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

Approximate Nearest Neighbor Search via Group Testing
Authors
Introduction
Locality Sensitive Hashing
Distance-Sensitive Bloom Filters

Contents

1. Introduction
2. Locality Sensitive Hashing
3. Distance-Sensitive Bloom Filters

Nearest Neighbor Search

- Nearest neighbor search is a fundamental problem with many applications in machine learning systems.
- **Task:** Given a dataset $D = \{x_1, x_2, \ldots, x_N\}$, 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

- 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.

Approximate Nearest Neighbor Search via Group Testing
Authors
Introduction
Locality Sensitive Hashing
Distance-Sensitive Bloom Filters

Formal Problem Statement

- **(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

A hash function $h(x) \to \{1, \ldots, R\}$ 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 $s(x, y) = sim(x, y)$.
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 $[h_1(x), h_2(x), ..., h_L(x)]$. If the original hash family had the range $[1, R]$, the new hash family has the range $[1, R^L]$.

Locality Sensitive Hashing

- **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

- (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 $sim(x, y) > S_H$, the structure returns true w.p. $\geq p$
  False Positive Rate: If there is no $x \in D$ with $sim(x, y) > S_L$, 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 threeparameters: 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

- 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 $s(x, y) = sim(x, y)$, the distance-sensitive Bloom filter solves the approximate membership query problem with

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

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