EE 381V: Large Scale Optimization **Fall 2012**

Lecture 2 — January 17

Lecturer: Caramanis & Sanghavi *Scribe: Dinesh Jayaraman*

2.1 Locally sensitive hashing: Overview

Last class, we began to discuss locality-sensitive hashing (LSH). The objective of LSH is to map high dimensional points into a lower dimensional space in such a way that points that are close to each other (as measured by some specified distance measure) map to the same point ("collide") and those that are not, map to different points. Formally, a family of functions $\mathcal{H}$ is said to be a (r_1, r_2, p_1, p_2) locally sensitive family, if $\forall h : \mathbb{A} \to \mathbb{B} \in \mathcal{H}$ and $\forall$ points $p, q \in \mathbb{A}$, $(r_1 < r_2, p_1 > p_2)$

- if $\|p - q\| \leq r_1$, then $p_{\mathcal{H}}(h(p) = h(q)) \geq p_1$
- if $\|p - q\| \geq r_2$, then $p_{\mathcal{H}}(h(p) = h(q)) \leq p_2$

2.2 Amplification of hashing functions

Ideally, we will have $p_1 = 1, p_2 = 0$. However, a family of LSH functions $\mathcal{H}$ as defined above (satisfying $p_1 > p_2$) can be combined using AND and OR operations to produce new functions that approach this ideal LSH function.

2.2.1 The AND operation

Defn 2.2.1. AND : $\mathcal{H} \to \mathcal{H}'$ - *Pick r functions $h_1, h_2, \ldots h_r$ without replacement from $\mathcal{H}$. Define h' by $h'(x) = h'(y) \Leftrightarrow h_i(x) = h_i(y)\ \forall\ 1 \leq i \leq r$. We denote the family of such h' functions $\mathcal{H}'$.*

By requiring that the h_is collide all at once, the AND operation reduces the chances of a collision in h'. If, for two points x and y, $p_{\mathcal{H}}(h(x) = h(y)) = p$, then $p_{\mathcal{H}'}(h'(x) = h'(y)) = p^r$. Furthermore, for probabilities $p_1 > p_2$, $p_1 \xrightarrow{AND} p_1' = {p_1}^r$, $p_2 \xrightarrow{AND} p_2' = {p_2}^r$

$$\left(\frac{p_1'}{p_2'}\right)^r = \left(\frac{p_1}{p_2}\right)^r >> \left(\frac{p_1}{p_2}\right) \tag{2.1}$$

Thus, we see that AND, while reducing the probability of a collision, amplifies the difference in probabilities of collisions between nearby and far points.

Example 2.2.2. *Recall the example of a family of LSH functions for the case of Hamming distances between binary vectors:* $\mathcal{H} = \{h_i : h_i(x) = x_i, i \text{ selected randomly from } [1 \dots n]\}$. *To produce a function* $h' \in \mathcal{H}'$ *in this case, we pick* r *coordinates (without replacement) from* $[1 \dots n]$ *and set* $h'(x) = h'(y)$ *if and only if all the* r *coordinates collide.*

2.2.2 The OR operation

Defn 2.2.3. AND : $\mathcal{H} \to \tilde{\mathcal{H}}$ - *Pick* b *functions* $h_1, h_2, \dots h_b$ *without replacement from* $\mathcal{H}$. *Define* $\tilde{h}$ *by* $\tilde{h}(x) = \tilde{h}(y) \Leftrightarrow h_i(x) = h_i(y)$ *for some* i. *We denote the family of such* $\tilde{h}$ *functions* $\tilde{\mathcal{H}}$.

The OR operation boosts the chances of a collision in $\tilde{h}$. If, for two points x and y, $p_{\mathcal{H}}(h(x) = h(y)) = p$, then $p_{\tilde{\mathcal{H}}}(\tilde{h}(x) = \tilde{h}(y)) = 1 - (1-p)^b$. In other words, if $p_{\mathcal{H}}(h(x) \neq h(y)) = 1 - p$, then $p_{\tilde{\mathcal{H}}}(\tilde{h}(x) \neq \tilde{h}(y)) = (1-p)^b$. For probabilities $p_1 > p_2$ as before, $\left(\frac{1-p_1}{1-p_2}\right) < 1$, $p_1 \xrightarrow{OR} \tilde{p_1} = 1 - (1-p_1)^b$, $p_2 \xrightarrow{OR} \tilde{p_2} = 1 - (1-p_2)^b$

$$\left(\frac{1-\tilde{p_1}}{1-\tilde{p_2}}\right) = \left(\frac{1-p_1}{1-p_2}\right)^b << \left(\frac{1-p_1}{1-p_2}\right) \tag{2.2}$$

Thus OR, while boosting the probability of a collision, also boosts the probability of collision more fo nearby points than for points farther away, and thus is also an amplifying operation for hash functions.

For our binary vector Hamming distance example, an OR involves simply picking b coordinates and setting $\tilde{h}(x) = \tilde{h}(y)$ if and only if there is at least one matching coordinate.

2.2.3 Concatenation of AND and OR

Concatenation of AND and OR can be used to combine hash functions to produce amplified hash functions that are near-ideal. The following equation shows a family of hash functions $\mathcal{H}$ with collision probabilities p transforms after AND and OR in sequence.

$$(\mathcal{H}, p) \xrightarrow{AND} (\mathcal{H}', p^r) \xrightarrow{OR} (\tilde{\mathcal{H}}', 1 - (1-p^r)^b) \tag{2.3}$$

As seen in the plots in Figure 2.1, AND and OR operations tend to take a family of hash functions $\mathcal{H}$ towards ideality asymptotically. For instance if p_1 and p_2 were 0.3 and 0.8 before amplification, then with $r = 10$ and $b = 100$, these probabilities go to 0 and 1 respectively. That is, after amplification, the hash functions in the family *always* collide for

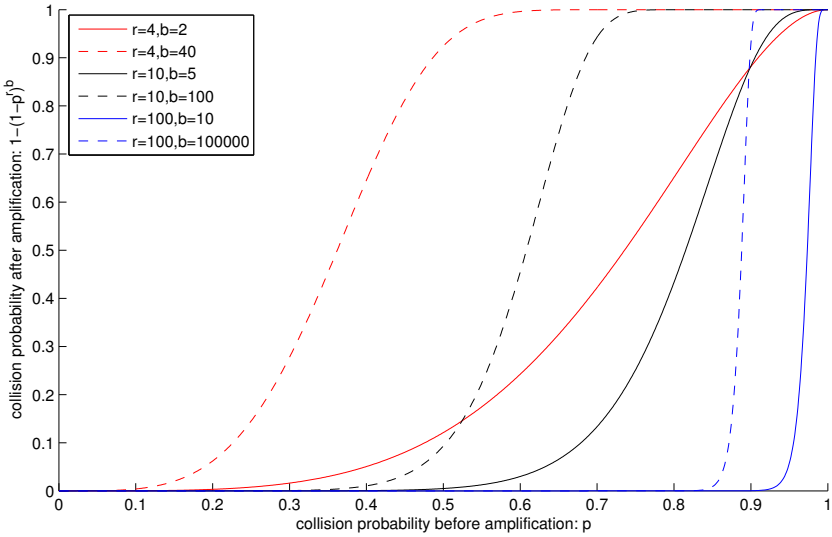

Figure 2.1. Evolution of collision probabilities with different amplification factors

distances below r_1, and *never* for distances above r_2. It is also possible to change the order of concatenation to achieve the same effect.

If $\mathcal{H}$ is bad to begin with (i.e it has low separation between p_1 and p_2), the convergence to ideality will be very slow. For example, say, $p_1 = 0.85$ and $p_2 = 0.9$. Then, from Figure 2.1, these probabilities get close to 0 and 1 respectively only around $r = 100$ and $b = 100000$. Combining such large numbers of hash functions can be computationally expensive(for each sample, a large number of elementary hash functions need to be computed), and is also limited by the size of the family $|\mathcal{H}|$.

2.2.4 Hash functions for the nearest neighbor problem

Defn 2.2.4. ***Optimization problem*** - *Given query point q and a set of points $\mathbb{A}$, find the nearest neighbor p to q in $\mathbb{A}$.*

Defn 2.2.5. ***Decision problem*** - *Given query point q, is there a point $p \in \mathbb{A}$.*

Defn 2.2.6. ***Randomized c-approx R-near-neighbor problem*** - *Given distance measure d on a space $\mathbb{X}$, for any point q, distance threshold R, and set $\mathbb{A} \in \mathbb{X}$, if $\exists\ p \in \mathbb{A}$ such that $d(q,p) \leq R$, return a point s s.t. $d(q,s) \leq cR$ $(c > 1)$ with probability $1 - \delta$.*

This problem, depicted in Figure 2.2, can be solved by using a family of hash functions as defined in section 2.1, that has $r_2 = cR$ and $r_1 = R$. Note that the problem as defined above says nothing about whether the algorithm should return a point when there exists no point in the set within the given distance threshold. False positives are allowed in this way

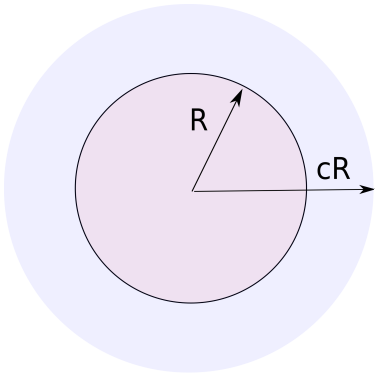

Figure 2.2. The c-approx nearest R-near-neighbor problem setup

because they can always be eliminated afterwards by computing the distances of all returned points from q.

Defn 2.2.7. ***Randomized R-near-neighbor problem*** - *Given distance measure d on a space $\mathbb{X}$, for any point q, distance threshold R, and set $\mathbb{A} \in \mathbb{X}$, if $\exists\ p \in \mathbb{A}$ such that $d(q,p) \leq R$, report a point s s.t. $d(q,s) \leq R$ with probability $1-\delta$.*

We cannot get better than $O(n^2)$ guarantee on such problems if we do not allow the probability tolerance δ. Together with the tolerance, these problems can be solved in sublinear time.

Example 2.2.8. *We have already discussed LSH functions for binary vectors and Hamming distances. Let us now look at another family of LSH functions, again for binary vectors, but this time with the Jaccard distance defined below being our choice of the distance measure.*

Defn 2.2.9. ***Jaccard distance*** - *For any binary vector u, define $S_u = \{i : u_i = 1\}$. Now for binary vectors u and v, define:*

$$d_J(u,v) = 1 - \frac{|S_1 \cup S_2|}{|S_1 \cap S_2|} \tag{2.4}$$

A common use of the Jaccard distance is in recommender systems. For eg., suppose that a supermarket wants to identify customers similar to some query customer. Each customer a may be represented as a binary vector v_a of n elements, where n is the number of items and the ith coordinate of v_a is set if and only if a has purchased that item. The Jaccard distance between v_a and v_b provides a useful notion of similarity between a and b in such contexts.

To define an LSH function for this case, select a random permutation π of the n coordinates in the binary vector space. Define hash functions $h(v) \equiv h(S(v)) = \arg\min_{i \in S(v)} \pi(i)$, where $S(v) = \{i : v_i = 1\}$

We will see how this function satisfies the LSH conditions in a later class.

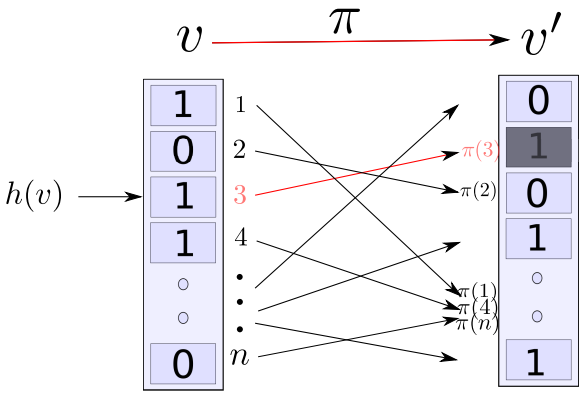

Figure 2.3. A schematic of the c-approx nearest R-near-neighbor problem definition