From NNS to MIPS

CS 5968/6968: Data Str & Alg for Scalable Computing Spring 2023

Benwei Shi

University of Utah

2023-02-27

Learning Outcomes

Nearest neighbor

What is the nearest neighbor?

Nearest neighbor

What is the nearest neighbor?

Definition (Nearest Neighbor)

Given a universe Ω and a distance function $D : \Omega^2 \to \mathbb{R}$, the nearest neighbor of a query point $q \in \Omega$ in a finite set of points $P \subseteq \Omega$ is,

$$p^* = \arg\min_{p_i \in P} D(p_i, q)$$

Nearest neighbor

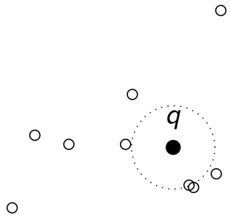
What is the nearest neighbor?

Definition (Nearest Neighbor)

Given a universe Ω and a distance function $D : \Omega^2 \to \mathbb{R}$, the nearest neighbor of a query point $q \in \Omega$ in a finite set of points $P \subseteq \Omega$ is,

$$p^* = \arg\min_{p_i \in P} D(p_i, q)$$

For example, $\Omega = \mathbb{R}^2$



Nearest neighbor search (NNS)

Preprocess P so that one can efficiently find the nearest neighbor of q in P.

Nearest neighbor search (NNS)

Preprocess P so that one can efficiently find the nearest neighbor of q in P.

- Reasonable preprocessing time: $\text{poly}(n, d)$

Nearest neighbor search (NNS)

Preprocess P so that one can efficiently find the nearest neighbor of q in P.

- Reasonable preprocessing time: $\mathrm{poly}(n, d)$
- Fast query time: $\mathrm{poly}(\log n, d)$

Similarity search?

- Similarity is an opposing concept to the distance.
- Similar elements should have small distance.
- Far apart elements should not be similar.

Similarity search?

- Similarity is an opposing concept to the distance.
- Similar elements should have small distance.
- Far apart elements should not be similar.
- e.g., the Gaussian kernel similarity $K(p, q) = \exp\left(-\frac{D(p,q)^2}{2\sigma^2}\right)$. When distance $D = 0$, the kernel is 1 and when distance D is large, the kernel is almost 0.

Similarity search?

- Similarity is an opposing concept to the distance.
- Similar elements should have small distance.
- Far apart elements should not be similar.
- e.g., the Gaussian kernel similarity $K(p,q) = \exp\left(-\frac{D(p,q)^2}{2\sigma^2}\right)$. When distance $D = 0$, the kernel is 1 and when distance D is large, the kernel is almost 0.
- Minimizing the distance is equivalent to maximizing the similarity.

Similarity search?

- Similarity is an opposing concept to the distance.
- Similar elements should have small distance.
- Far apart elements should not be similar.
- e.g., the Gaussian kernel similarity $K(p, q) = \exp\left(-\frac{D(p,q)^2}{2\sigma^2}\right)$. When distance $D = 0$, the kernel is 1 and when distance D is large, the kernel is almost 0.
- Minimizing the distance is equivalent to maximizing the similarity.
- When the objective is to maximizing a similarity function, we call it similarity search.

Common distance functions

- For $\mathbb{R}^d$:
  - L_p distance: $L_p(p,q) := \left(\sum_{i=1}^{d} |p_i - q_i|^p\right)^{1/p}$ for $p \geq 1$. When $p = 2$ we get the Euclidean distance. Other common choices are $p = 1$ (Manhattan distance) and $p = \infty$ (Chebyshev distance).
  - Cosine similarity: $S_C(p,q) = \frac{p^T q}{\|p\|_2 \|q\|_2}$.
  - Dot product (similarity): $S_D(p,q) = p^T q$.
- For strings:
  - Edit distance, Hamming distance.
- For the power set of a finite set:
  - Jacard similarity, Jacard distance,
- For nodes in a graph:
  - Length of the shortest path.
- For point sets:
  - Kernel distance, Gromov-Hausdorff distance.

Applications

What can you do with an efficient NNS?

Applications

What can you do with an efficient NNS?

- Learning: Pattern recognition, prediction, classification
- Matching: DNA sequencing, point cloud registration, compression, clustering
- Searching: information retrieval, web searching, map searching, recommendation, plagiarism detection

Applications

What can you do with an efficient NNS?

- Learning: Pattern recognition, prediction, classification
  - Cons: Need to store all the data points.
  - Model based learning methods are better at this.
- Matching: DNA sequencing, point cloud registration, compression, clustering
- Searching: information retrieval, web searching, map searching, recommendation, plagiarism detection

Applications

What can you do with an efficient NNS?

- Learning: Pattern recognition, prediction, classification
  - Cons: Need to store all the data points.
  - Model based learning methods are better at this.
- Matching: DNA sequencing, point cloud registration, compression, clustering
  - These are NNS problems in general.
  - Domain specific data $\Rightarrow$ Specical algorithms.
- Searching: information retrieval, web searching, map searching, recommendation, plagiarism detection

Applications

What can you do with an efficient NNS?

- Learning: Pattern recognition, prediction, classification
  - Cons: Need to store all the data points.
  - Model based learning methods are better at this.
- Matching: DNA sequencing, point cloud registration, compression, clustering
  - These are NNS problems in general.
  - Domain specific data $\Rightarrow$ Specical algorithms.
- Searching: information retrieval, web searching, map searching, recommendation, plagiarism detection
  - A lot of data, but we have to store them anyway.
  - Vector data (low/high dimensional).
  - Efficient NNS system is crucial.

Applications

What can you do with an efficient NNS?

- Learning: Pattern recognition, prediction, classification
  - Cons: Need to store all the data points.
  - Model based learning methods are better at this.
- Matching: DNA sequencing, point cloud registration, compression, clustering
  - These are NNS problems in general.
  - Domain specific data $\Rightarrow$ Specical algorithms.
- Searching: information retrieval, web searching, map searching, recommendation, plagiarism detection
  - A lot of data, but we have to store them anyway.
  - Vector data (low/high dimensional).
  - Efficient NNS system is crucial.

I'd see NNS as a searching problem, as the name suggests.

Applications

The rest lecture mostly focuses on the most common data space, d-dimensional vectors, i.e. $\Omega = \mathbb{R}^d$.

$d = 1$

- Sort the data points. Binary search.
  - Space: $O(n)$; query time: $O(\log n)$.
- Balanced binary search tree.
  - Space: $O(n)$; query time: $O(\log n)$.
- vEB-tree, x-fast trie...
  - Space: $O(n)$; query time: $O(\log \log u)$.

Space: $O(n)$; query time: $O(\log n)$.

$d = 2$

- Voronoi diagram
  - Space: $O(n)$; query time: $O(\log n)$. [Lipton-Tarjan'80]
- kd-Tree
  - Space: $O(n)$; query time: $O(\log n)$. [Lipton-Tarjan'80]

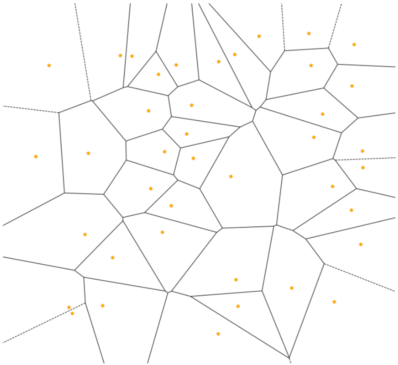


Image by Hristo Hristov

Space: $O(n)$; query time: $O(\log n)$ on average.

Curse of dimensionality

- All known data structures that beat $O(dn)$ linear scan query time require $O(2^d)$ space.
- Observation (Number of partitions). Splits the data space in each dimension into two halves, end with 2^d partitions.
- Think $n = 10^6$, $d = 100$.

Point Location in Equal Balls (PLEB)

Definition (ε-NN)

Given a point set P and a query point q, we say $\hat{p} \in P$ is an ε-NN of q in P if

$$D(\hat{p}, q) \leq (1+\varepsilon) \min_{p \in P} D(p, q)$$

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

Given a point set P and $r \in \mathbb{R}^+$, for any query point q,

- if $\min_{p \in P} D(p, q) \leq r$, return a point p' s.t. $D(p', q) \leq (1+\varepsilon)r$.
- if $(1+\varepsilon)r \leq \min_{p \in P} D(p, q)$, return $\emptyset$.
- if $r \leq \min_{p \in P} D(p, q) \leq (1+\varepsilon)r$, return p' or $\emptyset$.

Indyk-Motwani'98

Algorithm 1 Preprocess

Input: point set $P \subset \{0,1\}^d$, size ℓ
i.i.d. drawn $H \leftarrow \mathcal{H}^{\ell \times k}$
ℓ hash tables $T \leftarrow \{\} * \ell$
for $p \in P$ **do**
  for $j = 1$ to ℓ **do**
    $T_j[(H_{j,1}(p), \ldots, H_{j,k}(p))] \leftarrow p$
return T, H

$k = O(\log n), \ell = O(n^{1/\varepsilon})$
space: $O(nd + n^{1+1/\varepsilon})$
Where $\mathcal{H}$ is a family of LSH.

Algorithm 2 Query

Input: ℓ hash tables T, $\ell \times k$ hash functions H, query point $q \in \{0,1\}^d$
$A \leftarrow \emptyset$
for $j = 1$ to ℓ **do**
  $p \leftarrow T_j[(H_{j,1}(q), \ldots, H_{j,k}(q))]$
  if $D(q,p) \leq r_2$ **then**
    return p

time: $O(n^{1/\varepsilon} d)$

Indyk-Motwani'98 - Result

Definition (Locality-Sensitive Hashing)

A family $\mathcal{H} = \{h : \Omega \to S\}$ is called (r_1, r_2, p_1, p_2)-sesitive for metric D if for any $q, p \in \Omega$

- if $D(p, q) \le r_1$ then $\Pr_{h \sim \mathcal{H}}[h(q) = h(p)] \ge p_1$,
- if $D(p, q) \ge r_2$ then $\Pr_{h \sim \mathcal{H}}[h(q) = h(p)] \le p_2$,

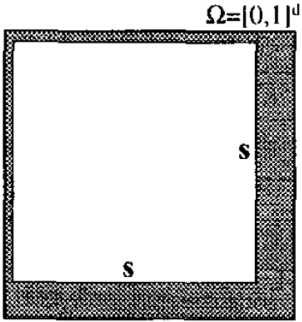
where $p_1 > p_2$ and $r_1 < r_2$.

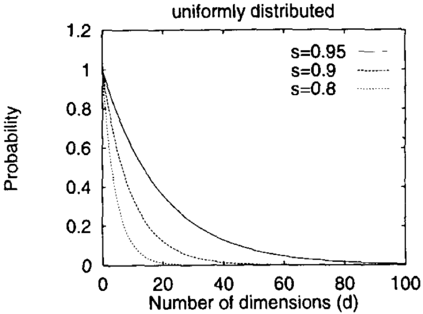
For d dimensional hamming space $\Omega = \{0, 1\}^d$ and hamming distance $D(p, q) = \sum_{i=1}^{d} \mathbb{1}(p_i \ne q_i)$, $\mathcal{H} = \{h_i : h_I(p) := p_i, i = 1, \dots, d\}$ is $(r, r(1+\varepsilon), 1 - \frac{r}{d}, 1 - \frac{r(1+\varepsilon)}{d})$-sensitive.

Theorem

For any $\varepsilon > 1$, there exists an algorithm for ε-PLEB in $\{0, 1\}^d$ using $O(dn + n^{1+1/\varepsilon})$ space and $O(dn^{1/\varepsilon})$ query time.

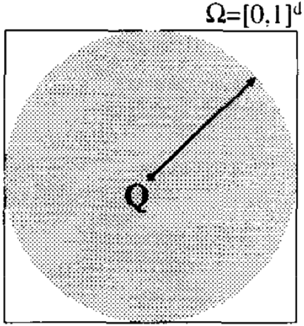
hyper-cube range query





Random sample a point from a hypercube $[0,1]^d$. Then the probability of the point lies within the lenght s hypercube.

hyper-sphere range query

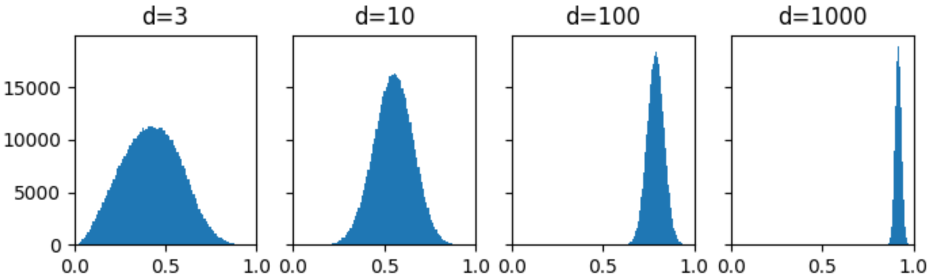


d	$P[R \in sp^d(Q, 0.5)]$
2	0.785
4	0.308
10	0.002
20	$2.461 \cdot 10^{-8}$
40	$3.278 \cdot 10^{-21}$
100	$1.868 \cdot 10^{-70}$

Random sample a point from a hypercube $[0, 1]^d$. Then the probability of the point lies within the largest ball that fits entirely within the hypercube.

Euclidean distance in high dimensional space

In the high dimensional space, the distance between two points is not a good indicator of their similarity.



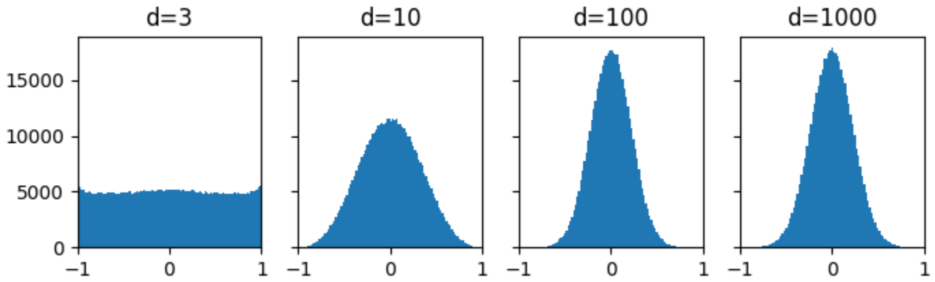
Randomly sample 1000 points from a uniform distribution in $[-1, 1]^d$. Then plot the histogram of the pairwise distances normalized by the max distance.

When $d = 1000$, $2/24$ error will include more than 50% of the points; $3/24$ error will include almost all the points.

Embedded data

- In the AI era, everything is a vector. And the dimension is quite large.
  - Google's Universal Sentence Encoder embeds sentences into a space of dimension 512.
  - GPT-3 embeds documents into space of dimension from 1024 to 12288.
- These embeds are not fixed, but learned by minimizing some loss function.
- The loss function can be defined in terms of the distances, or the similarities.

Cosine similarity in high dimensional space



Randomly sample 1000 points from a uniform distribution in $[-1, 1]^d$. Then plot the histogram of the pairwise consine similarities normalized by the max similarity.

High dimensional MIPS

There are two main tasks required to develop an efficient MIPS system.

- To reduce the number of candidates. e.g. space/data partitioning.
    - Tree search methods [Muja-Lowe'14; Dasgupta-Freund'08]
    - Locality sensitive hashing [Shrivastava-Li'14; Neyshabur-Srebro'15; Indyk-Motwani,'98; Andoni-Indyk-Laarhoven-Razenshteyn-Schmidt'15]
    - Graph search [Malkov-Yashunin'16; Harwood-Drummond'16]
- To speed up evaluation.
    - Quantization.
    - Random projection (dimension reduction) [Ailon-Liberty'13]

Conclusion

- A NNS system preprocesses the dataset for fast query.
- The objective can be minimizing a distance function or maximizing a similarity function.
- There are many methods exist for low dimensional data.
- In high dimensional space, the euclidean distance is not a good measurement of similarity, people usually use cosine similarity, e.g. MIPS.
- In addition to space(data) partitioning, speeding up evaluation is also important in high dimensional space.