Efficient hyper parameter selection for support vector regression using orthogonal array

Natsuki Sano*

Department of Economics, Management and Information Science,
Onomichi City University,
1600-2 Hisayamada-cho, Onomichi, Hiroshima, Japan
Email: sano@onomichi-u.ac.jp
*Corresponding author

Tomomichi Suzuki

Department of Industrial Administration,
Tokyo University of Science,
2641 Yamazaki, Noda, Chiba, Japan
Email: szk@rs.tus.ac.jp

Abstract: Support vector regression (SVR) is a nonlinear prediction method using kernel functions and has been widely applied to real-world problems. Although the accuracy of an effectively tuned SVR is high, its performance strongly depends on hyper parameters. Therefore, the determination of the parameters is important when applying SVR to real-world problems. Although the optimum parameters are usually determined by an exhaustive grid search, using this method is not realistic when the sample size is considerably large. To decrease the computational time required to determine the optimum parameters, we employ orthogonal array and propose two efficient methods for SVR parameter tuning based on variable selection in Taguchi method. The proposed methods can reduce the computational time to approximately one-twelfth of that taken by a grid search method. We also validated the actual computational time and accuracy of the proposed methods by applying it to five real datasets in UCI repository.

Keywords: machine learning; support vector regression; SVR; hyper parameter; orthogonal array; Taguchi method.

Reference to this paper should be made as follows: Sano, N. and Suzuki, T. (2017) 'Efficient hyper parameter selection for support vector regression using orthogonal array', *Int. J. Computational Intelligence Studies*, Vol. 6, No. 1, pp.40–51.

Biographical notes: Natsuki Sano is an Assistant Professor at Department of Economics, Management and Information Science, Onomichi City University. He holds a PhD in Engineering from the University of Tsukuba. He has worked for the Central Research Institute of Electric Power Industry, Institute of Statistical Mathematics, Osaka University, Kansai University, and Tokyo University of Science. His major is applied statistics especially in marketing and quality control. He is an Editor of *Journal of the Japanese Society for Quality Control*.

Tomomichi Suzuki is a Professor at Department of Industrial Administration, Tokyo University of Science. He holds a PhD in Engineering from the University of Tokyo. He has worked for the University of Tokyo and Tokyo University of Science. His major is statistics and quality management. He is the Chief Editor of *Journal of the Japanese Society for Quality Control*.

This paper is a revised and expanded version of a paper entitled 'Efficient parameter selection for support vector regression using orthogonal array' presented at 2014 IEEE International Conference on Systems, Man, and Cybernetics, San Diego, CA, USA, 5–8 October 2014.

1 Introduction

Support vector regression (SVR) is a nonlinear prediction method using kernel functions and its high prediction accuracy is well known. In addition, it has been widely applied to practical problems in the real world, such as understanding images (Ishiguro et al., 2012) and summarising documents (Ouyang et al., 2011).

Although the accuracy of an effectively tuned SVR model is high, its performance strongly depends on hyper parameters provided by a source outside the model, as shown in Figure 1. In this figure, the real function, training data, and predictive function are shown. It can be seen that the shape of the predictive function varies widely according to the value of the hyper parameters. Therefore, the determination of the parameters is important when applying SVR to real-world problems.

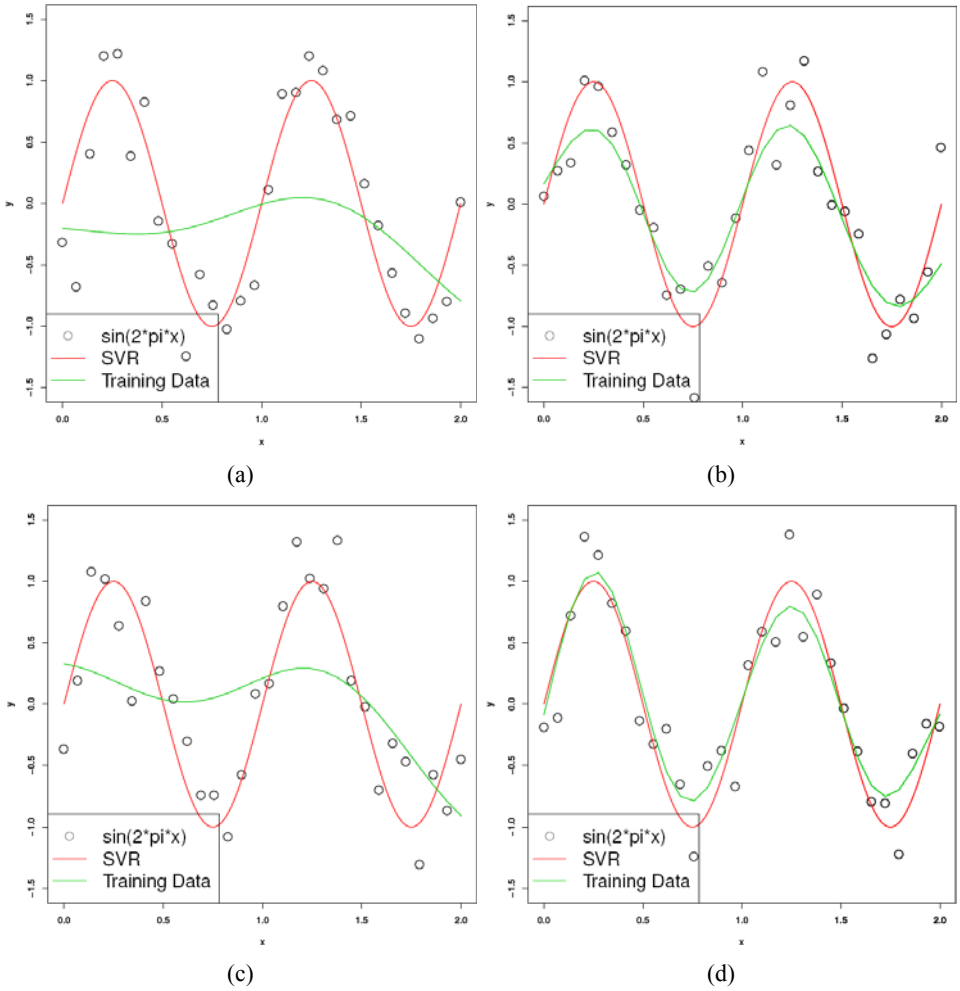
Although the optimum parameters are usually determined by an exhaustive grid search, using this method in big data analysis is not realistic when the sample size is considerably large, because the computational time of SVR scales quadratically at most with the number of samples. In order to decrease the computational time required to determine the optimum parameters, we conducted a particular sampling based on an orthogonal array (OA). In this paper, we propose an efficient method for SVR parameter tuning.

Staelin (2003) and Hsu and Yu (2012) employed OAs for the parameter tuning of a support vector machine (SVM). An SVM has two parameters: C, which controls slack variables, and a kernel parameter. Staelin employed an $L_{16}(8 \times 8 \times 2)$ OA to assign these parameters. However, SVR has an additional parameter: the error function, ε. Therefore, we employ an $L_{144}(12^7)$ OA to assign the three parameters.

The proposed method can reduce the computational time to approximately one-twelfth of that of a grid search method. We validated the accuracy of the proposed methods by applying it to five real datasets in UCI repository and compare those methods with random method. The results of the proposed methods are ranked in the top 10 among all combinations of hyper parameter set and the performance of the proposed method is superior to that of a random method except a dataset. In addition, we verified that the interaction effect between hyper parameters affects the performance of the proposed method.

We describe SVR in Section 2 and OAs in Section 3. In Section 4, we explain the proposed method and verify it using UCI repository datasets. In Section 5, we summarise this paper and present our conclusions and some recommendations for future research.

Figure 1 Effect of hyper parameters on prediction curve of SVR, (a) $C = 1$; $\varepsilon = 0.1$; $\sigma = 1$ (b) $C = 1$; $\varepsilon = 0.1$; $\sigma = 5$ (c) $C = 5$; $\varepsilon = 0.1$; $\sigma = 1$ (d) $C = 5$; $\varepsilon = 0.1$; $\sigma = 5$ (see online version for colours)



2 Support vector regression

We briefly describe SVR (Vapnik, 1998) and the problems related to it in this section.

2.1 Summary of SVR

Given a set of training data $S = \{(x_1, y_1), (x_2, y_2),\ldots,(x_n, y_n)\}$, where the input $x_i \in \Re^p$ constitutes continuous values, the objective of SVR is to find a function $f(x)$ that has at most ε deviation from actually obtained targets y_i for all the training data, and at the same time, is optimally accurate. In other words, while we are not concerned about errors,

provided that their value is less than ε, we do not accept a larger deviation. First, we describe the case of linear functions f, taking the form

$$f(x) = \langle w, x \rangle + b, \tag{1}$$

where $\langle \cdot, \cdot \rangle$ denotes the dot product in $\Re^p$ and w and b take the value of $\Re^p$ and R, respectively.

We can formulate this problem as the following convex optimisation problem

$$\begin{aligned} \text{minimise} \quad & \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} (\xi_i + \xi_i^*) \\ \text{subject to} \quad & y_i - \langle w, x_i \rangle - b \le \varepsilon + \xi_i \\ & \langle w, x_i \rangle - b - y_i \le \varepsilon + \xi_i^* \\ & \xi_i, \xi_i^* \ge 0, \end{aligned} \tag{2}$$

where the constant $C > 0$ determines the trade-off between the accuracy of f and the amount up to which deviations larger than ε are tolerated. It is also noted that slack variables, ξ_i, ξ_i^*, are introduced to obtain a feasible solution.

We can make the function $f(x)$ in equation (1) nonlinear by introducing map Φ: $\Re^p \to \mathcal{F}$ into some feature space $\mathcal{F}$. When substituting $\Phi(x_i)$ for x_i in equation (2), the optimisation problem requires an explicit expression of $\Phi(\bullet)$. However, we can avoid using the explicit expression by representing it as a dual problem and kernel $K(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$,

$$\begin{aligned} \text{maximise} \quad & \frac{1}{2} \sum_{i=1}^{n} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) K(x_i, x_j) - \varepsilon \sum_{i=1}^{n} (\alpha_i + \alpha_i^*) + \sum_{i=1}^{n} y_i (\alpha_i + \alpha_i^*) \\ \text{subject to} \quad & \sum_{i=1}^{n} (\alpha_i - \alpha_i^*) = 0 \\ & 0 < \alpha_i, \alpha_i^* < C \quad i = 1, \ldots, n, \end{aligned} \tag{3}$$

where α_i and α_i^* are Lagrange multipliers. This implicit mapping via a kernel is called a kernel trick (Boser et al., 1992) and any kernel satisfying Mercer's condition (Mercer, 1909) is applicable. A typical kernel is the RBF kernel

$$K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{\sigma^2} \right), \tag{4}$$

where σ^2 is a parameter of the RBF kernel. The predictive function is obtained by solving the dual problem in equation (3)

$$f(x) = \sum_{i=1}^{n} (\alpha_i - \alpha_i^*) K(x_i, x) + b.$$

2.2 Problems of SVR

Although SVR performs well when a nonlinear function is used, two primary problems arise.

First, the computational time of solving the quadratic programming optimisation problem in equation (3) usually scales quadratically with the number of samples n. Even the sequential minimisation optimisation (SMO) algorithm (Platt, 1999) is specialised for an SVM that scales somewhere between linear and quadratic. In the recently evolved big data environment, very large datasets need to be handled and thus the problem of execution time is significant.

Second, the performance of SVR depends on the hyper parameters C, ε and σ^2. Figure 1 shows the predictive function changed by the hyper parameters. Assuming a real function $\sin(2\pi x)$, we generate training data where x_i represents equally-spaced points between 0 and 2 and $y_i = \sin(2\pi x_i) + e_i$, $e_i \sim N(0, 0.3^2)$, $i = 1,\ldots,30$. In Figure 1, the real function, training data, and predictive function are shown. It can be seen that the shape of the predictive function varies greatly according to the combination of the hyper parameters, and the predictive function with $C = 5$, $\varepsilon = 0.1$ and $\sigma^2 = 5$ captures the nature of the data.

In order to tune the parameters to their optimal value, a grid search is often conducted (Hsu et al., 2003). However, parameter tuning by using a grid search is an exhaustive method that executes an algorithm for training data repeatedly, and therefore, it is very time consuming when applied to very large data. We propose a parameter tuning method that uses an OA to solve this problem.

3 Orthogonal array

Design of experiment (DOE) uses an OA to study the effect of each factor on the response variable efficiently (Montgomery, 2001). An OA is an array that has following characteristics.

- each level appears the same number of times in any column
- in any two selected columns, each combination (i, j) $i, j = 1, \ldots, l$ appears the same number of times, where l denotes the number of the level.

These characteristics are called orthogonality and the Mahalanobis-Taguchi (MT) system uses OAs to select meaningful variables for prediction from many variables (Taguchi and Jugulum, 2002) in the quality engineering field.

In this study, we employed an OA to select the hyper parameters C, ε and σ^2 of SVR efficiently. Although these hyper parameters are continuous variables originally, they are discretised to assign them to each level. We employ the $L_{144}(12^7)$ OA in the Kuhfeld collection of OAs (Kuhfeld Collection, http://support.sas.com) shown in Table 1, because the OA needs to have as many levels as possible to increase the granularity of the search.

Table 1 $L_{144}(12^7)$ OA

Run	*A*	*B*	*C*	*D*	*E*	*F*	*G*
1	1	1	1	1	1	1	1
2	1	2	7	4	9	5	2
3	1	3	9	7	2	12	3
4	1	4	3	2	12	11	4
5	1	5	8	10	6	3	5
6	1	6	2	12	10	8	6
7	1	7	10	3	4	9	7
8	1	8	12	9	11	7	8
9	1	9	5	6	3	10	9
10	1	10	11	5	8	2	10
11	1	11	6	8	7	4	11
12	1	12	4	11	5	6	12
13	2	1	3	7	11	9	6
14	2	2	2	2	2	2	1
15	2	3	8	5	10	6	2
:	:	:	:	:	:	:	:
142	12	10	1	3	11	8	5
143	12	11	7	5	3	1	6
144	12	12	12	12	12	12	1

In Table 1, the number in the leftmost column is that of the run, that is, the number of the experiment, and the numbers in the remaining columns are the level for each factor. $L_{144}(12^7)$ has 144 runs and 12 levels and can be assigned up to seven factors.

Table 2 Parameter level table

Level	*C*	ε	σ
1	0.0002	0.00003	0.1250
2	0.0011	0.00008	0.1824
3	0.0050	0.00023	0.2663
4	0.0228	0.00063	0.3886
5	0.1035	0.00172	0.5672
6	0.4695	0.00472	0.8278
7	2.130	0.01293	1.208
8	9.665	0.03545	1.763
9	43.85	0.09715	2.573
10	199.0	0.2663	3.756
11	902.7	0.7297	5.481
12	4096	2.000	8.000

We ignore the interaction effect and assign the three factors C, ε, and σ^2 to any columns A to G in the $L_{144}(12^7)$ OA. The parameter level table (Table 2) gives the specific value at each level (C_j, ε_k, σ_m^2), $j, k, m = 1,\ldots,12$. The levels of C constitute a sequence raised by 2 to the power of the arithmetic sequence between –12 and 12. Similarly, the levels of ε and σ are those between –12 and 1 and between –3 and 3, respectively.

We can determine the value of C, ε and σ^2 at the i^{th} run, ($C_{j(i)}$, $\varepsilon_{k(i)}$, $\sigma_{m(i)}^2$), $i = 1,\ldots,144$, in conjunction with the parameter level table. We execute SVR with parameter ($C_{j(i)}$, $\varepsilon_{k(i)}$, $\sigma_{m(i)}^2$) and evaluate the performance, p_i. We decompose the performance p_i by three-way layouts using the OA as follows.

$$p_i = \mu + \alpha_j + \beta_k + \gamma_m + e_i, \tag{5}$$

where μ denotes the mean effect, α_j, β_k, and γ_m denote the effects of C_j, ε_k, and σ_m^2, respectively, and e_i denotes the normally distributed random variable with zero mean. We denote the mean of p_i in the runs by C_j, ε_k, and σ_m^2 by $\overline{p}_{\alpha_j}$, $\overline{p}_{\beta_k}$, and $\overline{p}_{\gamma_m}$, respectively. By the orthogonality of the OA, they are written as

$$\begin{aligned}
\overline{p}_{\alpha_j} &= \mu + \alpha_j + \overline{\beta} + \overline{\gamma} + \overline{e}_j, \\
\overline{p}_{\beta_k} &= \mu + \overline{\alpha} + \beta_k + \overline{\gamma} + \overline{e}_k, \\
\overline{p}_{\gamma_m} &= \mu + \overline{\alpha} + \overline{\beta} + \gamma_m + \overline{e}_m, \\
& j, k, m = 1,\ldots,12;
\end{aligned} \tag{6}$$

where $\overline{\alpha} = \dfrac{\sum_{j=1}^{12} \alpha_j}{12}$, $\overline{\beta} = \dfrac{\sum_{k=1}^{12} \beta_k}{12}$, $\overline{\gamma} = \dfrac{\sum_{m=1}^{12} \gamma_m}{12}$, and $\overline{e}_j$, $\overline{e}_k$, and $\overline{e}_m$ denote the mean of ε_i in runs with C_j, ε_k, and σ_m^2, respectively. It should be noted that μ, $\overline{\beta}$, and $\overline{\gamma}$ is common among different levels of j in the first equation of equation (6). Therefore, the level j minimising $\overline{p}_{a_j}$ also minimises α_j, since $\overline{e}_j$ can be ignored by normality. This similarly holds true for β_k and γ_m. We denote the level that minimises $\overline{p}_{\alpha_j}$, $\overline{p}_{\beta_k}$, and $\overline{p}_{\gamma_m}$ by j^*, k^*, and m^*, respectively. We can determine (C_{j^*}, ε_{k^*} and $\sigma_{m^*}^2$) as the optimum combination of parameters and expect the performance

$$p^* = \mu + \alpha_{j^*} + \beta_{k^*} + \gamma_{m^*} \tag{7}$$

It should be noted that the combination of (C_{j^*}, ε_{k^*} and $\sigma_{m^*}^2$) can go beyond the parameter combination obtained by the $L_{144}(12^7)$ OA.

4 Proposed hyper parameter selection methods and verification

In this section, we propose two hyper parameter selection methods using an OA. In the first proposed method, we repeatedly execute SVR with parameters ($C_{j(i)}$, $\varepsilon_{k(i)}$, $\sigma_{m(i)}^2$) using the $L_{144}(12^7)$ OA. For measuring the performance of the proposed method, we employ ten-fold cross-validation and residual standard deviation. In ten-fold

cross-validation, we divide the original data into ten subsets: nine subsets are used as training data and the remaining subset is used as test data. We compute the residual standard deviation for the test data. This process is repeated ten times, where at each repetition a different subset is selected as test data. We evaluate p_i, the performance of SVR with parameters ($C_{j(i)}$, $\varepsilon_{k(i)}$, $\sigma^2_{m(i)}$), by using the residual standard deviation. The optimum hyper parameter set (C_{j*}, ε_{k*} and σ^2_{m*}) is determined as that which minimises p_i among 144 combinations in the OA. We call this procedure proposed method 1; it is summarised in Table 3.

Table 3 Procedure of proposed method 1

1	Repeat evaluation of SVR (i = 1,…,144)
	(a) List i^{th} parameter set from the i^{th} run from $L_{144}(12^7)$ and the parameter level table.
	(b) Execute SVR with the i^{th} parameter set by ten-fold cross validation.
	(c) Evaluate the performance p_i by residual standard deviation.
2	Determine the parameter set (C_{j*}, ε_{k*} and σ^2_{m*}) with which SVR minimises p_i.

Similarly to the first proposed method, the second proposed method computes the residual standard deviation for 144 combinations. Although the first proposed method selects the optimum hyper parameter set among 144 tested combinations, the second proposed method calculates the optimum hyper parameter levels. The optimum hyper parameter set (C_{j*}, ε_{k*} and σ^2_{m*}) is determined as that which minimises $\overline{p}_{\alpha_j}$, $\overline{p}_{\beta_k}$, and $\overline{p}_{\gamma_m}$ described in Section 3. We call this procedure proposed method 2, it is summarised in Table 4.

Table 4 Procedure of proposed method 2

1	Repeat evaluation of SVR (i = 1,…,144)
	(a) List i^{th} parameter set from the i^{th} run from $L_{144}(12^7)$ and the parameter level table.
	(b) Execute SVR with the i^{th} parameter set by ten-fold cross validation.
	(c) Evaluate the performance p_i by residual standard deviation.
2	Evaluate the mean of the performance at each parameter level, $\overline{p}_{\alpha_j}$, $\overline{p}_{\beta_k}$, $\overline{p}_{\gamma_m}$ j, k, m = 1, …, 12.
3	Determine the parameter set (C_{j*}, ε_{k*} and σ^2_{m*}) with which SVR minimises $\overline{p}_{\alpha_j}$, $\overline{p}_{\beta_k}$, and $\overline{p}_{\gamma_m}$.

Although the number of all the combinations of (C_j, ε_k, σ^2_m) is 12 × 12 × 12 = 1,728, the proposed method can determine the parameter set efficiently using only 144 combinations. This facilitates computational time saving, in particular, as the number of samples increases.

5 Numerical experiments using UCI repository data

We verified the accuracy of the proposed method as compared with a grid search method by using the UCI repository data published in UC Irvine Machine Learning Repository (http://archive.ics.uci.edu/ml/). Five datasets, *wine quality*, *Parkinson's telemonitoring*, *forest fires*, *airfoil self-noise*, and *concrete compressive strength* were employed for our numerical experiments. The number of samples and variables of these datasets are shown in Table 5.

Table 5 Experimental data (UCI repository)

Dataset	*Number of samples*	*Number of variables*
Wine quality	4,898	12
Parkinson's telemonitoring	5,875	22
Forest fires	517	11
Airfoil self-noise	1,503	6
Concrete compressive strength	1,030	9

Table 6 Comparison of computation time of proposed method 2 and grid search

Dataset	*Grid search*	*Proposed method 2*
Wine quality	62,022.45 s (17.23 h)	5,552.68 s (1.54 h)
Parkinson's telemonitoring	170,132.62 s (47.26 h)	14,118.89 s (3.92 h)
Forest fires	623.22 s (0.17 h)	52.00 s (0.01 h)
Airfoil self-noise	8,947.26 s (2.49 h)	727.58 s (0.20 h)
Concrete compressive strength	2,207.84 s (0.61 h)	169.69 s (0.05 h)

Table 6 shows the computation time of the proposed method as compared with that of grid search, which is an exhaustive method for sampling all the combinations. The numerical experiments were executed by a machine that had the following specifications: OS: Windows 7 Professional 64 bit, CPU: Intel (R) Core (TM) i7-2600S @ 2.80 Ghz, Memory: 8.00 GB.

We assigned C, ε, and σ^2 to (7, 5, 2) columns in the $L_{144}(12^7)$ OA, respectively, and used the kernlab package for kernel-based machine learning methods in R to execute SVR.

Table 7 Comparison of rank of the proposed methods and random method

Dataset	*Proposed method 1*	*Proposed method 2*	*Mean (random method)*	*SD (random method)*
Wine quality	7	1	13.84	12.78
Parkinson's telemonitoring	14	3	11.81	11.04
Forest fires	2	1	9.52	9.78
Airfoil self-noise	14	181	9.81	11.27
Concrete compressive strength	4	4	10.22	9.18

Table 8 Validation of interaction effect

Dataset	*β* / *p*	*Intercept*	*C*	*ε*	σ^2	$C\varepsilon$	$C\sigma^2$	$\varepsilon\sigma^2$	$C\varepsilon\sigma^2$
Wine quality	*β*	0.799	0.000	0.042	0.001	0.000	0.000	0.000	0.000
	p	0.000***	0.000***	0.000***	0.227	0.001**	0.702	0.897	0.820
Parkinsons telemonitoring	*β*	8.221	–0.001	0.115	0.272	0.000	0.000	–0.014	0.000
	p	0.000***	0.000***	0.417	0.000***	0.486	0.000***	0.748	0.787
Forest fires	*β*	63.686	0.000	–0.003	–0.004	0.000	0.000	0.000	0.000
	p	0.000***	0.000***	0.408	0.000***	0.948	0.000***	0.864	0.832
Airfoil self-noise	*β*	7.117	0.001	–0.100	–0.056	0.000	0.000	0.011	0.000
	p	0.000***	0.000***	0.007**	0.000***	0.000***	0.000***	0.335	0.000***
Concrete compressive strength	*β*	15.938	0.000	0.006	0.058	0.000	0.000	–0.001	0.000
	p	0.000***	0.000***	0.898	0.000***	0.852	0.000***	0.955	0.970

Notes: ***0.1% significant, **1% significant.

It can be seen that the proposed method reduced the computational time required to search for hyper parameters by one-twelfth of that of the grid search method for all data. However, in fact we can observe that a larger computational time is needed in the grid search method, as the number of samples increases. Note that the proposed method reduced the time required by 43.34 h for the Parkinson's telemonitoring data, the sample size of which is 5,875. Therefore as the number of samples increases, a larger time-saving effect can be expected.

Next, we compared the proposed method with a random method. The random method samples 144 candidate parameter sets from all the combinations randomly and determines which is optimal. The results of the comparison of the ranks of the proposed method and the random method are shown in Table 7. In this table the column presenting the random method shows the mean and standard deviation of rank among all the combinations when the random method was repeated 100 times. It should be noted that proposed methods 1 and 2 determine the hyper parameters uniquely and therefore there are no standard deviation columns for them in the table. It can be seen that proposed method 2 improved on proposed method 1, and both the proposed methods outperformed the random method, except in the case of the airfoil self-noise dataset. We consider that the interaction effect between the hyper parameters caused the inferior performance of the proposed methods for the airfoil self-noise dataset, since the proposed method ignores the interaction effect.

We conducted regression analyses that included the interaction effect term for the airfoil self-noise dataset to verify the interaction effect between hyper parameters. We considered the following regression model that includes the first-order interaction effect between C, ε, σ^2.

$$p_i = \beta_0 + \beta_1 C_i + \beta_2 \varepsilon_i + \beta_3 \sigma_i^2 + \beta_4 C_i \varepsilon_i + \beta_5 C_i \sigma_i^2 + \beta_6 \varepsilon \sigma_i^2 + \beta_7 C_i \varepsilon_i \sigma_i^2 + e_i,$$
$$i = 1, \ldots, n,$$

where p_i denotes the performance measure of the fitted model, i.e., the standard deviation of the residual, and e_i denotes the normally distributed error term. The performance of the model p_i was evaluated n = 1,728 times for all the combinations of hyper parameters C_i, ε_i, σ_i^2. The validation results of the interaction effects are shown in Table 8.

We can observe that C is 0.1% significant for all datasets and σ^2 is also 0.1% significant for almost all datasets, except for the wine quality data. Among the interaction effect terms, an interaction effect between C and σ^2 is significant for almost datasets, except for the wine quality dataset. In terms of datasets, we can observe that interaction effects among all hyper parameters are at least 1% significant in the airfoil self-noise dataset. Therefore, this interaction effect is considered the cause of the inferior performance of the proposed method for the airfoil self-noise dataset.

6 Conclusions

We proposed two efficient parameter tuning methods for SVR using an $L_{144}(12^7)$ OA. The proposed methods reduced the computational time to approximately one-twelfth of that of a grid search method.

We validated the accuracy of the proposed methods by applying it to five real datasets in UCI repository and compare those methods with random method. The results of the

proposed methods are ranked in the top ten among all combinations of hyper parameter set and the performance of the proposed method is superior to that of a random method except *airfoil self-noise* dataset. In addition, we verified that the interaction effect between hyper parameters affects the performance of the proposed method.

A larger effect in terms of computational time saving can be expected as the number of samples increases. Therefore, the proposed method has the potential to be a powerful tool for model building in big data analysis. Although we employed the $L_{144}(12^7)$ OA in this study, the characteristics of the OA for the proposed method and the principle of the assignment of hyper parameters in an OA were not clarified. These are subjects for future study.

Acknowledgements

This work was supported by JSPS KAKENHI Grant Number 25750131.

References

Boser, B.E., Guyon, I.M. and Vapnik, V.N. (1992) 'A training algorithm for optimal margin classifiers', *Proc of the Annual Conference on Computational Learning Theory*, pp.144–152.

Hsu, C.W., Chang, C.C. and Lin, C.J. (2003) *A Practical Guide to Support Vector Classification*, Technical Report, Department of Computer Science, National Taiwan University.

Hsu, W.C. and Yu, T.Y. (2012) 'Support vector machines parameter selection based on combined Taguchi method and Staelin method for e-mail spam filtering', *International Journal of Engineering and Technology Innovation*, Vol. 2, No. 2, pp.113–125.

Ishiguro, K., Kimura, A. and Takeuchi, K. (2012) 'Towards automatic image understanding and mining via social curation', *Proc of IEEE 12th International Conference on Data Mining*, pp.906–911.

Kuhfeld Collection [online] http://support.sas.com/techsup/technote/ts723.html (accessed 11 April 2016).

Mercer, J. (1909) 'Functions of positive and negative type and their connection with the theory of integral equations', *Philosophical Transaction of the Royal Society*, Series A, Vol. 209, pp.415–446.

Montgomery, D.C. (2001) *Design and Analysis of Experiments*, 5th ed., Wiley, NY.

Ouyang, Y., Li, W., Li, S. and Lu, Q. (2011) 'Applying regression models to query-focused multi-document summarization', *Information Processing and Management*, Vol. 47, No. 2, pp.227–237.

Platt, J. (1999) 'Fast training of support vector machines using sequential minimal optimization', in Schölkopf, B., Burges, C.J.C. and Smola, A.J. (Ed.): *Advances in Kernel Methods – Support Vector Learning*, pp.185–208, MIT Press, NY.

Staelin, C. (2003) *Parameter Selection for Support Vector Machines*, Technical Report, Hewlett-Packard Company.

Taguchi, G. and Jugulum, R. (2002) *The Mahalanobis-Taguchi Strategy: A Pattern Technology System*, Wiley, NY.

UC Irvine Machine Learning Repository [online] http://archive.ics.uci.edu/ml/ (accessed 11 April 2016).

Vapnik, V.N. (1998) *Statistical Learning Theory*, Wiley, NY.