Hashing Hyperplane Queries to Near Points with Applications to Large-Scale Active Learning Supplementary Material

Prateek Jain
Algorithms Research Group
Microsoft Research, Bangalore, India
prajain@microsoft.com

Sudheendra Vijayanarasimhan
Department of Computer Science
University of Texas at Austin
svnaras@cs.utexas.edu

Kristen Grauman
Department of Computer Science
University of Texas at Austin
grauman@cs.utexas.edu

Appendix A: Proof of LSH for Hyperplane Hashing

We first recall the data-structure used for LSH. We store l-hash tables and every hash table contains k-bit hash keys. So, the s-th hash table has a corresponding function $g_s : \mathbb{R}^d \rightarrow 0, 1^k$ that given a vector, maps the vector to k-bit hash keys. Each function g_s is obtained by randomly sampling $\mathcal{H}$ with replacement: $g_s = (h_{s_1}, h_{s_2}, \ldots, h_{s_k})$.

Here, we show that using locality-sensitive hash functions for the distance $d_\theta(\cdot, \cdot)$ along with hash tables, we can get a $(1 + \epsilon)$-approximate solution to our hyperplane-to-point search problem in sub-linear time.

In particular, we prove the following theorem:

Theorem 0.1. *Let $\mathcal{H}$ be a family of $(r, r(1+\epsilon), p_1, p_2)$-locality hash functions (see Definition 3.1 (*Main Text*)), with $p_1 > p_2$. Now given a database of N points, we set $k = \log_{1/p_2} N$ and $l = N^\rho$, where $\rho = \frac{\log p_1}{\log p_2}$. Now using $\mathcal{H}$ along with l-hash tables over k-bits, given a hyperplane query $\boldsymbol{w}$, with probability at least $\frac{1}{2} - \frac{1}{e}$, the algorithm solves the (r, ϵ)-neighbor problem, i.e., if there exists a point $\boldsymbol{x}$ s.t. $d_\theta(\boldsymbol{x}, \boldsymbol{w}) \leq (1+\epsilon)r$, then the algorithm will return the point with probability $\geq 1/2 - 1/e$. The retrieval time is bounded by $O(N^\rho)$.*

Proof. Our proof is a simple adaption of the proof of Theorem 1, Gionis et al. [1]. We present it here for the sake of completeness.

Following [1] we prove two properties:
P1: Let $\boldsymbol{x}^*$ be a point such that $d_\theta(\boldsymbol{x}^*, \boldsymbol{w}) \leq r$, then $g_j(\boldsymbol{x}^*) = g_j(\boldsymbol{w})$ for some $1 \leq j \leq l$ with probability $1/2 - 1/e$.
Proof: Now we know that

$$\Pr[g_j(\boldsymbol{x}^*) = g_j(\boldsymbol{w})] \geq p_1^k = p_1^{\log_{1/p_2} N} = N^{-\rho}.$$

Hence,

$$\Pr[g_j(\boldsymbol{x}^*) \neq g_j(\boldsymbol{w}), \forall j] = \Pi_j \Pr[g_j(\boldsymbol{x}^*) \neq g_j(\boldsymbol{w})] \leq (1 - N^{-\rho})^l = (1 - N^{-\rho})^{N^\rho} \leq 1/e.$$

Thus, P1 holds with probability $> 1 - 1/e$.

P2: Consider the set $S = \{\boldsymbol{y} \text{ s.t., } d_\theta(\boldsymbol{y}, \boldsymbol{w}) > r(1+\epsilon) \text{ and } g_j(\boldsymbol{y}) = g_j(\boldsymbol{w}) \text{ for some } j\}$. Then $|S| \leq cl$ with probability at least $1 - 1/c$.

Proof: Now if $d_\theta(\boldsymbol{y}, \boldsymbol{w}) > r(1+\epsilon)$, then $\Pr[h(\boldsymbol{y}) = h(\boldsymbol{w})] \leq p_2$. Hence, for any j,

$$\Pr[g_j(\boldsymbol{y}) = g_j(\boldsymbol{w})] \leq p_2^k = p_2^{\log_{1/p_2} n} = 1/N.$$

Thus the expected number of collisions for a single j is $N \cdot \Pr[g_j(\boldsymbol{y}) = g_j(\boldsymbol{w})] = 1$ and hence $E[|S|] = l$. Therefore, by Markov's inequality:

$$\Pr(|S| > cl) \leq 1/c.$$

Hence, P2 holds with probability $> 1 - 1/c$.

The theorem now immediately follows from P1 and P2, as by P1 we are assured of retrieving the point $\boldsymbol{x}^*$ with probability $> 1/2 - 1/e$, and by P2 we are assured of not looking at more than $cl = O(N^\rho)$ points. □

Appendix B: Comparison of approximation guarantees

In this section we compare the bounds on retrieval for both of our hashing methods. To recall, our H-Hash method guarantees the $(1+\epsilon)$-approximate solution in time N^ρ, where $\rho \leq \frac{1-\log(1-\frac{4r}{\pi^2})}{1+\frac{\epsilon}{1+\frac{\pi^2}{4r}\log 4}}$.

Similarly, our EH-Hash method guarantees the $(1+\epsilon)$-approximate solution in time N^ρ, where $\rho \leq \frac{\log\cos^{-1}\sin^2(\sqrt{r})-\log\pi}{\log\cos^{-1}\sin^2(\sqrt{r}(1+\epsilon/2))-\log\pi}$. Note that the function $\cos^{-1}\sin^2(\sqrt{r})$ behaves similarly to $\frac{1}{2} - \frac{2r}{\pi^2}$, which is twice the probability of collision for our H-Hash method when the points are within distance r (see Figure 1). This indicates that the bounds for our EH-Hash method should be significantly stronger than the corresponding bounds for our H-Hash method.

Figure 2 compares the values of ρ obtained by our two methods for different values of ϵ. We can clearly see that for our EH-Hash method the value of ρ is always smaller than the corresponding value for H-Hash method. Now, we give a concrete example. Let $\epsilon = 3.5$. Then it can be easily computed that if the closest point to the hyperplane is at angle of around 5^o, then H-Hash will return a point within 9^o in time $N^{0.97}$ while the corresponding bound for EH-Hash method will be $N^{0.89}$, a significant gain.

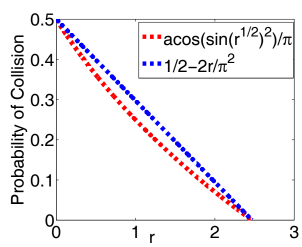

Figure 1: Comparison of the probability of collision p_1 for our EH-Hash method with the function $f(r) = \frac{1}{2} - \frac{2r}{\pi^2}$

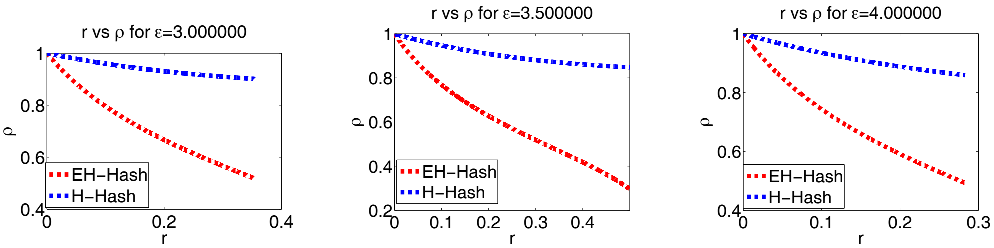

Figure 2: Comparision of the values of ρ for our H-Hash and EH-Hash methods with different values of $\epsilon = \{3.0, 3.5, 4.0\}$

Appendix C: Randomized Sampling

Proof of Lemma 3.4. Let i_k denote the randomly sampled index (using probability distribution p defined in the lemma) at the k-th round, i.e., i_k is index j with probability p_j. Next, we define a random variable G_k as,

$$G_k = v_{i_k} y_{i_k} / p_{i_k}.$$

Note that,

$$E[G_k] = \sum_j p_j v_j y_j / p_j = \boldsymbol{v}^T \boldsymbol{y}, \tag{1}$$

$$Var(G_k) = \sum_j p_j (v_j y_j / p_j)^2 - (\boldsymbol{v}^T \boldsymbol{y})^2 \leq \frac{v_j^2 t_j^2}{t_j^2 / \|\boldsymbol{y}\|^2} = \|\boldsymbol{v}\|^2 \|\boldsymbol{y}\|^2 = 1. \tag{2}$$

(3)

Now, our final approximation for $\boldsymbol{v}^T \boldsymbol{y}$ is obtained by averaging random variables G_k, i.e.,

$$\tilde{\boldsymbol{v}}^T \boldsymbol{x} = \frac{1}{t} \sum_k G_k.$$

Now, using Bernstein's inequality:

$$\Pr(|\sum_{k=1}^{t} (G_k - \boldsymbol{v}^T \boldsymbol{y})| \geq t\epsilon) < \exp(-t\epsilon^2).$$

Hence, if we select $t = \frac{c}{\epsilon^2}$, then with probability at least $1 - \log(1/c)$,

$$|\tilde{\boldsymbol{v}}^T \boldsymbol{y} - \boldsymbol{v}^T \boldsymbol{y}| \leq \epsilon.$$

□

References

[1] A. Gionis, P. Indyk, and R. Motwani. Similarity Search in High Dimensions via Hashing. In *Proceedings of the 25th Intl Conf. on Very Large Data Bases*, 1999.