Norm-Ranging LSH for Maximum Inner Product Search

Xiao Yan, Jinfeng Li, Xinyan Dai, Hongzhi Chen, James Cheng
Department of Computer Science
The Chinese University of Hong Kong
Shatin, Hong Kong
{xyan, jfli, xydai, hzchen, jcheng}@cse.cuhk.edu.hk

Abstract

Neyshabur and Srebro proposed SIMPLE-LSH [2015], which is the state-of-the-art hashing based algorithm for maximum inner product search (MIPS). 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 by partitioning a dataset into sub-datasets and building a hash index for each sub-dataset independently. We prove that NORM-RANGING LSH achieves lower query time complexity than SIMPLE-LSH under mild conditions. We also show that the idea of dataset partitioning can improve another hashing based MIPS algorithm. Experiments show that NORM-RANGING LSH probes much less items than SIMPLE-LSH at the same recall, thus significantly benefiting MIPS based applications.

1 Introduction

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

$$p = \arg \max_{x \in S} q^\top x. \quad (1)$$

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 when feature dimension is 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 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 lower query time complexity 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.
We compare the definition of locality sensitive hashing (LSH) \cite{Indyk1998,Andoni2018} where the entries of a random vector whose entries follow i.i.d. standard normal distribution. For angular similarity, sign random projection is an LSH. Its collision probability can be given as \cite{Goemans1995}:

\[ P_H[h(x) = h(y)] \leq c \cdot \min\{p_1, p_2\}, \]

where \( c \) is a constant. Given a family of \((S_0, c \cdot S_0, p_1, p_2)\)-LSH, a query for \( c \)-approximate nearest neighbor search\footnote{\(c\)-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 \( S \), return some \( c \cdot S_0 \)-near neighbor in \( S \) with probability at least \( 1 - \delta \).} can be processed with a time complexity of \( O(n^\rho \log n) \), where \( \rho = \frac{\log p_1}{\log p_2} \). For \( L_2 \)-distance, there exists a well-known family of LSH defined as follows:

\[ h_{a,b}(x) = \left\lfloor \frac{a^\top x + b}{r} \right\rfloor, \]

where \( \lfloor \cdot \rfloor \) 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 \([0, r]\). When a hash function is drawn randomly and independently for each pair of vectors \cite{Wang2013}, the collision probability of (2) is given as:

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

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 \cite{Goemans1995}:

\[ h_a(x) = \text{sign}(a^\top x), \quad P_{H_a}[h_a(x) = h_a(y)] = 1 - \frac{1}{\pi} \cos^{-1} \left( \frac{x^\top y}{\|x\|\|y\|} \right), \]

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

## 2 Locality Sensitive Hashing for MIPS

### 2.1 Locality Sensitive Hashing

A definition of locality sensitive hashing (LSH) \cite{Indyk1998,Andoni2018} is given as follows:

**Definition 1. (LSH)** A family \( H \) is called \((S_0, c \cdot S_0, p_1, p_2)\)-sensitive if, for any two vectors \( x, y \in \mathbb{R}^d \):

- if \( \text{sim}(x, y) \geq S_0 \), then \( P_{H}[h(x) = h(y)] \geq p_1 \),
- if \( \text{sim}(x, y) \leq c \cdot S_0 \), then \( P_{H}[h(x) = h(y)] \leq p_2 \).

Note the original LSH is defined for distance function, we adopt a formalization adapted for similarity function \cite{Shrivastava2014}, which is more suitable for MIPS. For a family of LSH to be useful, it is required that \( p_1 > p_2 \) and \( 0 < c < 1 \). Given a family of \((S_0, c \cdot S_0, p_1, p_2)\)-LSH, a query for \( c \)-approximate nearest neighbor search\footnote{\(c\)-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 \( S \), return some \( c \cdot S_0 \)-near neighbor in \( S \) with probability at least \( 1 - \delta \).} can be processed with a time complexity of \( O(n^\rho \log n) \), where \( \rho = \frac{\log p_1}{\log p_2} \). For \( L_2 \)-distance, there exists a well-known family of LSH defined as follows:

\[ h_{a,b}(x) = \left\lfloor \frac{a^\top x + b}{r} \right\rfloor, \]

where \( \lfloor \cdot \rfloor \) 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 \([0, r]\). When a hash function is drawn randomly and independently for each pair of vectors \cite{Wang2013}, the collision probability of (2) is given as:

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

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 \cite{Goemans1995}:

\[ h_a(x) = \text{sign}(a^\top x), \quad P_{H_a}[h_a(x) = h_a(y)] = 1 - \frac{1}{\pi} \cos^{-1} \left( \frac{x^\top y}{\|x\|\|y\|} \right), \]

where the entries of \( a \) follow i.i.d. standard normal distribution.\footnote{\(c\)-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 \( S \), return some \( c \cdot S_0 \)-near neighbor in \( S \) 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 $\mathbb{R}^d$ \cite{Shrivastava2014}. They applied a pair of asymmetric transformations, $P(x)$ and $Q(q)$, to the items and the query, respectively.

$$P(x) = [U x; \|x\|^2; \|U x\|^4; ...; \|U x\|^{2^m}]$$

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

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

As the scaling factor $U$ is common for all items and the last term vanishes with sufficiently large $m$ because $\|Ux\| < 1$. \cite{Shrivastava2014} shows that the problem of MIPS is transformed into finding the nearest neighbor of $Q(q)$ in terms of $L_2$ distance, which can be solved using the hash function in \cite{Shrivastava2014}. Given $S_0$ and $c$, a query time complexity of $O(n^\rho \log n)$ can be obtained for $c$-approximate MIPS with:

$$\rho = \frac{\log F_r(\sqrt{1 + m/4 - 2US_0(S_0)^{2m+1}})}{\log F_r(\sqrt{1 + m/4 - 2cUS_0})}.$$  

It is suggested to use a grid search to find the parameters ($m$, $U$ and $r$) that minimize $\rho$.

2.3 SIMPLE-LSH

Neyshabur and Srebro proved that L2-ALSH is not a universal LSH for MIPS, that is \cite{Neyshabur2015}, 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 $P_H[h_{a,b}^L(P(x))] = h_{a,b}^L(Q(q)) < P_H[h_{a,b}^L(P(y))] = h_{a,b}^L(Q(q))$. 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:

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

They apply the sign random projection in \cite{Shrivastava2014} to $P(x)$ and $P(q)$ to obtain an LSH for $c$-approximate MIPS with a query time complexity $O(n^\rho \log n)$ and $\rho$ is given as:

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

They called their algorithm 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) $\rho$ values than L2-ALSH and SIGN-ALSH in theory and outperforms both of them empirically \cite{Shrivastava2014}.

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 $\rho$ and $S_0$ for SIMPLE-LSH in Figure 1(a). Recall that the query time complexity of SIMPLE-LSH is $O(n^\rho \log n)$ and observe that $\rho$ is a decreasing function of $S_0$. As $\rho$ 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. Before applying the transformation in \cite{Shrivastava2014}, SIMPLE-LSH requires the 2-norm of the items to be bounded by 1, which is achieved by normalizing the items with the maximum 2-norm $U = \max_{x \in S} \|x\|$. Assuming $q^\top x = S$ for item vector $x$, we
We plot the distribution of the 2-norm of a real dataset in Figure 1(b). The distribution has a long tail when many items have the same 2-norm. Instead of using Algorithm 2, respectively. To solve the excessive normalization problem of simple-lsh, so as to keep the inner products of the queries large. In Figure 1(d), we plot 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 \( S \), dataset size \( n \), number of sub-datasets \( m \)
2: **Output**: A hash index \( I_j \) for each of the \( m \) sub-datasets
3: Rank the items in \( S \) according to their 2-norms;
4: Partition \( S \) into \( m \) sub-datasets \( \{S_1, S_2, \ldots, S_m\} \) such that \( S_j \) holds items whose 2-norms ranked in the range \( \left( \frac{j-1}{m}, \frac{j}{m} \right) \);
5: **for** every sub-dataset \( S_j \) **do**
6: Use \( U_j = \max_{x \in S_j} \|x\| \) to normalize \( S_j \);
7: Apply simple-lsh to build index \( I_j \) for \( S_j \);
8: **end for**

have \( q^T x = S/U \) after normalization. If \( U \) is significantly larger than \( \|x\| \), the inner product will be scaled to a small value, and small inner product leads to high query complexity.

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 \( \rho \) and poor theoretical performance.

The long tail in 2-norm distribution 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 \( 4 \times 10^9 \) 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 \( U_j = \max_{x \in S_j} \|x\| \) in each sub-dataset for normalization, so as to keep the inner products of the queries large. In Figure 1(d), we plot the
Algorithm 2 Norm-Ranging LSH: Query Processing

1: **Input:** Hash indexes \{I_1, I_2, ..., I_m\} for the sub-datasets, query q
2: **Output:** A c-approximate MIPS \(x^*\) to q
3: **for** every hash index \(I_j\) **do**
4: Conduct MIPS with q to get \(x_j^*\);
5: **end for**
6: Select the item in \([x_1^*, x_2^*, ..., x_m^*]\) that has the maximum inner product with q as the answer \(x^*\);

maximum inner product of the queries after the normalization process of RANGE-LSH. Comparing with Figure 1(c), the inner products are significantly larger. As a result, given Algorithm 2, the inner products are significantly larger. As a result, given the dataset is partitioned into \(m = n^α\) sub-datasets and there are at most \(n^β\) sub-datasets with \(U_j = U\), where \(0 < \alpha < \min\{\alpha, \frac{\beta - \rho^*}{1 - \rho}\}\), \(0 < \beta < \alpha\rho\), \(\rho^* = \max_{\rho_j < \rho} \rho_j\), \(\rho_j = G(c, S_0/U_j)\) and \(\rho = G(c, S_0/U)\).

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 \(x^*\) 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 \(x^*\) 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 query time complexity of RANGE-LSH. For each sub-dataset \(S_j\), it contains \(n^{1-\alpha}\) items and the query time complexity upper bound of \(c\)-approximate MIPS is \(O(n^{(1-\alpha)\rho_1} \log n^{1-\alpha})\) with \(\rho_j = G(c, S_0/U_j)\). As there are \(m = n^α\) sub-datasets, the time complexity of selecting the optimal one from the answers of all sub-datasets is \(O(n^α)\). Considering \(\rho_j\) is an increasing function of \(U_j\) and there are at most \(n^β\) sub-datasets with \(U_j = U\), the query time complexity of RANGE-LSH can be expressed as:

\[
f(n) = n^α + \sum_{j=1}^{n^α} n^{(1-\alpha)\rho_j} \log n^{1-\alpha} < n^α + \sum_{j=1}^{n^α} n^{(1-\alpha)\rho_j} \log n
\]

\[
= n^α + \sum_{j=1}^{n^α-n^β} n^{(1-\alpha)\rho_j} \log n + n^β n^{(1-\alpha)\rho} \log n
\]

\[
< n^α + n^α n^{(1-\alpha)\rho^*} \log n + n^β n^{(1-\alpha)\rho} \log n
\]

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 \(O(n^ρ \log n)\) complexity of SIMPLE-LSH,

\[
\frac{f(n)}{n^ρ \log n} < \frac{n^α + (n^α n^{(1-α)\rho^*} + n^β n^{(1-α)\rho}) \log n}{n^ρ \log n}
\]

\[
= n^{α-ρ}/ \log n + n^{α+(1-α)\rho^*} - \rho + n^{β-αρ}\rho
\]

(11) goes to 0 with sufficiently large \(n\) when \(α ≤ ρ, α + (1 - α)ρ < ρ\) and \(β - αρ < 0\), which is satisfied by \(α < \min\{ρ, \frac{\beta - ρ^*}{1 - ρ}\}\) and \(β < αρ\).
Although the theoretical guarantee of LSH only holds when using multiple hash tables, in practice, with sign random projection rather than SIMPLE RANGE, the lower theoretical query time complexity of LSH also translates into much better bucket balance in practice. On the ImageNet dataset, 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 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 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

Although the theoretical guarantee of LSH only holds when using multiple hash tables, in practice, LSH is usually used in a single-table multi-probe fashion for candidate generation for similarity search [Andoni et al., 2015; Lv et al., 2007]. The buckets/items are ranked according to the number of identical hashes they have with the query (e.g., Hamming ranking) and the top-ranked buckets are queried first. Multi-probing is challenging for LSH as different sub-datasets use different normalization constants and buckets from different sub-datasets cannot be ranked simply according to their number of identical hashes. To support multi-probe in LSH, we formulate a similarity metric for bucket ranking that is efficient to manipulate.

Combining the index building process of LSH and the collision probability of sign random projection in (4), the probability that an item \( x \in S_j \) and the query collide on one bit is \( p = 1 - \frac{1}{\pi} \cos^{-1} \left( \frac{q \cdot x}{U_j} \right) \), where \( U_j \) is the maximum 2-norm in sub-dataset \( S_j \). Denote the code length as \( L \) and the number of identical hashes bucket \( b \) has with the query as \( l \), we can obtain an estimate of the collision probability \( p \) as \( \hat{p} = l/L \). Plug \( \hat{p} \) into the collision probability, we get an estimate \( \hat{s} \) of the inner product between \( q \) and the items in bucket \( b \) (from sub-dataset \( S_j \)) as:

\[
\hat{s} = U_j \cos \left[ \pi \left( 1 - \frac{l}{L} \right) \right].
\]  

(12)

Therefore, we can compute \( \hat{s} \) for the buckets/items and use it for ranking. When \( l > L/2 \), \( \cos \left[ \pi \left( 1 - \frac{l}{L} \right) \right] > 0 \), thus larger \( U_j \) indicates higher inner product while the opposite is true when \( l < L/2 \). Since the code length is limited and \( l/L \) can diverge from the actual collision probability \( p \), it is possible that a bucket has large \( U_j \) and large inner product with \( q \), but it happens that \( l < L/2 \). In this case, it will be probed late in the query process, which harms query performance. To alleviate this problem, we adjust the similarity indicator to \( \hat{s} = U_j \cos \left[ \pi \left( 1 - \epsilon \right) \left( 1 - \frac{l}{L} \right) \right] \), where \( 0 < \epsilon < 1 \) is a small number. For the adjusted similarity indicator, \( \cos \left[ \pi \left( 1 - \epsilon \right) \left( 1 - \frac{l}{L} \right) \right] < 0 \) only when \( l < L \left[ \frac{1}{2} - \frac{\epsilon}{2(1-\epsilon)} \right] \), which leaves some room to accommodate the randomness in hashing.

Note that the similarity metric in (12) can be manipulated efficiently with low complexity. We can calculate the values of \( \hat{s} \) for all possible combinations of \( l \) 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. When a query comes, query processing can be conducted by traversing the sorted structure in ascending order. For a pair \((U_j, l)\), \( U_j \) determines the sub-dataset while \( l \) is used to choose the buckets to probe in that sub-dataset with standard hash lookup. We also provide an efficient method to rank the items when code length is large and there are many empty buckets in the supplementary material.

\[ l \text{ can take } L + 1 \text{ values, } U_j \text{ can take } m \text{ values, so the size of the sorted structure is } mL + m. \]
11 bits are generated by hashing. We partitioned the dataset into 32, 64 and 128 sub-datasets under a 16-bit code of range. We plot the probed item-recall curves in Figure 2. The results show that SIMPLE-LSH can be found in the supplementary material.

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. Following existing researches, we mainly compare the performance of the algorithms for single-table based multi-probing. While a comparison of the multi-table single probe performance between RANGE-LSH and SIMPLE-LSH can be found in the supplementary material.

We plot the probed item-recall curves in Figure 2. The results show that RANGE-LSH probes significantly less items compared with SIMPLE-LSH and L2-ALSH at the same recall. Due to space limitation, we only report the performance of top 10 MIPS, the performance under more configurations can be found in the supplementary material.

Recall that Algorithm 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...
We also experimented the influence of the number of sub-datasets on performance in Figure 3(b). As shown in the figure, the performance improves with the number of sub-datasets when the number of sub-datasets varies from 32 to 256. RH32 means RANGE-LSH with 32 sub-datasets.

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 partitions 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 (smaller) $\rho$ 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 $S_j$, if we have the range of its 2-norms as $u_{j-1} < \|x\| \leq u_j$, $u_j < S_0$ and $u_{j-1} > 0$, we can obtain the $\rho_j$ of $S_j$ as:

$$
\rho_j = \frac{\log F_r(\sqrt{1 + m/4 - 2U_jS_0 + (U_ju_j)^{2^m+j}})}{\log F_r(\sqrt{1 + m/4 - 2cU_jS_0 + (U_ju_{j-1})^{2^m+j}})}.
$$

As $u_j < S_0$ and $u_{j-1} > 0$, the collision probability in the numerator increases while the collision probability in the denominator decreases if we compare (13) with (7). Therefore, we have $\rho_j < \rho$. Moreover, partitioning the original dataset into sub-datasets allows us to use different normalization factor $U_j$ (in addition to $m$ and $r$) for each sub-dataset and we only need to satisfy $U_j < 1/u_j$ rather than $U < 1/\max_{x \in \mathbb{S}} \|x\|$, 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. We show empirically that dataset partitioning improves the performance of L2-ALSH in the supplementary material.

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 under mild conditions. 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 L2-ALSH. The superior performance of RANGE-LSH can benefit many applications that involve MIPS.
Acknowledgments

We thank the reviewers for their valuable comments. This work was supported in part by Grant CUHK 14222816 from the Hong Kong RGC.

References


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.
