**Similarity Search in High Dimension via Hashing**

**LSH算法：**

1、算法思想：将高维空间中的元素视为点并赋以坐标值，坐标值为正整数。通过一族哈希函数将空间所有点映射到n个哈希表中，n=||，即每个哈希函数f对应一个哈希表，每个哈希表都存放着空间所有的点。对于给定的查询子q，分别计算、、…、，，i=1,2,…,n 。以所有落入的哈希表中的桶中所有点作为候选集，比较其与q之间的距离，选出距离最近的K个点（K-NNS）。

2、算法步骤

（1）预处理：对于任意一点P，P为d维空间。设={,,…,}，将空间P映射到维Hamming空间，映射方法如下：

，

=()()…()

其中()表示个1紧跟C-个0，C表示空间P中任意点 坐标的最大值。

这种映射是保距的，即p,qP，=。其中表示定义在空间P上准则下的欧几里得距离，表示定义在空间下的Hamming距离。因此原问题——找出空间P中与查询子q距离最近的K个点——转化为在空间中的-NNS问题。在算法实现中，并没有显式将p转化为，而是用别的计算方法利用了这种转化，使算法易于描述并且实现的时空效率高。下面定义哈希函数的过程中，p代表了原始点的向量形式（即），即不会区分p与。

（2）定义哈希函数族：定义I={1,2,…,},定义正整数，取个I的子集，分别记为，，…，。定义p|I为向量p在坐标集I上的投影，即以坐标集I中每个坐标为位置索引，取向量p对应位置的比特值并将结果串联起来。比如I={1,5,7,8}，p =110010001110（对应原始点p={2,1,3}，d=3,C=4），则p|I=1100。记=p|I，我们将空间P中的所有点利用哈希函数族都存入哈希表的哈希桶中。哈希族的形式如下图所示：

|  |  |  |
| --- | --- | --- |
|  |  | …… |
|  |  | …… |
|  |  | …… |
|  |  |  |
|  |  | …… |

表 1

表中有个哈希表，第i个哈希表有个哈希桶，即第i个哈希表将空间P中的点映射到了个不同位置。因此表中共有个哈希桶。

因为哈希桶的数量可能会很大，因此采用第二层哈希，利用标准哈希函数将所有的哈希桶映射到一个大小为M的哈希表中。记该哈希表中的哈希桶最大容量为B，在算法中采取的冲突解决方法是：当一个哈希桶内点的个数超过B时，则新分配一个大小为B的桶并将该新桶连接到原来的桶中，而在实现的过程中，采取了更简单的方法：当一个哈希桶中点的个数超过B时，则不能再有点插入，当有新点分配到该桶中时，该点会被分配到其他未满的哈希桶中。新的哈希表如下表所示：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |

中逻辑上存储的是表1中的哈希桶，物理上存储的是表1的哈希桶中存储的空间P中的点。

（3）查询操作：对于给定的查询子q，计算，，…，，取出对应哈希桶中所有的点（即满足=的点p的集合）作为候选集。最后在候选集中选出K个距离查询子q最近的K个点。

（4）小问题

1）集合I的子集，，…，，对于j=1,2，…，，由从集合{1,2,…, }随机的均匀的可放回的选取k个元素组成。最好的k要使距离近的元素映射到同一个桶中，距离远的元素映射到不同的桶中。

2）前面提到，实际实现中不必显式的将d维空间P中的点p映射到维Hamming空间的向量，而是使用其他的方法隐式的进行。方法是这样的：记d维空间P中的点p映射到维Hamming空间的向量为,坐标集I{1,2,…, }，的含义如前所述。定义I|i，其中i{1,2,…,d}，I|i的含义为对应到p的第i个坐标的排好序的I中的坐标。举例说明，I={1,2,5,8,13}，p={2,3,1,4}，d=4，C=5，=11000111001000011110，I|1={1,2,5}，I|2={8}，I|3={13}，I|4=，因为d=4，所以I有四个映射：I|1，I|2，I|3，I|4。因为C=5，所以I|1表示I中范围在1-5中的坐标，I|2表示I中范围在6-10中的坐标，以此类推。显然，在I|i上的投影是一串1紧跟一串0的形式，比如上面讨论中在I|1上的投影是110，在I|2上的投影是1，这是显然的。因为I中坐标是排好序的，而如果将中每C个比特为一组划分的话，每一组是一串1加一串0组成的串，I|i是I中坐标与第i组中索引的交集，在I|i上的投影是从的第i组中从左至右的选择索引为交集中元素的比特值，所得投影必定是一串1紧跟一串0组成的串。而在I上的投影即为将在I|i(i=1,2,…,d)上投影的串联起来得到的串。因此，要得到在I上的投影，只需得到在I|i上的投影；要得到在I|i上的投影，只需得知在I|i上的投影中1的个数；要想得到1的个数，只需比较I|i和点p的第i个坐标值，=|{I|i}-C\*(i-1) |，||表示集合的元素个数。即比较I|i中小于等于的元素个数，因为I{1,2,…,}，=Cd，因此对于i>1的情况，要减去C的i-1倍，才能与的值在同一度量上。举例说明：d=4，C=5，=20，p={2,4,3,5}，=11000111101110011111,I={1,2,3,5,7,8}，因此I|1={1,2,3,5}，I|2={7,8}，I|3=I|4=，在I|1，I|2上的投影分别为1100,11。观察I|1与，I|2与，I|1中小于等于=2的个数为2，因此在I|1上的投影有两个1，I|2元素减去C=5后小于等于=4的个数为2，因此在I|2上的投影有两个1。0的个数为I|i的元素个数减去1的个数。最后得到的结果是，在I上的投影|I=110011。整个过程中，用到的变量有d，C，I，这里的I对应算法步骤中的。然后就可以计算I|i，通过比较I|i中元素与p的坐标就可以得到在I|i上的投影，最后串联得到在I上的投影。因此没有显式的用到。主要运算在比较p的坐标与I|i中的元素，即找到I|i中小于等于p的第i个坐标的所有元素，因为I|i是排好序的，找到I|i中小于等于p的第i个坐标的临界元素即可，使用二分法查找，每次查找的时间复杂度为，因为i=1,2,…,d，共需查找d次，故计算每个哈希函数所用的时间复杂度为。

**LSH算法分析**

**一、符号定义：**1、集合S

2、S中元素p,q距离记为D(p,q)

3、，p的r邻域记为。

**二、局部敏感的一般化定义：**

从S映射到U的函数族称为对距离D是(,,,)-敏感的，如果满足以下两个条件：

1. if , then 
2. if , then 

**三、维Hamming空间中的点p,q，对于任意的r,,选取的函数族为={:((,,…, ))=,for i=1,2,…, }，则其在Hamming距离下是(r,r(1+),1-,1-)-敏感的。**

**四、将前面算法的应用推广到一般的函数族**，即将函数定义为如下形式：

=(,,…, )

即从函数族中可重复的选取k个函数组成一个g函数，并按这种方法生成个g函数。当函数族为函数族时，即得到上面算法中用到的g函数：=。

**五、将LSH算法用来解决(r,)-Neighbor问题：**

问题描述：给定一个点q，确定在q的r邻域内是否存在一点p，若存在，返回q的r(1+****)邻域内一点；若不存在，判断是否在q的r(1+****)邻域内不存在其它点。记=r, = r(1+****)。记为P中与q的距离大于的点的集合。

LSH算法能解决(r,****)-Neighbor问题，当其能满足一下两个条件：

P1：如果在q的邻域内有点存在，则对某些j=1,2,…，使得  =（把q与距q较近的点映射到同一块中，是应该追求的，这种情况越多越好。）

P2：q经过g函数映射落入的块中仅包含中点的块数小于c。（把q 与距q较远的点映射到同一个块中，是应该避免的，这种情况越少越好。）

P1和P2中的块对应二级哈希表中的每个哈希桶。每个点经过g函数映射得到一个k维的哈希值，因为有个g函数，因此每个点会有个不同的k维哈希值分别存放在每个g函数中。在二层哈希的时候，利用下面哈希函数将每个点的每个k维向量（即所有g函数所有哈希桶的所有内容）映射到大小为M的二级哈希表中：



为点的k维向量值，，i=1,2,…,k为[0,M-1]中随机k个数。

一级哈希表中同一个哈希桶中的点一定在二级哈希表中的同一个哈希桶中

一级哈希表不同哈希桶中但相近的点可能会在二级哈希表的同一个哈希桶中

假设****为(,,,)-敏感的，定义=，则有如下定理：

定理：设定，，可以保证条件P1和P2同时满足的概率至少为。

备注：对于任意的>0，重复将LSH算法进行次，则至少会有一次将成功的概率提高到1-。

证明：设条件P1发生的概率为，条件P2发生的概率为，以下将证明和都比较大。不妨设查询子q的邻域内存在一点，反之证明类似。设，则对于中的任一点，由局部敏感的定义，式子成立的概率最大是=。对于每个g函数，将二级哈希中分配给的只包含中点的块（每个块对应一个哈希桶）的数量的期望值设为2，则分配给所有的符合条件的块的数量不超过2，因此，由马尔可夫不等式：得知，符合条件的块数超过4的概率小于1/2（），若选择c=4，则条件2满足的概率>。

由于事先假设的条件是查询子q的邻域内存在一点，所以下面讨论式子成立的概率。是以下面式子为下限的：



前面已设，故式子的概率小于等于1-，故对于所有j=1,2,…, ，都有式子成立的概率为，故至少存在一个j使得成立的概率（即）大于1-。

因此条件1和条件2同时成立的概率是1-[(1-)+(1-)]=+-1=。

**整体思路：**提出了LSH算法，证明了LSH算法可以利用在解决(r,****)-Neighbor问题上，而文章的本意是要解决****-NNS问题，而后一个问题是前一个问题在r等于查询子q距离数据集最近点距离时的特殊情况，通过在MIN D(q,P)到MAX D(q,P)之间取若干个不同的r，来实现解决****-NNS问题。