Disorder Inequality: A Combinatorial Approach to Nearest Neighbor Search

Navin Goyal (Georgia Tech)  
**Yury Lifshits** (Caltech)  
Hinrich Schütze (Stuttgart University)

Web Search and Data Mining 2008  
Stanford, February 11, 2008
Nearest Neighbors: an Example

**Input:** Set of objects

**Task:** Preprocess it
Nearest Neighbors: an Example

**Input:** Set of objects

**Task:** Preprocess it

**Query:** New object

**Task:** Find the most similar one in the dataset
Nearest Neighbors: an Example

Input: Set of objects
Task: Preprocess it

Query: New object
Task: Find the most similar one in the dataset
Nearest Neighbors

From computational perspective almost all algorithmic problems in the Web represent some form of nearest neighbor problem:

**Search space:** object domain $\mathbb{U}$, similarity function $\sigma$

**Input:** database $S = \{ p_1, \ldots, p_n \} \subseteq \mathbb{U}$

**Query:** $q \in \mathbb{U}$

**Task:** find $\text{argmax} \, \sigma(p_i, q)$
Contribution

- Combinatorial framework: new approach to data mining problems that does not require triangle inequality
- New algorithms for nearest neighbor search
- Experiments
- Tutorial, website
Outline

1. Motivation
2. Combinatorial Framework
3. New Algorithms
4. Directions for Further Research
Motivation
Similarity Search for the Web

- Recommendations
- Personalized news aggregation
- Ad targeting
- “Best match” search
  Resume, job, BF/GF, car, apartment
- Co-occurrence similarity
  Suggesting new search terms
Nearest Neighbors: Prior Work

Sphere Rectangle Tree  Orchard’s Algorithm  k-d-B tree
Geometric near-neighbor access tree  Excluded
middle vantage point forest .mvp-tree  Fixed-height
fixed-queries tree  AESA  Vantage-point
tree  LAESA  R*-tree  Burkhard-Keller tree  BBD tree
Navigating Nets  Voronoi tree  Balanced aspect ratio
tree  Metric tree  vp*-tree  M-tree
Locality-Sensitive Hashing  SS-tree
R-tree  Spatial approximation tree
Multi-vantage point tree  Bisector tree  mb-tree  Cover
tree  Hybrid tree  Generalized hyperplane tree  Slim tree
Spill Tree  Fixed queries tree  X-tree  k-d tree  Balltree
Quadtrees  Octree  Post-office tree
Challenge: Separation Effect

In theory:
Triangle inequality
Doubling dimension is $o(\log n)$
Challenge: Separation Effect

In theory:
Triangle inequality
Doubling dimension is $o(\log n)$

Typical web dataset has separation effect
For almost all $i, j : 1/2 \leq d(p_i, p_j) \leq 1$
Challenge: Separation Effect

**In theory:**
Triangle inequality
Doubling dimension is \( o(\log n) \)

Typical web dataset has separation effect

For almost all \( i, j \):
\[
\frac{1}{2} \leq d(p_i, p_j) \leq 1
\]

Classic methods fail:
In general metric space exact problem is intractable
Branch and bound algorithms visit every object
Doubling dimension is at least \( \log n/2 \)
Combinatorial Framework
Comparison Oracle

- Dataset $p_1, \ldots, p_n$
- Objects and distance (or similarity) function are NOT given
- Instead, there is a comparison oracle answering queries of the form:

  **Who is closer to $A$: $B$ or $C$?**
Disorder Inequality

Sort all objects by their similarity to \( p \):

\[ \text{rank}_p(r) \]

\[ \text{rank}_p(s) \]
Disorder Inequality

Sort all objects by their similarity to $p$:

Then by similarity to $r$:
Disorder Inequality

Sort all objects by their similarity to $p$:

Then by similarity to $r$:

Dataset has disorder $D$ if

$$\forall p, r, s : \ rank_r(s) \leq D(rank_p(r) + rank_p(s))$$
Combinatorial Framework

\[
= \text{Comparison oracle}
\]

Who is closer to A: B or C?

+ 

\[
\text{Disorder inequality}
\]

\[
\text{rank}_r(s) \leq D(\text{rank}_p(r) + \text{rank}_p(s))
\]
Combinatorial Framework: FAQ

- Disorder of a metric space? Disorder of $\mathbb{R}^k$?
- In what cases disorder is relatively small?
- Experimental values of $D$ for some practical datasets?
- Disorder constant vs. other concepts of intrinsic dimension?
Advantages:

- Does not require triangle inequality for distances
- Applicable to any data model and any similarity function
- Require only comparative training information
- Sensitive to “local density” of a dataset
Advantages:

- Does not require triangle inequality for distances
- Applicable to any data model and any similarity function
- Require only comparative training information
- Sensitive to “local density” of a dataset

Limitation: worst-case form of disorder inequality
Disorder vs. Others

- If expansion rate is $c$, disorder constant is at most $c^2$
- Doubling dimension and disorder dimension are incomparable
- Disorder inequality implies combinatorial form of “doubling effect”
3

New Algorithms
Ranwalk Informally (1/2)
Ranwalk Informally (1/2)
Ranwalk Informally (1/2)
Ranwalk Informally (1/2)
Ranwalk Informally (1/2)
Hierarchical greedy navigation:

1. Start at random city $p_1$
Hierarchical greedy navigation:

1. Start at random city $p_1$
2. Among all airlines choose the one going most closely to $q$, move there (say, to $p_2$)
Hierarchical greedy navigation:

1. Start at random city $p_1$
2. Among all airlines choose the one going most closely to $q$, move there (say, to $p_2$)
3. Among all railway routes from $p_2$ choose the one going most closely to $q$, move there ($p_3$)
Hierarchical greedy navigation:

1. Start at random city $p_1$

2. Among all airlines choose the one going most closely to $q$, move there (say, to $p_2$)

3. Among all railway routes from $p_2$ choose the one going most closely to $q$, move there ($p_3$)

4. Among all bus routes from $p_3$ choose the one going most closely to $q$, move there ($p_4$)
Hierarchical greedy navigation:

1. Start at random city \( p_1 \)
2. Among all airlines choose the one going most closely to \( q \), move there (say, to \( p_2 \))
3. Among all railway routes from \( p_2 \) choose the one going most closely to \( q \), move there (\( p_3 \))
4. Among all bus routes from \( p_3 \) choose the one going most closely to \( q \), move there (\( p_4 \))
5. Repeat this \( \log n \) times and return the final city
Hierarchical greedy navigation:

1. Start at random city $p_1$
2. Among all airlines choose the one going most closely to $q$, move there (say, to $p_2$)
3. Among all railway routes from $p_2$ choose the one going most closely to $q$, move there ($p_3$)
4. Among all bus routes from $p_3$ choose the one going most closely to $q$, move there ($p_4$)
5. Repeat this $\log n$ times and return the final city

Transport system: for level $k$ choose $c$ random arcs to $\frac{n}{2^k}$ neighborhood
Ranwalk Algorithm

Preprocessing:
- For every point \( p \) in database we sort all other points by their similarity to \( p \)

Data structure: \( n \) lists of \( n - 1 \) points each.

Query processing:
1. Step 0: choose a random point \( p_0 \) in the database.
2. From \( k = 1 \) to \( k = \log n \) do Step \( k \): Choose \( D' := 3D(\log \log n + 1) \) random points from \( \min(n, \frac{3Dn}{2^k}) \)-neighborhood of \( p_{k-1} \). Compute similarities of these points w.r.t. \( q \) and set \( p_k \) to be the most similar one.
3. If \( \text{rank}_{p_{\log n}}(q) > D \) go to step 0, otherwise search the whole \( D^2 \)-neighborhood of \( p_{\log n} \) and return the point most similar to \( q \) as the final answer.
Analysis of Ranwalk

Assume that database points together with query point $S \cup \{q\}$ satisfy disorder inequality with constant $D$:

$$\text{rank}_x(y) \leq D(\text{rank}_z(x) + \text{rank}_z(y)).$$

Then Ranwalk algorithm always answers nearest neighbor queries correctly.

**Resources:**

- Preprocessing space: $\mathcal{O}(n^2)$
- Preprocessing time: $\mathcal{O}(n^2 \log n)$
- Expected query time: $\mathcal{O}(D \log n \log \log n + D^2)$
Variation: Arwalk

**Arwalk:** moving all random choices to preprocessing

Assume that database points together with query point $S \cup \{q\}$ satisfy disorder inequality with constant $D$

Then for any probability of error $\delta$ Arwalk algorithm answers nearest neighbor query within the following constraints:

- Preprocessing space: $\mathcal{O}(nD \log n (\log \log n + \log 1/\delta))$
- Preprocessing time: $\mathcal{O}(n^2 \log n)$
- Query time: $\mathcal{O}(D \log n (\log \log n + \log 1/\delta))$
Experiment

Reuters-RCV1 corpus:

1. Fix range $R$
2. Choose random $a, b \in [1..R]$
3. Choose random $p \in S$
4. Take $r$ s.t. $\text{rank}_p(r) = a$
5. Take $s$ s.t. $\text{rank}_r(s) = b$
6. Let $c = \text{rank}_p(s)$
7. Return $\frac{c}{a+b}$
3
Directions for Further Research
Recent Results

Yury Lifshits and Shengyu Zhang
Similarity Search via Combinatorial Nets

- Better nearest neighbors:
  - Deterministic
  - Preprocessing $\text{poly}(D)n \log^2 n$ time
  - Price: search time increases to $D^4 \log n$

- Combinatorial algorithms for other problems:
  - Near duplicates
  - Navigation in a small world
  - Clustering
Future of Combinatorial Framework

- Other problems in combinatorial framework:
  - Low-distortion embeddings
  - Closest pairs
  - Community discovery
  - Linear arrangement
  - Distance labelling
  - Dimensionality reduction

- What if disorder inequality has exceptions, but holds in average?
- Insertions, deletions, changing metric
- Metric regularizations
- Experiments & implementation
Yury Lifshits and Shengyu Zhang
Similarity Search via Combinatorial Nets

Navin Goyal, Yury Lifshits, Hinrich Schütze
Disorder Inequality: A Combinatorial Approach to Nearest Neighbor Search

Benjamin Hoffmann, Yury Lifshits, Dirk Novotka
Maximal Intersection Queries in Randomized Graph Models
Summary

- Combinatorial framework: comparison oracle + disorder inequality
- New algorithms: Random walk with nearly $D \log n$ steps
- Further work: Implementing combinatorial algorithms Disorder in average
Summary

- **Combinatorial framework:**
  comparison oracle + disorder inequality

- **New algorithms:**
  Random walk with nearly $D \log n$ steps

- **Further work:**
  Implementing combinatorial algorithms
  Disorder in average

---

**Thanks for your attention!**

**Questions?**