AdaLSH: Adaptive LSH for Solving $c$-Approximate Maximum Inner Product Search Problem

Kejing LU$^{(a)}$, Nonmember and Mineichi KUDO$^{(b)}$, Fellow

SUMMARY Maximum inner product search (MIPS) problem has gained much attention in a wide range of applications. In order to overcome the curse of dimensionality in high-dimensional spaces, most of existing methods first transform the MIPS problem into another approximate nearest neighbor search (ANNS) problem and then solve it by Locality Sensitive Hashing (LSH). However, due to the error incurred by the transmission and incomprehensive search strategies, these methods suffer from low precision and have loose probability guarantees. In this paper, we propose a novel search method named Adaptive-LSH (AdaLSH) to solve MIPS problem more efficiently and more precisely. AdaLSH examines objects in the descending order of both norms and (the probably correctly estimated) cosine angles with a query object in support of LSH with extendable windows. Such extendable windows bring not only efficiency in searching but also the probability guarantee of finding exact or approximate MIP objects. AdaLSH gives a better probability guarantee of success than those in conventional algorithms, bringing less running times on various datasets compared with them. In addition, AdaLSH can even support exact MIPS with probability guarantee.

key words: image retrieval, locality sensitive hashing, maximum inner product search, high-dimensional spaces

1. Introduction and Related Work

The maximum inner product search (MIPS) problem has been viewed as a fundamental problem in many applications such as recommender system [1], multi-label prediction [2], reasoning about extracted facts in open relation extraction [3], deep learning [4] and structural SVM [5]. Given a large dataset $D$ in $\mathbb{R}^d$ with $L_2$ norm and a query $q$, the objective of MIPS is to find the MIP object $o$ in $D$ such that the inner product $\langle o, q \rangle$ is maximum. We call such problem exact MIPS problem. Although the objective is straightforward, exact MIPS problem is hard to be solved efficiently in high-dimensional spaces due to the curse of dimensionality. In order to overcome this difficulty, researchers turned to focus on the approximate MIPS problem, for which approximate MIP objects are also acceptable, and proposed various approximate MIPS methods. For some of these approximate methods, the approximation ratio $c$ is introduced to control the difference between the true MIP object and the approximate MIP object. To be specific, an approximate MIP object is called the $c$-MIP object (of $q$) if its inner product with $q$ is not less than $c$ times the true maximum inner product, and we call those approximate MIPS methods which could return $c$-MIP objects as $c$-approximate MIPS methods.

In recent years, many exact and $c$-approximate MIPS methods have been proposed. Although they adopt different techniques, they can be roughly classified into following two classes:

(1) Methods that exploit suitable index structures to realize an efficient search, such as ball and tree methods [9], [10]. Although these methods work well in low-dimensional spaces, they are defeated by the brute-force search in high-dimensional spaces due to the curse of dimensionality, which limits their applications in practice.

(2) Methods that transform the MIPS problem into an approximate nearest neighbor search (ANNS) problem or a maximum cosine similarity (MCS) problem, such as L2-ALSH [11], Sign-ALSH [6], Simple-ALSH [7], XBOX [12] and H2-ALSH [8]. Due to the utilization of Locality Sensitive Hashing (LSH) technique, these methods can return $c$-MIP objects with probability guarantees. However, the probability guarantees of these methods are obtained in the worst case and are inconsistent with the real performances, which leads to a significant gap between the theory and the application.

In addition to the methods mentioned above, some MIPS methods of other types have been proposed very recently, such as graph based methods [13], [14] and vector quantization based methods [15]. Compared with LSH based methods, they require less memory consumption [15] or enjoy higher searching efficiency [13], [14], [16], but they can neither solve $c$-approximate MIPS problems nor provide any theoretical guarantee on query results. Thus, we only focus on LSH-based methods in this paper.

In order to overcome the limitations of existing LSH-based methods, we propose a novel MIPS method named AdaptiveLSH (AdaLSH). In the indexing phase, similar to H2-ALSH, we construct a homocentric hypersphere structure of disjoint subsets $\{S_i\}_{i=1}^n$. However, instead of doing the transformation in each subset as H2-ALSH does, we only normalize each object to norm one and then construct an index structure by the random projection. To measure the cosine angle efficiently, we resort to dynamic counting technique [17]. Specifically, we estimate the cosine angle by counting how frequently the collision happens between $o$ and $q$ in hash tables when $o$ and $q$ are both normalized to norm one and are mapped to the hash tables. If the collision number of some object reaches the threshold determined be-

Manuscript received June 17, 2020.
Manuscript revised August 28, 2020.
Manuscript publicized October 13, 2020.

The authors are with the Graduate School of Information Science and Technology, Hokkaido University, Sapporo-shi, 060–0814 Japan.

a) E-mail: lkejing@ist.hokudai.ac.jp
b) E-mail: mine@ist.hokudai.ac.jp
DOI: 10.1587/transinf.2020EDP7132
forehand, it will be viewed as a candidate and its exact inner product with \( q \) will be computed. Finally, the most promising MIP candidate will be returned as the MIP object. It is notable that, unlike many existing LSH based methods depending on the transformation to a one-dimensional higher space, such as simple-LSH [7] and OASIS [16], AdaLSH works in the original space and indexes only the direction of data points by LSH, which not only avoids the transmission error but also makes our method control the widths of hash buckets by the norms of real vectors.

A brief comparison between AdaLSH and other LSH based methods is shown in Table 1. Specifically, AdaLSH has following two features:

1. The probability guarantee of AdaLSH makes practical sense. As discussed above, although almost all methods in the second class have probability guarantees, these guarantees could not guide the MIPS in practice. Take the state-of-the-art methods of LSH type. In the preprocessing phase, H2-ALSH partitions a dataset into disjoint subsets \( S_1, S_2, \ldots, S_n \), which can be viewed as concentric rings geometrically (\( S_1 \) is the smallest, followed by \( S_2 \), and so on). Subsequently, in each subset \( S_i \), objects (in \( S_i \)) are mapped to a one-dimension-higher \((d + 1)\) space, such that an object with a larger value of inner product with \( q \) in the original space will have a smaller distance to \( q \) in the mapped space. In the query phase, from \( S_n \) to \( S_1 \), H2-ALSH repeats the ANN search.

Although H2-ALSH performs well in practice, it has following two limitations. (1) Its probability guarantee is based on not always correct assumptions on data distributions. In other words, for some specific data distributions, the real performances of H2-ALSH may be inconsistent with the predicted performances. (2) As mentioned in Sect. 1, the success probability guaranteed by H2-ALSH is low (only \( 1/2 - 1/e \)). Therefore, in order to fulfill the user-specified precision, a number of repetitions are needed to increase the success probability, which incurs much additional cost.

### 2. Preliminaries

#### 2.1 Notations and Problem Setting

We consider a set \( D \subseteq \mathbb{R}^d \) of objects \( o \)'s and a query \( q \in \mathbb{R}^d \). Our goal is to find \( o^* \in D \) whose inner product with \( q \) is maximal. Let the angle between \( o \) and \( q \) be \( \theta \) (\( 0 \leq \theta \leq \pi \)). We use the tilde (~) for normalization, i.e., \( \bar{o} = o/\|o\| \), and the hat (\( \wedge \)) for “optimal candidates”, such as \( \hat{o} \) for \( o^* \).

The goal of the \( c \)-approximate maximum inner product \( (c\text{-AMIP}) \) problem is to find object \( \hat{o} \) satisfying \( \langle \hat{o}, q \rangle \geq c \langle o^*, q \rangle \) for given \( c \) (\( 0 < c \leq 1 \)) (\( c = 1 \) for Exact MIPS). In this paper, we mainly consider how to find a single \( c \)-AMIP object, but it is easy to extend the algorithm so as to find \( k \) \( c \)-AMIP objects (We only need to correspond the first MIP object to the \( k \)th one). In addition, if \( \langle o^*, q \rangle \) is negative, we only need a little modification in our algorithm, such as replacing \( c \) by \( 1/c \).

#### 2.2 Brief Review of H2-ALSH

The recently proposed H2-ALSH [8] outperforms the other state-of-the-art methods of LSH type. In the preprocessing phase, H2-ALSH partitions a dataset into disjoint subsets \( S_1, S_2, \ldots, S_n \), which can be viewed as concentric rings geometrically \((S_1) \) is the smallest, followed by \( S_2 \), and so on). Subsequently, in each subset \( S_i \), objects (in \( S_i \)) are mapped to a one-dimension-higher \((d + 1)\) space, such that an object with a larger value of inner product with \( q \) in the original space will have a smaller distance to \( q \) in the mapped space. In the query phase, from \( S_n \) to \( S_1 \), H2-ALSH repeats the ANN search.

### Table 1 Comparison among state-of-the-art LSH-based methods

| Methods   | Dimension elevation | Dataset partition | Support exact search? |
|-----------|---------------------|-------------------|-----------------------|
| Sign-LSH [6] | Yes                | No                | No                    |
| Simple-LSH [7] | Yes                | No                | No                   |
| H2-ALSH [8] | Yes                | Yes               | No                    |
| AdaLSH(proposed) | No                | Yes               | Yes                   |

### Table 2 Some notations

| Notations | Explanations |
|-----------|--------------|
| \( D \)   | The original dataset |
| \( q \)   | The query |
| \( d \)   | The dimension of dataset \( D \) |
| \( \delta \) | The error rate specified by users \((0 < \delta < 1)\) |
| \( c \) | The approximation ratio specified by users \((c < 1)\) |
| \( o^* \) | The true MIP object of \( q \) |
| \( \hat{o} \) | The MIP object among candidates |
| \( n \) | The number of rings |
| \( S_i \) | The \( i \)th ring |
| \( m \) | The number of hash functions |
| \( a_i \) | The \( i \)th projection vector |
| \( h_{ij} \) | The \( i \)th hash function generated randomly |
| \([-w_i, w_i]\) | The search window in \( S_i \) |
| \#Col(o_1, o_2) | The collision number of objects \( o_1 \) and \( o_2 \)
| \( t' \) | The threshold for \( w_i \) at the \( i \)th round |

\( ^1 \)In Theorem 4.2 [8], they limit the cosine angle in \([0, \pi]\) to be \( \leq 1/4 \). In contrast, AdaLSH could directly support any error rate and approximation ratio specified by users.

In Table 2, we list some important notations in this paper for reference. Some of them will be introduced later.
AdaLSH divides the dataset $D$ into $n$ distinct subsets $S_1, S_2, \ldots, S_n$ that are separated by concentric circles centered at zero with radii $r_0, r_1, \ldots, r_n$ in the descending order (max$_{o \in D} \|o\| = r_0 > r_1 > \cdots > r_n > 0$), that is, $S_i = \{o|r_i < \|o\| \leq r_{i-1}\}$. Let $\ell_i = \min_{o \in S_i} \|o\|$ and $u_i = \max_{o \in S_i} \|o\|$, respectively. Then, AdaLSH normalizes objects falling inside the same ring and project them onto the generated vectors by hash functions $h_o(\tilde{o}) = \langle a, \tilde{o} \rangle$, where $\tilde{o}$ is the normalized object in the ring. We apply such operation to all rings and thus obtain $n$ groups of hash functions, each of which corresponds to some ring (See Fig. 1 for an example).

### 3.2 The Query Phase

#### 3.2.1 The Overview

The searching process is carried out in multiple rounds. In each round, we examine $S_1, S_2, \ldots, S_n$ from outside to inside. That is, we limit first the objects by their norms. In each subset $S_i$ whose member $o$ has a norm in $[\ell_i, u_i]$, we furthermore limit the objects by their cosine angles $\cos \theta$, which are estimated probably correctly in support of the query-centric LSH technique. The estimation of a cosine angle is made through the collision testing in multiple hash functions (projection lines). As will be explained later in detail, with a random vector $a \sim N(0, I_d)$, $h_o(\tilde{o}) - h_o(\tilde{q}) \sim N(0, 2 - 2 \cos \theta)$ (Fig. 1). That is, $\tilde{o}$ and $\tilde{q}$ have a high collision probability in a window with fixed size $2w$ if they are close, i.e., $\cos \theta = 1$, while they have a low collision probability if they are distant, i.e., $\cos \theta \approx -1$. Based on this property, we estimate the value of $\cos \theta$ by the collision number in $m$ projection lines associated to randomly generated $a_1, \cdots, a_m$. We set up an anchor window $[-w, w]$ in common to $m$ projection lines for the collision testing of $o \in S_i$ and $q$. For checking the objects in the descending order of cosine angles, the value of $w_i$ is increased gradually by $\Delta w$. When $w_i$ becomes larger than a round threshold $\tau_i$ in the $r$th round, we terminate searching objects in $S_i$ and move to check the next subset $S_{i+1}$ (Fig. 2). When $S_n$ has been processed, the next round starts. The details will be explained in Sect. 3.2.4.

#### 3.2.2 Basics

Let us start by summarizing the facts that will be used. The inner product is defined as

$$\langle o, q \rangle = \|o\| \|q\| \cos \theta.$$ 

The solution of the MIP (Maximum Inner Product) problem, $\text{MIP}(D, q)$, is given as

$$o^* = \arg \max_{o \in D} \|o\| \|q\| \cos \theta = \arg \max_{o \in D} I(o) = \|o\| \cos \theta.$$

Note that the inner product $I(o)$ is evaluated as the product of the norm of $o$ and the cosine angle with the query.

Let us consider a random hash function $h_o(o) = \langle a, o \rangle$, where $a$ is generated randomly according to the $d$-dimensional standard Gaussian distribution $N(0, I_d)$. Then, it is well known [18] that for two points $o$ and $q$,

$$h_o(o - q) = h_o(o) - h_o(q) \sim N(0, \|o - q\|^2).$$

Suppose that $o$ and $q$ have been normalized to $\tilde{o}$ and $\tilde{q}$, respectively. In query-aware search [19], we view $h_o(\tilde{q})$ as the origin, i.e., $h_o(\tilde{q}) = 0$. Thus, for $\tilde{o}$ with distance $s = \|\tilde{o} - \tilde{q}\| = \sqrt{2(1 - \cos \theta)}$, we have

$$h_o(\tilde{o}) \sim N(0, s^2).$$

Once we set up an (anchor) window with width $2w$ centered at the origin on the line of direction vector $a$, the probability that $h_o(\tilde{o})$ falls inside the anchor window is given by

$$P(h_o(\tilde{o}) \in [-w, w]) = f(w/s) = \frac{1}{\sqrt{2\pi}} \int_{-w/s}^{w/s} e^{-x^2/2} dx.$$

Let us rewrite this probability by $p(s, w)$ as

$$p(s, w) = f(w/s).$$

---

**Fig. 1** The indexing phase of AdaLSH. Any object $o$ is normalized to $\tilde{o}$ ($\|\tilde{o}\| = 1$), then projected onto a line with a random Gaussian vector $a_j$ for collision testing.

**Fig. 2** Multi-round search strategy, where $\tau_r$ denotes the threshold for window size $w_i$ at $r$th round.
According to (3), \( p(s, w) \) is decreasing in \( s \) and increasing in \( w \). For a set of hash functions \( \mathcal{H} = \{ h_{i1}, h_{i2}, \ldots, h_{in} \} \) with \( \{ a_j \}_{j=1}^m \) randomly generated, we denote the number of collisions (of \( o \)) by \( \#\text{Col}(\tilde{o}, \tilde{q}) \) (if \( \tilde{o} \) falls inside the window \([-w, w]\), we say that \( \tilde{o} \) collides with \( \tilde{q} \) in this window). In the following discussion, we will consider a collision testing that examines if a normalized object \( \tilde{o} \) collides with \( \tilde{q} \), \( m/2 \) times or more.

### 3.2.3 Fundamental Relationships

Equation (2) shows that we have to control the window size \( 2w \) appropriately depending on the distance \( s \) to assure of the validity of the collision testing. More concretely, the following question arises: what size of \( w \) is necessary to find a specific object \( o \), with distance \( s = ||\tilde{o} - \tilde{q}|| \), by the collision testing with \( m \) hash functions. From (2) and the Hoeffding’s inequality on the sum of independent bounded random variables, for \( o \) with distance \( s = ||\tilde{o} - \tilde{q}|| \) and \( w \), we have

\[
P(\#\text{Col}(\tilde{o}, \tilde{q}) \geq m/2) \geq 1 - e^{-2m(p(w/s)−1/2)^2}. \tag{4}
\]

According to (4), for given number of projections \( m \) and error rate \( \delta \) (\( 0 < \delta < 1 \)), to make the probability that \( o \) passes the collision testing above at least \( 1 - \delta \), we require

\[
p(s, w) = f(w/s) \geq 1/2 + \sqrt{\frac{1}{2m} \log \frac{1}{\delta}}. \tag{5}
\]

Let \( p_0 = 1/2 + \sqrt{\frac{1}{2m} \log \frac{1}{\delta}} \) as a required rate of collision in a single projection. Hereafter, we assume that \( m \) is taken so as to \( p_0 < 1 \) for given \( \delta \). Then (5) means that we have to make \( w \) satisfy

\[
w/s \geq f^{-1}(p_0). \tag{6}
\]

since \( f \) is bijective and monotonically increasing in \( w/s \).

With notation \( I(o) = ||o|| \cos \theta \), distance \( s \) between \( \tilde{o} \) and \( \tilde{q} \) can be written as

\[
s = ||\tilde{o} - \tilde{q}|| = \sqrt{2(1 - \cos \theta)} = \sqrt{2(1 - I(o)/||o||)}. \tag{7}
\]

We define a function \( g \) as

\[
g\left( \frac{I(o)}{||o||}; p_0 \right) = \sqrt{2\left( 1 - \frac{I(o)}{||o||} \right)} f^{-1}(p_0). \tag{8}
\]

Then, condition (6) becomes

\[
w \geq \sqrt{s f^{-1}(p_0)} = g(I(o)/||o||; p_0). \tag{9}
\]

That is, \( g(I(o)/||o||; p_0) \) shows a necessary size of the window to find \( o \) with the cosine angle \( \cos \theta = I(o)/||o|| \) to \( q \) with probability at least \( 1 - \delta \). Note that \( g(I/r; p) \) is bijective (in the range \( 0 \leq \theta \leq \pi \)) in \( I/r \) and, decreasing in \( I \) and increasing in \( r \). Note also that \( g(I/r; p) \) is increasing in the parameter \( p \). Hence, we have

\[
g(I(o)/||o||; p_0) \leq g(I(o)/u_i; p_0), \quad o \in S_i. \tag{10}
\]

Our goal is to determine an appropriate value of \( w \) for finding \( o^* \). To achieve this, from (9), it suffices to determine the value of \( w \) such that

\[
w \geq g(I(o^*)/||o^*||; p_0). \tag{11}
\]

However, we do not know in advance \( I(o^*) \) nor \( ||o^*|| \), hence we cannot directly determine \( w \) by (11). We will describe a strategy to deal with this difficulty in the following section.

### 3.2.4 Searching Process

First, let us consider a simpler problem: if we know the subset \( S_r \) in which \( o^* \) lies, then how do we determine the value of \( w_r \) in \( S_r \)? This problem can be solved efficiently by dynamic counting technique [17] as follows. At the beginning of the search, we set \( w_r \) to 0. As the search proceeds, we increase the value of \( w_r \) step by step with \( \Delta w_r \). If some object \( o \) passes the collision testing (\( \#\text{Col}(\tilde{o}, \tilde{q}) \geq m/2 \)), we view \( o \) as a candidate for \( o^* \) and compute \( I(o) \) (Note that \( \#\text{Col}(\tilde{o}, \tilde{q}) \) is non-decreasing in \( w_r \)). During this process, we update \( \tilde{o} \), the current most promising candidate of \( o^* \), and the corresponding inner product \( I = I(\tilde{o}) \) when an object \( o \) with \( I(o) > I \) is found. Since \( I(o^*) \geq I \) and \( ||o^*|| \leq u_i \), we have

\[
g(I/u_i; p_0) \geq g(I(o^*)/||o^*||; p_0). \tag{12}
\]

Thus, in order to make (11) hold, it suffices to set \( w_r \) to \( g(I/u_i; p_0) \). The searching step is illustrated in Fig. 3.

Since the value of \( i^* \) is unknown in practice, we need to check every \( i \) in this way. That is, we set \( w_i \) (\( 1 \leq i \leq n \))
to $T_i = g(\hat{I}/u_i; p_0)$, resulting in $w_i$ to $T_r = g(\hat{I}/u_r; p_0)$. If $o \in S_i$ satisfies $I(o) > \hat{I}$, then the normalized object $\hat{o}$ passes the collision testing with window size $2T_i$ with probability at least $1 - \delta$, and as a special case, $I(o^*)$ is found in $S_r$. However, it is costly to use $T_i$ directly, because there are a lot of objects passing the collision testing with $T_i$. In order to make the searching more efficient, we adopt a multi-round search strategy (Fig. 2). That is, in the $r$th round, we stop the examination of $S_i$ when $w_i$ reaches $\hat{t}_i(r \leq T_i)$, and move to $S_{i+1}$. Then the problem is how to set the values of $\hat{t}_i$. We consider an increasing sequence $\tau(1) < \tau(2) < \cdots < \tau(R) = 1$, where $\tau(r) = r/R$. Then, on the basis of the current $\hat{I}$, which is appropriately initialized, we determine an upper bound $\hat{t}_i$ of $w_i$ at round $r$ as follows. First, we estimate an expected inner product $I_b(r)$ in $S_i$ at round $r$ so as to

$$I_b(r)/u_i = g^{-1}(\hat{I}/u_i; \tau(r); 1/2).$$  (13)

Here, for a fixed value of $x$, since $g^{-1}(g(x; \tau), 1/2)$ is decreasing in $\tau$, we have $1 > I_b(1) > \cdots > I_b(R) = \infty$. Thus, we determine the value of $\hat{t}_i$ ($1 \leq i \leq n$) by

$$\hat{t}_i = g(I_b(r)/u_i; 1/2).$$  (14)

In this way, we share the same criterion, that is, $I_b(r)$ in all subsets $S_1, \ldots, S_n$. This $\hat{t}_i$ guarantees that an object $o \in S_i$ with $I(o) \geq I_b(r)$ has the expected collision number at least $m/2$ if $\tau(r) > 1/2$. This can be confirmed from $\hat{t}_i = g(I_b(r)/u_i; 1/2) \geq g(\hat{I}/u_i; 1/2) \geq g(I_b(\cdot)/\|o\|; 1/2)$ for $\tau(r) > 1/2$ (The first inequality is derived from (13)). This multi-round searching is expected to accelerate the termination in an earlier round. Note also that, since $\tau(r)$ to 1, deriving $I_b(r) = -\infty$ and $\hat{t}_i = +\infty$ ($1 \leq i \leq n$), the termination condition $w_i \geq g(\hat{I}/u_i; p_0) = T_i$ is necessarily satisfied before reaching the $R$th round (Fig. 3).

Now we are ready to show the complete searching process (Fig. 2). We divide the searching process into $K$ rounds and introduce a round bound $\hat{t}_i$ for $w_i$ ($i = 1, \ldots, n$) at the $r$th round by (13) and (14). At the $r$th round ($r = 1, 2, \cdots, R$ in $S_i$ ($i = 1, \ldots, n$), we check every object passes the collision testing in $[w_i, w_i]$ (a candidate), and then increase gradually the value of $w_i$ with $\Delta w_i$ until $w_i$ reaches $\hat{t}_i$'. When $w_i$ exceeds $\hat{t}_i'$, we move to $S_{i+1}$. After the examination of $S_n$ in this round, we return to $S_1$ and restart the search at the $(r+1)$th round.

In the middle of the searching process, we can terminate the examination of $S_i$ if one of following three termination conditions is satisfied.

**TC1.** We can skip the examination of $S_{i+1}, S_{i+2}, \ldots, S_n$, if $u_i < \hat{I}$. This is because any $o \in \cup_{j \geq i}S_j$ satisfies $\|o\| \leq u_i$, thus, $I(o) = \|o\| \cos \theta \leq u_i$.

**TC2.** If $\hat{t}_i' \geq g(I/\|u_i\|; p_0) = T_i$, we can terminate the search in $S_i$ since (11) is satisfied (see Fig. 3). This is a special case of the condition TC3.

**TC3.** For any $c$ ($0 < c \leq 1$), we can terminate the search in $S_i$ if $w_i \geq g(I/(c\|u_i\|); p_0)$. The reason will be explained in Theorem 1 later.

---

**Algorithm 1: Query phase of AdaLSH**

**Input:** $q$: query;
- $c$: approximation ratio ($0 < c \leq 1$);
- $\delta$: error rate ($0 < \delta < 1$);
- $m$: the number of generated hash functions;
- $R$: maximum round times;
- $\{I(S_i, u_i, \tau_i)^w\}_i$: divided subsets of objects;
- $\mathcal{H} = [h_o]_0^\infty$: hash functions;

**Output:** $\delta$: a $c$-approximate MIP object for $q$.

1. $S \leftarrow \{1, 2, \ldots, n\}$;
2. $I \leftarrow (o^*, \hat{q})$, where $o^* = \arg \max \{\|o\|\}$;
3. $w_i \leftarrow 0$ ($i = 1, 2, \ldots, n$); $r \leftarrow 1$;
4. $p_0 \leftarrow 1/2 + \sqrt{(1/2m)\log(1/\delta)}$;

5. While $S \neq \emptyset$ and $r \leq R$ do
6. $I_b \leftarrow u_i g^{-1}(g(I/u_i; \tau(r); 1/2)); \%$ see (13)
7. for $i \in S$ in ascending order do
8. if $u_i < \hat{I}$ then
9. $S \leftarrow S\{i\};$
10. break
11. $u_i \leftarrow g(I_b(u_i; 1/2)); \%$ see (14)
12. while $w_i \leq u_i$ do
13. $w_i \leftarrow w_i + \Delta w_i; \%$ extend windows
14. for $o \in S$ do
15. if $\#Col(\hat{I}, \hat{q}) \geq m/2$ then
16. $I \leftarrow (\hat{q}, \hat{q})$;
17. if $I > \hat{I}$ then
18. update $\hat{o}$ and $\hat{I}$ with $o$ and $I$;
19. if $\|\hat{t}_i - g(I/(c\|u_i\|); p_0)\|$ holds
20. $S \leftarrow S\{i\};$
21. break
22. $r \leftarrow r + 1$
23. return $\delta$

---

### 3.2.5 Performance Analysis

We can guarantee the performance of AdaLSH as follows:

**Theorem 1.** For given error rate $\delta$ ($0 < \delta < 1$), AdaLSH solves $c$-AMIP problem AMIP$(D, q, c)$ with probability at least $1 - \delta$, where $0 < c \leq 1$.

**Proof.** First note that the main loop steps (7–21) checks every $i$ in order of $i = 1, 2, \ldots, n$. For each subset $S_i$, the examination of $S_i$ can be terminated when either TC1 or condition TC3 (TC2 is a special case) is satisfied (Note that $\hat{t}_i'$ is $+\infty$ in the $R$th round, which ensures for each $S_i$ that either of these two conditions is necessarily satisfied). On the other hand, only when examinations of all subsets have been terminated, we can terminate the query phase. Thus, it suffices to consider $S_F$ only. First note that condition TC1 cannot be applied to $S_i$ since $u_i > \hat{I}(o^*) > \hat{I}$. When condition TC3 was applied to $S_F$, either $w_F \geq g(I(o^*)/\|o^*\|; p_0)$ or $w_F \leq g(I(o^*)/\|o^*\|; p_0)$ holds. According to (11), the former case implies that we can find $o^*$ with probability at least $1 - \delta$. In the latter case, we have $g(I/(c\|u_i\|); p_0) \leq w_i \leq g(I(o^*)/\|o^*\|; p_0) \leq g(I(o^*)/u_i; p_0)$. This directly means...
that \( \hat{I} = I(o) \geq cI(o^*) \), implying that some object \( o \) has already been found such that \( I(o) \geq cI(o^*) \). Thus, no matter in which case, we can find a \( c \)-AMIP object with probability at least \( 1 - \delta \).

Although we have not considered the case \( \hat{I} < 0 \), in that case, it suffices to replace the termination condition in Step \#19 with \( w_i \geq g(\hat{I}/\ell_i; p_0) \). Under this modification, Theorem 1 still holds. For finding \( k \) \( (k > 1) c \)-AMIP objects, we need to maintain \( k \) MIP objects among candidates and replace \( \hat{I} \) in Algorithm 1 with \( \hat{I}_k = I(\delta_k) \), where \( \delta_k \) is the \( k \)th MIP object among candidates. By applying our discussion on \( o^* \) to \( o^*_i \), where \( o^*_i \) is the true \( i \)th MIP object, it is easy to extend Theorem 1 for \( c-k \)-AMIP problem. In addition, we have following result on recalls:

**Corollary 1.** For given error rate \( \delta \) \((0 < \delta < 1) \), AdaLSH can find \( k \) MIP objects with expectation of recall rate at least \( 1 - \delta \) when \( c = 1 \).

### 4. Experimental Evaluation

AdaLSH was coded in C++. All experiments were carried on a PC with Intel(R) 3.40GHz i7-4770 eight-Cores processor with 8 GB RAM in Ubuntu 16.04.

#### 4.1 Experimental Setup

##### 4.1.1 Datasets and Queries

We chose ten real datasets for experiments (Table 3). Since the objects in Deep/Ukbench have almost the same norm, we call these two datasets homocentric datasets. For each dataset, we chose 200 queries randomly from its corresponding query set and generated the ground truth set in by linear scan. All results are average results over 200 queries.

##### 4.1.2 Performance Metrics

- **Running time.** The running time is used to evaluate the speed of each method.
- **Recall rate.** For high-dimensional datasets and large approximation ratios (close to 1), overall ratios of compared methods are very close. In that case, in order to distinguish them better, we use the recall rate that is equal to the ratio of the number of returned true MIP objects to \( k \).

##### 4.1.3 The Performance of AdaLSH under Different Parameters

For several combinations of \( k, c \) and \( \delta \), we show the speeds of AdaLSH in Fig. 4. Here, we only present the results on Nusw since similar trends were observed on other datasets. We have two observations. (1) As \( k \) increases, AdaLSH needs more query time. This is a natural expense for finding more MIP objects. (2) The smaller \( c \) is and the larger \( \delta \) is, the faster AdaLSH is. Especially, the reduction of \( c \) is more effective than the increase of \( \delta \). On the other hand, the exact MIPS \((c = 1.0)\) is also carried out efficiently.

#### 4.2 The Comparison Study

In this subsection, we will compare AdaLSH with other state-of-the-art MIPS methods. The codes of benchmark methods were downloaded from the link \(^1\). In Sect. 4.2.1, we compare AdaLSH with L2-ALSH \([11]\), Simple-ALSH \([7]\), Sign-ALSH \([6]\) and Xbox \([12]\). These four benchmark methods fall into the second category, as mentioned before, but they could not support arbitrary approximation ratio \( c \).

In Sect. 4.2.2, we compare AdaLSH with H2-ALSH, a state-of-the-art LSH method which could support any \( c \) in \((0,1)\).

The internal parameters of benchmark methods were all set suitably, as suggested by their authors. To be more specific, we chose \( c = 0.5 \) as the standard settings of L2-ALSH, Simple-ALSH, Sign-ALSH. As for AdaLSH, \( b \) and \( m \) were set to 0.98 and 40, respectively, and the error rate \( \delta \) was fixed to 0.1 in all experiments.

##### 4.2.1 AdaLSH vs. Other LSH-Based Methods

The experimental results are shown in Table 4 and Table 5. We have following observations:

(1) On all non-homocentric datasets except for Trevi, AdaLSH runs fastest and achieves highest recall rates. This shows that, by suitably choosing the value of \( c \), AdaLSH could work more efficiently than other LSH-based methods.

![Table 3 Real datasets](image)

| Dataset   | Dimension | Size (×10\(^3\)) | Type |
|-----------|-----------|------------------|------|
| Cifar     | 512       | 50               | Image|
| Sun       | 512       | 79               | Image|
| Enron     | 1369      | 95               | Text |
| Trevi     | 4096      | 100              | Image|
| Nusw      | 500       | 269              | Image|
| Msong     | 420       | 922              | Audio|
| Ghost     | 960       | 1000             | Image|
| Ukbench   | 128       | 1000             | Image|
| Deep      | 256       | 1000             | Image|
| ImageNet  | 150       | 2340             | Image|

![Fig. 4 Speeds of AdaLSH under different parameters on Nusw. The naive searching takes around 107ms](image)

---

\(^1\)https://github.com/HuangQiang/H2_ALSH
(2) On homocentric datasets, Ukbench and Deep. SignALSH and SimpleLSH outperform AdaLSH because on these two datasets, the homocentric hypersphere structure lose effectiveness. However, as will be seen later, even on these two datasets, AdaLSH could achieve very high recalls by choosing a larger value of $c$.

### 4.2.2 AdaLSH vs. H2-ALSH

Since H2-ALSH could support any $c$ lies in $(0, 1)$, we compare AdaLSH and H2-ALSH under $c = 0.5$ and $c = 0.99$ respectively, where $c = 0.5$ is the recommended setting of H2-ALSH and $c = 0.99$ represents the high precision level specified by users (Note that H2-ALSH can not support $c = 1$). According to the results in Table 6 and Table 7, we have following observations.

(1) We can see that, AdaLSH requires much less running time to achieve the goal of 0.5-ANNS than H2-ALSH, especially on non-homocentric datasets. This is because the termination condition of AdaLSH is adjusted dynamically based on current search results, which makes AdaLSH terminate much earlier than H2-ALSH when the search goal has been achieved.

(2) Under $c = 0.99$, both H2-ALSH and AdaLSH can achieve very high recalls (over 99%) on non-homocentric datasets while on homocentric datasets, AdaLSH achieves higher recalls (99%) than H2-ALSH (91% and 96%). As for speeds, AdaLSH requires less running time than H2-ALSH on non-homocentric datasets except for ImageNet, and runs slower than H2-ALSH on homocentric datasets. This shows that, AdaLSH can reach the same precision level of H2-ALSH but incurs less running time on non-homocentric datasets for high user-specified precision level.

(3) We can see that, as $c$ increases from 0.5 to 0.99, the query accuracies and required running times of H2-ALSH do not change significantly due to incorrect probability guarantees, especially on homocentric datasets. In contrast, the performance of AdaLSH is sensitive to the change of $c$, and thus, users can choose suitable $c$ to achieve the desired trade-off between the precision and the cost. This shows AdaLSH owns better tuning ability of $c$ than H2-ALSH.

### 5. Conclusion

We have proposed a novel search method named AdaLSH which supports both exact MIPS and $c$-approximate MIPS with probability guarantee. Compared with the state-of-the-art LSH methods, AdaLSH owns a tighter probability guarantee in the general case, which fills the gap between the real performance and the theoretical guarantee. In experiments, AdaLSH works very well on non-homocentric datasets due to the efficient searching strategy. For homocentric datasets, although the performance of AdaLSH degrades, AdaLSH can still achieve high recalls by choosing large approximation ratios thanks to the theoretical guarantee.

### Acknowledgments

This work was partially supported by JSPS KAKENHI Grant Number 19H04128.

### References

[1] N. Koenigstein, P. Ram, and Y. Shavitt, “Efficient retrieval of recommendations in a matrix factorization framework,” Proc. 21st ACM
International Conference on Information and Knowledge Management, Maui, HI, USA, pp.535–544, ACM, Oct. 2012.

[2] T. Dean, M.A. Razon, M. Segal, J. Shlens, S. Vijayanarasimhan, and J. Yagnik, “Fast, accurate detection of 100,000 object classes on a single machine,” Proc. IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, pp.1814–1821, IEEE, June 2013.

[3] S. Riedel, L. Yao, B.M. Marlin, and A. McCallum, “Relation extraction with matrix factorization and universal schemas,” Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Atlanta, Georgia, USA, pp.74–84, The Association for Computational Linguistics, June 2013.

[4] P. Ram and A.G. Gray, “Maximum inner-product search using cone trees,” Proc. 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Foster City, Silicon Valley, CA, USA, pp.257–264, ACM, Oct. 2014.

[5] A. Shrivastava and P. Li, “Asymmetric lsh for maximum inner product search (mips),” The Conference of the North American Chapter of the Association of Computational Linguistics, Montreal, Quebec, Canada, pp.931–939, ACM, Aug. 2012.

[6] M. Datar, N. Immorlica, P. Indyk, and V.S. Mirrokni, “Locality-sensitive hashing scheme based on p-stable distributions,” Proc. 14th Annual Symposium on Computational Geometry - SCG ‘04, Brooklyn, New York, USA, pp.253–262, ACM, June 2004.

[7] Q. Huang, J. Feng, Y. Zhang, Q. Fang, and W. Ng, “Query-aware locality-sensitive hashing for approximate nearest neighbor search,” Proc. VLDB Endowment, Hawai‘i, USA, vol.9, no.1, pp.1–12, Sept. 2015.

Kejing Lu received his M.S. degree in software engineering from Donghua University, China, in 2017. He is currently pursuing the D.E. degree in computer science at Hokkaido University. His research interest is data mining, especially the similarity search.

Mineichi Kudo received his Dr. Eng. degree in Information Engineering from the Hokkaido University in 1988. Starting from an instructor in 1988, since 2001 he is a professor in Hokkaido University. In 2001 he received with professor Jack Sklansky the twenty-seventh annual pattern recognition society award. He was elected to a fellow of the International Association for Pattern Recognition on December 10, 2008. His current research interests include design of pattern recognition systems, image processing, data mining and computational learning theory. He is a member of the IEICE and the IEEE.