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$\mathcal{F}$将空间所有点映射到 n 个哈希表$\mathcal{T}_i$中，n=|$\mathcal{F}$|，即每个哈希函数 f∈$\mathcal{F}$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 $f_1(q)$、$f_2(q)$、…、$f_n(q)$，$f_i \in \mathcal{F}$，i=1,2,…,n 。以所有 $f_i(q)$ 落入的哈希表$\mathcal{T}_i$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$\mathfrak{p} \in$P，P 为 d 维空间。设$\mathfrak{p}$={$x_1, x_2, \cdots, x_d$}，将空间 P 映射到d'维 Hamming 空间$\mathcal{H}^{d'}$，映射方法如下：

$$\mathfrak{p} \to \mathfrak{p}' \in P,$$

$$\mathfrak{p}' = Unary_c(x_1)\, Unary_c(x_2) \cdots Unary_c(x_d)$$

其中$Unary_c(x_i)$表示x_i个 1 紧跟 C-x_i个 0，C 表示空间 P 中任意点 $\mathfrak{p}$坐标的最大值。

这种映射是保距的，即∀p,q∈P，$d_1(p,q)=d_H(p,q)$。其中d_1表示定义在空间 P 上l_1准则下的欧几里得距离，d_H表示定义在$\mathcal{H}^{d'}$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$\mathcal{H}^{d'}$空间中的ϵ-NNS 问题。在算法实现中，并没有显式将 $\mathfrak{p}$ 转化为$\mathfrak{p}'$，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$\mathfrak{p}'$），即不会区分 p 与$\mathfrak{p}'$。

（2）定义哈希函数族：定义 I={1,2,…,d'},定义正整数l，取l个 I 的子集，

分别记为I_1，I_2，…，I_l。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$g_j(p)$=p|I，我们将空间 P 中的所有点利用哈希函数族$g_j(p)$都存入哈希表的哈希桶中。哈希族的形式如下图所示：

T_1	$g_1(\bullet)=\bullet \mid I_1$	$g_1(p_{1i})$	$g_1(p_{2i})$	……	$g_1(p_{k_1 i})$
T_2	$g_2(\bullet)=\bullet \mid I_2$	$g_2(p_{1i})$	$g_2(p_{2i})$	……	$g_2(p_{k_2 i})$
T_3	$g_3(\bullet)=\bullet \mid I_3$	$g_3(p_{1i})$	$g_3(p_{2i})$	……	$g_3(p_{k_3 i})$
⋮	⋮		⋮		
T_l	$g_l(\bullet)=\bullet \mid I_l$	$g_l(p_{1i})$	$g_l(p_{2i})$	……	$g_l(p_{k_l i})$

表 1

表中有l个哈希表，第 i 个哈希表有k_i个哈希桶，即第 i 个哈希表将空间 P 中的点映射到了k_i个不同位置。因此表中共有$\sum_{i=1}^{l} k_i$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：

B_1	B_2	B_3	$\bullet\bullet\bullet$	B_M

B_i 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 $g_1(q)$，$g_2(q)$，…，$g_l(q)$，取出 $g_i(q)$ 对应哈希桶中所有的点（即满足 $g_i(q)=g_i(p)$ 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 I_1，I_2，…，I_l，对于 j=1,2,…，l，I_j 由从集合{1,2,…，d'}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 d' 维 Hamming 空间的向量 $\mathfrak{p}'$，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 d' 维 Hamming 空间的向量为 $\mathfrak{p}'$，坐标集 I⊆{1,2,…，d'}，$\mathfrak{p}'|\mathrm{I}$ 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，$\mathfrak{p}'$=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，$\mathfrak{p}'$ 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 $\mathfrak{p}'$ 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 $\mathfrak{p}'$ 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 $\mathfrak{p}'$ 第 i 组中索引的交集，$\mathfrak{p}'$ 在 I|i 上的投影

是从$\mathfrak{p}'$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$\mathfrak{p}'$在 I 上的投影即为将$\mathfrak{p}'$在 I|i(i=1,2,…,d)上投影的串联起来得到的串。因此，要得到$\mathfrak{p}'$在 I 上的投影，只需得到$\mathfrak{p}'$在 I|i 上的投影；要得到$\mathfrak{p}'$在 I|i 上的投影，只需得知$\mathfrak{p}'$在 I|i 上的投影中 1 的个数o_i；要想得到 1 的个数，只需比较 I|i 和点 p 的第 i 个坐标值x_i，o_i=|{I|i}-C*(i-1) $\leq x_i$|，|•|表示集合•的元素个数。即比较 I|i 中小于等于x_i的元素个数，因为 I$\subseteq${1,2,…, d'}， d'=Cd，因此对于 i>1 的情况，要减去 C 的 i-1 倍，才能与x_i的值在同一度量上。举例说明：d=4，C=5 ， d' =20 ， p={2,4,3,5} ， $\mathfrak{p}'$ =11000111101110011111,I={1,2,3,5,7,8}，因此 I|1={1,2,3,5}，I|2={7,8}，I|3=I|4=$\varnothing$，$\mathfrak{p}'$在 I|1，I|2 上的投影分别为 1100,11。观察 I|1 与x_1，I|2 与x_2，I|1 中小于等于x_1=2 的个数为 2，因此$\mathfrak{p}'$在 I|1 上的投影有两个 1，I|2 元素减去 C=5 后小于等于x_2=4 的个数为 2，因此$\mathfrak{p}'$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，$\mathfrak{p}'$在 I 上的投影$\mathfrak{p}'$|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I_j。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$\mathfrak{p}'$在 I|i 上的投影，最后串联得到$\mathfrak{p}'$在 I 上的投影。因此没有显式的用到$\mathfrak{p}'$。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$O(\log_2 C)$，因为 i=1,2,…,d，共

需查找 d 次，故计算每个哈希函数$g_i(p)$所用的时间复杂度为$O(d\log_2 C)$。

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、$\forall p \in S$，p 的 r 邻域记为$O(p,r)$。

二、局部敏感的一般化定义：

从 S 映射到 U 的函数族$\mathcal{H}$称为对距离 D 是(r_1, r_2, p_1, p_2)-敏感的，如果满足以下两个条件：

（1） if $p \in O(q,r_1)$, then $\Pr_{\mathcal{H}}(h(q)=h(p)) \geq p_1$

（2） if $p \notin O(q,r_2)$, then $\Pr_{\mathcal{H}}(h(q)=h(p)) \leq p_2$

三、d'维 Hamming 空间中的点 p,q，对于任意的 r,ϵ，选取的函数族为$\mathcal{H}_{d'}=\{h_i: h_i((b_1, b_2, \cdots, b_{d'}))=b_i, \text{for } i=1,2,\cdots, d'\}$，则其在 Hamming 距离下是$(r, r(1+\epsilon), 1-\frac{r}{d'}, 1-\frac{r(1+\epsilon)}{d'})$-敏感的。

四、将前面算法的应用推广到一般的函数族$\mathcal{H}$，即将函数$g_j(p)$定义为如下形式：

$$g_j(p)=(h_{i_1}(p), h_{i_2}(p), \cdots, h_{i_k}(p))$$

即从函数族$\mathcal{H}$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l个 g 函数。当函数族$\mathcal{H}$为函数族$\mathcal{H}_{d'}$时，即得到上面算法中用到的 g 函数：$g_j(p)=p|I_j$。

五、将 LSH 算法用来解决(r,ϵ)-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+ϵ)邻域内一点 p'；若不存在，判断是否在 q 的 r(1+ϵ)邻域内不存在其它点。记 r_1=r， r_2= r(1+ϵ)。记 P' 为 P 中与 q 的距离大于 r_2 的点的集合。

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

P1：如果在 q 的 r_1 邻域内有点 p^* 存在，则对某些 j=1,2,$\cdots l$，使得 $g_j(p^*)$ =$g_j(q)$（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 P' 中点的块数小于 cl。（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 l 个 g 函数，因此每个点会有 l 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：

$$h(v_1,v_2,\cdots,v_k)=a_1v_1+a_2v_2+\cdots+a_kv_k \bmod M$$

$(v_1,v_2,\cdots,v_k)$ 为点的 k 维向量值，a_i，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 $\mathcal{H}$ 为(r_1，r_2，p_1，p_2)-敏感的，定义 $\rho=\frac{\ln 1/p_1}{\ln 1/p_2}$，则有如下定理：

定理：设定 $k=\log_{1/p_2}(n/B)$，$l=(\frac{n}{B})^{\rho}$，可以保证条件 P1 和 P2 同时满足的

概率至少为$\frac{1}{2}-\frac{1}{e}\geq 0.132$。

备注：对于任意的δ>0，重复将 LSH 算法进行$O(1/\delta)$次，则至少会有一次将成功的概率提高到 1-δ。

证明：设条件 P1 发生的概率为P_1，条件 P2 发生的概率为P_2，以下将证明P_1和P_2都比较大。不妨设查询子 q 的r_1邻域内存在一点p^*，反之证明类似。设$k=\log_{1/p_2}(n/B)$，则对于P'中的任一点p'，由局部敏感的定义，式子$g(p')=g(q)$成立的概率最大是$p_2^k=\frac{B}{n}$。对于每个 g 函数g_j，将二级哈希中分配给g_j的只包含P'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g_j的符合条件的块的数量不超过 2l，因此，由马尔可夫不等式：$\Pr(X\geq a)\leq\frac{E(X)}{a}$得知，符合条件的块数超过 4$l$的概率小于 1/2（$\Pr(X\geq 4l)\leq\frac{E(X)}{4l}<\frac{2l}{4l}=\frac{1}{2}$），若选择 c=4，则条件 2 满足的概率$P_2>\frac{1}{2}$。

由于事先假设的条件是查询子 q 的r_1邻域内存在一点p^*，所以下面讨论式子$g_j(p^*)=g_j(q)$成立的概率P_1。P_1是以下面式子为下限的：

$$p_1^k=p_1^{\log_{1/p_2}n/B}=(n/B)^{-\frac{\log 1/p_1}{\log 1/p_2}}=(n/B)^{-\rho}$$

前面已设$l=(\frac{n}{B})^{\rho}$，故式子$g_j(p^*)\neq g_j(q)$的概率小于等于 1-p_1^k，故对于所有 j=1, 2, …, l，都有式子$g_j(p^*)\neq g_j(q)$成立的概率为$(1-p_1^k)^l=(1-(n/B)^{-\rho})^{(n/B)^{\rho}}<1/e$，故至少存在一个 j 使得$g_j(p^*)=g_j(q)$成立的概率（即$P_1$）大于 1-$1/e$。

因此条件 1 和条件 2 同时成立的概率是 1-[(1-P_1)+(1-P_2)]=P_1+P_2-1=

$\frac{1}{2}-\frac{1}{e}$。

整体思路：提出了 LSH 算法，证明了 LSH 算法可以利用在解决(r,ϵ)-Neighbor 问题上，而文章的本意是要解决ϵ-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ϵ-NNS 问题。