软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2023,34(10):4821−4829 [doi: 10.13328/j.cnki.jos.006649]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

近似最近邻归约问题在泊松点过程上的再研究*

马恒钊[1,3], 闫 跃[2], 李建中[1]

[1](哈尔滨工业大学 计算机科学与技术学院, 黑龙江 哈尔滨 150001)

[2](哈尔滨金融学院 计算机系, 黑龙江 哈尔滨 150001)

[3](中国科学院深圳先进技术研究院 先进计算与数字工程研究所, 广东 深圳 518055)

通信作者: 李建中, E-mail: lijzh@hit.edu.cn

摘 要: 在已发表文献中, 研究了基于图灵归约求解ε-NN 的问题, 即给定查询点 q、点集 P 及近似参数ε, 找到 q 在 P 中近似比不超过$1+\varepsilon$的近似最近邻, 并提出了一个具有 O(log$n$)查询时间复杂度的图灵归约算法, 这里的查询时间是调用神谕的次数. 经过对比, 此时间优于所有现存的归约算法. 但是已发表文献中提出的归约算法的缺点在于, 其预处理时间和空间复杂度中有 O($(d/\varepsilon)^d$) 的因子, 当维度数 d 较大或者近似参数ε较小时, 此因子将变得不可接受. 因此, 重新研究了该归约算法, 在输入点集服从泊松点过程的情况下, 分析算法的期望时间和空间复杂度, 将算法的期望预处理时间复杂度降到 O($n\log n$), 期望空间复杂度降到O($n\log n$), 而期望查询时间复杂度保持O(log$n$)不变, 从而完成了在已发表文献中所提出的未来工作.

关键词: 近似最近邻; 归约; 泊松点过程; 复杂度

中图法分类号: TP311

中文引用格式: 马恒钊, 闫跃, 李建中. 近似最近邻归约问题在泊松点过程上的再研究. 软件学报, 2023, 34(10): 4821–4829. http://www.jos.org.cn/1000-9825/6649.htm

英文引用格式: Ma HZ, Yan Y, Li JZ. Revised Algorithm Based on Turing Reduction for Solving ε-NN in Possion Point Process. Ruan Jian Xue Bao/Journal of Software, 2023, 34(10): 4821–4829 (in Chinese). http://www.jos.org.cn/1000-9825/6649.htm

Revised Algorithm Based on Turing Reduction for Solving ε-NN in Possion Point Process

MA Heng-Zhao[1,3], YAN Yue[2], LI Jian-Zhong[1]

[1](School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China)

[2](Department of Computer Science, Harbin Finance University, Harbin 150001, China)

[3](Institute of Advanced Computing and Digital Engineering, Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China)

Abstract: In a published study, the problem of using Turing reduction to solve ε-NN is studied. In other words, given a query point q, a point set P, and an approximate factor ε, the purpose is to return the approximate nearest neighbor of q in P with an approximation ratio of not more than $1+\varepsilon$. Moreover, a Turing reduction algorithm with O(log$n$) query time complexity is proposed, where the query time is the number of times that the oracle is invoked. The comparison indicates that the O(log$n$) query time is the lowest compared to that of all the existing algorithms. However, the disadvantage of the proposed algorithm is that there is a factor of O($(d/\varepsilon)^d$) in the preprocessing time complexity and space complexity. When the number of dimensions d is high, or the approximation factor ε is small, the factor would become unacceptable. Therefore, this study revises the reduction algorithm and analyzes the expected time complexity and space complexity of the algorithm when the input point set follows the Poisson point process. As a result, the expected preprocessing time complexity is reduced to O($n$log$n$), and the expected space complexity is reduced to O($n$log$n$), while the expected query time complexity remains O(log$n$). In this sense, the future work raised in the published study is completed.

Key words: approximate nearest neighbor; reduction; Poisson point process; complexity

---

* 基金项目: 国家自然科学基金 (61732003, 61832003, U1811461)
收稿时间: 2020-09-06; 修改时间: 2020-10-30; 采用时间: 2022-01-29; jos 在线出版时间: 2023-01-18
CNKI 网络首发时间: 2023-01-19

1 前 言

在文献 [1] 中, 我们提出了从 ε-NN 到 (c, r)-NN 的一种 $O(\log n)$ 时间的图灵归约, 给出了一种高效的求解 ε-NN 问题的算法. 这里, ε-NN 是近似最近邻问题, 定义如下: 给定点集 P、查询点 q 以及近似因子 ε, 输出一个距离 q 不超过 $(1+\varepsilon)r^*$ 的点, 其中, r^* 是 q 到 P 中最近的点的距离. (c, r)-NN 是近似近邻问题, 定义如下: 给定点集 P、查询点 q、查询范围 r 和近似因子 $c>1$, 输出满足以下条件: 若存在 p 使得 p 到 q 的距离不超过 r, 则返回 P 中某一个距离 q 不超过 cr 的点; 若 P 中所有点距离 q 都超过 cr, 则返回否定答案 No. 在以上定义中, 所有的点都来自于 R^d, 即每个输入点都是 d 维的向量, 而距离用 l_k 距离来表达, 即两个点 $p=(p^{(1)},\ldots,p^{(d)})$, $q=(q^{(1)},\ldots,q^{(d)})$ 之间的距离为 $D(p,q)=\left(\sum_{i=1}^{d}|p^{(i)}-q^{(i)}|^k\right)^{1/k}$.

文献 [1] 中的算法的查询时间复杂度为 $O(\log n)$, 即调用神谕的次数或调用 (c, r)-NN 算法的次数为 $O(\log n)$. 在文献 [1] 中我们已经证明, 该算法的查询时间复杂度优于所有现存算法[2–4]. 但是, 文献 [1] 中算法的预处理时间复杂度为 $O((d/\varepsilon)^d\cdot n\log n)$, 空间复杂度为 $O((d/\varepsilon)^d\cdot n)$. 若将 d 视为常数, 则此算法的复杂度接近文献 [5] 中提出的最优复杂度. 然而在维数 d 比较高或 ε 比较小的情况下, $O((d/\varepsilon)^d)$ 的因子将变得不可接受. 因此, 在文献 [1] 中, 我们提出未来的工作是降低此因子.

本文延续文献 [1] 的工作, 解决降低算法预处理时间复杂度及空间复杂度中 $O((d/\varepsilon)^d)$ 因子的问题. 实际上, 此因子的出现是因为文献 [1] 中分析的是最坏复杂度. 本文的研究结果表明, 当输入点集服从泊松点过程的情况下, 算法的期望预处理时间复杂度从 $O((d/\varepsilon)^d\cdot n\log n)$ 降低到 $O(n\log n)$, 而且算法的期望空间复杂度为 $O(n\log n)$, 与 d 和 ε 独立.

现介绍本文的贡献如下.

(1) 本文针对输入点集服从泊松点过程的情况, 提出了不同于文献 [1] 中的算法, 并分析了算法的正确性和时空复杂度.

(2) 本文提出的算法的期望预处理时间复杂度为 $O(n\log n)$, 期望空间复杂度为 $O(n\log n)$, 在移除了 $O((d/\varepsilon)^d)$ 因子的同时, 仍保持比较低的复杂度.

(3) 本文提出的算法的期望查询时间复杂度仍为 $O(\log n)$, 与文献 [1] 中的算法相同.

本文第 1 节的其余部分介绍相关工作. 第 2 节介绍预备知识及问题定义. 第 3 节介绍本文提出的算法. 第 4 节证明本文算法的正确性, 并分析其时间和空间复杂度. 最后第 5 节给出结论.

1.1 相关工作

最近邻问题 (或 NN 问题) 及其变种是计算几何的重要基础问题[6], 并在许多领域具有重要的应用[7–10]. 精确最近邻问题可以追溯到 1960 年代[11]. 研究者们发现, 解决精确最近邻问题面临着“维度诅咒”, 即当维度数增高时, 算法的理论性能和实验表现都会极大地降低, 甚至无法优于简单的直接搜索[12,13]. 为了解决“维度诅咒”, 许多研究者开始研究近似最近邻问题, 即本文中研究的 ε-NN 问题. 研究者们提出了一系列算法[5,14,15], 但是维度诅咒问题只是减轻而没有被解决. 更多的相关工作可以参阅文献 [1,4].

目前解决“维度诅咒”问题的最好方法当属 Indyk 等人提出的基于图灵归约的方法[4]. 他们定义了 (c, r)-NN 问题, 并提出了第一个从 ε-NN 问题到 (c, r)-NN 问题的图灵归约. 于是 ε-NN 问题的求解被分为两部分, 一部分是求解 (c, r)-NN 问题, 另一部分是解决从 ε-NN 问题到 (c, r)-NN 问题的图灵归约. 如果两部分都能够避免“维度诅咒”, 则 ε-NN 问题的“维度诅咒”随之解决. 下面我们分别介绍这两部分工作.

Indyk 等人[4]提出的基于局部敏感哈希 (LSH) 的方法是目前解决 (c, r)-NN 问题最有效的方法. 简言之, 局部敏感哈希函数能够使得距离较近的点对以较大概率被散列到同一个桶中, 距离较远的点对以较大概率散列到不同的桶中. 在文献 [4] 提出了局部敏感哈希的定义之后, Datar 等人[16]提出了第一个实用的局部敏感哈希函数, 适用于欧氏空间. 此后, 局部敏感哈希的研究获得了长足的发展, 既有关于不同数据、不同距离函数的局部敏感哈希函数设计的研究[17–19], 也有关于局部敏感哈希函数的性能下界的研究[20,21], 也有关于局部敏感哈希在不同领域的应用的研究[22]. 文献 [23,24] 给出了详细的综述. 最后要说明的是, 基于局部敏感哈希方法解决 (c, r)-NN 的算法可以

达到伪亚线性时间, 即经过多项式时间的预处理之后, 可以在亚线性时间内得到结果.

Indyk 等人[4]提出了第一个从 ε-NN 问题到 (c, r)-NN 问题的归约. 请注意, 这里的归约是图灵归约, 即以一个解决 (c, r)-NN 问题的算法作为神谕, 通过反复调用 (c, r)-NN 问题的算法来解决 ε-NN 问题. 设计图灵归约一般关注其查询复杂度, 即调用神谕算法的次数. 文献 [4] 中给出的归约的查询时间复杂度为 $O(\log^2 n)$. 文献 [2,3] 中提出新的图灵归约, 其查询时间复杂度分别为 $O(\log(n/\varepsilon))$ 和 $O(\log^{O(1)} n)$, 见文献 [25]. 我们在文献 [1] 中提出另一种图灵归约, 得到了目前最低的查询时间复杂度 $O(\log n)$. 但是, 我们在文献 [1] 中提出的归约的预处理时间和空间复杂度中都有 $O((d/\varepsilon)^d)$ 的因子. 在本文中, 我们就将致力于降低以至消除此因子.

泊松点过程是一种十分重要的随机过程, 它有着很多良好的几何性质, 因此在计算几何领域有很多相关的研究. 比如文献 [26] 中研究了服从泊松点过程的点集上构建的 Delaunay 三角剖分的性质, 文献 [27] 中研究了服从泊松点过程的点集上的寻路问题, 文献 [28] 研究了服从泊松点过程的网络节点的覆盖问题. 由于泊松点过程的良好性质, 通常能够使分析得到简化, 并能够得到比较好的理论结果. 据我们所知, 目前还没有在服从泊松点过程的数据上的近似最近邻问题的研究.

2 预备知识

2.1 d 维矩形和 d 维超球

一个 d 维矩形 R 是 R^d 空间上 d 个闭区间的笛卡尔积, 即对于 $1 \leqslant i \leqslant d$, 设 $I_i = [a_i, b_i]$, 则 $R = I_1 \times I_2 \times \ldots \times I_d$. 称 I_i 为 R 在第 i 维上的定义区间, $l_i(R) = b_i - a_i$ 为 R 在第 i 维上的边长. 若 $l_i(R) = l$ 对所有 $1 \leqslant i \leqslant d$ 都成立, 则称 R 为一个 d 维超立方体.

d 维超球是 2 维的圆和 3 维的球在高维上的拓展. 令 $c \in R^d$, $r > 0$, 记 $B(c, r) = \{x \in R^d | D(x, c) \leqslant r\}$ 为以 c 为球心、r 为半径的 d 维超球. 对于点 p 和 d 维超球 $B(c, r)$, 若 $D(p, c) < r$ 则称 $p \in B(c, r)$ 或 $B(c, r)$ 包含点 p; 若 $D(p, c) = r$ 则称 $B(c, r)$ 通过点 p.

2.2 泊松点过程

对泊松点过程的严格定义需要 σ-代数、Borel 集、勒贝格测度等概念, 这里不做过多赘述, 请参考文献 [29]. 简而言之, 一个定义在 $S \subseteq R^d$ 上的、速度为 λ 的泊松点过程是满足以下条件的过程.

(1) 落到任意区域 $A \subseteq S$ 之内的点数满足泊松形式的分布, 形式化地, 记 $N(A)$ 为落到区域 A 内的点数, $vol(A)$ 为区域 A 的体积, 则 $\Pr[N(A) = k] = e^{-\lambda vol(A)} \dfrac{(\lambda vol(A))^k}{k!}$.

(2) 任意 $A, B \subseteq S$, 若 $A \cap B = \emptyset$, 则 $\Pr[N(A) = k_1]$ 和 $\Pr[N(B) = k_2]$ 相互独立.

(3) 在给定 $N(A) = k$ 的条件下, 这 k 个点在区域 A 内均匀分布.

在下面的讨论中, 我们定义一个点集 P. 设存在一个定义在 R^d 上、速度 $\lambda = 1$、点数无限的泊松点过程, S 是边长为 $n^{1/d}$ 的 d 维超立方体, 服从泊松点过程而落在 S 内的点集即为 P. 由泊松点过程的性质, P 的期望大小为 n.

2.3 Delaunay 三角剖分

基于点集 P 构造的 Delaunay 三角剖分是一张图 $DT = (V, E)$, 其中 $V = P$, 对 $\forall p, p' \in P$, 边 $(p, p') \in E$ 当且仅当不存在 $B(c, r)$ 通过 p, p' 且 $B(c, r)$ 内包含 P 中其他的点. 细节内容请参阅文献 [25].

我们援引文献 [25] 中的如下定理.

定理 1. 设输入点集 P 如第 2.2 节中所描述, 则依 P 所构造的 Delaunay 三角剖分中最长边的期望长度为 $\Theta(\log^{1/d} n)$.

推论 1. 设输入点集 P 如第 2.2 节中所描述, 则存在常数 C_P, 使得依 P 所构造的 Delaunay 三角剖分中最长边的期望长度不超过 $C_P \log^{1/d} n$.

2.4 问题定义

我们首先给出 ε-NN 问题和 (c, r)-NN 问题的形式化定义. 在以下定义中使用的距离函数 $D(\cdot, \cdot)$ 均为欧氏距离

函数.

定义 1 (ε-NN 问题). 设输入点集 P 如第 2.2 节所描述, $q \in S$ 为查询点, ε 为近似参数, 则 ε-NN 问题的输出为点 $p' \in P$, 使得 $D(p',q) \leqslant (1+\varepsilon)D(p^*,q)$, 其中 p^* 是 P 中距离 q 最近的点, 即 $D(p^*,q) = \min_{p \in P} D(p,q)$.

定义 2 ((c, r)-NN 问题). 设输入点集 P 如第 2.2 节所描述, $q \in S$ 为查询点, $r > 0$ 为查询范围, $c > 1$ 为近似参数, 则 (c, r)-NN 问题的输出如下: 若存在 p 使得 $D(p,q) < r$, 则输出某点 p' 满足 $D(p',q) \leqslant cr$; 若 $\forall p \in P$ 都有 $D(p,q) > cr$ 则输出否定答案 No.

本文所研究的问题为设计从 ε-NN 问题到 (c, r)-NN 的高效图灵归约, 即以一个求解 (c, r)-NN 的算法 A_{crnn} 作为神谕, 设计求解 ε-NN 问题的高效算法. 设计的要点在于使用恰当的预处理算法和数据结构, 从而尽量减少调用 A_{crnn} 的次数. 算法的时间和空间复杂度分为 3 部分, 一是预处理算法的时间复杂度, 二是通过预处理算法构建的数据结构所占用的空间复杂度, 三是查询时间复杂度, 也即调用 A_{crnn} 的次数.

3 算 法

本文将要介绍的算法分为两部分, 即预处理算法和查询算法.

3.1 预处理算法

预处理算法接受如定义 1 中所述的输入, 返回一棵树型结构 T, 用于支持查询阶段的算法. 这棵树称为分裂树, 其结构形式化定义如下.

定义 3 (分裂树). 一棵分裂树 T 满足如下条件.

(1) 树中的每个结点都代表一个 d 维矩形 $R \subseteq S$, 且根结点为 S.

(2) 任一非叶结点 $R \in T$ 都有两个子结点, 其构造方法如下: 取 R 的最长边 $I_i(R) = [a_i, b_i]$, 将其从中点分成两部分, $I_i^{(1)}(R) = [a_i, (a_i+b_i)/2]$, $I_i^{(2)}(R) = [(a_i+b_i)/2, b_i]$, 则 $R^{(1)} = I_1(R) \times \ldots \times I_i^{(1)}(R) \times \ldots \times I_d(R)$, $R^{(2)} = I_1(R) \times \ldots \times I_i^{(2)}(R) \times \ldots \times I_d(R)$, 它们即为 R 的两个子结点.

(3) 每个叶结点 R 满足 $|R \cap P| = 1$.

对于分裂树 T 的每个结点 R, 算法对其构造了一个数据结构 $Nbr(R)$, 保存和 R 满足一定位置关系的 T 中的其他结点. 在下面的定义中, 记 $root(T)$ 为 T 的根结点. 对于 T 中某结点 R, 记 $dep_T(R)$ 为结点 R 在树 T 中的深度, 也即从 R 到 $root(T)$ 上的路径上边的数量. $Nbr(R)$ 的定义如下.

定义 4. 对于 $\forall R \in T$, $\forall R' \in Nbr(R)$ 满足以下条件.

(1) $dep_T(R') = dep_T(R)$.

(2) 设 $[a_i^{(1)}, b_i^{(1)}]$ 为 R 在第 i 维上的定义区间, $[a_i^{(2)}, b_i^{(2)}]$ 为在 R' 第 i 维上的定义区间, 则对于 $\forall i \in [1, d]$, 有 $([a_i^{(1)}, b_i^{(1)}] \cap [a_i^{(2)}, b_i^{(2)}] \neq \emptyset) \vee (|a_i^{(1)} - b_i^{(2)}| \leqslant 2C_p \log^{1/d} n) \vee (|a_i^{(2)} - b_i^{(1)}| \leqslant 2C_p \log^{1/d} n)$ 为真.

下面给出算法 1 即为预处理算法的伪代码. 预处理算法分为两个主要过程, 一是进行构造分裂树 T, 二是对 T 中每个结点建立其 Nbr 集. 构造分裂树的过程是一个比较简单的递归过程, 其伪代码在算法 2 中给出. 建立 Nbr 集是一个层序遍历的过程, 使用队列数据结构来完成层序遍历. 预处理算法的详细描述请见下面的算法 1. 在算法 1 的第 11 行中, 常数 C_P 的值根据推论 1 选取.

算法 1. 预处理算法.

输入: 点集 P, 查询点 q, 超立方体 S;
输出: 一棵树结构 T.

1 初始化一棵树 T, 其根为 S
2 Split($root(T)$)
3 $Nbr(root(T)) \leftarrow \varnothing$
4 初始化空队列 Q, 将 $root(T)$ 入队

```
5 While 队列 Q 非空 do
6     R_c←队列 Q 中队顶元素
7     if |R_c ∩ P|>1 then
8       Sons_c← {R_c } ⋃Nbr(R_c) 中结点的所有子结点
9       令 R_1, R_2 为 R_c 的两个子结点
10      foreach R′∈ Sons_c\ {R_1} do
11        if TestNbr(R′, R_1)==true then
12            Nbr(R_1)← Nbr(R_1) ⋃ {R′}
13        end
14      end
15      foreach R′∈ Sons_c\{R_2} do
16        if TestNbr(R′, R_2}==true) then
17            Nbr(R_2)← Nbr(R_2) ⋃ {R′}
18        end
19      end
20      将 R_c 从 Q 中出队, 将 R_1, R_2 入队
21    end
22 end
```

算法 2. 算法 1 中用到子程序.

```
1 Procedure Split(R):
2     取 R 的最长边, 将其从中点一分为二, 得到两个较小的矩形 R_1 和 R_2
3     将 R_1, R_2 作为 R 的子结点加入到树 T 中
4     Split(R_1)
5     Split(R_2)
6 end
7 Procedure TestNbr(R_1, R_2):
8     for i=1 to d do
9         [a_i^(1), b_i^(1)]← R_1 在第 i 维上的定义区间
10        [a_i^(2), b_i^(2)]← R_2 在第 i 维上的定义区间
11        if ([a_i^(1), b_i^(1)]∩[a_i^(2), b_i^(2)]=∅) ∧ (|a_i^(1)−b_i^(2)| ≥ 2C_P log^(1/d) n ∧ |a_i^(2)−b_i^(1)| ≥ 2C_P log^(1/d) n) then
12            return false
13        end
14    end
15    return true
16 end
```

3.2 查询算法

在预处理算法结束之后, 查询算法基于预处理算法所返回的分裂树 T 来解决第 2.4 节中给出的归约问题, 算法的执行过程如下.

(1) 设当前考虑的结点 R_c 为 T 的根.

(2) 设当前考虑的点集 P_c 为 R_c 和 $Nbr(R_c)$ 中包含的所有点.

(3) 调用 A_{crnn}, 调用的输入点集为 P_c, 查询点为 q, 查询范围为 $C_P\log^{1/d}n$, 近似参数为 $1+\varepsilon$, 其中 C_P 是推论 1 中所给出的值, ε 是 ε-NN 问题的输入参数. 在后面的引理 4 中我们将证明, 此次调用一定会返回一个点 p'.

(4) 检视 $R_c \cup Nbr(R_c)$ 的所有子结点, 设子结点 R' 满足 $p' \in R'$, 令 $R_c \leftarrow R'$.

(5) 若 R_c 中包含的点数多于 1, 则返回第 2 步, 否则继续.

(6) 在 P_c 中直接搜索 ε-NN 的结果.

算法的伪代码在下面的算法 3 中给出.

算法 3. 查询算法.

输入: 输入点集 P, 查询点 q, 解决 (c, r)-NN 的算法 A_{crnn}, 算法 1 的输出 T;
输出: ε-NN 的结果.

$R_c \leftarrow root(T)$
while $|R_c \cap P| > 1$ do
    $P_c \leftarrow \bigcup_{R' \in \{R_c\} \cup Nbr(R_c)} \{R' \cap P\}$
    调用 $A_{\text{crnn}}(P_c, q, C_P\log^{1/d}n, 1+\varepsilon)$, 得到返回点 p'
    检视所有 $\{R_c\} \cup Nbr(R_c)$ 的所有子结点, 设子结点 R' 满足 $p' \in R'$
    $R_c \leftarrow R'$
end
在 P_c 中用直接搜索找到 ε-NN 的结果并返回

4 分 析

4.1 准备工作

引理 1. 取 C_P 为推论 1 中所给出的常数, 则 $E[|B(q, C_P\log^{1/d}n) \cap P|] \geqslant 1$.

证明: 由问题定义, 输入点集 P 和查询点 q 都位于边长为 $n^{1/d}$ 的区域 S 中, 则根据推论 1, 由 $P \cup \{q\}$ 构造的 Delaunay 三角剖分的期望最长边为 $C_P\log^{1/d}n$. 于是在期望情况下, 在以 q 为中心、$C_P\log^{1/d}n$ 为半径的范围之内至少存在一个点, 引理得证.

引理 2. 对 T 中任意结点 R, $Nbr(R)$ 集合的期望大小为 $O(\log n)$, 即对于 $\forall R \in T$, $E[|Nbr(R)|] = O(\log n)$.

证明: 由泊松点过程的性质, 体积为 1 的 d 维矩形中包含的期望点数为 1. 由于 T 的每个结点中至少包含 P 中的一个点, 所以每个结点的体积在期望情况下不小于 1. $Nbr(R)$ 的定义可知, $Nbr(R)$ 所包含的区域在各维上的边长都比 R 多 $O(\log^{1/d}n)$, 进而 $Nbr(R)$ 所包含的区域的体积为 $O(\log n)$. 于是, 对于体积为 1 的结点 R, $Nbr(R)$ 中包含的体积为 1 的结点数最多为 $O(\log n)$. 对于体积大于 1 的结点 R, $Nbr(R)$ 中所包含的结点数则一定少于 $O(\log n)$. 总之, T 中任意结点的 Nbr 集合的期望大小为 $O(\log n)$, 引理得证.

引理 3. 预处理算法返回的分裂树 T 具有如下性质.

(1) T 中最多有 $O(n)$ 个结点.

(2) T 的期望高度为 $O(\log n)$.

证明: 根据定义 3 可知, T 的每个叶结点中有 1 个数据点, 进而 T 中有 $O(n)$ 个叶结点. T 中每个非叶结点都有两个子结点, 则当 T 是一棵完全二叉树时, T 中有 $2n-1$ 个结点, 若 T 不是完全二叉树则 T 中结点会更少. 于是 T 中最多有 $O(n)$ 个结点, 性质 1 得证.

由 T 的构造过程可知, 结点 R 的子结点是通过把 R 的最长边二等分得到的. 因此两个子结点 R_1 和 R_2 所代表的 d 维矩形具有相同的体积. 由泊松点过程的性质, R_1 和 R_2 中包含的期望点数相同. 也即树 T 中每一层的结点中

包含的点数的期望都比上一层少一半, 则 T 的期望高度为 $O(\log n)$.

4.2 正确性分析

引理 4. 在期望情况下, 算法 3 的第 4 行每次调用 A_{crnn} 都能返回一个点作为结果.

证明: 在第 1 次循环时, $P_c = P$. 又由引理 1, 在期望情况下 $B(q, C_P \log^{1/d} n)$ 至少包含一个点, 于是根据 (c, r)-NN 问题的定义, 第 1 次调用 A_{crnn} 能够返回一个点. 现假设在第 k 次调用时返回了一个点 p'. 由 R_c 和 P_c 的设置方法可知, 在第 $k+1$ 次调用时, p' 仍在第 $k+1$ 轮循环中所更新的 P_c 中. 由 (c, r)-NN 问题的定义, 此次调用 A_{crnn} 算法仍能返回一个点. 总之, 算法每次调用 A_{crnn} 在期望情况下每次都能返回一个点作为结果.

引理 5. 设 p^* 为 q 在 P 中的最近邻, 即 $D(p^*, q) = \min_{p \in P} D(p, q)$, 则在算法 3 的每一次循环中, 总有 $p^* \in P_c$.

证明: 由引理 4, 算法 3 的第 4 行每次都能返回一个点 p'. 由 (c, r)-NN 的定义可知, (c, r) 满足 $D(p', q) \leqslant C_P \log^{1/d} n$. 另由引理 1, 查询点 q 的最近邻距离不超过 $C_P \log^{1/d} n$. 即 $D(p^*, q) \leqslant C_P \log^{1/d} n$. 于是有 $D(p', p^*) \leqslant D(p', q) + D(p^*, q) \leqslant 2C_P \log^{1/d} n$.

由算法 3 的伪代码, R' 是包含 p' 的子结点. 由定义 4 可知, $Nbr(R')$ 覆盖的区域在每个维度上都比 R' 拓展了 $2C_P \log^{1/d} n$, 从而 $Nbr(R')$ 可以覆盖 $B(q, 2C_P \log^{1/d} n)$. 因为 $D(p', p^*) \leqslant 2C_P \log^{1/d} n$, 则 p^* 也能够被 $Nbr(R')$ 覆盖. 由 P_c 的设置方法, p^* 能够被 $Nbr(R')$ 覆盖等价于 $p^* \in P_c$, 引理得证.

定理 2. 算法 3 能够正确地返回 ε-NN 的结果.

证明: 由引理 5, 在算法的每一次循环中都有 $p^* \in P_c$, 其中 p^* 是 q 在 P 中的最近邻. 于是, 算法第 8 行的直接搜索一定能够能产生 ε-NN 的正确结果.

4.3 复杂度分析

定理 3. 算法 1 的期望时间复杂度为 $O(n \log n)$.

证明: 算法 1 的时间复杂度分为两部分, 一是构造分裂树, 二是对分裂树的所有结点建立 *Nbr* 集.

构造分裂树的部分是分治算法, 根据算法 1 的内容和引理 3 中的证明, 可以写出关于此分治算法的期望时间复杂度的递归公式: $E[T(n)] = 2E[T(n/2)] + O(n)$. 公式中 $2E[T(n/2)]$ 来自于子结点中期望点数为父结点的一半, 已经在引理 3 的证明过程中得以证明. 关于 $O(n)$ 项, 观察算法 1 可知, 将父结点分裂为两个子结点需要 $O(n)$ 时间, 遍历 $Nbr(R)$ 需要 $O(|Nbr(R)|) = O(\log n)$ 时间, 则分治算法每次循环体需要 $O(n)$ 时间. 容易证明, 此递归公式的解为 $E[T(n)] = O(n \log n)$.

现在分析对分裂树的所有结点建立 *Nbr* 集的复杂度. 由引理 3, 分裂树 T 中总共有 $O(n)$ 个结点. 由引理 2, 每个结点的 *Nbr* 集的期望大小都为 $O(\log n)$. 由算法 1 的伪代码可知, 每个结点的 *Nbr* 集最多被遍历一次. 从而建立所有结点的 *Nbr* 集的期望时间复杂度 $O(n \log n)$.

将以上两部分时间复杂度相加, 则得到算法 1 的期望时间复杂度为 $O(n \log n)$.

定理 4. 算法 1 需要 $O(n \log n)$ 期望空间.

证明: 算法 1 需要的空间即为存储分裂树 T 所需的空间. 由引理 3, T 中共有 $O(n)$ 个结点. 由引理 2, 对 T 中每个结点 R 存储的 $Nbr(R)$ 集合的期望大小为 $O(\log n)$. 于是, 存储 T 总共需要 $O(n \log n)$ 期望空间.

定理 5. 算法 3 需要调用 A_{crnn} 算法 $O(\log n)$ 次, 也即算法以 $O(\log n)$ 的查询时间解决第 2.4 节中给出的归约问题.

证明: 算法 3 在每一次循环中调用 $O(\log n)$ 一次, 而循环的执行次数等于分裂树 T 的高度. 由引理 3, T 的期望高度为 $O(\log n)$, 于是算法 3 需要调用 A_{crnn} 算法 $O(\log n)$ 次得证.

5 结 论

本文是我们发表的文献 [1] 中工作的延续, 研究了基于图灵归约求解 ε-NN 问题在输入点集服从泊松点过程条件下的期望复杂度. 我们提出了与文献 [1] 中不同的算法, 并分析了其复杂度, 证明了它具有 $O(n \log n)$ 期望预处理时间、$O(n \log n)$ 期望空间和 $O(\log n)$ 期望查询时间. 本文解决了我们在文献 [1] 中提出的未来工作.

References:

[1] Ma HZ, Li JZ. An algorithm for reducing approximate nearest neighbor to approximate near neighbor with O(log*n*) query time. In: Proc. of the 12th Int'l Conf. on Combinatorial Optimization and Applications. Atlanta: Springer, 2018. 465–479. [doi: 10.1007/978-3-030-04651-4_31]

[2] Har-Peled S. A replacement for voronoi diagrams of near linear size. In: Proc. of the 42nd IEEE Symp. on Foundations of Computer Science. Newport Beach: IEEE, 2001. 94–103. [doi: 10.1109/SFCS.2001.959884]

[3] Har-Peled S, Indyk P, Motwani R. Approximate nearest neighbor: Towards removing the curse of dimensionality. Theory of Computing, 2012, 8(1): 321–350. [doi: 10.4086/toc.2012.v008a014]

[4] Indyk P, Motwani R. Approximate nearest neighbors: Towards removing the curse of dimensionality. In: Proc. of the 30th Annual ACM Symp. on Theory of Computing. Dallas: ACM, 1998. 604–613. [doi: 10.1145/276698.276876]

[5] Arya S, Mount DM, Netanyahu NS, Silverman R, Wu AY. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. Journal of the ACM, 1998, 45(6): 891–923. [doi: 10.1145/293347.293348]

[6] Goel A, Indyk P, Varadarajan K. Reductions among high dimensional proximity problems. In: Proc. of the 12th Annual ACM-SIAM Symp. on Discrete Algorithms. Washington: Society for Industrial and Applied Mathematics, 2001. 769–778.

[7] Iscen A, Tolias G, Avrithis Y, Chum O. Mining on manifolds: Metric learning without labels. In: Proc. of the 2018 IEEE/CVF Conf. on Computer Vision and Pattern Recognition. Salt Lake City: IEEE, 2018. 7642–7651. [doi: 10.1109/CVPR.2018.00797]

[8] Lu XL. Improving search using proximity-based statistics. In: Proc. of the 38th Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval. Santiago: ACM, 2015. 1065. [doi: 10.1145/2766462.2767847]

[9] Luo YC, Zhu J, Li MX, Ren Y, Zhang B. Smooth neighbors on teacher graphs for semi-supervised learning. In: Proc. of the 2018 IEEE/CVF Conf. on Computer Vision and Pattern Recognition. Salt Lake City: IEEE, 2018. 8896–8905. [doi: 10.1109/CVPR.2018.00927]

[10] Zheng YX, Guo Q, Tung AKH, Wu S. LazyLSH: Approximate nearest neighbor search for multiple distance functions with a single index. In: Proc. of the 2016 Int'l Conf. on Management of Data. San Francisco: ACM, 2016. 2023–2037. [doi: 10.1145/2882903.2882930]

[11] Minsky M, Papert S. Perceptrons. Cambridge: MIT Press, 1969.

[12] Clarkson KL. An algorithm for approximate closest-point queries. In: Proc. of the 10th Annual Symp. on Computational Geometry. Stony Brook: ACM, 1994. 160–164. [doi: 10.1145/177424.177609]

[13] Weber R, Schek HJ, Blott S. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In: Proc. of the 24th Int'l Conf. on Very Large Data Bases. New York City: Morgan Kaufmann Publishers Inc., 1998. 194–205.

[14] Arya S, Mount DM. Approximate nearest neighbor queries in fixed dimensions. In: Proc. of the 4th Annual ACM-SIAM Symp. on Discrete Algorithms. Austin: Society for Industrial and Applied Mathematics, 1993. 271–280.

[15] Kleinberg JM. Two algorithms for nearest-neighbor search in high dimensions. In: Proc. of the 29th Annual ACM Symp. on Theory of Computing. El Paso: ACM, 1997. 599–608. [doi: 10.1145/258533.258653]

[16] Datar M, Immorlica N, Indyk P, Mirrokni VS. Locality-sensitive hashing scheme based on p-stable distributions. In: Proc. of the 20th Annual Symp. on Computational Geometry. Brooklyn: ACM, 2004. 253–262. [doi: 10.1145/997817.997857]

[17] Charikar MS. Similarity estimation techniques from rounding algorithms. In: Proc. of the 34th Annual ACM Symp. on Theory of Computing. Montreal: ACM, 2002. 380–388. [doi: 10.1145/509907.509965]

[18] Li P, Mitzenmacher M, Shrivastava A. Coding for random projections. In: Proc. of the 31th Int'l Conf. on Machine Learning. Beijing: JMLR.org, 2014. II-676–II-684.

[19] Terasawa K, Tanaka Y. Spherical LSH for approximate nearest neighbor search on unit hypersphere. In: Proc. of the 10th Int'l Conf. on Algorithms and Data Structures. Halifax: Springer, 2007. 27–38. [doi: 10.1007/978-3-540-73951-7_4]

[20] Andoni A, Razenshteyn IP. Tight lower bounds for data-dependent locality-sensitive hashing. In: Proc. of the 32nd Int'l Symp. on Computational Geometry. Dagstuhl: Schloss Dagstuhl—Leibniz-Zentrum fuer Informatik, 2016. 1–11. [doi: 10.4230/LIPIcs.SoCG.2016.9]

[21] Motwani R, Naor A, Panigrahy R. Lower bounds on locality sensitive hashing. SIAM Journal on Discrete Mathematics, 2008, 21(4): 930–935. [doi: 10.1137/050646858]

[22] Tao YF, Yi K, Sheng C, Kalnis P. Efficient and accurate nearest neighbor and closest pair search in high-dimensional space. ACM Trans. on Database Systems, 2010, 35(3): 20. [doi: 10.1145/1806907.1806912]

[23] Andoni A. Nearest neighbor search: The old, the new, and the impossible [Ph.D. Thesis]. Cambridge: Massachusetts Institute of Technology, 2009.

[24] Wang JD, Shen HT, Song JK, Ji JQ. Hashing for similarity search: A survey. arXiv:1408.2927, 2014.

[25] Andoni A, Indyk P. Nearest neighbors in high-dimensional spaces. In: Goodman JE, O'Rourke J, Tóth CD, eds. Handbook of Discrete and Computational Geometry. 3rd ed., Boca Raton: Chapman and Hall/CRC, 2017. 1135–1155.

[26] Bern M, Eppstein D, Yao F. The expected extremes in a Delaunay triangulation. Int'l Journal of Computational Geometry & Applications, 1991, 1(1): 79–91. [doi: 10.1142/S0218195991000074]

[27] Bordenave C. Navigation on a Poisson point process. The Annals of Applied Probability, 2008, 18(2): 708–746. [doi: 10.1214/07-AAP472]

[28] Ahsen M, Ali Hassan S. A Poisson point process model for coverage analysis of multi-hop cooperative networks. In: Proc. of the 2015 Int'l Wireless Communications and Mobile Computing Conf. (IWCMC). Dubrovnik: IEEE, 2015. 442–447. [doi: 10.1109/IWCMC.2015.7289124]

[29] Kingman JFC. Poisson Process. Oxford: Oxford University Press, 1993.

马恒钊(1995－), 男, 博士, 主要研究领域为大数据计算, 亚线性算法.

李建中(1950－), 男, 博士, 教授, 博士生导师, CCF 会士, 主要研究领域为数据库, 大数据计算, 无线传感网.

闫跃(1964－), 男, 讲师, 主要研究领域为数据挖掘.