A Learning Algorithm of Boosting Kernel Discriminant Analysis for Pattern Recognition

Kita, Shinji
Ozawa, Seiichi
Maekawa, Satoshi
Abe, Shigeo

PAPER

A Learning Algorithm of Boosting Kernel Discriminant Analysis for Pattern Recognition

Shinji KITA[†], ***Nonmember***, **Seiichi OZAWA**[†a)], **Satoshi MAEKAWA**[††], ***Members***, ***and*** **Shigeo ABE**[†], ***Nonmember***

SUMMARY In this paper, we present a new method to enhance classification performance of a multiple classifier system by combining a boosting technique called AdaBoost.M2 and Kernel Discriminant Analysis (KDA). To reduce the dependency between classifier outputs and to speed up the learning, each classifier is trained in a different feature space, which is obtained by applying KDA to a small set of hard-to-classify training samples. The training of the system is conducted based on AdaBoost.M2, and the classifiers are implemented by Radial Basis Function networks. To perform KDA at every boosting round in a realistic time scale, a new kernel selection method based on the class separability measure is proposed. Furthermore, a new criterion of the training convergence is also proposed to acquire good classification performance with fewer boosting rounds. To evaluate the proposed method, several experiments are carried out using standard evaluation datasets. The experimental results demonstrate that the proposed method can select an optimal kernel parameter more efficiently than the conventional cross-validation method, and that the training of boosting classifiers is terminated with a fairly small number of rounds to attain good classification accuracy. For multi-class classification problems, the proposed method outperforms both Boosting Linear Discriminant Analysis (BLDA) and Radial-Basis Function Network (RBFN) with regard to the classification accuracy. On the other hand, the performance evaluation for 2-class problems shows that the advantage of the proposed BKDA against BLDA and RBFN depends on the datasets.

key words: *boosting, kernel methods, kernel discriminant analysis, pattern classification, neural networks, feature selection*

1. Introduction

Recently, boosting has been widely known as a powerful method to construct a strong classifier by combining several weak classifiers [7], each of which performs slightly better than random guessing. It is demonstrated that boosting can enhance the overall performance by increasing the so-called margin, which is defined as the distance of the closest training sample to the decision surface of a classifier [4], [26]. In addition, Murua [23] reveals that the upper error bound is represented by not only the margin but also the dependence of the classifier outputs. In the boosting algorithms, this classifier dependence is kept low by providing training samples with different weights that are increased based on the difficulty in classification [8].

Although the boosting can construct accurate and robust classification systems, there still remain some open questions. One question is how we can determine the number of boosting rounds properly (i.e., the criterion of the training convergence) [16], and another is how a weak classifier should be created [17]. On the first question, the training convergence has often been determined by the training error or the training has been terminated at a predetermined round. However, these methods are not always suitable to determine the learning convergence because it has been known that a test error can be decreased even after the training error converges to zero [25], and that a proper boosting round generally depends on datasets. Therefore, we still need to find another valid criterion leading to the desired generalization performance with less computation costs. On the second question, Lu et al. [18] have proposed an interesting approach in which at each boosting round a new Linear Discriminant Analysis (LDA) is performed to construct a low-dimensional feature space called *LDA subspace* by focusing on hard-to-separate training samples, and the features projected to this LDA subspace are trained by a weak learner model. In this approach, low dependency of week learners' outputs is promoted by the variety of LDA subspaces constructed from different training sets.

On the other hand, the kernel methods have also been widely known as a powerful approach to solving difficult classification tasks [1], [22], [28]. The advantage of the kernel methods originates from the nonlinear mapping of an input space to a high-dimensional feature space. If a proper kernel function is selected, the subjects belonging to the same class can be separated from the others completely, and the class separability is maximized in the feature space. However, the classification using such a high-dimensional feature space often suffers from noise and outliers; thus some dimensional reduction techniques (i.e., feature extraction) such as Kernel Principal Component Analysis [1], [27] and Kernel Discriminant Analysis (KDA) have been widely used [2], [20]. KDA is a promising method of feature extraction in the sense that the class separability for any data distribution can be largely enhanced in the feature space.

We [14] previously extended Lu's work above [18] by adopting KDA instead of LDA as the subspace learning. For notational convenience, we call this approach "Boosting Kernel Discriminant Analysis (BKDA)." In the proposed BKDA, to satisfy the weak learner condition, a small number of training samples are selected based on a weight function and they are applied to KDA. Training such a small training set also reduces the computation costs of KDA

Manuscript received June 6, 2007.

[†]The authors are with Graduate School of Science and Technology, Kobe University, Kobe-shi, 657–8501 Japan.

[††]The author is with National Institute of Information and Communications Technology, Kyoto-fu, 619–0289 Japan.

a) E-mail: ozawasei@kobe-u.ac.jp

DOI: 10.1093/ietisy/e90–d.11.1853

within a feasible range even when many classifiers are created. However, it is well known that the performance of the kernel methods generally depends on the selection of a kernel function [12]. In the conventional methods, a proper kernel function is often searched based on the cross-validation method in which the classifier performance is estimated for all possible candidates of kernel functions. Needless to say, such a cross-validation method can impose immense computation costs to the proposed BKDA because the feature extraction by KDA and the training of classifiers must be carried out for every candidate of kernel functions.

In this paper, we improve our prior work [14] in the following points: (1) the kernel selection for KDA is carried out within a feasible time, (2) the criterion of the training convergence is defined so as to attain a desired performance with fewer boosting rounds. As for the first point, we evaluate the kernel function by measuring the class separability defined by the proportion of between-class scatter to within-class scatter; then a proper kernel parameter is searched within a certain range so as to maximize the separability. As for the second point, the training convergence is judged by the convergence of the minimum margins of classifiers. It is expected that these two points allow the proposed BKDA method to be more effective with regard to not only the training time but also the classification accuracy.

The rest of the paper is organized as follows. Section 2 gives a brief review of the two primary techniques used in the proposed method: Kernel Discriminant Analysis (KDA) and AdaBoost.M2. In Sect. 3, the basic idea of the proposed Boosting KDA (BKDA) is first presented, then the selection method of kernel functions and the criterion of the training convergence are proposed. In Sect. 4, the proposed BKDA is evaluated using several standard datasets through the comparisons with Boosting Linear Discriminant Analysis and Radial-Basis Function Network. Finally, conclusions and further research directions are addressed in Sect. 5.

2. Theoretical Background

As described in Sect. 1, we propose a novel learning algorithm for a multiple classifier system combining the following two methods: Kernel Discriminant Analysis (KDA) [2] and AdaBoost.M2 [8]. KDA carries out the feature extraction, while AdaBoost.M2 performs the training of a multiple classifier system in which a Radial Basis Function (RBF) network [19], [24] is adopted as a weak classifier. Before describing the proposed method, let us review these methods briefly.

2.1 Kernel Discriminant Analysis (KDA)

KDA [2], which is a nonlinear extension of Linear Discriminant Analysis (LDA), is a well-known technique that constructs a subspace where the class separability is maximized in a high-dimensional feature space. Suppose that training samples are given as a set $\{(\boldsymbol{x}_{ij}, y_{ij})_{j=1}^{C_i}\}_{i=1}^{C}$ where $\boldsymbol{x}_{ij}$ is the jth training sample of the ith class represented by an I dimensional column vector, y_{ij} is the class label of $\boldsymbol{x}_{ij}$, C and C_i $(i = 1, \ldots, C)$ are the number of classes and the number of training samples belonging to class i, respectively. Let $N = \sum_i C_i$ be the total number of training samples. Assume that observations have zero means in the feature space (if we cannot assume it, zero means observations can be obtained by adjusting a *kernel matrix* [27]). The inputs are mapped into a high dimensional feature space through a nonlinear mapping function $\phi : \boldsymbol{R}^I \rightarrow F$, where F is the feature space. A between-class scatter matrix $\boldsymbol{B}$ and a within-class scatter matrix $\boldsymbol{W}$ in F are defined as follows:

$$\boldsymbol{B} = \frac{1}{N}\sum_{i=1}^{C} C_i \tilde{\boldsymbol{m}}_i \tilde{\boldsymbol{m}}_i' \tag{1}$$

$$\boldsymbol{V} = \frac{1}{N}\sum_{i=1}^{C}\sum_{j=1}^{C_i}\left(\phi(\boldsymbol{x}_{ij}) - \tilde{\boldsymbol{m}}_i\right)\left(\phi(\boldsymbol{x}_{ij}) - \tilde{\boldsymbol{m}}_i\right)' \tag{2}$$

where $\tilde{\boldsymbol{m}}_i = \frac{1}{C_i}\sum_{j=1}^{C_i}\phi(\boldsymbol{x}_{ij})$ is the mean vector of class i in the feature space, and $'$ means the transpose of a vector or a matrix. To distinguish between two vectors in the input space and the high-dimensional feature space, we denote a vector in the feature space with tilde. The eigenvectors $\tilde{\boldsymbol{u}}$ spanning a subspace of F are obtained by solving the following eigenvalue problem:

$$\lambda \boldsymbol{V}\tilde{\boldsymbol{u}} = \boldsymbol{B}\tilde{\boldsymbol{u}} \tag{3}$$

where λ is the eigenvalue of $\tilde{\boldsymbol{u}}$. However, this eigenproblem is not solved directly to obtain $\tilde{\boldsymbol{u}}$ because the dimensionality of a feature space is usually extremely high or infinite.

To avoid the direct calculation in the feature space, so-called *kernel trick* is applied to Eq. (3). Since $\tilde{\boldsymbol{u}}$ is given by the linear combination of $\phi(\boldsymbol{x}_{ij})$, there exist coefficients α_{ij} $(i = 1, \ldots, C;\ j = 1, \ldots, C_i)$ such that the following relation holds:

$$\tilde{\boldsymbol{u}} = \sum_{i=1}^{C}\sum_{j=1}^{C_i}\alpha_{ij}\phi(\boldsymbol{x}_{ij}). \tag{4}$$

Substituting Eqs. (1), (2), and (4) into Eq. (3), the following eigenproblem is obtained:

$$\lambda \boldsymbol{K}\boldsymbol{K}\boldsymbol{\alpha} = \boldsymbol{K}\boldsymbol{W}\boldsymbol{K}\boldsymbol{\alpha} \tag{5}$$

where $\boldsymbol{\alpha} = [\alpha_{11}, \cdots, \alpha_{1C_1}, \cdots, \alpha_{C1}, \cdots, \alpha_{CC_C}]' \in \mathcal{R}^{N\times 1}$ and

$$\boldsymbol{W} = \begin{bmatrix} \boldsymbol{W}_1 & \cdots & \boldsymbol{0} \\ \vdots & \ddots & \vdots \\ \boldsymbol{0} & \cdots & \boldsymbol{W}_C \end{bmatrix} \in \mathcal{R}^{N\times N}.$$

Here, $\boldsymbol{W}_j$ $(j = 1, \cdots, C)$ is a $C_j \times C_j$ matrix whose diagonal elements are $1/C_j$. $\boldsymbol{K} \in \mathcal{R}^{N\times N}$ in Eq. (5) is called *kernel matrix*, and it is defined as

$$\boldsymbol{K} = \begin{bmatrix} \boldsymbol{K}_{11} & \cdots & \boldsymbol{K}_{1C} \\ \vdots & \ddots & \vdots \\ \boldsymbol{K}_{C1} & \cdots & \boldsymbol{K}_{CC} \end{bmatrix} \tag{6}$$

where

$$\boldsymbol{K}_{ii'} = \begin{bmatrix} k(\boldsymbol{x}_{i1}, \boldsymbol{x}_{i'1}) & \cdots & k(\boldsymbol{x}_{i1}, \boldsymbol{x}_{i'C_{i'}}) \\ \vdots & \ddots & \vdots \\ k(\boldsymbol{x}_{iC_i}, \boldsymbol{x}_{i'1}) & \cdots & k(\boldsymbol{x}_{iC_i}, \boldsymbol{x}_{i'C_{i'}}) \end{bmatrix} = \begin{bmatrix} \phi(\boldsymbol{x}_{i1})'\phi(\boldsymbol{x}_{i'1}) & \cdots & \phi(\boldsymbol{x}_{i1})'\phi(\boldsymbol{x}_{i'C_{i'}}) \\ \vdots & \ddots & \vdots \\ \phi(\boldsymbol{x}_{iC_i})'\phi(\boldsymbol{x}_{i'1}) & \cdots & \phi(\boldsymbol{x}_{iC_i})'\phi(\boldsymbol{x}_{i'C_{i'}}) \end{bmatrix} \in \mathcal{R}^{C_i \times C_{i'}}. \tag{7}$$

Let us consider the eigenvector decomposition of $\boldsymbol{K}$: $\boldsymbol{K} = \boldsymbol{PQP}'$ where $\boldsymbol{P} \in \mathcal{R}^{N \times L}$ is the eigenvector matrix of $\boldsymbol{K}$ and $\boldsymbol{Q} \in \mathcal{R}^{L \times L}$ is a diagonal matrix whose elements are non-zero eigenvalue. Here, L is the number of non-zero eigenvalue. In addition, let us define a vector $\boldsymbol{\beta}$ such that $\boldsymbol{\beta} = \boldsymbol{QP}'\boldsymbol{\alpha}$. Then, Eq. (5) is reduced to

$$\lambda\boldsymbol{\beta} = \boldsymbol{P}'\boldsymbol{WP}\boldsymbol{\beta}. \tag{8}$$

$\boldsymbol{\beta}$ is obtained by solving the new eigenproblem in Eq. (8), then $\boldsymbol{\alpha}$ is calculated by

$$\boldsymbol{\alpha} = \boldsymbol{PQ}^{-1}\boldsymbol{\beta}. \tag{9}$$

The feature z of a query input $\boldsymbol{x}$ is given by projecting $\phi(\boldsymbol{x})$ to the eigenvector $\tilde{\boldsymbol{u}}$ in Eq. (4) and it is given by

$$z = \phi'(\boldsymbol{x})\tilde{\boldsymbol{u}} = \sum_{i=1}^{C} \sum_{j=1}^{C_i} \alpha_{ij} k(\boldsymbol{x}, \boldsymbol{x}_{ij}). \tag{10}$$

Note that the feature vector $\boldsymbol{z} = (z_1, \cdots, z_L)'$ is obtained without calculating $\boldsymbol{B}$, $\boldsymbol{V}$, $\phi(\boldsymbol{x})$, and $\tilde{\boldsymbol{u}}$ explicitly.

2.2 AdaBoost.M2

AdaBoost is a powerful boosting algorithm developed by Freund and Schapire [8] in which the performance is boosted by the ensemble of weak learners whose accuracy is slightly better than random guessing. A week learner is constructed so as to focus on the data that are hard to discriminate. For this purpose, each training sample is provided with the weight function $D(i)$ $(i = 1, \cdots, N)$ which is increased when the ith training sample is misclassified, and the week learner is trained so as to minimize the error weighted by $D(i)$.

To extend AdaBoost to multi-class classification problems, the following two points are modified in AdaBoost.M2. First, the output of a weak hypothesis is represented by a vector of $[0, 1]^k$ where k is the number of classes. Second, the weight $D(i)$ of every sample $(\boldsymbol{x}_i, y_i)$ $(i = 1, \cdots, N)$ is distributed to the incorrect class labels with the rate of $q(i, y \neq y_i)$. By manipulating the weight function $D(i)$ and the label weight function $q(i, y)$, Ada-Boost.M2 allows a weak learner to focus not only on hard-to-classify samples but also on the incorrect class labels. Thus, the pseudo-loss $\varepsilon_q(h, i)$ of the hypothesis h on the training sample i weighted by $q(i, y)$ is defined as follows:

$$\varepsilon_q(h, i) = \frac{1}{2}\Big(1 - h(\boldsymbol{x}_i, y_i) + \sum_{y \neq y_i} q(i, y) h(\boldsymbol{x}_i, y)\Big) \tag{11}$$

where $\sum_{y \neq y_i} q(i, y) = 1$. The weak learner is trained such that the total pseudo-loss over all the N training samples is minimized, and the weight functions $D(i)$ and $q(i, y)$ are updated based on this pseudo-loss. The detail algorithm of AdaBoost.M2 is shown in [8].

3. Boosting Kernel Discriminant Analysis

3.1 Basic Idea

In this section, a novel approach to combining KDA and AdaBoost.M2 called *Boosting KDA* (BKDA) is described. A straightforward way to combine these two methods is that when a weak hypothesis is constructed, all training samples weighted by the weight function $D(i)$ are applied to KDA and a classifier is trained using the extracted eigen-features. More concretely, the kernel matrix is first obtained from Eq. (6), the eigenvalue problem in Eq. (8) is solved, and the coefficient vector $\boldsymbol{\alpha}$ is calculated based on Eq. (9). Then, the feature vector z of the training sample $\boldsymbol{x}_{ij}$ is calculated by projecting $\boldsymbol{x}_{ij}$ to an eigenvector $\tilde{\boldsymbol{u}}$ (see Eq. (10)). Finally the feature vector is applied to AdaBoost.M2 to train an individual classifier. Note that the above operation is carried out for all classifiers. A drawback of this method is that KDA must be carried out at every boosting round using all training samples; thus this method can cause immense computations especially when a hard classification problem, which generally requires a large number of classifiers, is provided to the classifier system.

To overcome this problem, we adopt the following practical approach to training classifiers: a part of training samples are selected based on $D_t(i)$, then they are applied to KDA to acquire a feature vector which is utilized for training the classifier. It is expected that this method has the following properties:

1. The useful features are extracted such that the class separability of hard-to-classify training samples are maximized.
2. The diversity of classifiers is increased because they are trained with different training sets selected based on $D_t(i)$.
3. The computational costs to solve an eigenproblem in KDA are reduced.
4. The computational costs of the kernel selection are reduced.

The first property enables individual classifiers to optimize their input features such that the features are linearly separable as much as possible. The second property can contribute to lowering the dependency of classifiers' outputs [18], and it is expected that this also leads to good classification performance. In addition, the third and fourth properties give us a practical implementation to combine two powerful techniques from the computational point of view.

The primary novelty of this paper is claimed on a practical and effective implementation for the combination of KDA and AdaBoost.M2. In addition, we present a new kernel selection method and a new criterion of the training convergence in order to make Boosting KDA more efficient and robust. These two methods will be described below.

3.2 Kernel Selection

When the Gaussian function given by

$$k(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|^2}{\sigma^2}\right). \tag{12}$$

is adopted as a kernel function, an optimal parameter σ is often obtained by using the Cross-Validation (CV) method in which the classification accuracy for a validation dataset is evaluated. Although several attempts to reduce the computation costs in the CV procedure have been made [5], [9], [15], they still need to evaluate the classification performance. In the proposed Boosting KDA, to test the performance, not only the training but also feature extraction must be carried out in all classifiers. Therefore, the kernel selection based on CV is not practical in terms of the computational costs because both KDA and the classifier training should be conducted for all the candidates of the parameter σ at every boosting round in AdaBoost.M2.

To overcome this problem, we propose a novel kernel selection method in which the search of the kernel parameter in Eq. (12) is not carried out based on the classification results, instead it is carried out based on the following class separability measure S of kernel features:

$$S = \frac{\mathrm{tr}(\boldsymbol{B})}{\mathrm{tr}(\boldsymbol{V})} \tag{13}$$

where $\mathrm{tr}(\cdot)$ is the trace of a matrix, $\boldsymbol{B}$ and $\boldsymbol{V}$ are the between-class scatter matrix and the within-class scatter matrix given by Eqs. (1) and (2), respectively. This separability measure [†] is the same as used in KDA, and an optimal kernel parameter σ is searched such that S is maximized. Since S includes only the calculation of kernel functions, the kernel selection based on S do not need to carry out both the classifier training and its performance evaluation; therefore, it is expected that the computation costs is greatly reduced.

When searching for an optimal parameter σ, we should select it to be neither too small nor too large in order to attain good classification performance. If σ is too small, the value of $k(\boldsymbol{x}, \boldsymbol{x}')$ is close to zero unless $\boldsymbol{x}$ and $\boldsymbol{x}'$ are almost identical. This means that almost all training samples in the high-dimensional feature space are orthogonal to each other; hence, an over-redundant feature space could be created, and this feature space generally causes the deterioration in the generalization performance. On the other hand, if σ is too large, most training samples in the high-dimensional feature space are confined in a very small region because $k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \phi(\boldsymbol{x}_1)'\phi(\boldsymbol{x}_2)$ has a similar value for any input pair $(\boldsymbol{x}_1, \boldsymbol{x}_2)$. Then, this causes degeneracy of the kernel matrix because many eigenvalue have almost zero, and this will lead to the creation of an uninformative feature space. To avoid such an ill-posed situation, we should select the kernel parameter σ such that the training samples are distributed over a feature space with appropriate dimensions. Our preliminary experiment suggests that σ should not be less than 10^{-4} and should not be larger than 10^{2} [††]. Next, let us explain how to find an optimal σ within this range.

The distribution of training samples can be quantified by the trace of the within-class scatter $\mathrm{tr}(\boldsymbol{V})$, and it is represented by the kernel function as follows:

$$\begin{aligned}\mathrm{tr}(\boldsymbol{V}) &= \frac{1}{N}\sum_{i=1}^{C}\sum_{j=1}^{C_i}\left\{\phi(\boldsymbol{x}_{ij}) - \boldsymbol{m}_i\right\}^T\left\{\phi(\boldsymbol{x}_{ij}) - \boldsymbol{m}_i\right\} \\ &= \frac{1}{N}\sum_{i=1}^{C}\left(C_i - \frac{1}{C_i}\sum_{j=1}^{C_i}\sum_{k=1}^{C_i} k(\boldsymbol{x}_{ij}, \boldsymbol{x}_{ik})\right)\end{aligned} \tag{14}$$

where $\boldsymbol{m}_i$ is the mean vector of class i training samples. If the Gaussian kernel in Eq. (12) is adopted, $k(\boldsymbol{x}_{ij}, \boldsymbol{x}_{ik})$ for two different vectors $\boldsymbol{x}_{ij}$ and $\boldsymbol{x}_{ik}$ monotonously increases from 0 to 1 as the kernel parameter σ increases from 0 to infinity. Thus, $\mathrm{tr}(\boldsymbol{V})$ in Eq. (14) has a larger value as σ becomes smaller. Considering that the range of σ is assumed to be $10^{-4} < \sigma < 10^{2}$, to avoid selecting an inappropriately small σ, we define the following empirical upper bound for the within-class scatter $\mathrm{tr}(\boldsymbol{V})$:

$$\mathrm{tr}(\boldsymbol{V}) < \mathrm{tr}(\boldsymbol{V})|_{\sigma=10^{-4}}. \tag{15}$$

This upper bound will be used for narrowing the search region of σ in the proposed kernel selection algorithm described later. Therefore, the final form of the proposed kernel selection problem is described as follows:

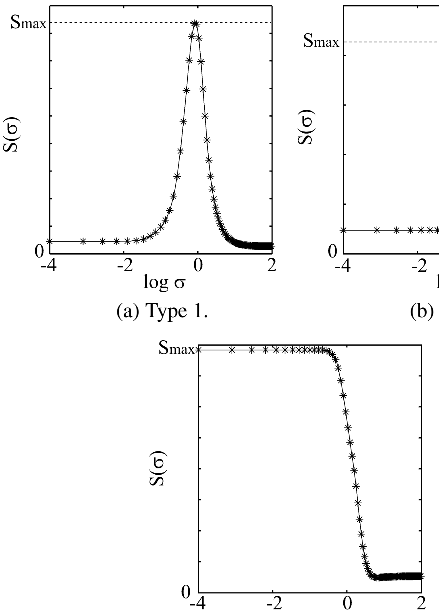
$$\begin{aligned}&\max_{\sigma} S = \frac{\mathrm{tr}(\boldsymbol{B})}{\mathrm{tr}(\boldsymbol{V})} \\ &\text{s. t.} \quad 10^{-4} < \sigma < 10^{2} \\ &\qquad\quad \mathrm{tr}(\boldsymbol{V}) < \mathrm{tr}(\boldsymbol{V})|_{\sigma=10^{-4}}\end{aligned} \tag{16}$$

Figures 1 (a)-(c) illustrate the three types of σ-S curves assumed in the proposed kernel selection. If a σ-S curve belongs to neither of the three types, we start over the selection of training samples. The reason why we restrict σ-S curves to these three types comes from the instability of obtaining optimal parameters. When a σ-S curve belongs to neither of the three types, no isolated maximum of S is found within a proper range of σ. Although we could select the maximum from the more than two local maxima, we often experienced in the preliminary experiments that the good classification performance was not stably obtained for such a σ.

For Type 1 in Fig. 1 (a), we select the σ which gives $S_{\max}$ in Eq. (16); however, as described above, we should not simply select the σ giving $S_{\max}$ for Type 2 or Type 3.

[†] There are some other criteria for the separability. See [10] for details.

[††] The original data are normalized such that the average is zero and the standard deviation is one.



(a) Type 1.

(b) Type 2.

(c) Type 3.

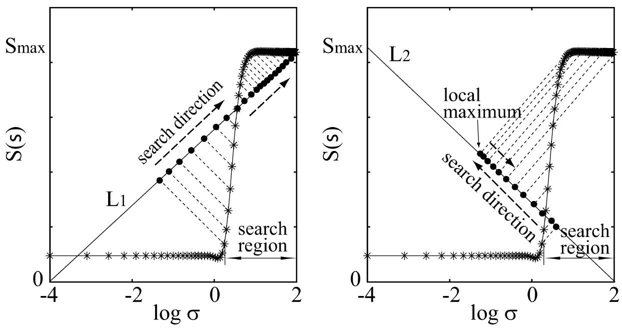
Fig. 1 Three types of σ-S curves.

As long as S is almost equivalent to $S_{\max}$, we should select as small σ as possible for Type 2, and select as large σ as possible for Type 3. In addition, since we have only a set of pairs $(\sigma, S(\sigma))$, we need to check if the σ-S curve is really one of the three types assumed here. Therefore, not only searching for an optimal σ but also checking if a σ-S curve belongs to either of the three types should be conducted by the algorithm itself. To make the algorithm simple, we introduce the following two straight lines with positive and negative slopes[†]:

$$L_1 : S(\sigma_j) = \frac{S_{\max}}{6}(\log \sigma_j + 4) \tag{17}$$

$$L_2 : S(\sigma_j) = -\frac{S_{\max}}{6}(\log \sigma_j - 2). \tag{18}$$

To explain how an optimal σ is searched by using L_1 or L_2, let us take an example of Type 2 in Fig. 1 (b). First, obtain all the pairs $(\sigma, S(\sigma))$ satisfying Eq. (15), and they are defined as a parameter set $\mathcal{P}$. Then, the pairs $(\sigma \in \mathcal{P}, S(\sigma))$ are projected to L_1 one after another from the smallest σ (see Fig. 2 (a)), and search for local maxima of the projected values by checking if an increasing trend of the values turns into a decreasing one. As seen from Fig. 2 (a), no local maximum on L_1 is found in this case. Then, they are projected to L_2 (see Fig. 2 (b)), and search for local maxima on L_2 in the same way. From Fig. 2 (b), we see that a single local maximum on L_2 is found, and that the smallest σ, which gives a little smaller value than the maximum separability $S_{\max}$, is found. In this example, we know that the σ-S curve belongs to Type 2 because a single local maximum is found on L_2. On the other hand, if a single local maximum is found on L_1, it means that the σ-S curve belongs to Type 1 or 3. Otherwise, it belongs to neither of the three types; then, the kernel selection should be started over by selecting different training samples.



(a)

(b)

Fig. 2 Examples of the parameter search using the straight line (a) L_1 and (b) L_2 when the σ-S curve belongs to Type 2.

[Kernel Selection]

1) Select r% of training samples based on the so-called roulette wheel strategy, in which the area of a roulette compartment for the ith training sample $(i = 1, \cdots, N)$ is allocated in proportion to $D_t(i)$.
2) Obtain a set of n candidate parameters σ_j $(j = 1, \cdots, n)$ as follows: $\sigma_j = j^3 \times 10^{-4}$.
3) Calculate $S(\sigma_j)$ for all σ_j based on Eq. (13).
4) Remove all σ_j that satisfy the following inequality: $\mathrm{tr}(\boldsymbol{V}) < \mathrm{tr}(\boldsymbol{V})|_{\sigma_j=10^{-4}}$. Then, put the selected σ_j into the parameter set $\mathcal{P}$, and let the number of $\sigma_j \in \mathcal{P}$ be n'.
5) Project all the points $(\sigma_j, S(\sigma_j))$ $(j = 1, \cdots, n')$ in the parameter set $\mathcal{P}$ to L_1 in Eq. (17).
6) If the projected values on L_1 have a single local maximum, select the σ_j giving the local maximum as an optimal parameter, and terminate this procedure. Otherwise, go to Step 7.
7) Project all the points $(\sigma_j, S(\sigma_j))$ in $\mathcal{P}$ to the following straight line L_2 in Eq. (18).
8) If the projected values on L_2 have a single local maximum, select the σ_j giving the local maximum of S as an optimal parameter, and terminate this procedure. Otherwise, go to Step 9.
9) Go back to Step 1 to redo the kernel selection.

Fig. 3 Algorithm of the proposed kernel selection.

To implement the above parameter search, we propose a heuristic algorithm in Fig. 3. To find an optimal σ, all the points $(\sigma, S(\sigma))$ are once projected to L_1 in Step 5, and a single local maximum is searched in Step 6. If no local maximum on L_1 is found or more than two maxima are found, it means that this σ-S curve belongs to neither Type 1 nor Type 3. Then, go to Step 7, and all the points $(\sigma, S(\sigma))$ are projected to L_2 to find local maxima. If no local maximum on L_2 is found or more than two maxima are found in Step 8, it means that this σ-S curve belongs to neither of the three types we assume here. Then, retry to select different training samples (see Step 9).

Since the calculation of S needs to perform neither KDA nor the classification test at every validation of the parameter σ, it is expected that the computation costs of the

[†] We can choose any non-zero absolute value of the slope for L_1 and L_2.

proposed kernel selection are greatly reduced as compared with the conventional CV-based method.

3.3 Criterion of Training Convergence

Defining an appropriate criterion of the training convergence has always been a controversial topic in the machine learning research. For neural classifiers such as Radial Basis Function (RBF) networks [3], [19], [24], many convergence criteria have been proposed so far based on both empirical and theoretical approaches [11]. However, since these criteria were proposed for a single neural classifier, an appropriate criterion for multiple classifier systems is still unclear. Hence, in many conventional boosting approaches, the training convergence has been judged based on a simple criterion such as training errors, or the training is just terminated when the training reaches to a predetermined round. However, these convergence criteria are sometimes inappropriate because it is known that the test performance can be improved even after the training error converges to zero [25], and because the training of neural networks with a fixed number of steps often causes the so-called overfitting.

Instead of these criteria, we adopt a new measure which is defined based on the *margin* of classifier outputs [26]. The margin $M_t(\boldsymbol{x})$ is defined by

$$M_t(\boldsymbol{x}) = h_t(\boldsymbol{x}, y) - \max_i\{h_t(\boldsymbol{x}, i) | i \in Y, i \neq y\} \tag{19}$$

where $h_t(\boldsymbol{x}, y)$ is the output of the tth classifier for a training sample $\boldsymbol{x}$ and class y, and Y is the set of class labels. There are two facts that we come up with this convergence criterion. The first fact is that the theoretical bound on the generalization errors of linear classifiers is determined by the margin $M_t(\boldsymbol{x})$ [26] for linearly separable problems. The second fact is that the margin $M_t(\boldsymbol{x})$ can increase even after the training error converges to zero [25] in some cases. These two facts allow us to conjecture that the margin can be a better measure of the test error than the training error although the first fact would not be applied to nonlinear classifiers like RBF networks we adopt here.

To define a convergence criterion, let us consider the distribution of the margins for all training samples called *margin distribution graph.* Figure 4 shows an example of the margin distribution graph for the ‘blood cell’ data [1]. The X axis corresponds to the margin $M_t(x) \in [-1, 1]$ and the Y axis means the accumulated number of training samples. Our preliminary experiments demonstrate that the training progress can be monitored more clearly by observing the transition of the minimum margin than that of the maximum margin. Figure 5 shows the time evolutions of the minimum margin, the test error, and the training error for the ‘blood cell’ data. As seen from Fig. 5, even after the training error converges to zero, the test error decreases and the minimum margin increases. In this case, the minimum margin converges at around 240 rounds and the test error also seems to converge at this round. This result reminds us of the second fact addressed above.

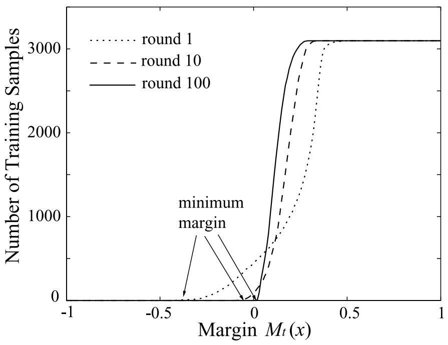


Fig. 4 Margin distribution graph at the rounds 1, 10, and 100 for the ‘blood cell’ dataset.

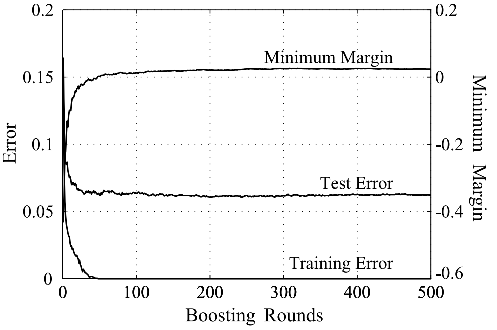


Fig. 5 Time evolutions of minimum margin, training error, and test error for the ‘blood cell’ dataset.

To detect the convergence of the minimum margin, we define the following average temporal variation ΔM_t of the margin $M_t(\boldsymbol{x})$:

$$\Delta M_t = \frac{1}{T_0} \sum_{t=T}^{T+T_0} | \min_j M_{t+1}(\boldsymbol{x}_j) - \min_j M_t(\boldsymbol{x}_j)| \tag{20}$$

where T is the current boosting round and T_0 is a period to average the minimum margin (in the later experiments, T_0 is set to 50). To make sure if the training really converges, the training is conducted while T_0 extra boosting rounds are completed. If the average margin variation ΔM_t becomes smaller than a threshold at $T + T_0$, we judge that the training converged at T. Since classifiers are generally created even during the extra boosting rounds, these created classifiers must be removed from the obtained classifier system. For example, if the convergence condition is satisfied at the 350th round, the training is terminated at the 350th round and the classifiers created from the 301st round to the 350th round are removed.

3.4 The Boosting KDA Algorithms

In Fig. 6, we summarize the proposed Boosting KDA (BKDA) algorithm. Steps 1, 2, 7, 8, 9, 10 correspond to the procedures of AdaBoost.M2, and Step 5 corresponds to the procedure of KDA. The proposed kernel selection (see Fig. 3 for details) and the convergence judgment is conducted at Step 4 and Step 11, respectively.

```
[Boosting KDA Algorithm]
Input
  • Training set $\mathcal{S} = \{(\boldsymbol{x}_{ij}, y_{ij})_{j=1}^{C_i}\}_{i=1}^{C}$ where $\boldsymbol{x}_{ij} \in \boldsymbol{R}^I$ is an $I$-dimensional
    input vector, $y_{ij} \in Y = \{1, \ldots, C\}$ is the class label, $C$ is the number
    of classes, and $C_i$ is the number of training samples in the $i$th class.
  • Weight function of samples $D_0(i, j) = 1/N$ for
    $i = 1, \ldots, C,\ j = 1, \ldots, C_i$
  • Kernel function $k(\boldsymbol{x}, \boldsymbol{x}')$ in Eq. (12).
  • Selection percentage $r$ of training samples.

Initialize
  • Training step: $t = 0$.
  • Weight value: $w_{ij,y}^{1} = D_0(i, j)/(C - 1)$ for
    $i = 1, \ldots, N,\ j = 1, \ldots, C_i,\ y \in Y - \{y_{ij}\}$.

Repeat
  1. Calculate $q_t(i, j, y) = \frac{w_{ij,y}^{t}}{\sum_{y' \neq y_{ij}} w_{ij,y'}^{t}}$ for all $y \in Y - \{y_{ij}\}$.
  2. Update the distribution probability $D_t(i, j) = \frac{W_{ij}^{t}}{\sum_{i=1}^{C} \sum_{j=1}^{C_i} W_{ij}^{t}}$.
  3. Select $r$% of training samples based on the probability $D_t(i, j)$, then
     put them into the training set $\mathcal{R}_t$.
  4. Do kernel selection in Fig. 3 using $\mathcal{R}_t$ and obtain an optimal kernel
     parameter $\sigma_t$.
  5. Carry out the following Kernel Discriminant Analysis:
       a. Calculate the kernel matrix $\boldsymbol{K}$ for $\mathcal{R}_t$ using Eq. (6), and
          decompose $\boldsymbol{K}$ into $\boldsymbol{P} \in \mathcal{R}^{N \times L}$ and $\boldsymbol{Q} \in \mathcal{R}^{L \times L}$ where $L$ is the
          number of non-zero eigenvalue.
       b. Obtain non-zero $\lambda_l$ $(l = 1, \cdots, L)$ and the $\boldsymbol{\beta}_l$ satisfying Eq. (8).
       c. Calculate $\boldsymbol{\alpha}_l$ $(l = 1, \cdots, L)$ based on Eq. (9).
  6. Calculate the feature vectors $\boldsymbol{z}_{ij} = (z_{ij,1}, \cdots, z_{ij,L})'$
     $(i = 1, \cdots, C; j = 1, \cdots, C_i)$ by projecting the training data $\boldsymbol{x}_{ij}$ to the
     KDA subspace spanned by $\{\tilde{\boldsymbol{u}}_1, \cdots, \tilde{\boldsymbol{u}}_L\}$ based on Eq. (10).
  7. Train the $t$th hypothesis $h_t$ (RBF network) using the training set
     $\{(\boldsymbol{z}_{ij}, y_{ij})_{j=1}^{C_i}\}_{i=1}^{C}$.
  8. Calculate the pseudo-loss of $h_t$ :
       $\epsilon_t = \frac{1}{2} \sum_{i=1}^{C} \sum_{j=1}^{C_i} D_t(i, j)$
           $\times [1 - h_t(\boldsymbol{z}_{ij}, y_{ij}) + \sum_{y \neq y_{ij}} q_t(i, j, y) h_t(\boldsymbol{z}_{ij}, y)]$.
  9. Update $\beta_t = \epsilon_t/(1 - \epsilon_t)$.
 10. Update a new weight vector to be
     $w_{ij,y}^{t+1} = w_{ij,y}^{t} \beta_t^{(1/2)(1+h_t(\boldsymbol{z}_{ij}, y_{ij}) - h_t(\boldsymbol{z}_{ij}, y))}$
     for $i = 1, \ldots, C,\ j = 1, \ldots, C_i,\ y \in Y - \{y_{ij}\}$.
 11. Calculate $\Delta M_t$ using Eq. (20). If the convergence condition $\Delta M_t < \kappa$
     ($\kappa$: small const.) is satisfied, terminate this algorithm. Otherwise,
     $t \leftarrow t + 1$ and go back to Step 1.

Output
The final hypothesis: $h_f(\boldsymbol{x}) = \arg\max_{y \in Y} \sum_{t=1}^{T} \left(\log \frac{1}{\beta_t}\right) h_t(\boldsymbol{x}, y)$.
```

Fig. 6 The proposed Boosting KDA algorithm.

4. Experiments

4.1 Experimental Setup

The performance evaluation is carried out using the fourteen datasets in Table 1, which are cited from the three data sources [1], [13], [29]. The first five datasets are multi-class problems, while the other nine datasets are two-class problems. To make a comparison as rigorously as possible in a statistical sense, we generate 100 different pairs of training and test sets from the original dataset, and the average test performance and its statistical significance are evaluated. Since some of the datasets are originally separated into training and test sets, these sets are once merged into a single dataset, and then it is randomly divided into a pair of training and test sets.

Table 1 Evaluation datasets.

	#attrib.	#class	#train.	#test
blood cell [1]	13	12	1197	5000
thyroid 1 [29]	21	3	1200	6000
segmentation [29]	19	7	500	1810
vehicle [29]	18	4	346	500
letter [29]	16	26	2000	18000
banana [13]	2	2	400	4900
breast cancer [13]	9	2	200	77
diabetis [13]	8	2	468	300
german [13]	20	2	700	300
heart [13]	13	2	170	100
image [13]	18	2	1300	1010
ringnorm [13]	20	2	400	7000
splice [13]	60	2	1000	2175
thyroid 2 [13]	5	2	140	75

In the proposed BKDA, an extended Radial Basis Function (RBF) network model [21] is adopted as a classifier due to its simplicity and high-performance although BKDA itself does not restrict the classifier model. In the extended RBF network, the centers of RBF are initialized by the k-means clustering, and the width is determined based on the distance between two nearest centers. In the training, the RBF network is updated by the conjugate gradient descent algorithm to minimize the square error weighted by the sample weights D_t. To obtain weak classifiers, the number of hidden units in every RBF network is restricted to twice as many as the number of classes.

There are several parameters to be set properly in the proposed BKDA. In the following experiments, we set the percentage r of selected training samples to 15 [%]. And the number of candidate parameters n in the kernel selection (see Fig. 3) are set to 100.

4.2 Evaluation of Kernel Selection and Convergence Criterion

In the proposed BKDA, a new method of the kernel selection and a new criterion of the training convergence are proposed. First, let us examine the effectiveness of these methods before evaluating the overall performance of BKDA.

4.2.1 Evaluation of Kernel Selection

As stated in Sect. 3.2, the optimal kernel parameter in KDA is searched by maximizing the separability S of training samples in a feature space. Therefore, the training of a boosting classifier is not actually carried out. Furthermore, only a part of training samples, which are selected based on a distribution probability $D_t(i, j)$, should be applied to KDA. Therefore, it is expected that these two characteristics in the proposed method allow the system to find an optimal kernel parameter much faster than the conventional parameter selection method based on Cross-Validation (CV).

Table 2 shows the computation time needed in the parameter selection when the 10-fold CV and the proposed selection methods are applied to the two-class problems. The training time is measured for each boosting round on a Pentium IV 1.8 GHz personal computer, and the average time is evaluated for the first five rounds. As seen from Table 2, the proposed method attains very fast parameter selection as compared with the 10-fold CV. Note that the computation time shown in Table 2 is required at every boosting round. Thus, from the practical point of view, the CV-based selection method could not be introduced in the proposed BKDA with typical high-performance computers[†].

4.2.2 Evaluation of Training Convergence Criterion

As explained in Sect. 3.3, the training convergence is judged by the convergence of the minimum margin (see Step 11 in Fig. 6). The threshold κ for the average margin variation is set to 0.001. To evaluate the proposed criterion, the test classification accuracy is examined for the Boosting KDA based on two different convergence criteria as well as the proposed BKDA. For notational convenience, the Boosting KDA whose convergence criterion is based on the training error is denoted as $\text{BKDA}_{\text{train}}$, and the Boosting KDA whose training is terminated at the 200th round is denoted as BKDA_{200}.

Table 3 shows the test classification performance and the convergence rounds for the three types of Boosting KDA. 'N/A' in the column of $\text{BKDA}_{\text{train}}$ means that the training was not converged within 2,000 rounds. As seen from Table 3, the test performance of BKDA is better than or almost equal to that of BKDA_{200} although BKDA needs more rounds than 200 for 'image' dataset. This result implies that it is difficult to determine a proper convergence round in advance. On the other hand, comparing between BKDA and $\text{BKDA}_{\text{train}}$, the proposed BKDA outperforms $\text{BKDA}_{\text{train}}$ in many cases; in addition, the required boosting rounds in $\text{BKDA}_{\text{train}}$ largely depend on the training datasets and sometimes it could not converge within 2,000 rounds for the 'breast cancer' dataset.

Table 2 Average training time (sec.) of a single boosting round in the two kernel selection methods: the proposed kernel selection based on the separability measure and the kernel selection based on the 10-fold CV.

	banana	breast cancer	diabetis	image
Proposed	0.5	0.55	0.7	0.66
CV-based	142	143	148	132

Table 3 Test classification accuracy [%] of the three types of Boosting KDA. The value in the parentheses ( ) means the boosting round needed for the convergence. 'N/A' means that the learning was not converged within 2,000 rounds.

	BKDA	$\text{BKDA}_{\text{train}}$	BKDA_{200}
blood cell	93.7 (99)	93.7 (48)	93.9
thyroid 1	97.8 (46)	97.9 (694)	97.9
segmentation	93.5 (83)	92.2 (5)	93.4
banana	86.7 (83)	87.1 (113)	86.7
breast cancer	67.5 (38)	N/A	66.2
image	98.4 (295)	97.5 (37)	98.3
thyroid 2	98.7 (133)	96.0 (3)	97.3

The above results support that the minimum margin gives valid information on the training convergence to attain a good test performance.

4.3 Performance Evaluation of Boosting KDA

The proposed BKDA is composed of the following two essential methods: kernel discriminant analysis and boosting. Hence, the comparison should be made with a non-kernel approach and a non-multiple classifier model. For this purpose, the test classification accuracy of the proposed BKDA is investigated through the comparison with the following two well-established methods: Boosting Linear Discriminant Analysis (BLDA) and Radial-Basis Function Network (RBFN). In BLDA, the feature extraction is carried out based on the conventional LDA algorithm instead of KDA in the proposed BKDA. In order to focus on the effectiveness of using the kernel method, the same convergence criterion is adopted in both BKDA and BLDA. On the other hand, for the latter, a single strong classifier model is constructed by RBFN whose input features are extracted by KDA.

Tables 4 (a) and (b) show the test classification accuracy [%] of BKDA, BLDA, and RBFN for the multi-class problems and the two-class problems, respectively. As mentioned in Sect. 4.1, the classification accuracy is averaged over 100 evaluations (the value after '±' means the standard deviation). To test the statistical significance between the average performances of the proposed BKDA and the two competitive methods (i.e., BLDA and RBFN), we perform the Wilcoxon signed-rank test. The asterisk after the values means that there is 1% level of significance for the average accuracy between BKDA and the two methods.

From the results in Table 4 (a), BKDA has significant improvement against both BLDA and RBFN for the four datasets out of five (i.e., 'blood cell', 'thyroid 1', 'vehicle', and 'letter'). On the other hand, the performance of BKDA is lower than that of BLDA for the 'segmentation' dataset; however, their difference is not very large. Thus, these results for the multi-class problems demonstrate that the combination of KDA and boosting in the proposed BKDA works effectively to attain good classification performance.

On the other hand, for the 2-class problems, the advantage of the proposed BKDA is a little obscure against both BLDA and RBFN (see Table 4 (b)). BKDA has sig-

[†] Due to such a long computation time for the CV-based selection method, the classification accuracy was not evaluated here.

Table 4 The results of the performance comparison for (a) multi-class problems and (b) two-class problems. The two values in each column mean the average classification accuracy [%] for the test dataset and the standard deviation, respectively. The asterisk after the values means that there is 1% level of significance for the average accuracy between BKDA and the two competitive methods. The best and the second best results are written in bold and italic fonts, respectively.

(a) Multi-class problems.

	BKDA	BLDA	RBFN
blood cell	**93.6 ± 0.34**	_93.36 ± 0.35_ *	92.62 ± 0.4*
thyroid 1	**97.34 ± 0.31**	_97.12 ± 0.46_ *	94.59 ± 0.77 *
segmentation	_94.95 ± 0.78_	**95.66 ± 0.58** *	92.82 ± 0.75 *
vehicle	**80.5 ± 1.56**	_79.73 ± 1.68_ *	78.39 ± 1.64 *
letter	**87.23 ± 0.47**	_85.83 ± 0.51_ *	85.47 ± 0.39 *

(b) Two-class problems.

	BKDA	BLDA	RBFN
banana	87.44 ± 0.6	_88.03 ± 1.13_ *	**89.15 ± 0.53** *
breast cancer	71.17 ± 4.25	_71.48 ± 4.52_	**71.77 ± 4.76**
diabetis	72.48 ± 1.7	_75.09 ± 2.26_ *	**75.26 ± 1.79** *
german	_74.71 ± 2.45_	74.55 ± 2.68	**75.18 ± 2.36**
heart	_79.96 ± 3.17_	**80.7 ± 3.2**	79.01 ± 3.21 *
image	**97.42 ± 0.48**	_91.99 ± 3.88_ *	88.24 ± 0.97 *
ringnorm	**97.94 ± 0.22**	81.26 ± 4.46 *	_90.66 ± 3.73_ *
splice	_88.07 ± 0.71_	81.42 ± 1.14 *	**89.02 ± 1.08** *
thyroid 2	_95.83 ± 2.21_	93.69 ± 2.6 *	**96.07 ± 1.98**

nificant improvement against both BLDA and RBFN for the two datasets out of nine (i.e., 'image' and 'ringnorm'), while BKDA has significant degradation against both BLDA and RBFN for the two datasets out of nine (i.e., 'banana' and 'diabetis'). And the best performance is attained by RBFN in many cases. However, considering that BKDA greatly outperforms both BLDA and RBFN for the 'image' and 'ringnorm' datasets, and that there is no statistical significance for 'breast cancer', 'german', and 'thyroid 2' datasets between BKDA and RBFN, we can say that the effectiveness of the proposed BKDA is not lost even for the two-class problems.

From the above experimental results in Table 4, we conclude that although the advantage of the proposed BKDA against BLDA and/or RBFN can be dependent to the given datasets for two-class problems, BKDA is still a promising method for pattern classification problems especially for multi-class problems.

5. Conclusions and Further Works

This paper proposed a novel approach to constructing a multiple classifier system by combining AdaBoost.M2 and Kernel Discriminant Analysis (KDA). In this approach, the feature extraction is first carried out to construct a particular feature space for every classifier based on the KDA algorithm, then the feature vectors are used for training the classifier. Since enormous computation costs are generally needed if all training samples are applied to KDA to train every classifier, the training samples are selected based on the distribution probability $D_t(i, j)$ which is also used for selecting data to train a classifier in AdaBoost.M2. This data selection leads not only to the reduction of the computation costs but also to making every classifier uncorrelated, resulting in boosting the performance of AdaBoost.M2.

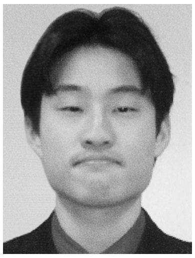
We also proposed a new kernel parameter selection for KDA and a new criterion of the training convergence for AdaBoost.M2. In the kernel selection, an optimal parameter is selected such that the class separability of feature vectors is maximized based on the same separability measure as used in KDA. Hence, the training of classifiers is unnecessary for testing the classification performance when an optimal kernel parameter is searched. It is expected that this contributes to alleviating immense computations that are often imposed to the kernel selection based on Cross-Validation (CV). As for the training convergence criterion, the minimum margin, which is defined as the minimum distance of samples to a separating hyper plane, was adopted as a measure of the generalization performance. In the first experiments, the effectiveness of the kernel parameter selection and the criterion of the training convergence was examined. The results showed that the proposed method could select an optimal parameter very quickly as compared with the CV-based kernel selection method, and that the training of boosting classifiers was terminated with fairly small rounds to attain good classification accuracy.

In the performance evaluation, the proposed Boosting KDA (BKDA) was compared with Boosting Linear Discriminant Analysis (BLDA) and Radial-Basis Function Network (RBFN) with regard to the classification accuracy. The experimental results for the multi-class classification problems demonstrated that BKDA outperformed both BLDA and RBFN in many cases. This result indicated that the combination of KDA as a feature extraction method and boosting as a classifier learning method works effectively to attain good classification performance. On the other hand, for the 2-class problems, the advantage of the proposed BKDA was a little obscure against both BLDA and RBFN. In many cases, the best performance is attained by RBFN; however, there are two datasets out of nine for which BKDA greatly outperforms both BLDA and RBFN. Therefore, we conclude that although the advantage of the proposed BKDA against BLDA and/or RBFN can be dependent to the given datasets, the proposed BKDA is still promising for many pattern classification problems.

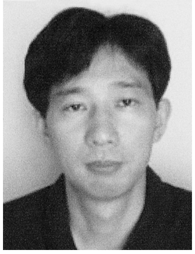
There still remain several problems that should be addressed as future works. First, the training time of BKDA is generally a little longer than that of both BLDA and RBFN to achieve good classification accuracy. However, the training time of BKDA is strongly related to the number of boosting rounds. As seen from Table 3, there was a case where BKDA needed over 200 boosting rounds although the classification accuracy was not largely improved from the 200th round. This fact means that the training speed can be enhanced by improving the criterion of training convergence. Finally, the data selection method for KDA should be improved. Our preliminary experiment showed that the proposed selection method based on the distribution probability was better than the random selection. However, this does not mean that the proposed method is optimal. Recently, Dai and Yeung [6] have proposed a new method to calculate optimal discriminant vectors for weak classifiers by weighting training samples of hard-to-separate classes, and this scheme could make classifier outputs uncorrelated. Since the data selection in KDA strongly affects the dependence of weak classifiers, further improvement in the proposed BKDA might be done with Dai's KDA.

References

[1] S. Abe, Support Vector Machines for Pattern Classification, Springer, 2005.
[2] G. Baudat and F. Anouar, "Generalized discriminant analysis using a kernel approach," Neural Comput., vol.12, no.10, pp.2385–2404, 2000.
[3] C.M. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, 1995.
[4] L. Breiman, "Arcing classifiers," The Annals of Statistics, vol.26, no.3, pp.801–849, 1998.
[5] N. Cristianini, C. Campbell, and J. Shawe-Taylor, "Dynamically adapting kernels in support vector machines," in Advances in Neural Information Processing Systems 11, ed. M.S. Kearns, S.A. Solla, and D.A. Cohn, pp.204–210, MIT Press, 1999.
[6] G. Dai and D.-Y. Yeung, "Boosting kernel discriminant analysis and its application to tissue classification of gene expression data," Proc. 20th Int. Joint Conf. on Artificial Intelligence, pp.744–749, 2007.
[7] Y. Freund and R.E. Schapire, "Experiments with a new boosting algorithm," Proc. Thirteenth Int. Conf. on Machine Learning, pp.148–156, 1996.
[8] Y. Freund and R.E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," J. Comput. Syst. Sci., vol.55, no.1, pp.119–139, 1997.
[9] F. Friedrichs and C. Igel, "Evolutionary tuning of multiple SVM parameters," Neurocomputing, vol.64, pp.107–117, 2005.
[10] K. Fukunaga, Introduction to Statistical Pattern Recognition (2nd ed.), Academic Press, 1990.
[11] S. Haykin, Neural Networks — A Comprehensive Foundation (2nd ed.), Prentice Hall, 1999.
[12] J. Huang, P.C. Yuen, W.-S. Chen, and J.-H. Lai, "Kernel subspace LDA with optimized kernel parameters on face recognition," Proc. Sixth IEEE Int. Conf. on Automatic Face and Gesture Recognition, pp.327–332, 2004.
[13] http://ida.first.fraunhofer.de/projects/bench/benchmarks.htm
[14] S. Kita, S. Maekawa, S. Ozawa, and S. Abe, "Boosting kernel discriminant analysis with adaptive kernel selection," Proc. 7th Int. Conf. on Adaptive and Natural Computing Algorithm, pp.429–432, 2005.
[15] G. Lebrun, C. Charrier, and H. Cardot, "SVM training time reduction using vector quantization," Proc. Seventeenth Int. Conf. on Pattern Recognition, vol.1, pp.160–163, 2004.
[16] W. Li, X. Gao, Y. Zhu, V. Ramesh, and T.E. Boult, "On the small sample performance of boosted classifiers," IEEE Computer Society Conf. on Computer Vision and Pattern Recognition, vol.2, pp.574–581, 2005.
[17] J. Lu, K.N. Plataniotis, and A.N. Venetsanopoulos, "Boosting linear discriminant analysis for face recognition," IEEE Int. Conf. on Image Processing, pp.657–660, 2003.
[18] J. Lu, K.N. Plataniotis, A.N. Venetsanopoulos, and S.Z. Li, "Ensemble-based discriminant learning with boosting for face recognition," IEEE Trans. Neural Netw., vol.17, no.1, pp.166–178, 2006.
[19] M.J.L. Orr, "Introduction to radial basis function networks," Tech. Rep., Centre for Cognitive Science, University of Edinburgh, 1996.
[20] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K. Muller, "Fisher discriminant analysis with kernels," Proc. IEEE Neural Networks for Signal Processing Workshop, pp.41–48, 1999.
[21] J. Moody and C. Darken, "Fast learning in networks of locally-tuned processing units," Neural Comput., vol.1, no.2, pp.281–294, 1989.
[22] K.-R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf, "An introduction to kernel-based learning algorithms," IEEE Trans. Neural Netw., vol.12, no.2, pp.181–201, 2001.
[23] A. Murua, "Upper bounds for error rates of linear combinations of classifiers," IEEE Trans. Pattern Anal. Mach. Intell., vol.24, no.5, pp.591–602, 2002.
[24] T. Poggio and F. Girosi, "Networks for approximation and learning," IEEE Trans. Neural Netw., vol.78, no.9, pp.1481–1497, 1990.
[25] G. Ratsch and M.K. Warmuth, "Efficient margin maximizing with boosting," Journal of Machine Learning Research, vol.6, pp.2131–2152, 2005.
[26] R.E. Schapire, Y. Freund, and P. Bartlett, "Boosting the margin: A new explanation for the effectiveness of voting methods," Annals of Statistics, vol.26, no.5, pp.1651–1686, 1998.
[27] B. Scholkopf, A. Smola, and K.-R. Muller, "Nonlinear component analysis as a kernel eigenvalue problem," Neural Comput., vol.10, no.5, pp.1299–1319, 1998.
[28] A.J. Smola, "On a kernel-based method for pattern recognition, regression, approximation, and operator inversion," Algorithmica, vol.22, no.1–2, pp.211–231, 1998.
[29] ftp://ftp.ics.uci.edu/pub/machine-learning-databases/



Shinji Kita received the B.S. and the M.S. degree in Electrical and Electronic Engineering from Kobe University, Kobe, Japan in 2003 and 2005, respectively. His research interests include pattern classification using neural networks, boosting, wavelet analysis, image processing, and kernel methods. He also has interests in the application to face recognition systems.



Seiichi Ozawa received the B.E. and M.E. degrees in instrumentation engineering from Kobe University, Kobe, Japan, in 1987 and 1989, respectively. In 1998, he received his Ph.D. degree in computer science from Kobe University, Kobe, Japan. He was a visiting researcher of Arizona State University, Arizona during 2005-2006. He is currently an associate professor with Graduate School of Engineering, Kobe University. His primary research interests include neural networks, machine learning, intelligent data processing, and pattern recognition. Dr. Ozawa is a member of IEEE, INNS, IEEJ, and SICE.

Satoshi Maekawa received a B.S. degree in Physics and M.E. degree in Electrical Engineering from Kyoto University in 1987 and 1990, respectively. In 1995, he received Ph.D. degree in Electrical Engineering from Kyoto University. He had been with NEC Corporation. He is now a senior researcher in National Institute of Information and Communications Technology.

Shigeo Abe received the B.S. degree in Electronics Engineering, the M.S. degree in Electrical Engineering, and the Dr. Eng. degree, all from Kyoto University, Kyoto, Japan in 1970, 1972, and 1984, respectively. After 25 years in the industry, he was appointed as full professor of Electrical Engineering, Kobe University in April 1997. He is now a professor of Graduate School of Engineering, Kobe University. His research interests include pattern classification and function approximation using neural networks, fuzzy systems, and support vector machines. He is the author of Neural Networks and Fuzzy Systems (Kluwer, 1996), Pattern Classification (Springer, 2001), and Support Vector Machines for Pattern Classification (Springer, 2005). Dr. Abe was awarded an outstanding paper prize from the Institute of Electrical Engineers of Japan in 1984 and 1995. He is a member of IEEE, INNS, and several Japanese Societies.