Learning Goals

- Nearest Neighbor Search
- Data structures with Pre-processing
- Reductions
- Streaming model
- ℓ_2 estimate in streaming model

(Approximate) Nearest Neighbor Search

- We are given n points $x_1, \ldots, x_n \in \mathbb{R}^d$

(Approximate) Nearest Neighbor Search

- We are given n points $x_1, \ldots, x_n \in \mathbb{R}^d$
- Task: Given a new point $y \in \mathbb{R}^d$, output $x^* = \mathrm{argmin}_i \, ||x_i - y||$.

(Approximate) Nearest Neighbor Search

- We are given n points $x_1, \ldots, x_n \in \mathbb{R}^d$
- Task: Given a new point $y \in \mathbb{R}^d$, output $x^* = \text{argmin}_i ||x_i - y||$.
- Assume $\min_{i \neq j} ||x_i - x_j|| \geq 1$, and $\max_{i,j} ||x_i - x_j|| \leq R$ for some $R > 0$.

(Approximate) Nearest Neighbor Search

- We are given n points $x_1, \ldots, x_n \in \mathbb{R}^d$
- Task: Given a new point $y \in \mathbb{R}^d$, output $x^* = \operatorname{argmin}_i ||x_i - y||$.
- Assume $\min_{i \neq j} ||x_i - x_j|| \geq 1$, and $\max_{i,j} ||x_i - x_j|| \leq R$ for some $R > 0$.
- Naïve solution: go over all data points, in time $O(nd)$.

(Approximate) Nearest Neighbor Search

- We are given n points $x_1, \ldots, x_n \in \mathbb{R}^d$
- Task: Given a new point $y \in \mathbb{R}^d$, output $x^* = \text{argmin}_i \, ||x_i - y||$.
- Assume $\min_{i \neq j} ||x_i - x_j|| \geq 1$, and $\max_{i,j} ||x_i - x_j|| \leq R$ for some $R > 0$.
- Naïve solution: go over all data points, in time $O(nd)$.
- In an *ϵ-approximate Nearest Neighbor* problem, given $y \in \mathbb{R}^d$, we must return $x^* \in \{x_1, \ldots, x_n\}$ such that $||y - x^*|| \leq (1 + \epsilon) \min_i ||y - x_i||$.

(Approximate) Nearest Neighbor Search

- We are given n points $x_1, \ldots, x_n \in \mathbb{R}^d$
- Task: Given a new point $y \in \mathbb{R}^d$, output $x^* = \text{argmin}_i \, ||x_i - y||$.
- Assume $\min_{i \neq j} ||x_i - x_j|| \geq 1$, and $\max_{i,j} ||x_i - x_j|| \leq R$ for some $R > 0$.
- Naïve solution: go over all data points, in time $O(nd)$.
- In an *ϵ-approximate Nearest Neighbor* problem, given $y \in \mathbb{R}^d$, we must return $x^* \in \{x_1, \ldots, x_n\}$ such that $||y - x^*|| \leq (1 + \epsilon) \min_i ||y - x_i||$.
- Goal: running time $O(d, \log n, 1/\epsilon)$.

Point Location in Equal Balls

We reduce ϵ-approximate nearest neighbor problem to the following problem:

Definition (Point Location in Equal Balls, ϵ-PLEB(r))

We are given n points $x_1, \ldots, x_n \in \mathbb{R}^d$ and radius r. Let $B(x, r) := \{z \in \mathbb{R}^d : ||z - x|| \leq r\}$ denote the Euclidean ball of radius r around x. Given a query point $y \in \mathbb{R}^d$:

- If there exists x_i such that $y \in B(x_i, r)$, we must return YES and an x_j such that $y \in B(x_j, (1 + \epsilon)r)$;
- If there exists no x_i such that $y \in B(x_i, (1 + \epsilon)r)$, we must return NO.
- Otherwise, we can say either YES or NO. If we return YES, we must also return an x_j such that $y \in B(x_j, (1 + \epsilon)r)$.

Reduction from ϵ-NN to PLEB

Claim

Given an algorithm $\mathcal{A}$ that solves ϵ-PLEB(r), we can solve ϵ-NN with $O(\log(\log R/\epsilon))$ calls to $\mathcal{A}$.

Reduction from ϵ-NN to PLEB

Claim

Given an algorithm $\mathcal{A}$ that solves ϵ-PLEB(r), we can solve ϵ-NN with $O(\log(\log R/\epsilon))$ calls to $\mathcal{A}$.

Proof.

We can do a binary search with an ϵ-PLEB(r) oracle and find an r^* such that ϵ-PLEB$(\frac{r^*}{1+\epsilon})$ returns No and ϵ-PLEB(r^*) returns YES with an x^*. This takes $\log(\log_{1+\epsilon} R) = O(\log(\frac{\log R}{\epsilon}))$ calls.

Reduction from ϵ-NN to PLEB

Claim

Given an algorithm $\mathcal{A}$ that solves ϵ-PLEB(r), we can solve ϵ-NN with $O(\log(\log R/\epsilon))$ calls to $\mathcal{A}$.

Proof.

We can do a binary search with an ϵ-PLEB(r) oracle and find an r^* such that ϵ-PLEB$(\frac{r^*}{1+\epsilon})$ returns No and ϵ-PLEB(r^*) returns YES with an x^*. This takes $\log(\log_{1+\epsilon} R) = O(\log(\frac{\log R}{\epsilon}))$ calls.
We then know $\min_j ||y - x_j|| \geq \frac{r^*}{1+\epsilon}$, and $||y - x^*|| \leq r^*(1+\epsilon)$. So $||y - x^*|| \leq (1+\epsilon)^2 \min_j ||y - x_j|| \leq (1+2\epsilon) \min_j ||y - x_j||$ for $\epsilon < 1$. □

Solving PLEB

Plan of attack:

1. Give a brute-force algorithm with *pre-processing*
2. Use JL-transform and run the brute-force algorithm in the low dimensional space

Solving PLEB

Plan of attack:

1. Give a brute-force algorithm with *pre-processing*
2. Use JL-transform and run the brute-force algorithm in the low dimensional space

Step 1: Brute-force algorithm for PLEB

Solving PLEB

Plan of attack:

1. Give a brute-force algorithm with *pre-processing*
2. Use JL-transform and run the brute-force algorithm in the low dimensional space

Step 1: Brute-force algorithm for PLEB

- Pre-processing:
  - Divide $\mathbb{R}^d$ into small cuboids with side length $\frac{\epsilon r}{\sqrt{d}}$.
    - The idea is that the longest distance between any two points in a cube is ϵr.
  - Create a hash table. For each x_i, and for each cuboid C that intersects with $B(x_i, r)$, hash the pair (C, x_i).
    - C is the *key*, x_i is the *satellite*
- Query:
  - To query y, calculate the cuboid C to which y belongs; query key value C.

Solving PLEB

Plan of attack:

1. Give a brute-force algorithm with *pre-processing*
2. Use JL-transform and run the brute-force algorithm in the low dimensional space

Step 1: Brute-force algorithm for PLEB

- Pre-processing:
  - Divide $\mathbb{R}^d$ into small cuboids with side length $\frac{\epsilon r}{\sqrt{d}}$.
    - The idea is that the longest distance between any two points in a cube is ϵr.
  - Create a hash table. For each x_i, and for each cuboid C that intersects with $B(x_i, r)$, hash the pair (C, x_i).
    - C is the *key*, x_i is the *satellite*
- Query:
  - To query y, calculate the cuboid C to which y belongs; query key value C.
  - If (C, x_i) exists in the hash table, return Yes and x_i; otherwise return No.

Analysis of Pre-processing

- Correctness:
  - When we return YES and x_i, we know for some point $y' \in C$, $||x - y'|| \leq r$, so $||x - y|| \leq ||x - y'|| + ||y' - y|| \leq (1 + \epsilon)r$.

Analysis of Pre-processing

- Correctness:
  - When we return YES and x_i, we know for some point $y' \in C$, $||x - y'|| \leq r$, so $||x - y|| \leq ||x - y'|| + ||y' - y|| \leq (1 + \epsilon)r$.
  - When we return NO, we know for all x_i, $||y - x_i|| \geq r$ (otherwise (C, x_i) should have been hashed).

Analysis of Pre-processing

- Correctness:
  - When we return Yes and x_i, we know for some point $y' \in C$, $||x - y'|| \leq r$, so $||x - y|| \leq ||x - y'|| + ||y' - y|| \leq (1 + \epsilon)r$.
  - When we return No, we know for all x_i, $||y - x_i|| \geq r$ (otherwise (C, x_i) should have been hashed).
- Running time:
  - Preprocessing: the volume of $B(x_i, r)$ is $2^{O(d)} r^d / d^{d/2}$; the volume of each cuboid is $(\frac{\epsilon r}{\sqrt{d}})^d$; so for each x_i hash $O(\frac{1}{\epsilon})^d$ cuboids.

Analysis of Pre-processing

- Correctness:
  - When we return Yes and x_i, we know for some point $y' \in C$, $||x - y'|| \leq r$, so $||x - y|| \leq ||x - y'|| + ||y' - y|| \leq (1 + \epsilon)r$.
  - When we return No, we know for all x_i, $||y - x_i|| \geq r$ (otherwise (C, x_i) should have been hashed).
- Running time:
  - Preprocessing: the volume of $B(x_i, r)$ is $2^{O(d)} r^d / d^{d/2}$; the volume of each cuboid is $(\frac{\epsilon r}{\sqrt{d}})^d$; so for each x_i hash $O(\frac{1}{\epsilon})^d$ cuboids.
    - For even d, the volume of a radius r Euclidean ball is $\frac{\pi^{d/2}}{(\frac{d}{2})!} r^d$.

Analysis of Pre-processing

- Correctness:
  - When we return YES and x_i, we know for some point $y' \in C$, $||x - y'|| \leq r$, so $||x - y|| \leq ||x - y'|| + ||y' - y|| \leq (1 + \epsilon)r$.
  - When we return NO, we know for all x_i, $||y - x_i|| \geq r$ (otherwise (C, x_i) should have been hashed).
- Running time:
  - Preprocessing: the volume of $B(x_i, r)$ is $2^{O(d)} r^d / d^{d/2}$; the volume of each cuboid is $(\frac{\epsilon r}{\sqrt{d}})^d$; so for each x_i hash $O(\frac{1}{\epsilon})^d$ cuboids.
    - For even d, the volume of a radius r Euclidean ball is $\frac{\pi^{d/2}}{(\frac{d}{2})!} r^d$.
  - Query: Computing C takes time $O(d)$. Querying the hash table takes time $O(1)$.

Analysis of Pre-processing

- Correctness:
  - When we return YES and x_i, we know for some point $y' \in C$, $||x - y'|| \le r$, so $||x - y|| \le ||x - y'|| + ||y' - y|| \le (1 + \epsilon)r$.
  - When we return NO, we know for all x_i, $||y - x_i|| \ge r$ (otherwise (C, x_i) should have been hashed).
- Running time:
  - Preprocessing: the volume of $B(x_i, r)$ is $2^{O(d)} r^d / d^{d/2}$; the volume of each cuboid is $(\frac{\epsilon r}{\sqrt{d}})^d$; so for each x_i hash $O(\frac{1}{\epsilon})^d$ cuboids.
    - For even d, the volume of a radius r Euclidean ball is $\frac{\pi^{d/2}}{(\frac{d}{2})!} r^d$.
  - Query: Computing C takes time $O(d)$. Querying the hash table takes time $O(1)$.
  - Query time is satisfactory, but pre-processing time is exponential in d!

Step 2: Dimension Reduction

- Using JL-transform, we can first map $x_1, \ldots, x_n$ to $z_1, \ldots, z_n \in \mathbb{R}^t$ where $t = O(\log n/\epsilon^2)$.

Step 2: Dimension Reduction

- Using JL-transform, we can first map $x_1, \ldots, x_n$ to $z_1, \ldots, z_n \in \mathbb{R}^t$ where $t = O(\log n/\epsilon^2)$.
- When querying $y \in \mathbb{R}^d$, first map it to $y' \in \mathbb{R}^t$ with the same random matrix. With high probability, $(1-\epsilon)||y' - z_i|| \leq ||y - x_i|| \leq (1+\epsilon)||y' - z_i||$ for every i.

Step 2: Dimension Reduction

- Using JL-transform, we can first map $x_1, \ldots, x_n$ to $z_1, \ldots, z_n \in \mathbb{R}^t$ where $t = O(\log n/\epsilon^2)$.
- When querying $y \in \mathbb{R}^d$, first map it to $y' \in \mathbb{R}^t$ with the same random matrix. With high probability, $(1-\epsilon)||y' - z_i|| \leq ||y - x_i|| \leq (1+\epsilon)||y' - z_i||$ for every i.
- Pre-processing now takes time $O(1/\epsilon)^t = n^{\log(1/\epsilon)/\epsilon^2}$.

Step 2: Dimension Reduction

- Using JL-transform, we can first map $x_1, \ldots, x_n$ to $z_1, \ldots, z_n \in \mathbb{R}^t$ where $t = O(\log n/\epsilon^2)$.
- When querying $y \in \mathbb{R}^d$, first map it to $y' \in \mathbb{R}^t$ with the same random matrix. With high probability, $(1-\epsilon)||y' - z_i|| \leq ||y - x_i|| \leq (1+\epsilon)||y' - z_i||$ for every i.
- Pre-processing now takes time $O(1/\epsilon)^t = n^{\log(1/\epsilon)/\epsilon^2}$.
- Each query for PLEB takes time $O(td) = O(\frac{d \log n}{\epsilon^2})$.

Step 2: Dimension Reduction

- Using JL-transform, we can first map $x_1, \ldots, x_n$ to $z_1, \ldots, z_n \in \mathbb{R}^t$ where $t = O(\log n/\epsilon^2)$.
- When querying $y \in \mathbb{R}^d$, first map it to $y' \in \mathbb{R}^t$ with the same random matrix. With high probability, $(1-\epsilon)||y' - z_i|| \leq ||y - x_i|| \leq (1+\epsilon)||y' - z_i||$ for every i.
- Pre-processing now takes time $O(1/\epsilon)^t = n^{\log(1/\epsilon)/\epsilon^2}$.
- Each query for PLEB takes time $O(td) = O(\frac{d \log n}{\epsilon^2})$.

The whole picture for solving ϵ-NN:

- Preprocessing time:

$$n^{O(\log(1/\epsilon)/\epsilon^2)} \cdot O\left(\frac{\log R}{\epsilon}\right).$$

Step 2: Dimension Reduction

- Using JL-transform, we can first map $x_1, \ldots, x_n$ to $z_1, \ldots, z_n \in \mathbb{R}^t$ where $t = O(\log n/\epsilon^2)$.
- When querying $y \in \mathbb{R}^d$, first map it to $y' \in \mathbb{R}^t$ with the same random matrix. With high probability, $(1-\epsilon)||y' - z_i|| \le ||y - x_i|| \le (1+\epsilon)||y' - z_i||$ for every i.
- Pre-processing now takes time $O(1/\epsilon)^t = n^{\log(1/\epsilon)/\epsilon^2}$.
- Each query for PLEB takes time $O(td) = O(\frac{d \log n}{\epsilon^2})$.

The whole picture for solving ϵ-NN:

- Preprocessing time:

$$n^{O(\log(1/\epsilon)/\epsilon^2)} \cdot O\left(\frac{\log R}{\epsilon}\right).$$

- Query time: $O(td) + O(\log(\frac{\log R}{\epsilon})) \cdot O(t)$.