IEEE Open Journal of the
Computer Society

Received 16 September 2022; accepted 14 October 2022. Date of publication 17 October 2022;
date of current version 31 October 2022. The review of this article was arranged by Associate Editor Peng Li.

Digital Object Identifier 10.1109/OJCS.2022.3215206

Cardinality Estimation in Inner Product Space

KOHEI HIRATA, DAICHI AMAGATA (Member, IEEE), AND TAKAHIRO HARA (Senior Member, IEEE)

Osaka University, Suita 565-0871, Japan

CORRESPONDING AUTHOR: KOHEI HIRATA (e-mail: hirata.kohei@ist.osaka-u.ac.jp)

This work was supported in part by JST PRESTO under Grant JPMJPR1931, in part by JSPS Grant-in-Aid for Scientific Research (A) under Grant 18H04095, and in part by JST CREST under Grant JPMJCR21F2.

ABSTRACT This article addresses the problem of cardinality estimation in inner product spaces. Given a set of high-dimensional vectors, a query, and a threshold, this problem estimates the number of vectors such that their inner products with the query are not less than the threshold. This is an important problem for recent machine-learning applications that maintain objects, such as users and items, by using matrices. The important requirements for solutions of this problem are high efficiency and accuracy. To satisfy these requirements, we propose a sampling-based algorithm. We build trees of vectors via transformation to a Euclidean space and dimensionality reduction in a pre-processing phase. Then our algorithm samples vectors existing in the nodes that intersect with a search range on one of the trees. Our algorithm is surprisingly simple, but it is theoretically and practically fast and effective. We conduct extensive experiments on real datasets, and the results demonstrate that our algorithm shows superior performance compared with existing techniques.

INDEX TERMS Cardinality estimation, high-dimensional data, inner product search.

I. INTRODUCTION

Artificial intelligence techniques have been recently applied to real-life applications. Deep learning and matrix factorization are representative examples. These techniques maintain objects (e.g., neurons, users, and items) by using matrices, so many studies with diverse fields, such as the World Wide Web [23], machine-learning [24], [29], databases [15], [18], [27], and recommender systems [1], [2], [12], have focused on inner product spaces. Among them, the inner product search problem has been extensively studied [6], [7], [9], [13], [16], [19], [21], [22], [26], [33], [35], because it is a primitive operator for extracting outputs from neural network models and matrix products. In this article, we address an estimation-version of the search problem, namely, the problem of cardinality estimation in inner product spaces. Given a set $\mathbf{X}$ of high-dimensional vectors, a query $\mathbf{q}$, and a threshold τ, this problem *estimates* $|\mathbf{X_q}|$, where $\mathbf{X_q}$ is a set of vectors $\mathbf{x} \in \mathbf{X}$ such that $\mathbf{x} \cdot \mathbf{q} \geq \tau$, and $\mathbf{x} \cdot \mathbf{q}$ is the inner product of $\mathbf{x}$ and $\mathbf{q}$. This problem has important applications in data science, management, and mining: for example, statistical density estimation [32], making a query processing schedule for inner product join [26], and understanding market sizes in recommender systems [1].

A straightforward approach to solving this problem is to run a state-of-the-art inner product search algorithm. Although this approach returns the exact cardinality, it is computationally expensive. This is because state-of-the-art search algorithms are based on a linear scan of $\mathbf{X}$ [15], [27], incurring $O(n)$ time, where $n = |\mathbf{X}|$. Actually, many works designed approximate inner product search algorithms [13], [19], [22], [33]. It is hence possible to utilize them for faster cardinality estimation. However, they still need to access many vectors, i.e., this approach is still time-consuming. Another approach is to employ estimation algorithms for other data spaces to which inner product spaces can be transformed. For example, we can use an estimation scheme for angular distance [32] when vectors are transformed to the corresponding data space. This is, however, not a good option, because it has many hyper-parameters that are difficult to tune in practice. Moreover, our empirical studies show that this approach incurs large errors.

As can be seen above, the inner product cardinality estimation problem is challenging, and existing techniques for inner product search and cardinality estimation cannot provide fast and accurate estimation. We therefore propose a new algorithm for cardinality estimation in inner product spaces.

Informally, our idea is to transform this problem into the problem of approximate range counting in a low-dimensional Euclidean space. Based on this idea, our algorithm samples vectors only from a local space that is sufficient to estimate an accurate cardinality. Thanks to this approach, with a small number of samples, our algorithm yields an accurate cardinality, i.e., fast and accurate estimation theoretically and empirically. To summarize, our main contributions are as follows:

1) We propose a new algorithm for cardinality estimation in inner product spaces.
2) We theoretically analyze the performance of our algorithm.
3) We conduct extensive experiments on real datasets. The experimental results demonstrate that our algorithm (i) estimates cardinality more accurately than the state-of-the-art cardinality estimation algorithm and (ii) is much faster than the state-of-the-art inner product search algorithms, while keeping a small estimation error.

The rest of this article is organized as follows. Section II formally defines the problem we address in this article. Section III reviews related studies. In Section IV, we present our algorithm. We report our experimental results in Section V. Finally, in Section VI, we conclude this article.

II. PROBLEM DEFINITION

Let $\mathbf{X}$ be a set of n vectors $\mathbf{x}_1,..., \mathbf{x}_n$. Each vector $\mathbf{x} \in \mathbf{X}$ has d-dimensions, i.e., $\mathbf{x} = \langle \mathbf{x}[1], \ldots, \mathbf{x}[d] \rangle$, and we assume that d is high. Given two vectors $\mathbf{x}$ and $\mathbf{x}'$, the inner product $\mathbf{x} \cdot \mathbf{x}'$ is computed as: $\mathbf{x} \cdot \mathbf{x}' = \sum_{i=1}^{d} \mathbf{x}[i] \cdot \mathbf{x}'[i]$.

Applications, which use inner products, are usually interested in vectors that have large inner products with a given query vector $\mathbf{q}$. For example, when $\mathbf{q}$ represents a new item, some applications may search for user vectors of which the inner products with $\mathbf{q}$ (e.g., their ratings for the item) are high. To estimate the number of such vectors, we address the following problem:

Definition 1 (Cardinality estimation in inner product spaces): Given a set $\mathbf{X}$ of vectors, a query vector $\mathbf{q}$, and a threshold τ, $\mathbf{X_q}$ is defined as:

$$\mathbf{X_q} = \{\mathbf{x} \mid \mathbf{x} \in \mathbf{X}, \mathbf{x} \cdot \mathbf{q} \geq \tau\}.$$

Then, the problem of cardinality estimation in inner product spaces is to estimate $|\mathbf{X_q}|$.

Our objective is to estimate $|\mathbf{X_q}|$ for an arbitrary query $\mathbf{q}$ quickly and accurately.

III. RELATED WORK

A. CARDINALITY ESTIMATION

This is an essential operator for many applications, so many studies developed fast and accurate cardinality estimation techniques. A typical example is cardinality estimation on a collection of sets via sketches [10]. Famous sketching techniques are Count-Min sketch [5] and HyperLogLog sketch [8]. Their variants were extensively studied, e.g., in [28]. However, these cannot be utilized for multi-dimensional vectors.

In the database community, there are a lot of studies that address the cardinality estimation problem in multi-dimensional spaces. A classic solution is to simply run random sampling from $\mathbf{X}$. In practice, this simple random sampling is slow and not accurate. Another approach is based on kernel density [17]. However, as it utilizes *density*, it supports only metric spaces, whereas inner product spaces do not follow metric.[1] Machine-learning (ML) models for cardinality estimation have also been devised recently [25], [30], [31], and most of them focused on relational database tables. These are hence hard to be employed in our problem, suggesting the necessity of designing a technique that is specific to inner product spaces. Moreover, we do not consider learning models for inner product cardinality estimation. This is because existing works, e.g., [25], show that training cardinality estimation models incur a long time, e.g., more than a few *hours* (whereas our pre-processing needs less than a few *minutes*, see Section B).

The cardinality estimation algorithm proposed in [32] can be employed in our problem. Actually, inner product search can be transformed into similarity search under angular distance constraint. (In [2], it was proved that inner product search can be transformed into the cosine similarity search, and the angular distance between two points is obtained from the cosine similarity between them.) This algorithm utilizes locality-sensitive hashing (LSH) [3] and computes concatenated binary hash values for each vector. Each bin of the LSH table maintains those vectors with the same hash values. Given a query vector, this algorithm first computes the hash values of the query vector. Then, it samples vectors from the bins such that the Hamming distance between their hash values and those of the query is not larger than a threshold w.r.t. Hamming distance. This algorithm computes the angular distance between each sample vector and the query, and then estimates the cardinality from the sampling result. Unfortunately, [32] does not provide how to tune parameters w.r.t. LSH and the Hamming distance threshold, although these directly relate to the accuracy of the estimation result and are not trivial to tune.

B. INNER PRODUCT SEARCH

A straightforward way to solve our problem is to run an inner product search algorithm. The state-of-the-art exact search algorithms employ linear scan and develop early termination techniques [15], [27] and beat the other approaches, such as [21]. However, this approach is "too much" for cardinality estimation, as it incurs $O(n)$ time.

For high-dimensional vectors, exact solutions are usually computationally expensive, so approximate solutions have also been devised. A typical approach is to utilize LSH [13], [19], [22], [33]. This approach transforms the problem of

[1] In Section A, we use a Euclidean transformation, but it holds only between a vector and a query, i.e., it does not hold between any two vectors in $\mathbf{X}$.

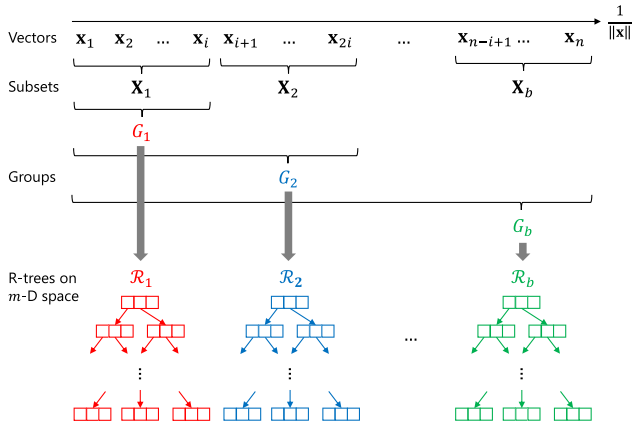

FIGURE 1. Visualization of our data structure. Assume that $\|\mathbf{x}_1\| \geq \|\mathbf{x}_2\| \geq \cdots \geq \|\mathbf{x}_n\|$. $\mathbf{X}$ is partitioned into disjoint subsets $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_b$. Each group G_i contains $\mathbf{X}_1, \ldots, \mathbf{X}_i$. For each group G_i, an R-tree $\mathcal{R}_i$ [20] is built on the m-dimensional space, i.e., the space projected via (2).

approximate inner product search into the problem of approximate nearest neighbor search. Another approach is to utilize proximity graphs [16], [35]. This approach employs a greedy algorithm. It traverses nodes of the proximity graphs that approach a given query the closest until no improvements are obtained. Different from these approaches, ScaNN [9], the state-of-the-art approximate inner product search algorithm, employs vector quantization. The idea of vector quantization is essentially similar to that of LSH: it partitions the data space into disjoint subspaces, and, for a given $\mathbf{q}$, it searches the query result only in the subspaces that correspond to $\mathbf{q}$. ScaNN quantizes the vectors while minimizing the loss incurred by the vector quantization. However, even this state-of-the-art is not efficient for cardinality estimation, because it still needs to access many vectors. Our problem hence needs a different technique, and accurately estimating the cardinality from a very small portion of $\mathbf{X}$ is required.

IV. OUR ALGORITHM

A desirable solution is to access only a subset $\mathbf{X}' \subset \mathbf{X}$ such that each $\mathbf{x} \in \mathbf{X}'$ has $\mathbf{x} \cdot \mathbf{q} \geq \tau$. This is unfortunately hard to achieve, because τ is not known in advance. We then have a question: how to *approximately* achieve this?

In a low-dimensional Euclidean space, a range search on a tree-index can prune many unnecessary points, thus is a promising approach. We hence use two transformations: from the inner product space to the Euclidean space, and from the high-dimensional space to a low-dimensional space. Note that simply employing a range search may lose efficiency, because this approach incurs $\Omega(|\mathbf{X}_\mathbf{q}|)$ time and $|\mathbf{X}_\mathbf{q}|$ can be large. To avoid this issue, we consider a sampling approach. However, how to incorporate sampling in such a range search on a tree-index to satisfy accurate estimation is not trivial. We design a non-trivial algorithm that is theoretically and practically fast and accurate.

Specifically, putting a similar observation to Johnson-Lindenstrauss lemma [14], we have the following result.

Theorem 1: There is a sampling algorithm yielding an estimation $|\mathbf{X}'_\mathbf{q}|$ such that (i) $\mathbb{E}[|\mathbf{X}'_\mathbf{q}|] = |\mathbf{X}_\mathbf{q}|$ and (ii) $(1-\epsilon)|\mathbf{X}_\mathbf{q}| \leq |\mathbf{X}'_q| \leq (1+\epsilon)|\mathbf{X}_\mathbf{q}|$ with at least probability $1-\delta$ by using $O\left(\frac{|\mathbf{V}_{\tilde{\mathbf{q}},\tilde{r}}|}{|\mathbf{X}_\mathbf{q}|\epsilon^2} \ln \frac{1}{\delta}\right)$ samples, where $|\mathbf{V}_{\tilde{\mathbf{q}},\tilde{r}}|$ is a sample pool size, with $O(n \log n)$ space, for arbitrary q and τ.

Remark 1: This novel result improves the space cost and sample complexity of a state-of-the-art algorithm [32]. Note that this algorithm needs $O\left(\frac{n}{\epsilon^2} \log\left(\frac{1}{\delta}\right)\right)$ space ($\log n < \frac{1}{\epsilon^2} \log\left(\frac{1}{\delta}\right)$) and an *additional* $O\left(\frac{1}{\alpha}\right)$ factor ($\alpha < 1$) for sample complexity to obtain the same theoretical estimation accuracy to ours.

In this section, we prove this theorem by using a novel approach that combines vector transformation, dimensionality reduction, and random sampling with tree-traversal.

A. PRE-PROCESSING ALGORITHM

To start with, we present our pre-processing algorithm that builds our data structure. This pre-processing is done only once, and our data structure is general to any $\mathbf{q}$ and τ.

1) SORT $\mathbf{X}$ BY NORM

We first compute L_2 norm $\|\mathbf{x}\|$ for each $\mathbf{x} \in \mathbf{X}$. Then, we sort $\mathbf{X}$ in descending order of this norm.

2) FROM INNER PRODUCT SPACE TO EUCLIDEAN SPACE

We use Xbox transformation [2] to achieve this transformation. It transforms each $\mathbf{x} \in \mathbf{X}$ into $\bar{\mathbf{x}}$ as follows:

$$\bar{\mathbf{x}} = \left\langle \mathbf{x}[1], \ldots, \mathbf{x}[d], \sqrt{M^2 - \|\mathbf{x}\|^2} \right\rangle,$$

where M is the largest norm in $\mathbf{X}$. Also, it transforms a given query $\mathbf{q}$ into $\bar{\mathbf{q}}$ as follows:

$$\bar{\mathbf{q}} = \langle \mathbf{q}[1], \ldots, \mathbf{q}[d], 0 \rangle . \tag{1}$$

Note that

$$\|\bar{\mathbf{x}} - \bar{\mathbf{q}}\|^2 = \|\mathbf{x} - \mathbf{q}\|^2 + M^2 - \|\mathbf{x}\|^2 = M^2 + \|\mathbf{q}\|^2 - 2\mathbf{x} \cdot \mathbf{q}.$$

and

$$\mathbf{x} \cdot \mathbf{q} \geq \tau \Leftrightarrow \|\bar{\mathbf{x}} - \bar{\mathbf{q}}\|^2 \leq M^2 + \|\mathbf{q}\|^2 - 2\tau.$$

Define r as:

$$r = \sqrt{M^2 + \|\mathbf{q}\|^2 - 2\tau},$$

and $|\mathbf{X}_\mathbf{q}|$ corresponds to the number of $\bar{\mathbf{x}}$ such that $\|\bar{\mathbf{x}} - \bar{\mathbf{q}}\| \leq r$. Now it can be seen that the problem of inner product cardinality estimation can be transformed into a range search in the Euclidean space. We use $\bar{\mathbf{X}}$ to denote the set of transformed vectors.

3) DIMENSIONALITY REDUCTION

Because the problem of range search in a high-dimensional Euclidean space cannot be solved efficiently, we reduce the dimensionality of $\bar{\mathbf{X}}$ (from $d+1$ to m). We utilize a random projection function $h(\bar{\mathbf{x}})$. Specifically,

$$h(\bar{\mathbf{x}}) = \mathbf{a} \cdot \bar{\mathbf{x}},$$

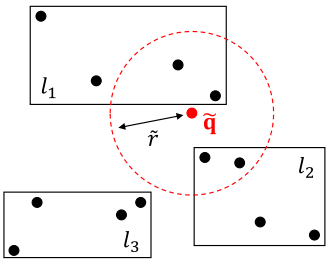

FIGURE 2. Visualization of our range search. The dotted circle shows the search range centered at a projected query $\tilde{\mathbf{q}}$, which is shown by the red point. Leaf nodes of an R-tree, l_1, l_2, and l_3, are represented by rectangles, and black points show projected vectors. The result of this range search is $\{l_1, l_2\}$.

where $\mathbf{a}$ is a $(d+1)$-dimensional vector and each of its dimensions is a random value drawn from a normal distribution $\mathcal{N}(0, 1)$. Let $\tilde{\mathbf{x}}$ be $\mathbf{x}$ in the projected space, i.e.,

$$\tilde{\mathbf{x}} = \langle h_1(\bar{\mathbf{x}}), \ldots, h_m(\bar{\mathbf{x}}) \rangle. \tag{2}$$

4) BUILDING TREES IN AN *m*-DIMENSIONAL SPACE

Based on the norm order, we equally partition $\mathbf{X}$ into b disjoint subsets $\mathbf{X}_1$,..., $\mathbf{X}_b$ where $\bigcup_{i=1}^{b} \mathbf{X}_i = \mathbf{X}$ and $\mathbf{X}_i \cap \mathbf{X}_j = \varnothing$ $(i \neq j)$. We set $b = O(\log n)$. We then make b groups G_1,..., G_b, and the group G_i contains $\mathbf{X}_1$,..., $\mathbf{X}_i$ (i.e., $G_i \subseteq G_{i+1}$). Last, for each group G_i, we build an in-memory R-tree $\mathcal{R}_i$ on the set of projected vectors $\tilde{\mathbf{x}}$ of $\mathbf{x} \in \bigcup_{j=1}^{i} \mathbf{X}_j$. Fig. 1 depicts our data structure.

Let M_i be the largest norm in $\mathbf{X}_i$. From Cauchy–Schwarz inequality, i.e., $\mathbf{x} \cdot \mathbf{q} \leq \|\mathbf{x}\|\|\mathbf{q}\|$, we can filter the subsets $\mathbf{X}_i$ if $M_i\|\mathbf{q}\| < \tau$. (All $\mathbf{x} \in \bigcup_{j=i}^{b} \mathbf{X}_j$ do not satisfy $\mathbf{x} \cdot \mathbf{q} \leq \tau$.) Therefore, in this case, we can focus only on $\mathcal{R}_{i-1}$. For example, in Fig. 1, if $M_3\|\mathbf{q}\| < \tau$, we use only $\mathcal{R}_2$. One may notice that this also can be achieved by building an R-tree for each subset $\mathbf{X}_i$ (not each group). However, if we do this, we have to traverse multiple R-trees (i.e., from the R-tree on $\mathbf{X}_1$ to the one on $\mathbf{X}_{i-1}$), which involves many node traversal costs. We avoid this drawback by making the groups, and this approach reduces the sample complexity (see Remark 2).

B. ONLINE ALGORITHM

We next describe our online algorithm that exploits our data structure to quickly and accurately estimate the cardinality.

1) FILTERING UNNECESSARY VECTORS

Given a query $\mathbf{q}$ and a threshold τ, we first compute the norm $\|\mathbf{q}\|$. Next, we filter unnecessary vectors from Cauchy–Schwarz inequality, as explained before. That is, we obtain the group G_i where

$$i = \max_{1 \leq j \leq b} j \; s.t. \; M_j\|q\| \geq \tau,$$

and all vectors in $\mathbf{X}_{i+j}$ $(j \geq 1)$ are filtered.

2) R-TREE TRAVERSAL

Then, we search R-tree *nodes* that intersect with our search range. The query $\mathbf{q}$ is transformed to $\bar{\mathbf{q}}$ via (1), and we project it into the m-dimensional space via (2) to obtain $\tilde{\mathbf{q}}$. Along with this, we compute $\tilde{r} = \sqrt{m}r$, where $r = \sqrt{M^2 + \|\mathbf{q}\|^2 - 2\tau}$. (We later explain this setting of $\tilde{r}$.) We next run a range search on $\mathcal{R}_i$. Note that, different from usual range searches which retrieve points, our range search retrieves the *leaf nodes* of $\mathcal{R}_i$ that intersect with the hypersphere centered at $\tilde{\mathbf{q}}$ and radius $\tilde{r}$. For example, in Fig. 2, our range search retrieves leaf nodes l_1 and l_2.

3) CARDINALITY ESTIMATION FROM SAMPLES

Last, we estimate the cardinality $|\mathbf{X_q}|$ by using the leaf nodes obtained above. Let $\mathbf{V}_{\tilde{\mathbf{q}},\tilde{r}}$ be the set of projected vectors that belong to the leaf nodes. We run random sampling in $\mathbf{V}_{\tilde{\mathbf{q}},\tilde{r}}$. (Assume that $|\mathbf{V}_{\tilde{\mathbf{q}},\tilde{r}}| \gg s$, where s is the sample size. If $|\mathbf{V}_{\tilde{\mathbf{q}},\tilde{r}}| \leq s$, we simply scan $\mathbf{V}_{\tilde{\mathbf{q}},\tilde{r}}$.) We prepare a counter c, and then do the following:

1) We randomly sample a projected vector from $\mathbf{V}_{\tilde{\mathbf{q}},\tilde{r}}$
2) Let $\tilde{\mathbf{x}}$ be the sampled vector, and we compute $\mathbf{x} \cdot \mathbf{q}$.
3) If $\mathbf{x} \cdot \mathbf{q} \geq \tau$, we increase c by one.
4) We perform (i) to (iii) s times.

Finally, we use $\frac{c \cdot |\mathbf{V}_{\tilde{\mathbf{q}},\tilde{r}}|}{s}$ as our estimation for $|\mathbf{X_q}|$.

C. ANALYSIS

Let $d = O(1)$ for simplicity. This section introduces our theoretical results. We first clarify the space complexity of our data structure.

Proposition 1 (Space complexity): Our data structure requires $O(n \log n)$ space.

Proof: Recall that we have $b = O(\log n)$ R-trees and the group G_i has $G_i \subset G_{i+1}$ for $i \in [1, b-1]$. The projected vector $\tilde{\mathbf{x}}$ of $\mathbf{x}$ is hence replicated at most $O(\log n)$ times. From this fact, this theorem holds. □

Next, we analyze the accuracy of our online algorithm. Let $\bar{t} = dist(\bar{\mathbf{x}}_1, \bar{\mathbf{x}}_2)$ for any two vectors $\mathbf{x_1}$ and $\mathbf{x_2}$, where $dist(\cdot, \cdot)$ is the Euclidean distance between two vectors. Also, let $\tilde{t} = dist(\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2)$. We introduce the following [34].

Lemma 1: Let $t = \frac{\tilde{t}}{\sqrt{m}}$. We have $\mathbb{E}[\tilde{t}] = \sqrt{m} \times \bar{t}$, and t is an unbiased estimator of $\bar{t}$.

From this lemma, we can expect that the vectors $\bar{\mathbf{x}}$ satisfying $\|\bar{\mathbf{x}} - \bar{\mathbf{q}}\| \leq r$ have $\|\tilde{\mathbf{x}} - \tilde{\mathbf{q}}\| \leq \tilde{r} = \sqrt{m}r$. In this case, the random sampling nature provides that

Corollary 1: $\mathbb{E}\left[\frac{c \cdot |\mathbf{V}_{\tilde{\mathbf{q}},\tilde{r}}|}{s}\right] = |\mathbf{X_q}|$.

From Lemma 1, we can derive the probability p of sampling a vector $\tilde{\mathbf{x}}$ of $\mathbf{x}$ satisfying $\mathbf{x} \cdot \mathbf{q} \geq \tau$:

$$p = \frac{|\mathbf{X_q}|}{|\mathbf{V}_{\tilde{\mathbf{q}},\tilde{r}}|}. \tag{3}$$

We use this observation to derive the variance of our estimation.

Lemma 2: $\mathrm{Var}\left[\frac{c \cdot |\mathbf{V}_{\tilde{\mathbf{q}},\tilde{r}}|}{s}\right] = \frac{|\mathbf{X_q}|(|\mathbf{V}_{\tilde{\mathbf{q}},\tilde{r}}| - |\mathbf{X_q}|)}{s}$.

Proof: Consider an independent random variable X_i, where $X_i = 1$ with probability p and $X_i = 0$ with probability $1 - p$.

Let $X = \sum_{i=1}^{s} X_i$. (This corresponds to c in our algorithm.) Because $\mathrm{Var}[X_i] = p(1-p)$, $\mathrm{Var}[X] = sp(1-p)$. By replacing X with $\frac{c \cdot |\mathbf{V}_{\tilde{\mathbf{q}},\tilde{r}}|}{s}$, we have

$$\mathrm{Var}\left[\frac{\mathrm{c} \cdot |\mathbf{V}_{\tilde{\mathbf{q}},\tilde{\mathrm{r}}}|}{\mathrm{s}}\right] = \left(\frac{|\mathbf{V}_{\tilde{\mathbf{q}},\tilde{r}}|}{s}\right)^2 \mathrm{Var}[\mathrm{c}]$$
$$= \frac{|\mathbf{V}_{\tilde{\mathbf{q}},\tilde{r}}|^2}{s} p(1-p).$$

From (3), this lemma holds. □

This lemma indicates that (i) larger samples reduce the variance of our estimation and (ii) the variance is low as long as we do not have $p \approx \frac{1}{2}$. Last, we need to show the following:

Lemma 3: Our algorithm returns $(1 \pm \epsilon)|\mathbf{X_q}|$ estimation with at least $1-\delta$ probability by setting $s = O\left(\frac{|\mathbf{V}_{\tilde{\mathbf{q}},\tilde{r}}|}{|\mathbf{X_q}|\epsilon^2} \ln \frac{1}{\delta}\right)$.

Proof: We use the same notation as in the proof of Lemma 2. Note that $\mathbb{E}[X] = sp$. We need s such that $\Pr[|X - \mathbb{E}[X]| \geq \epsilon\mathbb{E}[X]] \leq \delta$ for $0 < \epsilon < 1$ and $0 < \delta < 1$. From Chernoff bound, we have

$$\Pr\left[|X - \mathbb{E}[X]| \geq \epsilon\mathbb{E}[X]\right] \leq 2e^{-\frac{\mathbb{E}[X]\epsilon^2}{3}} = 2e^{-\frac{sp\epsilon^2}{3}}.$$

Therefore, we need

$$\delta \leq 2e^{-\frac{sp\epsilon^2}{3}}.$$

Then we have

$$s \geq \frac{3}{p}\left(\frac{1}{\epsilon}\right)^2 \ln\left(\frac{1}{\delta}\right) = O\left(\frac{1}{p\epsilon^2} \ln\left(\frac{1}{\delta}\right)\right).$$

From (3),

$$s = O\left(\frac{|\mathbf{V}_{\tilde{\mathbf{q}},\tilde{r}}|}{|\mathbf{X_q}|\epsilon^2} \ln\left(\frac{1}{\delta}\right)\right).$$

Consequently, this lemma holds. □

From the above results, Theorem 1 holds.

Remark 2: Our filtering based on Cauchy–Schwarz inequality improves the sampling complexity. If we do not filter any vectors, unnecessary vectors can be projected to the hypersphere, which increases the size of $\mathbf{V}_{\tilde{\mathbf{q}},\tilde{r}}$. Recall that we efficiently avoid this in $O(\log n)$ time.

Last, we introduce that our online algorithm is *sub*-linear to n in the worst case.

Proposition 2 (Time complexity): Our online algorithm runs in $O(n^{1-O(1)} + N + s)$ time, where N is the number of leaf nodes of $\mathcal{R}_i$ that intersect with the hypersphere.

Proof: The filtering step requires at most $O(\log n)$ time, because we have $b = O(\log n)$ subsets. The R-tree traversal step requires $O(n^{1-O(1)} + N)$ time [20], where N is the number of leaf nodes of $\mathcal{R}_i$ that intersect with the hypersphere. Last, the random sampling requires $O(s)$ time, and getting $|\mathbf{V}_{\tilde{\mathbf{q}},\tilde{r}}|$ needs $O(N)$ time. □

V. EXPERIMENT

This section reports our experimental results. All experiments were conducted on a Ubuntu 16.04 LTS machine with 3.0 GHz Core i9-9980XE CPU and 128 GB RAM.

TABLE 1. Dataset Statistics

	Amazon-M	Amazon-K	Netflix
#users	2,088,620	1,406,890	480,189
#items	200,941	430,530	17,770

A. SETTING

We used three real datasets: Amazon-M (movie), Amazon-K (Kindle) [11], and Netflix.[2] They are sets of ratings, and the rating scales are 1 to 5. To generate user and item vectors, we ran the Matrix Factorization in [4] by setting $d = 50$, similarly to [1], [15], [32]. The numbers of users and items are shown in Table 1. For evaluation, we used item vectors as queries such that $|\mathbf{X_q}| \geq 1$, and a set of user vectors was $\mathbf{X}$.

We compared our algorithm Facetts[3] with the following state-of-the-art and some variants of Facetts.

1) FEXIPRO [15]: This is a state-of-the-art *exact* algorithm for inner product search.
2) ScaNN [9]: This is a state-of-the-art approximation algorithm for k-maximum inner product search. Because this algorithm does not allow τ to be specified, we set $k = |\mathbf{X_q}|$. (This is not practical, but we can see its performance in the best setting for ScaNN.)
3) IS [32]: This is a state-of-the-art cardinality estimation algorithm for angular distance.
4) Facetts-wop: This is a variant of our algorithm, and it does not partition $\mathbf{X}$ (so Cauchy–Schwarz inequality cannot be used and a single R-tree is used).
5) Facetts-wog: This is also a variant of our algorithm, and it does not group multiple subsets (so an R-tree is built on each subset). After filtering unnecessary subsets, this algorithm runs tree traversal and sampling for each not-filtered R-tree, and the number of samples for each subset is $s' = s/a$, where a is the number of not-filtered subsets.

For ScaNN, we used the original implementation.[4] We implemented the others in C++ and complied by g++ 5.5.0 with −O3 flag.

B. RESULT OF PRE-PROCESSING

We first measured the pre-processing time of Facetts. On Amazon-M, Amazon-K, and Netflix, the pre-processing times were 66.5, 43.9, and 14.1 seconds, respectively. We see that the pre-processing time of Facetts is reasonable, and scales linearly to $|\mathbf{X}|$.

C. RESULT OF ONLINE PROCESSING

We measured (i) the average time to estimate the cardinality and (ii) the estimation error. We used absolute percentage

[2] [Online]. Available: https://www.cs.uic.edu/liub/Netflix-KDD-Cup-2007.html

[3] Fast and accurate cardinality estimation via tree traversal sampling. We set $m = 5$. Its code is available at https://github.com/peitaw22/facetts.

[4] [Online]. Available: https://github.com/google-research/google-research/tree/master/scann

TABLE 2. Impact of #samples on Avg. APE, Median APE, and Time [msec] on Amazon-M

#samples	2000			4000			6000		
	APE_{avg}	APE_{med}	Time	APE_{avg}	APE_{med}	Time	APE_{avg}	APE_{med}	Time
IS	0.988	0.902	1.7	0.980	0.891	2.8	0.976	0.884	4.0
Facetts-wop	0.141	0.018	3.0	0.145	0.015	4.2	0.113	0.019	5.3
Facetts-wog	8.014	8.995	4.7	7.980	3.045	5.7	7.954	9.064	6.7
Facetts	**0.120**	**0.012**	3.0	**0.104**	**0.011**	4.2	**0.083**	**0.008**	5.4

TABLE 3. Impact of #samples on Avg. APE, Median APE, and Time [msec] on Amazon-K

#samples	2000			4000			6000		
	APE_{avg}	APE_{med}	Time	APE_{avg}	APE_{med}	Time	APE_{avg}	APE_{med}	Time
IS	0.901	0.855	1.7	0.892	0.846	2.8	0.891	0.842	3.9
Facetts-wop	0.126	0.023	2.3	0.110	0.016	3.4	0.100	0.017	4.5
Facetts-wog	7.451	8.339	3.1	7.455	8.402	4.0	7.438	8.415	5.0
Facetts	**0.081**	**0.017**	2.3	**0.070**	**0.013**	3.4	**0.060**	**0.011**	4.5

TABLE 4. Impact of #samples on Avg. APE, Median APE, and Time [msec] on Netflix

#samples	2000			4000			6000		
	APE_{avg}	APE_{med}	Time	APE_{avg}	APE_{med}	Time	APE_{avg}	APE_{med}	Time
IS	0.440	0.357	1.0	0.421	0.348	1.8	0.410	0.342	2.6
Facetts-wop	0.209	0.061	1.2	0.181	0.047	2.2	0.141	0.039	3.1
Facetts-wog	3.690	2.733	0.9	3.667	2.711	1.7	3.645	2.676	2.4
Facetts	**0.116**	**0.036**	1.0	**0.091**	**0.026**	1.9	**0.076**	**0.026**	2.7

TABLE 5. Avg. APE, Median APE, and Time [msec] of FEXIPRO and ScaNN

Dataset	Amazon-M			Amazon-K			Netflix		
	APE_{avg}	APE_{med}	Time	APE_{avg}	APE_{med}	Time	APE_{avg}	APE_{med}	Time
FEXIPRO	0	0	521.8	0	0	325.8	0	0	52.0
ScaNN	0.106	0.047	81.0	0.105	0.052	50.0	0.187	0.118	4.6

error (APE) to measure the error, and

$$\text{APE} = \left| \frac{|\mathbf{X_q}| - C_{est}}{|\mathbf{X_q}|} \right|,$$

where C_{est} is the estimated cardinality. For ScaNN, C_{est} is the number of vectors in the result set that satisfy the threshold. (As ScaNN is an approximation algorithm, it has $C_{est} \leq |\mathbf{X_q}|$.) We computed the average (denoted by APE_{avg}) and median (denoted by APE_{med}).

1) IMPACT OF #SAMPLES

We investigated the impact of s, #samples, on the estimation time and error. We set $\tau = 4.0$. The experimental results are shown in Tables 2, 3, and 4 (bold shows the smallest error among the approximation algorithms). Since FEXIPRO and ScaNN do not employ sampling, we measured their running times when $\tau = 4.0$ (the domain of τ is [0,5]), and the result is shown in Table 5.

Generally, as the number of samples increases, each sampling-based algorithm returns a smaller error while the running time becomes longer. This is a natural result from the theoretical result. We have two important observations.

TABLE 6. Decomposed Time of Facetts [msec] ($s = 2000$)

Dataset	Amazon-M	Amazon-K	Netflix
Filtering	0.005	0.006	0.005
Tree trav.	1.353	0.811	0.159
Sampling	1.721	1.467	0.889

First, Facetts returns a much smaller error than IS on all datasets. For example, APE_{avg} of Facetts is about 10 times smaller than that of IS on Amazon-M and Amazon-K while keeping competitive running time. (The time difference is derived from tree traversal, because IS does not employ such an approach.) This result demonstrates the effectiveness of our approach and shows that simply utilizing a cardinality estimation algorithm for other data space is not promising. Second, the error of Facetts is sufficiently small even when the number of samples is small (e.g., $s = 2000$), whereas IS incurs a large error. Looking at APE_{med}, we see that Facetts provides less than 5% error for at least half queries.

Next, let us compare Facetts with the state-of-the-art search-based algorithms FEXIPRO and ScaNN. It can be seen

TABLE 7. Impact of Threshold on Average APE, Median APE, and Time [msec] on Amazon-M

τ	3.5			4			4.5		
	APE_{avg}	APE_{med}	Time	APE_{avg}	APE_{med}	Time	APE_{avg}	APE_{med}	Time
FEXIPRO	0	0	674.0	0	0	521.8	0	0	327.5
ScaNN	0.058	0.012	120.7	0.106	0.047	81.0	0.187	0.123	45.4
IS	1.063	0.961	1.8	0.988	0.902	1.7	0.861	0.766	1.5
Facetts-wop	0.074	0.009	3.1	0.141	0.018	3.0	0.244	0.046	2.8
Facetts	0.061	0.006	3.2	0.083	0.012	3.1	0.147	0.032	2.4

TABLE 8. Impact of Threshold on Average APE, Median APE, and Time [msec] on Amazon-K

τ	3.5			4			4.5		
	APE_{avg}	APE_{med}	Time	APE_{avg}	APE_{med}	Time	APE_{avg}	APE_{med}	Time
FEXIPRO	0	0	432.9	0	0	325.8	0	0	192.2
ScaNN	0.056	0.018	79.1	0.105	0.052	50.0	0.191	0.157	23.9
IS	0.969	0.909	1.8	0.901	0.855	1.7	0.769	0.714	1.5
Facetts-wop	0.091	0.013	2.3	0.126	0.023	2.3	0.216	0.052	2.2
Facetts	0.059	0.011	2.3	0.081	0.017	2.3	0.134	0.035	1.9

TABLE 9. Impact of Threshold on Average APE, Median APE, and Time [msec] on Netflix

τ	3.5			4			4.5		
	APE_{avg}	APE_{med}	Time	APE_{avg}	APE_{med}	Time	APE_{avg}	APE_{med}	Time
FEXIPRO	0	0	116.5	0	0	52.0	0	0	15.1
ScaNN	0.097	0.069	12.5	0.187	0.118	4.6	0.348	0.234	1.4
IS	0.497	0.463	1.2	0.440	0.357	1.0	0.525	0.363	0.9
Facetts-wop	0.109	0.035	1.3	0.209	0.061	1.2	0.432	0.155	1.1
Facetts	0.067	0.026	1.3	0.116	0.036	1.0	0.293	0.074	0.7

TABLE 10. Memory Usage [GB]

Dataset	Amazon-M	Amazon-K	Netflix
IS	9.93	6.69	2.28
Facetts	3.15	2.21	0.66

that Facetts is much faster than FEXIPRO, the exact state-of-the-art algorithm (so its error is 0). Also, Facetts is faster than ScaNN. As Table 6 shows, each component of Facetts does not incur a large overhead, so its estimation time is quite short. Notice that, even when $s = 2000$, Facetts returns a smaller error than ScaNN. This result confirms that search-based algorithms are not appropriate for cardinality estimation.

We compare Facetts with its variant Facetts-wop to evaluate the effectiveness of our partitioning strategy. Tables 2–4 show that Facetts always returns a smaller error. The reason is derived from filtering unnecessary vectors. Due to the projection into the m-dimensional space, vectors $\mathbf{x}$ such that $\mathbf{x} \cdot \mathbf{q} < \tau$ can also exist in the leaf nodes intersecting with the hypersphere. Facetts avoids this case as much as possible, so its estimation error is smaller.

Last, we compare Facetts with Facetts-wog. We see that Facetts significantly outperforms Facetts-wog, and Facetts-wog incurs a very large error. This result clarifies the effectiveness of our grouping approach (see Remark 2). Since Facetts-wog needs sampling in multiple R-trees and the number of samples for each R-tree becomes smaller, it cannot effectively sample vectors $\mathbf{x}$ such that $\mathbf{x} \cdot \mathbf{q} \geq \tau$. In addition, due to the traversals of multiple R-trees, its estimation time is slower than that of Facetts on Amazon-M and Amazon-K. On Netflix, on the other hand, Facetts-wog is a bit faster than Facetts. We found that the number of leaf nodes intersecting with the hypersphere in Facetts-wog is smaller than that in Facetts for the Netflix case. Besides, the number of not-filtered subsets is small in this case. These lead the shorter estimation time of Facetts-wog (but this does not make sense, as it incurs a large error).

2) IMPACT OF THRESHOLD τ

Tables 7–9 show the experimental results with varying τ. We set $s = 2000$ for the sampling-based algorithms. (We omit the result of Facetts-wog, because its error is too large.) As τ increases, the running times of FEXIPRO and ScaNN decrease. This is because their search space becomes smaller as the threshold becomes tight. On the other hand, the running times of the sampling-based algorithms do not change much, because their main overhead is to compute the inner product of sampled vectors and a given query vector, as we showed in Table 6.

We next observe that, as τ increases, the errors of ScaNN and the sampling-based algorithms increase. Notice that, for

a large τ, $|\mathbf{X_q}|$ decreases. Then, the probability of sampling vectors $\mathbf{x}$ such that $\mathbf{x} \cdot \mathbf{q} \geq \tau$ decreases. The errors of the sampling-based algorithms hence increase. However, Facetts retains small errors even for large τ, see its APE_{med}. ScaNN has a smaller APE_{avg} than Facetts when τ is small, but the difference is slight and Facetts has much shorter estimation time.

3) MEMORY USAGE

Table 10 compares the memory usage of Facetts with that of the state-of-the-art cardinality estimation algorithm IS. We see that Facetts needs less memory than IS, showing that Facetts outperforms IS w.r.t. estimation time, accuracy, and memory usage.

VI. CONCLUSION

In this article, we addressed the problem of cardinality estimation in inner product spaces. Because machine-learning techniques are widespread and they often use inner products of high-dimensional vectors, this problem also has important applications. Existing techniques for (approximate) inner product search and cardinality estimation in other data spaces actually can be employed to solve our problem. However, they are expensive for our problem as they involve many data accesses.

We therefore proposed a fast and accurate algorithm for inner product cardinality estimation. Our algorithm employs a new idea: transforming the inner product cardinality into range count estimation in a low-dimensional Euclidean space. We theoretically analyzed the performance of our algorithm, and it can enjoy a performance guarantee. Our extensive experiments also demonstrate that our algorithm yields accurate cardinality estimation with fast computation time on real datasets.

REFERENCES

[1] D. Amagata and T. Hara, "Reverse maximum inner product search: How to efficiently find users who would like to buy my item?," in *Proc. 15th ACM Conf. Recommender Syst.*, 2021, pp. 273–281.

[2] Y. Bachrach et al., "Speeding up the xbox recommender system using a euclidean transformation for inner-product spaces," in *Proc. 8th ACM Conf. Recommender Syst.*, 2014, pp. 257–264.

[3] M. S. Charikar, "Similarity estimation techniques from rounding algorithms," in *Proc. 34th Annu. ACM Symp. Theory Comput.*, 2002, pp. 380–388.

[4] W.-S. Chin, B.-W. Yuan, M.-Y. Yang, Y. Zhuang, Y.-C. Juan, and C.-J. Lin, "Libmf: A library for parallel matrix factorization in shared-memory systems," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 2971–2975, 2016.

[5] G. Cormode and S. Muthukrishnan, "An improved data stream summary: The count-min sketch and its applications," *J. Algorithms*, vol. 55, no. 1, pp. 58–75, 2005.

[6] X. Dai, X. Yan, K. K. Ng, J. Liu, and J. Cheng, "Norm-explicit quantization: Improving vector quantization for maximum inner product search," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 51–58.

[7] Q. Ding, H.-F. Yu, and C.-J. Hsieh, "A fast sampling algorithm for maximum inner product search," in *Proc. 22nd Int. Conf. Artif. Intell. Statist.*, 2019, pp. 3004–3012.

[8] P. Flajolet, É. Fusy, O. Gandouet, and F. Meunier, "Hyperloglog: The analysis of a near-optimal cardinality estimation algorithm," in *Proc. Discrete Math. Theor. Comput. Sci.*, 2007, pp. 137–156.

[9] R. Guo et al., "Accelerating large-scale inference with anisotropic vector quantization," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 3887–3896.

[10] H. Harmouch and F. Naumann, "Cardinality estimation: An experimental survey," *Proc. VLDB Endowment*, vol. 11, no. 4, pp. 499–512, 2017.

[11] R. He and J. McAuley, "Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering," in *Proc. 25th Int. Conf. World Wide Web*, 2016, pp. 507–517.

[12] K. Hirata, D. Amagata, and T. Hara, "Solving diversity-aware maximum inner product search efficiently and effectively," in *Proc. 16th ACM Conf. Recommender Syst.*, 2022, pp. 198–207.

[13] Q. Huang, G. Ma, J. Feng, Q. Fang, and A. K. Tung, "Accurate and fast asymmetric locality-sensitive hashing scheme for maximum inner product search," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov Data Mining*, 2018, pp. 1561–1570.

[14] W. B. Johnson and J. Lindenstrauss, "Extensions of Lipschitz mappings into a hilbert space 26," *Contemporary Math.*, vol. 26, pp. 189–206, 1984.

[15] H. Li, T. N. Chan, M. L. Yiu, and N. Mamoulis, "Fexipro: Fast and exact inner product retrieval in recommender systems," in *Proc. ACM Int. Conf. Manage. Data*, 2017, pp. 835–850.

[16] J. Liu, X. Yan, X. Dai, Z. Li, J. Cheng, and M.-C. Yang, "Understanding and improving proximity graph based maximum inner product search," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 139–146.

[17] M. Mattig, T. Fober, C. Beilschmidt, and B. Seeger, "Kernel-based cardinality estimation on metric data," in *Proc. Int. Conf. Extending Database Technol.*, 2018, pp. 349–360.

[18] H. Nakama, D. Amagata, and T. Hara, "Approximate top-k inner product join with a proximity graph," in *Proc. IEEE Int. Conf. Big Data*, 2021, pp. 4468–4471.

[19] B. Neyshabur and N. Srebro, "On symmetric and asymmetric LSHs for inner product search," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1926–1934.

[20] J. Qi, Y. Tao, Y. Chang, and R. Zhang, "Theoretically optimal and empirically efficient r-trees with strong parallelizability," *Proc. VLDB Endowment*, vol. 11, no. 5, pp. 621–634, 2018.

[21] P. Ram and A. G. Gray, "Maximum inner-product search using cone trees," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discov Data Mining*, 2012, pp. 931–939.

[22] A. Shrivastava and P. Li, "Asymmetric LSH (ALSH) for sublinear time maximum inner product search (MIPS)," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2321–2329.

[23] A. Shrivastava and P. Li, "Asymmetric minwise hashing for indexing binary inner products and set containment," in *Proc. World Wide Web*, 2015, pp. 981–991.

[24] R. Spring and A. Shrivastava, "Scalable and sustainable deep learning via randomized hashing," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discov Data Mining*, 2017, pp. 445–454.

[25] J. Sun, G. Li, and N. Tang, "Learned cardinality estimation for similarity queries," in *Proc. Int. Conf. Manage. Data*, 2021, pp. 1745–1757.

[26] C. Teflioudi and R. Gemulla, "Exact and approximate maximum inner product search with LEMP," *ACM Trans. Database Syst.*, vol. 42, no. 1, pp. 1–49, 2017.

[27] C. Teflioudi, R. Gemulla, and O. Mykytiuk, "LEMP: Fast retrieval of large entries in a matrix product," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2015, pp. 107–122.

[28] D. Ting, "Approximate distinct counts for billions of datasets," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2019, pp. 69–86.

[29] H. Wang et al., "A DNN-based cross-domain recommender system for alleviating cold-start problem in e-commerce," *IEEE Open J. Ind. Electron. Soc.*, vol. 1, pp. 194–206, 2020.

[30] X. Wang, C. Qu, W. Wu, J. Wang, and Q. Zhou, "Are we ready for learned cardinality estimation?," *Proc. VLDB Endowment*, vol. 14, no. 9, pp. 1640–1654, 2021.

[31] P. Wu and G. Cong, "A unified deep model of learning from both data and queries for cardinality estimation," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2021, pp. 2009–2022.

[32] X. Wu, M. Charikar, and V. Natchu, "Local density estimation in high dimensions," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 5296–5305.

[33] X. Yan, J. Li, X. Dai, H. Chen, and J. Cheng, "Norm-ranging lsh for maximum inner product search," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 2952–2961.

[34] B. Zheng, Z. Xi, L. Weng, N. Q. V. Hung, H. Liu, and C. S. Jensen, "PM-LSH: A fast and accurate LSH framework for high-dimensional approximate NN search," *Proc. VLDB Endowment*, vol. 13, no. 5, pp. 643–655, 2020.

[35] Z. Zhou, S. Tan, Z. Xu, and P. Li, "Möbius transformation for fast inner product search on graph," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 8218–8229.

KOHEI HIRATA is currently working toward the master's degree with the Department of Multimedia Engineering Graduate School of Information Science and Technology Osaka University, Osaka, Japan. His research focuses on high-dimensional data retrieval.

DAICHI AMAGATA (Member, IEEE) received the B.E., M.Sc., and Ph.D. degrees from Osaka University, Osaka, Japan, in 2012, 2014, and 2015, respectively. He is currently an Assistant Professor with the Department of Multimedia Engineering Graduate School of Information Science and Technology Osaka University. His research interests include fast algorithms for databases and AI technologies.

TAKAHIRO HARA (Senior Member, IEEE) received the B.E, M.E, and Dr.E. degrees in information systems engineering from Osaka University, Osaka, Japan, in 1995, 1997, and 2000, respectively. He is currently a Full Professor with the Department of Multimedia Engineering, Osaka University. His research interests include distributed databases, peer-to-peer systems, mobile networks, and mobile computing systems. He is a Distinguished Scientist of ACM and a Member of three other learned societies.