CS369G: Algorithmic Techniques for Big Data Spring 2015-2016

Lecture 17: Locality Sensitive Hashing and Dimension Reduction

Prof. Moses Charikar *Scribes: Joseph Shayani*

1 Overview

In this lecture, we give review the setup Locality Sensitive Hashing (LSH) and its connection to Nearest Neighbor Search (NNS) and give examples of LSH schemes for NNS with ℓ_2 metrics. We move on to discuss the time/space tradeoff in choice of LSH scheme and the connection between LSH and measures of similarity. We discuss how domain-specific knowledge can make LSH more effective in practice, and we mention one industry application of NNS that uses a heuristic alternative to LSH. In addition, we build on our earlier discussion of Dimension Reduction by presenting a stronger, distributional formulation of the lemma of Johnson and Lindenstrauss [JL84], and then we give examples of schemes that perform dimension reduction quickly.

2 Locality Sensitive Hashing

2.1 Definition of LSH and use for Approximate NNS

First we recall the definition of a family of (r, cr, p_1, p_2)-Locality Sensitive Hash functions.

Definition 1. *Let (X, d) be a metric space, and let Y be any set. Then family of hash function H with each $h \in H$ mapping $X \to Y$ is a (r, cr, p_1, p_2)-Locality Sensitive Hash family if for any $x_1, x_2 \in X$,*

$$P(h(x_1) = h(x_2)) \geq p_1 \text{ whenever } d(x_1, x_2) \leq r,$$
$$P(h(x_1) = h(x_2)) \leq p_2 \text{ whenever } d(x_1, x_2) \geq cr;$$

where the probability is over uniform random choice of h from H. [1]

Given a set of$n$ points $S \subset X$, we saw in the previous lecture that if we write $p_1 = p_2^\rho$, where $\rho = \rho(c, d)$, then given a LHS family, we can support approximate nearest neighbor search (NNS) queries in time $O(n^\rho)$, by using n^ρ hash tables for total space $O(n^{1+\rho})$).

Approximate nearest neighbor search requires that we return a neighbor within distance cr of the query point $q \in X$ if there is a point $x \in S$ with $d(x, q) = r$. Accordingly, we can view the relationship between ρ and c as a trade-off between space/time and accuracy.

We saw that if $X = \{0, 1\}^k$ and d is a Hamming distance, we can achieve $\rho = 1/c$. We can do as well for any ℓ_1 distance, and in fact we cannot do any better.

[1] The definition is only interesting for some conditions on the parameters (e.g. $p_1 < p_2$, $c > 1$).

2.2 LSH schemes for NNS with ℓ_2 distance

It is a fact that we can always map a metric space with an ℓ_2 distance to one with an ℓ_1 in an isometric (distance preserving) manner.[2] This fact guarantees that we can achieve $\rho = 1/c$, in the above sense, for LSH with ℓ_2 distance.

Several schemes do better than $\rho = 1/c$. The following are a few examples. X will be a vector space equipped with ℓ_2 distance d, i.e. X has an inner product and $d(x_1, x_2) = (x_1 - x_2) \cdot (x_1 - x_2)$.

- [DIIM04] Choose a random line ℓ ($\leftrightarrow$ unit vector), and split up the line into buckets of width w (this w will be the parameter we control). We can project each point $x \in X$ onto ℓ, and random noise, and identify x by the bucket along ℓ in which the resulting projection of the point lies. More precisely, for each unit vector ℓ, define $h_\ell(x) = \lfloor (x \cdot \ell + \alpha)/w \rfloor$, where $\alpha \in_R [0, w)$.
- [AI06] First map $\Pi : X \to \mathbb{R}^d$, and take a random sequence s of points $s_1, s_2, \cdots \in \mathbb{R}^d$. Fix a radius δ. Define $h_s(x) = \arg\min_{i>0} \Pi(x) \in B_\delta(s_i)$, i.e. the first ball contain $\Pi(x)$ in the sequence of balls of radius δ with center at the s_i. For a sufficiently large constant d (independent of the dimension of X), this scheme achieves $\rho \to 1/c^2$ as $n \to \infty$. This scheme is mostly of theoretical importance. As a more practical alternative, instead we can pick lattice points in $\mathbb{R}^d$ and identify x with the lattice point closest $\Pi(x)$ (ibid.).

The following scheme is useful in the case that we wish to hash the metric subspace of unit vectors. Let $X = S^n$.

- [AIL+15] First pick a random rotation R, then identify $x \in X$ with the index of the largest magnitude coordinate. This scheme, called the cross-polytope hash, achieves $\rho \approx 1/c^2$.

2.3 Space and query time tradeoff

We might have the alternative goal of finding a LSH scheme with low space. [Pan06] demonstrated such a scheme using $\tilde{O}(1)$ hash tables, i.e. $\tilde{O}(n)$ space. For ℓ_1 distance, the scheme requires query time of $O(n^{2.06/c})$. In Panigrahy's scheme, we investigate multiple buckets in each table for a single query. A similar idea was explored in [LJW+07] to reduce the space of the LSH data structure. In fact, Kapralov [Kap15] showed that there is a smooth tradeoff between storage space and query time.

2.4 Measures of similarity and Hash families

In the previous lecture, we mentioned that for the Jacard similarity between two multi-sets

$$\text{sim}(A, B) = \frac{|A \cap B|}{|A \cup B|},$$

[2] Moreover, if we allow distortion by a factor of $1 \pm \varepsilon$, then a scheme using random projections will be sufficient.

we can find a family of hash functions H with

$$P(h(A) = h(B)) = \text{sim}(A, B).$$

Question 2. *For which other measures of similarity does there exist a hash family H with $P(h(A) = h(B)) = sim(A, B)$?*

We can prove the following sufficient condition:

Proposition 3. *[Cha02] If there exists a hash family H with $P(h(A) = h(B)) = sim(A, B)$ for each $A, B \in X$, then $1 - sim(\cdot, \cdot)$ is a metric.*

Proof. We will prove that $1 - \text{sim}(A, B)$ satisfies the triangle inequality, i.e. that for each $A, B, C \in X$,

$$1 - \text{sim}(A, B) + 1 - \text{sim}(B, C) \geq 1 - \text{sim}(A, C).$$

By hypothesis, the condition is equivalent to

$$P(h(A) \neq h(B)) + P(h(B) \neq h(C)) \geq P(h(A) \neq h(C)).$$

It is sufficient to prove that for each $h \in H$

$$I_{h(A)\neq h(B)} + I_{h(B)\neq h(C)} \geq I_{h(A)\neq h(C)}, \tag{1}$$

where each I is an indicator function (then we average over all $h \in H$). (1) could be false only in if the left hand side is 0 and the right hand side is 1. If the left hand side is 0, it must be that $h(A) = h(B)$ and $h(B) = h(C)$, but then by transitivity $h(A) = h(C)$, so the right hand side is also 0. □

In fact, a stronger result can be proven:

Fact 4. *If there exists a hash family H with $P(h(A) = h(B))$ for each $A, B \in X$, then the metric space $(X, sim(\cdot, \cdot))$ is embeddable in a metric space with the generalized Hamming distance.*

Idea. Represent each $A \in X$ by $(h_1(A), h_2(A), \dots)$ for $h_1, h_2, \dots \in H$. More care must be taken if H is uncountable. □

See [CK12] for a further discussion of which measures of similarity correspond to hash families.

2.5 Data-aware versus data-oblivious hashing

There are two general philosophies for designing hash functions. We can either design hash functions with guarantees for any data set, or we can try to use extra information about particular data sets in order to design hash functions with better performance. LSH normally falls in the former category. In the setting of LSH, sometimes we can do better if we impose extra structure on our data. For example, if for each query q point there exists one nearest at distance r and all other points are distributed in a uniformly-random-like manner at the boundary of the ball of radius cr centered at q, then we can achieve $\rho = .5/c^2$. Moreover, it is possible to perform transformations on arbitrary point sets in a metric space in order to satisfy this condition [AR15]. See also [AINR14].

2.6 Application of NNS: PatchMatch

One remarkable industry application of NNS that uses a different (non-LSH) approach is Photoshop Intelligent Fill, which uses an algorithm called PatchMatch to find correspondences between small regions of images in order to support operations like removal of objects from photos. The algorithm constructs an approximate nearest neighbors graph and makes iterative improvements. Interested readers should see [BSFG09] and [DML11].

3 Dimension Reduction

Suppose we have n points in $\mathbb{R}^d$. We want to map the points to a smaller space $\mathbb{R}^k$ in a way such that distances between pairs of points are nearly preserved. We have seen the result due to Johnson and Lindenstrauss [JL84] that says this is possible to do such a map such that all distances are preserved within a factor of $1 \pm \varepsilon$ if we choose $k = O(\log(n)/\varepsilon^2)$. We reformulate the Johnson and Linderstrauss lemma in such a way that we eliminate consideration of the size n of a set of points.

More precisely, we seek a linear map $\Pi : \mathbb{R}^d \to \mathbb{R}^k$ such that for any unit vector $x \in \mathbb{R}^d$,

$$P(|||\Pi x||_2^2 - 1| > \varepsilon) < \delta.$$

Fact 5. *[JL84] Such a Π can be found for choice of $k = O(\log(1/\delta)/\epsilon^2)$.*

This choice of k is nearly tight: For a fixed choice of δ, we must have $k \geq \dfrac{1}{\varepsilon^2 \log(1/\varepsilon)}$. See Section 9 of [Alo03].

Proposition 6. *Fact 5 implies the Johnson-Lindenstrauss lemma.*

Proof. Given n points $x_1, \ldots, x_n \in \mathbb{R}^d$ and some fixed $\tilde{\varepsilon}, \tilde{\delta} > 0$, as in the set up of Johnson-Lindenstrauss, invoke Fact 5 with unit vectors $(x_i - x_j)/||x_i - x_j||$ for each $i \neq j \in [n]$. We get

$$||\Pi(x_i - x_j)||_2^2 \in [(1-\varepsilon)||x_i - x_j||_2^2, (1+\varepsilon)||x_i - x_j||_2^2] \tag{2}$$

for each pair i, j. We want (2) to hold $O(n^2)$ pairs i, j simultaneously with high probability. We set $\delta = 1/n^3$ in the statement of Fact 5 and apply the union bound so that we have failure probability at most $1/n \to 0 < \tilde{\delta}$. We can set $\varepsilon = \tilde{\varepsilon}$ because the interval only gets tighter when we take square roots in (2). Fact 5 requires that we take $k = O(\log(1/\delta)/\epsilon^2) = O(\log(n^3)/\epsilon^2) = O(\log(n)/\epsilon^2)$. This is exactly the Johnson-Lindenstrauss lemma. □

Question 7. *How quickly we can map $\mathbb{R}^d \to \mathbb{R}^k$?*

In the worst case, computing Πx takes time dk, where Π is $k \times d$ and $x \in \mathbb{R}^d$. We can do better. [Ach03] showed that if we can achieve Fact 5 with

$$\Pi_{ij} = \begin{cases} 1 & \text{with probability } 1/6 \\ -1 & \text{with probability } 1/6 \\ 0 & \text{with probability } 2/3 \end{cases}$$

Thus computing Πx takes time $dk/3$ on average. Recall that $k = O(\log(1/\delta)/\epsilon^2)$, so $dk = O(d\log(1/\delta)/\epsilon^2)$. Kane and Nelson [KN14] showed that we can do even better and save a factor of $1/\varepsilon$.

References

[Ach03] Dimitris Achlioptas. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *Journal of computer and System Sciences*, 66(4):671–687, 2003.

[AI06] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 459–468. IEEE, 2006.

[AIL+15] Alexandr Andoni, Piotr Indyk, Thijs Laarhoven, Ilya Razenshteyn, and Ludwig Schmidt. Practical and optimal lsh for angular distance. In *Advances in Neural Information Processing Systems*, pages 1225–1233, 2015.

[AINR14] Alexandr Andoni, Piotr Indyk, Huy L Nguyen, and Ilya Razenshteyn. Beyond locality-sensitive hashing. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1018–1028. SIAM, 2014.

[Alo03] Noga Alon. Problems and results in extremal combinatoricsi. *Discrete Mathematics*, 273(1):31–53, 2003.

[AR15] Alexandr Andoni and Ilya Razenshteyn. Optimal data-dependent hashing for approximate near neighbors. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, pages 793–801. ACM, 2015.

[BSFG09] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 28(3), August 2009.

[Cha02] Moses S Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388. ACM, 2002.

[CK12] Flavio Chierichetti and Ravi Kumar. Lsh-preserving functions and their applications. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 1078–1094. SIAM, 2012.

[DIIM04] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262. ACM, 2004.

[DML11] Wei Dong, Charikar Moses, and Kai Li. Efficient k-nearest neighbor graph construction for generic similarity measures. In *Proceedings of the 20th international conference on World wide web*, pages 577–586. ACM, 2011.

[JL84] W B Johnson and J Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary Mathematics*, 26:189–206, 1984.

[Kap15] Michael Kapralov. Smooth tradeoffs between insert and query complexity in nearest neighbor search. In *Proceedings of the 34th ACM Symposium on Principles of Database Systems*, pages 329–342. ACM, 2015.

[KN14] Daniel M Kane and Jelani Nelson. Sparser johnson-lindenstrauss transforms. *Journal of the ACM (JACM)*, 61(1):4, 2014.

[LJW+07] Qin Lv, William Josephson, Zhe Wang, Moses Charikar, and Kai Li. Multi-probe lsh: efficient indexing for high-dimensional similarity search. In *Proceedings of the 33rd international conference on Very large data bases*, pages 950–961. VLDB Endowment, 2007.

[Pan06] Rina Panigrahy. Entropy based nearest neighbor search in high dimensions. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 1186–1195. Society for Industrial and Applied Mathematics, 2006.