A Modeling Framework for Scalable Near-Duplicate Detection Using Random Projections and Locality-Sensitive Hashing

MA5770 Project Presentation

Group 3:

Ankit Gangwar (MM25D950)
Puneet (ID25S027)
Tanmoy Ghosh (MA25M026)
Rahul Ghosh (MA25M021)

IIT Madras

Outline

Project Introduction

The Problem

High-dimensional data is everywhere in Machine Learning (text, images, sequences), but traditional similarity search methods fail due to the **curse of dimensionality**.

Distance computations become unreliable, and algorithms like K-Means, DBSCAN, KNN become computationally prohibitive.

Our Goal

Develop a **scalable framework** for near-duplicate detection and similarity search that works efficiently even with:

- Millions of data points
- Hundreds/thousands of dimensions
- Real-time query requirements

Key Mathematical Concepts

Our Solution

Our approach leverages two powerful ideas that overcome the curse of dimensionality:

1. Johnson-Lindenstrauss (JL) Lemma

- Random projection theorem
- Reduces dimensions: $d \to k = O(\log n/\varepsilon^2)$
- Preserves distances within $(1 \pm \varepsilon)$
- Enables dimensionality reduction with guarantees

2. Locality-Sensitive Hashing (LSH)

- Hash similar items together
- Sublinear query time: $O(n^\rho)$, $\rho < 1$
- Probabilistic guarantees
- Fast approximate nearest neighbor search

Combined Power

Together, these techniques enable scalable similarity search in high dimensions

Key References

Foundational Papers

Our approach is built on these seminal works:

1. **Johnson, W. B., & Lindenstrauss, J.** (1984). Extensions of Lipschitz mappings into a Hilbert space. *Contemporary Mathematics*, 26(189-206), 1.
2. **Indyk, P., & Motwani, R.** (1998, May). Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing* (pp. 604-613).
3. **Gionis, A., Indyk, P., & Motwani, R.** (1999, September). Similarity search in high dimensions via hashing. In *VLDB* (Vol. 99, No. 6, pp. 518-529).

Note

These papers form the theoretical foundation for scalable similarity search in high-dimensional spaces.

The Reality of High-Dimensional Data

Where Do We Encounter High Dimensions?

Machine Learning practitioners constantly face high-dimensional data:

1. **Text Data**
   - TF-IDF Vectorizer
   - Count Vectorizer
   - One-Hot Encoding
   - Bag of Words

   $\Rightarrow$ Dimensionality grows with vocabulary size

2. **Image Data**
   - Pixel-by-pixel expansion
   - Each pixel becomes a feature
   - RGB channels multiply dimensions

   $\Rightarrow$ For 256$\times$256 RGB: 196,608 dimensions!

Reality Check

Real-world applications routinely handle data in hundreds to thousands of dimensions.

The Curse of Dimensionality

What Goes Wrong in High Dimensions?

1. **Distance Becomes Meaningless**
   - All points appear roughly equidistant
   - Cannot distinguish "near" from "far"
2. **Similarity Search Becomes Expensive**
   - Must compute distance to all points
   - Computational cost explodes
   - O(nd) for each query point
3. **Algorithms Slow Down Drastically**
   - K-Means clustering
   - DBSCAN
   - K-Nearest Neighbors (KNN)
   - HDBSCAN

Mathematical Intuition

In high dimensions (d):

$$\frac{\text{max dist} - \text{min dist}}{\text{min dist}} \to 0$$

as $d \to \infty$

Problem Statement

Core Challenge

Given: A large dataset of high-dimensional points

Goal: Efficiently find near-duplicate or similar items

Constraint: Must scale to millions of points and hundreds/thousands of dimensions

Traditional Approach Fails:

- Exact nearest neighbor: $O(nd)$ per query
- For $n = 1$M points, $d = 1000$:

  $\Rightarrow$ 1 billion computations per query!
- Tree-based methods degrade to linear

What We Need:

- Sublinear query time: $o(n)$
- Acceptable approximation
- Provable guarantees
- Practical implementation

Mathematical Formulation

Formal Problem Definition

Input:

- Dataset: $X = \{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}^d$
- Query point: $q \in \mathbb{R}^d$
- Similarity threshold: $r > 0$

Output:

- All points $x_i \in X$ such that $\|x_i - q\|_2 \leq r$
- Or: The k nearest neighbors to q

The Two-Pronged Solution

Step 1: Dimensionality Reduction

Johnson-Lindenstrauss Lemma

Randomly project data from $\mathbb{R}^d$ to $\mathbb{R}^k$:

$$k = O\left(\frac{\log n}{\varepsilon^2}\right)$$

Guarantee: All pairwise distances preserved within $(1 \pm \varepsilon)$ with high probability

- Projection matrix: $\Phi \in \mathbb{R}^{k \times d}$
- Entries: i.i.d. Gaussian $\mathcal{N}(0, 1/k)$
- Linear transformation: $y = \Phi x$

$\Rightarrow$ Reduces computational burden

Step 2: Fast Similarity Search

Locality-Sensitive Hashing (LSH)

- Similar points collide with high probability
- Dissimilar points collide with low probability
- Sublinear query time: $O(n^\rho)$ where $\rho < 1$
- **No inspection of entire dataset!**

$\Rightarrow$ Enables fast retrieval

Why Probabilistic Models?

Deterministic Algorithms

- × Exact but slow
- × Exponential in dimension
- × Don't scale to real data
- × Theoretical complexity: $O(2^d)$

Probabilistic Models (Our Approach)

- ✓ Approximate but fast
- ✓ Polynomial or sublinear time
- ✓ Proven guarantees
- ✓ Practical for real data

Example

k-d tree for d = 50:
$2^{50} \approx 10^{15}$ operations!

Our Guarantees

With high probability:
- Distance preserved within $(1 \pm \varepsilon)$
- Query time: $O(d \log n)$ or $O(dn^{\rho})$
- Works for millions of points

Project Goals & Methodology

What We Will Accomplish

1. **Theoretical Foundation**
   - Prove and understand the Johnson-Lindenstrauss Lemma
   - Analyze concentration inequalities and probabilistic guarantees
   - Understand LSH hash function families
2. **Algorithm Development**
   - Implement random projection (JL Lemma)
   - Design and implement LSH data structures
   - Optimize parameters: k (projection dim), L (hash tables), collision probability
3. **Real-World Validation**
   - Application: Near-duplicate text detection
   - Dataset: Publicly available document collections
   - Compare against: K-Means, HDBSCAN, exact search

Expected Outcomes

Performance Expectations

- **Speed:** 10–100$\times$ faster than exact search
- **Scalability:** Handle millions of documents
- **Accuracy:** $>$90% recall with small ε
- **Dimension Independence:** Performance stable as d increases

Comparison

Traditional clustering: Degrades rapidly with d
Our approach: Maintains efficiency

Broader Impact

Applications beyond text:

- **Plagiarism Detection** in code repositories
- **Recommendation Systems** content-based filtering
- **Search Engines** web-scale deduplication
- **Bioinformatics** sequence matching
- **Cybersecurity** anomaly detection

Johnson–Lindenstrauss Lemma: Statement

Theorem (Johnson–Lindenstrauss Lemma)

Let V be a set of n points in $\mathbb{R}^d$ and let ε satisfy $0 < \varepsilon < 1$. Then there exists a mapping

$$f : \mathbb{R}^d \to \mathbb{R}^k, \quad \textit{where} \quad k = O\left(\frac{\log n}{\varepsilon^2}\right),$$

such that for all $u, v \in V$,

$$(1-\varepsilon)\,\|u-v\|^2 \leq \|f(u)-f(v)\|^2 \leq (1+\varepsilon)\,\|u-v\|^2.$$

Understanding the JL Lemma

What Does It Say?

- We can project n points from $\mathbb{R}^d$ to $\mathbb{R}^k$
- Where $k \ll d$ (much smaller dimension)
- While preserving all pairwise distances within factor $(1 \pm \varepsilon)$
- The map f is a **random projection**

Key Insight

The target dimension k depends on:

- Number of points: $\log n$
- Desired accuracy: $1/\varepsilon^2$

But **NOT** on original dimension d!

Example

Suppose we have:

- $n = 1{,}000{,}000$ points
- $d = 10{,}000$ dimensions
- $\varepsilon = 0.1$ (10% error)

Then:

$$k = O\left(\frac{\log 10^6}{0.01}\right) = O(1{,}400)$$

Reduction

From 10,000 dimensions to $\sim$1,400 dimensions!

No Isometry Exists

Linear Transformation and Norm Preservation

Consider $R \in \mathbb{R}^{k \times d}$ with $u \mapsto Ru$. Goal: $\|u\|_2 = \|Ru\|_2$ for all $u \in \mathbb{R}^d$

Mathematical Analysis:

Expanding: $\|Ru\|_2^2 = u^T R^T R u$

For preservation: $u^T R^T R u = u^T u$

This requires: $R^T R = I_d$

Why This Fails

When $k < d$:

- R^T is $d \times k$, R is $k \times d$
- $R^T R$ is $d \times d$ but not full rank

Conclusion: Exact isometry is impossible for $k < d$

Our Goal: Approximate Isometry

Use **random** R to achieve: $(1 - \varepsilon)\|u\|_2^2 \leq \|Ru\|_2^2 \leq (1 + \varepsilon)\|u\|_2^2$ with high probability

In next class

We will begin with some more insightful mathematical aspect of JL-Lemma

Johnson-Lindenstrauss Lemma :

Approximate Isometry of JL-Lemma