COMS 4995-3: Advanced Algorithms — Feb 15, 2017

Lecture 9 – Nearest Neighbor Search: Locality Sensitive Hashing.

Instructor: *Alex Andoni* — Scribes: *Weston Jackson, Edo Roth*

1 Introduction

Today's lecture is about using Locality Sensitive Hashing to solve the problem of approximate Nearest Neighbor Search. The main problem is concerning performing an approximate Nearest Neighbor Search in approximately linear space and sublinear time. The motivation behind the problem is to be able to be able to quickly query a data set for nearest neighbor, while minimizing the complexity of our data structure.

2 Nearest Neighbor Search

Theorem 1 (KOR '98 [**?**]). $\forall d, r \in \mathbb{Z}, \exists \ \ \theta \in (\frac{1}{4}, \frac{3}{4})$, *can pick random* $\varphi : \{0,1\}^d \to \{0,1\}^k, k = O(\frac{lgn}{\epsilon^2})$ *s.t.* $\forall p, q \in \{0,1\}^d$ *:*

Close: if $\| p - q \| \leq r \Rightarrow Pr[\| \varphi(p) - \varphi(q) \| \leq \theta k] \geq 1 - \frac{1}{n^3}$

Far: if $\| p - q \| > cr \Rightarrow Pr[\| \varphi(p) - \varphi(q) \| > \theta(1+\epsilon)k] \geq 1 - \frac{1}{n^3}$

Where c is the approximation factor (i.e. $c = 1 + \epsilon$, $c = 2$*)*

2.1 Solution 1: Linear Search

Given data set $D \subset \{0,1\}^d$, precompute $\varphi(p)$, $\forall p \in D$.

- On query $q \in \{0,1\}^d$
- Compute $\varphi(q) \in \{0,1\}^k$
- Compare $\varphi(q)$ to all $\{\varphi(p)\}$, $p \in D$

Complexity:

- Space: $O(n \log n / \epsilon^2)$
- Query: $O(n \frac{\log n}{\epsilon^2})$

We note that while this is a good space complexity, our query time is larger than desired. We also note that the k bits produced from computing $\varphi(q)$ is the only information we need from the query, and we leverage this fact in our next solution.

2.2 Solution 2: Exhaustive Storage

Given data set $D \subset \{0, 1\}$, precompute answer for all possible k bit strings ($\varphi(q)$).

- Space: $O(2^k) = O(2^{\frac{\log n}{\epsilon^2}}) = n^{O(1/\epsilon^2)}$
- Query: Compute $\sigma = \varphi(q)$. $O(\text{time to compute } \varphi(q)) = O(dk) = O(d\frac{\log n}{\epsilon^2})$.

While this yields theoretically good bounds, unfortunately, the actual space in practice is closer to n^{4/ϵ^2}. Can we improve this?

3 Solution 3: Locality-Sensitive Hashing (Geometric Hashing)

We introduce the notion of locality-sensitive hashing (which can be thought of as a form of geometric hashing) to give a solution with better space complexity (albeit not quite as good query time). Intuitively, this specifies a hash family for which it is actually good to collide, a slightly different treatment from our earlier encounters with universal and perfect hashing.

Definition 2. *Hash family $\mathcal{H} = \{h : \mathbb{R}^d \to U\}$ is $(r, cr, p_1, p_2) - LSH$ if $\forall p, q \in \mathbb{R}^d$:*

Close: if $||p - q|| \leq r \to Pr_{h \in H}[h(p) = h(q)] \geq p_1$

Far: if $||p - q|| > cr \to Pr_{h \in H}[h(p) = h(q)] \leq p_2$

Where $p_1 > p_2, c > 1$.

3.1 Ideal LSH

Example 3. *Ideal Locality-Sensitive Hashing is the case where $p_1 = 1, p_2 = 0$. To solve, we use a hash table (independent from data set d).*

- *Build a dictionary on $h(p)$, $p \in D, h \in \mathcal{H}$ using a universal hash function*
- *On query q, look up bucket $h(q)$*
- *The bucket $h(q)$ contains precisely all nearest neighbors and no far neighbors*

Complexity:

- Space: $O(n) + O(nd)$
- Query: time to compute $h + O(1)$ on expectation

Can we build this? Consider the space partition of points p, q in hash function h. If $h(p) \neq h(q)$, there is a border separating p and q in the space partition. Consider two points x and y that are neighbors on either side of the space partition. These points are hashed separately despite being very close. **Need multiple hash functions!**

3.2 LSH for ANN: Proof Part 1

Theorem 4 (IM '98 [**?**]). *$(r, cr, p_1, p_2) - LSH$ implies solution for Approximate Nearest Neighbor with:*

Space: $O(nd + n^{1+\varrho/p_1})$

Query time: $O(n^{\varrho}/p_1 log(n))$

Where $\varrho = \frac{\log 1/p_1}{\log 1/p_2} \in (0, 1)$.

Proof. Property of any LSH $\mathcal{H}$: fix $k \in \mathbb{N}$. Build $G = \{g : \mathbb{R}^d \to U^k\}$ as follows:

$$g(p) := h_1(p) \cdot h_2(p)...h_k(p) \in U^k$$

Where $h_1...h_k$ are iid from $\mathcal{H}$ (and $\cdot$ denotes concatenation).

Fact: If $\mathcal{H}$ is $(r, cr, p_1, p_2) - LSH$, G is $(r, cr, p_1^k, p_2^k) - LSH$. If p,q are close:

$$Pr_g[g(p) = g(q)] = \prod_{i=1}^{k} Pr[h_i(p) = h_i(q)] \geq p_1^k$$

Fix k, take $L := (1/p_1)^k$. Use the following algorithm:

- Build L hash tables each with a fresh, random $g_i \in G_k$
- Store all $g_i(p)$ in a dictionary
- On query q, for tables $i = 1...L$
  - Retrieve points $p \in D$ where $g_i(p) = g_i(q)$
  - Return first p where $d(p, q) < cr$

3.3 LSH for ANN: Proof Part 2 (Analysis)

Correctness

1) Never reports a far point

2) If $\exists p^*$ which is distance $\leq r$ from q, fix $i \in L$:

$$Pr_{g_i \in G}[g_i(p^*) = g_i(q)] \geq p_1^k$$

Thus, the probability the algorithm fails is bounded by:

$$Pr[\nexists i \text{ s.t. } g_i(p^*) = g_i(q)] \leq (1 - p_1^k)^L = e^{-p_1^k L} = e^{-1} < 0.4$$

Query Time

The query time is Lk hash function evaluations + # far points colliding with q in all l hash functions. We first calculate the expected number of points colliding with q in one hash table.

$$E[\# \text{ far points colliding in 1 hash table}] \leq np_2^k$$

Calculating the total query time:

$$E[\text{total time}] = O(Lk\tau) + O(Lnp_2^k)$$

Where τ is the hash function evaluation time. Let $k = \lceil \log_{1/p_2} n \rceil = \lceil \frac{\log n}{\log(1/p_2)} \rceil$. Fixing k, take L from before:

$$L = (1/p_1)^k \leq (1/p_1)^{\log_{1/p_2} n+1}$$
$$= n^{\varrho}/p_1$$

Where $\varrho = \log_{1/p_2} 1/p_1$. Substituting $L = n^{\varrho}/p_1$:

$$E[\text{total time}] \leq O(\tau n^{\varrho}/p_1 \log n) + O(Lnp_2^k) \leq O(\tau n^{\varrho}/p_1 \log n) + O(n)$$

$$= O(\tau n^{\varrho}/p_1 \log n)$$

Thus, we achieve the following complexity:

- Space: $O(nd + Ln) = O(nd + n^{(1+\varrho)}/p_1)$
- Query: $O(\tau n^{\varrho}/p_1 \log n)$

□

4 LSH for ANN: Hamming Spaces

Can we build LSH? We can construct an LSH for Hamming spaces, $\{0,1\}^d$.

Hash family $\mathcal{H} = \{h : \mathbb{R}^d \to U = \{0,1\}\}$

Let $\delta = ||p-q||$ be the hamming distance. Fix p,q:

$$Pr_{h_1 \in H}[h_i(p) = h_i(q)] = \frac{\# \text{ that are close}}{\#\text{of total coordinates}} = \frac{d-\delta}{d} = 1 - \delta/d$$

Using Taylor Series expansion:

$$p_1 = 1 - r/d \approx e^{-r/d}$$
$$p_2 = 1 - cr/d \approx e^{-cr/d}$$

Thus:

$$\frac{\log 1/p_1}{\log 1/p_2} = \frac{r/d}{cr/d}$$
$$= 1/c$$

If we set $c = 2$, we get $\varrho = 1/2$. We achieve a data structure with the following properties:

- Space: $O(n^{1.5} + nd)$
- Query: $O(\sqrt{n} log(n) d)$