Support Vector Echo-State Machine for Chaotic Time Series Prediction

Zhiwei Shi and Min Han
School of Electronic and Information Engineering,
Dalian University of Technology, Dalian
Liaoning, 116023, China
minhan@dlut.edu.cn

Abstract: A novel chaotic time series prediction method based on support vector machines and echo state mechanisms is proposed. The basic idea is replacing "kernel trick" with "reservoir trick" in dealing with nonlinearity, that is, performing linear support vector regression in the high dimension "reservoir" state space, and the solution benefits from the advantages from structural risk minimization principle, and we call it SVESMs (Support Vector Echo State Machines). SVESMs belong to a special kind of recurrent neural networks with convex objective function, and its solution is global optimal and unique. SVESMs are especially efficient in dealing with real life nonlinear time series, and its generalization ability and robustness are obtained by regularization operator and robust loss function. The method is tested on the benchmark prediction problem of Mackey-Glass time series and applied to some real life time series such as monthly sunspots time series and runoff time series of the Yellow River, and the prediction results are promising.

Keywords: Support Vector Machines, Echo State Networks, Recurrent neural networks, Chaotic time series prediction

1. Introduction

Much interest has been attached to the forecasting problems in real life time series, and one has to predict ahead in time, up to a certain time horizon (called lead time or prediction horizon). The time series in nature are usually nonlinear or chaotic, so the prediction is an issue of dynamic modeling [1]. In neural networks and machine learning communities, several neural network models and kernel-based methods are applied to time series prediction task, such as the standard MLPs (Multi-layer Perceptrons) [2], RBF (Radial basis Function) neural network [3]-[5], FIR (Finite Impulse Response) neural networks [6], SVR (Support Vector Regression) [7]-[9], SOM (Self-Organization Map) [10], GP (Gaussian Process) [11]. RNNs (Recurrent Neural Networks) including NAR (Nonlinear AutoRegressive network) [12], Elman networks [13] and Jordan networks, RPNN (Recurrent Predictor Neural networks) [14], ESN (Echo State Networks) [15] are also studied for nonlinear time series prediction. Among these prediction models, the multi-step-ahead prediction task is achieved by either doing repeated one-step-ahead predictions up to the desired horizon, which is called the iterative method, or by explicitly

Submitted to IEEE T. Neural Network, July 9, 2005

training a direct model to predict h-steps-ahead, which is called the direct model [11], [16].

ESN iterative model, recently reported in [15], is a kind of RNN which is firstly trained as an accurate one-step predictor, and then iterates the prediction into different prediction horizons in an autonomous style. On a benchmark task of predicting Mackey-Glass chaotic time series, accuracy is improved by a factor of 2400 over previous techniques. ESN is of interest for the scientific community, but puzzles the engineers and practitioners, because in practice the time series often contains considerable noise and uncertainness. The Prediction error can never be smaller than the variance of the noise. Noise reduction method usually used is far from perfect, and noise reduced data is still noisy. Moreover, for an iterative model, successfully trained one-step predictors normally fail in the application of autonomous system, since the stability of the model, and even the equivalence of the models and systems attractor are difficult to guarantee [4]. These disadvantages are especially notable for real life nonlinear time series, such as sunspots time series and runoff time series of river.

Support vector machines, firmly grounded in the framework of statistical learning theory [17], [18], is proposed by Vapnik and his co-workers in 1995. For a short overview on the statistical learning theory, refer to [19], [20]. As a new learning system, SVMs are based on the structural risk minimization (SRM) principle which seeks to minimize an upper bound of the generalization error rather than minimize the empirical error. SVMs will achieve an optimum network structure by striking a right balance between the empirical error and the VC-confidence interval, and it enable the networks to generalize well to unseen data. With the introduction of Vapnik's ε-insensitive loss function, SVMs have been extended to solve nonlinear regression estimation problems [21], and then are introduced to nonlinear time series modeling and exhibit excellent performance [7]. For a recent tutorial on support vector regression, refer to [22]. Up to now, most of the applications involving SVMs appear in the pattern recognition and static function regression, and few cases appear in recurrent neural network structure. J.A.K. Suykens and J. Vandewalle proposed a kind of SVMs within the context of recurrent neural networks [23], and it is called recurrent least squares support vector machines (RLS-SVM). In RLS-SVM, essential features of SVMs remain, such as Mercer's condition and the fact that the output weights are a Lagrange multiplier weighted sum of the data points, however, some important properties of the SVMs are lost in RLS-SVM, for example, the parameter estimation problem becomes nonconvex and the solution is not sparse. Since SVMs are now successfully applied to static function approximation and regression, we would like to introduce SVMs into the framework of recurrent neural networks without losing any properties, so that the advantages of SVMs can be used for dynamical modeling and prediction.

In this study, we intend to build a kind of recurrent neural networks for modeling nonlinear dynamic system with the advantages of the SVMs and echo state mechanisms. The basic idea is to replace the “kernel trick” with the “reservoir trick”, namely, the linear support vector regression is performed in the high dimension “reservoir” state space, and we call it SVESMs (support vector echo state machines). In the framework of SVESMs, the final solution is obtained by solving a standard quadratic programming problem, and the solution is unique and the parameter estimation problem becomes convex. One of the remarkable features of the new method is the flexibility to use different loss functions. Sparse solution representation can be obtained if the ε-insensitive loss function is applied, and the robust loss functions such as the Huber loss function can be used to create a model which is less sensitive to outliers. These new features seldom appear in traditional RNNs.

We will show how to make *h*-step-ahead *direct* prediction of chaotic time series using SVESMs. We have modified the network topology of the original *iterative* method [15] by eliminating the feedback connections from the output neuron to the reservoir, and combing the embedded memory into the network inputs. We will show how the time series prediction problem can be converted into a dynamic system modeling task, so that SVESMs can be used.

This paper is organized as follows. In Section 2, the iterative method for chaotic system prediction based on ESNs is firstly revisited, and then the direct method is introduced. Section 3 presents the basic idea of SVESMs, and it will be shown how the new kind of RNNs is obtained by replacing “kernel” with “reservoir”. In Section 4, we will give three illustrative examples to show the working mechanisms of SVESMs for chaotic time series prediction. Section 5 presents the conclusions of this paper.

2. Chaotic Time Series Prediction Using “Reservoir”

The equations of the ESNs can be written as:

$$\mathbf{x}(k+1) = \mathbf{tansig}(\mathbf{W}_{\mathrm{x}} \cdot \mathbf{x}(k) + \mathbf{W}_{\mathrm{in}} \cdot \mathbf{u}(k) + \mathbf{v}(k+1)) \tag{1}$$

$$y(k) = \mathbf{w} \cdot \mathbf{x}(k) \tag{2}$$

where **tansig** denotes hyperbolic tangent sigmoid function which is applied elementwise, $\mathbf{x}(k)$ denotes the state variables in the “reservoir”, $\mathbf{u}(k)$ and $y(k)$ are the input and output to the ESNs, respectively, $\mathbf{v}(k+1)$ is an optional noise vector. $\mathbf{W}_{\mathrm{x}}$, $\mathbf{W}_{\mathrm{in}}$ and $\mathbf{w}$ are the internal connection weights of the reservoir, the input weights to the reservoir and the readout (output) weights from the reservoir, respectively.

Basic idea of the ESNs learning is the larger and fixed “reservoir”, from which the desired output is obtained by training suitable output weights. The “reservoir” has a large number of neurons which are randomly and sparsely connected.

Determination of optimal output weights is a linear task of MSE minimization [15], [24], [25].

$$\underset{\hat{\mathbf{w}}}{\text{minimize}}\left\|\mathbf{X}\hat{\mathbf{w}}-\mathbf{y}_{d}\right\| \tag{3}$$

where $\mathbf{X}=[\mathbf{x}^{T}(Init), \mathbf{x}^{T}(Init+1), \ldots, \mathbf{x}^{T}(Trn)]^{T}$, and $\mathbf{y}_{\mathrm{d}}=[y_{\mathrm{d}}(Init), y_{\mathrm{d}}(Init+1), \ldots, y_{\mathrm{d}}(Trn)]T$, *Init* and *Trn* are the beginning and ending index of the training examples, respectively, and the size of the training set is $N_{\mathrm{t}}=Trn\text{-}Init+1$. *Init* is usually set to certain value to discard the influence of reservoir initial transient.

In this study, SVESMs implement the direct prediction model. Before the introduction of direction method, it is necessary to briefly recall the iterative prediction model reported in [15].

2.1 Iterative Prediction of Chaotic System Based on "Reservoir"

In the iterative method, the multi-step prediction is based on the iteration of the exact single-step prediction [15]. The exact estimation of $y(k)$ is fed back to the reservoir ($\mathbf{u}(k)=y(k)$), and the autonomous system for the chaotic system prediction can be written as:

$$\mathbf{x}(k+1)=\mathbf{tansig}(\mathbf{W}_{\mathrm{x}}\cdot\mathbf{x}(k)+\mathbf{W}_{\mathrm{in}}\cdot y(k)) \tag{4}$$

$$y(k)=\mathbf{w}\cdot\mathbf{x}(k) \tag{5}$$

From Eqs.(4) and (5) we have:

$$\mathbf{x}(k+1)=\mathbf{tansig}((\mathbf{W}_{\mathrm{x}}+\mathbf{W}_{\mathrm{in}}\cdot\mathbf{w})\cdot\mathbf{x}(k)) \tag{6}$$

Eq.(6) shows the state evolution of the ESNs for iterative prediction, and it is similar to the state space neural network discussed in [26]-[28]. It can be proven that this kind of RNN can model chaotic time series without considering its initial state [29]. Fig.1 shows the diagram how the iterative predictor works for chaotic system prediction.

The stability of ESNs can be insured by constraining the spectral radius of $\mathbf{W}_{\mathrm{x}}$ within the unit cycle. However, the stability of the iterative predictor can not be guaranteed by simply limiting the spectral radius of $\mathbf{W}_{\mathrm{x}}$. From Eq.(6), it can be seen that the stability of the predictor is dependent on output weights **w**, which is the result of networks training, so the stability can not be insured before **w** are determined. This is one of the reasons why the direct prediction is introduced for chaotic time series.

The iteration method also has other potential dangers as well as the stability problem. The single-step predictor must be accurate enough that the accumulation error in iteration is as small as possible. However, it is usually impossible to get an accurate predictor trained by noisy training examples, and the prediction performance deteriorates significantly in the presence of noise and outliers.

2.2 Direct Prediction of Chaotic Time Series Based on "Reservoir"

Instead of the iterations, the direct prediction method relates the prediction origin and the prediction horizon in a straightforward way. Previous direct prediction methods such as MLPs treat the relationship between the prediction origin and the prediction horizon as a static mapping [2]. Because of the feedback and memory, RNNs are inherently dynamic systems. It is necessary to show how the direct prediction problem can be converted into a dynamic system modeling task.

Theorem: For a given chaotic time series and its proper embedding, there exists a nonlinear dynamic system realizing the dynamic mapping: $\mathbf{d}(k) \rightarrow x(k+h)$, where $\mathbf{d}(k)$ is:

$$\mathbf{d}(k)=[x(k), x(k-\tau), x(k-2\tau), \ldots, x(k-(m-1)\tau)]^T$$

where m and τ are the embedding parameters of the time series $\{x(k), k=1, 2, 3, \ldots\}$, and h is a positive integer.

Proof: For a given chaotic time series $\{x(k), k=1, 2, 3, \ldots\}$, the delay embedding state vector is defined as:

$$\mathbf{d}(k)=[x(k), x(k-\tau), x(k-2\tau), \ldots, x(k-(m-1)\tau)]^T,$$

where m and τ are the embedding parameters of the time series. According to the Takens' theorem [30], if the embedding dimension m is large enough, the evolution of delay embedding vector $\mathbf{d}(k)\rightarrow\mathbf{d}(k+1)$ can recover the original dynamic system without ambiguity. It is assumed that the evolution of the delay embedding vector is described by:

$$\mathbf{d}(k+1)=\mathbf{F}(\mathbf{d}(k)),$$

and we assume $x(k)$ is available through a measurement function $g(\cdot)$:

$$x(k)=g(\mathbf{d}(k)),$$

For a positive integer h, we have the following equations:

$$\begin{aligned}
&\mathbf{d}(k) = \mathbf{d}(k) \\
&\mathbf{d}(k+1) = \mathbf{F}(\mathbf{d}(k)) \\
&\mathbf{d}(k+2) = \mathbf{F}(\mathbf{d}(k+1)) \\
&\ldots \\
&\mathbf{d}(k+h) = \mathbf{F}(\mathbf{d}(k+h-1)) \\
&x(k+h) = g(\mathbf{d}(k+h))
\end{aligned} \qquad (7)$$

Further, we assume function $\mathbf{F}(\cdot)\equiv 0$, and then, Eq.(7) is rewritten as the following state space representation:

$$\begin{bmatrix} \mathbf{d}(k) \\ \mathbf{d}(k+1) \\ \mathbf{d}(k+2) \\ \ldots \\ \mathbf{d}(k+h) \end{bmatrix} = \begin{bmatrix} \mathbf{F_0}(\mathbf{d}(k-1)) \\ \mathbf{F}(\mathbf{d}(k)) \\ \mathbf{F}(\mathbf{d}(k+1)) \\ \ldots \\ \mathbf{F}(\mathbf{d}(k+h-1)) \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \\ \ldots \\ 0 \end{bmatrix} \cdot \mathbf{d}(k) \tag{8}$$

$$x(k+h) = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{d}(k) & \mathbf{d}(k+1) & \mathbf{d}(k+2) & \ldots & g(\mathbf{d}(k+h)) \end{bmatrix}^T$$

where the vector $[\mathbf{d}^T(k)\ \mathbf{d}^T(k+1)\ \mathbf{d}^T(k+2)\ \ldots \mathbf{d}^T(k+h)]^T$ is the augmented state variable, and note that this state space realization for the input-output sequences $\{\mathbf{d}(k), x(k+h)\}$ is not unique.

From the Theorem, it can be concluded that the input-output sequences of $\{\mathbf{d}(k), x(k+h)\}$ are from a dynamic system, and RNNs such as ESNs can be used for the modeling task. In the direct method, ESNs are used as an input-output dynamic system, rather than the autonomous running as an independent system.

The input-output system for the prediction is written as:

$$\mathbf{x}(k+1) = \mathbf{tansig}(\mathbf{W}_{\mathrm{x}} \cdot \mathbf{x}(k) + \mathbf{W}_{\mathrm{in}} \cdot \mathbf{d}(k) + \mathbf{v}(k+1)) \tag{9}$$

$$y(k) = \mathbf{w} \cdot \mathbf{x}(k) \tag{10}$$

The target output of the ESNs is the h-step-ahead value of the time series. The input-output training sequences may be $\{\mathbf{d}(k), x(k+h), k=1, 2, 3, ..., N_{\mathrm{t}}\}$.

The network structure for the direct prediction can be seen from Fig.2, and it is clear that the direct prediction method no longer closes the feedback loop. The stability of the ESNs can be guaranteed, if the spectral radius of the matrix $\mathbf{W}_{\mathrm{x}}$ is smaller than one, and is not dependent on the output weights $\mathbf{w}$.

[Fig.2]

2.3 Stable Memory and Transient State in the "Reservoir"

It is known that the time series is bounded, so Eq.(8) describes a BIBO stable system. To model a BIBO stable system, the network model is necessarily BIBO stable. Because the activation function in the "reservoir" is bounded, the output signal itself is always bounded. But the reservoir is not expected to run in the state of saturation, in fact, it needs to be run with proper transient state property and stable memory.

Consider the following linear system:

$$\mathbf{x}(k+1) = \mathbf{W}_{\mathrm{x}} \cdot \mathbf{x}(k) + \mathbf{W}_{\mathrm{in}} \cdot \mathbf{u}(k) \tag{11}$$

where $\mathbf{W}_{\mathrm{x}}$ and $\mathbf{W}_{\mathrm{in}}$ are the state transition matrix and input matrix, respectively. According to linear system theory, if the spectral radius of $\mathbf{W}_{\mathrm{x}}$ is smaller than 1(absolute value of the largest eigenvalue is within the unit cycle), and the linear system (11) will be stable. For different eigenvalues of state transition matrix $\mathbf{W}_\mathrm{x}$, the system (11) shows different transient properties. Within the unit cycle, the larger the magnitude the eigenvalue is, the longer the transient process will persist in.

For the "reservoir" neurons, the activation function is the hyperbolic tangent sigmoid function, and we can follow that for any k:

$$\left\|\mathbf{tansig}(\mathbf{W}_\mathrm{x} \cdot \mathbf{x}(k))\right\| \leq \left\|\mathbf{W}_\mathrm{x} \cdot \mathbf{x}(k)\right\| \qquad (12)$$

and it is concluded that the system (9) is more "stable" than system (11), if the spectral radius of $\mathbf{W}_\mathrm{x}$ is smaller than 1 (Note that it is not a necessary condition). The spectral radius of matrix $\mathbf{W}_\mathrm{x}$ is suggested to be smaller than 1 in [31]. It is shown by [32] that ESNs can be stable if the spectral radius is set to be slightly larger than 1 under some conditions, and recently, a rigorous bound for guaranteeing asymptotic stability of ESN is obtained in [33]. Anyway, the transient property and memory of "reservoir" can be controlled by the initial setting of $\mathbf{W}_\mathrm{x}$ in the preparation phase. So the following job will be the selection of proper reservoir within a stable region.

Connection to the static mapping by using feed-forward neural networks

Let $\mathbf{W}_\mathrm{x}$=0, and the spectral radius of $\mathbf{W}_\mathrm{x}$ will be 0. That is to say, if there is no feedback in reservoir, the prediction model will become a static mapping implemented by traditional feed-forward neural network, such as in [2] [7]. In other word, feedforward model is in fact a special case of the dynamic model which stated in section 2.2.

3. Support Vector Echo State Machines

3.1 Linear Support Vector Machine on the State Space from "Reservoir"

As stated by H. Jaeger in [15], because there are no cyclic dependencies between the trained output weights, training ESNs becomes a linear regression task. The prediction function for new network inputs is given by (without loss of generality, a bias term b is added to the regression function),

$$f(\mathbf{x}^{\text{test}}) = \mathbf{w}^T \mathbf{x}^{\text{test}} + b \qquad (13)$$

where $\mathbf{x}^{\text{test}}$ is the new reservoir state vector excited by the new network inputs. The reservoir used for new prediction is the same as the one used in the training phase. The output weights can be obtained by solving a standard least squares problem,

$$\hat{\mathbf{w}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}_d \qquad (14)$$

In practice, for ill-conditioned data, the least-squares method often yields a meaningless estimation for the output weights. The vector of (14), if it exists at all, is usually a bad approximation to real solution $\mathbf{w}$. Very large weights imply bad numerical stability. To avoid this problem, “noise injection” to the state can improve the stability of the solution, because noise term perturbs the matrix $\mathbf{X}^T\mathbf{X}$. However, if the matrix $\mathbf{X}$ has more columns than its rows, the matrix $\mathbf{X}^T\mathbf{X}$ is always singular, and the “noise injection” does not take effects. For such ill-posed problems, pseudoinverse or regularization techniques can be used to obtain meaningful solution. Good approximate inverses for regularization can be derived by modifying the standard least squares formula. The traditional Tikhonov regularization [34] (or ridge regression) is given by:

$$\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}_d \tag{15}$$

$\lambda \in R^+$ (posivtive number is added to $\mathbf{X}^T\mathbf{X}$), and it is equivalent to minimize the following objective function:

$$\sum_{j=1}^{N_t}(<\mathbf{w}\cdot\mathbf{x}_j> +b - y_{dj})^2 + \lambda\|\mathbf{w}\|^2 \tag{16}$$

where λ balances the two terms in Eq.(16). However, the objective function only contains the quadratic loss, which can be extended to a more general form.

The structural risk minimization principle suggests a tradeoff between the quality of the approximation and the complexity of the approximating function. The primal objective function is:

$$L_p(\mathbf{w},b) = C\sum_{j=1}^{N_t} L[\mathbf{x}_j, y_{dj}, f] + \|\mathbf{w}\|^2 \tag{17}$$

where L is a general loss function (to be defined later) and f is the prediction function defined by $\mathbf{w}$ and b. The second term in Eq.(17) is a regularizer and C is the regularization constant.

The regularizer in objective function (17) takes an implicit pruning role [35], [36] (compared with the constructive or growing method in [37], [38]). To obtain better generalization ability, H. Jaeger suggested increasing reservoir sizes until performance on test data deteriorates [25], and it is a growing algorithm.

“Reservoir Trick” instead of “Kernel Trick”

Kernel-based algorithms [39] such as SVMs have been developed in recent years, and it is a nonlinear version of a linear algorithm where the data have been previously (nonlinearly) transformed to a higher dimensional space in which we only need to be able to compute inner products (via a kernel function), and it is called the “kernel trick”. Similar to kernel-based algorithms, a nonlinear system problem is converted into a linear system problem in ESNs (via a reservoir). Also, the calculation of the linear problem occurs in a higher dimensional space called the “reservoir” state space, and we call it the “reservoir trick”.

However, there are some differences between the “kernel trick” and the “reservoir trick”: first of all, the “kernel trick” usually appears in the problems of static function regression or pattern recognition (the exception may be the RLS-SVM [23]), and the "reservoir trick" deals with the problems of dynamic system identification. Secondly, the "kernel trick" relies on kernel function, so there needs a careful selection of the hyperparameters [40]-[42] (for example, the width of RBF kernel or the order of Polynomial kernel), at the same time, "reservoir trick" needs a selection of the "reservoir" (for example, the sparseness and spectral radius of the reservoir weights)

3.2 Loss Function

For the choice of the loss function in Eq.(17), the following factors should be addressed: convexity, robustness and sparseness. Consider the loss function $|y\text{-}f(x)|^p$, if p<1, the problem becomes nonconvex and undesirable, because it is necessary to use a convex objective function; and if p>1, the solution may also not be desirable because the superlinear increase will lead to a loss of the robustness properties of the estimator. Generally speaking, measurement noise and outliers often pollute the time series, and the practically measured time series (even after a noise reduction procedure) used for modeling is far from perfect. So prediction model must be robust against such uncertainness and perturbations in the data.

Quadratic Loss Function

Using a quadratic loss function,

$$L_{quad}(f(x)-y)=(f(x)-y)^2 \tag{18}$$

We tend to minimize the function,

$$\begin{aligned} &\text{minimize}\|\mathbf{w}\|^2+C\sum_{j=1}^{N_\mathrm{t}}(\xi_j^2+\hat{\xi}_j^2)\\ &\text{subject to}\quad (<\mathbf{w}\cdot\mathbf{x}_j>+b)\text{-}y_j\le\xi_j\quad j=1,\ldots,N_\mathrm{t}\\ &\qquad\qquad y_j\text{-}(<\mathbf{w}\cdot\mathbf{x}_j>+b)\le\hat{\xi}_j\quad j=1,\ldots,N_\mathrm{t}\\ &\qquad\qquad \xi_j,\hat{\xi}_j\ge 0\quad j=1,\ldots,N_\mathrm{t}\end{aligned} \tag{19}$$

and the solution is given by solving the dual problem,

$$\underset{\beta}{\text{minimize}}\frac{1}{2}\sum_{i=1}^{N_\mathrm{t}}\sum_{j=1}^{N_\mathrm{t}}\beta_i\beta_j<\mathbf{x}_i\cdot\mathbf{x}_j>-\sum_{i=1}^{N_\mathrm{t}}\beta_i y_i+\frac{1}{2C}\sum_{i=1}^{N_\mathrm{t}}\beta_i^2 \tag{20}$$

with constraint,

$$\sum_{i=1}^{N_\mathrm{t}}\beta_i=0 \tag{21}$$

where β_i and β_j are the corresponding Lagrange multipliers.

In kernel-based learning algorithm, there is a kernel version of the statistical model of ridge regression, which is called kernel ridge regression. Correspondingly, there exists a reservoir version of ridge regression, which we call reservoir ridge regression. In fact, the optimization problem posed by Eq.(19) can also be resolved effectively by ridge regression without resorting to QP optimizers. Quadratic loss function is frequently used in the training of neural networks, however, it is sensitive to irregular values and over-emphasizes the outliers in the training data, even a single outlier can heavily corrupt the learned model [43]. That is to say, the quadratic loss function will result in a less robust solution.

ε-insensitive Loss Function

Using an ε-insensitive loss function,

$$L_\varepsilon(y)=\begin{cases}0 & \text{for } \left|f(x)-y\right|<\varepsilon \\ \left|f(x)-y\right|-\varepsilon & \text{otherwise}\end{cases} \tag{22}$$

we arrive at the formulation stated in Vapnik [17],

$$\begin{aligned}&\text{minimize}\left\|\mathbf{w}\right\|^2+C\sum_{j=1}^{N_\text{t}}(\xi_j+\hat{\xi}_j)\\ &\text{subject to}\quad (<\mathbf{w}\cdot\mathbf{x}_j>+b)\text{-}y_j\le\varepsilon+\xi_j \quad j=1,\ldots,N_\text{t}\\ &\qquad\qquad y_j\text{-}(<\mathbf{w}\cdot\mathbf{x}_j>+b)\le\varepsilon+\hat{\xi}_j \quad j=1,\ldots,N_\text{t}\\ &\qquad\qquad \xi_j,\hat{\xi}_j\ge 0 \quad j=1,\ldots,N_\text{t}\end{aligned} \tag{23}$$

and the solution is given by solving the following dual problem,

$$\underset{\alpha,\alpha^*}{\text{minimize}}\frac{1}{2}\sum_{i=1}^{N_\text{t}}\sum_{j=1}^{N_\text{t}}(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*)<\mathbf{x}_i\cdot\mathbf{x}_j>-\sum_{i=1}^{N_\text{t}}(\alpha_i-\alpha_i^*)y_i+\sum_{i=1}^{N_\text{t}}(\alpha_i+\alpha_i^*)\varepsilon \tag{24}$$

with constraints,

$$\begin{aligned}&0\le\alpha_i,\alpha_i^*\le C, i=1,\ldots,N_\text{t}\\ &\sum_{i=1}^{N_\text{t}}(\alpha_i-\alpha_i^*)=0\end{aligned} \tag{25}$$

where *a* and *a** are the corresponding Lagrange multipliers, and the same to the quadratic loss function, solving the minimization problem (24) with constrains (25) determines the Lagrange multipliers, α and α*.

The ε-insensitive loss function has the robust property as stated above. First of all, it allows an insensitive tube, and any uncertainness and perturbations within the insensitive tube have no loss; secondly it penalizes the error in a linear style outside the insensitive tube, which is better than quadratic loss function if there are many outliers.

Huber Loss Function

Using a Huber loss function,

$$L_{huber}(f(x)-y)=\begin{cases}\frac{1}{2}\left(f(x)-y\right)^2 & \text{for} \quad |f(x)-y|<\mu \\ \mu|f(x)-y|-\frac{\mu^2}{2} & \text{otherwise}\end{cases} \tag{26}$$

and the corresponding optimization problem is,

$$\underset{\beta}{\text{minimize}}\frac{1}{2}\sum_{i=1}^{N_{\mathrm{t}}}\sum_{j=1}^{N_{\mathrm{t}}}\beta_i\beta_j < x_i \cdot x_j > -\sum_{i=1}^{N_{\mathrm{t}}}\beta_i y_i + \frac{1}{2C}\sum_{i=1}^{N_{\mathrm{t}}}\beta_i^2 \mu \tag{27}$$

with constraints,

$$\begin{aligned} & -C \le \beta_i \le C, i=1,\dots,N_{\mathrm{t}} \\ & \sum_{i=1}^{N_{\mathrm{t}}}\beta_i = 0. \end{aligned} \tag{28}$$

Huber loss function can also produce robust solution, and it does not allow the superlinear increase of the loss by replacing it with a linear loss outside a specified region. However, sparseness of solution is not available.

There are also some other loss functions, for example, Laplacian, Polynomial and Piecewise polynomial. By using more control parameters, [44] introduces a unified loss function which includes Laplacian, Huber and ε-insensitive loss functions as special cases. The question is which loss function should be used for a particular time series prediction task. It is shown in [7] that the ε-insensitive loss is preferred if the additive noise is uniform, and Huber loss is preferred if the noise becomes Gaussian (refer to [22] for an overview over some common density models). However, when the noise distribution is unknown, it is difficult to decide which one is the best. Noise reduction technique is necessary for time series modeling, and it makes the data less noisy. However, for a short and noisy real life time series, the noise reduction method is far from perfect, the data may still contain considerable noise, and the noise density after the noise reduction is complex and it is better to try different loss functions for the given modeling task.

3.3 The Solution of Support Vector Echo State Machines

For different loss functions discussed above, the solution of SVESMs can be obtained by resolving the corresponding optimization problems. All these problems belong to the standard convex quadratic programming with linear constraints, and they can be solved in polynomial time. The solutions are given by a set of lagrange multipliers and the prediction function for the new reservoir test state variable $\mathbf{x}^{\text{test}}$ is generally written as:

$$f(\mathbf{x}^{\text{test}})=\sum_{i=1}^{s}\alpha_s(i)\mathbf{x}^T\mathbf{x}^{\text{test}}, \tag{29}$$

where $a_s(i)$ is the Lagrange multiplier corresponding to the reservoir state vector $\mathbf{x}(i)$. s is the number of support vectors, which is controlled by the size of the insensitive tube if the ε-insensitive loss function is applied.

SVESMs use a linear support vector regression in the reservoir state space, and its properties of convergence and time complexity can be analyzed in the same manner as the standard SVMs. In this paper, only batch learning is used. The time complexity of the algorithm is dependent on the number of training patterns. Standard quadratic programming requires $O(M^3)$ time complexity and $O(M^2)$ space complexity, where M is the number of training patterns. For practical implement of SVMs, many algorithms such as interior point method, chunking, decomposition, SMO (Sequential Minimal Optimization) or parallel SMO [45] can be used. State-of-the-art SVM implementations typically have a training time complexity that scales between $O(M)$ and $O(M^{2.3})$ from empirical observations [46]. The study of reducing the computational load for large scale problem becomes an active area in the recent research of SVMs. In SVESMs, the reservoir can be very large, but the optimization problem scale does not grow with the size of reservoir.

Support vector regression is a generalization of ridge regression and has several additional features. Ridge regression is equivalent to the linear support vector regression with quadratic loss function. When there is no regularization term ($C\rightarrow+\infty$), both of them reduce to the standard least-squares solution. The method of pseudoinverse also provides a solution in the sense of least square. So ridge regression as well as the method of pseudoinverse will suffer from the performance deterioration in the presence of outliers, and this will be demonstrated by SVESMs with quadratic loss function in the simulation. For the robustness against outliers, the ε-insensitive loss function or Huber loss function can be applied, the created predictor will be less sensitive to the outliers. In Huber and ε-insensitive loss function, the parameters μ and ε specify where the linear loss begins so that model is robust against outliers. When ε-insensitive loss function is used, the solution can be sparsified, and sparseness of the solution can be controlled by the insensitive parameter ε. Big reservoir and large C mean a complex model. If the reservoir or C is larger than necessary, and model tends to overfit the problem data. For a fixed reservoir, the model complexity can be controlled by the tuning of regularization parameter C. The techniques such as cross-validation, bootstrap or Bayesian method can be applied to determine which values are proper for these parameters.

By contrast, traditional RNNs suffer from many problems, such as weights estimation, outlier suppression and generalization capability control. First of all, in the training of traditional RNNs, one usually deals with the problem how the gradient is calculated and approximated in the presence of recurrent loops. The time complexity of these algorithms is usually between $O(N^2)$ and $O(N^4)$ for each data point [47], where N is the number of network nodes. Existing gradient-based algorithms usually use pure gradient-descent in weights updating, and the learning process may be slow. It should be noted that the weight parameters optimization of traditional RNNs are inherently nonconvex, and the global minimum is difficult to find. Secondly, the objective function of traditional RNNs is usually in the sense of least square, and the learned model may be very sensitive to outliers in the training examples. Finally, there are few cases in literatures where the regularization terms are introduced into the objective function of traditional RNNs, and the role of regularization has been proven effective for neural network model complexity control.

4. Simulation examples

In this section, we will give three examples to show the performance of the direct prediction method based on the proposed method. The first example is the Mackey-Glass benchmark testing, and we are particularly interested in the 84-step prediction performance. The investigation is also conducted into how the predictor is influenced by noise and outliers. The SVESM is also applied to real life problems such as the monthly sunspots prediction problem and yearly runoff time series of the Yellow River. For the presence of noise (such as monthly sunspots or yearly runoff time series), we perform a simple noise reduction method to preprocess the raw data before modeling and predicting the time series. The process of noise reduction can keep the chaotic characteristics of the raw data as many as possible and reduce significant noise and outliers [48]. Refer to [49], [50] for the practice of noise reduction technique in hydrological time series.

The simulation is conducted in the MATLAB environment running on HP Workstation xw8000, and the quadratic programming is implemented by MOSEK toolbox [51] with the MATLAB interface. Both the SVESM and SVM in the following simulation use the MOSEK package for quadratic programming. The package has the implementation of interior-point optimizer.

4.1 Multiple-Step-ahead Prediction of the Mackey-Glass Time series

The Mackey-Glass system is a time-delay differential system with the form:

$$\frac{dx}{dt} = \beta x(t) + \frac{\alpha x(t-\delta)}{1 + x(t-\delta)^{10}} \tag{30}$$

where $x(t)$ is the value of time-series at time t. The system is chaotic for d >16.8. The parameter values are chosen as β =−0.1, α =0.2 and δ=17. The data set is constructed using second-order Runge-Kutta method with a stepsize of 0.1.

The embedded data vector consists of four values of the time series:

$$\mathbf{d}(k) = \begin{bmatrix} x(k) & x(k-\tau) & x(k-2\tau) & x(k-3\tau) \end{bmatrix}^T$$

where τ=6 in the simulation. The target output is the 84-step-ahead value ($y_d(k)=x(k+84)$) of the time series.

In this simulation, the prediction performance is measured by the root mean squared error on the test sequence airs normalized by the variance of the original time series (NRMSE).

$$\textbf{NRMSE} = \sqrt{\frac{\sum_{i=1}^{T_n} (y_{\mathrm{d}}(i) - y(i))^2}{T_n \cdot \sigma^2}}$$

where $y_{\mathrm{d}}(i)$ denotes the 84-step target value, $y(i)$ is the corresponding prediction output, T_n is the number of the test examples, and σ^2 is the variance of the original time-series.

In practice, time series often contain considerable noise and outliers, and they influence the prediction in two manners. First of all, the accuracy of the predictor is influenced by noisy training examples; secondly, the prediction is influenced by the noisy test input when the predictor is practically applied to real prediction task.

In this section, three experiments will be conducted to test the performances of SVESMs. In the first experiment, the classical 84-step prediction problem (noiseless time series) is tested on the SVESMs, and several previous results are collected for comparisons.

The aim of the second experiment is to show how the ESNs predictors are influenced by the training examples polluted by noise (and outliers). The test sequences are noiseless. In the experiment, the ESNs iterative predictor [15] and direct predictor (by SVESMs) are compared.

In the last experiment, we try to simulate the realistic situation, where both the predictor and the test input are influenced by noise, and the results are compared with those of SVMs with different loss functions.

SVESM trained and tested by noiseless Mackey-Glass time series

The size of the reservoir is 700×700 and 1200×1200, and the parameter settings of the reservoir are listed in Table 1. Length of training sequence pairs $\{\mathbf{d}(k), y_{\mathrm{d}}(k)\}$ is 2200. The first 200 steps are discarded to wash out initial transient, and echo states $\mathbf{x}(k)$ are sampled from remaining 2000 steps. Since the time series is noiseless, only the quadratic loss function is applied. A noiseless validation data set (data set length is 100) is also created. The NRMS error is measured by a new testing data set (700 samples).

[Table 1]

[Fig.3]

At first, the direct predictor is tried to be implemented by the standard least square method (Eq.(3)), and the solution is far from satisfying in the experiment. It is because the matrix $\mathrm{X}^T\mathrm{X}$ is singular that regularization is needed. In the SVESM implement, the regularization term is included in the loss function and the parameter C is carefully chosen by cross-validation. By SVESM, the NRMS error on the test sequence is about 0.0159 (700×700 reservoir) and 0.0085

(1200×1200 reservoir). Fig.3 shows the 84-step prediction curve (a) and prediction error (b) (using 700×700 reservoir). Several previous results [10], [52] are collected in Table 2. Though the direct comparisons on different training sets and parameter configurations may be unfair, the result is still convincing. On the similar length of the training sequences, the SVESM shows comparable prediction performance to the previous methods except for the iterative predictor in [15].

[Table 2]

ESNs predictors influenced by noise and outliers

The second experiment is conducted to investigate how the ESNs predictors are influenced by noise and outliers. Refer to [4], [7], [11] for other studies of neural network predictors trained by noisy time series. The influence can be simply measured by feeding the noiseless input to the predictor and calculating the test error between the target time series and the corresponding predicted values. The size of reservoir is chosen as 700×700 for both iterative and direct predictors. The parameter settings of the reservoir are listed in Table 1.

Length of the training sequence is 1200 (first 200 examples will be discarded to wash out initial transient). Zero mean Gaussian noise is added to the original Mackey-Glass time series, and the noise level (ratio between standard deviation of noise and signal standard deviation) is 20%, and this time series is denoted by **MG_N**. Outliers are further added to the noisy time series to create new training sequence, and it is denoted by **MG_NAO**. There are three outliers in the training sequences, and Fig 4 describes the time series polluted both by noise and outliers (**MG_NAO**).

The length of validation set is 500. The validation set is still noisy (noise level is 20%), because the noiseless test data set is not available before the predictor is created.

[Fig.4]

[Fig.5]

For the iterative predictor, the output weights are determined by ridge regression, and the regularization parameter is chosen by cross validation. The prediction performances are tested on noiseless test sequences. The NRMS errors for different prediction horizons are calculated for the iterative predictor. Fig.5 gives the NRMS error of the ESNs iterative predictor, and the error is averaged over 80 runs of random training and testing. In Fig.5, the solid curve is the NRMS error produced by the predictor trained by **MG_N**, and the dotted curve is the NRMS error obtained by the predictor trained by **MG_NAO**. Compared with the noiseless prediction reported in [15], the result about the noisy time series is poor. Both the noise and outliers have a strong impact on the predictor accuracy. For multi-step-ahead prediction, large error is acquired because of the iteration on the inaccurate one-step predictor. For the performance deterioration, we would like to cite [11] as another example for the uncertainty propagation through an iterative predictor. In fact, the iteration on inaccurate predictor often leads to the unexpected attractors, such as a limit cycle.

[Table 3]

For the direct predictor implemented by SVESMs, the three loss functions are used, respectively, and the parameters such as C, ε and m are chosen by cross validation. For a clear comparison with the iterative predictor, the 84-step prediction results of SVESMs are obtained. The SVESM is also trained by the same time series containing noise (and outliers), and the prediction performance is tested over a separate noiseless time series with the length of 700. The comparison results are listed in Table 3. Although the performances of SVESMs are also heavily influenced by the noise and outliers, it is much better than that of the ESNs iterative predictor.

Both the predictor and the test input are influenced by noise

The objective of the third experiment is to investigate the prediction performance of SVESMs when both the predictor and the test input are influenced by noise. The SVESM predictors are the results of the second experiment, the only difference is that the prediction performance is measured by feeding the noisy input to the predictor and calculating the test error between the target time series and the corresponding predicted values. The noise level in the test input is 20%, which is the same as in the training sequences.

SVMs are also trained for the same direct prediction task. Since there is no initial transient to wash out, SVMs use all the 1200 training examples. Like in SVESMs, the three loss functions are also all applied. The kernel functions include the Gaussian and Polynomial kernel, and the parameters in the kernel functions are chosen by cross-validation. The prediction results are listed in Table 4.

As for the noisy time series, similar conclusions can be drawn from for SVESMs and SVMs. The predictors trained by the Quadratic loss function have the worst prediction results among the three loss functions. Better results can be obtained by the ε-insensitive or Huber loss function. As for the robustness, the predictors trained by the Quadratic loss function are very sensitive to the outliers in the training examples, and performance deterioration is considerable. However, the robust loss functions such as ε-insensitive or Huber loss function indicate the insensitivity to the outliers. From the results of the comparisons, SVESMs give better results than SVMs in the experiments.

[Table 4]

Noise and outliers limit the accuracy of prediction methods especially the ESNs iterative predictor. In the framework of SVESM, several robust loss functions can be used so that the model can take advantage of the robustness. SVESMs are a kind of recurrent neural networks, which have a different information processing style from its feedforward counterpart such as SVMs. The damping properties of RNNs (by reservoir) can alleviate the deleterious effect of noisy input, however, the SVMs direct transfer the effect of the noisy input to the predicted output.

In practical applications, the effects of noise or outliers can also be alleviated by the nonlinear noise reduction techniques. So the noise reduction is another source of performance improvements as well as finding a good model.

4.2 Forecasting the Monthly Sunspots Time Series

The sunspot number data used in the paper are monthly mean Wolf sunspots numbers. Like [14] and [53], we use the noise-reduced sunspots numbers for the prediction task. For the monthly sunspots number, in our simulation, $\mathbf{d}(k)$ is chosen as:

$$\mathbf{d}(k) = \left[x(k) \quad x(k-\tau) \quad x(k-2\tau)\right]^T,$$

where τ=10 in the simulation, the target output is the 6-month, 10-month, 15-month and 20-month ahead values (y_d is $x(k+6)$, $x(k+10)$, $x(k+15)$ and $x(k+20)$, respectively) of the sunspots time series so that it is compared with the previous results in [14] and [53]. The time series are rescaled to the range of (-1, 1) before presented to the networks.

The size of the reservoir is 600×600, and the parameter settings of the reservoir are listed in Table 5. Length of training sequence pairs$\{\mathbf{d}(k), y_\mathrm{d}(k)\}$ is 1000, first 100 steps are discarded to wash out initial transient, echo signals $\mathbf{x}(k)$ are sampled from the remaining 900 steps.

[Table 5]

In this section, the three kinds of loss functions are all used for the comparisons for the modeling and prediction. For the quadratic loss function, only the regularization parameter C need to be determined, and for the ε-insensitive and Huber loss function, the additional parameters are ε and μ, respectively. For different prediction horizons, the parameters C, ε and μ are determined by cross validation. Data points from 1001 to 1300 are used as the validation set. The prediction performance is measured by the root mean squared (RMS) error on the test sequences (data points >1300).

SVMs are also tested on the same prediction task. SVMs use all the 1000 training examples. The three loss functions are also applied like SVESMs. The kernel functions used include RBF and Polynomial kernel, and the parameters in the kernel functions are chosen by cross-validation. Both the prediction results of SVESMs and SVMs are collected in Table 6.

Existing results of RPNN and ULN for the same prediction tasks are also presented.

Among the three loss functions, the best prediction result is given by ε-insensitive loss function for the same prediction horizons. For example, for 10-month-ahead prediction, the RMS errors are 3.329, 3.560 and 3.548 by ε-insensitive, Quadratic and Huber loss function in the framework of SVESMs, respectively. And for SVMs, the corresponding RMS errors are 6.415, 6.948 and 6.815 for RBF kernel, and 6.451, 6.620 and 6.617 for Polynomial kernel, respectively. For RPNN and ULN, the RMS errors are 7.050 and 10.584, respectively. Since Quadratic loss function is usually used in the traditional RNNs, the results of robust loss function such as ε-insensitive are not available for RPNN or ULN. Both SVMs and SVESMs can find their optimal parameters in the convex optimization, and the model complexity can be properly controlled by the regularization, and it is why it exceeds the gradient-based learning method such as RPNN or ULN.

Both in SVMs and SVESMs, the best results are obtained by ε-insensitive loss function, and the results of Huber loss function are followed. On the whole, the robust loss functions always have better performance than the quadratic loss function in SVMs and SVESMs. In the simulation, SVMs and SVESMs use the same QP optimizer and loss functions, however, SVESMs obtain better results than SVMs, and this can be explained by the modeling power of reservoir and the merits of RNNs. There are many recurrent connections in the reservoir, and the output of SVESMs relies on not only the current input vector, but also the history input vectors, which makes it less sensitive to the noise in input vectors. However, SVMs directly compute the output via a kernel. So noise in input vector will have less influence on "reservoir" than "kernel". It is also observed that the results of Quadratic loss function is not so bad when compared with those of robust loss function in SVMs and SVESMs, and this can be explained by the positive roles of the noise reduction procedure.

[Table 6]

Fig.6 shows the 10-month-ahead prediction results based on the SVESMs with ε-insensitive loss function, and we are particularly interested in the parameter setting of this loss function, so we make an investigation to the influences of the parameter ε on the prediction performance and the sparseness. A group of simulations are done with different ε values while keep the parameter *C* and prediction horizon constant. As shown in Fig.7(a), with the ε increases, the test error firstly decreases gradually and then grows up rapidly around ε=0.007, at the same time the sparseness decreases from 93% to about 36% monotonously as shown in Fig.7(b). The result is consistent with the discussions in [8]. It is obvious that the best result is gained when ε=0.005, because any more improvement in sparseness will result in the deterioration of the prediction performance.

In our simulations, the iterative prediction method [15] fails to generate a proper chaotic attractor as expected. Also, the least square method using Eq.(3) fails to make a proper prediction and it is proven that the additional constraints of the weight decay term in the objective function is necessary.

[Fig.6]

[Fig.7]

4.3 Forecasting the Yearly Runoff Time Series of the Yellow River

As the second longest river of China, the Yellow River is important to the people life, so the runoff time series forecasting for the Yellow River is necessary. As shown in literatures of hydrology, runoff time series of many rivers indicate nonlinear or chaotic properties, so the nonlinear or chaos theory can be an aid to forecast. We use the noise-reduced yearly runoff time series of the Yellow River. In the simulation, $\mathbf{d}(k)$ is chosen as:

$$\mathbf{d}(k) = \left[x(k) \quad x(k-1) \quad \ldots \quad x(k-5)\right]^T ,$$

where τ=1 in the simulation, the target outputs are 1-year, 2-year ahead values (y_{d} is $x(k+1)$ and $x(k+2)$, respectively, note that most literatures deal with 1-step prediction, because the accuracy prediction for runoff time series is difficult).

The size of the reservoir is 200×200, and the parameter settings of the reservoir are listed in Table 7. Length of training sequence pairs $\{\mathbf{d}(k), y_{\mathrm{d}}(k)\}$ is 450, first 50 steps are discarded to wash out initial transient, and echo signals $\mathbf{x}(k)$ are sampled from remaining 400 steps. We use the examples of 450~500 as a validation set, and the remaining about 20 samples as the testing set (totally about 520 data points).

[Table 7]

All the three loss functions are used for the comparisons for the modeling and predicting. The parameters such as C, ε and μ are chosen by cross validation. The prediction performance is measured by the root mean squared (RMS) error on the test sequence pairs. Table 8 lists the prediction errors on the test sequences.

[Fig.8]

[Table 8]

SVMs are also tested on the same prediction task for a comparison. The three loss functions are also applied like SVESMs. The kernel functions include RBF and Polynomial kernel, and the parameters in the kernel functions are also chosen by cross-validation. The prediction results are also collected in Table 8.

Among the three loss functions, the worst prediction result is given by Quadratic loss function for both SVESMs and

SVMs. For SVESMs, the RMS error for 1-year-ahead prediction is 24.422 when Quadratic loss is used, and results of ε-insensitive and Huber function are 20.682 (with sparseness of 55%) and 20.429, respectively. For SVMs trained by Quadratic loss function, the RMS errors for 1-year-ahead prediction is 29.691 (RBF kernel) and 26.738 (Polynomial kernel), and results of ε-insensitive and Huber function have relatively better results, for example, if Polynomial kernel is used, the RMS errors of ε-insensitive and Huber function are 20.671 and 20.776, respectively. The 1-year-ahead prediction and 2-year-ahead prediction error obtained by RPNN is 37.920 and 50.720, respectively [54]. Fig.8 is 1-year-ahead prediction curve generated by SVESM (with Huber loss function).

Because the raw time series is short (less than 600) and noisy, and the method of nonlinear noise reduction has limited performance, the noise reduced time series may still have considerable noise or outliers, and this leads to the significant performance deterioration of the quadratic loss function when compared with the robust loss functions, which deal with the outliers in a linear rather than a superlinear style.

For the prediction of the yearly runoff time series of the Yellow River, it is found that larger spectral radius will deteriorate the modeling performance of SVESMs, and the spectral radius is kept small (about 0~0.1). When the spectral radius is small, it becomes a mapping with weak transient property. In this case, SVESMs bridge the static mapping method and the dynamic mapping method. In fact, if there is no feedback in the reservoir (spectral radius of $\mathbf{W}_x$ will be 0), the training of SVESMs will be the training of MLPs (input weights are fixed). It can be an explanation why the SVESMs have the similar performances with SVMs.

For the runoff time series, the iterative prediction method [15] fails to generate a proper chaotic attractor as expected in our simulation. Also, the standard least square methods fails to make a proper prediction because of the ill-conditioned data, so it is necessary to use ridge regression or linear support vector machine to improve the performance of solution.

4.4 Practical considerations of the parameter settings for the SVESM modeling

Up to now, the SVESM modeling involves the selection of following parameters: C, ε and μ for the linear SVMs, embedding dimension m and delay time τ, and the reservoir parameters such as the spectral radius. Firstly, the C, ε and μ can follow the traditional parameter tuning method [55] such as cross validation. Secondly, the embedding dimension and the delay time can be chosen in a wide range. According to Takens' theorem, the original dynamic properties can be retained under the topological transform, as long as the embedding dimension $m \geq 2d+1$ (d is the correlation dimension of the attractor). But this is only a sufficient but not necessary condition. In our simulation, for example, for Mackey-Glass time series, the embedding dimension m can be 3, 4, 5, 6, 7, … and the delay time τ can be 1, 2, 3, 4, …; for sunspots time series the embedding dimension m can be 2, 3, 4, 5, … and the delay time τ can be 1, 2, 3, …; for the runoff time series, the embedding dimension m is not necessary to be 6. For a large range of the embedding parameters, the prediction performance of SVESM does not deteriorate, and sometime it can improve to some degree.

The issue of the reservoir parameters selection is an important thing that should be paid special attention to in the SVESM modeling. The suggestion for the preparation of the reservoir can refer to [25], [31]. However, what we would like to highlight is the control of the transient property by using the size of the spectral radius. Generally speaking, the spectral radius falls in the range of 0~1. The larger the spectral radius is, the longer the transient process will persist in, and the best selection of the spectral radius should be adapted to different problems. It seems that the selection of the spectral radius is easier than the kernel parameters in the traditional SVM modeling (such as width parameters in the radial basis kernel function), because the spectral radius is a parameter within a fixed range. It is suggested to try different spectral radius for a given problem, for example, try the spectral radius as 0.3, 0.6 and 0.9, and find the best and refine the spectral radius like a line search procedure.

5. Conclusions

For chaotic time series prediction, a novel direct prediction method based on echo state mechanisms and support vector machine is proposed. The basic idea is replacing the “kernel trick” with “reservoir trick” in nonlinear system modeling, that is to say, performing linear support vector regression in the high dimension “reservoir” state space. The objective function of SVESMs is convex, so the solution is optimal and unique. SVESMs belong to RNNs and benefit from the advantages of SVMs. SVESN is especially efficient in dealing with real life time series, and the generalization ability and robustness is obtained by the regularization operator and robust loss function. The basic idea of our direct method is to use m-dimension embedded data vectors ($\mathbf{d}(k)$) and its history information to predict the h-step-ahead value $x(t+h)$ directly. We have proven that there exists a dynamic system whose input is the m-dimension embedded data vectors ($\mathbf{d}(k)$) at time k, and output is the h-step target value ($x(k+h)$). Since the sequences pairs$\{\mathbf{d}(k), x(k+h)\}$ are the input and output pairs of a dynamic system, so the direct prediction problem can be converted into a dynamic system modeling task, and the SVESMs can be applied.

Benefiting from the echo state mechanisms and support vector machine, the proposed method has the following features:

- Easy to train, optimal and unique solution and no local minimum problem
- Sparse modeling, Improved generalization ability and high prediction accuracy

- Insensitivity to uncertainness and robustness against outlier

The method has been tested on both the benchmark problem (Mackey-Glass time series) and real life time series (Wolf monthly sunspots number) and yearly runoff time series of the Yellow River. The prediction result is satisfying.

Acknowledgments

This research is supported by the project (60374064) of the National Nature Science Foundation of China, and this support is appreciated. The author would like to thank the Associate Editor and three anonymous referees for all their detailed comments and suggestions.

References

[1] J.C. Principe, A. Rathie, and J.M. Kuo, "Prediction of Chaotic Time Series with Neural Networks and the Issue of Dynamic Modeling", *Int. J. Bifurcation and Chaos*, vol. 2, pp. 989-996, 1992.

[2] A. Lapades, and R. Farbar, How neural nets work, in *Advances in Neural Information Processing Systems*, pp. 442-456, 1987

[3] M. Casdagli, "Nonlinear prediction of chaotic time series," *Physica D*, vol. 35, pp. 335-356, 1989.

[4] S. Haykin, J. Principe, Making sense of a complex world, *IEEE Signal Processing Magazine*, vol. 15, no.3, pp.66-68,1998.

[5] M.R. Cowper, B. Mulgrew, and C. P. Unsworth, "Nonlinear prediction of chaotic signals using a normalized radial basis function network," *Signal Processing*, vol. 82, no.5, pp. 775-789, 2002.

[6] E.A. Wan, "Time series prediction by using a connectionist network with internal delay lines (data set A)," in *Proceedings of the NATO Advanced Research Workshop on Comparative Time Series Analysis, May 14-17 1992*, pp. 195-217, 1993.

[7] K.-R. Muller, A. J. Smola, G. Ratsch, B. Scholkopf, J. Kohlmorgen, and V. N. Vapnik, "Predicting time series with support vector machines" in the Proceedings of the 1997 7th International Conference on Artificial Neural Networks, ICANN'97, Oct 8-10 1997," *Lecture Notes in Computer Science*, vol. 1327 ed. Lausanne, Switz, 1997, pp. 999-1004.

[8] S. Mukherjee, E. Osuna, and F. Girosi, "Nonlinear prediction of chaotic time series using support vector machines," in the *Proceedings of the 1997 7th IEEE Workshop on Neural Networks for Signal Processing*, NNSP'97, Sep 24-26 1997, Amelia Island, FL, USA, 1997.

[9] L. J. Cao and F. E. H. Tay, "Support vector machine with adaptive parameters in financial time series forecasting," *IEEE Transactions on Neural Networks*, vol. 14, no.6, pp. 1506-1518, 2003.

[10] J. Vesanto, Using the SOM and local models in time-series prediction. In *Proceedings of WSOM'97*, *Workshop on Self-Organizing Maps*, Espoo, Finland, June 4-6, pp. 209-214, 1997.

[11] A. Girard, C. E. Rasmussen and J. Qui, Gaussian Process Priors with Uncertain Inputs – Application to Multiple-Step Ahead Time Series Forecasting, in *Advances in Neural Information Processing Systems 15,* pp.529-536, Cambridge, MA: MIT Press, 2003.

[12] J.C. Principe, J.M. Kuo, Dynamic modeling of chaotic time series with neural networks, *Advances in Neural Information Processing Systems 7*, pp.311-318, Cambridge, MA: MIT Press, 1995.

[13] J. Zhang, K. S. Tang, and K. F. Man, "Recurrent NN model for chaotic time series prediction," in *Proceedings of the 1997 23rd Annual International Conference on Industrial Electronics, Control, and Instrumentation, IECON. Part 3 (of 4), Nov 9-14 1998*, Vol.3, pp. 1108-1112, 1997.

[14] M. Han, J. H. Xi, S. G. Xu and F. L. Yin, Prediction of chaotic time series based on the recurrent predictor neural network, *IEEE Transactions on Signal Processing*, vol. 52, no.12, pp.3409-3416, 2004.

[15] H. Jaeger, and H. Haas, Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication, *Science,* Vol.304, No.5667, pp.78-80, 2004.

[16] A.F. Atiya, S. M. El-Shoura. A comparison between neural-network forecasting techniques-case study: river flow forecasting. *IEEE Transactions on Neural Networks*, vol. 10, no. 2, pp.402-409, 1999.

[17] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1998.

[18] C. Cortes and V. Vapnik, "Support-Vector Networks," *Machine Learning*, vol. 20, no.3, pp. 273-297, 1995.

[19] V. N. Vapnik, "Overview of statistical learning theory," *IEEE Transactions on Neural Networks*, vol. 10, no.5, pp. 988-999, 1999.

[20] V. Cherkassky and F. Mulier, "Vapnik-Chervonenkis (VC) learning theory and its applications," *IEEE Transactions on Neural Networks*, vol. 10, no.5, pp. 985-987, 1999.

[21] V. N. Vapnik, S.E. Golowich, and A. J. Smola, “Support vector method for function approximation, regression estimation and signal processing,” in *Advances in Neural Information Processing Systems 9*, pp.281-287, Cambridge, MA: MIT Press, 1996.

[22] A. J. Smola and B. Scholkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, no.3, pp. 199-222, 2004.

[23] J. A. K. Suykens and J. Vandewalle, "Recurrent least squares support vector machines," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 47, no.7, pp. 1109-1114, 2000.

[24] H. Jaeger, Adaptive Nonlinear System Identification with Echo State Networks, in *Advances in Neural Information Processing Systems 15*, pp:593-600, Cambridge, MA: MIT Press, 2003.

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

[26] J.A.K. Suykens and J. Vandewalle, "Learning a simple recurrent neural state space model to behave like Chua's double scroll," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 42, no.8, pp. 499-502, 1995.

[27] M. Han, Z. W. Shi, and W. Wang, "Modeling dynamic system by recurrent neural network with state variables," in *Advances in Neural Networks - ISNN 2004, Pt 2*, vol. 3174, *Lecture Notes in Computer Science*. Berlin: SPRINGER-VERLAG BERLIN, 2004, pp. 200-205.

[28] J.M. Zamarreno, and P. Vega. "State space neural network. Properties and application," *Neural Networks*, vol.11, no.6, pp. 1099-1112, 1998.

[29] Z.W. Shi, M. Han and J.H. Xi, Exploring the neural state space learning from one-dimension chaotic time series, in *Proceedings of IEEE international conference on Networking, Sensing and Control*, Tucson, Arizona, U.S.A, pp.437-442, 2005.

[30] F. Takens, Detecting Strange Attractors in Fluid Turbulence, in D. Rand and L.-S. Young(eds.) *Dynamical Systems and Turbulence*, Berlin, Springer, 1981.

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

[32] M. C. Ozturk and J. C. Principe, "Computing with transiently stable states," *Proceedings of the International Joint Conference on Neural Networks* 2005, pp. 1467-1472, vol. 3, 2005.

[33] M. Buehner and P. Young, "A tighter bound for the echo state property," *IEEE Transactions on Neural Networks*, vol. 17, no. 3, pp. 820-824, 2006.

[34] A. N. Tikhonov and V. Y. Arsenin. *Solution of Ill--Posed Problems*. Winston, Washington, DC, 1977.

[35] R. Reed, "Pruning Algorithms - a Survey," *IEEE Transactions on Neural Networks*, vol. 4, no.5, pp. 740-747, 1993.

[36] T.N. Lin, C. L. Giles, B. G. Horne, and S. Y. Kung, "A delay damage model selection algorithm for NARX neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no.11, pp. 2719-2730, 1997.

[37] S.E. Fahlman, and C. Leibiere, "The cascade-correlation learning architecture", In *Advances In Neural Information Processing Systems*, vol. 2, pp. 524--532, 1990.

[38] M. Lehtokangas, "Modified cascade-correlation learning for classification," *IEEE Transactions on Neural Networks*, vol. 11, no.3, pp. 795-798, 2000.

[39] K.-R. Muller, S. Mika, G. Ratsch, K. Tsuda, and B. Scholkopf, "An introduction to kernel-based learning algorithms," *IEEE Transactions on Neural Networks*, vol.12, no.2, pp. 181-201, 2001.

[40] K. Ikeda, "Effects of kernel function on Nu support vector machines in extreme cases," *IEEE Transactions on Neural Networks*, vol.17, no.1, pp. 1-9, 2006.

[41] I. W.-H. Tsang and J. T.-Y. Kwok, "Efficient hyperkernel learning using second-order cone programming," *IEEE Transactions on Neural Networks*, vol.17, no.1, pp. 48-58, 2006.

[42] H. Xiong, M. N. S. Swamy, and M. O. Ahmad, "Optimizing the kernel in the empirical feature space," *IEEE Transactions on Neural Networks*, vol.16, no.2, pp. 460-474, 2005.

[43] 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*, vol.13, no.6, pp. 1322-1330, 2002.

[44] W. Chu, S. S. Keerthi, and C. J. Ong, "Bayesian support vector regression using a unified loss function," *IEEE Transactions on Neural Networks*, vol.15, no. 1 pp. 29-44, 2004.

[45] L. J. Cao, S. S. Keerthi, C.-J. Ong, J. Q. Zhang, U. Periyathamby, X. J. Fu, and H. P. Lee, "Parallel sequential minimal optimization for the training of support vector machines," *IEEE Transactions on Neural Networks*, vol.17, no.4, pp. 1039-1049, 2006.

[46] I. W. Tsang, J. T. Kwok, and P.-M. Cheung, "Core vector machines: Fast SVM training on very large data sets," *Journal of Machine Learning Research*, vol. 6, pp. 363-392, 2005.

[47] A. F. Atiya and A. G. Parlos, "New results on recurrent network training: unifying the algorithms and accelerating convergence," *IEEE Transactions on Neural Networks*, vol. 11, no.3, pp. 697-709, 2000.

[48] H. Kantz, T. *Schreiber*, *Nonlinear time series analysis*, Cambridge University Press, 1997.

[49] B. Sivakumar, K. K. Phoon, S. Y. Liong, and C. Y. Liaw, "A systematic approach to noise reduction in chaotic hydrological time series," *Journal of Hydrology*, vol. 219, no.3-4, pp. 103-135, 1999.

[50] A. W. Jayawardena and A. B. Gurung, "Noise reduction and prediction of hydrometeorological time series: dynamical systems approach vs. stochastic approach," *Journal of Hydrology*, vol. 228, no.3-4, pp. 242-264, 2000.

[51] MOSEK Aps, The MOSEK optimization toolbox for MATLAB version 3.2 (Revision 8) User’s guide and reference manual. 2004.

[52] X. Yao and Y. Liu, "New evolutionary system for evolving artificial neural networks," *IEEE Transactions on Neural Networks*, vol. 8, no.3, pp. 694-713, 1997.

[53] K. Jinno, S. G. Xu, A KAWAMURA and M. MATSUMOTO, Prediction of Sunspots Using Reconstructed Chaotic System Equations. *Journal of Geophysical Research-Space Physics*, vol.100, no.A8, pp.14773-14781, 1995.

[54] J.H. Xi, Study on Long-term Prediction Technology of Chaotic Time Series, PhD thesis, Dalian University of Technology, 2005

[55] V. Cherkassky and Y. Ma, "Practical selection of SVM parameters and noise estimation for SVM regression," *Neural Networks*, vol. 17, no.1, pp. 113-126, 2004.

List of tables

Table 1 Parameter setting of the reservoir for the Mackey-Glass time series

Items	Values
Size of the reservoir	700×700 and 1200×1200
Sparseness of $\mathbf{W}_x$	2%
Spectral radius of $\mathbf{W}_x$	0.98
Input weight connections	Uniform over(-0.25 0.25)

Table 2 NRMS errors by various methods in predicting the $x(k+84)$ of the noiseless Mackey-Glass time-series

TYPE	DESCRIPTIONS	NRMS ERROR
ESNs (iterative method)	1000×1000 reservoir; Training sequence length is 3000	6.30×10^{-5}
SVESM (direct method)	700×700 reservoir; Training sequence length is 2200	0.0159
	1200×1200 reservoir; Training sequence length is 2200	0.0085
SOM	35×35 Map size; Training sequence length is 3001	0.022
SOM	35×35 Map size; Training sequence length is 1001	0.035
MLP	Training sequence length is 500	0.060

Table 3 Prediction performances (NRMSE) of ESNs (iterative method) and SVESM (direct method) when the predictor is influenced by noise and outliers

Model	**ESNs (Iterative Predictor)**	**SVESM(Direct Predictor)**		
Loss	**Quadratic**	**Quadratic**	**ε-insensitive**	**Huber**
Trained by MG_N	0.3769	0.2206	0.2062	0.2056
Trained by MG_NAO	0.7743	0.2802	0.2231	0.2235

Table 4 Prediction performances (NRMSE) of SVESMs and SVMs when the predictors are influenced by noise (and outliers) and the test examples input contains noise

Model	**SVESM**			**SVMs**					
Loss	**Quadratic**	**ε-insensitive**	**Huber**	**Quadratic**		**ε-insensitive**		**Huber**	
Kernel	–	–	–	**RBF**	**POLY**	**RBF**	**POLY**	**RBF**	**POLY**
Trained by MG_N	0.2778	0.2636	0.2646	0.3162	0.3403	0.3061	0.3242	0.3067	0.3254
Trained by MG_NAO	0.3260	0.2703	0.2684	0.3400	0.3601	0.3083	0.3257	0.3078	0.3274

Table 5 Parameter settings of the reservoir for the monthly sunspots time series

Items	Values
Size of the reservoir	600×600
Sparseness of $\mathbf{W}_x$	2%
Spectral radius of $\mathbf{W}_x$	0.75
Input weight connections	Uniform over (-0.25 0.25)

Table 6 RMS errors for monthly sunspots time series using SVESMs with different loss functions and the comparison with previous results

Model	**SVESM**			**SVMs**						**RPNN**	**ULN**
Loss	**Quadratic**	**ε-insensitive**	**Huber**	**Quadratic**		**ε-insensitive**		**Huber**		**Quadratic**	**Quadratic**
Kernel	–	–	–	**RBF**	**POLY**	**RBF**	**POLY**	**RBF**	**POLY**	–	–
6-m	1.026	0.982	1.121	3.659	3.485	3.335	3.434	3.648	3.485	4.419	5.816
10-m	3.560	3.329	3.548	6.948	6.620	6.415	6.451	6.815	6.617	7.050	10.584
15-m	7.944	7.308	7.818	11.468	11.207	10.921	10.658	10.960	11.179	13.658	19.613
20-m	12.622	11.717	12.431	15.689	16.469	15.150	15.863	15.291	16.460	16.793	22.292

Table 7 Parameter settings of the reservoir for the yearly runoff time series of the Yellow River

Items	Values
Size of the reservoir	200×200
Sparseness of $\mathbf{W}_x$	2%
Spectral radius of $\mathbf{W}_x$	0.06
Input weight connections	Uniform over(-0.25 0.25)

Table 8 RMS errors for yearly runoff time series of the Yellow River using SVESMs with different loss functions and the comparison with previous results

Model	**SVESM**			**SVMs**						**RPNN**
Loss	**Quadratic**	**ε-insensitive**	**Huber**	**Quadratic**		**ε-insensitive**		**Huber**		**Quadratic**
Kernel	–	–	–	**RBF**	**POLY**	**RBF**	**POLY**	**RBF**	**POLY**	–
1-y	24.422	20.682	20.429	29.691	26.738	22.315	20.671	22.808	20.776	37.920
2-y	54.209	48.832	48.569	58.408	55.865	54.649	49.299	55.090	49.895	50.720

List of figures

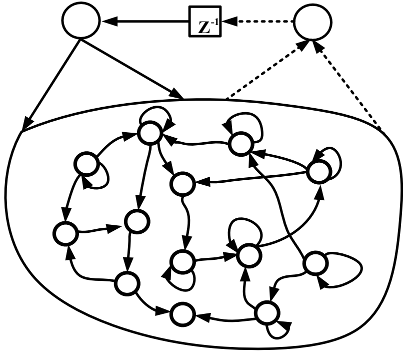


Fig.1 Iterative prediction method based on Echo State Networks

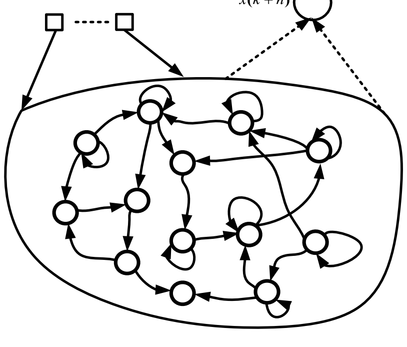
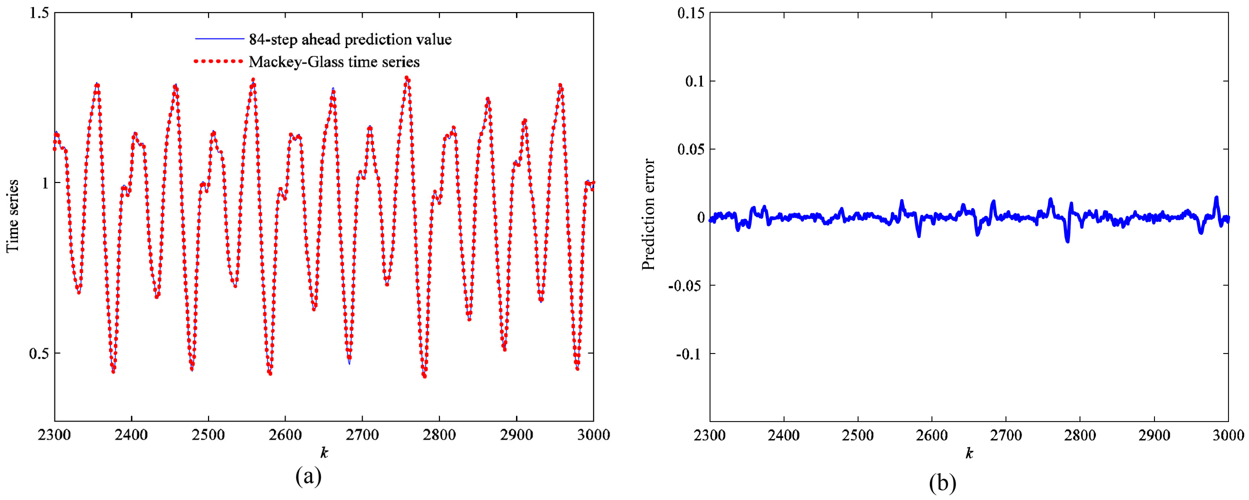


Fig.2 Direct prediction method based on Echo State Networks



(a)

(b)

Fig.3 84-step-ahead prediction for Mackey-Glass time series using direct method based on the SVESMs, (a) prediction curve (b) prediction error

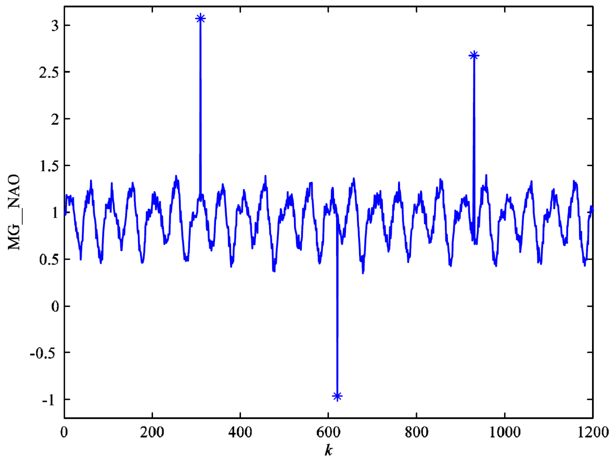


Fig.4 The noisy time series and the three outliers

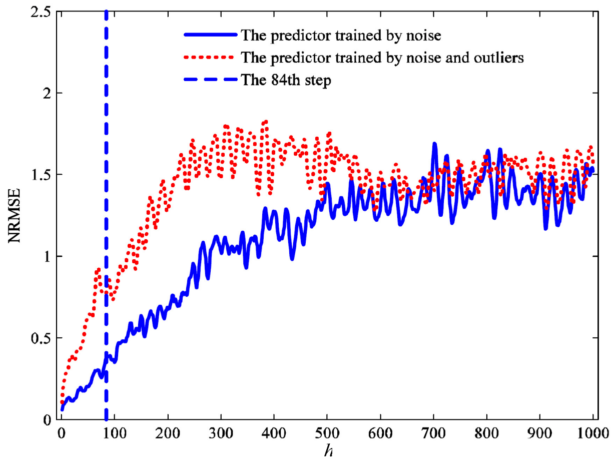
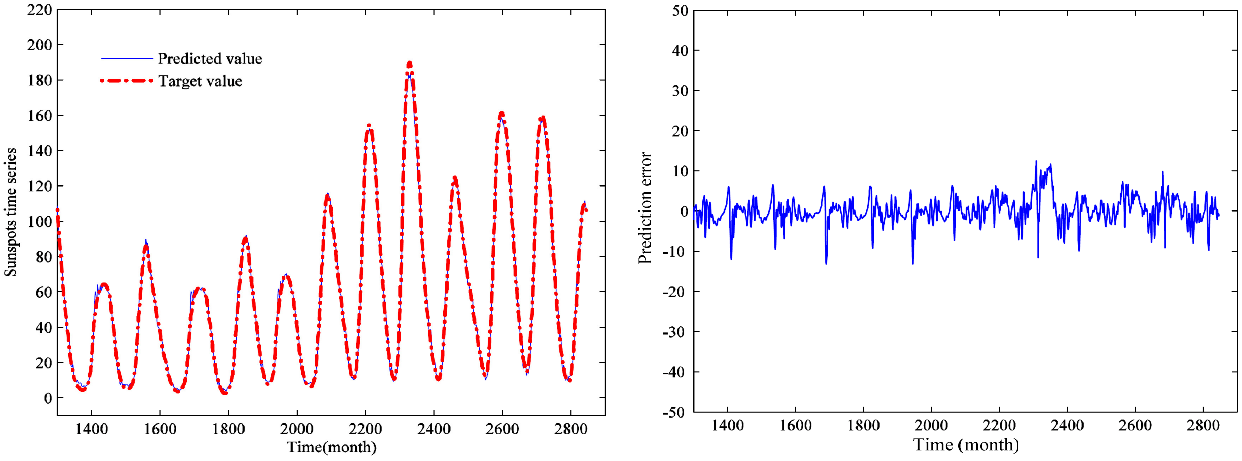


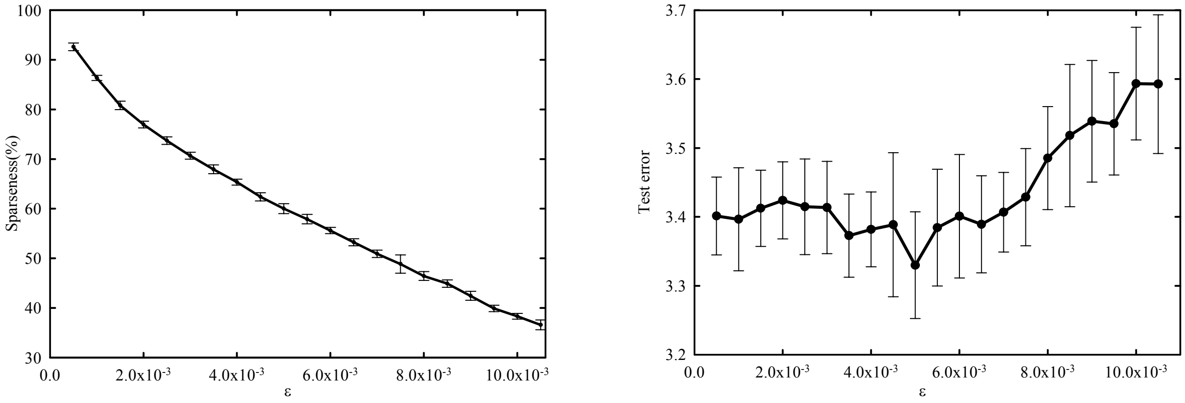
Fig.5 Average NRMS error for Mackey-Glass time series iterative prediction when the predictors are influenced by noise (noise and outliers)



(a)

(b)

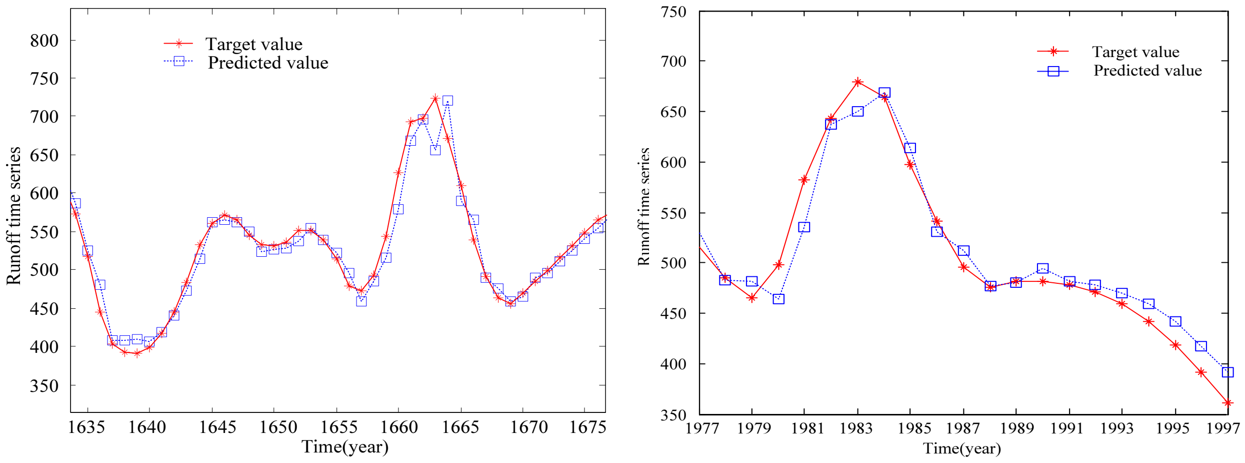
Fig.6 10-month-ahead prediction for monthly sunspots time series using direct method based on the SVESM, (a) prediction curve (b) prediction error



(a)

(b)

Fig.7 The influences of different ε parameters on the prediction results, (a) The 10-month-ahead prediction error based on SVESM with different ε-insensitive tube (b) Sparseness of the SVESM for the 10-month-ahead prediction error with different ε-insensitive tube.



(a) (b)

Fig.8 The one-year-ahead predicted value by SVESM and the target value of the runoff time series of the Yellow River, (a) Training sequence (this panel contains 40 year data) (b) Testing sequence(this panel contains 20 year data)

Zhiwei Shi was born in Luoyang, China, in 1980. He received his B.S. degree from Department of Electrical and Information Zhengzhou University in 2002. He is currently a PhD candidate in Department of Electrical Engineering at the Dalian University of Technology, China. His current research interests include recurrent neural networks, support vector machines and kernel method, chaotic time series analysis.

Min Han was born in Beijing, China, in 1959. She received her B.S. and M.S degree from Department of Electrical Engineering Dalian University of Technology respectively in 1982 and 1993, and received the M.S degree and Ph.D. degree in Kyushu University, Japan, in 1996 and 1999. She is a Professor at School of Electronic and Information Engineering, Dalian University of Technology. Her current research interests are neural network, chaos and their applications to control and identification. She is a member of IEEE.