Recent Developments in Algorithm Design: Graph-Based Nearest Neighbor Search

Prof. Christopher Musco, New York University

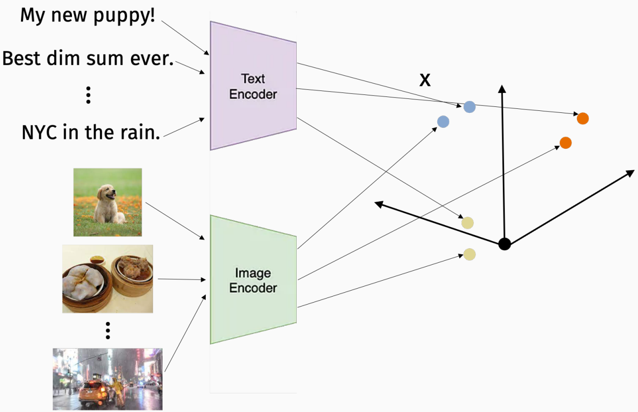
Characteristics of recent AI systems: Used at internet scale, demand real-time performance, significant test-time compute.

Algorithms for machine learning have gotten a lot more interesting in the past 3 years! Focus is no longer just on efficient training.

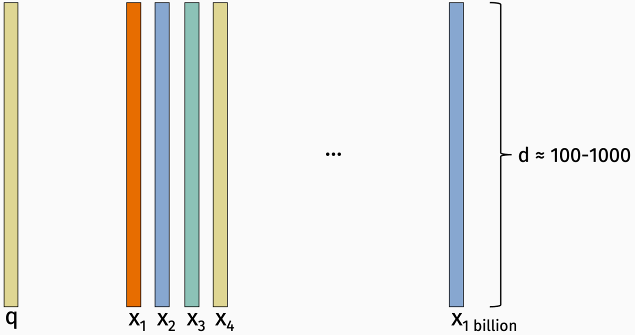
Goal for next three lectures: Three vignettes on recent algorithms relevant in modern machine learning.

- **High-Dimensional Vector Search.**
- Fast Autoregressive Language Generation.
- Sampling from high-dimensional distributions given an oracle (for image generation, Bayesian inference, private learning, and more)

Focus on recent. In many cases, methods in use are poorly understood and theory is in its very early stages.



Use neural network (BERT, CLIP, etc.) to convert documents, images, etc. to high dimensional vectors. Matching results should have similar vector embeddings.



Finding results for a query reduces to finding the nearest vector in a vector database $\mathcal{X}$, with similarity typically measured by Euclidean distance. I.e., return:

$$\arg\min_{x \in \mathcal{X}} \|x - q\|_2.$$

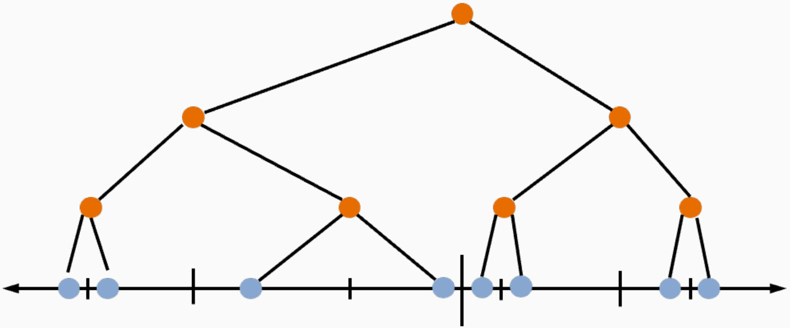
Vector search has been studied for a long time, but it is now used far more pervasively than even a few years ago:

- **Web-scale image search** and even text document search.
- **Retrieval Augmented Generation** for language models and AI autocomplete.
- **Multi-media search** on Amazon, Wayfair, etc.

Weaviate qdrant Pinecone zilliz

Goal: Let $\mathcal{X}$ be a database of n vectors in $\mathbb{R}^d$. Find $\mathbf{x} \in \mathcal{X}$ minimzing $\|\mathbf{x} - \mathbf{q}\|_2$ for a query $\mathbf{q}$.

- **Naive linear scan:** $O(nd)$ time.
- **kd trees:** $O(d \log(n) \cdot 2^d)$ time.



When d is large, we now have lots of other options available:

- Locality-sensitive hashing [Indyk, Motwani, 1998]
- Spectral hashing [Weiss, Torralba, and Fergus, 2008]
- Vector quantization/IVF data structures [Jégou, Douze, Schmid, 2009]
- Graph-based vector search [Malkov, Yashunin, 2016, Subramanya et al., 2019]

Key ideas behind all of these methods:

1. Allow for approximation.
2. Trade worse space-complexity + preprocessing time for better time-complexity. I.e., preprocess database in data structure that uses $\Omega(n)$ space.

When d is large, we now have lots of other options available:

- Locality-sensitive hashing [Indyk, Motwani, 1998]
- Spectral hashing [Weiss, Torralba, and Fergus, 2008]
- Vector quantization/IVF data structures [Jégou, Douze, Schmid, 2009]
- Graph-based vector search [Malkov, Yashunin, 2016, Subramanya et al., 2019]

Key ideas behind all of these methods:

1. Allow for approximation.
2. Trade worse space-complexity + preprocessing time for better time-complexity. I.e., preprocess database in data structure that uses $\Omega(n)$ space.

EXAMPLE WORST-CASE GUARANTEE

Theorem (Andoni, Indyk, FOCS 2006)

For any approximation factor $c \geq 1$, there is a data structure based on ***locality sensitive hashing*** *that, for any query* $\mathbf{q}$*, returns* $\tilde{\mathbf{x}}$ *satisfying:*

$$\|\tilde{\mathbf{x}} - \mathbf{q}\|_2 \leq c \cdot \min_{\mathbf{x} \in \mathcal{X}} \|\mathbf{x} - \mathbf{q}\|_2$$

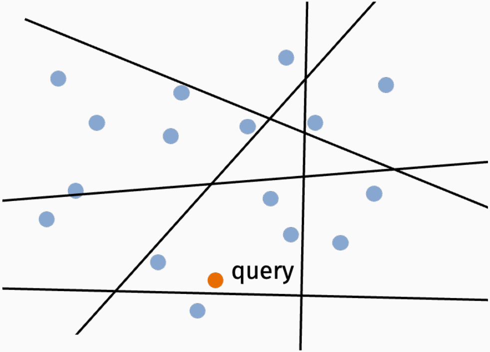
and uses:

- *Time:* $\tilde{O}\left(dn^{1/c^2}\right)$.
- *Space:* $\tilde{O}\left(nd + n^{1+1/c^2}\right)$.

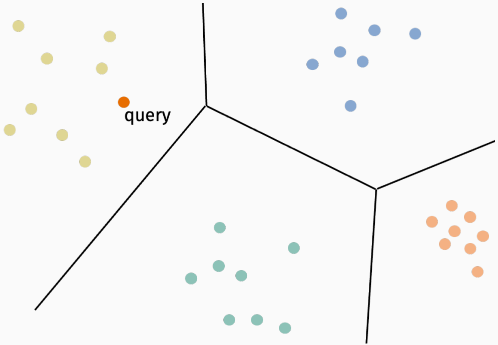
$\tilde{O}(\cdot)$ hides $\log(\Delta)$ factor where $\Delta = \frac{\max_{\mathbf{x},\mathbf{y} \in \mathcal{X}} \|\mathbf{x}-\mathbf{y}\|_2}{\min_{\mathbf{x},\mathbf{y} \in \mathcal{X}} \|\mathbf{x}-\mathbf{y}\|_2}$ is the dynamic range of our dataset.

Rough idea behind LSH:

1. Pick a bunch of random hyperplanes.
2. Check which side of each hyperplane q lies on.
3. Return closest point that lies in the same region as q.
4. Repeat multiple times to avoid missing anything.



In practice, we can often get partitions with better margin by partitioning in a data-dependent way, e.g. via clustering.



Main idea behind the improvements I listed earlier. Used in state-of-thea-art near-neighbor search libraries like Meta's FAISS and Google's SCANN.

NEAREST-NEIGHBOR SEARCH IN PRACTICE

New(ish) kid on the block: Graph-based near-neighbor search.

- **Navigating Spreading-out Graphs (NSG)** [Fu, Xiang, Wang, Cai, 2017]
- **Hierarchical Navigable Small World (HNSW)** [Malkov, Yashunin, 2016]
- **Microsoft DiskANN** [Subramanya, Devvrit, Kadekodi, Krishaswamy, Simhadri 2019]

Inspired by Milgram's famous "small world" experiments from the 1960s and later work on the small world phenomenon by Watts, Strogatz, Bobby Kleinberg, and others.

Similar methods proposed for low-dimensions in 1990s by Arya, Mount, Kleinberg and others.

1. Construct a directed search graph over our dataset.

2. Run greedy search in the graph.

Let $G = (V, E)$ be our graph where each node $1, \ldots, n$ is associated with a vector $\mathbf{x}_i \in \mathbb{R}^d$. Consider a query $\mathbf{q} \in \mathbb{R}^d$.

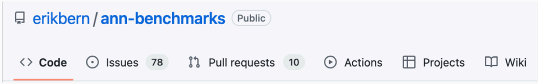
Let $\mathcal{N}(i) = \{j : (i, j) \in E\}$ be the out-neighborhood of i.

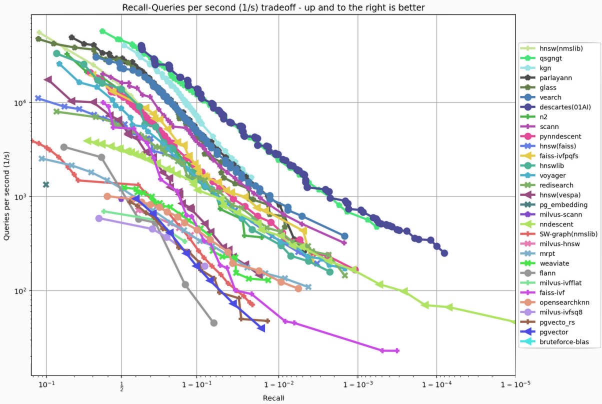
Greedy Search:

- Choose arbitrary starting node $\mathbf{s}$.
- Loop until termination:
  - Let $\mathbf{c} = \arg\min_{\mathbf{y} \in \mathcal{N}(\mathbf{s})} \|\mathbf{y} - \mathbf{q}\|_2$.
  - If $\|\mathbf{c} - \mathbf{q}\|_2 < \|\mathbf{s} - \mathbf{q}\|_2$, set $\mathbf{s} \leftarrow \mathbf{c}$.
  - Else, terminate loop and return $\mathbf{s}$.

Stanley Milgram

Winning all of the competitions!





Winning all of the competitions!

Results of the NeurIPS'21 Challenge on Billion-Scale Approximate Nearest Neighbor Search

Harsha Vardhan Simhadri[1] HARSHASI@MICROSOFT.COM
George Williams[2] GWILLIAMS@IEEE.ORG
Martin Aumüller[3] MAAU@ITU.DK
Matthijs Douze[4] MATTHIJS@FB.COM
Artem Babenko[5] ARTEM.BABENKO@PHYSTECH.EDU
Dmitry Baranchuk[5] DBARANCHUK@YANDEX-TEAM.RU
Qi Chen[1] CHEQI@MICROSOFT.COM
Lucas Hosseini[4] LUCAS.HOSSEINI@GMAIL.COM
Ravishankar Krishnaswamy[1] RAKRI@MICROSOFT.COM
Gopal Srinivasa[1] GOPALSR@MICROSOFT.COM
Suhas Jayaram Subramanya[6] SUHASJ@CS.CMU.EDU
Jingdong Wang[7] WANGJINGDONG@BAIDU.COM

[1] Microsoft Research [2] GSI Technology [3] IT University of Copenhagen
[4] Meta AI Research [5] Yandex [6] Carnegie Mellon University [7] Baidu

Results of the Big ANN: NeurIPS'23 competition

Harsha Vardhan Simhadri, Microsoft, harshasi@microsoft.com
Martin Aumüller, IT University of Copenhagen, maau@itu.dk
Amir Ingber, Pinecone, ingber@pinecone.io
Matthijs Douze, Meta AI Research, matthijs@meta.com
George Williams
Magdalen Dobson Manohar, Carnegie Mellon University
Dmitry Baranchuk, Yandex
Edo Liberty, Pinecone
Frank Liu, Zilliz
Ben Landrum, University of Maryland
Mazin Karjikar, University of Maryland
Laxman Dhulipala, University of Maryland
Meng Chen, Yue Chen, Rui Ma, Kai Zhang, Yuzheng Cai, Jiayang Shi, Yizhuo Chen, Weiguo Zheng, Fudan University
Zihao Wang, Shanghai Jiao Tong University
Jie Yin, Baidu
Ben Huang, Baidu

Open theory challenge: Can we explain the empirical success of graph-based nearest-neighbor search methods?

1. **Formalize desirable properties for a nearest-neighbor search graph.** Discuss some of my recent work with Torsten Suel, Haya Diwan, Jerry Gou, and Cameron Musco (NeurIPS 2024) on constructing graphs with these properties.
2. **Dive into a recent result of Indyk and Xu (NeurIPS 2023) on worst-case theoretical guarantees for graph-based search.** Currently, require strong (?) assumptions on the dataset $\mathcal{X}$ (low intrinsic dimension).

c-approximate nearest neighbor search: Return $\tilde{\mathbf{x}}$ satisfying $\|\tilde{\mathbf{x}} - \mathbf{q}\|_2 \leq c \cdot \min_{i \in \{1,\ldots,n\}} \|\mathbf{x}_i - \mathbf{q}\|_2$ for some $c \geq 1$.

Standard and reasonable guarantee for LSH methods.
Although people care about other metrics too.

Observation: Assuming there are no duplicates in $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, if query $\mathbf{q} = \mathbf{x}_i$ for some i, we must return $\mathbf{x}_i$.

Search graph G should be chosen to at least ensure that we find q if it is in the dataset.

Ideally, G should also be sparse and require few steps to find q (i.e, the graph should be “small-world”).

Definition (Navigable Graph)

A directed graph G for a point set $\mathbf{x}_1, \ldots, \mathbf{x}_n$ is navigable if, for all $i, j \in \{1, \ldots, n\}$, greedy search run on G with start node $\mathbf{x}_i$ and query $\mathbf{x}_j$ returns $\mathbf{x}_j$.

Listed as a desirable property in many empirical papers, including work on Navigable Spreading-our Graphs and Hierarchical Navigable Small World Graphs.

But none of this work produces provably navigable graphs.

Question: What is the sparsest navigable graph that can be constructed for a dataset $\mathbf{x}_1, \ldots, \mathbf{x}_n$?

Known results when $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are in low-dimensional Euclidean space:

- **2-dimensions:** The Delaunay graph can be proven to be navigable. This graph has average degree $O(n)$.

- **d-dimensions:** The Sparse Neighborhod Graph of Arya and Mount [SODA, 1993] is navigable and has average degree $O(2^d)$.

Claim (Upper Bound, DGMMS, 2024)

For any dataset $\mathbf{x}_1, \ldots, \mathbf{x}_n$, *it is possible to construct in* $O(n^2 \log n)$ *time a navigable graph* G *with average out-degree* $O(\sqrt{n \log n})$. *In fact, holds for any distance function.*

We will prove this under the mild assumption that, for all i, j, k, $\|\mathbf{x}_i - \mathbf{x}_j\|_2 \neq \|\mathbf{x}_i - \mathbf{x}_k\|_2$. Eliminates tedious corner cases related to tie-breaking. Can be ensured by adding arbitrarily small random perturbation to every data point.

Claim (Nearly Matching Lower Bound)

Let $\mathbf{x}_1, \ldots, \mathbf{x}_n$ *be random vectors in* $\{-1, 1\}^m$ *where* $m = O(\log n)$. *With high probability, any navigable graph for* $\mathbf{x}_1, \ldots, \mathbf{x}_n$ *requires average out-degree* $\Omega(n^{1/2-\epsilon})$ *for any fixed constant* ϵ.

Definition (Equivalent Navigabability Definition)

A directed graph G for a point set $\mathbf{x}_1, \ldots, \mathbf{x}_n$ is navigable if, for all nodes i, for all $j \neq i$, there is some $k \in \mathcal{N}(i)$ satisfying:

$$\|\mathbf{x}_j - \mathbf{x}_k\|_2 < \|\mathbf{x}_j - \mathbf{x}_i\|_2.$$

The above property is purely local! We can construct a navigable graph by separately checking the out-neighborhood of each node.

Can view graph construction as n seperate instances of set cover. For instance i, our elements to cover are $\{1, \ldots, n\} \setminus \{i\}$. We have a set S_k for all $k \neq i$.

$$S_k =$$

Definition (Equivalent Navigabability Definition)

A directed graph G for a point set $\mathbf{x}_1, \ldots, \mathbf{x}_n$ is navigable if, for all nodes i, for all $j \neq i$, there is some $k \in \mathcal{N}(i)$ satisfying:

$$\|\mathbf{x}_j - \mathbf{x}_k\|_2 < \|\mathbf{x}_j - \mathbf{x}_i\|_2.$$

Unfortunately, we can come up with point sets where any particular $\mathbf{x}_i$ necessarily has high-degree:

Approach: Consider all set cover instances in aggregate.

Distance-Based Permutation Matrix:

Node 1	x_1 x_{10} x_2 x_3 x_5 x_9 x_6 x_8 x_4 x_7
Node 2	x_2 x_4 x_6 x_1 x_{10} x_5 x_3 x_9 x_7 x_8
	⋮
Node n	x_{10} x_9 x_1 x_5 x_7 x_2 x_4 x_8 x_3 x_6

Requirement: Need at least one "left pointing" edge from every node in every list.

Construction: Choose $m < n$.

1. For all i, add an edge from j to i if j is one of i's m closest neighbors.
2. Add $3\frac{n}{m}\log n$ uniformly random out-edges from every node.

Node 1	$\mathbf{x}_1$ $\mathbf{x}_{10}$ $\mathbf{x}_2$ $\mathbf{x}_3$ $\mathbf{x}_5$ $\mathbf{x}_9$ $\mathbf{x}_6$ $\mathbf{x}_8$ $\mathbf{x}_4$ $\mathbf{x}_7$
Node 2	$\mathbf{x}_2$ $\mathbf{x}_4$ $\mathbf{x}_6$ $\mathbf{x}_1$ $\mathbf{x}_{10}$ $\mathbf{x}_5$ $\mathbf{x}_3$ $\mathbf{x}_9$ $\mathbf{x}_7$ $\mathbf{x}_8$
	$\vdots$
Node n	$\mathbf{x}_{10}$ $\mathbf{x}_9$ $\mathbf{x}_1$ $\mathbf{x}_5$ $\mathbf{x}_7$ $\mathbf{x}_2$ $\mathbf{x}_4$ $\mathbf{x}_8$ $\mathbf{x}_3$ $\mathbf{x}_6$

Fix a node i.

Claim 1: Suppose $\mathbf{x}_j$ is one of $\mathbf{x}_i$'s m closest neighbors. Then $\mathbf{x}_j$ has an out-edge to some $\mathbf{x}_k$ with $\|\mathbf{x}_k - \mathbf{x}_i\|_2 < \|\mathbf{x}_j - \mathbf{x}_i\|_2$.

Claim 2: Suppose $\mathbf{x}_j$ is <u>not</u> one of $\mathbf{x}_i$'s m closest neighbors. Then, with probability $\geq 1 - \frac{1}{n^3}$, $\mathbf{x}_j$ has an out-edge to some $\mathbf{x}_k$ with $\|\mathbf{x}_k - \mathbf{x}_i\|_2 < \|\mathbf{x}_j - \mathbf{x}_i\|_2$.

Claim 2: Suppose $\mathbf{x}_j$ is <u>not</u> one of $\mathbf{x}_i$'s m closest neighbors. Then, with probability $\geq 1 - \frac{1}{n^3}$, $\mathbf{x}_j$ has an out-edge to some $\mathbf{x}_k$ with $\|\mathbf{x}_k - \mathbf{x}_i\|_2 < \|\mathbf{x}_j - \mathbf{x}_i\|_2$.

Node 1	$\mathbf{x}_1$	$\mathbf{x}_{10}$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_5$	$\mathbf{x}_9$	$\mathbf{x}_6$	$\mathbf{x}_8$	$\mathbf{x}_4$	$\mathbf{x}_7$
Node 2	$\mathbf{x}_2$	$\mathbf{x}_4$	$\mathbf{x}_6$	$\mathbf{x}_1$	$\mathbf{x}_{10}$	$\mathbf{x}_5$	$\mathbf{x}_3$	$\mathbf{x}_9$	$\mathbf{x}_7$	$\mathbf{x}_8$
					⋮					
Node n	$\mathbf{x}_{10}$	$\mathbf{x}_9$	$\mathbf{x}_1$	$\mathbf{x}_5$	$\mathbf{x}_7$	$\mathbf{x}_2$	$\mathbf{x}_4$	$\mathbf{x}_8$	$\mathbf{x}_3$	$\mathbf{x}_6$

Node 1	$\mathbf{x}_1$ $\mathbf{x}_{10}$ $\mathbf{x}_2$ $\mathbf{x}_3$ $\mathbf{x}_5$ $\mathbf{x}_9$ $\mathbf{x}_6$ $\mathbf{x}_8$ $\mathbf{x}_4$ $\mathbf{x}_7$
Node 2	$\mathbf{x}_2$ $\mathbf{x}_4$ $\mathbf{x}_6$ $\mathbf{x}_1$ $\mathbf{x}_{10}$ $\mathbf{x}_5$ $\mathbf{x}_3$ $\mathbf{x}_9$ $\mathbf{x}_7$ $\mathbf{x}_8$
	⋮
Node n	$\mathbf{x}_{10}$ $\mathbf{x}_9$ $\mathbf{x}_1$ $\mathbf{x}_5$ $\mathbf{x}_7$ $\mathbf{x}_2$ $\mathbf{x}_4$ $\mathbf{x}_8$ $\mathbf{x}_3$ $\mathbf{x}_6$

By a union bound, we have a left-pointing edge for every node in every permutation with probability $\geq 1 - \frac{1}{n}$, so our graph is navigable.

Total degree of constructed graph:

Claim (Upper Bound)

For any dataset $\mathbf{x}_1, \ldots, \mathbf{x}_n$, it is possible to construct in $O(n^2 \log n)$ time a navigable graph G with average out-degree $O(\sqrt{n \log n})$. In fact, holds for any distance function.

Oberservation: The graph we constructed is “small-world”. Only two hops required for any starting node and query.

Claim (Nearly Matching Lower Bound)

Let $\mathrm{x}_1, \ldots, \mathrm{x}_n$ *be random vectors in* $\{-1, 1\}^m$ *where* $m = O(\log n)$. *With high probability, any navigable graph for* $\mathrm{x}_1, \ldots, \mathrm{x}_n$ *requires average out-degree* $\Omega(n^{1/2-\epsilon})$ *for any fixed constant* ϵ.

"hard" region

Node 1	$\mathbf{x}_1$	$\mathbf{x}_{10}$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_5$	$\mathbf{x}_9$	$\mathbf{x}_6$	$\mathbf{x}_8$	$\mathbf{x}_4$	$\mathbf{x}_7$
Node 2	$\mathbf{x}_2$	$\mathbf{x}_4$	$\mathbf{x}_6$	$\mathbf{x}_1$	$\mathbf{x}_{10}$	$\mathbf{x}_5$	$\mathbf{x}_3$	$\mathbf{x}_9$	$\mathbf{x}_7$	$\mathbf{x}_8$
					⋮					
Node n	$\mathbf{x}_{10}$	$\mathbf{x}_9$	$\mathbf{x}_1$	$\mathbf{x}_3$	$\mathbf{x}_7$	$\mathbf{x}_2$	$\mathbf{x}_4$	$\mathbf{x}_8$	$\mathbf{x}_5$	$\mathbf{x}_6$

Observation: Hard region involves $n^{3/2}$ edge constraints.

For sake of proof sketch, assume permutations are uniformly random. In the paper, we show that this is “close” to true for random data points in $O(\log n)$ dimensions.

Claim: Adding any edge (i, j) to the graph only covers at most $O(\log(n))$ of the $n^{3/2}$ hard constraints with high probability.

Claim: Under random permutations, adding any (i, j) to G only covers at most $O(\log(n))$ hard constraints with high prob.

"hard" region

Node 1	x_1	x_{10}	x_2	x_3	x_5	x_9	x_6	x_8	x_4	x_7
Node 2	x_2	x_4	x_6	x_1	x_{10}	x_5	x_3	x_9	x_7	x_8
					⋮					
Node n	x_{10}	x_9	x_1	x_3	x_7	x_2	x_4	x_8	x_5	x_6

Completing the argument:

Claim (Nearly Matching Lower Bound)

Let $\mathbf{x}_1, \ldots, \mathbf{x}_n$ *be random vectors in* $\{-1, 1\}^m$ *where* $m = O(\log n)$*. With high probability, any navigable graph for* $\mathbf{x}_1, \ldots, \mathbf{x}_n$ *has average out-degree* $\Omega(n^{1/2-\epsilon})$ *for any fixed constant* ϵ*.*

Positives:

- For queries $\mathbf{q} \in \mathcal{X}$, greedy search + navigable graph returns exact result.
- Data structure takes $O(n^{1.5})$ space.
- Runtime for $\mathbf{q} \in \mathcal{X}$ should be roughly $O(\sqrt{n})$ given small-world property, but difficult to say anything formally.

Negatives:

- $\sqrt{n}$ degree is still pretty dense. In practice, graphs can be pruned and yield good empirical results.
- No approx. guarantees for queries not in the data set $\mathcal{X}$.
- No formal runtime guarantees.

NeurIPS 2023 paper: *"Worst-case Performance of Popular Approximate Nearest Neighbor Search Implementations: Guarantees and Limitations"* by Piotr Indyk and Haike Xu.

Addresses these issues, albeit under additional assumptions about the dataset $\mathcal{X}$.

Two components of result:

1. If G is α-shortcut reachable then, for any query $\mathbf{q}$, greedy search converges to an $\left(\frac{\alpha+1}{\alpha-1}+\epsilon\right)$-approximate nearest neighbor in $\sim \log(1/\epsilon)$ steps.

2. Any dataset with doubling dimension d has an α-shortcut reachable graph with maximum degree $\tilde{O}\left((8\alpha)^d\right)$.

First introduced in the DiskANN paper out of Microsoft Research. Strictly strengthens navigability.

Definition (Navigabability, aka 1-shortcut reachability)

A directed graph G for a point set $\mathbf{x}_1, \ldots, \mathbf{x}_n$ is navigable if, for all nodes i, for all $j \neq i$, there is some $k \in \mathcal{N}(i)$ satisfying:

$$\|\mathbf{x}_j - \mathbf{x}_k\| < \|\mathbf{x}_j - \mathbf{x}_i\|.$$

Definition (α-shortcut reachability)

A directed graph G for a point set $\mathbf{x}_1, \ldots, \mathbf{x}_n$ is α-shortcut reachability for $\alpha \geq 1$ if, for all nodes i, for all $j \neq i$, there is some $k \in \mathcal{N}(i)$ satisfying:

$$\|\mathbf{x}_j - \mathbf{x}_k\| < \frac{1}{\alpha}\|\mathbf{x}_j - \mathbf{x}_i\|.$$

Theorem (ANN from Shortcut Reachability)

Let $c = \frac{\alpha+1}{\alpha-1}$. If greedy search is run on an α-shortcut reachable graph G with arbitrary start node and query $\mathbf{q}$, after $\log_\alpha(c\Delta/\epsilon)$ steps it returns a point $\tilde{\mathbf{x}}$ satisfying:

$$\|\tilde{\mathbf{x}} - \mathbf{q}\| \leq (c + \epsilon) \min_{j \in \{1,\ldots,n\}} \|\mathbf{x}_j - \mathbf{q}\|.$$

$\Delta = \frac{d_{\max}}{d_{\min}} = \frac{\max_{i,j} \|\mathbf{x}_i - \mathbf{x}_j\|}{\min_{i,j} \|\mathbf{x}_i - \mathbf{x}_j\|}$ is the dynamic range of our dataset.

Intuitive why larger α leads to faster convergence. Less clear why it leads to a better approximate nearest neighbor.

Why does navigability fail to return provable approximate nearest neigbhors for queries outside the data set?

Why would α-shortcut reachability fix this hard case?

Let $\mathbf{v}_0, \mathbf{v}_1, \ldots$ be the iterates of greedy search run on a graph G. So $\mathbf{v}_i$ is an out-neighbor of $\mathbf{v}_{i-1}$ and $\|\mathbf{q} - \mathbf{v}_0\| > \|\mathbf{q} - \mathbf{v}_1\| > \|\mathbf{q} - \mathbf{v}_2\| > \ldots$.

Claim (Almost Monotonic Convergence of Greedy Search)

If G is α-shortcut reachable then:

$$\|\mathbf{v}_i - \mathbf{q}\| \leq \frac{1}{\alpha}\|\mathbf{v}_{i-1} - \mathbf{q}\| + (1 + \frac{1}{\alpha})\|\mathbf{x}^* - \mathbf{q}\|.$$

Let $\mathbf{v}_0, \mathbf{v}_1, \ldots$ be the iterates of greedy search run on a graph G. So $\mathbf{v}_i$ is an out-neighbor of $\mathbf{v}_{i-1}$ and $\|\mathbf{q} - \mathbf{v}_0\| > \|\mathbf{q} - \mathbf{v}_1\| > \|\mathbf{q} - \mathbf{v}_2\| > \ldots$.

Claim (Almost Monotonic Convergence of Greedy Search)

If G is α-shortcut reachable then:

$$\|\mathbf{v}_i - \mathbf{q}\| \leq \frac{1}{\alpha}\|\mathbf{v}_{i-1} - \mathbf{q}\| + (1 + \frac{1}{\alpha})\|\mathbf{x}^* - \mathbf{q}\|.$$

Proof:

Claim (Almost Monotonic Convergence of Greedy Search)

If G is α-shortcut reachable then:

$$\|\mathbf{v}_i - \mathbf{q}\| \leq \frac{1}{\alpha}\|\mathbf{v}_{i-1} - \mathbf{q}\| + (1 + \frac{1}{\alpha})\|\mathbf{x}^* - \mathbf{q}\|.$$

Consequence 1: Greedy search eventually converges to some $\tilde{\mathbf{x}}$ with:

$$\|\tilde{\mathbf{x}} - \mathbf{q}\| \leq \frac{\alpha + 1}{\alpha - 1} \cdot \|\mathbf{x}^* - \mathbf{q}\|.$$

Proof:

Claim (Almost Monotonic Convergence of Greedy Search)

If G is α-shortcut reachable then:

$$\|\mathbf{v}_i - \mathbf{q}\| \leq \frac{1}{\alpha}\|\mathbf{v}_{i-1} - \mathbf{q}\| + (1 + \frac{1}{\alpha})\|\mathbf{x}^* - \mathbf{q}\|.$$

Consequence 2: For all $i \geq 1$,

$$\|\mathbf{v}_i - \mathbf{q}\| \leq \frac{\|\mathbf{v}_0 - \mathbf{q}\|}{\alpha^i} + \frac{\alpha + 1}{\alpha - 1} \cdot \|\mathbf{x}^* - \mathbf{q}\|.$$

Theorem (ANN from Shortcut Reachability (Indyk, Xu))

Let $c = \frac{\alpha+1}{\alpha-1}$. If greedy search is run on an α-shortcut reachable graph G with arbitrary start node and query $\mathbf{q}$, *after* $O(\log_\alpha(c\Delta/\epsilon))$ *steps it returns a point* $\tilde{\mathbf{x}}$ *satisfying* $\|\tilde{\mathbf{x}} - \mathbf{q}\| \leq (c+\epsilon)\min_j \|\mathbf{x}_j - \mathbf{q}\|$.

Key Lemma: For all $i \geq 1$, $\|\mathbf{v}_i - \mathbf{q}\| \leq \frac{\|\mathbf{v}_0 - \mathbf{q}\|}{\alpha^i} + \frac{\alpha+1}{\alpha-1} \cdot \|\mathbf{x}^* - \mathbf{q}\|$.

Case 1: $\|\mathbf{v}_0 - \mathbf{q}\| \geq \frac{\alpha+1}{2} d_{\max}$.

Theorem (ANN from Shortcut Reachability (Indyk, Xu))

Let $c = \frac{\alpha+1}{\alpha-1}$. If greedy search is run on an α-shortcut reachable graph G with arbitrary start node and query $\mathbf{q}$, after $O(\log_\alpha(c\Delta/\epsilon))$ steps it returns a point $\tilde{\mathbf{x}}$ satisfying $\|\tilde{\mathbf{x}} - \mathbf{q}\| \leq (c+\epsilon)\min_j \|\mathbf{x}_j - \mathbf{q}\|$.

Key Lemma: For all $i \geq 1$, $\|\mathbf{v}_i - \mathbf{q}\| \leq \frac{\|\mathbf{v}_0 - \mathbf{q}\|}{\alpha^i} + \frac{\alpha+1}{\alpha-1} \cdot \|\mathbf{x}^* - \mathbf{q}\|$.

Case 2: $\|\mathbf{v}_0 - \mathbf{q}\| \leq \frac{\alpha+1}{2} d_{\max}$ and $\|\mathbf{x}^* - \mathbf{q}\| \leq \frac{\alpha-1}{4(\alpha+1)} d_{\min}$.

Theorem (ANN from Shortcut Reachability (Indyk, Xu))

Let $c = \frac{\alpha+1}{\alpha-1}$. If greedy search is run on an α-shortcut reachable graph G with arbitrary start node and query $\mathbf{q}$, after $O(\log_\alpha(c\Delta/\epsilon))$ steps it returns a point $\tilde{\mathbf{x}}$ satisfying $\|\tilde{\mathbf{x}} - \mathbf{q}\| \leq (c+\epsilon)\min_j \|\mathbf{x}_j - \mathbf{q}\|$.

Key Lemma: For all $i \geq 1$, $\|\mathbf{v}_i - \mathbf{q}\| \leq \frac{\|\mathbf{v}_0 - \mathbf{q}\|}{\alpha^i} + \frac{\alpha+1}{\alpha-1} \cdot \|\mathbf{x}^* - \mathbf{q}\|$.

Case 3: $\|\mathbf{v}_0 - \mathbf{q}\| \leq \frac{\alpha+1}{2} d_{\max}$ and $\|\mathbf{x}^* - \mathbf{q}\| \geq \frac{\alpha-1}{4(\alpha+1)} d_{\min}$.

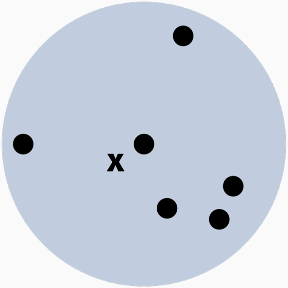
In contrast to navigability, it is possible to come up with datasets where any α-shortcut reachable graph (for any $\alpha > 1$) must have $\Omega(n^2)$ edges:

Fortunately, Indyk and Xu show that this not possible if the doubling dimension of our dataset is low. Doubling dimension is a natural measure of "intrinsic dimension" that has been considered in prior work on NN-search (e.g. [Beygelzimer, Kakade, Langford, ICML 2006]).

For a point $\mathbf{x}$, let $\mathcal{B}(\mathbf{x}, r)$ be a ball of radius r centered around $\mathbf{x}$.

Definition (Doubling Dimension)

The doubling constant of a point set $\mathcal{X}$ is the smallest C such that, for any r and any $\mathbf{x} \in \mathcal{X}$, $\mathcal{B}(\mathbf{x}, r) \cap \mathcal{X}$ can be covered with C balls of radius $r/2$. The doubling dimension, d', of $\mathcal{X}$ equals $d' = \log_2(C)$.

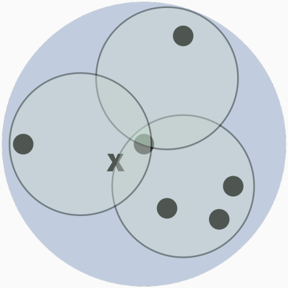


We always have that $d' \leq d$ if $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$, and often (?) $d' \ll d$.

For a point $\mathbf{x}$, let $\mathcal{B}(\mathbf{x}, r)$ be a ball of radius r centered around $\mathbf{x}$.

Definition (Doubling Dimension)

The doubling constant of a point set $\mathcal{X}$ is the smallest C such that, for any r and any $\mathbf{x} \in \mathcal{X}$, $\mathcal{B}(\mathbf{x}, r) \cap \mathcal{X}$ can be covered with C balls of radius $r/2$. The doubling dimension, d', of $\mathcal{X}$ equals $d' = \log_2(C)$.



We always have that $d' \leq d$ if $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$, and often (?) $d' \ll d$.

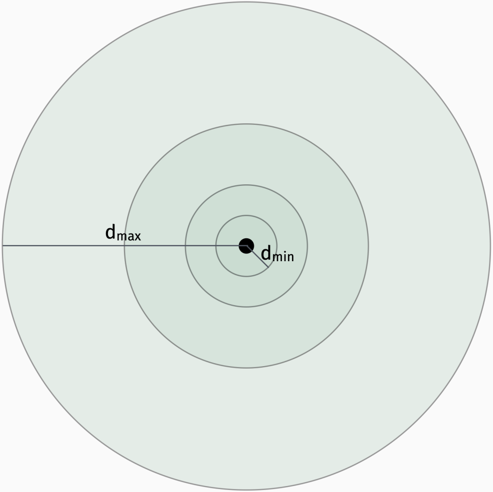
Theorem (Shortcut Reachability from Doubling Dim. (Indyk, Xu))

Any points set $\mathcal{X}$ with doubling dimension d' and dynamic range Δ has an α-shortcut reachable graph G with maximum degree:

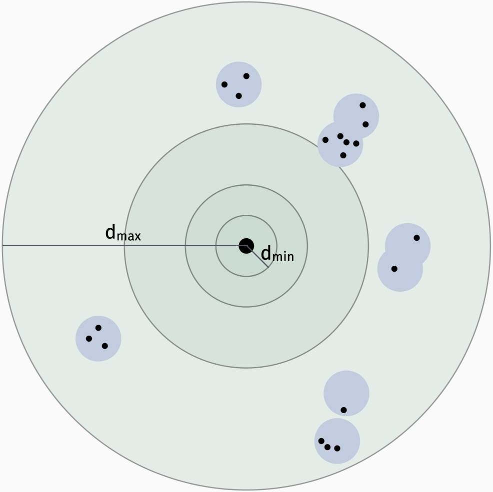
$$(8\alpha)^{d'} \log \Delta$$

Simple fact: If $\mathcal{X}$ has doubling dimension d', then for any $\mathbf{x} \in \mathcal{X}$ and any r, $\mathcal{B}(\mathbf{x}, r) \cap \mathcal{X}$ can be covered with $(2k)^{d'}$ balls of radius r/k.

Construction: Cover points in ring with outer radius r (inner radius $r/2$) with balls of radius $r/4\alpha$. Connect **x** to any point in each ball.



Construction: Cover points in ring with outer radius r (inner radius $r/2$) with balls of radius $r/4\alpha$. Connect $\mathbf{x}$ to any point in each ball.



Construction: Cover points in ring with outer radius r (inner radius $r/2$) with balls of radius $r/4\alpha$. Connect $\mathbf{x}$ to any point in each ball.

By previous fact, we need $(2 \cdot 4\alpha)^{d'}$ such balls to cover each ring. There are $\log_2 \Delta$ rings.

Theorem (Shortcut Reachability from Doubling Dim. (Indyk, Xu))

Any points set $\mathcal{X}$ with doubling dimension d' and dynamic range Δ has an α-shortcut reachable graph G with maximum degree:

$$(8\alpha)^{d'} \log \Delta$$

Two components of [Indyk, Xu, 2023] result: Let $c = \frac{\alpha+1}{\alpha-1}$.

1. If G is α-shortcut reachable then, for any query q, greedy search converges to an $\left(\frac{\alpha+1}{\alpha-1} + \epsilon\right)$-approximate nearest neighbor in $O(\log_\alpha(c\Delta/\epsilon))$ steps.

2. Any dataset with doubling dimension d' has an α-shortcut reachable graph with maximum degree $O\left((8\alpha)^{d'} \log \Delta\right)$.

Final space complexity:

Final runtime:

Positives:

- Theoretical tradeoff between time/space and accuracy.
- Covering-based graph can be constructed greedily in polynomial time. In fact, the algorithm was already proposed in DiskANN (NeurIPS, 2019).

Negatives:

- Not clear how small doubling dimension d' is in practice and it's difficult to verify / people haven't really tried thoroughly.
- No approx. guarantees for queries not in the data set $\mathcal{X}$.
- No formal runtime guarantees.