Backward Variable Selection of Support Vector Regressors by Block Deletion

Nagatani, Takashi

Abe, Shigeo

Backward Variable Selection of Support Vector Regressors by Block Deletion

Takashi Nagatani and Shigeo Abe

Abstract— **In function approximation, if datasets have many redundant input variables, various problems such as deterioration of the generalization ability and an increase of the computational cost may occur. One of the methods to solve these problems is variable selection. In pattern recognition, the effectiveness of backward variable selection by block deletion is shown. In this paper, we extend this method to function approximation. To prevent the deterioration of the generalization ability, we use the approximation error of a validation set as the selection criterion. And to reduce computational cost, during variable selection we only optimize the margin parameter by cross-validation. If block deletion fails we backtrack and start binary search for efficient variable selection. By computer experiments using some datasets, we show that our method has performance comparable with that of the conventional method and can reduce computational cost greatly. We also show that a set of input variables selected by LS-SVRs can be used for SVRs without deteriorating the generalization ability.**

I. Introduction

Function approximation estimates a continuous value for a given input based on the relationship acquired from a set of input-output pairs. Recently, as a tool to perform function approximation, Support Vector Machines (SVMs) [1], [2], [3] proposed by Vapnik attract much attention. Although SVMs are developed for learning methods for pattern recognition, they are extended to solving function approximation problems such as Support Vector Regressors (SVRs) [4] and Least Squares Support Vector Regressors (LS-SVRs) [5].

In developing a regressor, we may encounter problems such as the high computational cost caused by a large number of input variables and deterioration of the generalization ability by redundant input variables. Variable selection is one of the effective ways in reducing computational complexity and improving the generalization ability of the regressor.

The goal of variable selection is to obtain the smallest set of variables that realizes the generalization ability comparable with the original set of variables. But since it will be time consuming to use the generalization ability as the selection criterion, we usually use another selection criterion with less computational burden. This method is called a filter method [6], [7]. Although the computational cost may be small, it will take a risk of selecting a subset of input variables that may deteriorate the generalization ability of the regressor.

Recently, along with the improvement of computational power, wrapper methods, which use the generalization ability as the selection criterion, start to attract much attention. Wrapper methods provide good generalization ability but spend much computational cost [9]. To alleviate this, a combination of both methods [10], [11], [12], selecting variables during training [13], and using Gaussian processes [14] are considered.

In general, the generalization ability by wrapper methods is higher than that by filter methods. But usually, wrapper methods are not efficient. To speed up wrapper methods, backward feature selection by block deletion [15] was proposed for pattern classification. This method uses as the selection criterion the generalization ability estimated by cross-validation, which optimizes the margin parameter and the kernel parameter. And, to speed up feature selection, it deletes multiple candidate features while keeping the generalization ability high.

In this paper, we extend this method to function approximation. We also use as the selection criterion the generalization ability estimated by cross-validation. In function approximation using SVRs, we have to optimize three parameters: the margin parameter, the error threshold, and the kernel parameter. Hence cross-validation may bring a high computational cost. To speed up variable selection, at the start of variable selection we optimize the three parameters and determine the selection threshold, and during variable selection we only optimize the margin parameter.

In addition, to speed up backtracking when block deletion fails, we introduce binary search. Namely, half of the variables that are previously deleted are deleted. And if block deletion fails again, the above procedure is iterated until the deleted variables do not deteriorate the generalization ability.

To further accelerate variable selection, we empirically examine whether input variable selected by LS-SVRs can be applied to SVRs without deteriorating the generalization ability.

In Section Ⅱ, we summarize architectures of SVRs and LS-SVRs. And in Section Ⅲ, we discuss variable selection. Then, in Section Ⅳ we discuss backward variable selection for SVRs by block deletion. And in Section Ⅴ, we show the simulation results using benchmark datasets and finally in Section Ⅵ, we conclude our work.

II. Support Vector Regressors

In this section we briefly summarize architectures of SVRs and LS-SVRs.

Training of an SVR is done by solving a quadratic programming problem, which has inequality constraints [1].

In contrast to SVRs, LS-SVRs use equality constraints [5]. Thus, training of an LS-SVR can be done by solving

Takashi Nagatani and Shigeo Abe are with Graduate School of Engineering, Kobe University, Kobe 657-8501, JAPAN (email: 061t233n@stu.kobe-u.ac.jp, abe@kobe-u.ac.jp).

a set of simultaneous linear equations instead of a quadratic programming problem.

Let the M input-output pairs be $(\mathbf{x}_i, y_i)$ $(i = 1, ..., M)$ and the mapping function be $\mathbf{g}(\mathbf{x})$, in which the input vector $\mathbf{x}$ is mapped into the high dimensional feature space. Then the approximation function $f(\mathbf{x})$ is given by

$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{g}(\mathbf{x}) + b, \tag{1}$$

where $\mathbf{w}$ is the weight vector and b is the bias term. We consider determining them using an SVR or an LS-SVR.

A. Support Vector Regressors

In an SVR, we define the loss function as follows:

$$L(y, f(\mathbf{x})) = \begin{cases} 0 & \text{if } |y - f(\mathbf{x})| \le \varepsilon, \\ |y - f(\mathbf{x})| - \varepsilon & \text{otherwise,} \end{cases} \tag{2}$$

where ε is a user-defined error threshold. Now the regression problem is solved by

$$\text{minimize} \qquad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{M}(\xi_i + \xi_i^*) \tag{3}$$

$$\text{subject to} \qquad \begin{cases} y_i - \mathbf{w}^T\mathbf{g}(\mathbf{x}_i) - b \le \varepsilon + \xi_i, \\ \mathbf{w}^T\mathbf{g}(\mathbf{x}_i) + b - y_i \le \varepsilon + \xi_i^*, \\ \xi_i \ge 0, \quad \xi_i^* \ge 0 \\ \qquad\qquad \text{for} \quad i = 1, ..., M, \end{cases} \tag{4}$$

where C is the margin parameter and ξ_i and ξ_i^* are the slack variables for $\mathbf{x}_i$.

The dual problem of the SVRs is given by

$$\begin{aligned} \text{maximize} \qquad & -\frac{1}{2}\sum_{i,j=1}^{M}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\mathbf{g}^T(\mathbf{x}_i)\ \mathbf{g}(\mathbf{x}_j) \\ & -\varepsilon\sum_{i=1}^{M}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{N} y_i(\alpha_i - \alpha_i^*) \end{aligned} \tag{5}$$

$$\text{subject to} \qquad \begin{cases} \sum_{i=1}^{M}(\alpha_i - \alpha_i^*) = 0, \\ 0 \le \alpha_i, \alpha_i^* \le C, \end{cases} \tag{6}$$

where $H(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{g}^T(\mathbf{x}_i)\ \mathbf{g}(\mathbf{x}_j)$ is a kernel, and α_i and α_i^* are the Lagrange multipliers associated with $\mathbf{x}_i$. The resulting approximation function becomes

$$f(\mathbf{x}) = \sum_{i=1}^{M}(\alpha_i - \alpha_i^*)H(\mathbf{x}_i, \mathbf{x}) + b. \tag{7}$$

B. Least Squares Support Vector Regressors

In an LS-SVR, the primal problem is given as follows:

$$\text{maximize} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{C}{2}\sum_{i=1}^{M}\xi_i^2 \tag{8}$$

$$\text{subject to} \quad y_i = \mathbf{w}^T\mathbf{g}(\mathbf{x}_i) + b + \xi_i \quad \text{for} \quad i = 1, ..., M, \tag{9}$$

where ξ_i is the slack variable for $\mathbf{x}_i$.

Introducing the Lagrange multipliers α_i into (8) and (9), we obtain the following set of simultaneous linear equations for LS-SVRs:

$$\begin{pmatrix} \Omega & \mathbf{1} \\ \mathbf{1}^T & 0 \end{pmatrix}\begin{pmatrix} \boldsymbol{\alpha} \\ b \end{pmatrix} = \begin{pmatrix} \mathbf{y} \\ 0 \end{pmatrix}, \tag{10}$$

where $\mathbf{1}$ is the M-dimensional vector and

$$\Omega_{ij} = \mathbf{g}^T(\mathbf{x}_i)\ \mathbf{g}(\mathbf{x}_j) + \frac{\delta_{ij}}{C}, \tag{11}$$

$$\delta_{ij} = \begin{cases} 1 & i = j, \\ 0 & i \ne j, \end{cases} \tag{12}$$

$$\mathbf{y} = (y_1, ..., y_M)^T, \tag{13}$$

$$\mathbf{1} = (1, ..., 1)^T. \tag{14}$$

The obtained approximation function is given by

$$f(\mathbf{x}) = \sum_{i=1}^{M}\alpha_i H(\mathbf{x}_i, \mathbf{x}) + b. \tag{15}$$

III. Variable Selection

Variable selection is very important in function approximation. The generalization ability may be deteriorated if redundant variables are included. Variable selection, which solves the above problem, directly reduces the number of original input variables by selecting their subset that still retains the generalization ability comparable with that of the original input variables.

In general, variable selection methods are grouped into two categories: filter methods and wrapper methods [6]. Acquiring no feedback from a regressor, the filter method estimates the generalization ability by some indirect estimator, such as correlation coefficient [6]. On the other hand, the wrapper method directly estimates the generalization ability for the selected subset of input variables using the regressor [8].

In the wrapper methods, as a technique to select the subset of the available input variables, we usually use forward selection or backward selection, or combine both.

In the forward or backward selection, we first calculate the selection criterion using all the input variables and set this value as the stopping threshold. In the forward selection, starting from an empty set of input variables, we temporarily add one input variable, calculate the selection criterion, and permanently add the input variable that has the best selection criterion. And we repeat an addition of variables until a stopping threshold is satisfied for the obtained set of variables.

In the backward selection, starting from the initial set of input variables, we temporarily delete one input variable at a time, calculate the selection criterion, and permanently delete the variable with the best selection criterion from the set of input variables. We stop variable selection before the stopping threshold is violated.

Forward selection is faster than backward selection, but usually the former is less stable in selecting features than the latter [15].

IV. Backward Variable Selection of Support Vector Regressors by Block Deletion

Here we extend the method proposed in [15] to function approximation. In the following we discuss the approach of our extension.

- In the conventional method discussed in [15], during feature selection, the margin parameter and the kernel parameter are determined by grid search. In SVRs, we have to determine the margin parameter, kernel parameter, and error threshold. Therefore, the direct extension of the conventional method is as follows. We use grid search to determine a stopping threshold of variable selection and then we delete input variables estimating the generalization ability by grid search. We denote this cross-validation strategy "CV(ALL)." But it will be time consuming if we determine them by grid search during variable selection. To solve this problem, first we determine the stopping threshold using all the input variables optimizing the three parameters by grid search, and then during variable selection, we only optimize the margin parameter in estimating the generalization ability. We denote this cross-validation strategy "CV(C)."

- In the conventional method, if block deletion fails, we backtrack and resume backward selection but we only delete one input variable at a time. But this is inefficient if more than one variable are deletable. Therefore, we introduce binary search. Namely, if block deletion fails we backtrack and delete half of the variables previously deleted. And we iterate this procedure until block deletion succeeds. To determine the variables that are deleted after block deletion fails, we rank variables in the decreasing order of the generalization ability, which is evaluated by deleting one variable. And we select the variables with higher ranking for deletion.

- In the conventional method, an input variable is deleted temporarily and the generalization ability is estimated. If the generalization ability is worse than the stopping threshold, the input variable is excluded from the candidates of deletion afterwards. But since there is a possibility that this input variable become deletable after other variables are deleted. Therefore, we do not exclude variables from deletion candidates even if they deteriorate generalization ability at some stage of variable selection.

In the following we discuss backward variable selection for SVRs by block deletion using the generalization ability estimated by cross-validation as the selection criterion.

Step 1 Let the initial set of input variables be I^m, where m is the number of input variables, and the approximation error of the validation set by cross-validation be E^m. We delete the ith ($i = 1, ..., m$) input variable temporarily from I^m and estimate the approximation error of validation set by cross-validation. Let the error of the validation set be E_i^m. Setting $j = m$, we go to Step 2.

Step 2 Let I^j be the set of remaining j input variables. We set the set of input variables that are candidates for deletion by

$$S^j = \{i | E_i^j \leq E^m, i \in I^j\}. \tag{16}$$

If S^j is empty, we consider that there is no input variable to be deleted and stop variable selection. If only one input variable is included in S^j, this input variable can be deleted without deteriorating the generalization ability. Thus, we set $I^{j-1} = I^j - S^j$, $j \leftarrow j - 1$ and repeat Step 2. If S^j has more than two input variables, we generate a variable ranking list R^j, where R^j is ranked in the increasing order of E_i^j. And we go to Step 3.

Step 3 We delete all the variables in S^j from I^j:

$$I^k = I^j - S^j, \tag{17}$$

where $k = j - |S^j|$ and $|S^j|$ denotes the number of elements in S^j. If

$$E^k \leq E^m, \tag{18}$$

we consider that I^k is the set of input variables with the generalization ability equal to or higher than with the initial set of input variables I^m and we go to Step 2 updating j with k. If (18) is not satisfied, we go to Step 4.

Step 4 Let the variable ranking list $R^{'j}$ include the upper half elements of the variable ranking list R^j. And we set

$$I^l = I^j - \{R^{'j}\}, \tag{19}$$

where $\{R^{'j}\}$ is the set that includes all the variables in $R^{'j}$ and $l = j - |\{R^{'j}\}|$. If

$$E^l \leq E^j, \tag{20}$$

we delete input variables in $R^{'j}$. And we go to Step 2 updating j with l. If (20) is not satisfied, updating R^j with $R^{'j}$ we iterate Step 4 until (20) is satisfied.

V. Performance Evaluation

In this section, we show experimental results for SVRs and LS-SVRs. In the first experiment, using an artificial dataset, we show the effectiveness of the proposed method discussed in Section IV in comparison with the conventional backward selection method with single variable deletion. In the second experiment, we show that our proposed method is applicable to real datasets. In the third experiment, using the results of the second experiment, we examine whether the input variables selected by the LS-SVR can be used for the SVR

without deteriorating the generalization ability. Unless stated otherwise, we measure the approximation error of a regressor by the average absolute approximation error. We use an Athlon 64 XII 4800+ personal computer (2GB memory, Linux operating system) in measuring variable selection time.

A. Evaluation Conditions

We use the benchmark datasets listed in Table I. The Mackey-Glass dataset [16] is a time series dataset with chaotic behaviors. We add four artificial redundant input variables that are generated by a random variable with a uniform distribution in [0,1]. The Boston 14 dataset [17], [18], [19] predicts the average value of a home price in Boston, which is the 14th variable in the Boston dataset. The Boston dataset is not divided into training and test datasets. Therefore, we randomly divide the set into two with almost equal sizes.

For model selection and performance evaluation, we use RBF kernels:

$$H(\mathbf{x}, \mathbf{x}') = \exp(-\gamma||\mathbf{x} - \mathbf{x}'||^2), \tag{21}$$

where $\gamma(> 0)$ is a parameter for slope control, and Mahalanobis kernels:

$$H(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\delta}{m}(\mathbf{x} - \mathbf{x}')^T Q^{-1}(\mathbf{x} - \mathbf{x}')\right), \tag{22}$$

where δ is the scaling factor to control the Mahalanobis distance and Q is the full covariance matrix calculated using the training data. In the experiment, we determine the initial regressor with all the input variables using two methods of cross-validation; grid search for RBF kernels and line search for Mahalanobis kernels [19]. Using grid search for RBF kernels, we determine the margin parameter, the error threshold, and the kernel parameter by fivefold cross-validation. And using line search for Mahalanobis kernels, we determine the parameters as follows: we set $\delta = 1$ and determine the margin parameter and the error threshold by fivefold cross-validation, and setting the determined values to the margin parameter and the error threshold, determine the value of δ by fivefold cross-validation. According to [19], generalization ability of SVRs using Mahalanobis kernels by line search was comparable to, or better than that of RBF kernels by grid search.

TABLE I
SPECIFICATIONS OF DATASETS

Dataset	Inputs	Train	Test
Mackey-Glass	8	500	500
Boston 14	13	506	-

The stopping threshold is the cross-validation error for the selected model using all the input variables. In backward variable selection, we use two methods of cross-validation to estimate the generalization error; CV(ALL), in which the three parameters are determined by cross-validation and CV(C), in which only the margin parameter is determined. In cross-validation, we change $\gamma = \{0.1, 0.5, 1.0, 5.0, 10, 15, 20\}$ for RBF kernels, $\delta = \{0.1, 0.2, ..., 1.0, ..., 1.9, 2.0\}$ for Mahalanobis kernels, $\varepsilon = \{0.001, 0.01, 0.05, 0.1, 0.5, 1.0\}$, and $C = \{1, 10, 100, 1000, 5000, 10000, 100000\}$.

B. Artificial Dataset

In this section, we use the Mackey-Glass dataset. First we perform the experiment for SVRs. Tables II and III show the result for RBF kernels and Mahalanobis kernels, respectively.

The upper parts of the tables show the result of variable selection with the conventional method, in which CV(ALL) is used and one variable is deleted at a time, and the lower parts of the tables show that of the proposed method, in which CV(C) is used and block deletion of variables is adopted. Columns "γ" and "δ" list optimal kernel parameters for RBF kernels and Mahalanobis kernels, respectively. And columns "C" and "ε" list optimal margin parameter and the error threshold value, respectively. The Column "Vali." lists the approximation error for the cross-validation dataset. The column "Test" lists the approximation error for the test data measured by the Normalized Root Mean Square Error (NRMSE). The column "Deleted" lists the deleted input variables in the order of backward deletion.

TABLE II
PERFORMANCE COMPARISON FOR RBF KERNELS (SVRS)

Method	γ	C	ε	Vali.	Test	Deleted	Time[s]
	1.0	10^5	10^{-3}	0.012	0.061	-	
	5.0	10^2	10^{-3}	0.009	0.047	5	
Single	15	10^2	10^{-3}	0.006	0.029	5,7	
	5.0	10^5	10^{-3}	0.002	0.013	5,7,8	
	20	10^5	10^{-3}	0.001	0.004	5,7,8,6	
	20	10^5	10^{-3}	0.002	0.013	5,7,8,6,4	181192
	1.0	10^5	10^{-3}	0.012	0.061	-	
Block	1.0	10^5	10^{-3}	0.004	0.025	7,6,8,5	
	1.0	10^5	10^{-3}	0.006	0.034	7,6,8,5,4	8533

TABLE III
PERFORMANCE COMPARISON FOR MAHALANOBIS KERNELS (SVRS)

Method	δ	C	ε	Vali.	Test	Deleted	Time[s]
	0.6	10	10^{-2}	0.015	0.070	-	
	0.6	10	10^{-2}	0.010	0.049	7	
Single	0.6	10	10^{-2}	0.006	0.030	7,6	
	0.9	10	10^{-2}	0.003	0.017	7,6,5	
	0.7	10^3	10^{-3}	0.001	0.004	7,6,5,8	
	1.0	10^3	10^{-3}	0.003	0.012	7,6,5,8,4	37850
	0.6	10	10^{-2}	0.015	0.070	-	
Block	0.6	5000	10^{-2}	0.001	0.004	7,8,5,6	
	0.6	10^4	10^{-2}	0.003	0.013	7,8,5,6,4	4142

From the Tables, although the deletion sequences of variables are different, the proposed method deletes the same input variables as those by the conventional method. In addition, both methods delete added redundant variables. By using "CV(C)", the approximation error is slightly increased, but the variable selection time is about 20 times shorter than that of "CV(ALL)" for RBF kernels and about 10 times shorter for Mahalanobis kernels. Thus, for the artificial

dataset, the proposed method gives the same set of selected variables with much shorter calculation time.

Next, we perform the same experiment for LS-SVRs. Tables IV and V show the result for RBF kernels and Mahalanobis kernels, respectively. From the tables, the proposed method also gives the same set of variables as that of the conventional method with much shorter calculation time.

TABLE IV
PERFORMANCE COMPARISON FOR RBF KERNELS (LS-SVRS)

Method	γ	C	Vali.	Test	Deleted	Time[s]
	5.0	10^5	0.012	0.044	-	
	5.0	10^5	0.009	0.043	5	
Single	10	10^5	0.006	0.027	5,6	
	20	10^5	0.002	0.015	5,6,7	
	20	10^5	0.001	0.065	5,6,7,8	
	20	10^5	0.003	0.016	5,6,7,8,4	468
	5.0	10^5	0.012	0.044	-	
Block	5.0	10^5	0.004	0.019	5,6,7,8	
	5.0	10^5	0.005	0.025	5,6,7,8,4	53

TABLE V
PERFORMANCE COMPARISON FOR MAHALANOBIS KERNELS (LS-SVRS)

Method	δ	C	Vali.	Test	Deleted	Time[s]
	0.3	10^3	0.013	0.062	-	
	0.4	10^3	0.010	0.049	7	
Single	0.6	10^3	0.006	0.033	7,8	
	0.9	5000	0.004	0.017	7,8,5	
	1.4	10^5	0.001	0.002	7,8,5,6	
	1.4	10^5	0.003	0.011	7,8,5,6,4	383
	0.3	10^3	0.013	0.062	-	
Block	0.3	10^5	0.002	0.012	7,5,8,6	
	0.3	10^5	0.004	0.020	7,5,8,6,4	70

C. Real Dataset

We evaluate the proposed method for the real dataset. Since the Boston 14 dataset is not divided into training and test datasets, we randomly divide the set into training and test datasets with almost equal sizes. In this way, we make 20 datasets. And for each training data set we determine the optimal parameters as before and delete variables. Then we calculate the average approximation error and its standard deviation for 20 trials and average number of deleted variables and its standard deviation.

Tables VI and VII show the results for SVRs and LS-SVRs, respectively. Column "Vali." lists the average errors and the standard deviations before or after performing variable selection. "Mah." denotes Mahalanobis kernels and "Deleted" lists the average numbers and the standard deviations of the deleted input variables among 20 trials.

For all the cases the approximation errors for the test datasets after variable deletion are slightly worse than those with all the input variables, but six to nine variables are successfully deleted. Using RBF kernels more variables are deleted than using Mahalanobis kernels. But the approximation errors using Mahalanobis kernels are smaller than those of RBF kernels.

And the proposed method reduces the computation time by one tenth for SVRs and LS-SVRs. In addition, using LS-SVRs, the proposed method is 10 to 200 times faster than using SVRs. Thus if variable selection using LS-SVRs gives the similar set of variables as that using SVRs, we can further speed up variable selection for SVRs.

Tables VIII and IX list the number of times that the variables are deleted in variable selection among 20 trials for RBF and Mahalanobis kernels, respectively. The numbers of variables that we list in the tables are the average numbers of deleted variables. For instance from Table VI, the average number of deleted variables for SVR with RBF kernels is 8.9. Thus rounding down 8.9 to 8, we list 8 frequently deleted variables in Table VIII. In the table, the first row of the left table means that the fourth variable is deleted 20 times and the last row means that the seventh variable is deleted 18 times. From the table, for RBF kernels the first seven variables are the same for the SVR and LS-SVRs and only the eighth variables are different. From Table IX, six variables are deleted by SVRs and seven by LS-SVR. The first six variables by LS-SVR are the same with the six variables by SVRs.

TABLE VI
PERFORMANCE COMPARISON FOR BOSTON 14 DATASET (SVRS)

Method	Kernel	Vali.	Test	Deleted	Time[s]
Single	RBF	2.90 ± 0.16	2.71 ± 0.16	-	
		2.80 ± 0.16	2.90 ± 0.22	9.3 ± 0.8	52286 ± 9243
Block	RBF	2.90 ± 0.16	2.71 ± 0.16	-	
		2.67 ± 0.15	2.84 ± 0.19	8.9 ± 1.2	2780 ± 667
Single	Mah.	2.49 ± 0.21	2.50 ± 0.17	-	
		2.20 ± 0.76	2.72 ± 0.22	6.2 ± 2.7	7259 ± 2495
Block	Mah.	2.49 ± 0.21	2.50 ± 0.17	-	
		2.28 ± 0.56	2.68 ± 0.22	6.2 ± 2.4	740 ± 188

TABLE VII
PERFORMANCE COMPARISON FOR BOSTON 14 DATASET (LS-SVRS)

Method	Kernel	Vali.	Test	Deleted	Time[s]
Single	RBF	2.85 ± 0.20	2.71 ± 0.21	-	
		2.76 ± 0.17	2.79 ± 0.22	8.9 ± 0.9	269 ± 34
Block	RBF	2.85 ± 0.20	2.71 ± 0.21	-	
		2.77 ± 0.17	2.78 ± 0.22	8.8 ± 1.0	20 ± 6.0
Single	Mah.	2.54 ± 0.21	2.51 ± 0.15	-	
		2.46 ± 0.18	2.70 ± 0.18	7.7 ± 1.4	387 ± 46
Block	Mah.	2.54 ± 0.21	2.51 ± 0.15	-	
		2.48 ± 0.19	2.69 ± 0.22	7.5 ± 1.4	56 ± 14

D. Substitution Experiment

In this experiment, we evaluate whether the set of variables selected using LS-SVRs is also useful for SVRs. For this purpose, for each training dataset we delete frequently deleted variables, optimize the three parameters by grid search, and calculate the average errors for validation sets and test datasets. Table X shows the result. For instance, the first row of the table shows the result for RBF kernels using the set of selected variables by SVRs, and the second row

TABLE VIII

THE NUMBER OF TIMES THAT INPUT VARIABLES ARE DELETED FOR RBF KERNEL

SVR	Num.	LS.	Num.
#4	20	#1	20
#5	20	#3	20
#2	19	#4	20
#9	19	#5	20
#12	19	#12	20
#1	18	#2	19
#3	18	#9	18
#7	18	#8	16

TABLE IX

THE NUMBER OF TIMES THAT INPUT VARIABLES ARE DELETED FOR MAHALANOBIS KERNEL

SVR	Num.	LS.	Num.
#2	19	#1	20
#4	15	#2	19
#1	14	#4	19
#3	14	#3	17
#9	14	#9	17
#12	11	#12	13
		#10	12

shows the results for RBF kernels using the selected variables by LS-SVRs.

TABLE X

THE RESULT OF THE SUBSTITUTION EXPERIMENT

Kernel	Input	Vali.	Test
RBF	SVRs	2.71 ± 0.16	2.72 ± 0.16
RBF	LS-SVRs	2.74 ± 0.16	2.71 ± 0.18
Mah.	SVRs	2.47 ± 0.20	2.48 ± 0.18
Mah.	LS-SVRs	2.58 ± 0.16	2.57 ± 0.16

From the table, there is not much difference in approximation error even if we replace the set of variables selected by SVRs by that by LS-SVRs. Thus to speed up variable selection for SVRs, we can use a set of variables selected using LS-SVRs.

E. Discussions

From the experiments, it is shown that the proposed method has a comparable generalization ability with the conventional method and reduces the computational cost less than one tenth. Thus, the block deletion is effective for variable selection. For SVRs, in which initially we need to determine three parameters, we can speed up model selection using line search of Mahalanobis kernels instead of using RBF kernels. But for LS-SVRs, the opposite result is obtained. It may be caused as follows: initially we need to determine two parameters by cross-validation, the evaluation points for the parameter of Mahalanobis kernels are larger than those of RBF kernels, and we must recalculate the covariance matrix whenever each input variable is deleted temporarily. However, using Mahalanobis kernels, the approximation accuracy is superior to that using RBF kernels.

In the experiments, frequently selected variables for SVRs and LS-SVRs are quite similar and even if the set of variables selected by LS-SVRs is used for the set for SVRs, approximation performance is almost the same.

VI. CONCLUSIONS

We discussed backward variable selection of SVRs and LS-SVRs by block deletion. Namely, first, we determine the stopping threshold for variable selection by grid search using all the input variables. Then we delete variables that show the higher generalization ability if one is deleted, where we estimate the generalization ability of the regressor by optimizing the margin parameter. If block deletion of variables fails, we backtrack and begin deletion of variables by binary search.

The computer experiments using artificial and real datasets showed that performance of the backward variable selection by block deletion had a similar generalization ability with and smaller computational cost than the conventional backward selection method and the input variables selected by LS-SVRs could be used for SVRs without deteriorating the generalization ability.

REFERENCES

[1] V. Vapnik, *Statistical Learning Theory*, J. Wiley and Sons, 1998.

[2] S. Abe, *Support Vector Machines for Pattern Classification*, Springer-Verlag, London, 2005.

[3] S. Abe, *Pattern Classification: Neuro-fuzzy Methods and Their Comparison*, Springer, London, 2001.

[4] K. R. Müller, A. J. Smola, G. Rätsch, B. Schölkopf, J. Kohlmorgen, and V. Vapnik, "Predicting time series with support vector machines," In W. Gerstner, A. Germond, M. Hasler, and J. D. Nicoud, editors, *Artificial Neural Networks (ICANN '97) Proc. of the Seventh International Conference, Lausanne, Switzerland*, pp. 999-1004, Springer-Verlag, Berlin, 1997.

[5] J. A. K. Suykens, "Least squares support vector machines for classification and nonlinear modelling," *Neural Network World*, 10 (1-2), pp. 29-47, 2000.

[6] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of Machine Learning Research* 3, pp. 1157-1182, 2003.

[7] V. Sindhwani, S. Takshit, D. Deodhare, J. C. Principe, and P. Niyogi, "Feature selection in MLPs and SVMs based on maximum output information," *IEEE Transactions on Neural Networks*, 15 (4), pp. 937-948, 2004.

[8] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artificial Intelligence*, pp. 273-324, 1997.

[9] A. Rakotomamonjy, "Variable selection using SVM-based criteria," *Journal of Machine Learning Research*, Vol. 3, pp. 1357-1370, 2003.

[10] Y. Liu and Y. F. Zheng, "FS_SFS: A novel feature selection method for support vector machines," *Pattern Recognition*, 39 (7), pp. 1333-1345, 2005.

[11] L. J. Herrera, H. Pomares, I. Rojas, M. Verleysen, and A. Guilen, "Effective input variable selection for function approximation," In S. Kollias et al, editors, *Artificial Neural Networks (ICANN 2006)*, Part I, LNCS 4131, pp. 41-50, 2006.

[12] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Machine Learning*, Vol. 46, pp. 389-422, 2002.

[13] C. Gold, A. Holub, and P. Sollich, "Bayesian approach to feature selection and parameter tuning for support vector machine classifiers," *Neural Networks*, 18, pp. 693-701, 2005.

[14] L. F. Bo, L. Wang, and L. C. Jiao, "Sparse Gaussian processes using backward elimination," *Lecture Note in Computer Science (ISNN' 06)*, Vol. 3971, pp. 1083-1088, 2006.

[15] S. Abe, "Modified backward feature selection by cross validation," In *Proc. European Symposium on Artificial Neural Networks (ESANN2005)*, Bruges, Belgium, pp. 163-168, 2005.

[16] S. Abe, *Neural Networks and Fuzzy Systems: Theory and Applications*, Kluwer, Boston, MA, 1997.

[17] D. Harrison and D. L. Rubinfeld, "Hedonic prices and the demand for clean air," *Journal of Environmental and Management*, Vol. 5, pp. 81-102, 1978.

[18] http://www.cs.toronto.edu/ delve/data/datasets.html.

[19] Y. Kamada and S. Abe, "Support vector regression using Mahalanobis kernels," In F. Schwenker and S. Marinai, editors, *Artificial Neural Networks in Pattern Recognition*, pp. 144-152, 2006.