Subspace Based Support Vector Machines for Pattern Classification

Takuya Kitamura, Syogo Takeuchi, Shigeo Abe[a], Kazuhiro Fukui[b]

[a] *Graduate School of Engineering, Kobe University, Kobe, Japan*
[b] *Graduate School of Systems and Information Engineering, University of Tsukuba, Tsukuba, Japan*

Abstract

In this paper, we discuss subspace based support vector machines (SS-SVMs), in which an input vector is classified into the class with the maximum similarity. Namely, for each class we define the weighted similarity measure using the vectors called dictionaries that represent the class and optimize the weights so that the margin between classes is maximized. Because the similarity measure is defined for each class, for a data sample the similarity measure to which the data sample belongs needs to be the largest among all the similarity measures. Introducing slack variables we define these constraints either by equality constraints or inequality constraints. As a result we obtain subspace based least squares SVMs (SSLS-SVMs) and subspace based linear programming SVMs (SSLP-SVMs). To speedup training of SSLS-SVMs, which are similar to LS-SVMs by all-at-once formulation, we also propose SSLS-SVMs by one-against-all formulation, which optimize each similarity measure separately. Using two-class problems, we clarify the difference of

Email address: abe@kobe-u.ac.jp (Shigeo Abe), kfukui@cs.tsukuba.ac.jp (Kazuhiro Fukui)

SSLS-SVMs and SSLP-SVMs and evaluate the effectiveness of the proposed methods over the conventional methods with equal weights and with weights equal to eigenvalues.
Key words: Kernel methods, Least squares, Linear programming, Subspace based methods, Support vector machines

---

1. Introduction

In subspace methods (Watanabe & Pakvasa, 1973; Oja, 1983) each class region is defined by a set of basis vectors and the similarity of an input vector to a class is measured by the length of projection of the input onto the associated subspace. Similarities of the input vectors serve as discriminant functions.

Various subspace methods, such as class feature compression (CLAFIC) (Watanabe & Pakvasa, 1973) and learning subspace methods (Oja, 1983), have been proposed. In most cases, principal component analysis (PCA) is used to compute basis vectors of subspaces. The basic idea of PCA is to rotate the coordinates so that data samples are non-correlated and delete the axes that do not contribute in representing the data distribution. To extend PCA for nonlinear problems, kernel PCA (KPCA) has been proposed (Schölkopf et al., 1998, 1999). Recently, using KPCA variants of subspace methods are extended to kernel-based subspace methods, such as kernel mutual subspace methods (KMSMs) (Sakano et al., 2005; Fukui & Yamaguchi, 2007), kernel constrained mutual subspace methods (KCMSMs) (Fukui et al., 2006), and kernel orthogonal mutual subspace methods (KOSMSs) (Fukui & Yamaguchi, 2007).

In subspace methods using KPCA, we set the eigenvalues or 1 to the weights in the similarity measure of each class. However, because each subspace is defined separately and an overlap of subspaces or the margin between classes is not controlled after the definition of the subspaces, these weights may not be optimal from the standpoint of class separability. In our previous work (Kitamura et al., 2009; Takeuchi et al., 2009), we have developed least squares and linear programming support vector machines based on subspace methods.

In this paper, based on Kitamura et al. (2009); Takeuchi et al. (2009), we propose subspace based support vector machines (SS-SVMs), which optimize the weights in the similarity measure so that the margin between classes is maximized while minimizing the classification error for the training data. This is the same idea as that of support vector machines (SVMs) (Abe, 2005). We consider the similarity measure as the separating hyperplane that separates the associated class from the remaining classes and formulate the optimization problem under the linear constraints that the similarity measure associated with a data sample has the highest similarity among all the similarity measures. This formulation is the same as all-at-once SVMs, which are considered to be inefficient. However, because kernel evaluations are done when similarity measures are calculated, kernel evaluations are not necessary during optimization. We define two-types of SS-SVMs: subspace based least squares SVMs (SSLS-SVMs) and subspace based linear programming SVMs (SSLP-SVMs). For SSLS-SVMs with the quadratic objective function and the equality constraints, we derive a set of linear simultaneous equations. And we formulate SSLP-SVMs by the linear objective function and the in-

equality constraints so that they can be solved by linear programming.

To speed up training SSLS-SVMs for large data sets, we propose formulating SSLS-SVMs by one-against-all formulation. By this formulation we can optimize the weights of each class, separately.

This paper is organized as follows. In Section 2, we describe kernel-based subspace methods (KSMs) and how to calculate the similarity measures. In Section 3, we propose SSLS-SVMs and SSLP-SVMs that optimize the weights in similarity measures. In Section 4, we demonstrate the effectiveness of SSLS-SVMs and SSLP-SVMs through computer experiments. And we conclude our work in Section 5.

2. Kernel-based Subspace Methods

2.1. Kernel Principal Component Analysis

In kernel-based subspace methods, KPCA is used to represent a subspace of each class. Unlike conventional KPCA, in calculating the covariance matrix, the mean vector is not subtracted from the training data. We consider an n-class classification problem with the m-dimensional input vector $\mathbf{x}$. Let $\mathbf{x}$ be mapped into the l-dimensional feature space by the mapping function $\mathbf{g}(\mathbf{x})$. Thus, r_i dictionaries $\boldsymbol{\varphi}_{ik}$ for the subspace for class i are the eigenvectors of the following eigenvalue problem:

$$\frac{1}{|X_i|}\sum_{i\in X_i}\mathbf{g}(\mathbf{x}_j)\,\mathbf{g}^T(\mathbf{x}_j)\,\boldsymbol{\varphi}_{ik}=\lambda_{ik}\,\boldsymbol{\varphi}_{ik} \qquad (1)$$
$$\text{for} \quad i=1,...,n, \quad k=1,...,r_i,$$

where X_i is the index set for class i training data, $|X_i|$ is the number of elements in X_i, and λ_{ik} is the eigenvalue associated with $\boldsymbol{\varphi}_{ik}$. Here, we

assume that we select the first to the r_ith largest eigenvalues in (1).

To calculate (1) without using the variables in the feature space, we need to use kernel tricks. But it is time consuming. Thus to speed up calculations we use the concept of the empirical feature space (Xiong et al., 2005; Abe, 2007). The empirical feature space is spanned by the mapped training data and gives the same kernel value as that of the feature space.

Let the kernel be $H(\mathbf{x}, \mathbf{x}') = \mathbf{g}^T(\mathbf{x})\,\mathbf{g}(\mathbf{x}')$ and the number of data be M. For the M m-dimensional data $\mathbf{x}_i$ $(i = 1, \ldots, M)$, the $M \times M$ kernel matrix $H = \{H(\mathbf{x}_j, \mathbf{x}_k)\}\,(j, k = 1, ..., M)$ is symmetric and positive semidefinite. Let the rank of H be $N(\leq M)$. Then H is expressed by

$$H = USU^T, \tag{2}$$

where the column vectors of U are eigenvectors of H and S is given by

$$S = \begin{pmatrix} \sigma_1 & & 0 & \\ & \ddots & & 0_{N\times(M-N)} \\ 0 & & \sigma_N & \\ & 0_{(M-N)\times N} & & 0_{(M-N)\times(M-N)} \end{pmatrix}. \tag{3}$$

Here, $\sigma_j(> 0)$ are eigenvalues of H, whose eigenvectors correspond to the jth columns of U, and for instance $0_{(M-N)\times(M-N)}$ is the $(M - N) \times (M - N)$ zero matrix.

Defining the first N vectors of U as the $M \times N$ matrix P and Λ as the $N \times N$ diagonal matrix whose diagonal elements are $\sigma_j(j = 1, ...N)$, we can rewrite (2) as follows:

$$H = P\Lambda P^T, \tag{4}$$

where $P^T P = I_{N\times N}$ but $PP^T \neq I_{M\times M}$, and I is the unit matrix.

The mapping function to the N-dimensional empirical feature space is given by

$$\mathbf{h}(\mathbf{x}) = \Lambda^{-1/2} P^T \left(H(\mathbf{x}_1, \mathbf{x}), ..., H(\mathbf{x}_M, \mathbf{x})\right)^T . \tag{5}$$

It is proved that the empirical feature space gives the same kernel value as that of the feature space. Therefore, without inducing any error, instead of (1), we can carry out KPCA by

$$\frac{1}{|X_i|} \sum_{i \in X_i} \mathbf{h}^T(\mathbf{x}_j)\, \mathbf{h}(\mathbf{x}_j) \boldsymbol{\varphi}_{ik} = \lambda_{ik}\, \boldsymbol{\varphi}_{ik}^T. \tag{6}$$

Although the dimension of the coefficient matrix on the left-hand side of (1) may be infinite for RBF kernels, that of (6) is finite, i.e., N. Calculations of the eigenvalues and eigenvectors in (5) are not necessary if we obtain the N linearly independent data that span the empirical feature space. This can be done by the Cholesky factorization of the kernel matrix H deleting the linearly dependent data. Then, instead of (5) we use

$$\mathbf{h}(\mathbf{x}) = \left(H(\mathbf{x}_{k_1}, \mathbf{x}), ..., H(\mathbf{x}_{k_N}, \mathbf{x})\right)^T , \tag{7}$$

where $\mathbf{x}_{k_j}$ $(j = 1, ..., N)$ are linearly independent in the feature space.

For the N eigenvalues obtained by solving (6), we select, as the class i dictionaries, the first r_i largest eigenvalues and the associated eigenvectors according to the accumulation of eigenvalues. Namely, we select different dimensions for the different classes based on the cumulative proportion, $a(r_i)$, which is defined as follows:

$$a(r_i) = \frac{\sum_{j=1}^{r_i} \lambda_{ij}}{\sum_{j=1}^{N} \lambda_{ij}} \times 100\,(\%). \tag{8}$$

We set a threshold κ and determine r_i so that $a(r_i - 1) < \kappa \leq a(r_i)$.

2.2. Similarity Measures

We consider an n-class classification problem with the m-dimensional input vector $\mathbf{x}$ by kernel subspace methods in the empirical feature space. Let $\mathbf{x}$ be mapped into the r_i-dimensional subspace for class i in the N-dimensional empirical feature space mapped by $\mathbf{h}(\mathbf{x})$ and the kth dictionary for class i be $\boldsymbol{\varphi}_{ik}\,(k = 1, ..., r_i)$. Then the similarity measure is given by

$$S_i(\mathbf{x}) = \sum_{k=1}^{r_i} \frac{w_{ik}\,(\boldsymbol{\varphi}_{ik}^T\,\mathbf{h}(\mathbf{x}))^2}{\|\boldsymbol{\varphi}_{ik}\|^2\,\|\mathbf{h}(\mathbf{x})\|^2}, \tag{9}$$

where w_{ik} is the weight for the kth dictionary of class i.

Defining

$$\mathbf{f}_i(\mathbf{x}) = \left(\frac{(\boldsymbol{\varphi}_{i1}^T\,\mathbf{h}(\mathbf{x}))^2}{\|\boldsymbol{\varphi}_{i1}\|^2\,\|\mathbf{h}(\mathbf{x})\|^2}, ..., \frac{(\boldsymbol{\varphi}_{ir_i}^T\,\mathbf{h}(\mathbf{x}))^2}{\|\boldsymbol{\varphi}_{ir_i}\|^2\,\|\mathbf{h}(\mathbf{x})\|^2} \right)^T, \tag{10}$$

(9) becomes

$$S_i(\mathbf{x}) = \mathbf{w}_i^T \mathbf{f}_i(\mathbf{x}), \tag{11}$$

where $\mathbf{w}_i = (w_{i1}, ..., w_{ir_i})^T$.

Input vector $\mathbf{x}$ is classified into class

$$\arg \max_{i=1,...,n} S_i(\mathbf{x}). \tag{12}$$

3. Subspace Based Support Vector Machines

3.1. Idea

In conventional subspace methods using KPCA, we set the eigenvalues or 1 to the weights of the similarity measure of each class. However, these values

are not optimal from the standpoint of class separability, because weights are not determined to make class separability as large as possible.

We propose subspace based support vector machines (SS-SVMs) to solve this problem. Because (11) can be viewed as a decision function without a bias term we can borrow the idea of SVMs, namely maximizing margins between classes. But, unlike the decision functions of conventional SVMs, (11) is defined for each class. Thus we need to formulate SS-SVMs so that for a data sample the similarity measure for the associated class is the largest. This leads to formulating SS-SVMs by all-at-once formulation. To alleviate the computational burden of all-at-once formulation we also formulate SS-SVMs by one-against-all formulation. In the following we discuss SSLS-SVMs by all-at-once formulation and by one-against-all formulation and SSLP-SVMs by all-at-once formulation.

3.2. Subspace Based LS-SVMs by All-at-Once Formulation

Equation (11) can be viewed as the separating hyperplane in the class i subspace given by $\mathbf{f}_i(\mathbf{x})$, and it separates class i data from those belonging to other classes. Thus, we can maximize the margin in the subspace by minimizing $\|\mathbf{w}_i\|$. But, unlike the decision functions of conventional SVMs, each class has its decision function. Thus we need to formulate subspace based support vector machines according to all-at-once formulation.

Let for an n class problem M training data pairs be $\{(\mathbf{x}_1, y_1),..., (\mathbf{x}_M, y_M)\}$, where $\mathbf{x}_i$ and y_i are the m-dimensional input vectors and the associated class labels, respectively, and $y_i \in \{1, ..., n\}$. We formulate the SSLS-SVM with

all-at once formulation as follows:

$$\begin{aligned} \text{minimize} \quad & Q(\mathbf{w}, \boldsymbol{\xi}) = \frac{1}{2}\sum_{i=1}^{n} \|\mathbf{w}_i\|^2 \\ & + \sum_{i=1}^{n} \sum_{\substack{y_j \neq i, \\ j=1}}^{M} \frac{CM}{2n\,|X_{y_j}|} \xi_{ij}^2 \qquad (13) \\ \text{subject to} \quad & \mathbf{w}_{y_j}^T \mathbf{f}_{y_j}(\mathbf{x}_j) - \mathbf{w}_i^T \mathbf{f}_i(\mathbf{x}_j) = 1 - \xi_{ji} \\ & \text{for} \quad i \neq y_j, i = 1, ..., n, j = 1, ..., M, \qquad (14) \end{aligned}$$

where C is the margin parameter that determines the tradeoff between maximizing margins and minimizing misclassifications and ξ_{ij} are non-negative slack variables. The term $C\,M/(n\,|X_{y_j}|)$ in (13) is to avoid biased penalties for the unbalanced class data. In kernel subspace methods, the weights are assumed to be non-negative, but here we do not impose non-negativeness to increase freedom of solutions.

We solve the above optimization problem in the primal form. Substituting (14) into (13), we obtain

$$\begin{aligned} Q(\mathbf{w}, \xi) \;=\; & \frac{1}{2}\sum_{i=1}^{n} \|\mathbf{w}_i\|^2 + \sum_{i=1}^{n} \sum_{\substack{y_j \neq i, \\ j=1}}^{M} \frac{CM}{2n\,|X_{y_j}|} \\ & \times (1 - (\mathbf{w}_{y_j}^T \mathbf{f}_{y_j}(\mathbf{x}_j) - \mathbf{w}_i^T \mathbf{f}_i(\mathbf{x}_j)))^2. \qquad (15) \end{aligned}$$

Taking the partial derivative of (15) with respect to $\mathbf{w}_i$ and setting the resulting equation to $\mathbf{0}$, we obtain

$$\mathbf{w} \;=\; \begin{pmatrix} A_1 & B_{12} & \cdots & B_{1n} \\ B_{21} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ B_{n1} & \cdots & \cdots & A_n \end{pmatrix}^{-1} \mathbf{a}, \qquad (16)$$

where $\mathbf{w} = (\mathbf{w}_1, ..., \mathbf{w}_n)^T$, $\mathbf{a} = (\mathbf{a}_1, ..., \mathbf{a}_n)^T$, and

$$
\begin{aligned}
A_i &= \frac{n}{MC} I_{r_i} + \sum_{j=1, y_j \neq i}^{M} \frac{1}{|X_{y_j}|} \mathbf{f}_i(\mathbf{x}_j)\, \mathbf{f}_i^T(\mathbf{x}_j) \\
&\qquad \text{for} \quad i = 1, \ldots, n, \qquad (17) \\
B_{iy_j} &= -\sum_{j=1, y_j \neq i}^{M} \frac{1}{|X_{y_j}|} \mathbf{f}_i(\mathbf{x}_j)\, \mathbf{f}_{y_j}^T(\mathbf{x}_j) \\
&\qquad \text{for} \quad i \neq y_j, i = 1, \ldots, n, \qquad (18) \\
\mathbf{a}_i &= (n-1) \sum_{j=1, y_j = i}^{M} \frac{1}{|X_{y_j}|} \mathbf{f}_i(\mathbf{x}_j) \\
&\quad - \sum_{j=1, y_j \neq i}^{M} \frac{1}{|X_{y_j}|} \mathbf{f}_i(\mathbf{x}_j) \quad \text{for } i = 1, \ldots, n. \qquad (19)
\end{aligned}
$$

The size of the matrix that needs to be solved in (16) is $\sum_i^n r_i$. Thus, as the number of classes or the number of dictionaries increases, training becomes slow.

In our study we use RBF kernels: $\exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2)$ where γ is the width of the radius. Then, before training SSLS-SVMs, we need to determine the γ value, and the threshold value of the cumulative proportion, κ, and the value of margin parameter C. Because it is inefficient to determine the three values by cross-validation, we first determine γ value with $\kappa = 99.9\,(\%)$ changing the value of C by cross-validation. Then we determine κ and C values by cross-validation.

In addition to the above, to compare the proposed method with the conventional kernel subspace methods, first we determine γ and κ values for kernel subspace methods by fivefold cross-validation. Then, using these values, we optimize the C value for SSLS-SVMs by fivefold cross-validation. For this case, the algorithm of training SSLS-SVMs by all-at-once formulation is

as follows.

Algorithm 1

Step 1 Determine the γ and κ values for kernel subspace methods with equal weights or weights equal to eigenvalues by fivefold cross-validation. For the determined subspaces, determine the C value for the SSLS-SVM by fivefold cross-validation.

Step 2 Using the parameter values determined in Step 1, select the linearly independent data from the training data by the Cholesky factorization.

Step 3

Generate the mapping function to the empirical feature space using the linearly independent data obtained in Step 2. Calculate eigenvectors $\boldsymbol{\varphi}_{ik}$ and eigenvalues λ_{ik} for class i $(i = 1, \ldots, n)$ using (6).

Step 4 Determine the dimension of the subspace for class i, r_i, using the κ value determined in Step 1.

Step 5 Calculate $\mathbf{f}_i(\mathbf{x}_j)$ for $i = 1, \ldots, n$, $j = 1, \ldots, M$.

Step 6 Calculate weights $\mathbf{w}$ using (16).

3.3. Subspace Based LS-SVMs by One-against-All Formulation

We can optimize the weights in the similarity measures by the SSLS-SVM by all-at-once formulation. But if the size of the matrix in (16) is large, it will be difficult to train the SSLS-SVM. Therefore, to speed up training in such a situation, we consider formulating SSLS-SVMs in one-against-all

formulation, in which we separately optimize the weights of the similarity measure of each class.

To improve the classification ability which may be decreased because of approximation introduced by one-against-all formulation, instead of (11), we use the following decision function which include the class i bias term b_i:

$$S_i(\mathbf{x}) = \mathbf{w}_i \mathbf{f}_i(\mathbf{x}) + b_i. \tag{20}$$

Then for class $i\,(i = 1, \ldots, n)$ we formulate the SSLS-SVM with one-against-all formulation as follows:

$$\text{minimize} \quad Q(\mathbf{w}, \mathbf{b}, \boldsymbol{\xi}) = \frac{1}{2}||\mathbf{w}_i||^2 + \sum_{j=1}^{M} \frac{CM}{2n\,|X_{y_j}|}\xi_{ij}^2 \tag{21}$$

$$\text{subject to} \quad y_j(\mathbf{f}_i(\mathbf{x}_j) + b_i) = 1 - \xi_{ij} \quad \text{for} \;\; j = 1, ..., M. \tag{22}$$

Here, unlike ξ_{ij} in (13), ξ_{ij} in (21) is defined for $j = 1, \ldots, M$ and can be discarded when $\mathbf{w}_i$ is obtained. Thus, it may be possible to drop the subscript i in ξ_{ij}. But since they are different for different classes, we use ξ_{ij}.

We solve the above optimization problem in the primal form. Substituting (22) into (21), we obtain

$$Q(\mathbf{w}, \mathbf{b}) = \frac{1}{2}||\mathbf{w}_i||^2 + \sum_{j=1}^{M} \frac{C\,M}{2n\,|X_{y_j}|}(1 - y_j(\mathbf{f}_i(\mathbf{x}_j) + b_i))^2. \tag{23}$$

Taking the partial derivative of (23) with respect to $\mathbf{w}_i$ and b_i, and setting the resulting equation to $\mathbf{0}$, we obtain

$$\mathbf{w}_i = \Omega_i^{-1}\mathbf{a}_i', \tag{24}$$

$$b_i = \frac{1}{2}\left(\sum_{j=1}^{M} \frac{y_j}{|X_{y_j}|} - \mathbf{w}_i^T \sum_{j=1}^{M} \frac{1}{|X_{y_j}|}\mathbf{f}_i(\mathbf{x}_j)\right), \tag{25}$$

where

$$\Omega_i = \frac{n}{CM} I_{r_i} + \sum_{j=1}^{M} \frac{1}{|X_{y_j}|} \mathbf{f}_i(\mathbf{x}_j) \mathbf{f}_i^T(\mathbf{x}_j) \tag{26}$$

$$-\frac{1}{2} \sum_{j=1}^{M} \frac{1}{|X_{y_j}|} \mathbf{f}_i(\mathbf{x}_j) \sum_{j=1}^{M} \frac{1}{|X_{y_j}|} \mathbf{f}_i^T(\mathbf{x}_j), \tag{27}$$

$$\mathbf{a}_i' = \sum_{j=1}^{M} \frac{1}{|X_{y_j}|} y_j \mathbf{f}_i(\mathbf{x}_j)$$

$$-\frac{1}{M} \sum_{j=1}^{M} \frac{1}{|X_{y_j}|} y_j \sum_{j=1}^{M} \frac{1}{|X_{y_j}|} \mathbf{f}_i(\mathbf{x}_j). \tag{28}$$

Since we calculate the $r_i \times r_i$ matrixes for $i = 1, \ldots, n$, the number of matrix operations is $O(\sum_i^n r_i^3)$. While by all-at-once formulation the number of matrix operations is $O((\sum_{i=1}^n r_i)^3)$. Thus, by one-against-all formulation, the computational cost will be much cheaper. We call the SSLS-SVM by one-against-all formulation SSLS-SVM (O).

The algorithm of training SSLS-SVM (O) using the γ and κ values determined for kernel subspace methods is as follows.

Algorithm 2

Step 1 Determine the γ and κ values for the kernel subspace methods by fivefold cross-validation. And determine the value of the margin parameter for the SSLS-SVM (O) by fivefold cross-validation.

Step 2 Using the parameter values determined in Step 1, select the linearly independent data from the training data by the Cholesky factorization. Set $i = 1$.

Step 3 Calculate eigenvectors $\boldsymbol{\varphi}_{ik}$ and eigenvalues λ_{ik} for $i = 1, \ldots, n$ by (6).

Step 4 Determine the dimension of class i subspace, r_i, using the κ value determined in Step 1.

Step 5 Calculate $\mathbf{f}_i(\mathbf{x}_j)$ for $j = 1, \ldots, M$.

Step 6 Calculate weight vector $\mathbf{w}_i$ and bias term b_i using (24) and (25), respectively.

Step 7 If $i \neq n$, we set $i = i + 1$ and go to Step 3. If $i = n$, terminate the algorithm.

3.4. Subspace Based LP-SVMs by All-at-Once formulation

In this section, we formulate SSLP-SVMs. To introduce freedom into the similarity measure we use the decision function defined by (20). As will be shown in the "Experimental Results" section, in some cases inclusion of the bias term does not work well. In such cases, we delete the bias term.

We consider (20) as the separating hyperplane, in the dictionary space given by $\mathbf{f}_i(\mathbf{x})$ for class i, that separates class i data from those belonging to the other classes. Thus, minimizing $\|\mathbf{w}\|_1$ results in maximizing the margin in the dictionary space. By this definition, the difference from SVMs is that there are n distinct dictionary spaces and thus a data sample belonging to class i to be correctly classified, the value of (20) for class i must give the maximum value. This results in all-at-once formulation used for SVMs.

Accordingly, SSLP-SVMs are defined by

$$\begin{aligned} \text{minimize} \qquad & Q(\mathbf{w}, \mathbf{b}, \boldsymbol{\xi}) = \sum_{i=1}^{n} \sum_{k=1}^{r_k} w_{ik} + \\ & \sum_{j=1}^{M} \sum_{i \neq y_j, i=1}^{n} \frac{C\,M}{n\,|X_{y_j}|} \xi_{ji} \end{aligned} \tag{29}$$

$$\text{subject to} \quad \mathbf{w}_{y_j}^T \mathbf{f}_{y_j}(\mathbf{x}_j) + b_{y_j} - \mathbf{w}_i^T \mathbf{f}_i(\mathbf{x}_j) - b_i \geq 1 - \xi_{ji}$$
$$\text{for } i \neq y_j,\ i = 1, ..., n,\ j = 1, ..., M, \tag{30}$$
$$\xi_{ji} \geq 0$$
$$\text{for } i \neq y_j,\ i = 1, ..., n,\ j = 1, ..., M, \tag{31}$$
$$w_{ik} \geq 0 \text{ for } i = 1, ..., n,\ k = 1, ..., r_k, \tag{32}$$

where $\mathbf{b} = (b_1, \ldots, b_n)^T$, C is the margin parameter that determines the tradeoff between maximizing margins and minimizing misclassifications, y_j ($y_j \in \{1, \ldots, n\}$) are the class labels for the jth training data, $\xi_{ji}\,(i \neq y_j)$ are nonnegative slack variables for the jth training data for class i, and $\boldsymbol{\xi} = (\ldots, \xi_{ij}, \ldots)^T$. In (29), $M/(n\,|X_{y_j}|)$ is to set different values of the margin parameter for different classes for unbalanced training data. Namely, if $|X_{y_j}|$ is larger than those of the remaining classes, ξ_{ji} is multiplied by $C\,M/(n\,|X_{y_j}|)$. But if $|X_i|$ are the same for all classes, ξ_{ji} is multiplied by C.

Unlike regular SVMs, we make the weights nonnegative by (32). Since $\mathbf{f}_i(\mathbf{x}_j)$ are constants, the above optimization problem is equivalent to a linear all-at-once SVM with nonnegative weights.

To solve the above problem by linear programming we convert variables b_i that take negative values into the difference of nonnegative variables as follows:

$$b_i = {b_i}^+ - {b_i}^-, \tag{33}$$

where ${b_i}^+ \geq 0,\ {b_i}^- \geq 0$.

In addition, to transform inequality constraints into equality constraints, we introduce nonnegative slack variables u_{ji} ($i = 1, \ldots, n$, $i \neq y_j$, $j = 1, ..., M$). Then the optimization problem given by (29)–(32) is transformed

as follows:

$$\begin{aligned} \text{minimize} \quad & Q(\mathbf{w}, \mathbf{b}^+, \mathbf{b}^-, \boldsymbol{\xi}, \mathbf{u}) = \\ & \sum_{i=1}^{n} \sum_{k=1}^{r_k} w_{ik} + \sum_{j=1}^{M} \sum_{i \neq y_j, i=1}^{n} \frac{CM}{n\, N_{y_j}} \xi_{ji} \qquad (34) \\ \text{subject to} \quad & \mathbf{w}_{y_j}^T \mathbf{f}_{y_j}(\mathbf{x}_j) + b_{y_j}^+ - b_{y_j}^- \\ & -\mathbf{w}_i^T \mathbf{f}_i(\mathbf{x}_j) - b_i^+ + b_i^- = 1 - \xi_{ji} + u_{ji} \\ & \quad \text{for} \quad i \neq y_j,\ i = 1, ..., n,\ j = 1, \ldots, M, \qquad (35) \end{aligned}$$

where $\mathbf{u} = (\ldots, u_{ji}, \ldots)^T$. In the above optimization problem, all the variables are nonnegative and excluded from the constraints. Thus, (32) is deleted. In this formulation, the number of variables is $\sum_{i=1}^{n} r_i + (n-1)M + 2n$ and the number of constraints is $(n-1)M$.

We can solve the above optimization problem by linear programming using simplex methods or primal-dual interior-point methods.

When optimization is finished, if $w_{ik} = 0$, we assume $f_{ik}(\mathbf{x})$ does not contribute in recognition, where $f_{ik}(\mathbf{x})$ is the kth element of $\mathbf{f}_i(\mathbf{x})$. Therefore, we can delete $f_{ik}(\mathbf{x})$ from the dictionary. This means that we can carry out training and feature selection at the same time.

In training SSLP-SVMs, we need to determine the values of γ, κ, and C. Since SSLP-SVMs can perform dictionary selection during training, we set a large value to κ. In the computer experiments we set $\kappa = 99.9\,(\%)$ and make SSLP-SVMs select the optimum dictionaries.

In addition to optimizing the values of γ and C simultaneously, to make clear the improvement of the proposed method over the conventional methods with equal weights and with weights equal to the eigenvalues, we optimize weights for the parameter values optimized for conventional kernel subspace

methods. Namely, we first determine, in the empirical feature space determined by the Cholesky factorization, the kernel parameter value and the cumulative proportion for the conventional methods by fivefold cross-validation. Then using the subspaces determined by the conventional methods, we optimize the weights of the similarity measures by fivefold cross-validation. In the following we show the training algorithm for this case.

Algorithm 3

Step 1 For the conventional subspace method, determine the value of γ and cumulative proportion of eigenvalues, κ, by fivefold cross-validation. Namely, for a given value of γ, select linearly independent training data by the Cholesky factorization. Then, perform KPCA for each class and determine the subspace for a given value of κ. For all the combination of γ and κ values, select the values of γ and κ that realize the highest recognition rate for the validation data set. For the determined γ value, determine the value of C for the SSLP-SVM by fivefold cross-validation.

Step 2 Select the linearly independent data by performing the Cholesky factorization of the kernel matrix $H = H(\mathbf{x}_j, \mathbf{x}_k)\,(j, k = 1, \ldots, M)$ with the γ value determined in Step 1.

Step 3 For class i $(i = 1, \ldots, n)$, calculate eigenvectors $\boldsymbol{\varphi}_{ik}$ and eigenvalues λ_{ik} using (6).

Step 4 Using the κ value determined in Step 1, determine the number of eigenvalues, r_i, for class i $(i = 1, \ldots, n)$.

Step 5 Calculate $\mathbf{f}_i(\mathbf{x}_j)$ for $i = 1, \ldots, n$, $j = 1, \ldots, M$.

Step 6 Train the SSLP-SVM and obtain $\mathbf{w}$ and $\mathbf{b}$.

4. Experimental Results

4.1. Benchmark Data Sets and Evaluation Conditions

Unlike support vector machines, subspace methods are essentially for multiclass problems but because the two-class benchmark data sets (Rätsch et al., 2001) [1] include various types of classification problems, by evaluating the performance of all data sets we can get a clear view of classifier performance. Therefore, we used these benchmark data sets.

We compared the proposed SSLS-SVMs and SSLP-SVMS with SVMs, LS-SVMs, and the conventional kernel subspace methods (KSMs). Table 1 shows the number of inputs, training data, test data, and training and test data sets of the two-class data sets. We used RBF kernels and assumed that the diagonal element in the Cholesky factorization is zero if the argument of the square root in the diagonal element is less than or equal to 10^{-5}.

As conventional KSMs, we used (1) KSMs with weights equal to 1, KSMs (1) for short; and (2) KSMs with weights equal to eigenvalues, KSMs (E) for short. We compared SS-SVMs with KSMs in two ways. In the first method we optimized parameters of SS-SVMs and KSMs separately. In the second method, we first optimized the kernel parameter γ and the threshold of the cumulative proportion κ for KSMs by fivefold cross-validation. Then using the same values of γ and κ, we optimized the value of C for SS-SVMs. In this case, we can check how optimizing the weights improves the generalization

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

Table 1: Two-class Benchmark Data Sets

Data	Inputs	Training	Test	Sets
Banana	2	400	4900	100
B. cancer	9	200	77	100
Diabetes	8	468	300	100
German	20	700	300	100
Heart	13	170	100	100
Image	18	1300	1010	20
Ringnorm	20	400	7000	100
F. solar	9	666	400	100
Splice	60	1000	2175	20
Thyroid	5	140	75	100
Titanic	3	150	2051	100
Twonorm	20	400	7000	100
Waveform	21	400	4600	100

ability of KSMs.

We selected κ from $\{80, 85, 90, 95, 99, 99.9\}$, γ from $\{0.1, 0.5, 1, 1.5, 3, 5, 10, 15\}$, and C from $\{0.1, 0.5, 1, 3, 5, 10, 50, 100, 500, 10^3, 5 \times 10^3, 10^4, 2 \times 10^4, 5 \times 10^4, 10^5\}$ by fivefold cross-validation.

Table 2 shows the parameter values for KSMs determined by the above procedure. The γ values are the same for only three problems.

4.2. Comparison of SSLS-SVMs with KSMs

Table 3 shows the parameter values for SSLS-SVMs determined by fivefold cross-validation. In the table, "O" in the parentheses denotes one-against-all formulation and "1" and "E" denote, respectively, that the γ and κ values determined for KSM (1) and KSM (E) were used. Thus, SSLS and SSLS (O) denote that the γ, C, and κ values were determined independently from

Table 2: Parameter Values for KSMs

	KSM (1)		KSM (E)	
Data	$\kappa\,(\%)$	γ	$\kappa\,(\%)$	γ
Banana	99.9	15	99.9	15
B. cancer	99.9	0.5	85	3
Diabetes	99.9	3	80	5
German	85	3	85	10
Heart	95	1.5	80	3
Image	99	15	99.9	15
Ringnorm	99.9	0.1	99.9	15
F. solar	80	5	95	10
Splice	99	1	99	15
Thyroid	85	15	80	15
Titanic	80	0.1	95	5
Twonorm	80	0.1	80	3
Waveform	80	1.5	80	5

KSMs.

Table 3: Parameter Values for SSLS-SVMs

	SSLS			SSLS (O)			SSLS (1)	SSLS (E)	SSLS (O,1)	SSLS (O,E)
Data	γ	C	$\kappa\,(\%)$	γ	C	$\kappa\,(\%)$	C	C	C	C
Banana	15	50	99.9	15	1	99	50	50	1	1
B. cancer	3	0.1	85	1	5	80	10	0.1	5	1
Diabetes	3	10^3	95	3	50	95	10^3	1	50	0.5
German	3	10^4	80	10	500	85	10^4	0.5	100	500
Heart	1	1	80	0.5	0.1	80	10	0.1	500	0.5
Image	15	10^4	99	15	10^3	99.9	10^5	10^5	500	10^3
Ringnorm	0.1	10^4	99.9	0.1	1	80	10^4	500	1	0.1
F. solar	5	10^4	80	5	5	95	10^4	500	1	10
Splice	1	10^4	99	5	500	99	10^4	5×10^3	10^3	0.1
Thyroid	5	500	95	10	5	90	500	500	5	5
Titanic	5	0.1	80	5	5	95	500	0.1	0.5	5
Twonorm	0.1	50	80	0.5	0.1	80	50	10	1	0.1
Waveform	3	10^3	80	5	50	85	50	5×10^3	1	0.1

Table 4 shows the average recognition rates and their standard deviations of the validation data sets by SSLS-SVMs and KSMs for the first five training data sets. For each problem, the best average recognition rate is shown in boldface. And the bottom row shows the number of the best average recognition rate for each method. From this, SSLS, SSLS (1), and KSM (1) performed comparably and the methods based on eigenvalues and one-against-all formulation did not perform well.

For the ringnorm problem, by optimizing weights of KSM (1), the average recognition rates by SSLS (1) and SSLS (O, 1) were significantly improved.

But for the splice problem, optimizing weights of KSM (1) by SSLS (O,1) degraded the average recognition rate.

Table 4: Average Recognition Rates (%) and Their Standard Deviations of Validation Sets for SSLS-SVMs

Data	SSLS	SSLS (O)	SSLS (1)	SSLS (O,1)	SSLS (E)	SSLS (O,E)	KSM (1)	KSM (E)
Banana	89.7 ± 3.1	89.7 ± 3.5	89.7 ± 3.1	$\mathbf{90.1 \pm 3.2}$	89.7 ± 3.1	$\mathbf{90.1 \pm 3.2}$	89.8 ± 3.1	88.3 ± 2.9
B. cancer	75.1 ± 3.1	72.4 ± 4.4	74.4 ± 4.3	72.8 ± 4.3	75.1 ± 3.1	71.0 ± 4.6	72.6 ± 4.6	$\mathbf{75.7 \pm 3.5}$
Diabetes	73.6 ± 2.8	73.0 ± 3.3	$\mathbf{73.8 \pm 2.8}$	72.7 ± 3.9	70.6 ± 3.2	70.8 ± 3.0	73.5 ± 2.6	$\mathbf{73.8 \pm 3.0}$
German	72.9 ± 3.3	71.0 ± 3.0	72.6 ± 3.3	68.4 ± 3.0	71.0 ± 2.7	71.0 ± 3.0	$\mathbf{73.9 \pm 3.0}$	71.6 ± 2.8
Heart	83.0 ± 3.0	82.9 ± 3.8	83.2 ± 4.4	81.7 ± 5.4	$\mathbf{84.3 \pm 4.7}$	82.5 ± 4.3	82.1 ± 3.9	82.9 ± 3.9
Image	94.9 ± 1.2	94.1 ± 1.2	94.8 ± 1.0	93.7 ± 1.2	94.9 ± 1.2	94.1 ± 1.2	$\mathbf{95.7 \pm 0.9}$	88.1 ± 1.4
Ringnorm	$\mathbf{97.8 \pm 1.7}$	90.1 ± 2.5	$\mathbf{97.8 \pm 1.7}$	89.9 ± 2.5	63.2 ± 3.3	63.4 ± 3.2	52.8 ± 1.8	62.5 ± 4.0
F. solar	$\mathbf{66.1 \pm 3.5}$	65.1 ± 4.3	$\mathbf{66.1 \pm 3.5}$	62.6 ± 4.6	65.1 ± 3.8	65.1 ± 3.7	65.2 ± 2.8	64.4 ± 3.7
Splice	86.3 ± 2.1	77.1 ± 3.1	86.3 ± 2.1	57.2 ± 2.2	72.1 ± 3.1	72.4 ± 2.9	$\mathbf{87.4 \pm 2.5}$	72.0 ± 3.0
Thyroid	$\mathbf{96.3 \pm 2.3}$	96.2 ± 2.0	$\mathbf{96.3 \pm 2.4}$	96.2 ± 2.0	95.3 ± 2.5	96.0 ± 2.4	95.6 ± 2.5	96.0 ± 2.7
Titanic	79.0 ± 7.8	78.3 ± 8.1	79.0 ± 7.5	76.0 ± 5.9	78.4 ± 8.2	78.3 ± 8.1	$\mathbf{79.6 \pm 7.8}$	79.4 ± 7.0
Twonorm	$\mathbf{97.4 \pm 1.4}$	97.2 ± 1.2	$\mathbf{97.4 \pm 1.4}$	97.3 ± 1.4	97.2 ± 1.2	97.3 ± 1.2	97.1 ± 1.5	97.2 ± 1.4
Waveform	$\mathbf{89.9 \pm 2.7}$	88.2 ± 2.3	82.5 ± 3.2	81.9 ± 3.4	89.8 ± 2.2	84.2 ± 3.5	89.6 ± 3.0	89.0 ± 2.8
Best	4	0	5	1	1	1	4	2

Table 5 shows the average recognition rates and their standard deviations of the test data sets. We performed statistical test with the significant level of 5% and show the best average recognition rate in boldface, the second best in Roman, and the worst in italic. The best average recognition rates mean that there is no statistical difference among them and the second best recognition rates mean that there is no statistical difference among them but they are statistically different from at least one in the best group. And the worst recognition rates mean that they are statistically inferior to at least one

in the second best group. The bottom row of the table shows the numbers of the best, the second best, and the worst average recognition rates for each method. From this, SSLS, SSLS (1), and KSM (1) performed best, second best, third best, respectively, and KSM (E) performed the worst. From the standpoint of computational burden, SSLS-SVM (O) is better than SSLS-SVM but the classification performance of SSLS-SVM (O) was slightly worse.

Comparing Tables 4 and 5, the best or second best classifier for the validation data set coincides with the best classifier for the test data set. Therefore, the average recognition rate of the validation data set is useful in selecting the best classifier.

Table 5: Average Recognition Rates (%) and Their Standard Deviations of Test Data Sets for SSLS-SVMs

Data	SSLS	SSLS (O)	SSLS (1)	SSLS (O,1)	SSLS (E)	SSLS (O,E)	KSM (1)	KSM (E)
Banana	**88.9 ± 0.6**	88.7 ± 0.6	**88.9 ± 0.6**	**88.9 ± 0.6**	**88.9 ± 0**.6	**88.9 ± 0.6**	88.6 ± 0.6	*87.8 ± 0.7*
B. cancer	**75.1 ± 4.4**	*72.2 ± 4.6*	73.8 ± 4.6	*72.2 ± 4.6*	**75.1 ± 4.4**	*69.3 ± 4.8*	**75.0 ± 4.3**	**75.1 ± 4.4**
Diabetes	72.0 ± 2.3	*70.7 ± 2.4*	72.4 ± 2.2	72.5 ± 1.9	*71.1 ± 2.3*	*71.1 ± 2.2*	**73.5 ± 1.8**	*71.7 ± 2.3*
German	73.8 ± 2.2	*73.1 ± 2.1*	74.0 ± 2.2	*69.7 ± 2.8*	*72.7 ± 2.4*	*73.1 ± 2.1*	**75.1 ± 2.2**	73.6 ± 2.1
Heart	**82.6 ± 3.7**	**81.8 ± 3.6**	**82.6 ± 3.9**	**82.7 ± 3.5**	**82.1 ± 3.7**	**82.8 ± 3.7**	80.5 ± 3.3	**82.4 ± 3.6**
Image	95.4 ± 0.6	*94.9 ± 0.7*	95.1 ± 0.6	*94.5 ± 0.5*	95.4 ± 0.6	*94.9 ± 0.8*	**96.3 ± 0.6**	*88.1 ± 1.0*
Ringnorm	**97.6 ± 0.3**	91.8 ± 0.8	**97.6 ± 0.3**	91.5 ± 0.8	*64.7 ± 1.3*	*64.3 ± 2.3*	*76.6 ± 11.2*	*64.1 ± 2.4*
F. solar	**66.9 ± 1.6**	65.5 ± 1.8	**66.9 ± 1.6**	*62.4 ± 2.2*	65.1 ± 1.6	**65.5 ± 1.8**	65.1 ± 1.8	*63.5 ± 4.0*
Splice	86.2 ± 1.0	*75.8 ± 4.7*	86.2 ± 1.0	*57.4 ± 1.8*	*72.1 ± 1.6*	*72.3 ± 1.4*	**87.6 ± 0.8**	*72.0 ± 1.5*
Thyroid	**95.4 ± 2.1**	**95.4 ± 2.2**	**95.9 ± 2.1**	**95.5 ± 2.2**	**95.7 ± 2.1**	**95.4 ± 2.3**	**95.6 ± 2.1**	95.1 ± 2.4
Titanic	76.9 ± 1.0	**77.4 ± 0.7**	**77.2 ± 0.8**	76.9 ± 0.9	**77.3 ± 0.7**	**77.4 ± 0.7**	*76.6 ± 1.2*	**77.3 ± 0.6**
Twonorm	**97.7 ± 0.1**	**97.7 ± 0.1**	**97.7 ± 0.1**	**97.7 ± 0.1**	*97.4 ± 0.2*	*97.4 ± 0.2*	97.6 ± 0.1	*97.0 ± 0.5*
Waveform	**89.1 ± 0.8**	*87.0 ± 1.0*	*81.2 ± 0.9*	*81.7 ± 0.9*	*88.2 ± 0.6*	*87.5 ± 0.7*	88.5 ± 0.6	*88.0 ± 1.1*
B/S/W	8/5/0	4/3/6	7/5/1	4/3/6	5/2/6	5/0/8	6/5/2	3/2/8

4.3. Comparison of SSLP-SVMs with KSMs

We trained three types of SSLP-SVMs: (1) SSLP-SVM with $\kappa = 99.9\,(\%)$ and the γ and C values optimized by fivefold cross-validation; (2) SSLP-SVMs with the γ and κ values optimized by KSMs (1); and (3) SSLP-SVMs with the γ and κ values optimized by KSMs (E). The reason why we set $\kappa = 99.9\,(\%)$ for SSLP-SVMs is that we wanted to check whether the feature selection mechanism of linear programming formulation works. In training the SSLP-SVMs (E) we deleted the bias term because the recognition performance evaluated by cross-validation was better.

Table 6 shows the parameter values determined by the above procedure. The C values for SSLP (1) and SSLP (E) were determined using the γ and κ values determined for KSM (1) and KSM (E), respectively. From Tables 2 and 6, the optimal values of γ for KSM (1), KSM (E), and SSLP are different for most of the problems.

Table 7 shows the average recognition rates and their standard deviations of the validation data sets generated by the first five training data sets. For each problem, the best average recognition rate is shown in boldface. The bottom row shows the number of the best average recognition rates for each method. From this, SSLP is the best, KSM (1), the second best, and the SSLP (1) the third.

Usually, weights of 1 gave better results than those by the eigenvalues. And optimization of weights worked better for SSLP (1) than for SSLP (E). Especially, for the breast cancer and splice problems SSLP (E) performed worse than KSM (E). Namely, optimization of weights worsened classification performance. This may be due to the fact that the optimal values of γ are

Table 6: Parameter Values for SSLP-SVMs

	SSLP		SSLP (1)	SSLP (E)
Data	γ	C	C	C
Banana	5	10	5	5
B. cancer	0.5	1	1	1
Diabetes	0.5	100	10	3
German	5	10	3	50
Heart	3	0.5	3	0.5
Image	0.5	10000	100	50
Ringnorm	0.1	1000	1000	1
F. solar	0.5	1000	3	3
Splice	0.1	1000	10	1
Thyroid	0.5	50	5	5
Titanic	0.5	0.5	5	0.1
Twonorm	0.5	1	10	5
Waveform	1	50	50	1

very different for SSLP (E) and KSM (E) as seen from Tables 2 and 6.

Table 7: Average Recognition Rates (%) and Their Standard Deviations of Validation Sets for SSLP-SVMs

Data	SSLP	SSLP (1)	SSLP (E)	KSM (1)	KSM (E)
Banana	89.7 ± 2.8	89.0 ± 3.2	89.2 ± 3.3	**89.8 ± 3.1**	88.3 ± 2.9
B. cancer	74.1 ± 5.0	74.1 ± 5.0	68.1 ± 6.4	72.6 ± 4.6	**75.7 ± 3.5**
Diabetes	**74.8 ± 3.0**	71.6 ± 3.7	70.9 ± 3.0	73.5 ± 2.6	73.8 ± 3.0
German	71.6 ± 3.5	70.2 ± 5.9	71.5 ± 2.5	**73.9 ± 3.0**	71.6 ± 2.8
Heart	**83.5 ± 5.4**	82.2 ± 4.0	**83.5 ± 5.4**	82.1 ± 3.9	82.9 ± 3.9
Image	94.4 ± 1.1	83.2 ± 14.4	95.2 ± 1.0	**95.7 ± 0.9**	88.1 ± 1.4
Ringnorm	**98.4 ± 1.2**	**98.4 ± 1.2**	63.9 ± 5.7	52.8 ± 1.8	62.5 ± 4.0
F. solar	**65.6 ± 2.9**	63.0 ± 6.2	65.1 ± 3.2	65.2 ± 2.8	64.4 ± 3.7
Splice	**87.7 ± 1.8**	86.2 ± 1.4	62.4 ± 9.2	87.4 ± 2.5	72.0 ± 3.0
Thyroid	**96.6 ± 3.0**	95.7 ± 2.5	96.1 ± 2.4	95.5 ± 2.5	96.0 ± 2.7
Titanic	78.3 ± 8.6	78.7 ± 7.8	73.5 ± 9.0	**79.6 ± 7.8**	79.4 ± 7.0
Twonorm	97.4 ± 1.2	**97.5 ± 1.3**	97.1 ± 1.4	97.1 ± 1.5	97.2 ± 1.4
Waveform	89.2 ± 3.0	**90.0 ± 2.6**	89.1 ± 2.5	89.6 ± 3.0	89.0 ± 2.8
Best	6	3	1	4	1

Table 8 shows the average recognition rates and their standard deviations of the test data sets. We performed statistical test with the significant level of 5%. The bottom row shows the numbers of the best, the second best, and the worst recognition rates for each method. From this, SSLP performed best, KSM (1) the second best, and SSLP (1) the third best.

Comparing SSLP and KSM (1), SSLP performed better for seven problems. Especially for the ringnorm problem improvement was significant.

SSLP (1) and SSLP (E) performed better than KSM (1) and KSM (E) for four problems, respectively. Comparing the results in Tables 7 and 8,

improvement was decreased. Therefore, it is better to optimize the γ value for SSLP-SVM not using the value obtained for KSM (1) or KSM (E).

As seen from Tables 7 and 8, tendency to perform best is similar for the validation data sets and the test data sets and even if the classifiers that show best performance are different, the performance difference is not so large. Thus, we can select the suitable classifier according to the recognition rate of the validation data sets.

Table 8: Average Recognition Rates (%) and Their Standard Deviations of Test Data Sets for SSLP-SVMs

Data	SSLP	SSLP (1)	SSLP (E)	KSM (1)	KSM (E)
Banana	**89.0 ± 0.6**	88.6 ± 0.6	88.6 ± 0.6	88.6 ± 0.6	*87.8 ± 0.7*
B. cancer	73.3 ± 4.6	73.3 ± 4.6	*67.4 ± 5.0*	**75.0 ± 4.2**	**75.1 ± 4.3**
Diabetes	**73.5 ± 2.0**	*71.3 ± 4.9*	*70.1 ± 2.8*	**73.4 ± 1.7**	*71.7 ± 2.2*
German	*70.9 ± 6.6*	*71.2 ± 8.5*	*71.1 ± 8.5*	**75.1 ± 2.2**	73.7 ± 2.1
Heart	**83.1 ± 3.8**	**82.9 ± 3.7**	**83.1 ± 3.8**	80.4 ± 3.3	**82.4 ± 3.6**
Image	*95.1 ± 1.0*	*87.3 ± 13.5*	95.8 ± 0.8	**96.3 ± 0.6**	*88.0 ± 0.9*
Ringnorm	**98.2 ± 0.2**	**98.2 ± 0.2**	*64.1 ± 2.4*	76.5 ± 11.1	*64.1 ± 2.4*
F. solar	**64.7 ± 1.9**	63.4 ± 4.7	**65.0 ± 1.7**	**65.1 ± 1.8**	63.5 ± 3.9
Splice	**88.2 ± 0.6**	*86.7 ± 0.8*	*51.4 ± 4.5*	87.6 ± 0.8	*71.9 ± 1.5*
Thyroid	**96.3 ± 2.2**	95.3 ± *3.8*	**96.1 ± 2.1**	95.6 ± 2.0	95.0 ± 2.4
Titanic	76.8 ± 1.1	**77.0** ± 1.7	*76.1 ± 8.9*	76.6 ± 1.2	**77.3 ± 0.6**
Twonorm	**97.6** ± 0.2	97.5 ± 0.2	*97.3 ± 0.3*	**97.6 ± 0.1**	*97.0 ± 0.5*
Waveform	89.3 ± 0.7	**89.9 ± 0.7**	*88.2 ± 1.2*	88.5 ± *5.6*	*88.0 ± 1.1*
B/S/W	8/3/2	4/5/4	3/2/8	6/7/0	3/3/7

Table 9 shows the numbers of deleted eigenvalues per class by SSLP, SSLP (1), and SSLP (E). The "Class 1" column in SSLP lists the numbers of eigenvectors for class 1 selected by setting $\kappa = 99.9\,\%$ and the next column

Table 9: The Number of Deleted Eigenvectors for SSLP-SVMs

	SSLP				SSLP (1)				SSLP (E)			
Data	Class 1	Del	Class 2	Del	Class 1	Del	Class 2	Del	Class 1	Del	Class 2	Del
Banana	67.7	56.2	73.2	62.5	123.1	104.9	128.8	108.3	123.1	104.9	128.8	108.3
B. cancer	75.6	74.8	53.5	52.6	75.6	74.8	53.5	52.6	20.2	14.7	22.7	13.3
Diabetes	99.7	88.7	103.3	92.1	229.4	192.5	159.0	116.7	19.0	3.9	54.7	18.3
German	494.6	461.4	212.8	83.7	113.0	93.2	83.0	29.0	398.8	138.8	192.3	91.7
Heart	94.4	92.7	75.6	74.2	21.2	14.3	28.1	19.4	37.3	36.1	42.0	40.7
Image	65.7	42.1	70.7	50.5	201.4	158.4	135.6	109.1	485.9	418.0	572.9	491.0
Ringnorm	21.0	18.8	85.6	84.9	21.0	18.8	85.6	84.9	225.4	186.1	199.0	114.2
F. solar	17.2	8.8	26.6	20.0	2.0	0.2	3.0	0.4	13.4	4.7	27.8	12.6
Splice	108.3	100.8	63.5	51.1	314.7	304.7	188.8	173.8	517.0	490.8	459.9	317.2
Thyroid	15.3	13.3	33.5	31.6	11.8	6.0	29.7	2.6	8.8	2.8	27.1	1.0
Titanic	7.6	6.2	9.5	8.9	1.0	0	1.0	0.1	5.3	4.2	7.7	6.7
Twonorm	187.2	186.2	190.9	189.9	1.0	0	1.0	0	120.0	88.9	121.8	5.6
Waveform	265.3	255.0	132.3	121.1	3.0	0	2.0	0	187.1	184.1	92.0	54.8

shows the number of the deleted eigenvectors by training SSLP. And the "Class 1" column in SSLP (1) lists the numbers of eigenvalues for Class 1 selected by KSM (1), and the next column lists the numbers of deleted eigenvectors by training SSLP (1). For the breast cancer, heart, and twonorm problems, the number of selected eigenvalues per class is almost one and still SSLP performed very well. Thus, for these problems, the feature selection worked well.

Comparing the number of eigenvectors for KSM (1) and KSM (E), KSM (E) needed more eigenvectors for nine problems. But by optimizing weights by SSLP-SVM (E), many eigenvectors were deleted.

4.4. Comparison of SS-SVMs with SVMs

Table 10 lists the average recognition rates of the proposed methods, KSMs, L1 SVMs, and LS-SVMs. The column "SS-SVM" shows the recognition rate of the classifier selected from SSLP and SSLS according to the higher recognitions rate of the cross-validation data set listed in Tables 4 and 7. We performed the statistical test with the significance level of 5% and the best recognition rates are shown in boldface, the second best in Roman, and the worst recognition rates in italic. The bottom row shows the numbers of the best, the second best, and worst recognition rates for each method.

Among subspace methods, SSLP shows best performance and SSLS shows the second best performance. But compared to SVM and LS-SVM, they are inferior. But as shown in the "SS-SVM" column, selecting the classifier between SSLP and SSLS by the recognition rate of the validation data set, performance improved and comparable to SVM and LS-SVM except for the german and image data sets.

5. Discussions

One of the disadvantages of support vector machines is that there is no mechanism of analyzing classification results. Namely, for two class problems the input is classified into either Class 1 or Class 2. And if an input is misclassified, we cannot explain the reason why it is misclassified. On the other hand by subspace methods using similarity measures we can analyze the classification results. Even if the input is correctly classified, by checking the highest similarity measure with the second highest similarity measure, we can check whether the classification results are firm or marginal. Or

Table 10: Comparison of Average Recognition Rates (%) and Their Standard Deviations of Test Data Sets

Data	SS-SVM	SSLS	SSLS (O)	SSLP	KSM (1)	KSM (E)	SVM	LS-SVM
Banana	89.0 ± 0.6	88.9 ± 0.6	*88.7 ± 0.6*	89.0 ± 0.6	*88.6 ± 0.6*	*87.8 ± 0.7*	**89.3 ± 0.5**	**89.4 ± 0.5**
B. cancer	**75.1 ± 4.4**	**75.1 ± 4.4**	72.2 ± 4.6	73.3 ± 4.6	**75.0 ± 4.2**	**75.1 ± 4.3**	72.4 ± 4.6	**74.0 ± 4.7**
Diabetes	73.5 ± 2.0	*72.0 ± 2.3*	*70.7 ± 2.4*	73.5 ± 2.0	73.4 ± 1.7	*71.7 ± 2.2*	76.3 ± 1.8	**76.9 ± 1.7**
German	*73.8 ± 2.2*	*73.8 ± 2.2*	*73.1 ± 2.1*	*70.9 ± 6.6*	75.1 ± 2.2	*73.7 ± 2.1*	**76.2 ± 2.2**	**76.4 ± 2.2**
Heart	**83.1 ± 3.8**	82.6 ± 3.7	81.8 ± 3.6	**83.1 ± 3.8**	*80.4 ± 3.3*	82.4 ± 3.6	**83.7 ± 3.4**	**83.8 ± 3.1**
Image	*95.4 ± 0.6*	*95.4 ± 0.6*	*94.9 ± 0.7*	*95.1 ± 1.0*	96.3 ± 0.6	*88.0 ± 0.9*	**97.3 ± 0.4**	**97.5 ± 0.3**
Ringnorm	**98.2 ± 0.2**	*97.6 ± 0.3*	*91.8 ± 0.8*	**98.2 ± 0.2**	*76.5 ± 11.1*	*64.1 ± 2.4*	97.8 ± 0.3	*96.3 ± 0.4*
F. solar	66.9 ± 1.6	66.9 ± 1.6	*65.5 ± 1.8*	*64.7 ± 1.9*	*65.1 ± 1.8*	*63.5 ± 3.9*	**67.6 ± 1.7**	66.7 ± 1.6
Splice	88.2 ± 0.6	*86.2 ± 1.0*	*75.8 ± 4.7*	88.2 ± 0.6	*87.6 ± 0.8*	*71.9 ± 1.5*	**89.2 ± 0.7**	**89.4 ± 0.7**
Thyroid	**96.3 ± 2.2**	*95.4 ± 2.1*	*95.4 ± 2.2*	**96.3 ± 2.2**	*95.6 ± 2.0*	*95.0 ± 2.4*	**96.1 ± 2.0**	**95.9 ± 2.1**
Titanic	76.9 ± 1.0	76.9 ± 1.0	**77.4 ± 0.7**	76.8 ± 1.1	76.6 ± 1.2	**77.3 ± 0.6**	**77.2** ± 1.1	**77.3** ± 1.1
Twonorm	**97.7 ± 0.1**	**97.7 ± 0.1**	**97.7 ± 0.1**	97.6 ± *0.2*	97.6 ± 0.1	*97.0 ± 0.5*	97.6 ± 0.1	*97.4 ± 0.2*
Waveform	89.1 ± 0.8	89.1 ± 0.8	*87.0 ± 1.0*	89.3 ± 0.7	*88.5 ± 5.6*	*88.0 ± 1.1*	**90.0 ± 0.4**	**89.9** ± 0.5
B/S/W	5/6/2	2/5/6	2/2/9	3/7/3	1/5/7	2/1/10	9/4/0	10/1/2

introducing the threshold to classification, we can reject classification if the difference between the first and the second similarity is within the threshold value.

Another advantage of the subspace method is that using dictionaries we can visually inspect the obtained subspace if linear kernels are used. For instance for character recognition, by displaying dictionaries for each subspace (character), we can check whether the dictionaries well represent the character. This will increase the reliability of the developed classifier.

Now we compare the computational complexity of the proposed methods with support vector machines using SSLS-SVMs and LS-SVMs as examples. To make comparison simple, we estimate the complexity of training classifiers for two-class problems with given parameter values. For an LS SVM, training is done by solving a set of linear equations with M variables, where M is the number of training data. If we solve the set of linear equation by the Cholesky factorization the number of matrix operations is $O(M^3)$. For the SSLS-SVM, first we need to select independent variables by the Cholesky factorization. If $N\,(N \leq M)$ data are selected, the number of matrix operation is $O(N^3)$ if the incremental Cholesky factorization is used. Then, we calculate the largest r_i $(i = 1, 2, r_i \leq N)$ eigenvalues. The number of matrix operations is $O(r_i\,N^2)$ if the eigenvalues are calculated from the largest to the smallest. Training of SSLS-SVM is done by solving a set of linear equations and the number of matrix operations is $O((r_1 + r_2)^3)$ for all-at-once formulation and $O(r_1^3 + r_2^3)$ for one-against-all formulation. In our experiments, we calculated the relation of M, N, r_1, and r_2. For the SSLS-SVM, $N = 0.845\,M$, $r_1 = 0.125\,N$, and $r_2 = 0.150\,N$. For the SSLS-SVM (O),

$N = 0.845\,M$, $r_1 = 0.217\,N$, and $r_2 = 0.215\,N$. And for the SSLP-SVM, $N = 0.845\,M$, $r_1 = 0.352\,N$, and $r_2 = 0.348\,N$. Here the coefficients were the averages of the 13 classification problems. Because the sizes of r_1 and r_2 were small, independent data selection and eigenvalue calculations occupied the major computation of the SS-SVMs.

For the SSLS-SVM, we need to determine the vales of κ, γ, and C by cross-validation. Thus, including cross-validation, usually, training an SSLS-SVM is slower than an LS-SVM. For the SSLP-SVM, the κ value is set to 99.9% and, two parameter values are determined.

From the computer experiments, the generalization ability of the proposed methods are comparable or better than that of the SVM for some problems but in some cases the proposed methods show inferior results. In the following we discuss the possibilities that degrade the generalization ability using a two-dimensional case with linear kernels. Figure 1 shows classification by the subspace method. Here we assume that we only use one dictionary for each class, i.e., $\boldsymbol{\psi}_1$ and $\boldsymbol{\psi}_2$. Because the projection length of the filled circle on $\boldsymbol{\psi}_1$ is shorter than that on $\boldsymbol{\psi}_2$, it is classified into Class 1. Therefore the class boundary is given by the line as shown in the figure, which bisects $\boldsymbol{\psi}_1$ and $\boldsymbol{\psi}_2$. Therefore, to realize high generalization ability, the dot product of the dictionaries $\boldsymbol{\psi}_1$ and $\boldsymbol{\psi}_2$ need to be as large as possible.

Now consider the cases where the subspace method does not work well. Figure 2 shows an example, which is separable by an SVM. But because the two subspaces overlap heavily, classification by the subspace method is difficult. This may happen with linear kernels. But using RBF kernels, this will not happen because the mapping function associated with the RBF

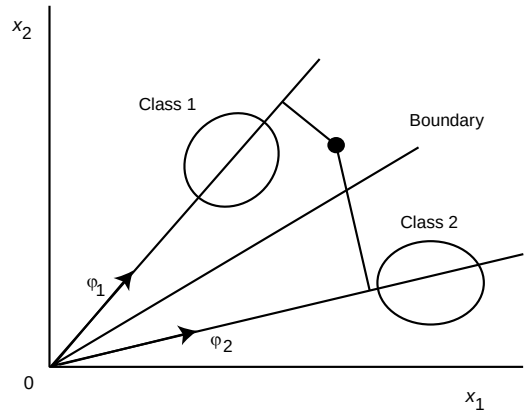

Figure 1: Classification by subspace methods

kernel maps the input onto the surface of the unit hypersphere with the center at the origin. In the feature space the training data are linearly separable by a hyperplane if the value of γ is properly set. Therefore, if the problem is difficult to classify by the subspace method it is also difficult by the SVM.

In the subspace method, the dictionaries are defined in the non-negative space. In Fig. 3, Classes 1 and 2 are in the opposite directions and thus $\boldsymbol{\psi}_1$ and $\boldsymbol{\psi}_2$ are also in the opposite directions. In this case, since the two subspace are identical, classification is impossible. In the input space, if we confine the range of variable to be non-negative, we can avoid this situation. For polynomial kernels: $(\mathbf{x}^T \mathbf{x}' + 1)^d$, where d is an integer, we can avoid this if the range of the input space is restricted to the non-negative region. For RBF kernels, because $H(\mathbf{x}, \mathbf{x}') = \mathbf{g}^T(\mathbf{x})\,\mathbf{g}(\mathbf{x}) > 0$ for any $\mathbf{x}$ and $\mathbf{x}'$, all the elements of $\mathbf{g}(\mathbf{x})$ are non-negative. Thus, for RBF kernels this problem does not happen.

From the above discussions, there seems to be no serious defect that

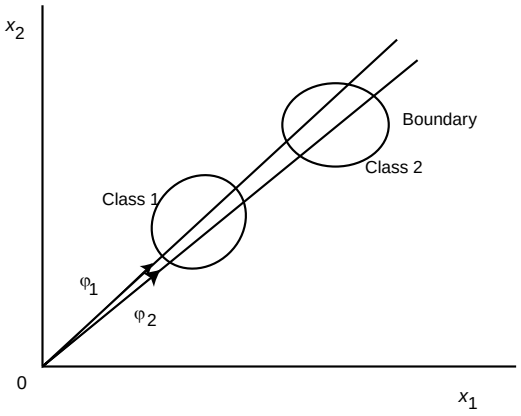

Figure 2: Difficult classification by the subspace method with close $\boldsymbol{\psi}_1$ and $\boldsymbol{\psi}_2$

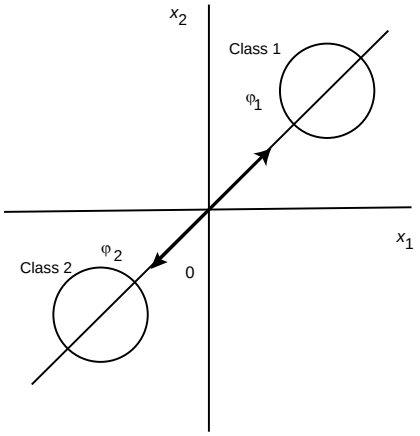

Figure 3: Difficult classification by the subspace method with two classes in the opposite directions

deteriorates the generalization ability of the subspace method. Therefore the reason of non-uniform performance compared to SVMS may be largely caused by the similarity measure given by (10). This is one of the similarity measures and a large number of similarity measures have been developed. Therefore, before developing a subspace-based classifier, it is better to select the most suitable similarity measure. This can be done by evaluating the similarity of dictionaries for each subspace to other subspaces and selecting the similarity measures that give the most non-overlapping subspaces. The proposed weight optimization method can then be used to develop the final subspace based classifier. We leave the detailed discussions to future study.

6. Conclusions

In this paper, we proposed two types of subspace based SVMs (SS-SVMs): subspace based least squares SVMs (SSLS-SVMs) and subspace based linear programming SVMs (SSLP-SVMs). In SS-SVMs, the similarity measure for each class is assumed as the separating hyperplane that separates the associated class with the remaining classes. Then the margin between classes is maximized under the constraints that the similarity measure associated with the class to which a data sample belongs is the largest among all the similarity measures. This leads to a linear all-at-once SVM. Because all-at-once formulation is inefficient, we also formulated SSLS-SVMs by one-against-all formulation.

According to the computer experiments for two-class problems, both SSLS-SVMs and SSLP-SVMs showed better recognition rates of the test data than the conventional kernel subspace method with equal weights or weights

set by the eigenvalues. We also show that although training of SSLS-SVMs by one-against-all formulation is efficient, the recognition rates of the test data were inferior to those of SSLS-SVMs by all-at-once formulation. We also showed that SSLP-SVMs can perform feature selection during training.

References

Abe, S. (2005). *Support Vector Machines for Pattern Classification*. Springer-Verlag, London.

Abe, S. (2007). Sparse least squares support vector training in the reduced empirical feature space. *Pattern Analysis and Applications*, 10 (3), 203–214.

Fukui, K., & Yamaguchi, O. (2007). The kernel orthogonal mutual subspace method and its application to 3D object recognition. In: *Proceedings of Asian Conference on Computer Vision (ACCV'07)* (pp. 467-476).

Fukui, K., Stenger, B., & Yamaguchi, O. (2006). A framework for 3D object recognition using the kernel constrained mutual subspace method. In: *Proceedings of Asian Conference on Computer Vision (ACCV'06)* (pp. 315-324).

Kitamura, T., Takeuchi, S., Abe, S., & Fukui, K. (2009). Subspace based least squares support vector machines for pattern classification. In: *Proceedings of International Joint Conference on Neural Networks (IJCNN 2009)* (pp. xx–yy).

Oja, E. (1983). *Subspace Methods of Pattern Recognition*. Research Studies Press.

Rätsch, G., Onda, T., & Müller, K.R. (2001). Soft margins for AdaBoost. *Machine Learning*, 42 (3), 287–320.

Schölkopf, B., Mika, S., Burges, C.J.C., Knirsch, P., Müller, K.R., Ratsch, C., Tsuda, K., & Smola, A.J. (1999). Input space versus feature space in kernel-based methods. *IEEE Transactions on Neural Networks*, 10 (5), 1000–1016.

Schölkopf, B., Smola, A.J., & Müller, K.R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10, 1299–1319.

Sakano, H., Mukawa, N., & Nakamura, T. (2005). Kernel mutual subspace method and its application for object recognition. *Electronics and Communication in Japan, Part 2*, 88 (6), 45–53.

Takeuchi, S., Kitamura, T., Abe, S., & Fukui, K. (2009). Subspace based linear programming support vector machines. In: *Proceedings of International Joint Conference on Neural Networks (IJCNN 2009)* (pp. xx-yy).

Teukolsky, S.A., & Press, W.H. (1993). *Numerical Recipes in C*, Cambridge University Press, pp. 430–444.

Watanabe, S., & Pakvasa, N. (1973). Subspace methods of pattern recognition. In: *Proceedings of 1st International Joint Conference on Pattern Recognition* (pp. 283-328).

Xiong, H., Swamy, M.N.S., & Ahmad, M.O. (2005). Optimizing the kernel in the empirical feature space. *IEEE Transactions on Neural Networks*, 16 (2), 460–474.