Nearest Neighbors in Doubling Metrics

Algorithmic Problems Around the Web #7

Yury Lifshits
http://yury.name

CalTech, Fall'07, CS101.2, http://yury.name/algoweb.html

Making Nearest Neighbors Easier

Tractable solution: $\mathit{poly}(n)$ preprocessing, $\mathit{poly}\log(n)$ search time

General case of nearest neighbors seems to be intractable

Making Nearest Neighbors Easier

Tractable solution: $poly(n)$ preprocessing, $poly\log(n)$ search time

General case of nearest neighbors seems to be intractable

Any assumption that makes the problem easier?

Making Nearest Neighbors Easier

Tractable solution: $poly(n)$ preprocessing, $poly\log(n)$ search time

General case of nearest neighbors seems to be intractable

Any assumption that makes the problem easier?

Two approaches:

- Define intrinsic dimension of search domain and assume it is small (usually constant or $\mathcal{O}(\log\log n)$)
- Fix some probability distribution over inputs and queries. Find an algorithm which is fast with high probability over inputs/query

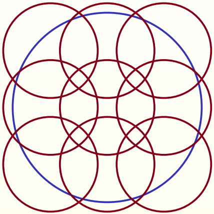
Nearest Neighbors in Small Doubling Dimension

Mini-plan:

Notion of doubling dimension
Solving 3-approximate nearest neighbors
From 3-approximation to $(1+\varepsilon)$-approximation

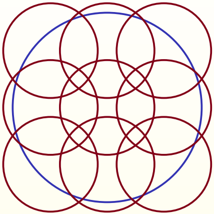
Notion of Doubling Dimension

Doubling constant λ for search domain $\mathbb{U}$: minimal value such that for every r and every object $p \in \mathbb{U}$ the ball $B(p, 2r)$ has cover of at most λ balls of radius r



Notion of Doubling Dimension

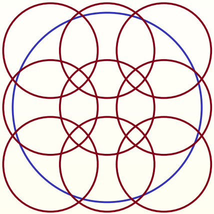
Doubling constant λ for search domain $\mathbb{U}$: minimal value such that for every r and every object $p \in \mathbb{U}$ the ball $B(p, 2r)$ has cover of at most λ balls of radius r



Doubling dimension: logarithm of doubling constant $\dim(\mathbb{U}) = \log \lambda$

Notion of Doubling Dimension

Doubling constant λ for search domain $\mathbb{U}$: minimal value such that for every r and every object $p \in \mathbb{U}$ the ball $B(p, 2r)$ has cover of at most λ balls of radius r

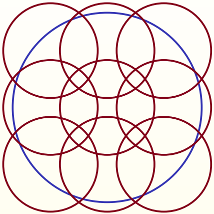


Doubling dimension: logarithm of doubling constant
$\dim(\mathbb{U}) = \log \lambda$

Exercise: Prove that for Euclidean space $\dim(\mathbb{R}^d) = \mathcal{O}(d)$

Notion of Doubling Dimension

Doubling constant λ for search domain $\mathbb{U}$: minimal value such that for every r and every object $p \in \mathbb{U}$ the ball $B(p, 2r)$ has cover of at most λ balls of radius r



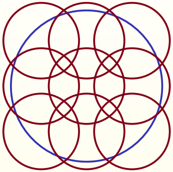
Doubling dimension: logarithm of doubling constant
$\dim(\mathbb{U}) = \log \lambda$

Exercise: Prove that for Euclidean space $\dim(\mathbb{R}^d) = \mathcal{O}(d)$

Exercise: Prove that $\forall S \subset \mathbb{U}: \quad \dim(S) \leq 2\dim(\mathbb{U})$

Doubling Dimension and r-Nets

Set $T \subset \mathbb{U}$ is an r-net for $S \subset \mathbb{U}$ iff
(1) $\forall p, p' \in T : \; d(p, p') > r$
(2) $\forall s \in S \;\; \exists p \in T : \; d(s, p) < r$

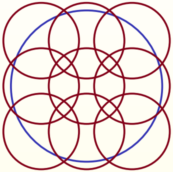


Doubling Dimension and r-Nets

Set $T \subset \mathbb{U}$ is an r-net for $S \subset \mathbb{U}$ iff

(1) $\forall p, p' \in T : \ d(p, p') > r$

(2) $\forall s \in S \ \ \exists p \in T : \ d(s, p) < r$



Lemma (Cover Lemma)

Every ball $B(p, r)$ has δr-net of cardinality at most $\left(\frac{1}{\delta}\right)^{\mathcal{O}(dim(\mathbb{U}))}$

Cover Lemma: Proof

Greedy algorithm:

1. Start from empty T
2. Find some object in S which is still δr-far from all objects in T, add it to T
3. Stop when all objects in S are within δr from some point in T

Cover Lemma: Proof

Greedy algorithm:

1. Start from empty T
2. Find some object in S which is still δr-far from all objects in T, add it to T
3. Stop when all objects in S are within δr from some point in T

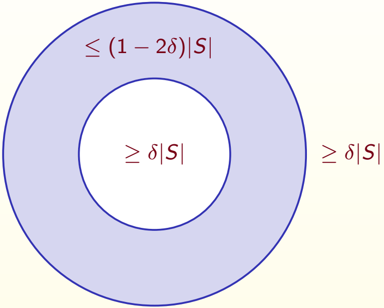
Upper bound on size:

- Apply definition of doubling constant to $B(p, r)$ recursively until getting $\frac{\delta r}{3}$-cover
- This cover has size $(\frac{1}{\delta})^{\mathcal{O}(\dim(\mathbb{U}))}$
- Every element of this cover can contain at most one object from T

Ring-Separator Lemma

Triple $(p, r, 2r)$ is
δ-**ring-separator**
for S iff

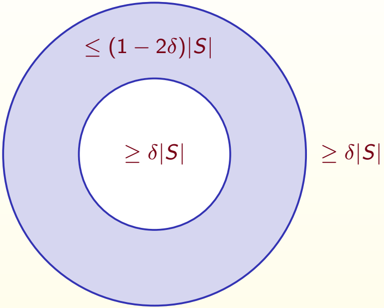
1. $|S \cap B(p, r)| \geq \delta |S|$
2. $|S / B(p, 2r)| \geq \delta |S|$



Ring-Separator Lemma

Triple $(p, r, 2r)$ is
δ-**ring-separator**
for S iff

1. $|S \cap B(p, r)| \geq \delta|S|$
2. $|S / B(p, 2r)| \geq \delta|S|$



Lemma (Ring-Separator Lemma)

For every S there is ring-separator with $\delta \geq \left(\frac{1}{2}\right)^{\mathcal{O}(dim(S))}$

Ring-Separator Lemma: Proof

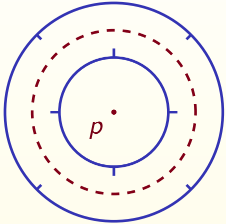
- Fix $\delta = (\frac{1}{2})^{c\dim(S)}$ for some large c
- For every p choose the maximal r_p such that $|B(p, r_p)| < \delta|S|$
- Let p_0 be the one having minimal r_{p_0}
- If none of triples $(p, r_p, 2r_p)$ is δ ring-separator build an r_{p_0}-net for $B(p_0, 2r_{p_0})$:
  - Start from r_0, and set $A := B(p_0, 2r_{p_0})/B(p_0, r_{p_0})$
  - Iteratively add some point p from A to net, update $A := A/B(p, r)$
- Since A decreased by at most $2\delta|S|$ points each time there must be many points in cover. Since it is r_{p_0}-net for $B(p_0, 2r_{p_0})$ there must be few points. Contradiction

Ring-Separator Tree

Krauthgamer&Lee'05

Preprocessing:

1. Find $(\frac{1}{2})^{\mathcal{O}(\dim(S))}$ ring-separator $(p, r, 2r)$ for S
2. Put objects from $B(p, 2r)$ to inner branch
3. Put objects from $S/B(p, r)$ to outer branch
4. Recursively repeat

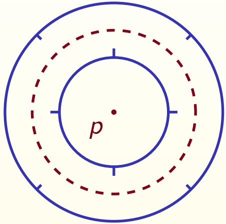


Ring-Separator Tree

Krauthgamer&Lee'05

Preprocessing:

1. Find $(\frac{1}{2})^{\mathcal{O}(\dim(S))}$ ring-separator $(p, r, 2r)$ for S
2. Put objects from $B(p, 2r)$ to inner branch
3. Put objects from $S/B(p, r)$ to outer branch
4. Recursively repeat



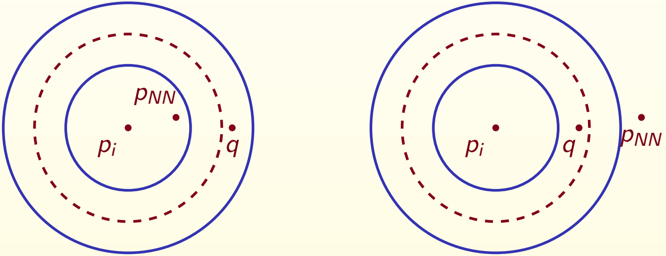
Search:

1. For every node $(p, r, 2r)$: if $d(q, p) \leq 3r/2$ go only to inner branch otherwise go only to outer branch
2. Return the best object considered in search

3-NN via Ring-Separator Tree

Notation: $p_1, \ldots, p_k$ are the centers of visited rings

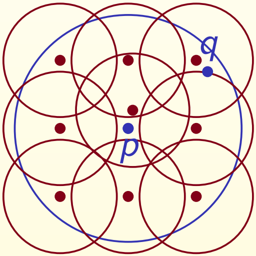
- If $p_{NN}(q) = p_k$ we are done
- If not, let us consider p_i where we miss the right branch. There are two cases:



- Anyway, p_i at most 3 time worse than $p_{NN}(q)$

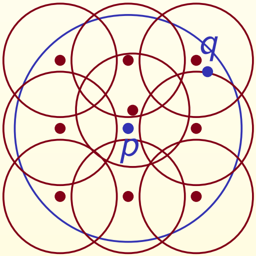
From 3-NN to r-NN: Reduction Algorithm

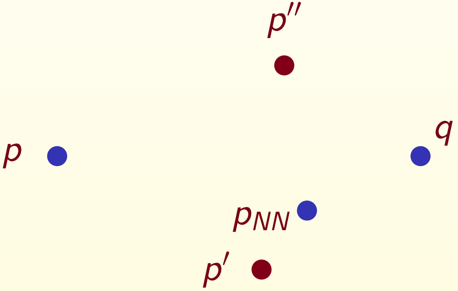
1. Find 3-approximate nearest neighbor p for q
2. Quickly build a $\varepsilon\frac{d(p,q)}{3}$ cover for $B(p, 4\frac{d(p,q)}{3})$. See the next slide
3. Return an object in cover that is the closest to q



From 3-NN to r-NN: Reduction Algorithm

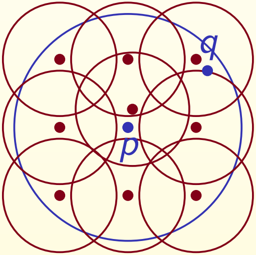
1. Find 3-approximate nearest neighbor p for q
2. Quickly build a $\varepsilon\frac{d(p,q)}{3}$ cover for $B(p, 4\frac{d(p,q)}{3})$. See the next slide
3. Return an object in cover that is the closest to q

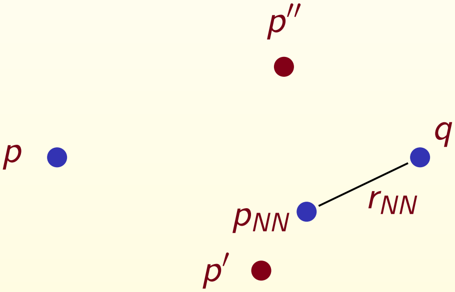




From 3-NN to r-NN: Reduction Algorithm

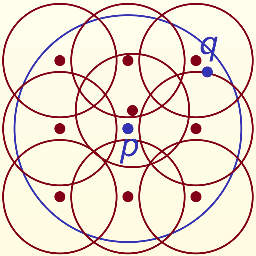
1. Find 3-approximate nearest neighbor p for q
2. Quickly build a $\varepsilon\frac{d(p,q)}{3}$ cover for $B(p, 4\frac{d(p,q)}{3})$. See the next slide
3. Return an object in cover that is the closest to q

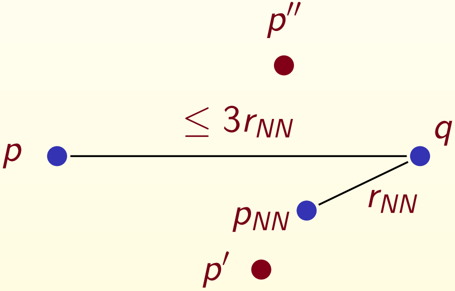




From 3-NN to r-NN: Reduction Algorithm

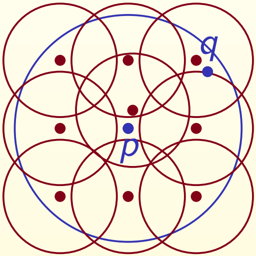
1. Find 3-approximate nearest neighbor p for q
2. Quickly build a $\varepsilon\frac{d(p,q)}{3}$ cover for $B(p, 4\frac{d(p,q)}{3})$. See the next slide
3. Return an object in cover that is the closest to q

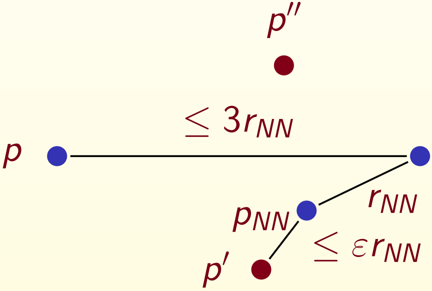




From 3-NN to r-NN: Reduction Algorithm

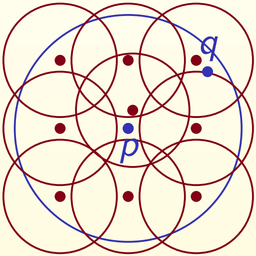
1. Find 3-approximate nearest neighbor p for q
2. Quickly build a $\varepsilon\frac{d(p,q)}{3}$ cover for $B(p, 4\frac{d(p,q)}{3})$. See the next slide
3. Return an object in cover that is the closest to q

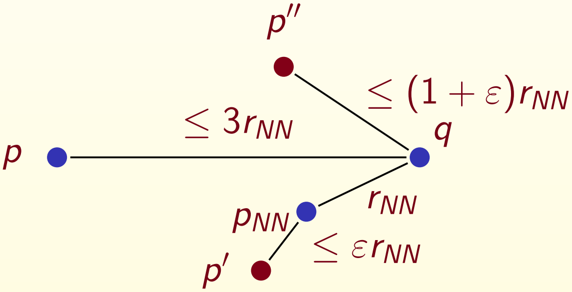




From 3-NN to r-NN: Reduction Algorithm

1. Find 3-approximate nearest neighbor p for q
2. Quickly build a $\varepsilon\frac{d(p,q)}{3}$ cover for $B(p, 4\frac{d(p,q)}{3})$. See the next slide
3. Return an object in cover that is the closest to q





From 3-NN to r-NN: Net Construction

Preprocessing:

1. For every i build 2^i-net for S (every lower level contains all points from the higher level)
2. Compute children pointers: from every element p of 2^i-net to all balls of 2^{i-1}-net required to cover $B(p, 2^i)$
3. Compute brother pointers: from every element p of 2^i-net to all elements p' from 2^i-net needed for covering $B(p, 2^i)$
4. Compute parent pointers: from every element p of 2^{i-1}-net to the element p' from 2^i-net within 2^i from it

From 3-NN to r-NN: Net Construction

Preprocessing:

1. For every i build 2^i-net for S (every lower level contains all points from the higher level)
2. Compute children pointers: from every element p of 2^i-net to all balls of 2^{i-1}-net required to cover $B(p, 2^i)$
3. Compute brother pointers: from every element p of 2^i-net to all elements p' from 2^i-net needed for covering $B(p, 2^i)$
4. Compute parent pointers: from every element p of 2^{i-1}-net to the element p' from 2^i-net within 2^i from it

On-line net construction:

1. Go up by parent pointers until meeting ball big enough
2. Use brother pointer
3. Go by children pointers until getting cover small enough

Other Definitions of Intrinsic Dimension

- Box dimension is the minimal d that for every r our domain $\mathbb{U}$ has r-net of size at most $(1/r)^{d+o(1)}$
- Karger-Ruhl dimension of database $S \subset \mathbb{U}$ is the minimal d that for every $p \in S$ and every r the following inequality holds: $|B(p, 2r) \cap S| \leq 2^d |B(p, r) \cap S|$
- Measure-based dimensions
- Disorder dimension (see next chapter)

Other Definitions of Intrinsic Dimension

- Box dimension is the minimal d that for every r our domain $\mathbb{U}$ has r-net of size at most $(1/r)^{d+o(1)}$
- Karger-Ruhl dimension of database $S \subset \mathbb{U}$ is the minimal d that for every $p \in S$ and every r the following inequality holds: $|B(p, 2r) \cap S| \leq 2^d |B(p, r) \cap S|$
- Measure-based dimensions
- Disorder dimension (see next chapter)

Exercise: prove that
$$\forall S \subset \mathbb{U} : \quad \dim_{\mathrm{Doub}}(S) \leq 4 \dim_{\mathrm{KR}}(S)$$

References

R. Krauthgamer and J.R. Lee
The black-box complexity of nearest-neighbor search Theoretical Computer Science, 2005
http://www.cs.berkeley.edu/~jrl/papers/nnc.pdf

R. Krauthgamer and J.R. Lee
Navigating nets: simple algorithms for proximity search SODA'04
http://www.cs.berkeley.edu/~robi/papers/KL-NavNets-SODA04.pdf

K.L. Clarkson
Nearest-Neighbor Searching and Metric Space Dimensions
In Nearest-Neighbor Methods for Learning and Vision: Theory and Practice, MIT Press, 2006
http://www.cs.bell-labs.com/who/clarkson/nn_survey/p.pdf