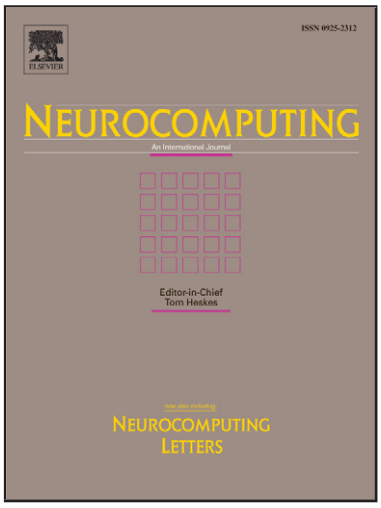


www.elsevier.com/locate/neucom

γ-C plane and robustness in static reservoir for nonlinear regression estimation

Zhiwei Shi, Min Han

PII: S0925-2312(08)00409-8
DOI: doi:10.1016/j.neucom.2008.08.002
Reference: NEUCOM 11231

To appear in: *Neurocomputing*

Received date: 4 March 2008
Revised date: 15 June 2008
Accepted date: 17 August 2008

Cite this article as: Zhiwei Shi and Min Han, γ-C plane and robustness in static reservoir for nonlinear regression estimation, *Neurocomputing* (2008), doi:10.1016/j.neucom.2008.08.002

γ-C Plane and Robustness in Static Reservoir for Nonlinear Regression Estimation

Zhiwei Shi and Min Han
School of Electronic and Information Engineering, Dalian University of Technology, Liaoning 116023, China (E-mail: minhan@dlut.edu.cn).

Abstract: Reservoir method is applied to the feed-forward learning machines for nonlinear regression estimation. Inspired by the existing experience from extreme learning machine (ELM), the new method inherits the basic idea from support vector echo-state machines, but eliminates the internal feedback matrix to adapt for the feed-forward usage. Based on the analysis of nonlinearity in reservoir and regularization in readout weights, the parameters of input scaling and penalty regularization are taken as the hyper-parameters to characterize a static reservoir (ELM); and then a proper reservoir is identified on the γ-C plane based on a generalization error criterion. For outlier suppression, the regularized robust regression is applied in the reservoir feature space, and it leads to an efficient algorithm for large-scale problems, which can be solved by Cholesky decomposition. The proposed method is compared with the classical kernel method and ELM method on several benchmark nonlinear regression datasets, and the results indicate the method is comparable with the existing methods.

Keywords: Reservoir method, Extreme Learning machine, Feed-forward Neural Networks, Support Vector Machines, Kernel Method

1. Introduction

Recently, Echo state networks (ESN)[1-3] and extreme learning machine (ELM)[4-8] are independently investigated but related closely with each other. ESN is a large scale, randomly generated recurrent neural networks and it has many features in a biological brains. ESN is mainly used for temporal information processing. At the same time, ELM is a large scale feed-forward neural networks, which originates from the single-hidden layer feed-forward networks (SLFNs) with randomly generated hidden nodes. In ESN and ELM, the only trainable part is the readout weights, which can be determined by a simple linear regression algorithm. It is worthwhile to investigate the common mysteriousness shared between ELM and its recurrent counterpart.

Theoretical analysis for ESN can be found in [3]. The necessary and sufficient condition to generate echo states is based on the information of the dynamic reservoir, such as the spectral radius of the internal matrix. ESN shares the name of reservoir computing with other similar learning methods, for example, the liquid state machine (LSM)[13], Backpropagation Decorrelation[14, 15] etc. ESN now finds wide applications, such as signal processing[16], nonlinear system identification[3], time series prediction[1, 17]. Recent reviews can be found in [18] and [19]. Rigorous proof has been shown that the input weights and hidden layer biases of SLFN can be randomly assigned if the hidden layer activation functions are infinitely differentiable[4], piecewise continuous[5], nondifferentiable and noncontinuous[6]. ELM method now has been successfully applied to protein secondary structure prediction [9], terrain reconstruction [10], cancer diagnosis [11] and channel equalizer [12] etc.

Support vector machines (SVMs) has a long research history [20-22], which can be used in regression estimation [23, 24] as well as in classification, and it has become one of the classic pattern recognition tools in machine learning. Reservoir method is a new machine-learning tool, which has been shown related closely to SVMs [2, 17, 25]. The combination of SVMs and reservoir may lead to the following two topics of research.

The first topic is to improve the learning performance of recurrent neural networks by using the theory and

experience of SVMs [17, 25]. The main benefit is to obtain a more meaningful training result for reservoir. For example, one can use regularization method to control large weight and improve generalization ability.

The second topic is to investigate the reservoir method as an alternative to the kernel method in the classic SVMs. In fact, the roles of kernel and static reservoir are similar. It is assumed that the input data **u** is mapped to the feature space via a nonlinear mapping $\Phi(\cdot)$, and the algorithm only needs the inner products between the nonlinear mappings. Kernel method allows the direct computation of $\Phi^T(\mathbf{u})\Phi(\mathbf{u}')$ via a nonlinear kernel $K(\mathbf{u}, \mathbf{u}')$, in fact, when a kernel function is used, the mapping $\Phi(\cdot)$ may be "ugly" and usually difficult to guess. By contrast, reservoir method allows the direct creation of the nonlinear mapping $\Phi(\cdot)$ by the mechanisms of echo state property or extreme learning machine. Therefore, the mapping in kernel method is implicit, while the mapping in reservoir method is explicit one, and they have the same role, which translates the nonlinear structure into a linear one.

Up to now, the second topic is not well explored. The study of extreme learning machine offers much experience for static pattern recognition problems. Since the hidden-layer is fixed, it is nice to develop incremental learning methods for the ELM, such as in [5, 7, 8], and it has been witnessed that lots of problems have been efficiently dealt with by ELM. However, there are two problems in static reservoir or extreme learning machines for feed-forward applications. The first problem is how to construct a static reservoir (a term to denote the ELM in this paper) efficiently for a better prediction performance. For different problems, the key problem is how to determine the reservoir size (number of hidden layer neurons), activation function, style of generating the input weights and the magnitude of hidden node bias. It has been shown that the classic ELM is still less stable and accurate than the classic SVMs [4]. The second problem is how to train the readout weights under different loss functions and for large-scale problems efficiently. In fact, if it involves linear SVMs, the popular sequential minimal optimization (SMO) method may not be the best choice [26].

In this paper, we will concentrate the work on the second topic of combing reservoir and support vector machines. Attention will be focused on how to characterize, select and train a static reservoir (ELM) for a given regression estimation problem. The method is called feed-forward support vector echo-state machines (FF-SVESMs). First, inspired by the experience of model selection in the classic kernel method and existing tips to create a reservoir, we take the scaling parameter γ and regularization parameter C as the hyper-parameters to characterize a reservoir. Then a proper reservoir is identified on the γ-C plane based on a generalization performance measure, for example the cross-validation error. Finally, an efficient algorithm is applied to minimize the cost, which can deal with large-scale training examples under robust loss functions.

This paper is organized as follows. Section 2 will present the main results, which starts from the basics of the static reservoir, analysis and determination of the two hyper-parameters, and training of output weights for different loss functions and large-scale problems; Section 3 is the simulation results. The γ-C plane of the static reservoir will be shown firstly, and the γ-C plane in kernel method will also be given. The FF-SVESMs will be compared with the classic ε-insensitive support vector regression and classic ELM in both prediction error and computation time, and the robustness against outliers is demonstrated in section 3. In section 4, conclusions will be given.

2. Nonlinear Regression Estimation by Reservoir Method

2.1 The basics of static reservoir for regression estimation

The network equation of echo state networks can be written as:

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{sig}(W_x\mathbf{x}(k) + W_{in}\mathbf{u}(k)+\mathbf{b}) \\ y(k) &= \mathbf{w}^T\mathbf{x}(k) + b_o \end{aligned} \tag{1}$$

where $\mathbf{x}(k)$, $\mathbf{u}(k)$ and $y(k)$ are the reservoir internal state, input and output variables at time k; $\mathbf{sig}(\cdot)$ denotes the

sigmoid function which applied elementwise, W_x is the internal weight matrix, W_{in} is the input weight matrix, **b** is the hidden layer bias vector. These connections and bias are generated at random so that the reservoir exhibits some special characteristics and decodes the nonlinear dynamics well. The output weights **w** and output bias b_o can be determined by a simple linear regression, for examples, by pseudoinverse or regularization learning.

The network equation of extreme learning machine can be written as:

$$y = \mathbf{w}^T \mathbf{sig}(W_{in}\mathbf{u}+\mathbf{b}) + b_o \tag{2}$$

The input weight matrix W_{in} and bias vector **b** are also generated at random, which is similar with ESN except for the missing of internal feedback matrix W_x. The simulation results in [4] use a relatively large number of hidden layer neurons than the classic MLPs, which is similar with ESN when comparing with the classic recurrent neural networks. It is obvious that the ELM is the feed-forward counterpart of ESN, and ESN is the recurrent counterpart of ELM.

These two networks can be well bridged. When the spectral radius of W_x is set to zero, the ESN will share the same network equation as the ELM. A static reservoir may have wide application fields, since many problems can be taken as a static mapping problem. In this paper, we will focus on the study of static reservoir for nonlinear regression estimation problems.

2.2 Input scaling parameter γ

Classic MLP models use adaptive basis functions with sigmoidal nonlinearities, which can adapt the parameters so that the regions of input space over which the basis functions vary corresponds to the data manifold [27]. A MLP model with one hidden layer can be written as:

$$y = \mathbf{w}^T \mathbf{sig}(\sum_{j}^{p}\sum_{i}^{N} \gamma_{ij}^{w} \cdot W_{in}^{ij} u_j + \sum_{i}^{N} \gamma_i^b \cdot b^i) + b_o$$

Where p is the input dimension, and u_j is the jth input; N is the number of hidden layer neurons; the input weight connections and bias are initialized to W_{in}^{ij} and b^i (i=1, 2, …, N, j=1, 2, …, p), respectively. γ_{ij}^{w} is the local scaling parameter for the input weight W_{in}^{ij}, and γ_i^b is the local scaling parameter for the bias b^i. The training of MLPs is equivalent to the determination of each scaling parameters. The dimension of the scaling parameter matrix is $N\times(p+1)$, and the gradient based algorithm can be applied to train the MLPs.

By reservoir method, it is unnecessary to determine exactly the scaling parameter for each input connection weight and bias, and only a rough global scaling parameter is needed. Two basic measures have been suggested [28]: (a) Use an extra bias input; (b) Shift a scale input. These methods can "shift many internal units towards one of the extremer outer ranges of their sigmoids", and "be advisable" when modeling "strong nonlinear behavior". These measures are applicable to a static reservoir.

In this paper, standard form of static reservoir is written as:

$$y = \mathbf{w}^T \mathbf{sig}(\gamma W_{in}\mathbf{u}) + b_o \tag{3}$$

where the **u** is the net input, bias vector **b** has been included in the input weight matrix W_{in} for conveniences, at the same time, a scalar "1" is absorbed into the vector **u** to make the bias vector implicit, that is, $\mathbf{u}=[\mathbf{u}^T\ 1]^T$. The input weight matrix W_{in} is generated to a special interval, for example, uniform over [–1, 1]. γ is the parameter to control the activation potential of sigmoidal nonlinearities. And it is called the global input scaling parameter. In fact, the scaling parameter not only scales the input but also exert a scalable bias vector to the activation potential, which can be shown in Fig.1.

Activation matrix A and target output vector $\mathbf{y}_d$ are defined as follows:

$$A = [\mathbf{a}_1 \quad \mathbf{a}_2 \quad \ldots \quad \mathbf{a}_i \quad \ldots \quad \mathbf{a}_l]^T$$
$$\mathbf{y}_d = [y_{d1} \quad y_{d2} \quad \cdots \quad y_{di} \quad \cdots \quad y_{dl}]^T \tag{4}$$

where $\mathbf{a}_i = \mathbf{sig}(\gamma W_{in}\mathbf{u}_i)$, y_{di} and $\mathbf{u}_i$ are respectively the target output and input vector of the ith training example, $0 \leqslant i \leqslant l$, l is the length of training examples.

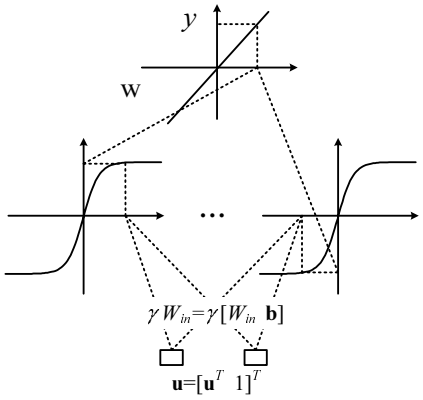


Fig. 1 Static reservoir

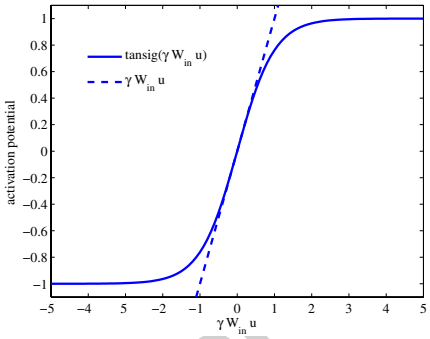


Fig. 2 Hyperbolic tangent sigmoid function

If the input scaling parameter γ is small, each vector of $\gamma W_{in}\mathbf{u}$ will lie in a small interval, for example in [–1, +1]. The neural networks model will work in a linear region as shown in Fig.2 (hyperbolic tangent sigmoid function), and in fact it is an approximate linear regression or weak nonlinear regression. On the contrary, when γ is large, the neural network model will indicate strong nonlinearity and obtain a good fit for nonlinear data set.

It is obvious that the classic MLPs and the static reservoir share the same network equation. Bases on the above analysis, the relationships between them can be summarized in Table 1.

The hyper-parameter γ is global for each weight and bias in W_{in}, and it can be local for each hidden neuron in a static reservoir. This idea is much like the generalized kernel regression model discussed in [29], in which the kernel model has an individually tuned diagonal covariance matrix. Similarly, a generalized reservoir regression model can be written as:

$$y = \mathbf{w}^T \mathbf{sig}(\sum_{j}^{N} \gamma_j W_{in}^j \mathbf{u}) + b_o$$

Where γ_j is the jth local scaling parameter, and $W_{in}^j\mathbf{u}$ is the weighted sum that will be fed to the jth neuron in reservoir. However, the procedures to determine each γ_j may be complex, and in this paper, we will simply consider the global scaling parameter γ as a hyper-parameter. The single scaling parameter γ is also beneficial for hardware realization, since the matrix W_{in} does not change after it is initialized.

Table 1 Comparison between classic MLPs and static reservoir

	classic MLPs	static reservoir
Fundamentals	Adaptive learning comes before the weight initialization	Random weight initialization comes before the adaptive learning
Advantages	Enjoy the small network structure	Enjoy the easy and fast training style
Disadvantages	Local minimum exists; Difficulty in training γ_{ij}	Large memory to store the hidden node activation; Hyper-parameter γ

2.3 Regularization in reservoir

In the previous section, the global scaling parameter γ is taken as a hyper-parameter, and in this section, another hyper-parameter will be chosen.

The first problem will focus on the reservoir size N. The number of hidden layer neurons in ELM has been paid much attention, and Huang gives the suggestion for some benchmark datasets [4]: For common applications, ELM needs a larger reservoir size, which varies from 10 to 190; comparatively speaking, the classical MLPs use smaller network size, which varies from 5 to 45.

If the reservoir size N can be roughly chosen without performance deterioration, the burden will be released for this hyper-parameter tuning. Human brain is complex, but it can deal with both simple and complex task based on a self-adaptive mechanism. Relatively large reservoir is preferred if the complexity can be adjusted based on some treatments to output weights. These treatments are also beneficial for the hardware realization, since reservoir can be built large enough for general purpose.

In general, a relatively large reservoir will lead to severe ill-posed problem, and regularization method for output weight training is needed so that the modeling performance does not deteriorate. Two commonly used methods are truncated singular value decomposition (TSVD) [30] and Tikhonov-type regularization. The second hyper-parameter will be one of the two: (a) the truncation length h in TSVD; (b) the regularization parameter C in Tikhonov-type regularization.

Table 2 Comparison between two regularization methods for reservoir output weight learning

	TSVD	Tikhonov-type regularization
Implement	SVD(A) and discard small singular values	Cholesky(A^TA+I/C)
Advantages	Singular value information is available	Lower computation load
Disadvantages	Need to determine the truncation length h; Larger computation load	Need to determine the regularization parameter C

The TSVD method is implemented by discarding small singular values of matrix A, while the Tikhonove-type regularization is implemented by penalizing larger output weights. Previous ESNs and ELM take pseudoinverse method as a training algorithm for output weight training, which is a special case of TSVD[1]. As shown in Table 2, the advantage of TSVD is that one can access the singular values of the matrix A.

We prefer the Tikhonov-type regularization, because it involves a Cholesky decomposition of matrix $A^T A + I / C \in \mathrm{R}^{N\times N}$, which is very efficient even if the number of training examples is huge. The solution can be solved as

$$\begin{aligned} &A^T A + I / C = L^T L \\ &L^T \mathbf{z} = A^T \mathbf{y}_d, L^T \mathbf{w} = \mathbf{z} \end{aligned} \tag{5}$$

where $\mathbf{z}$ is the intermediate variable, L is the lower triangular matrix in Cholesky decomposition.

In the training algorithm above, the computation load consists of two main parts: the computation of the symmetric matrix A^TA and solution to the linear equation. The computation of the matrix A^TA involves a large number of floating operations, with a time complexity $O(lN^2)$, and the Cholesky decomposition has a time complexity $O(N^3/6)$. The computation of matrix A^TA will dominate the training time if the training example size $l >> N/6$, and this feature is favorable for large-scale problems since the training time is linear with the l.

Compared with the traditional ELM method, the proposed method is computational efficient. The present

[1]In matlab, the default tolerance for singular value truncation is max(size(A))×norm(A)× eps(class(A))

batch learning mode of ELM is based on singular value decomposition, and it has a time complexities of $O(14lN^2+8N^3)$ (GR-SVD) or $O(6lN^2+20N^3)$ (R-SVD), which is very high for large-scale problems. In addition, the classic ELM method has to save the activation matrix $A \in R^{l\times N}$, which needs large memory resource for large l (for example, about 762 megabyte physical memory is needed for l=10^5 and N=10^3). For the proposed method, the whole activation matrix A is not necessary to keep in memory, since the objective is to compute matrix A^TA, which needs less than 10 megabyte in the above example.

Adding state noise is another method to use regularization [28, 31, 32], and it can reduce the condition number of matrix A, but it may perturb the solution, which is not as stable as the standard Tikhonov-type regularization.

In summary, two hyper-parameters have been suggested: the global scaling parameter γ and the regularization parameter C. In the next section, we will discuss the model section and static reservoir construction.

2.4 "γ-C" plane and practical method for reservoir construction

"For valid generalization, the size of the weights is more important than the size of the network", we begin the discussion of model selection by citing Peter L. Bartlett's work on neural network's learning [33]. The results indicate that if a large neural network is used for a pattern recognition problem, and the learning algorithm finds a network with small squared error on the training patterns, the generalization ability depends on the size of the weights rather than the number of weights.

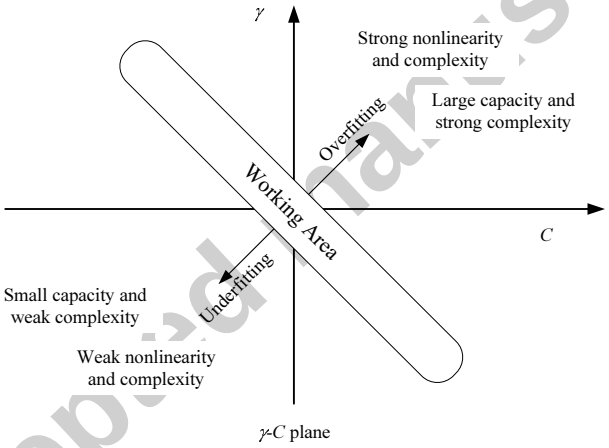


Fig.3 γ-C plane in constructing a reservoir

However, the reservoir method favors relatively large network size, which seems inconsistent with the above results. In fact, the magnitudes of the weights in reservoir can be controlled in the above-mentioned γ-C plane. On the one hand, the weight connection W_{in} is initialized and fixed, and the actual input weight connection is γW_{in}, so the magnitudes of input weights are fully dependent on the scaling parameter γ; on the other hand, the magnitudes of output weights are determined by the regularization parameter C given the activation matrix A and target vector $\mathbf{y}_d$. As a result, on the γ-C plane, small γ and C indicate weak nonlinearity and small capacity, which lead to underfitting; on the contrary, large γ and large C result in strong complexity and overfitting. The approximate working area for good performance is indicated in Fig.3. Note that the two parameters are greater than zero, and the origin only appears in logarithmic coordinates.

In practice, finding such a working area will involve two basic steps: (a) define a generalization error measure on the γ-C plane; (b) follow a procedure to search a good γ-C pair.

As for the generalization error measure, one can resort to many existing results in neural network or kernel methods, for example, the Allen's PRESS statistic [34]:

$$\text{PRESS} = \sum_{i=1}^{l} e_{(i)}^2$$

Where $e_{(i)}$ is the residual for the ith training example in leave-one-out cross-validation. In fact, these residuals are need not evaluated for each trail and can be simply written as:

$$e_{(i)} = \frac{e_i}{1 - h_{ii}}$$

Where e_i *is* the residual for the ith training pattern for a reservoir trained on the entire dataset, and h_{ii} is the ith element of the leading diagonal of hat matrix:

$$A(A^T A + I / C)^{-1} A^T$$

Note that the $A^T A + I / C \in \mathrm{R}^{N \times N}$, where N is the reservoir size and the inverse can be efficiently computed without the reliance on the training example size l. In the study of sparse kernel machines, the leave-one-out error can also be efficiently computed since it only involves a sub-matrix from the whole kernel matrix [35].

To find a good pair of γ and C parameter, one can use many existing methods, such as the grid search method or Nelder and Mead's downhill simplex method. As for the grid search, the process can be easily parallelized, for example using OpenMP or message passing interface (MPI) [36]. In the simulation, we will show how the downhill simplex is used to find a good reservoir model.

2.5 Loss functions in static reservoir method

The structural risk minimization principle suggests a tradeoff between the quality of the approximation and the complexity of the approximating function. The objective function can be written as:

$$C \sum_{i=1}^{l} L(\mathbf{u}_i, y_{di}, f) + \|\mathbf{w}\|^2$$

where L is a general loss function and f is the prediction function defined by:

$$f(\mathbf{w}, b_o, \mathbf{u}) = \mathbf{w}^T \mathbf{sig}(\gamma W_{in} \mathbf{u}) + b_o$$

There are many choices for loss function L. For quadratic loss, the problem can be efficiently solved by Cholesky decomposition with low time complexity, which has been discussed in section 2.3. However, the quadratic loss is super-linear and very sensitive to outliers in training examples, and it is necessary to use other robust loss functions [37, 38].

In the previous results of SVESMs [17], two types of robust loss functions have been applied: ε-insensitive and Huber loss. Huber loss is defined as:

$$L_{huber}(\mathbf{u}_i, y_{di}, f) = \begin{cases} \frac{1}{2}(f(\mathbf{w}, b_o, \mathbf{u}_i) - y_{di})^2, & |f(\mathbf{w}, b_o, \mathbf{u}_i) - y_{di}| < \mu \\ \mu |f(\mathbf{w}, b_o, \mathbf{u}_i) - y_{di}| - \frac{\mu^2}{2}, & \text{otherwise} \end{cases} \quad (6)$$

where μ is the parameter in the Huber loss function. The related optimization problem can be formulated as the following convex quadratic programming (QP):

$$\min_{\beta} \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\beta_i\beta_j \mathbf{sig}(\gamma W_{in}\mathbf{u}_i)^T \mathbf{sig}(\gamma W_{in}\mathbf{u}_j) - \sum_{j=1}^{l}\beta_i y_d(i) + \frac{1}{2C}\sum_{j=1}^{l}\beta_i^2\mu$$

$$s.t. \qquad (7)$$

$$0 \le \beta_i \le C, \quad i = 1, \cdots, l$$

$$\sum_{j=1}^{l}\beta_i = 0$$

Where β_i (i=1, …, l) is the variables to be determined.

The problem in Eq.(7) can be generally solved by a QP optimizer. However, for large-scale problems, QP optimizer is not a good choice. SMO algorithm is a decomposition algorithm for SVMs, which is very efficient for large-scale problems [39]. At present, it is a reasonable choice for training the ε-insensitive loss induced optimization problem. However, it has been shown that the decomposition algorithm is not a good idea for linear SVMs especially for large regularization parameter C [26].

In classic kernel method, the $\mathbf{sig}(\gamma W_{in}\mathbf{u}_i)^T \mathbf{sig}(\gamma W_{in}\mathbf{u}_j)$ will be substituted by a kernel function: $k(\mathbf{u}_i, \mathbf{u}_j)$, which is generally called the "kernel trick". However, in the reservoir method, the explicit mapping $\Phi(\mathbf{u})=\mathbf{sig}(\gamma W_{in}\mathbf{u})$ can be computed directly without resort to a certain kernel function. If the mapping is linear: $\Phi(\mathbf{u})= \mathbf{u}$, the FF-SVESMs is the same as the classic SVMs with linear kernel.

Since the static mapping $\Phi(\mathbf{u})(=\mathbf{sig}(\gamma W_{in}\mathbf{u}))$ is explicit, the weight vector $\mathbf{w}$ and bias b_o can be explicitly trained instead of the variables β_i (i=1, …, l). In this paper, the iteratively re-weighted least squares (IRWLS) algorithm is used to minimize the above cost function.

In the IRWLS algorithm, the optimization is solved by iteration. In each iteration, a weighted linear equation is solved:

$$(A^T\Phi(\mathbf{e})A + I/C)\mathbf{w} = A^T\Phi(\mathbf{e})\mathbf{y}_d \qquad (8)$$

where $\Phi(\mathbf{e}) = \mathbf{diag}\{\phi(e_1), \ \phi(e_2), \ ..., \ \phi(e_l)\}$ is the weight matrix, and the element in the weight matrix is defined as:

$$\phi(e_i) = \frac{\mathrm{d}L_{huber}(e_i)}{e_i} \qquad (9)$$

where e_i is the prediction error obtained by the previous update of $\mathbf{w}$ and b_o. The function $\mathrm{d}L_{huber}(e)$ is defined as follows:

$$\mathrm{d}L_{huber}(e) = \begin{cases} e, & |e| < \mu \\ \mu\,\mathrm{sgn}(e), & \text{otherwise} \end{cases} \qquad (10)$$

The algorithm starts from the solution of ridge regression, and terminates when the estimated coefficients converge. It is similar to the standard robust regression except for its form of regularization.

In section 2.4, the leave-one-out cross-validation error has been used as a performance measure. For Huber loss function, an approximate leave-one-out error can be used if the weight matrix is assumed unchanged by the deletion of a single training example at each iteration [40].

2.5 A bridge from feed-forward neural networks to support vector machines

Based on the above analysis, it is concluded that the FF-SVESMs model stays between traditional feed-forward neural networks and support vector machines. To demonstrate this, comparisons between the FF-SVESMs and the some existing methods are given in Table 3. The comparisons focus on the nonlinearity treatment, complexity control, loss function and training algorithm.

For nonlinearity treatment, the classic MLPs and SVMs use an adaptive basis function and kernel function,

while the ELM and FF-SVESMs are based on a static reservoir and stay in the same side.

Table 3 Comparisons between FF-SVESMs and existing supervised learning machines

Model	Nonlinearity treatment	Complexity control	Loss function	Training Algorithms
MLPs	Adaptive basis functions	Parsimonious in number or by penalty	Quadratic (usually)	Gradient descent
ELM	Randomly generated basis, roughly adjusted	Parsimonious in number	Quadratic	Pseudoinverse (by SVD), etc
FF-SVESMs		Parsimonious by penalty	Quadratic and robust	Determine γ-C pair + Cholesky decomposition, QP optimizer, SMO, etc
Kernel SVMs	Common and adjustable variance			Determine γ-C pair +QP optimizer, SMO, etc
Generalized Kernel Model	Individually tuned diagonal covariance	Parsimonious in number	Quadratic	Determine kernel covariance +OLS

As for complexity control and loss function, the FF-SVESMs model "betrays his ELM brother" and shares the same complexity control method with SVMs. In FF-SVESMs, it is easy to apply a robust loss function in the penalty-type regularization, and it is the vantage in dealing with outliers.

Training algorithm for FF-SVESMs is more dramatic. At first, FF-SVESMs model follows the similar procedure in model selection on the γ-C plane as the SVMs, and then the QP optimizer and SMO method can be naturally used to minimize the cost. However, it is unfair to use SMO algorithm in FF-SVESMs, since the SMO may not be the best choice for linear SVMs. One suggestion in this paper is to use IRWLS method to minimize the Huber induced cost function, since it can be efficiently implemented by Cholesky decomposition.

Table 4 Datasets used in the simulation

Dataset	Training Example Number	Testing Example Number	Feature Number
Auto Price	**80**	**79**	**15**
Wisconsin Breast Cancer	**100**	**94**	**32**
Machine CPU	**100**	**109**	**6**
Triazines	**100**	**86**	**60**
Abalone	**2000**	**2177**	**8**
Computer Activity	**4000**	**4192**	**12**
Census(house8L)	**10000**	**12784**	**8**
Delta Ailerons	**3000**	**4129**	**5**
Delta Elevators	**4000**	**5517**	**6**
California Housing-I	**8000**	**12640**	**8**
California Housing-II	**16000**	**4640**	**8**
SinC Dataset	**300**	**5000**	**1**
SinC Dataset with Outliers	**302**	**5000**	**1**
Motorcycle	**133**	-	**1**

3. Simulation cases

The simulation is mainly in the Matlab environment running on ordinary PC with Windows XP operation system. The PC has a Pentium 4, 1.7GHZ CPU, and 512M physical memory. The main algorithm is also written in *C*-code for efficiency. The matrix operations such as matrix-matrix product, Cholesky factorization are implemented by the library of BLAS\LAPACK. For the implement of classical ε-SVR-SMO, we use the latest *C*-code version of LIBSVM(v2.85)[41]. All the *C* programs are complied and linked in Microsoft visual studio 2005.

Most of the dataset in this paper can be downloaded from the website[2]. The datasets used in this paper are listed in Table 4, and we split the whole data for training and testing according to [7], and the input vector is normalized into the range of [–1, +1], while the target output are normalized to [0, 1]. For each dataset, the training data set and testing data set are randomly generated at each trial. The prediction performances are measured by the root mean square (RMS) error.

3.1 "γ-C" Plane

In the simulation, we use the Logsig activation function in reservoir. Instead of selecting different reservoir sizes for different datasets, we use a fixed reservoir size (N=200) for each dataset. Comparing with the dynamic reservoir, there are no the sparseness and spectral radius parameters, and the input weight connections in [17] are now dependent on the hyper-parameters γ, that is to say, the input weight connections will be uniformly distributed over ($-\gamma$, γ).

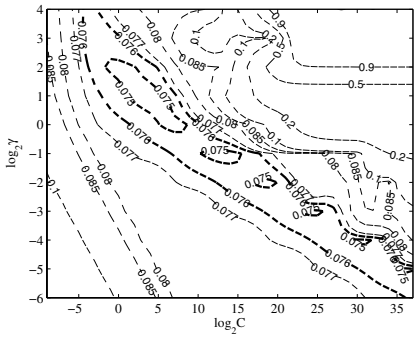
Now the "γ-C" plane is demonstrated by calculating the prediction error of the trained model on the testing examples. The two hyper-parameters are combinations of exponentially growing sequences of γ and C: γ=[…, 2^{-10}, 2^{-9}, 2^{-8}, …, 2^{8}, 2^{9}, 2^{10}, …] and C=[…, 2^{-10}, 2^{-9}, 2^{-8}, …, 2^{8}, 2^{9}, 2^{10}, …]. Fig 4(a) and (c) are the "γ-C" planes for the Abalone and Wisconsin breast cancer datasets. To make comparisons with kernel method, the "γ-C" planes of ε-SVR (with Gaussian kernel function) for the same datasets are indicated in Fig 4(b) and (d). In Fig.4, each contour line is labeled by the cross validation errors on the plane. For both datasets, proper working areas are indicated (the contour plot at the level 0.076 for abalone dataset and level 0.265 for Wisconsin breast cancer dataset), and it is obvious that both reservoir and kernel method have a wide working area.

Because of the regularization, the model performance does not change with the reservoir size when it is large enough. There is no stable and regular working area when the reservoir size is too small. According to the experience in [4], the size 200 should be large enough for them. In fact, the rough working areas are established when the reservoir size is around 25 for abalone and 10 for Wisconsin breast cancer, however, the working areas at this stage are not very stable subject to the change of reservoir size. It is also observed that the prediction error on the testing example is relatively small and stable on a larger reservoir, and similar results can be obtained for the ELM method if the TSVD algorithm is applied.

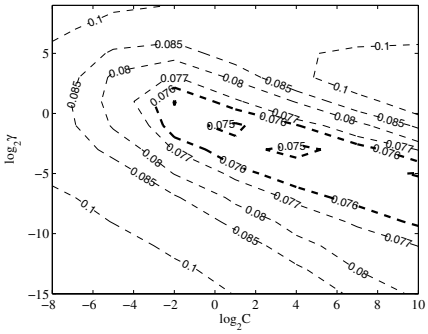
For both reservoir and kernel methods, the grid search method can be easily parallelized in searching a good pair of (γ, C). In classical SVMs, if the regularization C is too large, it will need more iterations and longer time than a small C. The computation time depends on the nodes responsible for the large C, so there is a problem to distribute the task in a cluster. In Fig.4(b), we cut a part of the working area of the ε-SVR, since the computation time with large C is long and not recommended. In reservoir method, large regularization C does not have a strong impact to the computation load, and each point on the grid has a similar training time, so the areas with large C are also recommended. In addition, the computation in reservoir method involves a single Cholesky decomposition (or several iterations of Cholesky decomposition in IRWLS algorithm for Huber loss function),

[2] http://www.liaad.up.pt/~ltorgo/Regression/DataSets.html

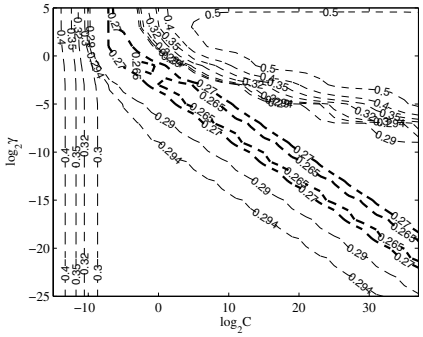
and it is a more efficient computation.



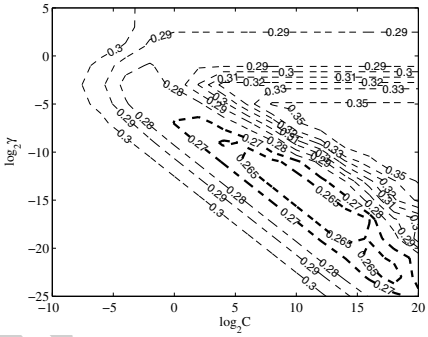
(a) Reservoir γ-C plane for abalone dataset



(b) Kernel γ-C plane for abalone dataset



(c) Reservoir γ-C plane for Wisconsin breast cancer dataset



(d) Kernel γ-C plane for Wisconsin breast cancer dataset

Fig.4 Typical reservoir γ-C plane and kernel γ-C plane by cross validation error

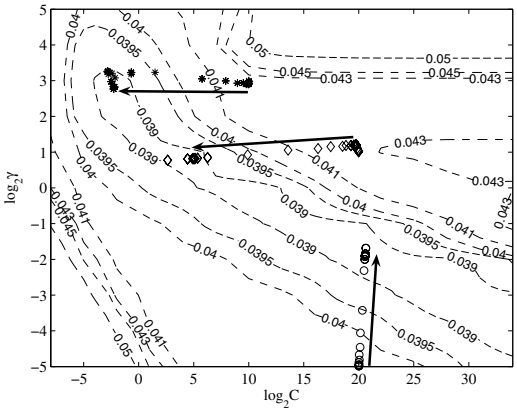


Fig 5 Find the working area by downhill simplex method

Fig.5 demonstrates the process of finding the working area using downhill simplex method on Delta Ailerons dataset. The contour plot shows the leave-one-out generalization error, which is computed with a dense grid. The objective of the downhill simplex algorithm is to minimize the leave-one-out generalization error. The initial values for (γ-C) are set to different places on the plane, as shown in Fig.5, they converge into the proper working

area (contour plot at the level of 0.039) after several iterations, and the final leave-one-out generalization error are very similar.

3.2 Comparison with the classic nonlinear kernel method

The training example size of the Auto Price, Wisconsin Breast Cancer, Machine CPU and Triazines are small (≤100), and they can be trained by a QP optimizer [17]. The remaining datasets have relatively larger training examples, and it will need more training time even the SMO algorithm is applied. This issue has been addressed in [26] that the decomposition algorithm is not a good choice for linear SVMs when the regularization parameter *C* is large.

Table 5 Comparison with the Classic ε-SVR with nonlinear Kernel in Prediction Error Mean (Standard Deviation)

Dataset	FF-SVESMs (Reservoir size is 200)		SVR-SMO (Gaussian kernel)
	Quadratic	Huber	ε-insensitive
Auto Price	**0.0897(0.0132)**	**0.0896(0.0103)**	**0.0913(0.0138)**
Wisconsin Breast Cancer	**0.2597(0.0128)**	**0.2594(0.0113)**	**0.2645(0.0102)**
Machine CPU	**0.0540(0.0171)**	**0.0503(0.0149)**	**0.0592(0.0227)**
Triazines	**0.1821(0.0159)**	**0.1772(0.0166)**	**0.1859(0.0191)**
Abalone	**0.0766(0.0014)**	**0.0765(0.0013)**	**0.0775(0.0016)**
Computer Activity	**0.0341(0.0012)**	**0.0334(0.0012)**	**0.0342(0.0004)**
Census(house8L)	**0.0654(0.0014)**	**0.0650(0.0013)**	**0.0651(0.0012)**
Delta Ailerons	**0.0388(0.0004)**	**0.0388(0.0004)**	**0.0389(0.0006)**
Delta Elevators	**0.0533(0.0006)**	**0.0530(0.0005)**	**0.0536(0.0005)**

Table 6 Comparison with the Classic ε-SVR with nonlinear Kernel in Time Consumption

Dataset	FF-SVESMs (Reservoir size is 200)		SVR-SMO (Gaussian kernel)
	Quadratic	Huber	ε-insensitive
Auto Price	**0.013s**	**0.096s**	**0.049(66sv)**
Wisconsin Breast Cancer	**0.017s**	**0.088s**	**0.040(31sv)**
Machine CPU	**0.017s**	**0.178s**	**0.052(38sv)**
Triazines	**0.017s**	**0.113s**	**0.024s(94sv)**
Abalone	**0.122s**	**0.861s**	**0.896s(786sv)**
Computer Activity	**0.231s**	**3.388s**	**6.192s(1334sv)**
Census(house8L)	**0.585s**	**4.623s**	**13.358s(1903sv)**
Delta Ailerons	**0.178s**	**1.506s**	**1.542s(691sv)**
Delta Elevators	**0.250s**	**2.016s**	**1.127s(502sv)**

Table 5 and Table 6 give the comparison results with the classic ε-SVR with Gaussian kernel. The RMS errors on the testing examples are given in Table 5. For FF-SVESMs, the reservoir size *N* is set to 200, and Cholesky decomposition and IRWLS algorithm are used to solve the optimization problems related to the Quadratic and Huber loss functions. The optimal γ-*C* pair is identified by a grid searching procedure. The similar procedures are applied to the γ-*C* pair in ε-SVR, in which the ε parameter is also adjustable to obtain the best cross

validation error, and the numbers of support vectors are recorded in Table 6. The RMS error is the average over 50 random trails, and standard deviations for each dataset are computed and listed in the brackets after the RMS error. The prediction performance of the FF-SVESMs is very competitive with the classic ε-SVR-SMO method in prediction error. The results indicate that the static reservoir has the same ability to process nonlinearity as its kernel counterpart.

The training time of the FF-SVESMs and ε-SVR-SMO are listed in Table 6. It is very interesting that the training of the FF-SVESMs is very punctual. Since the reservoirs have the same dimension for all the datasets, the problem scales for Cholesky decomposition are the same, and the training time depends on the size of training examples, in other words, with a complexity $O(l)$. In comparison, the training time of classic ε-SVR-SMO depends on the number of training example size, number of support vector and regularization parameter C.

The training time for Huber loss function is several time longer than that of Quadratic loss function, and it can be easy concluded since the solving of linear equation of Eq.(8) has the same computation load as Eq.(5). The computation of the weight matrix $\Phi(\mathbf{e})$ can be neglected since most elements are equal to one.

In fact, according to [4], the reservoir size (N=200) is larger than necessary for the datasets of the Auto Price, Wisconsin breast cancer, Machine CPU, Triazines, Abalone and Delta Ailerons. If the reservoir size is 50, the computation time can be further reduced without any performance deterioration.

Since the Huber loss and ε-insensitive loss are robust against outliers, it is necessary to make comparisons between the two methods. For Computer Activity dataset, the training time is 3.388 seconds in FF-SVESMs under Huber loss function, which is 45.2% shorter than that of the ε-SVR-SMO method. For Census (house8L) dataset, the training time of the proposed robust method is 65.4% shorter than its kernel counterpart, which needs 13.358 seconds to get approximately 1903 support vectors.

From Table 5, it is clear that the Huber loss function does not have significant better prediction performance than the Quadratic loss function. It can be explained as follows: (1) there are no reports about outlier information in these datasets. Most of them are noisy, but may be outlier free; (2) if there are some outliers in the datasets, the outlier may appear in testing set as well as in training set. In Table 5, the whole dataset is split into training set and testing set at random. If the outliers are in training set, the Huber loss will result in a better than the Quadratic loss function; otherwise, if the outliers appear in the testing set, the RMS error will not be a good indication of prediction performance.

3.3 Comparison with the classic ELM method

This section presents the comparison results with the classic ELM method. The classic ELM code is from the Huang's websites[3], which is implemented in matlab by calling a single LAPACK routine GESVD. We also implement the FF-SVESMs in matlab, in which the matrix-matrix product (A^TA) and the Cholesky factorization are implemented by calling the BLAS\LAPACK routines by a matlab C-MEX utility.

The dataset used in this section is the California Housing data, which contains 20640 examples. In the simulation, two regression tasks are produced by setting the training example size l=8000 and l=16000 respectively. The reservoir size N is chosen as 80 (recommended by literature [4]), 200, 600 and 1200 to test the complexity growing with N. The simulation experiment will focus on the training time and accuracy at different problem scales.

Table 7 lists the comparison results between FF-SVESMs and classic ELM. The computation time used in the proposed FF-SVESMs method is much shorter, which is about 1/10 that used in classic ELM. For example, when l=8000 and N=80, the training time used in proposed method is 0.19 second, while the classic ELM needs

[3] http://www.ntu.edu.sg/home/EGBHuang

1.88 second. The training time in FF-SVESMs grows as $O(lN^2)$ approximately, while the classic ELM needs a SVD of the large matrix A and it is relatively time-consuming.

As for the prediction accuracy, two methods are comparable when the reservoir size is small (N=80 in this example). However, the prediction performance of classic ELM method deteriorates when reservoir size grow, for example, the prediction results are meaningless when N is 600 or 1200. It is found that the 2 norm condition number of matrix A (cond (A) in Table 7) grows to a very large scale, usually in the order of 10^7~10^8, and it leads to an even higher eigenvalue spread of correlation matrix, which is one of the problems in the reservoir riddles[32].

In the proposed FF-SVESMs method, the ill-posed problem can be tackled by the Tikhonov-type regularization (by finding a proper area in γ-C plane). If the reservoir size N is large, for many datasets, the condition number of matrix A is large and meaningless results will be obtained by the classic ELM method, and it is the reasons why no comparison results with the classic ELM are listed in Table 6. In fact, the truncation regularization method can be used in classic ELM to improve the prediction accuracy.

Table 7 Comparison of the FF-SVESMs and Classic ELM on the California Housing Dataset

l	N	FF-SVESMs			Classic ELM			
		Train error	Test error	Time(s)	Train error	Test error	Time(s)	cond(A)
8000	80	**0.1261**	**0.1312**	**0.19**	**0.1256**	**0.1314**	**1.88**	**5.5E+4**
	200	**0.1227**	**0.1250**	**0.53**	**0.1171**	**0.1643**	**6.32**	**6.8E+5**
	600	**0.1157**	**0.1210**	**3.01**	-	-	**41.86**	**2.8E+7**
	1200	**0.1127**	**0.1194**	**10.81**	-	-	**191.25**	**5.7E+8**
16000	80	**0.1264**	**0.1296**	**0.34**	**0.1265**	**0.1295**	**3.91**	**4.9E+4**
	200	**0.1223**	**0.1244**	**0.94**	**0.1186**	**0.1424**	**12.83**	**5.6E+5**
	600	**0.1160**	**0.1194**	**5.82**	-	-	**65.14**	**1.9E+7**
	1200	**0.1129**	**0.1176**	**21.07**	-	-	**326.36**	**3.0E+8**

Comparing with the classic ELM method, the third attractive feature in the proposed method is the small memory consumption. The full matrix A are not necessary to be computed, since the algorithm only needs A^TA. For example, when l=16000 and N=200, the memory consumption in the proposed method is only 1/80 that in classic ELM method, if the matrix-matrix multiply is implemented in the manner of vector outer product. The feature is very useful for large-scale problems.

3.4 Complexity of reservoir

In this section, an illustrated example is given to show the complexity control in a static reservoir. The γ-C plane is investigated when the reservoir size varies. At the same time, target functions with different nonlinearities are used to test the model capacity of the static reservoir.

Three datasets with different nonlinearities are generated by the artificial SinC function:

$$y(x) = \begin{cases} \sin(x)/x, & x \neq 0 \\ 1, & x = 0 \end{cases}$$

on the following three different intervals:

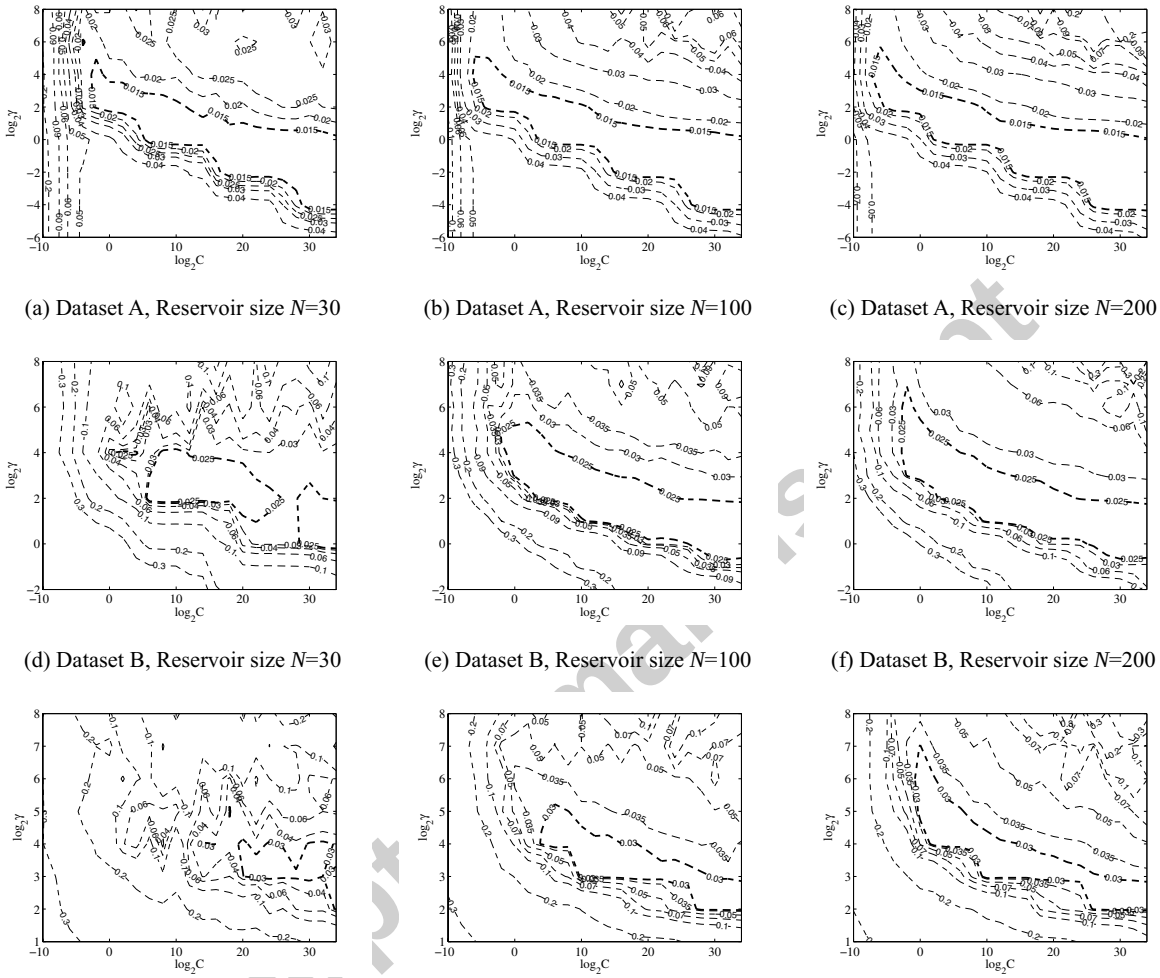
Dataset A: $x \in [-1, +1]$;

Dataset B: $x \in [-5, +5]$;

Dataset C: $x \in [-15, +15]$;

Three training sets with 300 data points are created when x is uniformly distributed on the above intervals. The

noise added to the training target is Gaussian distributed with a mean of zero and a standard deviation of 0.1. Before the three datasets are used as the training examples, the input is normalized into the range of [–1, +1]. It is clear that the complexity of the dataset grows with the length of its interval, that is to say, Dataset A is much easy to model, while Dataset C is hard to fit.



(a) Dataset A, Reservoir size N=30 (b) Dataset A, Reservoir size N=100 (c) Dataset A, Reservoir size N=200

(d) Dataset B, Reservoir size N=30 (e) Dataset B, Reservoir size N=100 (f) Dataset B, Reservoir size N=200

(g) Dataset C, Reservoir size N=30 (h) Dataset C, Reservoir size N=100 (i) Dataset C, Reservoir size N=200

Fig.6 Complexity control of reservoir for SinC function approximation

Reservoir sizes are selected as 30, 100 and 200 in the simulation, and there will be 3×3 γ-C planes. Generalization error measures are calculated on a noiseless testing set with 5000 data points, and the 9 γ-C planes are shown in Fig.6.

Dataset A is easy to fit, and the RMS error 0.015 is used to indicate the working area. The working areas are long and broad(as shown in Fig.6(a)-(c)), and it seems that the three reservoirs are all applicable for dataset A. Dataset B is relatively difficult to model, and the working area is given by a contour plot at the level of 0.025 in RMS error. From Fig.6, it is observed that the working areas for N= 30 are smaller than those for N=100 and 200. Dataset C is hard to learn, and we use the RMS error 0.03 to specify a working area. For dataset C, the size of working area grows monotonously with the reservoir size N.

The role of the global scaling parameter γ is to control the location where the activation functions response to an input signal. The nonlinearities of the reservoir will be strong if the parameter γ is large. Among the three

datasets, Dataset C is the most difficult to fit, and so the scaling parameter γ used for dataset C should be the largest. This is confirmed by the simulation, in fact, the minimal parameter γ for the three datasets are respectively 2^{-4}, 2^0 and 2^2. On the γ-C planes in section 3.1, it is observed that the value 2^0 for parameter γ is always across the working areas for these benchmark datasets. So it is lucky for the classic ELM to set the input weights into [–1, +1]. For the dataset C in this section, the fixed setting for ELM may not be so lucky.

The regularization parameter C is another factor to control the reservoir complexity. Small C limits the model capacity of a reservoir, while large C frees all the energy of a reservoir. So there is a balance between γ and C in γ-C plane, and a proper working area will be the combinations of the "small γ and large C", "moderate γ and C" and "large γ and small C", and it is why the working area is broad and long.

To obtain a broad and long working area is a luxurious objective; however, a wide working area will make it easy to search proper hyper-parameters. So it is recommended to use a relatively large reservoir.

3.5 Robustness against outliers

In this section, two illustrated examples are given to show the robustness of the FF-SVESMs. The first example is the extension of the previous SinC function approximation problem. A training set with 300 data points is created where x is uniformly distributed on the interval [–15, +15]. The noise added to the training target is Gaussian distributed with a mean of zero and a standard deviation of 0.1. Two outliers at (–11, 2.5) and (0, –1) are also added to the training examples. The 302 training set is shown in Fig.7. To verify model performance, a noiseless testing set with 5000 data points are generated.

The reservoir size is 100 in this example, and both Quadratic loss and Huber loss function are used. The fitting results are shown in Fig.8. As shown in the figure, the outliers have a bad impact on the fitted model if the Quadratic loss is used, and the fitted curve is pulled to the direction of the outliers at –11 and 0. However, the outliers have little effects on the fitted model by Huber loss function. It is observed that the $\phi(e_i)$ related to the two outliers have a tendency to very small values in the IRWLS algorithm.

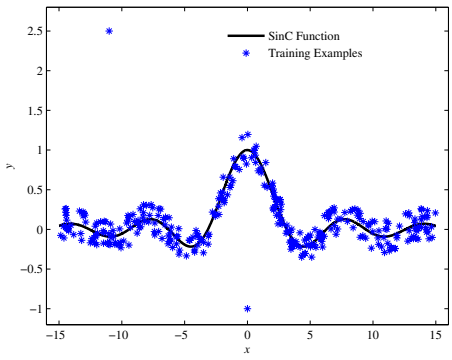


Fig. 7 Training Examples of SinC Function

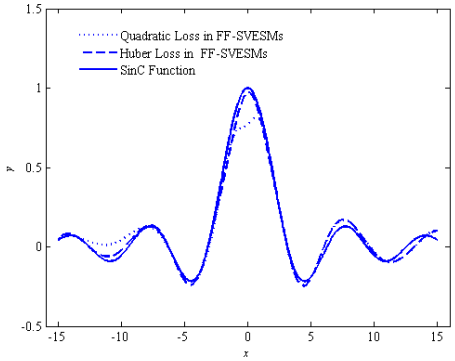


Fig. 8 Comparison Results of Quadratic loss and Huber loss on SinC dataset

The quantitative index is calculated to obtain the fitting performance. Table 8 lists the RMS error the testing set calculated on the fitted model by Quadratic and Huber loss functions. When there are no outliers, the performances of the two loss are the same (0.0305), since the best result by Huber loss is obtained by setting the μ large enough (In fact it reduces to a Quadratic loss function).

The RMS error almost doubles if the outliers are added to the training set when the Quadratic loss function is used. The performance of Huber loss will not be worse than the Quadratic loss, since at least the Huber loss can reduce to the Quadratic loss if the parameter μ is large enough and all the elements in the weight matrix are one.

Once there are some outliers in the training example, the Huber loss function will take advantage of robustness and has good fitting performance.

The second simulation is carried on the motorcycle data [38], a well-known benchmark data set in statistics. The reservoir size is set to 100, and the two loss functions are applied. The fitted results are shown in Fig.9. It can be seen that both loss functions have satisfying fitting results. The slight difference in the fitted curve between the two loss functions, bases on numerous simulation trails, is located around 35 in Fig.9. The fitted model by Quadratic loss is pulled to the point (35.2, –54.9), however, the Huber loss is not sensitive to the point, which is indicated in the weight matrix in the IRWLS procedure.

Table 8 Comparison of Test Error for different Loss Functions on SinC dataset

Items	Quadratic	Huber
Noisy SinC without Outliers	**0.0305**	**0.0305**
Noisy SinC with Outliers	**0.0628**	**0.0322**

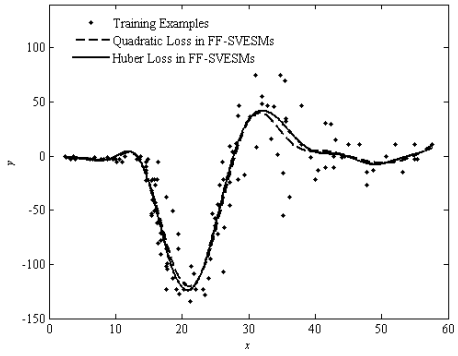


Fig.9 Comparison results of Quadratic loss and Huber loss on Motorcycle dataset

4. Conclusions

The mapping in kernel function is implicit, while the mapping in reservoir is explicit one. Base on this idea, the role kernel function now is replaced by a static reservoir in a new learning machine. The proposed method inherits the basic idea from support vector echo-state machines, but applies for static nonlinear regression estimation problem. Based on the experience of echo state networks and extreme learning machine, the scaling parameter γ and the regularization parameter C are used to characterize a reservoir, and the proper reservoir is identified on the γ-C plane. After obtaining the proper parameter pair of γ-C, an efficient learning algorithm for output weights is used, and it needs short computation time for large-scale training examples under different loss functions. The FF-SVESMs method is compared with the classic support vector machines trained by sequential minimal optimization and the classic extreme learning machine trained by pseudoinverse, and the simulation indicates that the prediction error and training time of FF-SVESMs are comparable to the existing methods.

Acknowledgement

This work was supported by the project (60674073) of the National Nature Science Foundation of China, the project (2006BAB14B05) of the National Major Technology R&D Program of China and the project (2006CB403405) of the National Basic Research Program of China (973 Program). All of these supports are appreciated. The author would like to thank the Associate Editor and three anonymous referees for all their

detailed comments and suggestions.

References

[1] H. Jaeger and H. Haas, Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication, Science 304(2004) 78-80.

[2] H. Jaeger, W. Maass, and J. Principe, Special issue on echo state networks and liquid state machines, Neural Networks 20(2007) 287-289.

[3] H. Jaeger, The "echo state" approach to analysing and training recurrent neural networks. Technical Report GMD Report 148, *German National Research Center for Information Technology* 148, 2001.

[4] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, Extreme learning machine: Theory and applications, Neurocomputing 70(2006) 489-501.

[5] G.-B. Huang, L. Chen, and C.-K. Siew, Universal approximation using incremental constructive feedforward networks with random hidden nodes, IEEE Transactions on Neural Networks 17(2006) 879-892.

[6] G. B. Huang, Q. Y. Zhu, K. Z. Mao, C. K. Siew, P. Saratchandran, and N. Sundararajan, "Can threshold networks be trained directly?," *IEEE Transactions on Circuits and Systems Ii-Express Briefs* 53 (2006) 187-191.

[7] G. B. Huang and L. Chen, "Convex incremental extreme learning machine," *Neurocomputing* 70 (2007) 3056-3062.

[8] G.-B. Huang and L. Chen, "Enhanced random search based incremental extreme learning machine," *Neurocomputing,* In Press.

[9] G. R. Wang, Y. Zhao and D. Wang. "A protein secondary structure prediction framework based on the extreme learning machine," Neurocomputing, in Press.

[10] C.-W. T. Yeu, M.-H. Lim, G.-B. Huang, A. Agarwal, and Y.-S. Ong, "A new machine learning paradigm for terrain reconstruction," *IEEE Geoscience and Remote Sensing Letters* 3 (2006) 382-386.

[11] R. Zhang, G.-B. Huang, N. Sundararajan, and P. Saratchandran, "Multicategory classification using an extreme learning machine for microarray gene expression cancer diagnosis," *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 4(2007) 485-494.

[12] J. Lim, "Recursive DLS solution for extreme learning machine-based channel equalizer," *Neurocomputing* 71(2008) 592-599.

[13] W. Maass, T. Natschlager, and H. Markram, Real-time computing without stable states: A new framework for neural computation based on perturbations, Neural Computation 14(2002) 2531-2560.

[14] J. J. Steil, Online reservoir adaptation by intrinsic plasticity for backpropagation-decorrelation and echo state learning, Neural Networks 20(2007) 353-364.

[15] J. J. Steil, Backpropagation-decorrelation: online recurrent learning with *O*(*N*) complexity, in Proceedings of the 2004 International Joint Conference on Neural Networks, 2004, pp. 843-848 vol.2.

[16] M. D. Skowronski and J. G. Harris, Noise-Robust Automatic Speech Recognition Using a Predictive Echo State Network, IEEE Transactions on Audio, Speech and Language Processing 15 (2007) 1724-1730.

[17] Z. W. Shi and M. Han, Support vector echo-state machine for chaotic time-series prediction, IEEE Transactions on Neural Networks 18(2007) 359-372.

[18] B. Schrauwen, D. Verstraeten, and J. V. Campenhout, An overview of reservoir computing: theory, applications and implementations, in Proceedings of the 15th European Symposium on Artificial Neural Networks, 2007, pp. 471-482.

[19] M. Lukosevicius and H. Jaeger, Overview of Reservoir Recipes-A survey of new RNN training methods that follow the Reservoir paradigm, School of Engineering and Science, Jacobs University Bremen Technical Report 11, 2007.

[20] B. E. Boser, I. M. Guyon, and V. N. Vapnik, A training algorithm for optimal margin classifier, in Proceedings of the 5th ACM Workshop on Computational Learning Theory, Pittsburgh, PA, 1992, pp. 144-152.

[21] K. R. Muller, S. Mika, G. Ratsch, K. Tsuda, and B. Scholkopf, An introduction to kernel-based learning algorithms, IEEE Transactions on Neural Networks 12(2001) 181-201.

[22] F. Perez-Cruz and O. Bousquet, Kernel methods and their potential use in signal processing, IEEE Signal Processing Magazine 21(2004) 57-65.

[23] A. J. Smola and B. Scholkopf, "A tutorial on support vector regression," Statistics and Computing 14(2004) 199-222.

[24] V. Vapnik and S. E. Golowich, Support Vector Method for Function Approximation, Regression Estimation, and Signal Processing, in Advances in Neural Information Processing Systems 9, Cambridge, MA, 1997, pp. 281-287.

[25] J. Schmichuber, D. Wierstra, M. Gagliolo, and F. Gomez, Training recurrent networks by Evolino, Neural Computation 19(2007) 757-779.

[26] W. C. Kao, K. M. Chung, C. L. Sun, and C. J. Lin, Decomposition methods for linear support vector machines, Neural Computation 16(2004) 1689-1704.

[27] C. M. Bishop, Pattern Recognition and Machine Learning. Berlin: Springer-Verlag, 2006.

[28] H. Jaeger, Tutorial on training recurrent neural networks, covering BPTT, RTRL, EKF and the "echo state network approach. GMD Report 159," German National Research Center for Information Technology 159, 2002.

[29] S. Chen, X. Hong, C. J. Harris, and X. X. Wang, Identification of nonlinear systems using generalized kernel models, IEEE Transactions on Control Systems Technology 13(2005) 401-411.

[30] P. C. Hansen, The truncated SVD as a method for regularization, BIT Numerical Mathematics 27(1987) 534-553.

[31] C. M. Bishop, Training with noise is equivalent to tikhonov regularization, Neural Computation 7(1995) 108-116

[32] H. Jaeger, Reservoir riddles: Suggestions for echo state network research, in Proceedings of the 2005 International Joint Conference on Neural Networks, Montreal, QC, Canada, 2005, pp. 1460-1462.

[33] P. L. Bartlett, Sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network, IEEE Transactions on Information Theory 44(1998) 525-536.

[34] R. H. Myers, Classical and Modern Regression with Applications. 2nd Edition, Boston: PWS-KENT, 1990.

[35] G. C. Cawley and N. L. C. Talbot, Fast exact leave-one-out cross-validation of sparse least-squares support vector machines, *Neural Networks* 17 (2004) 1467-1475.

[36] M. J. Quinn, Parallel Programming in C with MPI and OpenMP, McGraw-Hill, 2004.

[37] C. C. Chuang, S. F. Su, J. T. Jeng, and C. C. Hsiao, Robust support vector regression networks for function approximation with outliers, IEEE Transactions on Neural Networks 13(2002) 1322-1330.

[38] J. A. K. Suykens, J. De Brabanter, L. Lukas, and J. Vandewalle, Weighted least squares support vector machines: Robustness and sparce approximation, Neurocomputing 48(2002) 85-105.

[39] G. W. Flake and S. Lawrence, Efficient SVM regression training with SMO, Machine Learning 46(2002) 271-290.

[40] G. C. Cawley and N.L.C. Talbot, Efficient model selection for kernel logistic regression, in Proceedings of the 17th International Conference on Pattern Recognition, vol.2, pp. 439-442, 23-26 Aug. 2004

[41] C. C. Chang and C. J. Lin, LIBSVM: A library for support vector machines. http://www.csie.ntu.edu.tw/~cjlin/libsvm., 2007.

Photo of authors

Zhiwei Shi received the B.S. degree from the Department of Electrical Information, Zhengzhou University, Henan, China, in 2002. He is currently working towards the Ph.D. degree at the Department of Electrical Engineering, Dalian University of Technology, LiaoNing, China. His current research interests include RNNs, SVMs and kernel method, and chaotic time series analysis.

Min Han received the B.S. and M.S. degrees from the Department of Electrical Engineering, Dalian University of Technology, LiaoNing, China, in 1982 and 1993, respectively, and the M.S. and Ph.D. degrees from Kyushu University, Fukuoka, Japan, in 1996 and 1999, respectively. She is a Professor at School of Electronic and Information Engineering, Dalian University of Technology. Her current research interests are neural network and chaos and their applications to control and identification.