Proximity in the Age of ~~Destruction~~ Distraction: Robust Approximate Nearest Neighbor Search

Sariel Har-Peled[1] Sepideh Mahabadi[2]

[1]University of Illinois, Shampoo-Banana
[2]MIT

2: k robust nearest neighbor

ANN = Approximate nearest neighbor

$\mathbf{P} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\} \subseteq \mathbb{R}^{\boldsymbol{d}}$: Points in high dimensions.
$\boldsymbol{k}$: Parameter
Q: Given query $\boldsymbol{q}$, find ANN in $\mathbf{P}$, when ignoring $\boldsymbol{k}$ largest coordinates.

3: Robust distance

But without the noisy coordinates

Definition

$\boldsymbol{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$
$\pi = \mathrm{sort}(\boldsymbol{x})$: $|x_{\pi(1)}| \geq |x_{\pi(2)}| \geq \cdots \geq |x_{\pi(d)}|$.
i-tail of $\boldsymbol{x}$: $\boldsymbol{x}_{\setminus i} = \big(0, \ldots, 0, |x_{\pi(i+1)}|, |x_{\pi(i+2)}|, \ldots, |x_{\pi(d)}|\big)$.

computing $\boldsymbol{x}_{\setminus i}$ takes $O(d)$ time.

$x, y \in \mathsf{P}$: $\mathrm{dist}_{\setminus k}(x, y) = \|(x - y)_{\setminus k}\|$

Definition

k-robust nearest-neighbor to $q \in \mathbb{R}^d$ in P:

$$\mathrm{nn}_{\setminus k}(\mathsf{P}, q) = \arg\min_{y \in \mathsf{P}} \mathrm{dist}_{\setminus k}(q, y)$$

3: Robust distance

But without the noisy coordinates

Definition

$\boldsymbol{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$
$\pi = \mathrm{sort}(\boldsymbol{x})$: $|x_{\pi(1)}| \geq |x_{\pi(2)}| \geq \cdots \geq |x_{\pi(d)}|$.
i-tail of $\boldsymbol{x}$: $\boldsymbol{x}_{\setminus i} = \big(0, \ldots, 0, |x_{\pi(i+1)}|, |x_{\pi(i+2)}|, \ldots, |x_{\pi(d)}|\big)$.

computing $\boldsymbol{x}_{\setminus i}$ takes $O(d)$ time.

$\boldsymbol{x}, \boldsymbol{y} \in \mathsf{P}$: $\mathrm{dist}_{\setminus k}(\boldsymbol{x}, \boldsymbol{y}) = \big\|(\boldsymbol{x} - \boldsymbol{y})_{\setminus k}\big\|$

Definition

k-robust nearest-neighbor to $\boldsymbol{q} \in \mathbb{R}^d$ in P:

$$\mathrm{nn}_{\setminus k}(\mathsf{P}, \boldsymbol{q}) = \arg\min_{\boldsymbol{y} \in \mathsf{P}} \mathrm{dist}_{\setminus k}(\boldsymbol{q}, \boldsymbol{y})$$

3: Robust distance

But without the noisy coordinates

Definition

$\boldsymbol{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$
$\pi = \mathrm{sort}(\boldsymbol{x})$: $|x_{\pi(1)}| \geq |x_{\pi(2)}| \geq \cdots \geq |x_{\pi(d)}|$.
i-tail of $\boldsymbol{x}$: $\boldsymbol{x}_{\setminus i} = \big(0, \ldots, 0, |x_{\pi(i+1)}|, |x_{\pi(i+2)}|, \ldots, |x_{\pi(d)}|\big)$.

computing $\boldsymbol{x}_{\setminus i}$ takes $O(d)$ time.
$\boldsymbol{x}, \boldsymbol{y} \in \mathbf{P}$: $\mathrm{dist}_{\setminus k}(\boldsymbol{x}, \boldsymbol{y}) = \big\|(\boldsymbol{x} - \boldsymbol{y})_{\setminus k}\big\|$

Definition

k-robust nearest-neighbor to $\boldsymbol{q} \in \mathbb{R}^d$ in $\mathbf{P}$:

$$\mathbf{nn}_{\setminus k}(\mathbf{P}, \boldsymbol{q}) = \arg\min_{\boldsymbol{y} \in \mathbf{P}} \mathrm{dist}_{\setminus k}(\boldsymbol{q}, \boldsymbol{y})$$

4: Why robust NN?

1. High dim data common.
2. Some coordinates of $\boldsymbol{x}$ and $\boldsymbol{v}$ are irrelevant features.
3. Ignore noisy coordinates.

4: Why robust NN?

1. High dim data common.
2. Some coordinates of $\boldsymbol{x}$ and $\boldsymbol{v}$ are irrelevant features.
3. Ignore noisy coordinates.

4: Why robust NN?

1. High dim data common.
2. Some coordinates of $\boldsymbol{x}$ and $\boldsymbol{v}$ are irrelevant features.
3. Ignore noisy coordinates.

5: Results I

Reduction from Robust ANN to ANN.

1. General reduction from the robust ANN problem to the "standard" ANN problem.
2. L_1: $\exists q^* \in \mathbf{P}$ s.t. $\mathbf{dist}_{\setminus k}(q, q^*) \leq r \implies$
   1. $\delta \in (0,1)$: parameter
   2. return x s.t. $\mathbf{dist}_{\setminus O(k/\delta)}(q, x) \leq O(r/\delta)$
   3. $O(n^\delta)$ ANN queries in 2-ANN DS.
3. L_ρ: $\exists$NN with dist $\leq r$
   1. report point in distance $O(r(c + \frac{1}{\delta})^{1/\rho})$
   2. ignoring $O(k(\frac{1}{\delta} + c))$ coordinates
   3. n^δ of $c^{1/\rho}$-ANN queries.

5: Results I

Reduction from Robust ANN to ANN.

1. General reduction from the robust ANN problem to the "standard" ANN problem.
2. L_1: $\exists q^* \in P$ s.t. $\mathrm{dist}_{\setminus k}(q, q^*) \leq r \implies$
   1. $\delta \in (0,1)$: parameter
   2. return x s.t. $\mathrm{dist}_{\setminus O(k/\delta)}(q, x) \leq O(r/\delta)$
   3. $O(n^\delta)$ ANN queries in 2-ANN DS.
3. L_ρ: $\exists$NN with dist $\leq r$
   1. report point in distance $O(r(c + \frac{1}{\delta})^{1/\rho})$
   2. ignoring $O(k(\frac{1}{\delta} + c))$ coordinates
   3. n^δ of $c^{1/\rho}$-ANN queries.

5: Results I

Reduction from Robust ANN to ANN.

1. General reduction from the robust ANN problem to the "standard" ANN problem.
2. L_1: $\exists q^* \in P$ s.t. $\mathrm{dist}_{\setminus k}(q, q^*) \leq r \implies$
   1. $\delta \in (0, 1)$: parameter
   2. return x s.t. $\mathrm{dist}_{\setminus O(k/\delta)}(q, x) \leq O(r/\delta)$
   3. $O(n^\delta)$ ANN queries in 2-ANN DS.
3. L_ρ: $\exists$NN with dist $\leq r$
   1. report point in distance $O(r(c + \frac{1}{\delta})^{1/\rho})$
   2. ignoring $O(k(\frac{1}{\delta} + c))$ coordinates
   3. n^δ of $c^{1/\rho}$-ANN queries.

5: Results I

Reduction from Robust ANN to ANN.

1. General reduction from the robust ANN problem to the "standard" ANN problem.
2. L_1: $\exists q^* \in P$ s.t. $\mathbf{dist}_{\setminus k}(q, q^*) \leq r \implies$
   1. $\delta \in (0, 1)$: parameter
   2. return x s.t. $\mathbf{dist}_{\setminus O(k/\delta)}(q, x) \leq O(r/\delta)$
   3. $O(n^\delta)$ ANN queries in 2-ANN DS.
3. L_ρ: $\exists$NN with dist $\leq r$
   1. report point in distance $O(r(c + \frac{1}{\delta})^{1/\rho})$
   2. ignoring $O(k(\frac{1}{\delta} + c))$ coordinates
   3. n^δ of $c^{1/\rho}$-ANN queries.

5: Results I

Reduction from Robust ANN to ANN.

1. General reduction from the robust ANN problem to the "standard" ANN problem.
2. L_1: $\exists q^* \in P$ s.t. $\mathbf{dist}_{\setminus k}(q, q^*) \leq r \implies$
   1. $\delta \in (0, 1)$: parameter
   2. return x s.t. $\mathbf{dist}_{\setminus O(k/\delta)}(q, x) \leq O(r/\delta)$
   3. $O(n^\delta)$ ANN queries in 2-ANN DS.
3. L_ρ: $\exists$NN with dist $\leq r$
   1. report point in distance $O(r(c + \frac{1}{\delta})^{1/\rho})$
   2. ignoring $O(k(\frac{1}{\delta} + c))$ coordinates
   3. n^δ of $c^{1/\rho}$-ANN queries.

5: Results I

Reduction from Robust ANN to ANN.

1. General reduction from the robust ANN problem to the "standard" ANN problem.
2. L_1: $\exists q^* \in \mathbf{P}$ s.t. $\mathbf{dist}_{\setminus k}(q, q^*) \leq r \implies$
   1. $\delta \in (0, 1)$: parameter
   2. return x s.t. $\mathbf{dist}_{\setminus O(k/\delta)}(q, x) \leq O(r/\delta)$
   3. $O(n^\delta)$ ANN queries in 2-ANN DS.
3. L_ρ: $\exists$NN with dist $\leq r$
   1. report point in distance $O(r(c + \frac{1}{\delta})^{1/\rho})$
   2. ignoring $O(k(\frac{1}{\delta} + c))$ coordinates
   3. n^δ of $c^{1/\rho}$-ANN queries.

6: Results II

$(1+\varepsilon)$-approximation

L_1: $\exists q^* \in \mathbf{P}$ s.t. $\mathbf{dist}_{\parallel k}(q, q^*) \leq r \implies$

1. $\varepsilon, \delta \in (0,1)$: parameters
2. return x s.t. $\mathbf{dist}_{\parallel O(\frac{k}{\delta\varepsilon})}(q, x) \leq (1+\varepsilon)r$
3. $\widetilde{O}(n^{\delta}/\varepsilon)$ ANN queries in $(1+O(\varepsilon))$-ANN DS.

6: Results II

$(1+\varepsilon)$-approximation

L_1: $\exists q^* \in \mathbf{P}$ s.t. $\mathbf{dist}_{\setminus k}(q, q^*) \leq r \implies$

1. $\varepsilon, \delta \in (0,1)$: parameters
2. return x s.t. $\mathbf{dist}_{\setminus O(\frac{k}{\delta\varepsilon})}(q, x) \leq (1+\varepsilon)r$
3. $\widetilde{O}(n^{\delta}/\varepsilon)$ ANN queries in $(1+O(\varepsilon))$-ANN DS.

6: Results II

$(1+\varepsilon)$-approximation

L_1: $\exists q^* \in \mathbf{P}$ s.t. $\mathbf{dist}_{\setminus\setminus k}(q, q^*) \leq r \implies$

1. $\varepsilon, \delta \in (0,1)$: parameters
2. return x s.t. $\mathbf{dist}_{\setminus\setminus O(\frac{k}{\delta\varepsilon})}(q, x) \leq (1+\varepsilon)r$
3. $\widetilde{O}(n^{\delta}/\varepsilon)$ ANN queries in $(1+O(\varepsilon))$-ANN DS.

6: Results II

$(1+\varepsilon)$-approximation

L_1: $\exists \boldsymbol{q}^* \in \mathbf{P}$ s.t. $\mathbf{dist}_{\backslash\backslash k}(\boldsymbol{q}, \boldsymbol{q}^*) \leq r \implies$

1. $\varepsilon, \delta \in (0,1)$: parameters
2. return $\boldsymbol{x}$ s.t. $\mathbf{dist}_{\backslash\backslash O(\frac{k}{\delta\varepsilon})}(\boldsymbol{q}, \boldsymbol{x}) \leq (1+\varepsilon)r$
3. $\widetilde{O}(n^{\delta}/\varepsilon)$ ANN queries in $(1+O(\varepsilon))$-ANN DS.

7: Results III

Budgeted version

1. Every coordinate has a cost of ignoring it.
2. $\exists \boldsymbol{q}^* \in \mathbf{P}$, s.t., $\mathbf{dist}_{\|_1}(\boldsymbol{q}, \boldsymbol{q}^*) \leq r \implies$
3. Algorithm returns a point in distance $O(r)$
4. Cost of ignored coordinates is $O(1)$.
5. in L_1 norm.
6. Computing exact distance is NP-Hard even for two points.

7: Results III

Budgeted version

1. Every coordinate has a cost of ignoring it.
2. $\exists \boldsymbol{q}^* \in \mathbf{P}$, s.t., $\mathbf{dist}_{\setminus 1}(\boldsymbol{q}, \boldsymbol{q}^*) \leq r \implies$
3. Algorithm returns a point in distance $O(r)$
4. Cost of ignored coordinates is $O(1)$.
5. in L_1 norm.
6. Computing exact distance is NP-Hard even for two points.

8: Results III

Data sensitive LSH queries

1. known ANN DS perform better in practice.
   1. Many cases return *exact* NN!
   2. Find it early!
   3. Spend more time "proving" its ANN.
2. **Q:** If query is easy, why work hard?
3. **Result:** (for $\{0,1\}^d$).
   1. query time: $O(d \exp(\Delta) \log n)$
   2. Δ is smallest value s.t.

$$\sum_{i=1}^{n} \exp\left(-\Delta \frac{\mathrm{dist}(q, x_i)}{r}\right) \leq 1,$$

   3. works quickly on low dimensional data.

8: Results III

Data sensitive LSH queries

1. known ANN DS perform better in practice.
   1. Many cases return *exact* NN!
   2. Find it early!
   3. Spend more time "proving" its ANN.
2. **Q:** If query is easy, why work hard?
3. **Result:** (for $\{0,1\}^d$).
   1. query time: $O(d \exp(\Delta) \log n)$
   2. Δ is smallest value s.t.

$$\sum_{i=1}^{n} \exp\left(-\Delta \frac{\mathrm{dist}(q, x_i)}{r}\right) \leq 1,$$

   3. works quickly on low dimensional data.

8: Results III

Data sensitive LSH queries

1. known ANN DS perform better in practice.
   1. Many cases return *exact* NN!
   2. Find it early!
   3. Spend more time "proving" its ANN.
2. **Q:** If query is easy, why work hard?
3. **Result:** (for $\{0,1\}^d$).
   1. query time: $O(d \exp(\Delta) \log n)$
   2. Δ is smallest value s.t.

$$\sum_{i=1}^{n} \exp\left(-\Delta \frac{\mathrm{dist}(q, x_i)}{r}\right) \leq 1,$$

   3. works quickly on low dimensional data.

8: Results III

Data sensitive LSH queries

1. known ANN DS perform better in practice.
   1. Many cases return *exact* NN!
   2. Find it early!
   3. Spend more time "proving" its ANN.
2. **Q:** If query is easy, why work hard?
3. **Result:** (for $\{0,1\}^d$).
   1. query time: $O(d \exp(\Delta) \log n)$
   2. Δ is smallest value s.t.

$$\sum_{i=1}^{n} \exp\left(-\Delta \frac{\mathrm{dist}(q, x_i)}{r}\right) \leq 1,$$

   3. works quickly on low dimensional data.

8: Results III

Data sensitive LSH queries

1. known ANN DS perform better in practice.
   1. Many cases return *exact* NN!
   2. Find it early!
   3. Spend more time "proving" its ANN.
2. **Q:** If query is easy, why work hard?
3. **Result:** (for $\{0,1\}^d$).
   1. query time: $O(d \exp(\Delta) \log n)$
   2. Δ is smallest value s.t.

$$\sum_{i=1}^{n} \exp\left(-\Delta \frac{\mathrm{dist}(q, x_i)}{r}\right) \leq 1,$$

   3. works quickly on low dimensional data.

8: Results III

Data sensitive LSH queries

1. known ANN DS perform better in practice.
   1. Many cases return *exact* NN!
   2. Find it early!
   3. Spend more time "proving" its ANN.
2. **Q:** If query is easy, why work hard?
3. **Result:** (for $\{0,1\}^d$).
   1. query time: $O(d \exp(\Delta) \log n)$
   2. Δ is smallest value s.t.

$$\sum_{i=1}^{n} \exp\left(-\Delta \frac{\mathrm{dist}(q, x_i)}{r}\right) \leq 1,$$

   3. works quickly on low dimensional data.

8: Results III

Data sensitive LSH queries

1. known ANN DS perform better in practice.
   1. Many cases return *exact* NN!
   2. Find it early!
   3. Spend more time "proving" its ANN.
2. **Q:** If query is easy, why work hard?
3. **Result:** (for $\{0,1\}^d$).
   1. query time: $O(d \exp(\Delta) \log n)$
   2. Δ is smallest value s.t.

$$\sum_{i=1}^{n} \exp\left(-\Delta \frac{\mathrm{dist}(q, x_i)}{r}\right) \leq 1,$$

   3. works quickly on low dimensional data.

Part I

Locality sensitive hashing revisited

9: Locality sensitive hashing (LSH)

1. Technique due to Indyk and Motwani **[Indyk and Motwani, 1998]**.
2. Input: $\mathbf{P} \subseteq \{0,1\}^d$: set of n points.
3. Parameters r and ε.
4. **Q:** Given query point $q \in \{0,1\}^d$:
   1. If $\mathbf{d}_H(q, \mathbf{P}) \leq r$ then return $v \in \mathbf{P}$ s.t. $\mathbf{d}_H(q, v) \leq (1+\varepsilon)r$.
   2. If $\mathbf{d}_H(q, \mathbf{P}) > (1+\varepsilon)r$ then return "$\mathbf{d}_H(q, \mathbf{P}) > r$".
   3. If $\mathbf{d}_H(q, \mathbf{P}) \in \big(r, (1+\varepsilon)r\big)$ return either of the above.

10: A random projection...

1. Sequence $m \equiv i_1, i_2, \ldots, i_\ell \in [\![d]\!] = \{1, \ldots, d\}$.
2. $x = (x_1, \ldots, x_d) \in \mathbb{R}^d$
3. **projection** by m is

$$mx = (x_{i_1}, \ldots, x_{i_\ell}) \in \mathbb{R}^\ell.$$

4. $mP = \{mx \mid x \in P\}$.

10: A random projection...

1. Sequence $m \equiv i_1, i_2, \ldots, i_\ell \in [\![d]\!] = \{1, \ldots, d\}$.
2. $\boldsymbol{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$
3. **projection** by m is

$$m\boldsymbol{x} = (x_{i_1}, \ldots, x_{i_\ell}) \in \mathbb{R}^\ell.$$

4. $m\mathbf{P} = \{m\boldsymbol{x} \mid \boldsymbol{x} \in \mathbf{P}\}$.

10: A random projection...

1. Sequence $m \equiv i_1, i_2, \ldots, i_\ell \in [\![d]\!] = \{1, \ldots, d\}$.
2. $\boldsymbol{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$
3. **projection** by m is

$$m\boldsymbol{x} = (x_{i_1}, \ldots, x_{i_\ell}) \in \mathbb{R}^\ell.$$

4. $m\mathbf{P} = \{m\boldsymbol{x} \mid \boldsymbol{x} \in \mathbf{P}\}$.

10: A random projection...

1. Sequence $m \equiv i_1, i_2, \ldots, i_\ell \in [\![d]\!] = \{1, \ldots, d\}$.
2. $\boldsymbol{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$
3. **projection** by m is

$$m\boldsymbol{x} = (x_{i_1}, \ldots, x_{i_\ell}) \in \mathbb{R}^\ell.$$

4. $m\mathbf{P} = \{m\boldsymbol{x} \mid \boldsymbol{x} \in \mathbf{P}\}$.

11: A splay

1. τ: a probability
2. include ith coordinate, for $i = 1, \ldots, d$, in m with probability τ
3. $\mathcal{D}_\tau$: Distribution of such projections.
4. $m = i_1, \ldots, i_\ell$ and $n = j_1, \ldots, j_{\ell'}$:
   $m|n = i_1, \ldots, i_\ell, j_1, \ldots, j_{\ell'}$.
5. $\mathcal{D}_\tau^t$: distribution of sequences concatenating t sequences of $\mathcal{D}_\tau$.

11: A splay

1. τ: a probability
2. include ith coordinate, for $i = 1, \ldots, d$, in m with probability τ
3. $\mathcal{D}_\tau$: Distribution of such projections.
4. $m = i_1, \ldots, i_\ell$ and $n = j_1, \ldots, j_{\ell'}$:
   $m|n = i_1, \ldots, i_\ell, j_1, \ldots, j_{\ell'}$.
5. $\mathcal{D}_\tau^t$: distribution of sequences concatenating t sequences of $\mathcal{D}_\tau$.

11: A splay

1. τ: a probability
2. include ith coordinate, for $i = 1, \ldots, d$, in m with probability τ
3. $\mathcal{D}_\tau$: Distribution of such projections.
4. $m = i_1, \ldots, i_\ell$ and $n = j_1, \ldots, j_{\ell'}$:
   $m|n = i_1, \ldots, i_\ell, j_1, \ldots, j_{\ell'}$.
5. $\mathcal{D}_\tau^t$: distribution of sequences concatenating t sequences of $\mathcal{D}_\tau$.

11: A splay

1. τ: a probability
2. include ith coordinate, for $i = 1, \ldots, d$, in m with probability τ
3. $\mathcal{D}_\tau$: Distribution of such projections.
4. $m = i_1, \ldots, i_\ell$ and $n = j_1, \ldots, j_{\ell'}$:
   $m|n = i_1, \ldots, i_\ell, j_1, \ldots, j_{\ell'}$.
5. $\mathcal{D}_\tau^t$: distribution of sequences concatenating t sequences of $\mathcal{D}_\tau$.

12: Preprocessing

1. Set $\beta = 1/(1+\varepsilon) \in (0,1)$.
2. $\tau = 1/r, \quad t = \beta \log n, \quad$ and $L = O(n^{\beta} \log n)$.
3. Pick randomly L sequences $M_1, \ldots, M_L \in \mathcal{D}_{\tau}^{t}$.
4. Algorithm computes $\mathbf{Q}_i = M_i \mathbf{P}_i$, for $i = 1, \ldots, L$.
5. Store $\mathbf{Q}_i$ in a hash table $\mathcal{H}_i$ for $i = 1, \ldots, L$.
6. **Answering a query.**
   1. Given $q \in \{0,1\}^d$: compute $q_i = M_i q$, for $i = 1, \ldots, L$.
   2. Retrieve list ℓ_i from $\mathcal{H}_i$, of all points that collide with q_i.
   3. Scan $\ell_1, \ldots, \ell_L$.
      If any point is in dist $\leq (1+\varepsilon)r$, return it.
   4. Otherwise, return "$\mathbf{d}_H(q, \mathbf{P}) > r$".

12: Preprocessing

1. Set $\beta = 1/(1+\varepsilon) \in (0,1)$.
2. $\tau = 1/r, \quad t = \beta \log n, \quad$ and $L = O(n^{\beta} \log n)$.
3. Pick randomly L sequences $M_1, \ldots, M_L \in \mathcal{D}_{\tau}^{t}$.
4. Algorithm computes $\mathbf{Q}_i = M_i \mathbf{P}_i$, for $i = 1, \ldots, L$.
5. Store $\mathbf{Q}_i$ in a hash table $\mathcal{H}_i$ for $i = 1, \ldots, L$.

---

6. **Answering a query.**
   1. Given $q \in \{0,1\}^d$: compute $q_i = M_i q$, for $i = 1, \ldots, L$.
   2. Retrieve list ℓ_i from $\mathcal{H}_i$, of all points that collide with q_i.
   3. Scan $\ell_1, \ldots, \ell_L$.
      If any point is in dist $\leq (1+\varepsilon)r$, return it.
   4. Otherwise, return "$\mathbf{d}_H(q, \mathbf{P}) > r$".

12: Preprocessing

1. Set $\beta = 1/(1+\varepsilon) \in (0,1)$.
2. $\tau = 1/r, \quad t = \beta \log n, \quad$ and $L = O(n^{\beta} \log n)$.
3. Pick randomly L sequences $M_1, \ldots, M_L \in \mathcal{D}_{\tau}^{t}$.
4. Algorithm computes $\mathbf{Q}_i = M_i \mathbf{P}_i$, for $i = 1, \ldots, L$.
5. Store $\mathbf{Q}_i$ in a hash table $\mathcal{H}_i$ for $i = 1, \ldots, L$.

---

6. **Answering a query.**
   1. Given $\boldsymbol{q} \in \{0,1\}^d$: compute $\boldsymbol{q}_i = M_i \boldsymbol{q}$, for $i = 1, \ldots, L$.
   2. Retrieve list ℓ_i from $\mathcal{H}_i$, of all points that collide with $\boldsymbol{q}_i$.
   3. Scan $\ell_1, \ldots, \ell_L$.
      If any point is in dist $\leq (1+\varepsilon)r$, return it.
   4. Otherwise, return "$\mathbf{d}_H(\boldsymbol{q}, \mathbf{P}) > r$".

13: What is going on?

Collision probabilities

$\boldsymbol{q}$, $\boldsymbol{y}$: Two points.

$\varepsilon = 1 \qquad x = \mathbf{d}_H(\boldsymbol{q}, \boldsymbol{y}). \qquad t = \frac{\log n}{1+\varepsilon}$

$$f(x) = \mathbf{Pr}[M\boldsymbol{q} = M\boldsymbol{x}] = \left(\left(1 - \frac{1}{r}\right)^x\right)^t \approx \mathbf{exp}\left(-\frac{xt}{r}\right)$$

1

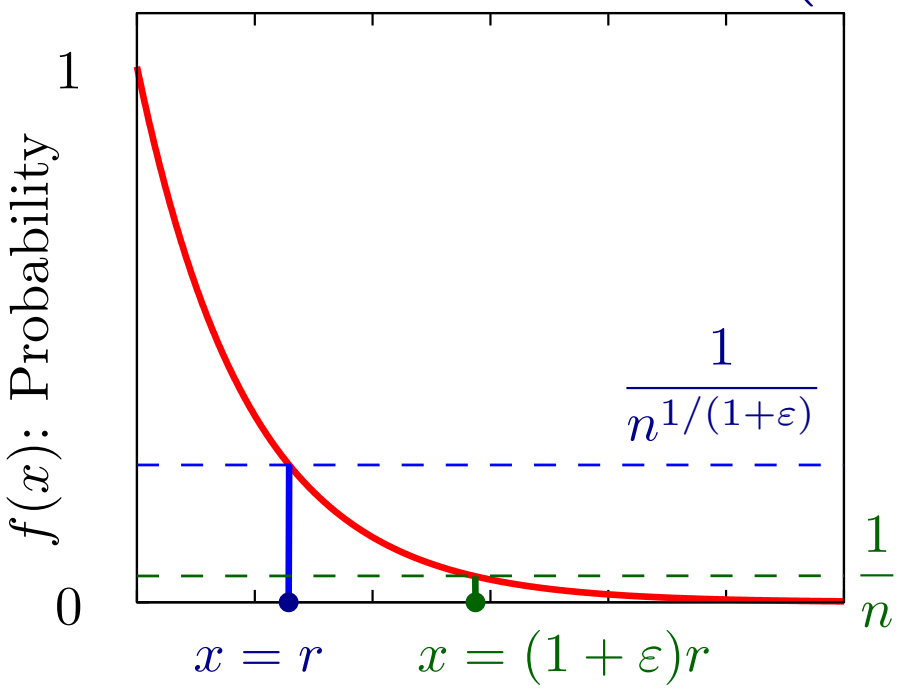
13: What is going on?

Collision probabilities

$\boldsymbol{q}, \boldsymbol{y}$: Two points.

$\varepsilon = 1 \qquad x = \mathbf{d}_H(\boldsymbol{q}, \boldsymbol{y}). \qquad t = \frac{\log n}{1+\varepsilon}$

$$f(x) = \mathbf{Pr}[M\boldsymbol{q} = M\boldsymbol{x}] = \left(\left(1 - \frac{1}{r}\right)^x\right)^t \approx \exp\left(-\frac{xt}{r}\right)$$



1

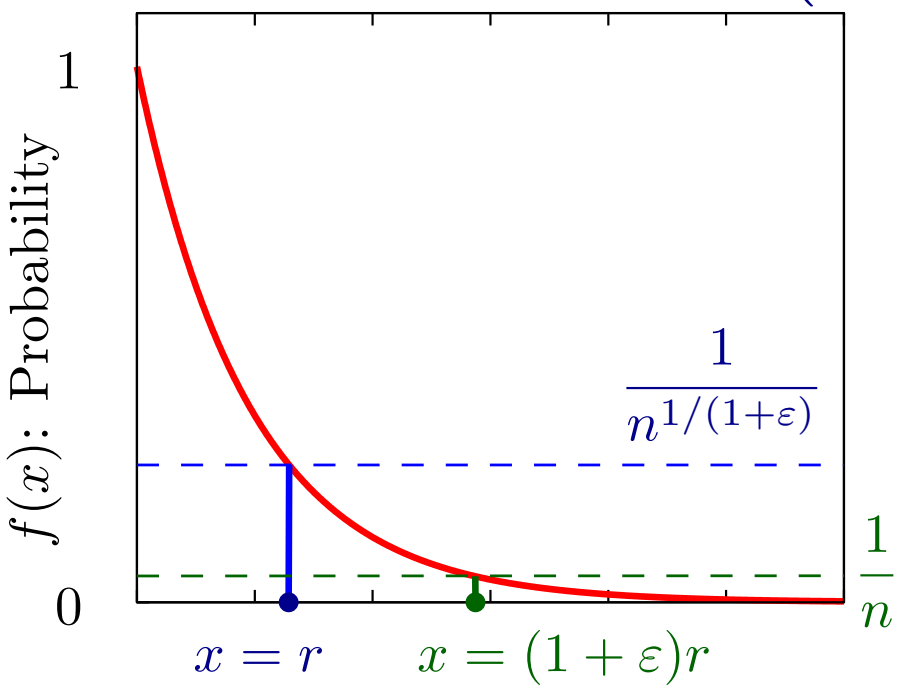
13: What is going on?

Collision probabilities

$\boldsymbol{q}, \boldsymbol{y}$: Two points.

$\varepsilon = 1 \qquad x = \mathbf{d}_H(\boldsymbol{q}, \boldsymbol{y}). \qquad t = \frac{\log n}{1+\varepsilon}$

$$f(x) = \mathbf{Pr}[M\boldsymbol{q} = M\boldsymbol{x}] = \left(\left(1 - \frac{1}{r}\right)^x\right)^t \approx \exp\left(-\frac{xt}{r}\right)$$



1. $f(x) = \exp\left(-\frac{x \log n}{(1+\varepsilon)r}\right)$

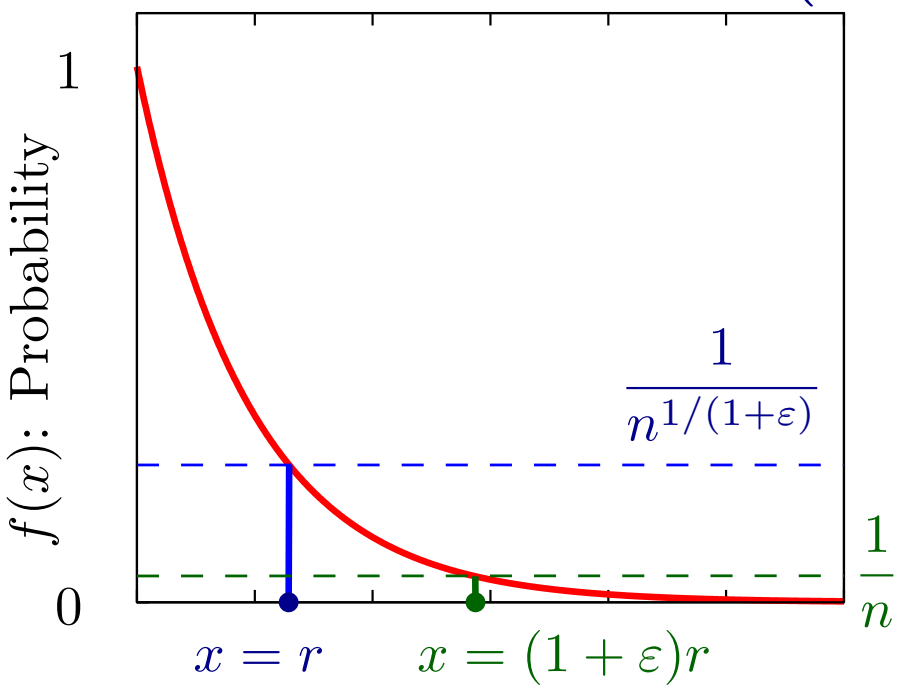
13: What is going on?

Collision probabilities

$\boldsymbol{q}, \boldsymbol{y}$: Two points.

$\varepsilon = 1 \qquad x = \mathbf{d}_H(\boldsymbol{q}, \boldsymbol{y}). \qquad t = \frac{\log n}{1+\varepsilon}$

$$f(x) = \mathbf{Pr}[M\boldsymbol{q} = M\boldsymbol{x}] = \left(\left(1 - \frac{1}{r}\right)^x\right)^t \approx \exp\left(-\frac{xt}{r}\right)$$



1. $f(x) = \exp\left(-\frac{x \log n}{(1+\varepsilon)r}\right)$
2. $f(r) = 1/n^{1/(1+\varepsilon)}$

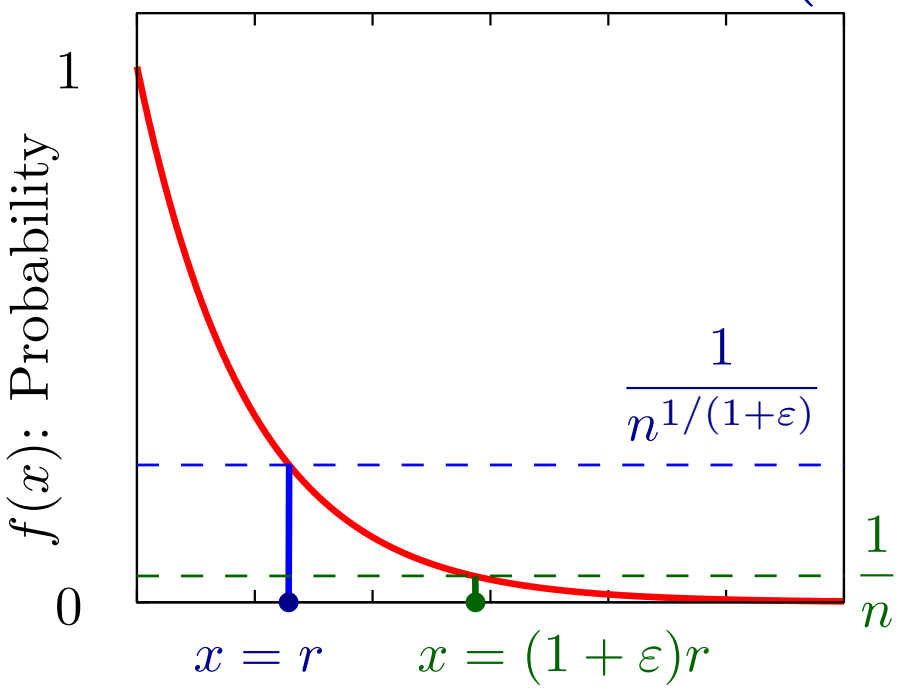
13: What is going on?

Collision probabilities

$\boldsymbol{q}$, $\boldsymbol{y}$: Two points.

$\varepsilon = 1$ $\quad x = \mathbf{d}_H(\boldsymbol{q}, \boldsymbol{y})$. $\quad t = \frac{\log n}{1+\varepsilon}$

$$f(x) = \mathbf{Pr}[M\boldsymbol{q} = M\boldsymbol{x}] = \left(\left(1 - \frac{1}{r}\right)^x\right)^t \approx \exp\left(-\frac{xt}{r}\right)$$



1. $f(x) = \exp\left(-\frac{x \log n}{(1+\varepsilon)r}\right)$
2. $f(r) = 1/n^{1/(1+\varepsilon)}$
3. $f\Big((1+\varepsilon)r\Big) = 1/n$

14: Correctness – avoiding few bad coordinates

Lemma

K: set of r forbidden coordinates. Probability $M = (m_1, \ldots, m_t) \in \mathcal{D}_\tau^t$ does not sample any coordinate of K is $\geq n^{-\beta(1+1/(2r))} \approx 1/n^\beta = 1/n^{1/(1+\varepsilon)}$.

Proof.

Probability m_i not contain any of coordinates of K is $(1-\tau)^r$.
Desired probability is

$$\left(1-\frac{1}{r}\right)^{rt} = \left(1-\frac{1}{r}\right)^{r\beta \log n} \geq \exp(-\beta \log n)\left(1-\frac{1}{r}\right)^{\beta \log n}$$
$$\geq n^{-\beta} \exp\left(-\frac{1}{2r}\beta \log n\right) = n^{-\beta-\beta/(2r)},$$

by $(1-1/m)^{m-1} \geq \frac{1}{e} \geq (1-1/m)^m$, $e^{-2x} \leq 1-x \leq e^x$. □

15: Proof

Lemma

(A) *If* $\mathbf{d}_H(q, \mathbf{P}) \leq r$. *W.h.p. return* $x \in \mathbf{P}$ *s.t.* $\mathbf{d}_H(q, x) \leq (1+\varepsilon)r$.

(B) *If* $\mathbf{d}_H(q, \mathbf{P}) \geq (1+\varepsilon)r$. *Then,* $\mathbf{E}[\sum_i |\ell_i|] = O(L)$.

Proof.

(A) By previous lemma. Probability $M_i x = M_i q$ is at least $1/n^\beta$. Probability all L data-structures fails, is $\leq (1 - 1/n^\beta)^L < 1/n^{O(1)}$, as $L = O(n^\beta \log n)$.

(B) For $v \in \mathbf{P}$, s.t. $\mathbf{d}_H(v, q) \geq (1+\varepsilon)r$. Probability for $M \in \mathcal{D}_\tau^t$ misses all the $\geq (1+\varepsilon)r$ coordinates (i.e., collision!) is

$$\leq (1-\tau)^{(1+\varepsilon)rt} \leq \exp\left(-(1+\varepsilon)\frac{tr}{r}\right) =$$

$\exp(-(1+\varepsilon)\beta \log n) = 1/n$, as $\beta = 1/(1+\varepsilon)$. Expected size of ℓ_i is $\leq n \times (1/n) = 1$. □

15: Proof

Lemma

(A) *If* $\mathbf{d}_H(\boldsymbol{q}, \mathbf{P}) \leq \boldsymbol{r}$. *W.h.p. return* $\boldsymbol{x} \in \mathbf{P}$ *s.t.* $\mathbf{d}_H(\boldsymbol{q}, \boldsymbol{x}) \leq (1+\varepsilon)\boldsymbol{r}$.

(B) *If* $\mathbf{d}_H(\boldsymbol{q}, \mathbf{P}) \geq (1+\varepsilon)\boldsymbol{r}$. *Then,* $\mathbf{E}[\sum_i |\ell_i|] = O(L)$.

Proof.

(A) By previous lemma. Probability $M_i \boldsymbol{x} = M_i \boldsymbol{q}$ is at least $1/n^{\beta}$. Probability all L data-structures fails, is $\leq (1-1/n^{\beta})^L < 1/n^{O(1)}$, as $L = O(n^{\beta} \log n)$.

(B) For $\boldsymbol{v} \in \mathbf{P}$, s.t. $\mathbf{d}_H(\boldsymbol{v}, \boldsymbol{q}) \geq (1+\varepsilon)\boldsymbol{r}$. Probability for $M \in \mathcal{D}_{\tau}^{t}$ misses all the $\geq (1+\varepsilon)\boldsymbol{r}$ coordinates (i.e., collision!) is

$$\leq (1-\tau)^{(1+\varepsilon)rt} \leq \exp\left(-(1+\varepsilon)\frac{tr}{r}\right) =$$

$\exp(-(1+\varepsilon)\beta \log n) = 1/n$, as $\beta = 1/(1+\varepsilon)$. Expected size of ℓ_i is $\leq n \times (1/n) = 1$. □

15: Proof

Lemma

(A) *If* $\mathbf{d}_H(q, \mathbf{P}) \leq r$. *W.h.p. return* $x \in \mathbf{P}$ *s.t.* $\mathbf{d}_H(q, x) \leq (1+\varepsilon)r$.

(B) *If* $\mathbf{d}_H(q, \mathbf{P}) \geq (1+\varepsilon)r$. *Then,* $\mathbf{E}[\sum_i |\ell_i|] = O(L)$.

Proof.

(A) By previous lemma. Probability $M_i x = M_i q$ is at least $1/n^\beta$. Probability all L data-structures fails, is $\leq (1 - 1/n^\beta)^L < 1/n^{O(1)}$, as $L = O\left(n^\beta \log n\right)$.

(B) For $v \in \mathbf{P}$, s.t. $\mathbf{d}_H(v, q) \geq (1+\varepsilon)r$. Probability for $M \in \mathcal{D}_\tau^t$ misses all the $\geq (1+\varepsilon)r$ coordinates (i.e., collision!) is

$$\leq (1-\tau)^{(1+\varepsilon)rt} \leq \exp\left(-(1+\varepsilon)\frac{tr}{r}\right) =$$

$\exp(-(1+\varepsilon)\beta \log n) = 1/n$, as $\beta = 1/(1+\varepsilon)$. Expected size of ℓ_i is $\leq n \times (1/n) = 1$. □

15: Proof

Lemma

(A) *If* $\mathbf{d}_H(q, \mathbf{P}) \leq r$. *W.h.p. return* $x \in \mathbf{P}$ *s.t.* $\mathbf{d}_H(q, x) \leq (1+\varepsilon)r$.

(B) *If* $\mathbf{d}_H(q, \mathbf{P}) \geq (1+\varepsilon)r$. *Then,* $\mathbf{E}[\sum_i |\ell_i|] = O(L)$.

Proof.

(A) By previous lemma. Probability $M_i x = M_i q$ is at least $1/n^\beta$. Probability all L data-structures fails, is $\leq (1 - 1/n^\beta)^L < 1/n^{O(1)}$, as $L = O(n^\beta \log n)$.

(B) For $v \in \mathbf{P}$, s.t. $\mathbf{d}_H(v, q) \geq (1+\varepsilon)r$. Probability for $M \in \mathcal{D}_\tau^t$ misses all the $\geq (1+\varepsilon)r$ coordinates (i.e., collision!) is

$$\leq (1-\tau)^{(1+\varepsilon)rt} \leq \exp\left(-(1+\varepsilon)\frac{tr}{r}\right) =$$

$\exp(-(1+\varepsilon)\beta \log n) = 1/n$, as $\beta = 1/(1+\varepsilon)$. Expected size of ℓ_i is $\leq n \times (1/n) = 1$. □

15: Proof

Lemma

(A) *If* $\mathbf{d}_H(\boldsymbol{q}, \mathbf{P}) \leq \boldsymbol{r}$. *W.h.p. return* $\boldsymbol{x} \in \mathbf{P}$ *s.t.* $\mathbf{d}_H(\boldsymbol{q}, \boldsymbol{x}) \leq (1+\varepsilon)\boldsymbol{r}$.

(B) *If* $\mathbf{d}_H(\boldsymbol{q}, \mathbf{P}) \geq (1+\varepsilon)\boldsymbol{r}$. *Then,* $\mathbf{E}[\sum_i |\ell_i|] = O(L)$.

Proof.

(A) By previous lemma. Probability $M_i\boldsymbol{x} = M_i\boldsymbol{q}$ is at least $1/n^{\beta}$. Probability all L data-structures fails, is $\leq (1-1/n^{\beta})^{L} < 1/n^{O(1)}$, as $L = O\left(n^{\beta}\log n\right)$.

(B) For $\boldsymbol{v} \in \mathbf{P}$, s.t. $\mathbf{d}_H(\boldsymbol{v}, \boldsymbol{q}) \geq (1+\varepsilon)\boldsymbol{r}$. Probability for $M \in \mathcal{D}^t_\tau$ misses all the $\geq (1+\varepsilon)\boldsymbol{r}$ coordinates (i.e., collision!) is

$$\leq (1-\tau)^{(1+\varepsilon)rt} \leq \exp\left(-(1+\varepsilon)\frac{tr}{r}\right) =$$

$\exp(-(1+\varepsilon)\beta \log n) = 1/n$, as $\beta = 1/(1+\varepsilon)$. Expected size of ℓ_i is $\leq n \times (1/n) = 1$. □

15: Proof

Lemma

(A) *If* $\mathbf{d}_H(\boldsymbol{q}, \mathbf{P}) \leq \boldsymbol{r}$. *W.h.p. return* $\boldsymbol{x} \in \mathbf{P}$ *s.t.* $\mathbf{d}_H(\boldsymbol{q}, \boldsymbol{x}) \leq (1+\varepsilon)\boldsymbol{r}$.

(B) *If* $\mathbf{d}_H(\boldsymbol{q}, \mathbf{P}) \geq (1+\varepsilon)\boldsymbol{r}$. *Then,* $\mathbf{E}[\sum_i |\ell_i|] = O(L)$.

Proof.

(A) By previous lemma. Probability $M_i\boldsymbol{x} = M_i\boldsymbol{q}$ is at least $1/n^{\beta}$. Probability all L data-structures fails, is $\leq (1-1/n^{\beta})^{L} < 1/n^{O(1)}$, as $L = O\left(n^{\beta}\log n\right)$.

(B) For $\boldsymbol{v} \in \mathbf{P}$, s.t. $\mathbf{d}_H(\boldsymbol{v}, \boldsymbol{q}) \geq (1+\varepsilon)\boldsymbol{r}$. Probability for $M \in \mathcal{D}^t_\tau$ misses all the $\geq (1+\varepsilon)\boldsymbol{r}$ coordinates (i.e., collision!) is

$$\leq (1-\tau)^{(1+\varepsilon)rt} \leq \exp\left(-(1+\varepsilon)\frac{tr}{r}\right) =$$

$\exp(-(1+\varepsilon)\beta \log n) = 1/n$, as $\beta = 1/(1+\varepsilon)$. Expected size of ℓ_i is $\leq n \times (1/n) = 1$. □

15: Proof

Lemma

(A) *If* $\mathbf{d}_H(\boldsymbol{q}, \mathbf{P}) \leq \boldsymbol{r}$. *W.h.p. return* $\boldsymbol{x} \in \mathbf{P}$ *s.t.* $\mathbf{d}_H(\boldsymbol{q}, \boldsymbol{x}) \leq (1+\varepsilon)\boldsymbol{r}$.

(B) *If* $\mathbf{d}_H(\boldsymbol{q}, \mathbf{P}) \geq (1+\varepsilon)\boldsymbol{r}$. *Then,* $\mathbf{E}[\sum_i |\ell_i|] = O(L)$.

Proof.

(A) By previous lemma. Probability $M_i\boldsymbol{x} = M_i\boldsymbol{q}$ is at least $1/n^{\beta}$. Probability all L data-structures fails, is $\leq (1-1/n^{\beta})^{L} < 1/n^{O(1)}$, as $L = O\left(n^{\beta}\log n\right)$.

(B) For $\boldsymbol{v} \in \mathbf{P}$, s.t. $\mathbf{d}_H(\boldsymbol{v}, \boldsymbol{q}) \geq (1+\varepsilon)\boldsymbol{r}$. Probability for $M \in \mathcal{D}^t_\tau$ misses all the $\geq (1+\varepsilon)\boldsymbol{r}$ coordinates (i.e., collision!) is

$$\leq (1-\tau)^{(1+\varepsilon)rt} \leq \exp\left(-(1+\varepsilon)\frac{tr}{r}\right) =$$

$\exp(-(1+\varepsilon)\beta \log n) = 1/n$, as $\beta = 1/(1+\varepsilon)$. Expected size of ℓ_i is $\leq n \times (1/n) = 1$. □

16: LSH: Query running time

1. Compute $\forall i \quad M_i q$. Takes $O(dt) = O(d \log n)$ time.
2. Repeat above L times: $O(Ld \log n) = O\left(n^{1/(1+\varepsilon)} \log^2 n\right)$ time overall.
3. In expectation, after scanning $O(L)$ points, either encountered point in distance r, or all points are in distance larger than $(1+\varepsilon)r$.
4. Expected query time is $O(Ld + Ld \log n) = O\left(n^{1/(1+\varepsilon)} \log^2 n\right)$.

16: LSH: Query running time

1. Compute $\forall i \quad M_i q$. Takes $O(dt) = O(d \log n)$ time.
2. Repeat above L times: $O(Ld \log n) = O\left(n^{1/(1+\varepsilon)} \log^2 n\right)$ time overall.
3. In expectation, after scanning $O(L)$ points, either encountered point in distance r, or all points are in distance larger than $(1+\varepsilon)r$.
4. Expected query time is $O(Ld + Ld \log n) = O\left(n^{1/(1+\varepsilon)} \log^2 n\right)$.

Part II

Robust nearest neighbor

17: The good, the bad, and the truncated.

1. technique probes coordinates, trying to detect "hidden" mass of a bad point.
2. mass might concentrate in few coordinates (i.e., $k+1$).
3. Such point still good?
4. Point one has to ignore many more than k coordinates is "bad".
5. Bound the influence of a single coordinate on distance to q.
6. $r = \mathrm{dist}_{\setminus k}(q, \mathsf{P})$.
7. Truncate coordinate of $x - q$ if larger than r/k.

17: The good, the bad, and the truncated.

1. technique probes coordinates, trying to detect "hidden" mass of a bad point.
2. mass might concentrate in few coordinates (i.e., $k+1$).
3. Such point still good?
4. Point one has to ignore many more than k coordinates is "bad".
5. Bound the influence of a single coordinate on distance to q.
6. $r = \mathrm{dist}_{\setminus k}(q, \mathsf{P})$.
7. Truncate coordinate of $x - q$ if larger than r/k.

17: The good, the bad, and the truncated.

1. technique probes coordinates, trying to detect "hidden" mass of a bad point.
2. mass might concentrate in few coordinates (i.e., $k+1$).
3. Such point still good?
4. Point one has to ignore many more than k coordinates is "bad".
5. Bound the influence of a single coordinate on distance to q.
6. $r = \mathrm{dist}_{\setminus k}(q, \mathsf{P})$.
7. Truncate coordinate of $x - q$ if larger than r/k.

17: The good, the bad, and the truncated.

1. technique probes coordinates, trying to detect "hidden" mass of a bad point.
2. mass might concentrate in few coordinates (i.e., $k+1$).
3. Such point still good?
4. Point one has to ignore many more than k coordinates is "bad".
5. Bound the influence of a single coordinate on distance to q.
6. $r = \operatorname{dist}_{\setminus k}(q, \mathsf{P})$.
7. Truncate coordinate of $x - q$ if larger than r/k.

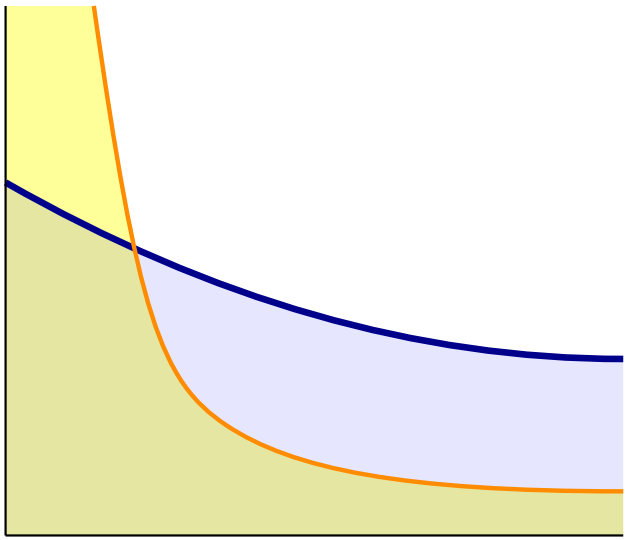
17: The good, the bad, and the truncated.

1. technique probes coordinates, trying to detect "hidden" mass of a bad point.
2. mass might concentrate in few coordinates (i.e., $k+1$).
3. Such point still good?
4. Point one has to ignore many more than k coordinates is "bad".
5. Bound the influence of a single coordinate on distance to q.
6. $r = \mathrm{dist}_{\setminus k}(q, \mathsf{P})$.
7. Truncate coordinate of $x - q$ if larger than r/k.

17: The good, the bad, and the truncated.

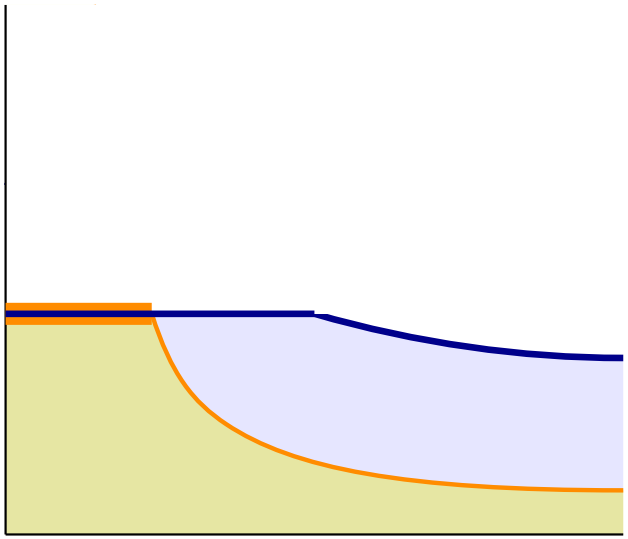
1. technique probes coordinates, trying to detect "hidden" mass of a bad point.
2. mass might concentrate in few coordinates (i.e., $k+1$).
3. Such point still good?
4. Point one has to ignore many more than k coordinates is "bad".
5. Bound the influence of a single coordinate on distance to q.
6. $r = \mathrm{dist}_{\setminus k}(q, \mathsf{P})$.
7. Truncate coordinate of $x - q$ if larger than r/k.

18: Truncation



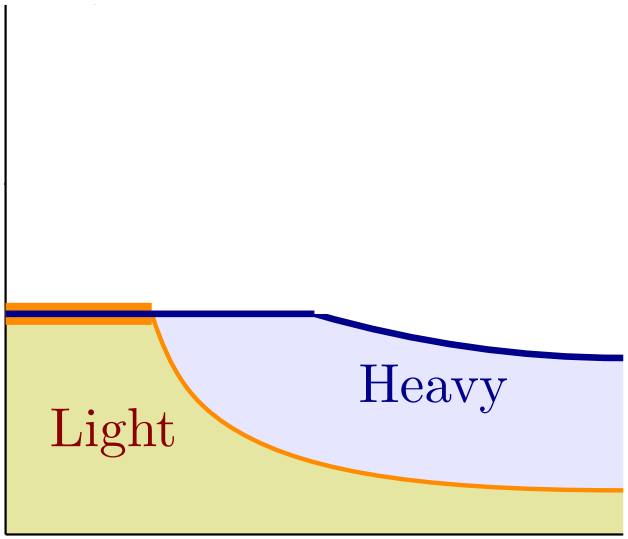
1. Pretend that points are truncated.
2. Prove good approximation (slightly refined LSH analysis).
3. Interpret the results for robust ANN.

18: Truncation



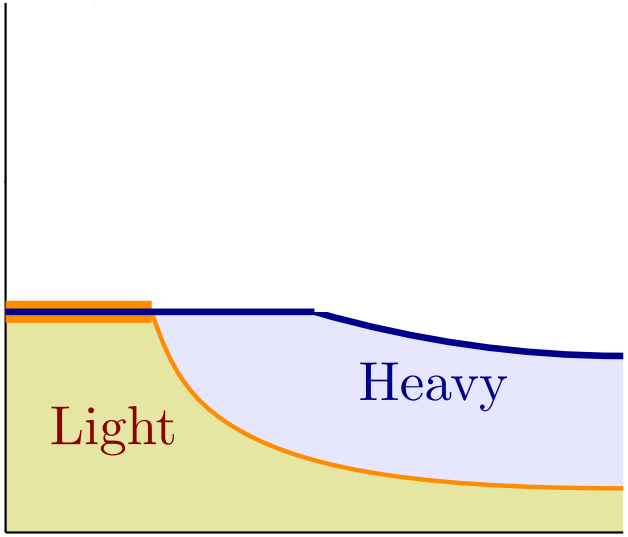
1. Pretend that points are truncated.
2. Prove good approximation (slightly refined LSH analysis).
3. Interpret the results for robust ANN.

18: Truncation



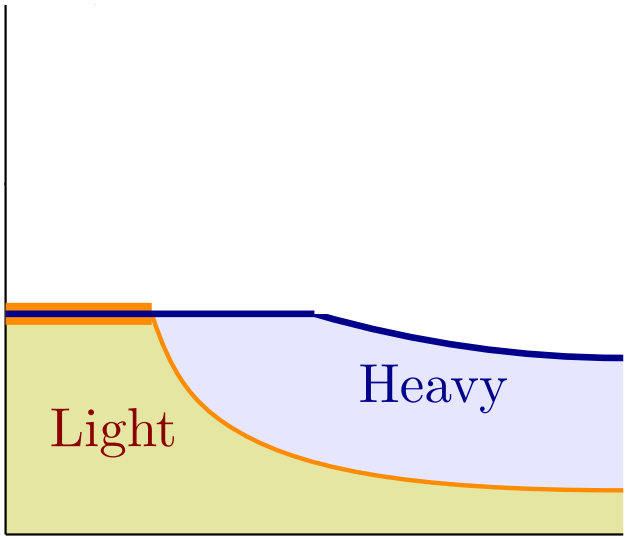
1. Pretend that points are truncated.
2. Prove good approximation (slightly refined LSH analysis).
3. Interpret the results for robust ANN.

18: Truncation



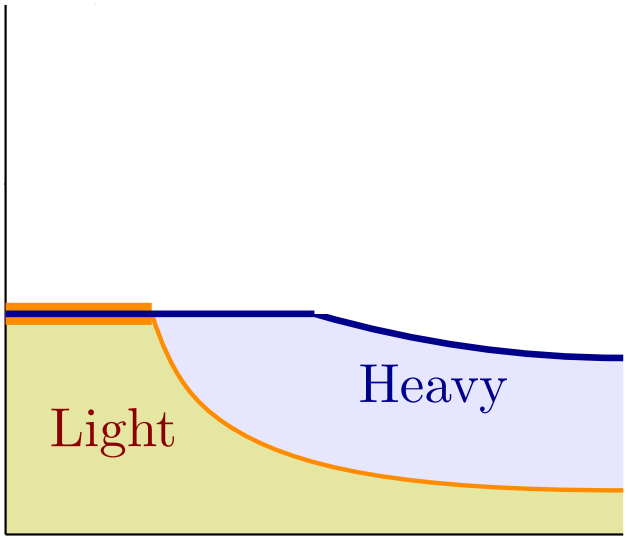
1. Pretend that points are truncated.
2. Prove good approximation (slightly refined LSH analysis).
3. Interpret the results for robust ANN.

18: Truncation



1. Pretend that points are truncated.
2. Prove good approximation (slightly refined LSH analysis).
3. Interpret the results for robust ANN.

18: Truncation



1. Pretend that points are truncated.
2. Prove good approximation (slightly refined LSH analysis).
3. Interpret the results for robust ANN.

19: The good, the bad, and the truncated.

1. Algorithm is bicriterion approximation.
   Returns a point where ignore slightly more coordinates than k
   ... but resulting point is constant approximation to NN when ignoring k coordinates.
2. Bad point after truncation, is amenable to random probing.
3. Heavy points have many bad coordinates.

19: The good, the bad, and the truncated.

1. Algorithm is bicriterion approximation.
   Returns a point where ignore slightly more coordinates than k
   ... but resulting point is constant approximation to NN when ignoring k coordinates.
2. Bad point after truncation, is amenable to random probing.
3. Heavy points have many bad coordinates.

19: The good, the bad, and the truncated.

1. Algorithm is bicriterion approximation.
   Returns a point where ignore slightly more coordinates than k
   ... but resulting point is constant approximation to NN when ignoring k coordinates.
2. Bad point after truncation, is amenable to random probing.
3. Heavy points have many bad coordinates.

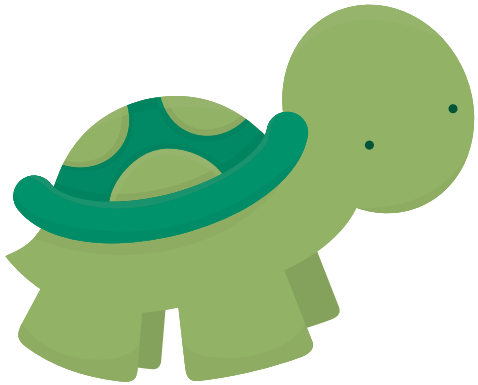
19: The good, the bad, and the truncated.

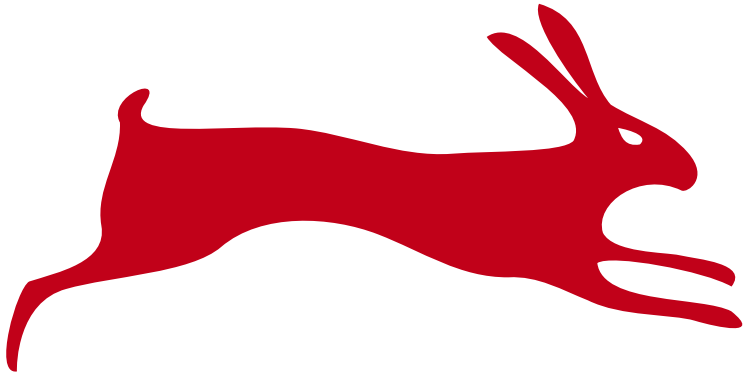
1. Algorithm is bicriterion approximation.
   Returns a point where ignore slightly more coordinates than k
   ... but resulting point is constant approximation to NN when ignoring k coordinates.
2. Bad point after truncation, is amenable to random probing.
3. Heavy points have many bad coordinates.

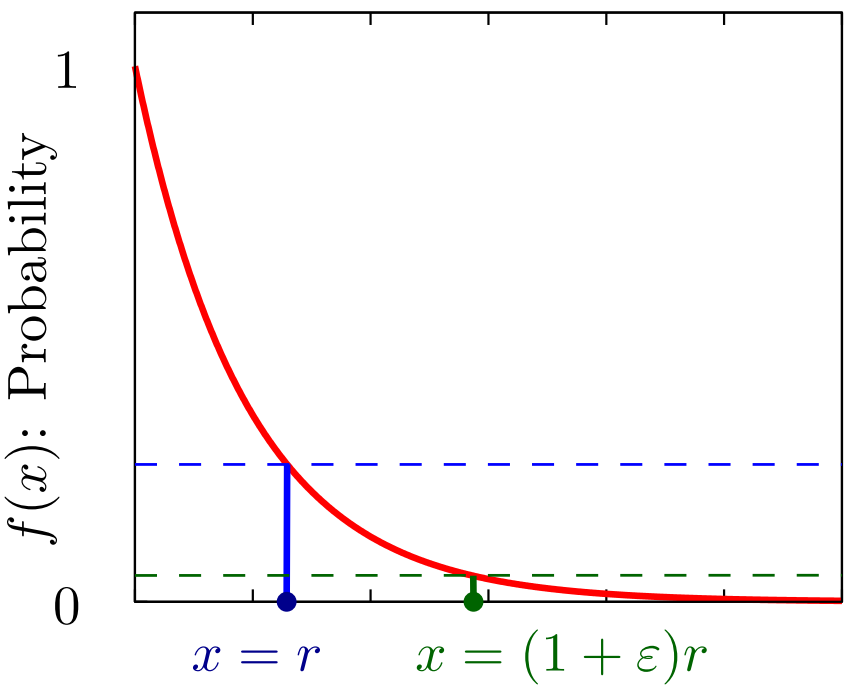
Part III

Data sensitive LSH

20: The tortoise and the hare

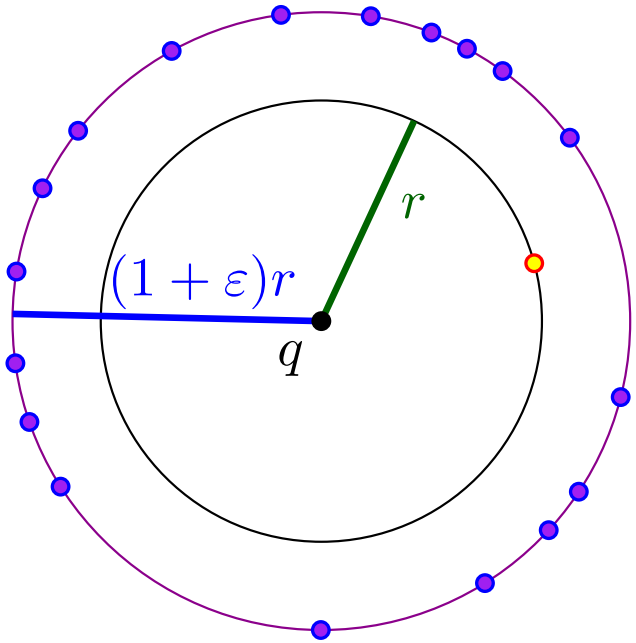






1. Pick coordinate prob $1/d$
2. Repeat t times to "push away" far points. $f((1+\varepsilon)r) \leq 1/n$.
3. Repeat $L \approx 1/f(r)$ times, to succeed w.h.p.

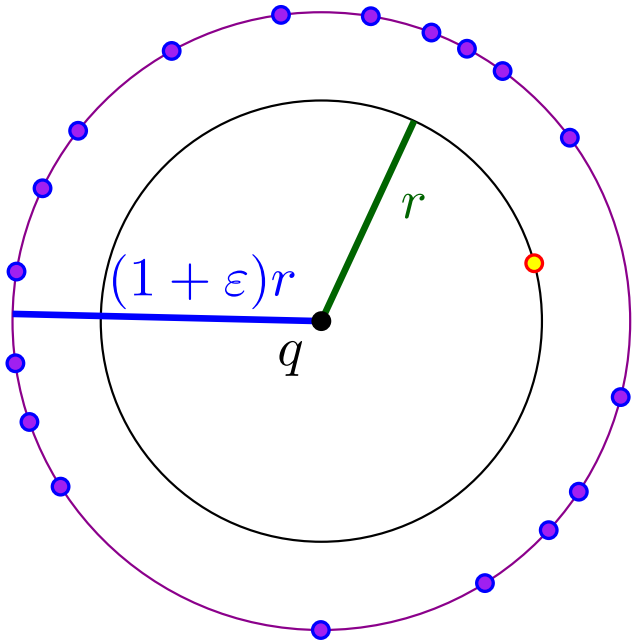
21: The worst/good case for LSH



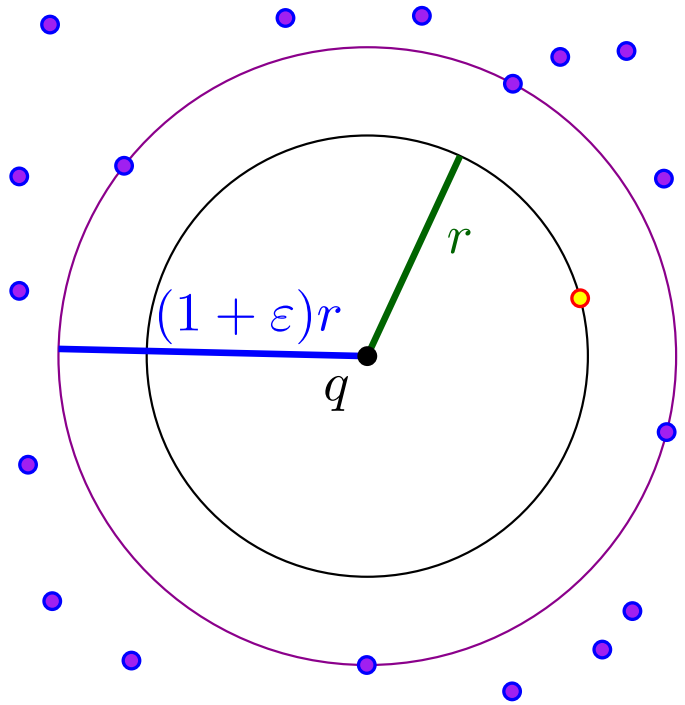
Worst case

Work hard to push away a single point...

21: The worst/good case for LSH



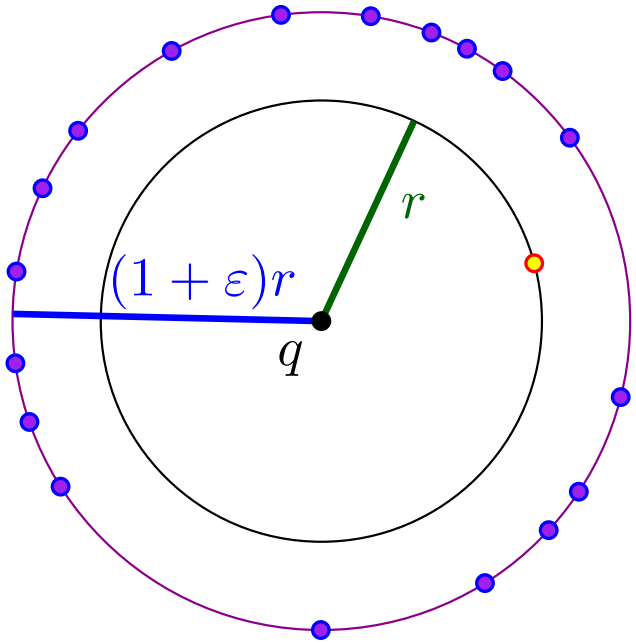
Worst case



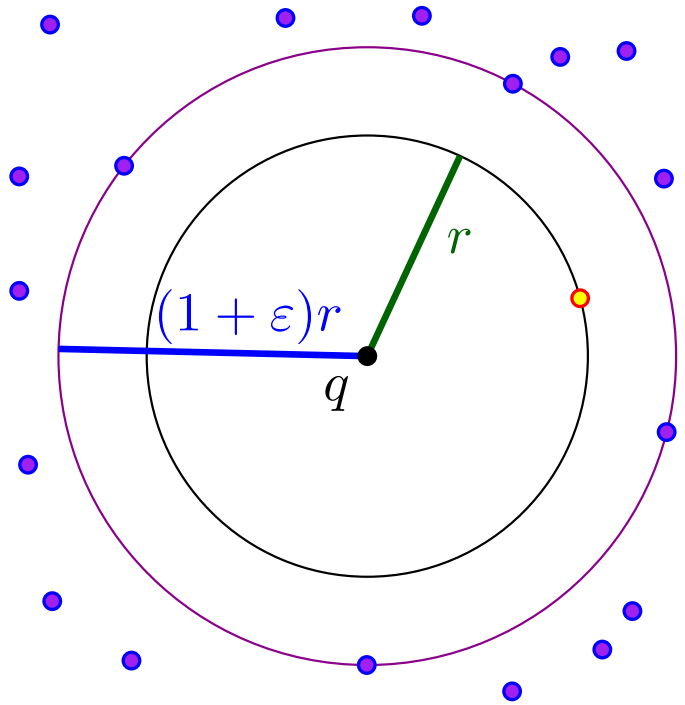
Maybe realistic case

Work hard to push away a single point...

21: The worst/good case for LSH



Worst case



Maybe realistic case

Work hard to push away a single point...

22: Data sensitive LSH.

1. **Idea**: interpret LSH as estimator of local density.
2. If data set is sparse near query point, LSH would hit NN point quickly...
3. ...density estimation would tell us that there are no more close by points.
4. Do exponential search – start with an insensitive LSH scheme (that is fast).
5. ... use more sensitive LSHs, till density estimation tell us we are done.
6. if all fails, the last LSH data-structure used is essentially the old LSH scheme.

22: Data sensitive LSH.

1. **Idea**: interpret LSH as estimator of local density.
2. If data set is sparse near query point, LSH would hit NN point quickly...
3. ...density estimation would tell us that there are no more close by points.
4. Do exponential search – start with an insensitive LSH scheme (that is fast).
5. ... use more sensitive LSHs, till density estimation tell us we are done.
6. if all fails, the last LSH data-structure used is essentially the old LSH scheme.

22: Data sensitive LSH.

1. **Idea**: interpret LSH as estimator of local density.
2. If data set is sparse near query point, LSH would hit NN point quickly...
3. ...density estimation would tell us that there are no more close by points.
4. Do exponential search – start with an insensitive LSH scheme (that is fast).
5. ... use more sensitive LSHs, till density estimation tell us we are done.
6. if all fails, the last LSH data-structure used is essentially the old LSH scheme.

22: Data sensitive LSH.

1. **Idea**: interpret LSH as estimator of local density.
2. If data set is sparse near query point, LSH would hit NN point quickly...
3. ...density estimation would tell us that there are no more close by points.
4. Do exponential search – start with an insensitive LSH scheme (that is fast).
5. ... use more sensitive LSHs, till density estimation tell us we are done.
6. if all fails, the last LSH data-structure used is essentially the old LSH scheme.

22: Data sensitive LSH.

1. **Idea**: interpret LSH as estimator of local density.
2. If data set is sparse near query point, LSH would hit NN point quickly...
3. ...density estimation would tell us that there are no more close by points.
4. Do exponential search – start with an insensitive LSH scheme (that is fast).
5. ... use more sensitive LSHs, till density estimation tell us we are done.
6. if all fails, the last LSH data-structure used is essentially the old LSH scheme.

22: Data sensitive LSH.

1. **Idea**: interpret LSH as estimator of local density.
2. If data set is sparse near query point, LSH would hit NN point quickly...
3. ...density estimation would tell us that there are no more close by points.
4. Do exponential search – start with an insensitive LSH scheme (that is fast).
5. ... use more sensitive LSHs, till density estimation tell us we are done.
6. if all fails, the last LSH data-structure used is essentially the old LSH scheme.

23: Setup

1. $\mathbf{P} \subseteq \{0,1\}^d$: n points.
   r: radius, $\varepsilon > 0$.
2. Preprocessing:
   $i = 1, \ldots, \mathcal{N} = \frac{\log n}{1+\varepsilon}$:
   1. $\mathrm{f}_i(\ell) = (1 - 1/r)^{\ell i} \leq \exp(-\ell i / r)$
   2. $\mathrm{s}(i) = \frac{c \log n}{\mathrm{f}_i(r)} = O\left(e^i \log n\right)$,
   3. $\mathcal{F}_i = \left\{ m_1, m_2, \ldots, m_{\mathrm{s}(i)} \in \mathcal{D}^i_{1/r} \right\}$
   4. $\forall m \in \mathcal{F}_i$: store $m\mathbf{P}$ in hash table.
   5. *colliding* with q: $\mathcal{C}_m(q) = \{x \in \mathbf{P} \mid mq = mx\}$,
   6. Can extract $|\mathcal{C}_m(q)|$ in $O(d)$ time.

23: Setup

1. $\mathbf{P} \subseteq \{0,1\}^d$: n points.
   r: radius, $\varepsilon > 0$.
2. Preprocessing:
   $i = 1, \ldots, \mathcal{N} = \frac{\log n}{1+\varepsilon}$:
   1. $\mathrm{f}_i(\ell) = (1 - 1/r)^{\ell i} \leq \exp(-\ell i / r)$
   2. $\mathrm{s}(i) = \frac{c \log n}{\mathrm{f}_i(r)} = O\left(e^i \log n\right)$,
   3. $\mathcal{F}_i = \left\{ m_1, m_2, \ldots, m_{\mathrm{s}(i)} \in \mathcal{D}^i_{1/r} \right\}$
   4. $\forall m \in \mathcal{F}_i$: store $m\mathbf{P}$ in hash table.
   5. *colliding* with q: $C_m(q) = \{x \in \mathbf{P} \mid mq = mx\}$,
   6. Can extract $|C_m(q)|$ in $O(d)$ time.

23: Setup

1. $\mathbf{P} \subseteq \{0,1\}^d$: n points.
   r: radius, $\varepsilon > 0$.
2. Preprocessing:
   $i = 1, \ldots, \mathcal{N} = \frac{\log n}{1+\varepsilon}$:
   1. $\mathrm{f}_i(\ell) = (1 - 1/r)^{\ell i} \leq \exp(-\ell i / r)$
   2. $\mathrm{s}(i) = \frac{c \log n}{\mathrm{f}_i(r)} = O\left(e^i \log n\right)$,
   3. $\mathcal{F}_i = \left\{ m_1, m_2, \ldots, m_{\mathrm{s}(i)} \in \mathcal{D}^i_{1/r} \right\}$
   4. $\forall m \in \mathcal{F}_i$: store $m\mathbf{P}$ in hash table.
   5. *colliding* with q: $C_m(q) = \{x \in \mathbf{P} \mid mq = mx\}$,
   6. Can extract $|C_m(q)|$ in $O(d)$ time.

23: Setup

1. $\mathbf{P} \subseteq \{0,1\}^d$: n points.
   r: radius, $\varepsilon > 0$.
2. Preprocessing:
   $i = 1, \ldots, \mathcal{N} = \frac{\log n}{1+\varepsilon}$:
   1. $\mathrm{f}_i(\ell) = (1 - 1/r)^{\ell i} \leq \exp(-\ell i / r)$
   2. $\mathrm{s}(i) = \frac{\mathrm{c} \log n}{\mathrm{f}_i(r)} = O\left(e^i \log n\right)$,
   3. $\mathcal{F}_i = \left\{ m_1, m_2, \ldots, m_{\mathrm{s}(i)} \in \mathcal{D}^i_{1/r} \right\}$
   4. $\forall m \in \mathcal{F}_i$: store $m\mathbf{P}$ in hash table.
   5. *colliding* with $\boldsymbol{q}$: $\mathcal{C}_m(\boldsymbol{q}) = \{\boldsymbol{x} \in \mathbf{P} \mid m\boldsymbol{q} = m\boldsymbol{x}\}$,
   6. Can extract $|\mathcal{C}_m(\boldsymbol{q})|$ in $O(d)$ time.

23: Setup

1. $\mathbf{P} \subseteq \{0,1\}^d$: n points.
   r: radius, $\varepsilon > 0$.
2. Preprocessing:
   $i = 1, \ldots, \mathcal{N} = \frac{\log n}{1+\varepsilon}$:
   1. $\mathrm{f}_i(\ell) = (1 - 1/r)^{\ell i} \leq \exp(-\ell i / r)$
   2. $\mathrm{s}(i) = \frac{c \log n}{\mathrm{f}_i(r)} = O\left(e^i \log n\right)$,
   3. $\mathcal{F}_i = \left\{ m_1, m_2, \ldots, m_{\mathrm{s}(i)} \in \mathcal{D}^i_{1/r} \right\}$
   4. $\forall m \in \mathcal{F}_i$: store $m\mathbf{P}$ in hash table.
   5. *colliding* with q: $C_m(q) = \{x \in \mathbf{P} \mid mq = mx\}$,
   6. Can extract $|C_m(q)|$ in $O(d)$ time.

23: Setup

1. $\mathbf{P} \subseteq \{0,1\}^d$: n points.
   r: radius, $\varepsilon > 0$.
2. Preprocessing:
   $i = 1, \ldots, \mathcal{N} = \frac{\log n}{1+\varepsilon}$:
   1. $\mathrm{f}_i(\ell) = (1 - 1/r)^{\ell i} \leq \exp(-\ell i / r)$
   2. $\mathrm{s}(i) = \frac{\mathrm{c} \log n}{\mathrm{f}_i(r)} = O\left(e^i \log n\right)$,
   3. $\mathcal{F}_i = \left\{ m_1, m_2, \ldots, m_{\mathrm{s}(i)} \in \mathcal{D}^i_{1/r} \right\}$
   4. $\forall m \in \mathcal{F}_i$: store $m\mathbf{P}$ in hash table.
   5. *colliding* with $\boldsymbol{q}$: $\mathcal{C}_m(\boldsymbol{q}) = \{\boldsymbol{x} \in \mathbf{P} \mid m\boldsymbol{q} = m\boldsymbol{x}\}$,
   6. Can extract $|\mathcal{C}_m(\boldsymbol{q})|$ in $O(d)$ time.

24: Query.

1. Query: $\boldsymbol{q} \in \mathbb{H}^{\boldsymbol{d}}$
2. For $i = 1 \ldots$
    1. $X_i = \sum_{m \in \mathcal{F}_i} |\mathcal{C}_m(\boldsymbol{q})|$. (# of colliding points.)
    2. If $X_i > 4s(i)$ then $i++$, continue to next iteration.
    3. If $X_i \leq 4s(i)$, then
        1. extract the X_i points
        2. Return closest point among them.
    4. If $i = \mathcal{N}$ then do standard LSH query.

24: Query.

1. Query: $q \in \mathbb{H}^d$
2. For $i = 1 \ldots$
   1. $X_i = \sum_{m \in \mathcal{F}_i} |\mathcal{C}_m(q)|$. (# of colliding points.)
   2. If $X_i > 4s(i)$ then $i++$, continue to next iteration.
   3. If $X_i \leq 4s(i)$, then
      1. extract the X_i points
      2. Return closest point among them.
   4. If $i = \mathcal{N}$ then do standard LSH query.

24: Query.

1. Query: $q \in \mathbb{H}^d$
2. For $i = 1 \ldots$
   1. $X_i = \sum_{m \in \mathcal{F}_i} |\mathcal{C}_m(q)|$. (# of colliding points.)
   2. If $X_i > 4s(i)$ then $i++$, continue to next iteration.
   3. If $X_i \leq 4s(i)$, then
      1. extract the X_i points
      2. Return closest point among them.
   4. If $i = \mathcal{N}$ then do standard LSH query.

24: Query.

1. Query: $q \in \mathbb{H}^d$
2. For $i = 1 \ldots$
   1. $X_i = \sum_{m \in \mathcal{F}_i} |\mathcal{C}_m(q)|$. (# of colliding points.)
   2. If $X_i > 4s(i)$ then $i++$, continue to next iteration.
   3. If $X_i \leq 4s(i)$, then
      1. extract the X_i points
      2. Return closest point among them.
   4. If $i = \mathcal{N}$ then do standard LSH query.

25: Intuition

Expected number of collisions with $\boldsymbol{q}$, for a $m \in \mathcal{F}_i$ is

$$\gamma_i = \mathbf{E}\Big[|\mathcal{C}_m(\boldsymbol{q})|\Big] = \sum_{\boldsymbol{x}\in \mathbf{P}} \mathbf{f}_i\big(\mathbf{d}_H(\boldsymbol{q},\boldsymbol{x})\big) \leq \sum_{\boldsymbol{x}\in \mathbf{P}} \exp\left(-\frac{\mathbf{d}_H(\boldsymbol{q},\boldsymbol{x})i}{r}\right)$$

A convolution over the input set...

Expected number collisions with $\boldsymbol{q}$ (with all projections $\mathcal{F}_i$):
$\Gamma_i = \mathbf{E}[X_i] = \mathbf{s}(i)\gamma_i$.

If scan collision lists in ith level: query time $O\big(d(X_i + \mathbf{s}(i))\big)$.

If $X_i = O(\mathbf{s}(i))$, then query time is small.

Intuitively: Bigly i stronger DS push points away from $\boldsymbol{q}$.

If lucky, need to push little.

Standard LSH: ends up immediately with $\mathcal{F}_{\mathcal{N}}$.

25: Intuition

Expected number of collisions with $\boldsymbol{q}$, for a $m \in \mathcal{F}_i$ is

$$\gamma_i = \mathbf{E}\left[|\mathcal{C}_m(\boldsymbol{q})|\right] = \sum_{\boldsymbol{x}\in\mathbf{P}} \mathbf{f}_i\left(\mathbf{d}_H(\boldsymbol{q},\boldsymbol{x})\right) \leq \sum_{\boldsymbol{x}\in\mathbf{P}} \exp\left(-\frac{\mathbf{d}_H(\boldsymbol{q},\boldsymbol{x})i}{r}\right)$$

A convolution over the input set...

Expected number collisions with $\boldsymbol{q}$ (with all projections $\mathcal{F}_i$):
$\Gamma_i = \mathbf{E}[X_i] = \mathbf{s}(i)\gamma_i$.

If scan collision lists in ith level: query time $O\left(d(X_i + \mathbf{s}(i))\right)$.

If $X_i = O(\mathbf{s}(i))$, then query time is small.

Intuitively: Bigly i stronger DS push points away from $\boldsymbol{q}$.

If lucky, need to push little.

Standard LSH: ends up immediately with $\mathcal{F}_{\mathcal{N}}$.

25: Intuition

Expected number of collisions with $\boldsymbol{q}$, for a $m \in \mathcal{F}_i$ is

$$\gamma_i = \mathbf{E}\left[|\mathcal{C}_m(\boldsymbol{q})|\right] = \sum_{\boldsymbol{x}\in\mathbf{P}} \mathrm{f}_i\left(\mathrm{d}_H(\boldsymbol{q},\boldsymbol{x})\right) \leq \sum_{\boldsymbol{x}\in\mathbf{P}} \exp\left(-\frac{\mathrm{d}_H(\boldsymbol{q},\boldsymbol{x})i}{r}\right)$$

A convolution over the input set...

Expected number collisions with $\boldsymbol{q}$ (with all projections $\mathcal{F}_i$):

$$\Gamma_i = \mathbf{E}[X_i] = \mathrm{s}(i)\gamma_i.$$

If scan collision lists in ith level: query time $O\left(d(X_i + \mathrm{s}(i))\right)$.

If $X_i = O(\mathrm{s}(i))$, then query time is small.

Intuitively: Bigly i stronger DS push points away from $\boldsymbol{q}$.

If lucky, need to push little.

Standard LSH: ends up immediately with $\mathcal{F}_{\mathcal{N}}$.

25: Intuition

Expected number of collisions with $\boldsymbol{q}$, for a $m \in \mathcal{F}_i$ is

$\gamma_i = \mathbf{E}\left[|\mathcal{C}_m(\boldsymbol{q})|\right] = \sum_{\boldsymbol{x}\in\mathbf{P}} \mathbf{f}_i\left(\mathbf{d}_H(\boldsymbol{q},\boldsymbol{x})\right) \leq \sum_{\boldsymbol{x}\in\mathbf{P}} \exp\left(-\frac{\mathbf{d}_H(\boldsymbol{q},\boldsymbol{x})i}{r}\right)$

A convolution over the input set...

Expected number collisions with $\boldsymbol{q}$ (with all projections $\mathcal{F}_i$):

$\boldsymbol{\Gamma}_i = \mathbf{E}[X_i] = \mathbf{s}(i)\gamma_i$.

If scan collision lists in ith level: query time $O\left(d(X_i + \mathbf{s}(i))\right)$.

If $X_i = O(\mathbf{s}(i))$, then query time is small.

Intuitively: Bigly i stronger DS push points away from $\boldsymbol{q}$.

If lucky, need to push little.

Standard LSH: ends up immediately with $\mathcal{F}_{\mathcal{N}}$.

25: Intuition

Expected number of collisions with $\boldsymbol{q}$, for a $m \in \mathcal{F}_i$ is
$\gamma_i = \mathbf{E}\left[|\mathcal{C}_m(\boldsymbol{q})|\right] = \sum_{\boldsymbol{x}\in\mathbf{P}} \mathbf{f}_i\left(\mathbf{d}_H(\boldsymbol{q},\boldsymbol{x})\right) \leq \sum_{\boldsymbol{x}\in\mathbf{P}} \exp\left(-\frac{\mathbf{d}_H(\boldsymbol{q},\boldsymbol{x})i}{r}\right)$
A convolution over the input set...

Expected number collisions with $\boldsymbol{q}$ (with all projections $\mathcal{F}_i$):
$\boldsymbol{\Gamma_i = \mathbf{E}[X_i] = \mathbf{s}(i)\gamma_i}$.

If scan collision lists in $\boldsymbol{i}$th level: query time $O\left(d(X_i + \mathbf{s}(i))\right)$.

If $\boldsymbol{X_i = O(\mathbf{s}(i))}$, then query time is small.

Intuitively: Bigly i stronger DS push points away from $\boldsymbol{q}$.

If lucky, need to push little.

Standard LSH: ends up immediately with $\mathcal{F}_{\mathcal{N}}$.

25: Intuition

Expected number of collisions with $\boldsymbol{q}$, for a $m \in \mathcal{F}_i$ is
$\gamma_i = \mathbf{E}\left[|\mathcal{C}_m(\boldsymbol{q})|\right] = \sum_{\boldsymbol{x}\in\mathbf{P}} \mathbf{f}_i\left(\mathbf{d}_H(\boldsymbol{q},\boldsymbol{x})\right) \leq \sum_{\boldsymbol{x}\in\mathbf{P}} \exp\left(-\frac{\mathbf{d}_H(\boldsymbol{q},\boldsymbol{x})i}{r}\right)$
A convolution over the input set...

Expected number collisions with $\boldsymbol{q}$ (with all projections $\mathcal{F}_i$):
$\boldsymbol{\Gamma_i = \mathbf{E}[X_i] = \mathbf{s}(i)\gamma_i}$.

If scan collision lists in $\boldsymbol{i}$th level: query time $O\left(d(X_i + \mathbf{s}(i))\right)$.

If $\boldsymbol{X_i = O(\mathbf{s}(i))}$, then query time is small.

Intuitively: Bigly $\boldsymbol{i}$ stronger DS push points away from $\boldsymbol{q}$.

If lucky, need to push little.

Standard LSH: ends up immediately with $\mathcal{F}_{\mathcal{N}}$.

25: Intuition

Expected number of collisions with $\boldsymbol{q}$, for a $m \in \mathcal{F}_i$ is
$\gamma_i = \mathbf{E}\left[|\mathcal{C}_m(\boldsymbol{q})|\right] = \sum_{\boldsymbol{x}\in\mathbf{P}} \mathbf{f}_i\left(\mathbf{d}_H(\boldsymbol{q},\boldsymbol{x})\right) \leq \sum_{\boldsymbol{x}\in\mathbf{P}} \exp\left(-\frac{\mathbf{d}_H(\boldsymbol{q},\boldsymbol{x})i}{r}\right)$
A convolution over the input set...

Expected number collisions with $\boldsymbol{q}$ (with all projections $\mathcal{F}_i$):
$\boldsymbol{\Gamma}_i = \mathbf{E}[X_i] = \mathbf{s}(i)\gamma_i$.

If scan collision lists in ith level: query time $O\left(d(X_i + \mathbf{s}(i))\right)$.

If $X_i = O(\mathbf{s}(i))$, then query time is small.

Intuitively: Bigly i stronger DS push points away from $\boldsymbol{q}$.

If lucky, need to push little.

Standard LSH: ends up immediately with $\mathcal{F}_{\mathcal{N}}$.

25: Intuition

Expected number of collisions with $\boldsymbol{q}$, for a $m \in \mathcal{F}_i$ is
$\gamma_i = \mathbf{E}\left[|\mathcal{C}_m(\boldsymbol{q})|\right] = \sum_{\boldsymbol{x}\in\mathbf{P}} \mathbf{f}_i\left(\mathbf{d}_H(\boldsymbol{q},\boldsymbol{x})\right) \leq \sum_{\boldsymbol{x}\in\mathbf{P}} \exp\left(-\frac{\mathbf{d}_H(\boldsymbol{q},\boldsymbol{x})i}{r}\right)$
A convolution over the input set...

Expected number collisions with $\boldsymbol{q}$ (with all projections $\mathcal{F}_i$):
$\boldsymbol{\Gamma}_i = \mathbf{E}[X_i] = \mathbf{s}(i)\gamma_i$.

If scan collision lists in ith level: query time $O\left(d(X_i + \mathbf{s}(i))\right)$.

If $X_i = O(\mathbf{s}(i))$, then query time is small.

Intuitively: Bigly i stronger DS push points away from $\boldsymbol{q}$.

If lucky, need to push little.

Standard LSH: ends up immediately with $\mathcal{F}_{\mathcal{N}}$.

26: Example: If the data is locally low dimensional

Γ_i depends on how the data looks like near the query.

Assume locally near query point, data is a uniform low dimensional point set.

Assume # points in distance ℓ from $q \leq \#(\ell) = O\big((\ell/r)^k\big)$.
k: small, r: distance to NN to q.

$\Gamma_i = \mathrm{s}(i) \sum_{x \in P} \mathrm{f}_i\big(\mathrm{d}_H(q,x)\big)$
$\leq O(e^i \log n) \sum_{j=1}^{\infty} \#(jr) \exp(-ji)$
$= \sum_{j=1}^{\infty} O\big(j^k \exp(i - ji) \log n\big).$

Setting $i = k$:

$$\Gamma_k \leq \sum_{j=1}^{\infty} O\left(\left(\frac{j}{e^{j-1}}\right)^k \log n\right) \leq \sum_{=1}^{\infty} O\left(\frac{j}{2^j} \log n\right) \leq O(\log n)$$

Expectation: Stop $O(k)$ rounds. Query time: $O(d \log n)$.

26: Example: If the data is locally low dimensional

Γ_i depends on how the data looks like near the query.

Assume locally near query point, data is a uniform low dimensional point set.

Assume # points in distance ℓ from $q \leq \#(\ell) = O\left((\ell/r)^k\right)$.
k: small, r: distance to NN to q.

$\Gamma_i = s(i) \sum_{x \in P} f_i\left(d_H(q,x)\right)$
$\leq O(e^i \log n) \sum_{j=1}^{\infty} \#(jr) \exp(-ji)$
$= \sum_{j=1}^{\infty} O\left(j^k \exp(i - ji) \log n\right).$

Setting $i = k$:

$$\Gamma_k \leq \sum_{j=1}^{\infty} O\left(\left(\frac{j}{e^{j-1}}\right)^k \log n\right) \leq \sum_{=1}^{\infty} O\left(\frac{j}{2^j} \log n\right) \leq O(\log n)$$

Expectation: Stop $O(k)$ rounds. Query time: $O(d \log n)$.

26: Example: If the data is locally low dimensional

Γ_i depends on how the data looks like near the query.

Assume locally near query point, data is a uniform low dimensional point set.

Assume # points in distance ℓ from $q \leq \#(\ell) = O\left((\ell/r)^k\right)$.
k: small, r: distance to NN to q.

$$\Gamma_i = s(i) \sum_{x \in P} f_i\left(d_H(q,x)\right)$$
$$\leq O(e^i \log n) \sum_{j=1}^{\infty} \#(jr) \exp(-ji)$$
$$= \sum_{j=1}^{\infty} O\left(j^k \exp(i - ji) \log n\right).$$

Setting $i = k$:

$$\Gamma_k \leq \sum_{j=1}^{\infty} O\left(\left(\frac{j}{e^{j-1}}\right)^k \log n\right) \leq \sum_{=1}^{\infty} O\left(\frac{j}{2^j} \log n\right) \leq O(\log n)$$

Expectation: Stop $O(k)$ rounds. Query time: $O(d \log n)$.

26: Example: If the data is locally low dimensional

Γ_i depends on how the data looks like near the query.

Assume locally near query point, data is a uniform low dimensional point set.

Assume # points in distance ℓ from $q \leq \#(\ell) = O\left((\ell/r)^k\right)$.
k: small, r: distance to NN to q.

$\Gamma_i = s(i) \sum_{x \in P} f_i\left(d_H(q,x)\right)$
$\leq O(e^i \log n) \sum_{j=1}^{\infty} \#(jr) \exp(-ji)$
$= \sum_{j=1}^{\infty} O\left(j^k \exp(i - ji) \log n\right).$

Setting $i = k$:

$$\Gamma_k \leq \sum_{j=1}^{\infty} O\left(\left(\frac{j}{e^{j-1}}\right)^k \log n\right) \leq \sum_{=1}^{\infty} O\left(\frac{j}{2^j} \log n\right) \leq O(\log n)$$

Expectation: Stop $O(k)$ rounds. Query time: $O(d \log n)$.

26: Example: If the data is locally low dimensional

Γ_i depends on how the data looks like near the query.

Assume locally near query point, data is a uniform low dimensional point set.

Assume # points in distance ℓ from $q \leq \#(\ell) = O\left((\ell/r)^k\right)$.
k: small, r: distance to NN to q.

$\Gamma_i = s(i) \sum_{x \in P} f_i\left(d_H(q,x)\right)$
$\leq O(e^i \log n) \sum_{j=1}^{\infty} \#(jr) \exp(-ji)$
$= \sum_{j=1}^{\infty} O\left(j^k \exp(i - ji) \log n\right)$.

Setting $i = k$:

$$\Gamma_k \leq \sum_{j=1}^{\infty} O\left(\left(\frac{j}{e^{j-1}}\right)^k \log n\right) \leq \sum_{=1}^{\infty} O\left(\frac{j}{2^j} \log n\right) \leq O(\log n)$$

Expectation: Stop $O(k)$ rounds. Query time: $O(d \log n)$.

26: Example: If the data is locally low dimensional

Γ_i depends on how the data looks like near the query.

Assume locally near query point, data is a uniform low dimensional point set.

Assume # points in distance ℓ from $q \leq \#(\ell) = O\big((\ell/r)^k\big)$.
k: small, r: distance to NN to q.

$\Gamma_i = s(i) \sum_{x \in P} f_i\big(d_H(q,x)\big)$
$\leq O(e^i \log n) \sum_{j=1}^{\infty} \#(jr) \exp(-ji)$
$= \sum_{j=1}^{\infty} O\big(j^k \exp(i - ji) \log n\big)$.

Setting $i = k$:

$$\Gamma_k \leq \sum_{j=1}^{\infty} O\left(\left(\frac{j}{e^{j-1}}\right)^k \log n\right) \leq \sum_{=1}^{\infty} O\left(\frac{j}{2^j} \log n\right) \leq O(\log n)$$

Expectation: Stop $O(k)$ rounds. Query time: $O(d \log n)$.

26: Example: If the data is locally low dimensional

Γ_i depends on how the data looks like near the query.

Assume locally near query point, data is a uniform low dimensional point set.

Assume # points in distance ℓ from $q \leq \#(\ell) = O\left((\ell/r)^k\right)$.
k: small, r: distance to NN to q.

$$\Gamma_i = s(i) \sum_{x \in P} f_i\left(d_H(q,x)\right)$$
$$\leq O(e^i \log n) \sum_{j=1}^{\infty} \#(jr) \exp(-ji)$$
$$= \sum_{j=1}^{\infty} O\left(j^k \exp(i - ji) \log n\right).$$

Setting $i = k$:

$$\Gamma_k \leq \sum_{j=1}^{\infty} O\left(\left(\frac{j}{e^{j-1}}\right)^k \log n\right) \leq \sum_{=1}^{\infty} O\left(\frac{j}{2^j} \log n\right) \leq O(\log n)$$

Expectation: Stop $O(k)$ rounds. Query time: $O(d \log n)$.

26: Example: If the data is locally low dimensional

Γ_i depends on how the data looks like near the query.

Assume locally near query point, data is a uniform low dimensional point set.

Assume # points in distance ℓ from $q \leq \#(\ell) = O\left((\ell/r)^k\right)$.
k: small, r: distance to NN to q.

$$\Gamma_i = \mathbf{s}(i) \sum_{x \in \mathbf{P}} \mathbf{f}_i\left(\mathbf{d}_H(q, x)\right)$$
$$\leq O(e^i \log n) \sum_{j=1}^{\infty} \#(jr) \exp(-ji)$$
$$= \sum_{j=1}^{\infty} O\left(j^k \exp(i - ji) \log n\right).$$

Setting $i = k$:

$$\Gamma_k \leq \sum_{j=1}^{\infty} O\left(\left(\frac{j}{e^{j-1}}\right)^k \log n\right) \leq \sum_{=1}^{\infty} O\left(\frac{j}{2^j} \log n\right) \leq O(\log n)$$

Expectation: Stop $O(k)$ rounds. Query time: $O(d \log n)$.

27: Analysis

Lemma

If $\exists$ point with distance $\leq r$ from q, then algorithm computes, w.h.p., point x s.t., $\mathbf{d}_H(q, x) \leq (1+\varepsilon)r$.

Proof.

$q^* \in P$: nearest neighbor to q.

For DS_i: probability q^* not collide with q
$\leq (1 - f_i(r))^{s(i)} \leq \exp(-f_i(r)s(i)) = \exp(-O(\log n)) \leq 1/n^{O(1)}$.

Since $s(i) = O(e^i \log n)$

$i < \mathcal{N}$: Algorithm stops and scans all lists.

$i = \mathcal{N}$: Just reg LSH. **(Approximation only in this case!)**

27: Analysis

Lemma

If $\exists$ point with distance $\leq r$ from q, then algorithm computes, w.h.p., point x s.t., $\mathbf{d}_H(q, x) \leq (1+\varepsilon)r$.

Proof.

$q^* \in \mathbf{P}$: nearest neighbor to q.

For DS_i: probability q^* not collide with q
$\leq (1 - \mathbf{f}_i(r))^{\mathbf{s}(i)} \leq \exp(-\mathbf{f}_i(r)\mathbf{s}(i)) = \exp(-O(\log n)) \leq 1/n^{O(1)}$.

Since $\mathbf{s}(i) = O\left(e^i \log n\right)$

$i < \mathcal{N}$: Algorithm stops and scans all lists.

$i = \mathcal{N}$: Just reg LSH. **(Approximation only in this case!)**

27: Analysis

Lemma

If $\exists$ point with distance $\leq r$ from q, then algorithm computes, w.h.p., point x s.t., $d_H(q,x) \leq (1+\varepsilon)r$.

Proof.

$q^* \in P$: nearest neighbor to q.

For DS_i: probability q^* not collide with q
$\leq (1-f_i(r))^{s(i)} \leq \exp(-f_i(r)s(i)) = \exp(-O(\log n)) \leq 1/n^{O(1)}$.

Since $s(i) = O(e^i \log n)$

$i < \mathcal{N}$: Algorithm stops and scans all lists.

$i = \mathcal{N}$: Just reg LSH. **(Approximation only in this case!)**

27: Analysis

Lemma

If $\exists$ point with distance $\leq r$ from q, then algorithm computes, w.h.p., point x s.t., $d_H(q,x) \leq (1+\varepsilon)r$.

Proof.

$q^* \in P$: nearest neighbor to q.

For DS_i: probability q^* not collide with q
$\leq (1-f_i(r))^{s(i)} \leq \exp(-f_i(r)s(i)) = \exp(-O(\log n)) \leq 1/n^{O(1)}$.

Since $s(i) = O(e^i \log n)$

$i < \mathcal{N}$: Algorithm stops and scans all lists.

$i = \mathcal{N}$: Just reg LSH. **(Approximation only in this case!)**

27: Analysis

Lemma

If $\exists$ point with distance $\leq r$ from q, then algorithm computes, w.h.p., point x s.t., $d_H(q,x) \leq (1+\varepsilon)r$.

Proof.

$q^* \in P$: nearest neighbor to q.

For DS_i: probability q^* not collide with q
$\leq (1-f_i(r))^{s(i)} \leq \exp(-f_i(r)s(i)) = \exp(-O(\log n)) \leq 1/n^{O(1)}$.

Since $s(i) = O(e^i \log n)$

$i < \mathcal{N}$: Algorithm stops and scans all lists.

$i = \mathcal{N}$: Just reg LSH. **(Approximation only in this case!)**

27: Analysis

Lemma

If $\exists$ point with distance $\leq r$ from q, then algorithm computes, w.h.p., point x s.t., $d_H(q,x) \leq (1+\varepsilon)r$.

Proof.

$q^* \in P$: nearest neighbor to q.

For DS_i: probability q^* not collide with q
$\leq (1-f_i(r))^{s(i)} \leq \exp(-f_i(r)s(i)) = \exp(-O(\log n)) \leq 1/n^{O(1)}$.

Since $s(i) = O(e^i \log n)$

$i < \mathcal{N}$: Algorithm stops and scans all lists.

$i = \mathcal{N}$: Just reg LSH. **(Approximation only in this case!)**

28: Value of $\mathbf{s(i)}$

Lemma

For any $i > 0$, *we have* $\mathbf{s}(i) = O\left(e^i \log n\right)$.

Proof.

$$\mathbf{f}_i(r) = (1 - 1/r)^{(r-1)ri/(r-1)} \geq \exp\left(-r\tfrac{i}{r-1}\right) = \exp\left(-i - \tfrac{i}{r-1}\right)$$

since $\forall m > 0$: $(1 - 1/m)^{m-1} \geq 1/e \geq (1 - 1/m)^m$.

Can assume that $r > \log n \geq i$ (duplicate coordinates)...

$\implies \mathbf{f}_i(r) = \Theta(\exp(-i))$.

$\implies \mathbf{s}(i) = O(\frac{\log n}{\mathbf{f}_i(r)}) = O\left(e^i \log n\right)$. □

28: Value of $\mathbf{s(i)}$

Lemma

For any $i > 0$, *we have* $\mathbf{s}(i) = O\left(e^i \log n\right)$.

Proof.

$$\mathbf{f}_i(r) = (1 - 1/r)^{(r-1)ri/(r-1)} \geq \exp\left(-r\tfrac{i}{r-1}\right) = \exp\left(-i - \tfrac{i}{r-1}\right)$$

---

since $\forall m > 0$: $(1 - 1/m)^{m-1} \geq 1/e \geq (1 - 1/m)^m$.

Can assume that $r > \log n \geq i$ (duplicate coordinates)...

$\implies \mathbf{f}_i(r) = \Theta(\exp(-i))$.

$\implies \mathbf{s}(i) = O(\frac{\log n}{\mathbf{f}_i(r)}) = O\left(e^i \log n\right)$. □

28: Value of $\mathbf{s(i)}$

Lemma

For any $i > 0$, *we have* $\mathbf{s}(i) = O\left(e^i \log n\right)$.

Proof.

$$\mathbf{f}_i(r) = (1 - 1/r)^{(r-1)ri/(r-1)} \geq \exp\left(-r\tfrac{i}{r-1}\right) = \exp\left(-i - \tfrac{i}{r-1}\right)$$

---

since $\forall m > 0$: $(1 - 1/m)^{m-1} \geq 1/e \geq (1 - 1/m)^m$.

Can assume that $r > \log n \geq i$ (duplicate coordinates)...

$\implies \mathbf{f}_i(r) = \Theta(\exp(-i))$.

$\implies \mathbf{s}(i) = O(\frac{\log n}{\mathbf{f}_i(r)}) = O\left(e^i \log n\right)$. □

28: Value of $\mathbf{s(i)}$

Lemma

For any $i > 0$, *we have* $\mathbf{s}(i) = O\left(e^i \log n\right)$.

Proof.

$$\mathbf{f}_i(r) = (1 - 1/r)^{(r-1)ri/(r-1)} \geq \exp\left(-r\tfrac{i}{r-1}\right) = \exp\left(-i - \tfrac{i}{r-1}\right)$$

---

since $\forall m > 0$: $(1 - 1/m)^{m-1} \geq 1/e \geq (1 - 1/m)^m$.

Can assume that $r > \log n \geq i$ (duplicate coordinates)...

$$\implies \mathbf{f}_i(r) = \Theta(\exp(-i)).$$

$$\implies \mathbf{s}(i) = O\left(\tfrac{\log n}{\mathbf{f}_i(r)}\right) = O\left(e^i \log n\right). \qquad \square$$

28: Value of $\mathbf{s(i)}$

Lemma

For any $i > 0$, *we have* $\mathbf{s}(i) = O\left(e^i \log n\right)$.

Proof.

$$\mathbf{f}_i(r) = (1 - 1/r)^{(r-1)ri/(r-1)} \geq \exp\left(-r\tfrac{i}{r-1}\right) = \exp\left(-i - \tfrac{i}{r-1}\right)$$

---

since $\forall m > 0$: $(1 - 1/m)^{m-1} \geq 1/e \geq (1 - 1/m)^m$.

Can assume that $r > \log n \geq i$ (duplicate coordinates)...

$\implies \mathbf{f}_i(r) = \Theta(\exp(-i))$.

$\implies \mathbf{s}(i) = O(\frac{\log n}{\mathbf{f}_i(r)}) = O\left(e^i \log n\right)$. □

29: Worst case query time

Lemma

For a query point $\boldsymbol{q}$*, the worst case query time is* $O(dn^{1/(1+\epsilon)} \log n)$*, with high probability.*

30: Data-dependent query time

Lemma

$\mathbf{P} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\} \subseteq \mathbb{H}^d$
$\ell_i = \mathbf{d}_H(\boldsymbol{q}, \boldsymbol{x}_i)$, *for* $\boldsymbol{i} = \mathbf{1}, \ldots, \boldsymbol{n}$,
$\boldsymbol{\Delta}$*: smallest value s.t.*

$$\sum_{i=1}^{n} \mathbf{exp}\left(-\frac{\boldsymbol{\Delta}\ell_i}{r}\right) \leq \mathbf{1}.$$

$\Longrightarrow$ *expected query time* $O\left(de^{\boldsymbol{\Delta}} \log n\right)$.

31: Proof

Condition implies: $\gamma_j \leq \gamma_\Delta \leq 1$, for any $j \geq \Delta$.

$\Longrightarrow$ $j \geq \Delta$: $\Gamma_j = \mathbf{E}[X_j] = \gamma_j s(j) \leq s(j)$.

Markov's inequality, $\mathbf{Pr}[X_j \leq 4s(j)] \geq 1 - 1/4$.
Then the algorithm would terminate in this iteration.

$Y_j = 1$: indicator the algorithm reached jth iteration.

Algorithm files in iterations $\Delta, \Delta + 1, \ldots, j - 1$.

$\Longrightarrow$ $\mathbf{Pr}[Y_j = 1] \leq 1/4^{j-\Delta}$.

jth iteration (if happens): $O(ds(j))$ time.

Overall expected running time is $O\left(\mathbf{E}\left[d \sum_{j=1}^{\mathcal{N}} Y_j s(j)\right]\right)$.

31: Proof

Condition implies: $\gamma_j \leq \gamma_\Delta \leq 1$, for any $j \geq \Delta$.

$\implies$ $j \geq \Delta$: $\Gamma_j = \mathbf{E}[X_j] = \gamma_j s(j) \leq s(j)$.

Markov's inequality, $\mathbf{Pr}[X_j \leq 4s(j)] \geq 1 - 1/4$.
Then the algorithm would terminate in this iteration.

$Y_j = 1$: indicator the algorithm reached jth iteration.

Algorithm files in iterations $\Delta, \Delta + 1, \ldots, j - 1$.

$\implies$ $\mathbf{Pr}[Y_j = 1] \leq 1/4^{j-\Delta}$.

---

jth iteration (if happens): $O(ds(j))$ time.

Overall expected running time is $O\left(\mathbf{E}\left[d \sum_{j=1}^{\mathcal{N}} Y_j s(j)\right]\right)$.

31: Proof

Condition implies: $\gamma_j \leq \gamma_\Delta \leq 1$, for any $j \geq \Delta$.

$\implies$ $j \geq \Delta$: $\Gamma_j = \mathbf{E}[X_j] = \gamma_j s(j) \leq s(j)$.

Markov's inequality, $\mathbf{Pr}[X_j \leq 4s(j)] \geq 1 - 1/4$.
Then the algorithm would terminate in this iteration.

$Y_j = 1$: indicator the algorithm reached jth iteration.

Algorithm files in iterations $\Delta, \Delta + 1, \ldots, j - 1$.

$\implies$ $\mathbf{Pr}[Y_j = 1] \leq 1/4^{j-\Delta}$.

---

jth iteration (if happens): $O(ds(j))$ time.

Overall expected running time is $O\left(\mathbf{E}\left[d \sum_{j=1}^{\mathcal{N}} Y_j s(j)\right]\right)$.

31: Proof

Condition implies: $\gamma_j \leq \gamma_\Delta \leq 1$, for any $j \geq \Delta$.

$\implies$ $j \geq \Delta$: $\Gamma_j = \mathbf{E}[X_j] = \gamma_j s(j) \leq s(j)$.

Markov's inequality, $\mathbf{Pr}[X_j \leq 4s(j)] \geq 1 - 1/4$.
Then the algorithm would terminate in this iteration.

$Y_j = 1$: indicator the algorithm reached jth iteration.

Algorithm files in iterations $\Delta, \Delta + 1, \ldots, j - 1$.

$\implies$ $\mathbf{Pr}[Y_j = 1] \leq 1/4^{j-\Delta}$.

jth iteration (if happens): $O(ds(j))$ time.

Overall expected running time is $O\left(\mathbf{E}\left[d \sum_{j=1}^{\mathcal{N}} Y_j s(j)\right]\right)$.

31: Proof

Condition implies: $\gamma_j \leq \gamma_\Delta \leq 1$, for any $j \geq \Delta$.

$\implies$ $j \geq \Delta$: $\Gamma_j = \mathbf{E}[X_j] = \gamma_j s(j) \leq s(j)$.

Markov's inequality, $\mathbf{Pr}[X_j \leq 4s(j)] \geq 1 - 1/4$.
Then the algorithm would terminate in this iteration.

$Y_j = 1$: indicator the algorithm reached jth iteration.

Algorithm files in iterations $\Delta, \Delta + 1, \ldots, j - 1$.

$\implies$ $\mathbf{Pr}[Y_j = 1] \leq 1/4^{j-\Delta}$.

---

jth iteration (if happens): $O(ds(j))$ time.

Overall expected running time is $O\left(\mathbf{E}\left[d \sum_{j=1}^{\mathcal{N}} Y_j s(j)\right]\right)$.

31: Proof

Condition implies: $\gamma_j \leq \gamma_\Delta \leq 1$, for any $j \geq \Delta$.

$\implies$ $j \geq \Delta$: $\Gamma_j = \mathbf{E}[X_j] = \gamma_j s(j) \leq s(j)$.

Markov's inequality, $\mathbf{Pr}[X_j \leq 4s(j)] \geq 1 - 1/4$.
Then the algorithm would terminate in this iteration.

$Y_j = 1$: indicator the algorithm reached jth iteration.

Algorithm files in iterations $\Delta, \Delta + 1, \ldots, j - 1$.

$\implies$ $\mathbf{Pr}[Y_j = 1] \leq 1/4^{j-\Delta}$.

---

jth iteration (if happens): $O(ds(j))$ time.

Overall expected running time is $O\left(\mathbf{E}\left[d \sum_{j=1}^{\mathcal{N}} Y_j s(j)\right]\right)$.

31: Proof

Condition implies: $\gamma_j \leq \gamma_\Delta \leq 1$, for any $j \geq \Delta$.

$\implies$ $j \geq \Delta$: $\Gamma_j = \mathbf{E}[X_j] = \gamma_j s(j) \leq s(j)$.

Markov's inequality, $\mathbf{Pr}[X_j \leq 4s(j)] \geq 1 - 1/4$.
Then the algorithm would terminate in this iteration.

$Y_j = 1$: indicator the algorithm reached jth iteration.

Algorithm files in iterations $\Delta, \Delta + 1, \ldots, j - 1$.

$\implies$ $\mathbf{Pr}[Y_j = 1] \leq 1/4^{j-\Delta}$.

---

jth iteration (if happens): $O(ds(j))$ time.

Overall expected running time is $O\left(\mathbf{E}\left[d \sum_{j=1}^{\mathcal{N}} Y_j s(j)\right]\right)$.

31: Proof

Condition implies: $\gamma_j \leq \gamma_\Delta \leq 1$, for any $j \geq \Delta$.

$\Longrightarrow$ $j \geq \Delta$: $\Gamma_j = \mathbf{E}[X_j] = \gamma_j s(j) \leq s(j)$.

Markov's inequality, $\mathbf{Pr}[X_j \leq 4s(j)] \geq 1 - 1/4$.
Then the algorithm would terminate in this iteration.

$Y_j = 1$: indicator the algorithm reached jth iteration.

Algorithm files in iterations $\Delta, \Delta + 1, \ldots, j - 1$.

$\Longrightarrow$ $\mathbf{Pr}[Y_j = 1] \leq 1/4^{j-\Delta}$.

---

jth iteration (if happens): $O(ds(j))$ time.

Overall expected running time is $O\left(\mathbf{E}\left[d \sum_{j=1}^{\mathcal{N}} Y_j s(j)\right]\right)$.

32: Proof continued...

$$O\left(d\sum_{j=1}^{\Delta} s(j) + d\sum_{j=\Delta+1}^{\mathcal{N}} \frac{s(j)}{4^{j-\Delta}}\right)$$

$$= O\left(de^{\Delta}\log n + de^{\Delta}\sum_{j=\Delta+1}^{\mathcal{N}} \frac{e^{j-\Delta}\log n}{4^{j-\Delta}}\right)$$

$$= O\left(de^{\Delta}\log n\right),$$

using the bound $s(j) = O\left(e^{j}\log n\right)$.

32: Proof continued...

$$O\left(d\sum_{j=1}^{\Delta} s(j) + d\sum_{j=\Delta+1}^{\mathcal{N}} \frac{s(j)}{4^{j-\Delta}}\right)$$

$$= O\left(de^{\Delta}\log n + de^{\Delta}\sum_{j=\Delta+1}^{\mathcal{N}} \frac{e^{j-\Delta}\log n}{4^{j-\Delta}}\right)$$

$$= O\left(de^{\Delta}\log n\right),$$

using the bound $s(j) = O\left(e^{j}\log n\right)$.

32: Proof continued...

$$
\begin{aligned}
& O\left(d \sum_{j=1}^{\Delta} \mathbf{s}(j) + d \sum_{j=\Delta+1}^{\mathcal{N}} \frac{\mathbf{s}(j)}{4^{j-\Delta}} \right) \\
&= O\left(d e^{\Delta} \log n + d e^{\Delta} \sum_{j=\Delta+1}^{\mathcal{N}} \frac{e^{j-\Delta} \log n}{4^{j-\Delta}} \right) \\
&= O\left(d e^{\Delta} \log n \right),
\end{aligned}
$$

using the bound $\mathbf{s}(j) = O\left(e^{j} \log n\right)$.

32: Proof continued...

$$
\begin{aligned}
& O\left(d \sum_{j=1}^{\Delta} \mathbf{s}(j) + d \sum_{j=\Delta+1}^{\mathcal{N}} \frac{\mathbf{s}(j)}{4^{j-\Delta}} \right) \\
&= O\left(d e^{\Delta} \log n + d e^{\Delta} \sum_{j=\Delta+1}^{\mathcal{N}} \frac{e^{j-\Delta} \log n}{4^{j-\Delta}} \right) \\
&= O\big(d e^{\Delta} \log n \big),
\end{aligned}
$$

using the bound $\mathbf{s}(j) = O\big(e^{j} \log n\big)$.

■

33: The result

$\mathbf{P} \subseteq \mathbb{H}^d = \{0,1\}^d$: set of n points,
$\varepsilon > 0$ and r.
Preprocess in $O\left(dn^{1+1/(1+\varepsilon)} \log n\right)$ time/ space.

Given $q \in \mathbb{H}^d$, decide (approximately) if $\mathrm{d}_H(q, \mathbf{P}) \leq r$, in $O\left(dn^{1/(1+\varepsilon)} \log n\right)$.

If query is "easy", DS returns *exact* nearest neighbor.

If $\mathrm{d}_H(q, \mathbf{P}) \leq r$, and $\exists \Delta < (\log n)/(1+\varepsilon)$ s.t. $\sum_{x \in \mathbf{P}} \exp(-\mathrm{d}_H(q,x)\Delta/r) \leq 1$

$\Longrightarrow$ Exact NN in $O\left(d \exp(\Delta) \log n\right)$ expected time.

33: The result

$\mathbf{P} \subseteq \mathbb{H}^d = \{0,1\}^d$: set of n points,
$\varepsilon > 0$ and r.
Preprocess in $O\left(dn^{1+1/(1+\varepsilon)} \log n\right)$ time/ space.

Given $q \in \mathbb{H}^d$, decide (approximately) if $\mathbf{d}_H(q, \mathbf{P}) \leq r$, in $O\left(dn^{1/(1+\varepsilon)} \log n\right)$.

If query is "easy", DS returns *exact* nearest neighbor.

If $\mathbf{d}_H(q, \mathbf{P}) \leq r$, and $\exists \Delta < (\log n)/(1+\varepsilon)$ s.t.
$\sum_{x \in \mathbf{P}} \exp(-\mathbf{d}_H(q,x)\Delta/r) \leq 1$

$\Longrightarrow$ Exact NN in $O\left(d \exp(\Delta) \log n\right)$ expected time.

33: The result

$\mathbf{P} \subseteq \mathbb{H}^d = \{0,1\}^d$: set of n points,
$\varepsilon > 0$ and r.
Preprocess in $O\left(dn^{1+1/(1+\varepsilon)} \log n\right)$ time/ space.

Given $q \in \mathbb{H}^d$, decide (approximately) if $\mathbf{d}_H(q, \mathbf{P}) \leq r$, in $O\left(dn^{1/(1+\varepsilon)} \log n\right)$.

If query is "easy", DS returns *exact* nearest neighbor.

If $\mathbf{d}_H(q, \mathbf{P}) \leq r$, and $\exists \Delta < (\log n)/(1+\varepsilon)$ s.t.
$\sum_{x \in \mathbf{P}} \exp(-\mathbf{d}_H(q,x)\Delta/r) \leq 1$

$\Longrightarrow$ Exact NN in $O\left(d \exp(\Delta) \log n\right)$ expected time.

33: The result

$\mathbf{P} \subseteq \mathbb{H}^d = \{0,1\}^d$: set of n points,
$\varepsilon > 0$ and r.
Preprocess in $O\left(dn^{1+1/(1+\varepsilon)} \log n\right)$ time/ space.

Given $q \in \mathbb{H}^d$, decide (approximately) if $\mathbf{d}_H(q, \mathbf{P}) \leq r$, in $O\left(dn^{1/(1+\varepsilon)} \log n\right)$.

If query is "easy", DS returns *exact* nearest neighbor.

If $\mathbf{d}_H(q, \mathbf{P}) \leq r$, and $\exists \Delta < (\log n)/(1+\varepsilon)$ s.t.
$\sum_{x \in \mathbf{P}} \exp(-\mathbf{d}_H(q,x)\Delta/r) \leq 1$

$\Longrightarrow$ Exact NN in $O\left(d \exp(\Delta) \log n\right)$ expected time.

33: The result

$\mathbf{P} \subseteq \mathbb{H}^d = \{0,1\}^d$: set of n points,
$\varepsilon > 0$ and r.
Preprocess in $O\left(dn^{1+1/(1+\varepsilon)} \log n\right)$ time/ space.

Given $q \in \mathbb{H}^d$, decide (approximately) if $\mathbf{d}_H(q, \mathbf{P}) \leq r$, in $O\left(dn^{1/(1+\varepsilon)} \log n\right)$.

If query is "easy", DS returns *exact* nearest neighbor.

If $\mathbf{d}_H(q, \mathbf{P}) \leq r$, and $\exists \Delta < (\log n)/(1+\varepsilon)$ s.t.
$\sum_{x \in \mathbf{P}} \exp(-\mathbf{d}_H(q, x)\Delta/r) \leq 1$

$\Longrightarrow$ Exact NN in $O\left(d \exp(\Delta) \log n\right)$ expected time.

34: Remarks

1. If data is k dimensional (bounded growth)
   Then query time $O(d \log n)$.
   Known: **[Datar, Immorlica, Indyk, and Mirrokni, 2004]**.
   New DS more general. Handles this case with no modification.
2. Fine tuning the LSH scheme to the hardness of the given data is not a new idea.
   1. **[Andoni, Datar, Immorlica, Indyk, and Mirrokni, 2006]**
   2. Fine tune LSH construction parameters for set of queries, to optimize overall query time.
   3. New DS: adapts parameters on the fly during query process, depending on query.

34: Remarks

1. If data is k dimensional (bounded growth)
   Then query time $O(d \log n)$.
   Known: **[Datar, Immorlica, Indyk, and Mirrokni, 2004]**.
   New DS more general. Handles this case with no modification.
2. Fine tuning the LSH scheme to the hardness of the given data is not a new idea.
   1. **[Andoni, Datar, Immorlica, Indyk, and Mirrokni, 2006]**
   2. Fine tune LSH construction parameters for set of queries, to optimize overall query time.
   3. New DS: adapts parameters on the fly during query process, depending on query.

34: Remarks

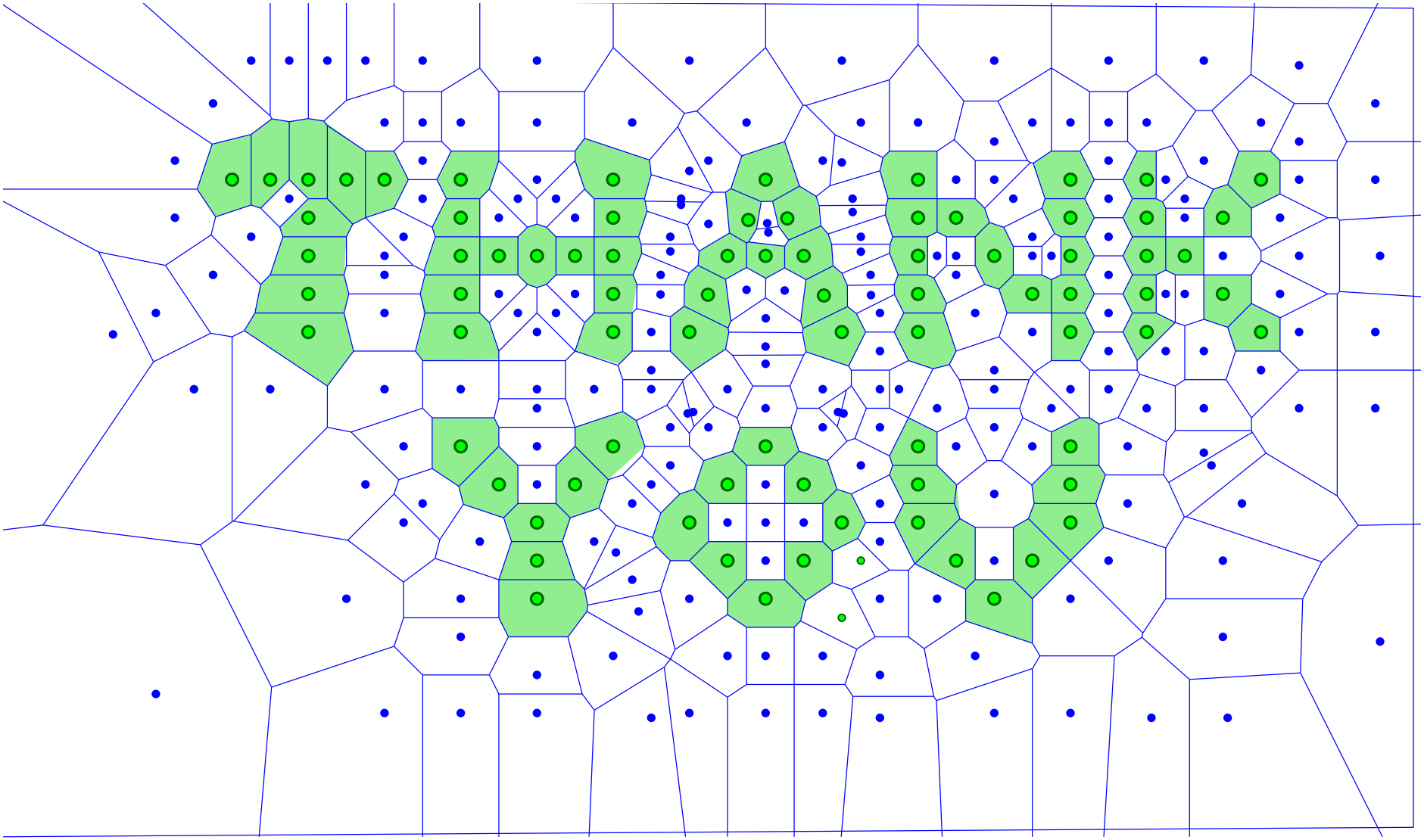
1. If data is k dimensional (bounded growth)
   Then query time $O(d \log n)$.
   Known: **[Datar, Immorlica, Indyk, and Mirrokni, 2004]**.
   New DS more general. Handles this case with no modification.
2. Fine tuning the LSH scheme to the hardness of the given data is not a new idea.
   1. **[Andoni, Datar, Immorlica, Indyk, and Mirrokni, 2006]**
   2. Fine tune LSH construction parameters for set of queries, to optimize overall query time.
   3. New DS: adapts parameters on the fly during query process, depending on query.

34: Remarks

1. If data is k dimensional (bounded growth)
   Then query time $O(d \log n)$.
   Known: **[Datar, Immorlica, Indyk, and Mirrokni, 2004]**.
   New DS more general. Handles this case with no modification.
2. Fine tuning the LSH scheme to the hardness of the given data is not a new idea.
   1. **[Andoni, Datar, Immorlica, Indyk, and Mirrokni, 2006]**
   2. Fine tune LSH construction parameters for set of queries, to optimize overall query time.
   3. New DS: adapts parameters on the fly during query process, depending on query.

34: Remarks

1. If data is k dimensional (bounded growth)
   Then query time $O(d \log n)$.
   Known: **[Datar, Immorlica, Indyk, and Mirrokni, 2004]**.
   New DS more general. Handles this case with no modification.
2. Fine tuning the LSH scheme to the hardness of the given data is not a new idea.
   1. **[Andoni, Datar, Immorlica, Indyk, and Mirrokni, 2006]**
   2. Fine tune LSH construction parameters for set of queries, to optimize overall query time.
   3. New DS: adapts parameters on the fly during query process, depending on query.



A. Andoni, M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing using stable distributions. In T. Darrell, P. Indyk, and G. Shakhnarovich, editors, *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice*, pages 61–72. MIT Press, 2006. URL https://graphics.stanford.edu/courses/cs468-06-fall/Papers/13%20lsh06.pdf.

M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on ***p***-stable distributions. In *Proc. 20th Annu. Sympos. Comput. Geom.* (SoCG), pages 253–262, 2004. doi: 10.1145/997817.997857.

P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proc. 30th Annu. ACM Sympos. Theory Comput.* (STOC), pages 604–613, 1998. doi: 10.1145/276698.276876.