Randomized Algorithms 2013A
Lecture 12 – Nearest Neighbor Searching (NNS) in High Dimension*

Robert Krauthgamer

Let's briefly recall our discussion of sketching algorithms.

What is Sketching: We have some input x, which we want to "compress" into a *sketch* $s(x)$ (much smaller), but want to be able to later compute some $f(x)$ only from the sketch. Often, randomization helps.

1 Sketching for estimating ℓ_1 distance

Theorem 1: For all $0 < \varepsilon < 1$ there is a randomized sketching algorithm for the decision version of estimating the ℓ_1 (or Hamming) distance between vectors within factor $1 + \varepsilon$ with sketch size $O(1/\varepsilon^2)$ bits. Formally, this algorithm determines whether $\|x - y\|_1 \leq R$ or $\|x - y\|_1 > (1 + \varepsilon)R$.

Proof: As seen in class, the sketching algorithm is based on subsampling the coordinates at a rate of $1/R$.

Exer: Show that the error probability can be reduced to $1/n^3$ by further increasing the sketch size to $m = O(\varepsilon^{-2} \log n)$ bits.

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 that then, given a query point $q \in \mathbb{R}^d$, we can quickly find the closest data point to the query, i.e. report x_i that minimizes $\|q - x_i\|_1$.

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

*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.

Two naive solutions: exhaustive search with query time $O(n)$, and preparing all answer in advance with preprocessing space 2^d (at least).

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

Approximate version (factor $c \geq 1$): find x_{i^*} such that $\|q - x_{i^*}\|_1 \leq c \cdot \min_i \|q - x_i\|_1$.

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

Remark 1: We shall neglect the precise polynomial dependence on d, as it depends on the implementation.

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

Remark 3: WLOG, we only need to solve the decision version i.e. there is a target distance $R > 0$, and if there is data point x_{i^*} such that $\|q - x_{i^*}\|_1 \leq R$ then we need to find 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.

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

Review of key points:

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

Open problem: What about other ℓ_p norm, when $2 < p < \infty$?

Remark: $1 \leq p \leq 2$ and $p = \infty$ are known.

3 NNS via LSH (polynomial query time)

Consider again the context of doing NNS, in the decision version where there is a 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 $\|x - y\|_1 \geq cR$ then $\Pr[h(x) = h(y)] \leq p'$.

Here, R, p are given as input, 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 Hamming distance in $\{0,1\}^d$, such that $p' \leq O(p^c)$.

Proof: As seen in class, in the case $p = 1/e$, $h(x)$ is constructed by sampling $t = d/R$ coordinates from $[d]$ independently ar random.

Theorem 4 [c-NNS scheme from c-LSH]: Consider the decision version (i.e. we have target distance R) 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 sketch: The main idea is to use the LSH to hash the data points $x_1, \ldots, x_n$, and then given a query q, hash also q and check (by computing the actual distance) all the x_i that are in the same bucket.