

# 一种基于双重距离尺度的高维索引结构

庄 毅,翁建广,庄越挺,吴 飞

(浙江大学 计算机科学与技术学院,浙江 杭州 310027)

**摘 要:** 为了提高高维数据相似查询的效率,提出一种基于双重距离尺度(DDM)的新型高维索引结构.通过建模得到该 DDM 的四元组数据结构,对于高维空间中的数据点,通过  $k$  平均聚类算法将数据点聚成若干类,分别计算每个点对应的始点和质心距离,得到基于加权的质心距离,并将加权的质心距离作为每个数据点的索引键值,且用基于分片的  $B^+$  树建立索引,得到了该索引的创建算法.高维空间的查询就转变成对一维空间的检索,并研究了数据点的维数、数据量和查询请求参数对查询性能的影响.结果表明,该 DDM 能更有效地缩小搜索空间,减少距离计算的开销,特别适合海量高维数据的查询.

**关键词:**  $k$  近邻查询;类超球;质心距离;始点距离

中图分类号: TP301

文献标识码: A

文章编号: 1008-973X(2007)03-0380-06

## Novel high-dimensional indexing structure based on dual-distance metric

ZHUANG Yi, WEN Jian-guang, ZHUANG Yue-ting, WU Fei

(College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China)

**Abstract:** To speed up high-dimensional similarity search efficiency, a novel high-dimensional indexing structure based on dual distance metric (DDM) was proposed. A four-tuple data structure of the DDM was obtained after modelling. Every point in high-dimensional space was grouped into some clusters using  $k$ -means cluster algorithm, then the weighted centroid distance of every point was computed based on the start distance and centroid distance of every point. The index key value of every point was inserted by a partition-based  $B^+$ -tree, and the index construction algorithm was obtained. Queries in high-dimensional space were transformed into queries in single-dimensional space, and the effects of dimensionality, data size and query request parameter on query performance were investigated. The results show that DDM can effectively reduce search space and the distance computation cost. The index structure is particularly fit for querying large-scale high-dimensional data.

**Key words:**  $k$ -nearest neighbor search; cluster hypersphere; centroid distance; start distance

随着 Internet 上多媒体信息的爆炸性增长,基于内容的海量多媒体信息检索成为一个热门的研究领域,其中海量高维数据的有效索引和查询技术是关键.而目前高维索引技术的查询效率往往还不够理想<sup>[1]</sup>,特别对于维数较高的数据.

高维索引技术经历了 20 多年的研究<sup>[1]</sup>,所采用的技术主要分为 3 类:

1) 基于数据和空间分片的树形索引,如 R-tree<sup>[2]</sup> 及其变种<sup>[3-4]</sup> 等.然而这些方法仅适合维数较低的情况,随着维数的增加,索引的性能下降使得顺序检索的效率往往优于树形索引.该类现象称为维数灾难.

2) 采用近似的方法来表示原始向量,如 VAF<sup>[5]</sup> 和 IQ-tree<sup>[6]</sup> 等.其基本思想是通过对高维数

据进行压缩和近似存储来加速顺序查找速度.然而数据压缩和量化带来的信息丢失,使得首次过滤后的查询精度并不令人满意.尽管它能显著减少磁盘的 I/O 次数,但由于对位串解码和对查询点距离的上、下界计算,会导致很高的 CPU 运算开销.

3) 采用基于距离尺度的方法,即通过将高维数据转化为一维数据或距离值来进行高维检索,包括 NB-Tree<sup>[7]</sup>和 iDistance<sup>[8]</sup>.NB-Tree<sup>[7]</sup>通过计算高维空间中的每个点与原点  $O(0,0,\dots,0)$  的距离,将高维数据点映射到一维空间,然后对这些距离值建立 B<sup>+</sup> 树索引,使得高维检索转变为一维空间的检索.尽管该方法能够快速得到查询结果,但是由于它不能有效地缩减查询空间,特别当维数很高时,查询效率较差.iDistance<sup>[8]</sup>是一种基于多参考点的高维索引方法,该方法通过引入多参考点和结合聚类等方法有效地缩小了数据搜索空间范围,提高了查询效率.然而该方法的查询效率很大程度上取决于参考点的选取,并且依赖数据聚类和分片.最坏的情况下,查询空间几乎会覆盖整个高维空间.

本文介绍一种基于双重距离尺度(dual distance metric, DDM)的高维索引方法,用于支持高效的相似查询.该方法的基本思想是通过聚类和基于双重距离尺度(即始点距离和质心距离)的方法,将高维空间的每个点转化成一维空间对应的距离值.

## 1 双重距离尺度

### 1.1 问题定义及动机

给定高维数据库  $\Omega = \{V_1, V_2, \dots, V_n\}$ , 其中  $V_i$  为第  $i$  个数据点, 每个点的维数为  $d$ , 且  $\forall V_i \in \Omega$ . 不失一般性, 对于任意给定的 2 个点  $V_i$  与  $V_j$ , 它们间的距离用  $d(V_i, V_j)$  来表示. 同时给定一个查询点  $V_q$  和查询半径  $r$ , 该超球表示为  $\Theta(V_q, r)$ .

**定义 1** 始点距离. 给定点  $V_i$ , 它的始点距离(start-distance, SD)为它到原点  $V_o$  的距离, 记作  $\sigma(V_i) = d(V_i, V_o)$ , 其中  $V_o = \{0, 0, \dots, 0\}$ .

假设  $n$  个点通过  $k$  平均聚类得到  $T$  个类. 对于任意一个类  $C_j$ , 其中  $j \in [1, T]$ , 该类中点的个数记为  $\|C_j\|$ , 且满足

$$\sum_{j=1}^T \|C_j\| = n.$$

**定义 2** 类半径. 对于任意一个类  $C_j(V_1, V_2, \dots, V_x)$ , 其质心  $O_j$  与该类中距离其最远点的距离为它的类半径, 记作  $R_j$ , 其中  $j \in [1, T]$ , 且  $x = \|C_j\|$ .

**定义 3** 类超球. 给定任意一个类  $C_j$  和类半径  $R_j$ , 类超球表示为  $\Theta(O_j, R_j)$ .

**定义 4** 质心距离. 给定一个点  $V_i$ , 它的质心距离(centroid-distance, CD)为该点到所在类  $C_j$  的质心  $O_j$  的距离, 表示为  $\delta(V_i) = d(V_i, O_j)$ , 且  $V_i \in \Theta(O_j, R_j), i \in [1, \|C_j\|], j \in [1, T]$ .

DDM 的提出是基于以下 2 点: 1) 在高维空间中, 点之间的相似性可以通过该点与某个参考点之间的距离来度量和排序; 2) 由于距离是一维值, 这样可以用其来近似表示高维空间对应的点, 同时可以使用一维索引 B<sup>+</sup> 树来对这些距离值建立索引.

### 1.2 数据结构

本文 DDM 方法首先通过  $k$  平均聚类, 将数据点聚成  $T$  类, 然后求得每个点的始点距离  $\sigma$  和质心距离  $\delta$ , 这样每个点  $V_i$  可以表示为一个四元组:

$$V_i ::= \langle i, \eta, \sigma, \delta \rangle. \quad (1)$$

式中:  $i$  为点  $V_i$  的编号,  $\eta$  为该点所属类的编号.

由于每个类超球中点对应的  $\sigma$  和  $\delta$  无法进行有效组合表达成一个统一的索引键值, 以便能够进一步缩小搜索空间. 为此本文提出基于加权质心距离的方法, 该方法通过对类超球的切分来得到索引键值, 如图 1 所示. 假设查询超球  $\Theta(V_q, r)$  与第  $j$  个类超球  $\Theta(O_j, R_j)$  相交, 首先将该类超球中点的始点距离  $\sigma$  平均切分为  $\lambda$  片. 对于该类超球中第  $l$  个分片的点来说, 当

$$\sigma(V_i) \in [\sigma(O_j) - R_j + 2R_j l / \lambda, \sigma(O_j) - R_j + 2R_j (l+1) / \lambda],$$

其中  $l \in [1, \lambda]$ , 且满足

$$l = (\lambda \times \sigma(V_i) - \sigma(O_j) + R_j) / (2R_j) + 1,$$

该点对应的加权质心距离, 记为  $\rho$ , 表示为

$$\rho(V_i) = l + \delta(V_i) / M. \quad (2)$$

由于  $\delta(V_i)$  可能大于 1, 需要通过对其分别除以  $M$  进行归一化, 使得其值小于 1, 其中对于真实数据来说,  $M$  取  $\sqrt{2}$ . 而对于均匀分布的随机数据来说,  $M$  取  $\sqrt{d}$ . 这样使得每个点对应的质心距离的值域尽量不

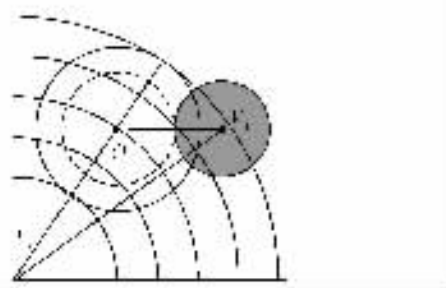


图 1 类超球的切分

Fig. 1 Slices of cluster hypersphere

重叠,最后对  $n$  个键值建立基于分片的  $B^+$  树索引.

## 2 相关算法

### 2.1 索引生成算法

DDM 的索引结构如图 2 所示,它由一张哈希表和  $T$  个子索引构成,其中  $T$  为聚类个数.通过  $k$  平均聚类后,每个类超球中的点分别采用一棵  $B^+$  树建立索引,作为 DDM 的一个子索引.当存在  $T$  个类时,就需要建立  $T$  棵  $B^+$  树,同时需要建立一张哈希表,根据点所在类的编号快速地找到对应的子索引.一般采用最简单的一一对应的方式来完成哈希映射,即点所在类的编号为其子索引的编号.

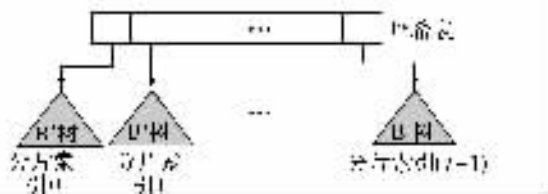


图 2 DDM 的索引框架

Fig. 2 Index architecture of DDM

图 3 中 DDM 索引的创建分 2 步: 1) 首先对  $n$  个点进行  $k$  平均聚类,得到  $T$  个类,然后通过  $T$  次循环,用 *newDDMFile()* 生成子  $B^+$  树索引头文件; 2) 接着对于每个类中的点,计算它的质心距离,并且通过转换得到对应的键值,并将其插入对应的子  $B^+$  树索引.尽管得到的这  $k$  个子索引在物理上是离散存储,但通过哈希表的统一映射,使得其在逻辑上是完整的. DDM 索引生成算法如下:

#### 算法 1 DDM 索引创建

输入:  $\Omega$ : 高维数据库,  $\lambda$ : 类超球的分片个数;  
输出:  $bt(1$  to  $T)$ : DDM 高维索引;  
1 points are grouped into  $T$  clusters by  $k$ -Means algorithm;  
2 for  $j := 1$  to  $T$  do  
3  $bt(j) \leftarrow \text{newDDMFile}()$ ;  
4 for each point  $V_i$  in the  $j$ -th cluster  
5 compute the centroid-distance and slice ID( $l$ ) of  $V_i$ ;  
6  $\rho(V_i) = l + \delta(V_i)/M$ ;  
7 insert  $\rho(V_i)$  to the  $j$ -th  $B^+$  tree  $bt(j)$ ;  
8 end for  
9 return  $bt(j)$ ;  
10 end for

图 3 DDM 索引生成算法

Fig. 3 Index construction algorithm for DDM index

### 2.2 $k$ 近邻查询算法

基于 DDM 的  $k$  近邻查询( $k$ -nearest neighbor,  $k$ -NN)在开始之前,假设查询超球  $\Theta(V_q, r)$  与第  $j$

个类超球  $\Theta(O_j, R_j)$  相交,研究  $\Theta(V_q, r)$  与  $\Theta(O_j, R_j)$  中的哪些分片相交.如图 1 所示,  $\Theta(V_q, r)$  对应的始点距离范围为  $[\sigma(V_q) - r, \sigma(V_q) + r]$ , 与其相交的类超球  $\Theta(O_j, R_j)$  对应的始点距离范围为  $[\sigma(O_j) - R_j, \sigma(O_j) + R_j]$ . 通过推导,可以得到该类超球中实际与  $\Theta(V_q, r)$  相交的分片是从第  $L(j)$  个到第  $U(j)$  个,其中  $L(j) \leq U(j)$ , 满足下式:

$$L(j) = \begin{cases} (\sigma(V_q) - r - \sigma(O_j) + R_j)\lambda / (2R_j) + 1; \\ \text{当 } \sigma(O_j) - R_j < \sigma(V_q) - r < \sigma(O_j) + R_j, \\ 1, \text{ 当 } \sigma(V_q) - r \leq \sigma(O_j) - R_j. \end{cases} \quad (3)$$

$$U(j) = \begin{cases} (\sigma(V_q) + r - \sigma(O_j) + R_j)\lambda / (2R_j) + 1; \\ \text{当 } \sigma(V_q) + r < \sigma(O_j) + R_j, \\ \lambda, \text{ 当 } \sigma(V_q) + r \geq \sigma(O_j) + R_j. \end{cases} \quad (4)$$

$k$ -NN 查询较范围查询稍复杂,其本质是通过嵌套地调用范围查询算法来得到  $k$  个最近邻点.  $k$  近邻查询算法如图 4:

#### 算法 2 $k$ NNSearch( $V_q, k$ )

输入: 查询点  $V_q, k$   
输出: 查询结果  $S$   
1  $r \leftarrow 0, s \leftarrow \Phi$ ; /\* 初始化/  
2 while ( $\|S\| < k$ ) /\* 当返回候选点个数小于  $k$ , 继续循环/  
3  $r \leftarrow r + \Delta r$ ; /\*  $\Delta r$  很小, 用于逐步增加半径值/  
4  $S \leftarrow \text{SRSearch}(V_q, r)$ ; /\* 嵌套调用 RangeSearch()/  
5 if ( $\|S\| > k$ ) then /\* 当返回候选点个数大于  $k$  /  
6 for count := 1 to  $\|S\| - k - 1$   
7  $V_{\text{far}} \leftarrow \text{Farthest}(S, V_q)$ ;  
8  $S \leftarrow S - V_{\text{far}}$ ; /\* 将  $V_{\text{far}}$  从候选点集  $S$  中删除/  
9 end for  
10 end if  
11 end while  
  
 $\text{RSearch}(V_q, r)$   
12 for  $j := 1$  to  $T$  do /\*  $T$  为总的聚类个数/  
13 if  $\Theta(O_j, R_j)$  contains  $\Theta(V_q, r)$  then  
14  $S1 \leftarrow \text{Search}(V_q, r, j)$ ;  
15  $S \leftarrow S \cup S1$ ;  
16 end; /\* 结束循环/  
17 else if  $\Theta(O_j, R_j)$  intersects  $\Theta(V_q, r)$  then  
18  $S1 \leftarrow \text{Search}(V_q, r, j)$ ;  
19  $S \leftarrow S \cup S1$ ;  
20 else /\* 表明查询超球与第  $j$  类超球不相交/  
21 break; /\* 跳出本次循环/  
22 end if  
23 end for  
24 return  $S$ ; /\* 返回候选点/  
  
 $\text{Search}(V_q, r, j)$   
25  $\text{left} \leftarrow L(j) + (d(V_q, O_j) - r)/M$ ; /\*  $L(j)$  由式(3)得到/  
26  $\text{right} \leftarrow U(j) + R_j/M$ ; /\*  $U(j)$  由式(4)得到/  
27  $S \leftarrow \text{BRSearch}[\text{left}, \text{right}, j]$ ;  
28 for each point  $V_i$  in the candidate points  $S$   
29 if  $d(V_q, V_i) > r$  then  $S \leftarrow S - V_i$ ;  
30 end for  
31 return  $s$ ; /\* 返回候选点/

图 4  $k$  近邻查询算法

Fig. 4  $k$ -NN algorithm

与范围查询不同的是,算法中  $k$ -NN 近邻查询算法开始是用一个较小的半径  $r$  去进行范围查询,当得到的候选点个数小于  $k$  时,再重新增大查询半径  $\Delta r$ ,由于通过上述方法得到的候选点个数不一定正好为  $k$  个,可能会大于  $k$ . 当遇到该情况时,需要进行  $(\|S\| - k - 1)$  次循环(第 6 行),依次找到在该候选点集  $S$  中距离查询点  $V_q$  最远的  $(\|S\| - k - 1)$  个点  $V_{far}$  并且将它们删除. 这样恰好得到  $k$  个最近邻点. 其中函数  $RSearch(V_q, r)$  为以  $V_q$  为中心,  $r$  为半径的范围查询函数.

图 4 中函数  $Search(V_q, r, j)$  用于具体执行,并且返回第  $j$  个子索引范围查询得到的候选点集;当查询超球与第  $j$  个类超球相交时,与其相交的第  $j$  个类超球中的分片为从第  $L(j)$  个分片到第  $U(j)$  个分片,其中  $L(j) \leq U(j)$ ,而质心距离的查询范围为  $[d(V_q, O_j) - r, R_j]$ ,这样将两者结合,就得到总的查询范围(第 25、26 行).

图 4 中函数  $Farthest(S, V_q)$  用于返回候选点集  $S$  中离  $V_q$  最远的点.  $BRSearch(left, right, j)$  用于对第  $j$  个子索引进行标准的范围查询.

### 3 性能分析

如 2.2 节所述,  $k$ -NN 查询是通过嵌套调用范围查询来完成,为简单起见,本节通过分析 3 种基于距离尺度的索引在相同半径  $r$  的范围查询情况下的搜索空间的大小来比较它们的查询性能.

当查询超球  $\Theta(V_q, r)$  与  $T'$  个类超球相交 ( $T' \leq T$ ),如图 5 所示,则对 DDM 来说,当每个与查询超球相交的类超球的分片数目足够多时,其对应的搜索空间可以近似表示为

$$(\Theta(V_o, \sigma(V_q) - \epsilon) \cap \Theta(V_o, \sigma(V_q) + \epsilon)) \cap \sum_{j=1}^{T'} \overline{\Theta(O_j, d(V_q, O_j) - \epsilon)} \cap \Theta(O_j, R_j). \quad (5)$$

即查询超球  $\Theta(V_q, r)$  可近似用式(5)来表示. 对于相

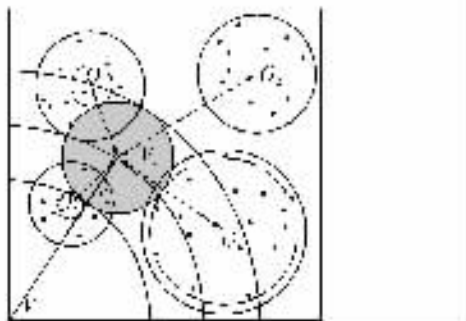


图 5  $k$ -NN 查询例子

Fig. 5 Example of  $k$ -NN search

同半径  $r$  的范围查询, NB-tree<sup>[7]</sup> 对应的搜索空间为  $\overline{\Theta(V_o, \sigma(V_q) - r)} \cap \Theta(V_o, \sigma(V_q) + r)$ . (6)

iDistance<sup>[8]</sup> 则为

$$\sum_{j=1}^{T'} \overline{\Theta(O_j, d(V_q, O_j) - r)} \cap \Theta(O_j, R_j). \quad (7)$$

较上述两种方法,可知 DDM 的搜索空间大大减少,从而有助于提高查询效率.

### 4 实验分析

下面通过实验来验证该算法的有效性,同时与其他索引作比较. 本文用 C 语言实现了基于双重距离尺度的索引 DDM,同时实现或下载了 NB-Tree、iDistance、VA-file 和 X-tree 等高维索引算法. 采用 B<sup>+</sup> 树作为单维索引结构. 所有实验的运行环境为 Pentium IV CPU 2.0GHz, 256 MB 内存, 硬盘大小为 80G 且 7 200 r/min,同时索引页大小设为 4 096 字节.

实验中的测试数据分为 2 类: 1) UCI 提供的颜色直方图数据<sup>[9]</sup> 作为实验数据,它包含了从 Corel 图片库提取 68 040 个 32 维的颜色直方图特征,每一维值的范围都在 0 和 1 之间. 2) 计算机随机产生的 100 000 个 64 维的均匀分布的合成数据,其中每一维值的范围也在 0 和 1 之间. 在下面的一系列实验中,本文分别将索引磁盘块访问数及 CPU 运算开销作为衡量查询性能的 2 个指标.

#### 4.1 维数对查询效率的影响

在第一组实验中,研究维数对 10-NN 查询性能的影响. 实验采用 100 000 个合成数据作为测试数据,其中维数  $D$  分别设为 16、32、48 和 64.

图 6(a)和 6(b)分别从 CPU 开销  $T_{CPU}$  和被访问磁盘块数的 I/O 开销  $P_{I/O}$  两个指标来比较查询性能. 从图中看出,随着维数  $D$  的增加,与其他 5 种方法相比,本文 DDM 索引的  $T_{CPU}$  和  $P_{I/O}$  最高,将使查询时间及磁盘块的访问次数大大减少. 这是因为它

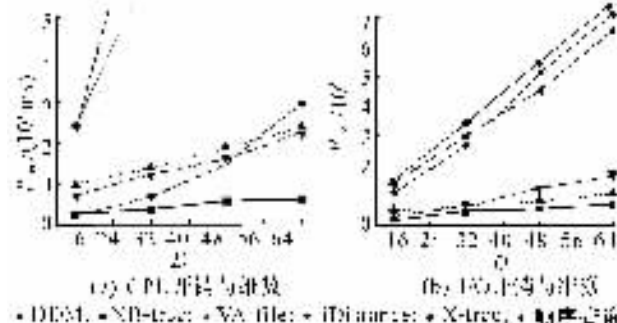


图 6 查询效率与维数关系

Fig. 6 Performance efficiency versus Dimensionality

能够有效地缩减查询过程中的搜索空间. 尽管 iDistance 索引采用聚类和数据分片方法来缩小高维搜索空间, 但随着维数  $D$  的增加, 越来越难找到一个好的聚类方式, 使其始终保持较好的查询响应时间. 而对于本文 DDM 索引来说,  $D$  对其查询效率影响相对较小, 这样使得两者在查询性能上的差异变大. 从图中还可以看出随着  $D$  的增加, NB-tree 的查询性能越来越差, 因为  $D$  增加使得其搜索空间呈指数级增长. 同时 DDM 的  $T_{CPU}$  和  $P_{I/O}$  大大少于 VA-file、X-tree 和顺序查询, 因此随着  $D$  的增加, DDM 方法的开销变化不大,  $D$  对本文 DDM 方法查询效率影响最小.

4.2 数据量对查询性能的影响

本次实验研究数据量对查询性能的影响. 采用两类数据作为测试数据集, 分别将它们分成 5 组执行 10-NN 查询. 图 7(a)、(b) 从 CPU 和 I/O 开销两方面比较了 6 种方法在查询性能上的差异,  $D=32$ . 实验表明, 随着数据量  $S$  的增加, 本文 DDM 方法索引的 CPU 开销  $T_{CPU}$  要低于其他 5 种方法, 其原因与 4.1 节所述相似, 同时可以看出 VA-file 的  $T_{CPU}$  要远远高于 iDistance 和 DDM, 因为它在查询过程中需要进行 CPU 密集运算的解码操作. 综上所述, 与其他 5 种方法相比, 随着  $S$  的增加, DDM 方法的开销  $T_{CPU}$  和  $P_{I/O}$  最小, 查询效率最高, 且  $S$  对其查询效率影响较小. 同时 DDM 在真实数据下的查询性能比合成数据要好, 这是因为真实数据的数据偏

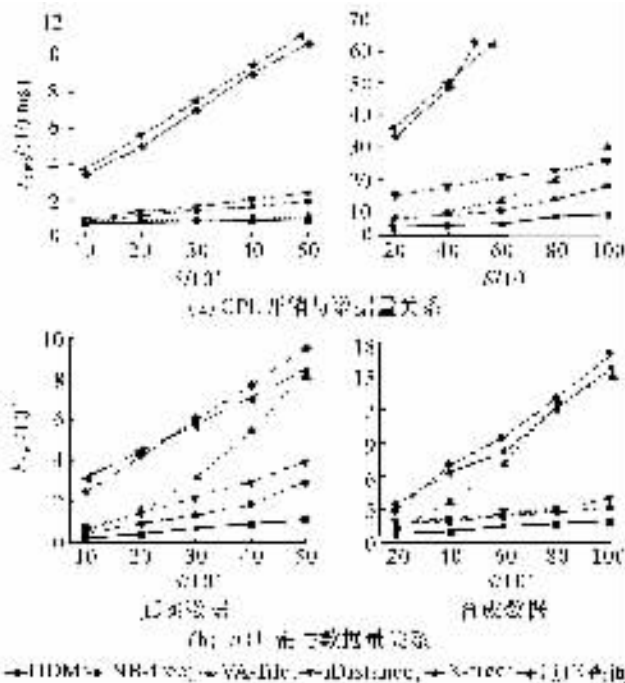


图7 CPU 和 I/O 开销与数据量关系  
Fig. 7 CPU and I/O costs versus data size

斜性比合成数据要好.

4.3  $k$  近邻查询的性能比较

在本次实验中, 本文对其  $k$ -NN 查询性能做一个评估. 同样采用真实和合成数据作为测试数据集, 研究不同查询参数  $k$  值对  $k$ -NN 查询性能的影响, 如图 8(a)、(b) 所示. 其中真实数据维数  $D=32$ , 合成数据中  $D=64$ . 可以看出, 随着  $k$  值的增加, DDM 无论在 I/O 还是 CPU 计算开销方面都要明显优于其他 5 种方法. 在图 8(a)、(b) 的这些索引中, X-tree 和 NB-tree 的查询开销仍然非常高. iDistance 和 VA-file 的查询开销非常接近. 这是因为当维数  $D$  增大到 64 时, iDistance 的 CPU 开销  $T_{CPU}$  提高, 对候选点过滤能力在下降, 导致其查询性能接近 VA-file, 如图 8(a) 右图合成数据所示. 同时也可以看出, DDM 对于  $D$  较低的真实数据具有较好的过滤效果, 使得它优于其他索引方法, 因此随着  $k$  值的增加, 本文 DDM 方法始终保持较低的查询开销.

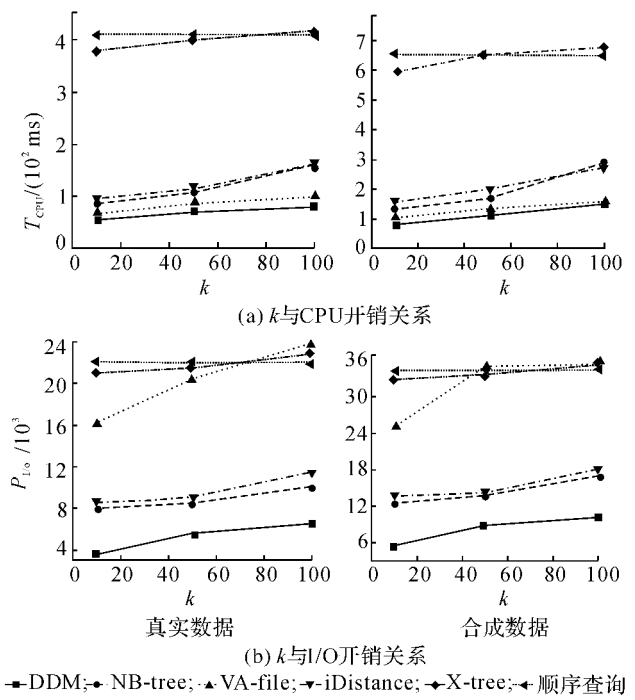


图8  $k$  与 CPU 和 I/O 开销关系  
Fig. 8  $k$  versus CPU and I/O costs

4.4 聚类个数对  $k$ -NN 查询性能的影响

本节研究聚类个数  $T$  对 10-NN 查询性能的影响. 图 9(a) 和 (b) 分别为  $T$  对 I/O 和 CPU 开销的影响. 随着  $T$  的增加, 包括 I/O 开销  $P_{I/O}$  和 CPU 开销  $T_{CPU}$  的查询效率开始是缓慢减少. 因为随着  $T$  的增加, 平均搜索空间在减少, 但减少的幅度是缓慢的. 当  $T$  超过一定数目时, 会使得各个类超球相互重叠, 导致查询的  $T_{CPU}$  提高, 查询的  $P_{I/O}$  基本保持不变, 因此可以将  $T$  作为一个查询性能优化的调整因子.

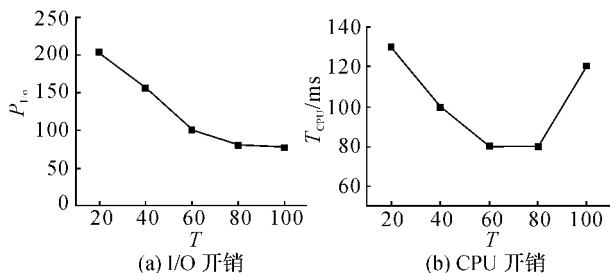


图 9 T 对 k-NN 查询性能的影响

Fig. 9 T versus k-NN query performance

### 5 结 语

本文提出一种基于 DDM 的高维索引方法. 该方法通过预先对高维数据进行聚类, 然后对其每个点分别计算对应的始点和质心距离, 同时结合其所在类的编号生成索引键值, 并采用 B<sup>+</sup> 树对其建立索引. 较其他方法, 理论和实验都表明, DDM 能够有效地缩小搜索空间, 从而明显减少距离计算的开销, 优于其他同类索引方法, 如 iDistance、NB-Tree 和 VA-file.

### 参考文献 (References):

[1] BOHM C, BERCHTOLD S, KEIM D. Searching in high-dimensional spaces; index structures for improving the performance of multimedia databases [J]. *ACM Computing Surveys*, 2001, 33(3): 322 - 373.

[2] GUTTMAN A. R-tree: A dynamic index structure for spatial searching [C]// *ACM SIGMOD International Conference on Management of Data*. Boston: ACM, 1984: 47 - 54.

[3] BECKMANN N, KRIEGEL H P, SCHNEIDER R, et al. The R\* -tree: An efficient and robust access method for points and rectangles [C]// *SIGMOD International Conference on Management of Data*. Boston: ACM, 1990: 322 - 331.

[4] BERCHTOLD S, KEIM D A, KRIEGEL H P. The X-tree: An index structure for high-dimensional data [C]// *22th International Conference on Very Large Data Bases*. Boston: ACM, 1996: 28 - 37.

[5] WEBER R, SCHEK H, BLOTT S. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces [C]// *24th International Conference on Very Large Data Bases*. New York: Morgan Kaufmann Publishers, 1998: 194 - 205.

[6] BERCHTOLD S, BOHM C, KRIEGEL H P, et al. Independent quantization: An index compression technique for high-dimensional data spaces [C]// *16th International Conference on Data Engineering*. [S. l.]: IEEE Computer Society, 2000: 577 - 588.

[7] FONSECA M J, JORGE J A. NB-Tree: An indexing structure for content-based retrieval in large databases [C]// *The 8th International Conference on Database Systems for Advanced Applications*. Kyoto, Japan: IEEE Computer Society, 2003: 267 - 274.

[8] JAGADISH H V, OOI B C, TAN K L, et al. iDistance: An adaptive B<sup>+</sup>-tree based indexing method for nearest neighbor search [J]. *ACM Transactions on Database Systems*, 2005, 30(2): 364 - 397.

[9] University of California, Irvine, UCI KDD Archive [EB/OL]. [2005-10-10]. www.kdd.ics.uci.edu.

下期论文摘要预登

## 两级雾化高压细水雾灭火喷头的实验研究

邓 东<sup>1,2</sup>, 周 华<sup>1</sup>, 杨华勇<sup>1</sup>

(1. 浙江大学 流体传动及控制国家重点实验室, 浙江 杭州 310027; 2. 博世汽车柴油系统股份有限公司, 江苏 无锡 214028)

摘 要: 为了解决细水雾灭火喷头喷雾保护半径小的问题, 研制了一种新型两级雾化高压细水雾灭火喷头. 计算了喷头流量和索太尔雾滴直径  $d_{sauter}$ . 实验测量了喷头在地面的喷雾密度分布, 并与另外 3 种形式雾化喷头的测量结果进行了对比. 采用风速计测量了喷雾速度, 并采用相位多普勒粒子测速仪测量验证了  $d_{sauter}$ . 考虑油盆火焰位置的变化, 进行了一系列的细水雾扑灭“2B”油火实验, 并测量了灭火过程中空间温度场的变化. 结果表明, 该喷头可以产生  $d_{sauter}$  小于 250  $\mu\text{m}$  的细水雾, 喷雾保护半径为 2 m, 在半径为 0~1.5 m 时都具有高效的灭火能力.

关键词: 细水雾; 喷头; 雾化; 灭火; 温度测量