Subspace based linear programming support vector machines

Takeuchi, Syogo
Kitamura, Takuya
Abe, Shigeo
Fukui, Kazuhiro

(Citation)
Neural Networks, 2009. IJCNN 2009. International Joint Conference on:3067-3073

(Issue Date)
2009-06

(Resource Type)
conference paper

(Version)
Version of Record

(URL)
https://hdl.handle.net/20.500.14094/90000930

Subspace Based Linear Programming Support Vector Machines

Syogo Takeuchi, Takuya Kitamura, Shigeo Abe, and Kazuhiro Fukui

***Abstract*— In subspace methods, the subspace associated with a class is represented by a small number of vectors called dictionaries and using the dictionaries the similarity measure is defined and an input is classified into the class with the highest similarity. Usually, each dictionary is given an equal weight. But if subspaces of different classes overlap, the similarity measures for the overlapping regions will not give useful information for classification.**

In this paper, we propose optimizing the weights for the dictionaries using the idea of support vector machines (SVMs). Namely, first we map the input space into the empirical feature space, perform kernel principal component analysis (KPCA) for each class, and define a similarity measure. Then considering that the similarity measure corresponds to the hyperplane, we formulate the optimization problem as maximizing the margin between the class associated with the dictionaries and the remaining classes. The optimization problem results in all-at-once formulation of linear SVMs. We demonstrate the effectiveness of the proposed method with that of the conventional methods for two-class problems.

I. Introduction

In subspace methods [1] each class is considered to be confined in a specific subspace and the subspace for a class is defined by a small number of linearly independent vectors called dictionaries. In classifying an input, the similarity of the input to the set of dictionaries for each class is calculated and the input is classified into the class with the highest similarity. Because subspace methods do not directly control the overlap between classes, selection of dictionaries directly influence classification performance.

To improve classification ability various variants have been developed, such as learning subspace methods [2]. In most cases, the dictionaries are generated by principal component analysis (PCA). To improve class separability, many kernel-based subspace methods have been developed, such as kernel mutual subspace methods [3], kernel-based learning (KBL) algorithms [4], prototype reduction schemes (PRS) to optimize kernel-based nonlinear subspace methods [5], selecting the optimum dimension of subspaces using heuristic functions called overlap criterion [6], kernel constrained mutual subspace methods (KCMSM), which provide a framework for 3D object recognition [7], and kernel orthogonal subspace methods (KOMSM), which classify sets of patterns such as video frames or multi-view images [8]. Most of the methods are concentrated on the definition of the subspace so that each class separates one another but little effort is done to improve separability after subspaces are defined.

Syogo Takeuchi, Takuya Kitamura, and Shigeo Abe are with Graduate School of Engineering, Kobe University, Kobe, Japan (email: {080t233t@stu., 084t219t@stu., abe@}kobe-u.ac.jp). Kazuhiro Fukui is with Graduate School of Systems and Information Engineering, University of Tsukuba, Tsukuba, Japan (email: kfukui@cs.tsukuba.ac.jp).

To solve this problem, in this paper, we optimize the weights in the similarity measure so that the margin between classes is maximized while minimizing the classification error for the training data: the same idea as optimizing support vector machines (SVMs). Namely, 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 constraints that for a data sample belonging to a class, the associated similarity measure is maximized among the similarity measures for all the classes. This formulation is the same as all-at-once formulation, which is considered to be inefficient. But even if kernels are used, kernel evaluations are done when similarity measures are calculated. Thus, during optimization, kernel evaluations are not necessary. We use the linear objective function, instead of quadratic objective function, so that the problem can be solve by linear programming. We call this method subspace based linear programming SVMs, SSLP-SVMs for short.

The paper is organized as follows. In Section II, we summarize conventional kernel based subspace methods. And in Section III, we propose SSLP-SVMs. In Section IV, we evaluate the proposed method using two-class benchmark data sets, and in Section V we conclude our work.

II. Kernel Subspace Methods

In kernel subspace methods, each class is defined by a small number of dictionaries that define the subspace in the feature space associated with the class, and using the dictionaries a similarity measure is defined for each class. And for an input, the similarity measures are calculated for all the classes and the input is classified into the class with the maximum similarity.

Let the kth dictionary for class i in the feature space be $\boldsymbol{\varphi}_{ik}$ ($k = 1, ..., r_i$). Then the similarity measure for class i, $S_i(\mathbf{x})$, is defined by

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

where w_{ik} is the weight for the kth similarity of class i and $\mathbf{g}(\mathbf{x})$ is the mapping function that maps the input space into the feature space.

Defining

$$\mathbf{f}_i(\mathbf{x}) = \left(\frac{(\boldsymbol{\varphi}_{i\mathbf{1}}^T \, \mathbf{g}(\mathbf{x}))^2}{\|\boldsymbol{\varphi}_{i\mathbf{1}}\|^2 \, \|\mathbf{g}(\mathbf{x})\|^2}, ..., \frac{(\boldsymbol{\varphi}_{ir_i}^T \, \mathbf{g}(\mathbf{x}))^2}{\|\boldsymbol{\varphi}_{ir_i}\|^2 \, \|\mathbf{g}(\mathbf{x})\|^2} \right)^T, \tag{2}$$

(1) becomes

$$S_i(\mathbf{x}) = \mathbf{w}_i^T \mathbf{f}_i(\mathbf{x}), \tag{3}$$

978-1-4244-3553-1/09/$25.00 ©2009 IEEE

where $\mathbf{w}_i = (w_{i1}, ..., w_{ir_i})^T$. Input vector $\mathbf{x}$ is classified into class

$$\arg_i \max_{i=1,...,n} S_i(\mathbf{x}). \tag{4}$$

In generating dictionaries, PCA is often used. But, because PCA performs linear transformation of the input space, separability of the classes does not change. Thus, if different classes overlap in the input space, PCA may not give useful information for the regions where different classes overlap. Kernel PCA (KPCA) can avoid this situation by mapping the input space into the high-dimensional feature space and performing the PCA in the feature space.

In kernel subspace methods using KPCA, each class is defined by the eigenvectors of KPCA performed for the training data associated with the class, and using the eigenvectors a similarity measure is defined for each class. Unlike conventional KPCA, in calculating the covariance matrix, the mean vector is not subtracted from the training data. Thus, a dictionary $\boldsymbol{\varphi}_{ik}$ is the eigenvector of the following eigenvalue problem:

$$\frac{1}{|X_i|} \sum_{i \in X_i} \mathbf{g}(\mathbf{x}_i) \mathbf{g}^T(\mathbf{x}_i) \boldsymbol{\varphi}_{ik} = \lambda_{ik} \boldsymbol{\varphi}_{ik}, \tag{5}$$

where X_i is the index set of class i data and λ_{ik} is the eigenvalue of $\boldsymbol{\varphi}_{ik}$.

In the above calculation, we usually use kernel tricks to avoid explicit treatment of variables in the feature space because the dimension of mapping function becomes very large or for RBF kernels infinite. Instead of using kernel tricks we use the concept of empirical feature spaces [9]. In the following, we describe the procedure.

Let the kernel be $H(\mathbf{x}, \mathbf{x}') = \mathbf{g}^T(\mathbf{x}) \mathbf{g}(\mathbf{x}')$. For M training data, the associated kernel matrix is the $M \times M$ symmetric, positive semi-definite matrix $H = \{H(\mathbf{x}_j, \mathbf{x}_k)\}$ $(j, k = 1, ..., M)$ given by

$$H = USU^T, \tag{6}$$

where the column vectors of U are eigenvectors of H. Because it is orthonormal, S is expressed by

$$S = \begin{bmatrix} \sigma_1 & \cdots & 0 & \\ \vdots & \ddots & \vdots & 0_{N \times (M-N)} \\ 0 & \cdots & \sigma_N & \\ & 0_{(M-N) \times N} & & 0_{(M-N) \times (M-N)}, \end{bmatrix}, \tag{7}$$

where $\sigma_j (> 0)$ are eigenvalues of H, whose eigenvectors are the jth columns of U.

Defining the N vectors of U associated with the nonzero eigenvalues as the $M \times N$ matrix P and

$$\Lambda = \begin{bmatrix} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_N \end{bmatrix}, \tag{8}$$

we can rewrite (6) as follows:

$$H = P \Lambda P^T. \tag{9}$$

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

$$\mathbf{h}(\mathbf{x}) = \Lambda^{-1/2} P^T \left(H(\mathbf{x}_1, \mathbf{x}), \ldots, H(\mathbf{x}_M, \mathbf{x}) \right)^T. \tag{10}$$

The kernel for the empirical feature space is defined by

$$H_{\mathrm{e}}(\mathbf{x}, \mathbf{x}') = \mathbf{h}^T(\mathbf{x}) \, \mathbf{h}(\mathbf{x}'). \tag{11}$$

It is proved that

$$H(\mathbf{x}_i, \mathbf{x}_j) = H_{\mathrm{e}}(\mathbf{x}_i, \mathbf{x}_j) \quad \text{for} \quad i, j = 1, \ldots, M. \tag{12}$$

Namely, both kernels give the same values for the training data and this is extended for the case where one of the arguments is a training data sample [10]. Thus, kernel methods in the feature space can be treated in the empirical feature space without any approximation.

The covariance matrix calculated by (5) is equivalent to the following formula:

$$\frac{1}{|X_i|} \sum_{j \in X_i} \mathbf{h}(\mathbf{x}_j) \mathbf{h}^T(\mathbf{x}_j) \boldsymbol{\varphi}_{ik} = \lambda_{ik} \boldsymbol{\varphi}_{ik}. \tag{13}$$

Although the dimension of the coefficient matrix on the left-hand side of (5) is infinite for RBF kernels, we can calculate the equivalent covariance matrix using (13). But because by this method we need to calculate the eigenvalues and eigenvectors, we use the following mapping function:

$$\mathbf{h}(\mathbf{x}) = \left(H(\mathbf{x}_{k_1}, \mathbf{x}), \ldots, H(\mathbf{x}_{k_N}, \mathbf{x}) \right)^T, \tag{14}$$

where $k_i \, (i = 1, \ldots, N) \in \{1, \ldots, M\}$ and $\mathbf{x}_{k_1}, \ldots, \mathbf{x}_{k_N}$ are N training data, which are linearly independent in the feature space. We select these training data by the Cholesky factorization. Namely, during factorization of the kernel matrix, if the diagonal element becomes zero, we consider the associated training data sample is expressed by the linear combination of previously factorized training data. And we delete the associated column and row vectors and proceed the Cholesky factorization [11], [12].

For the kernel subspace methods, because each subspace is defined separately using the training data for the class, we may use different kernels and different kernel parameters. But, by this method, optimization of the kernels and the kernel parameters is difficult. Therefore, we use the same kernel and parameter value for all the subspaces.

For the given kernel and the parameter value, we need to determine the number of eigenvalues, namely the number of dictionaries, r_i, for class i. To select the number of eigenvalues, we use the cumulative proportion of eigenvalues [3]:

$$\alpha(r_i) = \frac{\sum_{j=1}^{r_i} \lambda_j}{\sum_{j=1}^{N} \lambda_j} \times 100 \, (\%). \tag{15}$$

Defining the threshold κ, which takes the value between 0 and 100, we determine r_i that satisfies $\alpha(r_i - 1) < \kappa \leq \alpha(r_i)$.

We use the same κ value in determining r_i for $i = 1, \ldots, n$, where n is the number of classes. The parameter value for a given kernel and the κ value are determined by cross-validation.

III. Subspace Based Linear Programming Support Vector Machines

In conventional subspace methods, equal weights are used for dictionaries, i.e., w_{ik} = 1, or if KPCA is used, the eigenvalues are set to w_{ik}. However, these values are not optimal from the standpoint of class separability. In this section, to solve this problem, we discuss SSLP-SVMs, which use the idea of SVMs, that is, maximizing margins between the class associated with a subspace and the remaining classes. We use a linear objective function, instead of a quadratic objective function, so that the optimization problem can be solved by linear programming.

A. Definition of SSLP-SVMs

To introduce freedom into the similarity measure given by (3), we use the following decision function, which includes the bias term b_i:

$$D_i(\mathbf{x}) = \mathbf{w}_i^T \mathbf{f}_i(\mathbf{x}) + b_i. \tag{16}$$

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.

Equation (16) can be considered as the separating hyperplane in the dictionary space given by $\mathbf{f}_i(\mathbf{x})$ for class i, and it 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 (16) for class i must give the maximum value. This results in all-at-once formulation used for SVMs.

Accordingly, SSLP-SVMs are defined by

$$\begin{aligned}
\text{minimize} \quad & Q(\mathbf{w}, \mathbf{b}, \boldsymbol{\xi}) = \sum_{i=1}^{n} \sum_{k=1}^{r_k} w_{ik} + \\
& \sum_{j=1}^{M} \sum_{i \neq y_j, i=1}^{n} \frac{C\,M}{n\,|X_{y_j}|} \xi_{ji} && (17) \\
\text{subject to} \quad & \mathbf{w}_{y_j}^T \mathbf{f}_{y_j}(\mathbf{x}_j) + b_{y_j} - \mathbf{w}_i^T \mathbf{f}_i(\mathbf{x}_j) - b_i \geq 1 - \xi_{ji} \\
& \quad \text{for } i \neq y_j,\ i = 1, ..., n,\ j = 1, ..., M, && (18) \\
& \xi_{ji} \geq 0 \\
& \quad \text{for } i \neq y_j,\ i = 1, ..., n,\ j = 1, ..., M, && (19) \\
& w_{ik} \geq 0 \text{ for } i = 1, ..., n,\ k = 1, ..., r_k, && (20)
\end{aligned}$$

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, ξ_{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 (17), $M/(n\,|X_{y_j}|)$ 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 (20). 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:

$$b_i = {b_i}^+ - {b_i}^-, \tag{21}$$

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 (17)–(20) is transformed as follows:

$$\begin{aligned}
\text{minimize} \quad & Q(\mathbf{w}, \mathbf{b}^+, \mathbf{b}^-, \boldsymbol{\xi}, \mathbf{u}) = \\
& \sum_{i=1}^{n} \sum_{k=1}^{r_k} w_{ik} + \sum_{j=1}^{M} \sum_{i \neq y_j, i=1}^{n} \frac{CM}{n\,N_{y_j}} \xi_{ji} && (22) \\
\text{subject to} \quad & \mathbf{w}_{y_j}^T \mathbf{f}_{y_j}(\mathbf{x}_j) + b_{y_j}^+ - b_{y_j}^- \\
& -\mathbf{w}_i^T \mathbf{f}_i(\mathbf{x}_j) - b_i^+ + {b_i}^- = 1 - \xi_{ji} + u_{ji} \\
& \quad \text{for} \quad i \neq y_j,\ i = 1, ..., n,\ j = 1, \ldots, M, && (23)
\end{aligned}$$

where $\mathbf{u} = (\ldots, u_{ji}, \ldots)^T$. In the above optimization problem, all the variables are nonnegative and excluded from the constraints. Thus, (20) 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.

B. Training Algorithm

In our following study we use radial basis function (RBF) kernels: $\exp(-\gamma\|\mathbf{x} - \mathbf{x}'\|^2)$, where γ is the width of the radius. 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 of for this case.

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 (13).

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 $i = 1, \ldots, n$, $j = 1, \ldots, M$ replacing $\mathbf{g}(\mathbf{x})$ by $\mathbf{h}(\mathbf{x})$ in (2).

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

IV. Experimental Results

We compared the proposed SSLP-SVMs with the conventional subspace methods using the two-class problems [13], [14] shown in Table I, which lists the numbers of inputs, training data, test data, and training and test data sets. We used RBF kernels and in selecting the linearly independent data by the Cholesky factorization, we assumed the diagonal element is zero when the argument of the square root is less than or equal to 10^{-5}. We trained the SSLP-SVMs by the simplex method [15]. As conventional kernel subspace methods (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 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). When we say SSLP-SVMs we mean the first type, and last two SSLP-SVMs are abbreviated as SSLP-SVMs (1) and SSLP-SVM (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.

We determined the values of γ and κ for KSMs (1) and KSMs (E) by fivefold cross-validation using the first five training data sets. Then for SSLP-SVMs (1) and SSLP-SVMs (E), we used the same values of κ and γ, and determined the value of C by fivefold cross-validation.

TABLE I
Benchmark Data Sets

Data	Inputs	Train.	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	140	75	100
Twonorm	20	400	7000	100
Waveform	21	400	4600	100

We selected the values of κ from $\kappa = \{80.0, 85.0, 90.0, 95.0, 99.0, 99.9\}$, C from $C = \{0.1, 0.5, 1, 3, 5, 10, 50, 100, 500, 1000, 5000, 10000, 20000, 50000, 100000\}$, and $\gamma = \{0.1, 0.5, 1, 1.5, 3, 5, 10, 15\}$ by fivefold cross-validation.

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

Table III 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. From the table, SSLP-SVM performed best for six problems and KSM (1) did for four problems. Comparing SSLP-SVM and KSM (1), SSLP-SVM performed better than KSM (1) for eight problems. Thus, SSLP-SVM performed best. SSLP-SVM (1) performed better than KSM (1) for six problems and SSLP-SVM (E) performed better than KSM (E) for six problems.

Usually, weights of 1 gave better results than those by the eigenvalues. And optimization of weights worked better for SSLP-SVM (1) than for SSLP-SVM (E). Especially, for the breast cancer and splice problems SSLP-SVM (E) performed especially 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 very different for SSLP-SVM (E) and KSM (E) as seen from Table II.

Table IV shows the average recognition rates and their standard deviations of test data sets. It also includes the results for the regular SVMs. The better results excluding those of SVMs are shown in boldface.

The best performance was obtained by SSLP-SVM and the second best was KSM (1). Comparing SSLP-SVM and KSM (1), SSLP-SVM performed better for seven problems. Especially for the ringnorm problem improvement was significant.

SSLP-SVM (1) and SSLP-SVM (E) performed better than KSM (1) and KSM (E) for four problems, respectively.

TABLE II
PARAMETER VALUES

	KSM (1)		KSM (E)		SSLP-SVM		SSLP-SVM (1)	SSLP-SVM (E)
Data	γ	κ (%)	γ	κ (%)	γ	C	C	C
Banana	15	99.9	15	99.9	5	10	5	5
B.cancer	0.5	99.9	3	85.0	0.5	1	1	1
Diabetes	3	99.9	5	80.0	0.5	100	10	3
German	3	85.0	10	85.0	5	10	3	50
Heart	0.1	95.0	3	85.0	3	0.5	3	0.5
Image	15	99.0	15	99.9	0.5	10000	100	50
Ringnorm	0.1	99.9	15	99.9	0.1	1000	1000	1
F. solar	5	80.0	10	95.0	0.5	1000	3	3
Splice	1	99.0	15	99.9	0.1	1000	10	1
Thyroid	15	85.0	15	80.0	0.5	50	5	5
Titanic	0.1	80.0	5	95.0	0.5	0.5	5	0.1
Twonorm	0.1	80.0	3	80.0	0.5	1	10	5
Waveform	1.5	80.0	5	80.0	1	50	50	1

TABLE III
AVERAGE RECOGNITION RATES (%) AND THEIR STANDARD DEVIATIONS OF VALIDATION SETS

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

Comparing the results in Tables III and IV, 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).

Comparing SVM and SSLP-SVM, SSLP-SVM performed better than or comparable to SVM for eight problems. The average recognition rate of SSLP-SVM was 5.3 lower for the german problem. Since the standard deviation was 6.6, in some cases very low recognition rates were obtained.

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

Table V shows the numbers of deleted eigenvalues per class by SSLP-SVM, SSLP-SVM (1), and SSLP-SVM (E). The "Class 1" column in SSLP-SVM lists the numbers of eigenvectors for class 1 selected by setting $\kappa = 99.9\,\%$ and the next column shows the deleted eigenvectors by training SSLP-SVM. And the "Class 1" column in SSLP-SVM (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-SVM. For the breast-cancer, heart, and twonorm problems, the number of selected eigenvalues per class is almost one and still SSLP-SVM 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 eigenvalues for nine problems. But by optimizing weights by SSLP-SVM (E), many eigenvectors were deleted.

TABLE IV
AVERAGE RECOGNITION RATES (%) AND THEIR STANDARD DEVIATIONS OF TEST DATA SETS

Data	SSLP-SVM	SSLP-SVM (1)	SSLP-SVM (E)	KSM (1)	KSM (E)	SVM
Banana	**89.0 ± 0.6**	88.6 ± 0.6	88.6 ± 0.6	88.6 ± 0.6	87.8 ± 0.7	89.3 ± 0.5
B.cancer	73.3 ± 4.6	73.3 ± 4.6	67.4 ± 5.0	75.0 ± 4.2	**75.1 ± 4.3**	72.4 ± 4.6
Diabetes	**73.5 ± 2.0**	71.3 ± 4.9	70.1 ± 2.8	73.4 ± 1.7	71.7 ± 2.2	76.3 ± 1.8
German	70.9 ± 6.6	71.2 ± 8.5	71.1 ± 8.5	**75.1 ± 2.2**	73.7 ± 2.1	76.2 ± 2.2
Heart	**83.1 ± 3.8**	82.9 ± 3.7	**83.1 ± 3.8**	80.4 ± 3.3	82.4 ± 3.6	83.7 ± 3.4
Image	95.1 ± 1.0	87.3 ± 13.5	95.8 ± 0.8	**96.3 ± 0.6**	88.0 ± 0.9	97.3 ± 0.4
Ringnorm	**98.2 ± 0.2**	**98.2 ± 0.2**	64.1 ± 2.4	76.5 ± 11.1	64.1 ± 2.4	97.8 ± 0.3
F. solar	64.7 ± 1.9	63.4 ± 4.7	65.0 ± 1.7	**65.1 ± 1.8**	63.5 ± 3.9	67.6 ± 1.7
Splice	**88.2 ± 0.6**	86.7 ± 0.8	51.4 ± 4.5	87.6 ± 0.8	71.9 ± 1.5	89.2 ± 0.7
Thyroid	**96.3 ± 2.2**	95.3 ± 3.8	96.1 ± 2.1	95.6 ± 2.0	95.0 ± 2.4	96.1 ± 2.0
Titanic	76.8 ± 1.1	77.0 ± 1.7	76.1 ± 8.9	76.6 ± 1.2	**77.3 ± 0.6**	77.2 ± 1.1
Twonorm	97.6 ± 0.2	97.5 ± 0.2	97.3 ± 0.3	**97.6 ± 0.1**	97.0 ± 0.5	97.6 ± 0.1
Waveform	89.3 ± 0.7	**89.9 ± 0.7**	88.2 ± 1.2	88.5 ± 5.6	88.0 ± 1.1	90.0 ±0.4

TABLE V
THE NUMBER OF DELETED EIGENVECTORS FOR SSLP-SVMS

	SSLP-SVM				SSLP-SVM (1)				SSLP-SVM (E)			
Data	Class1	Del	Class2	Del	Class1	Del	Class2	Del	Class1	Del	Class2	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

V. CONCLUSIONS

In this paper, we proposed subspace based linear programming SVMs (SSLP-SVMs). In SSLP-SVMs, the weights for the dictionaries are optimized based on the idea of maximizing margins. We formulate the optimization problem by linear programming and all-at-once formulation. By formulating the problem by linear programming, we can select dictionaries during training.

By the computer experiments for 13 two-class problems, we showed that the SSLP-SVM performed better than the kernel subspace method with equal weights for eight problems and optimum dictionaries were shown to be selected.

REFERENCES

[1] S. Watanabe and N. Pakvasa, "Subspace methods of pattern recognition," *Proc. 1st IJCPR*, pp. 25–32, 1973.

[2] E. Oja, *Subspace Methods of Pattern Recognition*, Research Studies Press, 1983.

[3] H. Sakano, N. Mukawa, and T. Nakamura, "Kernel mutual subspace method and its application for object recognition," *Electronics and Communication in Japan, Part 2*, vol. 88, no. 6, pp. 45–53, 2005.

[4] K.R. Müller, S. Mika, G. Ratsch, K. Tsuda, and B. Schölkopf, "An introduction to kernel-based learning algorithm," *IEEE Trans. Neural Networks*, vol. 12, no. 2, pp. 181–201, 2001.

[5] S.W. Kim and B.J. Oommen, "On using prototype reduction schemes to optimize kernel-based nonlinear subspace method," *Pattern Recognition*, vol. 37, no. 2, pp. 227–239, 2004.

[6] S.W. Kim and B.J. Oommen, "On utilizing search methods to select subspace dimensions for kernel-based nonlinear subspace classifiers," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 27, no. 1, pp. 136–141, 2005.

[7] K. Fukui, B. Stenger, and O. Yamaguchi, "A framework for 3D object recognition using the kernel constrained mutual subspace method," *Proc. ACCV06*, pp. 315–324, 2006.

[8] K. Fukui and O. Yamaguchi, "The kernel orthogonal mutual subspace method and its application to 3D object recognition," *Proc. ACCV07*, pp. 467–476, 2007.

[9] H. Xiong, M.N.S. Swamy, and M.O. Ahmad, "Optimizing the kernel in the empirical feature space," *IEEE Trans. Neural Networks*, vol. 16, no. 2, pp. 460–474, 2005.

[10] S. Abe, "Sparse least squares support vector training in the reduced

empirical feature space," *Pattern Analysis & Applications*, vol. 10, no. 3, pp. 203–214, 2007.

[11] S. Abe, *Support Vector Machines for Pattern Classification*, Springer, 2005.

[12] K. Kaieda and S. Abe, "KPCA-based training of a kernel fuzzy classifier with ellipsoidal regions," *International Journal of Approximate Reasoning*, vol. 37, no. 3, pp. 145–253, 1999.

[13] G. Rätsch, T. Onda, and K.R. Müller, "Soft margins for AdaBoost," *Machine Learning*, vol. 42, no. 3. pp. 287–320, 2001.

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

[15] S.A. Teukolsky and W.H. Press, *Numerical Recipes in C*, Cambridge University Press, pp. 430–444, 1993.