Blockcourse on High-Dimensional Computational Geometry Spring 2013

Lecture 6: Approximate Nearest Neighbor — July 29, 2013

Wolfgang Mulzer *Scribe: Christoph Krüger, Max Wisniewski*

1 Volume of Convex Bodies

1.1 Dyer-Frieze-Kamman

A FPRAS for the Volume of a convex body $K \subseteq \mathbb{R}^d$ can be obtained as follows.
Let

$$B = K_0 \dots K_{q-1} \supseteq K_q = K \supseteq B'$$

be as sequence of convex sets.

For every $K_{i-1} \supset K_i$ we take random points from K_{i-1} and check if they are in K_i. In the last lecture we saw how to obtain a random cell with a random work of polynomial length such that the distribution is uniform.

1.2 Proof (continued).

1.2.1 Computing the K_i.

We use an additional factor such that

$$r \cdot B = K_0 \dots K_{q-1} \supseteq K_q = K \supseteq B \qquad , r = \sqrt{d}(d+1)$$

.

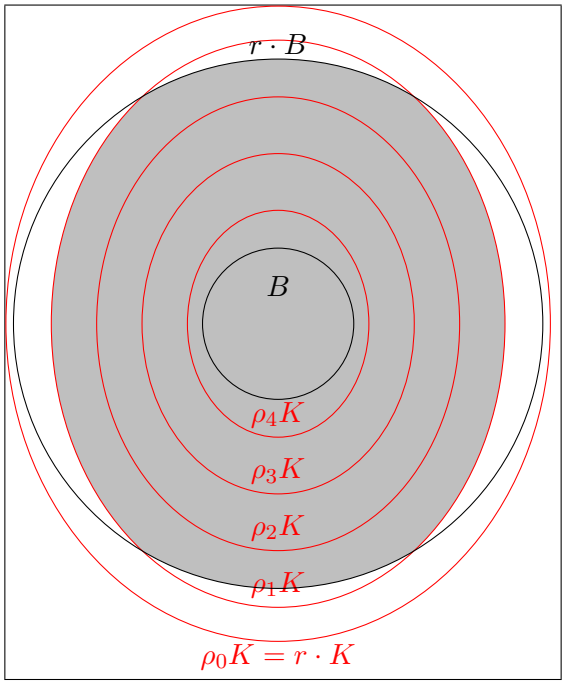

Figure 1: Series of $\rho_i K$ with highlighted K_1

Let

$$\rho = 1 - \frac{1}{d}$$
$$q = \lceil \log_{\frac{1}{\rho}} r \rceil$$
$$\rho_i = \max(\rho^i r, 1)$$
$$K_i = \rho_i K \cap rB$$

To compute the volume of $K_q = K$ we want to compute the product

$$\mathrm{Vol}(K) = \prod_{r=1}^{q} \frac{\mathrm{Vol}(K_i)}{\mathrm{Vol}(K_{i-1})} \cdot vol(rB)$$

.

To determine each $\frac{\mathrm{Vol}(K_i)}{\mathrm{Vol}(K_{i-1})}$ we proceed as follows

1. perform a sequence of trials for each of the fractions. In each we
   (a) perform a technical random walk on cubes of K_{i-1} for steps that ends in some cube c.
   (b) put small grid of cell size on c and get a random grid point of $x_0 \in c$.
   (c) check whether $x_0 \in K_{i-1}$
   (d) and if yes whether $x_0 \in K_i$
2. use this trials to approximate $\frac{\text{Vol}(K_i)}{\text{Vol}(K_{i-1})}$ which is a approximation for $\text{Vol}(K)$

The ratio itself $\left(\frac{\text{Vol}(K_i)}{\text{Vol}(K_{i-1})}\right)$ is bounded by ρ^d due to $\frac{1}{4} \leq \rho^d \leq \frac{1}{e}$.
The runtime altogether is

$$O\left(d^{23}\left(\log d\right)^5 \epsilon^{-2} \log \frac{1}{\delta}\right) \qquad \text{for} (\epsilon, \delta) - \text{FPRAS}$$

.

It has been improved to

$$O\left(d^8 \epsilon^{-2} \log\left(\frac{d}{\epsilon}\right) \log \frac{1}{\delta}\right)$$

.

For polytopes the a membership oracle can be simulated (we have to consider its running time as it is no longer an oracle) for

H-polytopes in $O(n \cdot d)$ with n halfspaces by checking each inequality $\langle h, x\rangle \leq 1$ for each $h \in H$ where we need $O(d)$ operations for each halfspace.

V-polytopes. To check whether the point x is in the convex hull of a given set of points ($x \in CH(P)$) we have to solve the following query

$$\begin{aligned} & \exists \alpha_0, \ldots \alpha_n \in [0,1] & \\ \text{s.t.} \quad & \alpha_1 v_1 + \cdots + \alpha_n v_n & = x \\ & \sum_{i=1}^{n} \alpha_i & = 1. \end{aligned}$$

This is a linear program where we are only interested in the existence of a solution. A linear program can be solved by the ellipsoid method or the inner point method in polynomial time in the size of the input.

Observe that only α_i are variables. The v_i are constants given by the problem.

2 Nearest Neighbor Searching

Given: $P \subset \mathbb{R}^d, |P| = n$.

Want: A data Structure to Nearest-Neighbor(NN)-queries such that
given $q \in \mathbb{R}^d$,
find $NN_P(Q) = \text{argmin}_{p \in P} d(q,p)$; the nearest neighbor of q in P.

2.1 Simple Data structure

Store all points P in a list.

Space: $\Theta(n \cdot d)$ enumerating every point and its coordinates.

Query: $\Theta(n \cdot d)$ for computing all distances of q to every point $p \in P$ and returning the maximum.

The space requirement of this data structure is very good, but the query time on the other hand is bad.

2.2 Voronoi Diagram (Ken Clarkson)

As seen last week we can compute the Voronoi Diagram of P by lifting the points to the unit paraboloid and using a point location data structure to find the nearest hyperplane in each step of the triangulation construction.

Space: $\Theta(n^{\lceil \frac{d}{2} \rceil + \epsilon})$.

Query: $O(2^O(d) \log q)$ (naive) or $d^{O(1)} \log n$ (improved by S meiser). For any $\epsilon > 0$ the constant in query time depends on ϵ.

The ϵ can be removed but query time gets worse. (claimed and not verified in exercise: $O(\log^2 n)$)

In this case the query time is very good but the space requirement is exponential in the dimension.

2.3 Improvement

The question after these two datastructures is, whether there is some data structure in between.

But for deterministic exact algorithm this is still an open question.

Conjecture 1 *"Curse of dimensionality"*
If we want a query time $d^{O(1)} poly(\log n)$ the space needs to depend exponentially on d.

There are only partial results but there is yet no proof of the conjecture, but Klee showed that voronoi-diagrams in $\mathbb{R}^d$ can have complexity $\Omega(n^{\lceil \frac{d}{2} \rceil})$.

2.4 Approximation & Randomness

Because there are no exact algorithms known, we look if we can get better bounds on space- and query time if we allow randomness and and approximate result.

2.4.1 Definitions

First we define the approximation of the result.

Definition 2 *(c-approximate nearest neighbor or C-NN)*
Given $P \subseteq \mathbb{R}^d$*, points* $c \geq 1$ *and* $q \in \mathbb{R}^d$*.*
A point $p \in P$ *is a c-approximate NN for* q *iff* $d(p,q) \subseteq c \cdot d(q,P)$*.* ⌟

And second the randomness to the algorithm.

Definition 3 *(c-approximate nearest neighbor with failure probability f)*
Given $P \subseteq \mathbb{R}^d$*, points* $c \geq 1$ *and* $q \in \mathbb{R}^d$*,* $f \in (0,1)$*.*
Build a DS such that

$$\forall q \in \mathbb{R}^d : \ \textit{Pr[DS returns a c-approximate NN for q]} \ \geq 1 - f.$$

⌟

The error will be with respect to an in advanced fixed $q \in \mathbb{R}^d$.

2.5 Solution(Indyk, Motwani & Har Peled)

The algorithm will be presented in two steps

1. Reduce to $a(n)$ approximate decision version.
2. Solve the decision version.

2.5.1 Decision-version of NN-problem

Definition 4 *(C-approximate r-near neighbor problem)*

Given: $P \subseteq \mathbb{R}^d, c \geq 1, r \geq 0$ *fixed, failure probability* f*.*

Goal: *Build a Data structure that does the following:*

For a query point q

1. *if* $d(q,P) \leq r$ *return* $p \in P$ *with* $d(q,p) \leq c \cdot r$ *with probability* $> 1 - f$*.*
2. *if* $d(q,P) \geq c \cdot r$ *always return* $\perp$*.*
3. *if* $r \leq d(q,P) \leq c \cdot r$ *either one can happen.*

⌟

Sometimes we want a dynamic data structure which allows to insert and delete points.

2.5.2 Reduction to decision version

With help of the c-approximate r-near neighbor structure we want to get a search structure for c-approximate nearest neighbor without using that much more space and time for the query.

Naive approach
First compute

$$d_{\max} = \max_{p_1,p_2 \in P} d(p_1, p_2)$$
$$d_{\min} = \min_{p_1,p_2 \in P} d(p_1, p_2)$$

For $i = 0, \dots M$, build c-approximate r_i-near neighbors structure Z_i

$$r_i = c^i \frac{d_{\min}}{2 \cdot c}$$
$$M = \log_c \left(\frac{d_{\max}}{d_{\min}} 2 \left(1 + \frac{1}{c+1} \right) \right)$$

which leads to:

$$r_0 = \frac{d_{\min}}{2 \cdot c},$$
$$r_1 = \frac{d_{\min}}{2},$$
$$r_2 = c \cdot \frac{d_{\min}}{2},$$
$$\vdots$$
$$r_M = \frac{1}{c} \cdot d_{\max}$$

Let $q \in \mathbb{R}^d$ be a query.

(a) if $d(q, P) \leq \frac{d_{\min}}{2c}$ if we query Z_0 with q, we got a point $p^* \in P : d(q, p^*) \leq \frac{d_{\min}}{2}$.

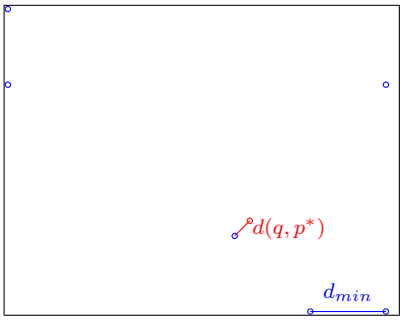

Figure 2: Case (b), if q is very close to a point in P.

It follows p^* is a nearest neighbor of q in P.
Otherwise there would be $p^{**} \in P$ with $d(q, p^{**}) \leq \frac{d_{\min}}{2} \Rightarrow d(p^*, p^{**}) \leq d(p^*, q) + d(q, p^{**}) \leq d_{\min}$ ↯

(b) if $d(q,p) \geq N_M = \frac{1}{(c-1)} d_{\max}$ all our datastructures Z_i might return $\perp$.

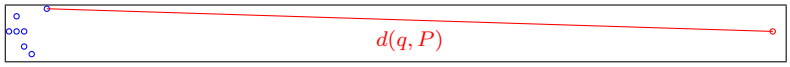

Figure 3: Case (c), if q is far away from P.

Take any $p \in P$ then we know

$$0 \leq d(q,p) - d(p, nn_p(q)) \leq d_{\max} \underbrace{\leq}_{\text{by assumption}} (c-1)d(q,p)$$

Thus:

$$d(q,p) \leq d(q,P) + (c-1) \cdot d(q,P) = c \cdot d(q,P)$$

$\Rightarrow$ p is a c-approximate nearest neighbor.

(c) if $d(q,p) \in [r_0, r_M]$

Suppose $d(q,p) \in (r_j, r_{j-1}]$ for some j. $r_j = c^j \frac{d_{\min}}{2c}$
Now we know that

i. for all $i \leq j-1 : Z_i$ returns $\perp$
   (caused by $d(q,p) > r_j \Rightarrow r_{j-1} < \frac{d(q,p)}{c}$)
ii. for all $i \geq j+1 : Z_i$ return some point p with distance at most $c \cdot r_i$
iii. the only data structure where we are not sure is Z_j.

Let Z_{j^*} be the first Datastructure that doesn't return bottom. This can be found in $\log M$ queries to the Z_i by using binary search.

Let p^* be the point returned by Z_{j^*}, then p^* is c^2-approximate nearest neighbor, because it might be the case, that Z_{j^*-1} has a point p' with $r \leq d(q,p') \leq c \cdot r$ which has the probability to be false positive, then in Z_{j^*} we can get at most c times the distance from Z_{j^*-1} which is $d(q,p') \leq c^2 \cdot r$.

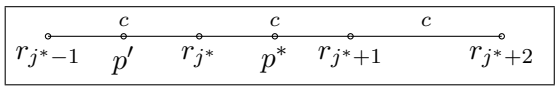

Figure 4: Approximation result of p^* in comparrison to p'.

Query Algorithm:

1. Query Z_0 : if return a result p^*, we have exact NN.
2. Query Z_M : if return $\perp$, any $p \in P$ is good enough.
3. Binary search yield c^2-approximate NN in $\log M$ queries

The big problem is that M depends on $\frac{d_{\max}}{d_{\min}}$ (the spread) which might be arbitrary large (with arbitrary precision in the representation).

$$M = \log \frac{d_{\max}}{d_{\min}} \Rightarrow \text{ query time } = \log \log \frac{d_{\max}}{d_{\min}}$$

2.6 approximate interval nearest neighbor search

Lemma 5 *Suppose we have a c-approximate near-neighbor DS with failure probability f', then we can do the following: Given a range $[a, b]$ and $\in (0, 1)$ and $P \subseteq \mathbb{R}^d$ we can build a DS for the following query: given $q \in \mathbb{R}^d$*

1. *if $d(q, P) \leq a$ this can be determined in 1 query + witness*
2. *if $d(q, P) \geq b$ this can be determined in 2 queries*
3. *if $d(q, p) \in [a, b]$, we can find a point $p^* \in^P$ with $d(q, p^*) \leq c \cdot (1 +) \cdot d(q, P)$ in $O(\log \frac{1}{} \log \frac{c \cdot b}{a})$ queries*

All this cases have the failure-probability is $f \leq O\left(\log\left(\frac{1}{\gamma} \log \frac{cb}{a}\right)\right) \cdot f'$ ⌟

Proof 5.
By construction of the previous algorithm. □

This is called the **approximate interval nearest neighbor search.**
Now we need to cluster P to obtain a sequence of reasonable interval NN-data-structures with a structure such as in figure 5 that groups $R \subseteq P$ where the points in R are "relativly" near to each other and "relativly" far away from every point not in R.

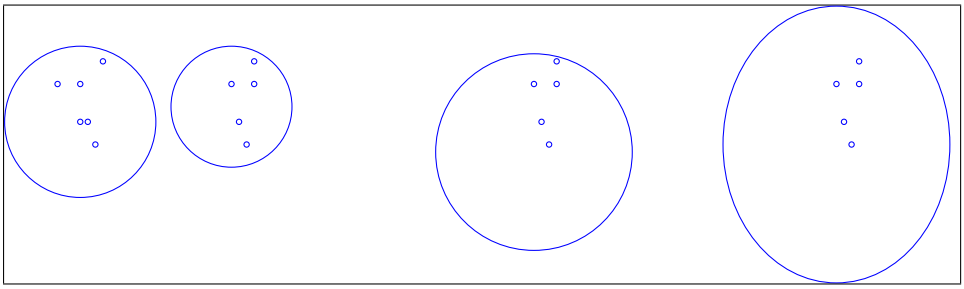

Figure 5: A possible clustering for a pointset

2.6.1 Generating Clusters

Next we will look at the generating of the clusters.

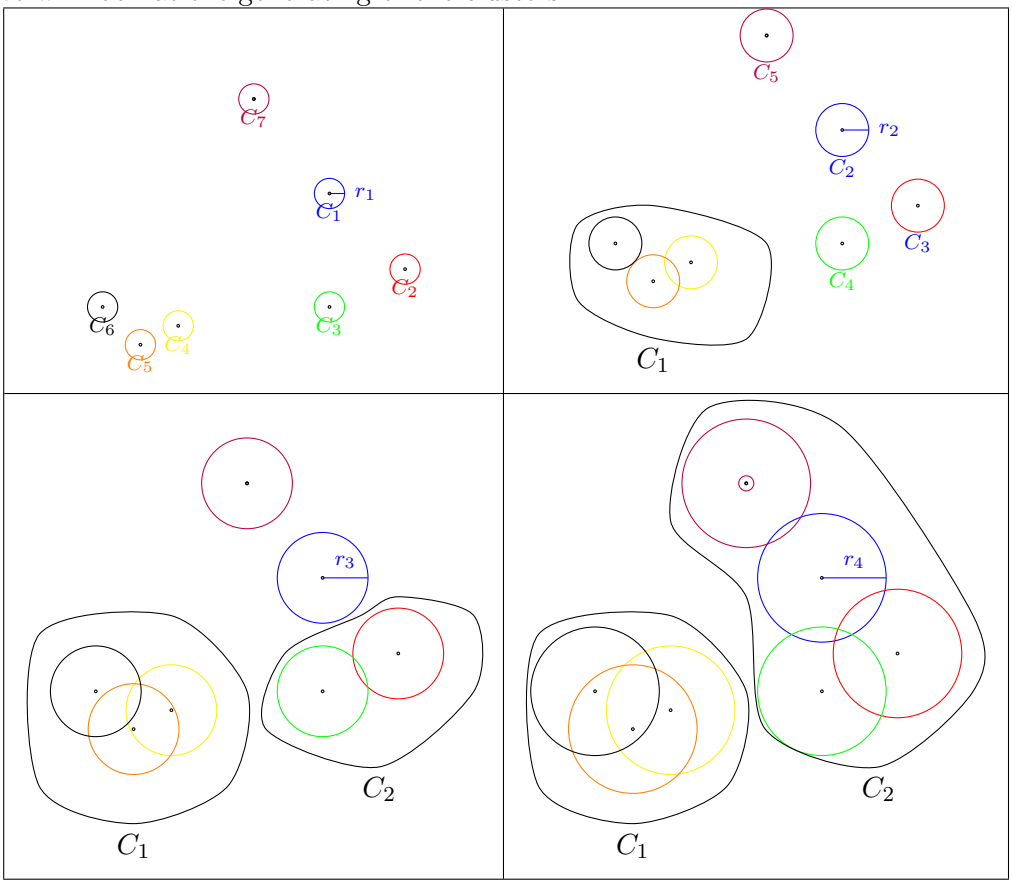

Figure 6: Clusters CC_P for radi r_1, r_2, r_3, r_4

Let the balls centered a the points in P grow.
For a fixed radius r let $CC_P(r)$ be the connected components of the set

$$\bigcup_{p \in P} B(q, r)$$

or equivalent the connected components of the graph $G(P, E \subset P^2)$ with

$$e = (p_1, p_2) \in E :\Leftrightarrow B(p_1, r) \cap B(p_2, r) \neq \emptyset$$

.
This is illustraded in figure 6.
Let $r_{\text{med}} = \min_r \{CC_p(r) \text{ has a component of size } \geq \frac{n}{2} + 1\}$.
Our clustering will (almost) be $CC_p(r_{\text{med}})$.