MODELLING OF NONLINEAR DYNAMIC SYSTEMS USING SUPPORT VECTOR NEURAL NETWORKS

W. C. Chan*, C. W. Chan*, K. C. Cheung* and C. J. Harris**

**Department of Mechanical Engineering, The University of Hong Kong, Pokfulam Road, Hong Kong, China*
***Department of Electronics and Computing, University of Southampton, UK*
Email: mechan@hkucc.hku.hk Fax: (852) 28585415

Abstract: Neural networks provide good generalization results only if the structure is suitably chosen. Therefore, it is important to derive methods to choose the 'best' structure of the networks. This paper proposes the Support Vector Neural Networks (SVNN) where the structure is determined using the approach of the Support Vector Regression and the weights are estimated by the linear least squares method. It is shown here that the SVNNs have bounded modelling errors and give unbiased estimate of the dynamic systems. The performance of the SVNN is illustrated by a simulation example involving a nonlinear system. *Copyright © 2000 IFAC*

Keywords: Radial basis function networks, optimization, least-squares method, modelling errors, dynamic systems.

1. INTRODUCTION

In the implementation of associative memory networks (AMNs), an important consideration is the choice of the structure of the network, such as the number of basis functions and their locations in the input space, as these factors can affect the performance of the AMN (Brown and Harris, 1994). This is a difficult problem particularly when no *a-priori* knowledge of the system is available. An approach to solve this problem is the Adaptive Spline Modelling of Observation Data (ASMOD) algorithm proposed by Kavli and Weyer (1995). It is an iterative search algorithm, which prunes the unnecessary centres or basis functions to improve the performance of the network. In this paper, a direct construction algorithm based on the support vector regression (Drezet and Harrison, 1998; Schölkopf *et al.*, 1999) is proposed for selecting the centres and the number of the multivariate basis functions.

Support Vector Machine (SVM) derived from the statistical learning theory is a linear-in-weights network with self-organizing structure (Vapnik, 1995). The weights and the structure of the SVM are obtained by minimizing a cost function for a given precision level. The kernels for data points within the error bounds are removed from the network leaving kernels that are on or outside the error bounds in the network, which is known as the Support Vectors (Vapnik, 1995). It is shown in this paper that the Support Vector Regression (SVR) reduces to a neural network based on the Support Vectors (SVs). For simplicity, this class of neural network is referred to as the Support Vector Neural Networks (SVNN). The SVNN can also be interpreted as a radial basis function (RBF) network with scatter partitioned input space (Jang *et al.*, 1997). It is shown here that the SVNN reserves the property of bounded cost function, as in the SVR, and that its output is an unbiased estimate of the nonlinear system.

This paper is organized as follows. In §2, the RBF network with scatter partitioned input space is presented. In §3, the derivation of the SVR is presented. In §4, Support Vector Neural Networks are developed from the SVR. In §5, the training of the SVNN using the least squares method is derived, followed by the properties of SVNN, which is discussed in §6. The performance of the SVNN is illustrated by the example shown in §7.

2. REVIEW OF ASSOCIATIVE MEMORY NETWORKS

Consider a nonlinear dynamic system given by,

$$y(k)=f(X(k))+e(k) \tag{1}$$

where $X(k)=[y(k-1),\ldots,y(k-m_y),u(k-1),\ldots,u(k-m_u)]^T$; $y(k)$ and $u(k)$ are respectively the output and the input; $e(k)$, a white noise, $e(k)\sim N(0,\sigma^2)$; m_y and m_u, the orders of the system; $f(.)$, a well-defined nonlinear function. It is assumed that m_y and m_u are known and the nonlinear function $f(.)$ is modelled by an AMN. Let $m=m_y+m_u$. A common choice of the structure of the AMN is that the centres of the basis functions of the AMN are evenly distributed on an m-dimensional lattice.

Unfortunately, this structure leads to the well known "curse of dimensionality" problem (Brown and Harris, 1994). For example, the Radial Basis Function (RBF) network (Chen and Billings, 1992) consists of m inputs with each input represented by n_i univariate basis functions, the number of multivariate basis functions, and hence the number of weights is,

$$n=\prod_{i=1}^{m} n_i \tag{2}$$

Clearly, n increases exponentially as m is increased. For $m=2$, and $n_1=n_2=4$, the number of weights is 16. The corresponding input space is shown in Fig. 1. In this paper, RBF networks with scatter partitioned input space (Jang *et al.*, 1997) are considered. For this class of RBF networks, the centres of the basis functions can be freely chosen from the given data.

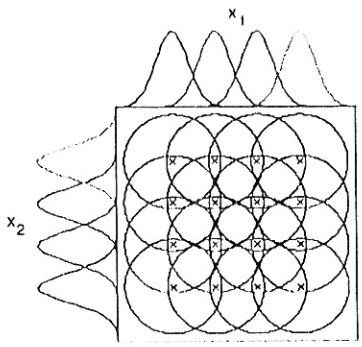

Fig. 1. Grid partitioning.

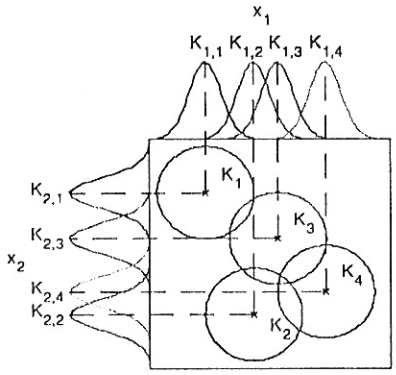

Fig. 2. Scatter partitioning.

Let $X(k)=[x_1(k),\ldots,x_m(k)]^T$. The output of the RBF network is,

$$y(k)=\frac{\sum_{i=1}^{n}\theta_i\mu_i(X(k))}{\sum_{i=1}^{n}\mu_i(X(k))} \tag{3}$$

where θ_i is the i^{th} weight; $\mu_i(.)$, the i^{th} Gaussian basis function given by,

$$\mu_i(X(k))=\exp\left(-\frac{\|X(k)-X'(i)\|^2}{2\gamma^2}\right) \tag{4}$$

where γ and $X'(i)=[x'_1(i),\ldots,x'_m(i)]^T$ are respectively the spread and the centre of the Gaussian function. The i^{th} multivariate basis function is a product of the univariate basis functions for each input variable, i.e.,

$$\mu_i(X(k))=\prod_{j=1}^{m}\mu_{i,j}(x_j(k)) \tag{5}$$

where $\mu_{i,j}(x_j(k))=\exp(-(x_j(k)-x'_j(i))/2\gamma^2)$. The spread γ is chosen to ensure a thorough coverage of the input space. The centre $X'(i)$ is chosen amongst the available data $\{X(k)\}$. Note that at $X'(i)$, $\mu_i(X'(i))=1$. The main advantage of scattered partitioning of the input space is that, in general, less basis functions are required in the network. The input space of the RBF network with scatter partitioning input space for the case: $n_1=n_2=4$ is shown in Fig. 2. Only 4 instead of 16 multivariate basis functions are required.

3. SUPPORT VECTOR MACHINE

To obtain an accurate estimate of the function $f(.)$, the training algorithm and the network structure should be chosen suitably. A general principle is that the simplest structure that gives an acceptable precision is chosen. Therefore, the choice of the structure of the network is often a compromise between modeling errors and the complexity of the network. In Vapnik (1995), Support Vector Machines (SVMs) are proposed with the optimal network structure selected

for a given precision. The SVM (Vapnik, 1995; and Gunn *et al.*, 1999) for estimating the nonlinear function $f(.)$ is

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

where N is the length of the data; $\overline{\alpha}_i$ and $\underline{\alpha}_i$ are the Lagrange multipliers and b is the bias. The kernel $K(X(k),X(i))$ is a nonlinear function satisfying the Mercer's condition (Burges *et al.*, 1998). The parameters $\overline{\alpha}_i$, $\underline{\alpha}_i$ and b are obtained by minimizing the following cost function (Gunn and Brown, 1999).

$$L(\overline{\alpha},\underline{\alpha})=\frac{1}{2}\sum_{i,j=1}^{N}(\overline{\alpha}_i-\underline{\alpha}_i)(\overline{\alpha}_j-\underline{\alpha}_j)K(X(i),X(j)) - \sum_{i=1}^{N}(\overline{\alpha}_i-\underline{\alpha}_i)y(i)+\sum_{i=1}^{N}(\overline{\alpha}_i+\underline{\alpha}_i)\varepsilon \quad (7)$$

subject to $\sum_{i=1}^{N}(\overline{\alpha}_i-\underline{\alpha}_i)=0$, $0\le\overline{\alpha}_i,\underline{\alpha}_i\le C/N$. Minimizing (7) implies that $\overline{\alpha}_i$ and $\underline{\alpha}_i$ lie in certain ranges corresponding to the following conditions.

(C1) If $|y(i)-f(X(i))|<\varepsilon$, then $\overline{\alpha}_i=\underline{\alpha}_i=0$,

(C2) If $y(i)-f(X(i))=\varepsilon$, then $\begin{cases}0<\overline{\alpha}_i<C/N\\ \underline{\alpha}_i=0\end{cases}$, or

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

(C3) If $y(i)-f(X(i))>\varepsilon$, then $\begin{cases}\overline{\alpha}_i=C/N\\ \underline{\alpha}_i=0\end{cases}$, or

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

The bias b can be evaluated by taking into account the equalities in (C2). Through the minimization of $L(.)$ given by (7), only those $\overline{\alpha}_i-\underline{\alpha}_i$ that satisfy (C2) and (C3) are nonzero. In the SVM, each pair of $\{\overline{\alpha}_i,\underline{\alpha}_i\}$ is associated with a kernel with a centre at $X(i)$. The i^{th} kernel is retained if $\overline{\alpha}_i-\underline{\alpha}_i\neq0$, but is removed if both $\overline{\alpha}_i$ and $\underline{\alpha}_i$ are zero. The centres $X(i)$'s of the remaining kernels in the SVM are referred to as the Support Vectors (Vapnik, 1995).

Let

(i) n be the number of SVs.

(ii) $\alpha_j=\overline{\alpha}_i-\underline{\alpha}_i$

(iii) $K_j(X(k))=K(X(k),X(i))$

where $j=1,\ldots,n$ and $X(i)$ is the j^{th} SV. The SVM (6) can be simplified as,

$$f(X(k))=\sum_{j=1}^{n}\alpha_j K_j(X(k))+b \quad (8)$$

Note that $K_j(X(k))$ is the Gaussian function given by (4) with centres at the SVs obtained from the minimization algorithm, i.e. $X'(j)=X(i)$. The model given by (8) is referred to as the Support Vector Regression (SVR) (Drezet and Harrison, 1998; Schölkopf *et al.*, 1999), by which the number of the kernels and their centres are determined for a given ε.

4. SUPPORT VECTOR NEURAL NETWORKS

The kernels of the SVR discussed previously do not necessarily form a partition of unity. Further, the output of the SVR is biased, as shown later in Section 5. Since the SVR (8) can be considered as a two-layer neural network linear in its weights, it is intuitive to reformulate it as a RBF network given by (3) using normalized basis functions. The weights can also be estimated using linear least squares method to overcome the biased problem of the SVR. For convenience, let $[\theta'_1,\ \ldots,\ \theta'_{n+1}]^T$ be the weights of the network. Then (8) can be rewritten as,

$$y(k)=\sum_{j=1}^{n}\theta'_j\,K_j(X(k))+\theta'_{n+1} \quad (9)$$

Let $$\lambda(X(k))=\sum_{i=1}^{n}K_i(X(k)), \quad (10)$$

$$\Psi=\begin{bmatrix}K_1(X(1)) & \cdots & K_n(X(1))\\ \vdots & & \vdots\\ K_1(X(N)) & \cdots & K_n(X(N))\end{bmatrix}, \quad (11)$$

and $$L=\begin{bmatrix}\lambda(X(1)) & & \mathbf{O}\\ & \ddots & \\ \mathbf{O} & & \lambda(X(N))\end{bmatrix} \quad (12)$$

Note that the kernels in (9) are multivariate basis functions (3) and the SV are the respective centres. In matrix form, (9) becomes

$$Y=\Psi\begin{bmatrix}\theta'_1\\ \vdots\\ \theta'_n\end{bmatrix}+\begin{bmatrix}\theta'_{n+1}\\ \vdots\\ \theta'_{n+1}\end{bmatrix} \quad (13)$$

where $Y=[y(1),\ldots,y(N)]^T$. Since $\sum_{i=1}^{n}\frac{K_i(X(k))}{\lambda(X(k))}=1$,

$$L^{-1}\Psi\begin{bmatrix}1\\ \vdots\\ 1\end{bmatrix}=\begin{bmatrix}1\\ \vdots\\ 1\end{bmatrix} \quad (14)$$

Equation (13) can be rewritten as,

$$Y=\Psi\begin{bmatrix}\theta'_1\\ \vdots\\ \theta'_n\end{bmatrix}+L^{-1}\Psi\begin{bmatrix}\theta'_{n+1}\\ \vdots\\ \theta'_{n+1}\end{bmatrix}=L^{-1}\Psi\begin{bmatrix}\theta_1\\ \vdots\\ \theta_n\end{bmatrix} \quad (15)$$

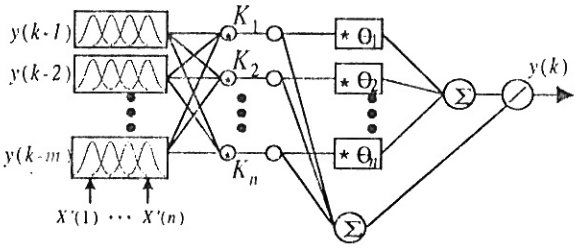

Fig. 3. The architecture of the SVNN based on the Support Vectors, $X'(i)$.

where $\{\theta_1,\ldots,\theta_n\}$ are the weights given by

$$\Psi\begin{bmatrix}\theta_1\\ \vdots\\ \theta_n\end{bmatrix} = L\Psi\begin{bmatrix}\theta'_1\\ \vdots\\ \theta'_n\end{bmatrix} + \Psi\begin{bmatrix}\theta'_{n+1}\\ \vdots\\ \theta'_{n+1}\end{bmatrix} \tag{16}$$

Rearranging (16), the weights $\{\theta_1, \ldots,\theta_n\}$ can be expressed in terms of $\{\theta'_1, \ldots,\theta'_{n+1}\}$ as

$$\begin{bmatrix}\theta_1\\ \vdots\\ \theta_n\end{bmatrix} = \left(\Psi^T\Psi\right)^{-1}\Psi^T L\Psi\begin{bmatrix}\theta'_1\\ \vdots\\ \theta'_n\end{bmatrix} + \begin{bmatrix}1\\ \vdots\\ 1\end{bmatrix}\theta'_{n+1} \tag{17}$$

From (17), the following network is obtained.

$$y(k) = \frac{\sum_{i=1}^{n}\theta_i K_i(X(k))}{\sum_{i=1}^{n}K_i(X(k))} = \sum_{i=1}^{n}\theta_i N_i(X(k)) \tag{18}$$

where $N_i(X(k))=K_i(X(k))/\lambda(X(k))$ is the normalized basis function. For convenience, the network (18) is referred to as the Support Vector Neural Network (SVNN) in the following discussion. The architecture of the SVNN for $m_u=0$ and $m=m_y$ is shown in Fig. 3.

5. TRAINING OF SUPPORT VECTOR NEURAL NETWORKS

Rewrite (18) in matrix form,

$$y(k) = B^T(X(k))\theta \tag{19}$$

where $B(X(k))=[N_1(X(k)),\ldots,N_n(X(k))]^T$ and $\theta=[\theta_1,\ldots,\theta_n]^T$. Denote the cost function by,

$$V(\theta) = \frac{1}{N}\sum_{k=1}^{N}\left(y(k) - \hat{y}(k)\right)^2 \tag{20}$$

where $\hat{y}(k)$ is the estimate of $y(k)$ computed by

$$\hat{y}(k) = B^T(X(k))\hat{\theta} \tag{21}$$

$\hat{\theta}$ is the linear least squares estimate of θ given by,

$$\hat{\theta} = \left[\Phi^T\Phi\right]^{-1}\Phi^T Y \tag{22}$$

where

$$\Phi = \begin{bmatrix} N_1(X(1)) & \cdots & N_n(X(1))\\ N_1(X(2)) & \cdots & N_n(X(2))\\ \vdots & & \vdots\\ N_1(X(N)) & \cdots & N_n(X(N))\end{bmatrix} \text{ and } Y = \begin{bmatrix} y(1)\\ y(2)\\ \vdots\\ y(N)\end{bmatrix}.$$

6. PROPERTIES OF SUPPORT VECTOR NEURAL NETWORKS

The properties of the SVNN trained by the linear least squares method given in §5 are derived here.

Theorem 1 Consider the dynamic system given by (1) with $e(k)=0$. Let the SVs of the SVNN be obtained by minimizing (7) for a given ε, and the weights computed from (22), then $V(\hat{\theta}) < \varepsilon^2$.

Proof: Since

$$\hat{\theta} = \arg\min_{\theta}\frac{1}{N}\sum_{k=1}^{N}\left(y(k) - B^T(X(k))\theta\right)^2 \tag{23}$$

it implies that

$$V(\hat{\theta}) \le \frac{1}{N}\sum_{k=1}^{N}\left(y(k) - f(X(k))\right)^2 \tag{24}$$

where $f(X(k))=B^T(X(k))\alpha'$ denotes the SVR where $\alpha'=(\Psi^T\Psi)^{-1}\Psi^T L\Psi[\alpha_1,\ldots,\alpha_n]^T+[b,\ldots,b]^T$ according to (17). Let $\{[X(1),y(1)], \ldots,[X(n),y(n)],\ldots,[X(N),y(N)]\}$ be the data set, and $\{X(1),\ldots,X(n)\}$, the support vectors. For a sufficiently large C, the output of the SVNN is subject to conditions C1 and C2 only. From C1, $|y(k) - f(X(k)| < \varepsilon$, for $k = n+1,\ldots,N$,

$$\sum_{k=n+1}^{N}\left|y(k) - f(X(k))\right|^2 < (N-n)\varepsilon^2 \tag{25}$$

From C2, $|y(k)-f(X(k)|=\varepsilon$, for $k=1,\ldots,n$,

$$\sum_{k=1}^{n}\left|y(k) - f(X(k))\right|^2 = n\varepsilon^2 \tag{26}$$

The sum of (25) and (26) is,

$$\frac{1}{N}\sum_{k=1}^{N}\left(y(k) - f(X(k))\right)^2 < \varepsilon^2 \tag{27}$$

From (24) and (27), it gives $V(\hat{\theta}) < \varepsilon^2$. □

The output of the SVNN in matrix form is,

$$Y = \Phi\theta + \xi \tag{28}$$

where $\xi=[e(1),\ldots,e(N)]^T$. It is shown below that the output of the SVNN (21) gives an unbiased estimate of $y(k)$.

Theorem 2 For the system given by (1). The output of the SVNN is an unbiased estimate of Y, whilst that from the SVR is biased.

Proof: Let $\hat{\theta}$ be the least squares estimate of θ. The output of the SVNN is given by,

$$\hat{Y} = \Phi\hat{\theta} \tag{29}$$

From (22), (28), and (29), the modelling error is

$$Y - \hat{Y} = \left(I_N - \Phi\left(\Phi^T\Phi\right)\Phi^T\right)\xi \tag{30}$$

where I_N is the NxN identity matrix. Since Φ and ξ are independent, it can be readily shown that the expected value of the modeling error is

$$E\left[Y - \hat{Y}\right] = 0 \tag{31}$$

hence the output of the SVNN is unbiased. However, the modeling error of the SVR is

$$Y - F = \begin{bmatrix} y(1) - f\left(X(1)\right) \\ \vdots \\ y(N) - f\left(X(N)\right) \end{bmatrix} \tag{32}$$

where $F = [f(X(1)), \ldots, f(X(N))]^T$. For non-trivial solution, there exist at least two data points, $[X(i),y(i)]$ and $[X(j),y(j)]$, satisfying condition C2, i.e., $y(i)-f(X(i))=\varepsilon$ and $y(j)-f(X(j))=-\varepsilon$, giving

$$E\left[y(i) - f\left(X(i)\right)\right] = \varepsilon \tag{33}$$

$$E\left[y(j) - f\left(X(j)\right)\right] = -\varepsilon \tag{34}$$

Since, in general

$$E\left[Y - F\right] \neq 0 \tag{35}$$

hence the output of the SVR is biased. □

7. EXAMPLE

Consider the nonlinear dynamic system (Chen and Billings, 1992; Brown and Harris, 1994),

$$\begin{aligned} y(k) = &\left(0.8 - 0.5\exp\left(-y^2(k-1)\right)\right)y(k-1) \\ &-\left(0.3 + 0.9\exp\left(-y^2(k-1)\right)\right)y(k-2) \\ &+ 0.1\sin\left(\pi y(k-1)\right) + e(k) \end{aligned} \tag{36}$$

where $e(k)$ denotes the white noise. From (36), $m=2$ and $X(k)=[y(k-1),y(k-2)]^T$. When $e(k)\equiv 0$, the iterative map of $X(k)$ spirals out from the initial condition $X(1)=[0.1,0.1]^T$ towards the globally attracting limit cycle, as shown in Fig. 4. Let $e(k)\sim N(0,0.1^2)$, 300 data are generated by (36) with an initial condition of $X(1)=[0,0]^T$, as shown in Fig. 5.

The Gaussian kernel with $\gamma=0.7071$ is chosen, whilst ε and C are set to 0.2 and 300 respectively. From Fig. 5, the output $y(k)$ is on the limit cycle for the last 250 data points. Consequently, it is sufficient to use only the first 50 data points to identify the structure of the network, or the SVs. The algorithm for selecting the SVs presented in section 3 is applied, and a set of SVs with 10 elements is selected from the data points, as marked by circles in Fig. 6.

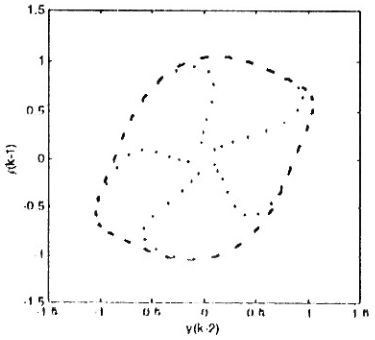

Fig. 4. Iterative map of the output of the nonlinear dynamic system (39) for $e(k)\equiv 0$.

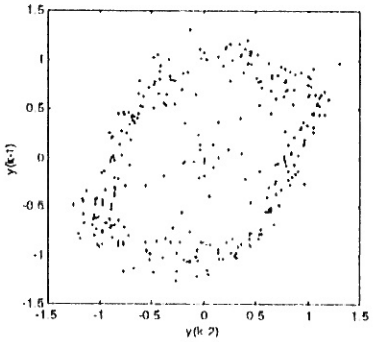

Fig. 5. Iterative map of the data for $e(k)\sim N(0,0.1^2)$.

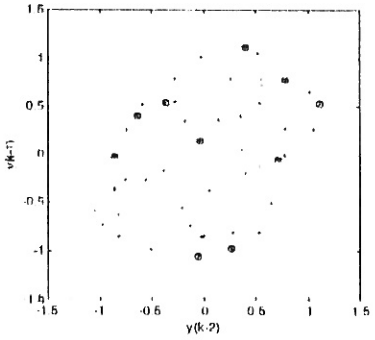

Fig. 6. Support vectors selected from 50 data points.

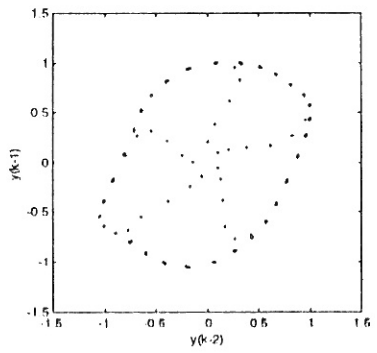

Fig. 7. Iterative map of the outputs of the SVNN.

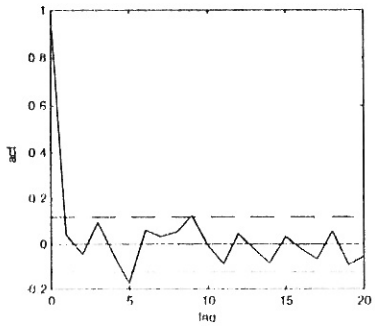

Fig. 8. The auto-correlation functions of $e(k)$.

The following SVNN is obtained,

$$y(k)=\frac{\sum_{i=1}^{10}\theta_i \exp\left(-\left\|X(k)-X'(i)\right\|^2\right)}{\sum_{i=1}^{10}\exp\left(-\left\|X(k)-X'(i)\right\|^2\right)} \qquad (40)$$

where $X'(i)$ is the i^{th} SV. The weights of the SVNN are obtained by (22). The iterative map of the output of the SVNN shown in Fig. 7 is a good approximation of the output of the nonlinear system (c.f. Fig. 4). From the auto correlation functions shown in Fig. 8, the training error is an uncorrelated random sequence. The cost function obtained from the training algorithm is 0.0099. Since ε^2 0.01, it is clear that $V\left(\hat{\theta}\right)<\varepsilon^2$, confirming the result given in Theorem 1. The mean of the errors is approximately zero, indicating that the output of the SVNN is unbiased, as given in Theorem 2.

8. CONCLUSION

In this paper, support vector neural networks are derived from the support vector machine. The main advantage of the SVNN is that the structure of the network, i.e., the SVs is selected based on the error bound specified by the user. As the SVs are usually scattered over the input space, good approximation of a nonlinear system can be obtained even though a smaller number of basis functions are used. It is shown also that the output of the SVR is biased, whilst that of the SVNN is not. The performance of the SVNN is illustrated by an example involving a nonlinear dynamic system contaminated by a white noise. The simulation result showed that the cost function was bounded by the square of the given precision, and the output of SVNN was unbiased, confirming the results given respectively in Theorems 1 and 2.

ACKNOWLEDGEMENT

The work was supported by the Research Grants Council of Hong Kong.

REFERENCES

Vapnik, V. (1995). *The nature of statistical learning theory*. Springer-Verlag, New York.

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

Gunn, S.R. and M. Brown (1999). SUPANOVA - A Sparse, Transparent Modelling Approach. *Proc. IEEE Neural Networks for Signal Processing Workshop*, pp. 21-30.

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

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

Chen, S. and S.A. Billings (1992). Neural networks for nonlinear dynamic system modelling and identification. *Int. J. Control*, **56**, 319-346.

Brown, M. and C.J. Harris (1994). *Neurofuzzy Adaptive Modelling and Control*. Prentice Hall.

Jang, J.-S.R., C.-T. Sun and E. Mizutani (1997). *Neuro-fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*. Prentice Hall.

Kavli, T. and E. Weyer (1995). On ASMOD - an algorithm for building multivariable spline models. *Neural Networks Engineering in Control*, pp. 83-104. Springer-Verlag.