**k-Nearest Neighbor Search (k-NNS).**

**Dataset:** \( X \subseteq (\mathbb{R}^d, \| \cdot \|_1) \ | X \mid = n \)

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

**Goal:** return \( k \) points from \( X \) closest to the query \( q \).

**Motivation:**
- Similarity search
- Database of \( n \) objects
- Given a query object, retrieve \( k \) most similar objects from the database, quickly

**Feature representation:**
- Objects \( \leftrightarrow \) Points in \( \mathbb{R}^d \)
- Similarity \( \leftrightarrow \) Norm over \( \mathbb{R}^d \)
- Similarity search queries \( \leftrightarrow \) Nearest Neighbor Queries

**Important cases:**
- \( \| \cdot \|_1 \quad (\| x \|_1 = \sum |x_i|) \quad \ell_1 \)
- \( \| \cdot \|_2 \quad (\| x \|_2 = (\sum x_i^2)^{1/2}) \quad - \text{focus today} \ell_2 \)
Parameters:  

- Space $(O(nd))$, ideally  
- Query time $(O(d))$

Linear scan

Curse of dimensionality:

- All the known NNS data structures faster than linear scan require $2^{n^{2(d)}}$ space (infeasible for $d \gg \log n$).
- 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".

What to do? Theory and "practice" give different but related answers. Theory: let's allow answers that are approximate.

Practice: let's require exact answers, but only for nice instances (e.g., when nearest neighbors are noticeably closer to the query than a typical point).
Curse of dimensionality holds only for uninteresting instances.

Approximate NNS: lots of work. See:
- Piotr's ICM 2018 paper
- My class notes

Today: practical algorithms (that use ML). Objective function: fraction of k-NN returned correctly

Modern pipeline:

\[
\begin{align*}
\text{query } q \in \mathbb{R} & \quad \xrightarrow{} \quad \text{index} \\
& \quad \xrightarrow{\text{list of candidate points}} \quad \text{sketches} \\
& \quad \xrightarrow{T \ll n} \quad \text{shorter list} \\
& \quad \xrightarrow{T' \ll T} \quad \text{exact dist.} \\
& \quad \xrightarrow{} \quad k \text{ points}
\end{align*}
\]

Sketches:

\[
\begin{align*}
q & \rightarrow \text{sk}(q) \\
\Uparrow & \quad \text{crude estimate} \\
x & \rightarrow \text{sk}(x) \\
& \quad \|q - x\|_2 \quad \text{very quickly}
\end{align*}
\]

Plan:
- Overview of sketches (random projections, ITQ, PQ).
- Overview of indexing techniques +
Fresh results on ML-based approaches.

Sketches (aka learn-to-hash).
Two major groups:
1. Estimator is Hamming distance.
   \[ x \in \mathbb{R}^d \overset{??}{\rightarrow} \{0,1\}^s \quad + : \text{Very fast} \]
   \[ y \in \mathbb{R}^d \overset{??}{\rightarrow} \{0,1\}^s \quad - : \text{Not so accurate} \]
2. Quantization-based sketches
   + : Can be pretty accurate
   - : Relatively slow.

- Random rotation sketch: \( \text{sgn}(Ax) \) where \( A \) is random orthogonal matrix

- Center
- More than \( d \) bits \( \Rightarrow \) repeat
- Can use FFT to speed-up rotations.

- PCA+RR
  - first project on top PCA directions
Learn the rotation

\[ A \text{-orth st.} \]
\[ E \| \text{sgn}(Ax) - x \|_2^2 \rightarrow \min \]

\[ \implies \]
\[ E \| B_x - A x \|_2^2 \rightarrow \min \]

\[ B_x = \text{sgn}(Ax) \]
Fix \( B_x \) update \( A \)

\[ \Rightarrow \text{can be solved using SVD.} \]

Product quantization (PQ) \[ \text{[Jegou, Douze, Schmid 2011]} \]

\[ ||q - p||_2^2 \approx ||q - c_p||_2^2 \]

\[ k \text{-means} \]

\[ B \text{ blocks} \]

\[ k_B \text{ centers total} \]

\[ k \text{-means in each} \]
Estimation:
\[ O(dk) + O(B) \] per point.

Indexing:

- Focus on partition-based (graph-based).
- \( P \)-partition of \( \mathbb{R}^d \)

Consider data points in the part with the query (and nearby parts, multi-probe).

Want:
- Look up few points
- Most nearest neighbors are in the same or one of the close parts
- Algorithmically simple
- Approximately balanced wrt dataset
Often, built from several simple partitions tree of hyperplanes $2^t$ parts dot products.

Focus on a single partition
- Locality-Sensitive Hashing (data-independent)
- $k$-means
- Random projection
- PCA tree

[Dong, Indyk, R., Wagner 2019]: new approach
- Graph partitioning ($k$-NN graph)
- Supervised learning (to extend to $\mathbb{R}^d$ in a "simple" way).

Highlights:
- Optimize obj function directly
- Unsup $\rightarrow$ Sup.
- Any learning
- Balanced
- Soft labels
Open problems:
  • Other metrics
  • Learn graph-based approaches.