Rank Approximate Nearest Neighbor Search

Retaining Meaning and Speed in High Dimensions

Parikshit Ram, Dongryeol Lee, Hua Ouyang, Alexander G. Gray

School of Computational Science and Engineering

[College of Computing, Georgia Institute of Technology]
Overview

● Nearest Neighbor Search in High Dimensions
● Applications
● What's new and improved!!
● Outline

Introduction

Rank Approximation

Experiments and Results

Conclusion and Future
Nearest neighbor problem
Nearest Neighbor Search in High Dimensions

- Nearest neighbor problem
- Easy!
Nearest Neighbor Search in High Dimensions

- Nearest neighbor problem
- Easy!
- High dimensions
Nearest Neighbor Search in High Dimensions

- Nearest neighbor problem
- Easy!
- High dimensions
- Not so easy anymore
Applications
Applications

- **Manifold learning**

![Manifold learning diagram](image)
Applications

- Manifold learning
- k-Nearest Neighbor classification
Applications

- Manifold learning
- k-Nearest Neighbor classification
- Kernel Density Estimation
Applications

- Manifold learning
- k-Nearest Neighbor classification
- Kernel Density Estimation
- Computer vision
What’s new and improved!!
What’s new and improved!!

- Novel idea: Rank approximation of Nearest Neighbor (RANN) search.
What’s new and improved!!

- Novel idea: Rank approximation of Nearest Neighbor (RANN) search.
- RANN vs. Exact Search: much better speed for low approximations!
What’s new and improved!!

- Novel idea: Rank approximation of Nearest Neighbor (RANN) search.
- RANN vs. Exact Search: much better speed for low approximations!
- RANN vs. Locality Sensitive Hashing (LSH): Better accuracy for same speed.
Outline

Introduction
Outline

- Introduction
  - The Problem
Outline

- Introduction
  - The Problem
  - The curse of dimensionality
Outline

Introduction

- The Problem
- The curse of dimensionality
- Distance approximate formulation
Outline

Introduction

- The Problem
- The curse of dimensionality
- Distance approximate formulation
- Rank approximate formulation
# Outline

- **Introduction**
  - The Problem
  - The curse of dimensionality
  - Distance approximate formulation
  - Rank approximate formulation

- **Rank Approximation**
Outline

- Introduction
  - The Problem
  - The curse of dimensionality
  - Distance approximate formulation
  - Rank approximate formulation

- Rank Approximation
  - Order Statistics Magic
Outline

- Introduction
  - The Problem
  - The curse of dimensionality
  - Distance approximate formulation
  - Rank approximate formulation

- Rank Approximation
  - Order Statistics Magic
  - Rank approximate NN search
Overview

- Nearest Neighbor Search in High Dimensions
- Applications
- What's new and improved!!

Outline

- Introduction
  - The Problem
  - The curse of dimensionality
  - Distance approximate formulation
  - Rank approximate formulation

- Rank Approximation
  - Order Statistics Magic
  - Rank approximate NN search
  - The Algorithm
Outline

- Introduction
  - The Problem
  - The curse of dimensionality
  - Distance approximate formulation
  - Rank approximate formulation

- Rank Approximation
  - Order Statistics Magic
  - Rank approximate NN search
  - The Algorithm

- Experiments and Results
Outline

- Introduction
  - The Problem
  - The curse of dimensionality
  - Distance approximate formulation
  - Rank approximate formulation

- Rank Approximation
  - Order Statistics Magic
  - Rank approximate NN search
  - The Algorithm

- Experiments and Results
  - Comparison with Exact
Overview

Nearest Neighbor Search in High Dimensions
Applications
What's new and improved!!

Outline

Introduction
  - The Problem
  - The curse of dimensionality
  - Distance approximate formulation
  - Rank approximate formulation

Rank Approximation
  - Order Statistics Magic
  - Rank approximate NN search
  - The Algorithm

Experiments and Results
  - Comparison with Exact
  - Comparison with LSH
Outline

- Introduction
  - The Problem
  - The curse of dimensionality
  - Distance approximate formulation
  - Rank approximate formulation

- Rank Approximation
  - Order Statistics Magic
  - Rank approximate NN search
  - The Algorithm

- Experiments and Results
  - Comparison with Exact
  - Comparison with LSH

- Conclusion and Future Work
Outline

- Introduction
  - The Problem
  - The curse of dimensionality
  - Distance approximate formulation
  - Rank approximate formulation

- Rank Approximation
  - Order Statistics Magic
  - Rank approximate NN search
  - The Algorithm

- Experiments and Results
  - Comparison with Exact
  - Comparison with LSH

- Conclusion and Future Work
  - Conclusions
Introduction
The Problem

Dataset $S \subset X$, $|S| = N$ in $(X, d)$, $q \in X$, find $p \in S$ such that

$$p = \arg \min_{r \in S} d(r, q).$$
The Problem

Dataset $S \subset X$, $|S| = N$ in $(X, d)$, $q \in X$, find $p \in S$ such that

$$p = \arg \min_{r \in S} d(r, q).$$

- Linear search: $O(N)$
The Problem

Dataset $S \subset X$, $|S| = N$ in $(X, d)$, $q \in X$, find $p \in S$ such that

$$p = \arg\min_{r \in S} d(r, q).$$

- Linear search: $O(N)$
- Hashing: $O(1)$
The Problem

Dataset $S \subset X$, $|S| = N$ in $(X, d)$, $q \in X$, find $p \in S$ such that

$$p = \arg \min_{r \in S} d(r, q).$$

- Linear search: $O(N)$
- Hashing: $O(1)$
- Spatial partitioning: Using trees data structures
The Problem

Dataset $S \subset X$, $|S| = N$ in $(X, d)$, $q \in X$, find $p \in S$ such that

$$p = \arg \min_{r \in S} d(r, q).$$

- Linear search: $O(N)$
- Hashing: $O(1)$
- Spatial partitioning: Using trees data structures

Use hypercubes (kd-trees). Search in expected $O(\log N)$

(Friedman, Bentley, Finkel, 1977)
The Problem

Dataset \( S \subset X, \ |S| = N \) in \( (X, d) \), \( q \in X \), find \( p \in S \) such that

\[
p = \arg \min_{r \in S} d(r, q).
\]

- Linear search: \( O(N) \)
- Hashing: \( O(1) \)
- Spatial partitioning: Using trees data structures

Use balls instead (Ball trees)

(Omohundro,'89)
The Problem

Dataset \( S \subset X, |S| = N \) in \( (X, d) \), \( q \in X \), find \( p \in S \) such that

\[
p = \arg \min_{r \in S} d(r, q).
\]

- Linear search: \( O(N) \)
- Hashing: \( O(1) \)
- Spatial partitioning: Using trees data structures

(Beygelzimer, Kakade, Langford, '06)

Cover trees are cooler. They have \( O(\log N) \) bound for search!!
The Problem

Dataset $S \subset X$, $|S| = N$ in $(X, d)$, $q \in X$, find $p \in S$ such that

$$p = \arg \min_{r \in S} d(r, q).$$

- Linear search: $O(N)$
- Hashing: $O(1)$
- Spatial partitioning: Using trees data structures

What about $O(N)$ queries? It would be at best $O(N \log N)$
The Problem

Dataset \( S \subset X, \quad |S| = N \) in \((X, d), \quad q \in X\), find \( p \in S \) such that

\[
p = \arg \min_{r \in S} d(r, q).
\]

- Linear search: \( O(N) \)
- Hashing: \( O(1) \)
- Spatial partitioning: Using trees data structures

How about using 2 trees?

It's environment friendly too!

(Gray, Moore, '00)
The Problem

Dataset $S \subset X$, $|S| = N$ in $(X, d)$, $q \in X$, find $p \in S$ such that

$$p = \arg \min_{r \in S} d(r, q).$$

- Linear search: $O(N)$
- Hashing: $O(1)$
- Spatial partitioning: Using trees data structures

I bet they are $O(N)$ for the cool Cover trees

(Beygelzimer, Kakade, Langford, ’06)
The Curse of Dimensionality
The Curse of Dimensionality

(Hammersley,’50) Distribution of distances in a hypersphere of radius $a$ is given by

$$N(a\sqrt{2}, a^2/2dim)$$
(Hammersley,'50) Distribution of distances in a hypersphere of radius $a$ is given by

$$N(a\sqrt{2}, a^2/2\dim)$$
The Curse of Dimensionality

(Hammersley,'50) Distribution of distances in a hypersphere of radius $a$ is given by

$$N(a\sqrt{2}, a^2/2\text{dim})$$

(Cayton,'08)
(Hammersley,’50) Distribution of distances in a hypersphere of radius \( a \) is given by

\[ N(a\sqrt{2}, a^2/2\dim) \]

Sad news: Efficiency without approximation seems impossible!!

(Cayton,’08)
Approximation: Distance based
Approximation: Distance based

Find any point $p' \in S$ such that

$$d(p', q) \leq (1 + \epsilon) \min_{r \in S} d(r, q).$$
Approximation: Distance based

Find any point $p' \in S$ such that

$$d(p', q) \leq (1 + \epsilon) \min_{r \in S} d(r, q).$$

Use same trees but with `looser` pruning

(Beygelzimer, Kakade, Langford, ’06)
Approximation: Distance based

Find any point $p' \in S$ such that

$$d(p', q) \leq (1 + \epsilon) \min_{r \in S} d(r, q).$$

Why don't we have overlapping tree nodes to get "Spill" trees. This would avoid backtracking.

(Liu, Moore, Gray, '04)
Approximation: Distance based

Find any point $p' \in S$ such that

$$d(p', q) \leq (1 + \epsilon) \min_{r \in S} d(r, q).$$

Why don't we apply PAC learning paradigm to NN?

(Ciaccia, Patella, ’00)
Approximation: Distance based

Find any point $p' \in S$ such that

$$d(p', q) \leq (1 + \epsilon) \min_{r \in S} d(r, q).$$

Johnson-Lindenstrauss lemma states that random projection has little distortion of pairwise distances

(Johnson, Lindenstrauss, ’84)
Approximation: Distance based

Find any point $p' \in S$ such that

$$d(p', q) \leq (1 + \epsilon) \min_{r \in S} d(r, q).$$

(Liu, Moore, Gray, '04)

Let's use random projection for reducing dimension of the data set.

And make Hybrid Spill trees.
Approximation: Distance based

Find any point $p' \in S$ such that

$$d(p', q) \leq (1 + \epsilon) \min_{r \in S} d(r, q).$$

(Indyk, Motwani, '98)

<table>
<thead>
<tr>
<th>0011</th>
<th>0111</th>
<th>1111</th>
</tr>
</thead>
<tbody>
<tr>
<td>0010</td>
<td>0110</td>
<td>1110</td>
</tr>
<tr>
<td>0000</td>
<td>0100</td>
<td>1100</td>
</tr>
</tbody>
</table>

Why don't we make hash tables of the projected data?

Hash tables great for low dimensions
Need
Rank Approximation Formulation

Need

But I said distances lose meaning in High Dimensions

(Hammersley,’50)
Need

But I said distances lose meaning in high dimensions (Hammersley, ’50)

Ranks are still meaningful though!! (Cayton, ’08)
Need

But I said distances lose meaning in High Dimensions. (Hammersley,’50)

Ranks are still meaningful though!! (Cayton,’08)

Let $D = \{D_1, \ldots, D_N\}$, where $D_i = d(q, r_i)$, $r_i \in S$, $\forall i = 1, \ldots, N$, $D_{(r)}$ - $r^{th}$ order statistics. Then find a point $p' \in S$ such that
Need

But I said distances lose meaning in High Dimensions

(Rank Approximation Formulation)

Let $D = \{D_1, \ldots, D_N\}$, where $D_i = d(q, r_i), r_i \in S, \forall i = 1, \ldots, N$, $D_{(r)} - r^{th}$ order statistics. Then find a point $p' \in S$ such that

$$d(q, p') \leq D_{(1+\epsilon)}$$

for a given $\epsilon \in \mathbb{Z}^+$. (Hammersley, ’50)

Ranks are still meaningful though!!

(Cayton, ’08)
Rank Approximation
Order Statistics Magic

\[ d(q, p') \leq D_{(1+\epsilon)} \]
Order Statistics Magic

\[ d(q, p') \leq D_{(1+\epsilon)} \]

(Sedransk, Meyer, '78)

\[ P(d_{(r)} \leq D_{(t)}) = \sum_{i=0}^{t-r} \binom{t-i-1}{r-1} \binom{N-t+i}{n-r} / \binom{N}{n} \]
Order Statistics Magic

\[ d(q, p') \leq D_{(1+\epsilon)} \]

(Sedransk, Meyer, ’78)

\[
P(d_{(r)} \leq D_{(t)}) = \sum_{i=0}^{t-r} \binom{t-i-1}{r-1} \binom{N-t+i}{n-r} / \binom{N}{n}
\]

Random stratification into \( N_1 \) and \( N_2 \) points. Samples of sizes \( n_1 \) and \( n_2 \) respectively. The above equation holds with \( N = N_1 + N_2 \) and \( n = n_1 + n_2 \).
Order Statistics Magic

\[ d(q, p') \leq D_{(1+\epsilon)} \]

- (Sedransk, Meyer, '78)

\[
P(d_{(r)} \leq D_{(t)}) = \sum_{i=0}^{t-r} \binom{t-i-1}{r-1} \binom{N-t+i}{n-r} / \binom{N}{n}
\]

- Random stratification into \( N_1 \) and \( N_2 \) points. Samples of sizes \( n_1 \) and \( n_2 \) respectively.
  The above equation holds with \( N = N_1 + N_2 \) and \( n = n_1 + n_2 \).

- Use PAC model!!
Rank Approximate Nearest Neighbor Search

\[ P(d_{(r)} \leq D_{(1+\epsilon)}) \geq \alpha \]
Rank Approximate Nearest Neighbor Search

\[ P(d_{(r)} \leq D_{(1+\epsilon)}) \geq \alpha \]

\[ P(d_{(1)} \leq D_{(1+\epsilon)}) = \sum_{i=0}^{\epsilon} \binom{N - \epsilon + i - 1}{n - 1} / \binom{N}{n} \]
**Rank Approximate Nearest Neighbor Search**

\[ P(d_{(r)} \leq D_{(1+\epsilon)}) \geq \alpha \]

\[ P(d_{(1)} \leq D_{(1+\epsilon)}) = \sum_{i=0}^{\epsilon} \binom{N - \epsilon + i - 1}{n - 1} / \binom{N}{n} \]

- Compute sample size ‘n’ for a given \( \epsilon, \alpha \)
Rank Approximate Nearest Neighbor Search

\[ P(d_{(r)} \leq D_{(1+\epsilon)}) \geq \alpha \]

\[ P(d_{(1)} \leq D_{(1+\epsilon)}) = \sum_{i=0}^{\epsilon} \binom{N - \epsilon + i - 1}{n - 1} / \binom{N}{n} \]

- Compute sample size ‘\( n \)’ for a given \( \epsilon, \alpha \)
  - Binary search in the range \([1 + \epsilon, N]\)
Rank Approximate Nearest Neighbor Search

\[ P(d_{(r)} \leq D_{(1+\epsilon)}) \geq \alpha \]

\[ P(d_{(1)} \leq D_{(1+\epsilon)}) = \sum_{i=0}^{\epsilon} \left( \begin{array}{c} N - \epsilon + i - 1 \\ n - 1 \end{array} \right) / \left( \begin{array}{c} N \\ n \end{array} \right) \]

- Compute sample size ‘\( n \)’ for a given \( \epsilon, \alpha \)
  - Binary search in the range \([1 + \epsilon, N]\)
  - Random sample \( n \) points from the dataset
Rank Approximate Nearest Neighbor Search

\[ P(d_{(r)} \leq D_{(1+\epsilon)}) \geq \alpha \]

\[ P(d_{(1)} \leq D_{(1+\epsilon)}) = \sum_{i=0}^{\epsilon} \binom{N - \epsilon + i - 1}{n - 1} / \binom{N}{n} \]

- Compute sample size \( n \) for a given \( \epsilon, \alpha \)
  - Binary search in the range \([1 + \epsilon, N]\)
- Random sample \( n \) points from the dataset
- Can we do better?
The Algorithm
The Algorithm
The Algorithm
The Algorithm

Why don't we use a tree?
The Algorithm
The Algorithm
The Algorithm
The Algorithm
The Algorithm
The Algorithm
The Algorithm
The Algorithm
The Algorithm
The Algorithm
The Algorithm
The Algorithm
The Algorithm
The Algorithm
The Algorithm

How about dual tree?
The Algorithm

How about dual tree?

Rrrrgh!
Why not!
Rrrrgh!
The Algorithm

How about dual tree?

Make sure you sample enough for all the queries
Experiments and Results
Comparison with Exact Search
Comparison with Exact Search

- Speedups over naive
- Rank error relative to size of dataset
- Error $\varepsilon$ from $0.001\% - 10\%$. 
Comparison with Exact Search

- Speedups over naive
- Rank error relative to size of dataset
- Error $\varepsilon$ from $0.001\% - 10\%$

Datasets:
- Bio (300k×74)
- Corel (40k×32)
- Cover type (600k×55)
- Images (700×4096)
- MNist (60k×784)
- Physics (150k×78)
- Uniform Random (1m×20)
Comparison with Exact Search

- Speedups over naive
- Rank error relative to size of dataset
- Error $\varepsilon$ from $0.001\% - 10\%$

Datasets:
- Bio ($300k \times 74$)
- Corel ($40k \times 32$)
- Cover type ($600k \times 55$)
- Images ($700 \times 4096$)
- MNist ($60k \times 784$)
- Physics ($150k \times 78$)
- Uniform Random ($1m \times 20$)
Comparison with LSH
Comparison with LSH
Comparison with LSH

- Rank errors for same query time
Comparison with LSH

- Rank errors for same query time
  - Datasets: Subsets of size 10k
  - Layout Histogram (dim = 32)
  - MNist (dim = 784)
Comparison with LSH

- Rank errors for same query time
- Datasets: Subsets of size 10k
- Layout Histogram (dim = 32)
- MNist (dim = 784)
Comparison with LSH

- Rank errors for same query time
  Datasets: Subsets of size 10k
- Layout Histogram (dim = 32)
- MNist (dim = 784)

What about variance?
Comparison with LSH

- Rank errors for same query time
  Datasets: Subsets of size 10k
- Layout Histogram (dim = 32)
- MNist (dim = 784)
Conclusion and Future
Conclusion
Conclusion

- Novel idea: Approximating ranks with desired probability
Conclusion

- Novel idea: Approximating ranks with desired probability
- Only two parameters to care about: No new data structure needed
Conclusion

- Novel idea: Approximating ranks with desired probability
- Only two parameters to care about: No new data structure needed
- Fast and meaningful in practice, even for high dimensions
Future directions
Future directions

- Extend to k-NN, maybe also a non-trivial $k^{th}$-NN.
Future directions

- Extend to k-NN, maybe also a non-trivial $k^{th}$-NN.
- Extend to other data structures: Ball trees, Cover trees
Future directions

- Extend to k-NN, maybe also a non-trivial $k^{th}$-NN.
- Extend to other data structures: Ball trees, Cover trees
- Non-euclidean metric
Future directions

- Extend to k-NN, maybe also a non-trivial $k^{th}$-NN.
- Extend to other data structures: Ball trees, Cover trees
- Non-euclidean metric
- Sampling on the query set !!!?????
Thank You

Any questions