Simple estimate of the width in Gaussian kernel with adaptive scaling technique

Simple Estimate of the Width in Gaussian Kernel with Adaptive Scaling Technique

Satoshi Kitayama[1], Koetsu Yamazaki[2]

[1] *College of Science and Engineering, Kanazawa University, Kakuma-machi, Kanazawa, 920-1192, Japan*

+81-76-234-4758

kitagon@t.kanazawa-u.ac.jp (Corresponding Author)

[2] *College of Science and Engineering, Kanazawa University, Kakuma-machi, Kanazawa, 920-1192, Japan*

Abstract

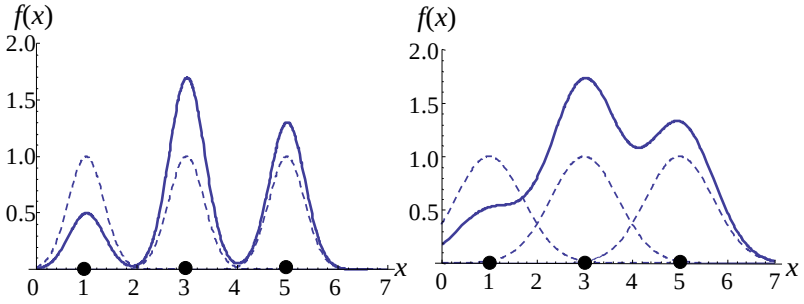
This paper presents a simple method to estimate the width of Gaussian kernel based on an adaptive scaling technique. The Gaussian kernel is widely employed in Radial Basis Function (RBF) network, Support Vector Machine (SVM), Least Squares Support Vector Machine (LS-SVM), Kriging models, and so on. It is widely known that the width of the Gaussian kernel in these machine learning techniques plays an important role. Determination of the optimal width is a time-consuming task. Therefore, it is preferable to determine the width with a simple manner. In this paper, we first examine a simple estimate of the width proposed by Nakayama et al.. Through the examination, four sufficient conditions for the simple estimate of the width are described. Then, a new simple estimate for the width is proposed. In order to obtain the proposed estimate of the width, all dimensions are equally scaled. A simple technique called the adaptive scaling technique is also developed. It is expected that the proposed simple method to estimate the width is applicable to wide range of machine learning techniques employing the Gaussian kernel. Through examples, the validity of the proposed simple method to estimate the width is examined.

Keywords: Gaussian Kernel, Width, LS-SVM, RBF network

1. Introduction

Machine learning techniques such as radial basis function (RBF) network, support vector machine (SVM), support vector regression (SVR), and least squares support vector machine (LS-SVM) are widely employed in various areas. These techniques commonly employ the Gaussian kernel. The equivalence between SVM and ordinary Kriging has been reported under the assumption that the covariance function is used as the kernel function [1]. The equivalence between SVM and the regularization neural network has been also reported [2]. This equivalence can be extended to RBF network, considering the suggestions of Poggio and Girosi [3]. Thus, it is easily assumed that the equivalence between SVM and RBF network can be established. LS-SVM overcomes the complex procedure for finding the support vectors of SVM and SVR, and LS-SVM is now widely studied [4-9]. In SVM and SVR, the support vectors are determined by solving the quadratic programming (QP) problem, while LS-SVM can find the support vectors by solving a simpler linear system.

In the Gaussian kernel, the width plays an important role. As an example, let us consider the regression by RBF network. The effect of the width is shown in Fig.1 (a) and (b). In Fig.1, the black dots represent the training data, the dashed line represents the Gaussian kernel, and the solid line denotes the regression. The following weights are assigned to the training data: $w_1 = 0.5$ at $x = 1$, $w_2 = 1.7$ at $x = 3$, and $w_3 = 1.3$ at $x = 5$. The difference between the graphs in Fig.1 is the width of the Gaussian kernel. The widths in Fig.1(a) and (b) are set to 0.5 and 1.0, respectively. Small value of the width leads to the non-smooth regression, while the smooth regression can be obtained with large value of the width. It is clear from Fig.1 that the determination of the width plays an important role.



(a) Small value of the width
width=0.5

(b) Large value of the width
width=1.0

Fig.1 Effect of width in the Gaussian kernel

It is possible to determine the optimal width with optimization techniques. Therefore, optimization with respect to the width will be performed. However, the task to optimize the width is much time-consuming. When we determine the width to each Gaussian kernel, the number of decision variables of the optimization problem is equal to the number of training data. If we have one hundred training data, the optimization problem has one hundred decision variables. Thus, we have to solve the optimization problem with one hundred decision variables. In addition, the function with respect to the width becomes a multi-modal function due to the Gaussian kernel. This leads to the discussion on global optimization techniques. Therefore, which are the best global optimization techniques? In order to avoid the discussions on global optimization techniques, it is preferable to determine the width in the Gaussian kernel with a simple manner. Few papers focus on the width in the Gaussian kernel, and some simple estimates of the width have been proposed [10-13]. These are heuristic approaches, and it may be difficult to provide rigorous proofs from the mathematical point of view. In numerical computation, we have to determine the width in some ways, so that it is an important topic for determining the width with a simple manner. In [10], "*p*-nearest neighbours heuristic" is employed to determine the width. In this approach, it is possible to determine the width to each Gaussian kernel. The problem of this approach is to determine the parameter *p* in advance. *p* is set to 2, but no clear reasons could be found in [10]. In [11], the nearest neighbor approach is also employed. In this approach, the minimum distance among the training data is considered. The width of each Gaussian kernel can be determined by the product between the minimum distance and a constant parameter. It is also possible to determine the width to each Gaussian kernel, but a constant parameter may depend on the problems. In [12], the following simple estimate of the width is proposed:

$$r = \frac{d_{\max}}{\sqrt{2M}} \tag{1}$$

where $d_{\max}$ denotes the maximum distance among the training data, and M is the number of hidden neurons. A similar estimate of the width is also proposed by Nakayama et al. [13], and is given as follows:

$$r = \frac{d_{\max}}{\sqrt[n]{nM}} \tag{2}$$

In Eq(2), *n* denotes the dimensions. Eq.(2) can be regarded as the generalization and the extension to *n* dimensions, in comparison with Eq.(1). Eq.(2) consists of three elements: (1) the dimensions, (2) the number of hidden neurons, and (3) the maximum distance among the training data. In [13], for simplicity, the number of hidden neurons is assumed to be the number of training data. This reason will be discussed at the beginning of section 3. In addition, this assumption is employed to apply Eq.(2) to SVM and SVR [14]. Eqs.(1) and (2) are equally applied to all Gaussian kernels, and the width has a constant value. If Eqs.(1) and (2) are employed to determine the width in the Gaussian kernels, non-uniform distribution of the training data cannot be taken into account. It has been reported that Eq.(2) can be applicable to RBF network, SVM, and SVR [14]. If the training data are uniformly distributed, it is expected that Eq.(2) will work well. However, most real problems show non-uniform distribution of the training data. In such cases, it may be difficult to obtain a good classifier and regression with Eq.(2). It is important to develop a simple estimate of the width considering the non-uniform distribution of the training data.

In this paper, a new simple estimate of the width in the Gaussian kernel is proposed. As described above, Eq.(2) can be regarded as the generalization and extension of Eq.(1), and then we mainly focus on Eq.(2). First, Eq.(2) is examined, and then four sufficient conditions for the simple estimate of the width will be described. According to these sufficient conditions, a new simple estimate of the width in the Gaussian kernel is proposed. In order to employ this new simple estimate of the width, all dimensions are equally scaled. A simple scaling technique, called the adaptive scaling technique, is also proposed. In this scaling technique, the scaling coefficient is adaptively adjusted. The proposed estimate of the width can deal with the non-uniform distribution of the training data. In numerical examples, the proposed estimate of the width with the adaptive scaling technique is applied to LS-SVM and RBF network. One application is the classifier by LS-SVM and RBF network, and the other is the sequential approximate optimization (SAO) by RBF network. In the classifier, three character recognition problems are handled. Through this numerical examples, it can be found that the proposed estimate of the width is applicable to LS-SVM and RBF network. It is apparent from numerical examples that the same constant value of the width sometimes leads to wrong results. In other words, the

numerical results show that individual assignment of the width to the Gaussian kernel is important and valid. The proposed estimate of the width clearly belongs to the heuristic approaches, so that it may be difficult to give rigorous proofs. Through numerical examples, the validity is examined.

The remainder of this paper is organized as follows: In section 2, brief review of some machine learning techniques is described. In section 3, the width in [12] is also examined, and four sufficient conditions for a simple estimate of the width are described. Then, a new equation of the width is proposed, and the adaptive scaling technique is also described. In section 4, a new simple estimate of the width is examined through examples.

2. RBF network and LS-SVM

2.1 Radial Basis Function Network

RBF network is a three-layer feed-forward network. The training data is expressed by $\{\boldsymbol{x}_j, y_j\}(\ j = 1, 2, \cdots, m\ )$, and M represents the number of hidden neurons. According to [13-15], the output of the network $f_a(\boldsymbol{x})$ is given by

$$f_a(\boldsymbol{x}) = \sum_{j=1}^{M} w_j h_j(\boldsymbol{x}) \tag{3}$$

where $h_j(\boldsymbol{x})$ is the j-th basis function, and w_j denotes the weight of the j-th basis function. The regression is given by Eq.(3). The following Gaussian kernel is often employed as the basis function.

$$h_j(\boldsymbol{x}) = \exp\left(-\frac{(\boldsymbol{x}-\boldsymbol{c}_j)^T(\boldsymbol{x}-\boldsymbol{c}_j)}{r_j^2}\right) \tag{4}$$

In Eq(4), r_j is the width of the j-th basis function, and $\boldsymbol{c}_j$ is the center of the j-th basis function. The learning of the RBF network is usually accomplished by solving the following optimization problem with respect to weights $\boldsymbol{w}$:

$$E = \sum_{j=1}^{m} (y_j - f_a(\boldsymbol{x}_j))^2 + \sum_{j=1}^{M} \lambda_j w_j^2 \rightarrow \min \tag{5}$$

where the second term is introduced for the purpose of regularization. It is recommended that λ_j in Eq.(5) have a sufficiently small value (e.g. $\lambda_j = 1.0 \times 10^{-3}$) [13]. Thus, the learning of the RBF network is equivalent to finding the weight vector $\boldsymbol{w}$ [15]. The necessary condition of Eq.(5) leads to the following equation:

$$\boldsymbol{w} = (\boldsymbol{H}^T \boldsymbol{H} + \boldsymbol{\Lambda})^{-1} \boldsymbol{H}^T \boldsymbol{y} \tag{6}$$

where $\boldsymbol{H}$, $\boldsymbol{\Lambda}$, and $\boldsymbol{y}$ are given as follows:

$$\boldsymbol{H} = \begin{bmatrix} h_1(\boldsymbol{x}_1) & h_2(\boldsymbol{x}_1) & \cdots & h_M(\boldsymbol{x}_1) \\ h_1(\boldsymbol{x}_2) & h_2(\boldsymbol{x}_2) & \cdots & h_M(\boldsymbol{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ h_1(\boldsymbol{x}_m) & h_2(\boldsymbol{x}_m) & \cdots & h_M(\boldsymbol{x}_m) \end{bmatrix} \tag{7}$$

$$\boldsymbol{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \lambda_M \end{bmatrix} \tag{8}$$

$$\boldsymbol{y} = (y_1, y_2, \cdots, y_m)^T \tag{9}$$

It is clear from Eq.(6) that the learning of the RBF network is equivalent to the matrix inversion $(\boldsymbol{H}^T \boldsymbol{H} + \boldsymbol{\Lambda})^{-1}$. In comparison with direct optimization of Eq.(5), the use of Eq.(6) can save the computational time, and results in simple calculation. Using the RBF network, it is easy to calculate the weight vector $\boldsymbol{w}$, because the additional learning is reduced to the incremental calculation of the matrix inversion. The detailed procedure is found in [15].

2.2 Least Squares Support Vector Machine

Least squares support vector machine (LS-SVM) considers equality constraints for the classification problem with a formulation in least squares sense. Thus, the solution can be obtained by solving a set of linear equations instead of solving QP problem. The extension of classical SVM to SVR is more complex because the epsilon insensitive loss function is introduced, while it is very easy to extend LS-SVM classifier to the regression version.

In LS-SVM classifier, the following simple linear system equation can be solved:

$$\begin{bmatrix} 0 & \boldsymbol{y}^T \\ \boldsymbol{y} & \boldsymbol{\Omega} + C^{-1}\boldsymbol{I} \end{bmatrix} \begin{Bmatrix} b \\ \boldsymbol{\alpha} \end{Bmatrix} = \begin{Bmatrix} 0 \\ \boldsymbol{1} \end{Bmatrix} \tag{10}$$

where $\boldsymbol{y} = (y_1, y_2, \cdots, y_m)^T$. The element of $\boldsymbol{y}$ is given by $y_j \in \{-1, +1\}$. $\boldsymbol{\Omega}$ is the m x m matrix with elements $\Omega_{i,j} = y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)$. K represents the kernel function, and the following Gaussian kernel is widely employed.

$$K(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\frac{(\boldsymbol{x}-\boldsymbol{x}')^T(\boldsymbol{x}-\boldsymbol{x}')}{r'^2}) \tag{11}$$

where r' represents the width in the Gaussian kernel. $\boldsymbol{I}$ represents the unit matrix. $\boldsymbol{1} = (1,1,\cdots,1)^T$ of the left hand side in Eq.(10) is a *m* x 1 column vector. The following separating hyperplane $f_a(\boldsymbol{x})$ can be obtained by solving Eq.(10):

$$f_a(\boldsymbol{x}) = \sum_{j=1}^{m} \alpha_j y_j K(\boldsymbol{x}, \boldsymbol{x}_j) + b \tag{12}$$

The training data is separated by $f_a(\boldsymbol{x}) = 0$.

Next, let us consider LS-SVM regression version. In LS-SVM regression, the following linear system can be solved in order to obtain the regression:

$$\begin{bmatrix} 0 & \boldsymbol{1}^T \\ \boldsymbol{1} & \boldsymbol{\Omega} + C^{-1}\boldsymbol{I} \end{bmatrix} \begin{Bmatrix} b \\ \boldsymbol{\alpha} \end{Bmatrix} = \begin{Bmatrix} 0 \\ \boldsymbol{y} \end{Bmatrix} \tag{13}$$

where $\boldsymbol{y} = (y_1, y_2, \cdots, y_m)^T$ is the column vector of the real values, and the element of $\boldsymbol{\Omega}$ is given by $\Omega_{i,j} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$. *K* represents the kernel function given by Eq.(11). LS-SVM regression $f_a(\boldsymbol{x})$ is expressed as follow:

$$f_a(\boldsymbol{x}) = \sum_{j=1}^{m} \alpha_j K(\boldsymbol{x}, \boldsymbol{x}_j) + b \tag{14}$$

By comparing Eqs.(10) and (13) with Eq.(6), it follows that parameter *C* in LS-SVM corresponds to λ_j in RBF network when the number of training data is equal to the number of hidden neurons. Therefore, it is recommended that parameter *C* have a sufficiently large value (e.g. *C*=1000).

3. Width of Gaussian Kernel and Adaptive Scaling Technique

In this section, a new simple estimate of the width in the Gaussian kernel is proposed. In order to develop the new simple estimate of the width, all dimensions are equally scaled. The adaptive scaling technique is also described in this section. In RBF network, the determination of the appropriate number of hidden neurons is also one of the important topics. This leads to the discussion on the center of the basis function. There are some methods to determine the center of Gaussian kernel such as SOM (Self-Organizing Map), *k*-means clustering and so on. However, the objective of this paper is to propose a simple estimate of the width with a simple manner. Then, for simplicity, we assume that the number of

hidden neurons is equal to the number of training data ($M = m$). In RBF network, this implies that we set the basis function upon the training data ($\boldsymbol{c}_j$ in Eq.(4) is replaced as j-th training data $\boldsymbol{x}_j$). This assumption can be extended to LS-SVM for the availability of width.

3.1 Width of Gaussian Kernel

Let us consider the K-level full factorial design [16], in which the regular interval is given by Δd . In this case, $d_{\max}$ is given by

$$d_{\max} = \sqrt{n}(K-1)\Delta d \tag{15}$$

Fig.2 shows an illustrative example in two dimensions. In this figure, the black dots denote the training data.

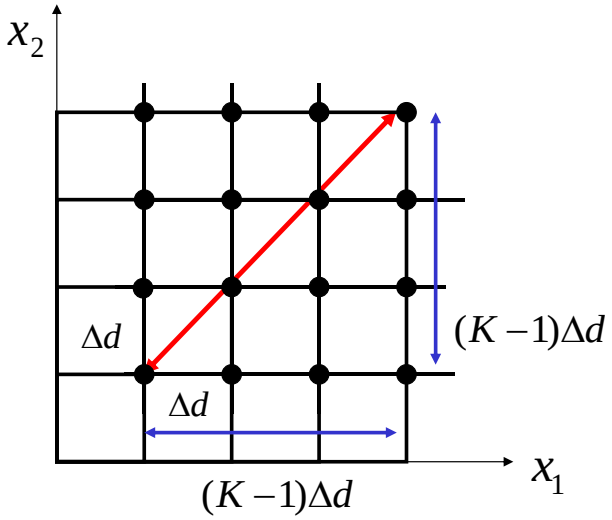


Fig.2 Training data with two dimensions

In the case of n dimensions, the number of training data, m, is simply calculated as follows:

$$m = K^n \tag{16}$$

Eqs.(15) and (16) are substituted into Eq.(2). Here, we solve Eq.(2) with respect to $r/\Delta d$, and then the following equation can be obtained:

$$\frac{r}{\Delta d} = n^{\frac{n-2}{2n}}\left(1-\frac{1}{K}\right) \tag{17}$$

In Eq.(17), $K \to \infty$ is considered. This implies a uniform distribution of the training data. Table 1 shows the convergence of $r/\Delta d$ at $K \to \infty$.

Table 1 Convergence of $r/\Delta d$ at $K \to \infty$

Dimension n	$r/\Delta d$
1	1.000
2	1.000
3	1.201
4	1.414
5	1.621
6	1.817
7	2.003
8	2.181

It is clear from Table 1 that the same value of $r/\Delta d$ can be achieved at $K \to \infty$ in the cases of n=1 and n =2. However, different values of $r/\Delta d$ can be obtained in the case of $n \geq 3$. It is assumed that the key factor to obtain a good classifier and regression is the uniform convergence of $r/\Delta d$, which implies $r/\Delta d \to 1$ at $K \to \infty$. Then, on the basis of Eq.(2), the following sufficient conditions for simple estimate of the width are summarized as follows:

(W1) It is preferable to consider the dimensions, n.

(W2) It is also preferable to consider the number of training data, m.

(W3) It is preferable to consider the maximum distance among the training data, $d_{\max}$.

(W4) $r/\Delta d \to 1$ can be achieved at $K \to \infty$

In order to satisfy the above sufficient conditions, the following simple estimate of the width in Gaussian kernel may be valid:

$$r_1 = r_2 = \cdots = r_m = \frac{d_{\max}}{\sqrt{n}\sqrt[n]{m}} \tag{18}$$

Since Eq.(18) satisfies the above sufficient conditions at $K \to \infty$, a good classifier and regression can be expected. Indeed, Eqs.(15) and (16) are substituted into Eq.(18), and we solve it with respect to $r/\Delta d$. As the result the following equation can be obtained:

$$\frac{r}{\Delta d} = (1 - \frac{1}{K}) \tag{19}$$

It is clear from Eq.(19) that $r/\Delta d \to 1$ can be achieved at $K \to \infty$, and the sufficient conditions (W1)~(W4) are satisfied. Note that Eq.(18) can be obtained under the assumption that is an uniform distribution of the training data. In addition, it is clear from Eq.(16) that numerous training data are required for a

good classifier and regression with Eq.(18). In addition, Eq.(18) is applied to all Gaussian kernels, so that width has a constant value. As described in introduction, a constant value cannot deal with the case of non-uniform distribution of the training data. Unfortunately, most real problems show non-uniform distribution of the training data, and available training data are also limited. Then, the following equation considering the non-uniform distribution of the training data is proposed in this paper.

$$r_j = \frac{d_{j,\max}}{\sqrt{n}\sqrt[n]{m-1}} \qquad j = 1, 2, \cdots, m \tag{20}$$

where r_j denotes the width of the j-th Gaussian kernel. $d_{j,\max}$ denotes the maximum distance between j-th training data and another training data in the training set. Eq.(20) is applied to each Gaussian kernel individually, unlike Eqs.(1) and (2). In Eq.(2), $d_{\max}$ is always constant in both the uniform and non-uniform distribution of the training data. This implies that non-uniform distribution of the training data cannot be taken into account. In order to consider the non-uniform distribution of the training data, $d_{j,\max}$ is employed in Eq.(20). j-th training data is already used to measure the distance $d_{j,\max}$, and the rest of training data is m-1. Thus, j-th training data is deleted among the all training data, and then the denominator in Eq.(20) is m-1.

We describe the application of Eq.(20) to the Gaussian kernel. As an example, let us consider the case of LS-SVM classifier. We must compute $\Omega_{i,j} = y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ to solve the linear system. In this case, $\Omega_{i,j}$ of r_j is computed as follow:

$$\Omega_{i,j} = y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) = y_i y_j \exp(-\frac{(\boldsymbol{x}_i - \boldsymbol{x}_j)^T (\boldsymbol{x}_i - \boldsymbol{x}_j)}{r_j^2}) \tag{21}$$

3.2 Adaptive Scaling Technique

As already described, all dimensions should be equally scaled in the development of Eq.(20). The following basic equation is used to scale all dimensions:

$$X_i = \frac{x_i - x_i^L}{x_i^U - x_i^L} \times s\,, \quad i = 1, 2, \cdots, n \tag{22}$$

where x_i is the i-th decision variable. x_i^U and x_i^L denote the upper and lower bounds on the i-th decision variable, respectively. s (>0) in Eq.(22) denotes the

scaling coefficient. Here, the scaling coefficient *s* plays an important factor. If the scaling coefficient *s* is fixed, W4 described above may not be satisfied. In order to determine *r* with a simple manner, Δd is set to 1. Please note that *r* depends on Δd, while $r/\Delta d \to 1$ at $K \to \infty$ should be satisfied. In addition, all variables should be equally scaled. Under these conditions, there are no guidelines to determine Δd, and then Δd is assumed to be 1 to ensure the distance among the training data in the scaled space. (There are no mathematical proofs on this assumption that Δd is equal to 1. This can be obtained through our numerical experiences.) Under this assumption, $r \to 1$ at $K \to \infty$ can be achieved. This implies that the minimum width with a simple manner is always greater than 1 when the decision variables are equally scaled with Eq.(22). As the result, the scaling coefficient *s* should be adaptively adjusted. Then, the adaptive scaling technique, in which the scaling coefficient is adaptively adjusted, is proposed. The algorithm of this technique is summarized as follows:

(STEP1) Initial scaling coefficient *s* (>0) is set up.

(STEP2) All dimensions are scaled by Eq.(22).

(STEP3) The width by Eq.(20) is calculated in the scaled space.

(STEP4) The minimum width $r_{\min}$ is found.

$$r_{\min} = \min_{1 \le j \le m} \{r_j\} \tag{23}$$

(STEP5) If $r_{\min} \le 1$, then scaling coefficient *s* is updated as follows:

$$s = \alpha \times s \quad (\alpha > 1) \tag{24}$$

Otherwise, the adaptive scaling algorithm will be terminated.

When some new training data are added, the adaptive scaling technique is applied again. On the basis of the author's numerical experiences, $\alpha = 1.1$ is recommended.

4. Examples and Discussions

The proposed estimate of the width is applied to two problems: First example is the classifier by LS-SVM and RBF network, and second one is the Sequential Approximate Optimization (SAO) by RBF network. In RBF network, the parameter λ_j in Eq.(8) should be determined, while the parameter *C* in Eq.(10) should be determined in LS-SVM. In RBF network, it is recommended

that λ_j in Eq.(5) have a sufficiently small value (e.g. $\lambda_j = 1.0 \times 10^{-3}$) [13], and then λ_j is set to 1.0×10^{-3}. As described in section 2.2, it is possible to consider the equivalence between LS-SVM and RBF network when the number of training data is assumed to be the number of hidden neurons. As the result, the parameter *C* in LS-SVM is set to 1000. The proposed estimate of the width can be developed based on Eq.(2), so that results by the proposed width are compared with ones by Eq.(2). In addition, the same parameter values (λ_j in Eq.(5) and *C* in Eq.(10)) are employed through numerical examples. Thus, we would like to examine the validity of the proposed estimate of the width under same parameter conditions. It may be important to use the various parameter values in the numerical examples, but the objective of this paper is to propose and examine the simple estimate of the width.

4.1 Application to the classifier by LS-SVM and RBF network

The proposed estimate of the width is applied to the classifier by LS-SVM and RBF network. The training data is separated by $f_a(\boldsymbol{x}) = 0$. In numerical examples, the number of dimensions is set to 2 to visualize the separating curve. In this case, the width by Eqs.(1) and (2) has the same constant value. The numerical training data used in this section are listed in Table 2. The range of all variables is set as follow:

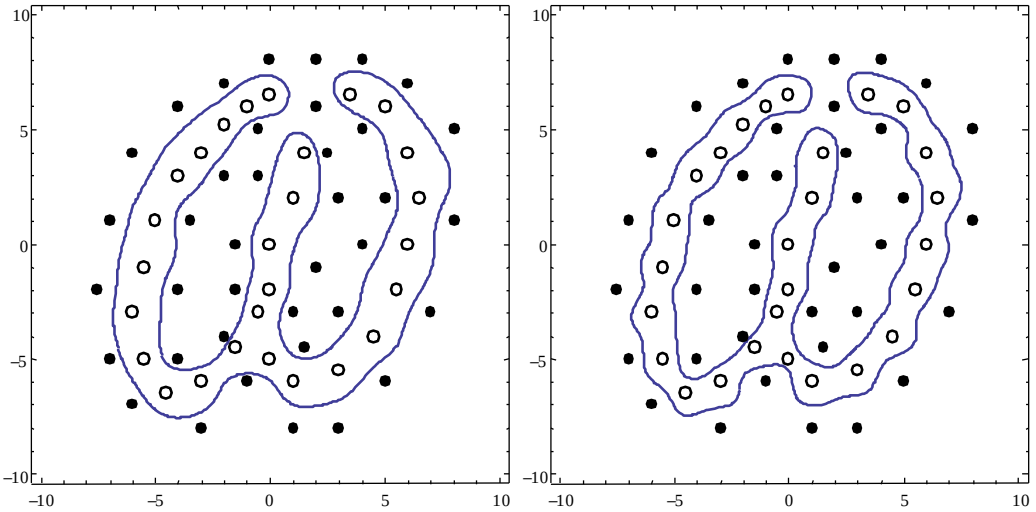
$$-10 \le x_1, x_2 \le 10 \tag{25}$$

Table 2 Numerical training data

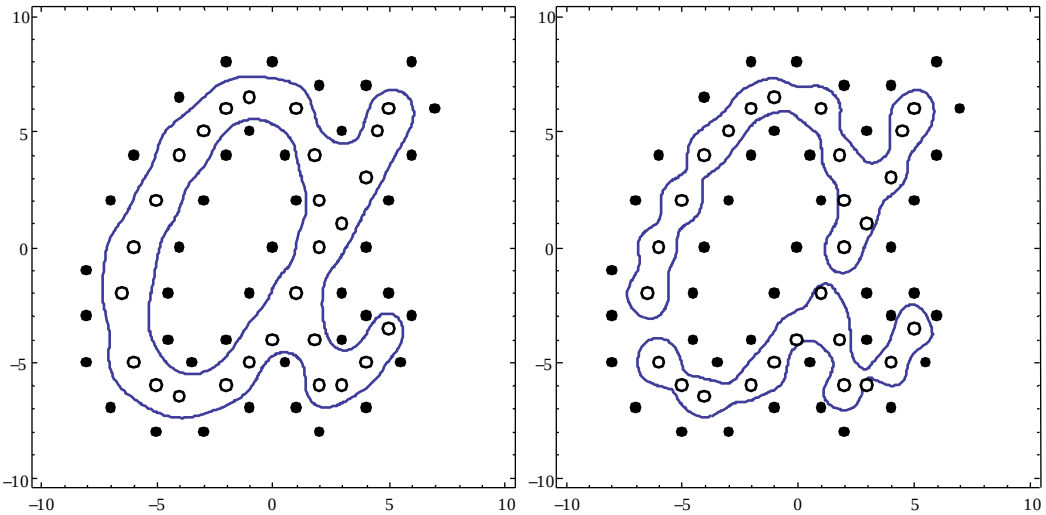
Example 1 (Omega)				Example 2 (Alpha)				Example 3 (Sigma)			
No.	x_1	x_2	Label	No.	x_1	x_2	Label	No.	x_1	x_2	Label
1	0	6.5	1	1	5	6	1	1	6	5	1
2	-1	6	1	2	4.5	5	1	2	4	5	1
3	-2	5.2	1	3	4	3	1	3	2	5	1
4	-3	4	1	4	3	1	1	4	1	5	1
5	-4	3	1	5	2	0	1	5	-1	5	1
6	-5	1	1	6	1	-2	1	6	-3	4	1
7	-5.5	-1	1	7	0	-4	1	7	-4	3	1
8	-6	-3	1	8	-1	-5	1	8	-5	2	1
9	-5.5	-5	1	9	-2	-6	1	9	-5.5	0	1
10	-4.5	-6.5	1	10	-4	-6.5	1	10	-6	-2	1
11	-3	-6	1	11	-5	-6	1	11	-5.5	-4	1
12	-1.5	-4.5	1	12	-6	-5	1	12	-5	-6	1
13	-0.5	-3	1	13	-6.5	-2	1	13	-3	-5.5	1
14	0	-2	1	14	-6	0	1	14	-1	-6	1
15	0	0	1	15	-5	2	1	15	0	-5.5	1
16	1	2	1	16	-4	4	1	16	1	-4	1
17	1.5	4	1	17	-3	5	1	17	2	-3	1
18	0	-5	1	18	-2	6	1	18	2.4	-1	1
19	1	-6	1	19	-1	6.5	1	19	2.8	1	1
20	3	-5.5	1	20	1	6	1	20	2	3	1
21	4.5	-4	1	21	1.8	4	1	21	5	8	-1
22	5.5	-2	1	22	2	2	1	22	2	8	-1
23	6	0	1	23	1.8	-4	1	23	0	8	-1
24	6.5	2	1	24	2	-6	1	24	-2	8	-1
25	6	4	1	25	3	-6	1	25	-3	7	-1
26	5	6	1	26	4	-5	1	26	-5	6	-1
27	3.5	6.5	1	27	5	-3.5	1	27	-6	4	-1
28	4	8	-1	28	6	-3	-1	28	-7	2	-1
29	6	7	-1	29	5.5	-5	-1	29	-8	0	-1
30	8	5	-1	30	4	-7	-1	30	-8	-3	-1
31	8	1	-1	31	2	-8	-1	31	-7	-6	-1
32	7	-3	-1	32	1	-7	-1	32	-6	-8	-1
33	5	-6	-1	33	0.5	-5	-1	33	-4	-8	-1
34	3	-8	-1	34	-1	-7	-1	34	-2	-8	-1
35	1	-8	-1	35	-3	-8	-1	35	0	-8	-1
36	-1	-6	-1	36	-5	-8	-1	36	2	-7	-1
37	-3	-8	-1	37	-7	-7	-1	37	3	-5	-1
38	-6	-7	-1	38	-8	-5	-1	38	4	-4	-1
39	-7	-5	-1	39	-8	-3	-1	39	5	-2	-1
40	-7.5	-2	-1	40	-8	-1	-1	40	5	0	-1
41	-7	1	-1	41	-7	2	-1	41	4.5	2	-1
42	-6	4	-1	42	-6	4	-1	42	4	3	-1
43	-4	6	-1	43	-4	6.5	-1	43	7	3	-1
44	-2	7	-1	44	-2	8	-1	44	8	4	-1
45	0	8	-1	45	0	8	-1	45	8	6	-1
46	2	8	-1	46	2	7	-1	46	0	3	-1
47	2	6	-1	47	3	5	-1	47	-1	3	-1
48	4	5	-1	48	4	7	-1	48	-3	2	-1
49	5	2	-1	49	6	8	-1	49	-4	0	-1
50	4	0	-1	50	7	6	-1	50	-4	-2	-1
51	3	-3	-1	51	6	4	-1	51	-4	-4	-1
52	1.5	-4.5	-1	52	5	2	-1	52	-2.5	-4	-1
53	1	-3	-1	53	4	0	-1	53	-1	-4	-1
54	2	-1	-1	54	3	-2	-1	54	0	-2	-1
55	3	2	-1	55	3	-4	-1	55	1	1	-1
56	2.5	4	-1	56	4	-3	-1				
57	-0.5	3	-1	57	5	-2	-1				
58	-1.5	0	-1	58	-1	5	-1				
59	-1.5	-2	-1	59	-2	4	-1				
60	-2	-4	-1	60	-3	2	-1				
61	-4	-5	-1	61	-4	0	-1				
62	-4	-2	-1	62	-4.5	-2	-1				
63	-3.5	1	-1	63	-4.5	-4	-1				
64	-2	3	-1	64	-3.5	-5	-1				
65	-0.5	5	-1	65	-2	-4	-1				
				66	-1	-2	-1				
				67	0	0	-1				
				68	1	2	-1				
				69	0.5	4	-1				

Fig.3 shows the results of separating curves by LS-SVM. In Fig.3, +1 is assigned to the white circles and -1 is assigned to the black circles. The left hand sides of Fig.3 show the result by Eq.(20), and the right hand sides show the one by Eq.(2). The solid line represents $f_a(\boldsymbol{x}) = 0$, which separates the training data. In addition, Fig.4 shows the three dimensional view of $f_a(\boldsymbol{x})$. The left hand sides show the $f_a(\boldsymbol{x})$ by Eq.(20), and the right hand sides show the one by Eq.(2).

Example 1 (Omega)



Example 2 (Alpha)



Example 3 (Sigma)

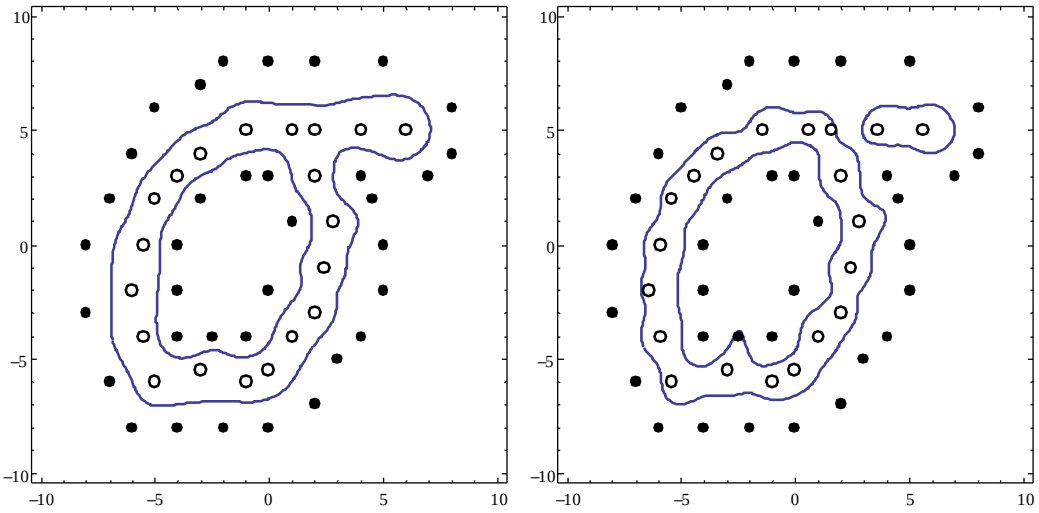
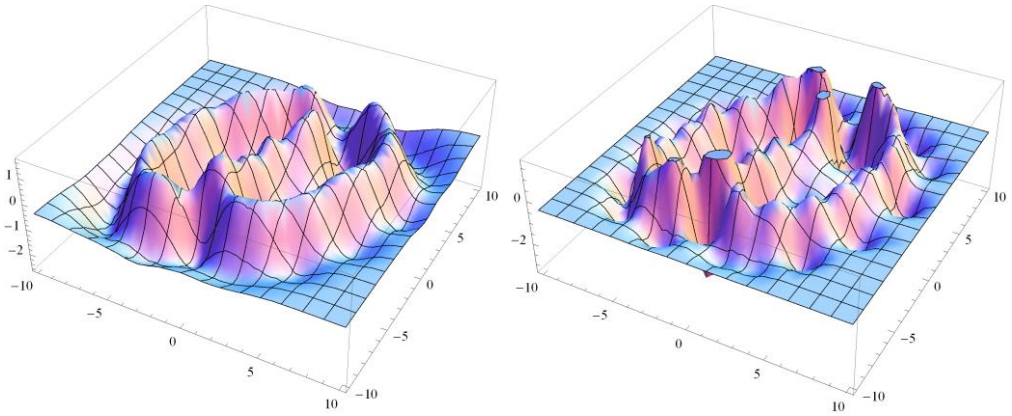
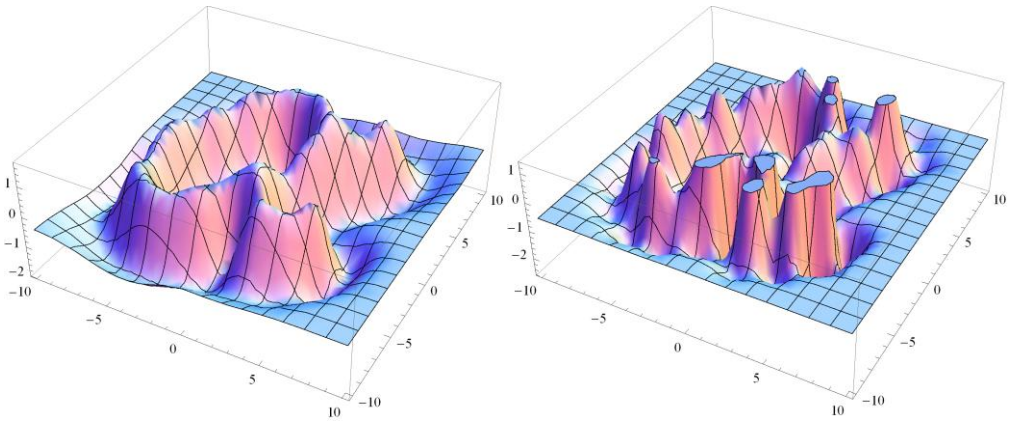


Fig.3 Separating hyperplane by LS-SVM

Example 1 (Omega)



Example 2 (Alpha)



Example 3 (Sigma)

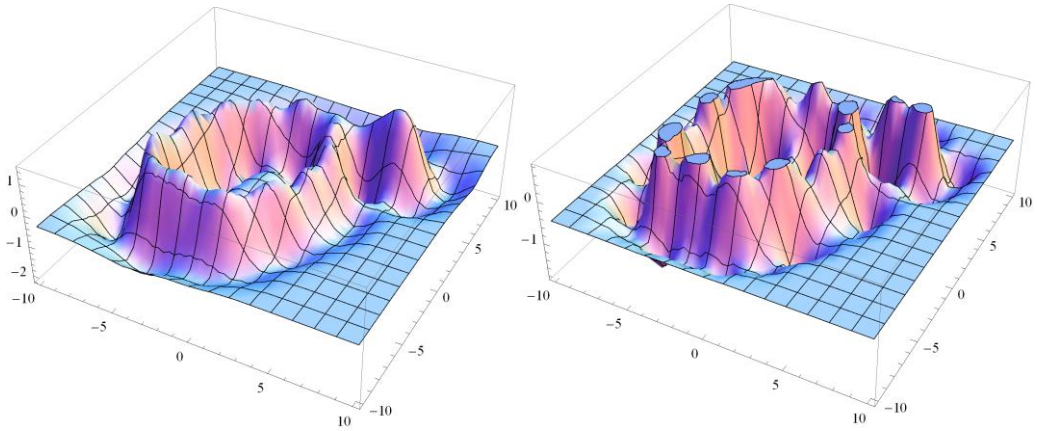
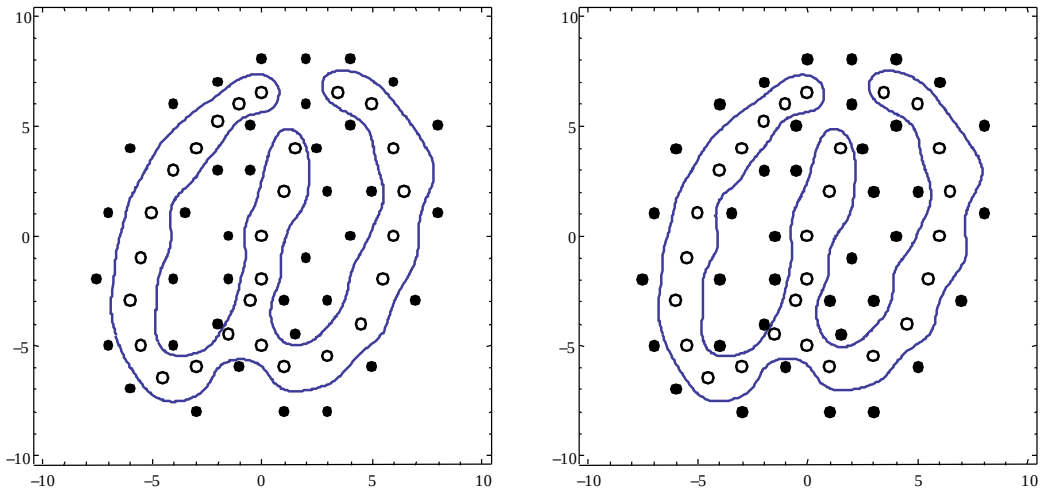


Fig.4 3D view of LS-SVM classifier

Next, separating curves by RBF network is obtained with the same training data. The objective of this is to examine the availability of the proposed estimate of the width to the RBF network. The results are shown in Fig.5. The left hand sides of Fig.5 show the results by LS-SVM, and the right hand sides show the ones by RBF network.



LS-SVM RBF network

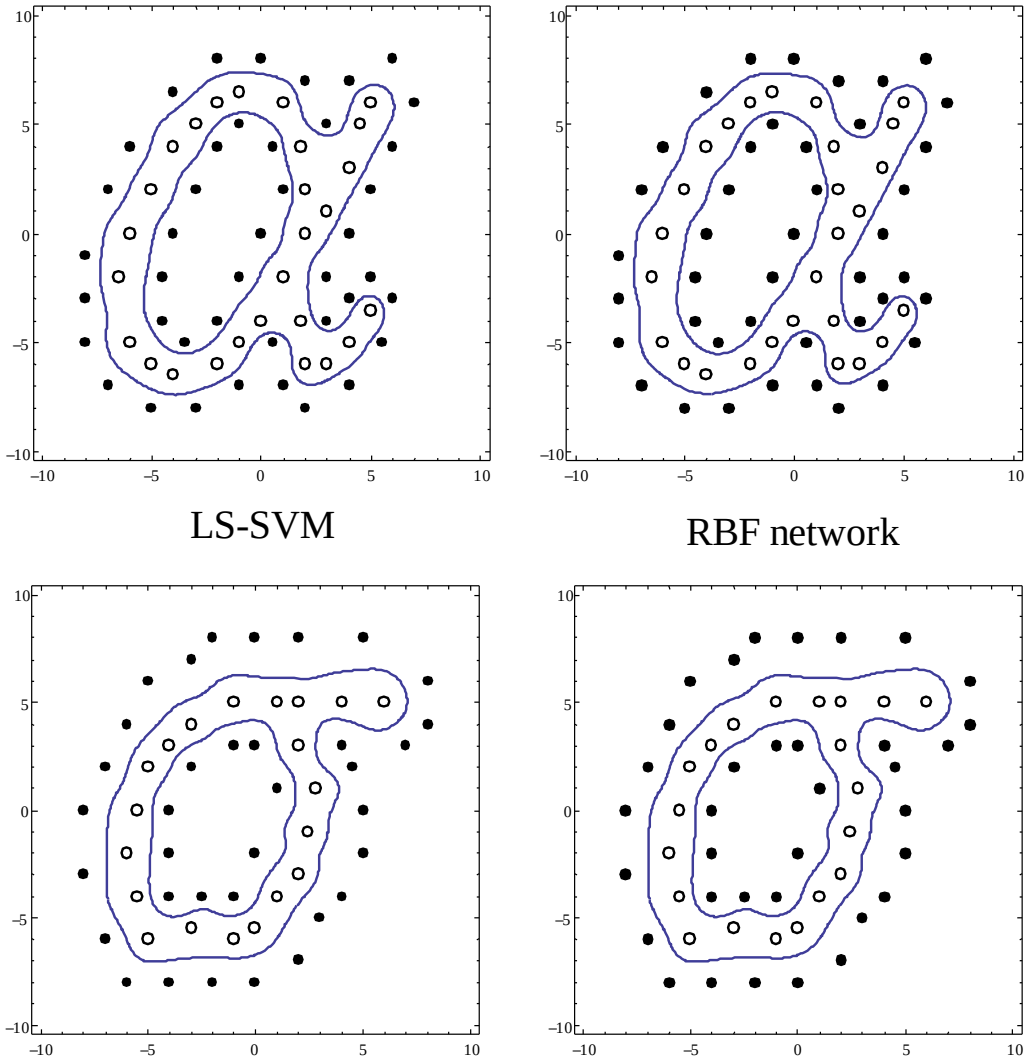


Fig.5 Comparison between LS-SVM and RBF network

4.2 Application to Sequential Approximate Optimization

In this section, the proposed estimate of the width with the adaptive scaling technique is applied to sequential approximate optimization (SAO) by the RBF network. SAO methodology is widely studied in the structural engineering field [17-21]. In the engineering optimization, it is very important to reduce the computational costs and the experiments. This is equivalent to the reduction of the training data. Thus, it is important to find the global minimum with a little number of training data.

General procedure of the SAO is summarized as follows:

(STEP1) The training data is distributed.

(STEP2) The objective is evaluated at training data. Therefore, the regression which is called the response surface is constructed.

(STEP3) In order to find the optimum of the response surface, optimization technique is applied to the response surface. The optimum of response surface is often called the approximate optimum.
(STEP4) The approximate optimum obtained in STEP3 is added as the new training data. In addition, a few training data are also added.

For better understanding of the SAO procedure, let us consider a following example:

$$f(x) = \sum_{i=1}^{5} i \cos\left[(i+1)x + i\right] \rightarrow \min \tag{26}$$

$$0 \le x \le 7.5 \tag{27}$$

In the SAO, some training data are distributed, and the function is evaluated at the training data. Then, the response surface is constructed by using the training data and the function values. In Fig.6, the dashed line shows the original function, and solid line represents the response surface. The black dots denote the training data. In Fig. 6, three training data are distributed. The global minimum of the response surface, which denotes the triangle in Fig.6, can be found.

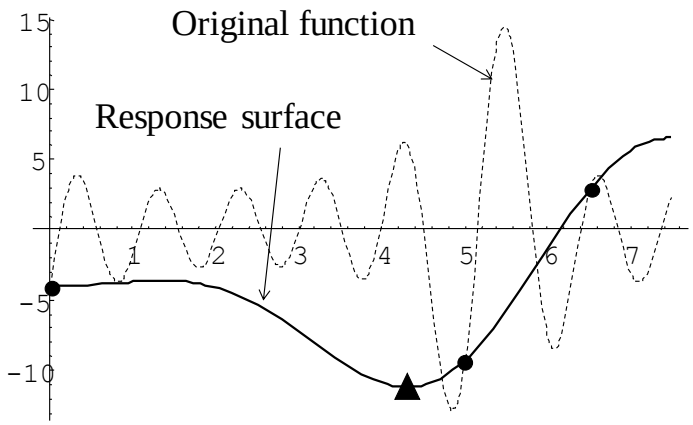
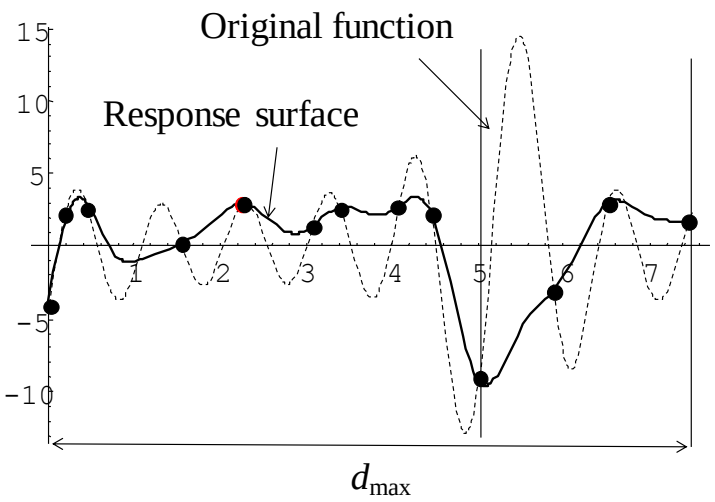
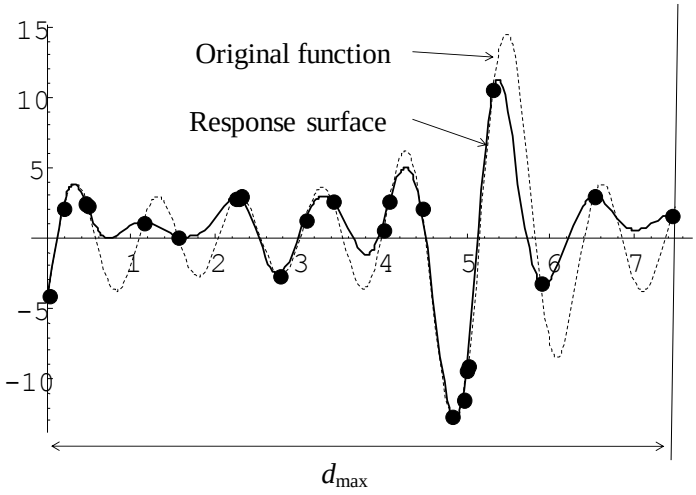


Fig. 6 Response surface

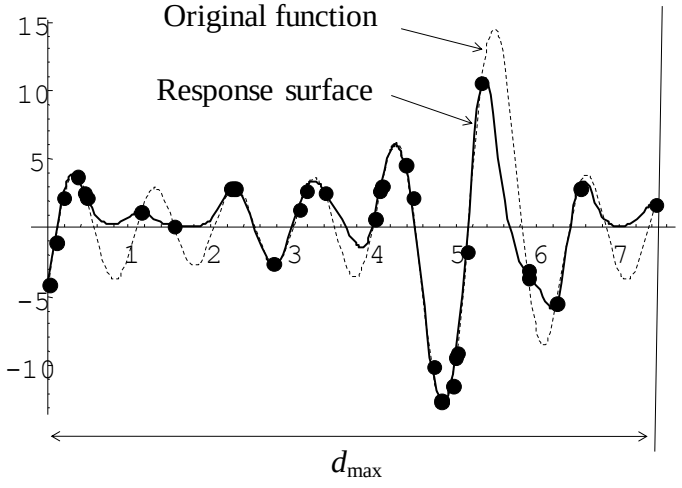
The global minimum of the response surface is added as the new training data. In addition, a few new training data are also added. Then, the response surface is constructed with new training data, and the global minimum of the response surface can be found. Through this iterative process, the global minimum of the original function can be found. The illustrative approximation processes are shown in Fig.7.



(a) Response surface at 13 training data



(b) Response surface at 23 training data



(c) Response surface at 33 training data

Fig. 7 Process of approximation of original function

It is clear from Fig. 7 that the training data show non-uniform distribution. When Eq.(2) is employed to construct the response surface, $d_{\max}$ is always constant. Since the training data is added in the SAO, the number of the training data is increased. As the result, the width in the Gaussian kernel will become smaller and smaller. On the other hand, $d_{j,\max}$is employed in the proposed width, and then the width in the Gaussian kernel does not always become smaller and smaller.

The following optimization problem is considered.

$$f(\boldsymbol{x}) = \frac{1}{2}\sum_{i=1}^{n}\left(x_i^4 - 16x_i^2 + 5x_i\right) \rightarrow \min \tag{37}$$

$$-5 \le \boldsymbol{x} \le 5 \tag{38}$$

The global minimum $\boldsymbol{x}_G$ is $\boldsymbol{x}_G = (-2.9035, -2.9035, \cdots, -2.9035)^T$. The dimension n is set to ten in this numerical example. In this case, the objective function at $\boldsymbol{x}_G$ is $f(\boldsymbol{x}_G) = -391.661$. The particle swarm optimization (PSO), which is a population-based optimization technique, is applied to this problem directly. The number of particles is set to 20, and the maximum number of iteration is set to 500. 20 particles move the positions at every search iteration, in order to find the global minimum. In other words, the global minimum can be actually found with 10000 ($= 20 \times 500$) training data. In section 4.1, the training data is fixed, but the training data (=particles) will move in PSO. This point is different in case of section 4.1. The PSO results are shown in Table 3 through 10 trials.

Table 3 Results of direct search by the PSO

Trial	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	obj.
1	-2.9035	-2.9035	-2.9035	-2.9035	-2.9035	-2.9035	-2.9035	-2.9035	-2.9035	-2.9035	-391.6600
2	-2.9036	-2.9036	-2.9036	-2.9035	-2.9035	-2.9035	-2.9035	2.7468	-2.9035	-2.9035	-377.5200
3	-2.9046	-2.9027	-2.9021	-2.9038	-2.9032	-2.9047	-2.9015	-2.9043	-2.9029	-2.9049	-391.6600
4	-2.9036	-2.9035	-2.9036	-2.9036	-2.9035	-2.9035	-2.9036	-2.9035	-2.9036	-2.9036	-391.6600
5	-2.9035	-2.9035	-2.9035	-2.9035	-2.9035	-2.9035	-2.9035	-2.9035	-2.9035	-2.9035	-391.6600
6	-2.9035	-2.9035	-2.9035	-2.9035	-2.9036	-2.9035	2.7468	-2.9036	-2.9035	-2.9035	-377.5200
7	-2.9035	-2.9035	-2.9035	-2.9035	-2.9035	-2.9035	-2.9035	-2.9035	-2.9035	-2.9035	-391.6600
8	-2.9035	-2.9035	-2.9035	-2.9035	-2.9035	-2.9035	-2.9035	-2.9035	-2.9035	-2.9035	-391.6600
9	-2.9036	-2.9034	-2.9036	-2.9036	-2.9034	-2.9036	-2.9035	-2.9035	-2.9036	-2.9036	-391.6600
10	-2.9035	-2.9035	-2.9035	-2.9035	-2.9035	-2.9035	-2.9035	-2.9035	-2.9035	-2.9035	-391.6600

First, 30 training data are distributed at random. The SAO algorithm in [22] is employed to find the global minimum. The maximum number of training data is set to 500. Therefore, the SAO algorithm will be started at 30 training data. The algorithm will be terminated at 500 training data. Ten trials with different

random seed are performed, and these results are listed in Table 4 and 5. Table 4 shows the result by Eq.(2), while Table 5 shows the result by Eq.(20).

Table 4 The result with the simple estimate by Nakayama

Trial	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	obj.
1	-2.6590	-2.7695	-2.1054	3.0079	-3.2740	-2.7108	-2.6121	-2.3852	2.7003	-0.9410	-314.1416
2	-2.0759	-2.7692	2.2905	-1.9245	2.7989	-2.6619	1.9850	-2.0531	2.4101	-1.4912	-273.9235
3	-2.9410	-2.8063	-3.0479	-2.6390	-2.9142	-2.8250	-2.8031	0.9497	-2.7844	0.7700	-318.2305
4	-2.8926	-2.9712	-2.0732	-3.2507	-2.8878	-2.7514	-3.3558	-1.2210	-2.2069	-1.5235	-324.6039
5	-2.9462	2.5448	-2.9124	-2.7911	-3.0628	2.3251	0.6797	-2.9464	-2.9779	2.7018	-308.3509
6	2.4241	-0.3086	3.1126	-3.6446	-1.6238	-2.6086	-3.4843	1.1949	-0.0281	1.9153	-192.1104
7	-2.8263	-2.9896	2.7270	-3.1419	-2.6376	-2.6945	-2.4389	-3.2174	-1.6055	-3.3261	-347.9492
8	-2.9634	-2.7063	2.5490	-2.6656	-2.5071	-2.9957	-2.9850	2.8242	-3.0073	2.2608	-341.3516
9	-2.6664	2.4891	-2.7121	-2.2098	-2.7480	-0.1432	1.5402	-2.7654	-3.2011	-3.0773	-300.2067
10	-2.3563	-2.7148	-2.7854	-2.6837	2.3436	-1.2683	2.4122	-2.6995	-2.7142	-2.1761	-321.3327

Table 5 The result with the proposed simple estimate

Trial	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	obj.
1	-2.8096	-2.8428	-2.8138	-2.8923	-2.9249	-2.8322	-2.7792	-2.9189	-2.9251	-2.8174	-390.8271
2	-2.8796	-2.7992	-2.9779	-2.9068	-2.8315	-2.9377	-2.9344	-2.9574	-2.9395	-2.8551	-391.1336
3	-2.9828	-2.8559	-2.8442	-2.7923	-2.8739	-2.9228	-2.7560	-2.9362	-2.9070	-2.8233	-390.7393
4	-2.8815	-2.8551	-2.9466	-2.8876	-2.8993	-2.8246	-2.8708	-2.8602	-2.8473	-2.8967	-391.3663
5	-2.9453	-2.9166	-2.7878	-2.8491	-2.8713	-2.7786	-2.9730	-2.7636	-2.9755	-2.9006	-390.5783
6	-2.9192	-2.8345	-2.9351	-2.8462	-2.9019	-2.8998	-2.8975	-2.9197	2.6884	-2.8685	-377.2915
7	-2.9376	2.6801	-2.8858	-2.8898	-2.8615	-2.8626	-2.9049	-2.8773	-2.8294	-2.8921	-377.2669
8	2.6569	-2.9470	-2.9554	-2.8557	-2.8986	-2.9416	-2.8419	-2.8627	-2.9532	-2.8738	-377.1143
9	-2.9211	-2.8614	2.7425	-2.9349	-2.8388	-2.8856	-2.8809	-2.8981	-2.9182	-2.8922	-377.3801
10	-2.8901	-2.8468	-2.8828	-2.9049	-2.9252	2.6898	-2.8235	-2.9735	-2.8813	-2.8938	-377.2004

4.3 Discussions

It is clear from above examples that the proposed estimate of the width works well, compared with Eq.(2). In section 4.1, more smooth separating curves obtained when the proposed estimate of the width is used. In section 4.1, the training data are distributed non-uniformly. In such cases, it is preferable to use different widths to each Gaussian kernel. The proposed estimate of the width is applicable to such cases, and shows the good results. The result implies that the individual assignment of the width to the Gaussian kernel is important and valid. The proposed estimate of the width is applied to RBF network with the same training data. The separating curves by RBF network can clearly separate the training data, and is qualitatively similar to one by LS-SVM under the parameter setting used in this paper. In section 4.2, the proposed simple estimate of the width with the adaptive scaling technique is applied to the sequential approximate optimization. RBF network is employed to construct the response surface. By comparing Table 4 with Table 5, it is clear that better results can be obtained.

Through five trials (Trial No.1 – Trial No.5 in Table 5), the approximate global minimum could be obtained. In addition, the other trials (Trial No.6 – Trial No.10 in Table 5) yield quasi-optima. However, the global minimum cannot be obtained with Eq.(23). These results also imply that it is preferable to apply a different width to each basis function.

5. Conclusions

The Gaussian kernel is widely employed in the machine learning techniques. It is important to determine the appropriate width in the Gaussian kernel with a simple manner. In this paper, a new simple estimate of the width in the Gaussian kernel has been proposed. The proposed estimate could be developed by examining the estimate by Nakayama. Through the examination, we considered four sufficient conditions for the simple estimate of the width. Therefore, it is preferable to consider (1) the dimensions, (2) the number of the training data, (3) the maximum distance among the training data, and (4) the uniform convergence. Based on these conditions, we first proposed the simple estimate of the width. However, this estimate does not consider the non-uniform distribution of the training data. Then, we have developed the new simple estimate of the width in the Gaussian kernel. The proposed width is applied to each Gaussian kernel, and the non-uniform distribution of the training data could be also taken into account. In addition, the adaptive scaling technique has also been proposed. In this technique, the scaling coefficient is adaptively adjusted, in order to determine the width. The proposed estimate of the width has been applied to LS-SVM and RBF network. The classifier and SAO have been taken as the numerical examples. In the classifier, two-dimensional problems have been handled, in order to visualize the separated curves. The training data are uniformly distributed. It has been confirmed that individual assignment of the width to the Gaussian kernel is important and valid. In the SAO, RBF network is employed to construct the response surface. The better result can be obtained by the proposed estimate of the width. The determination of the width in the Gaussian kernel still remains one of the important topics [23]. The proposed estimate of the width clearly belongs to heuristic approaches, and then we could not give rigorous proofs. Therefore the proposed estimate of the width is a proposal, and is not an absolute one.

Appendix

In section 4.2, particle swarm optimization (PSO) is employed as the optimization technique [24]. Several models of PSO have been proposed. The most popular among them is the g-best model. In this paper, the g-best model is employed. The position and velocity of particle d are represented by $\boldsymbol{x}_d^k$ and $\boldsymbol{v}_d^k$, respectively, and k represents the iteration step. The position and velocity of particle d at iteration k+1 are calculated by the following equations.

$$\boldsymbol{v}_d^{k+1} = w\boldsymbol{v}_d^k + c_1 r_1(\boldsymbol{p}_d^k - \boldsymbol{x}_d^k) + c_2 r_2(\boldsymbol{p}_g^k - \boldsymbol{x}_d^k) \tag{A1}$$

$$\boldsymbol{x}_d^{k+1} = \boldsymbol{x}_d^k + \boldsymbol{v}_d^{k+1} \tag{A2}$$

The coefficient w in (A1) is called the inertia term, and it linearly decreases as the search proceeds [25, 26]. Parameters r_1 and r_2 denote random numbers between [0,1]. Weighting coefficients c_1 and c_2 are recommended to maintain the following relationship.

$$c_1 + c_2 \le 4 \tag{A3}$$

c_1=c_2=2 is used in this paper, according to [27, 28]. The position vector $\boldsymbol{p}_d^k$, called the p-best, represents the best position of particle d until the k-th iteration, and $\boldsymbol{p}_g^k$, called the g-best, represents the best position in the swarm till the k-th iteration. The basic algorithm of PSO, which is called a g-best model, is described briefly below.

(STEP1) Define the search domain in advance. Determine the swarm population size and the maximum search iteration number, $k_{\max}$. Initialize the iteration counter k as k =1. Randomly generate the initial position and velocity of each particle in the search domain.

(STEP2) Calculate the objective function of each particle.

(STEP3) Select the p-best and g-best.

(STEP4) Update the velocity and position of each particle by (A1) and (A2).

(STEP5) Update the inertia term by using following equation.

$$w = w_{\max} - \frac{w_{\max} - w_{\min}}{k_{\max}} \times k \tag{A4}$$

where, $w_{\max}$=0.9 and $w_{\min}$=0.4 are used, in general [29].

(STEP6) If k is less than k_{max}, the iteration counter is increased as k=k+1, and the algorithm returns to STEP2. Otherwise, the algorithm terminates.

References:

1. W. An, Y. Sun, An Equivalence between SILF-SVR and Ordinary Kriging, Neural Processing Letters, 23 (2006) 133-141.
2. P. Andras, The Equivalence of Support Vector Machine and Regularization Neural Networks, Neural Processing Letters, 65 (2002) 97-104.
3. T. Poggio, F. Girosi, Networks for Approximation and Learning, Proc. of the IEEE, 78(9) (1990) 1481-1497.
4. J.A.K. Suykens, J. Vandewalle, Least Squares Support Vector Machine Classifiers, Neural Processing Letters, 9 (1999) 293-300.
5. J.A.K. Suykens, J.D. Brabanter, L. Lukas, J. Vandewalle, Weighted Least Squares Support Vector Machines: Robustness and Sparse Approximation, Neurocomputing, 48 (2002) 85-105.
6. Z. Wang, S. Chen, New Least Squares Support Vector Machines Based on Matrix Patterns, Neural Processing Letters, 26 (2007) 41-56.
7. H.M. Chi, O.K. Ersoy, Recursive Update Algorithm for Least Squares Support Vector Machines, Neural Processing Letters, 17 (2003) 165-173.
8. D. Anguita, A. Boni, Digital Least Squares Support Vector Machines, Neural Processing Letters, 18 (2003) 65-72.
9. R. Zhang, W. Wang, Y. Ma, C. Men, Least Square Transduction Support Vector Machine, Neural Processing Letters, 29 (2009) 133-142.
10. N. Benoudjit, M. Verleysen, On the Kernel Widths in Radial-Basis Function Networks, Neural Processing Letters, 18 (2003) 139-154.
11. A. Saha, J.D. Keeler, Algorithms for Better Representation and Faster Learning in Radial Basis Function Networks, Advances in Neural Information Processing Systems 2 (1989) Edited by Touretzky, D.S., 482-489.
12. Haykin, S. Neural Networks: A Comprehensive Foundation, Macmillan College Publishing Company, (1994), pp.236-284.
13. H. Nakayama, M. Arakawa, R. Sasaki, Simulation-Based Optimization Using Computational Intelligence, Optimization and Engineering, 3 (2002) 201-214.
14. H. Nakayama, Y. Yun, M. Yoon, Sequential Approximate Multiobjective Optimization Using Computational Intelligence, Springer, 2009.
15. M.J.L. Orr, Introduction to Radial Basis Function Networks, http://www.anc.ed.ac.uk/rbf/rbf.html (On-line available)
16. R.H. Myers, D.H. Montgomery, Response Surface Methodology, Wiley,New York, 1995.
17. M.J. Sasena, P.Y. Papalambros, P. Goovaerts, Exploration of metamodeling sampling criteria for constrained global optimization, Engineering Optimization, 34(3) (2002) 263-278.
18. G.G. Wang, Adaptive Response Surface Method Using Inherited Latin Hypercube Design Points, Journal of Mechanical Design, 125 (2003) 210-220.

19. H. Kurtaran, A. Eskandarian, D. Marzougui, N.E. Bedewi, Crashworthiness Design Optimization Using Successive Response Surface Approximations, Computational Mechanics, 29 (2002) 409-421.
20. G.G. Wang, S. Shan, Review of Metamodeling Techniques in Support of Egnieering Design Optimization, Journal of Mechanical Design, 129 (2007) 370-380.
21. A.A. Muller, A. Messac, Extended Radial Basis Functions: More Flexible and Effective Metamodeling, AIAA Journal, 43(6) (2005) 1306-1315.
22. S. Kitayama, M. Arakawa, K. Yamazaki, Sequential Approximate Optimization Using Radial Basis Function network for engineering optimization, Optimization and Engineering, (2010), (On-line available)
23. H.J. Wang, C.S. Leung, P.F. SUM, G. Wei, Kernel Width Optimization for Faulty RBF Neural Networks with Multi-node Open Fault, Neural Processing Letters, 32 (2010) 97-107.
24. J. Kennedy, R.C. Eberhart, Swarm Intelligence, Morgan Kaufmann Publishers, 2001.
25. G. Venter, J.S. Sobieski, Particle Swarm Optimization, AIAA Journal, 41(8) (2003) 1583-1589.
26. S. Kitayama, M. Arakawa, K. Yamazaki, Penalty Function Approach for the Mixed Discrete Non-Linear Problems by Particle Swarm Optimization, Structural and Multidisciplinary Optimization, 32(3) (2005) 191-202.
27. P.C. Fourie, A.A. Groenwold, The Particle Swarm Optimization Algorithm in Size and Shape Optimization, Structural and Multidisciplinary Optimization, 23-4 (2002) 259-267.
28. J.F. Schutte, A.A. Groenwold, Sizing Design of Truss Structures Using Particle Swarms, Structural and Multidisciplinary Optimization, 25 (2003) 261-269.
29. H. Yoshida, K. Kawata, Y. Fukuyama, S. Takayama, Y. Nakanishi, A Particle Swarm Optimization for Reactive Power and Voltage Control Considering Voltage Security Assessment, IEEE Trans Power Systems, 15(4) (2000) 1232–1239