Randomized Algorithms 2025A*
Lecture 6 – Nearest Neighbor Search in ℓ_1

Robert Krauthgamer

1 Sketching Distances

What is Sketching: We want to compress/summarize some input x into a *sketch* $s(x)$ (of small size), but want to be able to later compute some $f(x)$ only from the sketch. Often, randomization helps. We'll denote it as $s_r(x)$ where r is the sequence of random coins.

Examples:

1. In graphs: a Gomory-Hu tree can report min st-cuts; a spanner can report all pairwise distances (approximately); both are deterministic.

2. Sketching $x \in \mathbb{R}^n$ so that later we could estimate any x_i (usually the approximation is good only for large entries).

3. Sketching for equality testing: test whether $h(x) = h(y)$ use a hash function $h : \{0,1\}^n \to \{0,1\}^t$, for instance a random function or as in the exercise below. It's important here to choose h using public randomness, i.e., same h for both x, y.

Exer: Show that the hash function $h_r(x) = \sum_{i=1}^n x_i r_i \pmod 2$, where $\vec{r} \in \{0,1\}^n$ is random, is a good sketch for equality testing in the sense that

$$\forall x \neq y, \qquad \Pr_r[h_r(x) = h(y)] = 1/2.$$

4. Sketching for ℓ_p distance, namely, for all $x, y \in [\Delta]^d$,

$$\Pr[a(s_r(x), s_r(y)) = (1 \pm \varepsilon)\|x - y\|_p] \geq 2/3.$$

The JL transform offers such a sketch for ℓ_2 norm. We saw a specific implementation using a linear sketch $L : \mathbb{R}^d \mapsto \mathbb{Z}^k$ for $k = O(1/\varepsilon^2)$, hence $|s(x)| \leq O(\varepsilon^{-2} \log(d\Delta))$ bits.

Question: Can we use (for ℓ_1 or ℓ_2) only $O(\varepsilon^{-2})$ bits? No if we want an estimate. But maybe for a decision version (output is YES/NO)?

*These notes summarize the material covered in class, usually skipping proofs, details, examples and so forth, and possibly adding some remarks, or pointers. The exercises are for self-practice and need not be handed in. In the interest of brevity, most references and credits were omitted.

Definition: In *distance estimation*, the input is two inputs (e.g., vectors) and the goal is to approximate their distance within factor $1+\varepsilon$. In the decision (aka promise) version, the goal is to decide whether the distance is $\leq R$ or $> (1+\varepsilon)R$ for a parameter $R > 0$ given in advance.

Theorem 1 [Estimating ℓ_1 distance]: For every $0 < \varepsilon < 1$ there is a randomized sketch that can estimate the ℓ_1 (or Hamming) distance between two input vectors within $(1+\varepsilon)$-approximation in the decision version, with sketch size $O(1/\varepsilon^2)$ bits.

Proof: Was seen in class. The sketching algorithm has two steps, first choose $I \subset [n]$ to subsample the coordinates with rate $1/R$, and second apply on x_I, y_I the equality testing mentioned above.

Review of key points:

1. Design a single-bit sketch with small "advantage"
2. Amplify success probability using Chernoff bounds

2 NNS under ℓ_1 norm (logarithmic query time)

Problem definition (NNS): Preprocess a dataset of n points $x_1, \ldots, x_n \in \mathbb{R}^d$, so as to quickly find the closest data point to a query point $q \in \mathbb{R}^d$, i.e. report x_i that minimizes $\|q - x_i\|_1$.

Performance measure: Preprocessing (time and space) and query time.

Discretization: Assume all points come from $[\Delta]^d$, where $\Delta = \text{poly}(n)$.

Two naive solutions:

- Exhaustive search/Linear scan: query time is $O(nd)$, preprocessing is $O(nd)$
- Exhaustive storage: prepare all answers in advance with preprocessing space $[\Delta]^d$, then query time is $O(d)$.

Challenge: get query time sublinear (or polylog) in n, but still be polynomial in dimension d.

Approximate version (factor $c \geq 1$): find x_j such that $\|q - x_j\|_1 \leq c \cdot \min_i \|q - x_i\|_1$.

Theorem 2 [Indyk-Motwani'98, Kushilevitz-Ostrosvky-Rabani'98]: For every $\varepsilon > 0$ there is a randomized algorithm for $1+\varepsilon$ approximate NNS in $\mathbb{Z}^d$ under ℓ_1-norm with preprocessing space $n^{O(1/\varepsilon^2)} \cdot O(d)$ and query time $O(\varepsilon^{-2} d \,\text{polylog}\, n)$.

Remark 1: We shall omit/neglect the precise polynomial dependence on d.

Remark 2: The success probability is for a single query (assuming it's independent of the coins).

Remark 3: We only need to solve the decision version, i.e., there is a target distance $R > 0$, and if there is data point x_j such that $\|q - x_j\|_1 \leq R$ then the algorithm reports a point x_i such that $\|q - x_i\|_1 \leq cR$. If no point is within distance cR, then report NONE. Otherwise, can report either answer. This follows by preparing in advance for all powers of $1+\varepsilon$ as the value of R (then trying all of them or binary search).

Remark 4: WLOG x_i and q are in $\{0,1\}^d$.

Proof: Was seen in class. The main idea is to repeat the above single-bit sketching algorithm $k = O(\varepsilon^{-2} \log n)$ times to reduce the error probability to (say) $1/n^2$, and prepare in advance the answer for every $v \in \{0,1\}^m$ as a possible $s(q)$.

Review of key points:

1. "dimension reduction" to $O(\varepsilon^{-2} \log n)$.
2. Prepare all answers in advance (exponential in "reduced" dimension).

3 NNS via LSH (sublinear query time)

Consider again approximate NNS in the decision version, with target distance $R > 0$ and approximation factor $c > 1$, e.g. $c = 1 + \varepsilon$, but here we actually focus on larger c.

Locality Sensitive Hashing (LSH): A c-LSH is a family H of hash functions $h : \{0,1\}^d \to \mathbb{N}$ whose collision probability for all $x, y \in \{0,1\}^d$ is:

1. if $\|x - y\|_1 \leq R$ then $\Pr[h(x) = h(y)] \geq p$;
2. if $\|x - y\|_1\ cR$ then $\Pr[h(x) = h(y)] \leq p'$.

Think of R, p as given inputs, c is the approximation factor, and p' determines the performance (should be much smaller than p).

Note: We also need that $h \in H$ can be chosen quickly and $h(x)$ can be computed quickly. Here, we ignore this issue.

Theorem 3 [LSH for Hamming distance; Indyk-Motwani'98]: For every d, R, c and $p < 1/3$ there is c-LSH for ℓ_1 distance in $\{0,1\}^d$, such that $p' \leq O(p^c)$.

Proof: Was seen in class. For $p = 1/e$, we construct $h(x)$ by sampling $t = d/R$ coordinates from $[d]$ independently ar random.

We can get any desired smaller p by increasing the number of samples t.

Theorem 4 [c-NNS scheme from c-LSH]: Consider the decision version (with target distance $R > 0$) and fix an approximation $c > 1$. Let H be a c-LSH with some p and $p' = O(1/n)$. Then there is c-NNS with query time $O(1/p)$ and preprocessing $O(n/p)$.

Remark: For ℓ_1 norm $p = 1/n^{1/c}$.

Proof: Was discussed shown in class. The main idea is to use the LSH to compute the hash for all data points $x_1, \ldots, x_n$ (at the preprocessing stage), then for the q (at the query stage), and check points x_i in the same bucket with q (i.e., points that collide with the query) by computing the actual distance. A correct output x_i (if exists) has success probability p, which we can amplify by repeating the above $O(1/p)$ times. The time spent on checking points that are too far from q is in expectation by $O(p'n)$ per repetition, and we use Markov's inequality to bound it with high probability.