维普资讯 http://www.cqvip.com

·数据库 聚类索引树 检索算法 R树 索引结构 (18)

计算机科学1999Vol. 26№. 12

62-68

一种新的通用聚类索引树及其相似检索算法

A Newly Generalized and Dynamic Recursive Clustering Index Tree and Its Related Similarity Retrieval Algorithms

汪晓岩 胡庆生 庄镇泉 TP311.13

(中国科学技术大学电子技术部 合肥 230026)

Abstract This paper introduces a newly generalized structure for the similarity retrieval of high-dimensional feature vectors called the recursive clustering index tree, which is dynamics, simplicity and suited for various objects. Because the mutually overlapping bundling spheres of cluster may increase the number of search paths, we also propose a similarity retrieval method using pruning branches' strategy. Based on this structure, the retrieval effectiveness of this approach is less than that of the exhaustive search and the similarity retrieval method based on SS-tree, and the retrieval cost of this method is a third of that of the exhaustive search when the recall ratio is less than 10%.

Keywords High-dimensional feature vectors, Recursive clustering indexing tree, Similarity retrieval method, Pruning branches' strategy

1. 引言

数据库的查询操作一般依赖于物理层的特殊算法的支持，尤其在许多面向对象和空间数据库的应用领域，如在文献数据库、多媒体数据库、金融数据库、CAD数据库等的检索操作中，都需要特殊检索算法支持对象的相似检索。数据库中的对象常常用高维特征矢量表示，因此对象的相似检索问题实际上归结为高维特征矢量的相似检索问题。由于用于表示对象的多维特征矢量的维数一般都很高(从几十到几百都是可能的)，在进行相似检索时，相似度的计算和检索费用以及主存空间需求都很大，有时会导致系统过载。因此，适用于多维特征矢量的索引结构和存取方法已成为面向对象和空间数据库技术发展的具有挑战性的问题之一，到目前为止，这方面的研究成果已有不少，文[1]对到1998前的大部分主要的多维数据存取方法进行了系统的分类和总结。在其所介绍的方法中，主流方法都是采用树索引结构，主要有基于Hashing索引结构系列、基于R树索引结构系列和基于K-D树索引结构系列。

基于Hashing索引结构系列的典型范例是网格文件(grid files)[4]和Buddy树[1]。这种类型索引结构的缺点是桶块(buckets)分割或数据分类要按严格的要求进行，尤其对于不均匀和高维数据对象，分割结果会产生过低的空间利用率问题。虽然利用索引树结构能够缓解这一问题，但是索引树结构的节点数会随着特征矢量的维数增加而超线性增长。对于均匀分布的数据对象，平均索引节点数是$O(n^{1+(d-1)/(d+1)})$，其中，b是桶块的容量，d是特征矢量的维数，其平均空间利用率大约是69%。

基于R树的索引结构系列的典型范例有R树[5]、R*树[6]、X树[7]和SS树[2]等。这种类型的索引结构的特点是：索引树上每一个节点v都有相应的桶块和d维约束区间$I^d(v)$。如果v是一个内节点，节点v的儿子节点是v_i，则$I^d(v_i)\in I^d(v)$，在索引树的同一层上，内节点的约束区间可以相互重叠。如果v是叶节点，则$I^d(v_i)$是d维最小约束区间。这种类型的存取结构的优点是索引树结构是平衡的，且桶块分割的自由度较大。缺点是由于每一维的约束区间都要储存，需要较多的存储空间；此外，由于允许约束区间的相互重叠，由此也导致了搜索路径的平均数量的增加。因此，减少约束区间重叠，提高搜索效益是这类存取索引结构需要解决的主要问题。

基于K-D树索引结构是一种二分索引树结构，典

汪晓岩 博士研究生，主要从事信息与信号处理方面的研究。**胡庆生** 博士后，主要研究领域为VLSI设计，验证与优化算法。**庄镇泉** 教授，博士生导师，主要从事智能信息处理方面的教学和科研工作。

维普资讯 http://www.cqvip.com

型范例有 K-D 树[9]、K-D-B 树[7]、TV 树[10]和 LSD[5]树[1]等。这种类型的索引结构特点是：d 维数据桶块的分割用 $d-1$ 维超平面表示，在索引树结构中，内节点用超平面的维数和位置表示，很明显，这种分割方法避免了过大的索引树管理问题，但是，限制了分割的自由度。这种类型的存取索引结构的共同缺点是，对于给定的数据集分布，并非总能找到一个超平面能够平均分割数据集。此外，由于数据节点散落分布于整个索引树上，数据管理及检索都不方便。

综上所述，对于多维特征矢量的相似检索，主要问题是如何构造多维特征矢量的相似索引树。建立相似索引树的主要目的是为了减少相似查询所花费的时间，提高检索效益。然而，上述三种类型系列的索引树结构基本上是针对不同相似对象提出的索引树结构，分类方法依赖于样本数据，因此，严格地讲属于专用索引树。随着数据库技术的发展，通用索引树结构逐渐受到关注[1,2,4]，文[2,4]就试图总结出一般索引树结构的共同特征。从索引树结构的通用性考虑出发，本文提出了不依赖于样本数据对象的动态层次递归聚类索引树建立算法，并且基于该索引树结构，发展了一种"剪枝"相似查询算法。

本文提出的动态层次递归聚类索引树具有动态、简单、适合各种样本数据对象等特点，针对索引树上同一层的类区域有重叠，造成搜索路径平均数量增加的问题，相应地提出了"剪枝"相似检索算法。实验结果显示，本文提出的相似检索算法比基于 SS 树的相似检索和耗尽搜索法，具有更好的检索效益。

2. 动态层次递归聚类索引树结构

本文提出的动态层次递归聚类索引树建立算法，既考虑到了动态更新(插入和删除操作)的效益和空间利用率，还考虑到了通用性，即该算法通过简单地修正，就可以适用于不同的计算方法(如：欧氏距离和相似系数的计算)，以及各种分布特性的数据对象。递归聚类的基本思想是，先采用"最大距离法"从 N 个特征矢量中选择 M 个类中心矢量，然后，将剩余 $N-M$ 个特征矢量应用相似度相关准则与 M 个类进行合并，递归聚类过程从顶向下进行直到全部特征矢量聚类完成，形成聚类索引树。由于采用了"最大距离法"从 N 个特征矢量中选择 M 个类中心矢量，以及采用"最相似"合并准则等聚类策略，使得递归聚类方法不依赖于样本数据对象的分布特性，且索引树结构具有较好的平衡对称性。而且，分支数 M 对应着特征矢量的分布状态，因而可以按照桶块页的存储特性自由地设置聚类树的分支数。

2.1 聚类算法

设 N 个多维特征矢量集表示为 $X=\{X_1,X_2,\cdots,X_N\}$，聚类算法首先将 N 个特征矢量集设置成一个初始类，然后，将该初始类划分成 M 个子类，表示为 $C=\{C_1,\cdots,C_M\}$，如果子类 $C_i(i=1,\cdots,M)$ 所包含的特征矢量数超过 M 个，则继续下一层递归分类。聚类算法步骤如下：

第一步：从初始类中选择两个最不相似(欧氏距离最大或相似系数最小)的特征矢量 X_1 和 X_2，相似度为(以下如不特别说明则用距离表示相似度)，$D_{12}(X_1,X_2)=\mathrm{MAX}\{D_{ij}(X_i,X_j)\}$，$i\neq j$，$i,j=1,\cdots,N$，从 X 中删除 X_1 和 X_2，令 $M'=2$，并设 X_1 和 X_2 为两个类中心矢量。

第二步：假设已经确定了 M' 个类，求第 $M'+1$ 个类步骤如下：

a)计算剩余 $N-M'$ 个特征矢量中的任一特征矢量 X_i 到 M' 个类中心矢量 $X_j(j=1,\cdots,M')$ 之间距离 D_{ij}，然后，求这 M' 个距离中的最小距离及相应的特征矢量。即：

$$D_{ij}^{min}(X_i^{min})=\mathrm{MIN}\{D(X_i,X_j)\}\ i\in=[1,N-M'];j=1,\cdots,M';X_i,X_i^{min}\in X;X_j\in C$$

b)从 a)步求出的所有 $N-M'$ 个 $D_{ij}^{min}(X_i^{min})$ 中选择一个最大距离 D_{ij}^{max} 以及相应的特征矢量 $X_i^{M'+1}$ 作为候选的第 $M'+1$ 个类中心矢量。用公式表述就是：

$$D_{M'+1}^{max}(X_i^{M'+1})=\mathrm{MAX}\{D_{ij}^{min}(X_i^{min})\}\ i=1,\cdots,N-M';j\in[1,M'];X_i^{M'+1},X_i^{min}\in X$$

c)从 X 中删除 $X_i^{M'+1}$，令 $M'=M'+1$，将 $X_i^{M'+1}$ 作为第 $M'+1$ 个类中心矢量 X_M，$X_M\in C$。

第三步：如果 $M'<M$，转第二步，否则转第四步。

第四步：已知确定了 M 个类，将 X 中剩余的 $N-M$ 个特征矢量与 M 个类进行相似相关合并。令 $N=N-M$，分两步进行。

a)计算 X 中剩余的 N 个特征矢量中的任一特征矢量 X_i 到 M 个类中心矢量 X_j 之间距离 D_{ij}，然后，求这 M 个距离中的最小距离及对应的两个特征矢量。即：

$$D_{ij}^{min}(X_i,X_{M_j})=\mathrm{MIN}\{D_{ij}(X_i,X_j)\}\ i,i\in[1,N];j=1,\cdots,M;X_i,X_j\in X,X_{M_j},X_j\in C$$

b)将特征矢量 X_i 与类中心矢量为 X_{M_j} 的第 $M_j\in[1,M]$ 个类合并，从 X 中删除 X_i，令 $N=N-1$，如果 $N\neq 0$，转 a)继续合并，否则，结束递归聚类过程。

2.2 递归聚类索引树的建立

上面已经给出了聚类算法。下面给出建立递归聚类索引树的步骤如下：

第一步：定义初始 N 个特征矢量集作为一个初始类，设定为层次聚类索引树的根节点。

维普资讯 http://www.cqvip.com

第二步：用上述聚类算法将初始类分成 M 个子类。

第三步：用新生成的 M 个子类进行判别，如果某个类含有 M 个以上的特征矢量，则该类又作为一个新的"初始类"，转第二步，进行下一层递归聚类过程。

用上述递归聚类索引树建立算法生成递归聚类索引树，每一层的内节点数为 M^{level}，其中，*level* 为层号，全部特征矢量都分布在树的叶节点上，图1所示是一个4分支（$M=4$）的递归聚类索引树结构示意图，递归聚类索引树上的每一个节点都对应一个类，类参数由下面定义数据结构描述：

```
Struct Class_ptr{
int level;//类的层号
int Class_No;//在第 Level 层的类索引号
int* Centroid_ptr;//指向类中心矢量的指针
float radius;//类半径，这里指与类中心特征矢量最不
              相似的特征矢量之间的距离或相似系
              数的最低门限值
int* Parant_ptr;//指向该类的父节点的指针
int* Children_ptr[M-1];//指向该类的 M 个儿子节
                         点类索引号的指针
int Children_Sum;//所包含的特征矢量总数（包含儿
                   子节点）
```

2.3 递归聚类索引树的特性

N 个特征矢量划分成 M 个类有很多方法，如 k-平均算法，其分类准则是给定的目标函数最小，换句话说，其聚类条件是一致的，因此不能很好地适应各种数据分布。本文的递归聚类算法选择 M 个类中心矢量的策略是面向数据分布的，即在样本数据对象中，选择 M 个最不相似的特征矢量作为 M 个类的中心矢量，这一策略的优点是具有较好的通用性，即聚类算法适用于各种分布特征的数据对象，且生成的递归聚类索引树结构有较好的平衡对称性，其缺点是，生成 M 个类的每一个类区域大小不等，且可能相互重叠，进而导致相似搜索费用的增加，也可能出现将其它聚类算法（如最邻近算法或 k-平均算法[14]）当作是一个类的特征矢量分布在若干个类的现象。现总结递归聚类索引树结构的特点如下：

·递归聚类索引树是平衡对称的索引树结构，从根节点到叶节点，每层的类节点所包含的特征矢量数是逐渐减少的，同时，内节点对应的类区域半径也渐渐变小，即类中的特征矢量之间的相似度变高（欧氏距离变小或相似系数变大）。

·递归聚类索引树上每一个内节点都有一个对应的类，全部特征矢量都分布在树的叶节点上，每一个内节点都包含 M 个儿子节点，儿子节点对应的类是其父节点对应类的子类。

·树上同一层内节点对应的类，类区域半径大小不同，有可能相互重叠，但决不会存在包含关系，即其中一个类是另一个类的子类。

2.4 插入和删除操作

插入和删除操作对索引树结构的性能影响，是所有索引树结构必须考虑的重要问题。由于本文主要讨论基于递归聚类索引树的建立和相似检索问题，因此，这里仅简单介绍一下基于递归聚类索引树的插入和删除操作的基本方法。

递归聚类索引树的插入方法与 SS 树插入方法不同，当加入新的特征矢量时，首先用剪枝相似检索法查找到具有最小的重叠区域的类。然后，判别是否破坏递归聚类索引树的结构，确切地说，就是判别新的特征矢量与当前类的相关合并是否导致类半径的扩大，以致该类包含同一层的其他的类。因此，递归聚类索引树性能的保持比 SS 树好。其插入的基本步骤是：1）先将新的特征矢量与初始类合并，判断类半径是否扩大，如果类半径扩大了，则从初始类重新开始递归聚类过程，否则，合并有效，继续下一层合并，先求出新的特征矢量与 M 个类的中心矢量最小距离的类；2）将该类作为新的初始类按前述的方法继续进行相关合并直到叶节点为止；3）最后，判别该叶节点所包含的特征矢量数是否大于 M，如果大于 M，则从该内节点开始进行递归聚类过程，否则，整个合并过程结束，修改有关的类参数数据。

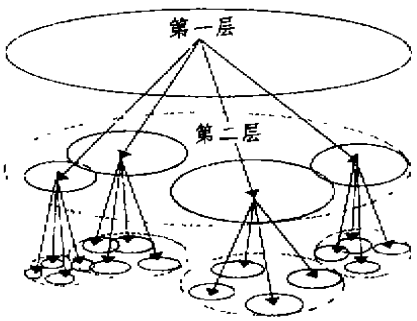


图1 4分支递归聚类索引树结构示意图

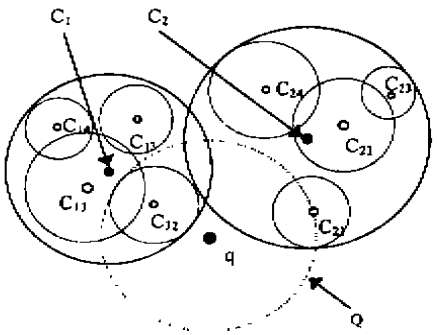


图2 "剪枝"相似检索说明图

维普资讯 http://www.cqvip.com

删除操作相对较容易一些。基本思想是，首先进行精确匹配检索，搜索到与样本特征矢量相匹配的特征矢量所在的叶节点，从该叶节点删除匹配的特征矢量，如果该叶节点所包含的特征矢量数为零，则从该叶节点的父节点开始重新递归聚类过程，然后，修改有关的类参数数据。否则，向上递归修改类的矢量个数参数，结束删除操作。

3. 基于递归聚类索引树的相似检索

相似查询简单地讲就是找出与给定检索对象最相似的对象。在实际应用中，常常用检索 k 个最相似的对象（k 最邻近法）和检索超过相似度门限值的所有相似对象的方法，控制返回的相似对象数目。在各种应用系统中，尤其在交互应用系统中，由于数据对象用高维特征矢量表示，相似判别复杂度非常高，常常希望能够用低概率的不精确性去换取整个相似检索效益的改善。一种通用的方法是给定相似检索对象及检索范围，求出在检索范围内的相似对象。

根据上述递归聚类索引树的特点，传统 B 树搜索算法或第一部分所讨论的其它索引树的搜索算法都不能很好地适用于递归聚类索引树的相似搜索。为此，本文根据递归聚类索引树上内节点对应的类区域可能相互重叠的特点，提出了"剪枝"相似搜索算法。

我们用图2说明"剪枝"相似搜索算法的基本思想。设 C_1 和 C_2 是对应递归聚类索引树某一层上两个内节点的相互重叠的类超球，C_{11}、C_{12}、C_{13}、C_{14} 是 C_1 的子类，C_{21}、C_{22}、C_{23}、C_{24} 是 C_2 的子类，Q 是检索超球。当自顶向下搜索递归聚类索引树时，如果在某一层上能够确定对应内节点的类超球与检索超球不相交，根据递归聚类索引树的下一层内节点对应的类是上一层的子类的特征，则该内节点所关联的路径可以被"剪枝"，这样就在很大程度上减少了要搜索的节点数。例如，在图2中，类超球 C_1 和 C_2 都与检索超球相交，进一步判别仅有 C_{11} 和 C_{21} 与检索超球部分相交，需要在下一层继续判别。而检索超球完全包含 C_{12}，C_{12} 所包含的全部特征矢量都作为相似对象输出。但是，C_{13}、C_{14}、C_{22}、C_{23}、C_{24} 与检索超球不相交，因此，它们所关联的路径被"剪枝"，它们的子类都不必搜索了。

"剪枝法"的关键问题是如何判别类超球与检索超球不相交，受文[11,12,13]的排除搜索思想的启发，本文分别给出了使用欧氏距离和相似系数判别不相似的准则。定理1给出的是使用欧氏距离和相似系数判别不相似的条件，定理2给出的是判别类超球完全相似于检索超球的条件。由于这两种判别条件中仅需要计算类超球的圆心矢量到检索超球的圆心矢量的欧氏距离或相似系数，因此判别效益非常高。在具体叙述"剪枝"相似检索算法之前，先给出有关的定义和定理。

3.1 相似判别的定义与定理

定义1 递归聚类索引树上内节点对应的类，在 d 维特征空间中定义成"类超球"，类中心矢量定义为类超球的"圆心"，称为类超球圆心矢量 C_0；类区域半径定义成类超球的"半径"，欧氏距离用 D_c 表示，相似系数用最低门限值 R_c 表示。

定义2 对于给定的相似检索对象 q 及其检索范围，在 d 维特征空间中定义成"检索超球"，将对应相似检索对象 q 的矢量定义成检索超球的"圆心"，称为检索特征矢量 q；检索范围定义成检索超球的"半径"，欧氏距离用 D_q 表示，相似系数用最低门限值 R_q 表示。

定义3 若 C 为对应递归聚类树上内节点的一个类超球，如果存在至少一个特征矢量 $X\in C$，满足 $D(q,X)\leqslant D_q$ 或 $R(q,X)\geqslant R_q$，则称检索超球 Q 与类超球 C 相似，其中，$D(q,X)$ 表示检索特征矢量 q 到特征矢量 X 的欧氏距离，$R(q,X)$ 表示 q 与 X 的相似系数。

定义4 设 $D(q,C_0)$ 是检索特征矢量 q 到类超球的中心矢量 C_0 的欧氏距离，$R(q,C_0)$ 是检索特征矢量 q 与类超球 C 的中心矢量 C_0 的相似系数。若 $D(q,C_0)\leqslant D_c$ 或 $R(q,C_0)\geqslant R_q$，则 q 在 C 的内部，称 q 与 C 相似，否则 q 在 C 的外部，称 q 与 C 不相似。

定义5 若检索超球 Q 包含类超球 C，则称 C 完全相似于 Q；若类超球 C 包含检索超球 Q，则称 C 部分完全相似于 Q。

要强调的是，检索特征矢量 q 与类超球 C 不相似并不能判定检索超球 Q 与类超球 C 不相似。为此，下面的定理1给出了检索超球 Q 与类超球 C 相似的判别方法。

定理1 当检索矢量 q 与类超球 C 不相似，即 $D(q,C_0)>D_c$ 或 $R(q,C_0)<R_q$ 时，如果 $D(q,C_0)-D_c>D_q$ 或 $R(q,C_0)-R_c+1<R_q$，则类超球 C 与检索超球 Q 不相似或不相交。

证明：如图3所示，当检索矢量 q 与类超球 C 不相似时，还不能判定类超球 C 与检索超球 Q 不相似。假如类超球 C 与检索超球 Q 相似，即类超球 C 与检索超球 Q 相交，如果从类超球圆心矢量 C_0 到检索超球圆心矢量 q 画一条直线相交 C 于 B 点，称为中间矢量。很显然，类超球 C 与检索超球 Q 不相似体现在几何关系是不相交。由定义3知，若对于任意特征矢量 $X\in C$，都有 $D(q,X)>D_q$ 或 $R(q,X)<R_q$，则类超球 C 与检索超球 Q 不相似或不相交。

首先证明欧氏距离的判别式。根据三角不等式，由图3(a)或图3(b)得：

$$D(X,q)\geqslant D(C_0,q)-D(X,C_0)=D_c+D(q,B)-D(X,C_0)$$

维普资讯 http://www.cqvip.com

因为 $X\in C$，有 $D(C_0,X)\leqslant D_c$

由上两式得：

$D(X,q)\geqslant D_c+D(q,B)-D(X,C_0)\geqslant D(q,B)$

因此，如果 $D(q,B)>D_q$，即，$D(q,B)=D(q-C_c)-D(C_c,B)=D(q,C_0)-D_c>D_q$，则类超球 C 与检索超球 Q 不相交或不相似。

下面证明相似系数的判别式：

设 $C_0=(c_1,\cdots,c_d)$；$q=(q_1,\cdots,q_d)$；$B=(b_1,\cdots,b_d)$，它们之间的关系为：$b_i=c_i+\lambda(q_i-c_i)$，不失一般性，假设特征矢量是归一化的，即 $\sum b_i^2=1$，$\sum q_i^2=1$，$\sum c_i^2=1$。C_0 与 B 之间的相似系数为：

$$R(C_0,B)=\frac{\sum c_ib_i}{\sum c_i^2\sum b_i^2}=\sum c_ib_i=\sum c_i(c_i+\lambda(q_i-c_i))=\sum c_i^2+\lambda(\sum q_ic_i-\sum c_i^2)=1+\lambda(R(q,C_0)-1)$$

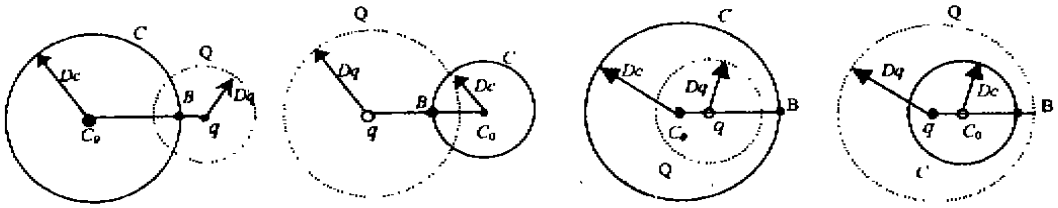
由于中间矢量 B 位于类超球 C 的球面上，因此，中间矢量 B 与类超球圆心矢量 C_0 的相似系数就是类超球的最低相似门限值，即：$R(C_0,B)=R_c=1+\lambda(R(q,C_0)-1)$

由此得出，$\lambda=(R_c-1)/[R(q,C_0)-1]$。如果中间矢量 B 不在检索超球 Q 内，则中间矢量 B 与检索超球圆心矢量 q 的相似系数 $R(q,B)<R_q$，即：

$$R(q,B)=\sum q_ib_i=\sum q_i(c_i-\lambda(q_i-c_i))=\sum q_ic_i+\lambda(\sum q_i^2-\sum q_ic_i)=R(q,C_0)+\lambda(1-R(q,C_0))<R_q$$

将 λ 表达式代入上式，得：$R(q,B)=R(q,C_0)-R_c+1<R_q$，即：

$R(q,X)\leqslant R(q,B)=R(q,C_0)-R_c+1<R_q$。 □



(a) q在C的外部，$D_q<D_c$ Q与C相交　(b) q在C的外部，$D_q>D_c$ Q与C相交　(c) q在C的内部，$D_q<D_c$ C包含Q　(d) q在C的内部，$D_q<D_c$ Q包含C

图3　类超球 C 与检索超球 Q 相交关系

定理2　当检索超球 Q 与类超球 C 相似时，如果 $D_c+D(q,C_0)\leqslant D_q$，且 $D_c\leqslant D_q$ 或 $R(q,C_0)-R_c+1\geqslant R_q$，且 $R_c\geqslant R_q$，则检索超球 Q 包含类超球 C，即类超球 C 完全相似于检索超球 Q。

证明：当检索超球 Q 与类超球 C 相似时，分两种情况考虑：第一种情况是 $D_q<D_c$ 或 $R_q>R_c$，如图3(c)所示，第二种情况是 $D_q>D_c$ 或 $R_q<R_c$，如图(d)所示。

对于图3(c)，当 $D(q,B)\geqslant D_q$ 或 $R(q,B)\leqslant R_q$ 时，C 完全包含 Q，即 C 部分完全相似于 Q。当 $D(q,B)<D_q$ 或 $R(q,B)>R_q$ 时，转变成类似图3(a)所示情况，即 C 部分相似于 Q。

对于图3(d)，当 $D(q,B)\leqslant D_q$ 或 $R(q,B)\leqslant R_q$ 时，Q 完全包含 C，即 C 完全相似于 Q。当 $D(q,B)<D_q$ 或 $R(q,B)>R_q$ 时，转变成类似图3(b)所示情况，即 C 部分相似于 Q。

因此，当 Q 完全包含 C 时，有 $D_c\leqslant D_q$ 或 $R_c\geqslant R_q$，而且，对于任意特征矢量 $X\in C$，必有 $X\in Q$，满足 $D(q,X)\leqslant D(q,B)\leqslant D_q$ 或 $R(q,X)\geqslant R(q,B)\geqslant R_q$。

用欧氏距离判别时，由图3(d)得：

$D(q,B)=D_c+D(q,C_0)$

因此可得：$D(q,X)\leqslant D(q,B)<D_q$，即：

$D_c+D(q,C_0)\leqslant D_q$

用相似系数判别时，由定理1证明可得：

$R(q,B)=R(q,C_0)-R_c+1$

同理可得：$R(q,X)\geqslant R(q,B)>R_q$，即：

$R(q,C_0)-R_c+1\geqslant R_q$ □

3.2　"剪枝"相似检索算法

"剪枝"相似检索算法实现时，自顶向下地搜索递归聚类索引树，首先判别检索超球 Q 与当前层的每一个类超球 $C^i(i=1,\cdots,M^{level})$ 是否相似，若不相似，则对应该类超球的内节点所关联的路径被"剪枝"；否则，再判 Q 是否包含 C^i，若包含，则按相似度大小排序放入输出队列中，若不包含，则执行下一层判别直到叶节点为止，在叶节点上，计算类中每一个特征矢量与检索特征矢量 q 的欧氏距离或相似系数，然后按相似度大小排序输出检索到的相似对象。设 S_Queue 表示判别初始类超球参数数据队列，Selected_Queue 表示候选类超球参数数据队列，Out_Queue 表示相似度排序特征矢量输出队列，level 表示递归聚类索引树的层数。我们考虑到实际应用情况，在下面的算法中增加了返回

维普资讯 http://www.cqvip.com

相似特征矢量数限制的判别。具体算法步骤如下：

第一步：初始化设定，将递归索引树上对应根节点的初始类超球及其参数数据放入 S_Queue 队列和 Selected_Queue 队列，Out_Queue 清空。

第二步：从 S_Queue 中选择一个与检索特征矢量 q 最邻近（欧氏距离最小或相似系数最大）的类超球 C' 及其参数数据，然后，从 S_Queue 中删除类超球 C' 及其参数数据。要说明的是最邻近不一定相似。

第三步：根据定义4，判别类超球 C' 与检索特征矢量 q 是否相似，再按判别结果分别执行下列(a)和(b)两步：

(a)若类超球 C' 与检索特征矢量 q 不相似，根据定理1判别类超球 C' 与检索超球 Q 是否相似，若不相似，转(d)；若相似，再根据定理2判别 C' 是否完全相似于 Q，若完全相似，则转(c)，若不完全相似，则转第四步。

(b)若类超球 C' 与检索特征矢量 q 相似，根据定理2判别类超球 C' 是否完全相似于检索超球 Q，若完全相似，则转(c)，若不完全相似，则转第五步。

(c)若 C' 完全相似于 Q，则将 C' 所包含的全部特征矢量取出，按相似度大小放入 Out_Queue 中，转(d)。

(d)执行"剪枝"操作，即从 Selected_Queue 队列中删除类超球 C' 及其参数数据，转第四步。

第四步：判别 Out_Queue 中包含的相似特征矢量数是否超过给定的 k 个返回值的限制，如果超过这个限制，转第七步；否则转第五步。

第五步：判 S_Queue 队列是否取空，若取空，转第六步；否则转第二步。

第六步：令 $level=level-1$，若 $level>0$，则在递归聚类索引树上执行下一层相似检索，将 Selected_Queue 队列中类超球的所有子类超球及其参数数据放入 S_Queue 队列中；否则，转第七步。

第七步：按相似度大小输出 Out_Queue 中包含的相似特征矢量，结束相似检索。

4. 性能评价和讨论

下面我们对递归聚类索引树的建立费用以及基于递归聚类索引树结构的相似检索费用进行评估和讨论。对于相似检索费用，由于特征矢量的分布特性不同，同等数量的特征矢量所生成的递归聚类索引树结构也会不同，因而相似检索费用也不同。此外，由于检索范围是根据不同应用需要而设定的，检索范围的大小直接影响相似检索费用的多少。因此，相似检索费用受多种因素影响，不能用简单地计算进行预估。下面我们分别讨论这两种费用。

4.1 递归聚类索引树的建立费用

根据递归聚类索引树的结构特点，建立费用由选择 M 个类中心矢量的费用和剩余 $N-M$ 个特征矢量与 M 个类相关合并的费用两部分组成：

a)选择 M 个类中心矢量的费用：

$$N(N+1)/2+\sum_{M'=2}^{M}(1-M')(N-M')$$

b)剩余 $N-M$ 个特征矢量与 M 个类相关合并的费用：

$$(N-M)M$$

每层递归聚类（划分 M 个类）的费用 C_M 是上述二者之和。整个递归聚类索引树建立费用根据特征矢量的分布状态不同而不同，即根据递归次数的不同而不同，对于均匀分布的特征矢量，递归聚类的次数为：$n_c=(\log N)/(\log M)-1$，对于不均匀分布的特征矢量，我们考虑最坏的情况，在每次递归聚类产生的 M 个新类中，$M-1$ 个类都只包含一个特征矢量，而剩余的特征矢量都分布在一个类中，这种情况下的递归聚类次数为：$n_c=(N-M)/(M-1)$。因此，整个递归聚类索引树建立费用为：$n_c\times C_M$，应该小于 $O(N^3)$。

本文的递归聚类索引树的建立算法所需费用比 k-平均算法的费用[14]要低2倍以上。而且，随着特征矢量维数的增加，后者所需费用的增长幅度要远远超过前者的增长幅度。

4.2 相似检索费用

相似检索费用一般是指在给定检索范围和要返回相似对象数目的条件下，获取相似检索结果的平均时间。但是，由于时间费用很大程度上依赖于具体的系统、硬件等多方面因素，因此，本文讨论的相似检索费是指检索的节点数和计算相似度的节点数之和作为费用评估指标。实验使用的样本集是控制随机数产生的随机分布（均匀分布和不均匀分布）的100维特征矢量集，特征矢量的每维数值都在[0,1)之间，且归一化。样本集的总数为100,000个100维特征矢量。

为了对本文算法与 SS 树和耗尽搜索的相似检索费用进行比较，相似检索费用都是用相对耗尽搜索的相似检索费用的比值表示，如图4和图5所示。本文提出的"剪枝"相似检索算法，在比较判断时，预先删除了大量不相似节点，减少了大量的节点计算费用，从而节省了整个相似检索费用。随着检索范围的增大（欧氏距离变大或相似系数变小），本文提出的"剪枝"相似检索算法的相似检索费用以及基于 SS 树的相似检索费用都相应地增大，不过在检索范围较小时，比较基于 SS 树的相似检索算法，本文算法的相似检索费用更小。在检索率为10%以下时，本文算法所需费用仅为耗尽搜索法的费用的1/3以下。

维普资讯 http://www.cqvip.com

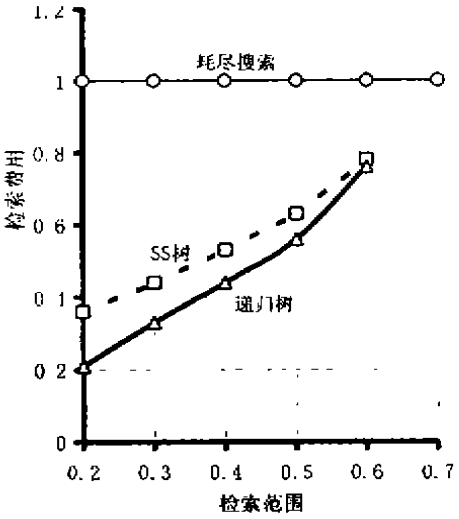


图4 检索范围采用欧氏距离的相似检索费用

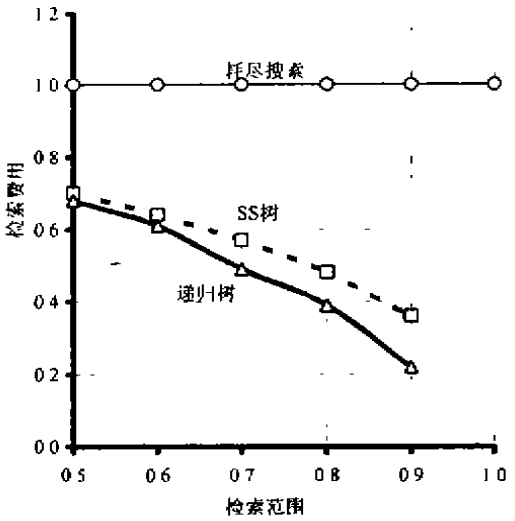


图5 检索范围采用相似系数的相似检索费用

结论 本文提出的通用动态递归聚类索引树，适用于各种分布特性的高维特征矢量集，聚类准则不依赖于具体的样本集，结构具有较好的平衡对称性，基于该索引树的"剪枝"相似检索算法，比较基于 SS 树的相似检索算法和耗尽搜索法，具有较好的相似检索效益。下一步，我们将尝试利用特征矢量集的一些先验知识，如分布状态、每维的重要性等特性，进一步改善聚类索引树的性能。

参 考 文 献

1 Gaede V, Gunther O. Multidimensional Access Methods. ACM Comput. Surv. ,1998,2(30):171～231

2 White D A, Jain R. Similarity Indexing with the SS-tree. In: Proc. 20th Intl. Conf. on Data Engineering. New Orleans, Louisiana, USA, 1996. 516～523

3 Henrich A. The LSD^h-tree. An Access structure for feature vectors. In: Proc. 14th Intl. Conf. on Data Engineering. Orlando, Florida, USA, 1998. 362～369

4 Aoki P M Generalizing "Search" in generalized search trees. In Same to[3] 381～389

5 Guttman A. R-tree: A dynamic index structure for spatial searching. In: Proc. of ACM SIGMOD Intl. Conf. on Management of Data. 1984. 47～57

6 Beckmann N, et al. The R*-tree: An efficient and robust access method for points and rectangles. In: Proc. of ACM SIGMOD Intl. Conf. on Management of Data. 1990. 322～331

7 Berchtold S, et al. The X-tree: An index structure for high-dimensional data. In: Proc. of the 22th Intl. Conf. on Very Large Data Bases, Bombay, India, 1996. 28～39

8 Bentley J L. Multidimensional binary search trees used for associative seraching. Commun. ACM, 1975, 18(9): 509～517

9 Robinson J T. The K-D-B-tree. A search structure for large multidimensional dynamic indexs. In: Proc. ACM SIGMOD Conf. 1981 10～18

10 Lin K -I, et al. The TV-tree-an index structure for high-dimensional data. VLDB Jounal, 1994, 3(4): 517～542

11 Shasha D, et al. New techniques for best-match retrieval. ACM Trans. on Information System, 1990, 8(2): 141～157

12 Burkhard W A, Keller R M. Some approaches to best-match file searching Comm. of ACM, 1973, 16(4): 230～236

13 Shasha D, et al. New tchniques for best-match retrieval. ACM Trans. on inf. sys. ,1990, 8(2): 140～158

14 Tsao E C, Bezdek J C Fuzzy Kohonen clustering network. Pattern Recognition, 1994, 27(5): 757～764

15 Salton G, et al A Vector Space Model for Automatic Indexing. Comm of the ACM, 1975, 18(11): 613～620