Support vector recurrent neurofuzzy networks in modeling nonlinear systems with correlated noise

W. C. Chan, C. W. Chan, K. C. Cheung
Department of Mechanical Engineering
The University of Hong Kong
Pokfulam Road, Hong Kong, China

C. J. Harris
Department of Electronics and Computing
University of Southampton, UK

Abstract
Good generalization results are obtained from neurofuzzy networks if its structure is suitably chosen. To select the structure of neurofuzzy networks, the authors proposed a construction algorithm that is derived from the Support Vector Regression. However, the modeling errors are assumed to be uncorrelated. In this paper, systems with correlated modeling errors are considered. The correlated noise is modeled separately by a recurrent network. The overall network is referred to as the support vector recurrent neurofuzzy networks. The prediction error method is used to train the networks, where the derivatives are computed by a sensitivity model. The performance of proposed networks is illustrated by an example involving a nonlinear dynamic system corrupted by correlated noise.

Keywords: Support vectors, recurrent neurofuzzy network, sensitivity model, correlated noise.

1. Introduction

Neurofuzzy networks were shown to have the transparency of fuzzy systems and the modeling ability of neural networks [12]. However, the goodness of the approximation depends not only on the training algorithm, but also on the structure of the network, i.e., the number of basis functions and their locations in the input space. If *a-priori* knowledge of the system is available, the structure of the network can be suitably chosen from this knowledge. If, however, no *a-priori* knowledge is available, the structure of the network is usually chosen from the training data, which a well-known problem. Support Vector Regression (SVR) developed from statistical learning theory is a linear-in-weights network [5]. The structure and the parameters of the network are obtained by minimizing a cost function that consists of the modeling errors and the complexity of the network. The structure of the SVR is now selected for a given error bound from the training data, which is known as the Support Vector (SV). It is shown that the SVR can be transformed to a radial basis function (RBF) neural network [13,15]. The structure of the transformed RBF network is the same as that of the SVR obtained objectively for a given error bound.

However, the noise of the system is assumed to be a white noise [13]. In this paper, this result is extended to systems with correlated noise. To model the correlated noise, an additional recurrent neurofuzzy network is used, yielding the support vector recurrent neurofuzzy network (SVRNFN). The training of the SVRNFN is derived from the prediction error method [6]. Newton-Raphson technique is used to optimize the cost function, with the derivatives obtained from the sensitivity model.

The organization of the paper is as follows. In section 2, the modeling of dynamic systems with correlated noise is presented. A brief description of the SVR algorithm is given in section 3, from which the SVRNFN is derived in section 4. The training of the SVRNFN derived from the prediction error method is derived in section 5. The performance of the SVRNFN and the proposed training algorithm is illustrated by an example given in section 6.

2. Modelling of Nonlinear Stochastic Systems

Consider the Nonlinear Auto-Regressive Moving Average model with eXogenous input (NARMAX) given by [8,10],

$$y(k) = F(y(k-1),\ldots, y(k-m), u(k-1),\ldots, u(k-p), e(k-1),\ldots, e(k-n)) + e(k) \quad (1)$$

where $y(k)$, $u(k)$ and $e(k) \sim N(0, \sigma^2)$ are respectively the output, input and white noise; m, p and n, the respective orders; and $F(\bullet)$, a smooth nonlinear function. There is a wide class of nonlinear stochastic systems that can be modeled by NARMAX (1) around an equilibrium point subject to the following conditions [8]:

A1: a state-space realization of the system exists,
A2: a linearized model exists when operating close to the given equilibrium point.

For simplicity, it is assumed that the correlated noise is independent of $y(k)$, $u(k)$, hence (1) can be simplified to

$$y(k) = f(X(k)) + g(E(k)) + e(k) \quad (2)$$

where $f(.)$ and $g(.)$ are smooth nonlinear functions, $X(k) = [y(k-1),..., y(k-m), u(k-1),..., u(k-p)]^T$ and $E(k) = [e(k-1),..., e(k-n)]^T$. The model given by (2) is referred to as the simplified NARMAX model [10]. Since the correlated noise is bounded with a mean of zero, hence we have,

$$g(\underline{0}) = 0 \quad (3)$$

$$\exists G \in \Re^+ \quad s.t. \quad |g(E)| < G \quad \forall E \in \Re^n \quad (4)$$

3. Support Vector Regression

Rewrite (2),

$$y(k) = f(X(k)) + d(k) \quad (5)$$

where $d(k) = g(E(k)) + e(k)$ is the correlated noise. The nonlinearity $f(X(k))$ can be approximated by a Support Vector Machine (SVM) as given below [5].

$$f(X(k)) = \sum_{i=1}^{N} (\overline{\alpha}_i - \underline{\alpha}_i) K(X(k), X(i)) + b \quad (6)$$

where N is the length of data for determining the structure of $f(\bullet)$; $X(k)=[x_1(k),\ldots,x_m(k)]^T$, $m=m_y+m_u$; $K(X(k), X(i))$ is a kernel satisfying the Mercer's condition [2]; $\overline{\alpha}_i$ and $\underline{\alpha}_i$ are the Lagrange multipliers, and b is the bias. The following fourth order B-spline kernels is used [2],

$$K(X(k), X(i)) = \prod_{j=1}^{m} \frac{B_4(x_j(k) - x_j(i))}{B_4(0)} \quad (7)$$

where

$$B_4(x) = \begin{cases} (2\lambda + x)^3 / 6\lambda^3 & -2\lambda \le x < -\lambda \\ (4\lambda^3 - 6\lambda x^2 - 3x^3) / 6\lambda^3 & -\lambda \le x < 0 \\ (4\lambda^3 - 6\lambda x^2 + 3x^3) / 6\lambda^3 & 0 \le x < \lambda \\ (2\lambda - x)^3 / 6\lambda^3 & \lambda \le x < 2\lambda \\ 0 & otherwise \end{cases} \quad (8)$$

where λ is the distance between two consecutive knots. The estimate of $\overline{\alpha}_i$ and $\underline{\alpha}_i$ are obtained by minimizing the cost function,

$$L(\overline{\alpha}_i, \underline{\alpha}_i) = \frac{1}{2} \sum_{i,j=1}^{N} (\overline{\alpha}_i - \underline{\alpha}_i)(\overline{\alpha}_j - \underline{\alpha}_j) K(X(i), X(j)) - \sum_{i=1}^{N} (\overline{\alpha}_i - \underline{\alpha}_i) y(i) + \sum_{i=1}^{N} (\overline{\alpha}_i + \underline{\alpha}_i) \varepsilon \quad (9)$$

subject to $0 \le \overline{\alpha}_i \le C / N$ and $0 \le \underline{\alpha}_i \le C / N$, where ε and C are the given precision level and regularization constant respectively. Minimizing $L(.,.)$ yields the following cases:

If $|y(i) - f(X(i))| < \varepsilon$, then $\overline{\alpha}_i = \underline{\alpha}_i = 0 \quad (10)$

If $y(i) - f(X(i)) = \varepsilon$, then $\begin{cases} 0 < \overline{\alpha}_i < C / N \\ \underline{\alpha}_i = 0 \end{cases} \quad (11)$

If $f(X(i)) - y(i) = \varepsilon$, then $\begin{cases} \overline{\alpha}_i = 0 \\ 0 < \underline{\alpha}_i < C / N \end{cases} \quad (12)$

If $y(i) - f(X(i)) > \varepsilon$, then $\begin{cases} \overline{\alpha}_i = C / N \\ \underline{\alpha}_i = 0 \end{cases} \quad (13)$

If $f(X(i)) - y(i) > \varepsilon$, then $\begin{cases} \overline{\alpha}_i = 0 \\ \underline{\alpha}_i = C / N \end{cases} \quad (14)$

Condition (10) implies that some of the Lagrange multipliers are zero. Consequently, the number of kernels in the SVM (6) can be reduced. Assuming n data points satisfy cases (11)-(14), the SVM is simplified as,

$$f(X(k)) = \sum_{j=1}^{n} \alpha_j K(X(k), X'(j)) + b \quad (15)$$

where $\{X'(1),..., X'(n)\}$ is a subset of the training data referred to as the Support Vectors (SV) [1-5],

$$S = \{X(i) : |y(i) - f(X(i))| \ge \varepsilon\} \quad (16)$$

and $\{\alpha_1,..., \alpha_n\}$ are given by,

$$A = \{\overline{\alpha}_i - \underline{\alpha}_i : |y(i) - f(X(i))| \ge \varepsilon\} \quad (17)$$

The bias b is computed from Conditions (11)-(14), the so-called Karush-Kuhn-Tucker conditions [1,2]. The model (8) is known as the support vector regression (SVR). It is shown that (13) and (14) do not occur if C is sufficiently large, implying that the estimate of $d(k)$ is bounded by $\pm\varepsilon$ [14]. Note that an optimal structure of $f(X(k))$ is suitably chosen subject to (4) if ε is less than G.

4. Support Vector Recurrent Neurofuzzy Network

Neurofuzzy networks are often used in modeling nonlinear systems for given input-output data. It is shown that under certain conditions, they are transparent and that the linguistic rules can be deduced from the network [12]. A recurrent network is used to implement a stochastic system represented by an ARMA model [9]. In this work, recurrent neurofuzzy networks are used to implement the NARMAX model. Since $f(.)$ and $g(.)$ are well-defined, they can be approximated by the following neurofuzzy networks.

$$\hat{f}(X(k)) = \sum_{i=1}^{n} \hat{\alpha}_i A_i(X(k)) + \sum_{i=1}^{n'} \hat{\beta}_i B_i(\underline{0}) \quad (18)$$

$$\hat{g}(E(k)) = \sum_{i=1}^{n'} \hat{\beta}_i B_i(E(k)) - \sum_{i=1}^{n'} \hat{\beta}_i B_i(\underline{0}) \quad (19)$$

where $\hat{\alpha}_i$ and $\hat{\beta}_i$ are the estimated weights; n and n', the respective numbers of weights; $A_i(X(k))$, the normalized kernel is given by,

$$A_i(X(k)) = \frac{K(X(k), X'(i))}{\sum_{i=1}^{n} K(X(k), X'(i))} \tag{20}$$

and $B_i(E(k))$ is the tensor product of the univariate B-spline functions [11,12],

$$B_i(E(k)) = \prod_{j=1}^{m_e} s_{\gamma_j}^{i}(e(k-j)) \tag{21}$$

where $s^i(.)$ is given by,

$$s_k^j(x) = \left(\frac{x - x_{j-k}}{x_{j-1} - x_{j-k}}\right) s_{k-1}^{j-1}(x) + \left(\frac{x_j - x}{x_j - x_{j-k-1}}\right) s_{k-1}^{j}(x)$$

$$s_1^j(x) = \begin{cases} 1 & \text{if} \quad x \in [x_{j-1}, x_j) \\ 0 & \text{otherwise} \end{cases} \tag{22}$$

It should be noted in (19) that $\hat{g}(0) = 0$, satisfying (3). From (18) and (19), the prediction of $y(k)$ is,

$$\begin{cases} \hat{y}(k) = \sum_{i=1}^{n} \hat{\alpha}_i A_i(X(k)) + \sum_{i=1}^{n'} \hat{\beta}_i B_i(E(k)) \\ E(k) = \begin{bmatrix} y(k-1) \\ \vdots \\ y(k-m_e) \end{bmatrix} - \begin{bmatrix} \hat{y}(k-1) \\ \vdots \\ \hat{y}(k-m_e) \end{bmatrix} \end{cases} \tag{23}$$

As $\hat{f}(X(k))$ is a Support Vector Neurofuzzy Network derived from the SVR, (23) is referred to as the support vector recurrent neurofuzzy network (SVRNFN), as shown in figure 1. The training of the network will be discussed in the next section.

5. Training of the SVRNFN

The prediction error (PE) method is a general parameter estimation method, which can be implemented by the Newton-Raphson technique [6]. In this paper, a training algorithm is proposed for training the SVRNFN, which is derived following closely the PE method. Similarly, the following assumptions are made.

S1: Let $e^k=[e(k),\ldots,e(0)]^T$ and Ω^k be σ–algebra generated by (e^k, u^k). So $E[e(k) \mid \Omega^{k-1}] = 0$.

S2: $E[e^k e^{k^T} \mid \Omega^{k-1}] \geq \delta I, \quad \delta > 0.$

Let D_M be a compact set, and M, the model set that consists of model (23),

$$M = \{M(\theta) : F(k,\theta) = r^T(k)\theta \mid \theta \in D_M\} \tag{24}$$

where $r(k) = [A_1(k) \ldots A_n(k) \quad B_1(k) \ldots B_{n'}(k)]^T$ and $\hat{\theta} = [\alpha_1 \ldots \alpha_n \quad \beta_1 \ldots \beta_{n'}]^T$. The training algorithm aims at finding the model with the smallest mean square errors from M,

$$J(\theta) = \frac{1}{2N} \sum_{k=1}^{N} e^2(k) \tag{25}$$

Since the mean of $e(k)$ is zero, the neurofuzzy network is trained from the input-output data, $\{u(1), \ldots, u(N), y(1), \ldots, y(N)\}$ by minimizing the cost function (25). The Newton-Raphson method is used here to estimate the weights in (24).

The prediction error of SVRNFN (23) is given by,

$$e(k) = y(k) - r^T(k)\hat{\theta} \tag{26}$$

where the weight $\hat{\theta}$ is computed iteratively by the Newton-Raphson technique,

$$\theta(l+1) = \theta(l) - (\Phi^T \Phi + S)\Phi\Psi \tag{27}$$

where l is the number of iterations, $\hat{\theta} = [\theta_1 \quad \theta_2 \ldots \theta_{n+n'}]$, $r(k)=[r_1(k)\ r_2(k) \ \ldots \ r_{n+n'}(k)]^T$, $\Psi = [e(1), e(2), \ldots, e(N)]^T$, and Φ and S are

$$\Phi_{i,j} = \frac{\partial e(i)}{\partial \theta_j}, \quad S_{i,j} = \sum_{k=1}^{N} e(k) \frac{\partial^2 e(k)}{\partial \theta_i \partial \theta_j} \tag{28}$$

The partial derivatives of the prediction errors with respect to θ can be generated by the sensitivity model, as given below [7].

5.1 Sensitivity model

The derivatives of prediction errors of the SVRNFN can be derived by dynamic back propagation [7]. Define

$$d(k,j) = \begin{cases} \theta^T \dfrac{\partial r(k)}{\partial e(j)}, & k\text{-}1 \geq j \geq k\text{-}m_e \\ 1, & k = j \\ 0, & \text{otherwise} \end{cases} \tag{29}$$

where $\dfrac{\partial r(k)}{\partial e(j)} = \left[0 \cdots 0 \quad \dfrac{\partial B_1(E(k))}{\partial e(j)} \cdots \dfrac{\partial B_{n'}(E(k))}{\partial e(j)}\right]^T$.

Differentiating (21) with respect to $e(j)$ gives

$$\frac{\partial B_i(E(k))}{\partial e(j)} = \frac{ds_{\gamma_j}^{i}(e(j))}{de(j)} \prod_{l=1, l \neq k-j}^{m_e} s_{\gamma_l}^{i}(e(k-l)) \tag{30}$$

For $s_{\gamma_j}^{i}(e(j)) \neq 0$, $\dfrac{\partial B_i(k)}{\partial e(j)} = \dfrac{B_i(E(k))}{s_{\gamma_j}^{i}(e(j))} \dfrac{ds_{\gamma_j}^{i}(e(j))}{de(j)}$.

The derivative of an univariate B-spline function can be obtained recursively as follows [11, 12].

$$\frac{ds_k^j(x)}{dx} = \left(\frac{k-1}{x_{j-1} - x_{j-k}}\right) s_{k-1}^{j-1}(x) - \left(\frac{k-1}{x_j - x_{j-k+1}}\right) s_{k-1}^{j}(x) \tag{31}$$

Assuming that the initial conditions of the first and second partial derivatives are zero, differentiating (26) with respect to θ_i, yields,

$$\frac{\partial e(k)}{\partial \theta_i} + d(k,k-1)\frac{\partial e(k-1)}{\partial \theta_i} + \ldots + d(k,k-m_e)\frac{\partial e(k-m_e)}{\partial \theta_i} = -r_i(k) \quad (32)$$

This linear dynamic model is known as the sensitivity model, with the first partial derivative $\partial e(k)/\partial \theta_i$ as the output, and the basis function as the input [7]. An advantage of using the neurofuzzy networks is that the coefficients of the dynamic model are given by the derivatives of the basis functions, therefore, the computation time will not be too large for small m_e. Define

$$c_i(k,j) = \begin{cases} \dfrac{\partial r_i(k)}{\partial e(j)}, & i = n+1,\ldots,n+n' \text{ and } k\text{-}1 \geq j \geq k\text{-}m_e \\ 0, & \text{otherwise} \end{cases} \quad (33)$$

Differentiating (32) with respect to θ_i, a sensitivity model of the second partial derivative is obtained.

$$\sum_{j=1}^{N} d(k,j)\frac{\partial^2 e(j)}{\partial \theta_h \partial \theta_i} = -\sum_{j=1}^{N} c_i(k,j)\cdot\frac{\partial e(j)}{\partial \theta_h} \quad (34)$$

5.2 Training Procedure

The procedure to train the recurrent neural networks is summarized below.

(1) Choose suitable $X(k)$, $E(k)$, m_e, ε, λ and C.
(2) Obtain SV by minimizing $L(.)$ given by (9).
(3) Initialize weights of $\hat{f}(X(k))$ by that of SVR.
(4) Compute $e(k)$, for k=1,...,N by (26).
(5) Compute Φ and S by (32) and (34).
(6) Update the parameters $\theta(l+1)$ by (27).
(7) Repeat (4)-(6) until $\left|\Delta J(\hat{\theta})/J(\hat{\theta})\right| < \varepsilon'$, say 10^{-3}.

6. Examples

Consider the following nonlinear system given in [12],

$$y(k) = \left(0.8\text{-}0.5e^{-y^2(k-1)}\right)y(k-1) - \left(0.3+0.9e^{-y^2(k-1)}\right)y(k-2) + 0.1\sin(\pi y(k-1)) - p\varepsilon(k-1) + \varepsilon(k) \quad (35)$$

where $\varepsilon(k) \sim N(0, 0.1^2)$, and p is a constant set to 0.5, 0,8 and 1 respectively. The iterative map of $(y(k-2), y(k-1))$ with an initial condition of (0.1, 0.1), when there is no noise, diverges from an unstable equilibrium near the origin towards a periodic attractor, as shown in Fig. 2. In the example, 1000 input-output data are generated from (35) with an initial condition of (0, 0). Since most of the data are on the periodic attractor, as shown in Fig. 2, only the first 50 data points are required to select the SV. The distance between the inner knots λ is set to 1.25, and the regularization constant C, to 100. The precision ε is chosen to be 0.122, 0.122 and 0.149 for the respective value of p. Note that ε for p = 1 is larger, as the variance of the correlated noise is larger. From Step 2 in training procedure, 16 SV are selected from the 50 data points, which are marked by circles in Fig. 3. In the SVRNFN, 16 normalized B-spline kernels are used to model $\hat{f}(\bullet)$, and 2 triangular B-spline functions are used to model $\hat{g}(\bullet)$. The initial estimate of α is now computed, as given in Step 3. The Newton-Raphson technique is now applied to train the SVRNFN with derivatives computed from the sensitivity model. The estimate iterative map of $f(.)$ in all three cases are shown in Figs. 4 to 6. The autocorrelation functions of the prediction errors for p = 0.5, as shown in Fig. 7, indicated that the prediction errors are statistically uncorrelated. The autocorrelation functions from the other two cases are similar, indicating that good generalization results are obtained.

7. Conclusion

The SVRNFN is proposed to model dynamic systems with correlated noise. The main feature of the SVRNFN is that the structure of the sub-network modeling the nonlinear system is determined by the SV, which is selected similar to that in the SVR. A recurrent neurofuzzy network is used to model the correlated noise. The training algorithm of the SVRNFN is derived based on the prediction error method. The Newton-Raphson technique is proposed to train the SVRNFN, and the derivation of the derivatives using the sensitivity model is given. The SVRNFN is applied to model a nonlinear system corrupted by correlated noise. Good generalization results are obtained using the SVRNFN, and the training algorithms based on the Newton-Raphson technique.

8. References

[1] B. Schölkopf, P. Bartlett, A.J. Smola, and R. Williamson, "Shrinking the Tube: A New Support Vector Regression Algorithm," *Advances in Neural Information Processing Systems*, 11, M.S. Kearns, S.A. Solla, and D.A. Cohn, (eds.), MIT Press, Cambridge, MA, 1999.

[2] C.J.C. Burges, "Geometry and Invariance in Kernel Based Methods," *Advanced in Kernel Methods - Support Vector Learning*, B. Schölkopf, C. Burges, and A. Smola (eds.), MIT Press, Cambridge, USA, 1998, pp 89-116.

[3] P.M.L. Drezet, and R.F. Harrison, "Support Vector Machines for System Identification," *UKACC International Conference on CONTROL '98*, 1-4 September 1998, pp 688-692.

[4] S.R. Gunn, and M. Brown, "SUPANOVA - A Sparse, Transparent Modelling Approach,"

Proc. IEEE Neural Networks for Signal Processing Workshop, 1999, pp 21-30.

[5] V. Vapnik, *The nature of statistical learning theory*, Springer-Verlag New York, 1995.

[6] L. Ljung, "Analysis of a General Recursive Prediction Error Identification Algorithm," *Automatica*, 17, 1981, pp 89-99.

[7] K.S. Narendra, and K. Parthasarathy, "Gradient methods for the optimization of dynamical systems containing neural networks," *Trans. IEEE on Neural Networks*, 2, 1991, pp 252-262.

[8] I.J. Leontaritis, and S.A. Billings, "Input–output parametric models for nonlinear systems Part II: stochastic nonlinear systems," *Int. J. Control*, 41, 1985, pp 329-344.

[9] J.T. Connor, R.D. Martin, and L.E. Atlas, "Recurrent Neural Networks and Robust Time Series Prediction," *Trans. IEEE on Neural Networks*, 5, 1991, pp 240-253.

[10] S. Chen, S.A. Billings, C.F.N. Cowan, and P.M. Grant, "Practical identification of NARMAX models using radial basis functions," *Int. J. Control*, 52, 1990, pp 1327-1350.

[11] L. Piegl, and W. Tiller, *The NURBS Book*, Springer-Verlag, Berlin Heidelberg, 1995.

[12] M. Brown, and C.J. Harris, *Neurofuzzy Adaptive Modelling and Control*, Prentice Hall, 1994.

[13] W.C. Chan, C.W. Chan, K.C. Cheung and C.J. Harris, "Modelling of nonlinear dynamic systems using support vector neural networks," *Preprints IFAC Symposium AIRTC '2000*, Budapest, Hungary, 2-4 October 2000, pp 217-222.

[14] V. Vapnik, *Statistical Learning Theory*, John Wiley & Sons, New York, 1998.

[15] W.C Chan, C.W. Chan, K.C. Cheung, and C.J. Harris, "On the modelling of nonlinear dynamic systems using support vector neural networks," *Engineering Applications of Artificial Intelligence*, 14, 2001, pp 105-113.

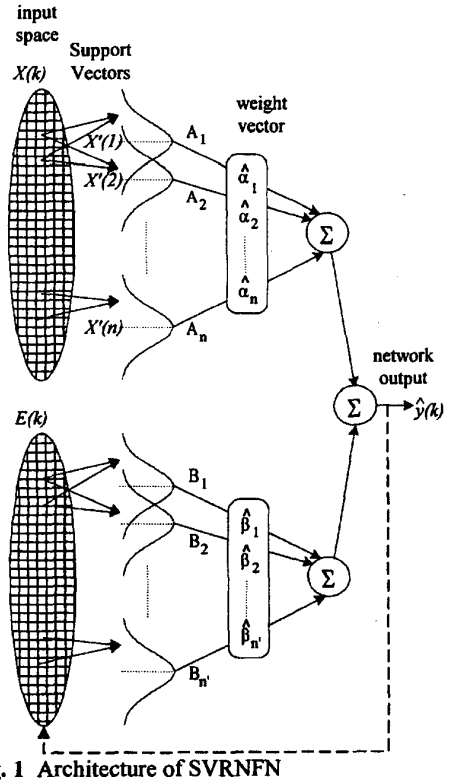

Fig. 1 Architecture of SVRNFN

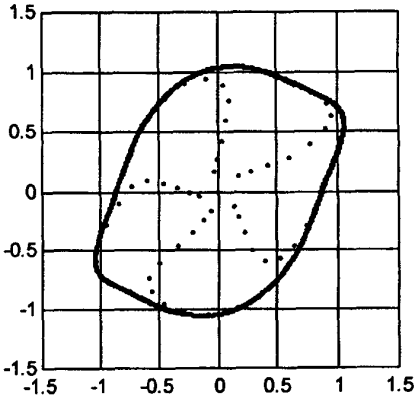

Fig. 2 The iterative map of $f(.)$.

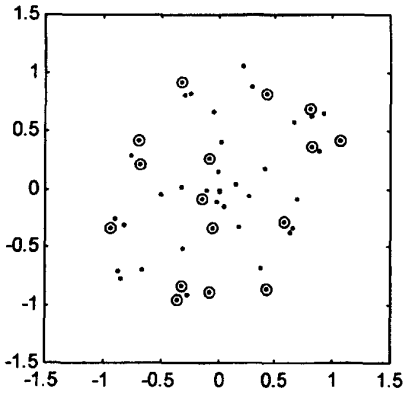

Fig. 3 Support Vectors for p=0.5 and ε=0.122.

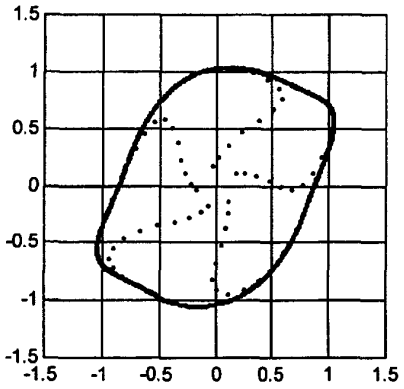

Fig. 4 Estimated iterative map of $f(.)$ for p=0.5.

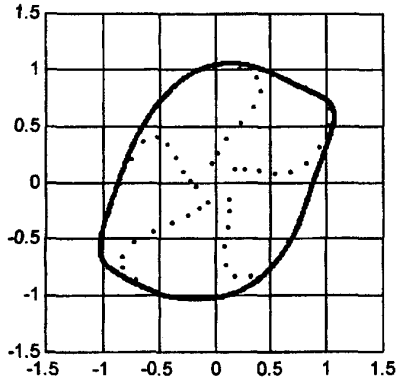

Fig. 5 Estimated iterative map of $f(.)$ for p=0.8.

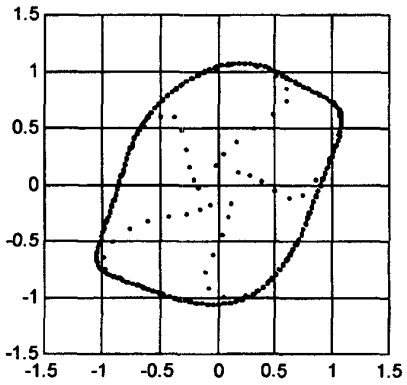

Fig. 6 Estimated iterative map of $f(.)$ for p=1.

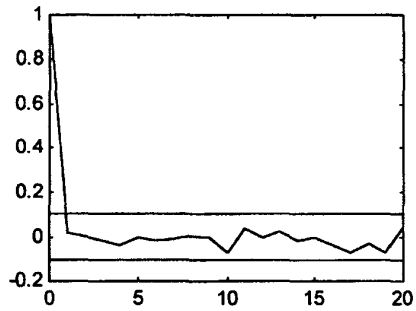

Fig. 7 The autocorrelation functions of $e(k)$ for p=0.5.