



Approximate
Nearest Neighbor
Search via Group
Testing

Authors

Introduction

Locality Sensitive
Hashing

Distance-
Sensitive Bloom
Filters

Algorithm

APPROXIMATE NEAREST NEIGHBOR SEARCH VIA GROUP TESTING

Saksham Rathi
(22B1003)

Kshitij Vaidya
(22B1829)

Ekansh Ravi Shankar
(22B1032)

CS754: ADVANCED IMAGE PROCESSING
UNDER PROF. AJIT RAJWADE

Indian Institute of Technology Bombay
Spring 2025

Contents



Approximate
Nearest Neighbor
Search via Group
Testing

Authors

Introduction

Locality Sensitive
Hashing

Distance-
Sensitive Bloom
Filters

Algorithm

- 1 Introduction
- 2 Locality Sensitive Hashing
- 3 Distance-Sensitive Bloom Filters
- 4 Algorithm

Nearest Neighbor Search



Approximate
Nearest Neighbor
Search via Group
Testing

Authors

Introduction

Locality Sensitive
Hashing

Distance-
Sensitive Bloom
Filters

Algorithm

- Nearest neighbor search is a fundamental problem with many applications in machine learning systems.
- **Task:** Given a dataset $D = \{x_1, x_2, \dots, x_N\}$, the goal is to build a data structure that can be queried with any point q to obtain a small set of points $x_i \in D$ that have high similarity (low distance) to the query. This structure is called an index.
- Such tasks frequently arise in genomics, web-scale data mining, machine learning, and other large-scale applications.

Group Testing



Approximate
Nearest Neighbor
Search via Group
Testing

[Authors](#)

Introduction

Locality Sensitive
Hashing

Distance-
Sensitive Bloom
Filters

Algorithm

- We are given a set D of N items, with k positives (“hits”) and $N - k$ negatives (“misses”).
- **Goal:** Identify all positive items using fewer than N group tests.
- A *group test* is positive iff at least one item in the group is positive.
- **Testing Variants:** Can be *noisy* (with false positives/negatives), *adaptive* (tests depend on previous results), or *non-adaptive* (all tests run in parallel).
- The paper uses a **doubly regular design**: Each item appears in an equal number of tests; each test has an equal number of items.

Formal Problem Statement



Approximate
Nearest Neighbor
Search via Group
Testing

[Authors](#)

Introduction

Locality Sensitive
Hashing

Distance-
Sensitive Bloom
Filters

Algorithm

- **(R, c)-Approximate Near Neighbor:** Given a dataset D , if there exists a point within distance R of a query y , return some point within distance $c \cdot R$, with high probability.
 - R is the distance threshold (radius).
 - $c > 1$ is the approximation factor.
- Any algorithm that solves the randomized nearest neighbor problem also solves the approximate near neighbor problem with $c = 1$ and any $R \geq$ distance to the nearest neighbor.
- (Definition) **Randomized Nearest neighbor:** Given a dataset D and a distance metric $d(\cdot, \cdot)$ and a failure probability $\delta \in [0, 1]$, construct a data structure which, given a query point y reports the point $x \in D$ with the smallest distance $d(x, y)$ with probability greater than $1 - \delta$.

Locality Sensitive Hashing



Approximate
Nearest Neighbor
Search via Group
Testing

[Authors](#)

Introduction

Locality Sensitive
Hashing

Distance-
Sensitive Bloom
Filters

Algorithm

A hash function $h(x) \rightarrow \{1, \dots, R\}$ is a function that maps an input x to an integer in the range $[1, R]$.

The two points x and y are said to collide if $h(x) = h(y)$.

$$s(x, y) = \Pr_H(h(x) = h(y))$$

For now, we will assume that $s(x, y) = \text{sim}(x, y)$.

For any positive integer L , we may transform an LSH family H with collision probability $s(x, y)$ into a new family having $s(x, y)^L$ by sampling L hash functions from H and concatenating the values to obtain a new hash code $[h_1(x), h_2(x), \dots, h_L(x)]$. If the original hash family had the range $[1, R]$, the new hash family has the range $[1, R^L]$.

Locality Sensitive Hashing



Approximate
Nearest Neighbor
Search via Group
Testing

[Authors](#)

Introduction

Locality Sensitive
Hashing

Distance-
Sensitive Bloom
Filters

Algorithm

- **Locality Sensitive Hashing (LSH)** algorithms use an LSH function to partition the dataset into buckets.
- The hash function is selected so that the distance between points in the same bucket is likely to be small.
- To find the near neighbors of a query, we hash the query and compute the distance to every point in the corresponding bucket.
- **Count-Based LSH** identifies neighbors by simply counting how many times two points land in the same hash bucket across multiple hash functions.

Distance-Sensitive Bloom Filters



Approximate
Nearest Neighbor
Search via Group
Testing

[Authors](#)

Introduction

Locality Sensitive
Hashing

Distance-
Sensitive Bloom
Filters

Algorithm

- (Definition) **Approximate Set Membership:** Given a set D of N points and similarity thresholds S_L and S_H , construct a data structure which, given a query point y , has:
True Positive Rate: If there is $x \in D$ with $\text{sim}(x, y) > S_H$, the structure returns true w.p. $\geq p$
False Positive Rate: If there is no $x \in D$ with $\text{sim}(x, y) > S_L$, the structure returns true w.p. $\leq q$
- The distance-sensitive Bloom filter solves this problem using LSH functions and a 2D bit array. The structure consists of m binary arrays that are each indexed by an LSH function. There are three parameters: the number of arrays m , a positive threshold $t \leq m$, and the number of concatenated hash functions L used within each array.

Distance-Sensitive Bloom Filters



Approximate
Nearest Neighbor
Search via Group
Testing

[Authors](#)

Introduction

Locality Sensitive
Hashing

Distance-
Sensitive Bloom
Filters

Algorithm

- To construct the filter, we insert elements $x \in D$ by setting the bit located at array index $[m, h_m(x)]$ to 1.
- To query the filter, we determine the m hash values of the query y . If at least t of the corresponding bits are set, we return true. Otherwise, we return false.
- (Theorem) Assuming the existence of an LSH family with collision probability $s(x, y) = \text{sim}(x, y)$, the distance-sensitive Bloom filter solves the approximate membership query problem with

$$p \geq 1 - \exp(-2m(-t + S_H^L)^2)$$

$$q \leq \exp(-2m(-t + NS_L^L)^2)$$



Input: Dataset D of size N , positive integers B and R , similarity threshold S

Output: A FLINNG search index consisting of membership sets $M_{r,b}$ and group tests $C_{r,b}$

- For $r = 0$ to $R - 1$:
 - Let $\pi(D)$ be a random permutation of D
 - For $b = 0$ to $B - 1$:
 - Define $M_{r,b} = \{\pi(D)_i \mid i \bmod B = b\}$
- For $r = 0$ to $R - 1$:
 - For $b = 0$ to $B - 1$:
 - Construct a classifier $C_{r,b}$ for membership set $M_{r,b}$ with true positive rate p and false positive rate q



- If we apply a similarity threshold to the dataset, we obtain a near neighbor set $K = \{x \in D | \text{sim}(x, y) \geq S\}$. We consider K to be the set of “positives” in the group testing problem.
- In order to do so, we split the dataset D into a set of groups, which we visualize as a $B \times R$ grid of cells. Each cell has a group of items $M_{r,b}$ and a corresponding group test $C_{r,b}$. To assign items to cells, we evenly distribute the N points among the B cells in each column of the grid, and we independently repeat this assignment process R times.

Index Query



Approximate
Nearest Neighbor
Search via Group
Testing

Authors

Introduction

Locality Sensitive
Hashing

Distance-
Sensitive Bloom
Filters

Algorithm

Input: A FLINNG index and a query y

Output: Approximate set \hat{K} of neighbors with similarity greater than the threshold S

- Initialize $\hat{K} = \{1, \dots, N\}$
- For $r = 0$ to $R - 1$:
 - Initialize $Y = \emptyset$
 - For $b = 0$ to $B - 1$:
 - If $C_{r,b}(y) = 1$ then: $Y = Y \cup M_{r,b}$
 - $\hat{K} = \hat{K} \cap Y$



- To query the index with a point y , we begin by querying each classifier. If $C_{r,b}(y) = 1$, then at least one of the points in $M_{r,b}$ has high similarity to y . We collect all of these “candidate points” by taking the union of the $M_{r,b}$ sets for which $C_{r,b}(y) = 1$.
- We repeat this process for each of the R repetitions to obtain R candidate sets, one for each column in the grid.
- With high probability, each candidate set contains the true neighbors, but it may also have some non-neighbors that were included in $M_{r,b}$ by chance. To filter out these points, we intersect the candidate sets to obtain our approximate near neighbor set \hat{K} .