Approximate Nearest Neighbors

from the TCS perspective

Erik Waingarten (Penn)

What this talk is

- Worst-case analysis for nearest neighbor search
    - possible vs impossible

What this talk is

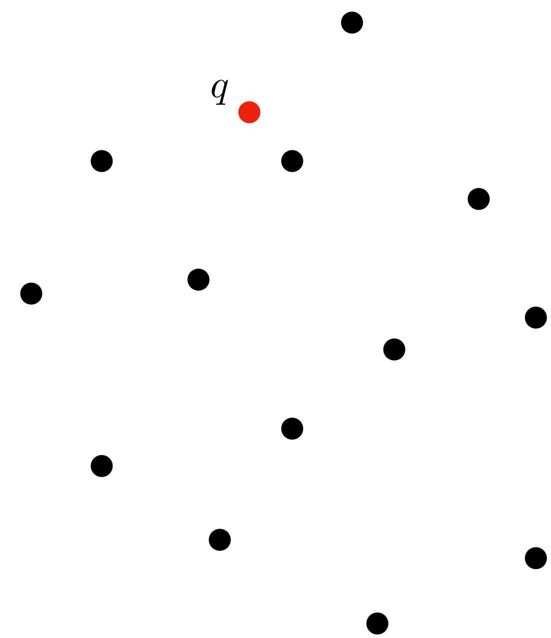
- Worst-case analysis for nearest neighbor search
  - possible vs impossible

Exact Algs	$(1+\epsilon)$-approx	c-approx
Curse of dimensionality Voronoi Diagram	Exponential-in dimension	LSH Data-dependent LSH

Nearest Neighbors
- a data structure problem

- Preprocessing:
    - dataset of points in a metric space

- Query:
    - new point, find closest dataset point

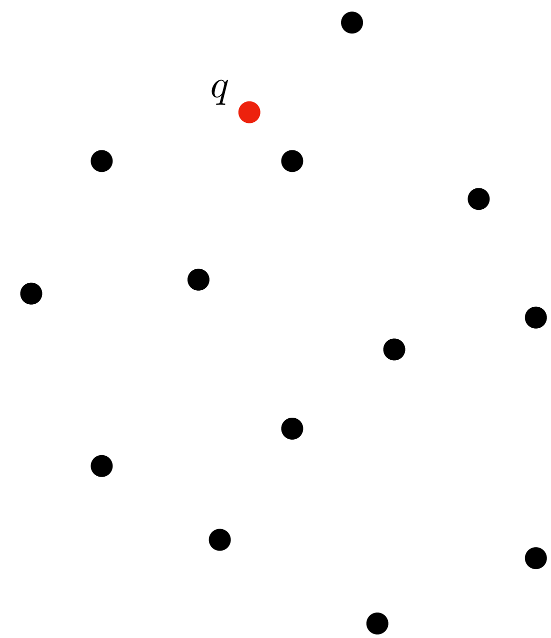


Nearest Neighbors

- a data structure problem has two trivial data structures

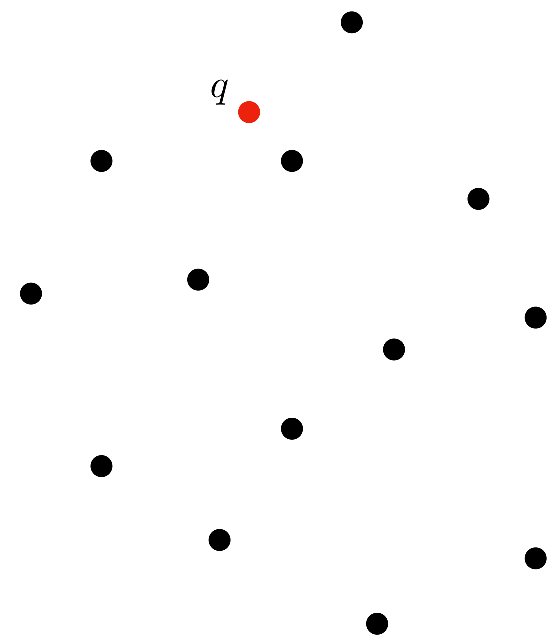
- Linear storage, linear scan.

- Prepare all answers, constant time.
  - for finite metric



Nearest Neighbors
- a data structure problem has two trivial data structures

- Linear storage, linear scan.



- Prepare all answers, constant time.
    - for finite metric

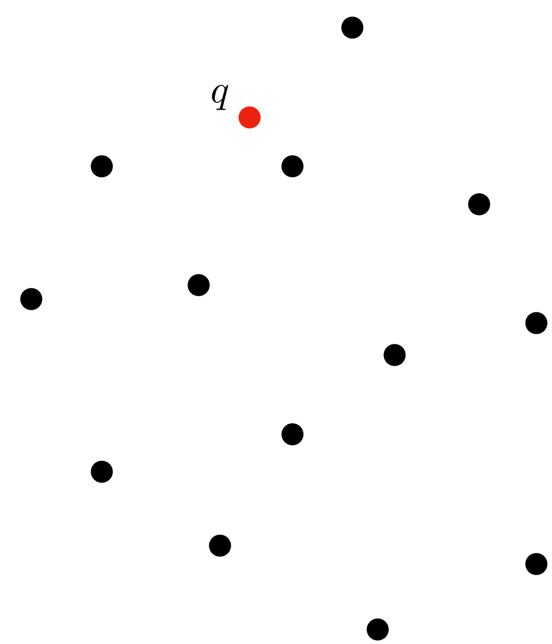
$\boxed{\text{Best of both worlds?}}$

Set Parameters and Priorities

- n: number of dataset points
- Metric space: $(\mathbb{R}^d, \ell_2)$ — generalize later

$$\omega(\log n) \leq d \leq n^{o(1)}$$

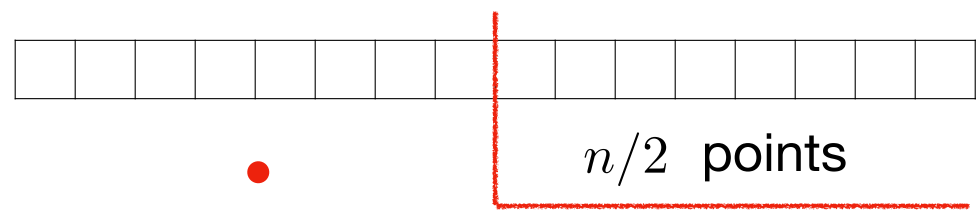
- Priorities:
  - Fast query time
  - Polynomial space
  - Preprocessing time



Nearest Neighbors in one dimension
- binary search

- Divide-and-Conquer
  - preprocessing time $O(n \log n)$
  - space $O(n)$
  - query time $O(\log n)$

Sorted dataset

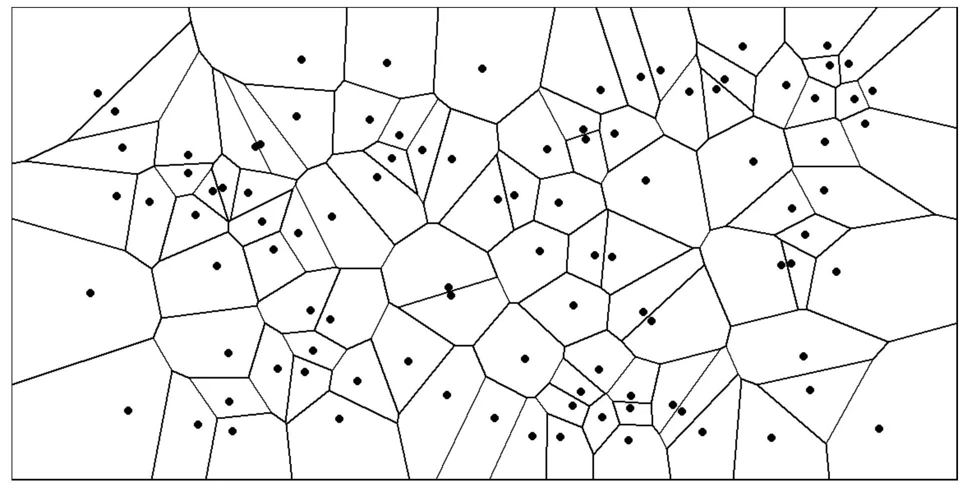


$$T(n) \leq T(n/2) + 1$$
$$= O(\log n)$$

Nearest Neighbors in higher dimension

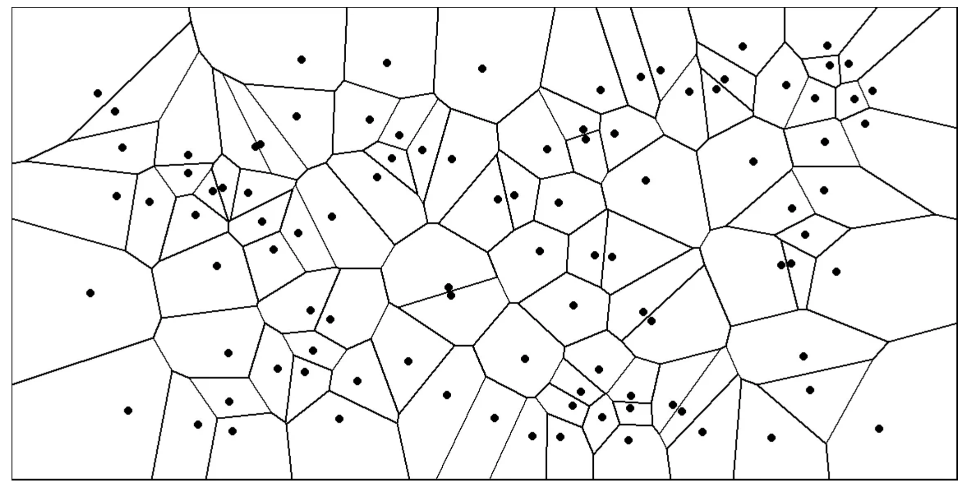
[Clarkson '88, Meiser '93]

- Query time: $O(d^5 \log n)$
- Space: $O(n^{d+\delta})$ for any $\delta > 0$
- Geometric complexity of Voronoi Diagram: $\Theta(n^{\lceil d/2 \rceil})$



Nearest Neighbors in higher dimension

[Clarkson '88, Meiser '93]



- Query time: $O(d^5 \log n)$
- Space: $O(n^{d+\delta})$ for any $\delta > 0$
- Geometric complexity of Voronoi Diagram: $\Theta(n^{\lceil d/2 \rceil})$

$\boxed{\text{Does there exist a poly-space sublinear query D.S?}}$

Nearest Neighbors in higher dimension

A strong negative answer

- Fine-grained complexity
  - [Williams ’05], [Ahle, Pagh, Razenshteyn, Silvestri ’16]

- **Theorem**: a data structure for nearest neighbor search with

  1. $\mathrm{poly}(nd)$ preprocessing time
  2. $\mathrm{poly}(d) \cdot n^{1-\epsilon}$ query time

  disproves SETH.

Approximate Nearest Neighbors

More space-efficient for $(1+\epsilon)$-approximations

- [Arya, Mount, Netanyahu, Silverman, Wu ’94], [Clarkson ’94], [Kleinberg ’97] [Kushilevitz, Ostrovsky, Rabani ’98, Indyk, Motwani ‘98]
- **Space:** $n^{\Theta(d)} \to n \cdot (d/\epsilon)^{O(d)} \to n \cdot (1/\epsilon)^{O(d)}$
- **Query Time:** $(1/\epsilon)^{O(d)} \log n \to \mathrm{poly}(d, \log n)$

Approximate Nearest Neighbors

More space-efficient for $(1+\epsilon)$-approximations

- [Arya, Mount, Netanyahu, Silverman, Wu ’94], [Clarkson ’94], [Kleinberg ’97], [Kushilevitz, Ostrovsky, Rabani ’98, Indyk, Motwani ‘98]
- **Space:** $n^{\Theta(d)} \to n \cdot (d/\epsilon)^{O(d)} \to n \cdot (1/\epsilon)^{O(d)} \to \ n \cdot n^{\tilde{O}(1/\epsilon^2)}$
- **Query Time:** $(1/\epsilon)^{O(d)} \log n \to \mathrm{poly}(d, \log n)$

Approximate Nearest Neighbors

More space-efficient for $(1+\epsilon)$-approximations

- [Arya, Mount, Netanyahu, Silverman, Wu ’94], [Clarkson ’94], [Kleinberg ’97], [Kushilevitz, Ostrovsky, Rabani ’98, Indyk, Motwani ‘98]
- **Space:** $n^{\Theta(d)} \to n \cdot (d/\epsilon)^{O(d)} \to n \cdot (1/\epsilon)^{O(d)} \to \; n \cdot n^{\tilde{O}(1/\epsilon^2)}$
- **Query Time:** $(1/\epsilon)^{O(d)} \log n \to \mathrm{poly}(d, \log n)$

Accurate approx., log-query time, large space

Approximate Nearest Neighbors

More space-efficient for $(1+\epsilon)$-approximations

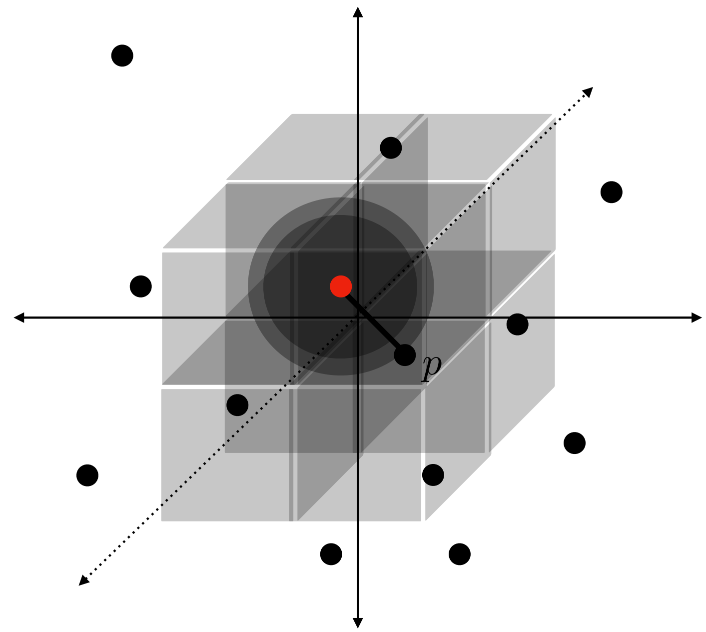
- [Arya, Mount, Netanyahu, Silverman, Wu ’94], [Clarkson ’94], [Kleinberg ’97], [Kushilevitz, Ostrovsky, Rabani ’98, Indyk, Motwani ‘98]
- **Space:** $n^{\Theta(d)} \to n \cdot (d/\epsilon)^{O(d)} \to n \cdot (1/\epsilon)^{O(d)} \to \; n \cdot n^{\tilde{O}(1/\epsilon^2)}$
- **Query Time:** $(1/\epsilon)^{O(d)} \log n \to \mathrm{poly}(d, \log n)$

Accurate approx., log-query time, large space

- **What makes dependence exponential?** [Clarkson ’99], [Karger, Ruhl ’02], [Krauthgamer, Lee ’04], … etc, survey [Clarkson ’05]

Approximate Nearest Neighbors

More space-efficient for $(1+\epsilon)$-approximations

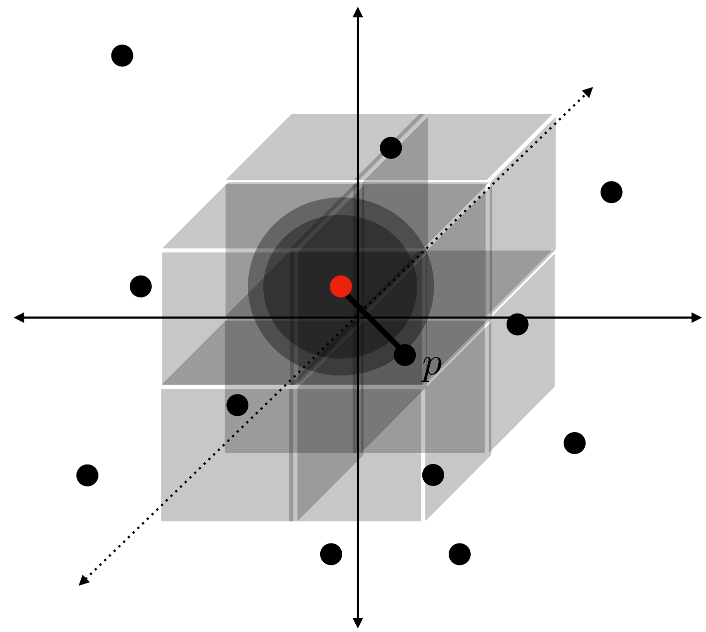


- Imagine sublinear algorithm
  - find point p
  - convince yourself no closer point - could intersect 2^d boxes

Approximate Nearest Neighbors

More space-efficient for $(1+\epsilon)$-approximations

- Imagine sublinear algorithm
  - find point p
  - convince yourself no closer point - could intersect 2^d boxes



- **Theorem** [Rubinstein ‘2018]: a data structure for $(1+\epsilon)$-approx nn with:
  - $\mathrm{poly}(nd)$ preprocessing time and $\mathrm{poly}(d) \cdot n^{1-\delta(\epsilon)}$ query time

disproves SETH.

Approximate Nearest Neighbors via LSH

Sublinear algorithms, better space

- “Randomized algorithms” perspective - [Indyk, Motwani ’98]
  - Large constant approx., $n^{0.1}$ -query time, $n^{1.1}$-space + preprocessing

Approximate Nearest Neighbors via LSH

Sublinear algorithms, better space

- "Randomized algorithms" perspective - [Indyk, Motwani '98]
  - Large constant approx., $n^{0.1}$ -query time, $n^{1.1}$-space + preprocessing

- **High Level Idea**:
  - Randomized hashing experiment, produces subset of dataset, return nn.
  - Why **correct**? Unlikely that significantly closer nn not in subset.
  - Why **fast**? Using approx. promise, limit the size of subset.

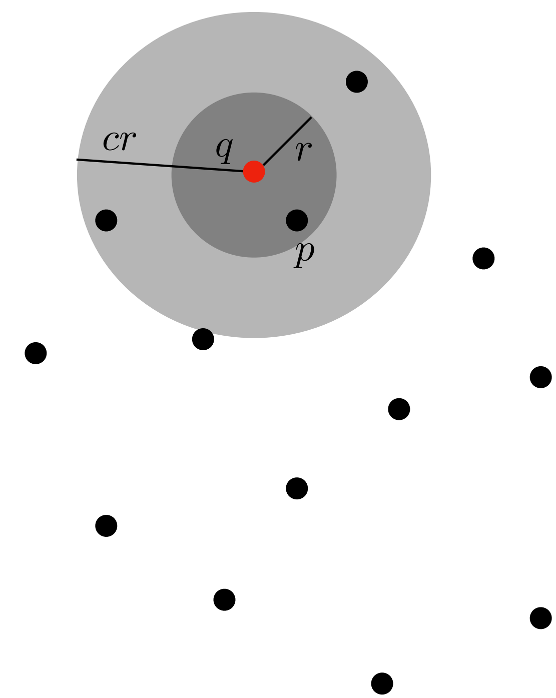
Approximate Near~~est~~ Neighbor

single-scale version of the problem

- Preprocess dataset with scale $r > 0$ and approx. $c > 1$.

- Query q.
  - Promise $\exists p : \|p - q\|_2 \leq r$,
  - $\Pr\left[\text{output } \|q - \hat{p}\|_2 \leq cr\right] \geq 1 - \delta$

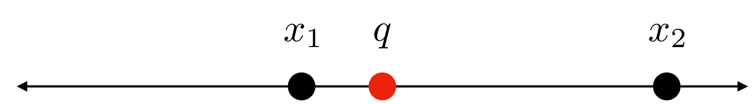
- As soon as find $\|q - \hat{p}\|_2 \leq cr$, return.



LSH: Randomized Space Partitions

what you need for randomized divide-and-conquer

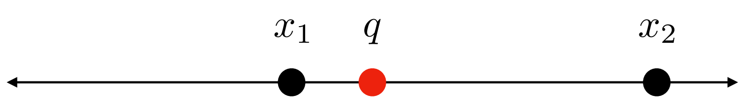
- Hash family $\mathcal{H}$ is (r, cr, p_1, p_2)-sensitive:
    - for any $\|x - q\|_2 \leq r$, $\quad \Pr_{\boldsymbol{h} \sim \mathcal{H}} [\boldsymbol{h}(x) = \boldsymbol{h}(q)] \geq p_1$
    - for any $\|x - q\|_2 \geq cr$, $\quad \Pr_{\boldsymbol{h} \sim \mathcal{H}} [\boldsymbol{h}(y) = \boldsymbol{h}(q)] \leq p_2$



LSH: Randomized Space Partitions

what you need for randomized divide-and-conquer

- Hash family $\mathcal{H}$ is (r, cr, p_1, p_2)-sensitive:
    - for any $\|x - q\|_2 \leq r$, $\Pr_{\boldsymbol{h}\sim\mathcal{H}}[\boldsymbol{h}(x) = \boldsymbol{h}(q)] \geq p_1$
    - for any $\|x - q\|_2 \geq cr$, $\Pr_{\boldsymbol{h}\sim\mathcal{H}}[\boldsymbol{h}(y) = \boldsymbol{h}(q)] \leq p_2$

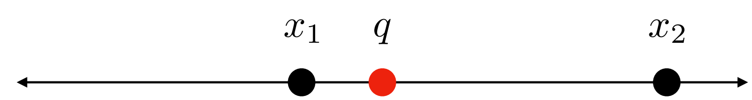


- Randomized Divide-and-Conquer:

LSH: Randomized Space Partitions

what you need for randomized divide-and-conquer

- Hash family $\mathcal{H}$ is (r, cr, p_1, p_2)-sensitive:
  - for any $\|x - q\|_2 \leq r$, $\Pr_{\boldsymbol{h} \sim \mathcal{H}} [\boldsymbol{h}(x) = \boldsymbol{h}(q)] \geq p_1$
  - for any $\|x - q\|_2 \geq cr$, $\Pr_{\boldsymbol{h} \sim \mathcal{H}} [\boldsymbol{h}(y) = \boldsymbol{h}(q)] \leq p_2$



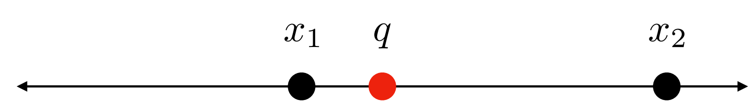
- Randomized Divide-and-Conquer:

$$T(n) \leq \frac{1}{p_1} \cdot (1 + T(n \cdot p_2))$$

LSH: Randomized Space Partitions

what you need for randomized divide-and-conquer

- Hash family $\mathcal{H}$ is (r, cr, p_1, p_2)-sensitive:
  - for any $\|x - q\|_2 \le r$, $\Pr_{\boldsymbol{h} \sim \mathcal{H}} [\boldsymbol{h}(x) = \boldsymbol{h}(q)] \ge p_1$
  - for any $\|x - q\|_2 \ge cr$, $\Pr_{\boldsymbol{h} \sim \mathcal{H}} [\boldsymbol{h}(y) = \boldsymbol{h}(q)] \le p_2$



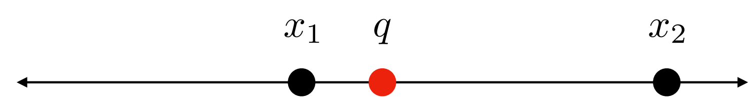
- Randomized Divide-and-Conquer:

$$T(n) \le \frac{1}{p_1} \cdot (1 + T(n \cdot p_2)) = \left(\frac{1}{p_1}\right)^{\log_{1/p_2} n}$$

LSH: Randomized Space Partitions

what you need for randomized divide-and-conquer

- Hash family $\mathcal{H}$ is (r, cr, p_1, p_2)-sensitive:
  - for any $\|x - q\|_2 \leq r$, $\quad \Pr_{\boldsymbol{h}\sim\mathcal{H}}[\boldsymbol{h}(x) = \boldsymbol{h}(q)] \geq p_1$
  - for any $\|x - q\|_2 \geq cr$, $\quad \Pr_{\boldsymbol{h}\sim\mathcal{H}}[\boldsymbol{h}(y) = \boldsymbol{h}(q)] \leq p_2$



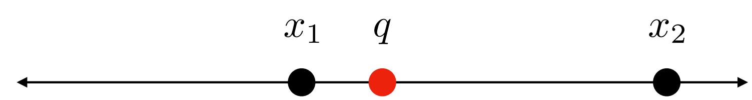
- Randomized Divide-and-Conquer:

$$T(n) \leq \frac{1}{p_1} \cdot (1 + T(n \cdot p_2)) = \left(\frac{1}{p_1}\right)^{\log_{1/p_2} n} = n^{\log(1/p_1)/\log(1/p_2)} = n^{\rho}.$$

LSH: Randomized Space Partitions

what you need for randomized divide-and-conquer

- Hash family $\mathcal{H}$ is (r, cr, p_1, p_2)-sensitive:
  - for any $\|x - q\|_2 \le r$, $\Pr_{\boldsymbol{h}\sim\mathcal{H}}[\boldsymbol{h}(x) = \boldsymbol{h}(q)] \ge p_1$
  - for any $\|x - q\|_2 \ge cr$, $\Pr_{\boldsymbol{h}\sim\mathcal{H}}[\boldsymbol{h}(y) = \boldsymbol{h}(q)] \le p_2$



- Randomized Divide-and-Conquer:

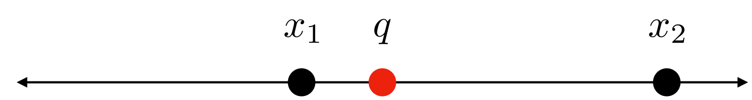
$$T(n) \le \frac{1}{p_1} \cdot (1 + T(n \cdot p_2)) = \left(\frac{1}{p_1}\right)^{\log_{1/p_2} n} = n^{\log(1/p_1)/\log(1/p_2)} = n^{\rho}.$$

$$\rho = \frac{\log(1/p_1)}{\log(1/p_2)} < 1$$

LSH: Randomized Space Partitions

what you need for randomized divide-and-conquer

- Hash family $\mathcal{H}$ is (r, cr, p_1, p_2)-sensitive:
  - for any $\|x - q\|_2 \le r$, $\Pr_{\boldsymbol{h} \sim \mathcal{H}}[\boldsymbol{h}(x) = \boldsymbol{h}(q)] \ge p_1$
  - for any $\|x - q\|_2 \ge cr$, $\Pr_{\boldsymbol{h} \sim \mathcal{H}}[\boldsymbol{h}(y) = \boldsymbol{h}(q)] \le p_2$



- Randomized Divide-and-Conquer:

$$T(n) \le \frac{1}{p_1} \cdot (1 + T(n \cdot p_2)) = \left(\frac{1}{p_1}\right)^{\log_{1/p_2} n} = n^{\log(1/p_1)/\log(1/p_2)} = n^{\rho}.$$

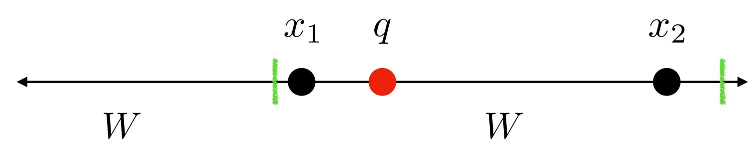
$$S(n) \le n \cdot \left(\frac{1}{p_1}\right)^{\log_{1/p_2} n} = n^{1+\rho}$$

$$\rho = \frac{\log(1/p_1)}{\log(1/p_2)} < 1$$

LSH: Randomized Space Partitions

what you need for randomized divide-and-conquer

- Hash family $\mathcal{H}$ is (r, cr, p_1, p_2)-sensitive:
  - for any $\|x - q\|_2 \le r$, $\Pr_{\boldsymbol{h}\sim\mathcal{H}}[\boldsymbol{h}(x) = \boldsymbol{h}(q)] \ge p_1$
  - for any $\|x - q\|_2 \ge cr$, $\Pr_{\boldsymbol{h}\sim\mathcal{H}}[\boldsymbol{h}(y) = \boldsymbol{h}(q)] \le p_2$



- Ex. $(\mathbb{R}^1, \ell_1)$: random shifted interval of width W

$$\Pr_{\boldsymbol{h}\sim\mathcal{H}}[\boldsymbol{h}(x) = \boldsymbol{h}(q)] = \min\left\{1 - \frac{|x-q|}{W}, 0\right\}$$

$$\frac{\log(1/p_1)}{\log(1/p_2)} = \frac{\log(1 - r/W)}{\log(1 - cr/W)} \approx \frac{-r/W}{-cr/W} = \frac{1}{c}$$

LSH: Randomized Space Partitions

what you need for randomized divide-and-conquer

- Hash family $\mathcal{H}$ is (r, cr, p_1, p_2)-sensitive:
  - for any $\|x-q\|_2 \leq r$, $\Pr_{\boldsymbol{h}\sim\mathcal{H}}[\boldsymbol{h}(x)=\boldsymbol{h}(q)] \geq p_1$
  - for any $\|x-q\|_2 \geq cr$, $\Pr_{\boldsymbol{h}\sim\mathcal{H}}[\boldsymbol{h}(y)=\boldsymbol{h}(q)] \leq p_2$

Quantity to optimize

$$\boxed{\rho = \frac{\log(1/p_1)}{\log(1/p_2)}}$$

- Upper bounds:
  - [Indyk-Motwani '98]: $\rho = 1/c$ for all $(\mathbb{R}^d, \ell_p), p \in [1,2]$
  - [Datar-Immorlica-Indyk-Mirrokni '04]: $\rho < 1/c$ for $(\mathbb{R}^d, \ell_2)$
  - [Andoni-Indyk '06]: $\rho = 1/c^p + o_d(1)$ for $(\mathbb{R}^d, \ell_p), p \in [1,2]$

LSH: Randomized Space Partitions

what you need for randomized divide-and-conquer

Quantity to optimize

$$\boxed{\rho = \frac{\log(1/p_1)}{\log(1/p_2)}}$$

- Lower bounds:
  - [Motwani-Naor-Panigrahy '06]: $\rho \geq \frac{e^{1/c^p} - 1}{e^{1/c^p} + 1} \to \frac{1}{2c^p}$
  - [O'Donnell-Wu-Zhou '09]: $\rho \geq \frac{1}{c^p} - o_d(1)$
- Upper bounds:
  - [Indyk-Motwani '98]: $\rho = 1/c$ for all $(\mathbb{R}^d, \ell_p), p \in [1,2]$
  - [Datar-Immorlica-Indyk-Mirrokni '04]: $\rho < 1/c$ for $(\mathbb{R}^d, \ell_2)$
  - [Andoni-Indyk '06]: $\rho = 1/c^p + o_d(1)$ for $(\mathbb{R}^d, \ell_p), p \in [1,2]$

LSH: Randomized Space Partitions

what you need for randomized divide-and-conquer

- Lower bounds:
  - [Motwani-Naor-Panigrahy '06]: $\rho \geq \frac{e^{1/c^p} - 1}{e^{1/c^p} + 1} \to \frac{1}{2c^p}$
  - [O'Donnell-Wu-Zhou '09]: $\rho \geq \frac{1}{c^p} - o_d(1)$

Quantity to optimize

$$\boxed{\rho = \frac{\log(1/p_1)}{\log(1/p_2)}}$$

- Upper bounds:
  - [Indyk-Motwani '98]: $\rho = 1/c$ for all $(\mathbb{R}^d, \ell_p), p \in [1, 2]$
  - [Datar-Immorlica-Indyk-Mirrokni '04]: $\rho < 1/c$ for $(\mathbb{R}^d, \ell_2)$
  - [Andoni-Indyk '06]: $\rho = 1/c^p + o_d(1)$ for $(\mathbb{R}^d, \ell_p), p \in [1, 2]$

Ex.: $c = 10$

Query time: $n^{0.01}$

Space: $n^{1+0.01}$

LSH: Randomized Space Partitions

next steps and subsequent questions

- Las Vegas LSH Algorithms:
    - Guarantee correctness, running time in expectation.
    - [Pagh ’16], [Ahle ’17], [Wei ‘19]
- Time-space tradeoffs:
    - Use more space, faster query time
    - [Kapralov ’15], [Becker-Ducas-Gama-Laarhoven ’16], [Christiani ’17], [Andoni-Razenshteyn-Laarhoven-Waingarten ’17]
- Practical LSH Algorithms: [Andoni-Indyk-Laarhoven-Razenshteyn-Schmidt ’15, Aumuller, Christiani, Pagh, Vesterli ’19]

The Landscape so Far…

Exact, Accurate-approx, Constant-approx

Exact Algs	$(1+\epsilon)$-approx	c-approx
$n^{\Theta(d)}$ space, $d^{O(1)} \log n$ time or, linear scan	$n \cdot (1/\epsilon)^{O(d)}$ space, $\mathrm{poly}(d, \log n)$ time	$dn^{1+\rho}$ space, dn^{ρ} time $\rho \to 0$ as $c \to \infty$

Beyond LSH?

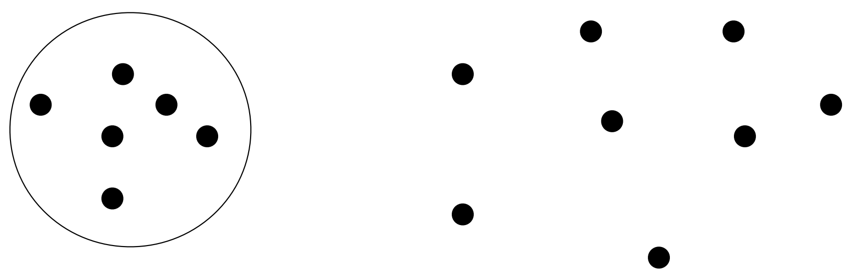
or is that all there is.

- LSH Recursion: $T(n) \leq \frac{1}{p_1} \cdot (1 + T(p_2 n))$
  - possible tradeoffs between p_1 vs. p_2 run into lbs.
  - [O'Donnell-Wu-Zhou '09] instances are very structured (contain dense region)

Beyond LSH?

or is that all there is.

- LSH Recursion: $T(n) \leq \frac{1}{p_1} \cdot (1 + T(p_2 n))$
  - possible tradeoffs between p_1 vs. p_2 run into lbs.
  - [O'Donnell-Wu-Zhou '09] instances are very structured (contain dense region)
- **Idea**: During preprocessing, identify special structure + adapt to it!



Beyond LSH?

or is that all there is.

- LSH Recursion: $T(n) \leq \dfrac{1}{p_1} \cdot (1 + T(p_2 n))$
  - possible tradeoffs between p_1 vs. p_2 run into lbs.
  - [O'Donnell-Wu-Zhou '09] instances are very structured (contain dense region)

- [Andoni-Indyk-Nguyen-Razenshteyn '14]: $\rho \leq 7/(8c^2)$
- [Andoni-Razenshteyn '15]: $\rho \leq 1/(2c^2 - 1)$

Beyond LSH?

or is that all there is.

- LSH Recursion: $T(n) \leq \frac{1}{p_1} \cdot (1 + T(p_2 n))$
  - possible tradeoffs between p_1 vs. p_2 run into lbs.
  - [O'Donnell-Wu-Zhou '09] instances are very structured (contain dense region)

- [Andoni-Indyk-Nguyen-Razenshteyn '14]: $\rho \leq 7/(8c^2)$
- [Andoni-Razenshteyn '15]: $\rho \leq 1/(2c^2 - 1)$

$$\frac{1}{c^2} \to \frac{1}{2c^2 - 1}$$

Beyond LSH?

or is that all there is.

- LSH Recursion: $T(n) \leq \frac{1}{p_1} \cdot (1 + T(p_2 n))$
  - possible tradeoffs between p_1 vs. p_2 run into lbs.
  - [O'Donnell-Wu-Zhou '09] instances are very structured (contain dense region)

- [Andoni-Indyk-Nguyen-Razenshteyn '14]: $\rho \leq 7/(8c^2)$
- [Andoni-Razenshteyn '15]: $\rho \leq 1/(2c^2 - 1)$

$$\frac{1}{c^2} \to \frac{1}{2c^2 - 1}$$

- For $d = \omega(\log n)$, $\rho \geq \frac{1}{2c^2 - 1}$ for unstructured instance.

The Twist on LSH

what can one ‘adapt’ to.

- LSH Recursion: $T(n) \leq \frac{1}{p_1} \cdot (1 + T(p_2 n))$

The Twist on LSH

what can one 'adapt' to.

size of recursive dataset

- LSH Recursion: $T(n) \leq \frac{1}{p_1} \cdot (1 + T(p_2 n))$

amplifying success prb
for *unknown* query.

The Twist on LSH

what can one 'adapt' to.

size of recursive dataset

- LSH Recursion: $T(n) \le \frac{1}{p_1} \cdot (1 + T(p_2 n))$

amplifying success prb
for *unknown* query.

- **Data-Dependent LSH:**
    - For any $x \in P, q \in \mathbb{R}^d \text{ s.t } \|x - q\|_2 \le r, \quad \Pr_{\boldsymbol{h} \sim \mathcal{H}(P)} [\boldsymbol{h}(x) = \boldsymbol{h}(q)] \ge p_1$
    - $\mathbb{E}_{\boldsymbol{x}_1, \boldsymbol{x}_2 \sim P} \left[\Pr_{\boldsymbol{h} \sim \mathcal{H}(P)} [\boldsymbol{h}(\boldsymbol{x}_2) = \boldsymbol{h}(\boldsymbol{x}_1)] \right] \le p_2$

Finding Structure in Arbitrary Datasets

the “data-dependent” approach

Finding Structure in Arbitrary Datasets

the “data-dependent” approach

- **Dense Ball-or-LSH**. Up to factor-2, easiest structure to handle is a *dense* ball.

Finding Structure in Arbitrary Datasets

the “data-dependent” approach

- **<u>Dense Ball-or-LSH</u>**. Up to factor-2, easiest structure to handle is a *dense* ball.
  - Thm [Andoni-Naor-Nikolov-Razenshteyn-Waingarten ’18]: For all $P \subset (\mathbb{R}^d, \ell_p)$,
    - Either there is an ℓ_p-ball of radius $O(p/\epsilon) \cdot r$ with half of the points.
    - Or, there is an LSH for approx. $O(p/\epsilon)$ with $\log(1/p_1)/\log(1/p_2) \leq \epsilon$.

Finding Structure in Arbitrary Datasets

the “data-dependent” approach

- **<u>Dense Ball-or-LSH</u>**. Up to factor-2, easiest structure to handle is a *dense* ball.
    - Thm [Andoni-Naor-Nikolov-Razenshteyn-Waingarten ’18]: For all $P \subset (\mathbb{R}^d, \overset{\|\cdot\|}{\cancel{\ell_p}})$,
        - Either there is an $\overset{\|\cdot\|}{\cancel{\ell_p}}$-ball of radius $\overset{O(\log d/\epsilon^2)}{\cancel{O(p/\epsilon)}} \cdot r$ with half of the points.
        - Or, there is an LSH for approx. $\underset{O(\log d/\epsilon^2)}{\cancel{O(p/\epsilon)}}$ with $\log(1/p_1)/\log(1/p_2) \le \epsilon$.

Finding Structure in Arbitrary Datasets

the "data-dependent" approach

- **<u>Dense Ball-or-LSH</u>**. Up to factor-2, easiest structure to handle is a *dense* ball.
  - Thm [Andoni-Naor-Nikolov-Razenshteyn-Waingarten '18]: For all $P \subset (\mathbb{R}^d, \overset{\|\cdot\|}{\cancel{\ell_p}})$,
    - Either there is an $\overset{\|\cdot\|}{\cancel{\ell_p}}$-ball of radius $\overset{O(\log d/\epsilon^2)}{\cancel{O(p/\epsilon)}} \cdot r$ with half of the points.
    - Or, there is an LSH for approx. $\underset{O(\log d/\epsilon^2)}{\cancel{O(p/\epsilon)}}$ with $\log(1/p_1)/\log(1/p_2) \leq \epsilon$.

- No Dense Ball $\longrightarrow$ Sparse cuts of potential query-nn pairs.

Finding Structure in Arbitrary Datasets

the “data-dependent” approach

- **<u>Dense Ball-or-LSH</u>**. Up to factor-2, easiest structure to handle is a *dense* ball.
  - Thm [Andoni-Naor-Nikolov-Razenshteyn-Waingarten ’18]: For all $P \subset (\mathbb{R}^d, \overset{\|\cdot\|}{\cancel{\ell_p}})$,
    - Either there is an $\overset{\|\cdot\|}{\cancel{\ell_p}}$-ball of radius $\overset{O(\log d/\epsilon^2)}{\cancel{O(p/\epsilon)}} \cdot r$ with half of the points.
    - Or, there is an LSH for approx. $\underset{\cancel{O(\log d/\epsilon^2)}\quad 2^{\tilde{O}(\sqrt{\log d}\cdot\log(1/\epsilon))}}{\cancel{O(p/\epsilon)}}$ with $\log(1/p_1)/\log(1/p_2) \le \epsilon$.

- No Dense Ball $\longrightarrow$ Sparse cuts of potential query-nn pairs.

  ^ $\mathrm{poly}(d)$-time oracle

Finding Structure in Arbitrary Datasets

the “data-dependent” approach

- **<u>Dense Ball-or-LSH</u>**. Up to factor-2, easiest structure to handle is a *dense* ball.
  - Thm [Andoni-Naor-Nikolov-Razenshteyn-Waingarten ’18]: For all $P \subset (\mathbb{R}^d, \overset{\|\cdot\|}{\cancel{\ell_p}})$,
    - Either there is an $\overset{\|\cdot\|}{\cancel{\ell_p}}$-ball of radius $\overset{O(\log d/\epsilon^2)}{\cancel{O(p/\epsilon)}} \cdot r$ with half of the points.
    - Or, there is an LSH for approx. $\underset{\cancel{O(\log d/\epsilon^2)} \quad 2^{\tilde{O}(\sqrt{\log d} \cdot \log(1/\epsilon))}}{\cancel{O(p/\epsilon)}}$ with $\log(1/p_1)/\log(1/p_2) \leq \epsilon$.

- No Dense Ball ⟶ Sparse cuts of potential query-nn pairs.

  ^ $\mathrm{poly}(d)$-time oracle

- [Kush-Nikolov-Tang ’21]: efficient preprocessing for $(\mathbb{R}^d, \ell_p)$ via average-distortion

Finding Structure in Arbitrary Datasets

the “data-dependent” approach

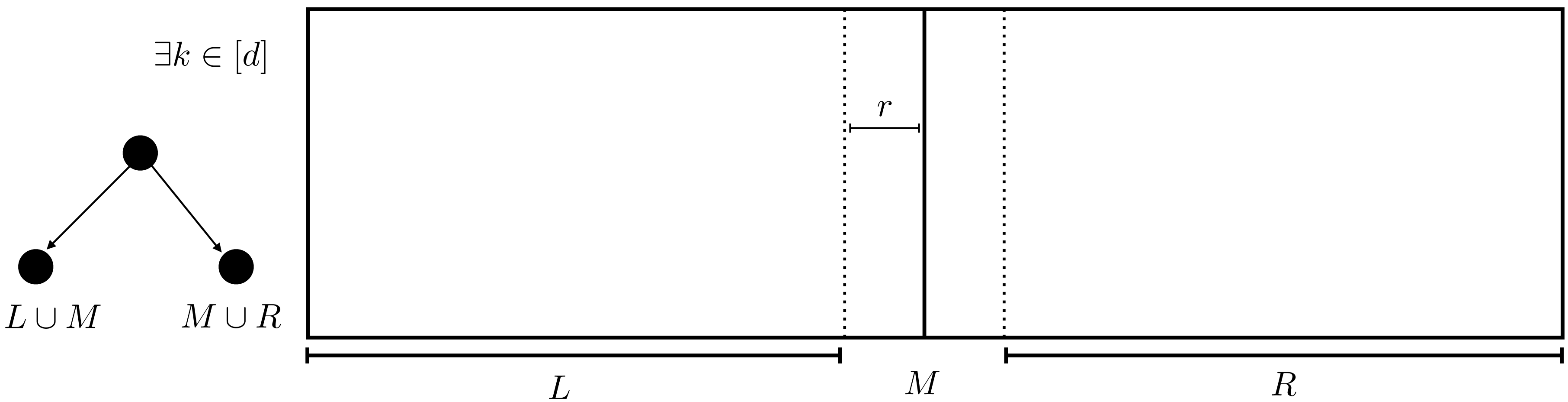
- **<u>Dense Ball-or-Decompose</u>**. [Indyk ’00]
- Thm: For any $P \subset (\mathbb{R}^d, \ell_\infty)$, either there is an ℓ_∞-ball of radius $O(\log\log d)/\epsilon \cdot r$, or

Finding Structure in Arbitrary Datasets

the "data-dependent" approach

- **<u>Dense Ball-or-Decompose</u>**. [Indyk '00]
- Thm: For any $P \subset (\mathbb{R}^d, \ell_\infty)$, either there is an ℓ_∞-ball of radius $O(\log \log d)/\epsilon \cdot r$, or

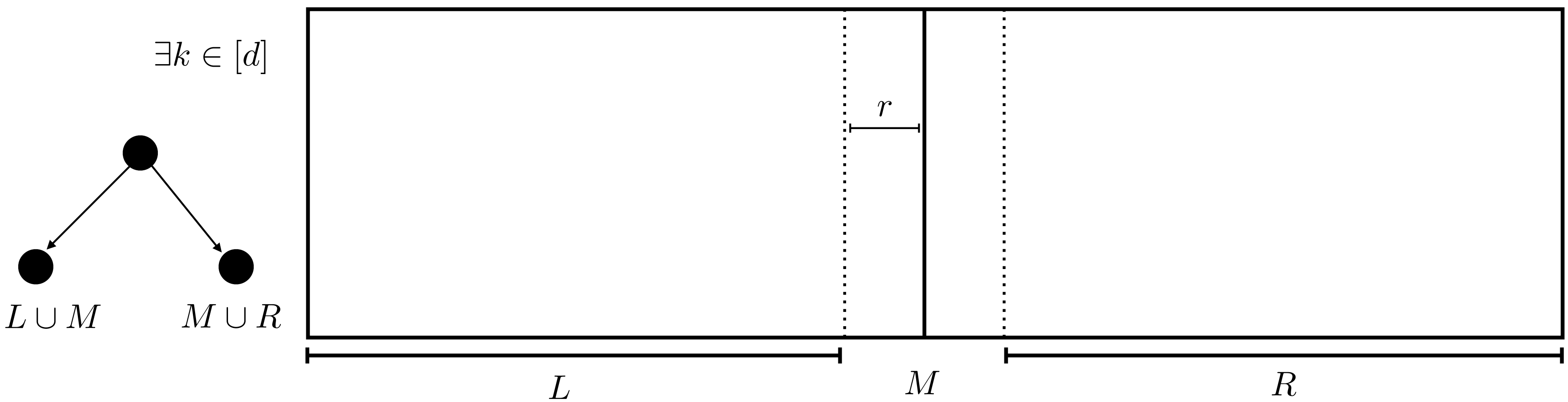
$\exists k \in [d]$



Finding Structure in Arbitrary Datasets

the "data-dependent" approach

- **<u>Dense Ball-or-Decompose</u>**. [Indyk '00]
- Thm: For any $P \subset (\mathbb{R}^d, \ell_\infty)$, either there is an ℓ_∞-ball of radius $O(\log \log d)/\epsilon \cdot r$, or



Query time: $T(n) \leq 1 + T((1 - 1/d)n)$

Space: $S(n) \leq S(L \cup M) + S(M \cup R) \leq n^{1+\epsilon}$

Finding Structure in Arbitrary Datasets

the “data-dependent” approach

- **<u>Dense Projected-Ball or Decompose</u>.** [Andoni-Nikolov-Razenshteyn-Waingarten ’21]
- Thm: For any $P \subset (\mathbb{R}^d, \ell_p)$, either:
  - $\exists$ coord. projection to $\mathbb{R}^{0.99d}$ with dense ℓ_p-ball of radius $O(\log p \cdot \log^{1/p} d/\epsilon)$, or
  - Random decomposition s.t. (i) overlap is small, and (ii) split query-nn rarely.

Finding Structure in Arbitrary Datasets

the “data-dependent” approach

- **<u>Dense Projected-Ball or Decompose</u>**. [Andoni-Nikolov-Razenshteyn-Waingarten ’21]
- Thm: For any $P \subset (\mathbb{R}^d, \ell_p)$, either:
  - $\exists$ coord. projection to $\mathbb{R}^{0.99d}$ with dense ℓ_p-ball of radius $O(\log p \cdot \log^{1/p} d/\epsilon)$, or
  - Random decomposition s.t. (i) overlap is small, and (ii) split query-nn rarely.

Query time:

i. $T(n,d) \leq T(n, 0.01d) + T(n/2, d)$

ii. $T(n,d) \leq n^\rho$

$$O(p/\epsilon) \to O(\log p \cdot \log^{2/p} d/\epsilon)$$

Finding Structure in Arbitrary Datasets

the “data-dependent” approach

- **<u>Dense Projected-Ball or Decompose</u>.** [Andoni-Nikolov-Razenshteyn-Waingarten ’21]
- Thm: For any $P \subset (\mathbb{R}^d, \ell_p)$, either:
  - $\exists$ coord. projection to $\mathbb{R}^{0.99d}$ with dense ℓ_p-ball of radius $O(\log p \cdot \log^{1/p} d/\epsilon)$, or
  - Random decomposition s.t. (i) overlap is small, and (ii) split query-nn rarely.

Query time:

i. $T(n,d) \leq T(n, 0.01d) + T(n/2, d)$

ii. $T(n,d) \leq n^\rho$

$$O(p/\epsilon) \to O(\log p \cdot \log^{2/p} d/\epsilon)$$

$$\to O(\log p/\epsilon)$$

[Andoni, Shekel-Nosatzki ’25]

Other aspects…

- Approximate nearest neighbors for …
  - Earth-Mover's Distance [Andoni-Indyk-Krauthgamer '08, Jayaram-Waingarten-Zhang '24]
  - Edit Distance [Ostrovsky-Rabani '05, Andoni-ShekelNosatzki '25]
  - Symmetric norms [Andoni-Nikolov-Nguyen-Razenshteyn-Waingarten '17]
- Geometric Spanners in high-dimension [HarPeled-Indyk-Sidropolous'13, Andoni-Zhang'23]
- Kernel Density Estimation [Charikar-Kapralov-Nouri-Siminelakis '20, Backurs-Charikar-Indyk-Siminelakis '18]
- Privacy and Fairness in Nearest Neighbors [Aumuller-HarPeled-Mahabadi-Pagh-Silvestri '22, Andoni-Indyk-Mahabadi-Narayanan '23]

Other aspects…

- Approximate nearest neighbors for …
  - Earth-Mover's Distance [Andoni-Indyk-Krauthgamer '08, Jayaram-Waingarten-Zhang '24]
  - Edit Distance [Ostrovsky-Rabani '05, Andoni-ShekelNosatzki '25]
  - Symmetric norms [Andoni-Nikolov-Nguyen-Razenshteyn-Waingarten '17]
- Geometric Spanners in high-dimension [HarPeled-Indyk-Sidropolous'13, Andoni-Zhang'23]
- Kernel Density Estimation [Charikar-Kapralov-Nouri-Siminelakis '20, Backurs-Charikar-Indyk-Siminelakis '18]
- Privacy and Fairness in Nearest Neighbors [Aumuller-HarPeled-Mahabadi-Pagh-Silvestri '22, Andoni-Indyk-Mahabadi-Narayanan '23]

Thanks!