Optimal Data-Dependent Hashing for Approximate Near Neighbors

Alexandr Andoni[1] **Ilya Razenshteyn**[2]

[1]Simons Institute

[2]MIT, CSAIL

April 20, 2015

Nearest Neighbor Search (NNS)

- Let P be an n-point subset of $\mathbb{R}^d$ equipped with some metric
- Given a query report the closest point from P

Nearest Neighbor Search (NNS)

- Let P be an n-point subset of $\mathbb{R}^d$ equipped with some metric
- Given a query report the closest point from P
- Space, query time
- Correct for a fixed query with probability 0.9

Nearest Neighbor Search (NNS)

- Let P be an n-point subset of $\mathbb{R}^d$ equipped with some metric
- Given a query report the closest point from P
- Space, query time
- Correct for a fixed query with probability 0.9
- Interesting metrics:
  - Euclidean (ℓ_2): $\|x - y\|_2 = \left(\sum_{i=1}^{d} |x_i - y_i|^2\right)^{1/2}$
  - Manhattan, Hamming (ℓ_1): $\|x - y\|_1 = \sum_{i=1}^{d} |x_i - y_i|$
  - ℓ_∞, edit distance, Earth Mover's distance etc.

The case $d = 2$ with Euclidean distance

- Build Voronoi diagram of P
- Given q perform point location

The case $d = 2$ with Euclidean distance

- Build Voronoi diagram of P
- Given q perform point location
- Performance:
  - Space $O(n)$
  - Query time $O(\log n)$

Applications

- Pattern recognition, statistical classification, computer vision, computational geometry, databases, recommendation systems, DNA sequencing, spell checking, plagiarism detection, clustering etc.

Applications

- Pattern recognition, statistical classification, computer vision, computational geometry, databases, recommendation systems, DNA sequencing, spell checking, plagiarism detection, clustering etc.
- Approximate string matching
  - A text T and a query S; substring of T closest to S
  - Dimension: $|S|$

Applications

- Pattern recognition, statistical classification, computer vision, computational geometry, databases, recommendation systems, DNA sequencing, spell checking, plagiarism detection, clustering etc.
- Approximate string matching
  - A text T and a query S; substring of T closest to S
  - Dimension: $|S|$
- Machine learning: nearest neighbor rule
  - The closest labeled example
  - Dimension: number of features

Applications

- Pattern recognition, statistical classification, computer vision, computational geometry, databases, recommendation systems, DNA sequencing, spell checking, plagiarism detection, clustering etc.
- Approximate string matching
  - A text T and a query S; substring of T closest to S
  - Dimension: $|S|$
- Machine learning: nearest neighbor rule
  - The closest labeled example
  - Dimension: number of features
- Near-duplicate document retrieval
  - Bag of words
  - Dimension: number of English words!

How to deal with many dimensions?

- Voronoi diagram takes $n^{\Omega(d)}$ space

How to deal with many dimensions?

- Voronoi diagram takes $n^{\Omega(d)}$ space
- All known data structures have space *exponential in d*

How to deal with many dimensions?

- Voronoi diagram takes $n^{\Omega(d)}$ space
- All known data structures have space *exponential in d*
- (Dobkin, Lipton 1976), (Yao, Yao 1985), (Clarkson 1988), (Agrawal, Matoušek 1992), (Matoušek 1992), (Meister 1993)

How to deal with many dimensions?

- Voronoi diagram takes $n^{\Omega(d)}$ space
- All known data structures have space *exponential in d*
- (Dobkin, Lipton 1976), (Yao, Yao 1985), (Clarkson 1988), (Agrawal, Matoušek 1992), (Matoušek 1992), (Meister 1993)
- Linear scan
- K-d trees, if d is relatively low

How to deal with many dimensions?

- Voronoi diagram takes $n^{\Omega(d)}$ space
- All known data structures have space *exponential in d*
- (Dobkin, Lipton 1976), (Yao, Yao 1985), (Clarkson 1988), (Agrawal, Matoušek 1992), (Matoušek 1992), (Meister 1993)
- Linear scan
- K-d trees, if d is relatively low
- Simplify the problem!

Outline

- **Approximate NNS**
- Locality-Sensitive Hashing (LSH) and Beyond
- Known LSH Families
- New Data Structure

The Approximate Nearest Neighbor Problem

- Let P be an n-point subset of $\mathbb{R}^d$ equipped with some metric, $\mathbf{c} > \mathbf{1}$
- Given a query $q \in \mathbb{R}^d$ report a c-approximate nearest neighbor from P

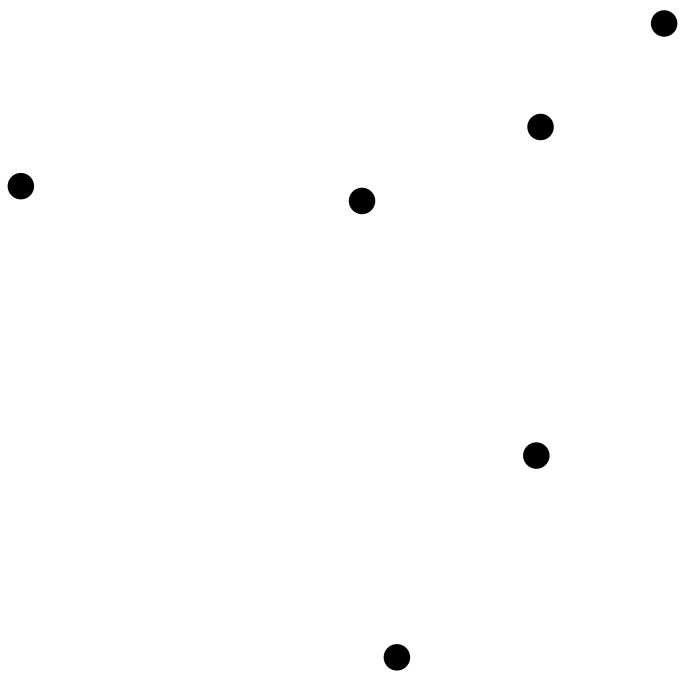
$$\|p - q\| \leq c \cdot \min_{p^* \in P} \|p^* - q\|$$

Literature

- **Exponential dependence on the dimension:**
  (Arya, Mount 1993), (Clarkson 1994),
  (Arya, Mount, Netanyahu, Silverman, We 1998), (Kleinberg 1997),
  (Har-Peled 2002)
- **Polynomial dependence on the dimension:**
  (Indyk, Motwani 1998), (Kushilevitz, Ostrovsky, Rabani 1998),
  (Indyk 1998), (Indyk 2001), (Gionis, Indyk, Motwani 1999),
  (Charikar 2002), (Datar, Immorlica, Indyk, Mirrokni 2004),
  (Chakrabarti, Regev 2004), (Panigrahy 2006), (Ailon, Chazelle 2006),
  (Andoni, Indyk 2006),
  (Andoni, Indyk, Nguyễn, Razenshteyn 2014),
  (Bartal, Gottlieb 2014), **(Andoni, Razenshteyn 2015)**

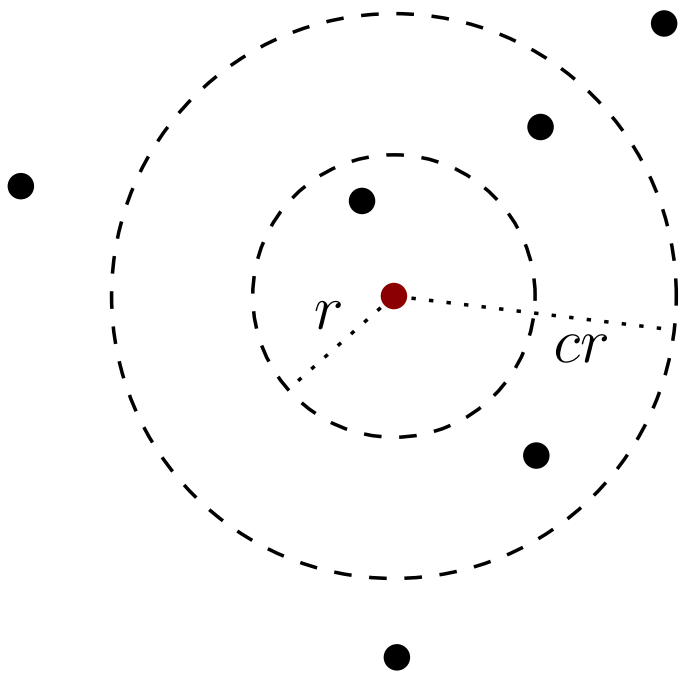
The Approximate **Near** Neighbor Problem (ANN)

- Let P be an n-point subset of $\mathbb{R}^d$, $\mathbf{r} > \mathbf{0}$, $c > 1$
- For $q \in \mathbb{R}^d$ find any point from P within cr, if one is promised to have a point within r



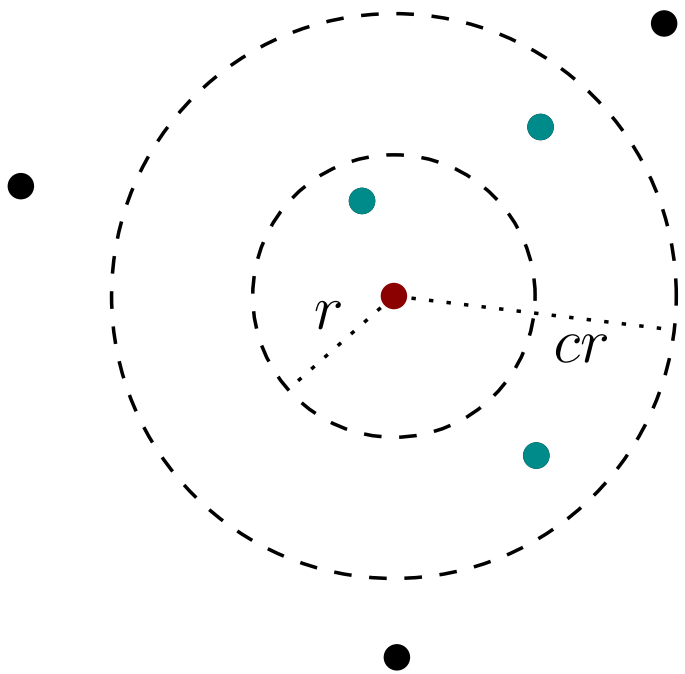
The Approximate **Near** Neighbor Problem (ANN)

- Let P be an n-point subset of $\mathbb{R}^d$, $\mathbf{r} > \mathbf{0}$, $c > 1$
- For $q \in \mathbb{R}^d$ find any point from P within cr, if one is promised to have a point within r



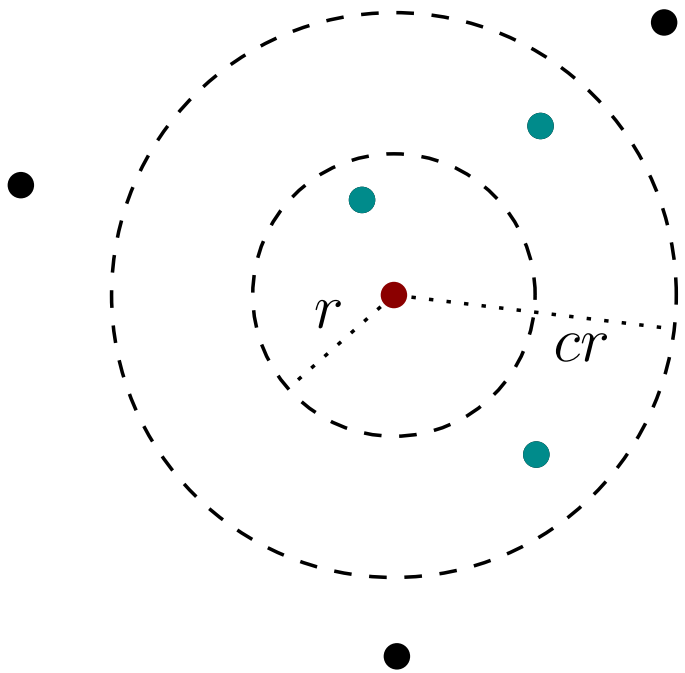
The Approximate **Near** Neighbor Problem (ANN)

- Let P be an n-point subset of $\mathbb{R}^d$, $\mathbf{r} > \mathbf{0}$, $c > 1$
- For $q \in \mathbb{R}^d$ find any point from P within cr, if one is promised to have a point within r



The Approximate **Near** Neighbor Problem (ANN)

- Let P be an n-point subset of $\mathbb{R}^d$, $\mathbf{r} > \mathbf{0}$, $c > 1$
- For $q \in \mathbb{R}^d$ find any point from P within cr, if one is promised to have a point within r
- There is a (non-trivial!) reduction from **Nearest** to **Near**



Outline

- Approximate NNS
- **Locality-Sensitive Hashing (LSH) and Beyond**
- Known LSH Families
- New Data Structure

Locality-Sensitive Hashing (LSH)

- The goal: ANN for $\mathbb{R}^d$ with
  - space and query time polynomial in d;
  - space subquadratic in n and query time sublinear in n.

Locality-Sensitive Hashing (LSH)

- The goal: ANN for $\mathbb{R}^d$ with
  - space and query time polynomial in d;
  - space subquadratic in n and query time sublinear in n.
- The only known technique (until recently):
  Locality-Sensitive Hashing (LSH) (Indyk, Motwani 1998)

Locality-Sensitive Hashing (LSH)

- The goal: ANN for $\mathbb{R}^d$ with
  - space and query time polynomial in d;
  - space subquadratic in n and query time sublinear in n.
- The only known technique (until recently):
  Locality-Sensitive Hashing (LSH) (Indyk, Motwani 1998)
- Hash functions $\leftrightarrow$ space partitions

Locality-Sensitive Hashing (LSH)

- The goal: ANN for $\mathbb{R}^d$ with
  - space and query time polynomial in d;
  - space subquadratic in n and query time sublinear in n.
- The only known technique (until recently):
  Locality-Sensitive Hashing (LSH) (Indyk, Motwani 1998)
- Hash functions $\leftrightarrow$ space partitions
- A distribution over partitions $\mathcal{P}$ of $\mathbb{R}^d$ is (r, cr, p_1, p_2)-*sensitive*, if for every $p, q \in \mathbb{R}^d$:
  - if $\|p - q\| \leq r$, then $\Pr_{\mathcal{P}}[\mathcal{P}(p) = \mathcal{P}(q)] \geq p_1$;
  - if $\|p - q\| \geq cr$, then $\Pr_{\mathcal{P}}[\mathcal{P}(p) = \mathcal{P}(q)] \leq p_2$
- Want: large gap between p_1 and p_2

From LSH to ANN

- Let $\mathcal{P}$ be a *reasonable* (r, cr, p_1, p_2)-sensitive (random) partition

From LSH to ANN

- Let $\mathcal{P}$ be a *reasonable* (r, cr, p_1, p_2)-sensitive (random) partition
- Define *quality* of $\mathcal{P}$ as

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

From LSH to ANN

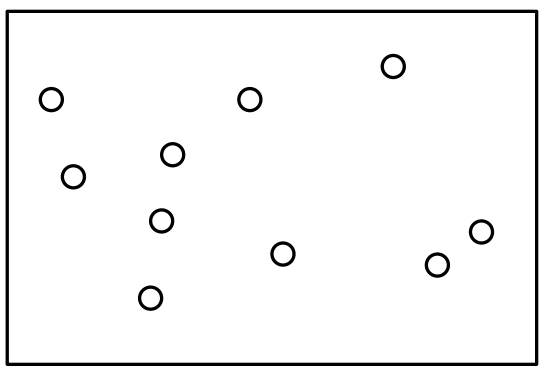
- Let $\mathcal{P}$ be a *reasonable* (r, cr, p_1, p_2)-sensitive (random) partition
- Define *quality* of $\mathcal{P}$ as

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

- Then, can solve ANN with roughly $O(n^{1+\rho} + dn)$ space and $O(dn^{\rho})$ query time (Indyk, Motwani 1998)

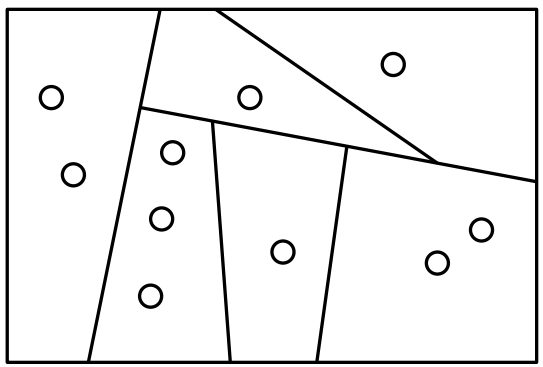
Proof idea

- Let $\mathcal{H}$ be a (r, cr, p_1, p_2)-sensitive family: for every $p, q \in X$
  - if $\|p - q\| \leq r$, then $\Pr_{h \sim \mathcal{H}}[h(p) = h(q)] \geq p_1$;
  - if $\|p - q\| \geq cr$, then $\Pr_{h \sim \mathcal{H}}[h(p) = h(q)] \leq p_2$



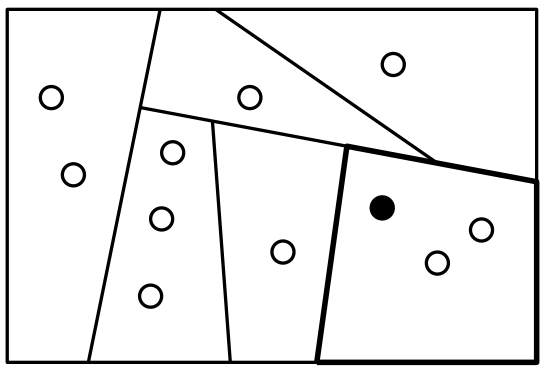
Proof idea

- Let $\mathcal{H}$ be a (r, cr, p_1, p_2)-sensitive family: for every $p, q \in X$
  - if $\|p - q\| \leq r$, then $\Pr_{h \sim \mathcal{H}}[h(p) = h(q)] \geq p_1$;
  - if $\|p - q\| \geq cr$, then $\Pr_{h \sim \mathcal{H}}[h(p) = h(q)] \leq p_2$
- Hash the dataset P using a concatenation of k functions from $\mathcal{H}$: $x \mapsto (h_1(x), h_2(x), \ldots, h_k(x))$



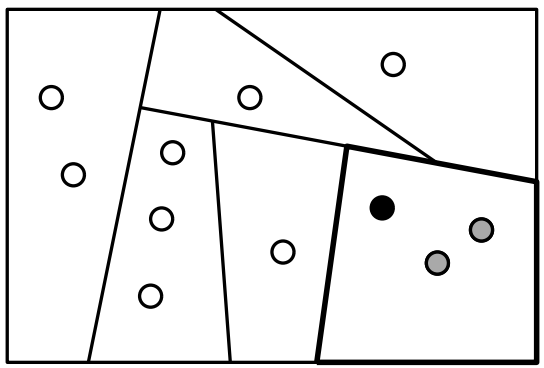
Proof idea

- Let $\mathcal{H}$ be a (r, cr, p_1, p_2)-sensitive family: for every $p, q \in X$
  - if $\|p - q\| \leq r$, then $\Pr_{h \sim \mathcal{H}}[h(p) = h(q)] \geq p_1$;
  - if $\|p - q\| \geq cr$, then $\Pr_{h \sim \mathcal{H}}[h(p) = h(q)] \leq p_2$
- Hash the dataset P using a concatenation of k functions from $\mathcal{H}$: $x \mapsto (h_1(x), h_2(x), \ldots, h_k(x))$
- Locate a query q and enumerate all points from the bucket



Proof idea

- Let $\mathcal{H}$ be a (r, cr, p_1, p_2)-sensitive family: for every $p, q \in X$
  - if $\|p - q\| \leq r$, then $\Pr_{h \sim \mathcal{H}}[h(p) = h(q)] \geq p_1$;
  - if $\|p - q\| \geq cr$, then $\Pr_{h \sim \mathcal{H}}[h(p) = h(q)] \leq p_2$
- Hash the dataset P using a concatenation of k functions from $\mathcal{H}$: $x \mapsto (h_1(x), h_2(x), \ldots, h_k(x))$
- Locate a query q and enumerate all points from the bucket
- The optimal choice of k leads to the need in n^ρ hash tables
- Overall: $n^{1+\rho}$ space, n^ρ query time



Prior work on LSH

(Indyk, Motwani 1998), (Andoni, Indyk 2006),
(Motwani, Naor, Panigrahy 2007), (O'Donnell, Wu, Zhou 2011)

Bounds on $\rho = \ln(1/p_1)/\ln(1/p_2)$ for various spaces:

Distance	Upper bound	Lower bound	$c = 2$
Hamming	$\rho \leq 1/c$	$\rho \geq 1/c - o(1)$	$1/2$
Euclidean	$\rho \leq 1/c^2 + o(1)$	$\rho \geq 1/c^2 - o(1)$	$1/4$

Can we do better for ANN?

Prior work on LSH

(Indyk, Motwani 1998), (Andoni, Indyk 2006),
(Motwani, Naor, Panigrahy 2007), (O'Donnell, Wu, Zhou 2011)

Bounds on $\rho = \ln(1/p_1)/\ln(1/p_2)$ for various spaces:

Distance	Upper bound	Lower bound	$c = 2$
Hamming	$\rho \leq 1/c$	$\rho \geq 1/c - o(1)$	$1/2$
Euclidean	$\rho \leq 1/c^2 + o(1)$	$\rho \geq 1/c^2 - o(1)$	$1/4$

Can we do better for ANN?

(Andoni, Indyk, Nguyễn, Razenshteyn 2014),
(Andoni, Razenshteyn 2015): **yes!**
ANN for ℓ_2 with space $O(n^{1+\tau} + dn)$ and time $O(dn^{\tau})$ with

$$\tau \leq 1/(2c^2 - 1) + o(1)$$

($1/7$ for $c = 2$).
The first data structures that overcome the lower bound for LSH.

How to do better than LSH?

- **The main idea: data-dependent space partitioning**

How to do better than LSH?

- **The main idea: data-dependent space partitioning**
- $\mathcal{P}$ is (r, cr, p_1, p_2)-*sensitive*, if for every $p, q \in \mathbb{R}^d$
  - if $\|p - q\| \leq r$, then $\Pr_{\mathcal{P}}[\mathcal{P}(p) = \mathcal{P}(q)] \geq p_1$;
  - if $\|p - q\| \geq cr$, then $\Pr_{\mathcal{P}}[\mathcal{P}(p) = \mathcal{P}(q)] \leq p_2$

How to do better than LSH?

- **The main idea: data-dependent space partitioning**
- $\mathcal{P}$ is (r, cr, p_1, p_2)-*sensitive*, if for every $p, q \in \mathbb{R}^d$
  - if $\|p - q\| \leq r$, then $\Pr_{\mathcal{P}}[\mathcal{P}(p) = \mathcal{P}(q)] \geq p_1$;
  - if $\|p - q\| \geq cr$, then $\Pr_{\mathcal{P}}[\mathcal{P}(p) = \mathcal{P}(q)] \leq p_2$
- **Too strong!** Enough to satisfy these for $p \in P$ and $q \in \mathbb{R}^d$. Can exploit the geometry of P to construct a partition

How to do better than LSH?

- **The main idea: data-dependent space partitioning**
- $\mathcal{P}$ is (r, cr, p_1, p_2)-*sensitive*, if for every $p, q \in \mathbb{R}^d$
  - if $\|p - q\| \leq r$, then $\Pr_{\mathcal{P}}[\mathcal{P}(p) = \mathcal{P}(q)] \geq p_1$;
  - if $\|p - q\| \geq cr$, then $\Pr_{\mathcal{P}}[\mathcal{P}(p) = \mathcal{P}(q)] \leq p_2$
- **Too strong!** Enough to satisfy these for $p \in P$ and $q \in \mathbb{R}^d$. Can exploit the geometry of P to construct a partition
- Our bounds are **optimal** for data-dependent hashing

How to do better than LSH?

- **The main idea: data-dependent space partitioning**
- $\mathcal{P}$ is (r, cr, p_1, p_2)-*sensitive*, if for every $p, q \in \mathbb{R}^d$
  - if $\|p - q\| \leq r$, then $\Pr_{\mathcal{P}}[\mathcal{P}(p) = \mathcal{P}(q)] \geq p_1$;
  - if $\|p - q\| \geq cr$, then $\Pr_{\mathcal{P}}[\mathcal{P}(p) = \mathcal{P}(q)] \leq p_2$
- **Too strong!** Enough to satisfy these for $p \in P$ and $q \in \mathbb{R}^d$. Can exploit the geometry of P to construct a partition
- Our bounds are **optimal** for data-dependent hashing
- Parallels with practice!
  - PCA trees (Sproull 1991), (McNames 2001), (Verma, Kpotufe, Dasgupta 2009)
  - Spectral Hashing (Weiss, Torralba, Fergus 2008)
  - Semantic Hashing (Salakhutdinov, Hinton 2009)
  - WTA Hashing (Yagnik, Strelow, Ross, Lin 2011)

Outline

- Approximate NNS
- Locality-Sensitive Hashing (LSH) and Beyond
- **Known LSH Families**
- New Data Structure

Hamming distance

- Hypercube $\{0,1\}^d$ with Hamming distance $\|x-y\| = \sum_{i=1}^{d} |x_i - y_i|$

Hamming distance

- Hypercube $\{0,1\}^d$ with Hamming distance $\|x-y\| = \sum_{i=1}^{d} |x_i - y_i|$
- Choose random $i \in \{1,2,\ldots,d\}$

Hamming distance

- Hypercube $\{0,1\}^d$ with Hamming distance $\|x-y\| = \sum_{i=1}^{d} |x_i - y_i|$
- Choose random $i \in \{1,2,\ldots,d\}$
- Partition $\{0,1\}^d = \{x_i = 0\} \cup \{x_i = 1\}$

Hamming distance

- Hypercube $\{0,1\}^d$ with Hamming distance $\|x-y\| = \sum_{i=1}^{d} |x_i - y_i|$
- Choose random $i \in \{1,2,\ldots,d\}$
- Partition $\{0,1\}^d = \{x_i = 0\} \cup \{x_i = 1\}$
- $\mathcal{P}$ is (r, cr, p_1, p_2)-sensitive, where $p_1 = 1 - r/d$ and $p_2 = 1 - cr/d$

11101110
10111101

Hamming distance

- Hypercube $\{0,1\}^d$ with Hamming distance $\|x-y\| = \sum_{i=1}^{d} |x_i - y_i|$
- Choose random $i \in \{1,2,\ldots,d\}$
- Partition $\{0,1\}^d = \{x_i = 0\} \cup \{x_i = 1\}$
- $\mathcal{P}$ is (r, cr, p_1, p_2)-sensitive, where $p_1 = 1 - r/d$ and $p_2 = 1 - cr/d$
- As a result,

$$\rho = \frac{\ln(1/p_1)}{\ln(1/p_2)} \approx \frac{r/d}{cr/d} = \frac{1}{c}$$

11101110
10111101

Hamming distance

- Hypercube $\{0,1\}^d$ with Hamming distance $\|x-y\| = \sum_{i=1}^{d} |x_i - y_i|$
- Choose random $i \in \{1,2,\ldots,d\}$
- Partition $\{0,1\}^d = \{x_i = 0\} \cup \{x_i = 1\}$
- $\mathcal{P}$ is (r, cr, p_1, p_2)-sensitive, where $p_1 = 1 - r/d$ and $p_2 = 1 - cr/d$
- As a result,

$$\rho = \frac{\ln(1/p_1)}{\ln(1/p_2)} \approx \frac{r/d}{cr/d} = \frac{1}{c}$$

- Overall, ANN with space $O(n^{1+1/c} + dn)$ and query time $O(dn^{1/c})$

11101110
10111101

Hamming distance

- Hypercube $\{0,1\}^d$ with Hamming distance $\|x-y\| = \sum_{i=1}^{d} |x_i - y_i|$
- Choose random $i \in \{1,2,\ldots,d\}$
- Partition $\{0,1\}^d = \{x_i = 0\} \cup \{x_i = 1\}$
- $\mathcal{P}$ is (r, cr, p_1, p_2)-sensitive, where $p_1 = 1 - r/d$ and $p_2 = 1 - cr/d$
- As a result,

$$\rho = \frac{\ln(1/p_1)}{\ln(1/p_2)} \approx \frac{r/d}{cr/d} = \frac{1}{c}$$

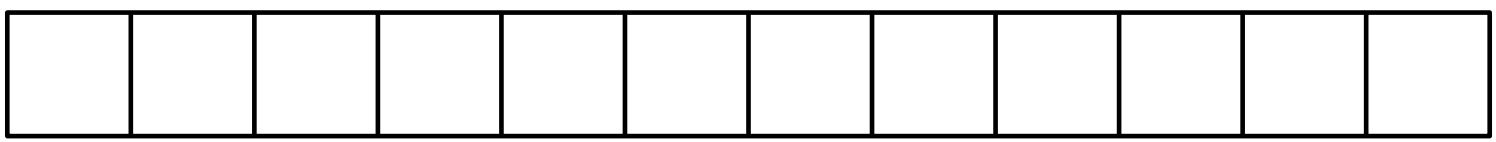
- Overall, ANN with space $O(n^{1+1/c} + dn)$ and query time $O(dn^{1/c})$
- Can be generalized to the whole ℓ_1

11101110
10111101

Full algorithm for the Hamming distance

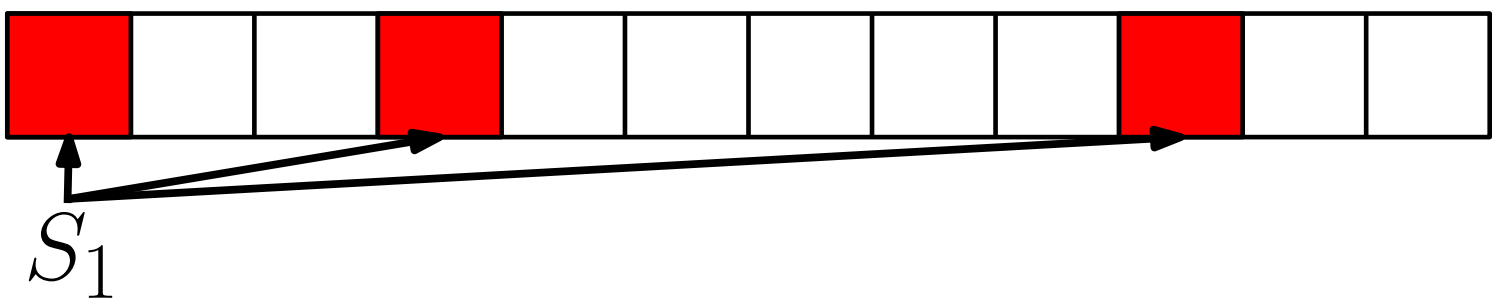
Full algorithm for the Hamming distance

- Sample L subsets of coordinates S_1, S_2, $\ldots$, S_L, each of size k



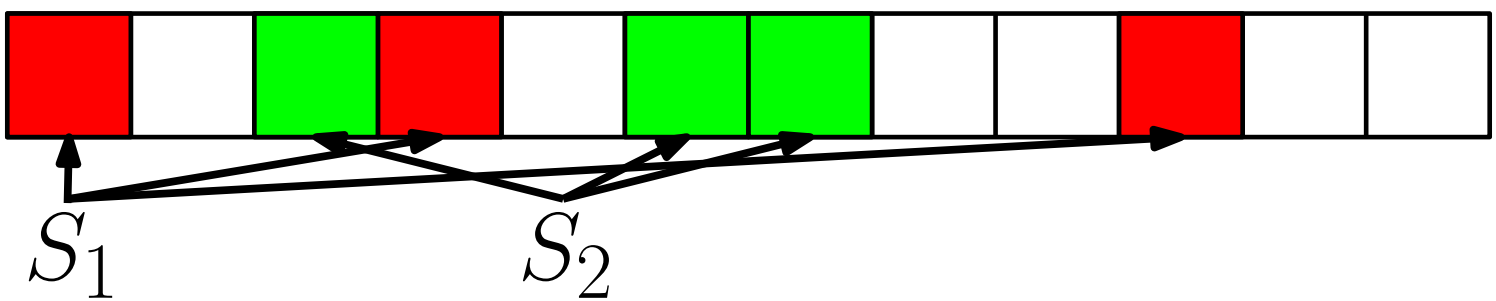
Full algorithm for the Hamming distance

- Sample L subsets of coordinates $S_1, S_2, \ldots, S_L$, each of size k



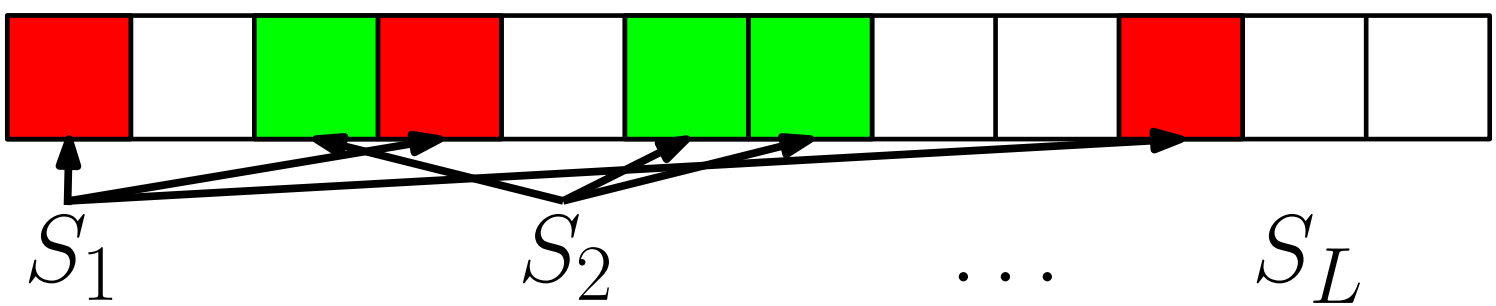
Full algorithm for the Hamming distance

- Sample L subsets of coordinates $S_1, S_2, \ldots, S_L$, each of size k



Full algorithm for the Hamming distance

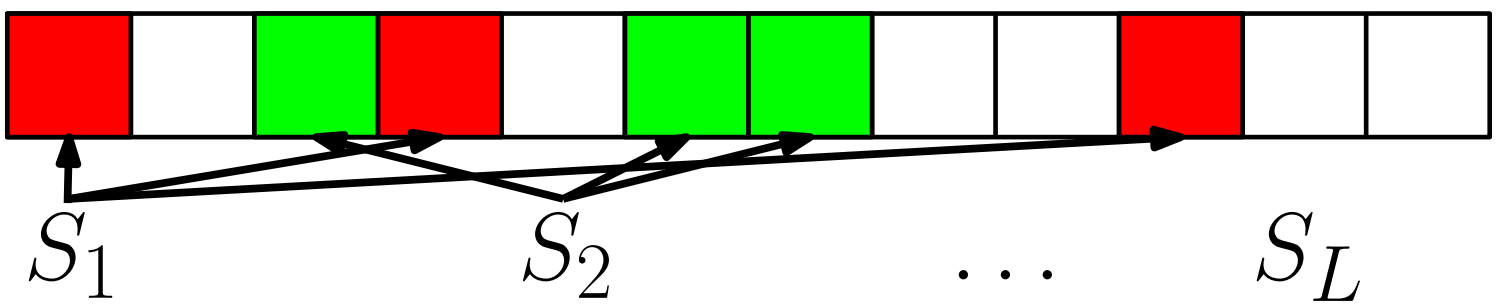
- Sample L subsets of coordinates $S_1, S_2, \ldots, S_L$, each of size k



Full algorithm for the Hamming distance

- Sample L subsets of coordinates $S_1, S_2, \ldots, S_L$, each of size k
- On a query q retrieve all the data points p such that

$$\textbf{there exists } S_i \textbf{ s.t. } q_{S_i} = p_{S_i}$$

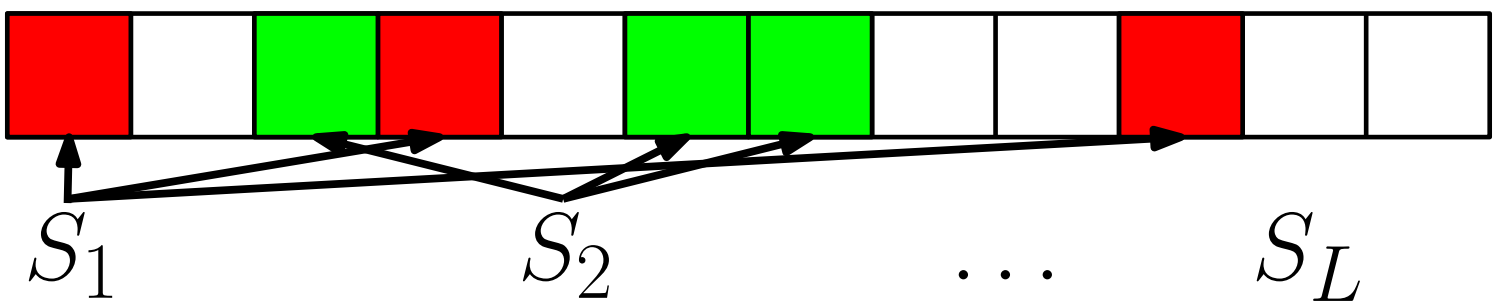


Full algorithm for the Hamming distance

- Sample L subsets of coordinates S_1, S_2, $\ldots$, S_L, each of size k
- On a query q retrieve all the data points p such that

$$\textbf{there exists } S_i \textbf{ s.t. } q_{S_i} = p_{S_i}$$

- $L \approx n^{1/c}$, $k \approx \log n$

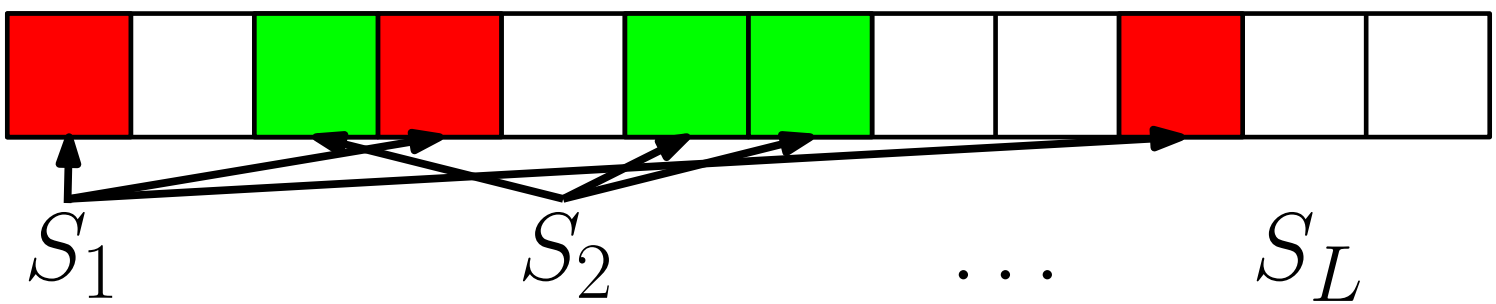


Full algorithm for the Hamming distance

- Sample L subsets of coordinates $S_1, S_2, \ldots, S_L$, each of size k
- On a query q retrieve all the data points p such that

$$\textbf{there exists } S_i \textbf{ s.t. } q_{S_i} = p_{S_i}$$

- $L \approx n^{1/c}$, $k \approx \log n$
- The new data structure is the *first improvement* upon bit sampling

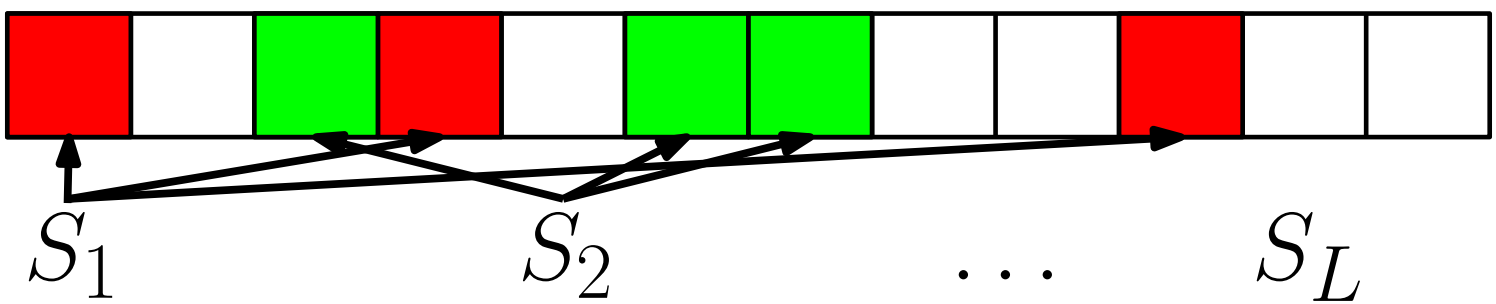


Full algorithm for the Hamming distance

- Sample L subsets of coordinates $S_1, S_2, \ldots, S_L$, each of size k
- On a query q retrieve all the data points p such that

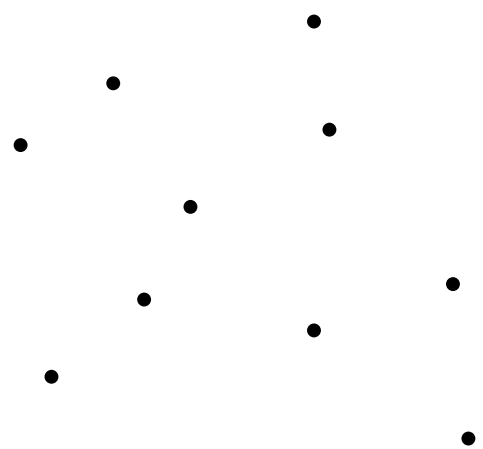
$$\textbf{there exists } S_i \textbf{ s.t. } q_{S_i} = p_{S_i}$$

- $L \approx n^{1/c}$, $k \approx \log n$
- The new data structure is the *first improvement* upon bit sampling
- In particular, for $c = 2$ improve from query time $O(n^{1/2})$ to $O(n^{1/3})$



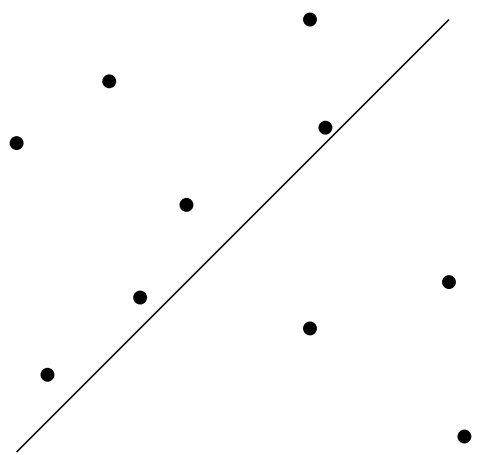
Euclidean distance

- LSH for ℓ_2 (with suboptimal ρ: $1/c$ instead of $1/c^2$) from (Datar, Immorlica, Indyk, Mirrokni 2004)



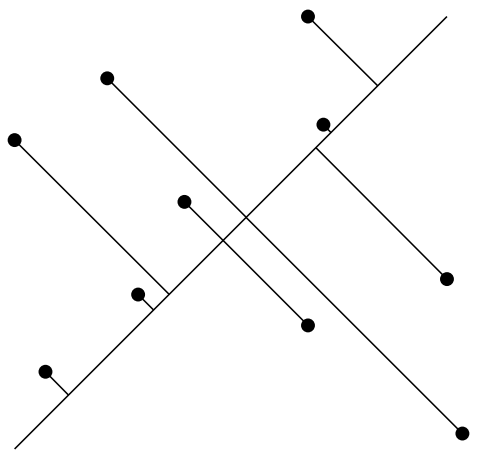
Euclidean distance

- LSH for ℓ_2 (with suboptimal ρ: $1/c$ instead of $1/c^2$) from (Datar, Immorlica, Indyk, Mirrokni 2004)
- Project $\mathbb{R}^d$ on a random line



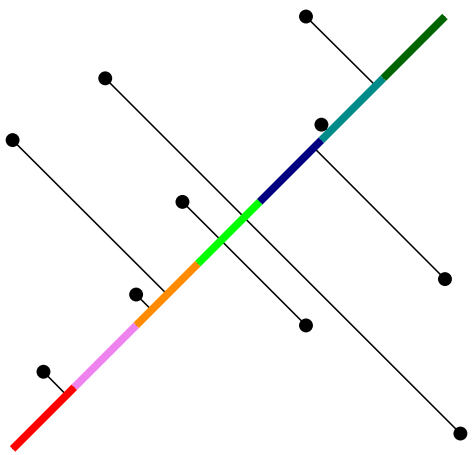
Euclidean distance

- LSH for ℓ_2 (with suboptimal ρ: $1/c$ instead of $1/c^2$) from (Datar, Immorlica, Indyk, Mirrokni 2004)
- Project $\mathbb{R}^d$ on a random line



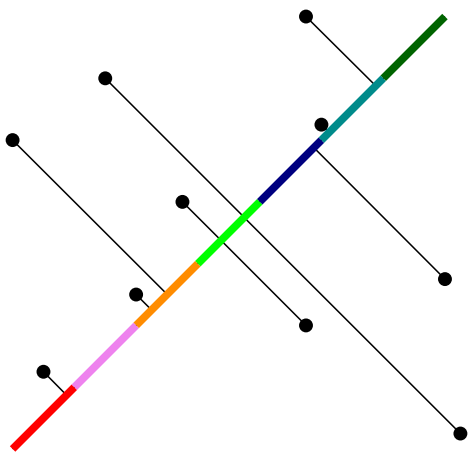
Euclidean distance

- LSH for ℓ_2 (with suboptimal ρ: $1/c$ instead of $1/c^2$) from (Datar, Immorlica, Indyk, Mirrokni 2004)
- Project $\mathbb{R}^d$ on a random line
- Partition the line starting from a random place in segments of length w



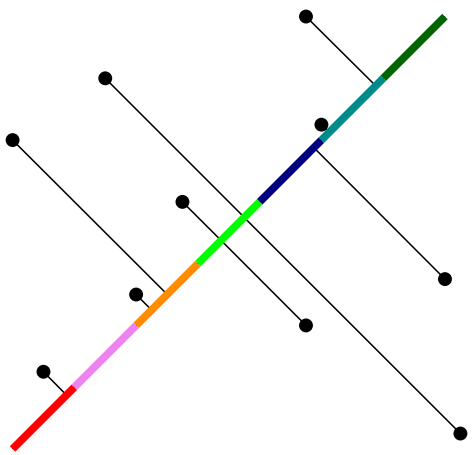
Euclidean distance

- LSH for ℓ_2 (with suboptimal ρ: $1/c$ instead of $1/c^2$) from (Datar, Immorlica, Indyk, Mirrokni 2004)
- Project $\mathbb{R}^d$ on a random line
- Partition the line starting from a random place in segments of length w
- Hash value is the segment that the projection falls into



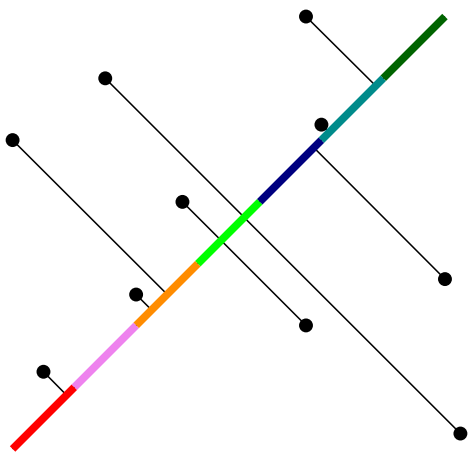
Euclidean distance

- LSH for ℓ_2 (with suboptimal ρ: $1/c$ instead of $1/c^2$) from (Datar, Immorlica, Indyk, Mirrokni 2004)
- Project $\mathbb{R}^d$ on a random line
- Partition the line starting from a random place in segments of length w
- Hash value is the segment that the projection falls into
- Optimizing $w = w(r, c)$, we get $\rho < 1/c$



Euclidean distance

- LSH for ℓ_2 (with suboptimal ρ: $1/c$ instead of $1/c^2$) from (Datar, Immorlica, Indyk, Mirrokni 2004)
- Project $\mathbb{R}^d$ on a random line
- Partition the line starting from a random place in segments of length w
- Hash value is the segment that the projection falls into
- Optimizing $w = w(r, c)$, we get $\rho < 1/c$
- Simple and practical: E^{2}LSH is based on this hashing scheme (Andoni, Indyk 2005)



Optimal LSH for Euclidean distance

- For ℓ_2 there is LSH with $\rho \leq 1/c^2 + o(1)$ (Andoni, Indyk 2006)

Optimal LSH for Euclidean distance

- For ℓ_2 there is LSH with $\rho \leq 1/c^2 + o(1)$ (Andoni, Indyk 2006)
- Unfortunately, fairly complicated

Optimal LSH for Euclidean distance

- For ℓ_2 there is LSH with $\rho \leq 1/c^2 + o(1)$ (Andoni, Indyk 2006)
- Unfortunately, fairly complicated
- Later, will see a simple family for a special case

Outline

- Approximate NNS
- Locality-Sensitive Hashing (LSH) and Beyond
- Known LSH Families
- **New Data Structure**

The plan

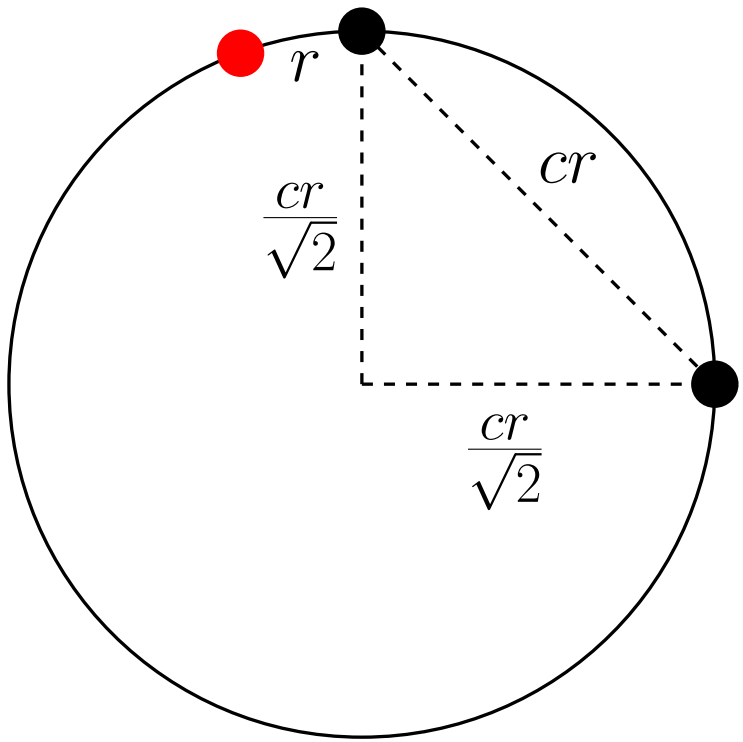
- A data structure based on *decision trees*

The plan

- A data structure based on *decision trees*
- First, show how to solve *nice* instances

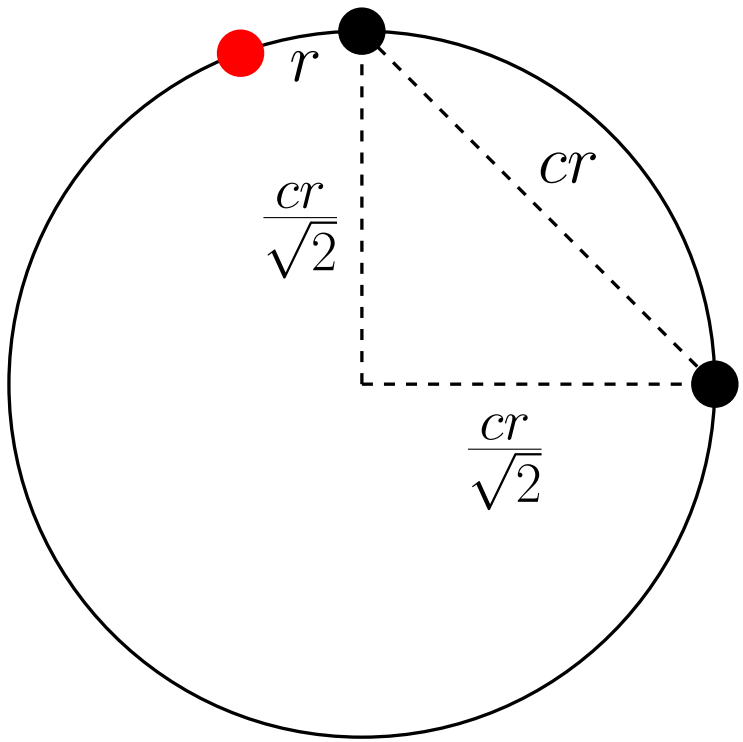
The plan

- A data structure based on *decision trees*
- First, show how to solve *nice* instances
  - Data points lie on a sphere of radius $cr/\sqrt{2}$



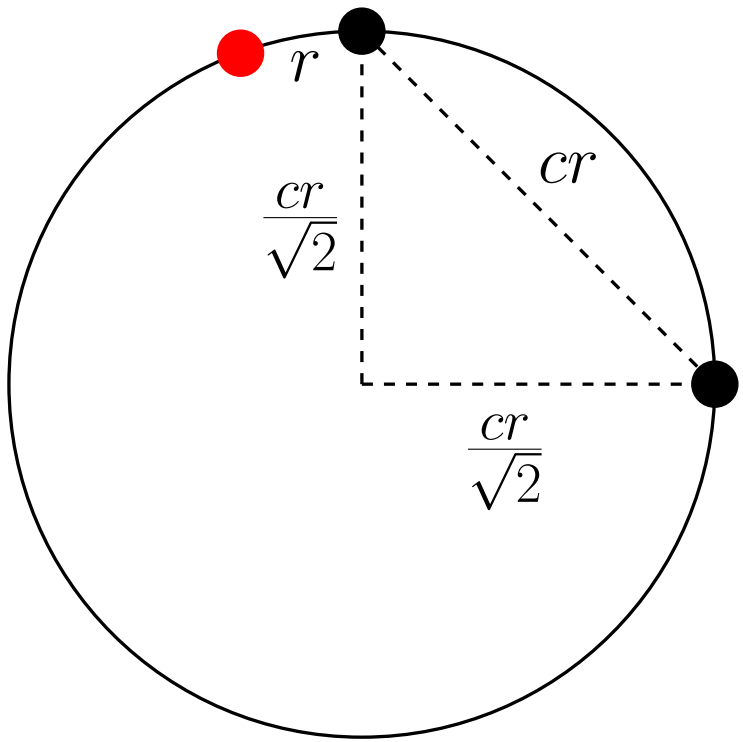
The plan

- A data structure based on *decision trees*
- First, show how to solve *nice* instances
  - Data points lie on a sphere of radius $cr/\sqrt{2}$
  - Intuition: this corresponds to a random instance



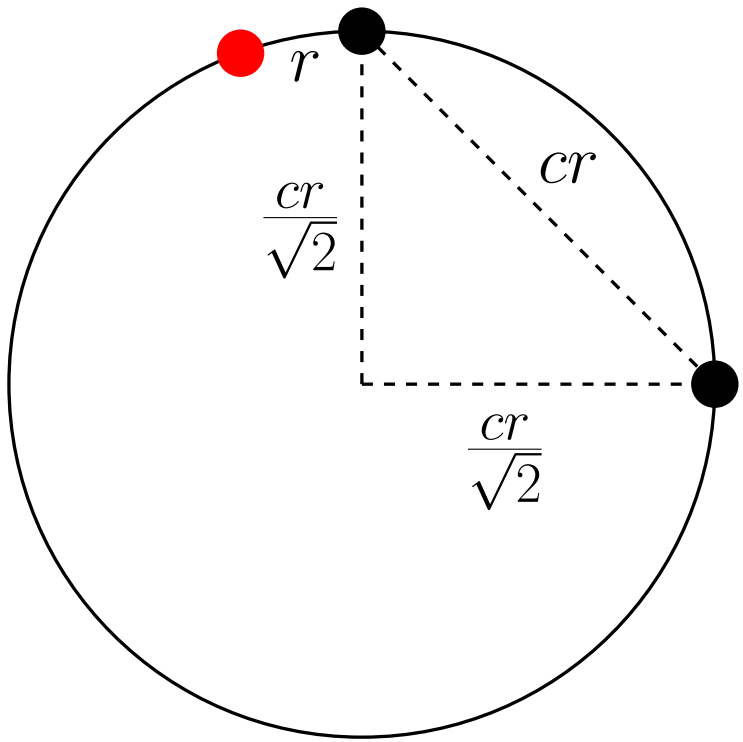
The plan

- A data structure based on *decision trees*
- First, show how to solve *nice* instances
  - Data points lie on a sphere of radius $cr/\sqrt{2}$
  - Intuition: this corresponds to a random instance
- Show how to reduce the general case to the nice case
  - Iterative clustering



The plan

- A data structure based on *decision trees*
- First, show how to solve *nice* instances
  - Data points lie on a sphere of radius $cr/\sqrt{2}$
  - Intuition: this corresponds to a random instance
- Show how to reduce the general case to the nice case
  - Iterative clustering
  - Data-dependent



The low-diameter case

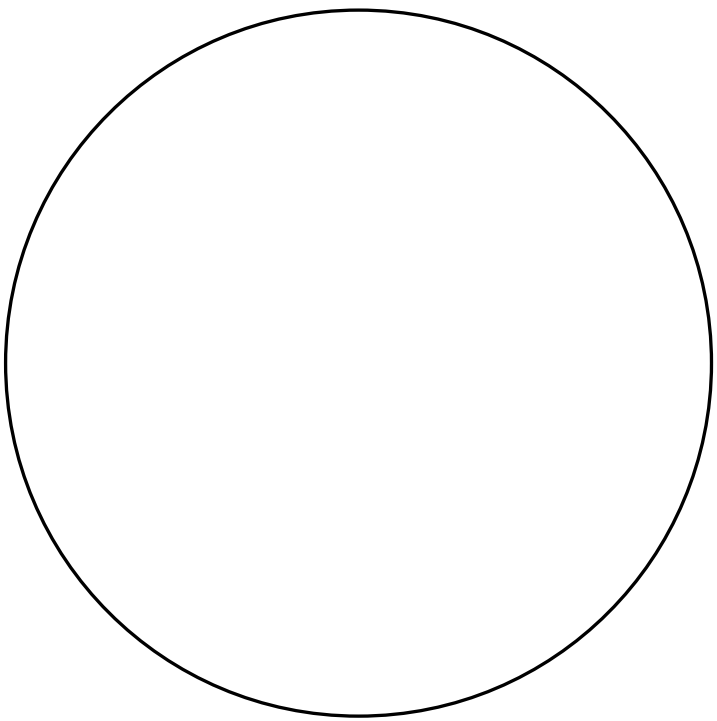
- All points and queries lie on a sphere of radius $cr/\sqrt{2}$

The low-diameter case

- All points and queries lie on a sphere of radius $cr/\sqrt{2}$
- Ball carving (Karger, Motwani, Sudan 1998), (Andoni, Indyk, Nguyen, Razenshteyn 2014)

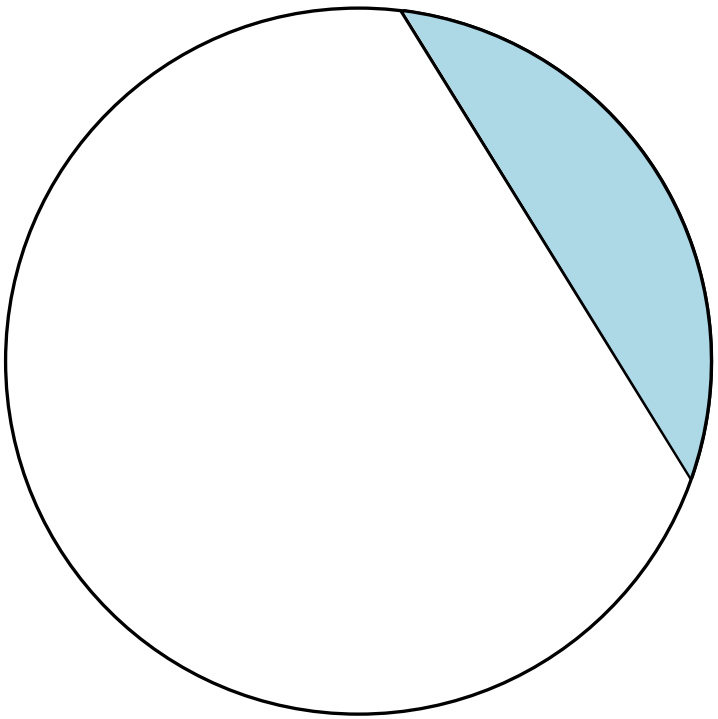
The low-diameter case

- All points and queries lie on a sphere of radius $cr/\sqrt{2}$
- Ball carving (Karger, Motwani, Sudan 1998), (Andoni, Indyk, Nguyen, Razenshteyn 2014)



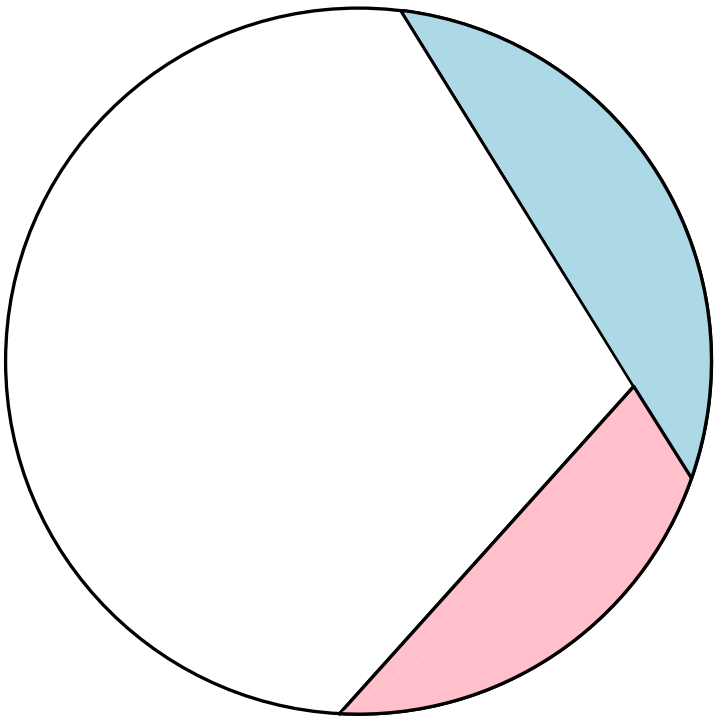
The low-diameter case

- All points and queries lie on a sphere of radius $cr/\sqrt{2}$
- Ball carving (Karger, Motwani, Sudan 1998), (Andoni, Indyk, Nguyen, Razenshteyn 2014)



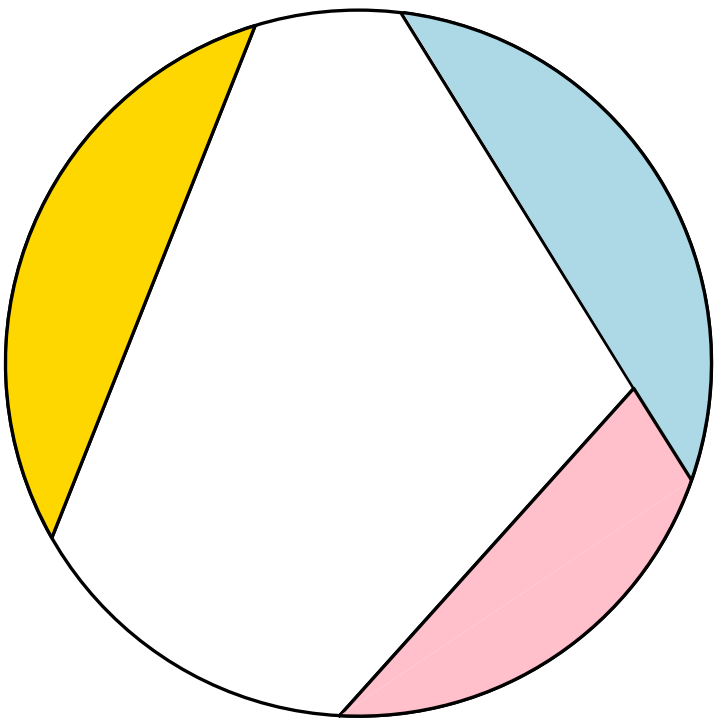
The low-diameter case

- All points and queries lie on a sphere of radius $cr/\sqrt{2}$
- Ball carving (Karger, Motwani, Sudan 1998), (Andoni, Indyk, Nguyen, Razenshteyn 2014)



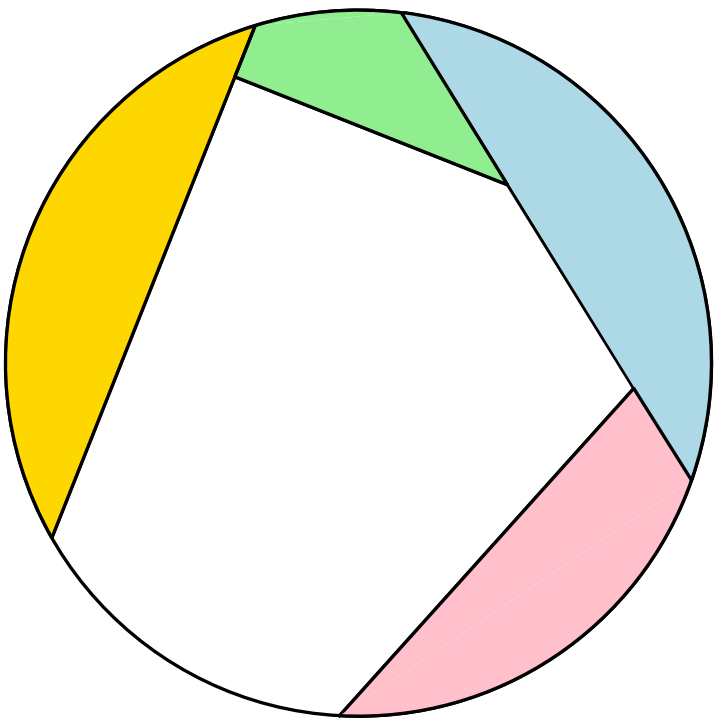
The low-diameter case

- All points and queries lie on a sphere of radius $cr/\sqrt{2}$
- Ball carving (Karger, Motwani, Sudan 1998), (Andoni, Indyk, Nguyen, Razenshteyn 2014)



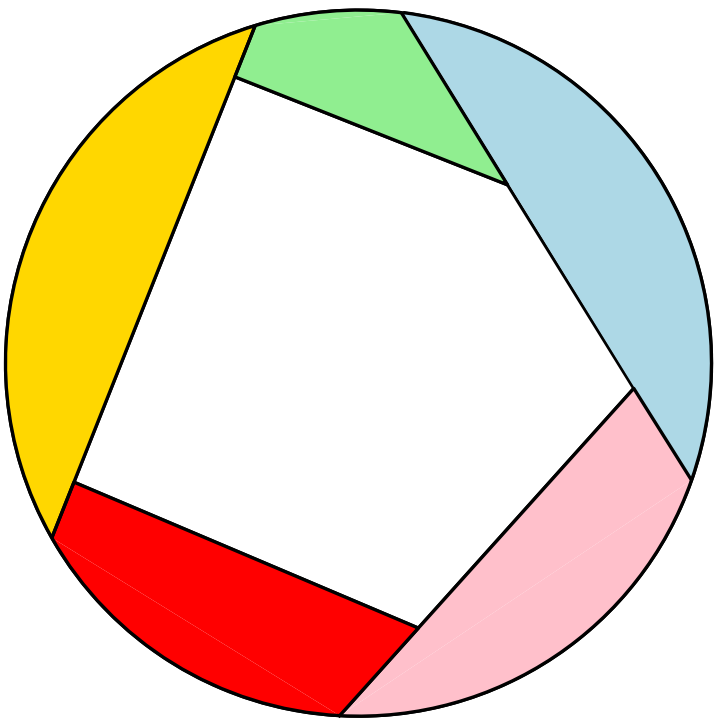
The low-diameter case

- All points and queries lie on a sphere of radius $cr/\sqrt{2}$
- Ball carving (Karger, Motwani, Sudan 1998), (Andoni, Indyk, Nguyen, Razenshteyn 2014)



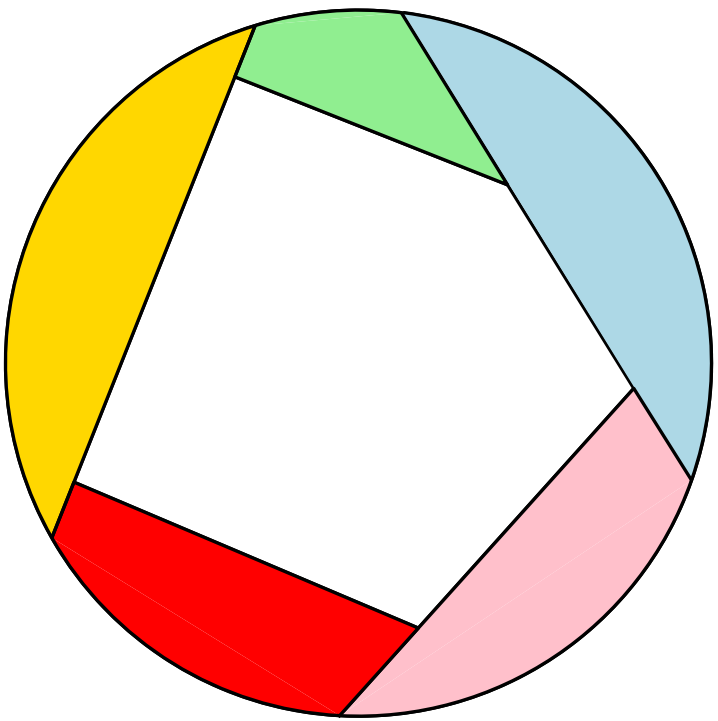
The low-diameter case

- All points and queries lie on a sphere of radius $cr/\sqrt{2}$
- Ball carving (Karger, Motwani, Sudan 1998), (Andoni, Indyk, Nguyen, Razenshteyn 2014)



The low-diameter case

- All points and queries lie on a sphere of radius $cr/\sqrt{2}$
- Ball carving (Karger, Motwani, Sudan 1998), (Andoni, Indyk, Nguyen, Razenshteyn 2014)
- Achieves $\rho = \frac{1}{2c^2-1}$



A glimpse of the analysis

- Sample $g_1, g_2, \ldots \sim N(0,1)^d$; the hash value $h(p) = \min i : \langle p, g_i \rangle \geq \eta\sqrt{d}$

A glimpse of the analysis

- Sample $g_1, g_2, \ldots \sim N(0,1)^d$; the hash value $h(p) = \min i : \langle p, g_i \rangle \geq \eta\sqrt{d}$
- Set $\eta = d^{-1/4}$: $\exp(\sqrt{d})$ caps in total

A glimpse of the analysis

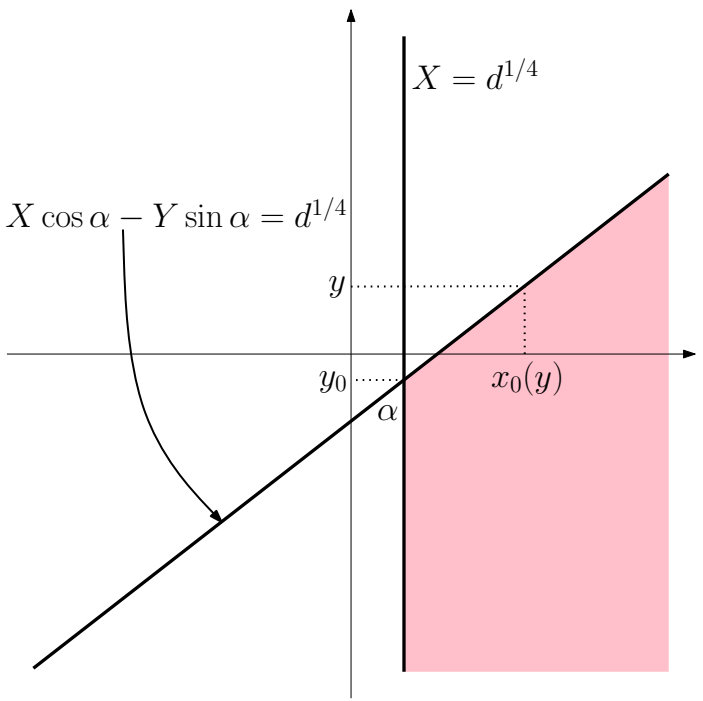
- Sample $g_1, g_2, \ldots \sim N(0,1)^d$; the hash value $h(p) = \min i : \langle p, g_i \rangle \geq \eta\sqrt{d}$
- Set $\eta = d^{-1/4}$: $\exp(\sqrt{d})$ caps in total
- One has

$$\Pr[p \text{ and } q \text{ collide}] = \frac{\Pr_g[\langle p, g\rangle \geq \eta\sqrt{d} \text{ and } \langle q, g\rangle \geq \eta\sqrt{d}]}{\Pr_g[\langle p, g\rangle \geq \eta\sqrt{d} \text{ or } \langle q, g\rangle \geq \eta\sqrt{d}]}$$

A glimpse of the analysis

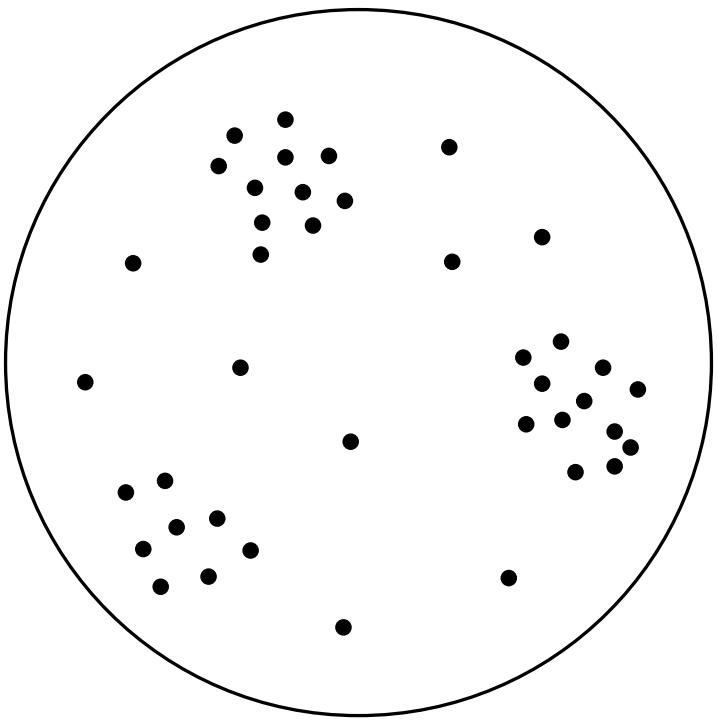
- Sample $g_1, g_2, \ldots \sim N(0,1)^d$; the hash value $h(p) = \min i : \langle p, g_i \rangle \geq \eta\sqrt{d}$
- Set $\eta = d^{-1/4}$: $\exp(\sqrt{d})$ caps in total
- One has

$$\Pr[p \text{ and } q \text{ collide}] = \frac{\Pr_g[\langle p, g\rangle \geq \eta\sqrt{d} \text{ and } \langle q, g\rangle \geq \eta\sqrt{d}]}{\Pr_g[\langle p, g\rangle \geq \eta\sqrt{d} \text{ or } \langle q, g\rangle \geq \eta\sqrt{d}]}$$



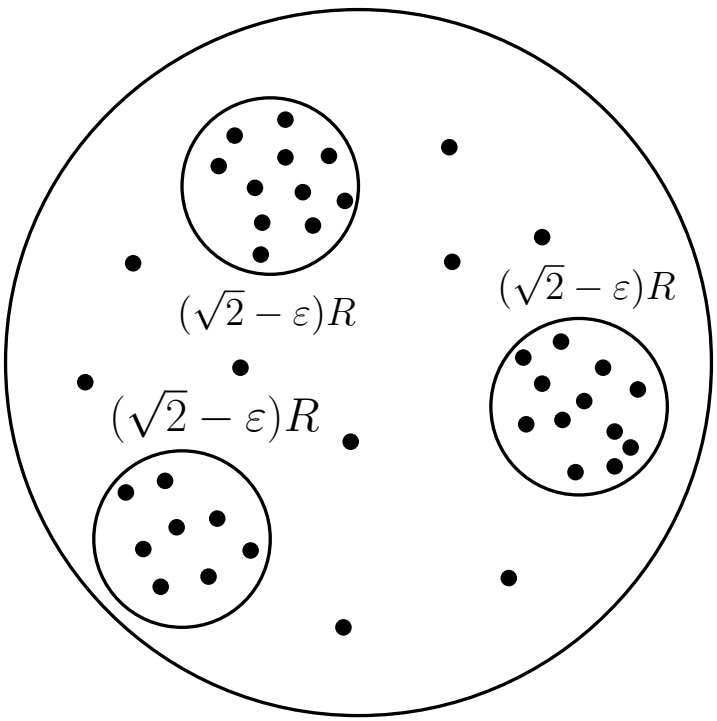
The general case

- Everything lies on a sphere of radius $R > cr/\sqrt{2}$



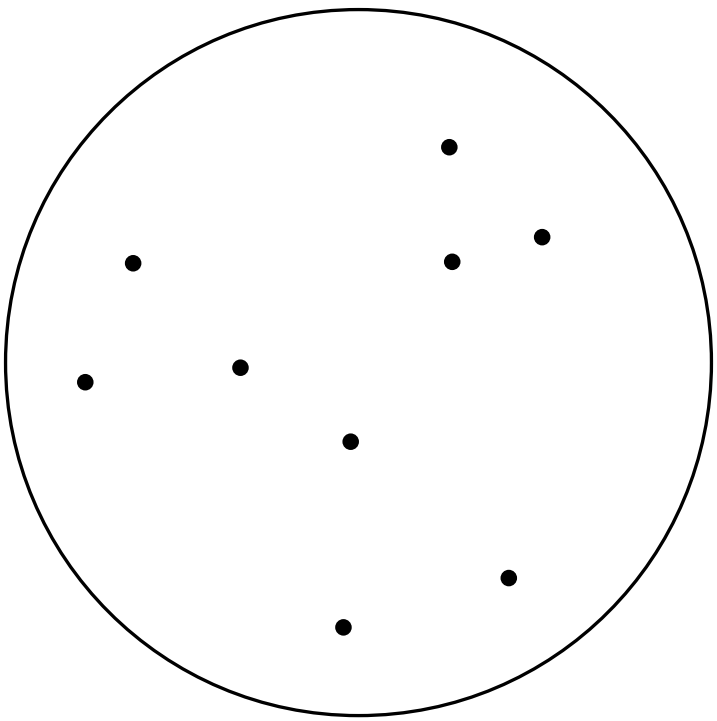
The general case

- Everything lies on a sphere of radius $R > cr/\sqrt{2}$
- Find dense clusters of radius $(\sqrt{2}-\varepsilon)R$ with centers on the sphere



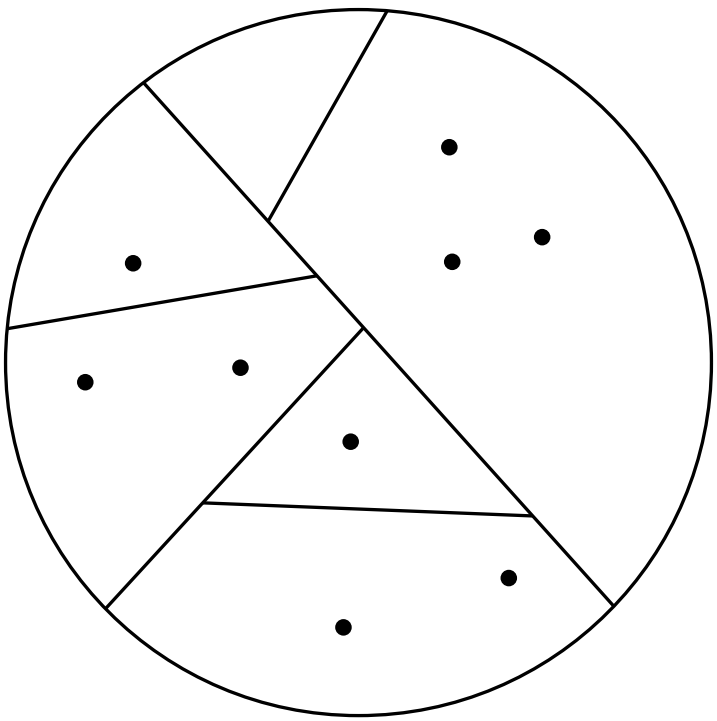
The general case

- Everything lies on a sphere of radius $R > cr/\sqrt{2}$
- Find dense clusters of radius $(\sqrt{2} - \varepsilon)R$ with centers on the sphere
- Treat them separately (see the next slide)



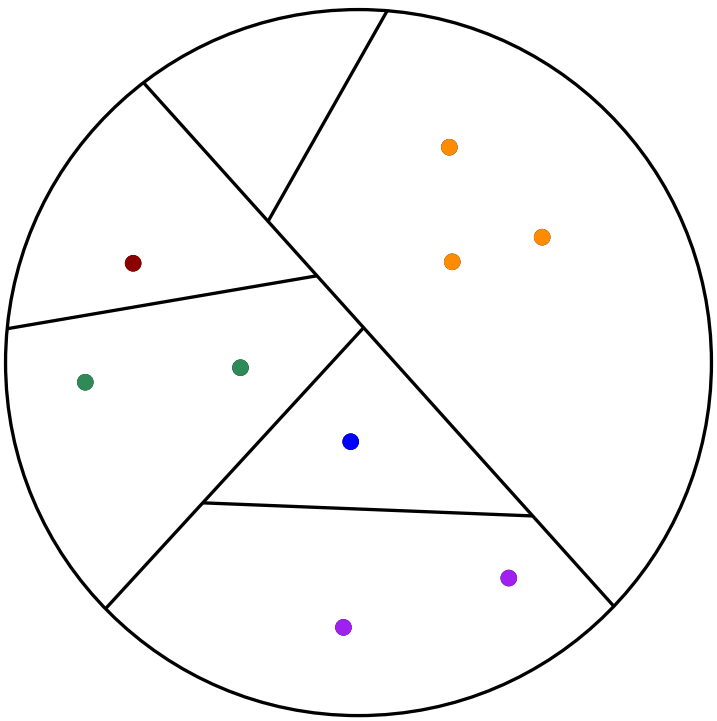
The general case

- Everything lies on a sphere of radius $R > cr/\sqrt{2}$
- Find dense clusters of radius $(\sqrt{2} - \varepsilon)R$ with centers on the sphere
- Treat them separately (see the next slide)
- For the remaining points, apply one iteration of ball carving



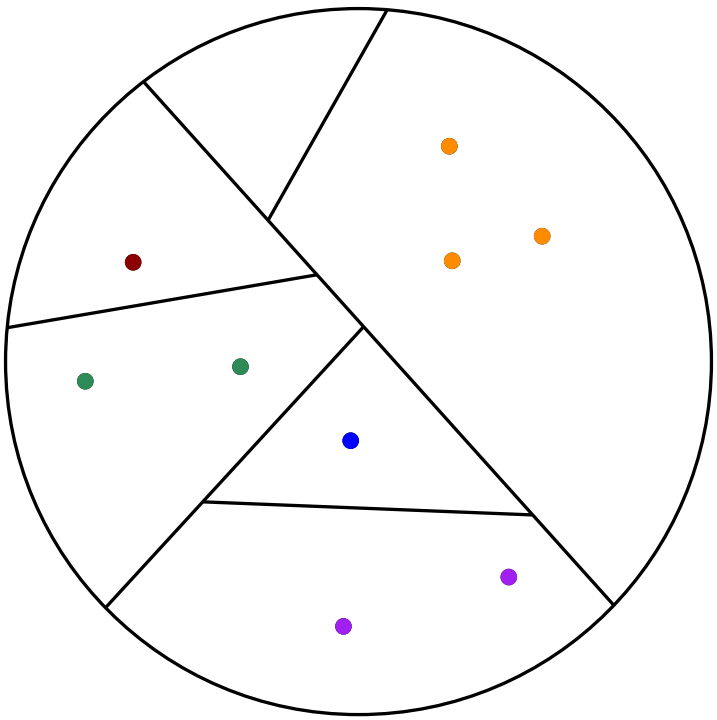
The general case

- Everything lies on a sphere of radius $R > cr/\sqrt{2}$
- Find dense clusters of radius $(\sqrt{2} - \varepsilon)R$ with centers on the sphere
- Treat them separately (see the next slide)
- For the remaining points, apply one iteration of ball carving
- Recurse on the parts (clusters may show up again)



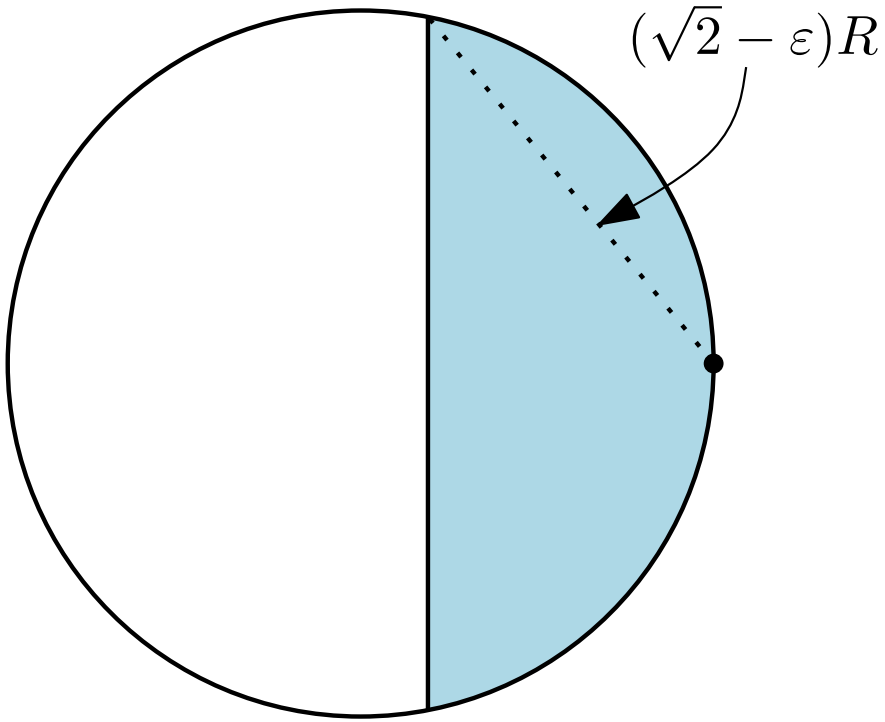
The general case

- Everything lies on a sphere of radius $R > cr/\sqrt{2}$
- Find dense clusters of radius $(\sqrt{2} - \varepsilon)R$ with centers on the sphere
- Treat them separately (see the next slide)
- For the remaining points, apply one iteration of ball carving
- Recurse on the parts (clusters may show up again)
- To answer a query q, query *all* the *dense* clusters, navigate q in the partition, and query the corresponding part *(only one!)*



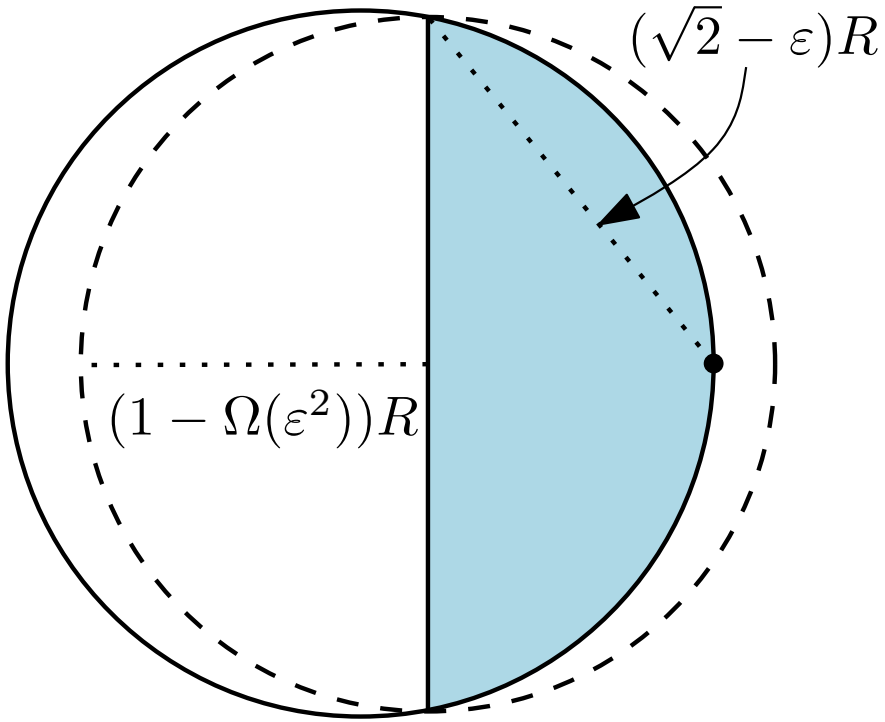
Treating dense clusters

- A cluster of radius $(\sqrt{2}-\varepsilon)R$ with center on the sphere can be covered with a ball of radius $(1-\Omega(\varepsilon^2))R$ (in spirit of the Ellipsoid Method)



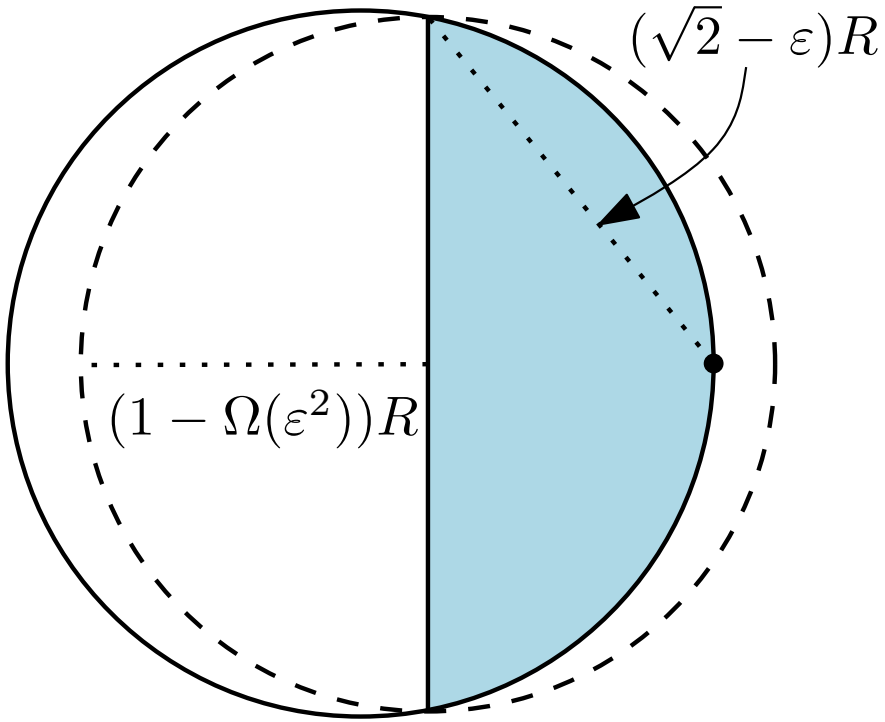
Treating dense clusters

- A cluster of radius $(\sqrt{2}-\varepsilon)R$ with center on the sphere can be covered with a ball of radius $(1-\Omega(\varepsilon^2))R$ (in spirit of the Ellipsoid Method)



Treating dense clusters

- A cluster of radius $(\sqrt{2}-\varepsilon)R$ with center on the sphere can be covered with a ball of radius $(1-\Omega(\varepsilon^2))R$ (in spirit of the Ellipsoid Method)
- Improve the geometry by reducing the radius



How do we make progress?

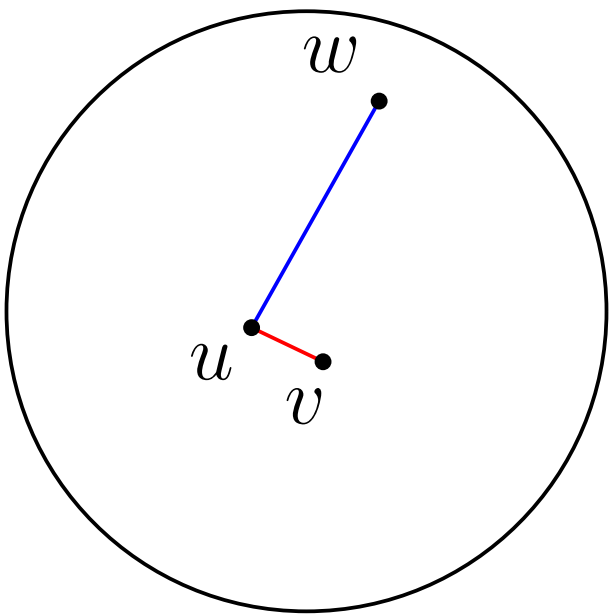
- For the clusters we decrease the radius non-trivially (by $1 - \Omega(\varepsilon^2)$)

How do we make progress?

- For the clusters we decrease the radius non-trivially (by $1 - \Omega(\varepsilon^2)$)
- For the remaining points, the ball carving is great: almost all the points are at distance $\geq (\sqrt{2} - \varepsilon)R$

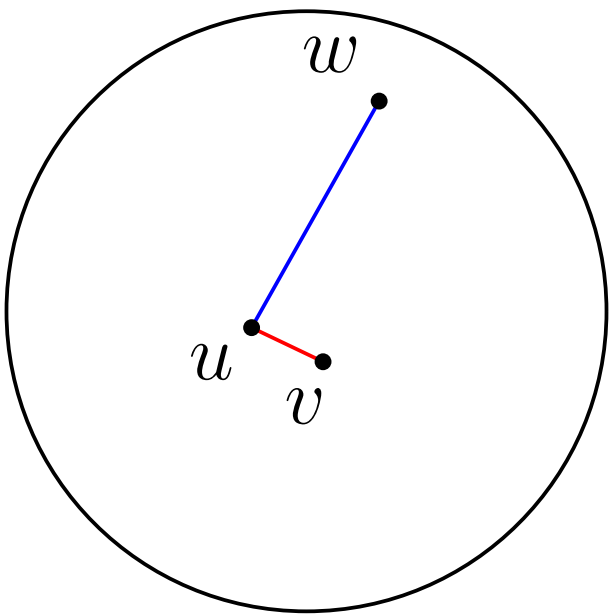
Ball carving for triples

- The analysis crucially relies on upper bounding $\Pr_{\mathcal{P}}[\mathcal{P}(u) = \mathcal{P}(w) \mid \mathcal{P}(u) = \mathcal{P}(v)]$



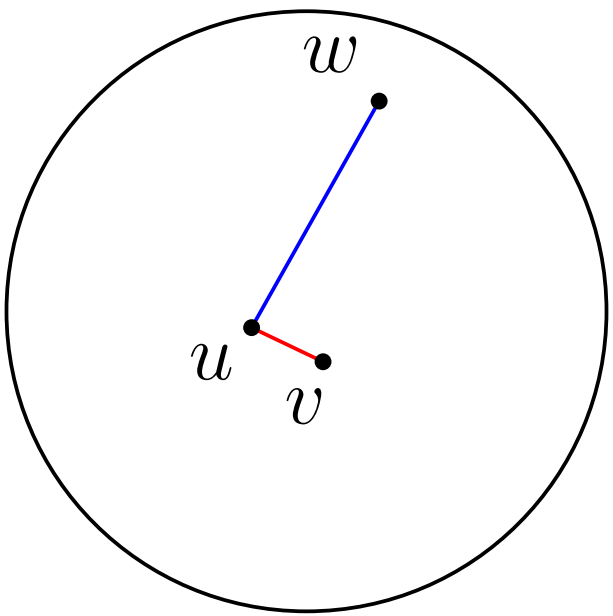
Ball carving for triples

- The analysis crucially relies on upper bounding $\Pr_{\mathcal{P}}[\mathcal{P}(u) = \mathcal{P}(w) \mid \mathcal{P}(u) = \mathcal{P}(v)]$
- The naïve bound $\frac{\Pr_{\mathcal{P}}[\mathcal{P}(u)=\mathcal{P}(w)]}{\Pr_{\mathcal{P}}[\mathcal{P}(u)=\mathcal{P}(v)]}$ merely gives the exponent $1/(2c^2 - 2) + o(1)$



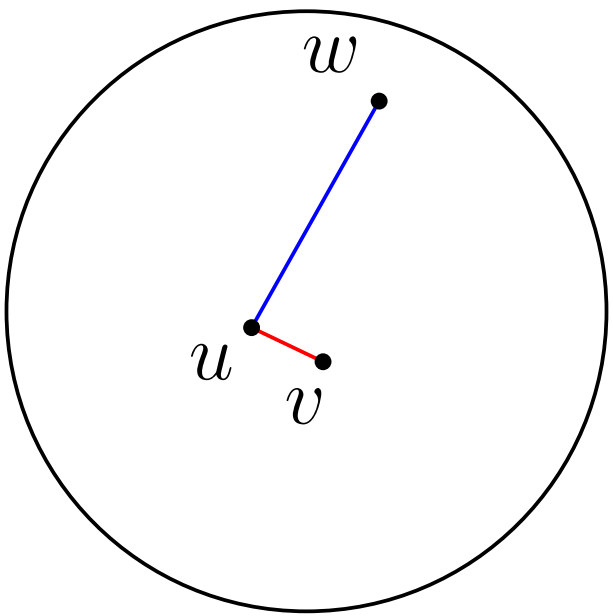
Ball carving for triples

- The analysis crucially relies on upper bounding $\Pr_{\mathcal{P}}[\mathcal{P}(u) = \mathcal{P}(w) \mid \mathcal{P}(u) = \mathcal{P}(v)]$
- The naïve bound $\frac{\Pr_{\mathcal{P}}[\mathcal{P}(u)=\mathcal{P}(w)]}{\Pr_{\mathcal{P}}[\mathcal{P}(u)=\mathcal{P}(v)]}$ merely gives the exponent $1/(2c^2 - 2) + o(1)$
- Unfortunately, tight for some cases



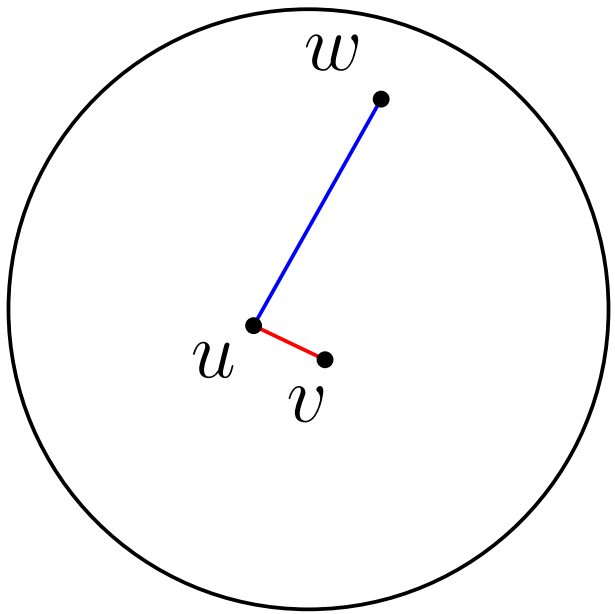
Ball carving for triples

- The analysis crucially relies on upper bounding $\Pr_{\mathcal{P}}[\mathcal{P}(u) = \mathcal{P}(w) \mid \mathcal{P}(u) = \mathcal{P}(v)]$
- The naïve bound $\frac{\Pr_{\mathcal{P}}[\mathcal{P}(u)=\mathcal{P}(w)]}{\Pr_{\mathcal{P}}[\mathcal{P}(u)=\mathcal{P}(v)]}$ merely gives the exponent $1/(2c^2 - 2) + o(1)$
- Unfortunately, tight for some cases
- Fortunately, for fixed u, v and *almost all* w can improve to essentially $\Pr_{\mathcal{P}}[\mathcal{P}(u) = \mathcal{P}(w)]$



Ball carving for triples

- The analysis crucially relies on upper bounding $\Pr_{\mathcal{P}}[\mathcal{P}(u) = \mathcal{P}(w) \mid \mathcal{P}(u) = \mathcal{P}(v)]$
- The naïve bound $\frac{\Pr_{\mathcal{P}}[\mathcal{P}(u)=\mathcal{P}(w)]}{\Pr_{\mathcal{P}}[\mathcal{P}(u)=\mathcal{P}(v)]}$ merely gives the exponent $1/(2c^2 - 2) + o(1)$
- Unfortunately, tight for some cases
- Fortunately, for fixed u, v and *almost all* w can improve to essentially $\Pr_{\mathcal{P}}[\mathcal{P}(u) = \mathcal{P}(w)]$
- No dense clusters $\Rightarrow$ can control the number of bad w's



Fast preprocessing

- Time $n^{2+o(1)}$ per decision tree: find only clusters with centers in data points (non-trivial analysis)

Fast preprocessing

- Time $n^{2+o(1)}$ per decision tree: find only clusters with centers in data points (non-trivial analysis)
- To achieve $n^{1+o(1)}$: observe that we only care about clusters with $n^{1-o(1)}$ points, thus, can subsample

Open problems

- Make the data structure dynamic

Open problems

- Make the data structure dynamic
- For ℓ_∞ the best data structure (Indyk 1998) is also based on decision trees; unify?

Open problems

- Make the data structure dynamic
- For ℓ_∞ the best data structure (Indyk 1998) is also based on decision trees; unify?
- *Questions?*