Asymmetric LSH (ALSH) for Sublinear Time Maximum Inner Product Search (MIPS)

Anshumali Shrivastava
Department of Computer Science
Computing and Information Science
Cornell University
Ithaca, NY 14853, USA
anshu@cs.cornell.edu

Ping Li
Department of Statistics and Biostatistics
Department of Computer Science
Rutgers University
Piscataway, NJ 08854, USA
pingli@stat.rutgers.edu

Abstract

We present the first provably sublinear time hashing algorithm for approximate *Maximum Inner Product Search* (MIPS). Searching with (un-normalized) inner product as the underlying similarity measure is a known difficult problem and finding hashing schemes for MIPS was considered hard. While the existing Locality Sensitive Hashing (LSH) framework is insufficient for solving MIPS, in this paper we extend the LSH framework to allow asymmetric hashing schemes. Our proposal is based on a key observation that the problem of finding maximum inner products, after independent asymmetric transformations, can be converted into the problem of approximate near neighbor search in classical settings. This key observation makes efficient sublinear hashing scheme for MIPS possible. Under the extended asymmetric LSH (ALSH) framework, this paper provides an example of explicit construction of provably fast hashing scheme for MIPS. Our proposed algorithm is simple and easy to implement. The proposed hashing scheme leads to significant computational savings over the two popular conventional LSH schemes: (i) Sign Random Projection (SRP) and (ii) hashing based on p-stable distributions for L_2 norm (L2LSH), in the collaborative filtering task of item recommendations on Netflix and Movielens (10M) datasets.

1 Introduction and Motivation

The focus of this paper is on the problem of *Maximum Inner Product Search (MIPS)*. In this problem, we are given a giant data vector collection $\mathcal{S}$ of size N, where $\mathcal{S} \subset \mathbb{R}^D$, and a given query point $q \in \mathbb{R}^D$. We are interested in searching for $p \in \mathcal{S}$ which maximizes (or approximately maximizes) the **inner product** $q^T p$. Formally, we are interested in efficiently computing

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

The MIPS problem is related to *near neighbor search (NNS)*, which instead requires computing

$$p = \arg\min_{x\in\mathcal{S}} ||q-x||_2^2 = \arg\min_{x\in\mathcal{S}}(||x||_2^2 - 2q^T x) \tag{2}$$

These two problems are equivalent if the norm of every element $x \in \mathcal{S}$ is constant. Note that the value of the norm $||q||_2$ has no effect as it is a constant and does not change the identity of $\arg\max$ or $\arg\min$. There are many scenarios in which MIPS arises naturally at places where the norms of the elements in $\mathcal{S}$ have significant variations [13] and cannot be controlled, e.g., (i) recommender system, (ii) large-scale object detection with DPM, and (iii) multi-class label prediction.

Recommender systems: Recommender systems are often based on collaborative filtering which relies on past behavior of users, e.g., past purchases and ratings. Latent factor modeling based on matrix factorization [14] is a popular approach for solving collaborative filtering. In a typical matrix factorization model, a user i is associated with a latent user characteristic vector u_i, and similarly, an item j is associated with a latent item characteristic vector v_j. The rating $r_{i,j}$ of item j by user i is modeled as the **inner product** between the corresponding characteristic vectors.

In this setting, given a user i and the corresponding learned latent vector u_i finding the right item j, to recommend to this user, involves computing

$$j = \arg\max_{j'} \ r_{i,j'} = \arg\max_{j'} \ u_i^T v_{j'} \tag{3}$$

which is an instance of the standard MIPS problem. It should be noted that we do not have control over the norm of the learned vector, i.e., $\|v_j\|_2$, which often has a wide range in practice [13].

If there are N items to recommend, solving (3) requires computing N inner products. Recommendation systems are typically deployed in on-line application over web where the number N is huge. A brute force linear scan over all items, for computing $\arg\max$, would be prohibitively expensive.

Large-scale object detection with DPM: Deformable Part Model (DPM) based representation of images is the state-of-the-art in object detection tasks [8]. In DPM model, firstly a set of part filters are learned from the training dataset. During detection, these learned filter activations over various patches of the test image are used to score the test image. The activation of a filter on an image patch is an inner product between them. Typically, the number of possible filters are large (e.g., millions) and so scoring the test image is costly. Recently, it was shown that scoring based only on filters with high activations performs well in practice [7]. Identifying those filters having high activations on a given image patch requires computing top inner products. Consequently, an efficient solution to the MIPS problem will benefit large scale object detections based on DPM.

Multi-class (and/or multi-label) prediction: The models for multi-class SVM (or logistic regression) learn a weight vector w_i for each of the class label i. After the weights are learned, given a new test data vector x_{test}, predicting its class label is basically an MIPS problem:

$$y_{test} = \arg\max_{i\in\mathcal{L}} \ x_{test}^T \, w_i \tag{4}$$

where $\mathcal{L}$ is the set of possible class labels. Note that the norms of the vectors $\|w_i\|_2$ are not constant. The size, $|\mathcal{L}|$, of the set of class labels differs in applications. Classifying with large number of possible class labels is common in multi-label learning and fine grained object classification, for instance, prediction task with $|\mathcal{L}| = 100,000$ [7]. Computing such high-dimensional vector multiplications for predicting the class label of a single instance can be expensive in, e.g., user-facing applications.

1.1 The Need for Hashing Inner Products

Solving the MIPS problem can have significant practical impact. [19, 13] proposed solutions based on tree data structure combined with branch and bound space partitioning technique similar to k-d trees [9]. Later, the same method was generalized for general max kernel search [5], where the runtime guarantees, like other space partitioning methods, are heavily dependent on the dimensionality and the expansion constants. In fact, it is well-known that techniques based on space partitioning (such as k-d trees) suffer from the curse of dimensionality. For example, [24] showed that techniques based on space partitioning degrade to linear search, even for dimensions as small as 10 or 20.

Locality Sensitive Hashing (LSH) [12] based randomized techniques are common and successful in industrial practice for efficiently solving NNS (*near neighbor search*). Unlike space partitioning techniques, both the running time as well as the accuracy guarantee of LSH based NNS are in a way independent of the dimensionality of the data. This makes LSH suitable for large scale processing system dealing with ultra-high dimensional datasets which are common in modern applications. Furthermore, LSH based schemes are massively parallelizable, which makes them ideal for modern "Big" datasets. The prime focus of this paper will be on efficient hashing based algorithms for MIPS, which do not suffer from the curse of dimensionality.

1.2 Our Contributions

We develop *Asymmetric LSH (ALSH)*, an extended LSH scheme for efficiently solving the approximate MIPS problem. Finding hashing based algorithms for MIPS was considered hard [19, 13]. We formally show that, under the current framework of LSH, there cannot exist any LSH for solving MIPS. Despite this negative result, we show that it is possible to relax the current LSH framework to allow asymmetric hash functions which can efficiently solve MIPS. This generalization comes with no extra cost and the ALSH framework inherits all the theoretical guarantees of LSH.

Our construction of asymmetric LSH is based on an interesting fact that the original MIPS problem, after asymmetric transformations, reduces to the problem of approximate near neighbor search in classical settings. Based on this key observation, we provide an example of explicit construction of asymmetric hash function, leading to the first provably sublinear query time hashing algorithm for approximate similarity search with (un-normalized) inner product as the similarity. The new ALSH framework is of independent theoretical interest. We report other explicit constructions in [22, 21].

We also provide experimental evaluations on the task of recommending top-ranked items with collaborative filtering, on Netflix and Movielens (10M) datasets. The evaluations not only support our theoretical findings but also quantify the obtained benefit of the proposed scheme, in a useful task.

2 Background

2.1 Locality Sensitive Hashing (LSH)

A commonly adopted formalism for approximate near-neighbor search is the following:

Definition: (c-Approximate Near Neighbor or c-NN) *Given a set of points in a D-dimensional space $\mathbb{R}^D$, and parameters $S_0 > 0$, $\delta > 0$, construct a data structure which, given any query point q, does the following with probability $1 - \delta$: if there exists an S_0-near neighbor of q in P, it reports some cS_0-near neighbor of q in P.*

In the definition, the S_0-near neighbor of point q is a point p with $Sim(q, p) \geq S_0$, where Sim is the similarity of interest. Popular techniques for c-NN are often based on *Locality Sensitive Hashing* (LSH) [12], which is a family of functions with the nice property that more similar objects in the domain of these functions have a higher probability of colliding in the range space than less similar ones. In formal terms, consider $\mathcal{H}$ a family of hash functions mapping $\mathbb{R}^D$ to a set $\mathcal{I}$.

Definition: (Locality Sensitive Hashing (LSH)) *A family $\mathcal{H}$ is called (S_0, cS_0, p_1, p_2)-sensitive if, for any two point $x, y \in \mathbb{R}^D$, h chosen uniformly from $\mathcal{H}$ satisfies the following:*

- *if $Sim(x, y) \geq S_0$ then $Pr_{\mathcal{H}}(h(x) = h(y)) \geq p_1$*
- *if $Sim(x, y) \leq cS_0$ then $Pr_{\mathcal{H}}(h(x) = h(y)) \leq p_2$*

For efficient approximate nearest neighbor search, $p_1 > p_2$ and $c < 1$ is needed.

Fact 1 [12]: Given a family of (S_0, cS_0, p_1, p_2) -sensitive hash functions, one can construct a data structure for c-NN with $O(n^\rho \log n)$ query time and space $O(n^{1+\rho})$, where $\rho = \frac{\log p_1}{\log p_2} < 1$.

2.2 LSH for L_2 Distance (L2LSH)

[6] presented a novel LSH family for all L_p ($p \in (0, 2]$) distances. In particular, when $p = 2$, this scheme provides an LSH family for L_2 distances. Formally, given a fixed (real) number r, we choose a random vector a with each component generated from i.i.d. normal, i.e., $a_i \sim N(0, 1)$, and a scalar b generated uniformly at random from $[0, r]$. The hash function is defined as:

$$h^{L2}_{a,b}(x) = \left\lfloor \frac{a^T x + b}{r} \right\rfloor \tag{5}$$

where $\lfloor \rfloor$ is the floor operation. The collision probability under this scheme can be shown to be

$$Pr(h^{L2}_{a,b}(x) = h^{L2}_{a,b}(y)) = F_r(d); \qquad F_r(d) = 1 - 2\Phi(-r/d) - \frac{2}{\sqrt{2\pi}(r/d)}\left(1 - e^{-(r/d)^2/2}\right) \tag{6}$$

where $\Phi(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx$ is the cumulative density function (cdf) of standard normal distribution and $d = ||x - y||_2$ is the Euclidean distance between the vectors x and y. This collision probability $F_r(d)$ is a monotonically decreasing function of the distance d and hence $h^{L2}_{a,b}$ is an LSH for L_2 distances. This scheme is also the part of LSH package [1]. Here r is a parameter. As argued previously, $||x-y||_2 = \sqrt{(||x||_2^2 + ||y||_2^2 - 2x^T y)}$ is not monotonic in the inner product $x^T y$ unless the given data has a constant norm. Hence, $h^{L2}_{a,b}$ is not suitable for MIPS.

The recent work on *coding for random projections* [16] showed that L2LSH can be improved when the data are normalized for building large-scale linear classifiers as well as near neighbor search [17]. In particular, [17] showed that 1-bit coding (i.e., sign random projections (SRP) [10, 3]) or 2-bit coding are often better compared to using more bits. It is known that SRP is designed for retrieving with cosine similarity: $Sim(x, y) = \frac{x^T y}{||x||_2 ||y||_2}$. Again, ordering under this similarity can be very different from the ordering of inner product and hence SRP is also unsuitable for solving MIPS.

3 Hashing for MIPS

3.1 A Negative Result

We first show that, under the current LSH framework, it is impossible to obtain a locality sensitive hashing scheme for MIPS. In [19, 13], the authors also argued that finding locality sensitive hashing for inner products could be hard, but to the best of our knowledge we have not seen a formal proof.

Theorem 1 *There cannot exist any LSH family for MIPS.*

Proof: *Suppose there exists such hash function h. For un-normalized inner products the self similarity of a point x with itself is $Sim(x,x) = x^Tx = ||x||_2^2$ and there may exist another points y, such that $Sim(x,y) = y^Tx > ||x||_2^2 + C$, for any constant C. Under any single randomized hash function h, the collision probability of the event $\{h(x) = h(x)\}$ is always 1. So if h is an LSH for inner product then the event $\{h(x) = h(y)\}$ should have higher probability compared to the event $\{h(x) = h(x)\}$, since we can always choose y with $Sim(x,y) = S_0 + \delta > S_0$ and $cS_0 > Sim(x,x)$ $\forall S_0$ and $c < 1$. This is not possible because the probability cannot be greater than 1. This completes the proof.* □

3.2 Our Proposal: Asymmetric LSH (ALSH)

The basic idea of LSH is probabilistic bucketing and it is more general than the requirement of having a single hash function h. The classical LSH algorithms use the same hash function h for both the preprocessing step and the query step. One assigns buckets in the hash table to all the candidates $x \in \mathcal{S}$ using h, then uses the same h on the query q to identify relevant buckets. The only requirement for the proof of Fact 1, to work is that the collision probability of the event $\{h(q) = h(x)\}$ increases with the similarity $Sim(q,x)$. The theory [11] behind LSH still works if we use hash function h_1 for preprocessing $x \in \mathcal{S}$ and a different hash function h_2 for querying, as long as the probability of the event $\{h_2(q) = h_1(x)\}$ increases with $Sim(q,x)$, and there exist p_1 and p_2 with the required property. The traditional LSH definition does not allow this asymmetry but it is not a required condition in the proof. For this reason, we can relax the definition of c-NN without losing runtime guarantees. [20] used a related (asymmetric) idea for solving 3-way similarity search.

We first define a modified locality sensitive hashing in a form which will be useful later.

Definition: (*Asymmetric* Locality Sensitive Hashing (ALSH)) A family $\mathcal{H}$, along with the two vector functions $Q : \mathbb{R}^D \mapsto \mathbb{R}^{D'}$ (*Query Transformation*) and $P : \mathbb{R}^D \mapsto \mathbb{R}^{D'}$ (*Preprocessing Transformation*), is called (S_0, cS_0, p_1, p_2)-sensitive if, for a given c-NN instance with query q and any x in the collection $\mathcal{S}$, the hash function h chosen uniformly from $\mathcal{H}$ satisfies the following:

- if $Sim(q,x) \geq S_0$ then $Pr_{\mathcal{H}}(h(Q(q))) = h(P(x))) \geq p_1$
- if $Sim(q,x) \leq cS_0$ then $Pr_{\mathcal{H}}(h(Q(q)) = h(P(x))) \leq p_2$

When $Q(x) = P(x) = x$, we recover the vanilla LSH definition with $h(.)$ as the required hash function. Coming back to the problem of MIPS, if Q and P are different, the event $\{h(Q(x)) = h(P(x))\}$ will not have probability equal to 1 in general. Thus, $Q \neq P$ can counter the fact that self similarity is not highest with inner products. We just need the probability of the new collision event $\{h(Q(q)) = h(P(y))\}$ to satisfy the conditions in the definition of c-NN for $Sim(q,y) = q^Ty$. Note that the query transformation Q is only applied on the query and the pre-processing transformation P is applied to $x \in \mathcal{S}$ while creating hash tables. It is this asymmetry which will allow us to solve MIPS efficiently. In Section 3.3, we explicitly show a construction (and hence the existence) of asymmetric locality sensitive hash function for solving MIPS. The source of randomization h for both q and $x \in \mathcal{S}$ is the same. Formally, it is not difficult to show a result analogous to Fact 1.

Theorem 2 *Given a family of hash function $\mathcal{H}$ and the associated query and preprocessing transformations P and Q, which is (S_0, cS_0, p_1, p_2) -sensitive, one can construct a data structure for c-NN with $O(n^\rho \log n)$ query time and space $O(n^{1+\rho})$, where $\rho = \frac{\log p_1}{\log p_2}$.*

3.3 From MIPS to Near Neighbor Search (NNS)

Without loss of any generality, let $U < 1$ be a number such that $||x_i||_2 \leq U < 1, \quad \forall x_i \in \mathcal{S}$. If this is not the case then define a scaling transformation,

$$S(x) = \frac{U}{M} \times x; \qquad M = max_{x_i \in \mathcal{S}} ||x_i||_2; \tag{7}$$

Note that we are allowed one time preprocessing and asymmetry, S is the part of asymmetric transformation. For simplicity of arguments, let us assume that $||q||_2 = 1$, the $\arg\max$ is anyway independent of the norm of the query. Later we show in Section 3.6 that it can be easily removed.

We are now ready to describe the key step in our algorithm. First, we define two vector transformations $P : \mathbb{R}^D \mapsto \mathbb{R}^{D+m}$ and $Q : \mathbb{R}^D \mapsto \mathbb{R}^{D+m}$ as follows:

$$P(x) = [x; ||x||_2^2; ||x||_2^4; ....; ||x||_2^{2^m}]; \qquad Q(x) = [x; 1/2; 1/2; ....; 1/2], \tag{8}$$

where [;] is the concatenation. $P(x)$ appends m scalers of the form $||x||_2^{2^i}$ at the end of the vector x, while Q(x) simply appends m "1/2" to the end of the vector x. By observing that

$$Q(q)^T P(x_i) = q^T x_i + \frac{1}{2}(||x_i||_2^2 + ||x_i||_2^4 + ... + ||x_i||_2^{2^m}); \quad ||P(x_i)||_2^2 = ||x_i||_2^2 + ||x_i||_2^4 + ... + ||x_i||_2^{2^{m+1}}$$

we obtain the following key equality:

$$||Q(q) - P(x_i)||_2^2 = (1 + m/4) - 2q^T x_i + ||x_i||_2^{2^{m+1}} \tag{9}$$

Since $||x_i||_2 \le U < 1$, $||x_i||^{2^{m+1}} \to 0$, at the tower rate (exponential to exponential). The term $(1 + m/4)$ is a fixed constant. As long as m is not too small (e.g., $m \ge 3$ would suffice), we have

$$\arg\max_{x \in \mathcal{S}} q^T x \simeq \arg\min_{x \in \mathcal{S}} ||Q(q) - P(x)||_2 \tag{10}$$

This gives us the connection between solving un-normalized MIPS and approximate near neighbor search. Transformations P and Q, when norms are less than 1, provide correction to the L_2 distance $||Q(q) - P(x_i)||_2$ making it rank correlate with the (un-normalized) inner product. This works only after shrinking the norms, as norms greater than 1 will instead blow the term $||x_i||_2^{2^{m+1}}$.

3.4 Fast Algorithms for MIPS

Eq. (10) shows that MIPS reduces to the standard approximate near neighbor search problem which can be efficiently solved. As the error term $||x_i||_2^{2^{m+1}} < U^{2^{m+1}}$ goes to zero at a tower rate, it quickly becomes negligible for any practical purposes. In fact, from theoretical perspective, since we are interested in guarantees for c-approximate solutions, this additional error can be absorbed in the approximation parameter c. Formally, we can state the following theorem.

Theorem 3 *Given a c-approximate instance of MIPS, i.e., $Sim(q, x) = q^T x$, and a query q such that $||q||_2 = 1$ along with a collection $\mathcal{S}$ having $||x||_2 \le U < 1\ \forall x \in \mathcal{S}$. Let P and Q be the vector transformations defined in (8). We have the following two conditions for hash function $h^{L2}_{a,b}$ (5)*
1) if $q^T x \ge S_0$ then $Pr[h^{L2}_{a,b}(Q(q)) = h^{L2}_{a,b}(P(x))] \ge F_r\left(\sqrt{1 + m/4 - 2S_0 + U^{2^{m+1}}}\right)$
2) if $q^T x \le cS_0$ then $Pr[h^{L2}_{a,b}(Q(q)) = h^{L2}_{a,b}(P(x))] \le F_r\left(\sqrt{1 + m/4 - 2cS_0}\right)$
where the function F_r is defined in (6).

Thus, we have obtained $p_1 = F_r\left(\sqrt{(1 + m/4) - 2S_0 + U^{2^{m+1}}}\right)$ and $p_2 = F_r\left(\sqrt{(1 + m/4) - 2cS_0}\right)$. Applying Theorem 2, we can construct data structures with worst case $O(n^\rho \log n)$ query time guarantees for c-approximate MIPS, where

$$\rho = \frac{\log F_r\left(\sqrt{1 + m/4 - 2S_0 + U^{2^{m+1}}}\right)}{\log F_r\left(\sqrt{1 + m/4 - 2cS_0}\right)} \tag{11}$$

We need $p_1 > p_2$ in order for $\rho < 1$. This requires us to have $-2S_0 + U^{2^{m+1}} < -2cS_0$, which boils down to the condition $c < 1 - \frac{U^{2^{m+1}}}{2S_0}$. Note that $\frac{U^{2^{m+1}}}{2S_0}$ can be made arbitrarily close to zero with the appropriate value of m. For any given $c < 1$, there always exist $U < 1$ and m such that $\rho < 1$. This way, we obtain a sublinear query time algorithm for MIPS.

We also have one more parameter r for the hash function $h_{a,b}$. Recall the definition of F_r in Eq. (6): $F_r(d) = 1 - 2\Phi(-r/d) - \frac{2}{\sqrt{2\pi}(r/d)}\left(1 - e^{-(r/d)^2/2}\right)$. Thus, given a c-approximate MIPS instance, ρ

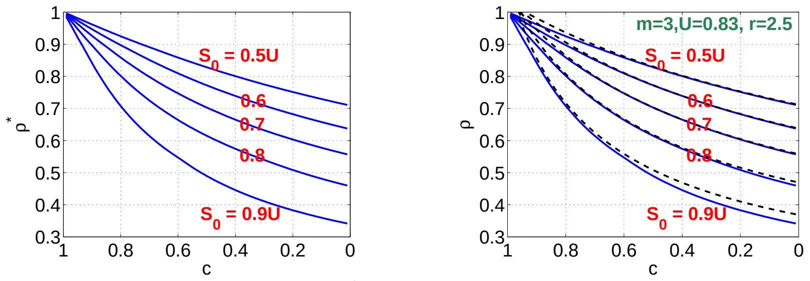


Figure 1: **Left panel**: Optimal values of ρ^* with respect to approximation ratio c for different S_0. The optimization of Eq. (14) was conducted by a grid search over parameters r, U and m, given S_0 and c. **Right Panel**: ρ values (dashed curves) for $m = 3$, $U = 0.83$ and $r = 2.5$. The solid curves are ρ^* values. See more details about parameter recommendations in *arXiv:1405.5869*.

is a function of 3 parameters: U, m, r. The algorithm with the best query time chooses U, m and r, which minimizes the value of ρ. For convenience, we define

$$\rho^* = \min_{U,m,r} \frac{\log F_r\left(\sqrt{1+m/4-2S_0+U^{2^{m+1}}}\right)}{\log F_r\left(\sqrt{1+m/4-2cS_0}\right)} \quad s.t. \quad \frac{U^{2^{m+1}}}{2S_0} < 1-c, \;\; m \in \mathbb{N}^+, \;\; 0 < U < 1. \qquad (12)$$

See Figure 1 for the plots of ρ^*. With this best value of ρ, we can state our main result in Theorem 4.

Theorem 4 *(***Approximate MIPS is Efficient***) For the problem of c-approximate MIPS with $||q||_2 = 1$, one can construct a data structure having $O(n^{\rho^*}\log n)$ query time and space $O(n^{1+\rho^*})$, where $\rho^* < 1$ is the solution to constraint optimization (14).*

3.5 Practical Recommendation of Parameters

Just like in the typical LSH framework, the value of ρ^* in Theorem 4 depends on the c-approximate instance we aim to solve, which requires knowing the similarity threshold S_0 and the approximation ratio c. Since, $||q||_2 = 1$ and $||x||_2 \le U < 1$, $\forall x \in \mathcal{S}$, we have $q^tx \le U$. A reasonable choice of the threshold S_0 is to choose a high fraction of U, for example, $S_0 = 0.9U$ or $S_0 = 0.8U$.

The computation of ρ^* and the optimal values of corresponding parameters can be conducted via a grid search over the possible values of U, m and r. We compute ρ^* in Figure 1 (Left Panel). For convenience, we recommend $m = 3$, $U = 0.83$, and $r = 2.5$. With this choice of the parameters, Figure 1 (Right Panel) shows that the ρ values using these parameters are very close to ρ^* values.

3.6 Removing the Condition $||q||_2 = 1$

Changing norms of the query does not affect the $\arg\max_{x\in\mathcal{C}} q^Tx$. Thus in practice for retrieving top-ranked items, normalizing the query should not affect the performance. But for theoretical purposes, we want the runtime guarantee to be independent of $||q||_2$. We are interested in the c-approximate instance which being a threshold based approximation changes if the query is normalized.

Previously, transformations P and Q were precisely meant to remove the dependency on the norms of x. Realizing the fact that we are allowed asymmetry, we can use the same idea to get rid of the norm of q. Let M be the upper bound on all the norms or the radius of the space as defined in Eq (7). Let the transformation $S : \mathbb{R}^D \to \mathbb{R}^D$ be the ones defined in Eq (7). Define asymmetric transformations $P' : \mathbb{R}^D \to \mathbb{R}^{D+2m}$ and $Q' : \mathbb{R}^D \to \mathbb{R}^{D+2m}$ as

$$P'(x) = [x; ||x||_2^2; ||x||_2^4; ....; ||x||_2^{2^m}; 1/2; ...1/2]; \quad Q'(x) = [x; 1/2; ....; 1/2; ||x||_2^2; ||x||_2^4; ....; ||x||_2^{2^m}],$$

Given the query q and data point x, our new asymmetric transformations are $Q'(S(q))$ and $P'(S(x))$ respectively. We observe that

$$||Q'(S(q)) - P'(S(x))||_2^2 = \frac{m}{2} + ||S(x)||_2^{2^{m+1}} + ||S(q)||_2^{2^{m+1}} - 2q^tx \times \left(\frac{U^2}{M^2}\right) \qquad (13)$$

Both $||S(x)||_2^{2^{m+1}}, ||S(q)||_2^{2^{m+1}} \le U^{2^{m+1}} \to 0$. Using exactly same arguments as before, we obtain

Theorem 5 *(***Unconditional Approximate MIPS is Efficient***) For the problem of c-approximate MIPS in a bounded space, one can construct a data structure having $O(n^{\rho_u^*}\log n)$ query time and space $O(n^{1+\rho_u^*})$, where $\rho_u^* < 1$ is the solution to constraint optimization (14).*

$$\rho_u^* = \min_{0<U<1, m\in N, r} \frac{\log F_r\left(\sqrt{m/2 - 2S_0\left(\frac{U^2}{M^2}\right) + 2U^{2^{m+1}}}\right)}{\log F_r\left(\sqrt{m/2 - 2cS_0\left(\frac{U^2}{M^2}\right)}\right)} \quad s.t. \quad \frac{U^{(2^{m+1}-2)}M^2}{S_0} < 1 - c, \tag{14}$$

Again, for any c-approximate MIPS instance, with S_0 and c, we can always choose m big enough such that $\rho_u^* < 1$. The theoretical guarantee only depends on the radius of the space M.

3.7 A Generic Recipe for Constructing Asymmetric LSHs

We are allowed any asymmetric transformation on x and q. This gives us a lot of flexibility to construct ALSH for new similarities $\mathcal{S}$ that we are interested in. The generic idea is to take a particular similarity $Sim(x, q)$ for which we know an existing LSH or ALSH. Then we construct transformations P and Q such $Sim(P(x), Q(q))$ is monotonic in the similarity $\mathcal{S}$ that we are interested in. The other observation that makes it easier to construct P and Q is that LSH based guarantees are independent of dimensions, thus we can expand the dimensions like we did for P and Q.

This paper focuses on using L2LSH to convert near neighbor search of L_2 distance into an ALSH (i.e., *L2-ALSH*) for MIPS. We can devise new ALSHs for MIPS using other similarities and hash functions. For instance, utilizing sign random projections (SRP), the known LSH for correlations, we can construct different P and Q leading to a better ALSH (i.e., *Sign-ALSH*) for MIPS [22]. We are aware another work [18] which performs very similarly to *Sign-ALSH*. Utilizing minwise hashing [2, 15], which is the LSH for resemblance and is known to outperform SRP in sparse data [23], we can construct an even better ALSH (i.e., *MinHash-ALSH*) for MIPS over binary data [21].

4 Evaluations

Datasets. We evaluate the proposed ALSH scheme for the MIPS problem on two popular collaborative filtering datasets on the task of item recommendations: (i) Movielens(10M), and (ii) Netflix. Each dataset forms a sparse **user-item matrix** R, where the value of $R(i, j)$ indicates the rating of user i for movie j. Given the user-item ratings matrix R, we follow the standard PureSVD procedure [4] to generate user and item latent vectors. This procedure generates latent vectors u_i for each user i and vector v_j for each item j, in some chosen fixed dimension f. The PureSVD method returns top-ranked items based on the inner products $u_i^T v_j$, $\forall j$. Despite its simplicity, PureSVD outperforms other popular recommendation algorithms [4]. Following [4], we use the same choices for the latent dimension f, i.e., $f = 150$ for Movielens and $f = 300$ for Netflix.

4.1 Ranking Experiment for Hash Code Quality Evaluations

We are interested in knowing, how the two hash functions correlate with the top-10 inner products. For this task, given a user i and its corresponding user vector u_i, we compute the top-10 gold standard items based on the actual inner products $u_i^T v_j$, $\forall j$. We then compute K different hash codes of the vector u_i and all the item vectors v_js. For every item v_j, we compute the number of times its hash values matches (or collides) with the hash values of query which is user u_i, i.e., we compute $Matches_j = \sum_{t=1}^{K} \mathbf{1}(h_t(u_i) = h_t(v_j))$, based on which we rank all the items.

Figure 2 reports the precision-recall curves in our ranking experiments for top-10 items, for comparing our proposed method with two baseline methods: the original L2LSH and the original sign random projections (SRP). These results confirm the substantial advantage of our proposed method.

4.2 LSH Bucketing Experiment

We implemented the standard (K, L)-parameterized (where L is number of hash tables) bucketing algorithm [1] for retrieving top-50 items based on PureSVD procedure using the proposed ALSH hash function and the two baselines: SRP and L2LSH. We plot the recall vs the mean ratio of inner product required to achieve that recall. The ratio being computed relative to the number of inner products required in a brute force linear scan. In order to remove the effect of algorithm parameters (K, L) on the evaluations, we report the result from the best performing K and L chosen from $K \in \{5, 6, ..., 30\}$ and $L \in \{1, 2, ..., 200\}$ for each query. We use $m = 3$, $U = 0.83$, and $r = 2.5$ for

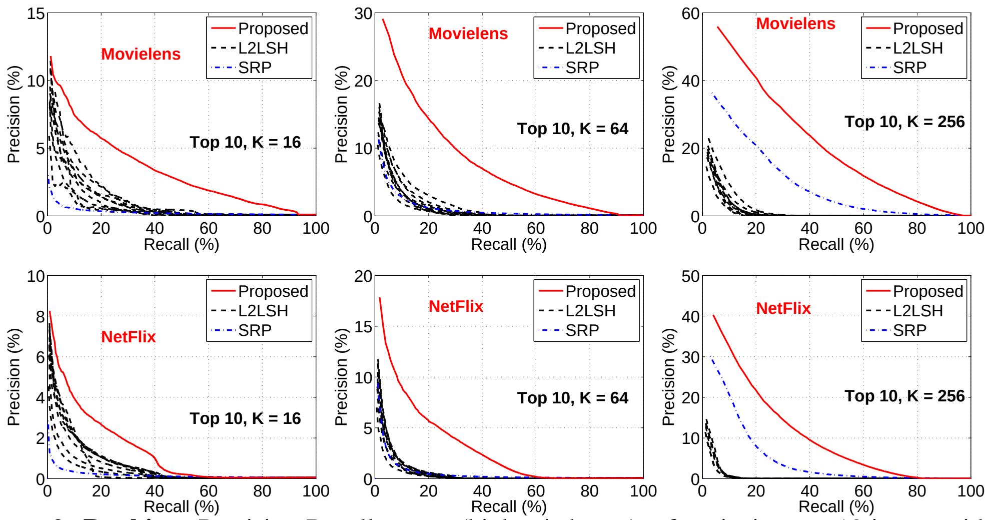


Figure 2: **Ranking.** Precision-Recall curves (higher is better), of retrieving top-10 items, with the number of hashes $K \in \{16, 64, 256\}$. The proposed algorithm (solid, red if color is available) significantly outperforms L2LSH. We fix the parameters $m = 3$, $U = 0.83$, and $r = 2.5$ for our proposed method and we present the results of L2LSH for all r values in $\{1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5\}$.

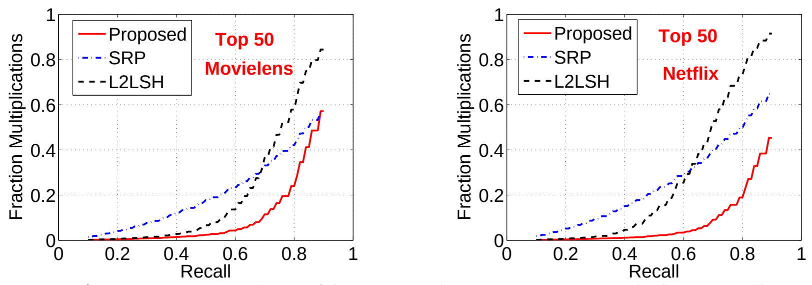


Figure 3: **Bucketing.** Mean number of inner products per query, relative to a linear scan, evaluated by different hashing schemes at different recall levels, for generating top-50 recommendations (Lower is better). The results corresponding to the best performing K and L (for a wide range of K and L) at a given recall value, separately for all the three hashing schemes, are shown.

our hashing scheme. For L2LSH, we observe that using $r = 4$ usually performs well and so we show results for $r = 4$. The results are summarized in Figure 3, confirming that the proposed ALSH leads to significant savings compared to baseline hash functions.

5 Conclusion

MIPS (maximum inner product search) naturally arises in numerous practical scenarios, e.g., collaborative filtering. This problem is challenging and, prior to our work, there existed no provably sublinear time *hashing* algorithms for MIPS. Also, the existing framework of classical LSH (locality sensitive hashing) is not sufficient for solving MIPS. In this study, we develop *ALSH* (asymmetric LSH), which generalizes the existing LSH framework by applying (appropriately chosen) asymmetric transformations to the input query vector and the data vectors in the repository. We present an implementation of ALSH by proposing a novel transformation which converts the original inner products into L_2 distances in the transformed space. We demonstrate, both theoretically and empirically, that this implementation of ALSH provides provably efficient as well as practical solution to MIPS. Other explicit constructions of ALSH, for example, ALSH through cosine similarity, or ALSH through resemblance (for binary data), will be presented in followup technical reports.

Acknowledgments

The research is partially supported by NSF-DMS-1444124, NSF-III-1360971, NSF-Bigdata-1419210, ONR-N00014-13-1-0764, and AFOSR-FA9550-13-1-0137. We appreciate the constructive comments from the program committees of KDD 2014 and NIPS 2014. Shrivastava would also like to thank Thorsten Joachims and the Class of CS6784 (Spring 2014) for valuable feedbacks.

References

[1] A. Andoni and P. Indyk. E2lsh: Exact euclidean locality sensitive hashing. Technical report, 2004.

[2] A. Z. Broder. On the resemblance and containment of documents. In *the Compression and Complexity of Sequences*, pages 21–29, Positano, Italy, 1997.

[3] M. S. Charikar. Similarity estimation techniques from rounding algorithms. In *STOC*, pages 380–388, Montreal, Quebec, Canada, 2002.

[4] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 39–46. ACM, 2010.

[5] R. R. Curtin, A. G. Gray, and P. Ram. Fast exact max-kernel search. In *SDM*, pages 1–9, 2013.

[6] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokn. Locality-sensitive hashing scheme based on p-stable distributions. In *SCG*, pages 253 – 262, Brooklyn, NY, 2004.

[7] T. Dean, M. A. Ruzon, M. Segal, J. Shlens, S. Vijayanarasimhan, and J. Yagnik. Fast, accurate detection of 100,000 object classes on a single machine. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 1814–1821. IEEE, 2013.

[8] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1627–1645, 2010.

[9] J. H. Friedman and J. W. Tukey. A projection pursuit algorithm for exploratory data analysis. *IEEE Transactions on Computers*, 23(9):881–890, 1974.

[10] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of ACM*, 42(6):1115–1145, 1995.

[11] S. Har-Peled, P. Indyk, and R. Motwani. Approximate nearest neighbor: Towards removing the curse of dimensionality. *Theory of Computing*, 8(14):321–350, 2012.

[12] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *STOC*, pages 604–613, Dallas, TX, 1998.

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

[14] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems.

[15] P. Li and A. C. König. Theory and applications b-bit minwise hashing. *Commun. ACM*, 2011.

[16] P. Li, M. Mitzenmacher, and A. Shrivastava. Coding for random projections. In *ICML*, 2014.

[17] P. Li, M. Mitzenmacher, and A. Shrivastava. Coding for random projections and approximate near neighbor search. Technical report, arXiv:1403.8144, 2014.

[18] B. Neyshabur and N. Srebro. A simpler and better lsh for maximum inner product search (mips). Technical report, arXiv:1410.5518, 2014.

[19] P. Ram and A. G. Gray. Maximum inner-product search using cone trees. In *KDD*, pages 931–939, 2012.

[20] A. Shrivastava and P. Li. Beyond pairwise: Provably fast algorithms for approximate k-way similarity search. In *NIPS*, Lake Tahoe, NV, 2013.

[21] A. Shrivastava and P. Li. Asymmetric minwise hashing. Technical report, 2014.

[22] A. Shrivastava and P. Li. An improved scheme for asymmetric lsh. Technical report, arXiv:1410.5410, 2014.

[23] A. Shrivastava and P. Li. In defense of minhash over simhash. In *AISTATS*, 2014.

[24] R. Weber, H.-J. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *VLDB*, pages 194–205, 1998.

Improved Asymmetric Locality Sensitive Hashing (ALSH) for Maximum Inner Product Search (MIPS)

Anshumali Shrivastava
Department of Computer Science
Computing and Information Science
Cornell University
Ithaca, NY 14853, USA
anshu@cs.cornell.edu

Ping Li
Department of Statistics and Biostatistics
Department of Computer Science
Rutgers University
Piscataway, NJ 08854, USA
pingli@stat.rutgers.edu

Abstract

Recently we showed that the problem of Maximum Inner Product Search (MIPS) is efficient and it admits provably sub-linear hashing algorithms. In [23], we used asymmetric transformations to convert the problem of approximate MIPS into the problem of approximate near neighbor search which can be efficiently solved using L2-LSH. In this paper, we revisit the problem of MIPS and argue that the quantizations used in L2-LSH is suboptimal for MIPS compared to signed random projections (SRP) which is another popular hashing scheme for cosine similarity (or correlations). Based on this observation, we provide different asymmetric transformations which convert the problem of approximate MIPS into the problem amenable to SRP instead of L2-LSH. An additional advantage of our scheme is that we also obtain LSH type space partitioning which is not possible with the existing scheme. Our theoretical analysis shows that the new scheme is significantly better than the original scheme for MIPS. Experimental evaluations strongly support the theoretical findings. In addition, we also provide the first empirical comparison that shows the superiority of hashing over tree based methods [21] for MIPS.

1 Introduction

In this paper, we revisit the problem of *Maximum Inner Product Search (MIPS)*, which was studied in our recent work [23]. In this work we present the first provably fast algorithm for MIPS, which was considered hard [21, 15]. Given an input query point $q \in \mathbb{R}^D$, the task of MIPS is to find $p \in \mathcal{S}$, where $\mathcal{S}$ is a giant collection of size N, which maximizes (approximately) the **inner product** q^Tp:

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

The MIPS problem is related to the problem of *near neighbor search (NNS)*. For example, L2-NNS

$$p = \arg\min_{x\in\mathcal{S}} ||q-x||_2^2 = \arg\min_{x\in\mathcal{S}}(||x||_2^2 - 2q^Tx) \tag{2}$$

or, correlation-NNS

$$p = \arg\max_{x\in\mathcal{S}} \frac{q^Tx}{\|q\|\|x\|} = \arg\max_{x\in\mathcal{S}} \frac{q^Tx}{\|x\|} \tag{3}$$

These three problems are equivalent if the norm of every element $x \in \mathcal{S}$ is constant. Clearly, the value of the norm $||q||_2$ has no effect for the argmax. In many scenarios, MIPS arises naturally at places where the norms of the elements in $\mathcal{S}$ have significant variations [15]. As reviewed in our prior work [23], examples of applications of MIPS include recommender system [16, 5, 15], large-scale object detection with DPM [9, 7, 14, 14], structural SVM [7], and multi-class label prediction [21, 15, 25].

Asymmetric LSH (ALSH): Locality Sensitive Hashing (LSH) [13] is popular in practice for efficiently solving NNS. In our prior work [23], the concept of "asymmetric LSH" (ALSH) was formalized and one can transform the input query $Q(p)$ and data in the collection $P(x)$ independently, where the transformations Q and P are different. In [23] we developed a particular set of transformations to convert MIPS into L2-NNS and then solved the problem by standard hashing i.e. L2-LSH [6]. In this paper, we name the scheme in [23] as **L2-ALSH**. Later we showed in [24] the flexibility and the power of the asymmetric framework developed in [23] by constructing a provably superior scheme for binary data. Prior to our work, asymmetry was applied for hashing higher order similarity [22], sketching [8], hashing different subspaces [3], and data dependent hashing [20] which unlike locality sensitive hashing do not come with provable runtime guarantees. Explicitly constructing asymmetric transformation tailored for a particular similarity, given an existing LSH, was the first observation made in [23] due to which MIPS, a sought after problem, became provably fast and practical.

It was argued in [17] that the quantizations used in traditional L2-LSH is suboptimal and it hurts the variance of the hashes. This raises a natural question that L2-ALSH which uses L2-LSH as a subroutine for solving MIPS could be suboptimal and there may be a better hashing scheme. We provide such a scheme in this work.

Our contribution: Based on the observation that the quantizations used in traditional L2-LSH is suboptimal, in this study, we propose another scheme for ALSH, by developing a new set of asymmetric transformations to convert MIPS into a problem of correlation-NNS, which is solved by "signed random projections" (SRP) [11, 4]. The new scheme thus avoids the use of L2-LSH. We name this new scheme as **Sign-ALSH**. Our theoretical analysis and experimental study show that Sign-ALSH is more advantageous than L2-ALSH for MIPS.

For inner products asymmetry is unavoidable. In case of L2-ALSH, due to asymmetry, we loose the capability to generate LSH like random data partitions for efficient clustering [12]. We show that for inner products with Sign-ALSH there is a novel formulation that allows us to generate such partitions for inner products. With existing L2-ALSH such formulation does not work.

Apart from providing a better hashing scheme, we also provide comparisons of the Sign-ALSH with cone trees [21]. Our empirical evaluations on three real datasets show that hashing based methods are superior over the tree based space partitioning methods. Since there is no existing comparison of hashing based methods with tree based methods for the problem of MIPS, we believe that the results shown in this work will be very valuable for practitioners.

2 Review: Locality Sensitive Hashing (LSH)

The problem of efficiently finding nearest neighbors has been an active research since the very early days of computer science [10]. Approximate versions of the near neighbor search problem [13] were proposed to break the linear query time bottleneck. The following formulation for approximate near neighbor search is often adopted.

Definition: (c-Approximate Near Neighbor or c-NN) *Given a set of points in a D-dimensional space $\mathbb{R}^D$, and parameters $S_0 > 0$, $\delta > 0$, construct a data structure which, given any query point q, does the following with probability $1 - \delta$: if there exists an S_0-near neighbor of q in $\mathcal{S}$, it reports some cS_0-near neighbor of q in $\mathcal{S}$.*

Locality Sensitive Hashing (LSH) [13] is a family of functions, with the property that more similar items have a higher collision probability. LSH trades off query time with extra (one time) preprocessing cost and space. Existence of an LSH family translates into provably sublinear query time algorithm for c-NN problems.

Definition: (Locality Sensitive Hashing (LSH)) *A family $\mathcal{H}$ is called (S_0, cS_0, p_1, p_2)-sensitive if, for any two points $x, y \in \mathbb{R}^D$, h chosen uniformly from $\mathcal{H}$ satisfies:*

- *if $Sim(x,y) \geq S_0$ then $Pr_{\mathcal{H}}(h(x) = h(y)) \geq p_1$*
- *if $Sim(x,y) \leq cS_0$ then $Pr_{\mathcal{H}}(h(x) = h(y)) \leq p_2$*

For efficient approximate nearest neighbor search, $p_1 > p_2$ and $c < 1$ is needed.

Fact 1: Given a family of (S_0, cS_0, p_1, p_2) -sensitive hash functions, one can construct a data structure for c-NN with $O(n^\rho \log n)$ query time and space $O(n^{1+\rho})$, where $\rho = \frac{\log p_1}{\log p_2} < 1$.

LSH is a generic framework and an implementation of LSH requires a concrete hash function.

2.1 LSH for L2 distance

[6] presented an LSH family for L_2 distances. Formally, given a fixed window size r, we sample a random vector a with each component from i.i.d. normal, i.e., $a_i \sim N(0,1)$, and a scalar b generated uniformly at random from $[0, r]$. The hash function is defined as:

$$h^{L2}_{a,b}(x) = \left\lfloor \frac{a^T x + b}{r} \right\rfloor \tag{4}$$

where $\lfloor\rfloor$ is the floor operation. The collision probability under this scheme can be shown to be

$$Pr(h^{L2}_{a,b}(x) = h^{L2}_{a,b}(y)) \tag{5}$$
$$= 1 - 2\Phi(-r/d) - \frac{2}{\sqrt{2\pi}(r/d)}\left(1 - e^{-(r/d)^2/2}\right) = F_r(d)$$

where $\Phi(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx$ and $d = \|x - y\|_2$ is the Euclidean distance between the vectors x and y.

2.2 LSH for correlation

Another popular LSH family is the so-called "sign random projections" [11, 4]. Again, we choose a random vector a with $a_i \sim N(0,1)$. The hash function is defined as:

$$h^{Sign}(x) = sign(a^T x) \tag{6}$$

And collision probability is

$$Pr(h^{Sign}(x) = h^{Sign}(y)) = 1 - \frac{1}{\pi}\cos^{-1}\left(\frac{x^T y}{\|x\|\|y\|}\right) \tag{7}$$

This scheme is known as *signed random projections (SRP)*.

3 Review of ALSH for MIPS and L2-ALSH

In [23], it was shown that the framework of locality sensitive hashing is restrictive for solving MIPS. The inherent assumption of the same hash function for both the transformation as well as the query was unnecessary in the classical LSH framework and it was the main hurdle in finding provable sub-linear algorithms for MIPS with LSH. For the theoretical guarantees of LSH to work there was no requirement of symmetry. Incorporating asymmetry in the hashing schemes was the key in solving MIPS efficiently.

Definition [23]: (*Asymmetric* Locality Sensitive Hashing (ALSH)) A family $\mathcal{H}$, along with the two vector functions $Q : \mathbb{R}^D \mapsto \mathbb{R}^{D'}$ (**Query Transformation**) and P :

$\mathbb{R}^D \mapsto \mathbb{R}^{D'}$ (**Preprocessing Transformation**), is called (S_0, cS_0, p_1, p_2)-sensitive if for a given c-NN instance with query q, and the hash function h chosen uniformly from $\mathcal{H}$ satisfies the following:

- if $Sim(q,x) \geq S_0$ then $Pr_{\mathcal{H}}(h(Q(q))) = h(P(x))) \geq p_1$
- if $Sim(q,x) \leq cS_0$ then $Pr_{\mathcal{H}}(h(Q(q)) = h(P(x))) \leq p_2$

Here x is any point in the collection $\mathcal{S}$.

Note that the query transformation Q is only applied on the query and the pre-processing transformation P is applied to $x \in \mathcal{S}$ while creating hash tables. By letting $Q(x) = P(x) = x$, we can recover the vanilla LSH. Using different transformations (i.e., $Q \neq P$), it is possible to counter the fact that self similarity is not highest with inner products which is the main argument of failure of LSH. We just need the probability of the new collision event $\{h(Q(q)) = h(P(y))\}$ to satisfy the conditions of definition of ALSH for $Sim(q,y) = q^Ty$.

Theorem 1 *[23] Given a family of hash function $\mathcal{H}$ and the associated query and preprocessing transformations P and Q, which is (S_0, cS_0, p_1, p_2) -sensitive, one can construct a data structure for c-NN with $O(n^\rho \log n)$ query time and space $O(n^{1+\rho})$, where $\rho = \frac{\log p_1}{\log p_2}$.*

[23] provided an explicit construction of ALSH, which we call **L2-ALSH**. Without loss of generality, one can assume

$$||x_i||_2 \leq U < 1, \quad \forall x_i \in \mathcal{S} \tag{8}$$

for some $U < 1$. If this is not the case, then we can always scale down the norms without altering the $\arg\max$. Since the norm of the query does not affect the $\arg\max$ in MIPS, for simplicity it was assumed $||q||_2 = 1$. This condition can be removed easily (see Section 5 for details). In L2-ALSH, two vector transformations $P: \mathbb{R}^D \mapsto \mathbb{R}^{D+m}$ and $Q: \mathbb{R}^D \mapsto \mathbb{R}^{D+m}$ are defined as follows:

$$P(x) = [x; ||x||_2^2; ||x||_2^4; ....; ||x||_2^{2^m}] \tag{9}$$
$$Q(x) = [x; 1/2; 1/2; ....; 1/2], \tag{10}$$

where [;] is the concatenation. $P(x)$ appends m scalers of the form $||x||_2^{2^i}$ at the end of the vector x, while Q(x) simply appends m "1/2" to the end of the vector x. By observing

$$||P(x_i)||_2^2 = ||x_i||_2^2 + ||x_i||_2^4 + ... + ||x_i||_2^{2^m} + ||x_i||_2^{2^{m+1}}$$
$$||Q(q)||_2^2 = ||q||_2^2 + m/4 = 1 + m/4$$
$$Q(q)^TP(x_i) = q^Tx_i + \frac{1}{2}(||x_i||_2^2 + ||x_i||_2^4 + ... + ||x_i||_2^{2^m})$$

one can obtain the following key equality:

$$||Q(q) - P(x_i)||_2^2 = (1 + m/4) - 2q^Tx_i + ||x_i||_2^{2^{m+1}} \tag{11}$$

Since $||x_i||_2 \leq U < 1$, we have $||x_i||^{2^{m+1}} \to 0$ at the tower rate (exponential to exponential). Thus, as long as m is not too small (e.g., $m \geq 3$ would suffice), we have

$$\arg\max_{x \in \mathcal{S}} q^Tx \simeq \arg\min_{x \in \mathcal{S}} ||Q(q) - P(x)||_2 \tag{12}$$

This scheme is the first connection between solving un-normalized MIPS and approximate near neighbor search. Transformations P and Q, when norms are less than 1, provide correction to the L2 distance $||Q(q) - P(x_i)||_2$ making it rank correlate with the (un-normalized) inner product.

3.1 Intuition for the Better Scheme : Why Signed Random Projections (SRP)?

Recently in [17, 18], it was observed that the quantization of random projections used by traditional L2-LSH scheme is not desirable when the data is normalized and in fact the shift b in Eq. (4) hurts the variance leading to less informative hashes. The sub-optimality of L2-LSH hints towards existence of better hashing functions for MIPS.

As previously argued, when the data are normalized then both L2-NNS and correlation-NNS are equivalent to MIPS. Therefore, for normalized data we can use either L2-LSH which is popular LSH for L2 distance or SRP which is popular LSH for correlation to solve MIPS directly. This raises a natural question "Which will perform better ?".

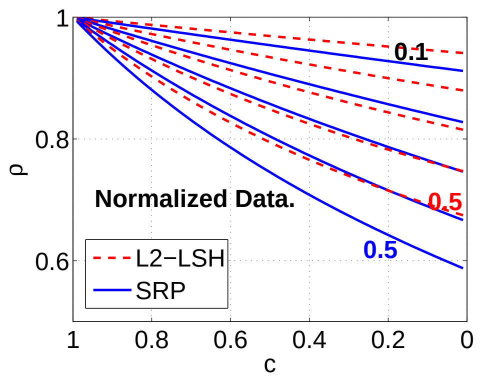
If we assume that the data are normalized, i.e., all the norms are equal to 1, then both SRP and L2-LSH are monotonic in the inner product and their corresponding ρ values for retrieving max inner product can be computed as

$$\rho_{SRP} = \frac{\log\left(1 - \frac{1}{\pi}\cos^{-1}(S_0)\right)}{\log\left(1 - \frac{1}{\pi}\cos^{-1}(cS_0)\right)} \tag{13}$$

$$\rho_{L2-LSH} = \frac{\log\left(F_r(\sqrt{2 - 2S_0})\right)}{\log\left(F_r(\sqrt{2 - 2cS_0})\right)} \tag{14}$$

where the function $F_r(.)$ is given by Eq. (5). The values of ρ_{SRP} and ρ_{L2-LSH} for different $S_0 = \{0.1, 0.2, .., 0.9, 0.95\}$ with respect to approximation ratio c is shown in Figure 1. We use standard recommendation of $r = 2.5$ for L2-LSH. We can clearly see that ρ_{SRP} is consistently better than ρ_{L2-LSH} given any S_0 and c. Thus, for MIPS with normalized data L2-LSH type of quantization given by equation 5 seems suboptimal. It is clear that when the data is normalized then SRP is always a better choice for MIPS as compared to L2-LSH. This motivates us to explore the possibility of better hashing algorithm for general (unnormalized) instance of MIPS using SRP, which will have impact in many applications as pointed out in [23].

Asymmetric transformations give us enough flexibility to modify norms without changing inner products. The transformations provided in [23] used this flexibility to convert MIPS to standard near neighbor search in L_2 space for which we have standard hash functions. For binary data, [24] showed a strictly superior construction, the asymmetric minwise hashing, which outperforms all ALSHs made for general MIPS.



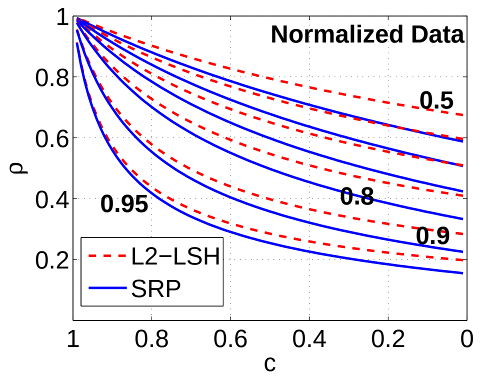


Figure 1: Values of ρ_{SRP} and ρ_{L2-LSH} (Lower is better) for normalized data. It is clear that SRP is more suited for retrieving inner products when the data is normalized

Signed random projections are popular hash functions widely adopted for correlation or cosine similarity. We use asymmetric transformations to convert approximate MIPS into approximate maximum correlation search and thus we avoid the use of sub-optimal L2-LSH. The collision probability of the hash functions is one of the key constituents which determine the efficiency of the obtained ALSH algorithm. We show that our proposed transformation with SRP is better suited for ALSH compared to the existing L2-ALSH for solving general MIPS instance.

4 The New Proposal: Sign-ALSH

4.1 From MIPS to Correlation-NNS

We assume for simplicity that $||q||_2 = 1$ as the norm of the query does not change the ordering, we show in the next section how to get rid of this assumption. Without loss of generality let $||x_i||_2 \leq U < 1, \quad \forall x_i \in \mathcal{S}$ as it can always be achieved by scaling the data by large enough number. We define two vector transformations $P : \mathbb{R}^D \mapsto \mathbb{R}^{D+m}$ and $Q : \mathbb{R}^D \mapsto \mathbb{R}^{D+m}$ as follows:

$$P(x) = [x; 1/2 - ||x||_2^2; 1/2 - ||x||_2^4; ....; 1/2 - ||x||_2^{2^m}] \tag{15}$$

$$Q(x) = [x; 0; 0; ....; 0], \tag{16}$$

Using $||Q(q)||_2^2 = ||q||_2^2 = 1$, $Q(q)^T P(x_i) = q^T x_i$, and

$$\begin{aligned}&||P(x_i)||_2^2\\ &= ||x_i||_2^2 + 1/4 + ||x_i||_2^4 - ||x_i||_2^2 + 1/4 + ||x_i||_2^8 - ||x_i||_2^4 + ...\\ &+ 1/4 + ||x_i||_2^{2^{m+1}} - ||x_i||_2^{2^m}\\ &= m/4 + ||x_i||_2^{2^{m+1}}\end{aligned}$$

we obtain the following key equality:

$$\frac{Q(q)^T P(x_i)}{\|Q(q)\|_2 \|P(x_i)\|_2} = \frac{q^T x_i}{\sqrt{m/4 + ||x_i||_2^{2^{m+1}}}} \tag{17}$$

The term $||x_i||^{2^{m+1}} \to 0$, again vanishes at the tower rate. This means we have approximately

$$\arg\max_{x \in \mathcal{S}} q^T x \simeq \arg\max_{x \in \mathcal{S}} \frac{Q(q)^T P(x_i)}{\|Q(q)\|_2 \|P(x_i)\|_2} \tag{18}$$

This provides another solution for solving MIPS using known methods for approximate correlation-NNS. Asymmetric transformations P and Q provide a lot of flexibility. Note that transformations P and Q are not unique for this task and there are other possibilities [2, 19]. It should be further noted that even scaling data and query differently is asymmetry in a strict sense because it changes the distribution of the hashes. Flexibility in choosing the transformations P and Q allow us to use signed random projections and thereby making possible to avoid suboptimal L2-LSH.

4.2 Fast MIPS Using Sign Random Projections

Eq. (18) shows that MIPS reduces to the standard approximate near neighbor search problem which can be efficiently solved by sign random projections, i.e., h^{Sign} (defined by Eq. (6)). Formally, we can state the following theorem.

Theorem 2 *Given a c-approximate instance of MIPS, i.e., $Sim(q,x) = q^T x$, and a query q such that $||q||_2 = 1$ along with a collection $\mathcal{S}$ having $||x||_2 \leq U < 1 \; \forall x \in \mathcal{S}$. Let P and Q be the vector transformations defined in Eq. (15) and Eq. (16), respectively. We have the following two conditions for hash function h^{Sign} (defined by Eq. (6))*

- *if $q^T x \geq S_0$ then*

$$\begin{aligned}&Pr[h^{Sign}(Q(q)) = h^{Sign}(P(x))]\\ &\geq 1 - \frac{1}{\pi}\cos^{-1}\left(\frac{S_0}{\sqrt{m/4 + U^{2^{m+1}}}}\right)\end{aligned}$$

- *if $q^T x \leq cS_0$ then*

$$\begin{aligned}&Pr[h^{Sign}(Q(q)) = h^{Sign}(P(x))]\\ &\leq 1 - \frac{1}{\pi}\cos^{-1}\left(\frac{\min\{cS_0, z^*\}}{\sqrt{m/4 + (\min\{cS_0, z^*\})^{2^{m+1}}}}\right)\end{aligned}$$

where $z^* = \left(\frac{m/2}{2^{m+1}-2}\right)^{2^{-m-1}}$.

Proof: *When $q^T x \geq S_0$, we have, according to Eq. (7)*

$$\begin{aligned}
&Pr[h^{Sign}(Q(q)) = h^{Sign}(P(x))] \\
&= 1 - \frac{1}{\pi}\cos^{-1}\left(\frac{q^T x}{\sqrt{m/4 + \|x\|_2^{2^{m+1}}}}\right) \\
&\geq 1 - \frac{1}{\pi}\cos^{-1}\left(\frac{q^T x}{\sqrt{m/4 + U^{2^{m+1}}}}\right)
\end{aligned}$$

When $q^T x \leq cS_0$, by noting that $q^T x \leq \|x\|_2$, we have

$$\begin{aligned}
&Pr[h^{Sign}(Q(q)) = h^{Sign}(P(x))] \\
&= 1 - \frac{1}{\pi}\cos^{-1}\left(\frac{q^T x}{\sqrt{m/4 + \|x\|_2^{2^{m+1}}}}\right) \\
&\leq 1 - \frac{1}{\pi}\cos^{-1}\left(\frac{q^T x}{\sqrt{m/4 + (q^T x)^{2^{m+1}}}}\right)
\end{aligned}$$

For this one-dimensional function $f(z) = \frac{z}{\sqrt{a+z^b}}$, where $z = q^T x$, $a = m/4$ and $b = 2^{m+1} \geq 2$, we know

$$f'(z) = \frac{a - z^b(b/2 - 1)}{(a + z^b)^{3/2}}$$

One can also check that $f''(z) \leq 0$ for $0 < z < 1$, i.e., $f(z)$ is a concave function. The maximum of $f(z)$ is attained at $z^* = \left(\frac{2a}{b-2}\right)^{1/b} = \left(\frac{m/2}{2^{m+1}-2}\right)^{2^{-m-1}}$ If $z^* \geq cS_0$, then we need to use $f(cS_0)$ as the bound. □

Therefore, we have obtained, in LSH terminology,

$$p_1 = 1 - \frac{1}{\pi}\cos^{-1}\left(\frac{S_0}{\sqrt{m/4 + U^{2^{m+1}}}}\right) \tag{19}$$

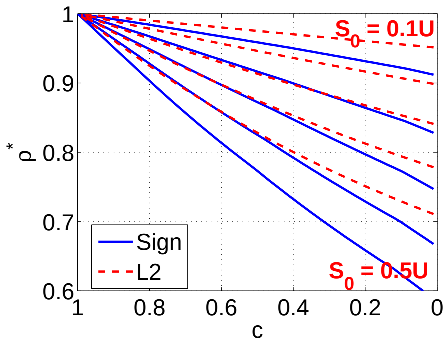
$$p_2 = 1 - \frac{1}{\pi}\cos^{-1}\left(\frac{\min\{cS_0, z^*\}}{\sqrt{m/4 + (\min\{cS_0, z^*\})^{2^{m+1}}}}\right), \tag{20}$$

$$z^* = \left(\frac{m/2}{2^{m+1} - 2}\right)^{2^{-m-1}} \tag{21}$$

Theorem 1 allows us to construct data structures with worst case $O(n^\rho \log n)$ query time guarantees for c-approximate MIPS, where $\rho = \frac{\log p_1}{\log p_2}$. For any given $c < 1$, there always exist $U < 1$ and m such that $\rho < 1$. This way, we obtain a sublinear query time algorithm for MIPS. Because ρ is a function of 2 parameters, the best query time chooses U and m, which minimizes the value of ρ. For convenience, we define

$$\rho^* = \min_{U,m} \frac{\log\left(1 - \frac{1}{\pi}\cos^{-1}\left(\frac{S_0}{\sqrt{m/4+U^{2^{m+1}}}}\right)\right)}{\log\left(1 - \frac{1}{\pi}\cos^{-1}\left(\frac{\min\{cS_0,z^*\}}{\sqrt{m/4+(\min\{cS_0,z^*\})^{2^{m+1}}}}\right)\right)} \tag{22}$$

See Figure 2 for the plots of ρ^*, which also compares the optimal ρ values for L2-ALSH in the prior work [23]. The results show that Sign-ALSH is noticeably better.



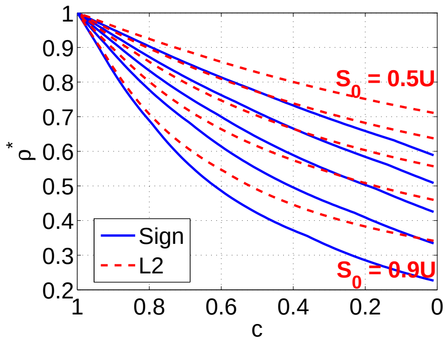


Figure 2: Optimal values of ρ^* (lower is better) with respect to approximation ratio c for different S_0, obtained by a grid search over parameters U and m, given S_0 and c. The curves show that Sign-ALSH (solid curves) is noticeably better than L2-ALSH (dashed curves) in terms of their optimal ρ^* values. The results for L2-ALSH were from the prior work [23]. For clarity, the results are in two figures.

4.3 Parameter Selection

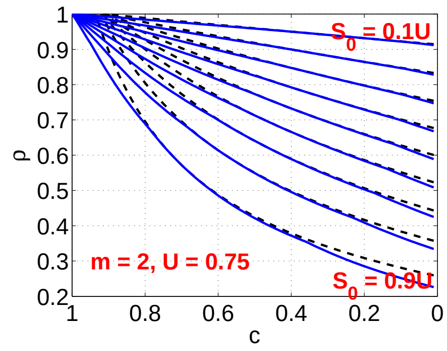


Figure 3: The solid curves are the optimal ρ values of Sign-ALSH from Figure 2. The dashed curves represent the ρ values for fixed parameters: $m = 2$ and $U = 0.75$ (left panel). Even with fixed parameters, the ρ does not degrade.

Figure 3 presents the ρ values for $(m, U) = (2, 0.75)$ We can see that even if we use fixed parameters, the performance would only degrade little. This essentially frees practitioners from the burden of choosing parameters.

5 Removing Dependency on Norm of Query

Changing norms of the query does not affect the $\arg\max_{x\in\mathcal{C}} q^T x$, and hence, in practice for retrieving top-k, normalizing the query should not affect the performance. But for theoretical purposes, we want the runtime guarantee to be independent of $||q||_2$. Note, both LSH and ALSH schemes solve the c-approximate instance of the problem, which requires a threshold $S_0 = q^T x$ and an approximation ratio c. These quantities change if we change the norms. We can use the same idea used in [23] to get rid of the norm of q. Transformations P and Q were precisely meant to remove the dependency of correlation on the norms of x but at the same time keeping the inner products same. Let M be the upper bound on all the norms i.e. $M = max_{x\in\mathcal{C}}||x||_2$. In other words M is the radius of the space.

Let $U < 1$, define the transformations, $T : \mathbb{R}^D \to \mathbb{R}^D$ as

$$T(x) = \frac{Ux}{M} \tag{23}$$

and transformations $P, Q : \mathbb{R}^D \to \mathbb{R}^{D+m}$ are the same for the Sign-ALSH scheme as defined in Eq (15) and (16).

Given the query q and any data point x, observe that the inner products between $P(Q(T(q)))$ and $Q(P(T(x)))$ is

$$P(Q(T(q)))^T Q(P(T(x))) = q^T x \times \left(\frac{U^2}{M^2}\right) \tag{24}$$

$P(Q(T(q)))$ appends first m zeros components to $T(q)$ and then m components of the form $1/2 - ||q||^{2^i}$. $Q(P(T(q)))$ does the same thing but in a different order. Now we are working in $D + 2m$ dimensions. It is not difficult to see that the norms of $P(Q(T(q)))$ and $Q(P(T(q)))$ is given by

$$||P(Q(T(q)))||_2 = \sqrt{\frac{m}{4} + ||T(q)||_2^{2^{m+1}}} \tag{25}$$

$$||Q(P(T(x)))||_2 = \sqrt{\frac{m}{4} + ||T(x)||_2^{2^{m+1}}} \tag{26}$$

The transformations are very asymmetric but we know that it is necessary.

Therefore the correlation or the cosine similarity between $P(Q(T(q)))$ and $Q(P(T(x)))$ is

$$Corr = \frac{q^T x \times \left(\frac{U^2}{M^2}\right)}{\sqrt{\frac{m}{4} + ||T(q)||_2^{2^{m+1}}}\sqrt{\frac{m}{4} + ||T(x)||_2^{2^{m+1}}}} \tag{27}$$

Note $||T(q)||_2^{2^{m+1}}, ||T(x)||_2^{2^{m+1}} \le U < 1$, therefore both $||T(q)||_2^{2^{m+1}}$ and $||T(x)||_2^{2^{m+1}}$ converge to zero at a tower rate and we get approximate monotonicity of correlation with the inner products. We can apply sign random projections to hash $P(Q(T(q)))$ and $Q(P(T(q)))$.

As $0 \le ||T(q)||_2^{2^{m+1}} \le U$ and $0 \le ||T(x)||_2^{2^{m+1}} \le U$, it is not difficult to get p_1 and p_2 for Sign-ALSH, without conditions on any norms. Simplifying the expression, we get the following value of optimal ρ_u (u for unrestricted).

$$\rho_u^* = \min_{U,m,} \frac{\log\left(1 - \frac{1}{\pi}\cos^{-1}\left(\frac{S_0 \times \left(\frac{U^2}{M^2}\right)}{\frac{m}{4} + U^{2^{m+1}}}\right)\right)}{\log\left(1 - \frac{1}{\pi}\cos^{-1}\left(\frac{cS_0 \times \left(\frac{4U^2}{M^2}\right)}{m}\right)\right)} \tag{28}$$

$$s.t.\quad U^{2^{m+1}} < \frac{m(1-c)}{4c},\ m \in \mathbb{N}^+, \text{ and } 0 < U < 1.$$

With this value of ρ_u^*, we can state our main theorem.

Theorem 3 *For the problem of c-approximate MIPS in a bounded space, one can construct a data structure having $O(n^{\rho_u^*} \log n)$ query time and space $O(n^{1+\rho_u^*})$, where $\rho_u^* < 1$ is the solution to constraint optimization (28).*

Note, for all $c < 1$, we always have $\rho_u^* < 1$ because the constraint $U^{2^{m+1}} < \frac{m(1-c)}{4c}$ is always true for big enough m. The only assumption for efficiently solving MIPS that we need is that the space is bounded, which is always satisfied for any finite dataset. ρ_u^* depends on M, the radius of the space, which is expected.

6 Random Space Partitioning for Inner Product

In this section, we show that due to the nature of the new transformations P and Q there is one subtle but surprising advantage of Sign-ALSH over L2-ALSH.

One popular application of LSH (Locality Sensitive Hashing) is random partitioning of the data for large scale clustering, where similar points map to the same partition (or bucket). Such partitions are very useful in many applications [12]. With classical LSH, we simply use $h(x)$ to generate partition for x. Since $Pr_{\mathcal{H}}(h(x) = h(y))$ is high if $sim(x, y)$ is high, similar points are likely to go into the same partition under the usual LSH mapping. For general ALSH, this property is lost because of asymmetry.

In case of ALSH, we only know that $Pr(h(P(x)) = h(Q(y))$ is high if $sim(x, y)$ is high. Therefore, given x we cannot determine whether to assign partition using $h(P(.))$ or $h(Q(.))$. Neither $Pr(h(P(x)) = h(P(y))$ nor $Pr_{\mathcal{H}}(h(Q(x)) = h(Q(y))$ strictly indicates high value of $sim(x, y)$ in general. Therefore, partitioning property of classical LSH does not hold anymore with general ALSHs. However for the case of inner products using Sign-ALSH, there is a subtle observation which allows us to construct the required assignment function, where pairs of points with high inner products are more likely to get mapped in the same partition while pairs with low inner products are more likely to map into different partitions.

In case of Sign-ALSH for MIPS, we have the transformations $P(Q(T(x)))$ and $Q(P(T(x)))$ given by

$$P(Q(T(x))) = [x; 1/2 - ||T(x)||_2^2; ....; 1/2 - ||T(x)||_2^{2^m}, 0, ..., 0]$$

$$Q(P(T(x))) = [x; 0, ..., 0, 1/2 - ||T(x)||_2^2; ....; 1/2 - ||T(x)||_2^{2^m}].$$

After this transformation, we multiply the generated $D + 2m$ dimensional vector by a random vector $a \in \mathbb{R}^{D+2m}$ whose entries are i.i.d. Gaussian followed by taking the sign. For illustration let $a = [w; s_1, ...s_m, t_1, ...t_m]$ where $w \in \mathbb{R}^D$ b_i and c_i are numbers. All components of a are i.i.d. from $N(0,1)$. With this notation, we can write the final Sign-ALSH as

$$h^{Sign}(P(Q(T(x)))) = Sign(w^T T(x) + \sum_{i=1}^{m} s_i(1/2 - ||T(x)||_2^{2^i}))$$

$$h^{Sign}(Q(P(T(x)))) = Sign(w^T T(x) + \sum_{i=1}^{m} t_i(1/2 - ||T(x)||_2^{2^i}))$$

The key observation here is that $h^{Sign}(P(Q(T(x))))$ does not depend on t_i and $h^{Sign}(Q(P(T(x))))$ does not depend on s_i. If we define

$$h_w(x) = Sign(w^T T(x) + \sum_{i=1}^{m} \alpha_i(1/2 - ||T(x)||_2^{2^i})) \quad (29)$$

where α_i are sampled i.i.d. from $N(0,1)$ for every x independently of everything else. Then, **under the randomization of** w, it is not difficult to show that

$$Pr_w(h_w(x) = h_w(y)) = Pr(h^{Sign}(P(x)) = h^{Sign}(Q(y)))$$

for any x, y. The term $Pr(h^{Sign}(P(x)) = h^{Sign}(Q(y)))$ satisfies the LSH like property and therefore, in any partitions using h_w, points with high inner products are more likely to be together. Thus, $h_w(x)$ is the required assignment. Note, h_w is not technically an LSH because we are randomly sampling α_i for all x independently. The construction of h_w using independent randomizations could be of separate interest. To the best of our knowledge, this is the first example of LSH like partition using hash function with independent randomization for every data point.

The function h_w is little subtle here, we sample w i.i.d from Gaussian and use the same w for all x, but while computing h_w we use α_i independent of everything for every x. The probability is under the randomization of w and independence of all α_i ensures the asymmetry. We are not sure if such construction is possible with L2-ALSH. For LSH partitions with binary data, the idea used here can be applied on asymmetric minwise hashing [24].

7 Ranking Evaluations

In [23], the L2-ALSH scheme was shown to outperform other reasonable heuristics in retrieving maximum inner products. Since our proposal is an improvement over L2-ALSH, in this section we first present comparisons with L2-ALSH, in particular on ranking experiments.

7.1 Datasets

We use three publicly available dataset MNIST, WEBSPAM and RCV1 for evaluations. For each of the three dataset we generate two independent partitions, the query set and the train set. Each element in the query set is used for querying, while the training set serves as the collection $\mathcal{C}$ that will be searched for MIPS. The statistics of the dataset and the partitions are summarized in Table 1

Dataset	Dimension	Query size	Train size
MNIST	784	10,000	60,000
WEBSPAM	16,609,143	5,000	100,000
RCV1	47,236	5,000	100,000

Table 1: Datasets used for evaluations.

7.2 Evaluations

In this section, we show how the ranking of the two ALSH schemes, L2-ALSH and Sign-ALSH, correlates with inner products. Given a query vector q, we compute the top-10 gold standard elements based on the actual inner products $q^T x$, $\forall x \in \mathcal{C}$, here our collection is the train set. We then generate K different hash codes of the query q and all the elements $x \in \mathcal{C}$ and then compute

$$Matches_x = \sum_{t=1}^{K} \mathbf{1}(h_t(Q(q)) = h_t(P(x))), \quad (30)$$

where $\mathbf{1}$ is the indicator function and the subscript t is used to distinguish independent draws of h. Based on $Matches_x$ we rank all the elements x. Ideally, for a better hashing scheme, $Matches_x$ should be higher for element x having higher inner products with the given query q. This procedure generates a sorted list of all the items for a given query vector q corresponding to the each of the two asymmetric hash functions under consideration.

For L2-ALSH, we used the same parameters used and recommended in [23]. For Sign-ALSH, we used the recommended choice shown in Section 4.3, which is $U = 0.75$, $m = 2$. Note that Sign-ALSH does not have parameter r.

We compute precision and recall of the top-10 gold standard elements, obtained from the sorted list based on $Matches_x$. To compute this precision and recall, we start at the top of the ranked item list and walk down in order. Suppose we are at the k^{th} ranked item, we check if this element belongs to the gold standard top-10 list. If it is one of the top-10 gold standard elements, then we increment the count of *relevant seen* by 1, else we move to $k + 1$. By k^{th} step, we have already seen k elements, so the *total items seen* is k. The precision and recall at that point are

$$Precision = \frac{\text{relevant seen}}{k}, \qquad Recall = \frac{\text{relevant seen}}{10}$$

We show performance for $K \in \{64, 128, 256, 512\}$. Note that it is important to balance both precision and recall. The

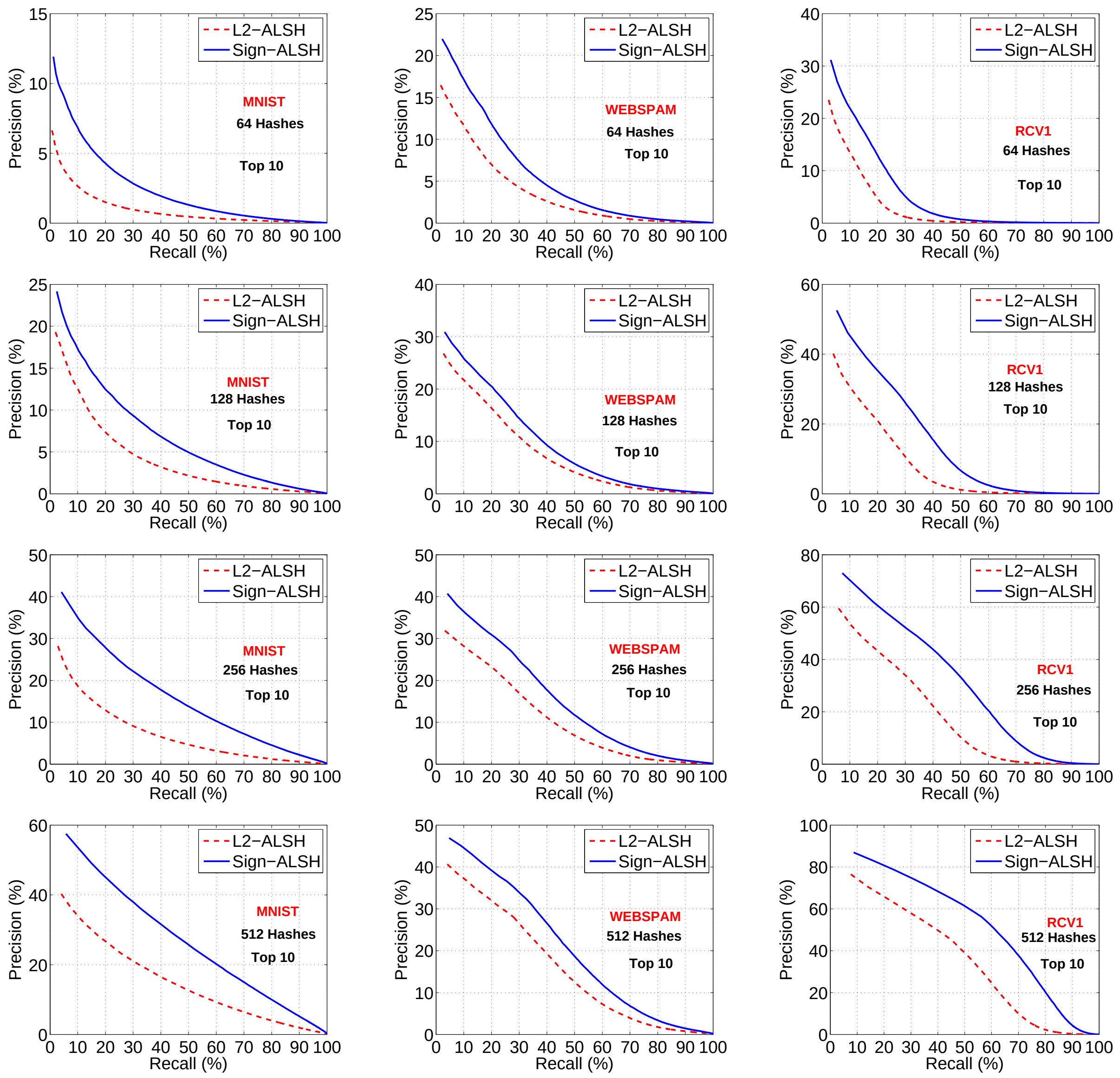


Figure 4: Precision-Recall curves (higher is better). We compare L2-ALSH (using parameters recommended in [23]) with our proposed Sign-ALSH using $(m = 2, U = 0.75)$ for retrieving top-10 elements. Sign-ALSH is noticeably better.

method which obtains higher precision at a given recall is superior. Higher precision indicates higher ranking of the top-10 inner products which is desirable. We report averaged precisions and recalls.

The plots for all the three datasets are shown in Figure 4. We can clearly see, that our proposed Sign-ALSH scheme gives significantly higher precision recall curves than the L2-ALSH scheme, indicating better correlation of top inner products with Sign-ALSH compared to L2-ALSH. The results are consistent across datasets.

8 Comparisons of Hashing Based and Tree Based Methods for MIPS

We have shown in the previous Section that Sign-ALSH outperforms L2-ALSH in ranking evaluations. In this Section, we consider the actual task of finding the maximum inner product. Our aim is to estimate the computational saving, in finding the maximum inner product, with Sign-ALSH compared to the existing scheme L2-ALSH. In addition to L2-ALSH which is a hashing scheme, there is an another tree based space partitioning method [21] for solving MIPS. Although, in theory, it is know that tree based methods perform poorly [25] due to their exponential dependence on the dimensionality, it is still important to understand the impact of such dependency in practice. Unfortunately no empirical comparison between hashing and tree based methods exists for the problem of MIPS in the literature. To provide such a comparison, we also consider tree based space partitioning method [21] for evaluations. We use the same three datasets as described in Section 7.1.

Tree based and hashing based methodologies are very different in nature. The major difference is in the stopping criteria. Hashing based methods create buckets and stop the search once they find a good enough point, they may not succeed with some probability. On the other hand, tree based methods use branch and bound criteria to stop exploring further. So it is possible that a tree based algorithm finds the optimal point but continues to explore further requiring more computations. The usual stopping criteria thus makes tree based methods unnecessarily expensive compared to hashing based methods where the criteria is to stop after finding a good point. Therefore, to ensure fair comparisons, we allow the tree based method to stop the evaluations immediately once the algorithm finds the maximum inner product and prevent it from exploring further. Also, in case when hashing based algorithm fails to find the best inner product we resort to the full linear scan and penalize the hashing based algorithm for not succeeding. All this is required to ensure that tree based algorithm is not at any disadvantage compare to hashing methods.

We implemented the bucketing scheme with Sign-ALSH and L2-ALSH. The bucketing scheme requires creating many hash tables during the preprocessing stage. During query phase, given a query, we compute many hashes of the query and probe appropriate buckets in each table. Please refer [1] for more details on the process. We use the same fixed parameters for all the evaluations, i.e., (m=2, U=0.75) for Sign-ALSH and (m=3, U=0.83, r=2.5) for L2-ALSH as recommended in [23]. The total number of inner products evaluated by a hashing scheme, for a given query, is the total number of hash computation for the query plus the total number of points retrieved from the hash tables. In rare cases, with very small probability, if the hash tables are unable to retrieve the gold standard maximum inner product, we resort to linear scan and also include the total number of inner products computed during the linear scan. We stop as soon as we reach the gold standard point.

We implemented Algorithm 5 from [21], which is the best performing algorithm as shown in the evaluations. For this algorithm, we need to select one parameter which is the minimum number of elements in the node required for splitting. We found that on all the three datasets the value of 100 for this parameter works the best among $\{500, 200, 100, 50\}$. Therefore, we use 100 in all our experiments. The total number of inner products evaluated by tree based algorithm is the total number of points reported plus the total number of nodes visited, where we compute the branch and bound constraint. Again we stop the search process as soon as we reach the point with gold standard maximum inner product. As argued, we need this common stopping condition to compare with hashing based methods, where we do not have any other stopping criteria [13].

For every query we compute the number of inner products evaluated by different methods for MIPS. We report the mean of the total number of inner products evaluated per query in Table 2. We can clearly see that hashing based

	Sign-ALSH	L2-ALSH	Cone Trees
MNIST	**7,944**	9,971	11,202
WEBSPAM	**2,866**	3,813	22,467
RCV1	**9,951**	11,883	38,162

Table 2: Average number of inner products evaluated per query by different MIPS algorithms. Both Sign-ALSH and L2-ALSH [23] outperform cone trees [21]. Sign-ALSH is always superior compared to L2-ALSH for MIPS.

methods are always better than the tree based algorithm. Except on MNIST dataset, hashing based methods are significantly superior, which is also not surprising because MNIST is an image dataset having low intrinsic dimensionality. Among the two hashing schemes Sign-ALSH is always better than L2-ALSH, which verifies our theoretical findings and supports our arguments in favor of Sign-ALSH over L2-ALSH for MIPS.

9 Conclusion

The MIPS (maximum inner product search) problem has numerous important applications in machine learning, databases, and information retrieval. [23] developed the framework of Asymmetric LSH and provided an explicit scheme (L2-ALSH) for approximate MIPS in sublinear time. L2-ALSH uses L2-LSH as a subroutine which uses suboptimal quantizations. In this study, we present another asymmetric transformation scheme (Sign-ALSH) which converts the problem of maximum inner products into the problem of maximum correlation search, which is subsequently solved by sign random projections, thereby avoiding the use of L2-LSH.

Theoretical analysis and experimental study demonstrate that **Sign-ALSH** can be noticeably more advantageous than **L2-ALSH**. The new transformations with Sign-ALSH can be adapted to generate LSH like random data partitions which is very useful for large scale clustering. Such an adaptation is not possible with existing L2-ALSH. This was a rather unexpected advantage of the proposed Sign-ALSH over L2-ALSH. We also establish by experiments that hashing based algorithms are superior to tree based space partitioning methods for MIPS.

It should be noted that for MIPS over binary data our recent work asymmetric minwise hashing [24] should be used. We showed that for binary domain asymmetric minwise hashing is both empirically and provably superior, please see [24] for more details.

10 Acknowledgement

The work is partially supported by NSF-III-1360971, NSF-Bigdata-1419210, ONR-N00014-13-1-0764, and AFOSR-FA9550-13-1-0137. We would like to thank the reviewers of AISTATS 2015 and UAI 2015. We also thank Sanjiv Kumar and Hadi Daneshmand for pleasant discussions.

References

[1] A. Andoni and P. Indyk. E2lsh: Exact euclidean locality sensitive hashing. Technical report, 2004.

[2] Y. Bachrach, Y. Finkelstein, R. Gilad-Bachrach, L. Katzir, N. Koenigstein, N. Nice, and U. Paquet. Speeding up the xbox recommender system using a euclidean transformation for inner-product spaces. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, 2014.

[3] R. Basri, T. Hassner, and L. Zelnik-Manor. Approximate nearest subspace search with applications to pattern recognition. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[4] M. S. Charikar. Similarity estimation techniques from rounding algorithms. In *STOC*, pages 380–388, Montreal, Quebec, Canada, 2002.

[5] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 39–46. ACM, 2010.

[6] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokn. Locality-sensitive hashing scheme based on p-stable distributions. In *SCG*, pages 253 – 262, Brooklyn, NY, 2004.

[7] T. Dean, M. A. Ruzon, M. Segal, J. Shlens, S. Vijayanarasimhan, and J. Yagnik. Fast, accurate detection of 100,000 object classes on a single machine. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 1814–1821. IEEE, 2013.

[8] W. Dong, M. Charikar, and K. Li. Asymmetric distance estimation with sketches for similarity search in high-dimensional spaces. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 123–130. ACM, 2008.

[9] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1627–1645, 2010.

[10] J. H. Friedman and J. W. Tukey. A projection pursuit algorithm for exploratory data analysis. *IEEE Transactions on Computers*, 23(9):881–890, 1974.

[11] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of ACM*, 42(6):1115–1145, 1995.

[12] T. H. Haveliwala, A. Gionis, and P. Indyk. Scalable techniques for clustering the web. In *WebDB*, pages 129–134, 2000.

[13] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *STOC*, pages 604–613, Dallas, TX, 1998.

[14] T. Joachims, T. Finley, and C.-N. J. Yu. Cutting-plane training of structural svms. *Machine Learning*, 77(1):27–59, 2009.

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

[16] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems.

[17] P. Li, M. Mitzenmacher, and A. Shrivastava. Coding for random projections. In *ICML*, 2014.

[18] P. Li, M. Mitzenmacher, and A. Shrivastava. Coding for random projections and approximate near neighbor search. Technical report, arXiv:1403.8144, 2014.

[19] B. Neyshabur and N. Srebro. On symmetric and asymmetric lshs for inner product search. Technical report, arXiv:1410.5518, 2014.

[20] B. Neyshabur, N. Srebro, R. R. Salakhutdinov, Y. Makarychev, and P. Yadollahpour. The power of asymmetry in binary hashing. In *Advances in Neural Information Processing Systems*, pages 2823–2831, 2013.

[21] P. Ram and A. G. Gray. Maximum inner-product search using cone trees. In *KDD*, pages 931–939, 2012.

[22] A. Shrivastava and P. Li. Beyond pairwise: Provably fast algorithms for approximate k-way similarity search. In *NIPS*, Lake Tahoe, NV, 2013.

[23] A. Shrivastava and P. Li. Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips). In *NIPS*, Montreal, CA, 2014.

[24] A. Shrivastava and P. Li. Asymmetric minwise hashing for indexing binary inner products and set containment. In *WWW*, 2015.

[25] R. Weber, H.-J. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proceedings of the 24rd International Conference on Very Large Data Bases*, VLDB '98, pages 194–205, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

Asymmetric Minwise Hashing for Indexing Binary Inner Products and Set Containment

Anshumali Shrivastava
Department of Computer Science
Computer and Information Science
Cornell University
Ithaca, NY 14853, USA
anshu@cs.cornell.edu

Ping Li
Department of Statistics and Biostatistics
Department of Computer Science
Rutgers University
Piscataway, NJ 08854, USA
pingli@stat.rutgers.edu

ABSTRACT

Minwise hashing (Minhash) is a widely popular indexing scheme in practice. Minhash is designed for estimating set resemblance and is known to be suboptimal in many applications where the desired measure is set overlap (i.e., inner product between binary vectors) or set containment. Minhash has inherent bias towards smaller sets, which adversely affects its performance in applications where such a penalization is not desirable. In this paper, we propose asymmetric minwise hashing (*MH-ALSH*), to provide a solution to this well-known problem. The new scheme utilizes asymmetric transformations to cancel the bias of traditional minhash towards smaller sets, making the final "collision probability" monotonic in the inner product. Our theoretical comparisons show that, for the task of retrieving with binary inner products, asymmetric minhash is provably better than traditional minhash and other recently proposed hashing algorithms for general inner products. Thus, we obtain an algorithmic improvement over existing approaches in the literature. Experimental evaluations on four publicly available high-dimensional datasets validate our claims. The proposed scheme outperforms, often significantly, other hashing algorithms on the task of near neighbor retrieval with set containment. Our proposal is simple and easy to implement in practice.

1. INTRODUCTION

Record matching (or linkage), data cleansing and plagiarism detection are among the most frequent operations in many large-scale data processing systems over the web. *Minwise hashing* (or minhash) [6, 7, 27] is a popular technique deployed by big data industries for these tasks. Minhash was originally developed for economically estimating the *resemblance* similarity between sets (which can be equivalently viewed as binary vectors). Later, because of its locality sensitive property [22], minhash became a widely used hash function for creating hash buckets leading to efficient algorithms for numerous applications including spam detection [6], collaborative filtering [4], news personalization [15], compressing social networks [13], graph sampling [14], record linkage [25], duplicate detection [21], all pair similarity [5], etc.

WWW 2015, May 18–22, 2015, Florence, Italy.
ACM 978-1-4503-3469-3/15/05.
http://dx.doi.org/10.1145/2736277.2741285 .

1.1 Sparse Binary Data, Set Resemblance, and Set Containment

Binary representations for web data are common, owing to the wide adoption of the "bag of words (n-gram)" representations for documents and images. It is often the case that a significant number of words (or combinations of words) occur rarely in a document and most of the higher-order n-grams in the document occur only once. Thus in practice, often only the presence or absence information suffices [9, 20, 24]. Leading search firms routinely use sparse binary representations in their large data systems, e.g., [8].

The underlying similarity measure of interest with minhash is the resemblance (also known as the Jaccard similarity). The resemblance similarity between two sets $x, y \subseteq \Omega = \{1, 2, ..., D\}$ is

$$\mathcal{R} = \frac{|x \cap y|}{|x \cup y|} = \frac{a}{f_x + f_y - a}, \tag{1}$$
$$\text{where } \ f_x = |x|, \ \ f_y = |y|, \ \ a = |x \cap y|.$$

Sets can be equivalently viewed as binary vectors with each component indicating the presence or absence of an attribute. The cardinality (e.g., f_x, f_y) is the number of nonzeros in the binary vector.

While the resemblance similarity is convenient and useful in numerous applications, there are also many scenarios where the resemblance is not the desired similarity measure [1, 11]. For instance, consider text descriptions of two restaurants:

1. "Five Guys Burgers and Fries Downtown Brooklyn New York"
2. "Five Kitchen Berkley"

Shingle (n-gram) based representations for strings are common in practice. Typical (first-order) shingle based representations of these names will be (i) {five, guys, burgers, and, fries, downtown, brooklyn, new, york } and (ii) {five, kitchen, berkley}. Now suppose the query is "Five Guys" which in shingle representation is {Five, Guys}. Suppose we hope to match and search the records, for this query "Five Guys", based on resemblance. Observe that the resemblance between query and record (i) is $\frac{2}{9}$ = 0.22, while that with record (ii) is $\frac{1}{4}$ = 0.25. Thus, simply based on resemblance, record (ii) is a better match for query "Five Guys" than record (i), which however should not be correct in this content.

Clearly the issue here is that resemblance penalizes the sizes of the sets involved. Shorter sets are unnecessarily favored over longer ones, which hurts the performance in (e.g.,) record matching [1]. There are other scenarios where such penalization is undesirable. For instance, in plagiarism detection, it is typically immaterial whether the text is plagiarized from a long or a short document.

To counter the often unnecessary penalization of the sizes of the sets with resemblance, a modified measure, the *set containment* (or Jaccard containment) was adopted [6, 1, 11]. Containment of set x and y with respect to x is defined as

$$\mathcal{J}_{\mathcal{C}} = \frac{|x \cap y|}{|x|} = \frac{a}{f_x}. \tag{2}$$

In the above example with query "Five Guys", the set containment with respect to query for record (i) will be $\frac{2}{2} = 1$ and with respect to record (ii) it will be $\frac{1}{2}$, leading to the desired ordering. It should be noted that for any fixed query x, the ordering under set containment with respect to the query, is the same as the ordering with respect to the intersection a (or binary inner product). Thus, near neighbor search problem with respect to $\mathcal{J}_{\mathcal{C}}$ is equivalent to the near neighbor search problem with respect to a.

1.2 Maximum Inner Product Search (MIPS) & Maximum Containment Search (MCS)

Formally, we state our problem of interest. We are given a collection $\mathcal{C}$ containing n sets (or binary vectors) over universe Ω with $|\Omega| = D$ (or binary vectors in $\{0,1\}^D$). Given a query $q \subset \Omega$, we are interested in the problem of finding $x \in \mathcal{C}$ such that

$$x = \arg\max_{x \in \mathcal{C}} |x \cap q| = \arg\max_{x \in \mathcal{C}} q^T x; \tag{3}$$

where $|\ |$ is the cardinality of the set. This is the so-called *maximum inner product search (MIPS)* problem.

For binary data, the MIPS problem is equivalent to searching with set containment with respect to the query, because the cardinality of the query does not affect the ordering and hence

$$x = \arg\max_{x \in \mathcal{C}} |x \cap q| = \arg\max_{x \in \mathcal{C}} \frac{|x \cap q|}{|q|}; \tag{4}$$

which we also refer to as the *maximum containment search (MCS)* problem.

1.3 Shortcomings of Inverted Index Based Approaches for MIPS (and MCS)

Owing to its practical significance, there have been many existing heuristics for solving the MIPS (or MCS) problem [31, 34, 12]. A notable recent work among them made use of the inverted index based approach [1]. Inverted indexes might be suitable for problems when the sizes of documents are small and each record only contains few words. This situation, however, is not always observed in practice. The documents over the web are large with huge vocabulary. Moreover, the vocabulary blows up very quickly once we start using higher-order shingles. In addition, there is an increasing interest in enriching the text with extra synonyms to make the search more effective and robust to semantic meanings [1], at the cost of a significant increase of the sizes of the documents. Furthermore, if the query contains many words then the inverted index is not very useful. To mitigate this issue several additional heuristics were proposed, for instance, the heuristic based on minimal infrequent sets [1]. Computing minimal infrequent sets is similar to the set cover problem which is hard in general and thus [1] resorted to greedy heuristics. The number of minimal infrequent sets could be huge in general and so these heuristics can be very costly. Also, such heuristics require the knowledge of the entire dataset before hand which is usually not practical in a dynamic environment like the web. In addition, inverted index based approaches do not have theoretical guarantees on the query time and their performance is very much dataset dependent. Not surprisingly, it was shown in [17] that simply using a sign of the projected document vector representation referred to as TOPSIG, which is also similar in nature to sign random projections (SRP) [18, 10], outperforms inverted index based approaches for querying.

1.4 Probabilistic Hashing

Locality Sensitive Hashing (LSH) [22] based randomized techniques are common and successful in industrial practice for efficiently solving NNS (*near neighbor search*). They are some of the few known techniques that do not suffer from the curse of dimensionality. Hashing based indexing schemes provide provably sub-linear algorithms for search which is a boon in this era of big data where even linear search algorithms are impractical due to latency. Furthermore, hashing based indexing schemes are massively parallelizable, which makes them ideal for modern distributed systems. The prime focus of this paper will be on efficient hashing based algorithms for binary inner products.

Despite the interest in set containment and binary inner products, there were no hashing algorithms for these measures for a long time and minwise hashing is still a popular heuristic [1]. Recently, it was shown that general inner products for real vectors can be efficiently solved by using asymmetric locality sensitive hashing schemes [35, 37]. The asymmetry is necessary for the general inner products and an impossibility of having a symmetric hash function can be easily shown using elementary arguments. Thus, binary inner product (or set intersection) being a special case of general inner products also admits provable efficient search algorithms with these asymmetric hash functions which are based on random projections. However, it is known that random projections are suboptimal for retrieval in the sparse binary domain [39]. Hence, it is expected that the existing asymmetric locality sensitive hashing schemes for general inner products are likely to be suboptimal for retrieving with sparse high dimensional binary-like datasets, which are common over the web.

1.5 Our Contributions

We investigate hashing based indexing schemes for the problem of near neighbor search with binary inner products and set containment. The impossibility of existence of LSH for general inner products shown in [35] also hold for the binary case.

Recent results on hashing algorithms for maximum inner product search [35] have shown the usefulness of asymmetric transformations in constructing provable hash functions for new similarity measures, which were otherwise impossible. Going further along this line, we provide a novel (and still very simple) asymmetric transformation for binary data, that corrects minhash and removes the undesirable bias of minhash towards the sizes of the sets involved. Such an asymmetric correction eventually leads to a provable hashing scheme for binary inner products, which we call *asymmetric minwise hashing (MH-ALSH)*. Our theoretical comparisons show that for binary data, which are common over the web, the new hashing scheme is provably more efficient that the recently proposed asymmetric hash functions for general inner products [35, 37]. Thus, we obtain a provable algorithmic improvement over the state-of-the-art hashing technique for binary inner products. The construction of our asymmetric transformation for minhash could be of independent interest in itself.

The proposed asymmetric minhash significantly outperforms existing hashing schemes, in the tasks of ranking and near neighbor search with set containment as the similarity measure, on four real-world high-dimensional datasets. Our final proposed algorithm is simple and only requires minimal modifications of the traditional minhash and hence it can be easily adopted in practice.

2. BACKGROUND

2.1 c-Approximate Near Neighbor Search and the Classical LSH

Past attempts of finding efficient algorithms, for exact near neighbor search based on space partitioning, often turned out to be a disappointment with the massive dimensionality of modern datasets [40]. Due to the curse of dimensionality, theoretically it is hopeless to obtain an efficient algorithm for exact near neighbor search. Approximate versions of near neighbor search problem were proposed [22] to overcome the linear query time bottleneck. One commonly adopted such formulation is the c-approximate Near Neighbor (c-NN).

DEFINITION 1. *(c-Approximate Near Neighbor or c-NN). [22] Given a set of points in a d-dimensional space $\mathbb{R}^d$, and parameters $S_0 > 0$, $\delta > 0$, construct a data structure which, given any query point q, does the following with probability $1 - \delta$: if there exists an S_0-near neighbor of q in P, it reports some cS_0-near neighbor.*

The usual notion of S_0-near neighbor is in terms of distance. Since we are dealing with similarities, we define S_0-near neighbor of point q as a point p with $Sim(q,p) \geq S_0$, where Sim is the similarity function of interest.

The popular technique, with near optimal guarantees for c-NN in many interesting cases, uses the underlying theory of *Locality Sensitive Hashing* (LSH) [22]. LSH relies on a family of functions, with the property that similar input objects in the domain of these functions have a higher probability of colliding in the range space than non-similar ones. More specifically, consider $\mathcal{H}$ a family of hash functions mapping $\mathbb{R}^D$ to some set $\mathcal{S}$.

DEFINITION 2. *(Locality Sensitive Hashing) A family $\mathcal{H}$ is called (S_0, cS_0, p_1, p_2) sensitive if for any two point $x, y \in \mathbb{R}^D$ and h chosen uniformly from $\mathcal{H}$ satisfies the following:*

- *if $Sim(x,y) \geq S_0$ then $Pr_{\mathcal{H}}(h(x) = h(y)) \geq p_1$*
- *if $Sim(x,y) \leq cS_0$ then $Pr_{\mathcal{H}}(h(x) = h(y)) \leq p_2$*

For approximate nearest neighbor search typically, $p_1 > p_2$ and $c < 1$ is needed. Note, $c < 1$ as we are defining neighbors in terms of similarity. To obtain distance analogy we can resort to $D(x,y) = 1 - Sim(x,y)$

FACT 1. *[22] Given a family of (S_0, cS_0, p_1, p_2) -sensitive hash functions, one can construct a data structure for c-NN with $O(n^\rho \log_{1/p_2} n)$ query time and space $O(n^{1+\rho})$, $\rho = \frac{\log 1/p_1}{\log 1/p_2} < 1$*

LSH trades off query time with extra preprocessing time and space that can be accomplished off-line. It requires constructing a one time data structure which costs $O(n^{1+\rho})$ space and further any c-approximate near neighbor queries can be answered in $O(n^\rho \log_{1/p_2} n)$ time in the worst case.

A particularly interesting sufficient condition for existence of LSH is the monotonicity of the collision probability in $Sim(x,y)$. Thus, if a hash function family $\mathcal{H}$ satisfies,

$$Pr_{h\in\mathcal{H}}(h(x) = h(y)) = g(Sim(x,y)), \tag{5}$$

where g is a monotonically increasing function, then the conditions of Definition 2 are automatically satisfied for all $c < 1$.

The quantity $\rho < 1$ is a property of the LSH family, and it is of particular interest because it determines the worst case query complexity of the c-approximate near neighbor search. It should be further noted, that the complexity depends on S_0 which is the operating threshold and c, the approximation ratio we are ready to tolerate. In case when we have two or more LSH families for a given similarity measure, then the LSH family with smaller value of ρ, for given S_0 and c, is preferred.

2.2 Minwise Hashing (Minhash)

Minwise hashing [6] is the LSH for the *resemblance*, also known as the *Jaccard similarity*, between sets. In this paper, we focus on binary data vectors which can be equivalent viewed as sets.

Given a set $x \in \Omega = \{1, 2, ..., D\}$, the minwise hashing family applies a random permutation $\pi : \Omega \to \Omega$ on x and stores only the minimum value after the permutation mapping. Formally minwise hashing (or minhash) is defined as:

$$h_\pi(x) = \min(\pi(x)). \tag{6}$$

Given sets x and y, it can be shown that the probability of collision is the resemblance $\mathcal{R} = \frac{|x\cap y|}{|x\cup y|}$:

$$Pr_\pi(h_\pi(x) = h_\pi(y)) = \frac{|x \cap y|}{|x \cup y|} = \frac{a}{f_x + f_y - a} = \mathcal{R}. \tag{7}$$

where $f_x = |x|$, $f_y = |y|$, and $a = |x \cap y|$. It follows from Eq. (7) that minwise hashing is (S_0, cS_0, S_0, cS_0)-sensitive family of hash function when the similarity function of interest is resemblance.

Even though minhash was really meant for retrieval with resemblance similarity, it is nevertheless a popular hashing scheme used for retrieving set containment or intersection for binary data [1]. In practice, the ordering of inner product a and the ordering or resemblance $\mathcal{R}$ can be different because of the variation in the values of f_x and f_y, and as argued in Section 1, which may be undesirable and lead to suboptimal results. We show later that by exploiting asymmetric transformations we can get away with the undesirable dependency on the number of nonzeros leading to a better hashing scheme for indexing set intersection (or binary inner products).

2.3 LSH for L2 Distance (L2LSH)

[16] presented a novel LSH family for all L_p ($p \in (0,2]$) distances. In particular, when $p = 2$, this scheme provides an LSH family for L_2 distance. Formally, given a fixed number r, we choose a random vector w with each component generated from i.i.d. normal, i.e., $w_i \sim N(0,1)$, and a scalar b generated uniformly at random from $[0, r]$. The hash function is defined as:

$$h^{L2}_{w,b}(x) = \left\lfloor \frac{w^T x + b}{r} \right\rfloor, \tag{8}$$

where $\lfloor\rfloor$ is the floor operation. The collision probability under this scheme can be shown to be

$$Pr(h^{L2}_{w,b}(x) = h^{L2}_{w,b}(y)) = F_r(d), \tag{9}$$

$$F_r(d) = 1 - 2\Phi(-r/d) - \frac{2}{\sqrt{2\pi}r/d}\left(1 - e^{-r^2/(2d^2)}\right) \tag{10}$$

where $\Phi(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx$ is the cumulative density function (cdf) of standard normal distribution and $d = ||x - y||_2$ is the Euclidean distance between the vectors x and y. This collision probability $F(d)$ is a monotonically decreasing function of the distance d and hence $h^{L2}_{w,b}$ is an LSH for $L2$ distances. This scheme is also the part of LSH package [2]. Here r is a parameter.

2.4 LSH for Cosine Similarity (SRP)

Sign Random Projections (SRP) or *simhash* is another popular LSH for the cosine similarity measure, which originates from the concept of **Sign Random Projections (SRP)** [18, 10]. Given a vector x, SRP utilizes a random w vector with each component generated from i.i.d. normal, i.e., $w_i \sim N(0,1)$, and only stores the sign of the projection. Formally simhash is given by

$$h^{sign}(x) = sign(w^T x). \tag{11}$$

It was shown in the seminal work [18] that collision under SRP satisfies the following equation:

$$Pr_w(h^{sign}(x) = h^{sign}(y)) = 1 - \frac{\theta}{\pi}, \tag{12}$$

where $\theta = cos^{-1}\left(\frac{x^T y}{||x||_2||y||_2}\right)$. The term $\frac{x^T y}{||x||_2||y||_2}$ is the popular **cosine similarity**. For sets (or equivalently binary vectors), the cosine similarity reduces to

$$\mathcal{S} = \frac{a}{\sqrt{f_x f_y}} \tag{13}$$

The recent work on *coding for random projections* [28] has shown the advantage of SRP (and 2-bit random projections) over L2LSH for both similarity estimation and near neighbor search. Interestingly, another recent work [39] has shown that for binary data (actually even sparse non-binary data), minhash can significantly outperform SRP for near neighbor search even as we evaluate both SRP and minhash in terms of the cosine similarity (although minhash is designed for resemblance). This motivates us to design asymmetric minhash for achieving better performance in retrieving set containments. But first, we provide an overview of asymmetric LSH for general inner products (not restricted to binary data).

2.5 Asymmetric LSH (ALSH)

The term "ALSH" stands for *asymmetric LSH*, as used in a recent work [35]. Through an elementary argument, [35] showed that it is not possible to have a Locality Sensitive Hashing (LSH) family for general unnormalized inner products.

For inner products between vectors x and y, it is possible to have $x^T y \gg x^T x$. Thus for any hashing scheme h to be a valid LSH, we must have $Pr(h(x) = h(y)) > Pr(h(x) = h(x)) = 1$, which is an impossibility. It turns out that there is a simple fix, if we allow asymmetry in the hashing scheme. Allowing asymmetry leads to an extended framework of asymmetric locality sensitive hashing (ALSH). The idea to is have a different hashing scheme for assigning buckets to the data point in the collection $\mathcal{C}$, and an altogether different hashing scheme while querying.

DEFINITION 3. *(Asymmetric Locality Sensitive Hashing (ALSH)) A family $\mathcal{H}$, along with the two vector functions $Q : \mathbb{R}^D \mapsto \mathbb{R}^{D'}$ (**Query Transformation**) and $P : \mathbb{R}^D \mapsto \mathbb{R}^{D'}$ (**Preprocessing Transformation**), is called (S_0, cS_0, p_1, p_2)-sensitive if for a given c-NN instance with query q, and the hash function h chosen uniformly from $\mathcal{H}$ satisfies the following:*

- *if $Sim(q,x) \geq S_0$ then $Pr_{\mathcal{H}}(h(Q(q))) = h(P(x))) \geq p_1$*
- *if $Sim(q,x) \leq cS_0$ then $Pr_{\mathcal{H}}(h(Q(q)) = h(P(x))) \leq p_2$*

Here x is any point in the collection $\mathcal{C}$. Asymmetric LSH borrows all theoretical guarantees of the LSH.

FACT 2. *Given a family of hash function $\mathcal{H}$ and the associated query and preprocessing transformations Q and P respectively, which is (S_0, cS_0, p_1, p_2) -sensitive, one can construct a data structure for c-NN with $O(n^\rho \log n)$ query time and space $O(n^{1+\rho})$, where $\rho = \frac{\log p_1}{\log p_2}$.*

[35] showed that using asymmetric transformations, the problem of **maximum inner product search (MIPS)** can be reduced to the problem of approximate near neighbor search in L_2. The algorithm first starts by scaling all $x \in \mathcal{C}$ by a constant large enough, such that $||x||_2 \leq U < 1$. The proposed ALSH family (**L2-ALSH**) is the LSH family for L_2 distance with the Preprocessing transformation $P : \mathbb{R}^D \mapsto \mathbb{R}^{D+m}$ and the Query transformation $Q : \mathbb{R}^D \mapsto \mathbb{R}^{D+2m}$ defined as follows:

$$P^{L2}(x) = [x; ||x||_2^2; ....; ||x||_2^{2^m}; 1/2; ...; 1/2] \tag{14}$$

$$Q^{L2}(x) = [x; 1/2; ...; 1/2; ||x||_2^2; ....; ||x||_2^{2^m}], \tag{15}$$

where [;] is the concatenation. $P^{L2}(x)$ appends m scalers of the form $||x||_2^{2^i}$ followed by m "1/2s" at the end of the vector x, while $Q^{L2}(x)$ first appends m "1/2s" to the end of the vector x and then m scalers of the form $||x||_2^{2^i}$. It was shown that this leads to provably efficient algorithm for MIPS.

FACT 3. *[35] For the problem of c-approximate MIPS in a bounded space, one can construct a data structure having $O(n^{\rho^*_{L2-ALSH}} \log n)$ query time and space $O(n^{1+\rho^*_{L2-ALSH}})$, where $\rho^*_{L2-ALSH} < 1$ is the solution to constrained optimization (16).*

$$\rho^*_{L2-ALSH} \tag{16}$$

$$= \min_{U<1, m\in N, r} \frac{\log F_r\left(\sqrt{m/2 - 2S_0\left(\frac{U^2}{V^2}\right) + 2U^{2^{m+1}}}\right)}{\log F_r\left(\sqrt{m/2 - 2cS_0\left(\frac{U^2}{V^2}\right)}\right)}$$

$$s.t. \quad \frac{U^{(2^{m+1}-2)}V^2}{S_0} < 1 - c,$$

Here the guarantees depends on the maximum norm of the space $V = \max_{x\in\mathcal{C}} ||x||_2$.

Quickly after, it was realized that a very similar idea can convert the MIPS problem in the problem of maximum cosine similarity search which can be efficiently solve by SRP leading to a new and better ALSH for MIPS **Sign-ALSH** [37] which works as follows: The algorithm again first starts by scaling all $x \in \mathcal{C}$ by a constant large enough, such that $||x||_2 \leq U < 1$. The proposed ALSH family (**Sign-ALSH**) is the SRP family for cosine similarity with the Preprocessing transformation $P^{sign} : \mathbb{R}^D \mapsto \mathbb{R}^{D+m}$ and the Query transformation $Q^{sign} : \mathbb{R}^D \mapsto \mathbb{R}^{D+2m}$ defined as follows:

$$P^{sign}(x) = [x; 1/2 - ||x||_2^2; ...; 1/2 - ||x||_2^{2^m}; 0; ...; 0] \tag{17}$$

$$Q^{sign}(x) = [x; 0; ...; 0; 1/2 - ||x||_2^2; ...; 1/2 - ||x||_2^{2^m}], \tag{18}$$

where [;] is the concatenation. $P^{sign}(x)$ appends m scalers of the form $1/2 - ||x||_2^{2^i}$ followed by m "0s" at the end of the vector x, while $Q^{sign}(x)$ appends m "0" followed by m scalers of the form $1/2 - ||x||_2^{2^i}$ to the end of the vector x. It was shown that this leads to provably efficient algorithm for MIPS.

As demonstrated by the recent work [28] on *coding for random projections*, there is a significant advantage of SRP over L2LSH for near neighbor search. Thus, it is not surprising that Sign-ALSH outperforms L2-ALSH for the MIPS problem. Similar to L2LSH, the runtime guarantees for Sign-ALSH can be shown as:

FACT 4. *For the problem of c-approximate MIPS, one can construct a data structure having $O(n^{\rho^*_{Sign-ALSH}} \log n)$ query time and space $O(n^{1+\rho^*_{Sign-ALSH}})$, where $\rho^*_{Sign-ALSH} < 1$ is the* *solution to constraint optimization problem*

$$\rho^*_{Sign-ALSH} = \min_{U,m,} \frac{\log\left(1 - \frac{1}{\pi}\cos^{-1}\left(\frac{S_0 \times \left(\frac{U^2}{V^2}\right)}{\frac{m}{4} + U^{2^{m+1}}}\right)\right)}{\log\left(1 - \frac{1}{\pi}\cos^{-1}\left(\min\{\frac{cS_0U^2}{V^2}, z^*\}\right)\right)} \tag{19}$$

$$z^* = \left[\frac{(m - m2^{m-1}) + \sqrt{(m - m2^{m-1})^2 + m^2(2^m - 1)}}{4(2^m - 1)}\right]^{2^{-m}}$$

There is a similar asymmetric transformation [3, 32] which followed by sign random projection leads to another ALSH having very similar performance to Sign-ALSH. The ρ values, which were also very similar to the $\rho_{Sign-ALSH}$ can be shown as

$$\rho_{Sign} = \frac{\log\left(1 - \frac{1}{\pi}\cos^{-1}\left(\frac{S_0}{V^2}\right)\right)}{\log\left(1 - \frac{1}{\pi}\cos^{-1}\left(\frac{cS_0}{V^2}\right)\right)} \tag{20}$$

Both L2-ALSH and Sign-ALSH work for any general inner products over $\mathbb{R}^D$. For sparse and high-dimensional binary dataset which are common over the web, it is known that minhash is typically the preferred choice of hashing over random projection based hash functions [39]. We show later that the ALSH derived from minhash, which we call asymmetric minwise hashing (*MH-ALSH*), is more suitable for indexing set intersection for sparse binary vectors than the existing ALSHs for general inner products.

3. SAMPLING BASED ALSH FOR INDEXING BINARY INNER PRODUCTS

In [35], it was shown that there cannot exist any LSH for general unnormalized inner product. Using a slightly different argument it can be shown that even for binary data we cannot have any LSH scheme. Note, for binary inner product $x^Ty \leq x^Tx$ and therefore we cannot use exactly the same argument as before. But we can have x,y and z such that $x^Ty \gg z^Tz$. Now, $Pr(h(x) = h(y)) > Pr(h(z) = h(z)) = 1$ is again impossible. However, with asymmetry it is not difficult to construct a provable hashing scheme for binary inner product.

The construction is based on sampling. Simply sampling a random component leads to the popular LSH for hamming distance [33]. The ordering of inner product is different from that of hamming distance. The hamming distance between x and query q is given by $f_x + f_q - 2a$, while we want the collision probability to be monotonic in the inner product a. f_x makes it non-monotonic in a. Note that f_q has no effect on ordering of $x \in \mathcal{C}$ because it is constant for every query. To construct an LSH monotonic in binary inner product, we need an extra trick.

Given a binary data vector x, we sample a random co-ordinate (or attribute). If the value of this co-ordinate is 1 (in other words if this attribute is present in the set), our hash value is a fixed number 0. If this randomly sampled co-ordinate has value 0 (or the attribute is absent) then ensure that the hash value of the query never matches the hash value of the data. Formally,

$$\mathcal{H}_S(f(x)) = \begin{cases} 0 & \text{if } x_i = 1, i \text{ drawn uniformly} \\ 1 & \text{if f = Q (for query)} \\ 2 & \text{if f = P (while preprocessing)} \end{cases} \tag{21}$$

Note the asymmetry, i.e., the hash functions are different for query and the dataset. We can also write it down more formally using $P(.)$ and $Q(.)$ but we avoid it for the sake of simplicity.

THEOREM 1. *Given two binary vectors x and y, we have*

$$Pr(\mathcal{H}_S(P(x)) = \mathcal{H}_S(Q(y))) = \frac{a}{D} \tag{22}$$

PROOF. The probability that both $H_S(P(x))$ and $H_S(Q(y))$ have value 0 is $\frac{a}{D}$. They cannot be equal otherwise □

COROLLARY 1. *$\mathcal{H}_S$ is $(S_0, cS_0, \frac{S_0}{D}, \frac{cS_0}{D})$-sensitive ALSH for binary inner product with $\rho_{\mathcal{H}_S} = \frac{\log\left(\frac{S_0}{D}\right)}{\log\left(\frac{cS_0}{D}\right)} < 1$*

3.1 Shortcomings

The above ALSH for binary inner product is likely to be very inefficient for sparse and high dimensional datasets. For those datasets, typically the value of D is very high and the sparsity ensures that a is very small. For modern web datasets, we can have D running into billions (or 2^{64}) while the sparsity is only in few hundreds or perhaps thousands [8]. Therefore, we have $\frac{a}{D} \simeq 0$ which essentially boils down to $\rho_{\mathcal{H}_S} \simeq 1$. In other words, the hashing scheme becomes worthless in sparse high dimensional domain. On the other hand, if we observe the collision probability of minhash Eq.(7), the denominator is $f_x + f_y - a$, which is usually of the order of a and much less than the dimensionality for sparse datasets.

Another way of realizing the problem with the above ALSH is to note that it is informative only if a randomly sampled co-ordinate has value equal to 1. For very sparse dataset with $a \ll D$, sampling a non zero coordinate has probability $\frac{a}{D} \simeq 0$. Thus, almost all of the hashes will be fixed numbers which are not informative.

3.2 Why Is Minhash Reasonable?

In this section, we argue why retrieving inner product based on plain minhash is a reasonable thing to do. Later, we will show a provable way to improve it using asymmetric transformations.

The number of nonzeros in the query, i.e., $|q| = f_q$ does not change the identity of $\arg\max$ in Eq.(4). Let us assume that we have data of bounded sparsity and define constant M as

$$M = \max_{x \in \mathcal{C}} |x| \tag{23}$$

where M is the maximum number of nonzeros (or maximum cardinality of sets) seen in the database. For sparse data seen in practice M is likely to be small compared to D. Outliers, if any, can be handled separately. By observing that $a \leq f_x \leq M$, we also have

$$\frac{a}{f_q + M - a} \leq \frac{a}{f_x + f_q - a} = \mathcal{R} \leq \frac{a}{f_q} \tag{24}$$

Thus, given the bounded sparsity, if we assume that the number of nonzeros in the query is given, then we can show that minhash is an LSH for inner products a because the collision probability can be upper and lower bounded by purely functions of a, M and f_q.

THEOREM 2. *Given bounded sparsity and query q with $|q| = f_q$, minhash is a $(S_0, cS_0, \frac{S_0}{f_q+M-S_0}, \frac{cS_0}{f_q})$ sensitive for inner products a with $\rho^q_{min} = \frac{\log \frac{S_0}{f_q+M-S_0}}{\log \frac{cS_0}{f_q}}$*

This explains why minhash might be a reasonable hashing approach for retrieving inner products or set intersection.

Here, if we remove the assumption that $|q| = f_q$ then in the worst case $\mathcal{R} \leq \frac{a}{f_q} \leq 1$ and we get $\log 1$ in the denominator. Note that the above is the worst case analysis and the assumption $|q| = f_q$ is needed to obtain any meaningful ρ with minhash. We show the power of ALSH in the next section, by providing a better hashing scheme and we do not even need the assumption of fixing $|q| = f_q$.

4. ASYMMETRIC MINWISE HASHING

In this section, we provide a very simple asymmetric fix to minhash, named *asymmetric minwise hashing (MH-ALSH)*, which makes the overall collision probability monotonic in the original inner product a. For sparse binary data, which is common in practice, we later show that the proposed hashing scheme is superior (both theoretically as well as empirically) compared to the existing ALSH schemes for inner product [35].

4.1 The New ALSH for Binary Data

We define the new preprocessing and query transformations $P' : [0,1]^D \rightarrow [0,1]^{D+M}$ and $Q' : [0,1]^D \rightarrow [0,1]^{D+M}$ as:

$$P'(x) = [x; 1; 1; 1; ...; 1; 0; 0; ...; 0] \tag{25}$$

$$Q'(x) = [x; 0; 0; 0; ...; 0], \tag{26}$$

For $P'(x)$ we append $M - f_x$ 1s and rest f_x zeros, while in $Q'(x)$ we simply append M zeros.

At this point we can already see the power of asymmetric transformations. The original inner product between $P'(x)$ and $Q'(x)$ is unchanged and its value is $a = x^T y$. Given the query q, the new resemblance R' between $P'(x)$ and $Q'(q)$ is

$$R' = \frac{|P'(x) \cap Q'(q)|}{|P'(x) \cup Q'(q)|} = \frac{a}{M + f_q - a}. \tag{27}$$

If we define our new similarity as $Sim(x, y) = \frac{a}{M+f_q-a}$, then the near neighbors in this new similarity are the same as near neighbors with respect to either set intersection a or set containment $\frac{a}{f_q}$. Thus, we can instead compute near neighbors in $\frac{a}{M+f_q-a}$ which is also the resemblance between $P'(x)$ and $Q'(q)$. We can therefore use minhash on $P'(x)$ and $Q'(q)$.

Observe that now we have $M + f_q - a$ in the denominator, where M is the maximum nonzeros seen in the dataset (the cardinality of largest set), which for very sparse data is likely to be much smaller than D. Thus, asymmetric minhash is a better scheme than $\mathcal{H}_S$ with collision probability $\frac{a}{D}$ for very sparse datasets where we usually have $M \ll D$.

From theoretical perspective, to obtain an upper bound on the query and space complexity of c-approximate near neighbor with binary inner products, we want the collision probability to be independent of the quantity f_q. This is not difficult to achieve. The asymmetric transformation used to get rid of f_x in the denominator can be reapplied to get rid of f_q.

Formally, we can define $P'' : [0,1]^D \rightarrow [0,1]^{D+2M}$ and $Q'' : [0,1]^D \rightarrow [0,1]^{D+2M}$ as :

$$P''(x) = Q'(P'(x)); \quad Q''(x) = P'(Q'(x)); \tag{28}$$

where in $P''(x)$ we append $M - f_x$ 1s and rest $M + |f_x|$ zeros, while in $Q''(x)$ we append M zeros, then $M - f_q$ 1s and rest zeros

Again the inner product a is unaltered, and the new resemblance then becomes

$$R'' = \frac{|P''(x) \cap Q''(q)|}{|P''(x) \cup Q''(q)|} = \frac{a}{2M - a}. \tag{29}$$

which is independent of f_q and is monotonic in a. This allows us to achieve a formal upper bound on the complexity of c-approximate maximum inner product search with the new asymmetric minhash.

From the collision probability expression, i.e., Eq. (29), we have

THEOREM 3. *Minwise hashing along with Query transformation Q'' and Preprocessing transformation P'' defined by Equation 28 is a $(S_0, cS_0, \frac{S_0}{2M-S_0}, \frac{cS_0}{2M-cS_0})$ sensitive asymmetric hashing family for set intersection.*

This leads to an important corollary.

COROLLARY 2. *There exists an algorithm for c-approximate set intersection, with bounded sparsity M, that requires space $O(n^{1+\rho_{MH-ALSH}})$ and query time $O(n^{\rho_{MH-ALSH}} \log n)$, where*

$$\rho_{MH-ALSH} = \frac{\log \frac{S_0}{2M-S_0}}{\log \frac{cS_0}{2M-cS_0}} < 1 \tag{30}$$

Given query q and any point $x \in \mathcal{C}$, the collision probability under traditional minhash is $R = \frac{a}{f_x+f_q-a}$. This penalizes sets with high f_x, which in many scenarios is not desirable. To balance this negative effect, asymmetric transformation penalizes sets with smaller f_x. Note, that $M - f_x$ ones added in the transformations $P'(x)$ gives additional chance in proportion to $M - f_x$ for minhash of $P'(x)$ not to match with the minhash of $Q'(x)$. This asymmetric probabilistic correction balances the penalization inherent in minhash. This is a simple way of correcting the probability of collision which could be of independent interest in itself. We will show in our evaluation section, that despite this simplicity such correction leads to significant improvement over plain minhash.

4.2 Efficient Sampling

Our transformations P'' and Q'' always create sets with $2M$ nonzeros. In case when M is big, hashing might take a lot of time. We can use (improved) consistent weighted sampling [30, 23] for efficient generation of hashes. We can instead use transformations P''' and Q''' that makes the data non-binary as follows

$$P'''(x) = [x; M - f_x; 0] \tag{31}$$

$$Q'''(x) = [x; 0; M - f_x]$$

It is not difficult to see that the weighted resemblance (or weighted Jaccard similarity) between $P'''(x)$ and $Q'''(q)$ for given query q and any $x \in \mathcal{C}$ is

$$\mathcal{R}_W = \frac{\sum_i \min(P'''(x)_i, Q'''(q)_i)}{\sum_i \max(P'''(x)_i, Q'''(q)_i)} = \frac{a}{2M - a}. \tag{32}$$

Therefore, we can use fast consistent weighted sampling for weighted resemblance on $P'''(x)$ and $Q'''(x)$ to compute the hash values in time constant per nonzero weights, rather than maximum sparsity M. In practice we will need many hashes for which we can utilize the recent line of work that make minhash and weighted minhash significantly much faster [29, 36, 38, 19].

5. THEORETICAL COMPARISONS

For solving the MIPS problem in general data types, we already know two asymmetric hashing schemes, *L2-ALSH* and *Sign-ALSH*, as described in Section 2.5. In this section, we provide theoretical comparisons of the two existing ALSH methods with the proposed asymmetric minwise hashing (*MH-ALSH*). As argued, the LSH scheme described in Section 3 is unlikely to be useful in practice because of its dependence on D; and hence we can safely ignore it for simplicity of the discussion.

Before we formally compare various asymmetric LSH schemes for maximum inner product search, we argue why asymmetric minhash should be advantageous over traditional minhash for retrieving inner products. Let q be the binary query vector, and f_q denotes the number of nonzeros in the query. The $\rho_{MH-ALSH}$ for asymmetric minhash in terms of f_q and M is straightforward from the collision probability Eq.(27):

$$\rho^q_{MH-ALSH} = \frac{\log \frac{S_0}{f_q+M-S_0}}{\log \frac{cS_0}{f_q+M-cS_0}} \tag{33}$$

For minhash, we have from theorem 2 $\rho^q_{min} = \frac{\log \frac{S_0}{f_q+M-S_0}}{\log \frac{cS_0}{f_q}}$. Since M is the upper bound on the sparsity and cS_0 is some value of inner product, we have $M - cS_0 \geq 0$. Using this fact, the following theorem immediately follows

THEOREM 4. *For any query q, we have $\rho^q_{MH-ALSH} \leq \rho^q_{min}$.*

This result theoretically explains why asymmetric minhash is better for retrieval with binary inner products, compared to plain minhash.

For comparing asymmetric minhash with ALSH for general inner products, we compare $\rho_{MH-ALSH}$ with the ALSH for inner products based on sign random projections. Note that it was shown that Sign-ALSH has better theoretical ρ values compared to L2-ALSH [37]. Therefore, it suffices to show that asymmetric minhash outperforms sign random projection based ALSH. Both $\rho_{MH-ALSH}$ and ρ_{sign} can be rewritten in terms of ratio $\frac{S_0}{M}$ as follows. Note that for binary data we have $M = \max_{x\in\mathcal{C}} ||x||^2 = V^2$

$$\rho_{MH-ALSH} = \frac{\log \frac{S_0/M}{2-S_0/M}}{\log \frac{cS_0/M}{2-cS_0/M}}; \quad \rho_{Sign} = \frac{\log\left(1 - \frac{1}{\pi}\cos^{-1}\left(\frac{S_0}{M}\right)\right)}{\log\left(1 - \frac{1}{\pi}\cos^{-1}\left(\frac{cS_0}{M}\right)\right)} \tag{34}$$

Observe that M is also the upper bound on any inner product. Therefore, we have $0 \leq \frac{S_0}{M} \leq 1$. We plot the values of $\rho_{MH-ALSH}$ and ρ_{sign} for $\frac{S_0}{M} = \{0.1,\ 0.2, ..., 0.8,\ 0.9,\ 0.95\}$ with c. The comparison is summarized in Figure 1. Note that here we use ρ_{Sign} based on the slightly more convenient transformation from [3, 32] instead of $\rho_{Sign-ALSH}$ for convenience although the two schemes perform essentially the same.

Clearly, irrespective of the choice of threshold $\frac{S_0}{M}$ or the approximation ratio c, asymmetric minhash outperforms sign random projection based ALSH in terms of the theoretical ρ values. This is not surprising, because it is known that minhash based methods are often significantly powerful for binary data compared to SRP (or simhash) [39]. Therefore ALSH based on minhash outperforms ALSH based on SRP as shown by our theoretical comparisons. Our proposal thus leads to an algorithmic improvement over state-of-the-art hashing techniques for retrieving binary inner products.

6. EVALUATIONS

In this section, we compare the different hashing schemes on the actual task of retrieving top-ranked elements based on set Jaccard containment. The experiments are divided into two parts. In the first part, we show how the ranking based on various hash functions correlate with the ordering of set containment. In the second part, we perform the actual LSH based bucketing experiment for retrieving top-ranked elements and compare the computational saving obtained by various hashing algorithms.

6.1 Datasets

We used four publicly available high dimensional sparse datasets: *EP2006*, *MNIST*, *NEWS20*, and *NYTIMES*. (Note that “EP2006” is a short name for “E2006LOG1P” from LIBSVM web site.) Except for MNIST, the other three datasets are binary “BoW" representation of the corresponding text corpus. MNIST is an image dataset consisting of 784 pixel image of handwritten digits. Binarized versions of MNIST are commonly used in literature. The pixel values in MNIST were binarized to 0 or 1 values. For each of the four datasets, we generate two partitions. The bigger partition was used to create hash tables and is referred as the *training partition*. The small partition which we call the *query partition* is used for querying. The statistics of these datasets are summarized in Table 1. The datasets cover a wide spectrum of sparsity and dimensionality.

Table 1: Datasets

Dataset	# Query	# Train	# Dim	nonzeros (mean $\pm$ std)
EP2006	2,000	17,395	4,272,227	6072 $\pm$ 3208
MNIST	2,000	68,000	784	150 $\pm$ 41
NEWS20	2,000	18,000	1,355,191	454 $\pm$ 654
NYTIMES	2,000	100,000	102,660	232 $\pm$ 114

6.2 Competing Hash Functions

We consider the following hash functions for evaluations:

1. **Asymmetric minwise hashing (Proposed):** This is our proposal, the asymmetric minhash described in Section 4.1.

2. **Traditional minwise hashing (MinHash):** This is the usual minwise hashing, the popular heuristic described in Section 2.2. This is a symmetric hash function, we use h_π as defined in Eq.(6) for both query and the training set.

3. **L2 based Asymmetric LSH for Inner products (L2-ALSH):** This is the asymmetric LSH of [35] for general inner products based on LSH for L2 distance.

4. **SRP based Asymmetric LSH for Inner Products (Sign-ALSH):** This is the asymmetric hash function of [37] for general inner products based on SRP.

6.3 Ranking Experiment: Hash Quality Evaluations

We are interested in knowing, how the orderings under different competing hash functions correlate with the ordering of the underlying similarity measure which in this case is the set containment. For this task, given a query q vector, we compute the top-100 gold standard elements from the training set based on the set containment $\frac{a}{f_q}$. Note that this is the same as the top-100 elements based on binary inner products. Give a query q, we compute K different hash codes of the vector q and all the vectors in the training set. We then compute the number of times the hash values of a vector x in the training set matches the hash values of query q defined by

$$Matches_x = \sum_{t=1}^{K} \mathbf{1}(h_t(q) = h_t(x)), \tag{35}$$

where $\mathbf{1}$ is the indicator function. t subscript is used to distinguish independent draws of the underlying hash function. Based on $Matches_x$ we rank all elements in the training set. This procedure generates a sorted list for every query for every hash function. For asymmetric hash functions, in computing total collisions, on the query vector we use the corresponding Q function (query transformation) followed by underlying hash function, while for elements in the training set we use the P function (preprocessing transformation) followed by the corresponding hash function.

We compute the precision and the recall of the top-100 gold standard elements in the ranked list generated by different hash functions. To compute precision and recall, we start at the top of the ranked item list and walk down in order, suppose we are at the p^{th} ranked element, we check if this element belongs to the gold standard top-100 list. If it is one of the top 100 gold standard elements, then we increment the count of *relevant seen* by 1, else we move to $p + 1$. By p^{th} step, we have already seen p elements, so the *total*

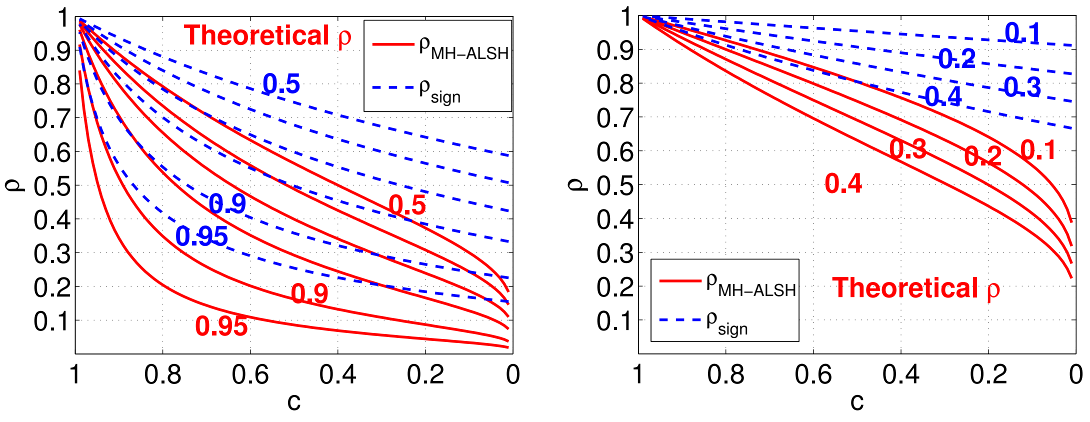


Figure 1: Values of $\rho_{MH-ALSH}$ and ρ_{sign} (lower is better) with respect to approximation ratio c for different $\frac{S_0}{M}$. The curves show that asymmetric minhash (solid curves) is noticeably better than ALSH based on sign random projection (dashed curves) in terms of their ρ values, irrespective of the choices of $\frac{S_0}{M}$ or c. For clarity, the results are shown in two panels.

elements seen is p. The precision and recall at that point is then computed as:

$$Precision = \frac{\text{relevant seen}}{p}, \qquad Recall = \frac{\text{relevant seen}}{100} \qquad (36)$$

It is important to balance both. Methodology which obtains higher precision at a given recall is superior. Higher precision indicates higher ranking of the relevant items. We finally average these values of precision and recall over all elements in the query set. The results for $K \in \{32,\ 64,\ 128\}$ are summarized in Figure 2.

We can clearly see, that the proposed hashing scheme always achieves better, often significantly, precision at any given recall compared to other hash functions. The two ALSH schemes are usually always better than traditional minwise hashing. This confirms that fact that ranking based on collisions under minwise hashing can be different from the rankings under set containment or inner products. This is expected, because minhash in addition penalizes the number of nonzeros leading to a ranking very different from the ranking of inner products. Sign-ALSH usually performs better than L2-LSH, this is in line with the results obtained in [37].

It should be noted that ranking experiments only validate the monotonicity of the collision probability. Although, better ranking is certainly a good indicator of good hash function, it does not always mean that we will achieve faster sub-linear LSH algorithm. For bucketing the probability sensitivity around a particular threshold is the most important factor, see [33] for more details. What matters is the **gap** between the collision probability of good and the bad points. In the next subsection, we compare these schemes on the actual task of near neighbor retrieval with set containment.

6.4 LSH Bucketing Experiment: Computational Savings in Near Neighbor Retrieval

In this section, we evaluate the four hashing schemes on the standard (K, L)-parameterized bucketing algorithm [2] for sub-linear time retrieval of near neighbors based on set containment. In (K, L)-parameterized LSH algorithm, we generate L different meta-hash functions. Each of these meta-hash functions is formed by concatenating K different hash values as

$$B_j(x) = [h_{j1}(x); h_{j2}(x); ...; h_{jK}(x)], \qquad (37)$$

where $h_{ij}, i \in \{1, 2, ..., K\}$ and $j \in \{1, 2, ..., L\}$, are KL different independent evaluations of the hash function under consideration. Different competing scheme uses its own underlying randomized hash function h.

In general, the (K, L)-parameterized LSH works in two phases:

i) **Preprocessing Phase:** We construct L hash tables from data by storing element x, in the training set, at location $B_j(P(x))$ in the hash-table j. Note that for vanilla minhash which is a symmetric hashing scheme $P(x) = x$. For other asymmetric schemes, we use their corresponding P functions. Preprocessing is a one time operation, once the hash tables are created they are fixed.

ii) **Query Phase:** Given a query q, we report the union of all the points in the buckets $B_j(Q(q))\ \forall j \in \{1, 2, ..., L\}$, where the union is over L hash tables. Again here Q is the corresponding Q function of the asymmetric hashing scheme, for minhash $Q(x) = x$.

Typically, the performance of a bucketing algorithm is sensitive to the choice of parameters K and L. Ideally, to find best K and L, we need to know the operating threshold S_0 and the approximation ratio c in advance. Unfortunately, the data and the queries are very diverse and therefore for retrieving top-ranked near neighbors there are no common fixed threshold S_0 and approximation ratio c that work for all the queries.

Our objective is to compare the four hashing schemes and minimize the effect of K and L, if any, on the evaluations. This is achieved by finding best K and L at every recall level. We run the bucketing experiment for all combinations of $K \in \{1, 2, 3, ...40\}$ and $L \in \{1, 2, 3, ..., 400\}$ for all the four hash functions independently. These choices include the recommended optimal combinations at various thresholds. We then compute, for every K and L, the mean recall of Top-T pairs and the mean number of points reported, per query, to achieve that recall. The best K and L at every recall level is chosen independently for different Ts. The plot of the mean fraction of points scanned with respect to the recall of top-T gold standard near neighbors, where $T \in \{5, 10, 20, 50\}$, is summarized in Figure 3.

The performance of a hashing based method varies with the variations in the similarity levels in the datasets. It can be seen that the proposed asymmetric minhash always retrieves much less number of points, and hence requires significantly less computations, compared to other hashing schemes at any recall level on all the four

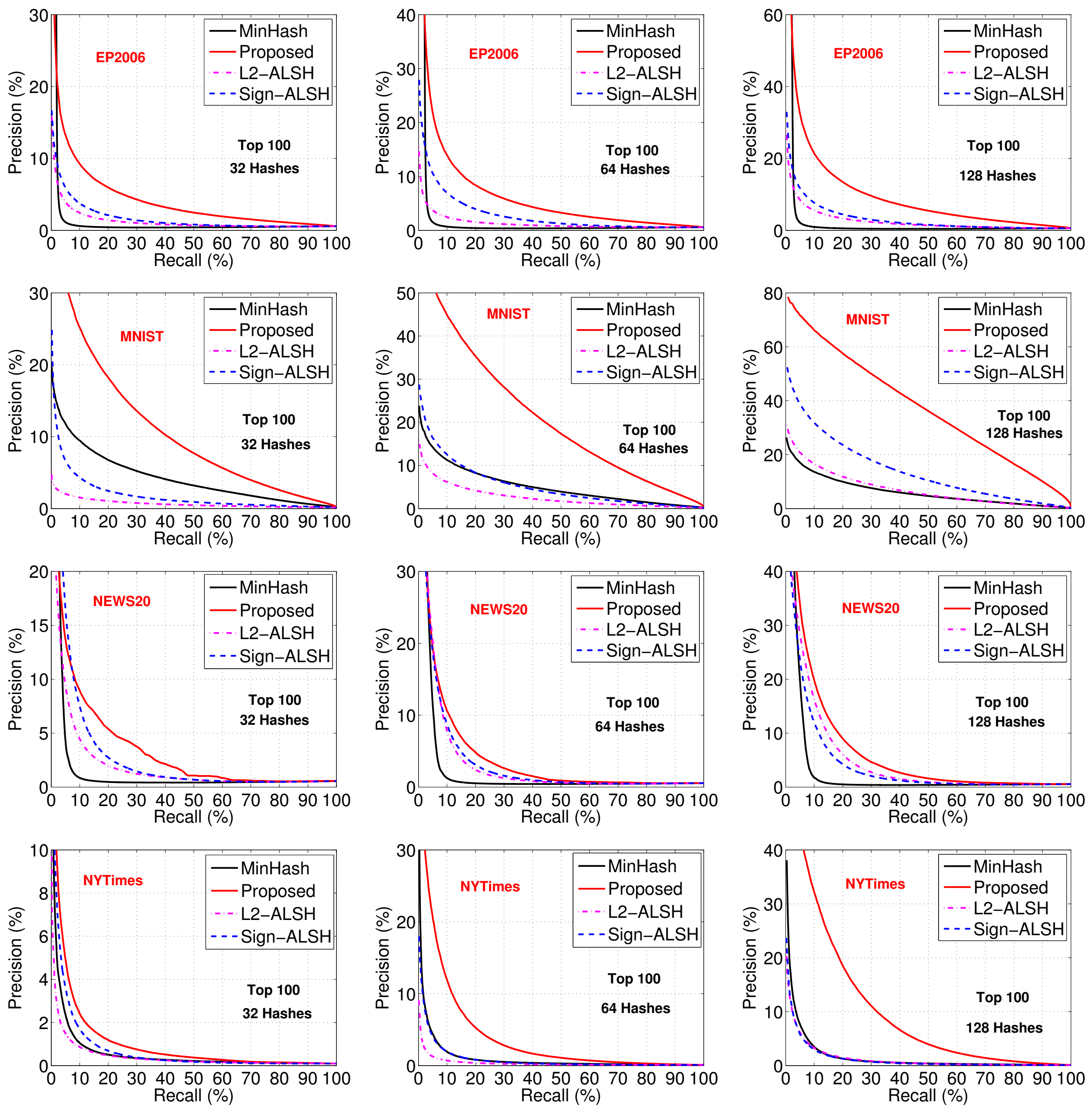


Figure 2: Ranking Experiments. Precision Vs Recall curves for retrieving top-100 items, for different hashing schemes on 4 chosen datasets. The precision and the recall were computed based on the rankings obtained by different hash functions using 32, 64 and 128 independent hash evaluations. Higher precision at a given recall is better.

datasets. Asymmetric minhash consistently outperforms other hash functions irrespective of the operating point. The plots clearly establish the superiority of the proposed scheme for indexing set containment (or inner products).

L2-ALSH and Sign-ALSH perform better than traditional minhash on EP2006 and NEWS20 datasets while they are worse than plain minhash on NYTIMES and MNIST datasets. If we look at the statistics of the dataset from Table 1, NYTIMES and MNIST are precisely the datasets with less variations in the number of nonzeros and hence minhash performs better. In fact, for MNIST dataset with very small variations in the number of nonzeros, the performance of plain minhash is very close to the performance of asymmetric minhash. This is of course expected because there is negligible effect of penalization on the ordering. EP2006 and NEWS20 datasets have huge variations in their number of nonzeros and hence minhash performs very poorly on these datasets. What is exciting is that despite these variations in the nonzeros, asymmetric minhash always outperforms other ALSH for general inner products.

The difference in the performance of plain minhash and asymmetric minhash clearly establishes the utility of our proposal which is simple and does not require any major modification over traditional minhash implementation. Given the fact that minhash is widely popular, we hope that our proposal will be adopted.

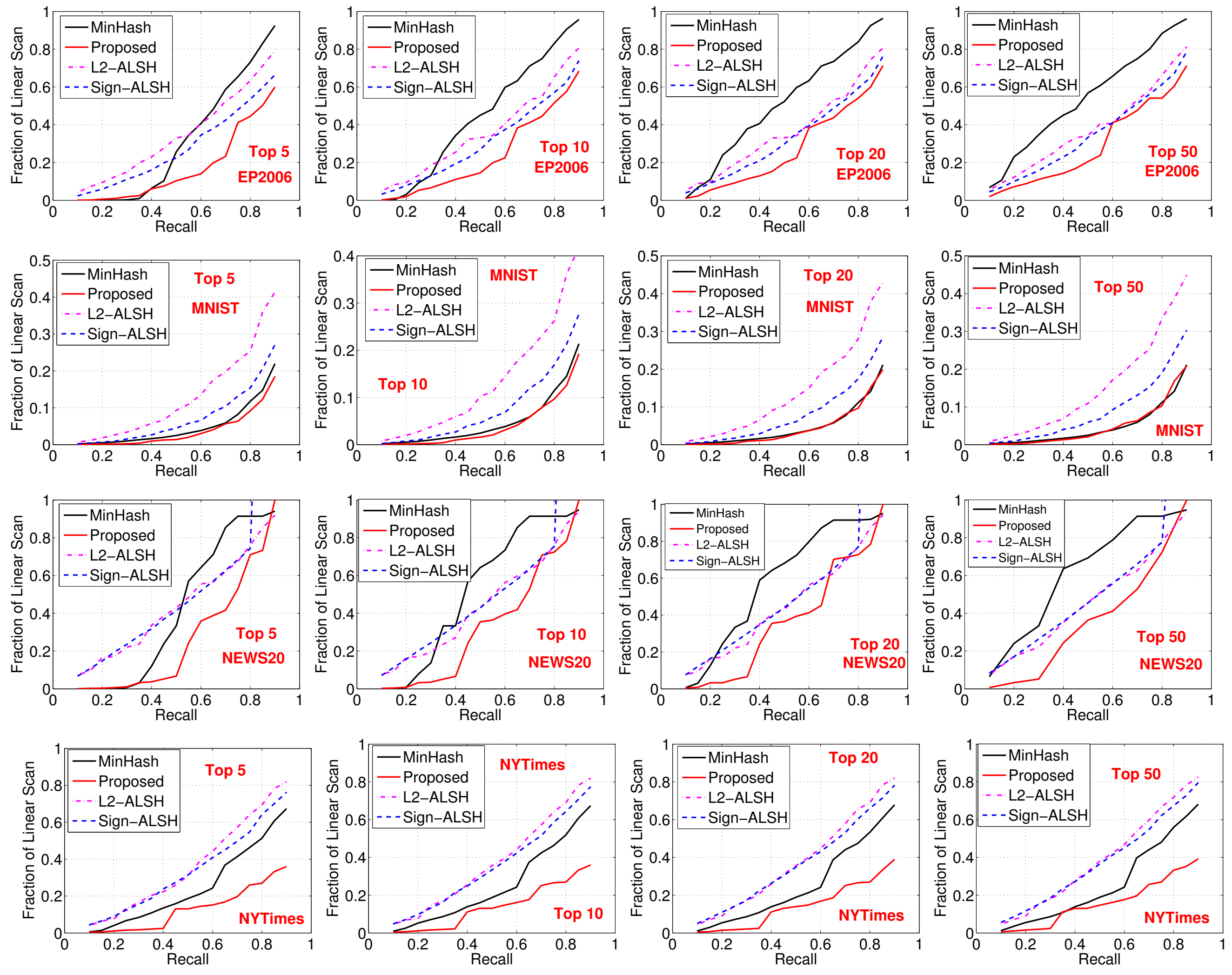


Figure 3: LSH Bucketing Experiments. Average number of points retrieved per query (lower is better), relative to linear scan, evaluated by different hashing schemes at different recall levels, for top-5, top-10, top-20, top-50 nearest neighbors based on set containment (or equivalently inner products), on four datasets. We show that results at the best K and L values chosen at every recall value, independently for each of the four hashing schemes.

7. CONCLUSION AND FUTURE WORK

Minwise hashing (minhash) is a widely popular indexing scheme in practice for similarity search. Minhash was originally designed for estimating set resemblance (i.e., normalized size of set intersections). In many applications the performance of minhash is severely affected because minhash has a bias towards smaller sets. In this study, we propose asymmetric corrections (asymmetric minwise hashing, or MH-ALSH) to minwise hashing that remove this often undesirable bias. Our corrections lead to a provably superior algorithm for retrieving binary inner products in the literature. Rigorous experimental evaluations on the task of retrieving maximum inner products clearly establish that the proposed approach can be significantly advantageous over the existing state-of-the-art hashing schemes in practice, when the desired similarity is the inner product (or containment) instead of the resemblance. Our proposed method requires only minimal modification of the original minwise hashing algorithm and should be straightforward to implement in practice.

Future work: One immediate direction for future work would be *asymmetric consistent weighted sampling* for hashing weighted intersection: $\sum_{i=1}^D \min\{x_i, y_i\}$, where x and y are general real-valued vectors. One proposal of the new asymmetric transformation is the following:

$$P(x) = [x; M - \sum_{i=1}^D x_i; 0], \qquad Q(x) = [x; 0; M - \sum_{i=1}^D x_i],$$

where $M = \max_{x \in \mathcal{C}} \sum_i x_i$. It is not difficult to show that the weighted Jaccard similarity between $P(x)$ and $Q(y)$ is monotonic in $\sum_{i=1}^D \min\{x_i, y_i\}$ as desired. At this point, we can use existing methods for consistent weighted sampling [30, 23, 19, 26]. on the new data after asymmetric transformations

Acknowledgement

The work is partially supported by NSF-DMS-1444124, NSF-III-1360971, ONR-N00014-13-1-0764, and AFOSR-FA9550-13-1-0137.

8. REFERENCES

[1] P. Agrawal, A. Arasu, and R. Kaushik. On indexing error-tolerant set containment. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 927–938. ACM, 2010.

[2] A. Andoni and P. Indyk. E2lsh: Exact euclidean locality sensitive hashing. Technical report, 2004.

[3] Y. Bachrach, Y. Finkelstein, R. Gilad-Bachrach, L. Katzir, N. Koenigstein, N. Nice, and U. Paquet. Speeding up the xbox recommender system using a euclidean transformation for inner-product spaces. In *RecSys*, 2014.

[4] Y. Bachrach, E. Porat, and J. S. Rosenschein. Sketching techniques for collaborative filtering. In *Proceedings of the 21st International Jont Conference on Artifical Intelligence*, IJCAI'09, 2009.

[5] R. J. Bayardo, Y. Ma, and R. Srikant. Scaling up all pairs similarity search. In *WWW*, pages 131–140, 2007.

[6] A. Z. Broder. On the resemblance and containment of documents. In *the Compression and Complexity of Sequences*, pages 21–29, Positano, Italy, 1997.

[7] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Min-wise independent permutations. In *STOC*, pages 327–336, Dallas, TX, 1998.

[8] T. Chandra, E. Ie, K. Goldman, T. L. Llinares, J. McFadden, F. Pereira, J. Redstone, T. Shaked, and Y. Singer. Sibyl: a system for large scale machine learning.

[9] O. Chapelle, P. Haffner, and V. N. Vapnik. Support vector machines for histogram-based image classification. *IEEE Transactions on Neural Networks*, 10(5):1055–1064, 1999.

[10] M. S. Charikar. Similarity estimation techniques from rounding algorithms. In *STOC*, pages 380–388, Montreal, Quebec, Canada, 2002.

[11] S. Chaudhuri, V. Ganti, and R. Kaushik. A primitive operatior for similarity joins in data cleaning. In *ICDE*, 2006.

[12] S. Chaudhuri, V. Ganti, and D. Xin. Mining document collections to facilitate accurate approximate entity matching. *Proceedings of the VLDB Endowment*, 2(1):395–406, 2009.

[13] F. Chierichetti, R. Kumar, S. Lattanzi, M. Mitzenmacher, A. Panconesi, and P. Raghavan. On compressing social networks. In *KDD*, pages 219–228, Paris, France, 2009.

[14] G. Cormode and S. Muthukrishnan. Space efficient mining of multigraph streams. In *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 271–282. ACM, 2005.

[15] A. S. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th international conference on World Wide Web*, pages 271–280. ACM, 2007.

[16] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokn. Locality-sensitive hashing scheme based on p-stable distributions. In *SCG*, pages 253 – 262, Brooklyn, NY, 2004.

[17] S. Geva and C. M. De Vries. Topsig: Topology preserving document signatures. In *CIKM*, pages 333–338, 2011.

[18] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of ACM*, 42(6):1115–1145, 1995.

[19] B. Haeupler, M. Manasse, and K. Talwar. Consistent weighted sampling made fast, small, and easy. Technical report, arXiv:1410.4266, 2014.

[20] M. Hein and O. Bousquet. Hilbertian metrics and positive definite kernels on probability measures. In *AISTATS*, pages 136–143, Barbados, 2005.

[21] M. R. Henzinger. Finding near-duplicate web pages: a large-scale evaluation of algorithms. In *SIGIR*, pages 284–291, 2006.

[22] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *STOC*, pages 604–613, Dallas, TX, 1998.

[23] S. Ioffe. Improved consistent sampling, weighted minhash and L1 sketching. In *ICDM*, pages 246–255, Sydney, AU, 2010.

[24] Y. Jiang, C. Ngo, and J. Yang. Towards optimal bag-of-features for object categorization and semantic video retrieval. In *CIVR*, pages 494–501, Amsterdam, Netherlands, 2007.

[25] N. Koudas, S. Sarawagi, and D. Srivastava. Record linkage: similarity measures and algorithms. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 802–803. ACM, 2006.

[26] P. Li. Min-max kernels. Technical report, arXiv:1503.0173, 2015.

[27] P. Li and A. C. König. Theory and applications b-bit minwise hashing. *Commun. ACM*, 2011.

[28] P. Li, M. Mitzenmacher, and A. Shrivastava. Coding for random projections and approximate near neighbor search. Technical report, arXiv:1403.8144, 2014.

[29] P. Li, A. B. Owen, and C.-H. Zhang. One permutation hashing. In *NIPS*, Lake Tahoe, NV, 2012.

[30] M. Manasse, F. McSherry, and K. Talwar. Consistent weighted sampling. Technical Report MSR-TR-2010-73, Microsoft Research, 2010.

[31] S. Melnik and H. Garcia-Molina. Adaptive algorithms for set containment joins. *ACM Transactions on Database Systems (TODS)*, 28(1):56–99, 2003.

[32] B. Neyshabur and N. Srebro. A simpler and better lsh for maximum inner product search (mips). Technical report, arXiv:1410.5518, 2014.

[33] A. Rajaraman and J. Ullman. *Mining of Massive Datasets*. http://i.stanford.edu/ ullman/mmds.html.

[34] K. Ramasamy, J. F. Naughton, and R. Kaushik. Set containment joins: The good, the bad and the ugly.

[35] A. Shrivastava and P. Li. Asymmetric LSH (ALSH) for sublinear time maximum inner product search (mips). In *NIPS*, Montreal, CA, 2014.

[36] A. Shrivastava and P. Li. Densifying one permutation hashing via rotation for fast near neighbor search. In *ICML*, Beijing, China, 2014.

[37] A. Shrivastava and P. Li. Improved asymmetric locality sensitive hashing (ALSH) for maximum inner product search (MIPS). *arXiv:1410.5410 (submitted to AISTATS)*, 2014.

[38] A. Shrivastava and P. Li. Improved densification of one permutation hashing. In *UAI*, Quebec City, CA, 2014.

[39] A. Shrivastava and P. Li. In defense of minhash over simhash. In *AISTATS*, 2014.

[40] R. Weber, H.-J. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *VLDB*, pages 194–205, 1998.