PAPER

Revisiting a Nearest Neighbor Method for Shape Classification

Kazunori IWATA[†a)], ***Member***

SUMMARY The nearest neighbor method is a simple and flexible scheme for the classification of data points in a vector space. It predicts a class label of an unseen data point using a majority rule for the labels of known data points inside a neighborhood of the unseen data point. Because it sometimes achieves good performance even for complicated problems, several derivatives of it have been studied. Among them, the discriminant adaptive nearest neighbor method is particularly worth revisiting to demonstrate its application. The main idea of this method is to adjust the neighbor metric of an unseen data point to the set of known data points before label prediction. It often improves the prediction, provided the neighbor metric is adjusted well. For statistical shape analysis, shape classification attracts attention because it is a vital topic in shape analysis. However, because a shape is generally expressed as a matrix, it is non-trivial to apply the discriminant adaptive nearest neighbor method to shape classification. Thus, in this study, we develop the discriminant adaptive nearest neighbor method to make it slightly more useful in shape classification. To achieve this development, a mixture model and optimization algorithm for shape clustering are incorporated into the method. Furthermore, we describe several helpful techniques for the initial guess of the model parameters in the optimization algorithm. Using several shape datasets, we demonstrated that our method is successful for shape classification.

key words: *shape classification, ordinary Procrustes sum of squares, nearest neighbor method, discriminant adaptive nearest neighbor method*

1. Introduction

Let a data point denote a point in a vector space. Generally, the goal of the classification problem in pattern recognition is to identify which class label an unseen data point has by referring to a training set. In this study, we simply call this decision prediction. The training set typically contains pairs that consist of a data point and its class label. The nearest neighbor (NN) method (e.g., see [1]–[3]) is a simple and flexible scheme for label prediction. Although the NN method does not require processing the training set in advance and uses the training set as it is, it sometimes provides significant performance in terms of prediction, even for complicated problems. Therefore, several derivatives of it have been presented. Among them, the discriminant adaptive (DA) NN method [4] in particular has had a large impact on the other derivatives. The DANN method can be regarded as a type of metric learning [5] that has recently attracted much attention in machine learning. Regarding statistical shape analysis (see [6], [7] for background material), shape classification is an interesting application [8], [9]. However, because a shape is typically expressed as a matrix, it may be non-trivial to apply the NN method to shape classification. Moreover, it must be non-trivial to apply the DANN method. This was our motivation to perform this study. In this study, we develop the DANN method to make it slightly more useful in shape classification. To achieve this development, a mixture model and optimization algorithm are incorporated into it. Additionally, we introduce several helpful techniques for the initialization of model parameters in the optimization algorithm. Using the shape datasets, we show that our DANN method is successful for shape classification. The structure of this paper is as follows: We briefly describe the basis of shape analysis in Sect. 2 and combine it with NN and DANN methods in Sect. 3. Then, we present several experimental results in Sect. 4. In Sect. 5, we provide our summary of this study.

Manuscript received April 1, 2020.
Manuscript revised August 1, 2020.
Manuscript publicized September 23, 2020.

[†]The author is with the Graduate School of Information Sciences, Hiroshima City University, Hiroshima-shi, 731–3194 Japan.
a) E-mail: kiwata@hiroshima-cu.ac.jp

DOI: 10.1587/transinf.2020EDP7074

2. Preliminaries

Generally, a shape is formed of contours, such as handwriting using a pen or object boundaries detected using picture image processing. A shape is typically expressed using a finite number of points on contours that are numbered or labeled in some manner. Such a point is referred to as a landmark [6], [7]. An annotated number assigned to a landmark indicates that there is correspondence between similar shapes for the same object.

2.1 Notation

We begin with the basic notation of statistical shape analysis. We use $\mathbb{R}$, $\mathbb{R}^+$, and $\mathbb{R}_0^+$ to denote the sets of real numbers, positive real numbers, and non-negative real numbers, respectively. Let ℓ be the number of landmarks for a shape. Let I_ℓ be the $\ell \times \ell$ identity matrix in which all the diagonal elements are one, and 1_ℓ be the ℓ-dimensional vector in which all the elements are one. When we represent a shape in m dimensions, we construct a numerical matrix denoted by

$$\begin{pmatrix} x_{11} & \cdots & x_{1m} \\ x_{21} & \cdots & x_{2m} \\ \vdots & \ddots & \vdots \\ x_{\ell 1} & \cdots & x_{\ell m} \end{pmatrix}, \tag{1}$$

where $(x_{a1}, \ldots, x_{am})$ denotes the coordinate of the a-th landmark for $a = 1, 2, \ldots, \ell$. We denote the $\ell \times m$ matrix by X, which is called a configuration matrix [7]. The centering matrix [7] defined as

$$C = I_\ell - \frac{1}{\ell} 1_\ell 1_\ell^{\mathrm{T}} \tag{2}$$

is such an $\ell \times \ell$ matrix that moves the center of gravity of coordinates in a configuration matrix to the origin of the coordinates. The superscript T denotes the transpose of a matrix or vector. The size of X is denoted by $\|X\|$ and is defined as

$$\|X\| = \sqrt{\mathrm{tr}\,(X^{\mathrm{T}} X)}, \tag{3}$$

where tr is the trace of a square matrix. The normalization of X implies $X/\|X\|$. For example, for a configuration matrix and its centering, if

$$X = \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 2 & 2 \\ 2 & 1 \end{pmatrix}, \tag{4}$$

then $C = I_4 - 1_4 1_4^{\mathrm{T}}/4$ and

$$CX = \begin{pmatrix} -1/2 & -1/2 \\ -1/2 & 1/2 \\ 1/2 & 1/2 \\ 1/2 & -1/2 \end{pmatrix}. \tag{5}$$

The center of gravity of coordinates in this configuration matrix is indeed the origin. Incidentally, because the size of the centered matrix is $\sqrt{2}$, its normalization results in

$$\frac{CX}{\|CX\|} = \begin{pmatrix} -1/\left(2\sqrt{2}\right) & -1/\left(2\sqrt{2}\right) \\ -1/\left(2\sqrt{2}\right) & 1/\left(2\sqrt{2}\right) \\ 1/\left(2\sqrt{2}\right) & 1/\left(2\sqrt{2}\right) \\ 1/\left(2\sqrt{2}\right) & -1/\left(2\sqrt{2}\right) \end{pmatrix}. \tag{6}$$

Clearly, the size of this matrix is one.

2.2 OSS

The shape distance is essentially some distance between landmarks on a shape and those on another shape that quantifies the dissimilarity between two shapes. The ordinary Procrustes sum of squares (OSS) [7] is one of the most typical shape distances. For any centered configuration matrices CX_1, CX_2, it is defined as

$$\begin{aligned} &\mathrm{OSS}\,(CX_1, CX_2) \\ &= \inf_{\Gamma \in \mathrm{SO}(m),\, \gamma \in \mathbb{R}^m,\, \beta \in \mathbb{R}^+} \left\| CX_2 - \beta CX_1 \Gamma - 1_\ell \gamma^{\mathrm{T}} \right\|^2, \end{aligned} \tag{7}$$

where $\mathrm{SO}(m)$ is the m-dimensional rotation group [10]; for example, if $m = 2$, then for all $\theta \in \mathbb{R}$,

$$\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \in \mathrm{SO}(2). \tag{8}$$

Intuitively, Γ, γ, and β in (7) imply a rotation around the origin, translation, and isotropic scale, respectively, in $\mathbb{R}^m$. Accordingly, to compute the OSS, we need to solve the optimization problem with respect to Γ, γ, and β, that is,

$$\begin{aligned} \text{min.} \quad & \left\| CX_2 - \beta CX_1 \Gamma - 1_\ell \gamma^{\mathrm{T}} \right\|^2 \\ \text{s.t.} \quad & \gamma \in \mathbb{R}^m, \\ & \Gamma \in \mathrm{SO}(m), \\ & \beta \in \mathbb{R}^+. \end{aligned}$$

In fact, according to [7], [11], the solution is analytically given by

$$\hat{\gamma} = \mathbf{0}, \tag{9}$$

$$\hat{\Gamma} = UV^{\mathrm{T}}, \tag{10}$$

$$\hat{\beta} = \frac{\mathrm{tr}\left((CX_2)^{\mathrm{T}} (CX_1) \hat{\Gamma} \right)}{\mathrm{tr}\left((CX_1)^{\mathrm{T}} (CX_1) \right)}, \tag{11}$$

where $U, V \in \mathrm{SO}(m)$ are $m \times m$ matrices such that

$$(CX_2)^{\mathrm{T}} (CX_1) = \|CX_1\| \|CX_2\| V \Lambda U^{\mathrm{T}}. \tag{12}$$

In some cases, shapes should be compared on the same scale as they are; that is, it is not of interest to consider scaling. The ordinary partial Procrustes sum of squares (PSS) [7] is one of the most typical shape distances and defined as

$$\begin{aligned} &\mathrm{PSS}\,(CX_1, CX_2) \\ &= \inf_{\Gamma \in \mathrm{SO}(m),\, \gamma \in \mathbb{R}^m} \left\| CX_2 - CX_1 \Gamma - 1_\ell \gamma^{\mathrm{T}} \right\|^2. \end{aligned} \tag{13}$$

To obtain the PSS, we need to solve

$$\begin{aligned} \text{min.} \quad & \left\| CX_2 - CX_1 \Gamma - 1_\ell \gamma^{\mathrm{T}} \right\|^2 \\ \text{s.t.} \quad & \gamma \in \mathbb{R}^m, \\ & \Gamma \in \mathrm{SO}(m), \end{aligned}$$

and the solution to this problem is given by (9) and (10) [7].

2.3 Shape Clustering

We have mentioned that a shape is expressed as a configuration matrix, a similarity transformation invariant distance between shapes can be given by the OSS, and a rotation and translation invariant distance can be given by the PSS. Shape clustering [12], [13] is an interesting application of their shape distances because clustering can only be performed if we can define a suitable distance for shapes. Now we briefly introduce a stochastic view of the generation of the configuration matrix. According to [13], we define the probability density for any configuration matrix X as

$$p\,(X; \theta) = \sum_{y=1}^{G} \alpha_y p_y \left(X; W_y \right), \tag{14}$$

where G denotes a constant parameter for the number of components, y denotes a class label, and α_y denotes the mixing parameter with respect to y. The component density p_y

Algorithm 1 OSS- or PSS- based EM for shape clustering

Input:
$\{X_1, \ldots, X_n\}$: set of configuration matrices
Output:
$\hat{\theta}^{(t)}$: set of estimated parameters of the model
function OSS- OR PSS-BASED EM($\{X_1, \ldots, X_n\}$)
Initialize $\hat{\theta}^{(0)}$ in some manner
$t \leftarrow 0$
repeat
$t \leftarrow t + 1$
calculate $p\left(y \mid X_i; \hat{\theta}^{(t-1)}\right)$ for all $y \in \{1, \ldots, G\}$ and all $i \in \{1, \ldots, n\}$
calculate $\hat{\alpha}_y^{(t)}$ and $\hat{W}_y^{(t)}$ for all $y \in \{1, \ldots, G\}$
until difference between $\hat{\theta}^{(t)}$ and $\hat{\theta}^{(t-1)}$ is insignificant
return $\hat{\theta}^{(t)}$
end function

is written as

$$p_y\left(X; W_y\right) = \frac{1}{c(\kappa)} \exp\left(-\kappa \operatorname{OSS}\left(CX, W_y\right)\right), \tag{15}$$

where $\kappa \in \mathbb{R}^+$ is a constant parameter that magnifies the OSS and $c(\kappa) \in \mathbb{R}^+$ is a normalization parameter. The $\ell \times m$ matrix W_y is referred to as a prototype matrix because, in this model, W_y acts as a template or prototype such that it corresponds to the mean matrix with respect to the probability density p_y. The PSS version is also defined by substituting the PSS for the OSS in (15). The form of the probability density in (14) is called a mixture model [1]. To summarize, G and κ are considered as constant parameters of the model. Although the approach used to determine these constant parameters when we have no prior knowledge about them might be a profound matter, it is not of immediate relevance for the main purpose of this study. Simultaneously, α_y and W_y are not constant and should be estimated to fit the model to the probability density of the configuration matrices in a dataset, which is denoted by $\{X_1, \ldots, X_n\}$, where n is the number of elements in the set. Let $\theta = \left\{W_y, \alpha_y \mid y = 1, \ldots, G\right\}$. The most popular method to estimate such a set of parameters of the mixture model is the expectation-maximization (EM) algorithm. Because the EM algorithm performs an iterative estimation, let t denote the number of iterations. Let $\hat{\theta}^{(0)} = \left\{\hat{W}_y^{(0)}, \hat{\alpha}_y^{(0)} \mid y = 1, \ldots, G\right\}$ be the initial guess of θ. For $t = 1, 2, \ldots$, based on the current estimated set $\hat{\theta}^{(t-1)}$, the probability that the label of configuration matrix X_i is y is drawn according to

$$p\left(y \mid X_i; \hat{\theta}^{(t-1)}\right) = \frac{\hat{\alpha}_y^{(t-1)} p_y\left(X_i; \hat{W}_y^{(t-1)}\right)}{\sum_{y=1}^{G} \hat{\alpha}_y^{(t-1)} p_y\left(X_i; \hat{W}_y^{(t-1)}\right)}. \tag{16}$$

Subsequently, based on this estimated probability, when we use the OSS, we modify all the elements in $\hat{\theta}^{(t-1)}$ using

$$\hat{\alpha}_y^{(t)} = \frac{1}{n} \sum_{i=1}^{n} p\left(y \mid X_i; \hat{\theta}^{(t-1)}\right), \tag{17}$$

$$\hat{W}_y^{(t+\frac{1}{2})} = \frac{\sum_{i=1}^{n} p\left(y \mid X_i; \hat{\theta}^{(t-1)}\right) \hat{\beta} C X_i \hat{\Gamma}}{\sum_{i=1}^{n} p\left(y \mid X_i; \hat{\theta}^{(t-1)}\right)}, \tag{18}$$

$$\hat{W}_y^{(t+1)} = \frac{\hat{W}_y^{(t+\frac{1}{2})}}{\left\|\hat{W}_y^{(t+\frac{1}{2})}\right\|}. \tag{19}$$

where $\hat{\Gamma}$ and $\hat{\beta}$ are determined using CX_i and $\hat{W}_y^{(t-1)}$, as defined in (10) and (11). Equation (19) defines the normalization of a prototype matrix. When we use the PSS, we modify all the elements in $\hat{\theta}^{(t-1)}$ using (17) and

$$\hat{W}_y^{(t+1)} = \frac{\sum_{i=1}^{n} p\left(y \mid X_i; \hat{\theta}^{(t-1)}\right) C X_i \hat{\Gamma}}{\sum_{i=1}^{n} p\left(y \mid X_i; \hat{\theta}^{(t-1)}\right)}, \tag{20}$$

where $\hat{\Gamma}$ is determined using CX_i and $\hat{W}_y^{(t-1)}$, as defined in (10). The normalization of a prototype matrix is not applied in the case of the PSS. These modified parameters are used in the next iteration step. We summarize the OSS- or PSS-based EM algorithm in Algorithm 1.

3. Main Results

As shown in (16), the EM algorithm contains the probability estimation of a label given a configuration matrix. The main idea of our method is to apply the probability estimation to the probability estimation in the DANN.

3.1 Simple NN Method

Throughout this paper, let k be a constant parameter in the set of positive integers. The k-NN method is well-known in pattern recognition as a classification method for data points in a vector space. An unseen data point to which we now refer for prediction is called a test point. The parameter k plays a role in determining the neighborhood of a test point. Now we consider predicting the label of a test point $\boldsymbol{x} \in \mathbb{R}^m$, exploiting the training set of pairs, denoted by $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$, where n is the number of pairs, $\boldsymbol{x}_i \in \mathbb{R}^m$ denotes the i-th data point, and y_i denotes its label. Then, using the ordinary Euclidean distance between points, we choose a radius of the hypersphere around $\boldsymbol{x}$ in $\mathbb{R}^m$ such that the hypersphere contains k data points in the training set. Finally, we assign a label to $\boldsymbol{x}$ using a majority rule for k labels inside the hypersphere. The entire NN method

Algorithm 2 NN method

Input:
$\boldsymbol{x}$: test point in $\mathbb{R}^m$
$\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$: training set of pairs
Output:
y: estimated label to $\boldsymbol{x}$
function k-NN($\boldsymbol{x}$, $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$)
find k-NNs in $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ using the Euclidean distance between $\boldsymbol{x}$ and $\boldsymbol{x}_i$ for all $i \in \{1, \ldots, n\}$
$y \leftarrow$ the most frequent label in $\{y_{i_1}, \ldots, y_{i_k}\}$, where $(i_1, \ldots, i_k)$ denote the indices of the k-NNs
return y
end function

Algorithm 3 DANN method

Input:
$\boldsymbol{x}$: test point in $\mathbb{R}^m$
$\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$: training set of pairs
k'; number of data points used for optimizing a distance function
Output:
y: estimated label to $\boldsymbol{x}$
function DAk-NN($\boldsymbol{x}$, $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$, k')
find k'-NNs in $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ using the Euclidean distance between $\boldsymbol{x}$ and $\boldsymbol{x}_i$ for all $i \in \{1, \ldots, n\}$
optimize a distance function using $\left\{(\boldsymbol{x}_{i'_1}, y_{i'_1}), \ldots, (\boldsymbol{x}_{i'_{k'}}, y_{i'_{k'}})\right\}$, where $(i'_1, \ldots, i'_{k'})$ denote the indices of the k'-NNs
find k-NNs in $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ using the distance function between $\boldsymbol{x}$ and $\boldsymbol{x}_i$ for all $i \in \{1, \ldots, n\}$
$y \leftarrow$ the most frequent label in $\{y_{i_1}, \ldots, y_{i_k}\}$, where $(i_1, \ldots, i_k)$ denote the indices of the k-NNs
return y
end function

is shown in Algorithm 2. There are various derivatives of the assignment rule, but this study does not address them in depth.

3.2 DANN Method

The simple k-NN method uses the Euclidean distance between points to choose a radius of the hypersphere around test point $\boldsymbol{x}$. However, for each $\boldsymbol{x}$, the DANN method uses another distance that is designed for the neighborhood of $\boldsymbol{x}$. First, using the Euclidean distance between points, we choose the radius of the hypersphere around $\boldsymbol{x}$ in $\mathbb{R}^m$ such that the hypersphere contains k' data points in the training set, where k' is a constant parameter in the set of positive integers smaller than n. Then, the data points inside the hypersphere are used to optimize a distance function between points, which is available inside the hypersphere. After optimizing the distance function in some manner (see [4] for an example of optimization), according to the distance function between points, we choose a radius of the hypersphere around $\boldsymbol{x}$ in $\mathbb{R}^m$ such that the hypersphere contains k data points in the training set. As in the simple k-NN, we finally assign a label to $\boldsymbol{x}$ using a majority rule for k labels inside the hypersphere. The entire DANN is shown in Algorithm 3. The key point of the DANN method is to adjust the distance function finely for each test point, which is involved in selecting the NNs, before using it for the prediction of the label. Hence, this method is a type of metric learning or metric adaptation. For each test data, this method requires processing k' data points in the training set of pairs, so it clearly results in a higher computational cost than the simple NN method explained above. However, it significantly improves the prediction, provided the distance function is adjusted well.

3.3 OSS-Based DANN Method

The purpose of this study is to apply the DANN method to shape classification. To the best of our knowledge, no studies in shape analysis have ever tried to make effective use of the DANN method. When we consider changing the DANN method to make it useful in shape classification, we are confronted with two problems. The first is determining what distance is standard for shapes. Because a shape is expressed as a matrix, the Euclidean distance defined over a vector space cannot straightforwardly adopt it. To overcome this problem, we use the OSS or PSS, which was explained in the previous section, instead of the Euclidean distance. Using the OSS or PSS, substituting configuration matrices for data points changes the NN method to its OSS- or PSS-based version given in Algorithm 4. The second problem is determining what distance function is suitable for the metric adaptation for a test point. This problem is not as simple to overcome as the first, but we propose using information divergence [1], [2], which is one of the most familiar probability-based pseudo-distances in pattern recognition. For any test configuration matrix X and any configuration matrix X_i in a training set, the information divergence is defined as

$$d(X, X_i) = \sum_{y=1}^{G} p(y \mid X) \log \frac{p(y \mid X)}{p(y \mid X_i)}, \tag{21}$$

Algorithm 4 OSS- or PSS-based NN method

Input:
X: test configuration matrix
$\{(X_1, y_1), \ldots, (X_n, y_n)\}$: training set of pairs
Output:
y: estimated label to X
function OSS- or PSS-based k-NN(X, $\{(X_1, y_1), \ldots, (X_n, y_n)\}$)
find k-NNs in $\{X_1, \ldots, X_n\}$ using the OSS or PSS between CX and CX_i for all $i \in \{1, \ldots, n\}$
$y \leftarrow$ the most frequent label in $\{y_{i_1}, \ldots, y_{i_k}\}$, where $(i_1, \ldots, i_k)$ denote the indices of the k-NNs
return y
end function

Algorithm 5 OSS- or PSS-based DANN method

Input:
X: test configuration matrix
$\{(X_1, y_1), \ldots, (X_n, y_n)\}$: training set of pairs
k'; number of data points used for optimizing the distance function d
Output:
y: estimated label to $\boldsymbol{x}$
function OSS- or PSS-based DAk-NN(X, $\{(X_1, y_1), \ldots, (X_n, y_n)\}$, k')
find k'-NNs in $\{X_1, \ldots, X_n\}$ using the OSS or PSS between CX and CX_i for all $i \in \{1, \ldots, n\}$
$\hat{\theta} \leftarrow$ OSS- or PSS-BASED EM$\left(\left\{X_{i'_1}, \ldots, X_{i'_{k'}}\right\} \cup \{X\}\right)$, where $(i'_1, \ldots, i'_{k'})$ denote the indices of the k'-NNs
find k-NNs in the k'-NNs $\left\{X_{i'_1}, \ldots, X_{i'_{k'}}\right\}$, where $k \le k'$, using $d\left(X, X_{i'}; \hat{\theta}\right)$ for all $i' \in \left\{i'_1, \ldots, i'_{k'}\right\}$
$y \leftarrow$ the most frequent label in $\{y_{i_1}, \ldots, y_{i_k}\}$, where $(i_1, \ldots, i_k)$ denote the indices of the k-NNs
return y
end function

where p denotes a probability density. However, this proposed approach causes another problem, which is how to implement $p(y \mid X)$ in (21) and fit it for a training set. Our subsequent proposed approach is to use the mixture model and EM algorithm in Algorithm 1. Accordingly, (21) may be rewritten as

$$d\left(X, X_i; \hat{\theta}\right) = \sum_{y=1}^{G} p\left(y \mid X; \hat{\theta}\right) \log \frac{p\left(y \mid X; \hat{\theta}\right)}{p\left(y \mid X_i; \hat{\theta}\right)}, \tag{22}$$

where p is obtained using (16), and $\hat{\theta}$ denotes the estimated set of parameters in the EM algorithm. Our method is thus summarized in Algorithm 5. Hereafter, we call it the OSS- or PSS-based DANN method. It is important to know the time complexity of our method. In the following, we use O-notation; refer to [14] for details. In fact, the computation of the OSS in (7) is decomposed into three operations: matrix centering, such as CX_1; matrix product-and-trace, such as $\|CX_1\|$; and singular value decomposition (SVD) on the right-hand side of (12). Assume that $\ell \ge m$, without loss of generality. The first operation takes $O\left(\ell^2 m\right)$ time using a straightforward matrix multiplication, the second operation takes $O\left(\ell m^2\right)$ time using a straightforward matrix multiplication and matrix trace, and the third operation takes $O\left(m^3\right)$ time using the Golub–Kahan method [15] or Demmel–Kahan method [16]. Consequently, the computation of the OSS takes $O\left(\ell^2 m\right)$ time. The same can be said of the computation of the PSS in (13). The NN method on the first line of Algorithm 5 consists of two parts: the computation of the OSS for all configuration matrices in the training set and sorting the OSSs. The time complexity for the first part is $O\left(n\ell^2 m\right)$, and the average time complexity for the second part is $O(n \log n)$. Next, regarding the EM on the second line of Algorithm 5, for all y and all i', obtaining $p\left(y \mid X_{i'}; \hat{\theta}^{(t-1)}\right)$ in (16) takes $O\left(Gk'\ell^2 m\right)$ time when the normalization parameter $c(\kappa)$ takes constant time. Because $\hat{\alpha}_y^{(t)}$ and $\hat{W}_y^{(t)}$ in (17) and (18)–(20) take $O(k')$ time and $O\left(k'\ell^2 m\right)$ time, respectively, calculating these for all y takes $O\left(Gk'\ell^2 m\right)$ time. Accordingly, the EM algorithm on the second line takes $O\left(Gk'\ell^2 m\right)$ time for each iteration in the algorithm. Furthermore, regarding the NN method on the third line of Algorithm 5, because the distance in (22) takes $O(G)$ time, the time complexity for all the distances is $O(Gk')$. Additionally, the average time complexity for sorting the distances is $O(k' \log k')$. The fourth line of Algorithm 5 for checking the most frequent label takes $O(k)$ time.

In the remainder of this section, we describe helpful techniques for setting the initial guess $\hat{\theta}^{(0)}$ in the EM algorithm in the second step of Algorithm 5. If there is no label y in the set of k'-NNs' labels to test configuration matrix X, then let $p\left(y \mid X; \hat{\theta}\right) = 0$ and remove y from the set of class labels in the mixture model. Otherwise, $\hat{\alpha}_y^{(0)}$ is set to the ratio of y to k' in the set of k'-NNs' labels to X. Additionally, we determine the nearest configuration matrix whose label is y to X, center it using the centering matrix C, normalize the centered matrix, and finally set the normalized matrix to $\hat{W}_y^{(0)}$. These techniques were used in the experiments described in the following section.

Algorithm 6 CD- or EMD-based method

Input:
$\mathcal{S}$: test set of points
$\{(\mathcal{S}_1, y_1), \ldots, (\mathcal{S}_n, y_n)\}$: training set of pairs
Output:
y: estimated label to $\mathcal{S}$
function CD- OR EMD-BASED k-NN($\mathcal{S}$, $\{(\mathcal{S}_1, y_1), \ldots, (\mathcal{S}_n, y_n)\}$)
find k-NNs in $\{\mathcal{S}_1, \ldots, \mathcal{S}_n\}$ using the CD or EMD between $\mathcal{S}$ and $\mathcal{S}_i$ for all $i \in \{1, \ldots, n\}$
$y \leftarrow$ the most frequent label in $\{y_{i_1}, \ldots, y_{i_k}\}$, where $(i_1, \ldots, i_k)$ denote the indices of the k-NNs
return y
end function

4. Experiments

Using three shape datasets, we verified whether the OSS- and PSS-based DANN methods worked well compared with some NN methods. The Chamfer distance (CD) [17] and earth mover's distance (EMD) [17], [18] are shape distances between sets of points on contours and used mainly in the three-dimensional model of computer-aided design. Compared with the OSS, these distances are defined for simple points in a vector space, but not for landmarks that are numbered to indicate a correspondence between similar shapes of the same object. Accordingly, the OSS yields a rotation, translation, and isotropic scale as reasons why the shapes are matched or not, whereas the CD and EMD do not yield these transformations. Additionally, the CD and EMD rely only on the Euclidean distance; hence, they are affected by the similarity transformation of one shape. As described in Algorithm 6, we constructed their NN methods. In the experiments, the methods serve as the baselines that we use to measure the prediction improvement that results from the OSS- and PSS-based NN and DANN methods.

4.1 ETHZ Shape Classes Dataset

The ETHZ shape classes dataset available in [19] was analyzed in [20] for object detection from picture images. This dataset contained the images of ground-truth outlines of 44 Apple logos, 55 bottles, 92 giraffes, 48 mugs, and 32 swans obtained by the Berkeley natural boundary detector. Thus, there were 271 ground-truth outlines that were classified into five classes ($G = 5$). We binarized and thinned the images of the ground-truth outline, and subsequently located six landmarks manually on each ground-truth outline ($\ell = 6, m = 2$). Several examples of landmarks are shown in Fig. 1. The dots in the figure denote landmarks. In fact, they are numbered to indicate a correspondence between shapes of the same class. Based on the coordinates of the landmarks, we constructed their 271 configuration matrices for the OSS- and PSS-based NN and DANN methods, and 271 point sets for the CD- and EMD-based NN methods.

4.2 Skewed Line Drawings Dataset

The skewed line drawings dataset available in [22] was used in [23] for shape retrieval. All the line drawings were classified into three shape classes, each of which contained 30 line drawings ($G = 3$). As shown in Fig. 2, some of the line drawings were skewed; hence, they cannot be matched under the similarity transformations. Each drawing was represented by the coordinates of four landmarks chosen manually in the drawing corners ($\ell = 4$, $m = 2$). Based on the coordinates of the landmarks, we constructed their 90 configuration matrices for the OSS- and PSS-based NN and DANN methods, and 90 point sets for the CD- and EMD-based NN methods.

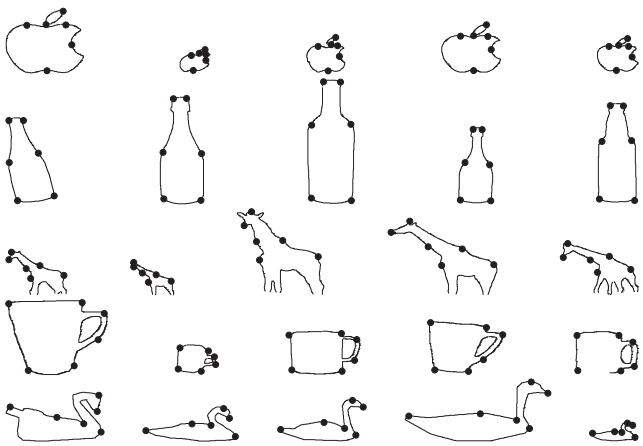


Fig. 1 Examples of six landmarks on the outlines of the Apple logo, bottle, giraffe, mug, and swan.

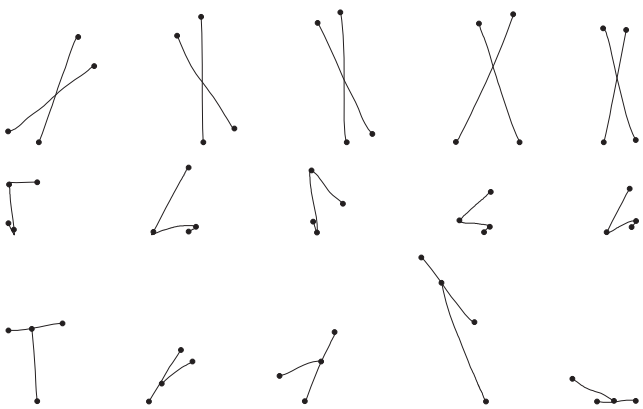
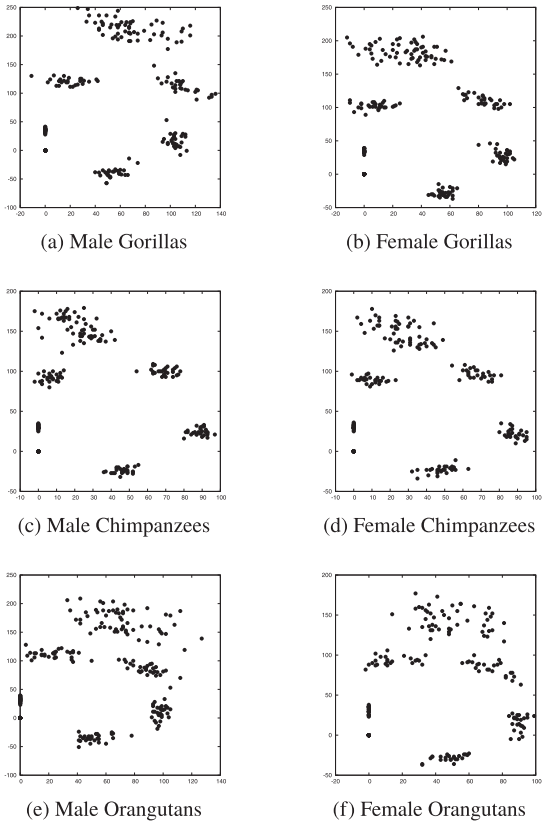


Fig. 2 Examples of four landmarks on the line drawings of X, square root, and T.



(a) Male Gorillas (b) Female Gorillas

(c) Male Chimpanzees (d) Female Chimpanzees

(e) Male Orangutans (f) Female Orangutans

Fig. 3 Examples of eight landmarks on the midline plane of each skull in [21].

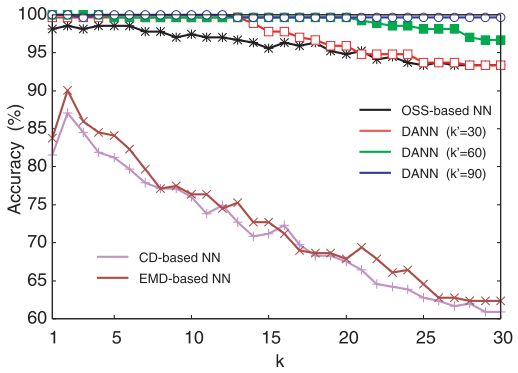


Fig. 4 Prediction accuracies of the OSS-based NN and OSS-based DANN for the ETHZ shape classes dataset.

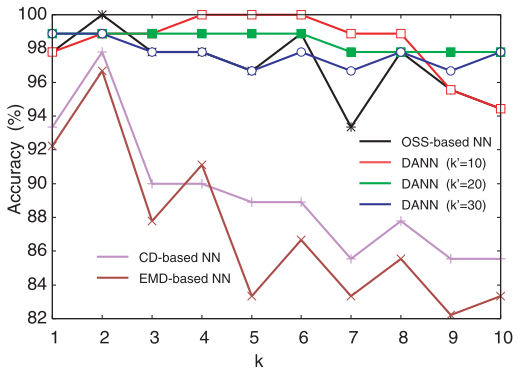


Fig. 5 Prediction accuracies of the OSS-based NN and OSS-based DANN for the skewed line drawings dataset.

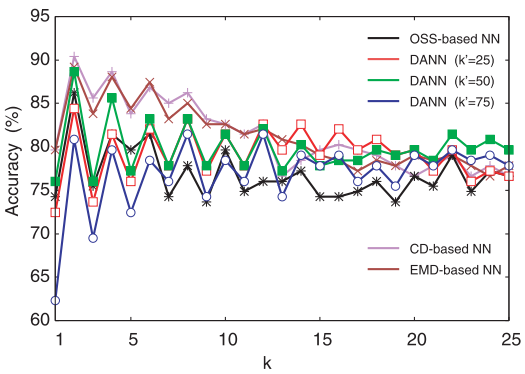


Fig. 6 Prediction accuracies of the OSS-based NN and OSS-based DANN for the great ape skulls dataset.

4.3 Great Ape Skulls Dataset

The great ape skulls dataset available in [21] was investigated in [24] to analyze the cranial differences between the sexes of ape: 29 male and 30 female gorillas, 28 male and 26 female chimpanzees, and 30 male and 24 female orangutans. Thus, 167 ape skulls were classified into six classes ($G = 6$). In fact, as shown later, the scale of the skull is vital for the prediction of its class label in this dataset; hence, the PSS is more suitable than the OSS. Each ape skull was represented by the coordinates of eight landmarks selected by an expert biologist in the midline plane of the skull ($\ell = 8$, $m = 2$). Several examples of landmarks are shown in Fig. 3. Based on the coordinates of the landmarks, we constructed their 167 configuration matrices for the OSS- and PSS-based NN and DANN methods, and 167 point sets for the CD- and EMD-based NN methods.

4.4 Assessment

Each configuration matrix or point set in a dataset was chosen as a test configuration matrix or set, and the others were used as a training set of pairs. Then, each method predicted the label of the chosen configuration matrix or set, respectively, and the number of correct predictions in each method was recorded. The overall prediction accuracy for some k was achieved by dividing that number by the total number of configuration matrices or sets when all the elements in a dataset were chosen in turn as test configuration matrices or sets. The effectiveness of the optimization of the distance function for the mixture model and EM algorithm in the OSS- and PSS-based DANN methods was assessed using the difference in prediction accuracy between all the NN methods.

4.5 Results

The prediction accuracies for the ETHZ shape classes dataset are summarized in Figs. 4 and 7, those for skewed line drawings dataset are summarized in Figs. 5 and 8, and those for great ape skulls dataset are summarized in Figs. 6 and 9. In these figures, the vertical and horizontal axes denote the prediction accuracy and the value of k, respectively. We can compare the OSS-based DANN method with the other NN methods in Figs. 4, 5, and 6. In the same way, we can compare the PSS-based DANN with the other NN methods in Figs. 7, 8, and 9. Table 1 provides the settings for κ and k' in the OSS- and PSS-based DANN methods on each dataset. Generally, the smaller the difference between classes in the landmarks, the larger the values of κ that should be selected. The runtime of each method is shown in Table 2. The runtime was measured on a personal computer with 2.9GHz Intel Core-i7 CPU, DDR3 8GB, and Windows 10, and the programs for the methods were implemented in R. For example, regarding the runtime of the OSS-based DANN method, it took at most 3.53 times more time in the entire runtime compared with the OSS-based NN method on the ETHZ shape classes dataset, 6.90 times more time for the skewed line drawings dataset, and 6.62 times more time for the great ape skulls dataset. Table 3 presents the peaks of prediction accuracy for each method on the datasets. According to the table, the OSS- and PSS-based DANN methods worked better than the OSS- and PSS-based NN methods, respectively, provided k and k' were chosen appropriately for the datasets. This means that the

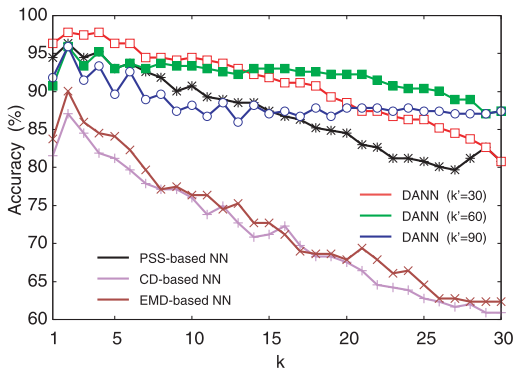


Fig. 7 Prediction accuracies of the PSS-based NN and PSS-based DANN for the ETHZ shape classes dataset.

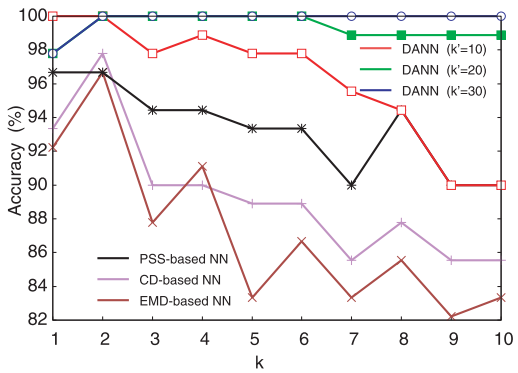


Fig. 8 Prediction accuracies of the PSS-based NN and PSS-based DANN for the skewed line drawings dataset.

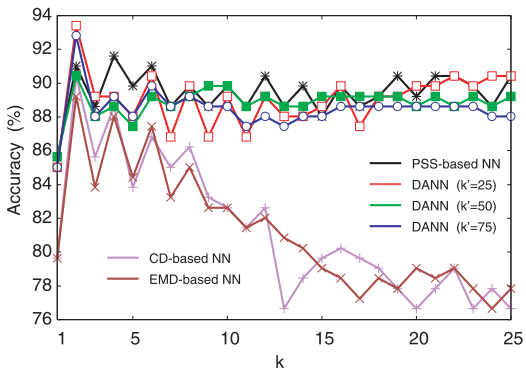


Fig. 9 Prediction accuracies of the PSS-based NN and PSS-based DANN for the great ape skulls dataset.

Table 1 Parameter settings for the OSS- and PSS-based DANN methods.

dataset \ method	OSS-based DANN	PSS-based DANN
ETHZ shape classes	$\kappa = 20$ $k' = 30, 60, 90$	$\kappa = 0.001$
skewed line drawings	$\kappa = 100$ $k' = 10, 20, 30$	$\kappa = 0.0005$
great ape skulls	$\kappa = 5000$ $k' = 25, 50, 75$	$\kappa = 0.025$

Table 2 Runtime (s).

dataset \ method	CD-based NN	EMD-based NN	OSS-based NN	PSS-based NN	OSS-based DANN	PSS-based DANN
ETHZ shape classes	553.2	29118.8	179.8	175.3	134.2 ($k' = 30$) 382.2 ($k' = 60$) 634.6 ($k' = 90$)	153.4 ($k' = 30$) 370.3 ($k' = 60$) 588.7 ($k' = 90$)
skewed line drawings	9.8	39.8	6.5	6.8	11.2 ($k' = 10$) 28.1 ($k' = 20$) 44.8 ($k' = 30$)	12.1 ($k' = 10$) 26.8 ($k' = 20$) 44.0 ($k' = 30$)
great ape skulls	307.7	3714.8	55.6	55.1	79.4 ($k' = 25$) 213.0 ($k' = 50$) 368.1 ($k' = 75$)	58.8 ($k' = 25$) 160.0 ($k' = 50$) 306.3 ($k' = 75$)

Table 3 Peaks of prediction accuracy (%).

dataset \ method	CD-based NN	EMD-based NN	OSS-based NN	PSS-based NN	OSS-based DANN	PSS-based DANN
ETHZ shape classes	87.1 (k = 2) –	90.0 (k = 2) –	98.5 (k = 2, 4, 5, 6) –	96.3 (k = 2) –	**100** (k = 1, 2, 4, ..., 12) (k' = 90)	97.8 (k = 2, 4) (k' = 30)
skewed line drawings	97.8 (k = 2) –	96.7 (k = 2) –	**100** (k = 2) –	96.7 (k = 1, 2) –	**100** (k = 4, 5, 6) (k' = 10)	**100** (k = 2, ..., 10) (k' = 30)
great ape skulls	90.4 (k = 2) –	89.2 (k = 2) –	86.2 (k = 2) –	91.6 (k = 4) –	88.6 (k = 2) (k' = 50)	**93.4** (k = 2) (k' = 25)

optimized distance function in the OSS- and PSS-based DANN methods exceeded the limit of performance of the OSS- and PSS-based NN methods, respectively. More importantly, from the figures, on average, the optimized distance function seemed to be effective for prediction for most k. In fact, it yielded such an additional merit that we did not have to be accurate in the choice of k. Regarding the great ape skulls dataset, because the scale of the skull was important for the prediction, the peak of accuracy for the OSS-based NN method was lower than the peaks for the CD- and EMD-based NN methods. For this dataset, the PSS was more suitable than the OSS. Indeed, from Fig. 9 and Table 3, we observe that the PSS-based NN method worked better than the CD- and EMD-based NN methods. Additionally, the PSS-based DANN method was even better than these NN methods. Thus, it is worth performing a trial using the OSS- and PSS-based DANN methods for shape classification, even though their computational costs are higher than those of the OSS- and PSS-based NN methods, respectively.

5. Conclusion

We developed the DANN method to make the OSS (or PSS) and optimization of distance function available for the configuration matrices of shapes. The mixture model and EM algorithm based on the OSS or PSS were applied to the optimization. Moreover, we introduced several techniques that are helpful for the initialization of model parameters in the EM algorithm. Using the datasets on shapes, we demonstrated that the optimization in the OSS- and PSS-based methods were effective in terms of shape classification.

Including an online EM algorithm, such as [25], [26], in the OSS- and PSS-based DANN methods is an interesting area of future study. Generally, compared with a batch EM algorithm, the online version reaps the benefits of frequent updates and fast adaptation in the early stage of parameter estimation [27]. It is more suitable in some shape classification tasks. Additionally, there are a number of machine learning approaches for the optimization of the distance function. As stated in (22), we used the EM algorithm to estimate the probability densities of the distance function, but the EM algorithm is not the only estimation method available and may be replaced with others. Accordingly, seeking an efficient optimization approach is also another area of future study. Moreover, classification problems for the shapes of three dimensions or more would pose a new challenge.

Acknowledgments

We thank Riku Hatanaka of Hiroshima City University for his support with the experiments. This work was supported in part by KAKENHI-KAKUTOKU-SHIENHI of Hiroshima City University. We thank Maxine Garcia, PhD, from Edanz Group (`https://en-author-services.edanzgroup.com/ac`) for editing a draft of this manuscript.

References

[1] C.M. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, New York, 1995.
[2] T.M. Cover and J.A. Thomas, Elements of Information Theory, 2nd ed., John Wiley & Sons, Inc., New York, 2006.
[3] G. James, D. Witten, T. Hastie, and R. Tibshirani, An Introduction to Statistical Learning with Applications in R, Springer Texts in Statistics, Springer-Verlag, New York, 2013.
[4] T. Hastie and R. Tibshirani, "Discriminant adaptive nearest neighbor classification," IEEE Trans. Pattern Anal. Mach. Intell., vol.18, no.6, pp.607–616, June 1996.
[5] B. Kulis, "Metric learning: A survey," Foundations and Trends in Machine Learning, vol.5, no.4, pp.287–364, July 2013.
[6] M.L. Zelditch, D.L. Swiderski, and H.D. Sheets, Geometric Morphometrics for Biologists: A Primer, 2nd ed., Elsevier Academic Press, Amsterdam, 2012.
[7] I.L. Dryden and K.V. Mardia, Statistical Shape Analysis with Applications in R, 2nd ed., Wiley Series in Probability and Statistics, Wiley, Chichester, UK, 2016.
[8] L. da Fontoura Costa and R.M. Cesar Jr., Shape Analysis and Classification: Theory and Practice, CRC Press, Boca Raton, FL, 2001.
[9] G. McNeill and S. Vijayakumar, "2D shape classification and retrieval," Proc. Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, pp.1483–1488, IJCAI, Aug. 2005.
[10] M.G. Kendall, A course in the geometry of n dimensions, Wiley Series in Probability and Statistics, Dover Publications, Mineola, NY, 2004.
[11] J.T. Kent and K.V. Mardia, "Shape, procrustes tangent projections and bilateral symmetry," Biometrika, vol.88, no.2, pp.469–485, June 2001.
[12] K. Iwata, "Shape clustering as a type of procrustes analysis," Proc. 25th International Conference on Neural Information Processing, ed. L. Cheng, A. Leung, and S. Ozawa, Lecture Notes in Computer Science, vol.11304, Siem Reap, Cambodia, pp.218–227, Asia Pacific Neural Network Society, Springer, Dec. 2018.

[13] K. Iwata, "EM algorithm for mixture models in shape analysis," Technical Report IBISML2019-33, IEICE, Tokyo, Japan, March 2020. vol.119, no.476, pp.1–7 (in Japanese).
[14] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, Introduction to Algorithms, 3rd ed., MIT Press, Cambridge, MA, July 2009.
[15] G.H. Golub and C.F.V. Loan, Matrix Computations, 4th ed., Johns Hopkins Studies in the Mathematical Sciences, Johns Hopkins University Press, Baltimore, Maryland, 2013.
[16] J.W. Demmel, Applied Numerical Linear Algebra, SIAM, Philadelphia, PA, 1997.
[17] H. Fan, H. Su, and L. Guibas, "A point set generation network for 3D object reconstruction from a single image," Proc. 2017 IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, pp.2463–2471, IEEE Computer Society, IEEE, July 2017.
[18] Y. Rubner, C. Tomasi, and L.J. Guibas, "The earth mover's distance as a metric for image retrieval," International Journal of Computer Vision, vol.40, no.2, pp.99–121, Nov. 2000.
[19] V. Ferrari, "ETHZ shape classes." http://www.vision.ee.ethz.ch/datasets/index.en.html, Sept. 2009.
[20] V. Ferrari, F. Jurie, and C. Schmid, "From images to shape models for object detection," International Journal of Computer Vision, vol.87, no.3, pp.284–303, May 2010.
[21] I.L. Dryden, "shapes package." http://www.R-project.org, 2018. Contributed package, Version 1.2.4.
[22] T. Okamoto and K. Iwata, "Data set of skewed line drawings." http://www.prl.info.hiroshima-cu.ac.jp/kiwata/okamotoiwata/, Sept. 2017.
[23] T. Okamoto, K. Iwata, and N. Suematsu, "Extending the full Procrustes distance to anisotropic scale in shape analysis," Proc. 4th IAPR Asian Conference on Pattern Recognition, Nanjing, Jiangsu, China, pp.634–639, IAPR, IEEE, Nov. 2017.
[24] P. O'Higgins and I.L. Dryden, "Sexual dimorphism in hominoids: further studies of craniofacial shape differences in pan, gorilla and pongo," Journal of Human Evolution, vol.24, no.3, pp.183–205, March 1993.
[25] R.M. Neal and G.E. Hinton, "A view of the EM algorithm that justifies incremental, sparse, and other variants," in Learning in Graphical Models, ed. M.I. Jordan, NATO ASI Series, vol.89, pp.355–368, Kluwer Academic Publishers, 1998.
[26] O. Cappé and E. Moulines, "On-line expectation-maximization algorithm for latent data models," Journal of the Royal Statistical Society: Series B (Statistical Methodology), vol.71, no.3, pp.593–613, June 2009. Corrigendum is given in vol.73, no.5, p.792.
[27] P. Liang and D. Klein, "Online EM for unsupervised models," Proc. 2009 Annual Conference of the North American Chapter of the ACL: Human Language Technologies, Boulder, Colorado, pp.611–619, Association for Computational Linguistics, June 2009.

Kazunori Iwata received BE and ME degrees from Nagoya Institute of Technology, Aichi, Japan, in 2000 and 2002, respectively, and a PhD degree in informatics from Kyoto University, Kyoto, Japan, in 2005. He was a JSPS research fellow from April 2002 to March 2005. He has been with Hiroshima City University, Hiroshima, Japan, since April 2005. His research interests include machine learning, pattern recognition, and information theory. He was an Associate Editor of IEICE Transactions on Information and Systems from July 2013 to June 2017.