Journal of Information Processing Vol.22 No.1 44–55 (Jan. 2014)

[DOI: 10.2197/ipsjjip.22.44]

Regular Paper

Markov Chain Monte Carlo for Arrangement of Hyperplanes in Locality-Sensitive Hashing

Yui Noma[1,a)] Makiko Konoshima[1,b)]

Received: March 18, 2013, Accepted: October 9, 2013

Abstract: Since Hamming distances can be calculated by bitwise computations, they can be calculated with a lighter computational load than L2 distances. Similarity searches can therefore be performed faster in Hamming distance space. On the other hand, the arrangement of hyperplanes induces a transformation from the feature vectors into feature bit strings, which are elements of the Hamming distance space. This transformation is a type of locality-sensitive hashing that has been attracting attention as a way of performing approximate similarity searches at high speed. Supervised learning of hyperplane arrangements enables us to devise a method that transforms the higher-dimensional feature vectors into feature bit strings that reflect the information about the labels applied to feature vectors. In this paper, we propose a supervised learning method for hyperplane arrangements in feature space that uses a Markov chain Monte Carlo (MCMC) method. We consider the probability density functions used during learning and evaluate their performance. We also consider the sampling method for data pairs needed in learning and evaluate its performance. The performance evaluations indicate that the accuracy of this learning method, when using a suitable probability density function and sampling method, is greater than those of existing learning methods.

Keywords: higher dimensional feature vector, locality-sensitive hashing, arrangement of hyperplanes, similarity search, Markov chain Monte Carlo, low-temperature limit

1. Introduction

Unstructured data such as audio and images include complex content. This makes it difficult to search unstructured data directly. A commonly used approach has been to perform searches based on feature vectors extracted from unstructured data. To reflect the complexity of the data, these feature vectors generally consist of higher-dimensional data with hundreds or even thousands of dimensions.

There are a wide range of applications for high-speed similarity searches using higher-dimensional feature quantities extracted from unstructured data. Examples include authentication of people by fingerprint recognition, speech recognition in call centers, management of products and components based on CAD data, and detecting abnormal situations from surveillance video. For these applications, there are two things that are very important. One is a high-speed similarity search method. The other is a data structure that permits high-speed similarity searches, together with a method for extracting feature quantities that reflect the properties of unstructured data.

To perform a similarity search, the feature space should be a metric space. In most cases, the feature space is treated as an L2 metric space. Studies [1], [2], have devised an index structure aimed at performing similarity searches at high speed. However, in higher-dimensional space, the so-called "curse of dimensionality" means that all distances between data items are of a similar value. Consequently, searches in higher-dimensional data using these methods end up having processing times that are similar to those of searches performed without using a special index [3].

Hamming distances can be calculated by bitwise operations, which means that similarity searches are fast in Hamming metric space without using a specific index structure. In locality-sensitive hashing [4], the feature vectors are transformed into hash values. For this transformation, methods that involve the use of hyperplanes in feature space have been intensively studied [5], [6], [7], [8]. These methods use multiple hyperplanes as a means of partitioning the feature space. A bit string, determined from the orientations of hyperplanes, is assigned to each partitioned region. Feature vectors extracted from the data are allocated in the same way as bit strings assigned to the regions that include the feature vectors. Similar feature vectors are included in neighboring regions, so that the bit strings allocated to these feature vectors are similar and are separated by small Hamming distances. In the following, we will use the term "hashing" to refer to the process of transforming higher-dimensional feature vectors into feature bit strings.

From the above discussion, we decided to use bit strings as feature quantities, since these are data structures that can be searched at high speed. Moreover, when the data has labels, supervised learning can be used to extract feature bit strings that reflect the

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

labeled information. Note however that a single item of unstructured data will not necessarily have just one label to reflect its complexity.

Studies aimed at increasing the precision of feature quantities associated with the hyperplane hashing method include the following references [9], [10], [11]. In the learning phase, the normal vectors of the hyperplanes are determined by making the Hamming distances smaller between data pairs with a common label and larger between data pairs that do not have any common labels. As the number of bits increases, the degree of freedom also increases so that greater precision becomes possible.

Based on this reasoning, we can draw the following conclusions regarding high-precision similarity searches of large quantities of unstructured data. High-speed similarity searches can be achieved by using bit strings in Hamming metric space as feature quantities. High precision can be achieved by using a large number of bits and performing supervised learning with labeled data as the training data.

In this paper, apart from the use of feature bit strings, no consideration is given to the processing time of the similarity search. Our main focus is on using supervised learning to improve the precision of feature bit strings.

The method proposed in this paper performs supervised learning using a Markov chain Monte Carlo (MCMC) method. The transformation of feature vectors into feature bit strings is a discontinuous vector-valued function. This makes it impossible to perform naïve learning based on gradients. Another approach involves introducing a loss function so that the transformation method can be approximated by a continuous function. However, the only loss functions found so far are strongly dependent on the properties of the data set. In our method, each normal vector is regarded as a particle on a unit sphere in feature space, and a random walk is performed on this unit sphere. In the random walk, a discontinuous function can be treated as an evaluation function. In this paper, we also describe sampling methods for training data pairs and evaluation functions for use in learning.

This paper is structured as follows. First, Section 2 describes the existing learning methods. Then, Section 3 describes our method. We discuss the evaluation functions needed during learning and the sampling methods for training data pairs. Section 4 describes the experiments we performed using various data sets. We also evaluate the evaluation functions and the sampling methods. Section 5 shows that our learning method outperforms existing methods. Finally, Section 6 summarizes our work and discusses the prospects of this approach.

2. Background and Related Work

In this section, we describe the use of hyperplanes for locality-sensitive hashing, which is the basis of the proposed technique. Then, we describe some of the related techniques.

2.1 Conventional Locality-sensitive Hashing with Hyperplanes

Here, we describe the hashing method using hyperplanes. A space V in which there are higher-dimensional feature quantities is regarded as an N-dimensional vector space. The configurations of multiple hyperplanes in V are referred to as hyperplane arrangements.

Consider B hyperplanes passing through the origin of V. Such hyperplanes are called linear hyperplanes. A linear hyperplane is identified by its normal vector. Since the length of the normal vector specifying a linear hyperplane is immaterial, these lengths are set to one. The configuration space of normal vectors corresponds to an $N-1$-dimensional hypersphere S^{N-1}. When distinguishing between B hyperplanes, the configuration space of the hyperplanes is $(S^{N-1})^B$. An N-dimensional feature vector $\vec{x}$ is transformed into a bit string by registering a 1 if its dot product with each normal vector is positive, and a zero otherwise. Therefore, the length of the bit string is equal to the number of hyperplanes B.

A hyperplane that does not pass through the origin is called an affine hyperplane. An affine hyperplane can easily be constructed from a linear hyperplane. In Ref. [11], an experiment is performed where the hashing of affine hyperplanes is achieved by learning the hashing of linear hyperplanes. When developing a new learning method for hyperplanes, it is easier to work with linear hyperplanes. In the following discussion, therefore, all hyperplanes are assumed to be linear.

When the data has labels, it is sometimes the case that the angles or L2 distances do not exhibit a suitable degree of dissimilarity. In such cases, the hyperplanes can be determined by supervised learning so that data pairs with a common label are separated by small Hamming distances and data pairs that do not have any common label are separated by large Hamming distances. In the following, we will refer to a data pair with a common label as a "positive pair" and a data pair without a common label as a "negative pair."

In one hashing method [5], the B hyperplanes are set randomly. In the following, this is referred to as the LSH method.

Other references such as Refs. [6], [7], [8], [9], [10] describe hashing methods that use hyperplanes. In particular, MLH [9] and S-LSH [10] are described in Sections 2.2 and 2.3.

2.2 Minimal Loss Hashing

Minimal Loss Hashing (MLH) [9] is a learning method whose aim is to minimize an empirical loss function on $(S^{N-1})^B$. However, the empirical loss function is discontinuous, so it is not possible to use learning methods based on gradients. As a result, the empirical loss is replaced with a differentiable upper bound function g, and gradients are used to minimize g instead. The point that gives the minimum value determines the coordinates of the B learned hyperplanes. The function g has several parameters that need to be adjusted. Some of these parameters depend on the data pairs used for training. All the data pairs for learning are chosen at random.

2.3 Locality-sensitive Hashing with Margin-based Feature Selection

In this subsection, we describe the concept of the learning method called locality-sensitive hashing with margin-based feature selection (S-LSH) [10]. In S-LSH, the normal vectors of hyperplanes are not used directly for learning. $\tilde{B}$ hyperplanes

($\tilde{B} > B$) are randomly provided. A degree of importance is allocated to each hyperplane, and these degrees of importance are calculated in the learning process. The degrees of importance are arranged in descending order, and the topmost B normal vectors are selected. The distance calculations during learning are of weighted Hamming distances. Two types of data pairs are used during learning. The learning data pairs are selected as follows. A feature vector a is randomly selected from the learning data. The first type of data pair consists of the pair (a, b), where b is the feature vector with the smallest weighted Hamming distance in the data set that has a common label as a. Later, we call such pairs NearHits (see Section 3.4.) The second type of data pair consists of the pair (a, c), where c is the feature vector with the smallest weighted Hamming distance in the data set that does not have any common label as a. Later, we call such pairs NearMisses (see Section 3.4.)

S-LSH has been shown to have good learning performance on many different data sets [10]. It is particularly effective in cases where there are many labels, and data with the same label have small cardinalities.

3. Proposed Method

3.1 Motivation

To make a high-speed similarity search that accurately represents the latent similarities of unstructured data, we perform learning with a greater number of bits B. The configuration space of an arrangement of B hyperplanes is $(S^{N-1})^B$. When there is an evaluation function U' on $(S^{N-1})^B$ that has the following property, learning the arrangement of hyperplanes can be regarded as an optimization problem that globally maximizes U'. The argument of U' is the arrangement of hyperplanes. Each hyperplane divides the feature space V into two regions. The value of U' increases with the number of positive pairs whose feature vectors are in the same region and negative pairs whose feature vectors are in different regions. In most cases, a function U' having this property is thought to have multiple local maxima. When B is large, the dimension of $(S^{N-1})^B$ increases and it becomes harder to solve the optimization problem. Since we are concerned here with hashing using hyperplanes, the values of U' can also be discrete.

Instead of solving an optimization problem in $(S^{N-1})^B$, we can consider a method where optimization problems in S^{N-1} are solved B times, and these solutions are bundled together. That is, instead of learning a set of B hyperplanes, the individual hyperplanes are separately learned and the results are bundled together. However, if we obtain B solutions to the optimization problem in S^{N-1}, the performance is severely impaired for the following reason. Consider an evaluation function U on S^{N-1}. Assume that the points S^{N-1} where the value of U is large correspond to a good hyperplane. That is, we assume that U has the following property. When the feature space V is partitioned into two regions by a single hyperplane, the value of U increases with the number of positive pairs whose feature vectors are in the same region and negative pairs whose feature vectors are in the different regions. An example of an evaluation function is shown in Section 3.3. We will assume that the evaluation function U has a global maximum value on $p_* \in S^{N-1}$. If all the hyperplanes exist in p_*, then they are all degenerate. In this case, the feature space V is only divided into two regions, and there are only two types of representative bit strings. Clearly, it would not be possible to capture the features of unstructured data with these bit strings. For this reason, when we consider bundling the learning results of individual hyperplanes, it can be said that individual hyperplanes are not necessarily learned by finding the point where the evaluation function U on S^{N-1} is globally maximized. Therefore, in the following, we find B points where the evaluation function U has a local maximum in S^{N-1} when learning is performed with individual hyperplanes. Multiple hyperplanes are learned by bundling these results. Here, we must ensure that the multiple hyperplanes are not oriented in the same direction.

In the remainder of this section, we describe the proposed method (called M-LSH), which is a hyperplane normal vector learning based on the Markov chain Monte Carlo (MCMC) method. We also discuss a number of evaluation functions (the choices of which have a strong influence on the performance of M-LSH learning) and data pair sampling methods.

3.2 Learning Hyperplanes with the Markov Chain Monte Carlo Method

Our method is a supervised learning of hyperplanes using MCMC. Its aim is to probabilistically determine the point where the evaluation function U reaches a local maximum value. Its advantage is that it does not require a differentiable evaluation function. Its disadvantage is that because it uses a Monte Carlo method, the point where the evaluation function is locally maximized cannot be determined with perfect accuracy. However, from the properties of MCMC, the learned results are highly likely to be close to the point where the evaluation function is locally maximized. The probability that a particle is in such a place is high so that the local maximum value is high, and the peak is sharp.

In the following, we will assume that the evaluation function U is positive and bounded. Also, we will assume that the details of the evaluation function depend on the training data pairs. Examples of U are given in Section 3.3.

Consider a particle on S^{N-1} whose position corresponds to the normal vector of a hyperplane. Since U is generally discontinuous and not differentiable, we cannot use optimization methods based on gradients, that is, continuous particle motions. Instead, we can try to find a minimum solution by using a random walk method. Because of the assumptions placed on U, we can regard U as the probability density function of S^{N-1} (except for a normalization constant) and use MCMC to evaluate the temporal evolution of particles. This is the essence of the M-LSH method.

M-LSH uses the Metropolis-Hastings algorithm as its MCMC method [12] and a normal distribution as its density function. In M-LSH, particles perform random walks a fixed number of times. We refer to this temporal evolution as a single batch process. Since the details of the evaluation function are determined by deciding on the training data pairs, the handling of the training data pairs may lead to incidental local maximum values of the evaluation function. To prevent the particles from becoming trapped at this sort of point, batch processing is performed a number of

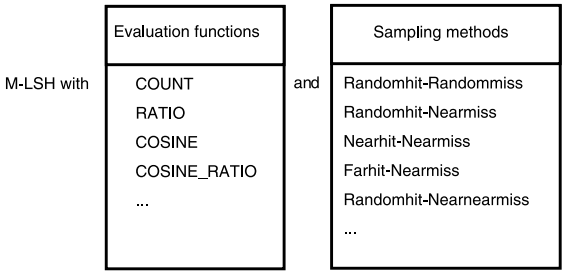

Fig. 1 M-LSH variants. The number of variants equals the number of combinations of evaluation functions and sampling methods.

times, and different learning data pairs are used in each batch process.

By performing learning with multiple hyperplanes, we obtain the points where the evaluation function is locally maximized. As described in Section 3.1, it is necessary to prevent points from being learned where multiple hyperplanes produce the same local maximum value. In M-LSH, this issue is resolved in the following way. MCMC exhibits a property whereby particles tend to accumulate at places where the probability density function is locally maximized. The sharper the peak is in the evaluation function close to the local maximum value, the more intense this trend becomes. In most cases, since U is multimodal, making its peaks sharper and randomly setting the initial positions of the particles will cause the particles to collect at peaks close to their initial positions. Therefore, we can prevent the particles from all moving towards the same point.

Many variants of M-LSH can be obtained by using different evaluation functions U and different sampling methods for training data pairs that determine the evaluation function U. **Figure 1** lists these combinations. These items are described below in Sections 3.3 and 3.4.

3.3 Evaluation Function

In M-LSH, the type of evaluation function must be determined. A number of possible function types are considered below. First, though, we will introduce some nomenclature. PP denotes the set of all given positive pairs, and NP denotes the set of all given negative pairs. The angles subtended by the two feature vectors of a pair p relative to the normal vector of a hyperplane are $\theta_1(p)$ and $\theta_2(p)$, respectively. The following subsets are defined.

$$PP_+ := \{p \in PP | \cos(\theta_1(p)) * \cos(\theta_2(p)) > 0\}, \quad (1)$$

$$NP_- := \{p \in NP | \cos(\theta_1(p)) * \cos(\theta_2(p)) < 0\}. \quad (2)$$

Note that the cardinality of a set A is denoted by $\#A$.

The following formulas assume an evaluation function $U = \exp(x/T)$ that uses an enumerated value x. Here, $T = 1$.

COUNT

$$x = \#PP_+ + \#NP_-. \quad (3)$$

RATIO

$$x = \frac{\#PP_+}{\#PP} + \frac{\#NP_-}{\#NP}. \quad (4)$$

COSINE

$$x = \sum_{p \in PP} |\cos(\theta_1(p)) + \cos(\theta_2(p))| + \sum_{p \in NP} |\cos(\theta_1(p)) - \cos(\theta_2(p))|. \quad (5)$$

COSINE_RATIO

$$x = \frac{1}{\#PP} \sum_{p \in PP} |\cos(\theta_1(p)) + \cos(\theta_2(p))| + \frac{1}{\#NP} \sum_{p \in NP} |\cos(\theta_1(p)) - \cos(\theta_2(p))|. \quad (6)$$

If we regard $-x$ and T as the particle's energy and temperature, respectively, U can be regarded as a Boltzmann weight in statistical physics. From this perspective, the low-temperature limit is where T is zero, and the high-temperature limit is where T is ∞. The energy, $-x$, of the function COUNT corresponds to the energy of Ising's model of (anti-)ferromagnetic matter. Positive pairs correspond to ferromagnetic matter. Negative pairs correspond to anti-ferromagnetic matter.

The function COSINE is a continuous version of COUNT up to a certain factor. When T is small, COSINE can be approximated by COUNT. The functions RATIO and COSINE_RATIO are scaled versions of COUNT and COSINE.

3.4 Sampling Method for Training Data

The evaluation function defined in Section 3.3 must include both PP and NP. PP and NP can be determined by considering all combinations of the training data. However, the number of such configurations is almost half the square of the number of training data. As the number of training data increases, the cardinalities of PP and NP become large and it takes longer to calculate the evaluation function. In this subsection, we consider a number of different selection methods for PP and NP and discuss their advantages and disadvantages. Here, we will use the term "distance" to refer to the L2 distance, unless otherwise noted. We will also use the following nomenclature. L is the set of all training data. L_a represents a data set having a common label as an element $a \in L$, and L_a^c represents the complement $L \setminus L_a$. The distance between two elements $a, b \in L$ is denoted by $\mathrm{dist}(a, b)$.

We will start by considering the following sampling methods for selecting NP.

RandomMiss

Select $a \in L$ randomly. Then select $b \in L_a^c$ randomly and form a negative pair (a, b).

NearMiss

Select $a \in L$ randomly. Then form a negative pair (a, b) such that $b := \arg\min_{c \in L_a^c}(\mathrm{dist}(a, c))$.

BoundaryMiss

Select $a \in L$ randomly. The form a negative pair (a', b) such that $b := \arg\min_{c \in L_a^c}(\mathrm{dist}(a, c))$ and $a' := \arg\min_{c \in L_a \cap L_b^c}(\mathrm{dist}(b, c))$.

RandomMiss is used in MLH [9], and NearMiss is used in S-LSH [10]. On the other hand, BoundaryMiss is a new sampling method, so we will describe it here in more detail. Consider two elements $a, b \in L$ that do not have any common label and $L_a \cap L_b \neq \emptyset$[*1]. Since the distributions of L_a and L_b overlap, it

[*1] Since the labels applied to unstructured data can be of more than one type, this sort of situation occurs frequently.

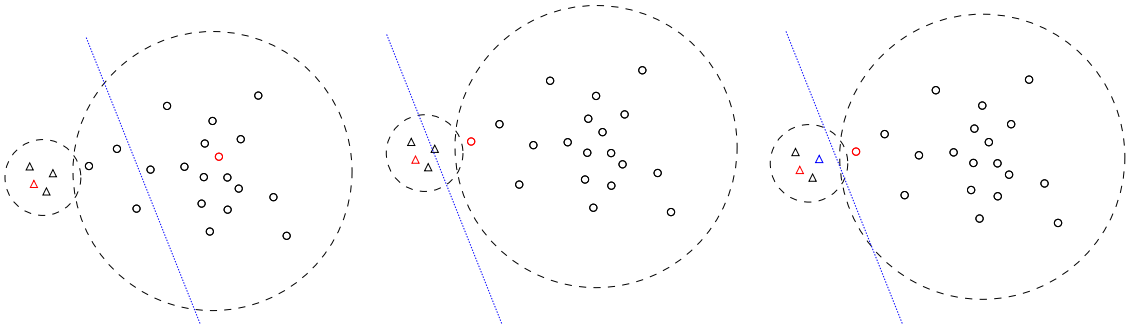

Fig. 2 Selection methods for negative pairs. From left to right RandomMiss, NearMiss, and BoundaryMiss. The red triangles, red circles, and blue triangles correspond to element a, element b, and element a', respectively. The blue dotted line represents the hyperplane expected from learning.

is not possible to obtain a hyperplane that separates them completely. If we are allowed to bisect $L_a \cap L_b$ with a hyperplane, then it may also be possible to separate the difference sets $L_a \setminus L_b$ and $L_b \setminus L_a$. To get a hyperplane that bisects $L_a \cap L_b$, we can form a negative pair by selecting one data item from $L_a \setminus L_b$ and $L_b \setminus L_a$. BoundaryMiss is one way in which negative pairs of this sort can be made. Furthermore, the points found by BoundaryMiss are expected to lie close to the boundary between $L_a \setminus L_b$ and $L_b \setminus L_a$ (**Fig. 2**.)

Figure 2 also shows typical data pairs obtained by RandomMiss sampling and the hyperplanes learned from them. The dotted circles in these figures show the approximate regions over which these sets are distributed. The figure shows that in this sort of sampling there is a strong possibility of selecting a pair comprising an element close to the center of gravity of L_a and an element close to the center of gravity of L_a^c. This makes it easier to learn a hyperplane that separates the center of gravity of L_a from the center of gravity of L_a^c. When L_a^c is distributed over a broader region than L_a, it is expected that the resulting hyperplane will deviate from the boundary of L_a and L_a^c. In particular, when the number of labels added to the training data is large and the sets of each label have similar cardinality, the distribution of L_a^c tends to become broader than that of L_a. As a result, the tendency for the learned hyperplanes to be away from the boundary is thought to become more pronounced as the number of labels increases.

In NearMiss sampling, an element selected from L_a^c lies close to the boundary of L_a and L_a^c, so it is possible to avoid the above drawback of the RandomMiss sampling method. However, when L_a is distributed over a wide region, there is a greater likelihood of $a \in L$ deviating from the boundary between L_a and L_a^c. Figure 2 shows some typical data pairs obtained by NearMiss sampling and the hyperplanes learned from these pairs.

BoundaryMiss sampling can compensate for the above-mentioned drawbacks of the NearMiss method, but it is likely to choose data pairs that are separated by smaller distances and is therefore more susceptible to noise in the data.

We will now describe the selection method for PP. For the reasons discussed below, it is better to consider positive pairs in terms of applying corrections to the discriminant planes used for discriminating negative pairs. For example, if we learn with only positive pairs, because the hyperplanes should not separate feature vectors in each positive pair, the hyperplanes should stay away from all the feature vectors. In such case, all the bit strings

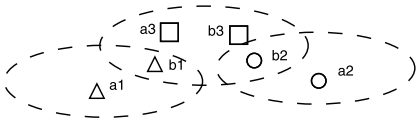

Fig. 3 Schematic illustration of overlapping data pairs.

of the training data will be identical, making it impossible to separate the feature vectors. We therefore consider that positive pairs help to prevent L_a from becoming separated by hyperplanes. We will consider the following sampling methods for positive pairs.

RandomHit

Select $a \in L$ randomly. Then randomly select $b \in L_a$ and form a positive pair (a, b).

NearHit

Select $a \in L$ randomly. Then form a positive pair (a, b) such that $b := \arg\min_{c \in L_a}(\mathrm{dist}(a, c))$.

FarHit

Select $a \in L$ randomly. Then form a positive pair (a, b) such that $b := \arg\max_{c \in L_a}(\mathrm{dist}(a, c))$.

RandomHit is used in MLH [9], and NearHit is used in S-LSH [10].

Let us consider a data set whose elements have a single label. In this case, FarHit sampling is vulnerable to outliers. Moreover, while RandomHit sampling is robust against outliers, NearHit sampling performs poorly because it is not possible to prevent data other than the selected data pair in L_a from being arranged in different directions of the hyperplane. The performance degradation is particularly severe when there are many data items with the same label.

Next, let us consider a data set whose elements have an arbitrary number of labels. Here, consider the case in which there are three positive pairs, $(a_1, b_1), (a_2, b_2), (a_3, b_3) \in PP$, such that $a_1 \notin L_{a_2} \wedge b_1 \in L_{b_2} \wedge a_3, b_3 \in L_{b_2}$. In particular, when a_3 and b_3 are close to b_1 and b_2 respectively, we shall refer to these data pairs as overlapping. This situation is summarized in **Fig. 3**. When learning is performed in this case, it becomes difficult to separate L_{a_1} and L_{a_2} from L_{b_2}. As the number of sampling pairs increases, while FarHit sampling and RandomHit sampling will make many overlapping data pairs, NearHit sampling will make less overlapping data pairs. Therefore, in this situation, while FarHit sampling and RandomHit sampling will worsen the performance, NearHit sampling will have less of a performance decrease.

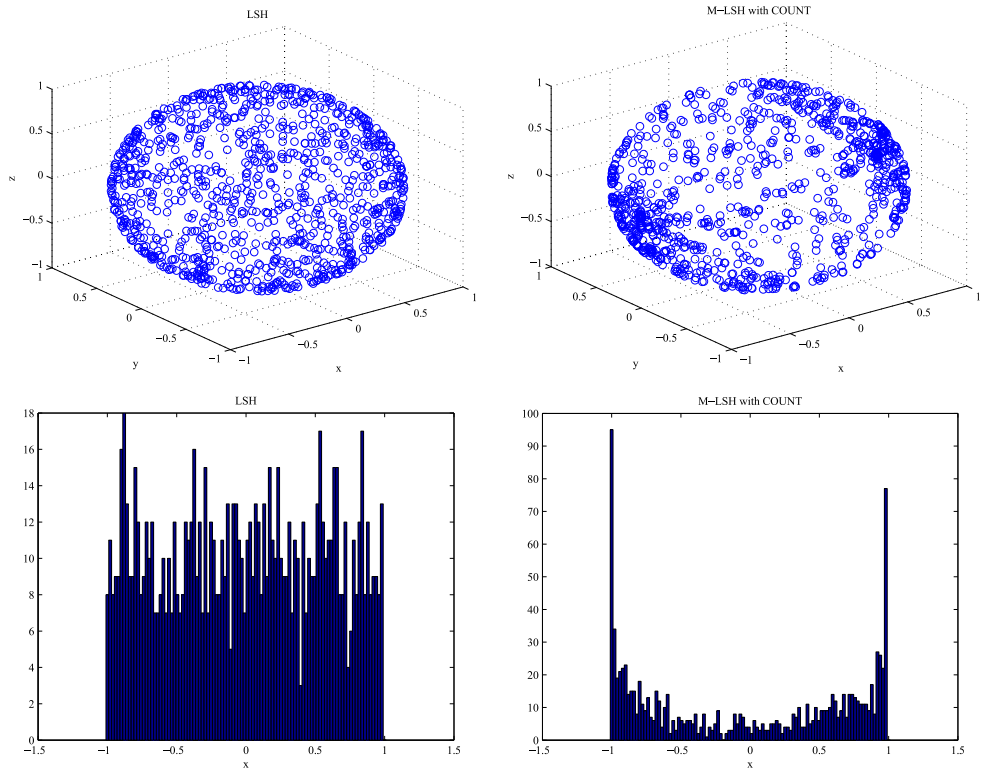

Fig. 4 Learning results of LSH and M-LSH in the experiment with an artificial data set. Top left: Scatter plot of normal vectors learned by LSH; Bottom left: Histogram of x components. Top right: Scatter plot of normal vectors learned by M-LSH; Bottom right: Histogram of x components.

Based on this reasoning, there are as many possible sampling methods as there are combinations of positive pair and negative pair sampling methods. The sampling methods that we evaluated are as follows[*2].

- RandomHit-RandomMiss
- RandomHit-NearMiss
- NearHit-NearMiss
- FarHit-NearMiss
- RandomHit-BoundaryMiss

4. Experiments and Evaluation

We performed experiments to measure the effect of the proposed method on a number of different data sets. We performed supervised learning on data that had already been labeled. The data labels were all known. The search results were obtained as follows. The Hamming distance between the query and data in the database was calculated, and ordered in ascending order of the distance. The high-ranked data were the search results. Thus, the performance depended on the number of data acquired by the search. To evaluate the performance, we used the precision rate and recall rate and F-measure as defined as follows: A is the number of data acquired by the search, B is the number of data items with a common label as the query for which the search results were obtained, and C is the number of data items in the database with a common label as the query. From this, we can define Precision and Recall as follows:

$$\text{Precision} := \frac{B}{A}, \tag{7}$$

$$\text{Recall} := \frac{B}{C}. \tag{8}$$

The precision-recall curve shows the variations in recall and precision as A changes. The F-measure is the harmonic mean of precision and recall.

In the experiments, by way of reference, we also calculated the precision and recall in similarity searches based on the L2 distance using the original feature vectors.

The remainder of this section is structured as follows. First, we illustrate the benefits of M-LSH on learning with an artificial data set that we prepared. Then, we assess the performance of the evaluation functions and sampling methods considered in Sections 3.3 and 3.4. For this performance evaluation, we used actual data sets instead of an artificial data set. Finally, we show how our method differs from existing learning methods.

4.1 Experiments with an Artificial Data Set

We assessed the effect of M-LSH on learning in an experiment using an artificial data set. This data set consisted of 300 data items sampled from a three-dimensional standard normal distribution. With the axes labeled x, y and z, we classified the data items into two classes according to whether the x component was positive or nonpositive. As can be seen from the way in which the data is labeled, we wanted a hyperplane whose normal vector $\vec{n}$ was $\vec{n} = (\pm 1, 0, 0)$.

Figure 4 shows the effect of LSH and M-LSH on learning with a bit string length of 1,024. The parameters of learning with M-LSH were as follows: number of processing batches: 5; number of temporal evolution steps in batch processing: 100; number of

[*2] We believe that our choices were natural ones.

Table 1 Experimental parameters.

Parameter \ Data set	MNIST	Fingerprint	Speech	LabelMe
Number of training data items	60,000	9,906	192,875	11,000
Number of data items for searching	5,000	12,138	192,683	5,500
Number of data items for queries	5,000	19,932	1,815	5,500
Dimension before dimensionality reduction	784	4096	200	512
Dimension after dimensionality reduction	149	276	30	20
Feature vector have unique labels	Yes	Yes	No	No
Approximate number of labels	10	1300	2000	300
Rough cardinality of the sets for each label	6000	7	100	40

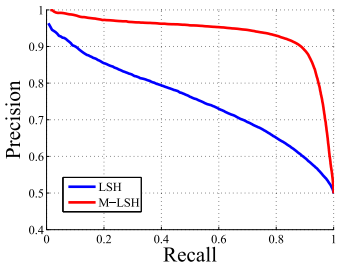

Fig. 5 Precision-recall curves of LSH and M-LSH learning on an artificial data set.

data pairs used for learning in each batch process: 2,000; number of evaluation functions used during learning: COUNT; sampling method: RandomHit-RandomMiss, with equal numbers of positive and negative pairs.

The scatter plot and x-component histogram of the normal vectors obtained by LSH in Fig. 4 reveal that the normal vectors are uniformly distributed on a two-dimensional sphere. In contrast, the scatter diagram and x-component histogram for M-LSH indicate that most of the normal vectors are distributed in the vicinity of $\vec{n} = (\pm 1, 0, 0)$. **Figure 5** shows the precision and recall of LSH and M-LSH. As expected from the distribution of normal vectors, this figure shows that M-LSH has a positive effect on learning.

4.2 Experimental Data

Here, we describe the experimental data used in the performance evaluations described in Sections 4.3 and 4.4.

The experimental data was obtained from the following sources.

- MNIST
  Scanned images of handwritten numerals 0–9 [13]. Each digit is stored as a 28×28-pixel 8-bit grayscale image and is labeled with the corresponding digit 0–9. We used the images themselves as feature quantities. Therefore, the feature quantities had 784 dimensions.
- Fingerprint images
  Fingerprint image data acquired using a fingerprint image scanner. The feature quantities consisted of 4,096-dimensional Fourier spectra of these fingerprint images. Since fingerprints are unique to each human, the data was labeled with the names of the corresponding individuals. In other words, each feature vector was given just one label. For details, see reference [10].
- Speech features
  A set of 200-dimensional mel frequency cepstral coefficient (MFCC) feature quantities extracted from a three-hour recording of a local government assembly published on the Internet [14]. The query was a spoken sound acquired separately. In supervised learning of speech, the contents of the speech are normally labeled with a text transcription. Instead, we treated the features with the top 0.1% shortest Euclidean distances from queries regarded to be in the same class. Each feature vector could have multiple labels.
- LabelMe
  LabelMe data using 512-dimensional Gist feature quantities [15] extracted from image data published in reference [16]. The labeling was applied to the dissimilarity matrices of distributed data, and the same labels were applied to data corresponding to the topmost 50 rows of data in each row. Each feature vector could have multiple labels.

The data sets were selected with the following applications in mind: handwritten number recognition (MNIST), biometric identification (fingerprint images), speech recognition (speech features), and automatic image classification (LabelMe).

The quality of data used in the experiments is summarized in **Table 1**. Here, we envisaged performing searches on a database with the data divided into three pairwise disjoint sets: one set for learning, another to be searched, and a third one for queries. The learning performance varied widely depending on the number of labels in the data set and on the cardinality of data sets having a common label. However, since the data sets were not all given unique labels, it was not possible to give a naïve definition of the cardinality of the sets for each label. We therefore reasoned as follows. For the data actually used for learning, the average value of the number of data items having a common label as one item of data can be regarded as the rough cardinality of the sets for each label. The rough number of labels can then be calculated by dividing the number of data items actually used for learning by the rough cardinality of the sets for each label.

Since data generally contains noise, noise reduction must be performed. Prior to the experiments, we subjected all the data to the following noise reduction processes. These processes are widely used as noise reduction methods. The feature quantities of the data are higher-dimensional data. Depending on the data set, each component of the data may be expressed in different units. Unless the feature vectors are made dimensionless, they cannot be used for calculationing distances or angles. We therefore subjected the data to an affine transformation so that the average value of each component of the feature vector of the learning data became 0 and the standard deviation of the learning data became 1. We also performed principal component analysis on the learning data. This was done by finding the subspace with a

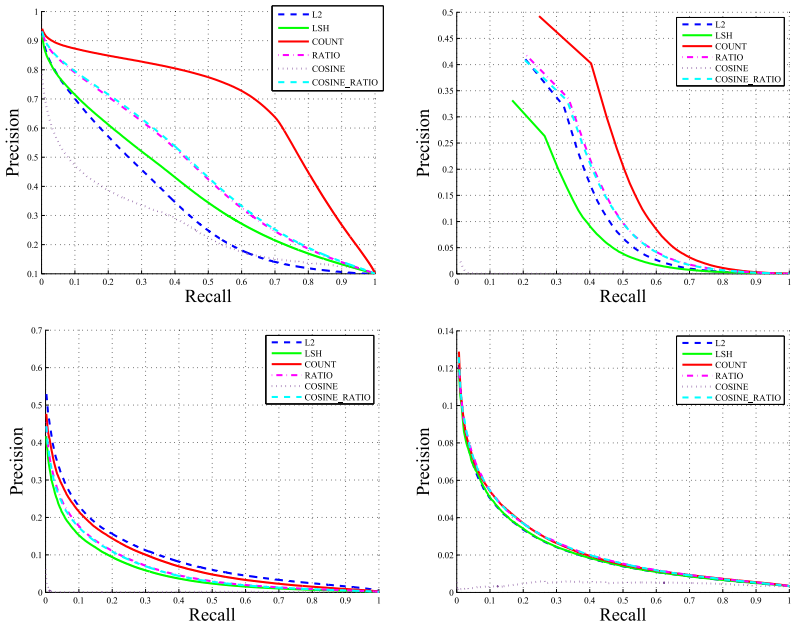

Fig. 6 Recall-Precision curves for each evaluation function for MNIST (upper left), fingerprint (upper right), speech (lower left), and LabelMe (lower right).

cumulative contribution rate of over 80%, and mapping all data to this space. After this noise reduction process, we performed learning and search tests. Since all the methods were evaluated using data that had been subjected to this noise reduction process, their performances were unaffected by it.

The parameters of the M-LSH experiments were as follows. Standard deviation of proposed density distribution: 0.01; number of processing batches: 10; number of bits: 1,024; number of temporal evolution steps in batch processing: 100; number of data pairs used for learning in each batch process: 2,000 or 20,000.

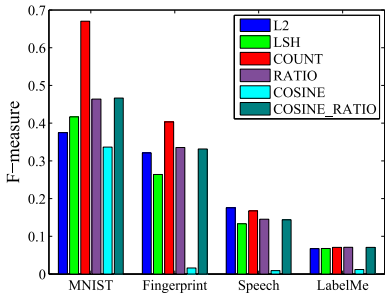

Fig. 7 Maxima of F-measures for each evaluation function for MNIST, fingerprint, speech, and LabelMe.

4.3 Performance of Evaluation Function

We evaluated the performance of the evaluation functions listed in Section 3.3. A natural choice of sampling method is the simplest one i.e., RandomHit-RandomMiss. However, as discussed in Section 4.4, RandomHit-RandomMiss sampling performs poorly. Therefore, we used the next simplest method, RandomHit-NearMiss sampling.

We used precision-recall curves and the maxima of F-measures to evaluate the performance. **Figure 6** shows the precision-recall curves for each data set. For comparison, the figure also shows curves of the search with LSH and the L2 distance. **Figure 7** shows the maxima of F-measures for each data set. The number of data pairs used in M-LSH was 2,000.

Figures 6 and 7 indicate that M-LSH performed about as well as LSH when it used RATIO or COSINE_RATIO. In contrast, it outperformed LSH on all data sets when it used COUNT, but performed worse than LSH on all data sets when it used COSINE.

The reason why M-LSH using RATIO or COSINE_RATIO performed poorly is thought to be as follows. The evaluation functions include a parameter T that is analogous to temperature. Since we used a fixed value of $T = 1$ in this evaluation, the exponents of the evaluation functions are confined to the range $[0, 2]$ or $[0, 4]$. In this range, a slight change in the normal vector will not cause a large change in the value of the evaluation function. Therefore, the normal vector moves approximately at random, and performance suffers as a result. In other words, in this evaluation function it can be said that $T = 1$ corresponds to a high temperature. To increase the performance of the evaluation function, we should use a smaller T (i.e., a lower temperature) and expand the range of the evaluation function exponent to make the maximum value peaks sharper. However, at this limit, the evaluation function can be approximated by COUNT. For this reason, at the low temperature limit, these two evaluation functions should exhibit more or less the same performance as M-LSH when using COUNT.

COSINE performed much worse than LSH for the following reason. In COSINE, there is a gentle evaluation function gradient at all points in the region where the normal vector is defined. Therefore, the normal vectors tend to be oriented toward the point that has a global maximum value. To see that the normal vector actually exists at such a point, we calculated the absolute values of the cosines between the normal vectors. A larger absolute value of the cosine means that the vectors point in similar directions. **Figure 8** shows the absolute values of the cosines between 32 normal vectors made by M-LSH using COUNT or COSINE. In this figure, a matrix is calculated with the absolute values of cosines between 32 normal vectors as its constituent values, and

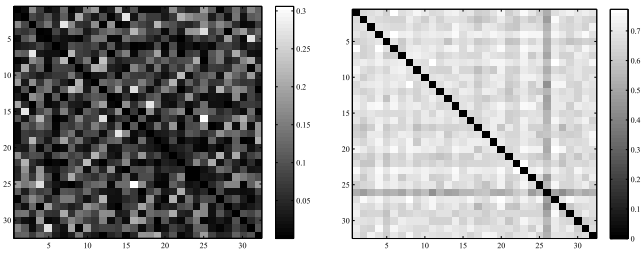

Fig. 8 Cosines between normal vectors in M-LSH results learned for MNIST, with COUNT (left) and COSINE (right) as the evaluation function.

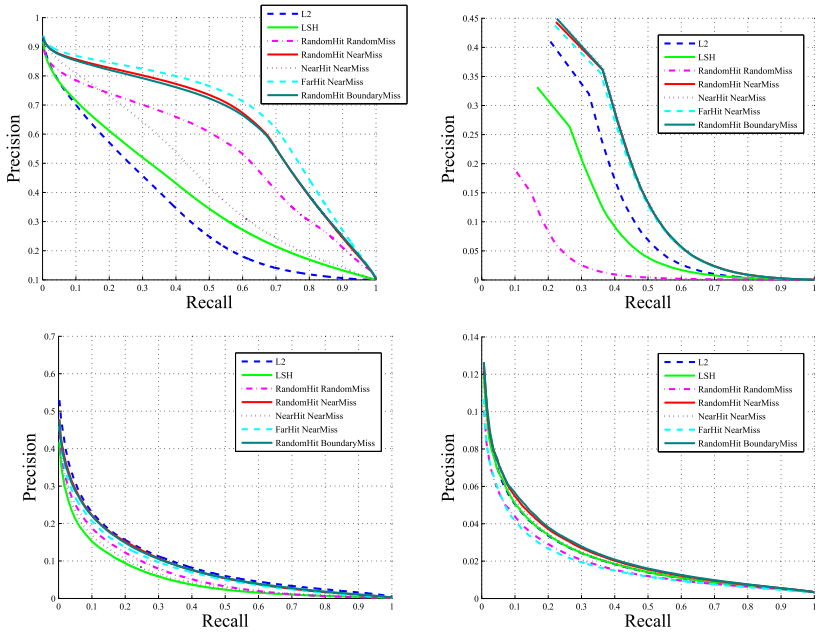

Fig. 9 Recall-Precision curves for each sampling method with 2,000 data pairs for MNIST (upper left), fingerprint (upper right), speech (lower left), and LabelMe (lower right).

these values are represented as a grayscale image. The diagonal elements are all zero. As Fig. 8 clearly shows, almost all of the normal vectors obtained with M-LSH are oriented in similar directions. In the case of COSINE, it is thought that the performance can be improved by taking a low temperature limit, as was the case for RATIO and COSINE_RATIO. However, at this limit, COSINE can be approximated by COUNT. Furthermore, since COSINE requires more processing time than COUNT, there is no reason to employ COSINE.

On the basis of the above calculations and discussion, we conclude that using COUNT as the evaluation function is more appropriate from the viewpoint of processing time and performance.

4.4 Evaluation of Sampling Methods

In light of the discussion of Section 4.3, we used the M-LSH method with the COUNT evaluation function to evaluate the performance of the sampling methods described in Section 3.4.

We used precision-recall curves and the maxima of F-measures to assess the performance of the evaluation functions. **Figure 9** shows the precision-recall curves of each sampling method for each data set. **Figure 10** shows the maxima of F-measures for each data set. The number of data pairs used in M-LSH was 2,000. To evaluate the dependence on the number of data pairs used for training, we also evaluated the performance with 20,000 data pairs (see **Fig. 11** and **12**).

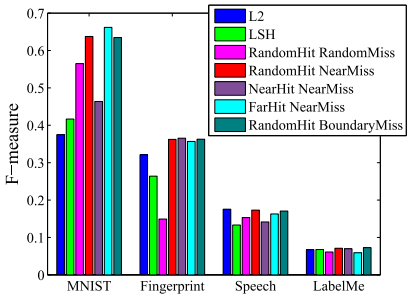

Fig. 10 Maxima of F-measures for each sampling method with 2,000 data pairs for MNIST, fingerprint, speech, and LabelMe.

Figures 9 to 11 indicate that the performance of M-LSH using RandomHit-RandomMiss sampling was very poor on data sets other than MNIST. The performance was worse than that of LSH.

M-LSH using FarHit-NearMiss sampling performed the best on MNIST. However, it performed the worst on LabelMe.

These results show that the performance of M-LSH with the NearHit-NearMiss sampling method depends strongly on the number of training data pairs. For the speech features and LabelMe data sets, the performance improves as the number of training data pairs increases. This performance improvement is thought to be due to the low probability of there being overlapping data pairs. For MNIST, the performance decreases as the number of training data pairs increases. This effect is thought

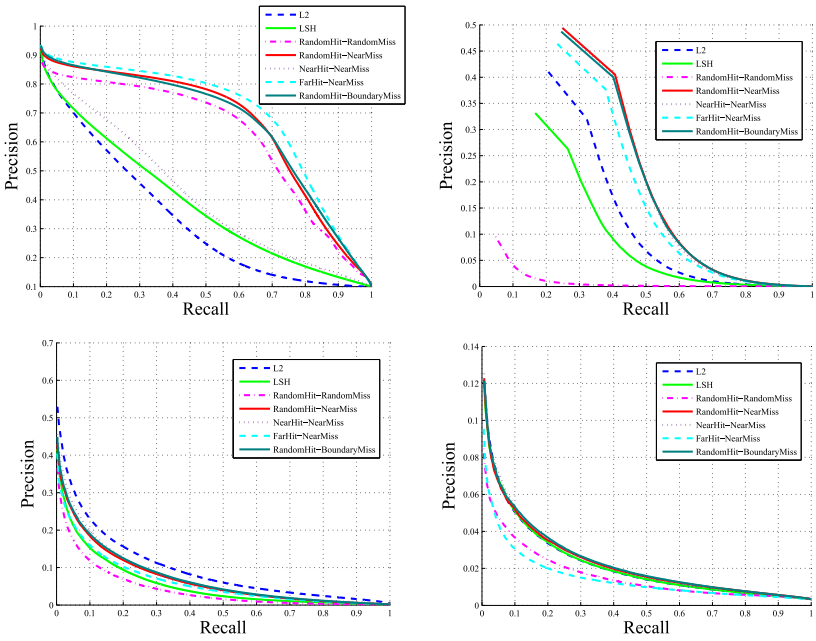

Fig. 11 Recall-Precision curves for each sampling method with 20,000 data pairs for MNIST (upper left), fingerprint (upper right), speech (lower left), and LabelMe (lower right).

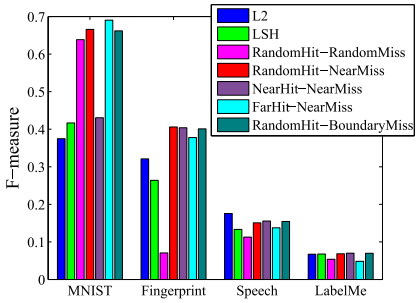

Fig. 12 Maxima of F-measures for each sampling method with 20,000 data pairs for MNIST, fingerprint, speech, and LabelMe.

to occur in the following way. As mentioned above, the role of positive pairs is to prevent data sets with a common label from being split by hyperplanes. It is therefore desirable that positive pairs are widely distributed over data sets having a common label. NearHit sampling creates positive pairs by choosing the closest feature vectors with a common label. Therefore, a large number of positive pairs are needed to widely distribute over data sets having a common label. In particular, MNIST requires more positive pairs than other data sets because there are many data items that have the same label. The role of negative pairs is to separate data sets having different labels. Therefore, a number of negative pairs roughly equal to the square of the number of labels is sufficient. Since the positive pairs and negative pairs were used in equal numbers in these experiments, it seems that the effect of negative pairs to separate data sets having different labels outweighed the effect of positive pairs to prevent the separation of data sets having a common label.

The M-LSH method using RandomHit-NearMiss sampling and RandomHit-BoundaryMiss sampling performed well on all data sets, regardless of the number of training data pairs. For MNIST and fingerprint images, the performance improved as the number of training data pairs increased. However, for the speech features and LabelMe data sets, performance decreased as the number of training data pairs increased. This is thought to be due to an increase in the number of overlapping data pairs. There were no large differences between the results of these two sampling methods, but M-LSH performed slightly better with RandomHit-BoundaryMiss sampling.

On the basis of these results, we conclude that the appropriate choice of sampling method depends on the properties of the data. Of the sampling methods we tried out in this study, it seems that the following choices are robust.

- For data sets in which each feature vector has a unique label: RandomHit-NearMiss sampling or RandomHit-BoundaryMiss sampling.
- For data sets in which each feature vector has multiple labels and there are not many training data pairs: RandomHit-BoundaryMiss sampling.
- For data sets in which each feature vector has multiple labels and there are very many training data pairs: NearHit-NearMiss sampling.

5. Comparison with Existing Learning Methods

We compared M-LSH with other learning methods (LSH, MLH, and S-LSH). In this experiment, M-LSH used the COUNT evaluation function and the RandomHit-BoundaryMiss sampling method. The number of sample data pairs was 10,000 for positive (negative) pairs.

We used the following parameters for MLH (see Ref. [9] for the meaning of each parameter): number of iterations of batch: 10,000, λ: 1.0, ρ divided by the number of bits: 0.2, ϵ: 1.0, η: 0.1, α: 0.9. We used the following parameters for S-LSH (see Ref. [10] for the meaning of each parameter): number of iterations: 10,000, $\tilde{B}$: 10,000.

Figures 13 and **14** plots the precision-recall curves and the

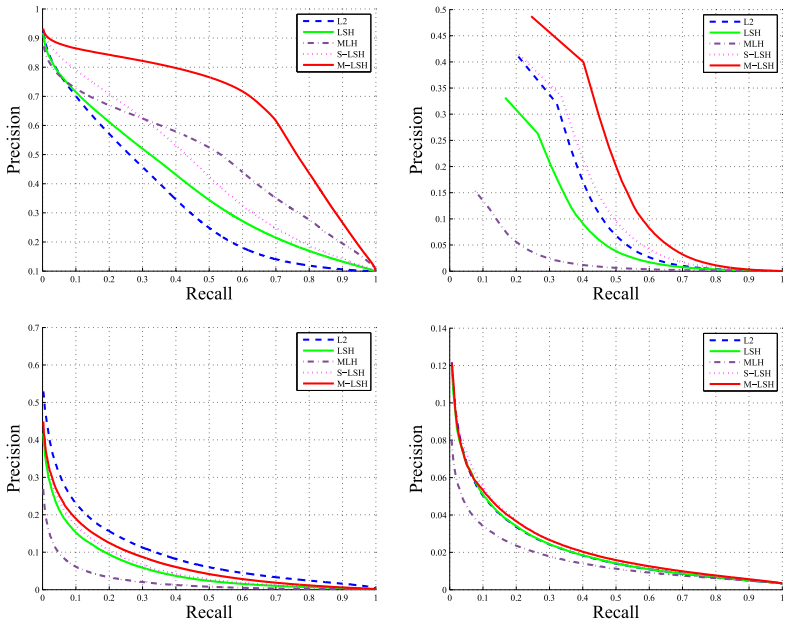

Fig. 13 Recall-Precision curves for MNIST (upper left), fingerprint (upper right), speech (lower left), and LabelMe (lower right).

Table 2 Processing time for learning (sec).

Algorithm \ Data set	MNIST	Fingerprint	Speech	LabelMe
LSH	0.0	0.0	0.0	0.0
MLH	2,742.00	5,437.82	519.05	503.30
S-LSH	55,293.61	9,142.87	56,162.03	8,780.21
M-LSH with 2,000 data pairs	3,900.80	9,445.55	1,945.66	531.77
M-LSH with 20,000 data pairs	51,078.65	120,032.97	21,627.17	6,378.24

maxima of F-measures for various data sets. The number of bits was 1,024. These results reveal that M-LSH outperformed the existing learning methods on all data sets.

One of the reasons for MLH's performance deterioration is the sampling method of the data pairs used for learning. In Ref. [9], the learning data pairs are selected at random. That sampling method corresponds to RandomHit-RandomMiss in our terminology. As we saw in Section 4.4, M-LSH with RandomHit-RandomMiss performed poorly. Therefore, we expect that MLH would perform better if the sampling method for data pairs were altered. However, we have to tune many parameters of MLH in order to improve its performance[*3].

Table 2 lists the processing times for learning of each method. The experiment's environment was as follows. The CPU was an Intel Xeon X5680 3.3 GHz. Each method was implemented using C++. Each process was a single thread. We used the clock() function to measure the processing time.

From the above results, we conclude that the M-LSH learning method has good performance.

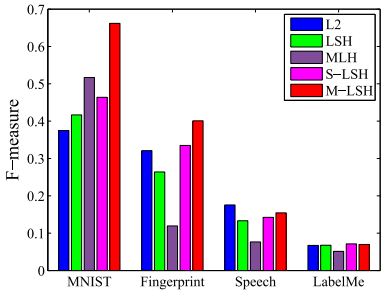

Fig. 14 Maxima of F-measures for MNIST, fingerprint, speech, and LabelMe.

6. Summary and Future Work

In this paper, we proposed a method for learning hyperplanes that uses MCMC. We also examined the evaluation functions and the sampling methods used in this learning method and evaluated their performance. The results show that our method outperforms existing learning methods.

Finally, let us mention the direction of future research. When using the MCMC method for learning, the ultimate positions of the particles do not lie at points that maximize the evaluation function. A possible way to resolve this problem involves recording the particle loci and finding out which point maximizes the evaluation function.

[*3] The reason why MLH performed differently from what is reported in Ref. [9] is the difference in the labeling strategy and usage of the data sets. In this study, we separated the data set into three pairwise disjoint sets: a data set for learning, one to be searched, and one for queries. In Ref. [9], two data sets were used: one for learning and searching, and the other for querying. As such, the performance of MLH appears to be better when the data set is divided in two rather than in three. The labeling strategy used in Ref. [9] for the MNIST data set was as follows. Calculate the L2 distances between feature vectors, determine the neighbors of each vector by thresholding the distances, and assign the same label to the neighbor. Each vector has, on average, 100 neighbors. Because this strategy is different from ours, the performance differs.

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 (online), DOI: 10.1145/293347.293348 (1998).

[2] Jagadish, H.V., Ooi, B.C., Tan, K.-L., Yu, C. and Zhang, R.: iDistance: An adaptive B+-tree based indexing method for nearest neighbor search, *ACM Trans. Database Syst.*, Vol.30, No.2, pp.364–397 (online), DOI: 10.1145/1071610.1071612 (2005).

[3] Weber, R., Schek, H.-J. and Blott, S.: A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces, *Proc. 24th International Conference on Very Large Data Bases*, *VLDB '98*, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc., pp.194–205 (1998) (online), available from ⟨http://dl.acm.org/citation.cfm?id=645924.671192⟩.

[4] 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 (online), DOI: 10.1145/276698.276876 (1998).

[5] Charikar, M.S.: Similarity estimation techniques from rounding algorithms, *Proc. 34th Annual ACM Symposium on Theory of Computing*, *STOC '02*, pp.380–388, ACM (online), DOI: 10.1145/509907.509965 (2002).

[6] Mu, Y., Chen, X., Chua, T.-S. and Yan, S.: Learning reconfigurable hashing for diverse semantics, *Proc. 1st ACM International Conference on Multimedia Retrieval*, *ICMR '11*, pp.7:1–7:8, ACM (online), DOI: 10.1145/1991996.1992003 (2011).

[7] Jiang, Y.-G., Wang, J. and Chang, S.-F.: Lost in binarization: query-adaptive ranking for similar image search with compact codes, *Proc. 1st ACM International Conference on Multimedia Retrieval*, *ICMR '11*, pp.16:1–16:8, ACM (online), DOI: 10.1145/1991996.1992012 (2011).

[8] Wang, X.-J., Zhang, L., Jing, F. and Ma, W.-Y.: AnnoSearch: Image Auto-Annotation by Search, *Proc. 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2*, *CVPR '06*, pp.1483–1490, IEEE Computer Society (online), DOI: 10.1109/CVPR.2006.58 (2006).

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

[10] Konoshima, M. and Noma, Y.: Locality-sensitive hashing with margin based feature selection, arXiv:1209.5833 (2012).

[11] Noma, Y. and Konoshima, M.: Hyperplane Arrangements and Locality-Sensitive Hashing with Lift, arXiv:1212.6110 (2012).

[12] Hastings, W.K.: Monte Carlo sampling methods using Markov chains and their applications, *Biometrika*, Vol.57, No.1, pp.97–109 (1970).

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

[14] : Internet relay broadcast of Kawasaki-city parliament, available from ⟨http://www.kawasaki-council.jp/⟩.

[15] Oliva, A. and Torralba, A.: Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope, *Int. J. Comput. Vision*, Vol.42, No.3, pp.145–175 (online), DOI: 10.1023/A:1011139631724 (2001).

[16] Torralba, A., Fergus, R. and Weiss, Y.: Small codes and large image databases for recognition, *Proc. IEEE Conf. Computer Vision and Pattern Recognition* (2008).

Yui Noma received his B.S. in physics from Kyoto University, Kyoto, Japan in 2003. He received his M.S. and Ph.D. in physics from Osaka University, Osaka, Japan in 2005, 2008. His interest was elementary particle physics. His current interest is data minings and machine learning.

Makiko Konoshima received her B.Eng. degree in electronics engineering from Yokohama National University, Japan in 1986. Her current research focuses on big data processing and analytics.