Nearest Neighbor Search (NNS)

Dataset: \( X \subseteq (\mathbb{R}^d, \|\|_1) \), \( |X| = n \)
\[
\log n \leq d \leq n^{\frac{3}{2}(\epsilon)}
\]

Query: \( q \in \mathbb{R}^d \)

Goal: return \( p \in X \) closest to \( q \) (wrt \( \|\|_1 \) or \( \|\|_2 \))

Motivation:

Similarity Search

database of \( n \) objects
given a query object, retrieve \( k \) most similar ones, quickly (focus on \( k=1 \)).

Feature representation:

Objects \( \leftrightarrow \) Points in \( \mathbb{R}^d \)

Similarity \( \leftrightarrow \) Norm in \( \mathbb{R}^d \)

Similarity search \( \leftrightarrow \) Nearest neighbor queries

Important cases:

1) \( \|\|_1 \), \( X \subseteq \mathbb{R}^d \), \( q \in \mathbb{R}^d \) - \( l_1 \) case

2) \( \|\|_1 \), \( X \subseteq \{0,1\}^d \), \( q \in \{0,1\}^d \) - Hamming case

3) \( \|\|_2 \), \( X \subseteq \mathbb{R}^d \), \( q \in \mathbb{R}^d \) - \( l_2 \) case.
$4) \| \cdot \|_2, X \subset S^{d-1}, q \in S^{d-1}$

Parameters:
1. Space (ideally, $O(nd)$)
2. Query time ($O(d)$)

Linear scan:
1. Space $O(nd)$
2. Query time $O(nd)$

Curse of dimensionality:
1. All the known data structures faster than linear scan require space $2^{\Omega(d)}$ (infeasible for $d \gg \log n$).
2. A data structure with preprocessing time $\text{poly}(n,d)$ and query time $\text{poly}(d) \times n^{0.99}$ would refute "Strong Exponential Time Hypothesis".

Approximation to the rescue!

Definition (Approximate NNS, ANN)

Dataset: $X \subset \mathbb{R}^d, |X| = n, d \leq n^{o(1)}$

Query: $q \in \mathbb{R}^d$

Goal: $\hat{p} \in X$ s.t. $\|q - \hat{p}\| \leq c \cdot \min_{p^* \in X} \|q - p^*\|$
When $c > 1$, curse of dimensionality approximation goes away!

Is approximation bad in practice?
* Features are a bit arbitrary anyway
* Techniques for ANN can be used to solve NNS when distances are distributed nicely
* Curse of dimensionality holds mostly for "uninteresting" instances.

We will show: ANN for $\{0,1\}^d$ with
1. Space: $O(n^{1+1/c} \cdot \text{poly}(d))$ (for $c = 2$, $\sqrt{n}$)
2. Query time: $O(n^{1/c} \cdot \text{poly}(d))$
3. Probability of success $\geq 0.9$
(for a fixed query, over preprocessing and query stages).

**Step 1: Reduction to a single distance scale**

* Promise: $q$ is within $sr$ from $X$
* Goal: find $p \in X$ within $cr$ from $q$
* New parameter: $0 \leq r \leq d$ (known in
Enough to handle a fixed scale with:
- Space: $O(n^{1+\frac{1}{c}} \cdot \text{poly}(d))$.
- Query time: $O(n^{\frac{1}{d}} \cdot \text{poly}(d))$.
- Probability of success: $1 - \frac{1}{100^d}$.

(d such data structures, querying log d of them).


Index data points and queries wrt. $S$.

Intuition: approximate match $\leftrightarrow$ exact match on $S$.

Data structure: hash table that given $q \in \{0, 1\}^d$, can retrieve all $p \in \mathbb{X}$ s.t.

$p|_S = q|_S$. 
Query: retrieve \( p \in X \) s.t. \( \| p - q \|_s = q \|_s \)
stop as soon as find \( p \) s.t.
\[ \| p - q \|_s \leq cr. \]

Space: \( O(nd) \)

Analysis

\[ \mathbb{E}[\# \text{ far points matching the query}] \leq n \left(1 - \frac{cr}{d}\right)^K \quad (*) \]

Choose smallest \( K \) s.t. \( (*) \leq 1 \)

\[ K = \left\lceil \frac{\log n}{\log \left(1 - \frac{cr}{d}\right)^{-1}} \right\rceil \]

For this \( K \), query time is \( O(d) \) in expectation

NB: Won't necessarily succeed.

Probability of success

\[ \geq \Pr \left[ \text{match with a point within } r \right] \geq \left(1 - \frac{r}{d}\right)^K \geq n \frac{\log \left(1 - \frac{r}{d}\right)^{-1}}{\log \left(1 - \frac{cr}{d}\right)^{-1}} \cdot \left(1 - \frac{r}{d}\right) \]
\[ \log \left( 1 - \frac{r}{d} \right)^{-1} \approx \frac{r/d}{cr/d} = \frac{1}{c} \] (always a valid upper bound).

\((1 - \frac{r}{d}) \geq \frac{1}{d}\)

Overall: \(\geq n^{-1/c} \cdot \frac{1}{d}\)

Repeat \(L = O(d \cdot n^{1/c})\) times to get prob. \(\geq 1 - \frac{1}{d}\log d\)

Space: \(O(n^{1+1/c} \cdot d^2 \log d)\)

Query time: \(O(n^{1/c} \cdot d^2 \log d)\)

Overall data structure:

Sample \(L\) sets \(S_1, \ldots, S_L\) of size \(K\).

Query: \(q \in \{0, 1\}^d\)

enumerate \(p \in X\) s.t. \(\exists i : p | S_i = q | S_i\) (using \(L\) hash tables).

Stop as soon as find \(p : \|q - p\|_1 \leq cr\).
This was an instantiation of **Locality-Sensitive Hashing (LSH)**

**Def** Distribution over partitions $\mathcal{P}$ is locality-sensitive with parameters $p_1, p_2$ if $\forall x_1, x_2$:

1) $\|x_1 - x_2\| \leq r \Rightarrow \Pr[\mathcal{P}(x_1) = \mathcal{P}(x_2)] \geq p_1$

2) $\|x_1 - x_2\| > cr \Rightarrow \Pr[\mathcal{P}(x_1) = \mathcal{P}(x_2)] < p_2$

**Example** $\mathcal{P}$-partition of $\{0, 1\}^d$:

$\{x| x_i = 0\} \cup \{x| x_i = 1\}$ for random $i$.

**Th** If $\mathcal{P}$ is LSH with $p_1, p_2$, then

3ANN data structure with:

1) Space $O(n^{1+p}/p_1) +$ dataset + $O(n^p/p_1)$ partitions

2) Query time $O(n^p/p_1)$ distances + $O(n^p/p_1)$ point locations

3) Probability of success $\geq 0.9$

$$p = \frac{\log p_1^{-1}}{\log p_2^{-1}} \quad (\leq \frac{1}{c} \text{ for } \{0, 1\}^d).$$