ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.18, No.6, June 2007, pp.1419−1428
DOI: 10.1360/jos181419

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

基于最大间隙空间映射的高维数据索引技术*

王国仁[1,2+], 黄健美[1], 王 斌[1], 韩东红[1], 乔百友[1], 于 戈[1]

[1](东北大学 信息科学与工程学院,辽宁 沈阳 110004)

[2](东北大学 计算中心(网络中心),辽宁 沈阳 110004)

A High Dimensional Data Indexing Technique Based on Max Gap Space Mapping

WANG Guo-Ren[1,2+], HUANG Jian-Mei[1], WANG Bin[1], HAN Dong-Hong[1], QIAO Bai-You[1], YU Ge[1]

[1](College of Information Science and Engineering, Northeastern University, Shenyang 110004, China)

[2](Computing Center (Networking Center), Northeastern University, Shenyang 110004, China)

+ Corresponding author: Phn: +86-24-83681250, Fax: +86-24-23893138, E-mail: wanggr@mail.neu.edu.cn, http://mitt.neu.edu.cn

Wang GR, Huang JM, Wang B, Han DH, Qiao BY, Yu G. A high dimensional data indexing technique based on max gap space mapping. *Journal of Software*, 2007,18(6):1419−1428. http://www.jos.org.cn/1000-9825/18/1419.htm

Abstract: In the similarity query processing based on high dimensional indexing, the searching space is usually narrowed down by pruning the inactive subspaces which do not contain any query results. However, among the active subspaces, some of them do not contain any query results at all, those are called false active subspaces. It is obvious that the performance of query processing degrades in the presence of false active subspaces. The problem becomes seriously in the case of high dimensional data. The experiment in this paper shows that the number of accesses to false active subspaces increases as the dimensionality increases. In order to overcome the problem, a space mapping approach is proposed to reduce such unnecessary accesses. For a given query, it can be refined by filtering within its mapped space. A maximal gap space mapping strategy, MaxGapMapping, is proposed to improve the efficiency of the refinement processing. An index structure——MS-tree, the algorithms of construction, and query processing based on this refining method are designed and implemented. Finally, the performance of MS-tree is systemically compared with that of other competitors in terms of range queries based on a real data set.

Key words: high dimensional index; query refining; false active subspace

摘 要: 在基于高维索引技术的相似性查询处理中,通常通过过滤那些不包含任何查询结果的非活动子空间来不断缩减搜索空间.但是在活动子空间中,有些可能根本就不包含任何查询结果,这样的活动子空间被称为假活动子空间.显然,查询处理性能会随着假活动子空间访问次数的增加而下降.这一问题在高维数据情况下将会变得更加严重,实验显示出随着维数的增加,假活动子空间的访问次数也会增加.为了解决这一问题,提出了一种空间映射方法来减少这种不必要的访问.对于一个给定的查询,可以通过在映射空间内进一步精炼该查询来过滤假活动子空间.为

* Supported by the National Natural Science Foundation of China under Grant Nos.60273079, 60473074, 60573089 (国家自然科学基金); the National Basic Research Program of China under Grant No. 2006CB303103 (国家重点基础研究发展计划(973)); the National Research Foundation for the Doctoral Program of Higher Education of China under Grant No.DP0345710 (国家教育部博士点基金)

Received 2005-11-07; Accepted 2006-08-16

了提高映射空间内查询精炼的处理效率,提出了一个最大间隙空间映射策略——MaxGapMapping.基于这种映射方法,设计并实现了一种新的索引结构——MS-tree,给出了索引的构建算法和范围查询处理算法.最后对 MS-tree 及其他索引结构的性能进行了详细的比较和分析.

关键词: 高维索引;查询精炼;假活动子空间
中图法分类号: TP311 文献标识码: A

随着在各应用领域中视听数据源的不断增长,在大型数据库中快速处理基于内容的相似性查找变得越来越重要.为了加速高维空间中的相似性查找,常用的方法是设计一个高维索引来支持这种类型的查询[1].高维索引方法可以分为两大类:基于向量空间的索引结构和基于度量空间的索引结构.R-tree 及其变种[2−5]是前者的代表,它们是基于向量空间中的相对位置来管理数据.其他类型的索引结构,包括 VP-tree[6],MVP-tree[7],M-tree[8],MB$^+$-tree[9],Slim-tree[10],M+-tree[11],是基于度量空间的索引技术,它们基于数据对象间的相对距离来管理数据.

VP-tree[6]是第一种支持相似性查询的层次索引结构,它使用数据对象到代表点之间的相对距离和三角不等式来进行数据空间的过滤.因为 VP-tree 索引结构较小的扇出(因而索引的高度很高)而引起了大量的距离计算,从而极大地影响了它的查询性能.应该指出的是,在度量空间中距离计算是非常复杂且非常耗时的.为了克服上述问题,MVP-tree 索引结构[7]使用多个代表点,从而增加了索引的扇出,降低了索引的高度.VP-tree 和 MVP-tree 都是静态的基于度量空间的索引结构,它们采用一种自上而下的方法来构建.这就意味着,这些索引无法支持数据的更新和删除.

M-tree[8]是基于度量空间的动态索引结构的代表,它是一种页面结构的平衡树,采用自下而上的索引构造方法,具有节点提升和分裂机制.因此,它适合作为一种磁盘索引结构,并能处理数据的更新而无须重构整个索引.M-tree 第一个认识到了距离计算的高代价,因此,它将大多数距离预计算好并存储在索引当中.这样就可以避免很多距离的动态计算.但是,M-tree 的兄弟节点索引空间的重叠是一个非常值得注意的问题,因为它对查询处理的性能有着非常大的影响.为此,基于 M-tree 索引结构的基本思想,几种改进的索引技术被提了出来,例如 MB$^+$-tree[9],Slim-tree[10],M$^+$-tree[11].文献[16]基于数据空间优化划分的概念,提出了基于距离的相似性结构 opt-树及其变种 η-树.文献[17]提出了一种新的高维索引结构 VA-Trie,该索引结构结合了 VA-File 和 A-Trees 中近似量化的思想来实现矢量数据的压缩,引入了 Trie 结构来组织和管理矢量数据.

Slim-tree 通过一个后处理过程减少子空间的交叠和索引节点的数目.MB$^+$-tree 采用另外一种不同的方法,即采用 B$^+$-tree 作为一个辅助索引结构,而不是采用单一的多维索引来处理高维空间中的查询问题.MB$^+$-tree 的空间划分是不相交的.

基于向量空间的索引和基于度量空间的索引这两类索引在数据维数不是太高的情况下效率非常高.但是随着数据维数的增加,大多数索引的查询性能会下降.为了解决这一问题,常会采用降维方法,也就是先降低数据的维数,然后使用现有的方法来索引低维的数据空间.降维技术主要分成两种类型:全局降维(global dimensionality reduction,简称 GDR)和局部降维(local dimensionality reduction,简称 LDR).由于在实际数据集中并不一定是全局相关的,因此 GDR 方法对于实际数据集效果并不是很好.为了克服 GDR 方法存在的问题,LDR 方法首先发现数据集中的局部相关性,然后在局部相关的数据集上分别进行降维.但 LDR 的局部相关聚类过程代价比较昂贵,尤其是对于更新比较频繁的数据库来讲,基于 LDR 方法的索引性能会不断下降,例如 Δ-tree[12].

1 本文的研究动机

本文从一个新的角度来研究如何改进高维索引的性能,即在查询处理的过程中如何尽量减少对假活动子空间的访问.在本节中,我们首先给出两个有用的定义:活动子空间和非活动子空间,然后给出在这两个概念之上的一个观察,并基于这一观察导出了假活动子空间的概念.然后讨论解决假活动子空间访问问题的基本思路.

定义 **1**(活动子空间和非活动子空间). 在基于高维索引的相似性查询处理过程中,如果一个子空间可能包含查询结果(即该子空间与查询空间相交),则该子空间被称为一个活动子空间;反之,如果一个子空间不可能包

含查询结果时(即该子空间与查询空间不相交),则该子空间被称为一个非活动子空间.

观察 **1**. 在相似性查询处理过程中,一个子空间没有被过滤掉,但它的所有孩子空间可能都能够被过滤掉.如图 1 所示,在某些情况下,子空间 A 是活动子空间,而它的所有孩子空间,即 B,C,D 可能都是非活动子空间.在这种情况下,如果能够早点过滤掉子空间 A,则可以极大地提高查询处理的性能.

定义 **2**(假活动子空间). 在相似性查询处理过程中,如果一个子空间 S 可能包含查询结果,但该子空间的所有孩子空间并不包含任何查询结果,则称该子空间 S 为假活动子空间.

如图 1 所示,假设子空间 A 是活动的.如果它的孩子空间 B,C,D 是非活动的,则称 A 是假活动的.

下面讨论几个有用的观察,它们非常有效地支持本文提出的新索引技术.

观察 **2**. 在空间中,数据的分布通常是非均匀的,而查询对象可能出现在不同位置.如图 2 所示,假设查询半径是 r,无论是查询 Q 还是查询 Q',子空间 A 都不能被三角不等式过滤原理过滤掉.但是,如果数据仅仅分布在椭圆区域 B,则对于查询 Q'来讲,子空间 A 是一个假活动子空间,因为子空间 A 并不包含查询 Q'的任何查询结果.

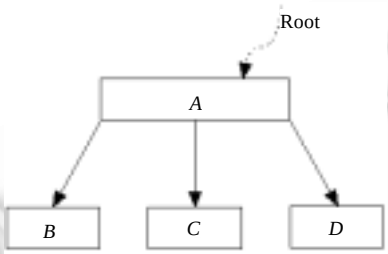

Fig.1 Example of subspaces

图 1 子空间示例

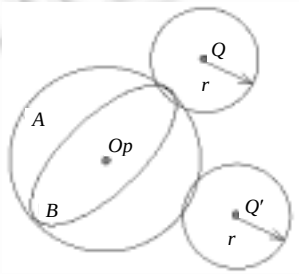

Fig.2 Illustrating false active subspaces

图 2 假活动子空间示例

表 1 给出了 M-tree 索引的 KNN 查询的子空间访问次数和假活动子空间访问次数的统计结果,从统计结果可以看出,有 70%以上的索引访问都是针对假活动子空间的访问.也就是说,由于假活动子空间的大量存在,一个查询不得不遍历许多假活动子空间,而它们实际上并不包含任何查询结果.这就引起了不必要的 I/O 开销和距离计算.而且从表 1 可以看出,数据维数越高,假活动子空间的访问频率就越大.因此,减少对假活动子空间的访问次数对改善索引相似性查询的性能至关重要.这是本文重点要解决的问题.

Table 1 Statistic on access to false active subspaces in M-tree

表 **1** M-tree 索引中假活动子空间的访问情况

Number of dims	Number of accesses to subspaces	Number of accesses to false active subspaces
5	715	507
10	1 953	1 406
15	3 460	2 321
20	5 013	3 381
25	6 627	4 518
30	7 942	4 935
35	10 494	6 175
40	10 699	6 367

观察 **3**. 对于一个假活动子空间,通过将原始空间映射到低维空间,就有可能避免这种对假活动子空间的访问.如图 3 所示,Q'无法剪枝数据空间 A.假设数据仅仅分布在区域 B 中,则在映射空间 $A'[p_1,p_2]$中,查询 Q'是非活动的,因为查询 Q'在映射空间中的投影与 A'并不相交.

很明显,在映射后的空间中进行查询精炼是可能的.我们知道,在高维空间中,相邻的子空间较之在低维空间中更容易相交.因此,通过高维空间到低维空间进行映射来解决假活动子空间访问是较为合理的.图 3 实际上就是本文给出的解决假活动子空间访问的基本解决思路,即将原始空间中的真实数据空间分布映射到低维空间,然后在低维空间中精炼查询.有关空间映射策略将在第 2.1 节中加以讨论.

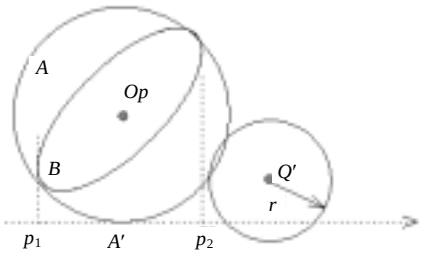

Fig.3 Filtering false active subspaces

图 3 假活动子空间的过滤

2 MS-tree

根据第 1 节中解决假活动子空间访问问题的基本思路,本文设计并实现了一种新的索引结构 MS-tree.与 M-tree 类似,在 MS-tree 索引结构中有两种类型的节点对象:路由对象(routing object)和叶子对象(leaf object).每一个叶子节点入口项包含 3 部分:数据对象 O_j 的特征值、对象标识符 $oid(O_j)$以及对象 O_j 到其父亲 $P(O_j)$的距离 $d(O_j,P(O_j))$.中间节点的入口项信息包含两部分:对象在原始空间的信息和该对象在投影空间的信息.其中前者包括:中间节点对象的特征值及其覆盖半径、距离其父节点的距离、指向孩子节点的指针.而后者则包含一个代表投影空间的队列、投影空间的覆盖半径以及该对象在投影空间中距离其父节点对象的投影距离.下面,首先介绍子空间映射策略,然后讨论 MS-tree 的构造算法和理论分析与评价,最后讨论 MS-tree 上的相似性查找算法.

2.1 空间映射策略

在 MS-tree 中,空间映射策略是影响查询精炼能力的一个关键因素,它将直接影响到查询精炼的效率.基于下面的观察,本文提出了一种最大间隙空间映射方法,称为 MaxGapMapping.

观察 **4**. 在一个数据空间中,沿着一个维的间隙越长,它成为假活动子空间的机会就越大,在其上精炼查询结果就越容易.如图 4 所示,假设阴影部分为数据分布空间,则可以清楚地看到,在垂直维上的间隙 d_1+d_2 比在水平维上的 D_1+D_2 要大.如果一个查询区域沿着垂直维与(L_1,L_3)或(L_2,L_4)相交,则可以通过该维来过滤该空间.而对于水平维,因为 D_1,D_2 非常小,通过它们来过滤假活动子空间的概率就要小得多.

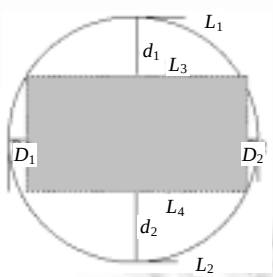

Fig.4 Space mapping

图 4 空间映射

基于观察 4,本文提出的最大间隙空间映射策略的基本思想是,将一个高维数据的维分成两部分:一部分是由对假活动空间有较大过滤能力的维构成,这些维具有比较大的维间隙值;另一部分是由对假活动空间过滤能力比较弱的维构成,这些维具有较小的维间隙值.显然,为了过滤假活动空间并提高查询精炼的效率,在映射空间中只需要保持具有较大维间隙值的那些维.

最大间隙空间映射算法描述如下:(1) 计算父亲数据空间 MBR 的最大边界值和最小边界值;(2) 计算当前节点 MBR 的最大边界值和最小边界值(L_3,L_4);(3) 计算球体的最大边界值和最小边界值(L_1,L_2);(4) 计算每个维上的维间隙值(即在图 4 中的 d_1+d_2 或 D_1+D_2);(5) 根据最大间隙原则,选择具有较大维间隙值的维来构造映射后的低维空间.算法 1 给出了最大间隙空间映射算法.

算法 **1**. 空间映射(space mapping)算法.

Max*Gap*(*N*)

(1) Get MinMax*Boundary* of parent;

(2) Get MinMax*Boundary* of *N*;

(3) Compute *Gap* over each *Dim*;

(4) Sort the *dimensions* by *Gaps*;

(5) Choose the mapped spaces;

(6) RETURN {mapped spaces}.

2.2 构造MS-tree

MS-tree 是一个动态索引结构,采用自下而上的构造途径.当一个新对象被插入时,MS-tree 首先找到适当的插入节点 N.如果该节点没有满,则该对象被直接插入到该节点中;如果该节点已满,则该节点将被分裂.然后检查插入对象是否在结点 N 的 MBR 中.如果不在,则更新 N 的 MBR 并将原始空间映射到一个低维空间,从 N 到根节点,递归地执行这一步直到不需要进行更新为止.算法 2 给出了 MS-tree 构造过程的简单描述.

算法 **2**. MS-tree 构造(constructing MS-tree)算法.

Buildtree(*entry*(*On*),*N*)

(1) Choose the N' from node N

(2) IF N' is not full

(3) THEN $N' \leftarrow entry(On)$

(4) ELSE split;

(5) IF *On* is not in MBR of *N*

(6) THEN *UpdateTheMBR*;

(7) *Mapping*(*N*).

2.3 理论分析

为了从理论上分析 MS-tree 的构建代价,我们主要对 MS-tree 的高度进行分析.为此,表 2 给出了估算参数.

Table 2 Parameters for cost estimation

表 **2** 估算参数

Name of parameter	Description of parameter
N	Number of dimensions of the data set
PageSize	Size of disk page
BytesOfDouble	Number of bytes of double type
BytesOfInt	Number of bytes of integer type
BytesOfPointer	Number of bytes of a pointer
DatasetSize	Size of the data set

对于一个给定的数据集,由于磁盘页面的大小是固定的,通常情况下,索引树的高度取决于索引入口项的大小和它的扇出.在 MS-tree 中,每个非叶子节点入口项包含一个 n 维路径对象 O_j,数据区域的最小覆盖半径 $r(O_j)$ 和距离父亲节点的距离 $d(O_j,P(O_j))$,路经对象 O_j 在投影空间中的投影对象 Q'_j,数据区域的最小投影覆盖半径 $r(Q'_j)$,该节点距离父亲节点的投影距离 $d(Q'_j,P(Q'_j))$,以及指向孩子节点的指针.在 MS-tree 索引中,我们使用 double 类型来表示非叶子节点中的数据类型和覆盖半径,用 int 类型来表示距离,假设投影空间的平均维数为 k,则 MS-tree 非叶子节点入口项的平均大小可以用下面的式(1)来估算:

$$EntrySize_{MS\text{-}tree}=(n+k+2)\times ByteOfDouble+2\times BytesOfInt+BytesOfPointer \tag{1}$$

我们假定节点的空间利用率,即索引节点的填充率为 2/3,则每个节点的扇出由式(2)来进行估算:

$$Fanout_{MS\text{-}tree}=\left\lfloor \frac{PageSize}{EntrySize_{MS\text{-}tree}} \right\rfloor \times \frac{2}{3} \tag{2}$$

这样,我们可以用式(3)来估算 MS-tree 的高度:

$$Height_{MS\text{-}tree}=\left\lceil \log_{Fanout_{MS\text{-}tree}}^{DataSize} \right\rceil \tag{3}$$

2.4 相似性查找算法

MS-tree 可以支持两种类型的相似性查找:*r*-range 查找和 KNN 查找.由于 KNN 查找与范围查找使用相同

的精炼机制,因此本文仅仅讨论了 MS-tree 的范围查询,以便测试本文提出的空间映射方法及索引结构的性能.

对于一个给定的范围查询(q,r),q 为查询对象,r 为查询半径.该查询的结果为所有到查询对象 q 的距离小于 r 的对象.该查询的执行过程是:从 MS-tree 的根节点开始,递归地执行查询处理过程直到叶子节点为止,并保留所有的匹配对象.在查询处理过程中,对于一个搜索空间,如果它是一个假活动子空间,则与之对应的子树为假活动子树.

在非叶子节点层,范围查询需要执行三级过滤(查询精炼):第一,在原始的高维空间中,根据节点中的入口项到其父亲节点路由对象的距离,利用三角不等式进行过滤;第二,对第 1 步不能被过滤的子树,在映射空间中,进一步精炼结果,通过三角不等式过滤掉那些假活动子树;最后,计算查询对象到未过滤掉的节点入口项之间的距离,并使用三角不等式进行进一步的过滤.对于叶子节点,执行一个两步过滤操作,其过滤原理与非叶子节点的前两步过滤原理相同,第 1 步在原始空间中过滤,第 2 步在映射空间中过滤.算法 3 给出了范围查询算法的描述.

算法 **3**. 范围查询算法(range search algorithm).

```
RangeSearch(N:node,Q,r)
(01)   O_p←ParentNode(N)
(02)   IF N is not leaf\
(03)      THEN IF |d(O_p,Q)−d(O_r,O_p)|≤r+r(O_r)
(04)                THEN RefiningInMappedSpace;
(05)                     Compute(d(O_r,Q));
(06)                     IF d(O_r,Q)<r+r(O_r)
(07)                        THEN RangeSearch(Ptr,Q,r);
(08)   ELSE
(09)   FOR all O_j in N do
(10)         IF |d(Op,Q)−d(Or,Op)|≤r
(11)            THEN RefiningInMappedSpace;
(12)                 Compute(d(O_j,Q));
(13)                 IF d(O_j,Q)≤r}
(14)                    THEN add oid(O_j) to results;
```

3 性能评价

本节主要从映射空间维数对性能的影响、I/O 开销、CPU 开销这 3 方面来评价 MS-tree 的查询精炼能力,同时也对 MS-tree,M-tree 和 Slim-tree 这 3 种索引结构的性能进行了比较分析.同时,我们也对Δ-tree 和 MS-tree 的性能进行了分析与比较.

本文实验选用的数据集生成如下:选取 20 000 幅真实图像,使用 MPEG-7 特征抽取工具将这些图像的 Color Layout 特征抽取出来构成一个高维数据集,Color Layout 的特征维数是 12.实验环境是一台 Pentinum IV 2.5GHz 的 PC 机,内存为 256MB.所有数据存储在一个对象数据库系统 Fish 中[13].在所有实验中,页面大小被设置为 4 096 字节.

3.1 映射空间维数对性能的影响

本节主要通过实验来测试映射空间的维数对性能的影响,并发现最好的映射空间维数,即究竟应该保留多少维是比较好的.由于提出 MS-tree 的主要目的是通过减少假活动子树的访问来提高查询性能,因此我们仅仅考虑了 I/O 开销和在映射空间维数变化情况下访问假活动子树的开销,因为 I/O 次数反映了假活动子树访问的数量.也就是说,I/O 次数越多,对假活动子树的访问也就越多.

图 5 和图 6 给出了查找半径为 0.1 时 I/O 开销和假活动子树访问的测试结果,图 7 和图 8 是查找半径为 0.15 时的 I/O 开销和假活动子树访问测试结果.从这些图中不难看出,随着维数的增加,I/O 次数和假活动子树访问次数开始降低,当维数等于 3 时,降到最低点,然后逐渐开始上升.我们可以看到,对于这一数据集,将原始空间映射

到一个三维空间是最佳的.另一点非常重要的是,对于一个给定数据集,我们可以发现,从原始数据空间到映射数据空间的最佳维数,并且这个最佳维数与原始数据空间的维数相比通常是非常小的.由于在后面的实验中使用的是相同的数据集,我们将映射空间的维数都设置为 3.

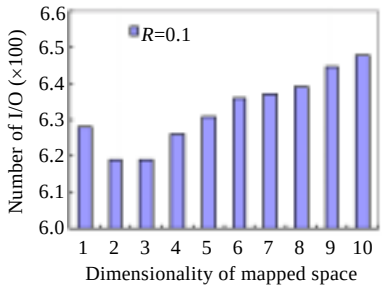

Fig.5 I/O overhead of range search

图 5 范围查询的 I/O 开销

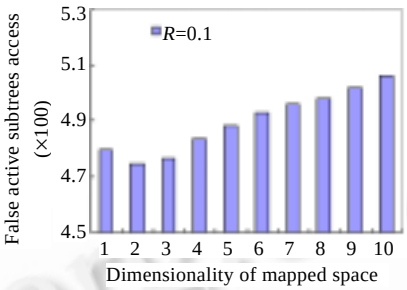

Fig.6 False active subspace access of range search

图 6 范围查询的假活动子树访问

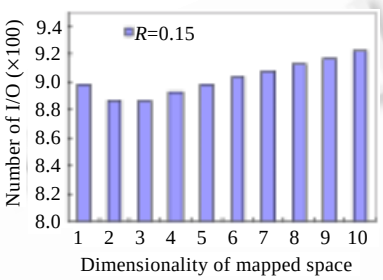

Fig.7 Mapped space dimensionality vs. I/O

图 7 映射空间维数对 I/O 的影响

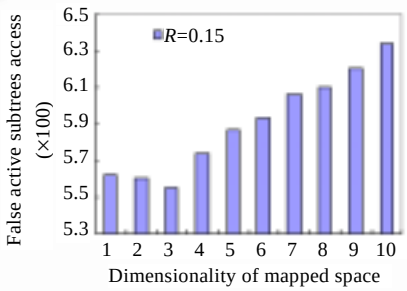

Fig.8 Mapped space dimensionality vs. false active subspace access

图 8 映射空间维数对假活动子树访问的影响

3.2 I/O代价

本节对 MS-tree 的范围查找算法的 I/O 开销与 M-tree 和 Slim-tree 进行了比较.我们的主要目标是要考察精炼过程对索引的有效性.在这个实验中,MS-tree 索引的映射空间维数被设定为 3,查找半径在 0.05~0.15 之间变化.图 9 和图 10 分别给出了 I/O 开销和假活动子树访问的测试结果.从这些图中可以看到,与 M-tree 相比,Slim-tree 减少了 I/O 和假活动子树的访问,因为它的后处理过程减少了索引节点的总数,这样访问索引节点的机会就会相应减少.

同时,显而易见的是,MS-tree 总是具有最好的 I/O 和假活动子空间的访问开销,因为在映射空间中的精炼操作可以减少 20%的假活动子树的访问.另一个值得指出的是,无论查询半径如何变化,MS-tree 对于所有查询都是最好的.与 Slim-tree 相比,MS-tree 是优化 M-tree 性能的一种更加有效的索引.

3.3 CPU代价

本节要对 MS-tree,Slim-tree 和 M-tree 中查询精炼过程对 CPU 代价的影响进行比较分析.我们设置映射空间的维数为 3,并且范围查询的查找半径的变化为 0.05~0.25.图 11 和图 12 给出了响应时间和距离计算的测试结果.可以清楚地注意到,尽管 Slim-tree 比 M-tree 具有较少的 I/O 开销,Slim-tree 仍然保留了与 M-tree 具有相同的距离计算次数,因为经过后处理后每个节点包含了更多的对象,需要更多的距离计算.因此,当执行热范围查询时,Slim-tree 无法加速查询响应,这一点可以从图 11 中反映出来.

与 Slim-trees 相比,MS-tree 更加有效地改进了 M-tree 的查询性能.由于映射空间中的非叶子节点和叶子节点的巨大精炼能力,一方面,在一个节点中的入口项保持稳定,这样它只需要较少的距离计算,因为它需要较少

的 I/O;另一方面,由于叶子节点中的过滤,某些叶子节点可以通过精炼过程将它们过滤掉而无须任何距离计算.因此,MS-tree 具有非常好的效率.与 Slim-tree 和 M-tree 相比,降低了 30%~50%的距离计算和一半的响应时间.

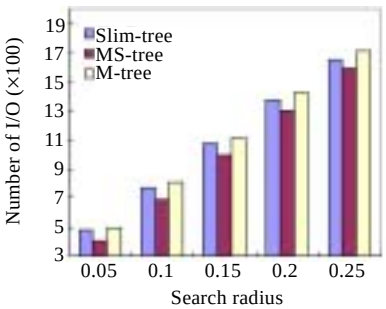

Fig.9 I/O overhead of range search

图 9 范围查询的 I/O 开销

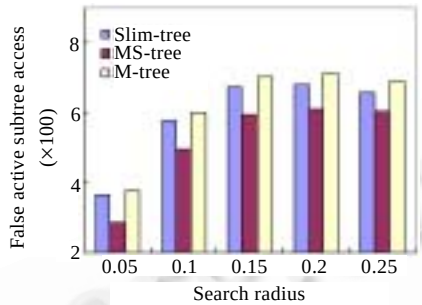

Fig.10 False active subspace access of range search

图 10 范围查询的假活动子树访问

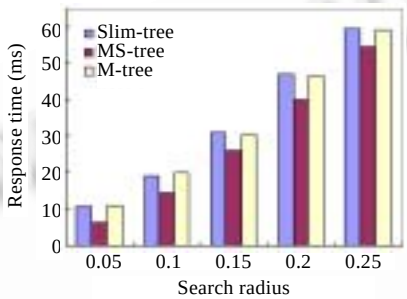

Fig.11 Response time of range search

图 11 范围查询响应时间

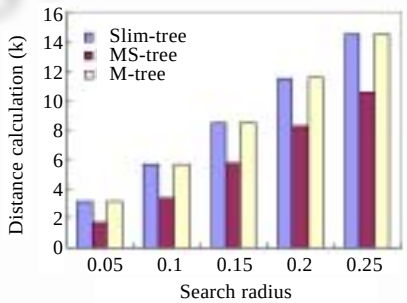

Fig.12 Distance calculation of range search

图 12 范围查询距离计算

3.4 与降维技术的比较

由于本文采取技术的核心思想是通过降维技术来提高索引技术的性能,因此在本节我们将分析和比较基于降维技术的Δ-tree 和 MS-tree 的性能,我们使用最近邻算法来比较和分析这两种索引技术.在该实验中,K 值从 10 到 50 变化,数据集的大小为 40 000.

图 13 和图 14 给出了最近邻算法在Δ-tree 和 MS-tree 两种索引结构上的响应时间和 I/O 计算代价.

从图中可以看出,随着最近邻查询中 K 值的不断增加,这两种索引技术的相应时间会随之增加,这应该是不言而喻的.同时,无论是响应时间,还是 I//O 计算代价,基于 MS-tree 的算法始终优于基于Δ-tree 的算法.但是这种性能的改进并没有随着 K 值的增加而增加,这主要是因为,随着 K 值的增加,基于最大间隙空间映射方法的过滤能力并未提高,而这因为概念上随着 K 值的增大,在索引树中查询所涉及到的索引结点也势必增多,搜索空间中的点也将增加,这样就会稍微降低基于最大间隙空间映射方法的过滤能力.

3.5 讨 论

本节我们主要讨论 MS-tree 索引结构的适应范围和批量构建问题.

本文提出的基于最大间隙空间映射的方法可以应用到任意的 R-tree 序列及其变种,以及任意基于空间包络过滤原理的索引结构上,因为基于最大间隙空间映射的降维方法是基于维的概念.它不仅可以适应于 MBR 包络过滤原理,也适合于其他空间包络方法,例如超球体的包络、MBR 和超球体交的包络、椭圆包络等.这样,最大间隙空间映射方法就不能适应于 M-tree 索引及其变种,因为这些索引方法是基于三角不等式过滤原理,而不是基于空间包络过滤原理的.

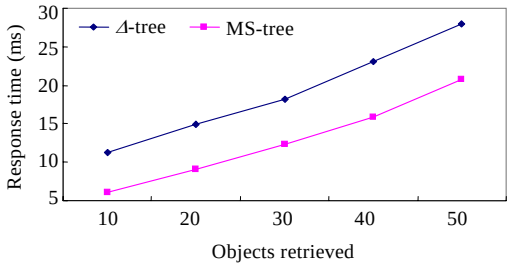

Fig.13 Response time of KNN: Δ-tree vs MS-tree

图 13 Δ-tree 和 MS-tree 最近邻查询的响应时间

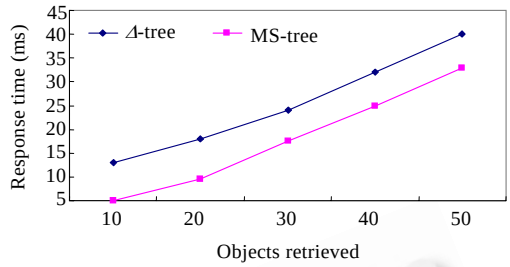

Fig.14 I/O overhead of KNN: Δ-tree vs MS-tree

图 14 Δ-tree 和 MS-tree 最近邻查询的 I/O 代价

索引技术的批量构建技术是反映索引性能的另一个非常重要的方面,已有很多研究成果,包括 R-tree 和 M-tree 的批量构建算法.这些算法的一个主要思想是通过聚类来获得更好的索引性能.文献[14]提出了一种动态构建 M-tree 的批量构建算法,该算法通过自底向上的批量装载技术来快速构建 M-tree 树,但该算法的一个主要缺陷是任意选取聚类中心,这样就很难保证聚类的效果,从而无法保证索引的查询性能.为了解决这一问题,我们在文献[15]中提出了一种基于多步聚类的 M^+-tree 批量构建算法.在该算法中,首先选择任意两个足够远的数据点作为两个聚类的中心,然后将待聚类的数据集中适当数据的点根据相对距离聚类到这两个类中;在剩下的数据集中重新选择两个足够远的点作为两个新的聚类中心,从剩下的数据集中聚出两个新类.以此类推,直到将数据集中所有数据聚类完毕.然后根据各聚类结果进行子树构建并合并成一棵更高的索引树.该方法的批量构建性能与 M-tree 的批量构建算法相似,但构建后的查询性能则要优越得多.尽管基于多步聚类的批量构建算法是为 M^+-tree 而提出来的,但它同样也适应于其他批量索引构建算法,例如 M-tree 和 MS-tree.

4 结 论

在高维空间中,传统的过滤技术引起了大量的假活动空间的访问.本文通过映射原始空间到一个低维空间且在低维空间中通过精炼查询来避免假活动子空间访问的方法,最大限度地克服了现有过滤方法的缺点.然后,我们提出了一种新的空间映射策略,它能够有效地过滤假活动子空间.最后,基于本文提出的空间映射策略设计并实现了一种高维索引结构,称为 MS-tree.它是一个完全动态的分页的平衡树.实验结果分析显示,MS-tree 较之其他方法具有更好的查询处理性能.

References:

[1] Bohm C, Berchtold S, Keim DA. Seaching in high-dimensional spaces. ACM Computing Surveys, 2001,33(3):322−373.

[2] Berkmann N, Krigel HP, Schneider R, Seeger B. The *R**-tree: An efficient and robust access method for points and rectangles. SIGMOD Record, 1990,19(2):322−331.

[3] Katayama N, Satoh S. The SR-tree: An index structure for high-dimensional nearest neighbor queries. SIGMOD Record, 1997, 26(2):369−380.

[4] White DA, Jain R. Similarity indexing with the SS-tree. In: Su SYW, ed. Proc. of the 20th Int'l Conf. on Data Engineering. New Orleans: IEEE Computer Society, 1996. 516−523.

[5] Lin KI, Jagadish HV, Faloutsos C. The TV-tree: An index structure for high-dimensional Data. VLDB Journal, 1994,3(4):517−542.

[6] Fu A, Chan P, Cheung Y, Moon Y. Dynamical VP-tree indexing for *n*-nearest neighbor search given pair-wise distances. VLDB Journal, 2000,9(2):154−173.

[7] Bozkaya T, Ozsoyoglu M. Distance-Based indexing for high-dimensional metric spaces. SIGMOD Record, 1997,26(2):357−368.

[8] Ciaccia P, Patella M, Zezula P. M-tree: An efficient access method for similarity search in metric spaces. In: Jarke M, Carey MJ, Dittrich KR, Lochovsky FH, Loucopoulos P, Jeusfeld MA, eds. Proc. of the 23rd Int'l Conf. on Very Large Data Bases. San Fransisco: Morgan Kaufamann Publishers, 1997. 426−435.

[9] Cui B, Ooi BC, Su J, Tan K. Contorting high dimensional data for efficient main memory processing. In: Halevy AY, Ives ZG, Doan A, eds. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. San Diego: ACM, 2003. 479−490.

[10] Jr. Traina C, Traina A, Seeger B, Faloutsos C. Slim-trees: High performance metric trees minimizing overlap between nodes. In: Zaniolo C, Lockemann PC, Scholl MH, Grust T, eds. Proc. of the 7th Int'l Conf. on Extending Database Technology. Springer-Verlag, 2000. 51−65.

[11] Zhou X, Wang G, Yu JX, Yu G. M^+-tree: A new dynamical multidimensional index for metric spaces. In: Schewe KD, Zhou X, eds. Proc. of the 14th Australasian Database Conf. Australian Computer Society, 2003. 161−168.

[12] Chakrabarti K, Mehrotra S. Local dimensionality reduction: A new approach to indexing high dimensional spaces. In: Abbadi AE, rodie ML, Chakravarthy S, Dayal U, Kamel N, Schlageter G, Whang KY, eds. Proc. of the 26th Int'l Conf. on Very Large Data Bases. San Fransisco: Morgan Kaufamann Publishers, 2000. 89−100.

[13] Wang G, Lu H, Yu G, Bao Y. Managing very large document collections using semantics. Journal of Computer Science and Technology, 2003,18(3):403−406.

[14] Ciaccia P, Patella M. Bulk loading the M-tree. In: Roddick JF, ed. Proc. of the 9th Australasian Database Conf. Perth: Springer-Verlag, 1998. 15−26.

[15] Zhou XM, Wang GR, Chang LZ, Fan D. Bulk-Loading M^+-tree. Mini-Micro Systems, 2006,27(2):295−299 (in Chinese with English abstract).

[16] Feng Y, Cao K, Cao Z. A multidimensional index structure for fast similarity retrieval. Journal of Software, 2002,13(8):1678−1685 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/13/1678.pdf

[17] Dong D, Liu Z, Xue X. VA-trie: A new and efficient high dimensional index structure for approximate *k* nearest neighbor query. Journal of Computer Research and Development, 2005,42(12):2213−2218 (in Chinese with English abstract).

附中文参考文献:

[15] 周项敏,王国仁,常立喆,范丹.批量构建 M^+-tree.小型微型计算机系统,2006,27(2):295−299.

[16] 冯玉才,曹奎,曹忠升.一种支持快速相似性检索的多维索引结构.软件学报,2002,13(8):1678−1685. http://www.jos.org.cn/1000-9825/13/1678.pdf

[17] 董道国,刘振中,薛向阳.VA-Trie:一种用于近似 *k* 近邻查询的高维索引结构.计算机研究与发展,2005,42(12):2213−2218.

王国仁(1966－),男,湖北崇阳人,博士,教授,博士生导师,CCF 高级会员,主要研究领域为多媒体数据管理,P2P 数据管理,生物信息学,XML 数据管理.

韩东红(1968－),女,讲师,主要研究领域为数据流管理.

黄健美(1966－),男,博士生,主要研究领域为数据挖掘.

乔百友(1972－),男,讲师,CCF 会员,主要研究领域为 P2P 数据管理.

王斌(1971－),男,讲师,CCF 会员,主要研究领域为 P2P 数据管理.

于戈(1962－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为数据挖掘与数据仓库,数据流管理,Web Services.