Norm-Ranging LSH for Maximum Inner Product Search

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Neyshabur and Srebro proposed SIMPLE-LSH [2015], which is the state-of-the-art hashing method for maximum inner product search (MIPS) with performance guarantee. We found that the performance of SIMPLE-LSH, in both theory and practice, suffers from long tails in the 2-norm distribution of real datasets. We propose NORM-RANGING LSH, which addresses the excessive normalization problem caused by long tails in SIMPLE-LSH by partitioning a dataset into multiple sub-datasets and building a hash index for each sub-dataset independently. We prove that NORM-RANGING LSH has lower query time complexity than SIMPLE-LSH. We also show that the idea of partitioning the dataset can improve other hashing based methods for MIPS. To support efficient query processing on the hash indexes of the sub-datasets, a novel similarity metric is formulated. Experiments show that NORM-RANGING LSH achieves an order of magnitude speedup over SIMPLE-LSH for the same recall, thus significantly benefiting applications that involve MIPS.

1 Introduction

Given a dataset $\mathcal{S} \subset \mathbb{R}^d$ containing n vectors (also called items) and a query $q \in \mathbb{R}^d$, maximum inner product search (MIPS) finds the vector in $\mathcal{S}$ that has the maximum inner product with q,

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

MIPS may require items with the top k inner products and it usually suffices to return approximate results (i.e., items with inner products close to the maximum). MIPS has many important applications including recommendation based on user and item embeddings obtained from matrix factorization [Koren et al., 2009], multi-class classification with linear classifier [Dean et al., 2013], filtering in computer vision [Felzenszwalb et al., 2010], etc.

MIPS is a challenging problem as modern datasets often have high dimensionality and large cardinality. Initially, tree-based methods [Ram and Gray, 2012, Koenigstein et al., 2012] were proposed for MIPS, which use the idea of branch and bound similar to k-d tree [Friedman and Tukey, 1974]. However, these methods suffer from the curse of dimensionality and their performance can be even worse than linear scan for feature dimension as low as 20 [Weber et al., 1998]. Shrivastava and Li proposed L2-ALSH [2014], which attains the first provable sub-linear query time complexity guarantee for approximate MIPS that is independent of dimensionality. L2-ALSH applies an asymmetric transformation [1] to transform MIPS into L_2 similarity search, which can be solved with well-known LSH functions. Following the idea of L2-ALSH, Shrivastava and Li formulated another pair of asymmetric transformations called SIGN-ALSH [2015] to transform MIPS into angular similarity search and obtained better query time complexity guarantee than that of L2-ALSH.

---

[1] Asymmetric transformation means that the transformations for the queries and the items are different, while symmetric transformation means the same transformation is applied to the items and queries.

Neyshabur and Srebro showed that asymmetry is not necessary when queries are normalized and items have bounded 2-norm [2015]. They proposed SIMPLE-LSH, which adopts a symmetric transformation and transforms MIPS into angular similarity search similar to SIGN-ALSH. However, they proved that SIMPLE-LSH is a universal LSH for MIPS, while L2-ALSH and SIGN-ALSH are not. SIMPLE-LSH is also parameter-free and avoids the parameter tuning of L2-ALSH and SIGN-ALSH. Most importantly, SIMPLE-LSH achieves superior performance over L2-ALSH and SIGN-ALSH in both theory and practice, and thus is the state-of-the-art hashing algorithm for MIPS.

SIMPLE-LSH requires the 2-norms of the items to be bounded, which is achieved by normalizing the items by the largest 2-norm in the dataset. However, real datasets often have long tails in the distribution of 2-norm, meaning that the largest 2-norm can be much larger than the majority of the items. As we will show in Section 3.1, the excessive normalization process of SIMPLE-LSH makes the maximum inner product between the query and the items small, which degrades the performance of SIMPLE-LSH in both theory and practice.

To solve this problem, we propose NORM-RANGING LSH. The idea is to partition the original dataset into multiple sub-datasets according to the percentiles of the 2-norm distribution. For each sub-dataset, NORM-RANGING LSH uses SIMPLE-LSH as a subroutine to build an index independent of other sub-datasets. As each sub-dataset is normalized by its own maximum 2-norm, which is usually significantly smaller than the maximum 2-norm in the entire dataset, NORM-RANGING LSH achieves a lower asymptotic query time complexity bound than SIMPLE-LSH. To support efficient query processing, we also formulate a novel similarity metric which defines a probing order for buckets from different sub-datasets. We compare NORM-RANGING LSH with SIMPLE-LSH and L2-ALSH on three real datasets and show empirically that NORM-RANGING LSH offers an order of magnitude speedup for achieving the same recall.

2 Locality sensitive hashing for MIPS

2.1 Locality sensitive hashing

A formal definition of locality sensitive hashing (LSH) [Indyk and Motwani, 1998] is given as follows:

Definition 1. *(Locality Sensitive Hashing) A family $\mathcal{H}$ is called (S_0, cS_0, p_1, p_2)-sensitive if, for any two vectors $x, y \in \mathbb{R}^d$:*

- *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 a family $\mathcal{H}$ of LSH functions to be useful, it is required that $p_1 > p_2$ and $0 < c < 1$. Given a family of (S_0, cS_0, p_1, p_2)-sensitive hash functions, one can answer a query for c-approximate nearest neighbor search [2] with a time complexity of $\mathcal{O}(n^\rho \log n)$, where $\rho = \frac{\log p_1}{\log p_2}$. For L_2 distance, there exists a well-known family of LSH functions defined as follows:

$$h_{a,b}^{L_2}(x) = \left\lfloor \frac{a^\top x + b}{r} \right\rfloor, \tag{2}$$

where $\lfloor \rfloor$ is the floor operation, a is a random vector whose entries follow i.i.d. standard normal distribution and b is generated by a uniform distribution over $[0, r]$. The probability that two vectors x and y are hashed to the same value under (2) is given as:

$$Pr\left[h_{a,b}^{L_2}(x) = h_{a,b}^{L_2}(y)\right] = F_r(d) = 1 - 2\Phi(-\frac{r}{d}) - \frac{2d}{\sqrt{2\pi}r}(1 - e^{-(r/d)^2/2}), \tag{3}$$

in which $\Phi(x)$ is the cumulative density function of standard normal distribution and $d = \|x - y\|$ is the L_2 distance between x and y. For angular similarity, sign random projection is an LSH. Its expression and collision probability can be given as [Goemans and Williamson, 1995]:

$$h_a(x) = \text{sign}(a^\top x), Pr[h_a(x) = h_a(y)] = 1 - \frac{1}{\pi}\cos^{-1}\left(\frac{x^\top y}{\|x\|\|y\|}\right), \tag{4}$$

where the entries of a follow i.i.d. standard normal distribution.

---

[2] c-approximate nearest neighbor search solves the following problem: given parameters $S_0 > 0$ and $\delta > 0$, if there exists an S_0-near neighbor of q in $\mathcal{S}$, return some cS_0-near neighbor in $\mathcal{S}$ with probability at least $1 - \delta$.

2.2 L2-ALSH

Shrivastava and Li proved that there exists no symmetric LSH for MIPS if the domain of the item x and query q are both $\mathbb{R}^d$ [2014]. They proposed to apply a pair of asymmetric transformations, $P(x)$ and $Q(q)$, to the items and the query, respectively.

$$P(x) = [Ux; \|Ux\|^2; \|Ux\|^4; ...; \|Ux\|^{2^m}]; \quad Q(q) = [q; 1/2; 1/2; ...; 1/2] \tag{5}$$

The scaling factor U should ensure that $\|Ux\| < 1$ for all $x \in \mathcal{S}$ and the query is normalized to unit 2-norm before the transformation. After the transformation, we have:

$$\|P(x) - Q(q)\|^2 = 1 + \frac{m}{4} - 2Ux^\top q + \|Ux\|^{2^{m+1}}. \tag{6}$$

As the scaling factor U is common for all items and the last term vanishes with sufficiently large m because $\|Ux\| < 1$, (6) shows that MIPS is transformed into finding the nearest neighbor of $Q(q)$ in L_2 distance, which can be solved using the hash function in (2). Given S_0 and c, a query time guarantee of $\mathcal{O}(n^\rho \log n)$ can be obtained for c-approximate MIPS with:

$$\rho = \frac{\log F_r(\sqrt{1 + m/4 - 2US_0 + (US_0)^{2^{m+1}}})}{\log F_r(\sqrt{1 + m/4 - 2cUS_0})}. \tag{7}$$

It is suggested to use a grid search to find the values of the parameters (m, U and r) that minimize ρ.

2.3 SIMPLE-LSH

Neyshabur and Srebro proved that L2-ALSH is not a universal LSH for MIPS, that is, for any setting of m, U and r, there always exists a pair of S_0 and c such that $x^\top q = S_0$ and $y^\top q = cS_0$ but $p_1 = [h_{a,b}^{L_2}(P(x)) = h_{a,b}^{L_2}(Q(q))] < p_2 = [h_{a,b}^{L_2}(P(y)) = h_{a,b}^{L_2}(Q(q))]$ [2015]. Moreover, they showed that asymmetry is not necessary if the items have bounded 2-norm and the query is normalized, which is exactly the assumption of L2-ALSH. They proposed a symmetric transformation to transform MIPS into angular similarity search as follows:

$$P(x) = [x; \sqrt{1 - \|x\|^2}]; P(q)^\top P(x) = [q; 0]^\top [x; \sqrt{1 - \|x\|^2}] = q^\top x. \tag{8}$$

They apply the sign random projection in (4) to $P(x)$ and $P(q)$ to obtain an LSH for c-approximate MIPS with a query time complexity $\mathcal{O}(n^\rho \log n)$ and ρ is given as:

$$\rho = G(c, S_0) = \frac{\log(1 - \frac{\cos^{-1}(S_0)}{\pi})}{\log(1 - \frac{\cos^{-1}(cS_0)}{\pi})}. \tag{9}$$

They called their scheme SIMPLE-LSH as it avoids the parameter tuning process of L2-ALSH. Moreover, SIMPLE-LSH is proved to be a universal LSH for MIPS under any valid configuration of S_0 and c. SIMPLE-LSH also obtains better (lower) ρ values than L2-ALSH and SIGN-ALSH in theory and outperforms both of them empirically on real datasets [Shrivastava and Li, 2015].

3 Norm-Ranging LSH

In this section, we first motivate norm-ranging LSH by showing the problem of SIMPLE-LSH on real datasets, then introduce how norm-ranging LSH (or RANGE-LSH for short) solves the problem.

3.1 SIMPLE-LSH on real datasets

We plot the relation between ρ and S_0 for SIMPLE-LSH under different values of c in Figure 1(a). Recall that the theoretical query time complexity of SIMPLE-LSH is $\mathcal{O}(n^\rho \log n)$ and observe that ρ is a decreasing function of S_0. As ρ is large when S_0 is small, SIMPLE-LSH suffers from poor query performance when the maximum inner product between a query and the items is small. To conduct the transformation in (8), SIMPLE-LSH requires the 2-norm of the items to be bounded by 1, which is usually achieved by dividing the items by the maximum 2-norm $U = \max_{x \in \mathcal{S}} \|x\|$. Assuming
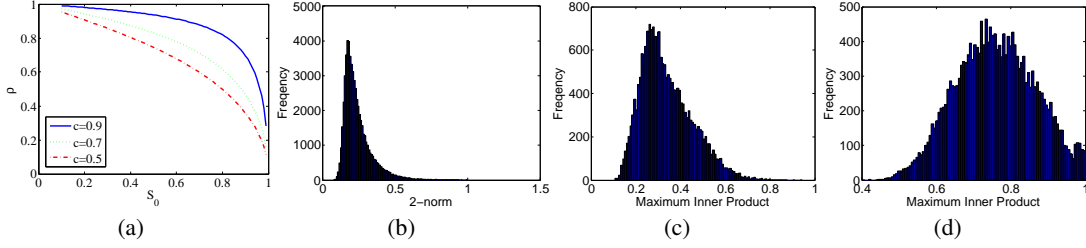
Figure 1: (a) The relation between ρ and S_0; (b) 2-norm distribution of the SIFT descriptors from the ImageNet dataset (maximum 2-norm scaled to 1); (c) The distribution of the maximum inner product of the queries after the normalization process of SIMPLE-LSH; (d) The distribution of the maximum inner product of the queries after the normalization process of RANGE-LSH (32 sub-datasets).

Algorithm 1 Norm-Ranging LSH: Index Building

1: **Input:** Dataset $\mathcal{S}$, size n, number of sub-datasets m
2: **Output:** A hash index $\mathcal{I}_j$ for each of the m sub-datasets
3: Rank the items in $\mathcal{S}$ according to their 2-norms;
4: Partition $\mathcal{S}$ into m sub-datasets $\{\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_m\}$ such that $\mathcal{S}_j$ holds items whose 2-norms ranked in the range $[\frac{(j-1)n}{m}, \frac{jn}{m}]$;
5: **for** every sub-dataset $\mathcal{S}_j$ **do**
6:     Use $U_j = \max_{x \in \mathcal{S}_j} \|x\|$ to normalize $\mathcal{S}_j$;
7:     Apply SIMPLE-LSH to build index $\mathcal{I}_j$ for $\mathcal{S}_j$;
8: **end for**

$q^\top x = S$ for an original item vector x, we have $q^\top x = S/U$ after normalization. If U is significantly larger than $\|x\|$, the inner product will be scaled to a small value.

We plot the distribution of the 2-norm of a real dataset in Figure 1(b). The distribution has a long tail and the maximum 2-norm is much larger than the majority of the items. We also plot in Figure 1(c) the distribution of the maximum inner product of the queries after the normalization process of SIMPLE-LSH. The results show that for the majority of the queries, the maximum inner product is small, which translates into a large ρ and poor theoretical query performance.

The long tail distribution of real datasets also harms the performance of SIMPLE-LSH in practice. If $\|x\|$ is small after normalization, the $\sqrt{1 - \|x\|^2}$ term, which is irrelevant to the inner product between x and q, will be dominant in $P(x) = [x; \sqrt{1 - \|x\|^2}]$. In this case, the result of sign random projection in (4) will be largely determined by the last entry of a, causing many items to be gathered in the same bucket. In our sample run of SIMPLE-LSH on the ImageNet dataset [Deng et al., 2009] with a code length of 32, there are only 60,000 buckets and the largest bucket holds about 200,000 items. Considering that the ImageNet dataset contains roughly 2 million items and 32-bit code offers approximately 4×10^9 buckets, these statistics show that the large $\sqrt{1 - \|x\|^2}$ term severely degrades bucket balance in SIMPLE-LSH. Bucket balance is important for the performance of binary hashing algorithms such as SIMPLE-LSH because they use Hamming distance to determine the probing order of the buckets [Cai, 2016, Gong et al., 2013]. If the number of buckets is small or some buckets contain too many items, Hamming distance cannot define a good probing order for the items, which results in poor query performance.

3.2 Norm-Ranging LSH

The index building and query processing procedures of RANGE-LSH are presented in Algorithm 1 and Algorithm 2, respectively. To solve the excessive normalization problem of SIMPLE-LSH, RANGE-LSH partitions the items into m sub-datasets according to the percentiles of the 2-norm distribution so that each sub-dataset contains items with similar 2-norms. Note that ties are broken arbitrarily in the ranking process of Algorithm 1 to ensure that the percentiles based partitioning works even when many items have the same 2-norm. Instead of using U, i.e., the maximum 2-norm in the entire dataset, SIMPLE-LSH uses the local maximum 2-norm $U_j = \max_{x \in \mathcal{S}_j} \|x\|$ in each sub-dataset for

Algorithm 2 Norm-Ranging LSH: Query

1: **Input:** Hash indexes $\{\mathcal{I}_1, \mathcal{I}_2, ..., \mathcal{I}_m\}$ for the sub-datasets, query q
2: **Output:** A c-approximate answer to q
3: **for** every hash index $\mathcal{I}_j$ **do**
4:     Conduct MIPS with q to get $x_j^\star$;
5: **end for**
6: Select the item in $\{x_1^\star, x_2^\star, ..., x_m^\star\}$ that has the maximum inner product with q as the answer;

normalization, so as to keep the inner products of the queries large. In Figure 1(d), we plot the maximum inner product of the queries after the normalization process of RANGE-LSH. Comparing with Figure 1(c), the values of the inner product are significantly larger. As a result, the ρ_j of sub-dataset $\mathcal{S}_j$ becomes $\rho_j = G(c, S_0/U_j)$, which is smaller than $\rho = G(c, S_0/U)$ if $U_j < U$. The smaller ρ values translate into better query performance. In the following, we prove that RANGE-LSH achieves a lower query time complexity bound than SIMPLE-LSH while providing the same quality guarantee.

Theorem 1. *For any $0 < S_0 < 1$ and $0 < c < 1$, the query time complexity upper bound of* RANGE-LSH *for solving c-approximate MIPS is lower than that of* SIMPLE-LSH *with sufficiently large n, if the dataset is divided into $m = n^\alpha$ sub-datasets with $\alpha < \min\{\rho, \frac{\rho - \rho^\star}{1 - \rho^\star}\}$ and there are at most n^β sub-datasets satisfying $U_j = U$ with $0 < \beta < \alpha\rho$. The parameters are given as $\rho = G(c, S_0/U)$, $\rho^\star = \max_{\rho_j < \rho} \rho_j$ and $\rho_j = G(c, S_0/U_j)$.*

Proof. Firstly, we prove the correctness of RANGE-LSH, that is, it indeed returns a cS_0 approximate answer with probability at least $1 - \delta$. Note that S_0 is a pre-specified parameter common to all sub-datasets rather than the actual maximum inner product in each sub-dataset. If there is an item $x^\star$ having an inner product of S_0 with q in the original dataset, it is certainly contained in one of the sub-datasets. When we conduct MIPS on all the sub-datasets, the sub-dataset containing $x^\star$ will return an item having inner product cS_0 with q with probability at least $1 - \delta$ according to the guarantee of SIMPLE-LSH. The final query result is obtained by selecting the optimal one (the one having the largest inner product with q) from the query answers of all sub-dataset according to Algorithm 2, which is guaranteed to be no less than cS_0 with probability at least $1 - \delta$.

Now we analyze the complexity of RANGE-LSH. For each sub-dataset $\mathcal{S}_j$, it contains $n^{1-\alpha}$ items and the query time complexity upper bound of c-approximate MIPS is $\mathcal{O}(n^{(1-\alpha)\rho_j} \log n^{1-\alpha})$ with $\rho_j = G(c, S_0/U_j)$. As there are $m = n^\alpha$ sub-datasets, the time complexity of selecting the optimal one from the answers of all sub-datasets is $\mathcal{O}(n^\alpha)$. Considering ρ_j is an increasing function of U_j and there are n^β sub-datasets with $U_j = U$, the query time complexity of RANGE-LSH can be expressed as:

$$
\begin{aligned}
f(n) &= n^\alpha + \sum_{j=1}^{n^\alpha} n^{(1-\alpha)\rho_j} \log n^{1-\alpha} < n^\alpha + \sum_{j=1}^{n^\alpha} n^{(1-\alpha)\rho_j} \log n \\
&= n^\alpha + \sum_{j=1}^{n^\alpha - n^\beta} n^{(1-\alpha)\rho_j} \log n + n^\beta n^{(1-\alpha)\rho} \log n \\
&< n^\alpha + n^\alpha n^{(1-\alpha)\rho^\star} \log n + n^\beta n^{(1-\alpha)\rho} \log n
\end{aligned}
\tag{10}
$$

Strictly speaking, the equal sign in the first line of (10) is not rigorous as the constants and non-dominant terms in the complexity of querying each sub-dataset are ignored. However, we are interested in the order rather than the precise value of query time complexity, so the equal sign is used for the conciseness of expression. Comparing $f(n)$ with the $\mathcal{O}(n^\rho \log n)$ complexity of SIMPLE-LSH,

$$
\begin{aligned}
\frac{f(n)}{n^\rho \log n} &< \frac{n^\alpha + \big(n^\alpha n^{(1-\alpha)\rho^\star} + n^\beta n^{(1-\alpha)\rho}\big) \log n}{n^\rho \log n} \\
&= n^{\alpha - \rho}/\log n + n^{\alpha + (1-\alpha)\rho^\star - \rho} + n^{\beta - \alpha\rho}
\end{aligned}
\tag{11}
$$

(11) tends to 0 with sufficiently large n when $\alpha \leq \rho$, $\alpha + (1 - \alpha)\rho^\star < \rho$ and $\beta - \alpha\rho < 0$, which is satisfied by $\alpha < \min\{\rho, \frac{\rho - \rho^\star}{1 - \rho^\star}\}$ and $\beta < \alpha\rho$. $\qquad\square$

Note that the conditions of Theorem (1) can be easily satisfied. Theorem (1) imposes an upper bound instead of a lower bound on the number of sub-datasets, which is favorable as we do not want to partition the original dataset into a large number of sub-datasets. Moreover, the condition that the number of sub-datasets with $U_j = U$ is smaller than $n^{\alpha\rho}$ is easily satisfied as very often only the sub-dataset that contains the items with the largest 2-norms has $U_j = U$. The proof also shows that RANGE-LSH is not limited to datasets with long tail in the 2-norm distribution. As long as $U > U_j$ holds for most sub-datasets, RANGE-LSH can provide better performance than SIMPLE-LSH. We acknowledge that RANGE-LSH and SIMPLE-LSH are equivalent when all items have the same 2-norm. However, MIPS is also equivalent to angular similarity search without transformation in this case, and thus can be solved with sign random projection instead of SIMPLE-LSH.

The lower theoretical query time complexity of RANGE-LSH also translates into much better bucket balance in practice. On the ImageNet dataset, RANGE-LSH with 32-bit code maps the items to approximately 2 million buckets and most buckets contain only 1 item. Comparing with the statistics of SIMPLE-LSH in Section 3.1, these numbers show that RANGE-LSH has much better bucket balance, and thus better ability to define a good probing order for the items. This can be explained by the fact that RANGE-LSH uses more moderate scaling factors for each sub-dataset than SIMPLE-LSH, thus significantly reducing the magnitude of the $\sqrt{1 - \|x\|^2}$ term in $P(x) = [x; \sqrt{1 - \|x\|^2}]$.

3.3 Similarity metric

Algorithm 2 needs to probe the hash indexes of all sub-datasets independently and the query result can only be obtained after all hash indexes are probed. In practice, it is more favorable to probe the buckets of different sub-datasets simultaneously in a manner similar to Hamming ranking, which motivates the design of a metric to rank the buckets across the indexes of different sub-datasets.

Combining the index building process of RANGE-LSH and the collision probability of sign random projection in (4), the probability that an item $x \in \mathcal{S}_j$ and the query collide on one bit can be given as $p = 1 - \frac{1}{\pi}\cos^{-1}\left(\frac{q^\top x}{U_j}\right)$, where U_j is the maximum 2-norm in sub-dataset $\mathcal{S}_j$. When the code length is K, the expectation of the number of identical bits between query q and item $x \in \mathcal{S}_j$ can be expressed as:

$$E(k) = Kp = K\left(1 - \frac{1}{\pi}\cos^{-1}\left(\frac{q^\top x}{U_j}\right)\right). \tag{12}$$

For $z > 0$, we have $\frac{\pi}{2} - \frac{\pi}{2}z \le \cos^{-1}(z) \le \frac{\pi}{2} - z$. Plugging it into (12) and rearranging the terms:

$$\frac{K}{\pi}q^\top x \le \left[E(k) - \frac{K}{2}\right]U_j \le \frac{K}{2}q^\top x. \tag{13}$$

(13) shows that $[E(k) - \frac{K}{2}]U_j$ can serve as an indicator of the inner product between item x and query q (extending to the case $q^\top x < 0$ is trivial). In practice, we cannot obtain $E(k)$, but the actual number of identical bits between bucket b and query q, denoted as $I(q, b)$, can be used as a surrogate. When $I(q, b) - \frac{K}{2} > 0$, larger U_j indicates higher inner product while the opposite is true when $I(q, b) - \frac{K}{2} < 0$. Since the code length is limited and $I(q, b)$ can diverge from $E(k)$, it is possible that a bucket has large U_j and large inner product with q, but it happens that $I(q, b) < \frac{K}{2}$. In this case, it will be probed late in the query process, hurting query performance. To alleviate this problem, we adjust the similarity indicator to $[I(q, b) - \frac{K}{2} + k_0]U_j$, where k_0 is a small positive integer.

Note that our similarity metric $[I(q, b) - \frac{K}{2} + k_0]U_j$ can be manipulated with a complexity similar to Hamming distance. We can calculate the values of $[I(q, b) - \frac{K}{2} + k_0]U_j$ for all possible combinations of $I(q, b)$ and U_j, and sort them during index building. Note that the sorted structure is common for all queries and does not take too much space [3]. When a query comes, query processing can be conducted by traversing the sorted structure in ascending order. For a pair of $(U_j, I(q, b))$, U_j determines the sub-dataset while $I(q, b)$ is used to choose the buckets to probe in that sub-dataset with standard hash lookup. Moreover, an analysis (see the supplementary material) shows that our similarity metric is also amendable to generate-to-probe scheme based on the inverted multi-index [Babenko and Lempitsky, 2012].

---

[3] $I(q, b)$ can take $K + 1$ values, U_j can take m values, so the size of the sorted structure is $mK + m$.
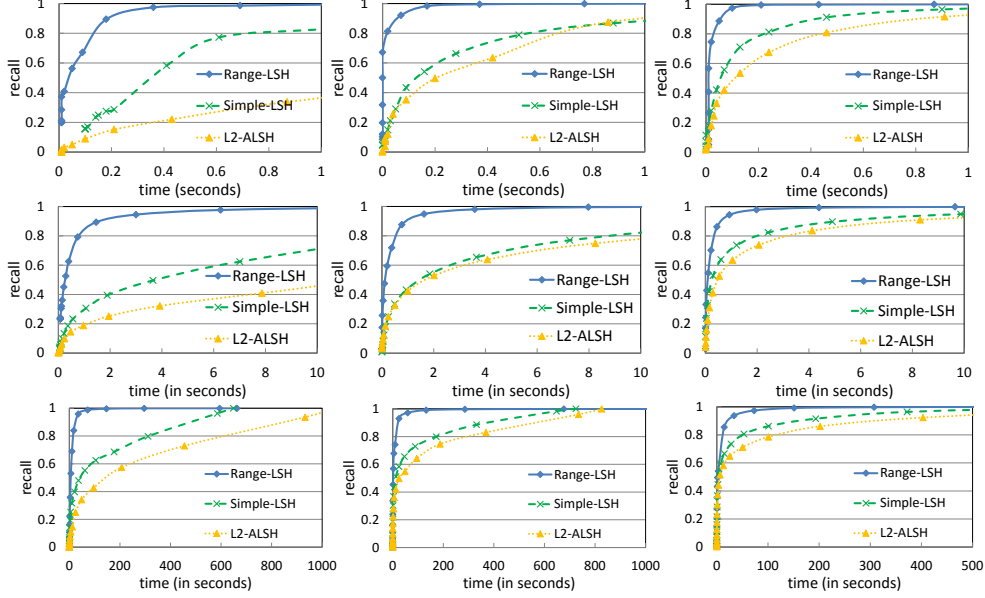
Figure 2: Recall versus running time for top 10 MIPS on Netflix (top row), Yahoo!Music (middle row), and ImageNet (bottom row). From left to right, the code lengths are 16, 32 and 64, respectively.

4 Experimental results

We used three popular datasets, i.e., Netflix, Yahoo!Music and ImageNet. For the Netflix dataset and Yahoo!Music dataset, the user and item embeddings were obtained using alternating least square based matrix factorization [Yun et al., 2013], and each embedding has 300 dimensions. We used the item embeddings as dataset items and the user embeddings as queries. The ImageNet dataset contains more than 2 million SIFT descriptors of the ImageNet images, and some randomly selected SIFT descriptors are used as queries and the rest are used as dataset items. Note that the 2-norm distributions of the Netflix and Yahoo!Music embeddings do not have long tail and the maximum 2-norm is close to the median (see the supplementary material), which helps verify the robustness of RANGE-LSH to different 2-norm distributions. Different from previous empirical studies [Neyshabur and Srebro, 2015, Shrivastava and Li, 2014], which used small datasets containing no more than 20,000 items, our experiments covered the spectrum from small-scale to large-scale datasets with Netflix (17,770 items), Yahoo!Music (136,736 items), and ImageNet (2,340,373 items). For each dataset, we report the average performance of 1,000 randomly selected queries.

We compared RANGE-LSH with SIMPLE-LSH and L2-ALSH. For L2-ALSH, we used the parameter setting recommended by its authors, i.e., $m = 3$, $U = 0.83$, $r = 2.5$. For RANGE-LSH, part of the bits in the binary code are used to encode the index of the sub-datasets and the rest are generated by hashing. For example, if the code length is 16 and the dataset is partitioned into 32 sub-datasets, the 16-bit code of RANGE-LSH consists of 5 bits for indexing the 32 sub-datasets, while the remaining 11 bits are generated by hashing. We partitioned the dataset into 32, 64 and 128 sub-datasets under a code length of 16, 32 and 64, respectively. For fairness of comparison, all algorithms use the same total code length.

We plot the time-recall curves in Figure 2 as it is critical to reach a high recall in a short time for many applications. The results show that RANGE-LSH achieves significant speedups compared with SIMPLE-LSH and L2-ALSH at the same recall. To be more specific, SIMPLE-LSH takes approximately 16, 18 and 18 times of the running time of RANGE-LSH to achieve a 90% recall under 32-bit code on the three datasets. Due to space limitation, we only report the performance of the top 10 MIPS, the performance under more configurations can be found in the supplementary material.

Recall that Algorithm 1 partitions a dataset into sub-datasets according to percentiles in the 2-norm distribution. We tested an alternative partitioning scheme, which divides the domain of 2-norms into *uniformly* spaced ranges and items falling in the same range are partitioned into the same sub-dataset.
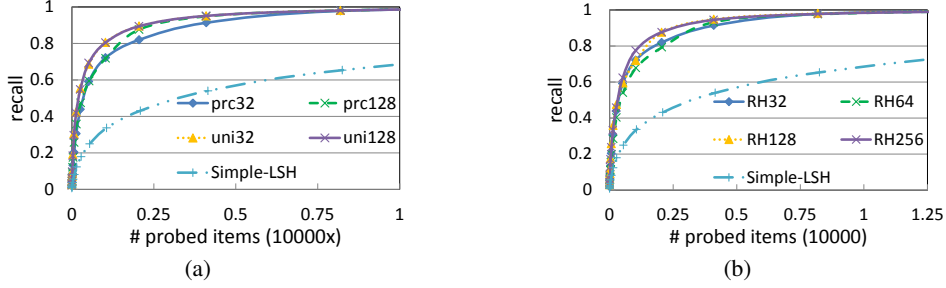
Figure 3: (a) Comparison between percentile based partitioning and uniform partitioning (best viewed in color), the code length is 32 bit and the dataset is Yahoo!Music. prc32 and uni32 mean percentile and uniform partitioning with 32 sub-datasets, respectively. (b) The influence of the number of sub-datasets on the performance on the Yahoo!Music dataset, the code length is 32 bit and the number of sub-datasets varies from 32 to 256. RH32 means RANGE-LSH with 32 sub-datasets.

The results are plotted in Figure 3(a), which shows that uniform partitioning achieves slightly better performance than percentile partitioning. This proves RANGE-LSH is general and robust to different partitioning methods as long as items with similar 2-norms are into the same sub-dataset. We also experimented the influence of the number of sub-datasets on performance in Figure 3(b). The results show that performance improves with the number of sub-datasets when the number of sub-datasets is still small, but stabilizes when the number of sub-datasets is sufficiently large.

5 Extension to L2-ALSH

In this section, we show that the idea of RANGE-LSH, which divides the original dataset into sub-datasets with similar 2-norms, can also be applied to L2-ALSH [Shrivastava and Li, 2014] to obtain more favorable ρ values than (7). Note that we get (7) from (6) as we only have $0 \leq \|x\| \leq S_0$ if the entire dataset is considered. For a sub-dataset $\mathcal{S}_j$, if we have the range of its 2-norms as $l_j \leq \|x\| \leq u_j$ and $l_j > 0$, we can obtain the ρ_j of $\mathcal{S}_j$ as:

$$\rho_j = \frac{\log F_r(\sqrt{1 + m/4 - 2U_j S_0 + (U_j u_j)^{2^{m+1}}})}{\log F_r(\sqrt{1 + m/4 - 2cU_j S_0 + (U_j l_j)^{2^{m+1}}})}. \tag{14}$$

As $u_j \leq S_0$ and $l_j > 0$, the denominator in (14) increases in absolute value while the numerator decreases in absolute value compared with (7). Therefore, we have $\rho_j < \rho$. Moreover, dividing the original dataset into sub-datasets allows us to use different normalization factor U_j for each sub-dataset and we only need to satisfy $U_j < 1/u_j$ rather than $U < 1/\max_{x \in \mathcal{S}} \|x\|$, which allows more flexibility for parameter optimization. Similar to Theorem (1), it can also be proved that dividing the dataset into sub-datasets results in an algorithm with lower query time complexity than the original L2-ALSH. The same idea can also be trivially extended to obtain more favorable ρ values for SIGN-ALSH [Shrivastava and Li, 2015].

6 Conclusions

Maximum inner product search (MIPS) has many important applications such as collaborative filtering and computer vision. We showed that, SIMPLE-LSH, the state-of-the-art hashing method for MIPS, has critical performance limitations due to the long tail in the 2-norm distribution of real datasets. To tackle this problem, we proposed RANGE-LSH, which attains provably lower query time complexity than SIMPLE-LSH. In addition, we also formulated a novel similarity metric that can be processed with low complexity. The experimental results showed that RANGE-LSH significantly outperforms SIMPLE-LSH, and RANGE-LSH is robust to the shape of 2-norm distribution and different partitioning methods. We also showed that the idea of SIMPLE-LSH hashing is general and can be applied to boost the performance of other hashing based methods for MIPS. The superior performance of RANGE-LSH can benefit many applications that involve MIPS.

8

References

A. Babenko and V. S. Lempitsky. The inverted multi-index. In *CVPR*, pages 3069–3076, 2012.

D. Cai. A revisit of hashing algorithms for approximate nearest neighbor search. *CoRR*, 2016.

T. L. 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 *CVPR*, pages 1814–1821, 2013.

J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and F. F. Li. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009.

P. F. Felzenszwalb, R. B. Girshick, D. A. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32:1627–1645, 2010.

J. H. Friedman and J. W. Tukey. A projection pursuit algorithm for exploratory data analysis. *IEEE Trans. Computers*, 23:881–890, 1974.

M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *JACM*, 42:1115–1145, 1995.

Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35: 2916–2929, 2013.

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

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

Y. Koren, R. M. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42:30–37, 2009.

B. Neyshabur and N. Srebro. On symmetric and asymmetric lshs for inner product search. In *ICML*, pages 1926–1934, 2015.

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

A. Shrivastava and P. Li. Asymmetric LSH (ALSH) for sublinear time maximum inner product search (MIPS). In *NIPS*, pages 2321–2329, 2014.

A. Shrivastava and P. Li. Improved asymmetric locality sensitive hashing (ALSH) for maximum inner product search (MIPS). In *UAI*, pages 812–821, 2015.

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

H. Yun, H. F. Yu, C.J. Hsieh, S. V. N. Vishwanathan, and I. S. Dhillon. NOMAD: non-locking, stochastic multi-machine algorithm for asynchronous and decentralized matrix completion. *CoRR*, 2013.