Journal of Information Processing Vol.25 366–375 (May 2017)

[DOI: 10.2197/ipsjjip.25.366]

Regular Paper

Inverse Stereographic Projecting Hashing for Fast Similarity Search

Yui Noma[1,a)]

Received: August 1, 2016, Accepted: February 9, 2017

Abstract: Fast similarity searches that use high-dimensional feature vectors for a vast amount of multi-media data have recently become increasingly important. However, ordinary similarity searches are slow because they require a large number of floating-point operations that are proportional to the number of record data. Many studies have been done recently that propose to speed up similarity searches by converting feature vectors to bit vectors. Such similarity searches are regarded as approximations of the similarity searches over the original data. However, some of those approximations are not theoretically guaranteed since no direct approximate relations between the Euclidean and Hamming distances are given. We propose a novel hashing method that utilizes inverse-stereographic projection and gives a direct approximate relation between the Euclidean and Hamming distances in a closed-form expression. Although some studies have discussed the relationship between the two distances, to the best of our knowledge, our hashing method is the first one to give a direct approximate relation between the two distances. We also propose parameter values that are needed for our proposal method. Furthermore, we show through experiments that the proposed method has more accurate approximation than the existing random projection-based and Hamming distance-based methods for many datasets.

Keywords: Inverse Stereographic Projection, similarity search, hypersphere, hashing, binary code

1. Introduction

Acquiring data has become extremely easy with the advent of the use of multiple sensors connected via networks in recent years. In particular, it is expected that the amount of multimedia data such as video and audio will increase in the future, and similarity searches over such data will be utilized in many applications. For example, scanned biometric information could be used for micro-payments, and surveillance camera images for anomaly detection.

Similarity searches work by extracting feature vectors from the multimedia data with a given (dis)-similarity function. The search process calculates similarities between the feature vectors and finds the most similar k-records from a database. A prevalent dissimilarity function is the Euclidean distance. A naive similarity search over a vast amount of data is usually slow since it requires a large number of floating-point operations that are proportional to the number of record data. Many methods using index structures have been proposed in order to speed up similarity searches [1], [2], [3], [4]. However, these index structures are outperformed by a linear scan for high-dimensional feature vector spaces [5].

However, it is not always necessary to find k-similar records from a database. For example, in the case of biometric authentication for micro-payments, similarity searches are used to find similar items of biometric information, and the retrieved items are passed to other authentication processes. In such cases, a requirement for the similarity search is that the retrieved items contain the biometric information of the person to be authenticated. As in this example, many system requirements of the actual applications of multimedia data can be met not only by similarity searches but also by approximate-similarity searches. Although the retrieved results of approximate-similarity searches are not the same as the results of the similarity searches, most of the items in the results are the same, and such approximate results can satisfy the system requirements. In such cases, it is preferable for the approximate-similarity searches to be processed at high speed and have a high accuracy of approximation. Much research has been done on fast approximate-similarity searches in a high-dimensional feature vector space. One area of research is based on Locality-Sensitive Hashing (LSH) [6], [7], [8]. The studies on LSH consider a family of hash functions that satisfy certain conditions, take several hash functions from the family, and calculate several hash values from the feature vectors. Those feature vectors are stored in multiple hash tables. For each query, the search process calculates hash values from the query feature vector and retrieves the record feature vectors from the multiple hash tables. The actual form of the family of hash functions depends on the (dis)-similarity function used for the similarity search. Many families of hash functions have been proposed for several (dis)-similarity functions. Research has been done on the families of hash functions for the Euclidean distances [7], [8], [9], [10]. However, these methods require a large amount of memory because the hash values are integer values, and LSH requires a large number of hash tables. Hence, they

[1] System Software Laboratories, FUJITSU LABORATORIES LTD., Kawasaki, Kanagawa 211–8588, Japan
[a)] noma.yui@jp.fujitsu.com

cannot be efficiently applied in many real systems [11].

Another area of research on approximate-similarity searches involves converting feature vectors to bit vectors and conducting similarity searches whose dissimilarities are Hamming distances between the bit vectors [12], [13], [14]. The conversion from feature vectors to bit vectors can be seen as a partitioning of the feature vector space by multiple hyperplanes, and an assignment of bit vectors to the partitioned regions and the feature vectors in the regions. Hereafter, we call such a conversion "hashing that uses hyperplanes". Hashing that uses hyperplanes requires floating-point operations, and consequently, the hashing process is slow. Nevertheless, by hashing the feature vectors in a database before querying, the calculation cost is reduced during searching. Although the query feature vectors are converted to bit vectors at querying time, the calculation cost does not depend on the number of record data. Furthermore, CPUs in recent years have contained SIMD (single instruction, multiple data) instructions, and the Hamming distances are calculated at high speed. Moreover, because the modern CPU contains Field Programable Gate Arrays (FPGA), and the size of the circuits of the Hamming distance calculations is small, the Hamming distances can be calculated in massively parallel with modern CPU [15]. Hence, similarity searches that use hashing can be done at high speed for a vast amount of data without any index structures and are promising techniques. Moreover, it is known that the Hamming distance between bit vectors that are hashed from the feature vectors is probabilistically proportional to the angle between the feature vectors. However, there is no guarantee that the similarity searches that use hyperplane hashing are an approximation of the similarity searches whose dissimilarities are the Euclidean distances, since the relation between the Hamming distances and the Euclidean distances is not clear [16]. Although there are studies that aim to preserve the Euclidean distances by machine learning depending on the data distribution [17], there is no direct relation between the Hamming and Euclidean distances. Hence, in order to use these techniques in real systems, one needs to evaluate its accuracy of approximations whenever new pieces of data are inserted because there is no theoretical guarantee. Such systems require a lot of work for the users and are not preferred for real systems, in particular for enterprise systems.

There is another research areas for approximate similarity search using Product Quantizer (PQ) [18]. Furthermore, fast index structures are studied [19], [20]. However, these methods require floating-point operations and the size of the circuits are larger than the one for the Hamming distances. Hence, the degrees of parallelism are not large on FPGA compared with the similarity searches with Hamming distances. Therefore, we focus on the similarity searches whose dissimilarities are Hamming distances.

In this paper, we present our novel method for approximate-similarity search that gives such a relation between the Hamming and Euclidean distances. We consider a space with one dimension larger than the dimensionality of the feature vector space. We also consider a unit sphere in the space and the inverse stereographic projection to the unit sphere from the original feature vector space. By applying hashing that uses hyperplanes to the projected feature vector, we convert the feature vectors to bit vectors. Using the proposed method, we found that the Euclidean distance between two feature vectors was related to the angle between the projected feature vectors in a closed-form expression. This leads to the relation between the Euclidean distance between two feature vectors and the Hamming distance between the bit vectors hashed from the feature vectors. To the best of our knowledge, this is the first work giving the direct approximate relation between the two distances. By the existence of the relation, users of real systems are not required to evaluate the accuracy of the approximation whenever new pieces of data are inserted. Moreover, the users can focus on improvement of the accuracy by tuning arrangements of hyperplanes with machine learning [13], [14], [17]. In this paper, in order to clarify the relation between the two distances, we eliminated the effect of the learning of arrangements of hyperplanes, and we focused on approximate similarity searches with random projection-based and Hamming distance-based methods. We also show by experiments that the proposed method has better accuracy of approximation than the existing random projection-based and Hamming distance-based methods.

2. Related Work

One of the well-known methods that convert feature vectors to bit vectors is random projection [21]. We consider an N-dimensional vector $\vec{n}$ whose components are sampled from the standard normal distribution. The random projection is performed for an N-dimensional vector $\vec{x}$ as follows:

$$h(\vec{x}) = \mathrm{thr}(\vec{n} \cdot \vec{x}), \tag{1}$$

where

$$\mathrm{thr}(a) := \begin{cases} 1 & \text{if } a > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Equation (1) can be seen as a space partitioning of the feature vector space by the linear hyperplane whose normal vector is $\vec{n}$. The probability of collision of the hash function h is as follows:

$$P\left(h(\vec{x}) = h(\vec{y})\right) = 1 - \frac{\theta_{xy}}{\pi}, \tag{3}$$

where θ_{xy} is the angle between $\vec{x}$ and $\vec{y}$.

We consider B hash functions and create a bit vector valued function $\vec{h}(\vec{x})$ as follows.

$$\vec{h}(\vec{x}) = \left(h^{(1)}(\vec{x}), h^{(2)}(\vec{x}), \cdots, h^{(B)}(\vec{x})\right) \tag{4}$$

The probability of the difference of a certain bit of $\vec{h}(\vec{x})$ and $\vec{h}(\vec{y})$ is $\frac{\theta_{xy}}{\pi}$. Hence, the Hamming distance $\mathrm{Hamm}(\vec{h}(\vec{x}), \vec{h}(\vec{y}))$ is an unbiased-estimator of $\frac{B}{\pi}\theta_{xy}$. Therefore, the similarity searches that use the bit vectors converted from the feature vectors with $\vec{h}$ are approximations of the similarity searches whose dissimilarities are the angles between the feature vectors.

One study on batch-orthogonal locality-sensitive hashing [22] is based on the random projection method where they grouped the hyperplanes and orthogonalized them to reduce the variance of the unbiased estimator.

Another study attempted to speed up the similarity search

whose dissimilarity is the Euclidean distance by using the random projection method [16]. They focused on the cosine term in the definition of the Euclidean distance.

$$(d_{\mathrm{Euc}}(\vec{x},\vec{y}))^2 = \|\vec{x}\|^2 + \|\vec{y}\|^2 - 2\|\vec{x}\| * \|\vec{y}\| * \cos(\theta_{xy}) \tag{5}$$

They reduced the number of floating-point operations to calculate the Euclidean distance by approximating the angle θ_{xy} based on the Hamming distance between the bit vectors generated by the random projection method.

$$(d_{\mathrm{Euc}}(\vec{x},\vec{y}))^2 \approx \|\vec{x}\|^2 + \|\vec{y}\|^2 - 2\|\vec{x}\| * \|\vec{y}\| * \cos\left(\frac{\pi}{B}\mathrm{Hamm}(\vec{h}(\vec{x}),\vec{h}(\vec{y}))\right) \tag{6}$$

However, in order to use this approximation, one needs to use the floating-point operations. Thus, this method is slower than the similarity searches that uses only the Hamming distance calculation.

Spherical Hashing [23] is a hashing method that uses hyperspheres instead of hyperplanes. This method uses machine learning to determine the positions and the radius of the hyperspheres. Hence, the relation between the Euclidean distances and the Hamming distances is not clear. There is also a hashing method involved with hyperspheres called spherical LSH [24]. This method treats only feature vectors that exist on a hypersphere. Furthermore, the hash value is not a binary. Hence, one cannot speed up the similarity searches by using the bit vectors and the Hamming distances with spherical LSH. A study was done on extending the random projection by combining kernel methods, referred to as Kernelized LSH [25], but again the relation between the Euclidean distances and the Hamming distances was not clear.

3. Proposed Method

We propose a hashing method that uses inverse stereographic projection in Section 3.2. Before we explain our hashing method, we introduce a higher dimensional analogue of the inverse stereographic projection. In the following, we assume that the mean vector of a dataset is zero.

3.1 Inverse Stereographic Projection

Let V be the feature vector space and let the dimensionality of V be N. We also denote the coordinate of V by x_i for $i = 1, \cdots, N$. We define the inverse stereographic projection f^{-1} from V to an $N+1$-dimensional space $\tilde{V}$ as follows:

$$f^{-1}(x_1, x_2, \cdots, x_N; d) = \left(\frac{2dx_1}{d^2+r^2}, \cdots, \frac{2dx_N}{d^2+r^2}, \frac{-d^2+r^2}{d^2+r^2}\right), \tag{7}$$

where $r^2 := \sum_{i=1}^{N} x_i^2$. The parameter d is a given positive number; we discuss the value of d in a later section. In the case of $N = 2$, the function f^{-1} is well known. For example, it is used to map a complex plane to a sphere. Equation (7) is a higher-dimensional analog of the well-known function.

Inverse stereographic projection can be considered from a geometric viewpoint. The projection is represented in **Fig. 1**. The details are as follows. Let us denote the coordinate of $\tilde{V}$ by $\tilde{x}_i$ for $i = 1, \cdots, N+1$. Place a unit sphere whose center is the origin of $\tilde{V}$ and denote it as S. The image of f^{-1} is S. We call the common set of S and the hyperplane defined by $\tilde{x}_{N+1} = 0$ the equator, $(0, \cdots, 1) \in \tilde{V}$ the north pole, and $(0, \cdots, -1) \in \tilde{V}$ the south pole. We embed V in the plane $\tilde{x}_{N+1} = -d + 1$. When one draws a segment from the north pole to the embedded space V, the segment has an intersection point in S. The function f^{-1} maps the intersection point of the segment and the embedded space V to the intersection point of the segment and S.

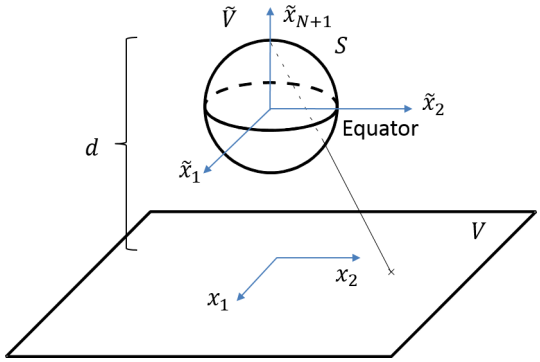

Fig. 1 Inverse-stereographic projection.

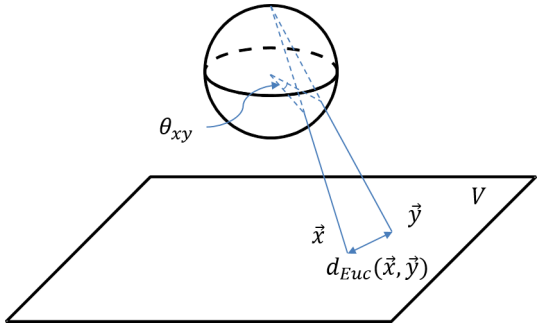

Fig. 2 Relation between θ_{xy} and $d_{Euc}(\vec{x},\vec{y})$.

We denote the angle between the projected feature vectors $f^{-1}(\vec{x}; d)$, $f^{-1}(\vec{y}; d)$ as θ_{xy}. See **Fig. 2** for detail. The cosine of θ_{xy} is as follows:

$$\cos(\theta_{xy}) = \frac{f^{-1}(\vec{x};d)\cdot f^{-1}(\vec{y};d)}{\|f^{-1}(\vec{x})\| * \|f^{-1}(\vec{y})\|} = \frac{\frac{4}{d^2}\vec{x}\cdot\vec{y} + \left(1-\frac{r_x^2}{d^2}\right)\left(1-\frac{r_y^2}{d^2}\right)}{\left(1+\frac{r_x^2}{d^2}\right)\left(1+\frac{r_y^2}{d^2}\right)} \tag{8}$$

We use the following identity.

$$(d_{Euc}(\vec{x},\vec{y}))^2 = r_x^2 + r_y^2 - 2\vec{x}\cdot\vec{y} \tag{9}$$

By using this identity, we arrange Eq. (8) as follows:

$$(d_{Euc}(\vec{x},\vec{y}))^2/d^2 = \left(1+\frac{r_x^2}{d^2}\right)\left(1+\frac{r_y^2}{d^2}\right)\frac{1}{2}\left(1-\cos(\theta_{xy})\right) \tag{10}$$

Equation (10) shows a direct relation between the Euclidean distance defined in V and the angles defined in $\tilde{V}$.

We denote the factor in Eq. (10) as follows:

$$F(r,d) := \left(1+\frac{r^2}{d^2}\right) \tag{11}$$

If the factor $F(r,d)$ is independent of the feature vector, Eq. (10)

is a proportional relation between the Euclidean distance in V and the cosine in $\tilde{V}$. Moreover, $1 - \cos(\theta)$ is a monotonically increasing function of θ. Hence, the similarity search whose dissimilarity is the angle in $\tilde{V}$ is equivalent to the similarity search whose dissimilarity is the Euclidean distance in V. If the value of factor $F(r, d)$ does not vary greatly for the feature vectors, we can regard the factor as a small disturbance. Therefore, in such a case, both similarity search results are almost the same. In Section 4.1, we discuss the dependence of the value of $F(r, d)$ on the feature vectors and on the value of d in order to reduce the variation of $F(r, d)$.

3.2 Inverse Stereographic Projection Hashing

In the random projection method, the angles in $\tilde{V}$ can be estimated by the Hamming distances between the bit vectors generated by a random projection in $\tilde{V}$. Hence, we propose the following hashing method, Inverse Stereographic Projection Hashing (ISPH):

$$\vec{\tilde{h}}(\vec{x}) := \left(\tilde{h}^{(1)}(\vec{x}), \tilde{h}^{(2)}(\vec{x}), \cdots, \tilde{h}^{(B)}(\vec{x})\right), \tag{12}$$

where

$$\begin{aligned}\tilde{h}^{(m)}(\vec{x}) &:= \mathrm{thr}(\vec{\tilde{n}}^{(m)} \cdot f^{-1}(\vec{x}; d)) \\ &= \mathrm{thr}\left(\frac{2d\tilde{n}_1^{(m)} x_1}{d^2 + r^2} + \cdots + \frac{2d\tilde{n}_N^{(m)} x_N}{d^2 + r^2} + \frac{\tilde{n}_{N+1}^{(m)}(-d^2 + r^2)}{d^2 + r^2}\right).\end{aligned} \tag{13}$$

The vectors $\vec{\tilde{n}}^{(m)}$ are $N+1$ dimensional vectors whose elements are sampled from the standard normal distribution. Because the denominators are always positive and function thr does not depend on the scalar multiplication, $\tilde{h}^{(m)}$ can be simplified as follows:

$$\tilde{h}^{(m)}(\vec{x}) = \mathrm{thr}\left(\tilde{n}_1^{(m)} x_1 + \cdots + \tilde{n}_N^{(m)} x_N + \frac{\tilde{n}_{N+1}^{(m)}(-d^2 + r^2)}{2d}\right).$$

Although Eq. (13) looks like kernelized LSH [25], it is not kernelized LSH because the function f^{-1} does not satisfy the Mercer condition.

By using ISPH $\vec{\tilde{h}}$ and Eq. (10) we arrive at the following relation:

$$\begin{aligned}(d_{Euc}(\vec{x}, \vec{y}))^2 / d^2 &\approx \left(1 + \frac{r_x^2}{d^2}\right)\left(1 + \frac{r_y^2}{d^2}\right) \\ &\times \frac{1}{2}\left(1 - \cos\left(\frac{\pi}{B}\mathrm{Hamm}(\vec{\tilde{h}}(\vec{x}), \vec{\tilde{h}}(\vec{y}))\right)\right)\end{aligned} \tag{14}$$

Equation (14) is a direct approximate relation between the Euclidean distance and the Hamming distance, and is one of our main contributions.

3.3 Inverse Stereographic Projection and Hyperspheres

We discuss here the interpretation of ISPH from a geometric viewpoint. In the case of $N = 2$, a common set of S and a hyperplane in $\tilde{V}$ correspond to a hypersphere of a hyperplane in V under the projection f^{-1}. In the following, we show that the same correspondence holds in the case of $N > 2$.

An affine hyperplane $\tilde{H}$ in $\tilde{V}$, e.g., whose dimensionality is N, is defined by the following equation:

$$\tilde{\phi}(\tilde{x}) = \vec{\tilde{n}} \cdot \vec{\tilde{x}} + \tilde{b} = 0 \tag{15}$$

where $\vec{\tilde{n}} = (\tilde{n}_1, \tilde{n}_2, \cdots, \tilde{n}_{N+1})$ is the normal vector of $\tilde{H}$ and $\tilde{b}$ is an offset. The affine hyperplane $\tilde{H}$ consists of $\tilde{x}$ that satisfies Eq. (15). When the affine hyperplane $\tilde{H}$ and S have a common set, the preimage of the common set is represented by the following equation:

$$0 = \tilde{\phi}(f^{-1}(x; d)) \Leftrightarrow 0 = \sum_{i=1}^{N} \tilde{n}_i \frac{2dx_i}{d^2 + r^2} + \tilde{n}_{N+1} \frac{-d^2 + r^2}{d^2 + r^2} + \tilde{b}. \tag{16}$$

In the case of $\tilde{n}_{N+1} = -\tilde{b}$, Eq. (16) is given as follows.

$$\sum_{i=1}^{N} \tilde{n}_i x_i + d\tilde{b} = 0 \tag{17}$$

The above equation expresses an affine hyperplane in V. In the case of $\tilde{n}_{N+1} \neq -\tilde{b}$, Eq. (16) is arranged as follows.

$$\sum_{i=1}^{N}\left(x_i + \frac{d\tilde{n}_i}{\tilde{n}_{N+1} + \tilde{b}}\right)^2 = \frac{d^2}{(\tilde{n}_{N+1} + \tilde{b})^2}\left(\sum_{i=1}^{N+1} \tilde{n}_i^2 - \tilde{b}^2\right). \tag{18}$$

The above equation expresses a hypersphere in V. In particular, when $\vec{\tilde{n}} = (0, \cdots, 0, 1)$, the common set is the equator of S. The preimage of the equator is a hypersphere in V whose radius is d. From the above discussion, we can conclude that the preimages of the common sets of S and hyperplanes in $\tilde{V}$ are hyperplanes or hyperspheres in V. We show the relation in **Fig. 3** from a geometric viewpoint.

Since the degree of freedom of a hyperplane in $\tilde{V}$ is equal to that of a hypersphere or a hyperplane in V, there is a one-to-one correspondence between hyperspheres/hyperplanes in V and hyperplanes in $\tilde{V}$ whose common sets with S are not empty. Furthermore, the two regions in S separated by a hyperplane in $\tilde{V}$ coincide with the two regions in V separated by the corresponding hypersphere/hyperplane because f and its inverse function are

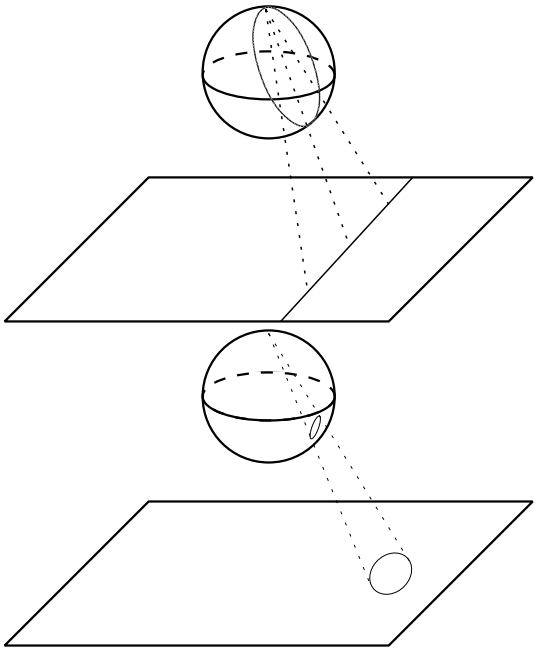

Fig. 3 Common sets of S and hyperplanes and their preimages. A hyperplane crossing the north pole (upper), a hyperplane that does not cross the north pole (lower).

continuous functions.

In particular, the hyperplanes appearing in Eq. (13) are linear hyperplanes in $\tilde{V}$. In such a case, any $N+1$ hyperplanes have at least one intersection point on S. Hence, the corresponding hyperspheres must have at least one intersection point in V. This property implies that any regions determined by a bit vector are connected.

4. Resolution of Angles

We discuss the resolution of the angles measured by the Hamming distances between the bit vectors generated by ISPH. Consider two feature vectors and the corresponding bit vectors. Let $\hat{\hat{X}}_i$ be a random variable that is 1 when the i-th bit of the two bit vectors is different, otherwise 0. We consider the following function J.

$$J(\hat{\hat{X}}_1, \cdots, \hat{\hat{X}}_1) := 1 - \cos(\frac{\pi}{B} \sum_{i=1}^{B} \hat{\hat{X}}_i) \tag{19}$$

Function J satisfies the following inequality.

$$\sup_{x_1, \cdots, x_B, x_i'} \left| J(x_1, \cdots, x_B) - J(x_1, \cdots, x_i', \cdots, x_B) \right| \leq \frac{1}{B} \tag{20}$$

By using McDiarmid's inequality,

$$P\left(\left| J(\hat{\hat{X}}_1, \cdots, \hat{\hat{X}}_B) - E[J(\hat{\hat{X}}_1, \cdots, \hat{\hat{X}}_B)] \right| \geq \epsilon \right) \leq 2\exp(-2B\epsilon^2). \tag{21}$$

Thus, with a probability of at least $1-\delta$, we have

$$\left| J(\hat{\hat{X}}_1, \cdots, \hat{\hat{X}}_B) - E[J(\hat{\hat{X}}_1, \cdots, \hat{\hat{X}}_B)] \right| \geq \left(\frac{1}{2B} \ln\left(\frac{2}{\delta}\right) \right)^{1/2} \tag{22}$$

The right hand side of the above inequality can be seen as a resolution of $(1-\cos(\theta_{xy}))$ measured by the Hamming distances. Hence, if B is large enough, one can estimate $(1-\cos(\theta_{xy}))$ accurately. In the context of the similarity search, we can identify $1-\cos(\theta)$ and θ because $1-\cos(\theta)$ is a monotonically increasing function of θ. Therefore, the resolution of angles measured by the Hamming distance becomes high as B becomes large.

4.1 Dependence of $F(r,d)$ on Feature Vectors and Optimal Parameter d

As we discussed in Section 3.1, if the value of factor $F(r,d)$ does not change greatly for each feature vector, the approximate relation between the Euclidean distances in V and the Hamming distances has good accuracy of approximation. The dependence of $F(r,d)$ on the feature vectors comes from the value r. Since r appears as r/d in $F(r,d)$, it is expected that the variation of $F(r,d)$ is reduced by a certain value d.

One of the most naive parameter regions to reduce the variation of $F(r,d)$ is $d \gg 1$. In such a case, the following relation holds.

$$(d_{Euc}(\vec{x},\vec{y}))^2 \approx d^2 \frac{1}{2}\left(1-\cos(\theta_{xy})\right) \text{ for } d \gg 1. \tag{23}$$

However, in such a large d, almost all of the feature vectors are mapped to the neighborhood of the south pole of S. We illustrate such a situation in **Fig. 4**. In that case, the angles between the projected feature vectors are small. If one can estimate the angles

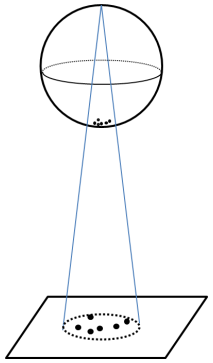

Fig. 4 Distribution of the projected feature vectors when $d \gg 1$.

in $\tilde{V}$ at high resolution, the accuracy of the approximation of the similarity search will be high for large d. However, as we showed by Eq. (22), in order to estimate the angles at high resolution, we need to use long bit vectors. Hence, the parameter region $d \gg 1$ is not practical.

We focused on the phenomenon called concentration on the sphere to reduce the variation of $F(r,d)$. We first explain the phenomenon and discuss its usage. When the dimensionality of a feature vector space is sufficiently high, almost all of the feature vectors will be located on a sphere whose center is the mean vector [26], [27], [28], [29]. We consider the distribution of the distance r of the feature vectors from the mean vector. Let us denote p-percentile of the distribution as $r^{(p)}$. If the concentration on a sphere occurs, almost all of the distances of the feature vectors from the mean vector will be $r_* := r^{(50)}$. We define the following parameter rrDiff in order to measure the variation of r from r_*.

$$\text{rrDiff} := (r^{(90)} - r^{(10)})/r_* \tag{24}$$

We used the percentile to reduce the effect of the outlier. If rrDiff is small enough, we can say that almost all of the feature vectors are on the sphere.

This phenomenon can be observed for many datasets. We used four real datasets, SHIFT-1M, GIST-1M [18], MNIST [30], and LabelMe [31], and the following two artificially made datasets.

Uniform Data were sampled from the uniform distribution whose domain is the interior of a unit sphere whose center is the origin in N-dimensional space. The number of feature vectors was 10,000.

Gauss Data were sampled from the N-dimensional multivariate standard normal distribution. Here also, there were 10,000 feature vectors.

The dimensionalities of the artificial dataset were 32, 64, 128, 256, and 512. We show rrDiff in **Fig. 5**. From the figure, we can say that rrDiff < 1 in high-dimensional spaces.

We now discuss the usage of the concentration on the sphere. The simplest situation is that all of data have a constant radius $r = r_c$. In such cases, rrDiff $= 0$ and $r_* = r_c$, and $F(r,d)$ is also a constant. Eq. (10) is given as follows.

$$(d_{Euc}(\vec{x},\vec{y}))^2/d^2 = \left(1+\frac{r_*^2}{d^2}\right)^2 \frac{1}{2}\left(1-\cos(\theta_{xy})\right) \tag{25}$$

We discuss the optimal value of d in this situation. Let $S(R)$ be a sphere in V whose center is the origin of V and whose radius

Table 1 Quantities of datasets.

Parameter \ Dataset	SHIFT1M	GIST1M	MNIST	LabelMe	Gauss	Uniform
Number of data for records	10,000	10,000	5,000	5,500	10,000	10,000
Number of data for queries	1,000	1,000	5,000	5,500	1,000	1,000
Dimensionality	128	960	784	511	512	512
k	100	100	50	55	100	100

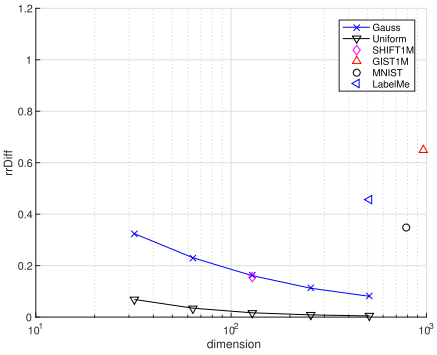

Fig. 5 rrDiff for various datasets.

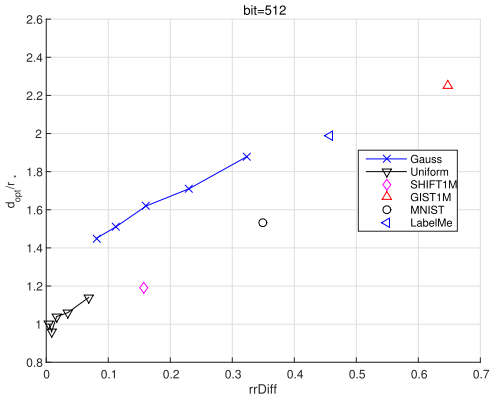

Fig. 6 rrDiff and d_{opt}/r_*.

is R, i.e. $S(R) := \{\vec{x} \in V | \|\vec{x}\| = R\}$. Because of $r = r_* = r_c$, all of the feature vectors are located on $S(r_*)$. If we set $d = r_*$, $S(r_*)$ is mapped to the equation of S by f^{-1} as we mentioned in Section 3.3. The equator is a great circle of S and the biggest image of $S(r_*)$ in various parameter ds. Hence, $d = r_*$ is the optimal. Although rrDiff is not zero for actual datasets because rrDiff is less than one, it is expected that the deviation of the optimal value of d from r_* is a function of rrDiff.

Let us consider the region where $d \leq r_*$. In this case, more than half of the feature vectors are mapped to the northern hemisphere of S. Even if there is a pair of feature vectors that are distant from each other and from the origin of V, they are mapped to S and have a small angle. Therefore, the accuracy of the approximation of the similarity searches deteriorates. Thus, the optimal value of d should be greater than r_*. As we have discussed, the region where $d \gg r_*$ is not practical.

From the above discussion, it is expected that the optimal value of d will be finite and greater than r_*, and is controlled by the concentration on the sphere. Hence, it is expected that the optimal value of d is a function of rrDiff because rrDiff is a measure of how data are concentrated on a sphere. Furthermore, it is expected that the optimal value of d will become large as the bit vector becomes long because the resolution of the angles is high for long bit vectors.

4.2 Proposed Value of d

We show by experiments that the discussion in Section 4.1 is correct. Moreover, we propose a value of d that does not depend on the details of the distribution of the feature vectors but only depends on the concentration on the sphere.

We used the four datasets and two artificial datasets mentioned in Section 4.1. We set the dimensionality of the artificial datasets as 512. From each dataset, we created two kinds of datasets by sampling from the original dataset a dataset for querying and a dataset for records, e.g., the feature vectors to be searched. The datasets are summarized in **Table 1**.

We used the following method to calculate the accuracy of the approximation. Let $\vec{q}$ be a query vector. We first performed the k-nearest neighbor search whose dissimilarity is the Euclidean distance and obtains a set $A(\vec{q})$ of kNN of $\vec{q}$. The retrieved results were treated as the ground truth. Parameter k were set to 1% of the number of record data and are summarized in Table 1. Then we performed the k-nearest neighbor search whose dissimilarity was the Hamming distance between bit vectors generated with ISPH. Let $B(\vec{q}, d)$ be the retrieved results with the parameter d. We calculated precision@k(d), which is a measure of the accuracy of approximation:

$$\text{precision@k}(d) := \#(A(\vec{q}) \cap B(\vec{q}, d))/k \tag{26}$$

We also performed the k-nearest neighbor search with bit vectors generated with the random projection method and calculated precision@k for comparison. The length of the bit vectors was 512. The experiment was performed five times for each value of d because the normal vectors were generated randomly. We plot the mean of precision@k in **Fig. 8**. RP represents the precision@k with the random projection method. From the figure, one can see that the accuracy of the approximation depends on the value of d.

Let us denote the value of d where precision@k is the maximum as d_{opt}; $d_{\text{opt}} := \arg\max_d(\text{precision@k}(d))$. We sought d_{opt} numerically by experiments. From a dimensional analysis, d and d_{opt} have the same unit as r. Hence, in order to compare d_{opt} for various datasets, we need to normalize it dividing by a parameter having the unit of r. We chose r_* for the parameter. As we discussed in Section 4.1, it is expected that d_{opt} is a function of rrDiff. Hence, we show the scatter plots of rrDiff and d_{opt}/r_* for various datasets in **Fig. 6**. The length of the bit vector was 512. From the figure, one can see that d_{opt} is greater than r_*, Furthermore, we found that rrDiff and d_{opt}/r_* have a linear correlation. This discovery is our second contribution.

Figure 7 shows the same scatter plot for the four real datasets when the lengths of the bit vectors are changed. The lengths of

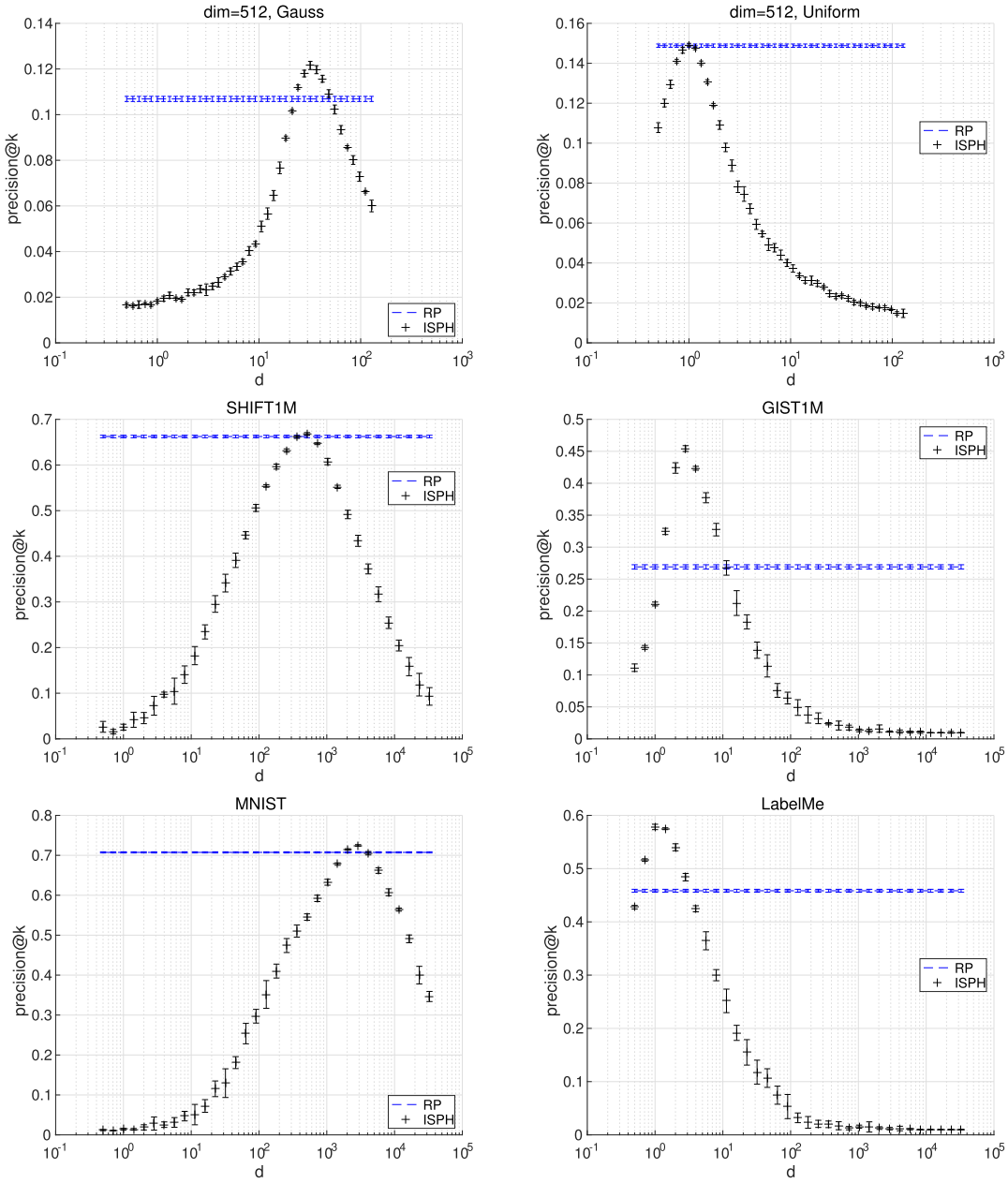

Fig. 8 Precision@k for various datasets.

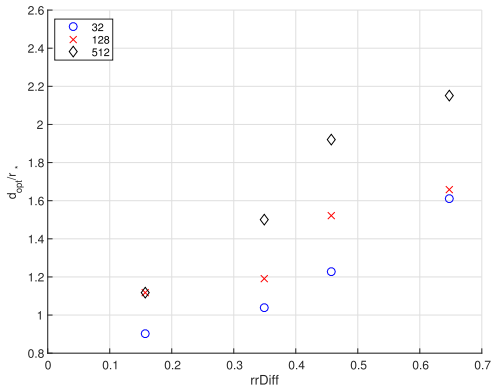

Fig. 7 Dependence of rrDiff and d_{opt}/r_* on bit vector length.

the bit vectors were 32, 128, and 512. From the figure, one can see that d_{opt}/r_* becomes large as the length of the bit vectors becomes long, and the growth rates depend on rrDiff.

From the above experiments, we propose the empirically optimal value of d as follows.

$$d_{prop} = r_* + (-1.0 + 0.374 * \log_2(B)) * (r^{(90)} - r^{(10)}) \qquad (27)$$

5. Experiments

5.1 Experimental Procedure

We examine the performance of the proposed method from the following three viewpoints: the processing time for learning of the hash function, the similarity search processing time, and the accuracy of the approximation.

The datasets used in this section are the ones mentioned in Section 4.2. Unless otherwise stated, parameter d is the value of Eq. (27).

We compared our proposed method with the following state-of-the-art methods: Random Projection (RP) [21], Angle Ap-

Table 2 Processing time for learning (msec).

Algorithm \ Data set	SHIFT1M	GIST1M	MNIST	LabelMe
ISPH	4.36	33.22	27.14	18.72
RP	2.80	20.90	17.16	11.23
AA	2.80	20.90	17.16	11.23
BOLSH	7.95	29.32	24.18	16.06

proximation for Euclidean Distance (AA) [16], and Batch-Orthogonal Locality-Sensitive Hashing (BOLSH) [22]. For BOLSH, we set the number of hyperplanes in a group as 32.

The experimental environment was as follows. The CPU was an Intel Xeon X5680 3.3 GHz, and the size of the main memory was 72.0 GB. The OS was Windows Server 2008 R2. Each method was implemented using C++, each process was a single thread, and the linear algebra library LINPACK was used. This experimental environment is the same through this section.

5.2 Processing Time for Learning

To determine parameter d, we need to calculate the percentiles $r^{(10)}$, $r^{(50)}$, and $r^{(90)}$. We regarded this calculation as a learning process of ISPH.

We measured the processing time for learning for each method. The length of the bit-vectors was 512. The data used for learning were sampled from the original dataset. The number of data used for learning was 10,000. The learning was done 100 times. **Table 2** summarizes the average processing time for learning. The processing time for learning for ISPH was less than twice the time for RP for all of the datasets.

5.3 Processing Time for Similarity Search

The hash function for ISPH defined in Eq. (13) can be given for fast calculation.

$$\tilde{h}^{(m)}(\vec{x}) = \mathrm{thr}\left(-\left(\frac{d\tilde{n}_{N+1}}{2}\right) + \left(\frac{\tilde{n}_{N+1}}{2d}\right) r^2 + \sum_{i=1}^{N} \tilde{n}_i x_i\right)$$

Since the first term and the coefficient of the second term do not depend on the feature vectors, they are calculated before querying. Although r^2 in the second term depends on the feature vectors, it does not depend on the hyperplanes and is calculated once at querying. The third term depends on the feature vectors and hyperplanes. However, it is an inner product of vectors, so we can use the linear algebra libraries.

Figure 9 plots the processing time for searching when changing the length of the bit vectors. Since the processing time for searching does not change greatly on the datasets and is proportional to the number of record data, we show the processing time for searching only for the 512-dimensional Gauss dataset. The values in Fig. 9 are the processing time per query. The processing times for searching for LHP, BOLSH, and ISPH are almost the same. In contrast, the AA processing time for searching is about twice that of LHP.

5.4 Accuracy of Approximation

We calculated precision@k to measure the accuracy of the approximation and show the precision@k as a function of processing time. These treatments were used in Ref. [19]. The procedure

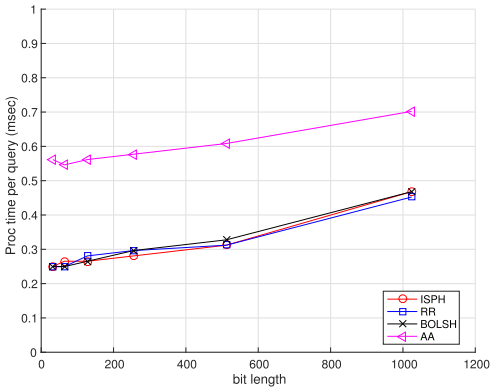

Fig. 9 Similarity search processing time.

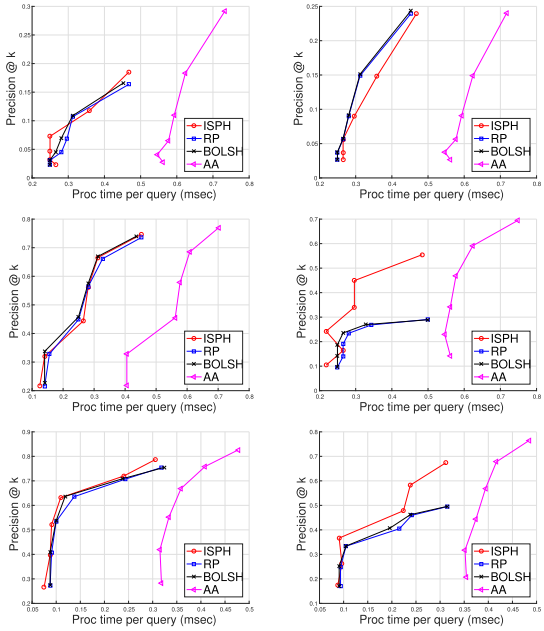

Fig. 10 Precision@k as a function of processing time for various datasets when changing the length of the bit vector. Each graph shows the results for various datasets: Gauss (upper left), Uniform (upper right), SHIFT1M (middle left), GIST1M (middle right), MNIST (lower left), LabeMe (lower right).

and parameters, for example k, are the same as the one described in Section 4.2. **Figure 10** shows the precision@k as a function of processing time for various datasets. The length of the bit vectors run from 32 to 1,024.

From Fig. 10, one can see that ISPH is the most accurate and high-speed for almost all datasets. The exception is for the Uniform dataset. This can be explained as follows. With the Uniform dataset, rrDiff is almost zero, as shown in Fig. 5, and almost all of the feature vectors are located on a sphere. Hence, the angles in V and the Euclidean distances in V are almost equal. Therefore, the precision@k of ISPH and AA are almost the same as that of RP, and ISPH is slightly slower than RP because the calculation

of the inverse stereographic projection requires more operations than RP. We cannot observe a difference in the precision@k values of BOLSH and RP. The reason for this is because BOLSH does not work well for high-dimensional feature spaces.

Even though 1,024 length bit vectors are long compared with the one used in other researches [17], such bit vectors are used in [14] and are shorter than the bit length of the original feature vectors. To store the original feature vector in memory, one needs 32 or 64 bits times the dimensionality of the feature vector. It is 4,096 bits for SHIFT1M to 61,440 bits for GIST1M, and is longer than 1,024. The processing times for the original similarity search per query are 6.64 msec for 512-dimensional Gauss, 6.77 msec for 512-dimensional Uniform, 1.99 msec for SHIFT1M, 24.74 msec for GIST1M, 6.14 msec for MNIST, and 4.18 msec for LabelMe. These processing times are 4-50 times longer than the approximate similarity searches with 1,024 length bit vector.

6. Conclusion

We have argued that the Euclidean distances in the feature vector space can be related to the angles in the higher-dimensional space by using inverse-stereographic projection. Moreover, we argued that the Euclidean distances in the feature vector space can be approximated by the Hamming distances between the bit vectors generated with the random projection method applied in the higher-dimensional space. To the best of our knowledge, this is the first work to give a direct approximate relation between the two distances. The relation is required for real system in particular for enterprise systems. The proposed method has better accuracy of approximation of the similarity search than the random projection method in the feature space and can be processed faster than the angle approximation method. We also discovered the relation between the optimal value of d and concentration on spheres. This discovery leads us to propose the value of d. With this proposal, the developers for real system are not required to tune the parameter. Such a feature is preferred for real system in particular for enterprise systems. Furthermore, we believe that the accuracy of the approximation of the proposed method will be improved by adjusting the normal vectors of the hyperplanes in the higher-dimensional space to the distribution of the feature vectors by using methods denoted in Refs. [13], [14], [17].

References

[1] Arya, S., Mount, D.M., Netanyahu, N.S., Silverman, R. and Wu, A.Y.: An optimal algorithm for approximate nearest neighbor searching fixed dimensions, *J. ACM*, Vol.45, No.6, pp.891–923 (1998).

[2] Guttman, A.: R-trees: A Dynamic Index Structure for Spatial Searching, *Proc. 1984 ACM SIGMOD International Conference on Management of Data*, *SIGMOD '84*, pp.47–57, ACM (1984).

[3] Novak, D. and Batko, M.: Metric Index: An Efficient and Scalable Solution for Similarity Search, *Proc. 2009 2nd International Workshop on Similarity Search and Applications*, *SISAP '09*, Washington, DC, USA, pp.65–73, IEEE Computer Society (online), DOI: 10.1109/SISAP.2009.26 (2009).

[4] Almeida, J., Torres, R.d.S. and Leite, N.J.: BP-tree: An Efficient Index for Similarity Search in High-dimensional Metric Spaces, *Proc. 19th ACM International Conference on Information and Knowledge Management*, *CIKM '10*, New York, NY, USA, pp.1365–1368, ACM (online), DOI: 10.1145/1871437.1871622 (2010).

[5] Weber, R., Schek, H.-J. and Blott, S.: A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces, *Proc. 24rd International Conference on Very Large Data Bases*, *VLDB '98*, pp.194–205, Morgan Kaufmann Publishers Inc. (1998).

[6] Indyk, P. and Motwani, R.: Approximate nearest neighbors: towards removing the curse of dimensionality, *Proc. 30th Annual ACM Symposium on Theory of Computing*, *STOC '98*, pp.604–613, ACM (1998).

[7] Datar, M., Immorlica, N., Indyk, P. and Mirrokni, V.S.: Locality-sensitive Hashing Scheme Based on P-stable Distributions, *Proc. 20th Annual Symposium on Computational Geometry*, *SCG '04*, pp.253–262, ACM (2004).

[8] Andoni, A. and Indyk, P.: Near-optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions, *Comm. ACM*, Vol.51, No.1, pp.117–122 (2008).

[9] Andoni, A., Indyk, P., Nguyen, H.L. and Razenshteyn, I.: Beyond Locality-Sensitive Hashing, *Proc. 25th Annual ACM-SIAM Symposium on Discrete Algorithms*, *SODA 2014*, pp.1018–1028 (2014).

[10] Andoni, A. and Razenshteyn, I.: Optimal Data-Dependent Hashing for Approximate Near Neighbors, *Proc. 47th Annual ACM on Symposium on Theory of Computing*, *STOC 2015*, pp.793–801 (2015).

[11] Weiss, Y., Torralba, A. and Fergus, R.: Spectral Hashing, *NIPS*, pp.1753–1760 (2008).

[12] Wang, J., Kumar, S. and Chang, S.-F.: Semi-supervised hashing for scalable image retrieval, *CVPR*, pp.3424–3431, IEEE (2010).

[13] Norouzi, M. and Fleet, D.J.: Minimal Loss Hashing for Compact Binary Codes, *ICML*, Getoor, L. and Scheffer, T. (Eds.), pp.353–360, Omnipress (2011).

[14] Noma, Y. and Konoshima, M.: Markov Chain Monte Carlo for Arrangement of Hyperplanes in Locality-Sensitive Hashing, *Journal of Information Processing*, Vol.22, No.1, pp.44–55 (2014).

[15] Matsumura, H., Sugimura, M., Yamasaki, H., Tomita, Y., Baba, T. and Watanabe, Y.: An FPGA-accelerated partial duplicate image retrieval engine for a document search system, *WACV 2016* (2016).

[16] Marukatat, S. and Methasate, I.: Fast nearest neighbor retrieval using randomized binary codes and approximate Euclidean distance, *Pattern Recognition Letters*, Vol.34, No.9, pp.1101–1107 (2013).

[17] Wang, J., Zhang, T., Song, J., Sebe, N. and Shen, H.T.: A Survey on Learning to Hash, arXiv:1606.00185 (2016).

[18] Jegou, H., Douze, M. and Schmid, C.: Product Quantization for Nearest Neighbor Search, *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol.33, No.1, pp.117–128 (online), DOI: 10.1109/TPAMI.2010.57 (2011).

[19] Iwamura, M., Sato, T. and Kise, K.: What is the Most Efficient Way to Select Nearest Neighbor Candidates for Fast Approximate Nearest Neighbor Search?, *The IEEE International Conference on Computer Vision* (*ICCV*) (2013).

[20] Matsui, Y., Yamasaki, T. and Aizawa, K.: PQTable: Fast Exact Asymmetric Distance Neighbor Search for Product Quantization Using Hash Tables, *The IEEE International Conference on Computer Vision* (*ICCV*) (2015).

[21] Charikar, M.S.: Similarity estimation techniques from rounding algorithms, *Proc. 34th Annual ACM Symposium on Theory of Computing*, *STOC '02*, pp.380–388, ACM (2002).

[22] Ji, J., Yan, S., Li, J., Gao, G., Tian, Q. and Zhang, B.: Batch-Orthogonal Locality-Sensitive Hashing for Angular Similarity, *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol.36 (2014).

[23] Heo, J.-P., Lee, Y., He, J., Chang, S.-F. and Yoon, S.-E.: Spherical hashing, *CVPR*, pp.2957–2964 (2012).

[24] Terasawa, K. and Tanaka, Y.: Spherical LSH for Approximate Nearest Neighbor Search on Unit Hypersphere, *WADS*, Lecture Notes in Computer Science, Vol.4619, pp.27–38 (2007).

[25] Kulis, B. and Grauman, K.: Kernelized locality-sensitive hashing for scalable image search, *IEEE International Conference on Computer Vision ICCV* (2009).

[26] Beyer, K.S., Goldstein, J., Ramakrishnan, R. and Shaft, U.: When Is "Nearest Neighbor" Meaningful?, *Proc. 7th International Conference on Database Theory*, *ICDT '99*, pp.217–235, Springer-Verlag (1999).

[27] Hall, P., Marron, J.S. and Neeman, A.: Geometric representation of high dimension, low sample size data, *J.R. Statist. Soc. B*, Vol.67 (2005).

[28] Ahn, J., Marron, J.S., Muller, K.M. and Chi, Y.-Y.: The high-dimension, low-sample-size geometric representation holds under mild conditions, *Biometrika*, Vol.94 (2007).

[29] Aoshima, M. and Yata, K.: Two-Stage Procedures for High-Dimensional Data, *Sequential Analysis*, Vol.30 (2011).

[30] THE MNIST DATABASE of Handwritten digits, available from ⟨http://yann.lecun.com/exdb/mnist/⟩.

[31] LabelMe: The open annotation tool, available from ⟨http://labelme.csail.mit.edu/⟩.

Yui Noma received a B.S. in physics from Kyoto University, Kyoto, Japan in 2003. He received an M.S. and Ph.D. in physics from Osaka University, Osaka, Japan in 2005 and 2008, where his interest was elementary particle physics. His current interest is data mining and information theory.