4)位置信息pos,代表该节点距离其父节点的中心点的距离和ε倍数关系。在该节点内部的任意数据记录x满足$pos\cdot\varepsilon\leqslant d(x,r_L)<(pos+1)\cdot\varepsilon$,其中,$L$是节点所在的层次,$r_L$是第$L$层的参考点,对应的是落在图2(b)准线的那一段中。

4.2 距离树构建

通过逐条插入的方式构建距离树,插入数据实际上就是把每条数据分发到某个叶节点的过程。插入每条数据x时,都要从根节点N_{R}开始直到某个叶节点结束。

如果N_{R}是一个叶节点,笔者简单地把x放在N_{R}中,记录数C加1,如果此时$C=\psi$,需要分裂N_{R}并且创建一些新的子节点。特别地,对于每个需要分裂的节点,本文用节点所在的层次的参考点r_i按照如下步骤分裂第i层中的节点:

(1)计算x和r_i的距离$d(x,r_i)$。

(2)计算满足$d(x,r_i)\in[j\cdot\varepsilon,(j+1)\cdot\varepsilon)$$(j\geqslant0)$的值$j$,如果不存在$pos=j$的子节点就创建一个位置信息为$j$的子节点,层次$L=i+1$,并把$x$分配到这个节点,数据个数$C=1$;如果存在$pos=j$的子节点则直接把$x$分发到这个节点,$C$自增1。检查获得$x$的节点中数据个数$C$,如果$C=\psi$,则从步骤(1)开始对这个节点进行分裂。

如果N_{R}是一个内部节点,查找是否存在位置信息pos满足$d(x,r_i)\in[pos\cdot\varepsilon,(pos+1)\cdot\varepsilon)$的子节点。如果存在,将$x$分发到这个节点,并判断节点是否要分裂;如果不存在,则创建一个新节点。

下面用一个例子来说明如何构建距离树。设定输入数据集是 $X=\{(1,1),(2,2),(3,4),(3,1),(2,0)\}$,中心点集合为 $R=\{(0,1),(0,0),(1,0),(3,2)\}$,叶节点最大数据容量 $\psi=3$,相似度阈值 $\varepsilon=2$。首先初始化根节点,层次 $L=0$,$pos=-1$,$C=0$。然后依次遍历 X 的数据,当 $C<\psi$ 时,直接将数据放入根节点中,因此,(1,1),(2,2),(3,4)3 条数据会直接分配到根节点中,当 $C=3$ 时,需要分裂根节点。按顺序计算 $d((1,1),(0,1))=1$,计算满足的 $j=0$,查找根节点的所有子节点,由于不存在 $pos=0$ 的子节点,因此创建一个子节点并初始化 $L=1$,$pos=0$,$C=1$,同时将(1,1)分发到该节点,数据(2,2),(3,4)按相同的方式进行计算和分配可以得到图 4 的结果。继续遍历剩下数据,此时的根节点为内部节点,因此,对于数据(3,1),首先计算得出 $d((3,1),(0,1))=3$ 和满足条件的 $j=1$,查找根节点的子节点,节点 N_2 的 $pos=1$ 满足条件,因此,将(3,1)插入到 N_2 中,并且 C 自增 1。继续按照上述方法插入数据,最后得到图 5 所示的完整的距离树。

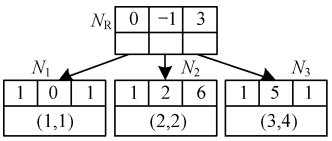


图 4　距离树构建过程

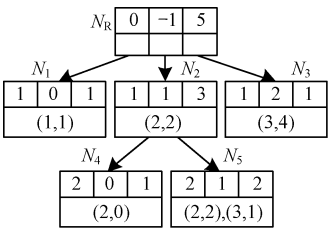


图 5　完整距离树

4.3　距离树中节点之间的相似关系

利用距离树中节点之间的位置关系可以精简同一分区中大量非必要计算。距离树中任意 2 条数据有 2 种位置关系,一种是在同一个节点中,另一种是在 2 个不同的节点中。显然,在同一个节点内的数据相似度很大,需要进行两两精确计算。对于节点之间的数据,根据以下定理,每个叶节点只需要和少量的几个临近叶节点中的数据进行精确计算就能保证计算结果的准确性。

定理　给定在 L_1 层和 L_2 层的 2 个叶节点 N_1 和 N_2,假设 $L_1\geqslant L_2$。从根节点 N_R 到 N_1 和 N_2 经过的节点的位置序列分别为 $\{p_1,p_2,\cdots,p_{L_1}\}$ 和 $\{q_1,q_2,\cdots,q_{L_2}\}$。如果对于任意的 $i\leqslant L_2$,都有 $p_i+2<q_i$ 或者 $p_i>q_i+2$,可以得出 N_1 中的任何数据和 N_2 中的任何数据之间的距离大于 ε。

证明:假设从根节点到 L_2 层的中心点序列为 $\{r_1,r_2,\cdots,r_{L_2}\}$,从 N_R 到 N_1 和 N_2 经过的节点序列分别为 $\{N_{p_1},N_{p2},\cdots,N_{p_{L_1}}\}$ 和 $\{N_{q_1},N_{q2},\cdots,N_{q_{L_2}}\}$,这里使用欧式距离作为度量距离。根据距离树的性质,可以得出在第 i 层的任意节点 N_{p_i} 和 N_{q_i} 有如下性质:

$$\forall x_i\in Np_i,p_i\times\varepsilon\leqslant d(x_i,r_i)<(p_i+1)\times\varepsilon$$

$$\forall y_i\in Nq_i,q_i\times\varepsilon\leqslant d(y_i,r_i)<(q_i+1)\times\varepsilon$$

根据三角不等式可以得出,如果 $p_i+2<q_i$ 或 $p_i>q_i+2$,则有:

$$d(x_i,y_i)>|d(x_i,r_i)-d(y_i,r_i)|$$
$$>|p_i-q_i-1|\times\varepsilon>\varepsilon$$

这里可以给出 2 个例子。如图 3 所示,从 N_R 到 N_7 和 N_3 经过的节点序列的位置信息为{2,7,3}和{5}。当 $i=1$ 时,$p_i+2<q_i$,可以安全的剪掉 N_7 和 N_3 中的数据之间的比较。然而,对从根节点到达 N_7 和 N_6所经过的节点位置序列为{2,7,3}和{2,8},可以发现,对于任意的 $i\leqslant 2$,不存在 $p_i+2<q_i$ 或者 $p_i>q_i+2$,因此,在 N_7 和 N_3 中的数据需要进行两两比较。

5　基于距离树的相似性连接算法

基于距离树的相似连接算法主要有 3 个阶段。在第 1 个阶段,使用主成分分析从数据集中选取几个参考点。第 2 阶段,根据选取的参考点和用户指定的阈值 ε,构建距离树。在第 3 个阶段,根据 4.3 节定理,精确计算每个分区中相似数据。算法描述如算法 1 所示,其中,第 4 行完成距离树构建,第 5 行~第 10 行遍历距离树中的叶节点,对于每个叶节点 N_{Leaf},第 7 行按照 4.3 节定理查找其所有的邻近节点 NN_{Leaf},第 8 行计算计算 N_{Leaf} 和 NN_{Leaf} 中数据对的欧氏距离,如果距离小于给定阈值则将结果对放入结果集合中。

算法 1　*DistanceTree*()　//距离树

输入　高维数据集 A,叶节点最大数据容量 ψ,相似性阈值 ε

输出　$Res=\{(x_i,x_j)\mid d(x_i,x_j)<\varepsilon,x_i,x_j\in A,i\neq j\}$

```
1. 使用 PCA 算法选取一组中心点 R = {r1, r2, ⋯, rn};
2. Res = {};
3. //构建距离树并返回根节点
4. Root = ConstructDistanceTree(A, R, ψ, ε);
5. foreach N_Leaf in DistanceTree do
6.     //依据 4.3 节定理
7.     查找 N_Leaf 的临近叶节点 NN_Leaf;
8.     精确计算 N_Leaf 和 NN_Leaf 中的数据对;
9.     if 距离小于 ε then
10.        Res.add(数据对);
11. Return Res
```

5.1 中心点选取

选取一组合适的参考点把给定的数据集划分成几个交互尽可能少的分区,可以减少大量的比较计算。在对数据进行分区时,最直观的想法是找到k个参考点$R=\{r_1,r_2,\cdots,r_k\}$,以这k个参考点为中心形成k个分区。分区k_i中的任意一条数据记录x,满足$d(x,r_i)\leqslant d(x,r_j)$,其中$1\leqslant i,j\leqslant k,i\neq j$。这就是 K-Center 问题,一个 NP-完全问题[11]。在实际应用中,这个算法需要尽可能减少 2 个不同分区之间相似的数据对。本文使用主成分分析(Principal Component Analysis,PCA)[12]算法来选取中心点,实验证明,PCA 是一种有效的分区方法。

主成分分析是将多个变量通过线性变换以选出较少个数重要变量的一种多元统计分析方法。在很多情形下,变量之间是有一定的相关关系的,当 2 个变量之间有一定相关关系时,可以解释为这 2 个变量反映的信息有一定的重叠。主成分分析是对于原先提出的所有变量,将重复的变量(关系紧密的变量)删去多余,建立尽可能少的新变量,使得这些新变量是两两不相关的,而且这些新变量在反映的信息方面尽可能保持原有的信息。

5.2 距离树构建

在构建距离树之前,除了已经获得的中心点集还需要指定距离度量函数d和给定 2 个参数。第 1 个参数是用户指定的相似度阈值ε,第 2 个参数是每个叶节点的最大容量ψ,这个参数可以根据输入数据量的大小做适当调整。之后按照 4.2 节描述的方法逐条插入数据构建距离树。算法 2 给出了构建距离树的伪代码。

算法 2 *ConstructDistanceTree*(A,R,ψ,ε)

//构建距离树

输出 距离树的根节点

```
1. 初始化根节点 N_R, N_R.pos = -1, N_R.C = 0, N_R.L = 0;
2. foreach x_i in A do
3.         appendData(x_i, N_R);
4. return N_R
```

在构建距离树之前,会把数据集A,ψ和ε这 3 个参数传进 *ConstructDistanceTree* 函数中,遍历A中的每条数据,并调用算法 3 中的 *appendData*( )方法往每个节点中插入数据。第 1 个参数是需要插入的数据,第 2 个参数是目标节点。第 1 行表示当节点中数据量小于ψ的时候直接将数据分发到节点N,当插入结束后,第 3 行检查节点是否需要分裂,如果需要分裂,取出N所在的层次的中心点r_{N_L}并计算N中所有的数据和r_{N_L}的距离,同时查找满足条件的子节点,如果子节点存在将数据插入到子节点否则创建新的子节点。算法第 11 行表示,此时的根节点为内部节点不能直接插入数据,需要查找满足条件的子节点,并把数据插入到子节点中。构建距离树的时间复杂度为$O(n)$,其中,n为输入数据条数。

算法 3 *appendData*(x_i,N)

```
1. if(N.C < ψ && N.children = 0) then
2.     分发 x_i 到 N; N.C + +;
3.     if(N.C = ψ) then
4.         foreach x_j in N do
5.             计算 distance = d(x_j, r_{N_L});
6.             查找 distance 对应的子节点 N_child;
7.             if(N_child 存在) then
8.                 appendData(x_j, N_child);
9.             else
10.                创建子节点;
11.    else
12.        计算 d(x_i, r_{N_L}) 并查找子节点 N_child;
13.        if(N_child 存在) then
14.        appendData(x_i, N_child)
15  else 创建子节点
```

5.3 相似连接

距离树构建完成后,按照先序序列依次访问所有的叶节点。对于访问到的每个叶节点N_i按下面的步骤进行精确计算:

(1)对N_i内部的所有数据进行两两精确计算。

(2)根据 4.3 节定理,找到所有满足相似条件的N_i的临近叶节点。把N_i中的数据和临近节点中的数据进行比较。

在查找N_i的临近节点时,为了不重复计算某些数据对,只需要查找先序列顺序在N_i后面的叶节点。查找临近叶节点时间复杂度为$O(\lg n)$,精确计算的时间复杂度为$K\cdot O(\psi^2)$,其中,K表示每个叶节点的相邻叶节点个数。

6 实验与结果分析

本节主要评估距离树的性能和算法中某些参数对算法效率的影响。将本文算法与 TOUCH 以及 PBSM 算法进行对比,尽管 PBSM 算法是基于磁盘设计的,但是现在内存空间越来越大,因此,用基于内存的方式实现 PBSM 算法。使用 PBSM 算法最高效的配置 PBSM-500,即把每一维度划分成 500 个单元格,然而这种配置会产生大量的副本也就会产生大量的额外计算。对于 TOUCH 算法也基于原文中的讨论使用它最高效的配置,设置 *fanout* = 2,分区个数为 1 024。距离树在构建之前需要获取 3 个参数:第 1 个是中心点集合,本文在 Matlab 中使用 PCA 算法选取中心点,Matlab 执行 PCA 算法非常高效,时间基本可以忽略不计,因此,在本文中,对于选取参考点的算法不做讨论;第 2 个参数是叶节点容量ψ,叶节点的容量是根据输入数据量动态调整的,6.2.4 节讨论ψ对算法的影响;6.2.5 节讨论第 3 个

参数相似性阈值 ε 对距离树执行时间的影响。

6.1　实验配置和数据集

本文的实验环境是一台 16 GB 内存,i7-3770CPU 的单台 Window 操作系统物理机。本文使用以下 2 组真实数据集来评估距离树:

(1) UCI,这个数据集来自机器学习仓库[13]。UCI 包含了采集于 6 个不同区域的个化学检测平台的 18 000 条时间序列数据。本文对这些数据进行了预处理,生成了 5×10^5 条 1 000 维的时间序列。

(2) Wiki,这个数据集是 Wikimedia 项目[14]中的页面访问统计信息。它包含了从 2007 年 – 2014 年大量维基百科页面的访问量。把某个页面在 1 h 内的访问量当作一个点,选取 5×10^5 个页面 1 000 h 内的访问量。

6.2　实验评估

6.2.1　运行时间评估

本节从 UCI 和 Wiki 数据集中分别取出不同大小的子数据集来评估每个算法的有效性和可扩展性,这 2 个指标是相似性连接最重要的 2 个方面。图 6 和图 7 分别展示了本文提出的距离树和其他 2 种算法在 UCI 和 Wiki 数据集上的运行时间,其中数据量即为数据条数。相似性阈值设置为 3。

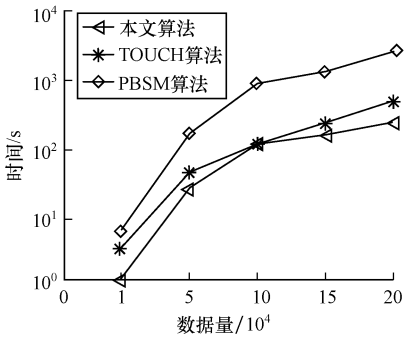


图 6　UCI 数据集运行时间

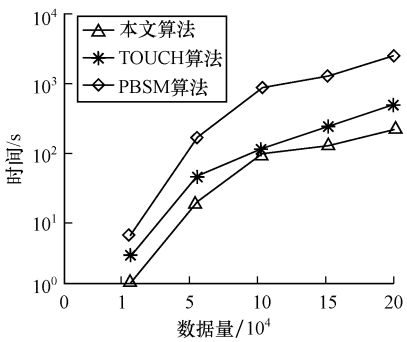


图 7　Wiki 数据集运行时间

尽管 TOUCH 和 PBSM 算法随着数据集的增长都有很好的扩展性,但是本文算法比 TOUCH 算法快 2 倍 ~ 3 倍,比 PBSM 算法快 10 倍 ~ 12 倍。在第 1 阶段,本文算法和 TOUCH 算法都把数据分成几个分区,运行时间的差距主要是在精确计算阶段。当结束分区时,TOUCH 算法需要把每个分区中的任意 2 条数据进行比较。而本文算法通过节点之间的位置关系保存了分区阶段的计算,在每个分区中,只有临近的节点之间的数据需要做比较。

6.2.2　计算量比较

计算量的评估使用的是和 6.2.1 节相同的数据集,由于 UCI 和 Wiki 数据集之间的比较次数相近,因此本文只给出 3 种算法在 UCI 上运行时的比较次数。相似性阈值设置为 3。如图 8 所示,其中,数据量即为数据条数。由图 8 可见,由于有效利用了分区阶段的计算,距离树中进行精确计算的量远小于 TOUCH 和 PBSM 算法,因此距离树的运行时间更快,基于距离树的相似性连接具有更高的效率和更好的可扩展性。

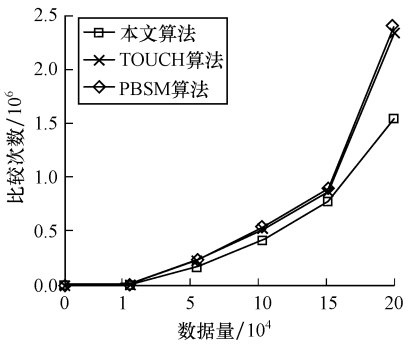


图 8　UCI 数据集计算量

6.2.3　数据分区性能评估

本文算法和 TOUCH 算法是基于树型结构的算法,建树过程可以看作是数据分区的过程。PBSM 算法的分区过程和距离树差别比较大,而且通过 6.2.1 节和 6.2.2 节的实验结果发现,PBSM 算法的性能远低于本文算法,因此,本节主要对比本文算法和 TOUCH 算法的分区阶段性能。

将相似性阈值设定为 3,本文算法和 TOUCH 算法的分区时间比较如图 9 所示,可以看出,在分区阶段,本文算法比 TOUCH 算法快 5 倍以上。

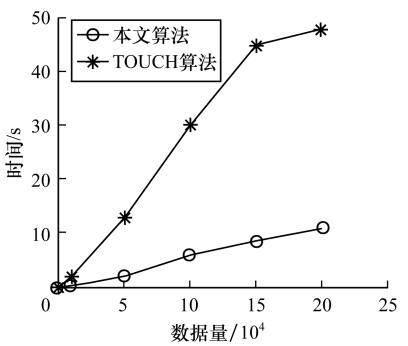


图 9　数据分区时间

距离树的建树时间复杂度仅为 $O(n)$。TOUCH

算法在构建 R 树的过程中使用了 STR[15] 分裂最小边界矩形(MBR),STR 需要对数据的每一维进行排序,仅分裂 MBR 的时间复杂度至少为 $K \cdot n\lg(n)$(K 为数据维度,n 为数据总个数)。当数据维度很大时,TOUCH 算法的性能会逐渐下降,而且在 R 树中查找满足条件的中间节点的时间复杂度为 $O(n\lg n)$,因此,分区阶段本文算法的性能远高于 TOUCH 算法。

6.2.4 叶节点容量变化

为了讨论叶节点容量对算法性能的影响,本文选取 2×10^5 条 UCI 数据进行实验,相似性阈值设为 3。图 10 展示了叶节点容量的变化对算法性能的影响。本文选取了原数据集 1‰ ~ 1% 之间的 6 个点作为叶节点的容量。可以发现,当叶节点最大数据容量为原数据集的 1‰ ~ 2‰之间时,距离树的效率最高。如果叶节点容量过小,则在建树的过程中就会产生大量的节点,树的高度也会随着增高,那么在查找可能相似的节点时需要访问大量的节点,造成时间的消耗;如果叶节点容量过大,虽然树的高度会降低,但是单个节点中包含的数据量过多,在精算阶段,数据计算量会急剧增长,造成大量时间消耗。

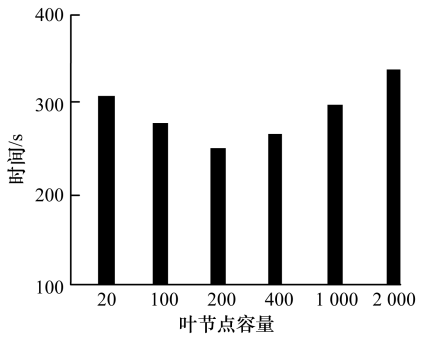


图 10 距离树叶节点容量变化

6.2.5 相似性阈值 ε 变化

本节从 UCI 数据集中选取 2×10^5 条数据作为输入数据集,由于 2 个数据集的运行时间变化不大,本节只使用 UCI 数据集。图 11 展示了不同的相似性阈值对于 3 种算法执行效率的影响。可以看出随着 ε 的增大,3 种算法的运行时间都显著增加,这是因为 ε 增大会产生更多的数据计算量。可以发现 PBSM 算法的时间增长非常显著,这是因为 ε 的增大导致了更大量的数据副本产生。距离树和 TOUCH 的执行时间随着 ε 的变化基本呈线性增长趋势。从图 11 可以看出,距离树的运行时间增长速度大于 TOUCH,这是因为 ε 越大,叶节点内的数据量就越大,因此数据比较量增长很快。但是整体情况下,在 ε 较小的时候,距离树还是比 TOUCH 要高效。对于相似性连接问题来说,只有小阈值的分析才是有意义的,通常不会把相似性阈值设置得很大。

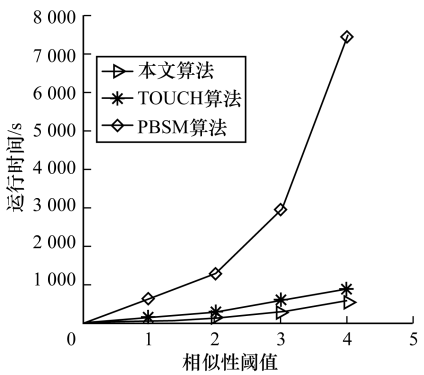


图 11 变化相似性阈值对算法的影响

7 结束语

随着物理设备性能的不断提升,单台物理机的内存也越来越大,内存算法成为很多应用的瓶颈。本文研究内存相似性连接算法,提出一种新型的索引结构——距离树。相似性连接问题面临的主要挑战是随着数据量的增长,计算量指数性增长,这阻止了很多相似性连接应用的扩展。基于距离树的相似性连接算法从大量减少计算量的角度出发,由 2 个阶段组成:第 1 阶段构建距离树,在构建距离树的同时把数据均匀地分发到不同节点中并形成分区;第 2 阶段对每个分区中的数据进行精确计算。不同于以往算法的是,第 1 阶段和第 2 阶段是 2 个相互联系的阶段,在第 1 阶段形成分区的同时还通过节点位置信息保存了计算结果,第 2 阶段直接利用这些计算结果,花费少量的计算就能对每个分区中的数据进行精确计算。本文在 2 个真实数据集上应用基于距离树的相似性连接算法,实验结果证明,该算法具有较高的效率和可扩展性。

基于磁盘的相似性连接算法,其内存计算阶段同样可以利用距离树。随着数据量的极速膨胀,距离树也很容易设计扩展成基于 MapReduce[16] 分布式框架的算法,这些都是下一步的研究方向。

参考文献

[1] Ubell M. The Montage Extensible DataBlade Architecture[C]//Proceedings of ACM SIGMOD International Conference on Management of Data. Minneapolis, USA: ACM Press, 1994: 482-493.

[2] Wang Fusheng. A Data Model and Database for High-resolution Pathology Analytical Image Informatics[J]. Journal of Pathology Informatics, 2011, 2(1): 32-40.

[3] Henzinger M R. Finding Near-duplicate Web Pages: A Large-scale Evaluation of Algorithms[C]//Proceedings of ACM SIGIR Conference on Research and Development in Information Retrieval. Seattle, USA: ACM Press, 2006: 284-191.

[4] Hoad T C. Methods for Identifying Versioned and Plagiarized Documents[J]. Journal of the American Society for Information Science and Technology, 2003, 54(3): 203-215.

(下转第 30 页)

实际处理的三维水声数据,对传统算法进行改进,提出采用三维滤波窗口的模糊中值滤波和考虑相邻切片小波系数相关性和小波系数尺度间相关性的小波软阈值滤波算法,然后根据复合噪声模型,利用改进后的模糊中值滤波和小波软阈值滤波,分层去除三维水声数据中的加性和乘性噪声。实验结果表明,与文献[12]算法相比,本文算法具有更好的降噪效果,能够在保留细节的同时抑制三维水声数据中加性和乘性噪声,并增强数据对比度,为后续的数据分割和识别等操作提供技术支持。

参考文献

[1] Thijssen J M. Ultrasonic Speckle Formation, Analysis and Processing Applied to Tissue Characterization[J]. Pattern Recognition Letters,2003,24(2):659-675.

[2] 喻 琪,夏顺仁,从卫华,等. 基于小波系数相关性和模糊理论的声纳图像处理[J]. 浙江大学学报:工学版,2008,12(42):2151-2155.

[3] Gonzalez R C,Woods R E. 数字图象处理[M]. 阮秋琦,阮宇智,译. 3 版. 北京:电子工业出版社,2011.

[4] Lopes A,Nezry E,Touzi R,et al. Maximum a Posteriori Speckle Filtering and First Order Texture Models in SAR Images[C]//Proceedings of International Geoscience and Remote Sensing Symposium. Washington D. C., USA:IEEE Press,1990:2409-2412.

[5] Frost V S,Stiles J A,Shanmugan K S,et al. A Model for Radar Images and Its Application to Adaptive Digital Filtering of Multiplicative Noise[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1982, 3(2):157-166.

[6] Kuan D T,Sawchuk A A,Strand T C,et al. Adaptive Noise Smoothing Filter for Images with Signal-dependent Noise[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence,1982,3(2):165-177.

[7] 王 芳,满益云. 基于模糊中值滤波的椒盐噪声去噪方法[J]. 模糊系统与数学,2012,2(1):166-174.

[8] Donohod D L,Johnstoneim I M. Ideal Spatial Adaptation by Wavelet Shrinkage[J]. Biometrika,1994,81(3):425-455.

[9] Donohod D L. De-noising by Soft-thresholding[J]. IEEE Transactions on Information Theory,1995,41(3):613-627.

[10] 李昌顺,杨 浩,裴 蕾. 基于高密度离散小波变换的改进图像降噪方法[J]. 计算机工程,2012,38(1):211-214.

[11] 郭德全,杨红雨,刘东权. 采用引导滤波的超声纹理补偿图像优化[J]. 计算机辅助设计与图形学学报,2014,26(1):40-46.

[12] 宋坤坡,夏顺仁,徐 清. 考虑小波系数相关性的超声图像降噪算法[J]. 浙江大学学报:工学版,2010,11(44):2203-2208.

[13] 周伟华,王 鑫,罗 斌. 基于双树复小波变换的相位保持 SAR 图像降噪[J]. 中国图象图形学报,2007,12(5):805-810.

[14] Yang X,Toh P S. Adapitve Fuzzy Multilevel Median Filter[J]. IEEE Transactions on Image Processing, 1995,5(4):680-682.

[15] Gupta S,Chauhan R C,Saxena S C. Locally Adaptive Wavelet Domain Bayesian Processor for Denoising Medical Ultrasound Images Using Speckle Modeling Based on Rayleigh Distribution[J]. Vision, Image and Signal Processing,2005,152(2):129-135.

[16] Hou Jianhua, Liu Xiangming, Xiong Chengyi, et al. Speckle Reduction Algorithm for Synthenic Aperture Radar Images Based on Bayesian Maximum a Posteriori Estimation in Wavelet Domain[J]. Optical Engineering, 2008,47(5).

[17] Jain A K. Fundamental of Digital Image Processing[M]. Upper Saddle River,USA:Prentice Hall,Inc.,1989.

[18] Kingsbury N. A Dual Tree Complex Wavelet Transform with Improved Orthogonality and Symmetry Properties[C]//Proceedings of IEEE International Conference on Image Processing. Washington D. C., USA: IEEE Press,2000:375-378.

编辑 陆燕菲

(上接第 24 页)

[5] Nobari S,Tauheed F,Heinis T. TOUCH:In-memory Spatial Join by Hierarchical Data-oriented Partitioning[C]//Proceedings of ACM SIGMOD International Conference on Management of Data. New York,USA:ACM Press,2013:701-712.

[6] Patel J M, DeWitt D J. Partition Based Spatial-merge Join[C]//Proceedings of ACM SIGMOD International Conference on Management of Data. New York, USA: ACM Press,1996:259-270.

[7] Ye Wang, Metwally A, Parthasarathy S. Scalable All-pairs Similarity Search in Metric Spaces[C]//Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York, USA:ACM Press,2013:829-837.

[8] 王晓晔. 时间序列数据挖掘中相似性和趋势预测的研究[D]. 天津:天津大学,2003.

[9] Guttman A. R-trees:A Dynamic Index Structure for Spatial Searching[C]//Proceedings of ACM SIGKDD International Conference on Management of Data. New York, USA:ACM Press,1984:47-57.

[10] Bryant V. Metric Spaces:Iteration and Application[M]. London,UK:Cambridge University Press,1985.

[11] Toussaint G T. A Simple Linear Algorithm for Intersecting Convex Polygons[J]. The Visual Computer,1985,1(2):118-123.

[12] Jolliffe I T. Principal Component Analysis[M]. 2nd ed. Berlin,Germany:Springer,2002.

[13] Aha D,Kibler D. Machine Learning Repository[EB/OL]. [2014-12-19]. http://archive.ics.uci.edu/ml/datasets.html.

[14] Mituzas D. Page View Statistics for Wikimedia Projects[EB/OL]. [2014-12-19]. http://dumps.wikimedia.org/other/pagecounts-raw/.

[15] Leutenegger S,Lopez M,Edgington J. STR:A Simple and Efficient Algorithm for R-Tree Packing[C]//Proceedings of ACM SIGMOD International Conference on Management of Data. Seattle,USA:ACM Press,1997:497-506.

[16] Dean J, Ghemawat S. Mapreduce: Simplified Data Processing on Large Clusters[C]//Proceedings of Conference on Symposium on Opearting Systems Design & Implementation. New York,USA:ACM Press,2004:10-21.

编辑 金胡考