On ϵ-Nets and Simplex Range Queries

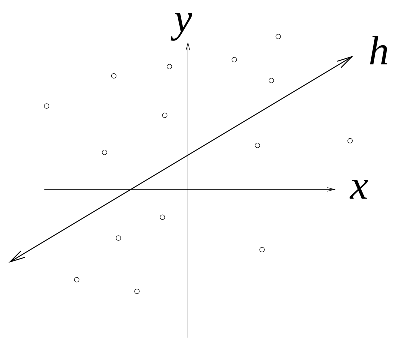
Mike Just

Half-Space Range Query

Given n points in E^2 and line h, determine

- (*counting problem*) the number of points
- (*reporting problem*) the points

in the half-space defined by h.



Generalizations include

- higher dimensions
- more complex regions, i.e. rectangles, triangles

Half-Space Range Query

Trivial Scheme

- determine which side of h, each of the n points lie
  - *storage* $O(n)$
  - *pre-computation* none
  - *query* $O(n)$

Our goal is to

- maintain linear storage
- achieve sub-linear query time

Build a *data structure* to optimize

- storage, pre-computation time, query time

[Haussler & Welzl 1987]

Results of paper are solutions for simplex range query in E^d with

- *storage* $O(n)$
- *pre-computation* $O(n \log n)$
- *query* $O(n^\alpha)$ where $\alpha = \frac{d(d-1)}{d(d-1)+1} + \gamma$, $\forall \gamma > 0$

Also of independent interest was the

- introduction of an ϵ-net
- and the use of *random sampling*

Relation to Other Work

dim.	Previous		Current
$d=2$	[Willard 1982]	$\alpha \approx 0.774$	
	[Edel., Welzl 1986]	$\alpha \approx 0.695$	$\alpha \approx 0.667$
$d=3$	[Yao 1983]	$\alpha \approx 0.936$	
	[Dob., Edel. 1984]	$\alpha \approx 0.916$	
	[Edel., Hub. 1984]	$\alpha \approx 0.909$	
	[Dob. et al. 1986]	$\alpha \approx 0.899$	$\alpha \approx 0.857$
$d=4$	[Cole 1985]	$\alpha \approx 0.977$	$\alpha \approx 0.923$
$d \geq 2$	[Yao & Yao 1985]	$\alpha = \frac{\log(2^d-1)}{d}$	$\alpha \approx \frac{d(d-1)}{d(d-1)+1}$

- best algorithm for solving simplex range queries (with linear storage) for $d \geq 2$

Data Structure and Algorithms

- Pre-computation is a *Las Vegas Algorithm*
- Query algorithm is deterministic but running time (i.e. γ) depends on the pre-computation
- *Data Structure – Partition Tree*
  - (*goal*) - partition plane into
    - a "small" number of regions
    - so that points are "evenly" distributed among the regions
  - *method* involves recursive partitioning

Partition Tree Construction (High-Level)

- set of points $A = \{p_1, p_2, \ldots, p_n\}$
- choose random subset $N \subseteq A$ where $|N| = v$
- form line arrangement for pairs of points in N
- create root of tree with a child for each cell in arrangement that has points in A
- if a cell contains $> v$ points, recursively partition this cell

Querying the Tree (High-Level)

Given a line h that partitions the plane, there are 3 types of cells,

1. those completely in the half-plane

2. those completely outside the half-plane

3. those intersected by the half-plane

Partition Tree Query involves

- count points from cells on the first level that are completely contained in the half-plane

- recurse on those cells intersected on h

Querying the Tree (cont'd)

Goals for Optimizing Query

1. points are distributed evenly among cells, i.e. "few" points in each cell (sort of)

2. small no. of cells intersected (bounded by $O(v^2)$)

Goals ultimately depend on choice of v.

$$v = \left\lfloor c\left(\frac{1}{\epsilon}\right)\left(\log\left(\frac{1}{\epsilon}\right) + \log\left(\frac{1}{\delta}\right)\right)\right\rfloor$$

v is "essentially" a constant

- As $\epsilon \rightarrow 0$, we achieve 1. with probability $1 - \delta$.

- γ is determined by ϵ

ϵ-Nets

The concept of an ϵ-net is useful for

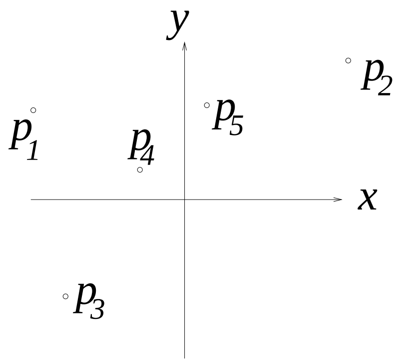
- allowing us to *choose* a “good” random sample
- and allowing us to efficiently *measure* when a good sample is obtained.

Definition

- Let R be the set of ranges determined by half-spaces in E^2.
- Let $A \subseteq E^2$ be a subset of points.
- $N \subseteq A$ is an $\epsilon -$ net of A for R if N contains a point in r for every $r \in R$ with

$$\frac{|A \cap r|}{|A|} > \epsilon$$

ϵ-Net Example



- $R = R_1 \cup R_2 \cup R_3$ where

$$
\begin{aligned}
R_1 &= \{\{p_1\}, \{p_2, p_3, p_4, p_5\}, \{p_2\}, \{p_1, p_3, p_4, p_5\}, \\
&\quad \{p_3\}, \{p_1, p_2, p_4, p_5\}\} \\
R_2 &= \{\{p_1, p_2\}, \{p_3, p_4, p_5\}, \{p_1, p_3\}, \{p_2, p_4, p_5\}, \\
&\quad \{p_2, p_3\}, \{p_1, p_4, p_5\}\} \\
R_3 &= \{\{p_1, p_2, p_5\}, \{p_3, p_4\}, \{p_1, p_3, p_4\}, \{p_2, p_5\}, \\
&\quad \{p_2, p_3, p_5\}, \{p_1, p_4\}\}
\end{aligned}
$$

ϵ-Net Example (cont'd)

For $A = \{p_1, p_2, p_3, p_4, p_5\}$, we have

- if $\epsilon = 0$ then smallest N is $\{p_1, p_2, p_3\}$
- if $\epsilon = \frac{1}{2}$ then smallest N is $\{p_3, p_5\}$

So,

- $|N|$ generally increases as ϵ decreases
- want to obtain (with high probability) the smallest such N for smallest ϵ
- can think of N as a *covering* or *sampling*

Existence of ϵ-Nets

For any $\epsilon > 0$, $A \subseteq E^d$, there exists an ϵ-net for A for half-spaces with at most

$$\left\lceil \frac{8(d+1)}{\epsilon} \log \frac{8(d+1)}{\epsilon} \right\rceil$$

points. In fact, if at least

$$\max\left(\frac{4}{\epsilon} \log \frac{2}{\delta}, \frac{8(d+1)}{\epsilon} \log \frac{8(d+1)}{\epsilon} \right)$$

points are independently and randomly drawn from A, these points form an ϵ-net for A with probability at least $1 - \delta$.

Partition Tree Structure

For each node p in T,

- $set(p)$ denotes the points in A that are in p (i.e. where the region of p intersects A)

Let $size(p) = |set(p)|$. If $size(p) > v$ then

- p is an internal node
- $arr(p)$ is an arrangement of $points(p) \subseteq set(p)$, where $|points(p)| = v$
- for each cell f in $arr(p)$, there is a child p_f of p

else

- p is a leaf

Partition Tree Analysis

Storage

- for each node $O(v^{d^2})$
- for entire tree $O(v^{d^2}n)$ or really $O(n)$ considering that v is dependent upon only d, ϵ and δ, i.e. not dependent upon $n = |A|$.

Important Property (i.e. think ϵ-nets) For every internal node p in T and *every* half-plane h

$$\sum_{f \in F} |f \cap set(p)| \leq \epsilon |set(p)|$$

where F is the set of cells f in $arr(p)$ with $f \cap h \neq \emptyset$.

In other words, the number of points in cells intersected by any line is small.

Partition Tree Construction

Set of points $A = \{p_1, p_2, \ldots, p_n\}$. If $|A| > v$, create internal node p (else create a leaf)

1. repeatedly select random sample $points(p) \subseteq A$ where $|points(p)| = v$ until the ϵ-net property is met

2. Form line arrangement for $H_d(points(p))$

3. recurse on the children of p

Can set $\delta = \frac{1}{2}$ so that 1. occurs with probability $\frac{1}{2}$.

Construction Cost

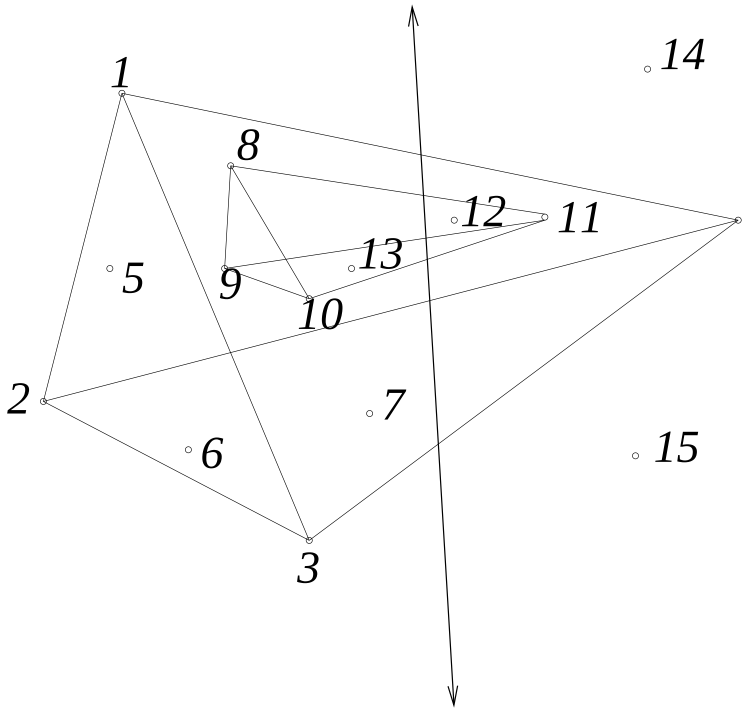
- Cost of computing $H_d(points(p))$ is constant since v is constant, and the number of cells depends on v.
- Verifying that the ϵ-net property is met takes constant time.
- If $\delta = \frac{1}{2}$, we expect to perform at most 2 verifications.
- Depth of tree is $O(\log n)$. (Note that the branching factor is bounded by the constant v.)
- At each level, there are at most n nodes.
- $O(n \log n)$ expected time to construct partition tree.

Queries

The count procedure is called as Count(r, h^*) where r is the root of the tree and h is the half-plane and h^* is the open half-space above h.

Count(p, h^*)
    result := result + $|points(p) \cap h^*|$;
    if p is an internal node **then**
        for each cell $f \in arr(p)$ for which p_f exists **do**
            if $f \subseteq h^*$ **then** result := result + size(p_f)
            else if f is intersected by h **then** Count(p_f, h^*)

Example



Query Analysis

Recall the goals for optimizing the query.

- no. of points in cells intersected by any line h is $< \epsilon l$ where l is the total no. of points at current level of recursion
- no. of cells intersected is small

By choosing an appropriate v, we obtain an ϵ-net which satisfies these properties.

Conclusion

- Easily extended to simplex range queries (both counting and reporting).
- Lots of hidden constants.
  - determining if a cell is intersected by a line
  - computing arrangements
  - as $\epsilon \to 0$, v becomes large (though indep of n)