KPCA-based training of a kernel fuzzy classifier with ellipsoidal regions

Kaieda, Kenichi

Abe, Shigeo

(Citation)
International Journal of Approximate Reasoning,37(3):189-217

(Issue Date)
2004-09

(Resource Type)
journal article

(Version)
Accepted Manuscript

(URL)
https://hdl.handle.net/20.500.14094/90000151

KPCA-Based Training of a Kernel Fuzzy Classifier with Ellipsoidal Regions

Kenichi Kaieda and Shigeo Abe
Graduate School of Science and Technology
Kobe University

March 29, 2004

Abstract

In a fuzzy classifier with ellipsoidal regions, a fuzzy rule, which is based on the Mahalanobis distance, is defined for each class. Then the fuzzy rules are tuned so that the recognition rate of the training data is maximized. In most cases, one fuzzy rule per one class is enough to obtain high generalization ability. But in some cases, we need to partition the class data to define more than one rule per class.

In this paper, instead of partitioning the class data, we map the input space into the high dimensional feature space and then generate a fuzzy classifier with ellipsoidal regions in the feature space. We call this classifier kernel fuzzy classifier with ellipsoidal regions.

To speed up training, first we select independent training data that span the subspace in the feature space and calculate the kernel principal components. By this method, we can avoid using singular value decomposition, which leads to training speedup.

In the feature space, training data are usually degenerate. Namely, the space spanned by the mapped training data is a proper subspace. Thus, if the mapped test data are in the complementary subspace, the Mahalanobis distance may become erroneous and thus the probability of misclassification becomes high. To overcome this problem, we propose transductive training: in training, we add basis vectors of the input space as unlabelled data; and in classification, if mapped unknown data are not in the subspace we expand the subspace so that they are included.

We demonstrate the effectiveness of our method by computer simulations.

1 Introduction

There are many fuzzy classifiers developed so far [1, 2]. Among them a fuzzy classifier with ellipsoidal regions [2, 3, 4] shows relatively good generalization performance for a wide range of applications. In training the classifier, first we define a fuzzy rule for each class, calculating the center and the covariance

matrix using the training data for each class. Then the membership functions, which are based on the Mahalanobis distance, are tuned so that the recognition rate of the training data is maximized. To improve generalization ability, we define more than one rule for each class, either by partitioning the training data before training [3] or after tuning the membership functions [4].

Support vector machines (SVMs) developed by Vapnik are known to be a powerful tool for generating pattern recognition systems with high generalization ability [5]. In an SVM, first the input space is mapped into a high dimensional feature space and the optimal hyperplane that maximizes a margin between two classes is determined. Inspired by the success of SVMs, many conventional techniques, mostly linear techniques, are reformulated in the feature space to enhance linear separability. These techniques are called kernel-based methods. Kernel least squares [6], kernel principal component analysis (KPCA) [7], and the kernel Mahalanobis distance [6] are examples of kernel-based methods. But in these methods, since we need to calculate the pseudo-inverse of a matrix, with the size equal to the number of training data, by singular value decomposition, the calculation time becomes longer as the number of training data increases. To avoid this, in [8] independent vectors are selected by sequential forward selection.

In this paper, to improve the generalization ability of the fuzzy classifier with ellipsoidal regions without explicitly partitioning the training data, we propose a kernel version [9], which is based on the kernel Mahalanobis distance. In the feature space we define a fuzzy rule for each class using the training data. We then tune fuzzy rules so that the recognition rate of the training data is improved in the similar way as discussed in [2, 3]. The hyper ellipsoid in the feature space corresponds to several clusters in the input space. Thus, clustering is automatically done by mapping.

To overcome the problem of long calculation time, we propose a new method based on KPCA to calculate a kernel Mahalanobis distance. According to the formulation of [7], all the training data have to be retained to calculate the kernel principal components. But in our proposed method, we first select the linearly independent vectors that form a basis of the subspace in the feature space and discard the remaining redundant vectors. The difference of our method from [8] is that we select independent vectors by the Cholesky factorization, not by forward selection. By our method, the covariance matrix in the feature space is not singular and we do not need to use singular value decomposition. Thus we achieve training speedup compared with the conventional kernel method.

Usually the space spanned by the mapped training data in the feature space forms a proper subspace of the feature space. In such a case, if the mapped test data exist in the complementary subspace, the kernel Mahalanobis distance may becomes erroneous and this results in misclassification. This may happen when we use the input space as feature space and the training data are degenerate [9]. Namely, the training data do not span the input space. To avoid this, we introduce transductive training: in training, we add the basis vectors of the input space as the unlabelled training data, and if the mappings of these data are in the complementary subspace, we add them as the basis vectors in the

feature space. Namely, first, we map the basis vectors into the feature space, and then we perform the Cholesky factorization to judge whether each mapped basis vector is included in the space spanned by the mapped training data. If it is not included in the space, we calculate the vector that is orthogonal to the space spanned by the mapped training data and modify the covariance matrix.

By this method, however, the complementary subspace may remain. Thus, in classification, whenever an unknown datum is given, we judge whether the mapped datum is included in the space spanned by the mapped training data and basis vectors by the Cholesky factorization. We can speed up factorization, storing the previous factorized matrices and factorizing the row and column associated with only the unknown datum. If it is not included, we calculate the vector orthogonal to the previously selected vectors by the Gram-Schmidt orthogonalization and modify the covariance matrix.

In Section 2 we describe a conventional fuzzy classifier with ellipsoidal regions and its kernel version, and in Section 3 we discuss the conventional kernel method to calculate Mahalanobis distance which is based on singular value decomposition. Then in Section 4, we describe a singular value decomposition technique and its variant for improving the generalization ability. In Section 5, we propose a KPCA-based Mahalanobis distance, and in Section 6, we propose transductive training. Then in Section 7 we discuss a fuzzy rule tuning. In Section 8, we evaluate our methods using many benchmark data sets.

2 Classifier Architecture

2.1 Conventional Fuzzy Classifier with Ellipsoidal Regions

In this section, we discuss the conventional fuzzy classifier with ellipsoidal regions, which is generated in the input space [2, 3, 10]. Here we consider classification of the m-dimensional input vector $\boldsymbol{x}$ into one of n classes. For each class, we define the following fuzzy rule:

$$R_i\text{: if } \boldsymbol{x} \text{ is } \boldsymbol{c}_i\text{, then } \boldsymbol{x} \text{ belongs to class } i, \tag{1}$$

where $\boldsymbol{c}_i$ is the center vector of class i:

$$\boldsymbol{c}_i = \frac{1}{M_i} \sum_{j=1}^{M_i} \boldsymbol{x}_{ij}. \tag{2}$$

Here, M_i is the number of training data for class i and $\boldsymbol{x}_{ij}$ is the jth training datum for class i. In the following equation, we omit the subscript i in $\boldsymbol{x}_{ij}$ to simplify notations.

We define a membership function $m_i(\boldsymbol{x})$ $(i = 1, \cdots, n)$ for input datum $\boldsymbol{x}$ by

$$\begin{aligned} m_i(\boldsymbol{x}) &= \exp(-h_i^2(\boldsymbol{x})) \\ h_i^2(\boldsymbol{x}) &= \frac{\delta_i^2(\boldsymbol{x})}{\alpha_i} \end{aligned} \tag{3}$$

$$= \frac{1}{\alpha_i}[\boldsymbol{x} - \boldsymbol{c}_i]^T Q_i^{-1}[\boldsymbol{x} - \boldsymbol{c}_i], \tag{4}$$

where $\delta_i(\boldsymbol{x})$: the Mahalanobis distance between $\boldsymbol{x}$ and $\boldsymbol{c}_i$; $h_i(\boldsymbol{x})$: the tuned distance; α_i: the tuning parameter for class i; Q_i: the covariance matrix for class i in the input space, and Q_i^{-1}: the inverse of Q_i.

The covariance matrix for class i in the input space is given by

$$Q_i = \frac{1}{M_i}\sum_{j=1}^{M_i}[\boldsymbol{x}_j - \boldsymbol{c}_i][\boldsymbol{x}_j - \boldsymbol{c}_i]^T. \tag{5}$$

We calculate the membership function of input datum $\boldsymbol{x}$ for each class. If the degree of membership for class j is maximum, the input datum is classified into class j. This is equivalent to finding the minimum Mahalanobis distance when α_i in (4) is equal to 1. Function (3) makes the output range of (3) lie in [0,1], and if $m_j(\boldsymbol{x})$ is equal to 1, the input vector $\boldsymbol{x}$ corresponds to the center of class j, $\boldsymbol{c}_j$. We tune the membership function using α_i in (4).

When Q_i is positive definite, the Mahalanobis distance (4) can be calculated using the symmetric Cholesky factorization [11]. Thus Q_i can be decomposed into

$$Q_i = L_i L_i^T, \tag{6}$$

where L_i is the real valued regular lower triangular matrix.

But when Q is positive semi-definite, the value in the square root of diagonal element of L_i is non-positive. To avoid this, during the Cholesky factorization, if the value in the square root is smaller than ζ, where ζ is a predefined threshold, the element is replaced by $\sqrt{\zeta}$ [2, 10]. This means that when the covariance matrix is positive semi-definite, the principal components in the subspace that the training data do not span, are taken into consideration to calculate the Mahalanobis distance. Thus by this method, the generalization ability does not decrease even when the training data are degenerate, namely, the training data do not span the input space.

2.2 Kernel Fuzzy Classifier with Ellipsoidal Regions

In a kernel fuzzy classifier with ellipsoidal regions [9], the input space is mapped into the feature space by a non-linear mapping function. Similar to a conventional fuzzy classifier, in the feature space the following fuzzy rule is defined for each class:

$$R_i\text{: if } \phi(\boldsymbol{x}) \text{ is } \boldsymbol{\mu}_i, \text{ then } \boldsymbol{x} \text{ belongs to class } i, \tag{7}$$

where ϕ is the nonlinear mapping function that maps $\boldsymbol{x}$ into the l-dimensional feature space and $\boldsymbol{\mu}_i$ is the center of class i in the feature space and is given by

$$\boldsymbol{\mu}_i = \frac{1}{M_i}\sum_{j=1}^{M_i}\phi(\boldsymbol{x}_j). \tag{8}$$

We define a membership function $m_i(\phi(\boldsymbol{x}))$ $(i = 1, \cdots, n)$ for input datum $\boldsymbol{x}$ by

$$
\begin{aligned}
m_i(\phi(\boldsymbol{x})) &= \exp(-h_i^2(\phi(\boldsymbol{x}))), && (9)\\
h_i^2(\phi(\boldsymbol{x})) &= \frac{\delta_i^2(\phi(\boldsymbol{x}))}{\alpha_i} \\
&= \frac{1}{\alpha_i}[\phi(\boldsymbol{x}) - \boldsymbol{\mu}_i]^T \phi_{C_i}^+ [\phi(\boldsymbol{x}) - \boldsymbol{\mu}_i], && (10)
\end{aligned}
$$

where $\delta_i(\phi(\boldsymbol{x}))$: the kernel Mahalanobis distance between $\boldsymbol{x}$ and $\boldsymbol{\mu}_i$; $h_i(\phi(\boldsymbol{x}))$: tuned distance; α_i: tuning parameter for class i; ϕ_{C_i}: the covariance matrix for class i in the feature space, $\phi_{C_i}^+$: the pseudo-inverse of ϕ_{C_i}.

In the similar way as the conventional fuzzy classifier, we calculate the membership function of input datum $\boldsymbol{x}$ for each class. If the degree of membership for class j is maximum, the input datum is classified into class j. Membership functions given by (9) are tuned in the feature space as well as in the input space by changing α_i in (10). This tuning method is described in Section 7.

3 SVD-based Kernel Mahalanobis Distance

To calculate the kernel Mahalanobis distance given by (10) without explicitly treating the variables in the feature space, we use the kernel method. Namely, we transform (10) so that only the dot products $\phi^T(\boldsymbol{x})\phi(\boldsymbol{x})$ appear in (10) [6].

The covariance matrix in the feature space is expressed in a matrix form as follows:

$$
\begin{aligned}
\phi_{C_i} &= \frac{1}{M_i}\sum_{j=1}^{M_i}(\phi(\boldsymbol{x}_j) - \boldsymbol{\mu}_i)(\phi(\boldsymbol{x}_j) - \boldsymbol{\mu}_i)^T \\
&= \phi^T(X_i)\left(\frac{1}{M_i}(I_{M_i} - \mathbf{1}_{M_i})\right)\phi(X_i), && (11)
\end{aligned}
$$

where $\boldsymbol{x}_j = [x_{j1} \cdots x_{jm}]^T$ is the jth training data for class i, I_{M_i} is the $M_i \times M_i$ dimensional unit matrix, $\mathbf{1}_{M_i}$ is the $M_i \times M_i$ dimensional matrix with all components equal to $1/M_i$, and $\phi(X_i)$ is an $M_i \times l$ matrix:

$$
\phi(X_i) = \begin{bmatrix} \phi^T(\boldsymbol{x}_1) \\ \vdots \\ \phi^T(\boldsymbol{x}_{M_i}) \end{bmatrix}. \qquad (12)
$$

Since $I_{M_i} - \mathbf{1}_{M_i}$ is a symmetric positive semi-definite matrix, we can define the square root of the matrix, Z_i:

$$
Z_i = \left(\frac{1}{M_i}(I_{M_i} - \mathbf{1}_{M_i})\right)^{\frac{1}{2}}. \qquad (13)
$$

Substituting (13) into (11) we obtain

$$\phi_{C_i} = \phi^T(X_i) Z_i^2 \phi(X_i). \tag{14}$$

Substituting the above equation into (10) gives

$$\delta_i^2(\phi(\boldsymbol{x})) = [\phi(\boldsymbol{x}) - \boldsymbol{\mu}_i]^T [\phi^T(X_i) Z_i^2 \phi(X_i)]^+ [\phi(\boldsymbol{x}) - \boldsymbol{\mu}_i]. \tag{15}$$

Now the following equation is valid for any integral number n and symmetric positive semi-definite matrix A and any vectors $\boldsymbol{t}$ and $\boldsymbol{u}$ [6]:

$$\boldsymbol{t}^T (X^T A X)^n \boldsymbol{u} = \boldsymbol{t}^T X^T (A^{\frac{1}{2}} (A^{\frac{1}{2}} H A^{\frac{1}{2}})^{n-1} A^{\frac{1}{2}}) X \boldsymbol{u}, \tag{16}$$

where $H = XX^T$. If n is negative, it means pseudo-inverse. We calculate the pseudo-inverse using singular value decomposition, which will be discussed in the next section.

Here since Z_i^2 in (15) is a symmetric positive semi-definite matrix, we can apply (16) to (15):

$$\begin{aligned} \delta_i^2(\phi(\boldsymbol{x})) &= [\phi(X_i)\phi(\boldsymbol{x}) - \phi(X_i)\boldsymbol{\mu}_i]^T \left[Z_i (Z_i \phi(X_i) \phi^T(X_i) Z_i)^{-2} Z_i \right] \\ &\quad \times [\phi(X_i)\phi(\boldsymbol{x}) - \phi(X_i)\boldsymbol{\mu}_i]. \end{aligned} \tag{17}$$

Since the above equation consists of only dot products in the feature space, we can replace them with kernel functions:

$$K(\boldsymbol{x}, \boldsymbol{y}) = \phi(\boldsymbol{x})^T \phi(\boldsymbol{y}). \tag{18}$$

The kernel functions that we use in the following study are:

1. linear kernels
$$K(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}^T \boldsymbol{y}, \tag{19}$$
2. polynomial kernels
$$K(\boldsymbol{x}, \boldsymbol{y}) = (1 + \boldsymbol{x}^T \boldsymbol{y})^d, \tag{20}$$
where d is a positive integer,
3. radial basis function (RBF) kernels
$$K(\boldsymbol{x}, \boldsymbol{y}) = \exp(-\gamma ||\boldsymbol{x} - \boldsymbol{y}||^2), \tag{21}$$
where γ is a positive parameter.

Using (18), we can rewrite the dot products in (17) as follows:

$$\begin{aligned} \phi(X_i)\phi(\boldsymbol{x}) &= \begin{bmatrix} K(\boldsymbol{x}_1, \boldsymbol{x}) \\ \vdots \\ K(\boldsymbol{x}_{M_i}, \boldsymbol{x}) \end{bmatrix} = K(X_i, \boldsymbol{x}), \\ \phi(X_i)\boldsymbol{\mu}_i &= \frac{1}{M_i} \phi(X_i) \sum_{j=1}^{M_i} \phi(\boldsymbol{x}_j) \end{aligned} \tag{22}$$

$$= \frac{1}{M_i}\sum_{j=1}^{M_i} K(X_i, \boldsymbol{x}_j), \tag{23}$$

$$\phi(X_i)\phi^T(X_i) = \begin{bmatrix} K(\boldsymbol{x}_1, \boldsymbol{x}_1) & \cdots & K(\boldsymbol{x}_1, \boldsymbol{x}_{M_i}) \\ \vdots & \ddots & \vdots \\ K(\boldsymbol{x}_{M_i}, \boldsymbol{x}_1) & \cdots & K(\boldsymbol{x}_{M_i}, \boldsymbol{x}_{M_i}) \end{bmatrix}$$

$$= K(X_i, X_i^T), \tag{24}$$

where, $K(X_i, X_i^T)$ is called kernel matrix.

Substituting (22), (23), and (24) into (17), we obtain

$$\delta_i^2(\phi(\boldsymbol{x})) = [K(X_i, \boldsymbol{x}) - \frac{1}{M_i}\sum_{j=1}^{M_i} K(X_i, \boldsymbol{x}_j)]^T [Z_i (Z_i K(X_i, X_i^T) Z_i)^{-2} Z_i]$$
$$\times [K(X_i, \boldsymbol{x}) - \frac{1}{M_i}\sum_{j=1}^{M_i} K(X_i, \boldsymbol{x}_j)]. \tag{25}$$

Using (25) we can calculate the kernel Mahalanobis distance without treating variables in the feature space. We call this equation SVD-based kernel Mahalanobis distance.

4 Singular Value Decomposition

In (25), $(Z_i K(X_i, X_i^T) Z_i)^{-2}$ needs to be calculated by singular value decomposition. In this section, we discuss singular value decomposition and its variant to improve generalization ability when the number of training data is small.

4.1 Conventional Method

Any matrix A is decomposed into $A = S\Lambda U^T$ by singular value decomposition, where S and U are orthogonal matrices ($SS^T = I, UU^T = I$, where I is a unit matrix) and Λ is a diagonal matrix. If A is an $m \times m$ positive semi-definite matrix, singular value decomposition is equivalent to the diagonalization of the matrix. Namely, S, Λ, and U are $m \times m$ square matrices and $S = U$. Since $Z_i K(X_i, X_i^T) Z_i$ is a symmetric positive semi-definite matrix, in the following discussion we assume that A is symmetric positive semi-definite.

If A is positive definite, the inverse of A is expressed as follows:

$$A^{-1} = (U\Lambda U^T)^{-1} = (U^T)^{-1}\Lambda^{-1}U^{-1} = U\Lambda^{-1}U^T. \tag{26}$$

Assume that A is positive semi-definite with rank r ($m > r$). In this case the pseudo-inverse of A, A^+, is used. Since $m > r$ holds, the $(r+1)$th to mth diagonal elements of Λ are zero. In the conventional pseudo-inverse, if a diagonal element λ_i is larger than or equal to σ, where σ is a predefined threshold, we set $1/\lambda_i$ to the ith element of Λ^+. But if it is smaller, we set 0.

4.2 Proposed Method

According to our computer experiments, if the number of training data was small, the generalization ability of the kernel fuzzy classifier was inferior to that of the conventional fuzzy classifier. The reason for this will be discussed in Section 6. In the conventional method, if a diagonal element is smaller than σ, the associated diagonal element of the pseudo-inverse is set to 0. It means that all components with small singular values are neglected. Namely, the subspace corresponding to the zero diagonal elements is neglected. This leads to decreasing the generalization ability. To avoid this we set $1/\sigma$ instead of 0. Namely we calculate the pseudo-inverse as follows:

$$\begin{aligned} A^+ &= U\Lambda^+ U^T \\ &= U \begin{bmatrix} \lambda_1^{-1} & & & & & & \\ & \ddots & & & & & \\ & & \lambda_r^{-1} & & & & \\ & & & \frac{1}{\sigma} & & & \\ & & & & \ddots & & \\ & & & & & \frac{1}{\sigma} \end{bmatrix} U^T. \end{aligned} \tag{27}$$

5 KPCA-based Mahalanobis Distance

In this section we propose the KPCA-based method to calculate a Mahalanobis distance in the feature space. This method is equivalent to the kernel Mahalanobis distance discussed in Section 3. But in this method since we can discard the redundant input vectors, calculation time can be shortened compared with that of the conventional kernel Mahalanobis distance given by (25).

5.1 Kernel Mahalanobis Distance Calculated by Orthogonal Transformation

Let λ_j and $\boldsymbol{z}_j$ be the jth eigenvalue and the associated eigenvector of the covariance matrix ϕ_{C_i} given by (11). (Here, we do not affix the subscript i to λ_j and $\boldsymbol{z}_j$ to simplify notations. It will be added if a confusion may arise.) Then

$$\phi_{C_i} \boldsymbol{z}_j = \lambda_j \boldsymbol{z}_j \qquad \text{for} \quad j = 1, \ldots, l, \tag{28}$$

where l is the dimension of the feature space.

In the pseudo-inverse, we set the inverse of the zero eigenvalue to be zero [11, pp. 257–258]. Then for the pseudo-inverse of ϕ_{C_i}, $\phi_{C_i}^+$, the following equation holds:

$$\phi_{C_i}^+ \boldsymbol{z}_j = \begin{cases} \lambda_j^{-1} \boldsymbol{z}_j & \lambda_j > 0, \\ 0 & \lambda_j = 0. \end{cases} \tag{29}$$

Then the jth kernel principal component of input $\boldsymbol{x}$ is defined by

$$y_j = \begin{cases} \boldsymbol{z}_j^t(\phi(\boldsymbol{x}) - \boldsymbol{\mu}_i) & \lambda_j > 0, \\ 0 & \lambda_j = 0. \end{cases} \tag{30}$$

From (29) and (30), the Mahalanobis distance in that space can be calculated as follows:

$$\delta_i^2(\phi(\boldsymbol{x})) = \frac{y_1^2}{\lambda_1} + \cdots + \frac{y_{M_i'}^2}{\lambda_{M_i'}}, \tag{31}$$

where M_i' is the number of non-zero eigenvalues. We call this equation KPCA-based Mahalanobis distance.

To calculate the component given by (30) efficiently, we use the KPCA method.

5.2 Kernel Principal Component Analysis

Here we explain the KPCA method to calculate (30). This is a sophisticated version of [7].

From (11), the eigenvalue equation is given by

$$\phi^T(X_i)\left(\frac{1}{M_i}(I_{M_i} - \mathbf{1}_{M_i})\right)\phi(X_i)\boldsymbol{z} = \lambda \boldsymbol{z}. \tag{32}$$

According to the formulation of [7], $\boldsymbol{z}$ is expressed by the linear sum of $\{\phi(\boldsymbol{x}_1), \cdots, \phi(\boldsymbol{x}_{M_i})\}$. But by this method, we need to retain all the training data. To overcome this problem, we select only linearly independent vectors, which form a basis of the subspace in the feature space. It is because linearly independent vectors $\{\phi(\boldsymbol{x}_1), \cdots, \phi(\boldsymbol{x}_{M_i'})\}\{M_i' \leq M_i\}$ can span the space generated by $\{\phi(\boldsymbol{x}_1), \cdots, \phi(\boldsymbol{x}_{M_i})\}$[8].[1] Therefore, $\boldsymbol{z}$ is expressed by

$$\boldsymbol{z} = (\phi(\boldsymbol{x}_1), \cdots, \phi(\boldsymbol{x}_{M_i'}))\begin{bmatrix} \rho_1 \\ \vdots \\ \rho_{M_i'} \end{bmatrix}, \tag{33}$$

where $\{\rho_1, \cdots, \rho_{M_i'}\}$ are scalars.

In (32), we are only interested in the components for $\phi(\boldsymbol{x}_j)$ $(1 \leq j \leq M_i')$. Multiplying both term of (32) by $\phi^T(\boldsymbol{x}_i)$ from the left-hand side and substitute (33) into (32),

$$\frac{1}{M_i}(K_{i1}, \cdots, K_{iM_i})(I_{M_i} - \mathbf{1}_{M_i})K\rho' = \lambda(K_{i1}, \cdots, K_{iM_i'})\rho', \tag{34}$$

where $K_{ij} = \phi^T(\boldsymbol{x}_i)\phi(\boldsymbol{x}_j)$, $K = \{K_{ij}\}(i = 1, \cdots, M_i, j = 1, \cdots, M_i')$ $\rho' = [\rho_1, \cdots, \rho_{M_i'}]^T$. Thus combining (34) for $i = 1, \cdots, M_i'$,

$$\frac{1}{M_i}K^T(I_{M_i} - \mathbf{1}_{M_i})K\rho' = \lambda K^s\rho', \tag{35}$$

[1] In theory, using the RBF kernels, $M_i' = M_i$. But relaxing the decision of independence, we can select the smaller number of independent training data.

where $K^s = K_{ij}$ $(i = 1, \cdots, M_i',\ j = 1, \cdots, M_i')$.

Since $\{\phi(\boldsymbol{x}_1), \cdots, \phi(\boldsymbol{x}_{M_i'})\}$ are linearly independent, K^s is positive definite. So K^s can be decomposed into LL^T with the Cholesky factorization. Multiplying both term of (35) by L^{-1} from the left-hand side, we obtain

$$\frac{1}{M_i} L^{-1} K^T (I_{M_i} - \mathbf{1}_{M_i}) K (L^T)^{-1} L^T \rho' = \lambda L^T \rho'. \tag{36}$$

In (36), λ and $L^T\rho'$ mean an eigenvalue and an eigenvector, respectively. Thus solving the eigenvalue λ and eigenvector $L^T\rho'$ of (36), we obtain the generalized eigenvalue λ and eigenvector ρ' of (35).

When the jth eigenvector of (35), ρ_j', is defined as $\left[\rho_{j1}, \cdots, \rho_{jM_i'}\right]^T$, the jth eigenvector of (11) is given by

$$\boldsymbol{z}_j = (\phi(\boldsymbol{x}_1), \cdots, \phi(\boldsymbol{x}_{M_i'})) \begin{bmatrix} \rho_{j1} \\ \vdots \\ \rho_{jM_i'} \end{bmatrix}. \tag{37}$$

Using the jth eigenvector $\boldsymbol{z}_j$, we can calculate the jth kernel principal component (30) as follows:

$$\begin{aligned} y_j &= \boldsymbol{z}_j^T (\phi(\boldsymbol{x}) - \boldsymbol{\mu}_i) \\ &= \left[\rho_{j1}, \cdots, \rho_{jM_i'}\right] \begin{bmatrix} \phi(\boldsymbol{x}_1)^T \\ \vdots \\ \phi(\boldsymbol{x}_{M_i'})^T \end{bmatrix} \phi(\boldsymbol{x}) - \boldsymbol{z}_j^T \boldsymbol{\mu}_i \\ &= \sum_{l=1}^{M_i'} \rho_{jl} K(\boldsymbol{x}, \boldsymbol{x}_l) - \boldsymbol{z}_j^T \boldsymbol{\mu}_i, \end{aligned} \tag{38}$$

where

$$\boldsymbol{z}_j^T \boldsymbol{\mu}_i = \left[\rho_{j1}, \cdots, \rho_{jM_i'}\right] K^T \begin{bmatrix} \frac{1}{M_i} \\ \vdots \\ \frac{1}{M_i} \end{bmatrix}. \tag{39}$$

With the kernel principal components (38) and the eigenvalue, we can calculate the KPCA-based Mahalanobis distance given by (31). This is equivalent to the SVD-based kernel Mahalanobis distance, discussed in Section 3. But in this method, since we can discard the redundant training data, computational time can be shortened compared with that of the SVD-based kernel Mahalanobis distance.

5.3 Selection of Linearly Independent Vectors

The dimensionality of the space spanned by the training data in the feature space is equivalent to the rank of the kernel matrix given by (24) [8]. In [8], a

set of linearly independent vectors is selected by sequential forward selection. In our method, since the kernel matrix (24) is guaranteed to be positive semi-definite, we adopt the symmetric Cholesky factorization.

The kernel matrix (24) can be decomposed into two triangular matrices by the symmetric Cholesky Factorization as follows:

$$K(X_i, X_i^T) = L_i L_i^T, \tag{40}$$

where L_i is the real-valued regular lower triangular matrix and each element of L_i is given by

$$\begin{aligned} l_{op} &= \frac{k_{op} - \sum_{n=1}^{p-1} l_{pn} l_{on}}{l_{pp}} \\ \text{for } o &= 1, \cdots, M_i, \quad p = 1, \cdots, o-1, \end{aligned} \tag{41}$$

$$l_{aa} = \sqrt{k_{aa} - \sum_{n=1}^{a-1} l_{an}^2} \quad \text{for } a = 1, 2, \cdots, M_i. \tag{42}$$

Here in (42), if

$$k_{jj} - \sum_{n=1}^{j-1} l_{jn}^2 < \tau, \tag{43}$$

where $\tau > 0$, we judge that the jth training datum $\boldsymbol{x}_j$ can be expressed by a linear sum of the previously processed data and discard it. We continue calculating l_{aa} until $a = M_i$. The training data that are not discarded are linearly independent vectors.

If we set a large value to τ, the approximation error of the subspace spanned by the training data increases. But since the number of independent variables decreases, the computation time becomes shorter. Therefore, there is a tradeoff between the approximation error and the computation time.

6 Transductive Training

6.1 Concept

In Sections 3 and 5, we discuss the SVD-based Mahalanobis distance (25) and the KPCA-based Mahalanobis distance (31), respectively. But with those methods, when the training data are degenerate, i.e., the space spanned by the mapped training data is a proper subspace in the feature space, the generalization ability of our classifier is decreased [9]. It is because the principal components are zero in the complementary subspace.

We now explain the reason using an example shown in Figure 1. In the figure, training data for class j are in the two-dimensional space but those of class i are in one dimension.

Assume that we have a datum $\boldsymbol{x}$ belonging to class j. This datum is in the subspace spanned by the mapped training data belonging to class j, but not in the subspace spanned by class i. Then the kernel Mahalanobis distance between $\boldsymbol{x}$ and $\boldsymbol{\mu}_j$ is correctly calculated by

$$\delta_j^2(\phi(\boldsymbol{x})) = \frac{y_{j1}^2}{\lambda_{j1}} + \frac{y_{j2}^2}{\lambda_{j2}}. \tag{44}$$

But for class i, since the second eigenvalue is zero due to the degeneracy of the training data, $\delta_i^2(\phi(\boldsymbol{x}))$ is erroneously given by

$$\delta_i^2(\phi(\boldsymbol{x})) = \frac{y_{i1}^2}{\lambda_{i1}}. \tag{45}$$

Since the kernel Mahalanobis distance for class i does not change even if $\boldsymbol{x}$ moves in the direction orthogonal to the eigenvector $\boldsymbol{z}_{i1}$, classification by the Mahalanobis distance becomes erroneous.

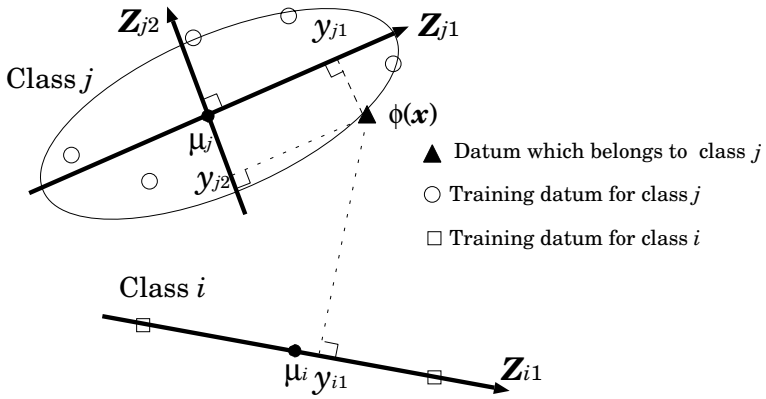

Figure 1: The training data for class i are degenerate. Since the second eigenvalue of class i is zero, the kernel Mahalanobis distance for class i, $\delta_i(\phi(\boldsymbol{x}))$, cannot be calculated correctly.

Similar to (27), we can overcome this problem, adding the vector that is orthogonal to the existing eigenvectors for the covariance matrix with a small equivalent eigenvalue. The next problem is how to select the appropriate vectors for addition. If the linear kernel is used and the training data are degenerate, i.e., the training data do not span the input space, we can add the basis vectors in the input space that are in the complementary subspace of the training data. If the nonlinear kernel is used, the mapped training data may not span the feature space even though the training data are not degenerate.

To solve this problem, we propose transductive training of the classifier. Namely, in training, we add the basis vectors of the input space as the unlabelled training data, and if the mappings of these data are in the complementary

subspace associated with the covariance matrix, we generate associated eigenvectors with small eigenvalues. Namely, first we map the basis vectors into the feature space, and then we perform the Cholesky factorization to judge whether each mapped basis vector is included in the subspace associated with the covariance matrix. If it is not included in the subspace, we calculate the vector that is orthogonal to the subspace and modify the covariance matrix.

By this method, however, the complementary subspace may remain. Thus, in classification, whenever an unknown datum is given, we judge whether the mapped datum is included in the subspace associated with the covariance matrix by the Cholesky factorization. If it is not included, we calculate the vector orthogonal to the previously selected vectors by the Gram-Schmidt orthogonalization and modify the covariance matrix. We can speed up factorization, storing the previously factorized matrices and factorizing the row and column associated with only the unknown datum.

6.2 Transductive Training Using Basis Vectors

In this section, we discuss how to improve approximation of the feature space using the basis vectors $\{\boldsymbol{e}_1, \cdots, \boldsymbol{e}_m\}$ in the input space in addition to the vector of the mapped training data $\phi(X_i)$. Since the labels of $\{\boldsymbol{e}_1, \cdots, \boldsymbol{e}_m\}$ are not known, these data should not affect the principal components associated with $\phi(X_i)$. Therefore, first we calculate the eigenvalues $\{\lambda_1, \cdots, \lambda_{M_i'}\}$ and eigenvectors $\{\boldsymbol{z}_1, \cdots, \boldsymbol{z}_{M_i'}\}$ of ϕ_{C_i}, and add the eigenvalues with small eigenvectors using some of the $\{\phi(\boldsymbol{e}_1), \cdots, \phi(\boldsymbol{e}_m)\}$ that are orthogonal to $\{\boldsymbol{z}_1, \cdots, \boldsymbol{z}_{M_i'}\}$.

Now we will explain the procedure more in detail. In the similar way as discussed in Section 5.3, we can select independent vectors by the Cholesky factorization of the kernel matrix for $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{M_i}\}$ and $\{\boldsymbol{e}_1, \cdots, \boldsymbol{e}_m\}$:

$$K(X_i', X_i'^T) = \phi(X_i')\phi^T(X_i'), \tag{46}$$

where

$$\phi(X_i') = \left[\begin{array}{ccccccc} \phi(\boldsymbol{x}_1) & \cdots & \phi(\boldsymbol{x}_{M_i}) & \phi(\boldsymbol{e}_1) & \cdots & \phi(\boldsymbol{e}_m) \end{array} \right]^T. \tag{47}$$

Placing $\{\phi(\boldsymbol{e}_1), \cdots, \phi(\boldsymbol{e}_m)\}$ after $\phi(X_i)$, we can select independent vectors from $\{\boldsymbol{e}_1, \cdots, \boldsymbol{e}_m\}$ that are not included in the subspace spanned by $\phi(X_i)$. Assume that m' basis vectors, $\phi(\boldsymbol{e}_r)(r = 1, \cdots, m')$ are selected as the independent vectors in addition to M_i' independent training vectors. From the M_i' independent training data, the orthogonal system $\{\boldsymbol{z}_1, \cdots, \boldsymbol{z}_{M_i'}\}$ is calculated by the method discussed in Section 5.

Next using $\phi(\boldsymbol{e}_r)$ we generate $(M_i' + 1)$th extra eigenvector $\boldsymbol{z}_{M_i'+1}$ by the Gram-Schmidt orthogonalization:

$$\boldsymbol{z}_{M_i'+1} = \frac{\boldsymbol{g}_r}{\|\boldsymbol{g}_r\|} \tag{48}$$

where $||\boldsymbol{g}_r||$ is the norm of $\boldsymbol{g}_r$:

$$\boldsymbol{g}_r = \phi(\boldsymbol{e}_r) - \sum_{i=1}^{M_i'} (\phi(\boldsymbol{e}_r)^T \boldsymbol{z}_i)\boldsymbol{z}_i$$

$$= \quad \phi(\boldsymbol{e}_r) - \sum_{i=1}^{M_i'} \omega_i \boldsymbol{z}_i, \tag{49}$$

Here, $\omega_i = \phi(\boldsymbol{e}_r)^T \boldsymbol{z}_i$.

By substituting (37) into (49), $\boldsymbol{z}_{M_i'+1}$ is expressed by the linear sum of $\{\phi(\boldsymbol{x}_1), \cdots, \phi(\boldsymbol{x}_{M_i'}), \phi(\boldsymbol{e}_r)\}$:

$$\boldsymbol{z}_{M_i'+1} = \Big[\phi(\boldsymbol{x}_1), \cdots, \phi(\boldsymbol{x}_{M_i'}), \phi(\boldsymbol{e}_r)\Big] \begin{bmatrix} \rho_{M_i'+1,\ 1} \\ \vdots \\ \rho_{M_i'+1,\ M_i'} \\ \frac{1}{||\boldsymbol{g}_r||} \end{bmatrix}, \tag{50}$$

where

$$\rho_{M_i'+1,\ j} = -\frac{1}{||\boldsymbol{g}_r||} \sum_{i=1}^{M_i'} \omega_i \rho_{ij} \quad \text{for} \quad j = 1, \cdots, M_i'. \tag{51}$$

Using (50), we can calculate the $(M_i'+1)$th principal component as follows:

$$\begin{aligned} y_{M_i'+1} &= \boldsymbol{z}_{M_i'+1}^T (\phi(\boldsymbol{x}) - \boldsymbol{\mu}_i) \\ &= \sum_{i=1}^{M_i'} \rho_{M_i'+1,\ i} K(\boldsymbol{x}, \boldsymbol{x}_i) + \frac{1}{||\boldsymbol{g}_r||} K(\boldsymbol{e}_r, \boldsymbol{x}) - \boldsymbol{z}_{M_i'+1}^T \boldsymbol{\mu}_i. \end{aligned} \tag{52}$$

We iterate this procedure until the rest of the principal components y_i $(i = M_i'+1, \cdots, M_i'+m')$ are calculated.

We add these new principal components to the equation of (31), and finally the Mahalanobis distance becomes as follows:

$$\delta_i^2(\phi(\boldsymbol{x})) = \frac{y_1^2}{\lambda_1} + \cdots + \frac{y_{M_i'}^2}{\lambda_{M_i'}} + \frac{y_{M_i'+1}^2}{\varepsilon} + \cdots + \frac{y_{M_i'+m'}^2}{\varepsilon}, \tag{53}$$

where for the added eigenvectors the associated eigenvalues are assumed to be small and we set a small value to ε. We call (53) KPCA-based Mahalanobis distance with Gram-Schmidt orthogonalization.

6.3 Transductive Training Using Unknown Data

In Section 6.2, we discuss approximation of a subspace using the mapped basis vectors $\phi(\boldsymbol{e}_1), \cdots, \phi(\boldsymbol{e}_m)$. With this method, the input space is spanned if linear kernels are used, but for nonlinear kernels the whole feature space cannot be covered. Thus approximation with the basis vectors may not be enough to prevent generalization ability from decreasing.

To overcome this problem, we need to make unknown data lie in the subspace spanned by the eigenvectors associated with the covariance matrix. To do so, when an unknown datum $\boldsymbol{t}$ is given, we judge whether this datum is included

in the space spanned by $\phi(X_i')$. If the datum is not in the space, we generate the extra eigenvector in the similar way as discussed in Section 6.2. Namely, for the kernel matrix

$$K(X_i'', X_i''^T) = \phi(X_i'')\phi^T(X_i'') \tag{54}$$

where

$$\phi(X_i'') = [\phi(X_i')\, \phi(\boldsymbol{t})]^T, \tag{55}$$

we perform the Cholesky factorization. But since the factorization of $K(X_i', X_i'^T)$ can be done off-line, we only have to calculate the (M_i+m+1)th row and column of $K(X_i'', X_i''^T)$ as follows:

$$l_{M_i+m+1,j} = \frac{k_{M_i+m+1,j} - \sum_{n=1}^{j-1} l_{M_i+m+1,n}\ l_{jn}}{l_{jj}} \quad \text{for} \quad j = 1, \cdots, M_i + m, \tag{56}$$

$$l_{M_i+m+1,M_i+m+1} = (k_{M_i+m+1,M_i+m+1} - \sum_{n=1}^{M_i+m} l_{M_i+m+1,n}^2)^{\frac{1}{2}}. \tag{57}$$

If the value in the square root in (57) is larger than τ in (43), it means that the unknown datum $\boldsymbol{t}$ is not included in the space spanned by $\phi(X_i')$. Thus we generate the extra eigenvector to span the subspace.

By this method, since we do not need to calculate the whole elements, it is not time consuming. When we are not given enough number of training data, this online approximation may be effective.

7 Fuzzy Rule Tuning

7.1 Concept

In training the conventional fuzzy classifier with ellipsoidal regions, slopes of the membership functions are tuned until there is no improvement in the recognition rate. Unlike the steepest ascent method, which is one of the common ways of optimizing parameters but is very slow, this method is shown to be very robust and extremely efficient [2, 3, 10].

For the kernel fuzzy classifier with ellipsoidal regions, since we can calculate a Mahalanobis distance in the feature space using any one of (25), (31), and (53), the tuning algorithm in the input space can be readily extended to the feature space. In this section we explain the tuning algorithm.

For example, when tuning parameter α_i in (10) is increased, the slope of $m_i(\phi(\boldsymbol{x}))$ decreases, and at the same time the degree of $m_i(\phi(\boldsymbol{x}))$ increases. This may lead to correct classification of the data which were misclassified before tuning, while some data which were correctly classified may be misclassified. Here we allow the data, which were classified correctly before tuning to be misclassified as long as the overall recognition rate of the training data is improved. We divide the training data into the following four cases:

Case 1. Input $\boldsymbol{x}$ is correctly classified into class i.

Case 2. Input $\boldsymbol{x}$ is correctly classified into a class other than class i.

Case 3. Input $\boldsymbol{x}$, which belongs to class i, is misclassified into another class.

Case 4. Input $\boldsymbol{x}$, which belongs to a class other than i, is misclassified into class i.

We define $X_i (i = 1, \cdots, 4)$ as a data set in case i.

7.2 Lower and Upper Bounds of α_i

Here we consider the training data for Case 1. Before tuning, the input datum $\boldsymbol{x}$ is correctly classified into class i. While maintaining the correct classification of $\boldsymbol{x}$, we can change α_i so long as the following equation is satisfied:

$$L_i(\phi(\boldsymbol{x})) = \frac{\delta_i^2(\phi(\boldsymbol{x}))}{\min\limits_{j \neq i} h_j^2(\phi(\boldsymbol{x}))} < \alpha_i, \tag{58}$$

where $L_i(\phi(\boldsymbol{x}))$ is defined as the lower bound of α_i to maintain correct classification of $\boldsymbol{x}$.

Next, we calculate the lower bounds for all data in Case 1, and the maximum lower bound among them is defined as follows:

$$L_i(1) = \max_{\boldsymbol{x} \in X_1} L_i(\phi(\boldsymbol{x})), \tag{59}$$

where $L_i(1)$ is the lower bound that does not lead to any misclassification.

In general, $L_i(l)$ which allow $l-1$ correctly classified data to be misclassified is defined as follows:

$$L_i(l) = \max_{\boldsymbol{x} \in X_1, L_i(\phi(\boldsymbol{x})) \neq L_i(1), \cdots, L_i(l-1)} L_i(\phi(\boldsymbol{x})). \tag{60}$$

Next, we consider the training data in Case 2. In a similar way we can define $U_i(\phi(\boldsymbol{x}))$ as the upper bound of α_i to maintain correct classification of $\boldsymbol{x}$:

$$U_i(\phi(\boldsymbol{x})) = \frac{\delta_i^2(\phi(\boldsymbol{x}))}{\min\limits_{j} h_j^2(\phi(\boldsymbol{x}))}. \tag{61}$$

Then we calculate the upper bounds for all data in Case 2.

In general, we define the upper bound of α_i, $U_i(l)$, that allows $l-1$ correctly classified data to be misclassified as follows:

$$U_i(l) = \min_{\boldsymbol{x} \in X_2, U_i(\phi(\boldsymbol{x})) \neq U_i(1), \cdots, U_i(l-1)} U_i(\phi(\boldsymbol{x})). \tag{62}$$

7.3 Maximizing Margins

In SVM training, the optimal hyperplane is determined so that the margin between two classes is maximized. Thus the optimal hyperplane is determined uniquely and it has the highest generalization ability. But in fuzzy rule tuning, we do not consider margins among classes. Thus, when classes do not overlap, there are infinite possibilities for the position of the boundary. Therefore, when the number of training data is small, the generalization ability may be worsened compared to SVMs. To improve the generalization ability when the number of training data is small, we maximize margins among classes in the same way as discussed in [10]. In training if we set α_i in $[L_i(1), U_i(1)]$, it does not cause any new misclassification. Thus to maximize margins we set α_i in the middle point of $[L_i(1), U_i(1)]$.

7.4 Bounds to Resolve Misclassification

In this section we check whether the misclassified data can be correctly classified by changing α_i.

First, we consider the training data for Case 3. Thus, input $\boldsymbol{x}$ which belongs to class i is misclassified into another class (e.g., class j ($j \neq i$)). If the following equation is satisfied, input $\boldsymbol{x}$ is correctly classified:

$$\frac{\delta_i^2(\phi(\boldsymbol{x}))}{\alpha_i} < h_j^2(\phi(\boldsymbol{x})). \tag{63}$$

We define $V_i(\phi(\boldsymbol{x}))$ as the lower bound of α_i to resolve misclassification of input $\boldsymbol{x}$:

$$V_i(\phi(\boldsymbol{x})) = \frac{\delta_i^2(\phi(\boldsymbol{x}))}{\min_j h_j^2(\phi(\boldsymbol{x}))} < \alpha_i. \tag{64}$$

We define $\mathrm{Inc}(l)$ as the number of the misclassified data that are correctly classified if we set the value of α_i in $[U_i(l-1), U_i(l))$. When $V_i(\phi(\boldsymbol{x}))$ is in the range of $(\alpha_i, U_i(l))$, $\mathrm{Inc}(l)$ is increased by one. Here we define $\beta_i(l)$ as follows:

$$\beta_i(l) = \max_{V_i(\phi(\boldsymbol{x}))<U_i(l)} V_i(\phi(\boldsymbol{x})). \tag{65}$$

If α_i is set to be larger than $\max(\beta_i(l), U_i(l-1))$, $\mathrm{Inc}(l)$ misclassified data are correctly classified. But on the other hand, $l-1$ correctly classified data are misclassified.

Next we consider the training data for Case 4. Thus input $\boldsymbol{x}$ which belongs to another class (e.g., class j ($j \neq i$)) is misclassified into class i. Here the tuned distance for class j need to be the second minimum among n classes. Otherwise we may not classify $\boldsymbol{x}$ correctly by changing α_i. Then the datum $\boldsymbol{x}$ is classified correctly if the following equation is satisfied:

$$\alpha_i < \frac{\delta_i^2(\phi(\boldsymbol{x}))}{\min_j h_j^2(\phi(\boldsymbol{x}))} = K_i(\phi(\boldsymbol{x})), \tag{66}$$

where $K_i(\phi(\boldsymbol{x}))$ is the upper bound of α_i to make misclassification of $\boldsymbol{x}$ be correctly classified. We define $\mathrm{Dec}(l)$ as the number of the misclassified data that are correctly classified if we set the value of α_i in $(L_i(l), L_i(l-1)]$. When $K_i(\phi(\boldsymbol{x}))$ is in the range of $[L_i(l), \alpha_i]$, $\mathrm{Dec}(l)$ is increased by one. Here we define $\gamma_i(l)$ as follows:

$$\gamma_i(l) = \min_{K_i(\phi(\boldsymbol{x}))>L_i(l)} K_i(\phi(\boldsymbol{x})). \tag{67}$$

If α_i is set to be smaller than $\min(\gamma_i(l), L_i(l-1))$, $\mathrm{Dec}(l)$ misclassified data are correctly classified, and $l-1$ correctly classified data are misclassified.

7.5 Modification of α_i

For $\mathrm{Inc}(l)$ $(l = 1, \cdots, l_M)$, where l_M is a positive integer, we find the l which satisfies the following equation:

$$\max_l(\mathrm{Inc}(l) - l + 1). \tag{68}$$

Similarly, for $\mathrm{Dec}(l)$ $(l = 1, \cdots, l_M)$ we find the l which satisfies the following equation:

$$\max_l(\mathrm{Dec}(l) - l + 1). \tag{69}$$

If the above equations are satisfied for plural l, we select the smallest l. First we consider the case where (68) is larger or equal to (69). If we set α_i larger than $\beta_i(l)$ in the range of $(\alpha_i, U_i(l))$, the net increase of the correctly classified data is $\mathrm{Inc}(l) - l + 1$. So we set α_i in the range of $[\beta_i(l), U_i(l))$ as follows:

$$\alpha_i = \beta_i(l) + \eta(U_i(l) - \beta_i(l)), \tag{70}$$

where $0 < \eta < 1$.

Next, we consider the case where (68) is smaller than (69). If we set α_i smaller than $\gamma_i(l)$ in the range of $(L_i(l), \alpha_i)$, the net increase of the correctly classified data is $\mathrm{Dec}(l) - l + 1$. So we set α_i in the range of $[L_i(l), \gamma_i(l))$ as follows:

$$\alpha_i = \gamma_i(l) - \eta(\gamma_i(l) - L_i(l)). \tag{71}$$

Here the recognition rate of test data does not depend on the value of η. So we can change η freely, but in the experiment in Section 8 we set $\eta = 0.1$.

7.6 Tuning Procedure

According to the above discussion, we change the tuning parameter α_i as follows:

1. We set a positive number to parameter l_M, where $l_M - 1$ is the maximum number of misclassifications allowed for tuning α_i, and set a value in (0,1) to η of (70) and (71). At first we set $\alpha_i (i = 1, \cdots, n)$ to 1.

2. Next, we change α_i from $i = 1$ to $i = n$ so that the margins are maximized, which is to say we set $\alpha_i = \frac{1}{2}(L_i(1) + U_i(1))$.

3. After maximizing margins, for $\alpha_i (i = 1, \cdots, n)$ calculate $L_i(l)$, $U_i(l)$, Inc(l), Dec(l), $\beta_i(l)$ and $\gamma_i(l)$ for $l = 1, \cdots, l_M$. Find l that maximize (68) or (69) and change α_i using (70) or (71). Then we iterate the procedure of Step (3) until there is no improvement in the recognition rate of the training data.

8 Performance Evaluation

We evaluated our methods using two groups of data sets: (1) multiclass data sets used in [2, pp. 19–20] and (2) two-class data sets [18] used in [19, 20]. In the first group, each data set consists of one set of training and test data, and in the second group, each data set consists of 100 sets of training and test data. We used a Pentium III 1GHz personal computer to evaluate our method.

8.1 Multiclass Data Sets

8.1.1 Model Selection

Table 1 lists the specification of blood cell data [12], numeral data, iris data, thyroid data [13], and hiragana data [14]. For the hiragana-105 data set, since the number of input variables is very large, we normalize the kernel functions as follows:

$$K(\boldsymbol{x}, \boldsymbol{y}) = \frac{(1 + \boldsymbol{x}^T \boldsymbol{y})^d}{(m+1)^d}, \tag{72}$$

$$K(\boldsymbol{x}, \boldsymbol{y}) = \exp(-\frac{\gamma}{m} ||\boldsymbol{x} - \boldsymbol{y}||). \tag{73}$$

Table 1: Benchmark data specification

Data	Inputs	Classes	Train.	Test
Blood cell	13	12	3097	3100
Numeral	12	10	810	820
Iris	4	3	75	75
Thyroid	21	3	3772	3428
Hiragana-50	50	39	4610	4610
Hiragana-105	105	38	8375	8356
Hiragana-13	13	38	8375	8356

For the conventional fuzzy classifier with ellipsoidal regions, we need to set the value of ζ, which detects the singularity of the covariance matrix and the value of l_M, which is the maximum number of misclassification for tuning. We selected the value of ζ so that the recognition rate of the test data was maximized. And we set $l_M = 10$, which usually gives good generalization ability.

For comparison, we trained the pairwise fuzzy support vector machine [15]. For linear kernels, polynomial kernels with $d = [2, 3, 4]$, or RBF kernels with

$\gamma = [0.1, 1, 10]$, we performed 5-fold cross validation for the margin parameter $C = [1, 10, 50, 100, 500, 1000, 2000, 3000, 5000, 7000, 10000, 100000]$, and selected the kernel, its parameter, and the value of C with the highest average recognition rate.

We compared performance of the four kinds of the kernel Mahalanobis distance:

1. the conventional singular value decomposition-based kernel Mahalanobis distance (25) (denoted hereafter as "SVD");
2. the proposed KPCA-based Mahalanobis distance given by (31) in which the kernel matrix is used to select the independent vectors ("KPCA I");
3. the proposed KPCA-based Mahalanobis distance (53) by transductive training using the basis vectors ("KPCA II"); and
4. the proposed KPCA-based Mahalanobis distance by transductive training using the basis vectors and test data ("KPCA III").

Since SVD-based training took long time, we determined the value of σ in (27) so that the recognition rates of the test data was the highest. Table 2 shows the difference between the conventional method (26) and the proposed method (27) to perform singular value decomposition for the numeral data. "Initial" and "Final" denote the initial recognition rates of the test (training) data with the tuning parameters $\alpha_i = 1$ and after tuning, respectively. "Num" denotes the number of selected diagonal elements. From the table, our proposed method of singular value decomposition is effective to prevent generalization ability from decreasing, especially for initial recognition rates for linear kernels. Thus in the following experiments, we use this method to calculate a pseudo-inverse when the training data are degenerate (numeral, thyroid, and hiragana-50 data sets).

Table 2: Comparison of singular value decomposition

Type	Kernel	Initial [%]	Final [%]	Num
Conventional	Linear	88.78(90.00)	98.78(98.40)	100
($\sigma = 10^{-8}$)	$\gamma = 0.01$	98.90(99.75)	**99.02**(100)	206
Proposed	Linear	98.54(97.28)	98.78(98.52)	810
($\sigma = 10^{-8}$)	$\gamma = 0.01$	99.39(99.88)	**99.27**(100)	810

In the training of KPCA II, we used the basis vectors to approximate the subspace that the training data do not span (46). On the other hand, in the training of KPCA III, we used all the test data in addition to the basis vectors in the input space as unlabelled data, and approximated the subspace all at once on the assumption that all the test data are given at once. This was evaluated only to determine whether KPCA II gave sufficient performance. Namely, comparing KPCA II and KPCA III, we evaluated how approximation with the basis vectors

is effective even though the training data and basis vectors cannot cover all the space.

To calculate a KPCA-based Mahalanobis distance (31) or (53), we need to set two parameters, ε, which determines the minimum eigenvalue and τ, which determines how strictly we select the independent vectors. From our computational experiments, we know that the generalization ability is more sensitive to the value of ε than that of τ. In addition, the best value of ε depends on the training data sets and kernel functions.

Thus we determined the value of τ by 5-fold cross validation for the blood cell data and made the value of τ common to all training data sets and kernel functions. Then we performed 5-fold cross validation to determine the value of ε for each training data set and kernel function.

For linear kernels, polynomial kernels with $d = [2, 3]$, or RBF kernels with $\gamma = [0.001, 0.01, 0.1, 1, 10]$, we performed 5-fold cross validation for $\varepsilon = [10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}]$. We iterated cross validation 5 times to decrease the influence of random selection, and determined the value of ε with the highest average recognition rate.

Figure 2 shows the relationship between the results of 5-fold cross validation and those of the test data, in determining the value of ε for KPCA II with $d = 2$. From this figure, the results of 5-fold cross validation are almost the same with those of the test data. This means that the 5-fold cross validation is reliable to determine the value of ε. From Figure 2, we set $\varepsilon = 10^{-7}$ for blood cell data when KPCA-based methods with $d = 2$ were used.

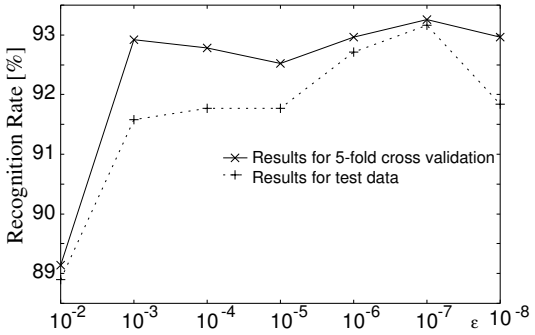

Figure 2: Relationship between the results of 5-fold cross validation and those of the test data. (KPCA II with $d = 2$)

Tables 3 and 4 show the kernels and parameters that were selected for each data set and method as discussed above, respectively. In Table 4, "Conventional" denotes the conventional fuzzy classifier with ellipsoidal regions. For KPCA I, II, and III, we determined the parameters for KPCA II and used the values for KPCA I and III.

Table 3: Kernels selected

Type	Blood	Num.	Iris	Thy.	H-50	H-105	H13
SVM	$d2$	$\gamma 1$	Linear	$d3$	$d2$	$d3$	$\gamma 10$
SVD (1)	Linear	Linear	Linear	Linear	Linear	Linear	Linear
(2)	$d2$	$d2$	$d2$	$d2$	$d2$	$d2$	$d2$
(3)	$\gamma 0.1$	$\gamma 0.01$	$\gamma 0.1$	$\gamma 0.1$	$\gamma 0.01$	$\gamma 0.01$	$\gamma 0.1$
KPCA (1)	Linear	Linear	Linear	Linear	Linear	Linear	Linear
(2)	$d2$	$d2$	$d2$	$d2$	$d2$	$d2$	$d2$
(3)	$\gamma 0.1$	$\gamma 0.01$	$\gamma 0.1$	$\gamma 0.1$	$\gamma 0.01$	$\gamma 0.01$	$\gamma 0.1$

Table 4: Parameters selected

Type (Parm.)	Blood	Num.	Iris	Thy.	H-50	H-105	H13
SVM (C)	10	10	10	10^5	5000	100	100
Conventional (ζ)	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-2}	10^{-2}	10^{-6}
SVD (1)	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-4}	10^{-4}	10^{-8}
(σ) (2)	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-2}	10^{-6}	10^{-8}
(3)	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-4}	10^{-8}	10^{-8}
KPCA (1)	10^{-4}	10^{-2}	10^{-3}	10^{-5}	10^{-3}	10^{-3}	10^{-5}
(ε) (2)	10^{-7}	10^{-2}	10^{-3}	10^{-7}	10^{-2}	10^{-5}	10^{-5}
(3)	10^{-5}	10^{-4}	10^{-4}	10^{-8}	10^{-5}	10^{-6}	10^{-6}

8.1.2 Evaluation Results

Table 5 shows the recognition rates of the test data. For each data set the highest recognition rate is shown in boldfaces.

If the training data set is not degenerate and KPCA I (1) selects independent vectors that span the input space, the conventional fuzzy classifier, SVD (1), and KPCA I (1) give the same recognition rate. Except for the numeral, thyroid, and hiragana-50 data sets, the training data sets are non-degenerate. Thus, for these sets, the recognition rates are almost the same.

For the degenerate training data sets, the recognition rates of SVD (1) and KPCA I (1) are inferior to that of the conventional fuzzy classifier. This is especially evident for the thyroid data set. Using nonlinear kernels, the recognition rates were improved but still lower for the thyroid data set. In KPCA II (1), since the basis vectors in the input space were added, the degeneracy was resolved and the recognition rates are comparable with those of the conventional fuzzy classifier. For the linear kernels there were no on-line addition of the basis vectors. Thus KPCA II (1) and KPCA III (1) give the same results. For nonlinear kernels KPCA II shows better performance than SVD. But there is no much difference between KPCA II and KPCA III. This means that the training data and the basis vectors in the input space are sufficient enough to represent the space spanned by the test data.

Performance of the SVM and that of the KPCA-based methods are comparable.

Table 5: Recognition rates of the test data [%]

Type		Blood	Num.	Iris	Thy.	H-50	H-105	H-13
SVM		92.71	**99.63**	94.67	**97.37**	**98.96**	**100**	99.78
Conventional		91.32	99.27	**97.33**	95.62	98.85	**100**	99.34
SVD	(1)	91.32	98.78	**97.33**	89.53	96.38	99.99	99.25
	(2)	92.35	97.68	94.67	90.58	95.99	99.90	98.96
	(3)	92.45	99.27	96.00	94.40	93.51	**100**	99.74
KPCA I	(1)	91.23	99.02	96.00	87.25	97.53	**100**	99.43
	(2)	92.87	99.27	**97.33**	90.14	97.25	99.99	99.86
	(3)	91.35	99.39	94.67	95.01	97.57	**100**	99.86
KPCA II	(1)	91.23	99.51	96.00	94.75	98.31	**100**	99.43
	(2)	93.16	99.51	**97.33**	96.82	97.61	**100**	99.86
	(3)	91.32	99.39	94.67	95.01	98.52	**100**	**99.88**
KPCA III	(1)	91.23	99.51	96.00	94.75	98.31	**100**	99.43
	(2)	**93.23**	99.51	**97.33**	96.76	97.55	**100**	99.86
	(3)	91.03	99.39	94.67	95.71	98.52	**100**	**99.88**

Table 6 shows the recognition rates of the test data without tuning the tuning parameters. This means that the classifier classifies a datum according to the (kernel) Mahalanobis distance. The recognition rates that are higher than those by tuning are shown in bold faces.

From Tables 5 and 6, in most cases by tuning the tuning parameters, the recognition rates of the test data are improved. The effect of tuning is especially evident for blood cell and thyroid data sets. But for the iris data set, the recognition rate does not improve by tuning for all the cases tried.

Table 7 lists the selected training data. For the SVD, the number denotes the number of selected diagonal elements. Since the numeral, thyroid, and hiragana-50 data sets are degenerate, the numbers for SVD for these data sets are equal to the numbers of the training data.

For KPCA-based methods the number denotes the number of selected independent data. For the KPCA II and KPCA III, this means the number of selected training data plus the number of added vectors by the Gram-Schmidt orthogonalization. Thus extra vectors may be generated at most n times per one vector. For example, for the blood cell data the difference of the numbers of KPCA I (2) and KPCA II (2) are 141. Thus 141 basis vectors were added in KPCA II. There are no much difference between KPCA I and KPCA II. But between KPCA II and KPCA III, in some cases many basis vectors are added but the improvement in the recognition rates is very small. This means that off-line addition of basis vectors is sufficient for the data sets tested.

Each classifier has different number of parameters to optimize. Thus, it is difficult to compare the training time of each classifier fairly. Therefore, here

Table 6: Recognition rates for test data by the Mahalanobis distance [%]

Type		Blood	Num.	Iris	Thy.	H-50	H-105	H-13
Conventional		87.45	99.63	**98.67**	86.41	98.79	100	98.36
SVD	(1)	87.45	98.54	**98.67**	74.77	80.89	99.98	98.36
	(2)	**92.42**	95.98	**97.33**	86.03	67.53	97.75	99.63
	(3)	91.58	**99.39**	**98.67**	83.46	82.43	99.99	**99.86**
KPCA I	(1)	88.65	96.34	**98.67**	72.40	94.84	100	99.15
	(2)	92.29	97.93	**98.67**	83.72	85.36	98.90	**99.90**
	(3)	89.52	97.20	**98.67**	80.25	81.13	99.96	99.84
KPCA II	(1)	88.65	99.27	**98.67**	84.92	97.85	100	99.15
	(2)	92.26	99.15	**98.67**	95.60	**97.96**	99.89	**99.90**
	(3)	89.48	99.39	**98.67**	87.95	98.50	100	99.86
KPCA III	(1)	88.65	99.27	**98.67**	84.92	97.85	100	99.15
	(2)	92.16	99.27	**98.67**	95.68	**98.42**	100	**99.89**
	(3)	89.58	99.39	**98.67**	90.32	98.48	100	99.87

we only compare the training time of fuzzy classifiers for the parameters given by Tables 3 and 4. Table 8 lists the training time of each method for the given conditions. In the table, 0 means that the calculation time is shorter than 0.5 second. Since training of the classifiers for the iris data set is very short, we do not include the result in the table.

We do not include the training time of KPCA III since all the test data were added at once. For KPCA III, we measured the time to process one unknown datum for classification by the fast on-line method, but it was too short to be measured correctly.

Training of the conventional fuzzy classifier is the shortest. And training of KPCA I and II is the second shortest. Training of SVD is the slowest due to singular value decomposition. From this table the effectiveness of our proposed methods is evident.

8.2 Two-class Data Sets

We compared recognition rates of the SVM, the conventional fuzzy classifier with ellipsoidal regions, KPCA I, and KPCA II using the two-class benchmark data sets in [18]. As a reference we used recognition performance of the SVM listed in [18]. Since the number of inputs for Splice is 60, we normalized the input using (73).

Throughout the experiment, we set $\tau = 10^{-5}$ for independent data selection. As for the determination of the values of γ and ϵ, first we fixed ϵ to 10^{-7}, and performed 5-fold cross validation of the first five training data sets for $\gamma = [10, 1, 0.1, 0.01, 0.001]$ and selected the median of the best value for each data set. Then fixing the value of γ with the median we performed 5-fold cross validation for $\epsilon = [0.5, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}]$. We also

Table 7: Number of data selected

Type		Blood	Num.	Iris	Thy.	H-50	H-105	H-13
SVD	(1)	156	810	12	3772	4610	1915	494
	(2)	1013	810	47	3772	4610	2265	2903
	(3)	638	810	30	3772	4610	2273	2253
KPCA I	(1)	156	108	12	50	1647	3521	494
	(2)	1017	420	37	308	3807	6988	3042
	(3)	545	162	19	433	2999	1779	2086
KPCA II	(1)	156	120	12	63	1951	3990	494
	(2)	1158	538	45	352	5757	10978	3499
	(3)	701	282	31	496	4949	5080	2580
KPCA III	(1)	156	120	12	63	1951	3990	494
	(2)	1260	908	45	557	51715	79821	3991
	(3)	1648	715	43	1209	45319	5101	11264

Table 8: Training time comparison [s]

Type		Blood	Num.	Thy.	H-50	H-105	H-13
Conventional		0	0	0	2	8	1
SVD	(1)	228	2	96660	98	906	991
	(2)	218	2	96120	101	933	885
	(3)	222	2	94440	100	841	983
KPCA I	(1)	2	0	31	7	52	5
	(2)	10	1	194	33	239	31
	(3)	5	0	364	21	40	21
KPCA II	(1)	2	0	31	8	69	6
	(2)	12	1	201	50	395	35
	(3)	6	0	384	33	72	23

determine the value of ζ by 5-fold cross validation. Table 9 lists the kernels and parameters selected by the above method. For Ringnorm and Titanic we performed cross validation including $d = [2, 3]$. The parameters for the SVM are from [18].

Table 10 lists the mean classification errors and the standard deviations with $\pm$. Comparing the conventional classifier and KPCA I, except for B. Cancer and F. Solar KPCA I is better than or comparable to the conventional fuzzy classifier with ellipsoidal regions. KPCA II is better than the conventional fuzzy classifier with ellipsoidal regions and better than or comparable to KPCA I. Thus, the generalization improvement of KPCA II over the conventional fuzzy classifier with ellipsoidal regions is clear. As for KPCA II and the SVM, except for Ringnorm and Splice, they are comparable. To improve the recognition performance for Ringnorm and Splice we need to do more extensive parameter survey. We tested the performance of KPCA III for these data sets, but the generalization

Table 9: Kernels and parameters selected

Data	SVM		Conventional	KPCA	
	C	$1/\sqrt{2\gamma}$	ζ	γ	ε
Banana	316.2	1	10^{-1}	10^{-1}	10^{-4}
B. Cancer	15.19	50	10^{-1}	10^{-1}	10^{-7}
Diabetes	N.A.	20	10^{-1}	10^{-2}	10^{-3}
German	3.162	55	10^{-1}	10^{-3}	0.5
Heart	3.162	120	10^{-1}	10^{-3}	0.5
Image	500	30	10^{-4}	10^{-2}	10^{-7}
Ringnorm	10^9	10	10^{-1}	$d3$	0.5
F. Solar	1.023	30	10^{-1}	10^{-3}	0.5
Splice	10^3	70	10^{-1}	10^{-3}	10^{-4}
Thyroid	10	3	10^{-1}	10^{-2}	10^{-7}
Titanic	10^5	2	10^{-1}	$d2$	10^{-2}
Twonorm	3.162	40	10^{-1}	10^{-3}	10^{-5}
Waveform	1	20	10^{-1}	10^{-3}	10^{-5}

performance was not improved. This may mean that the transductive training using the basis vectors was enough for these data sets.

9 Conclusions

In this paper we discussed a kernel fuzzy classifier with ellipsoidal regions, in which the input space is mapped into a high dimensional feature space and a fuzzy classifier is generated in the feature space.

To speedup training, we proposed the KPCA-based method to calculate a Mahalanobis distance in the feature space. To improve generalization ability when training data are degenerate, we proposed transductive training of the classifier. Namely, using the basis vectors in the input space as unlabelled data, we span the space by the Gram-Schmidt orthogonalization. We extend this method to online training.

Using the multiclass and two-class benchmark data sets, we confirmed the training speedup and improvement of the generalization ability over the conventional fuzzy classifier with ellipsoidal regions.

Acknowledgement

We are grateful to Associate Professor S. Ozawa and Assistant Professor M. Yoshimura of Kobe University for their valuable discussions. Thanks are also due to the Editor and anonymous reviewers for their constructive comments.

Table 10: Comparison among the four methods

Data	SVM	Conventional	KPCA I	KPCA II
Banana	11.5±0.7	35.8±4.2	**10.9**±0.6	**10.9**±0.6
B. Cancer	**26.0**±4.7	28.9±5.3	33.5±4.9	26.5±4.4
Diabetes	**23.5**±1.7	25.8±2.2	25.3±2.0	25.3±2.0
German	**23.6**±2.1	27.3±2.6	25.2±2.5	25.2±2.4
Heart	**16.0**±3.3	20.2±3.7	16.5±3.6	16.5±3.6
Image	3.0±0.6	11.4±1.2	3.1±0.9	**2.9**±0.7
Ringnorm	**1.7**±0.1	27.6±1.7	3.6±0.4	3.2±0.3
F. Solar	**32.4**±1.8	34.6±1.8	47.5±2.0	34.4±2.3
Splice	**10.9**±0.7	15.9±1.1	15.2±1.0	15.2±1.0
Thyroid	**4.8**±2.2	8.9±2.8	4.9±2.4	5.0±2.2
Titanic	**22.4**±1.0	23.0±1.1	23.0±1.3	22.5±1.2
Twonorm	3.0±0.2	3.6±0.4	**2.6**±0.2	**2.6**±0.3
Waveform	**9.9**±0.4	19.5±1.9	12.0±0.9	11.9±0.9

References

[1] S. K. Pal and S. Mitra, *Neuro-fuzzy Pattern Recognition: Methods in Soft Computing*, John Wiley & Sons, New York, NY, 1999.

[2] S. Abe, *Pattern Classification, Neuro-Fuzzy Method and Their Comparison*, Springer-Verlag, London, 2001.

[3] S. Abe and R. Thawonmas, "A Fuzzy Classifier with Ellipsoidal Regions," *IEEE Trans. Fuzzy Systems*, Vol. 5, No. 3, pp. 358–368, 1997.

[4] S. Abe, "Dynamic Cluster Generation for a Fuzzy Classifier with Ellipsoidal Regions," *IEEE Trans. Systems, Man, and Cybernetics—Part B*, Vol. 28, No. 6, pp. 869–876, 1998.

[5] V. Vapnik, *Statistical Learning Theory*, John Wiley & Sons, 1998.

[6] A. Ruiz and P. E. López-de-Teruel, "Nonlinear

Kernel-Based Statistical Pattern Analysis," *IEEE Trans. Neural Networks*, Vol. 12, No. 1. pp. 16–32, January 2001.

[7] B. Schölkopf, A. Smola, and K. Robert Müller, "Nonlinear Component Analysis as a Kernel Eigenvalue Problem," *Neural Computation*, 10, pp. 1299–1319, 1998.

[8] G. Baudat and F. Anouar, Kernel-Based Methods and Function Approximation, *Proc. International Joint Conference on Neural Networks (IJCNN'01)*, Vol. 2, pp. 1244–1249, 2001.

[9] K. Kaieda and S. Abe, "A Kernel Fuzzy Classifier with Ellipsoidal Regions," *Proc. International Joint Conference on Neural Networks (IJCNN'03)*, pp. 2043–2048, 2003.

[10] S. Abe and K. Sakaguchi, "Generalization Improvement of a Fuzzy Classifier with Ellipsoidal Regions," *Proc. 10th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2001)*, pp. 207–210, 2001.

[11] G. H. Golub and C. F. Van Loan, *Matrix Computation*, Johns Hopkins University Press, Baltimore, MD, Third Edition, 1996.

[12] A. Hashizume, J. Motoike, and R. Yabe, "Fully Automated Blood Cell Differential System and Its Application," *Proc. IUPAC 3rd International Congress on Automation and New Technology in the Clinical Laboratory*, pp. 297–302, 1988.

[13] ftp://ftp.ics.uci.edu/pub/machine-learning-databases/

[14] H. Takenaga et al., "Input Layer Optimization of Neural Networks by Sensitivity Analysis and Its Application to Recognition of Numerals," *Electrical Engineering in Japan*, Vol. 111, No. 4, pp. 130–138, 1991.

[15] S. Abe and T. Inoue," Fuzzy Support Vector Machines for Multiclass Problems," *Proc. European Symposium on Artificial Neural Networks (ESANN 2002)*, pp. 113–118, 2002.

[16] G. L. Cash and M. Hatamian, "Optical Character Recognition by the Method of Moments," *Computer Vision, Graphics, and Image Processing, 39*, pp. 291–310, 1989.

[17] M.-S. Lan, H. Takenaga, and S. Abe. "Character Recognition Using Fuzzy Rules Extracted from Data," *Proc. 3th IEEE International Conference on Fuzzy Systems*, Vol. 1, pp. 415-420, 1994.

[18] http://ida.first.fraunhofer.de/projects/bench/benchmarks.htm

[19] G. Rätsch, T. Onoda, K.-R. Müller, "Soft Margins for AdaBoost," *Machine Learning*, Vol. 42, No. 3, pp. 287-320, 2001.

[20] K.-R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf, "An Introduction to Kernel-Based Learning Algorithms," *IEEE Trans. Neural Networks*, Vol. 12, No. 2, pp. 181–201, 2001.