On Symmetric and Asymmetric LSHs for Inner Product Search

Behnam Neyshabur BNEYSHABUR@TTIC.EDU
Nathan Srebro NATI@TTIC.EDU
Toyota Technological Institute at Chicago, Chicago, IL 60637, USA

Abstract

We consider the problem of designing locality sensitive hashes (LSH) for inner product similarity, and of the power of asymmetric hashes in this context. Shrivastava and Li (2014a) argue that there is no *symmetric* LSH for the problem and propose an *asymmetric* LSH based on different mappings for query and database points. However, we show there *does* exist a simple *symmetric* LSH that enjoys stronger guarantees and better empirical performance than the asymmetric LSH they suggest. We also show a variant of the settings where asymmetry is in-fact needed, but there a different asymmetric LSH is required.

1. Introduction

Following Shrivastava and Li (2014a), we consider the problem of Maximum Inner Product Search (MIPS): given a collection of "database" vectors $\mathcal{S} \subset \mathbb{R}^d$ and a query $q \in \mathbb{R}^d$, find a data vector maximizing the inner product with the query:

$$p = \arg\max_{x \in \mathcal{S}} \quad q^\top x \tag{1}$$

MIPS problems of the form (1) arise, e.g. when using matrix-factorization based recommendation systems (Koren et al., 2009; Srebro et al., 2005; Cremonesi et al., 2010), in multi-class prediction (Dean et al., 2013; Jain et al., 2009) and structural SVM (Joachims, 2006; Joachims et al., 2009) problems and in vision problems when scoring filters based on their activations (Dean et al., 2013) (see Shrivastava and Li, 2014a, for more about MIPS). In order to efficiently find approximate MIPS solutions, Shrivastava and Li (2014a) suggest constructing a Locality Sensitive Hash (LSH) for inner product "similarity".

Locality Sensitive Hashing (Indyk and Motwani, 1998) is a popular tool for approximate nearest neighbor search and is also widely used in other settings (Gionis et al., 1999; Datar et al., 2004; Charikar, 2002). An LSH is a random mapping $h(\cdot)$ from objects to a small, possibly binary, alphabet, where collision probabilities $\mathbb{P}[h(x) = h(y)]$ relate to the desired notion of similarity $\text{sim}(x, y)$. An LSH can in turn be used to generate short hash words such that hamming distances between hash words correspond to similarity between objects. Recent studies have also explored the power of asymmetry in LSH and binary hashing, where two different mappings $f(\cdot), g(\cdot)$ are used to approximate similarity, $\text{sim}(x, y) \approx \mathbb{P}[h(x) = g(y)]$ (Neyshabur et al., 2013; 2014). Neyshabur et al. showed that even when the similarity $\text{sim}(x, y)$ is entirely symmetric, asymmetry in the hash may enable obtaining an LSH when a symmetric LSH is not possible, or enable obtaining a much better LSH yielding shorter and more accurate hashes.

Several tree-based methods have also been proposed for inner product search (Ram and Gray, 2012; Koenigstein et al., 2012; Curtin et al., 2013). Shrivastava and Li (2014a) argue that tree-based methods, such as cone trees, are impractical in high dimensions while the performance of LSH-based methods is in a way independent of dimension of the data. Although the exact regimes under which LSH-based methods are superior to tree-based methods and vice versa are not fully established yet, the goal of this paper is to analyze different LSH methods and compare them with each other, rather than comparing to tree-based methods, so as to understand which LSH to use and why, in those regimes where tree-based methods are not practical.

Considering MIPS, Shrivastava and Li (2014a) argue that there is no symmetric LSH for inner product similarity, and propose two distinct mappings, one of database objects and the other for queries, which yields an asymmetric LSH for MIPS. But the caveat is that they consider different spaces in their positive and negative results: they show nonexistence of a symmetric LSH over the entire space $\mathbb{R}^d$, but their asymmetric LSH is only valid when queries are normalized and data vectors are bounded. Thus, they do *not* actually show a situation where an asymmetric hash succeeds where a symmetric hash is not possible. In fact, in Section 4 we show a simple *symmetric* LSH that is also

Proceedings of the 32^{nd} International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.

valid under the same assumptions, and it even enjoys improved theoretical guarantees and empirical performance! This suggests that asymmetry might actually not be required nor helpful for MIPS.

Motivated by understanding the power of asymmetry, and using this understanding to obtain the simplest and best possible LSH for MIPS, we conduct a more careful study of LSH for inner product similarity. A crucial issue here is what is the space of vectors over which we would like our LSH to be valid. First, we show that over the entire space $\mathbb{R}^d$, not only is there no symmetric LSH, but there is also no asymmetric LSH either (Section 3). Second, as mentioned above, when queries are normalized and data is bounded, a symmetric LSH is possible and there is no need for asymmetry. But when queries and data vectors are bounded and queries are not normalized, we do observe the power of asymmetry: here, a symmetric LSH is not possible, but an asymmetric LSH exists (Section 5).

As mentioned above, our study also yields an LSH for MIPS, which we refer to as SIMPLE-LSH, which is not only symmetric but also parameter-free and enjoys significantly better theoretical and empirical compared to L2-ALSH(SL) proposed by Shrivastava and Li (2014a). In the supplementary material we show that all of our theoretical observations about L2-ALSH(SL) apply also to the alternative hash SIGN-LSH(SL) put forth by Shrivastava and Li (2014b).

The transformation at the root of SIMPLE-LSH was also recently proposed by Bachrach et al. (2014), who used it in a PCA-Tree data structure for speeding up the Xbox recommender system. Here, we study the transformation as part of an LSH scheme, investigate its theoretical properties, and compare it to LS-ALSH(SL).

2. Locality Sensitive Hashing

A **hash** of a set $\mathcal{Z}$ of objects is a random mapping from $\mathcal{Z}$ to some alphabet Γ, i.e. a distribution over functions $h : \mathcal{Z} \to \Gamma$. The hash is sometimes thought of as a "family" of functions, where the distribution over the family is implicit.

When studying hashes, we usually study the behavior when comparing any two points $x, y \in \mathcal{Z}$. However, for our study here, it will be important for us to make different assumptions about x and y—e.g., we will want to assume w.l.o.g. that queries are normalized but will not be able to make the same assumptions on database vectors. To this end, we define what it means for a hash to be an LSH over a pair of constrained subspaces $\mathcal{X}, \mathcal{Y} \subseteq \mathcal{Z}$. Given a similarity function $\text{sim} : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}$, such as inner product similarity $\text{sim}(x, y) = x^\top y$, an LSH is defined as follows[1]:

Definition 1 (Locality Sensitive Hashing (LSH))**.** *A hash is said to be a* (S, cS, p_1, p_2)**-LSH** *for a similarity function sim* **over the pair of spaces** $\mathcal{X}, \mathcal{Y} \subseteq \mathcal{Z}$ *if for any* $x \in \mathcal{X}$ *and* $y \in \mathcal{Y}$*:*

- *if* $\text{sim}(x, y) \geq S$ *then* $\mathbb{P}_h[h(x) = h(y)] \geq p_1$,
- *if* $\text{sim}(x, y) \leq cS$ *then* $\mathbb{P}_h[h(x) = h(y)] \leq p_2$.

When $\mathcal{X} = \mathcal{Y}$*, we say simply "***over the space** $\mathcal{X}$*".*

Here $S > 0$ is a threshold of interest, and for efficient approximate nearest neighbor search, we need $p_1 > p_2$ and $c < 1$. In particular, given an $(S, cS, p1, p2)$-LSH, a data structure for finding S-similar objects for query points when cS-similar objects exist in the database can be constructed in time $O(n^\rho \log n)$ and space $O(n^{1+\rho})$ where $\rho = \frac{\log p_1}{\log p_2}$. This quantity ρ is therefore of particular interest, as we are interested in an LSH with minimum possible ρ, and we refer to it as the *hashing quality*.

In Definition 1, the hash itself is still symmetric, i.e. the same function h is applied to both x and y. The only asymmetry allowed is in the problem definition, as we allow requiring the property for differently constrained x and y. This should be contrasted with a truly asymmetric hash, where two different functions are used, one for each space. Formally, an **asymmetric hash** for a pair of spaces $\mathcal{X}$ and $\mathcal{Y}$ is a joint distribution over pairs of mappings (f, g), $f : \mathcal{X} \to \Gamma$, $g : \mathcal{Y} \to \Gamma$. The asymmetric hashes we consider will be specified by a pair of deterministic mappings $P : \mathcal{X} \to \mathcal{Z}$ and $Q : \mathcal{Y} \to \mathcal{Z}$ and a single random mapping (i.e. distribution over functions) $h : \mathcal{Z} \to \Gamma$, where $f(x) = h(P(x))$ and $g(y) = h(Q(y))$. Given a similarity function $\text{sim} : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ we define:

Definition 2 (Asymmetric Locality Sensitive Hashing (ALSH))**.** *An asymmetric hash is said to be an* (S, cS, p_1, p_2)**-ALSH** *for a similarity function sim* **over** $\mathcal{X}, \mathcal{Y}$ *if for any* $x \in \mathcal{X}$ *and* $y \in \mathcal{Y}$*:*

- *if* $\text{sim}(x, y) \geq S$ *then* $\mathbb{P}_{(f,g)}[f(x) = g(y)] \geq p_1$,
- *if* $\text{sim}(x, y) \leq cS$ *then* $\mathbb{P}_{(f,g)}[f(x) = g(y)] \leq p_2$.

Referring to either of the above definitions, we also say that a hash is an (S, cS)**-LSH** (or **ALSH**) if there exists $p_2 > p_1$ such that it is an (S, cS, p_1, p_2)-LSH (or ALSH). And we say it is a **universal LSH** (or **ALSH**) if for every $S > 0, 0 < c < 1$ it is an (S, cS)-LSH (or ALSH).

3. No ALSH over $\mathbb{R}^d$

Considering the problem of finding an LSH for inner product similarity, Shrivastava and Li (2014a) first observe that for any $S > 0, 0 < c < 1$, there is no *symmetric* (S, cS)-

[1] This is a formalization of the definition given by Shrivastava and Li (2014a), which in turn is a modification of the definition of LSH for distance functions (Indyk and Motwani, 1998), where we also allow different constraints on x and y. Even though inner product similarity could be negative, this definition is only concerned with the positive values.

LSH for $\mathrm{sim}(x,y) = x^\top y$ over the entire space $\mathcal{X} = \mathbb{R}^d$, which prompted them to consider asymmetric hashes. In fact, we show that asymmetry doesn't help here, as there also isn't any ALSH over the entire space:

Theorem 3.1. *For any $d \geq 2$, $S > 0$ and $0 < c < 1$ there is no asymmetric hash that is an (S, cS)-ALSH for inner product similarity over $\mathcal{X} = \mathcal{Y} = \mathbb{R}^d$.*

Proof. Assume for contradiction there exists some $S > 0, 0 < c < 1$ and $p_1 > p_2$ for which there exists an (S, cS, p_1, p_2)-ALSH (f, g) for inner product similarity over $\mathbb{R}^2$ (an ALSH for inner products over $\mathbb{R}^d$, $d > 2$, is also an ALSH for inner products over a two-dimensional subspace, i.e. over $\mathbb{R}^2$, and so it is enough to consider $\mathbb{R}^2$). Consider the following two sequences of points:

$$x_i = [-i, 1]$$
$$y_j = [S(1-c), S(1-c)j + S].$$

For any N (to be set later), define the $N \times N$ matrix Z as follows:

$$Z(i,j) = \begin{cases} 1 & x_i^\top y_j \geq S \\ -1 & x_i^\top y_j \leq cS \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Because of the choice of x_i and y_j, the matrix Z does not actually contain zeros, and is in-fact triangular with $+1$ on and above the diagonal and -1 below it. Consider also the matrix $P \in \mathbb{R}^{N \times N}$ of collision probabilities $P(i,j) = \mathbb{P}_{(f,g)}[f(x_i) = g(x_j)]$. Setting $\theta = (p_1 + p_2)/2 < 1$ and $\epsilon = (p_1 - p_2)/2 > 0$, the ALSH property implies that for every i, j:

$$Z(i,j)(P(i,j) - \theta) \geq \epsilon \tag{3}$$

or equivalently:

$$Z \odot \frac{P - \theta}{\epsilon} \geq 1 \tag{4}$$

where $\odot$ denotes element-wise (Hadamard) product. Now, for a sign matrix Z, the *margin complexity* of Z is defined as $mc(Z) = \inf_{Z \odot X \geq 1} \|X\|_{\max}$ (see Srebro and Shraibman, 2005, and also for the definition of the max-norm $\|X\|_{\max}$), and we know that the margin complexity of an $N \times N$ triangular matrix is bounded by $mc(Z) = \Omega(\log N)$ (Forster et al., 2003), implying

$$\|(P - \theta)/\epsilon\|_{\max} = \Omega(\log N). \tag{5}$$

Furthermore, any collision probability matrix has max-norm $\|P\|_{\max} \leq 1$ (Neyshabur et al., 2014), and shifting the matrix by $0 < \theta < 1$ changes the max-norm by at most θ, implying $\|P - \theta\|_{\max} \leq 2$, which combined with (5) implies $\epsilon = O(1/\log N)$. For any $\epsilon = p_1 - p_2 > 0$, selecting a large enough N we get a contradiction. □

For completeness, we also include in the supplementary material a full definition of the max-norm and margin complexity, as well as the bounds on the max-norm and margin complexity used in the proof above.

4. Maximum Inner Product Search

We saw that no LSH, nor ALSH, is possible for inner product similarity over the entire space $\mathbb{R}^d$. Fortunately, this is not required for MIPS. As pointed out by Shrivastava and Li (2014a), we can assume the following without loss of generality:

- The query q is normalized: Since given a vector q, the norm $\|q\|$ does not affect the argmax in (1), we can assume $\|q\| = 1$ always.
- The database vectors are bounded inside the unit sphere: We assume $\|x\| \leq 1$ for all $x \in \mathcal{S}$. Otherwise we can rescale all vectors without changing the argmax.

We cannot, of course, assume the vectors x are normalized. This means we can limit our attention to the behavior of the hash over $\mathcal{X}_\bullet = \left\{x \in \mathbb{R}^d \;\middle|\; \|x\| \leq 1\right\}$ and $\mathcal{Y}_\circ = \left\{q \in \mathbb{R}^d \;\middle|\; \|q\| = 1\right\}$. Indeed, Shrivastava and Li (2014a) establish the existence of an *asymmetric* LSH, which we refer to as L2-ALSH(SL), over this pair of database and query spaces. Our main result in this section is to show that in fact there does exists a simple, parameter-free, universal, *symmetric* LSH, which we refer to as SIMPLE-LSH, over $\mathcal{X}_\bullet, \mathcal{Y}_\circ$. We see then that we *do* need to consider the hashing property asymmetrically (with different assumptions for queries and database vectors), but the same hash function can be used for both the database and the queries and there is no need for two different hash functions or two different mappings $P(\cdot)$ and $Q(\cdot)$.

But first, we review L2-ALSH(SL) and note that it is not universal—it depends on three parameters and no setting of the parameters works for all thresholds S. We also compare our SIMPLE-LSH to L2-ALSH(SL) (and to the recently suggested SIGN-ALSH(SL)) both in terms of the hashing quality ρ and empirically of movie recommendation data sets.

4.1. L2-ALSH(SL)

For an integer parameter m, and real valued parameters $0 < U < 1$ and $r > 0$, consider the following pair of mappings:

$$\begin{aligned} P(x) &= [Ux; \|Ux\|^2; \|Ux\|^4; \ldots; \|Ux\|^{2^m}] \\ Q(y) &= [y; 1/2; 1/2; \ldots; 1/2], \end{aligned} \tag{6}$$

combined with the standard L_2 hash function

$$h^{L_2}_{a,b}(x) = \left\lfloor \frac{a^\top x + b}{r} \right\rfloor \tag{7}$$

where $a \sim \mathcal{N}(0, I)$ is a spherical multi-Gaussian random vector, $b \sim \mathcal{U}(0, r)$ is a uniformly distributed random variable on $[0, r]$. The alphabet Γ used is the integers, the intermediate space is $\mathcal{Z} = \mathbb{R}^{d+m}$ and the asymmetric hash L2-ALSH(SL), parameterized by m, U and r, is then given by

$$(f(x), g(q)) = (h^{L_2}_{a,b}(P(x)), h^{L_2}_{a,b}(Q(q))). \tag{8}$$

Shrivastava and Li (2014a) establish[2] that for any $0 < c < 1$ and $0 < S < 1$, there exists $0 < U < 1$, $r > 0$, $m \geq 1$, such that L2-ALSH(SL) is an (S, cS)-ALSH over $\mathcal{X}_\bullet, \mathcal{Y}_\circ$. They furthermore calculate the hashing quality ρ as a function of m, U and r, and numerically find the optimal ρ over a grid of possible values for m, U and r, for each choice of S, c.

Before moving on to presenting a symmetric hash for the problem, we note that L2-ALSH(SL) is not *universal* (as defined at the end of Section 2). That is, not only might the optimal m, U and r depend on S, c, but in fact there is no choice of the parameters m and U that yields an ALSH for all S, c, or even for all ratios c for some specific threshold S or for all thresholds S for some specific ratio c. This is unfortunate, since in MIPS problems, the relevant threshold S is the maximal inner product $\max_{x \in S} q^\top x$ (or the threshold inner product if we are interested in the "top-k" hits), which typically varies with the query. It is therefore desirable to have a single hash that works for all thresholds.

Lemma 1. *For any m, U, r, and for any $0 < S < 1$ and*

$$1 - \frac{U^{2^{m+1}-1}(1 - S^{2^{m+1}})}{2S} \leq c < 1,$$

L2-ALSH(SL) is not an (S, cS)-ALSH for inner product similarity over $\mathcal{X}_\bullet = \{x | \|x\| \leq 1\}$ and $\mathcal{Y}_\circ = \{q | \|q\| = 1\}$.

Proof. Assume for contradiction that it is an (S, cS)-ALSH. For any query point $q \in \mathcal{Y}_\circ$, let $x \in \mathcal{X}_\bullet$ be a vector s.t. $q^\top x = S$ and $\|x\|_2 = 1$ and let $y = cSq$, so that $q^\top y = cS$. We have that:

$$p_1 \leq \mathbb{P}\left[h^{L_2}_{a,b}(P(x)) = h^{L_2}_{a,b}(Q(q))\right] = \mathcal{F}_r(\|P(x) - Q(q)\|_2)$$
$$p_2 \geq \mathbb{P}\left[h^{L_2}_{a,b}(P(y)) = h^{L_2}_{a,b}(Q(q))\right] = \mathcal{F}_r(\|P(y) - Q(q)\|_2)$$

where $\mathcal{F}_r(\delta)$ is a monotonically decreasing function of δ (Datar et al., 2004). To get a contradiction it is therefor enough to show that $\|P(y) - Q(q)\|^2 \leq \|P(x) - Q(q)\|^2$. We have:

$$\begin{aligned}\|P(y) - Q(q)\|^2 &= 1 + \frac{m}{4} + \|y\|^{2^{m+1}} - 2q^\top y \\ &= 1 + \frac{m}{4} + (cSU)^{2^{m+1}} - 2cSU\end{aligned}$$

using $1 - \frac{U^{2^{m+1}-1}(1-S^{2^{m+1}})}{2S} \leq c < 1$:

$$\begin{aligned}&< 1 + \frac{m}{4} + (SU)^{2^{m+1}} - 2cSU \\ &\leq 1 + \frac{m}{4} + U^{2^{m+1}} - 2SU \\ &= \|P(x) - Q(q)\|^2\end{aligned}$$

□

Corollary 4.1. *For any U, m and r, L2-ALSH(SL) is not a universal ALSH for inner product similarity over $\mathcal{X}_\bullet = \{x | \|x\| \leq 1\}$ and $\mathcal{Y}_\circ = \{q | \|q\| = 1\}$. Furthermore, for any $c < 1$, and any choice of U, m, r there exists $0 < S < 1$ for which L2-ALSH(SL) is not an (S, cS)-ALSH over $\mathcal{X}_\bullet, \mathcal{Y}_\circ$, and for any $S < 1$ and any choice of U, m, r there exists $0 < c < 1$ for which L2-ALSH(SL) is not an (S, cS)-ALSH over $\mathcal{X}_\bullet, \mathcal{Y}_\circ$.*

In the supplemental material, we show a similar non-universality result also for SIGN-ALSH(SL).

4.2. SIMPLE-LSH

We propose here a simpler, parameter-free, symmetric LSH, which we call SIMPLE-LSH.

For $x \in \mathbb{R}^d$, $\|x\| \leq 1$, define $P(x) \in \mathbb{R}^{d+1}$ as follows:

$$P(x) = \left[x; \sqrt{1 - \|x\|_2^2}\right] \tag{9}$$

For any $x \in \mathcal{X}_\bullet$ we have $\|P(x)\| = 1$, and for any $q \in \mathcal{Y}_\circ$, since $\|q\| = 1$, we have:

$$P(q)^\top P(x) = \left[q; 0\right]^\top \left[x; \sqrt{1 - \|x\|_2^2}\right] = q^\top x \tag{10}$$

Now, to define the hash SIMPLE-LSH, take a spherical random vector $a \sim \mathcal{N}(0, I)$ and consider the following random mapping into the binary alphabet $\Gamma = \{\pm 1\}$:

$$h_a(x) = \text{sign}(a^\top P(x)). \tag{11}$$

Theorem 4.2. SIMPLE-LSH *given in (11) is a universal LSH over $\mathcal{X}_\bullet, \mathcal{Y}_\circ$. That is, for every $0 < S < 1$ and $0 < c < 1$, it is an (S, cS)-LSH over $\mathcal{X}_\bullet, \mathcal{Y}_\circ$. Furthermore, it has hashing quality:*

$$\rho = \frac{\log\left(1 - \frac{\cos^{-1}(S)}{\pi}\right)}{\log\left(1 - \frac{\cos^{-1}(cS)}{\pi}\right)}.$$

Proof. For any $x \in \mathcal{X}_\bullet$ and $q \in \mathcal{Y}_\circ$ we have (Goemans and Williamson, 1995):

$$\mathbb{P}[h_a(P(q)) = h_a(P(x))] = 1 - \frac{\cos^{-1}(q^\top x)}{\pi}. \tag{12}$$

Therefore:

[2] Shrivastava and Li (2014a) have the scaling by U as a separate step, and state their hash as an (S_0, cS_0)-ALSH over $\{\|x\| \leq U\}, \{\|q\| = 1\}$, where the threshold $S_0 = US$ is also scaled by U. This is equivalent to the presentation here which integrates the pre-scaling step, which also scales the threshold, into the hash.

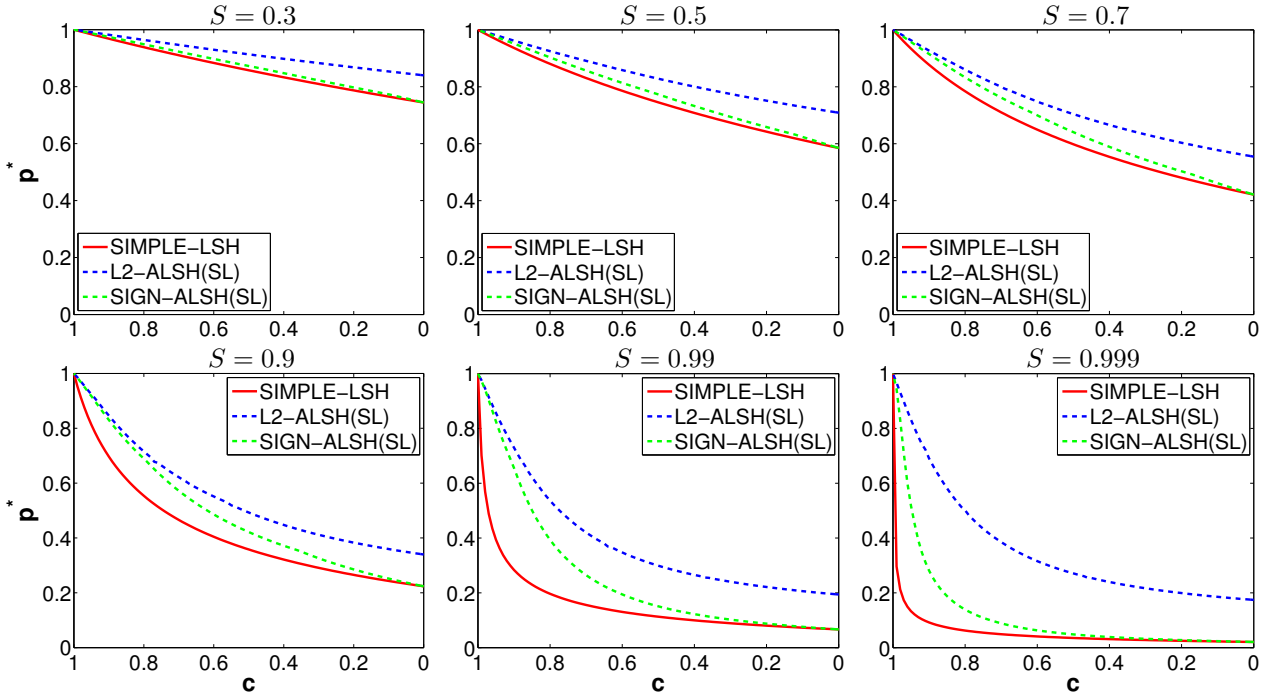

Figure 1. The optimal hashing quality ρ^* for different hashes (lower is better).

- if $q^\top x \geq S$, then

$$\mathbb{P}\big[h_a(P(q)) = h_a(P(x))\big] \geq 1 - \frac{\cos^{-1}(S)}{\pi}$$

- if $q^\top x \leq cS$, then

$$\mathbb{P}\big[h_a(P(q)) = h_a(P(x))\big] \leq 1 - \frac{\cos^{-1}(cS)}{\pi}$$

Since for any $0 \leq x \leq 1$, $1 - \frac{\cos^{-1}(x)}{\pi}$ is a monotonically increasing function, this gives us an LSH. □

4.3. Theoretical Comparison

Earlier we discussed that an LSH with the smallest possible hashing quality ρ is desirable. In this Section, we compare the best achievable hashing quality and show that SIMPLE-LSH allows for much better hashing quality compared to L2-ALSH(SL), as well as compared to the improved hash SIGN-LSH(SL).

For L2-ALSH(SL) and SIGN-ALSH(SL), for each desired threshold S and ratio c, one can optimize over the parameters m and U, and for L2-ALSH(SL) also r, to find the hash with the best ρ. This is a non-convex optimization problem and Shrivastava and Li (2014a) suggest using grid search to find a bound on the optimal ρ. We followed the procedure, and grid, as suggested by Shrivastava and Li (2014a)[3]. For SIMPLE-LSH no parameters need to be tuned, and for each S, c the hashing quality is given by Theorem 5.3. In Figure 1 we compare the optimal hashing quality ρ for the three methods, for different values of S and c. It is clear that the SIMPLE-LSH dominates the other methods.

4.4. Empirical Evaluation

We also compared the hash functions empirically, following the exact same protocol as Shrivastava and Li (2014a), using two collaborative filtering datasets, Netflix and Movielens 10M.

For a given user-item matrix Z, we followed the pureSVD procedure suggested by Cremonesi et al. (2010): we first subtracted the overall average rating from each individual rating and created the matrix Z with these average-subtracted ratings for observed entries and zeros for unobserved entries. We then take a rank-f approximation (top f singular components, $f = 150$ for Movielens and $f = 300$ for Netflix) $Z \approx W\Sigma R^\top = Y$ and define $L = W\Sigma$ so that $Y = LR^\top$. We can think of each row of L as the vector presentation of a user and each row of R as the presentation for an item.

The database S consists of all rows R_j of R (corresponding to movies) and we use each row L_i of L (corresponding to

[3]We actually used a slightly tighter bound—a careful analysis shows the denominator in equation 19 of Shrivastava and Li (2014a) can be $\log F_r(\sqrt{1 + m/2 - 2cSU + (cSU)^{2^{m+1}}})$

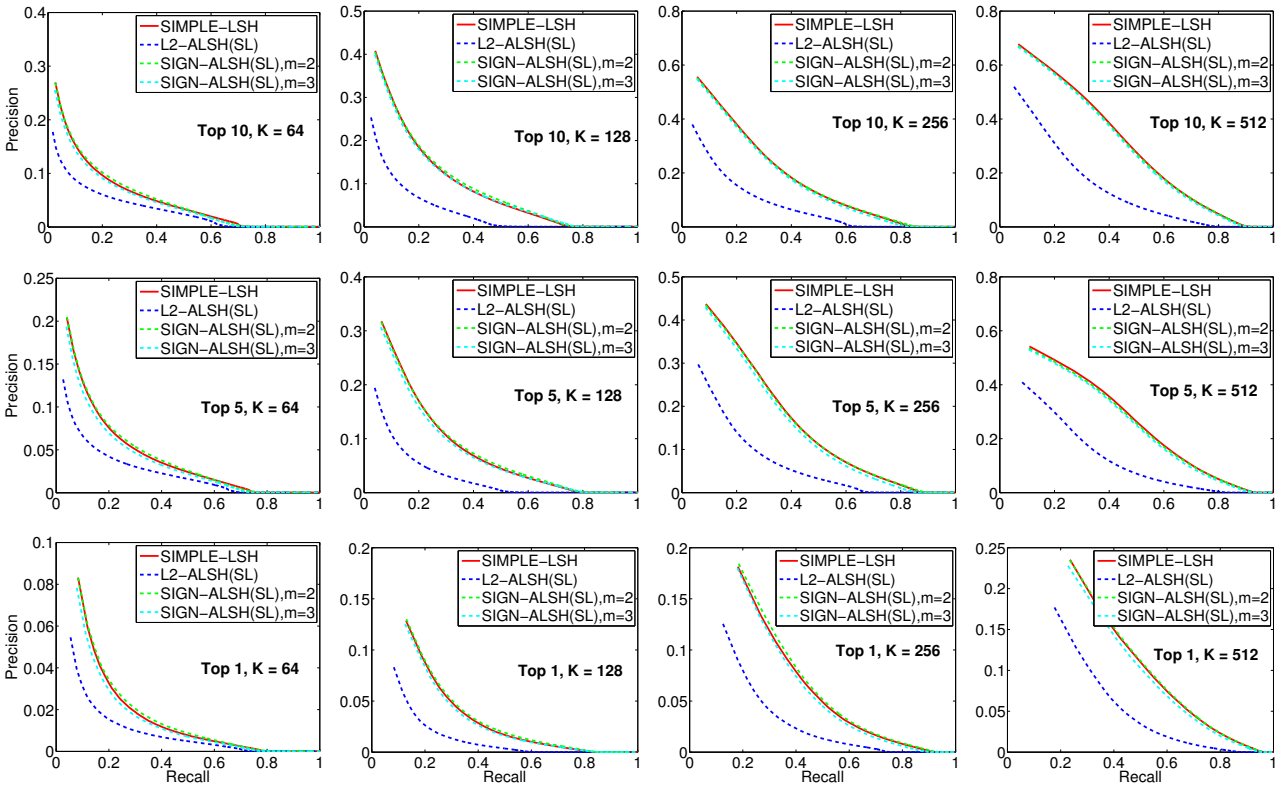

Figure 2. **Netflix**: Precision-Recall curves (higher is better) of retrieving top T items by hash code of length K. SIMPLE-LSH is parameter-free. For L2-ALSH(SL), we fix the parameters $m = 3$, $U = 0.84$, $r = 2.5$ and for SIGN-ALSH(SL) we used two different settings of the parameters: $m = 2, U = 0.75$ and $m = 3, U = 0.85$.

users) as a query. That is, for each user i we would like to find the top T movies, i.e. the T movies with highest $\langle L_i, R_j \rangle$, for different values of T.

To do so, for each hash family, we generate hash codes of length K, for varying lengths K, for all movies and a random selection of 60000 users (queries). For each user, we sort movies in ascending order of hamming distance between the user hash and movie hash, breaking up ties randomly. For each of several values of T and K we calculate precision-recall curves for recalling the top T movies, averaging the precision-recall values over the 60000 randomly selected users.

In Figures 2 and 3, we plot precision-recall curves of retrieving top T items by hash code of length K for Netflix and Movielens datasets where $T \in \{1, 5, 10\}$ and $K \in \{64, 128, 256, 512\}$. For L2-ALSH(SL) we used $m = 3, U = 0.83, r = 2.5$, suggested by the authors and used in their empirical evaluation. For SIGN-ALSH(SL) we used two different settings of the parameters suggested by Shrivastava and Li (2014b): $m = 2, U = 0.75$ and $m = 3, U = 0.85$. SIMPLE-LSH does not require any parameters.

As can be seen in the Figures, SIMPLE-LSH shows a dramatic empirical improvement over L2-ALSH(SL). Following the presentation of SIMPLE-LSH and the comparison with L2-ALSH(SL), Shrivastava and Li (2014b) suggested the modified hash SIGN-ALSH(SL), which is based on random projections, as is SIMPLE-LSH, but with an asymmetric transform similar to that in L2-ALSH(SL). Perhaps not surprising, SIGN-ALSH(SL) does indeed perform almost the same as SIMPLE-LSH (SIMPLE-LSH has only a slight advantage on Movielens), however: (1) SIMPLE-LSH is simpler, and uses a single symmetric lower-dimensional transformation $P(x)$; (2) SIMPLER-LSH is universal and parameter free, while SIGN-ALSH(SL) requires tuning two parameters (its authors suggest two different parameter settings for use). Therefor, we see no reason to prefer SIGN-ALSH(SL) over the simpler symmetric option.

5. Unnormalized Queries

In the previous Section, we exploited asymmetry in the MIPS problem formulation, and showed that with such asymmetry, there is no need for the hash itself to be asymmetric. In this Section, we consider LSH for inner product

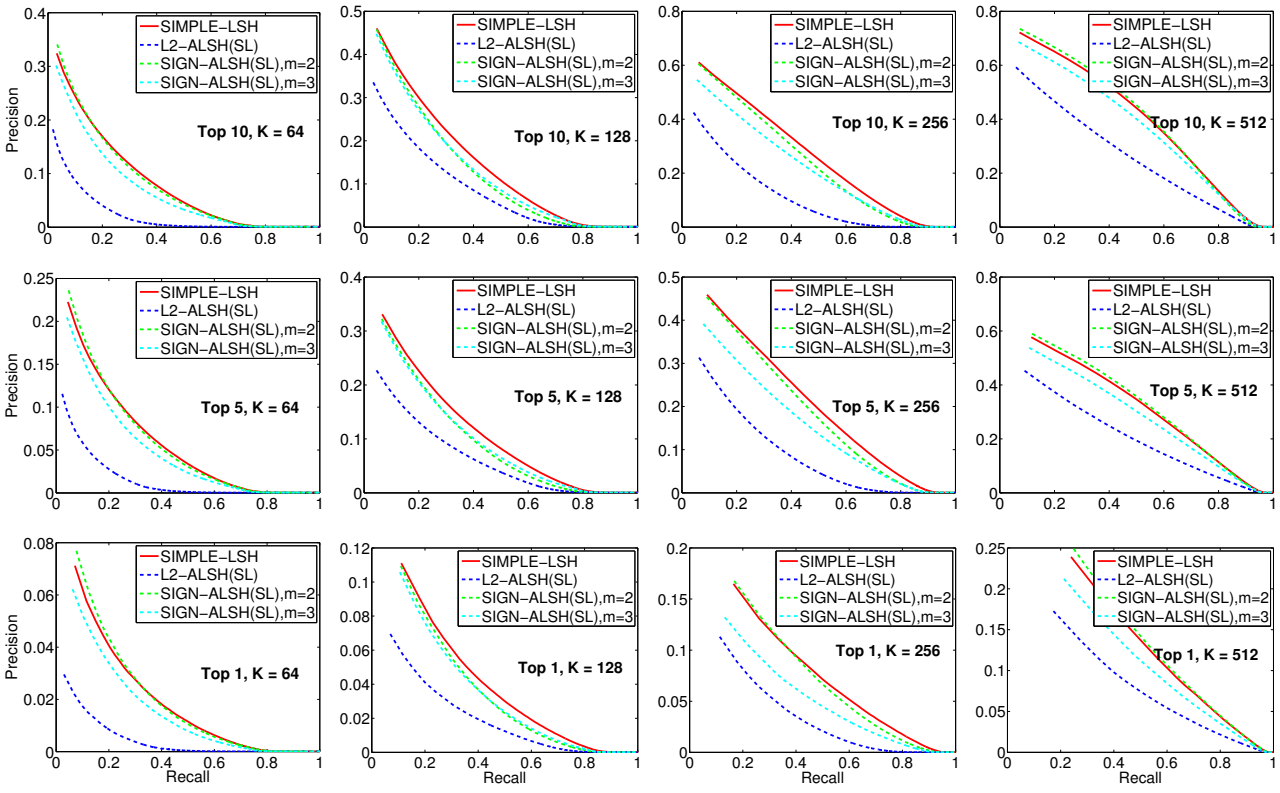

Figure 3. **Movielens**: Precision-Recall curves (higher is better) of retrieving top T items by hash code of length K. SIMPLE-LSH is parameter-free. For L2-ALSH(SL), we fix the parameters $m = 3, U = 0.84, r = 2.5$ and for SIGN-ALSH(SL) we used two different settings of the parameters: $m = 2, U = 0.75$ and $m = 3, U = 0.85$.

similarity in a more symmetric setting, where we assume no normalization and only boundedness. That is, we ask whether there is an LSH or ALSH for inner product similarity over $\mathcal{X}_\bullet = \mathcal{Y}_\bullet = \{x \mid \|x\| \leq 1\}$. Beyond a theoretical interest in the need for asymmetry in this fully symmetric setting, the setting can also be useful if we are interested in using sets $\mathcal{X}$ and $\mathcal{Y}$ interchangeably as query and data sets. In user-item setting for example, one might be also interested in retrieving the top users interested in a given item without the need to create a separate hash for this task.

We first observe that there is no symmetric LSH for this setting. We therefore consider asymmetric hashes. Unfortunately, we show that neither L2-ALSH(SL) (nor SIGN-ALSH(SL)) are ALSH over $\mathcal{X}_\bullet$. Instead, we propose a parameter-free asymmetric extension of SIMPLE-LSH, which we call SIMPLE-ALSH, and show that it is a universal ALSH for inner product similarity over $\mathcal{X}_\bullet$.

To summarize the situation, if we consider the problem *asymmetrically*, as in the previous Section, there is no need for the hash to be asymmetric, and we can use a single hash function. But if we insist on considering the problem *symmetrically*, we do indeed have to use an *asymmetric* hash.

5.1. No symmetric LSH

We first show we do not have a symmetric LSH:

Theorem 5.1. *For any $0 < S \leq 1$ and $0 < c < 1$ there is no (S, cS)-LSH (by Definition 1) for inner product similarity over $\mathcal{X}_\bullet = \mathcal{Y}_\bullet = \{x \mid \|x\| \leq 1\}$.*

Proof. The same argument as in Shrivastava and Li (2014a, Theorem 1) applies: Assume for contradiction h is an (S, cS, p_1, p_2)-LSH (with $p_1 > p_2$). Let x be a vector such that $\|x\| = cS < 1$. Let $q = x \in \mathcal{X}_\bullet$ and $y = \frac{1}{c}x \in \mathcal{X}_\bullet$. Therefore, we have $q^\top x = cS$ and $q^\top y = S$. However, since $q = x$, $\mathbb{P}_h(h(q) = h(x)) = 1 \leq p_2 < p_1 = \mathbb{P}_h(h(q) = h(y)) \leq 1$ and we get a contradiction. □

5.2. L2-ALSH(SL)

We might hope L2-ALSH(SL) is a valid ALSH here. Unfortunately, whenever $S < (c+1)/2$, and so in particular for all $S < 1/2$, it is not:

Theorem 5.2. *For any $0 < c < 1$ and any $0 < S < (c+1)/2$, there are no U, m and r such that L2-ALSH(SL) is an (S, cS)-ALSH for inner product similarity over $\mathcal{X}_\bullet =$*

$\mathcal{Y}_\bullet = \{x \mid \|x\| \leq 1\}$.

Proof. Let q_1 and x_1 be unit vectors such that $q_1^\top x_1 = S$. Let x_2 be a unit vector and define $q_2 = cSx_2$. For any U and m:

$$\begin{aligned}\|P(x_2) - Q(q_2)\|^2 &= \|q_2\| + \frac{m}{4} + \|Ux_2\|^{2^{m+1}} - 2q_2^\top x \\ &= c^2S^2 + \frac{m}{4} + U^{2^{m+1}} - 2cSU \\ &\leq 1 + \frac{m}{4} + U^{2^{m+1}} - 2SU \\ &= \|P(x_2) - Q(q_2)\|^2\end{aligned}$$

where the inequality follows from $S < (c+1)/2$. Now, the same arguments as in Lemma 1 using monotonicity of collision probabilities in $\|P(x) - Q(q)\|$ establish LS-ALSH(SL) is not an (S, cS)-ALSH. □

In the supplementary material, we show a stronger negative result for SIGN-ALSH(SL): for any $S > 0$ and $0 < c < 1$, there are no U, m such that SIGN-ALSH(SL) is an $(S, cS) - ALSH$.

5.3. SIMPLE-ALSH

Fortunately, we can define a variant of SIMPLE-LSH, which we refer to as SIMPLE-ALSH, for this more general case where queries are not normalized. We use the pair of transformations:

$$\begin{aligned}P(x) &= \left[x; \sqrt{1 - \|x\|_2^2}; 0\right] \\ Q(x) &= \left[x; 0; \sqrt{1 - \|x\|_2^2}\right]\end{aligned} \tag{13}$$

and the random mappings $f(x) = h_a(P(x))$, $g(y) = h_a(Q(x))$, where $h_a(z)$ is as in (11). It is clear that by these definitions, we always have that for all $x, y \in \mathcal{X}_\bullet$, $P(x)^\top Q(y) = x^\top y$ and $\|P(x)\| = \|Q(y)\| = 1$.

Theorem 5.3. SIMPLE-ALSH *is a universal ALSH over* $\mathcal{X}_\bullet = \mathcal{Y}_\bullet = \{x \mid \|x\| \leq 1\}$. *That is, for every* $0 < S, c < 1$*, it is an* (S, cS)*-ALSH over* $\mathcal{X}_\bullet, \mathcal{Y}_\bullet$.

Proof. The choice of mappings ensures that for all $x, y \in \mathcal{X}_\bullet$ we have $P(x)^\top Q(y) = x^\top y$ and $\|P(x)\| = \|Q(y)\| = 1$, and so $\mathbb{P}[h_a(P(x)) = h_a(Q(y))] = 1 - \frac{\cos^{-1}(q^\top x)}{\pi}$. As in the proof of Theorem 4.2, monotonicity of $1 - \frac{\cos^{-1}(x)}{\pi}$ establishes the desired ALSH properties. □

Shrivastava and Li (2015) also showed how a modification of SIMPLE-ALSH can be used for searching similarity measures such as set containment and weighted Jaccard similarity.

6. Conclusion

We provide a complete characterization of when symmetric and asymmetric LSH are possible for inner product similarity:

- Over $\mathbb{R}^d$, no symmetric nor asymmetric LSH is possible.
- For the MIPS setting, with normalized queries $\|q\| = 1$ and bounded database vectors $\|x\| \leq 1$, a universal symmetric LSH is possible.
- When queries and database vectors are bounded but not normalized, a symmetric LSH is not possible, but a universal asymmetric LSH is. Here we see the power of asymmetry.

This corrects the view of Shrivastava and Li (2014a), who used the nonexistence of a symmetric LSH over $\mathbb{R}^d$ to motivate an asymmetric LSH when queries are normalized and database vectors are bounded, even though we now see that in these two settings there is actually no advantage to asymmetry. In the third setting, where an asymmetric hash is indeed needed, the hashes suggested by Shrivastava and Li (2014a;b) are not ALSH, and a different asymmetric hash is required (which we provide). Furthermore, even in the MIPS setting when queries are normalized (the second setting), the asymmetric hashes suggested by Shrivastava and Li (2014a;b) are *not* universal and require tuning parameters specific to S, c, in contrast to SIMPLE-LSH which is symmetric, parameter-free and universal.

It is important to emphasize that even though in the MIPS setting an asymmetric hash, as we define here, is not needed, an asymmetric view of the problem *is* required. In particular, to use a symmetric hash, one *must* normalize the queries but not the database vectors, which can legitimately be viewed as an asymmetric operation which is part of the hash (though then the hash would not be, strictly speaking, an ALSH). In this regard Shrivastava and Li (2014a) do indeed successfully identify the need for an asymmetric view of MIPS, and provide the first practical ALSH for the problem.

References

Bachrach, Y., Finkelstein, Y., Gilad-Bachrach, R., Katzir, L., Koenigstein, N., Nice, N., and Paquet, U. (2014). Speeding up the Xbox recommender system using a euclidean transformation for inner-product spaces. *In Proceedings of the 8th ACM Conference on Recommender systems*, pages 257–264.

Charikar, M. S. (2002). Similarity estimation techniques from rounding algorithms. *STOC*.

Cremonesi, P., Koren, Y., and Turrin, R. (2010). Perfor-

mance of recommender algorithms on top-n recommendation tasks. *In Proceedings of the fourth ACM conference on Recommender systems, ACM*, page 3946.

Curtin, R. R., Ram, P., and Gray, A. G. (2013). Fast exact max-kernel search. *SDM*, pages 1–9.

Datar, M., Immorlica, N., Indyk, P., and Mirrokni, S. V. (2004). Locality-sensitive hashing scheme based on p-stable distributions. *In Proc. 20th SoCG*, pages 253–262.

Dean, T., Ruzon, M., Segal, M., Shlens, J., Vijayanarasimhan, S., and Yagnik, J. (2013). Fast, accurate detection of 100,000 object classes on a single machine. *CVPR*.

Forster, J., Schmitt, N., Simon, H., and Suttorp, T. (2003). Estimating the optimal margins of embeddings in euclidean half spaces. *Machine Learning*, 51:263281.

Gionis, A., Indyk, P., and Motwani, R. (1999). Similarity search in high dimensions via hashing. *VLDB*, 99:518–529.

Goemans, M. X. and Williamson, D. P. (1995). Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42.6:1115–1145.

Indyk, P. and Motwani, R. (1998). Approximate nearest neighbors: towards removing the curse of dimensionality. *STOC*, pages 604–613.

Jain, P., , and Kapoor, A. (2009). Active learning for large multi-class problems. *CVPR*.

Joachims, T. (2006). Training linear SVMs in linear time. *SIGKDD*.

Joachims, T., Finley, T., and Yu, C.-N. J. (2009). Cutting-plane training of structural SVMs. *Machine Learning*, 77.1:27–59.

Koenigstein, N., Ram, P., and Shavitt, Y. (2012). Efficient retrieval of recommendations in a matrix factorization framework. *CIKM*, pages 535–544.

Koren, Y., Bell, R., and Volinsky., C. (2009). Matrix factorization techniques for recommender systems. *Computer*, 42.8:30–37.

Neyshabur, B., Makarychev, Y., and Srebro, N. (2014). Clustering, hamming embedding, generalized LSH and the max norm. *ALT*.

Neyshabur, B., Yadollahpour, P., Makarychev, Y., Salakhutdinov, R., and Srebro, N. (2013). The power of asymmetry in binary hashing. *NIPS*.

Ram, P. and Gray, A. G. (2012). Maximum inner-product search using cone trees. *SIGKDD*.

Shrivastava, A. and Li, P. (2014a). Asymmetric LSH (ALSH) for sublinear time maximum inner product search (MIPS). *NIPS*.

Shrivastava, A. and Li, P. (2014b). Improved asymmetric locality sensitive hashing (ALSH) for maximum inner product search (MIPS). *arXiv:1410.5410*.

Shrivastava, A. and Li, P. (2015). Asymmetric minwise hashing for indexing binary inner products and set containment. *WWW*.

Srebro, N., Rennie, J., and Jaakkola, T. (2005). Maximum margin matrix factorization. *NIPS*.

Srebro, N. and Shraibman, A. (2005). Rank, trace-norm and max-norm. *COLT*.