Blockcourse on High-Dimensional Computational Geometry Spring 2013

Lecture 7: Introduction — July 30, 2013

Helmut Alt, Wolfgang Mulzer *Scribe: Heuna Kim, Sang Duk Yoon*

1 c-Approximate Near Neighbor Problem

We aim to build an algorithm for c-approximate near neighbor problems by using queries to the c-approximate interval near neighbor algorithm.

Let P be a set of n points in $\mathbb{R}^d$ and $r > 0$. We define $CC_P(r)$ as the set of connected components in $\bigcup_{p\in P} B_r(p)$ where $B_r(p)$ is the ball of radius r centered around p. We also denote $r_{med} := \min_r\{CC_P(r) \text{ has a component of size } \frac{n}{2}+1\}$.

Suppose that the largest connected component of $CC_P(r)$ has exactly $\frac{n}{2}+1$ points. To build an interval nearest neighbor structure, we need to define an interval. Set $r_{top} = \alpha \cdot n \cdot r_{med} \cdot \frac{\log n}{\gamma}$, where α is sufficiently large and $\gamma \in (0, \frac{1}{2})$. Build an interval near neighbor structure for the interval $[\frac{r_{med}}{2}, r_{top}]$. Recursively, build a data structure for each connected component of $CC_P(r_{med})$. Also, $P' = \{p'_1, p'_2, \ldots, p'_k\}$ that has exactly one point from each cluster in $cc_p(r_{med})$, recursively build data structures for p'.

Observation 1. *$|p'| \le \frac{n}{2}$ and the size of each connected component $\le \frac{n}{2}+1$*

Query processing

For a given query point $q \in \mathbb{R}^d$, query approximate interval near neighbor structure with q.

(i) $d(q,P) \le \frac{r_{med}}{2}$ with witness p^* : find the cluster c^* such that $p^* \in C^*$ and recurse on C^*.

(ii) $d(q,P) \ge r_{top}$: recurse on P'.

(iii) $d(q,P) \in (\frac{r_{med}}{2}, r_{top})$: get c-approximation which we return.

Analysis of approximation

(i) $d(q,P) \le \frac{r_{med}}{2}$

Claim 2. *C^* contains the nearest neighbor of q in P.*

Proof. Let C_i, C_j be clusters in $CC_P(r_{med})$. We can see that $d(C_i, C_j) > 2 \cdot r_{med}$, i.e., each point in C_i has distance larger than $2 \cdot r_{med}$ from each point in C_j. Since $d(q, C^*) \le \frac{r_{med}}{2}$, we have $d(q, C) \ge \frac{3}{2} \cdot r_{med}$ for all cluster $C \ne C^*$. Therefore, the nearest neighbor of q in P has to be in C^* as claimed. □

(ii) $d(q,P) \ge r_{top} = \alpha \cdot n \cdot r_{med} \cdot \frac{\log n}{\gamma}$

Observation 3. *Let C be a cluster, the diameter of C is at most $2 \cdot r_{med} \cdot (n-1)$.*

In particular, for any two points $p_i, p_j \in C$, $|d(q,p_i) - d(q,p_j)| \leq 2 \cdot r_{med} \cdot (n-1) \leq \frac{2}{\alpha} \cdot \frac{\gamma}{\log n} \cdot r_{top}$. By taking one representative for each cluster, we have a multiplicative error of at most $(1 + \frac{2}{\alpha} \cdot \frac{\gamma}{\log n})$, i.e., $d(q,P') \leq \frac{2}{\alpha} \cdot \frac{\gamma}{\log n} \cdot d(q,P)$.

(iii) $d(q,P) \in (\frac{r_{med}}{2}, r_{top})$

We get an approximate nearest neighbor in the current point set. Overall, the worst case is in each level of recursion, we have case (ii) and then case (iii); in this case, approximation factor will be $(1 + \frac{2}{\alpha} \cdot \frac{\gamma}{\log n})^l \cdot c$, where l is the number of the recursion. The problem size halves at each level of recursion, so $l = \log n$. also, we can bound the approximation factor:

$$(1 + \frac{2}{\alpha} \cdot \frac{\gamma}{\log n})^{\log n} = 1 + \sum_{i=2}^{\log n} \binom{\log n}{i} \cdot (\frac{2}{\alpha} \cdot \frac{\gamma}{\log n})^i \leq 1 + \sum_{i=1}^{\log n} (\frac{2\gamma}{\alpha})^i \leq 1 + \frac{2\gamma}{\alpha} \cdot o(1) \leq (1+\gamma) \cdot c,$$

with sufficiently large α.

2 Big Picture of the Reduction

c-approximate r-near neighbor problem

We suppose that we have an algorithm for c-approximate r-near neighbor problem. Reduce c-approximate r-near neighbor problem to the approximate interval near neighbor problem.

$(1+\gamma)c$-approximate interval near neighbor problem with range $[a,b]$, $\gamma \in (0, \frac{1}{2})$

For given query point q,

(i) if $d(q,P) \leq a$, return a witness,

(ii) if $d(q,P) \geq b$, done.

(iii) Otherwise, get $(1+\gamma) \cdot c$-approximation : $O(\log \frac{1}{\gamma} \log \frac{cb}{a})$ many r-NN queries are needed.

Reduce c-approximate interval near neighbor problem to c-approximate nearest neighbor problem.

$(1+\gamma)(1+\gamma')c$-approximate nearest neighbor problem with $\gamma' \in (0, \frac{1}{2})$

Build several interval near neighbor structures with $O(n \log n)$ points. We can get $(1+\gamma)(1+\gamma')c$-approximate nearest neighbor with $O(\log n)$ queries.

3 Issues in the Previous Query Processing

α. It assumed that the largest connected componenet has exactly $\frac{n}{2} + 1$ points.

β. An efficient preprocessing : how to find r_{med} and clusters.

γ. The failure probability was ignored.

($\alpha \Rightarrow$) Define $r_{bot} = \frac{r^*_{med}}{2n} \leq r_{med}$. Build the interval near neighbor structure for interval $[\frac{r_{bot}}{2}, r_{top}]$. For recursion in case (i), use clustering with respect to r_{bot} (i.e. $CC_p(r_{bot})$) for P' and use clusters with respect to r^*_{med}.

($\beta \Rightarrow$) Use r^*_{med} from the homework problem 6.1, do clustering with r^*_{med}. Don't compute $CC_p(r)$, but suitable approximation using a dynamic c-approximate r-near neighbor structure.

($\gamma \Rightarrow$) Use union bound and lose factor $O(\log \frac{n}{\gamma})$.

4 Pseudo Code and Query Algorithm

Pseudo-Code

```
% construct data structure
1:          if |P| = 1 then store P and return
2:          repeat
3:                  find r*_med as in Problem 6.1.
4:                  set r_bot = r*_med / 2nc, r_top = αn (log n / γ) r*_med
5:                  use dynamic r-near neighbor structure to compute partitioning
                        P = {P_1, P_2, ..., P_k} and P' = {p'_1, p'_2, ..., p'_k} at P
                        such that CC_P(r_bot) ⊑ P ⊑ CC_p(c r_bot)
                        where ⊑ represents refinements,
                        and CC_P(r*_med) ⊆ P' ⊆ CC_p(c r*_med)
6:          until |P_i| ≤ n/2 for i = 1, ..., k and k' ≤ n/2
% build interval near neighbor structure for [r_bot/2, r_top]
7:          build recursive structures for P_1, ..., P_k and for representatives in P'.
```

Query Algorithm:

input q:

query interval NN structure

(i) if we get an approximate near neighbor, return it.

(ii) if we get an answer $d(q, P) \geq r_{top}$, use the data structure for P'.

(iii) if we get $d(q, p^*) \leq r_{bot}/2$ for some $p^* \in P$, recurse on the data structure for the cluster in $\mathcal{P}$ with p^*

Theorem 4. *Let $P \subseteq \mathbb{R}^d, |P| = n$, let $c = 1 + \varepsilon$, $f \in (0, 1)$ and $\gamma \in (\frac{1}{n}, 1)$ be parameters. Assume we have c-approximate r-near neighbor structure with space $S(n, c, f)$, query time $Q(n, c, f)$, construction time $T(n, c, f)$ and deletion time $D(n, c, f)$. (Assume these functions are well-behaved.) Then we can build a data structure for $(1 + \gamma)c$-approximate nearest neighbor queries with*

- *Query time : $O(\log \frac{n}{\gamma}) \cdot Q(n, c, f)$*

- *Space requirement :* $\frac{S(n,c,f)}{\gamma}\log^2 n$
- *Construction time :* $\frac{T(n,c,f)}{\gamma}\log^2 n + n\log n[Q(n,c,t) + D(n,c,t)]$
- *Failure probability :* $f \cdot \log\frac{n}{\gamma}$

Note that the level of the recursion tree is $O(\log n)$.

5 c-approximate r-near neighbor

Approach using Locality Sensitive Hashing (LSH):

Definition 5. *Let (X, d) be a metric space. A family $\mathcal{H} = \{h : X \to U\}$ where U is a set of buckets is called (r_1, r_2, p_1, p_2)-**sensitive** if the following holds for all $x, y \in X$.*

(i) if $d(x,y) \le r_1$, then $Pr_{h\in\mathcal{H}}[h(x) = h(y)] \ge p_1$

(ii) if $d(x,y) \ge r_2$, then $Pr_{h\in\mathcal{H}}[h(x) = h(y)] \le p_2$

We need $r_1 \le r_2$ and $p_1 > p_2$.

Suppose we have a (r, cr, p_1, p_2)-sensitive family.

- Build a data structure $\Rightarrow$ Pick a random $h \in \mathcal{H}$ and build a hash table for P with h as the hash function
- Query $\Rightarrow$ Given $q \in \mathbb{R}^d$, compute $h(q)$, check the corresponding bucket and let P_B be the points in this bucket.
  For all $p \in P_B$, compute $d(p,q)$ if distance is at most cr return p.
  Otherwise there is no such point, return $\perp$.
- We hope the followings :
  - If $d(q,P) \le r$, P_B is likely to contain a point with $d(q,p) < cr$, and
  - P_B contains few points with $d(p,q) > cr$.