LBD:基于局部位码比较的高维空间 KNN 搜索算法

梁俊杰[1,2] 冯玉才[1]

(华中科技大学计算机学院 武汉 430074)[1] (湖北大学数学与计算机科学学院 武汉 430062)[2]

摘 要 利用高维数据空间合理划分,提出一种简单有效的 KNN 检索算法—LBD。通过聚类将数据划分成多个子集空间,对每个聚类子集内的高维向量,利用距离和位码定义简化表示形式。KNN 搜索时,首先利用距离信息确定候选范围,然后利用某些维上的位码不相同信息进一步缩小搜索范围,提高剪枝效率。位码字符串比较时,按照维度贡献优先顺序,大大加快非候选点过滤。LBD 利用特殊的 B^+ 树组织,降低 I/O 和距离计算代价。采用模拟数据和真实数据,实验验证了 LBD 具有更高的检索效率。

关键词 高维索引,KNN 查询,位码,近似向量

LBD:Exploring Local Bit-code Difference for KNN Search in High-dimensional Spaces

LIANG Jun-Jie[1,2] FENG Yu-Cai[1]

(College of Computer Science & Technology, Huazhong University of Science and Technology, Wuhan 430074)[1]

(Faculty of Mathematics & Computer Science, Hubei University, Wuhan 430062)[2]

Abstract Recent advances in research fields like multimedia and bioinformatics have brought about a new generation of high-dimensional databases. To support efficient querying and retrieval on such databases, we propose a methodology exploring Local Bit-code Difference (LBD) which can support k-nearest neighbors (KNN) queries on high-dimensional databases and yet co-exist with ubiquitous indices, such as B^+-trees. On clustering the data space into a number of partitions, LBD extracts a distance and a simple bitmap representation called Bit Code (BC) for each point in the database with respect to the corresponding reference point. Pruning during KNN search is performed by dynamically selecting only a subset of the bits from the BC based on which subsequent comparisons are performed. In doing so, expensive operations involved in computing L-norm distance functions between high-dimensional data can be avoided. Extensive experiments are conducted to show that our methodology offers significant performance advantages over other existing indexing methods on both real life and synthetic high-dimensional spaces.

Keywords High-dimensional index, KNN search, Bit code, Approximate vector

1 引言

近几十年来,在很多领域出现了高维数据应用,例如数据仓库和基于内容的多媒体信息检索。数据以高维向量的形式存在,有的维度甚至高达几百或几千维,并且数据量通常也较大。KNN 查询(K-Nearest Neighbor search)是这类应用中常见的检索方式,给定查询对象,要求在高维空间中查找出与查询对象距离最近的 K 个点[1]。如何提高高维向量空间的 KNN 检索效率成为很多学者的研究热点。

在高维向量空间中实施 KNN 查询,影响检索效率提高的一个重要因素是高维向量距离计算的代价相当大。为了减少距离计算,目前已有很多文献对此提出了不同的解决方法,其中以通过分割数据或空间建立索引为主要思路[2,3]。但是这类方法往往不可避免地遭遇"维数灾难"困扰,不能很好地应用于高维(超过几十维)数据空间检索[4]。因此,最近几年提出一种新思路:简化高维向量表示形式,例如利用近似向量代替高维向量的表示方法,或者将高维向量转换为一维表示形式等,降低维度增加带来的不利影响。VA-file 和 iDistance 索引结构分别体现了这两种思想,但是都存在不同方面的问题,影响了它们在实际中的应用。

本文结合近似向量表示法和一维向量转换法两者思想,提出利用局部位码不相同信息实现快速 KNN 搜索,称为 Local Bit-code Difference (简称 LBD)。通过特殊的 B^+ 树,LBD 有效组织高维向量的距离表示和位码表示,KNN 检索时缩小需要搜索的范围。在利用位码过滤阶段,按照各维距离贡献大小顺序,比较部分维上不相同位码值,加快非候选数据过滤速度。实验证明,利用 LBD 进行 KNN 搜索,能降低 I/O 和距离计算代价。

2 相关工作

多维或高维索引技术已经得到数据库研究领域很多学者的关注。随着更高维数据(上百甚至上千维)的出现,早期提出的高维索引技术逐渐显现出很多不足之处。特别是,针对 30~50 维数据设计的索引结构在处理更高维数据检索时,性能往往不如顺序扫描,这就是所谓的"维数灾难"影响[5,6]。为了克服该影响,新近提出的高维索引结构可以分为以下几类:

(1)数据降维方法:该类方法首先将高维数据映射到低维数据空间,然后利用多维索引结构组织低维空间数据[10,11]。映射过程通常会将原始数据压缩到信息保留最大的低维空间,但是不可避免地损失部分信息,因此这类方法适用于近似检索要求。

(2)近似向量表示法:通过采用简单向量近似地表示原高维向量,达到简化搜索空间目的,与本文索引有关的两个例子 VA-file[4] 和 BID[7]。VA-file 利用数据压缩加速顺序扫描,基

梁俊杰 博士研究生,主要研究方向:多媒体数据库、基于内容的息检索和高维索引结构;冯玉才 教授,博士生导师,主要研究方向:数据库、多媒体技术、GIS。

本思想是通过将数据空间分割成个矩形单元格，对每个单元格采用一个长度为 b 个位(Bit)的字符串标识。这样，所有的高维点向量就可以采用其所属的单元格标识符近似表示。因此，VA-file 的 KNN 搜索就只需在这些近似向量中进行顺序扫描，但是 VA-file 的搜索效率受数据分布影响较大。和 VA-file 不同的是，BID 利用与参考点的相对位置关系将高维向量近似表示成位码字符串，查询时根据查询点和某个点之间的位码不相同位数，判断该点是否为候选点，实现主存环境的近似 KNN 查询。虽然 BID 查询过程不需要距离计算，检索速度较快，但是不能保证较高的 KNN 检索精度。

(3)一维转换表示法：将高维向量转换成一维表示形式，金字塔技术(Pyramid-Technique，PT)[8] 和 iDistance[9] 体现了这种思想。金字塔技术将 d 维数据空间划分成 $2d$ 个金字塔，根据每个点所在的金字塔序号 i 以及距离顶点的高度 h，通过函数($key=i+h$)转换得到它的关键字值(key)，并将它们组织成 B^+ 树索引结构。金字塔技术的检索效率受维度影响不大，但是由于它只适用于矩形范围检索，限制了在实际中的应用。iDistance 利用与参考点的距离将高维点向量转换为一维表示，由于提供了灵活的空间划分和参考点选择方式，使得 iDistance 具有较好的检索性能。但是，由于转换方式引起的不同点可能具有相同一维数值，并且随着维度的增高(>30)，这一现象越来越明显，因此检索过程会引入过多的误中点，对这些点的距离计算大大降低检索效率。

3 LBD 索引树

为了利用 LBD 方法进行快速有效的 KNN 搜索，可以将 LBD 和 B^+ 树结合，这样可以利用 B^+ 树现有的很多算法。

LBD 索引树可以分为三层(如图 1)。第一层是 B^+ 树，存储高维向量转化的一维向量值，该过程可以利用多种方法实现。为了简单，我们选用文[9]中给出的算法，利用该算法每个高维数据被划分到某个聚类空间内。假设每个聚类子集有唯一的聚类标示符，则位于聚类 i 内的任意点 P 的一维索引值为 $\mathrm{Key}(P)=i*c+dist(P,O_i)$。其中 O_i 是聚类 i 的质心，c 是常数，$dist(.)$ 是距离函数。

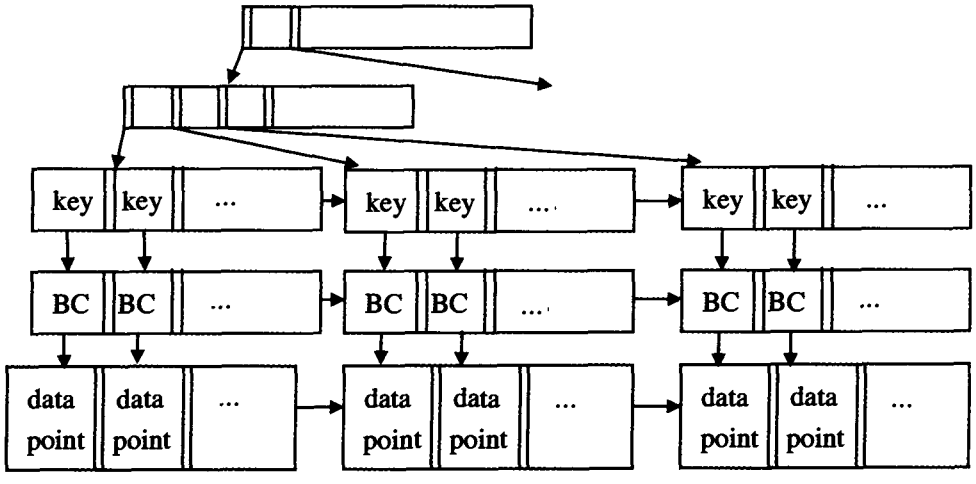

图 1 LBD 索引树结构

LBD 树的第二层是位码层。任意一个 N-维点向量的位码表示是一个长度为 N 的字符串，并且该字符串中的每一位代表高维向量相应维的数值与它所属的聚类质心该维数值的大小关系。该层的所有位码以顺序文件形式存储，和索引树是分开的，这样便于快速存取。

LBD 树的第三层是数据层。这一层存储的是真正的高维向量数据。

定义 1(位码，Bit Code) 任意一个 N-维点向量 $P(p_1,p_2,\cdots,p_N)$，若它所在的聚类质心为 $O(o_1,o_2,\cdots,o_N)$，则 P 的位码定义为

$$BC_p=(BC_{p_1},BC_{p_2},\cdots,BC_{p_N})$$

其中 $BC_{p_i}=\begin{cases}1 & p_i\geqslant o_i\\ 0 & p_i<o_i\end{cases}(1\leqslant i\leqslant N)$

相应地，LBD 索引树的构造过程可以分为三个阶段。首先，将整个数据空间划分成几个聚类子集，并计算各个点向量的索引关键值(距离值)，组织成 B^+ 树。然后，将 B^+ 树叶节点中的距离值和相应的位码值链接起来。最后，链接位码和对应的数据向量。

在 LBD 索引树中很容易实现插入和删除操作。当需要插入一个节点时，首先确定该点所属的聚类，计算出它的一维距离值和位码值。然后，利用常用的 B^+ 树插入方法，将该点的距离值插入 B^+ 树中。最后，将它的位码值和向量数据插入到相应的位码层和数据层。类似地，删除操作也可以实现。

4 KNN 搜索

4.1 利用位码不相同位求距离下界

定理 1(两点间的距离下界) 在 N 维聚类空间 $D=\{d_1,d_2,\cdots,d_N\}$ 中，聚类质心 $O(o_1,o_2,\cdots,o_N)$，对于任意两点 $P(p_1,p_2,\cdots,p_N)$、$Q(q_1,q_2,\cdots,q_N)$，若存在一个维度子集 $D'\subset D$，且 $\forall_i\in D'$，$BC_{p_i}\neq BC_{q_i}$(P、Q 在第 i 维上的位码不同)，则

$$\mathrm{dist}(P,Q)\geqslant\sqrt{\sum_{i\in D'}(q_i-o_i)^2}$$

证明：P、Q 在第 i 维上的位码不同，表示 P、Q 分别位于过质心与第 i 维坐标轴垂直的超平面的两侧，所以 $\mathrm{dist}(q_i,p_i)\geqslant\mathrm{dist}(q_i,o_i)$，$\forall i\in D'$。又因为 $D'\subset D$，所以

$$\mathrm{dist}(P,Q)=\sqrt{\sum_{i\in D}(q_i-p_i)^2}\geqslant\sqrt{\sum_{i\in D'}(q_i-p_i)^2}\geqslant\sqrt{\sum_{i\in D'}(q_i-o_i)^2}$$

由定理 1 不难看出，在判断某一点 P 是否需要剪枝时，如果能判断 P 和查询点 Q 间的距离下界大于剪枝距离

(Pruning Distance,pd),则可以得出 P 一定不是 Q 的候选节点,这样将 P 剪枝。在实际搜索过程中,可以变换一种方式使用该定理,在利用 LBD 索引树确定需要搜索的 key 值范围后,找出 Q 与该范围内各个聚类质心 O 在哪些维上的距离超过了剪枝距离 pd,则该聚类内凡是在这些维上位码与 Q 点的位码不相同的点都可以剪枝。

4.2 维度优先级

为了加快寻找 Q 与聚类质心 O 在哪些维上的距离超过 pd,需要区分聚类数据在哪些维上距离增长快(即这些维具有较大区分能力),并将它们优先用于距离比较。因此可以对维度进行排序,确定维度对距离的区分能力的优先级列表。

定义 2(维度优先级) 假设 $Q(q_1,q_2,\cdots,q_N)$ 为查询点,$O(o_1,o_2,\cdots,o_N)$ 为某个聚类质心,则 Q 相对于点的维度优先级列表可以采用一个数组 $rank[1..N]$ 来表示,数组中的每个元素代表空间中的某一维,并且

$$q_{rank[1]}-O_{rank[1]}\geqslant q_{rank[2]}-o_{rank[2]}\geqslant\cdots\geqslant q_{rank[N]}-o_{rank[N]}$$

有了维度优先级列表之后,在进行点过滤时,就可以先从优先级高的维度开始。若两点间的距离下界大于剪枝距离 pd,则可以将那些位码在这些维上不同于 Q 的点过滤,不必对它们进行距离计算。照这样,逐渐向优先级低的维度过渡,可以大大加快非候选点过滤速度。

4.3 KNN 搜索算法

根据上述思想,利用 LBD 实现 KNN 搜索可以分为两个阶段进行。

第一个阶段:粗筛选处理。主要实现剪枝处理,首先利用点向量的 key 值定义,界定需要搜索的 key 范围;然后利用 LBD 的位码定义以及定理 1 结论,结合维度优先级思想,进一步对搜索范围内的点进行过滤处理,尽可能减少下一阶段需要处理的点个数。

第二个阶段:细提炼处理。对第一阶段处理的结果集合中的点具体计算与 Q 点的距离,产生最后的最近邻集合。

具体的 KNN 搜索算法描述如下:

```
Algorithm KNNSearch(point Q, real step, integer K, point_sets
CurrentKNNList)
    R=r, CurrentKNNList={};
    Initialize lp[]=∞,rp[]=0;
    pd=MaxDist(CurrentKNNList,Q);
    While R<pd or |CurrentKNNList|<K do
        R=R+step;
        For each cluster cl_i with the centriod O_i do
          If cl_i intersects sphere(Q,R)then
            SearchRange=Determine_Range(cl_i,Q,key_low,key_high)
            SearchRange=BC_Pruning(SearchRange,O_i,Q, pd);
            If MinKey(SearchRange)≤lp[i] then
              lp[i]=SearchDown(lp[i],MinKey(SearchRange))
            If MaxKey(SearchRange)≥rp[i] then
              rp[i]=SearchUp(rp[i], MaxKey(SearchRange))

Function BC_Pruning(SearchRange,O,Q, pd)
  For t=1 to N do
    rank[t]=rank_dimension(Q,O)
  width=1;
  while(width<N)do
    n<-(N^width);
    List[n][width]<-(1..N^width);
    For i=1 to n do
      DimDist=0;
      For j=1 to width do
        DimDist=DimDist+dist(q_rank[List[i][j]],o_rank[List[i][j]])
      If (DimDist<pd)
        break
    else
      For each point P in SearchRange do
        diff_num=0;
        For j=0 to width do
        If BC_p,rank[List[i][j]] ≠ BC_q,rank[List[i][j]] then
          diff_num=diff_num+1;
        If diff_num=width then
          Remove P from SearchRange
        width++;
      Return SearchRange;

Function SearchUp(startkey, endkey)
  For each point P with key from startkey to endkey do
    If|CurrentKNNList|==K then
      CurrentKNNDist=MaxDist(CurrentKNNList,Q)
      If dist(P, Q)<CurrentKNNDist then
        Remove the point with largest distance from Current-
        KNNList;
        Insert P into CurrentKNNList
    Else
      If|CurrentKNNList|<K then
        Insert P into CurrentKNNList
return endkey;
```

上述算法中,函数 MaxDist(CurrentKNNList,Q)返回当前 KNN 结果集 CurrentKNNList 中所有点与 Q 的最远距离,并将该值作为剪枝判断依据。Determine_Range(cl_i,Q,key_{low},key_{high})确定在聚类 cl_i 中需要搜索的范围,并修改 key_{low},key_{high} 参数值。BC_Pruning ()是根据位码和维度优先级概念设计的,完成对 SearchRange 逐维剪枝。SearchUp (startkey, endkey)从 LBD 树中 key 值为 startkey 的点开始,沿着它的右兄弟节点继续往下搜索,直到遇到的 key 值大于 endkey 为止,并且将停止位置的 key 值存储在 rp[]数组中。SearchDown()和 SearchUp()的执行过程相类似,不同之处在于两者的搜索方向正好相反,SearchDown()的搜索是沿着左兄弟方向进行的。

4.4 一个实例

LBD 的 KNNSearch 算法看起来很复杂,为了更清楚地说明它的实现过程,现就一个实例进行具体解释。为了描述方便,在例子中用 L_1 距离代替欧氏距离。假设 Q(0.9,0.1,0.55,0.7,0.35),聚类质心 O(0.5,0.6,0.5,0.5,0.5),Q 的相关信息以及相对 O 的维度优先级列表如表 1 所示。

表 1 一个查询实例

Q	(0.9,0.1,0.55,0.7,0.35)
O	(0.5,0.6,0.5,0.5,0.5)
$\lvert q_i-o_i\rvert$ $(1\leqslant i\leqslant 5)$	(0.4,0.5,0.05,0.2,0.15)
Key(Q)	1.3
BC_Q	10110
rank	[2,1,4,5,3]

假设有 $P_1,P_2,\cdots,P_9$ 9 个点属于该聚类,各点的信息列于表 2。现在的问题是:如何在这 9 个点中找到距离 Q 最近的两个点(即 $K=2$)? 下面给出应用 LBD 方法求解该问题的过程。

假设剪枝距离 $pd=0.5$,则根据 LBD 索引树和定理 1 得出的结论,对该问题的解决分以下几个步骤完成:

(1)界定需要搜索的 key 值范围[key_Q-pd,key_Q+pd]。则需要搜索的 key 值区间为:[0.8,1.8],由于 $P_1,P_2,\cdots,P_9$ 9 个点的 key 都属于这个范围,因此这一过程剪枝的点:无。

(2)剩余点{$P_1,P_2,P_3,P_4,P_5,P_6,P_7,P_8,P_9$},则其元素个数 $9>K$,继续…

(3)根据 rank[]定义,首先判断剪枝不等式 dist($q_{rank[1]}$,$o_{rank[1]}$)$\geqslant pd$ 是否成立,事实上 dist($q_{rank[1]}$,$o_{rank[1]}$)=0.5$\geqslant pd$,所以根据定理 1 可以将 $P_1,P_2,\cdots,P_9$ 9 个点中与 Q 在第 2 维(rank[1])上具有不同 BC 码的点剪枝,即将 $BC_{p_{i,2}}\neq BC_{q_2}=0(1\leqslant i\leqslant 9)$ 的所有 P_i 剪枝,这样过滤掉的点有:P_1,P_4,P_6,P_7,P_9。

表 2　实例中的点对象

P	Data	key	BC
P_1	(0.1,0.9,0.3,0.55,0.0)	1.45	01010
P_2	(0.35,0.2,0.95,0.8,0.9)	1.7	00111
P_3	(0.85,0.15,0.6,0.65,0.45)	1.1	10110
P_4	(0.2,0.8,0.65,0.95,0.4)	1.2	01110
P_5	(0.92,0.15,0.4,0.6,0.25)	1.32	10010
P_6	(0.65,0.8,0.1,0.4,0.3)	1.05	11000
P_7	(0.15,0.9,0.3,0.1,0.7)	1.45	01001
P_8	(0.4,0.1,0.25,0.7,0.75)	1.3	00011
P_9	(1.0,0,0.99,0.05,0.95)	2.49	11011

(4)剩余点$\{P_2,P_3,P_5,P_8\}$,则其元素个数 $4>K$,继续…

(5)判断维度贡献数组的下一维距离。剪枝不等式 $\mathrm{dist}(q_{rank[2]},o_{rank[2]})=0.4\geqslant pd$ 不成立,并且可以推断出在 rank 数组中的后续某一维上,该不等式也不会成立,所以结束一维剪枝判断,进入二维剪枝判断。

(6)在进入二维剪枝判断时,除了已经处理的一维以外,其他维按照最有可能使得剪枝不等式成立的方向进行,因此从 rank[2]开始,依次判断它与 rank 数组中其他某一维的距离累加值和 pd 的关系。

(7)$\mathrm{dist}(q_{rank[2]},o_{rank[2]})+\mathrm{dist}(q_{rank[3]},o_{rank[3]})=0.4+0.2>pd$,则根据定理 1 可以将剩余点中与 Q 在第 1 维(rank[2])和第 4 维(rank[3])上具有不同 BC 码的点剪枝。即将 $BC_{p_{i,1}}\neq BC_{q_1}=1$ 且 $BC_{p_{i,4}}\neq BC_{q4}=1(1\leqslant i\leqslant 9)$的所有 $P_i(i\in\{2,3,5,8\})$剪枝,这样过滤掉的点有:无。

(8)$\mathrm{dist}(q_{rank[2]},o_{rank[2]})+\mathrm{dist}(q_{rank[4]},o_{rank[4]})=0.4+0.15>pd$,则根据定理 1 可以将 leftover 中与 Q 在第 1 维(rank[2])和第 5 维(rank[4])上具有不同 BC 码的点剪枝。即将 $BC_{p_{i1}}\neq BC_{q_1}=1$ 且 $BC_{p_{i5}}\neq BC_{q_5}=0(i\in\{2,3,5,8\})$的所有 P_I 剪枝,这样过滤掉的点有:P_2,P_8。

(9)剩余点$\{P_3,P_5\}$,则其个数 $2=K$,剪枝停止,因此候选点就只剩下 P_3,P_5 这两个点。

(10)计算 P_3,P_5 与 Q 点的距离:

$$\mathrm{dist}(P_3,Q)=\sqrt{(0.85-0.9)^2+(0.15-0.1)^2+(0.6-0.55)^2+(0.65-0.7)^2+(0.45-0.35)^2}=0.14$$

$$\mathrm{dist}(P_5,Q)=\sqrt{(0.92-0.9)^2+(0.15-0.1)^2+(0.4-0.55)^2+(0.6-0.7)^2+(0.25-0.35)^2}=0.21$$

因此 Q 的最近邻两个点为:$\{P_3,P_5\}$。

5　实验结果与评价

为了验证 LBD 方法的效果,分别使用人工模拟数据和真实数据进行实验,因为 LBD 方法综合了近似向量表示法和一维向量转换法,所以实验中采用代表这两种思想的典型索引 VA-file 和 iDistance 作为比较对象。虽然目前根据这两种思想的索引还有很多,但是我们选择的是与 LBD 相似的索引结构进行比较和分析。

5.1　模拟数据实验

实验 1(图 2)　考察数据量对查询时间的影响。实验中使用的数据是 20 维的向量,数据量从 5 万条到 30 万条,KNN 查询结果集大小 $K=10$。可以看出,无论是利用近似向量表示法还是一维向量转换法,检索性能都比顺序扫描好很多,这是因为顺序扫描需要计算的距离次数随数据量呈线性增长,而利用索引可以使需要搜索的范围缩小,减少距离计算。因此,在下面的实验中,我们重点比较 LBD 和 iDistance、VA-file 之间的区别。

图 2 是数据量对查询时间的影响。

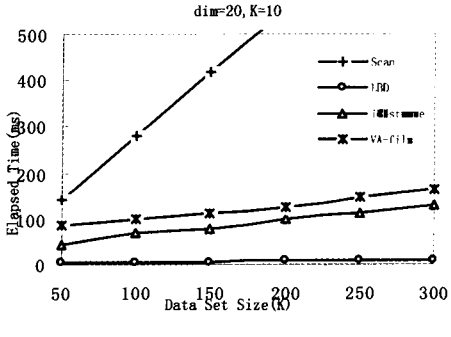

图 2　数据量对查询时间的影响

根据图 2,随着数据量的增加,LBD 的检索性能比只用一种思想设计的索引 iDistance 和 VA-file 都要好,查询花费时间至少减少 80%,并且随数据量增加,这种差距更加明显。这是因为 LBD 利用距离和位码信息进行剪枝处理,并结合局部位码比较的 KNN 检索算法,简单快捷地将大量非候选节点剪枝过滤掉,大大减少距离计算。

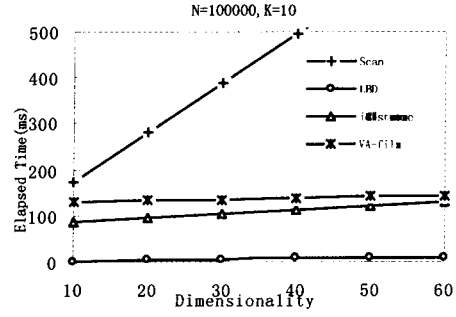

图 3　维度对查询时间的影响

实验 2(图 3)　考察维度对查询时间的影响。实验中生成了维度分别为 10、20、30、40、50、60 的数据,每个数据文件的数据量都是 10 万,KNN 结果集大小 $K=10$。可以看出,维度对于 LBD、VA-file 和 iDistance 的查询时间影响都比较小,维度由 10 变化到 60,查询时间的增加均小于 50%。但是,随维度增加,LBD 的查询时间增长显得更加缓慢;并且与 VA-file、iDistance 相比,LBD 查询时间始终少出 90%。维度的增加给 LBD 带来了更多的优势,从理论上分析,这是因为:随着维度的增加,数据分布稀疏现象愈加明显,分布在空间表面的点越来越多,VA-file 对数据空间均匀分割的单元个数快速增加,造成查询时需要扫描的单元增多,相应地时间也增多;iDistance 由于采用简单的一维距离数值代替高维向量搜索,维度增加使得更多的点具有相同的一维数值,这样检索时难免会引入过多的误中点,影响了检索速度。而从 LBD 索引值的构造过程可以看出,分区可以将那些距离表示无法区分的点进一步进行划分,提高剪枝过滤效率,因此 LBD 对高维空间稀疏数据的划分更加合理。

5.2　真实数据实验

采用来自 68040 幅图像[12]提取的 32 维颜色特征数据,

(下转第 161 页)

补齐法和扩充法的优劣后，认为扩充模型善于寻找知识之间的内在联系，而数据补齐可以提供知识属性的表层分析，二者的结合能更好地挖掘知识的潜在规律，提高决策规则的正确性和可靠性，因此提出了基于差异关系和数据部分补齐的处理方法。差异关系相对于相似关系，当用相似性无法明确两个对象的等价关系时，我们可以转而利用它们的差异性。本文定义了差异关系，对差异矩阵进行扩充使其能适用于不完备信息系统，证明了用差异关系进行属性约简和求核的可行性，并给出了相应的算法。数据部分补齐的目的是通过分析对象之间的差异性更进一步地挖掘信息之间的潜在联系，为后续的决策规则的求取提供更丰富更准确的信息。实验证明，在处理不完备信息系统时，基于差异关系和数据部分补齐的方法能获得更好的分类性能。

参 考 文 献

1 Grzymala-Busse J W, Fu M. A Comparison of several approaches to missing attribute values in data mining. In: Proceedings of the 2nd International Conference on Rough Sets and Current Trends in Computing, Berlin: Springer-Verlag, 2000. 378～385

2 傅媛. 一种基于决策的数据补全方法. 昆明理工大学学报(理工版), 2003, 28(6): 157～160

3 张士林, 毛海军, 赵秋艳. 对于粗糙集应用中数据不全的解决方法研究. 计算机工程与应用, 2003. 10～11, 50

4 Wong K C, Chiu K Y. Synthesizing statistical knowledge for incomplete mixed-mode data. IEEE Trans. on Pattern Analysis and Machine Intelligence, 1987, 9: 796～805

5 Kryszkiewicz M. Rough set approach to incomplete information system. Information Sciences, 1998, 112: 39～49

6 Stefanowski J, Tsoukias A. On the extension of rough sets under incomplete information. In: Proceedings of The Seventh International Workshop on Rough Sets, Fuzzy Sets, Data Mining, and Granular-Soft Computing (RSFDGrC'99), 1999. 73～81

7 朱小飞, 卓丽霞. 一种基于量化容差关系的不完备数据分析方法. 重庆工学院学报, 2005, 19(5): 23～25

8 余英泽, 王国胤, 吴渝. 一种基于 Rough 理论的不完备信息系统处理方法. 计算机科学, 2001, 28(5): 35～38

9 Witten I H, MacDonald B A. Using concept learning for knowledge acquisition. International Journal of Man-Machine Studies, 1988, 27: 349～370

(上接第 148 页)

这组实验(图 4、图 5)主要考察结果集大小对查询时间的影响。随着查询结果集增大，LBD、VA-file 和 iDistance 查询时间缓慢增加，但是总体上 LBD 查询时间明显少于 VA-file 和 iDistance 所用时间，只有 1/6 左右。LBD 索引利用分区，可以将数据合理地划分到不同的分区中，便于检索时剪枝，使得 LBD 更加适用于非均匀分布数据。

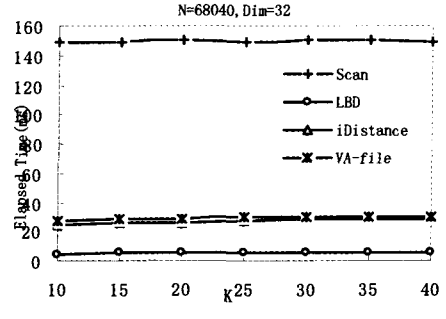

图 4 查询范围对查询时间的影响

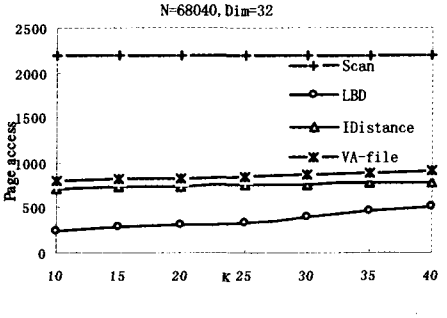

图 5 查询范围对页面访问次数的影响

综合可知，LBD 的查询效率远远优于 SCAN、iDistance 和 VA-file，尽管它们都是高维数据空间检索表现较好的索引结构，特别是随着数据量增大或维度增高这种优势更加明显，并且 LBD 索引结构不受数据分布影响，能很好地适用于各种数据分布情况，所以 LBD 是一种有效的针对高维空间 KNN 检索的索引结构。

结论 本文提出了一种有效的高维数据表示(LBD)，基于此给出一个简单有效的 KNN 搜索算法。在充分考虑到高维空间向量分布特点的前提下，LBD 对高维空间进行聚类划分利用距离索引组织 B^+ 树，并对每个聚类子空间合理分区，通过简单的部分维上的位码字符比较，大大提高 KNN 搜索剪枝效率。实验证明，LBD 的检索性能优于其他同类 KNN 搜索算法。

参 考 文 献

1 Chavez E, Navarro G, Baeza-Yates R, et al. Searching in metric spaces. ACM Computing Surveys, 2001, 33(3): 273～321

2 Fonseca M J, Jorge J A. Indexing high-dimensional data for content-based retrieval in large databases. In: Proceedings of the Eighth International Conference on Database Systems for Advanced Applications (DASFAA'03), 2003. 267～274

3 Hjaltason G R, Hanan S. Index-driven similarity search in metric spaces. ACM Transactions on Database Systems, 2003, 28(4): 517～580

4 Weber R, Schek H, Blott S. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In: Proc 24th International Conference on Very Large Data Bases, 1998. 194～205

5 Beyer K, Goldstein J, Ramakhrisnan R, et al. when is "Nearest neighbor" meaningful. In: Proc. the 7th Int'l Conf. Database Theory (ICDT'99), 1999. 1～11

6 Hinneburg A, Aggarwal C C, Keim D A. What is the nearest neighbor in high dimensional spaces. In: Proc. 26th International Conference on Very Large Data Bases, 2000. 506～515

7 BinCui, et al. Exploring Bit -Difference for Approximate KNN Search in High-dimensional Databases. In: Proc. 16th Australasian Database Conference, 2005. 165～174

8 Berchtold S, Bohm C, Kriegel H P. The pyramid-technique: Towards breaking the curse of dimensionality. In: Proc. ACM SIGMOD International Conference on Management of Data, 1998. 142～153

9 Yu Cui, et al. Indexing the Distance: An Efficient Method to KNN Processing. In: Proc. 27th International Conference on Very Large Data Bases, 2001. 421～430

10 Chakrabarti K, Mehrotra S. Local Dimensionality Reduction: A New Approach to Indexing High Dimensional Spaces. In: Proc. Conf Very Large Data Bases, 2000. 89～100

11 Jin H, Ooi B, Shen H, et al. An Adaptive and Efficient Dimensionality Reduction Algorithm for High-Dimensional Indexing. In: Proc. Int'l Conf. Data Eng, 2003. 87～98

12 Corel Image Features. http://kdd.ics.uci.edu