華中科技大學

操作系统原理课程实验报告

姓　　名：　彭嘉炜

学　　院：　计算机科学与技术

专　　业：　计算机科学与技术

班　　级：　本硕博 2101 班

学　　号：　U202115662

指导教师：　华宇

分数	
教师签名	

2024 年 4 月 19 日

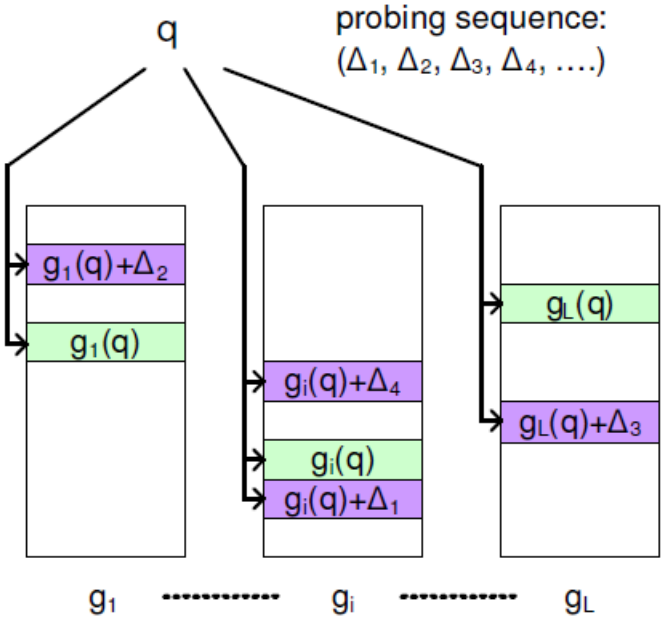
目　录

如何减少 LSH 的空间开销

对于高维相似性搜索，LSH 是一种常用的解决方法，位置敏感哈希(locality sensitive hash, LSH)的基本思想是使用哈希函数，将相似的对象以高概率映射到相同的哈希桶中。在 LSH 索引上执行相似度搜索查询包括两个步骤:(1)使用 LSH 函数为给定的查询 q 选择“候选”对象，(2)根据候选对象到 q 的距离对候选对象进行排序。

但是 LSH 存在一个问题：离散化的分桶方式仍然有概率将邻居分到不同桶中，这导致了基本 LSH 索引方法存在一个主要缺点：它可能需要大量的哈希表来覆盖最近的邻居。同时每个哈希表的大小与数据集大小成正比，因为每个表的条目数量与数据集中数据对象的数量一样多。当哈希表的空间需求超过主存大小时，查找哈希桶可能需要磁盘 I/O，这会对查询过程造成很大的延迟。

查阅文章有人提出了一种 Multi-Probe LSH，这个原理是在 LSH 中，对于查询点 q 邻近的数据点将以很高的概率落在相同或者邻近的值上(相差不超过 1)，因此对于查询点 q，我们定义一个扰动序列向量 $\Delta = \delta 1, \cdots, \delta m$，当我们应用扰动序列 Δ，我们将探询桶 $g(q)+\Delta$。如下图：$g\,i(q)$是在第 i 个表中的查询点 q 的哈希值，$(\Delta 1, \Delta 2, \cdots)$是一组探寻序列，$gi(q)+\Delta 1$ 是应用扰动向量 $\Delta 1$ 后产生的新的哈希值，它指向表中的一个新的哈希桶，通过使用多个扰动向量，我们可以获得多个与 q 指向的哈希桶“邻近”的桶，这些桶中很有可能含有与 q 邻近的元素。



1.1 主要算法的设计

Algorithm Generate T perturbation sets
1 A0 = {1}

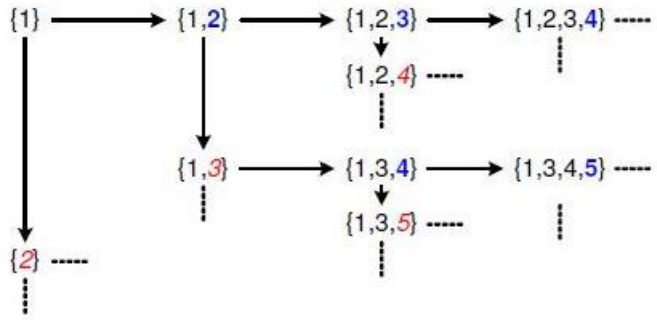
```
2 minHeap_insert(A0,score(A0))
3 for i = 1 to T do
4     repeat
5         Ai = minHeap_extractMin()
6         As = shift(Ai)
7         minHeap_insert(As,score(As))
8         Ae = expand(Ai)
9         minHeap_insert(Ae,score(Ae))
10    until valid(Ai)
11    output Ai
12 end for
```

其中：

shift(A)表示将 max(A)替换为 1+max(A)。例如，shift({1,3,4}) ={1,3,5}。

expand(A)表示将元素 1 + max(A)添加到集合 A 中，例如 expand({1,3,4}) ={1,3,4,5}。

上述算法描述了产生 T 个扰动集的过程。核心思想是使用一个最小堆来管理扰动集合，每次从最小堆中取出具有最小分数的集合进行操作，直到得到有效的扰动集合。产生过程如下：



1.2 操作流程分析

1. 初始化阶段 (init)

定义哈希函数族：选择一组局部敏感哈希（LSH）函数，这些函数能够将高维空间中的对象映射到较低维的哈希空间。

创建哈希表：为每个选定的 LSH 函数创建一个哈希表，这些表将用于存储数据对象的哈希值。

2. 数据插入阶段 (insert)

映射数据对象：对于每个数据对象，使用 LSH 函数计算其哈希值。

存储哈希值：将每个数据对象的哈希值存储在相应的哈希表的对应桶中。

3. 查询阶段 (query)

计算查询对象的哈希值：使用与数据插入相同的 LSH 函数为查询对象计算哈希值。

生成候选集：根据查询对象的哈希值，在每个哈希表中找到对应的桶，并从这些桶中收集数据对象形成候选集。

4. 多探测策略 (Multi-Probe)

生成探测序列：根据查询对象的哈希值，生成一个探测序列，该序列指导算法在哈希表中探测多个可能包含最近邻的哈希桶。

步进探测（Step-Wise Probing）：从与查询对象哈希值相差最小的桶开始，逐步探测距离越来越远的哈希桶。

查询指导探测（Query-Directed Probing）：根据查询对象在哈希表中的位置，计算出一个更优的探测序列，优先探测最有可能包含最近邻的哈希桶。

5. 成功概率估计

估计每个桶的成功概率：基于高斯分布的性质，估计每个哈希桶包含查询对象最近邻的概率。

6. 优化探测序列的构建

预计算排序：预计算哈希桶的排序，以便在查询时快速生成探测序列，减少查询时的计算开销。

7. 评估与排名

距离计算：对候选集中的对象计算与查询对象的距离。

排名：根据计算出的距离对候选集中的对象进行排名，选择距离最近的对象作为近似最近邻。

1.3 理论分析

Multi-Probe LSH 相比于传统的 LSH 方法在时间和空间效率上更优秀，主要原因在于它的设计和实现采用了以下几个关键策略：

1. 多探测策略（Multi-Probing）

传统 LSH：通常只考虑查询点映射到的单一哈希桶。

Multi-Probe LSH：不仅考虑查询点映射到的哈希桶，还考虑了一系列可能包含最近邻的其他哈希桶。这是通过预先计算的探测序列实现的，该序列基于点在哈希空间中的位置，智能地选择要检查的邻近哈希桶。

2. 探测序列的优化

Multi-Probe LSH：使用两种主要的探测序列生成方法：步进探测（Step-Wise Probing）和查询指导探测（Query-Directed Probing）。这些方法基于数据点在哈希空间的分布和查询点的位置，优化了探测的顺序和选择，从而提高了搜索效率。

1.4 运行与测试

引用论文中的比较如下图：

recall	method	error ratio	query time (s)	#hash tables	space ratio
0.96	basic	1.027	0.049	44	14.7
	entropy	1.023	0.094	21	7.0
	multi-probe	1.015	0.050	3	1.0
0.93	basic	1.036	0.044	30	15.0
	entropy	1.044	0.092	11	5.5
	multi-probe	1.053	0.039	2	1.0
0.90	basic	1.049	0.029	18	18.0
	entropy	1.036	0.078	6	6.0
	multi-probe	1.029	0.031	1	1.0

(a) image dataset

recall	method	error ratio	query time (s)	#hash tables	space ratio
0.94	basic	1.002	0.191	69	13.8
	entropy	1.002	0.242	44	8.8
	multi-probe	1.002	0.199	5	1.0
0.92	basic	1.003	0.174	61	15.3
	entropy	1.003	0.203	25	6.3
	multi-probe	1.002	0.163	4	1.0
0.90	basic	1.004	0.133	49	16.3
	entropy	1.003	0.181	19	6.3
	multi-probe	1.003	0.143	3	1.0

(b) audio dataset

Table 2: Search performance comparison of different LSH methods: multi-probe LSH is most efficient in terms of space usage and time while achieving the same recall score as other LSH methods.

上图说明了:在获得与其他 LSH 方法相同的召回分数的同时，多探针 LSH 在空间使用和时间方面效率最高。

```
查询点: [0.18306216 0.23057345 0.88494328 0.11343312 0.79367727 0.76173292
 0.75152515 0.59344293 0.82941624 0.91643572]
前十个近邻的数据点及其距离:
近邻索引: 440
距离: 0.5290341926750407
近邻数据点: [0.49526652 0.35601988 0.80929297 0.17904538 0.76260468 0.71554995
 0.97015268 0.37357985 0.60055663 0.98734419]
近邻索引: 667
距离: 0.6253146806313209
近邻数据点: [0.21271061 0.57685435 0.85808897 0.38254695 0.79067485 0.58564748
 0.46835298 0.69216454 0.6967764  0.6744756 ]
近邻索引: 988
距离: 0.6303490725679534
近邻数据点: [0.2022066  0.11927653 0.893045   0.07034755 0.78874732 0.38996917
 0.75343708 0.35416474 0.41476716 0.79298418]
近邻索引: 47
距离: 0.6719660413739457
近邻数据点: [0.35322808 0.08359939 0.71946842 0.24868172 0.59323702 0.81103358
 0.94238968 0.33715346 0.37489518 0.97965463]
```

我在这里最后实现了一个最简单的 Multi-Probe LSH，能成功返回最近的十个数据点的索引和距离。

参考文献

• "Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions" (by Alexandr Andoni and Piotr Indyk). Communications of the ACM, vol. 51, no. 1, 2008, pp. 117-122.

• Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li, “Multi- Probe LSH: Efficient Indexing for High-Dimensional Similarity Search,” Proc. 33rd Int’l Conf. Very Large Data Bases (VLDB ’07), pp. 950-961, 2007.