Robust Support Vector Regression Networks for Function Approximation With Outliers

Chen-Chia Chuang, Shun-Feng Su, *Member, IEEE*, Jin-Tsong Jeng, *Member, IEEE*, and Chih-Ching Hsiao

***Abstract*—Support vector regression (SVR) employs the support vector machine (SVM) to tackle problems of function approximation and regression estimation. SVR has been shown to have good robust properties against noise. When the parameters used in SVR are improperly selected, overfitting phenomena may still occur. However, the selection of various parameters is not straightforward. Besides, in SVR, outliers may also possibly be taken as support vectors. Such an inclusion of outliers in support vectors may lead to seriously overfitting phenomena. In this paper, a novel regression approach, termed as the robust support vector regression (RSVR) network, is proposed to enhance the robust capability of SVR. In the approach, traditional robust learning approaches are employed to improve the learning performance for any selected parameters. From the simulation results, our RSVR can always improve the performance of the learned systems for all cases. Besides, it can be found that even the training lasted for a long period, the testing errors would not go up. In other words, the overfitting phenomenon is indeed suppressed.**

***Index Terms*—Outliers, robust learning, support vector regression (SVR).**

I. Introduction

The SUPPORT vector machine (SVM) is a universal approach for solving the problems of multidimensional function estimation. Those approaches are all based on the Vapnik–Chervonenkis (VC) theory [1], [2]. Initially, it was designed to solve pattern recognition problems, where in order to find a decision rule with good generalization capability, a small subset of the training data, called the support vectors [1], [2], are selected. Experiments showed that it is easy to recognize high-dimensional identities using a small basis constructed from the selected support vectors [3]. Recently, SVM has also been applied to various fields successfully such as classification [4], [5], time prediction [6] and regression [7]–[11]. When SVM is employed to tackle the problems of function approximation and regression estimation, the approaches are often referred to as the support vector regression (SVR) [6]–[11]. The SVR type of function approximation is very effective, especially for the case of having a high-dimensional input space. Another important advantage for using SVR in function approximation is that the number of free parameters in the function approximation scheme is equal to the number of support vectors. Such a number can be obtained by defining the width of a tolerance band. Thus, the selection of the number of free parameters can be directly related to the approximation accuracy and does not have to depend on the dimensionality of the input space or other factors as that in the cases of multilayer feedforward neural networks.

In general, for any real-world applications, observations are always subject to noise or outliers. The intuitive definition of outliers is that "an observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism" [12]. Outliers may occur due to various reasons, such as erroneous measurements or noisy phenomenon appearing in the tail portion of some noise distribution functions. When the obtained observations contain noise or outliers, the learning process being unaware of those situations may try to fit those unwanted data and this behavior may lead to a corrupted approximation function. This phenomenon is often called overfitting [13]–[16], which usually can lead to the loss of generalization performance in the test phase. In [9], SVR has been shown to have better robust properties for various signal-to-noise ratios. In [6], SVR has also been shown to have excellent performance for both the ε-insensitive function and Huber's robust function matching the correct type of noise in an application of time series prediction. Besides, a general cost function for SVR has also been proposed in considering error distributions [10], [11]. In [16], the SVM with weighted least square (LS-SVM) is proposed to overcome the effects of outliers. However, when the parameters in those approaches are not properly chosen, the final results may be affected by its parameters. This property has also been mentioned in [16]. The selection of parameters in SVR is not straightforward. In fact, for different examples, the optimal sets of parameters are also different. Thus, while facing with real problems, some methods such as k-fold cross-validation, VC bounds, Xi–Alpha bound, or radius–margin bound [34], must be used to find those proper parameters. A fundamental problem of those validation approaches is that they need to use training data sets to determine whether the selected parameters are proper or not. If not, another set of parameters is tried again. First, such a trial-and-error procedure may involve extensive computation. Second, when the training data used for test contain outliers, the validation result may not be trustable. In addition, if LS-SVM or other cost functions are used, conjugate gradient methods must be used. Hence, the speed of convergence depends on

Manuscript received August 26, 1999; revised December 17, 2001 and April 1, 2002. This work was supported in part by the National Science Council of Taiwan under Grant NSC 89-2218-E-009-091.

C.-C. Chuang is with the Department of Electronic Engineering, Hwa-Hsia College of Technology and Commerce, Taipei 235, Taiwan, R.O.C.

S.-F. Su and C.-C. Hsiao are with the Department of Electrical Engineering, National Taiwan University of Science and Technology, Taipei 106, Taiwan, R.O.C.

J.-T. Jeng is with the Department of Computer Science and Information Engineering, National Huwei Institute of Technology, Huwei Jen, Yunlin 632, Taiwan, R.O.C.

Digital Object Identifier 10.1109/TNN.2002.804227

the condition numbers of matrices. This phenomenon is also mentioned in [16]. Besides, in those approaches, outliers may also possibly be taken as support vectors. We agree that good selections of the regularization constant and kernel parameters in SVR are capable of ensuring good generalization of the obtained model. However, when the regularization constant and kernel parameters are not properly selected, the resultant performances may not be good. Moreover, we are also aware of the fact that good generalization of the obtained SVR model can be ensured due to the optimization process. Nevertheless, the good generalization property comes from the use of the regularization term or the so-called weight decay term. The idea of such an approach is to minimize the values of weights during the process of minimizing errors. Such a process treats all training data equally. Even though the minimization of weights may indeed result in good generalization, the effects are different from and inferior to that of what have been traditionally used for robust learning against outliers [13], [14], [18]. Traditional robust learning is to find ways of reducing the effects of outliers. Our approach is to provide a mechanism that can facilitate such a concept.

In this paper, a novel regression approach, termed as the robust SVR (RSVR) network, is proposed to enhance the robust capability of SVR. The basic idea of the approach is to adopt the concept of traditional robust statistics [22]–[24] to fine tune the model obtained by SVR. Fundamentally, the proposed network is an SVR, but equipped with a robust learning algorithm. Simulation results of the proposed approach have shown the effectiveness of the approximated function in discriminating against outliers.

The remainder of this paper is outlined as follows. Section II describes the concept of robust learning and the problems encountered in traditional robust learning algorithms. The fundamental ideas for SVR are briefly introduced in Section III. The robust properties of SVR are also discussed in the section. In Section IV, the RSVR network is proposed and its training process is introduced. Section V gives experimental results. Various loss functions were used in our simulations. Those results all showed the superiority of RSVR to the original SVR. Concluding remarks are given in Section VI.

II. Robust Learning Concept

In neural-network applications, various robust learning algorithms that adopt robust statistics methods have been proposed to deal with outliers in the literature [14], [17]–[21]. Based on similar ideas, robust radial basis function networks [19], robust interval regression [21] and robust principal component analysis (PCA) algorithms [20], etc., have been proposed to deal with outliers existing in observations for various applications successfully. However, some problems exist in the use of those robust learning methods. A fundamental problem is how to determine the initial values of parameters. Robust learning algorithms are to discriminate against outliers in the learning process. Whether a point is considered as an outlier is determined by the estimated errors because an outlier is supposed to have a large error. Thus, those robust learning algorithms use the so-called robust cost function to discriminate against outliers from the majority by degrading the effects of those points whose estimated errors are large [22], [23]. However, the problem of such approaches is that what are desired or which points are the majorities are unknown, because the estimated errors used for identifying outliers may not be correct. Hence, when the initial weights are not properly selected, the robust cost function used in those algorithms may not correctly discriminate against those outliers. As a result, the learning process will be moving in a wrong direction and the learning performance then may become awful.

The second problem in traditional robust learning networks is that the number of hidden nodes is difficult to determine [25]–[28]. With too many hidden nodes, the approximated function often interpolates all training data including noise leading to overfitting. Consequently, even though training errors can be reduced to a certain extent, generalization error, however, may become worse. On the other hand, with too few hidden nodes, the network is too simple to capture the information bearing in a complicated data set leading to underfitting. As a result, the approximated function cannot truly represent the considered function. Summarily, the selection of the number of hidden nodes is important. However, a suitable number of nodes depends not only on some available information such as the numbers of input and outputs and the number of training patterns but also on some unknown information like the type of noise and the complexity of the function to be approximated. Hence, it is not likely to select a proper number of nodes in advance.

III. SVR and Its Robust Property

A function approximation problem can be formulated as to obtain a function f from a set of observations, $X = \{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \ldots, (\vec{x}_N, y_N)\}$ with $\vec{x}_i \in R^m$ and $y_i \in R$, where N is the number of training data, $\vec{x}_i$ is the ith input vector, and y_i is the desired output for the input $\vec{x}_i$. Based on the SVM theory, SVR is to approximate the given observations in an m-dimensional space by a linear function in another feature space F. The function in SVR is of the form

$$f\left(\vec{x}, \vec{\theta}\right) = \left\langle \vec{\theta}, \Phi(\vec{x}) \right\rangle + b \tag{1}$$

where $\langle \cdot, \cdot \rangle$ is an inner product defined on F, $\Phi(\cdot)$ is a nonlinear mapping function from R^m to F, $\vec{\theta} \in F$ is a weight vector to be identified in the function, and b is a threshold. Usually, the considered cost function is [31], [32]

$$R_{\mathrm{SV}}[f] = R_{\mathrm{emp}}[f] + C \cdot \left\| \vec{\theta} \right\|^2 \tag{2}$$

where $R_{\mathrm{emp}}[f] = (1/N)\sum_{i=1}^{N} L(y_i - f(\vec{x}_i, \vec{\theta})) = (1/N)\sum_{i=1}^{N} L(e_i)$ [29], [30], $L(y - f(\vec{x}, \vec{\theta}))$ is the loss function measuring the error between y and the estimated output $f(\vec{x}, \vec{\theta})$ for a given input $\vec{x}$, and $C > 0$ is a regular constant. The idea of adding the regularization term is to keep the weight vector $\vec{\theta}$ as small as possible in the approximation process. When overfitting phenomena occur, some unwanted information typically noise, has been modeled into the function.

Those unwanted signals usually are not smooth, and as a consequence, some parameters may become large to accommodate such behaviors. Thus, in (2), the cost function has included the intention to minimize $\vec{\theta}$, which in turn, reduces the model capacity. In others words, the regularization term in (2) controls the tradeoff between the model complexity and approximation accuracy in order to ensure good generalization performance [31].

In traditional SVR, the ε-insensitive function is used as the loss function in (2). It was first introduced in the original SV algorithm [2], [8]. The ε-insensitive function is defined as

$$L(e) = \begin{cases} 0, & \text{for } |e| \le \varepsilon \\ |e| - \varepsilon, & \text{otherwise} \end{cases} \tag{3}$$

for a previously chosen nonnegative number ε. The use of this loss function can lead to sparse decompositions [32] and a quadratic programming problem formulated in terms of inner products in F. Note that beside of the linear term used in (3), quadratic forms and infinities are also admissible for leading to the same type of problems [32]. Other loss functions found in the literature [10] are the polynomial loss function and the "piecewise polynomial and linear" loss function. Those loss functions are listed in Table I. In our simulation, those functions are all employed in the SVR algorithm for illustration.

It was shown in [2] that the solution of the above problem can be expressed in terms of support vectors, $\vec{\theta} = \sum_{i=1}^{N} \beta_i \Phi(\vec{x}_i)$ and the function f is then written as

$$f\left(\vec{x}, \vec{\theta}\right) = \sum_{i=1}^{N} \beta_i \left\langle \Phi\left(\vec{x}_i\right), \Phi\left(\vec{x}\right)\right\rangle + b. \tag{4}$$

In (4), the inner product $\langle \Phi(\vec{x}_i), \Phi(\vec{x})\rangle$ in the feature space is usually considered as a kernel function $K(\vec{x}_i, \vec{x})$ [33]. The kernel function determines the smoothness properties of solutions and should reflect a prior knowledge on data. The choice of the kernel function is usually left for users. The kernel function used in our study is Gaussian and defined as

$$K\left(\vec{x}, \vec{x}_i\right) = \exp\left[-\left\|\vec{x} - \vec{x}_i\right\|^2 \Big/ 2\tau^2\right] \tag{5}$$

where τ is a constant. The coefficients β_is in (4) can be solved by quadratic programming methods with suitable transformation of the above problem into constraint optimization problems and properly rearranging the equation into a matrix form [7], [32]. Note that only some of β_is are not zeros and the corresponding vectors $\vec{x}_i$s are called the support vectors. In (4), the constant b is unknown. Various forms can be found in the literature [9]. In our implementation, the following equation is used for b [9]:

$$b = \frac{1}{2}\left\{\min_i \left(y_i - \sum_{i=1}^{N} \beta_i K\left(\vec{x}_i, \vec{x}\right)\right) + \max_i \left(y_i - \sum_{i=1}^{N} \beta_i K\left(\vec{x}_i, \vec{x}\right)\right)\right\}. \tag{6}$$

TABLE I
LOSS FUNCTIONS ARE USED IN THIS PAPER

ε -insensitive loss function	$l(\upsilon) = \begin{cases} 0, for \lvert\upsilon\rvert \le \varepsilon \\ \lvert\upsilon\rvert - \varepsilon, for \lvert\upsilon\rvert > \varepsilon \end{cases}$
Polynomial loss function	$l(\upsilon) = \frac{1}{p}\upsilon^p, \quad p > 1$
Piecewise polynomial and linear loss function	$l(\upsilon) = \begin{cases} \gamma^{1-p}\frac{1}{p}\upsilon^p, \text{for } \upsilon < \gamma \\ \upsilon + \left(\frac{1}{p} - 1\right)\gamma, \text{for } \upsilon \ge \gamma \end{cases}$

According to the SVM theory, SVR has the advantage of self-determining its structure. Hence, there are no initialization problems for SVR. For training data sets with certain noise distributions, SVR also has shown excellent performance under the ε-insensitive function or Huber's robust function [6]–[11]. However, the robust effects against training data sets with outliers are not obvious in SVR.

In the SVR theory, the loss function and the parameters such as the regular constant C in (2), and τ in the kernel function (5), must be determined in advance. The parameter ε in the ε-insensitive function, and the regular constant C are powerful means for regularization and adaptation to the noise in training data [6]. Both control the network complexity and the generalization capability of SVR. However, as stated in [6], how to determining a set of proper parameters is still suboptimal and computationally extensive (if not clumsy). With different loss functions and/or parameter sets $\{\tau, C\}$, the SVR approaches may result in different optimum solutions under the same training data set with or without outliers. In Section V, we shall show that it is not straightforward to select those parameters properly. In [34], several validation approaches have been proposed for verifying the validation of the selected parameter set such as k-fold cross-validation, VC bounds, Xi–Alpha bound, and radius–margin bound. Those approaches still suffer from various problems mentioned in Section I. In this research, instead of developing algorithms to find suitable selections for those parameters, we propose to employ traditional robust learning approaches to improve the learning performance.

IV. RSVR NEURAL NETWORKS

In this paper, the robust learning concept and the SVR theory are combined to form the RSVR networks. The learning of the proposed RSVR network is divided into two phases, the initial phase and the robust learning phase. The initial phase is to determine the network structure and the corresponding initial network weights through the SVR theory. When the cost function in (2) and the kernel functions in (5) are chosen, the initial weights and the structure of RSVR can be determined by the SVR theory as stated in the previous section. In this paper, the kernel function is chosen as Gaussian function. After applying the SVR theory, an initial RSVR network is obtained as

$$\hat{y} = \sum_{i=1}^{P} \beta_i K\left(\vec{x}_i, \vec{x}\right) + b \tag{7}$$

where $\hat{y}$ is the output of the RSVR network, $K(\vec{x}, \vec{x}_i)$ is a kernel function of the SVR theory, P is the number of kernel functions, which is equivalent to the number of support vectors, and $\vec{\beta} = [\beta_1, \ldots, \beta_P]^T$ is the weight vector of the network. Note that (7) can also be regarded as a classical parameterized radial basis function networks with unknown β_i and b.

In the second phase, the algorithm is to adjust those weights via a robust learning approach. The robust learning algorithm is based on the algorithm proposed in [13]. It should be noted that this robust learning algorithm is employed after a period of traditional backpropagation training in [13]. This is because that approach needs a fair initial network before entering the robust learning phase. In our RSVR, SVR in fact is to replace such a procedure. There are also other similar robust learning approaches [14], [17]–[21]. An important feature of those robust learning algorithms is to use a robust cost function in the place of the quadratic form of the cost function in a standard backpropagation algorithm.

In the learning algorithm, a robust cost function $E_R(t)$ is defined as

$$E_R(t) = \frac{1}{N} \sum_{j=1}^{N} \sigma[e_j(t)] \tag{8}$$

where t is the epoch number, $e_j(t) = y_j - \hat{y}_j(t)$ is the error of the RSVR network at epoch t, and $\sigma(\cdot)$ is a robust cost function, which directly stems from the theory of robust statistics [22], [23]. In the RSVR learning algorithms, the gradient descent method is employed and $\vec{\beta}$ can be updated as

$$\vec{\beta}(t+1) = \vec{\beta}(t) + \Delta\vec{\beta}(t) \tag{9}$$

with

$$\Delta\vec{\beta}(t) = \eta \frac{\partial E_R(t)}{\partial \vec{\beta}} = \frac{\eta}{N} \sum_{j=1}^{N} \varphi[e_j(t)] \frac{\partial e_j(t)}{\partial \vec{\beta}} \tag{10}$$

where η is a learning constant, $\varphi[e_j(t)] = \partial\sigma\lfloor e_j(t)\rfloor / \partial e_j(t)$ is usually called the influence function of the algorithm, and $\partial e_j(t)/\partial\vec{\beta} = -K(\vec{x}_l, \vec{x}_j)$. Note that due to the iterative nature of the training procedure, (10) becomes $\Delta\vec{\beta}(t) = \eta\varphi[e_j(t)]\partial e_j(t)/\partial\vec{\beta}$ for each training pattern in the implementation. We concur that the above steepest descent approach is an inferior optimization tool. However, the considered scenario is that training data set contains outliers. Traditional approaches for solving such a problem are to introduce a robust cost function [13], [14], [18], and then, a steepest descent approach is applied. The idea of such an approach is to identify outliers and then to reduce the effects of outliers directly. Other optimization tools that can directly reduce the effects of outliers are hardly found in the literature. In fact, even without the existence of outliers, the steepest descent approach can still work well in various applications of neural networks. Since our approach starts from the model constructed from SVR, which is capable of finding a nice initial network, a steepest descent approach should be enough.

The $\tanh$-estimator [13], [17], [19] is used as the robust cost function in our implementation and is defined as

$$\sigma[e_i(t)] = \begin{cases} \frac{1}{2} e_i^2(t), & 0 \le |e_i(t)| < a(t) \\ \frac{1}{2} a^2(t) + \frac{c_1}{c_2} \ln\left[\frac{\cosh(c_2(b(t) - a(t)))}{\cosh(c_2(b(t) - |e_i(t)|))}\right], & a(t) < |e_i(t)| \le b(t) \\ \frac{1}{2} a^2(t) + \frac{c_1}{c_2} \ln[\cosh(c_2(b(t) - a(t)))], & b(t) < |e_i(t)| \end{cases} \tag{11}$$

where $a(t)$ and $b(t)$ are time-dependent cutoff points, and c_1 and c_2 are constants selected as 1.73 and 0.93, respectively, as those in [19]. The influence function of (13) is obtained as

$$\varphi[e_i(t)] = \begin{cases} e_i(t), & 0 \le |e_i(t)| < a(t) \\ c_1 \tanh[c_2(b(t) - |e_i(t)|)] \operatorname{sign}(e_i(t)), & a(t) < |e_i(t)| b(t) \\ 0, & b(t) < |e_i(t)|. \end{cases} \tag{12}$$

The shape of $\sigma(\cdot)$ depends on the probabilistic distribution of the obtained errors [29] and on the cutoff points $a(t)$ and $b(t)$ in (11). Ideal values of $a(t)$ and $b(t)$ basically depend on outliers. The *CUTOFF* algorithm used in [13] is also adopted in our research. The *CUTOFF* algorithm is stated as follows. Let q be an upper bound of the percentage of outliers in the training data set, and thus $a(t)$ and $b(t)$ can be defined as follows.

Step 1) Compute $e_i(t) = y_i(t) - f(\vec{x}_i, \vec{w})$, $i = 1, 2, \ldots, N$.
Step 2) Sort $e_i(t)$ in an increasing order and define $a(t) = e_{i^*}(t)$ for $i^* = (1-q)N$ and $b(t) = 2a(t)$.

In the RSVR learning algorithm, the following inputs are required:

1) a set of training data $Tr = \{(\vec{x}_i, y_i), i = 1, 2, \ldots, N\}$, $\vec{x}_i \in R^n$, $y_i \in R$;
2) a set of testing data $Te = \{(\vec{x}_k, y_k), k = 1, 2, \ldots, M\}$, $\vec{x}_k \in R^n$, $y_k \in R$;
3) the kernel function $K(\cdot, \cdot)$, such as Gaussian, $\tanh$, or B-spline;
4) the loss function $L(\cdot)$, such as the ε-insensitive function, the quadratic function, or the “piecewise polynomial and linear” loss function;
5) the threshold ε_R used for determining the termination condition of the robust learning algorithm;
6) the maximal percentage q of outliers in the training data set.

Note that the set of testing date Te is used to define the generalization errors.

The RSVR learning algorithm is summarized as follows.

Step 1) Initialize the RSVR structure with the given kernel functions, the loss function, and the constant C.

Step 2) For each training pattern, compute the estimated result $\hat{y}$ by (7) and its error.

Step 3) Update the weight vector $\vec{\beta}$ incrementally by (9) and (10). When it is in the first epoch, the robust cost function is the quadratic form by selecting $a(0)$ and $b(0)$ large enough to count in all points.

Step 4) Determine the cutoff points $a(t)$ and $b(t)$ by the *CUTOFF* algorithm.

Step 5) Compute the robust cost function E_R defined by (8).

Step 6) If the termination conditions are not satisfied, then go to Step 2; otherwise, terminate the learning process.

V. Simulation Results

The simulations were conducted in the Matlab environment. The SVR toolbox provided by Gunn [35] and obtained through the network service is used here. In this study, outliers are added artificially by moving some points away from designated locations. The root mean square error (RMSE) of the testing data is used to measure the performance of the learned network (generalization capability). In this study, the results of various cases with different loss functions (the ε-insensitive function with $\varepsilon = 0$, the ε-insensitive function with $\varepsilon = 0.01$, the quadratic function, and the "piecewise polynomial and linear loss function" with $\gamma = 0.1$), different τs, and different Cs are presented for illustration. The learning constant η and the maximum percentage of outliers q used in the simulation are 0.03 and 0.05, respectively.

In this paper, two functions are considered. The first one is the sinc function and is defined as

$$y = \frac{\sin(x)}{x} \quad \text{with} \quad x \in [-10, 10]. \tag{13}$$

This sinc function is often used in the literature [7]–[11]. Fifty one training patterns in Tr are generated from the function. Among those data, three artificial outliers whose deviation values are 0.5, 0.4, and 0.3, respectively, are created. The other used function has also been used in [14] and [18] and is defined as

$$y = x^{2/3} \quad \text{with} \quad x \in [-2, 2]. \tag{14}$$

The test data sets with 201 patterns are also generated for both examples. Note that the test data do not contain any outliers.

The sinc function is first considered. The testing RMSEs of SVR are obtained for various loss functions with different parameter sets $\{\tau, C\}$. Those errors are tabulated in the upper portion of each entry in Table II(a)–(d). The testing RMSEs of the proposed RSVR after 1000 epochs of learning are tabulated in the lower portion of each entry in Table II(a)–(d). From the results of SVR, by selecting a proper parameter set for $\{\tau, C\}$, the testing RMSE of SVR can reach a nice level. It is also evident that the proposed RSVR can further improve the generalization performance of the original SVR for those cases. The average percentages of the reduced RMSEs obtained by the model constructed from SVR and from RSVR for all situations are summarized and listed in Table III.

TABLE II
(a) Testing RMSEs of SVR (in the Upper Portion) and of RSVR (in the Lower Portion) After 1000 Epochs Training Under the ε-Insensitive Function With $\varepsilon = 0$ for the sinc Function. (b) Testing RMSEs of SVR (in the Upper Portion) and of RSVR (in the Lower Portion) After 1000 Epochs' Training Under the ε-Insensitive Function With $\varepsilon = 0.01$ for the sinc Function. (c) Testing RMSEs of SVR (in the Upper Portion) and of RSVR (in the Lower Portion) After 1000 Epochs' Training Under the Quadratic Function for the sinc Function. (d) Testing RMSEs of SVR (in the Upper Portion) and of RSVR (in the Lower Portion) After 1000 Epochs' Training Under the "Piecewise Polynomial and Linear Loss Function" With $\gamma = 0.1$ for the $sinc$ Function

RMSE	C=0.5	C=1	C=5	C=10	C=50	C=100
$\tau = 0.5$	0.09337	0.07214	0.05189	0.06221	0.09827	0.09827
	0.08890	**0.07010**	**0.05007**	**0.06221**	**0.09827**	**0.09827**
$\tau = 0.7$	0.02971	0.00728	0.00280	0.00196	0.00183	0.00342
	0.01517	**0.00621**	**0.00065**	**0.00163**	**0.00150**	**0.00220**
$\tau = 1$	0.00035	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0
	0.00010	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0
$\tau = 2$	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0
	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0
$\tau = 3$	0.02405	0.00141	0.00016	≈ 0	≈ 0	≈ 0
	0.00890	**0.00109**	≈ 0	≈ 0	≈ 0	≈ 0

Note: The symbol "≈ 0 " indicates that the value is less than 1×10^{-4}

(a)

RMSE	C=0.5	C=1	C=5	C=10	C=50	C=100
$\tau = 0.5$	0.09073	0.07603	0.05017	0.07592	0.09704	0.09704
	0.06015	**0.07355**	**0.05399**	**0.07326**	**0.09692**	**0.09692**
$\tau = 0.7$	0.02725	0.02082	0.01387	0.01376	0.01701	0.01953
	0.02051	**0.01606**	**0.01013**	**0.01127**	**0.01481**	**0.01690**
$\tau = 1$	0.01412	0.01103	0.00974	0.00997	0.00977	0.01008
	0.00216	**0.00216**	**0.00666**	**0.00827**	**0.00952**	**0.00984**
$\tau = 2$	0.00776	0.00772	0.00776	0.00769	0.00779	0.00760
	0.00075	**0.00131**	**0.00196**	**0.00256**	**0.00516**	**0.00655**
$\tau = 3$	0.03274	0.00939	0.00748	0.00761	0.00750	0.00770
	0.00846	**0.00437**	**0.00358**	**0.00362**	**0.00444**	**0.00399**

(b)

RMSE	C=0.5	C=1	C=5	C=10	C=50	C=100
$\tau = 0.5$	0.12789	0.09674	0.07939	0.08122	0.08819	0.09103
	0.02896	**0.04010**	**0.03464**	**0.04594**	**0.07023**	**0.09057**
$\tau = 0.7$	0.10937	0.08365	0.06941	0.07058	0.07597	0.07842
	0.00961	**0.01538**	**0.03810**	**0.02167**	**0.03914**	**0.04621**
$\tau = 1$	0.09498	0.07331	0.06021	0.06048	0.06477	0.06699
	0.00182	**0.00270**	**0.01372**	**0.02133**	**0.04400**	**0.05193**
$\tau = 2$	0.09398	0.07058	0.04800	0.04528	0.04507	0.04654
	0.00168	**0.00122**	**0.00396**	**0.00670**	**0.001535**	**0.01978**
$\tau = 3$	0.14216	0.11062	0.06159	0.05203	0.04372	0.04219
	0.01674	**0.01580**	**0.01268**	**0.01093**	**0.00823**	**0.00790**

(c)

RMSE	C=0.5	C=1	C=5	C=10	C=50	C=100
$\tau = 0.5$	0.08883	0.08129	0.07084	0.09001	0.09579	0.09688
	0.03536	**0.07548**	**0.05074**	**0.07791**	**0.09534**	**0.09664**
$\tau = 0.7$	0.04112	0.02381	0.02582	0.02901	0.03919	0.04545
	0.01874	**0.01475**	**0.01220**	**0.01603**	**0.02688**	**0.03334**
$\tau = 1$	0.01801	0.01515	0.01513	0.01609	0.01842	0.01942
	0.00205	**0.00300**	**0.00638**	**0.00793**	**0.01091**	**0.01207**
$\tau = 2$	0.01506	0.01073	0.00779	0.00784	0.00892	0.00946
	0.00051	**0.00078**	**0.00217**	**0.00301**	**0.00506**	**0.00579**
$\tau = 3$	0.04518	0.02276	0.00986	0.00815	0.00675	0.00658
	0.01218	**0.00804**	**0.00365**	**0.00278**	**0.00202**	**0.00189**

(d)

Next, we would like to point out that SVR have different optimal parameter sets $\{\tau, C\}$ for different loss functions. In the case of using the ε-insensitive function with $\varepsilon = 0.01$ as the loss function [Table II(b)], when $\{C, \tau\} = \{50, 3\}$, it has the best performance among various $\{\tau, C\}$s and the testing RMSE is

TABLE III
AVERAGE PERCENTAGES OF THE REDUCED RMSEs OBTAINED BY THE MODEL CONSTRUCTED FROM SVR AND FROM RSVR FOR THE *sinc* FUNCTION

	ε-insensitive function with $\varepsilon = 0$	ε-insensitive function with $\varepsilon = 0.01$	Quadratic function	"Piecewise polynomial and linear loss function" with $\gamma = 0.1$
$\tau = 0.5$	0.66%	6.60%	45.0%	11.87%
$\tau = 0.7$	41.79%	20.10%	65.10%	40.34%
$\tau = 1$	71.42%	40.33%	67.79%	58.58%
$\tau = 2$	0%	60.51%	86.07%	88.05%
$\tau = 3$	25.46%	60.70%	84.02%	69.22%

The average percentage of the reduced RMSE is defined as:
$\frac{\text{Sum of SVR RMSE - Sum of RSVR RMSE}}{\text{Sum of SVR RMSE}}$, for τ fixed and all C

TABLE IV
AVERAGE PERCENTAGES OF THE REDUCED RMSEs OBTAINED BY THE MODEL CONSTRUCTED FROM SVR AND FROM RSVR FOR THE FUNCTION $y = x^{2/3}$

	ε-insensitive function with $\varepsilon = 0$	ε-insensitive function with $\varepsilon = 0.1$	Quadratic function	"Piecewise polynomial and linear loss function" with $\gamma = 0.1$
$\tau = 0.5$	22.78%	34.46%	49.95%	22.09%
$\tau = 0.7$	14.03%	39.99%	44.28%	10.63%
$\tau = 1$	18.52%	33.61%	30.76%	6.55%
$\tau = 2$	17.52%	17.57%	17.53%	7.76%
$\tau = 3$	16.20%	15.17%	21.52%	12.69%

The average percentage of the reduced RMSE is defined as:
$\frac{\text{Sum of SVR RMSE - Sum of RSVR RMSE}}{\text{Sum of SVR RMSE}}$, for τ fixed and all C

0.007 48. In the case of using the quadratic form [Table II(c)], when $\{C, \tau\} = \{100, 3\}$, the performance is the best, and the testing error is 0.042 19. In the case of using the "piecewise polynomial and linear loss function" with $\gamma = 0.1$ [Table II(d)], when $\{C, \tau\} = \{100, 3\}$, the performance is the best and the testing error is 0.006 58. For those cases, the proposed RSVR still can significantly improve the generalization performance. For the case of using the ε-insensitive function with $\varepsilon = 0$ [Table II(a)], various combinations of $\{C, \tau\}$ all can reach nice performance. It should be noted that the number of support vectors when $\varepsilon = 0$ is equal to the number of training data. For the sinc function, SVR is robust against training data with outliers when using the ε-insensitive function with $\varepsilon = 0$ in a broad range of parameter sets $\{\tau, C\}$. However, it is not always true for other cases.

Now, the function $y = x^{2/3}$ is considered. In this case, various loss functions with different parameters sets $\{\tau, C\}$ similar to those used for the sinc function are also used. Detailed results are shown in [36]. From the simulations, it can found that the parameter sets for the best performance in different loss functions are different from those in the above example. For example, in the case of using the ε-insensitive function with $\varepsilon = 0.1$ as the loss function, when $\{C, \tau\} = \{5, 0.5\}$, the performance is the best and the testing RMSE is 0.074 37. For this case, after robust learning, the RSME becomes 0.048 46. In the case of using the quadratic form, when $\{C, \tau\} = \{50, 0.5\}$, the performance is the best and the testing RMSE is 0.065 45. For this case, after robust learning, the RSME becomes 0.044 91. The average percentages of the reduced RMSEs obtained by the model constructed from SVR and from RSVR for all situations are summarized and listed in Table IV. Again, the proposed RSVR can always improve the generalization performance of SVR. Finally, we should point out that the robust performance among parameters selected in the case of using the ε-insensitive function with $\varepsilon = 0$ shown for the sinc function is no longer true for this example.

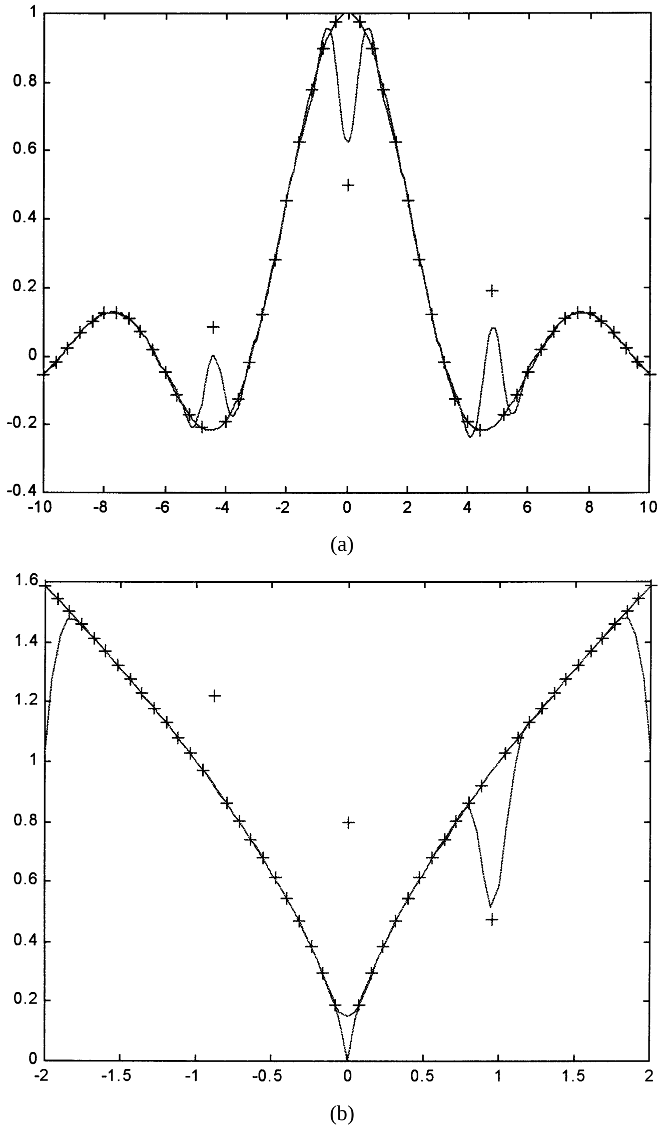
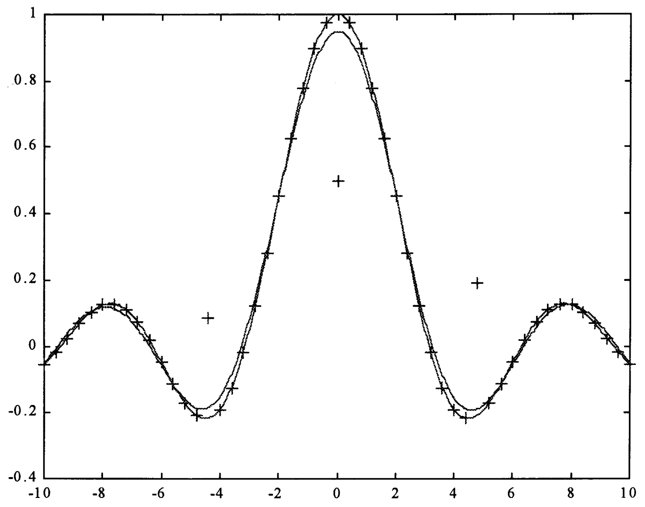


Fig. 1. (a) Obtained result of SVR using the quadratic loss function, $\tau = 0.5$ and $C = 10$ for the sinc function. In this case, all data are support vectors. (b) Obtained result of SVR using the ε-insensitive function with $\varepsilon = 0$, $\tau = 0.1$ and $C = 0.5$ for function $y = x^{2/3}$. In this case, all data are support vectors.

Two learned results are displayed in Fig. 1(a) and (b) for illustration. Fig. 1(a) is the case of using the quadratic loss function and the parameter sets $\{\tau, C\} = \{0.5, 10\}$ for the sinc function. Fig. 1(b) is the case of using the ε-insensitive function with $\varepsilon = 0$ and $\{\tau, C\} = \{0.1, 0.5\}$ for $y = x^{2/3}$. It is noted that since ε is zero in those shown examples, all training data are support vectors. From both figures, the approximated results appear oscillation around outliers. Such phenomena can be interpreted as the overfitting phenomenon. When the parameter sets


(a)

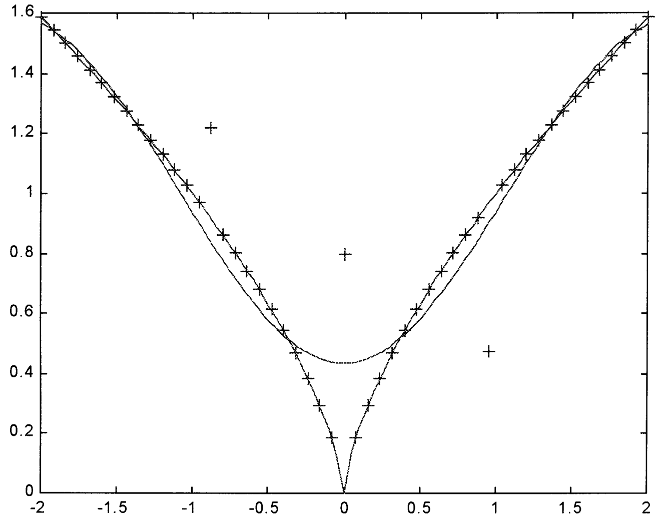
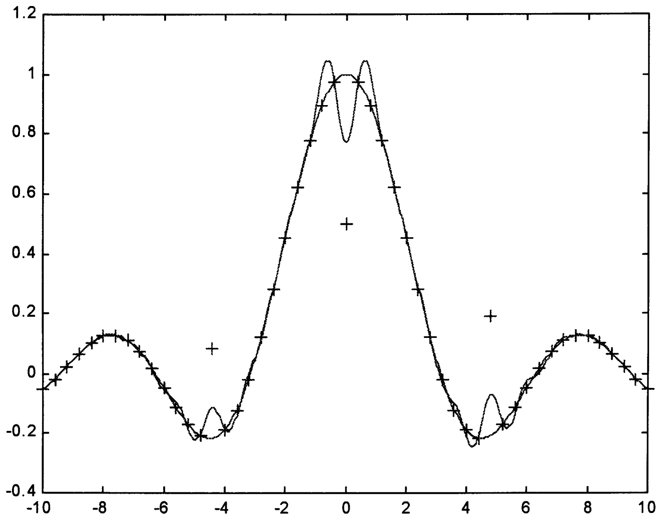

(b)

Fig. 2. (a) Obtained result of SVR using the ε-insensitive function with $\varepsilon = 0$, $\tau = 3$, and $C = 0.5$ for the sinc function. In this case, all data are support vectors. (b) Result of SVR using the ε-insensitive function with $\varepsilon = 0$, $\tau = 2$ and $C = 100$ for function $y = x^{2/3}$. In this case, all data are support vectors.


(a)

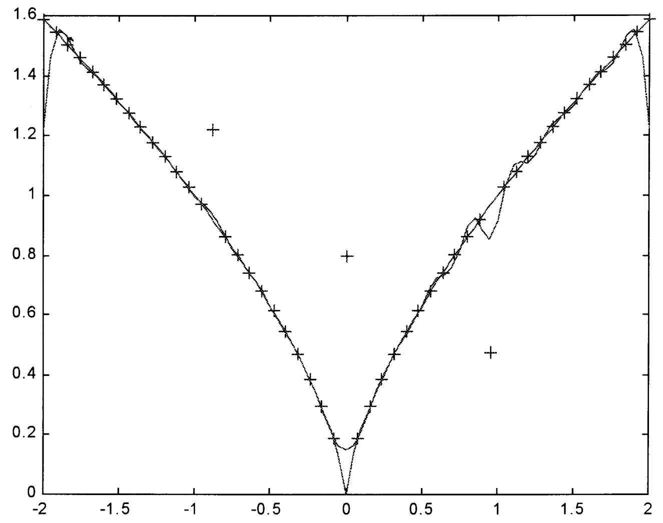

(b)

Fig. 3. (a) Result obtained by RSVR based on Fig. 1(a) for the sinc function. (b) Result obtained by RSVR based on Fig. 1(b) for function $y = x^{2/3}$.

$\{\tau, C\} = \{3, 0.5\}$ for the sinc function and $\{\tau, C\} = \{2, 100\}$ for $y = x^{2/3}$ both with the ε-insensitive function with $\varepsilon = 0$, the approximated results are still affected by outliers and are shown in Fig. 2(a) and (b), respectively. These results match with the concept discussed in [15]. Therefore, it can be concluded that the selection of parameters is not straightforward. The results by RSVR after 1000 epochs based on Figs. 1(a) and (b) and 2(a) and (b) are shown in Figs. 3(a) and (b) and 4(a) and (b), respectively. It can be found that the proposed RSVR can reduce the overfitting phenomena.

VI. Conclusion

In this paper, a novel regression approach (the RSVR network) was proposed to enhance the robust capability of the SVR approaches. The basic idea of the approach is to adopt the concept of traditional robust statistics to fine-tune the function obtained by SVR. In this paper, various loss functions have been used for illustration. From our simulation results, for some cases, the approximated results may oscillate around outliers. Such phenomena can be interpreted as the overfitting phenomenon. From the simulation examples, the selection of τ at the kernel function is more important than others. Different τs may have different performance. However, for different examples, the optimal τs are also different. In those examples, since the desired functions are exactly known, we were able to find which set of parameters can have the best performance. However, while facing with real problems, there are no ways of finding those proper parameters. Hence, we proposed to employ traditional robust learning approaches to improve the learning performance for whatever selected parameters. From the simulation results, it is evident that when improperly initial weights of the network were obtained by SVR, the improving rate of RSVR is significant. As a matter of fact, our RSVR can always improve the performance of the learned systems for all cases. Besides, it can be found that even the training lasted for a long period, the testing errors would not go up. In other word, the overfitting phenomenon is suppressed. Finally, as mentioned earlier, the number of hidden nodes in our approach can

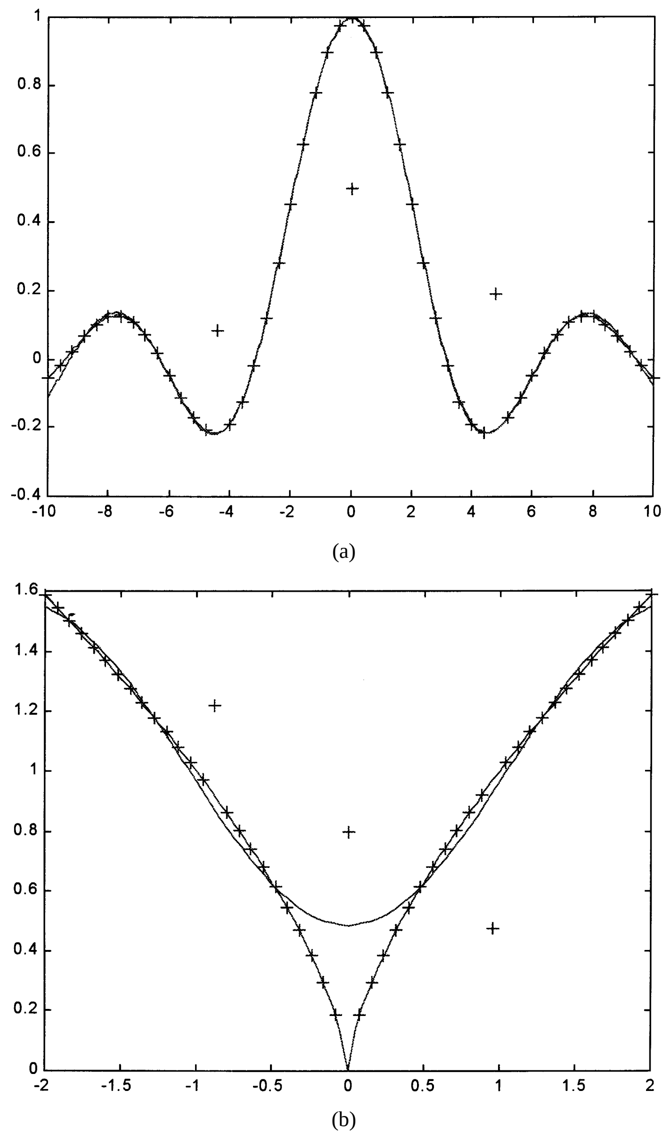


Fig. 4. (a) Result obtained by RSVR based on Fig. 2(a) for sinc function. (b) Result obtained by RSVR based on Fig. 2(b) for function $y = x^{2/3}$.

easily be determined by the SVR theory, instead of subjective selection by heuristics in traditional neural networks.

REFERENCES

[1] C. Cortes and V. Vapnik, "Support vector networks," *Machine Learning*, vol. 20, pp. 273–297, 1995.

[2] V. Vapnik, *The Nature of Statistical Learning Theory*. New York: Springer-Verlag, 1995.

[3] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining Knowledge Discovery*, vol. 2, no. 2, 1996.

[4] B. Schölkopf *et al.*, "Comparing support vector machines with Gaussian kernels to radial basis function classifiers," *IEEE Trans. Signal Processing*, vol. 45, pp. 2758–2765, Nov. 1997.

[5] B. Boser, I. Guyon, and V. Vapnik, "A training algorithm for optimal margin classifiers," presented at the 5th Annu. Workshop Comput. Learning Theory, 1992.

[6] S. Mukherjee, E. Osuna, and F. Girosi, "Nonlinear prediction of chaotic time series using a support vector machine," in *Proc. NNSP*, 1997, pp. 24–26.

[7] H. Drucker *et al.*, "Support vector regression machines," in *Neural Information Processing Systems*. Cambridge, MA: MIT Press, 1997, vol. 9.

[8] V. Vapnik, S. Golowich, and A. J. Smola, "Support vector method for function approximation, regression estimation, and signal processing," in *Neural Information Processing Systems*. Cambridge, MA: MIT Press, 1997, vol. 9.

[9] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," Royal Holloway College, London, U.K., Neuro COLT Tech. Rep. TR-1998-030, 1998.

[10] A. J. Smola, B. Schölkopf, and K. R. Müller, "General cost functions for support vector regression," presented at the ACNN, Australian Congr. Neural Networks, 1998.

[11] A. J. Smola, "Regression estimation with support vector learning machines," Master's thesis, Technical Univ. Munchen, Munich, Germany, 1998.

[12] D. M. Hawkins, *Identification of Outliers*. London, U.K.: Chapman and Hall, 1980.

[13] D. S. Chen and R. C. Jain, "A robust back-propagation learning algorithm for function approximation," *IEEE Trans. Neural Networks*, vol. 5, pp. 467–479, May 1994.

[14] C.-C. Chuang, S.-F. Su, and C.-C. Hsiao, "The annealing robust backpropagation (ARBP) learning algorithm," *IEEE Trans. Neural Networks*, vol. 11, pp. 1067–1077, Sept. 2000.

[15] F. E. H. Tay and L. Cao, "Application of support vector machines in financial time series forecasting," in *Int. J. Manage. Sci.*, 2001, pp. 309–317.

[16] J. A. K. Suykens, J. De Brabanter, L. Lukas, and J. Vandewalle, "Weighted least squares support vector machines: Robustness and sparse approximation," Neurocomput., 2001, to be published.

[17] J. T. Connor, R. D. Martin, and L. E. Atlas, "Recurrent neural networks and robust time series prediction," *IEEE Trans. Neural Networks*, vol. 5, pp. 240–254, Mar. 1994.

[18] K. Liano, "Robust error measure for supervised neural network learning with outliers," *IEEE Trans. Neural Networks*, vol. 7, pp. 246–250, Jan. 1996.

[19] V. David and A. Sanchez, "Robustization of learning method for RBF networks," *Neurocomput.*, vol. 9, pp. 85–94, 1995.

[20] C. Wang, H. C. Wu, and J. C. Principe, "A cost function for robust estimation of PCA," *Proc. SPIE*, vol. 2760, pp. 120–127, 1996.

[21] L. Huang, B. L. Zhang, and Q. Huang, "Robust interval regression analysis using neural network," *Fuzzy Sets Syst.*, pp. 337–347, 1998.

[22] P. J. Rousseeuw and M. A. Leroy, *Robust Regression and Outlier Detection*. New York: Wiley, 1987.

[23] P. J. Huber, *Robust Statistics*. New York: Wiley, 1981.

[24] C. GriffinM. Kendall and A. Stuart, *The Advanced Theory of Statistics*, A. Stuart, Ed., 1977.

[25] P. L. Bartlett, "For valid generalization, the size of the weights is more important than the size of the network," in *Advances in Neural Information Processing Systems*, —AUTHOR, PLEASE LIST FIRST INITIALS— Mozer, Jordan, and Petshe, Eds. Cambridge, MA: MIT Press, 1997, vol. 9, pp. 134–140.

[26] S. Lawrence, C. L. Giles, and A. C. Tsoi. (1996) What size neural network gives optimal generalization? Convergence properties of backpropagation. Univ. Maryland, College Park. [Online]UMIACS-TR-96-22, Tech. Rep.. Available: http://www.neci.nj.com/homepages/lawrence/papers/minima-tr96/minima-tr96.html.

[27] M. Smith, *Neural Networks for Statistical Modeling*. New York: Van Nostrand Reinhold, 1993.

[28] W. S. Sarle, "Stopped training and other remedies for overfitting," in *Proc. 27th Symp. Interface Comput. Sci. Statist.*, 1995, pp. 352–360.

[29] V. Vapnik, *Estimation of Dependences Based on Empirical Data*. New York: Springer-Verlag, 1982.

[30] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley, 1973.

[31] A. J. Smola and B. Schölkopf, "From regularization operators to support vector kernels," in *Neural Information Processing Systems*. Cambridge, MA: MIT Press, 1997, vol. 9.

[32] ——, (1988) On a kernel-based method for pattern recognition, regression, approximation and operator inversion. *Algorithmica* [Online]. Available: http://svm.first.gmd.de/papers.html.

[33] M. A. Aizerman, E. M. Braverman, and L. I. Rozonoér, "Theoretical foundations of the potential function method in pattern recognition learning," *Automat. Remote Contr.*, pp. 821–837, 1964.

[34] K. Duan, S. S. Keerthi, and A. N. Poo, "Evaluation of simple performance measures for tuning SVM hyperparameters," Nat. Univ. Singapore, Tech. Rep. CD-01-11, 2001.

[35] S. R. Gunn. (1999) Support vector regression-Matlab toolbox. Univ. Southampton, Southampton, U.K.. [Online]. Available: http://kernel-machines.org.

[36] C.-C. Chuang, "Robust modeling for function approximation under outliers," Ph.D. dissertation, Dept. Electr. Eng., Nat. Taiwan Univ. Sci. Technol., Taipei, 2000.

Chen-Chia Chuang received the B.S., M.S., and Ph.D. degrees in electrical engineering from the National Taiwan Institute of Technology, Taipei, Taiwan, R.O.C., in 1991, 1993, and 2000, respectively.

He is currently an Assistant Professor in the Department of Electrical Engineering, Hwa-Hsia College of Technology and Commerce, Taipei. His current research interests are neural networks, statistics learning theory, robust learning algorithms, and signal processing.

Shun-Feng Su (S'89–M'91) received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, R.O.C., in 1983 and the M.S. and Ph.D. degrees in electrical engineering from Purdue University, West Lafayette, IN, in 1989 and 1991, respectively.

He is currently a Professor in the Department of Electrical Engineering, National Taiwan University of Science and Technology, Taipei. His current research interests include neural networks, fuzzy modeling, machine learning, intelligent control, virtual reality simulation, data mining, and bioinformatics.

Jin Tsong Jeng (S'95–M'97) was born in Taiwan, R.O.C., in 1967. He received the B.S.E.E., M.S.E.E., and Ph.D. degrees all in electrical engineering from the National University of Science and Technology, Taipei, Taiwan, in 1991, 1993, and 1997, respectively.

He is currently an Associate Professor in the Department of Computer Science and Information Engineering, National Huwei Institute of Technology, Huwei Jen, Yunlin, Taiwan. His primary research interests include neural networks, fuzzy systems, intelligent control, support vector regression, magnetic bearing systems, magnetic materials, nonholonomic control systems, and device drivers.

Chih-Ching Hsiao received the B.S. and M.S. degrees from the Department of Electrical Engineering, National Taiwan Institute of Technology, Taipei, Taiwan, R.O.C., in 1991 and 1993, respectively. He is currently pursuing the Ph.D. degree at the National Taiwan University of Science and Technology, Taipei.

His current research interests include intelligent control and fuzzy systems.