A NEW SUPPORT VECTOR NEURAL NETWORK INFERENCE SYSTEM

Ling Wang and Zhi-Chun Mu
Information Engineering School, University of Science and Technology Beijing 100083,China

Abstract: In this paper, we present a new support vector neural network inference system (SVNNIS) for regression estimation. The structure of the proposed SVNNIS can be obtained similar to that in the support vector regression (SVR), while the output of the SVNNIS is unbiased compared with the SVR and the weights can be updated by the recursive least square method with forgetting factor. The advantage of this system is its good generalization capability. The simulation result illustrates the effectiveness of the proposed SVNNIS.

Key words: support vector machine(SVM); support vector regression(SVR); support vector neural network inference system (SVNNIS); regression estimation

1. INTRODUCTION

The Support vector machine (SVM) is derived from the Vapnik-Chervonenkis (VC) theory [1],[2]. Recently, SVM has also been applied to various fields such as classification, time prediction and regression. When SVM is employed to tackle the problems of function approximation and regression estimation, the approaches are often referred to as the support vector regression (SVR)[3]-[8].

The SVR type of function approximation is very effective, especially for solving the problems of multidimensional function regression estimation. Another important advantage for using SVR in function approximation is that the number of free parameters in the function approximation scheme is equal to the number of support vectors. Since the kernels of the SVR are similar to the basis functions of the RBF network, it is shown here that the

SVR can be reformulated as a RBF network with basis functions normalized. Jongcheol and Sangchul (2002)[9] proposed a new Support Vector Fuzzy inference system and used the SVM without bias. Nevertheless, the output of the SVR is biased, making it unsuitable for the neural network inference system. To overcome this limitation, we proposed a new support vector neural network inference system (SVNNIS). In contrast to the SVR, the output of the SVNNIS is unbiased. As the SVNNIS is a linear-in-weight network, the weights can be computed by the recursive least square method with forgetting factor.

This paper is organized as follows. Section 2 briefly introduced the fundamental ideas for SVM. In section 3, the new SVNNIS is proposed and its training process is described. Section 4 gives experimental results. Those results all showed the superiority of SVNNIS to the original SVR.
Conclusions are given in Section 5.

2. SUPPORT VECTOR MACHINES

The structure of the network and the training algorithm are important factors that affect the performance of the neural network. It is possible to reduce the modeling errors by increasing the complexity of the network. However, increasing the complexity of the network may overfit the data, leading to a degradation of its performance and reduced network transparency. between modeling errors and the complexity of the network. A principle generally adopted is to choose a network with the simplest structure, yet giving an acceptable precision. In [2] SVM are proposed with the network structure selected to satisfy a given precision. Suppose we have given data $\{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \cdots, (\vec{x}_N, y_N)\}$ with $\vec{x}_i \in R^m$ and $y_i \in R$, where N is the number of training data, x_i is the *ith* input vector, and y_i is the desired output for the input $\vec{x}_i$. The SVM for estimating the nonlinear function $f(\cdot)$ is

$$f(\vec{x}) = \sum_{i=1}^{N} (\alpha_i^* - \alpha_i) K(\vec{x}_i, \vec{x}) + b \qquad (1)$$

α_i^* and α_i are lagrange multipliers. The kernel function $K(x_i, x)$ is defined as a linear dot product of the nonlinear mapping,

$$K(\vec{x}_i, \vec{x}) = \phi(\vec{x}_i) \cdot \phi(\vec{x}) \qquad (2)$$

The coefficients α_i^* and α_i of (1) are obtained by minimizing the following regularized risk functional $R_{eg}[f]$, which is a combination of the model complexity and the empirical risk, for given error bound ε,

$$R_{eg}[f] = \frac{1}{2}\|w\|^2 + C \cdot \sum_{i=1}^{l} L_\varepsilon(y) \qquad (3)$$

Here, $\|w\|^2$ is a term which characterizes the model complexity, C is a constant determining the trade-off and the ε-insensitive loss function $L_\varepsilon(y)$ is given by

$$L_\varepsilon(y)=\begin{cases} 0 & for\left|f(\vec{x})-y\right|<\varepsilon \\ \left|f(\vec{x})-y\right|-\varepsilon & otherwise \end{cases} \qquad (4)$$

The minimization of regularized risk function in (3) can be converted to the following constrained optimization problem,

$$\left.\begin{array}{l} \min\limits_{\alpha,\alpha^*} w(\alpha,\alpha^*)=\min\limits_{\alpha,\alpha^*}\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\alpha_i^*-\alpha_i)(\alpha_j^*-\alpha_j)K(\vec{x}_i,\vec{x})-\sum_{i=1}^{N}(\alpha_i^*-\alpha_i)y_i+\varepsilon\sum_{i=1}^{N}(\alpha_i^*-\alpha_i) \\ \qquad subject\ to \begin{cases} \sum_{i=1}^{N}(\alpha_i^*-\alpha_i)=0 \\ \alpha,\alpha^*\in[0,c] \end{cases} \end{array}\right\} (5)$$

where the kernel function used is Gaussian and defined as

$$k(\vec{x}_i,\vec{x})=\exp(-\frac{\left\|\vec{x}-\vec{x}_i\right\|^2}{2\sigma^2}) \qquad (6)$$

where σ is a constant. Note that only some of lagrange multipliers are not zeros and the corresponding vectors are called the support vectors.

3. NEW SUPPORT VECTOR NEURAL NETWORK INFERENCE SYSTEM

In this section, the proposed SVNNIS is based on the multivariate nonlinear system, which is given by $y=f(x_1,x_2,\cdots,x_N)$. Now, we describe the learning algorithm adopted to train the SVNNIS, which is divided into two phases, the initial phase and the learning phase.

The initial phase is to determine the network inference system structure and the corresponding initial network weights through the SVR theory. When the cost function in (4) and the kernel functions in (6) are chosen, the initial weights and the structure of SVNNIS can be determined by the SVR theory as stated in the previous section. In this paper, After applying the SVR theory, an initial SVNNIS is obtained as

$$\hat{y}=\sum_{i=1}^{l}\beta_i K(\vec{x},\vec{x}_i)+b \qquad (7)$$

where $\beta_i = \alpha_i^* - \alpha_i$, $\hat{y}$ is the output of the SVNNIS, $K(x, x_i)$ is a kernel function of the SVR theory, l is the number of kernel functions, which is equivalent to the number of support vectors, and $\beta = [\beta_1, \beta_2, \cdots, \beta_l]$ is the weight vector of the network. Since the SVR given by (8)can be considered as a two-layer neural network linear in its weights, it is intuitive to reformulate it as a RBF network using normalized basis functions. Now, Let's review the RBF network. First, the RBF network consists of m inputs with each input represented by n_i univariate basis functions, the number of weights is $n = \prod n_i$. Let the centers of the basis function can be chosen at the input data, $\vec{x}_i = [x_{i1}, \cdots, x_{im}]$, This is not restrictive, as the centers of the basis functions for this class of RBF networks can be chosen freely. Then the RBF network is given by

$$y = \frac{\sum_{i=1}^{n} \theta_i \mu_i(\vec{x}_i')}{\sum_{i=1}^{n} \mu_i(\vec{x}_i)} \qquad (8)$$

where θ_i is the *ith* weight; and $\mu_i(\cdot)$ is the *ith* Gaussian basis function given by

$$\mu_i(\vec{x}_i) = \exp\left(- \frac{|\vec{x}_i - \vec{x}_i'|^2}{2\gamma^2} \right) \qquad (9)$$

where γ and $\vec{x}'$ are, respectively the spread chosen to ensure a thorough coverage of the input space, and the center of the Gaussian function. The centers $\vec{x}'$ can also be considered as the fuzzy basis vector [10], since $\mu_i(\cdot)$ is the fuzzy membership function. The *ith* multivariate basis function is a product of the univariate basis functions obtained from the inputvariables, i.e.

$$\mu_i(\vec{x}_i) = \prod_{j=1}^{m} \mu_{i,j}(x_{ij}) \qquad (10)$$

Where $\mu_{i,j}(x_{ij}) = \exp(-(x_{ij} - x_{ij}')^2 / 2\gamma^2)$. The centers $\vec{x}'$ are chosen from the input data $\vec{x}$. From (10), $\mu_i(\vec{x}') = 1$. It is assumed for simplicity that each input has the same number of basis functions, i.e.,

$$n = n_1 = \cdots = n_m \qquad (11)$$

If (8) can be reformulated to the form of (9), the weights can also be estimated using linear least-squares method to overcome the biased problem of the SVR. For convenience, let the weights of the network be denoted by $[\theta_1', \cdots, \theta_{n+1}']^T$.

Then (8) can be rewritten as

$$y_i = \sum_{j=1}^{n} \theta'_j K_j(\vec{x}'_i) + \theta'_{n+1} \tag{12}$$

Let

$$\lambda(\vec{x}_i) = \sum_{j=1}^{n} K_j(\vec{x}_i) \tag{13}$$

$$\psi = \begin{bmatrix} K_1(\vec{x}_1) & \cdots & K_n(\vec{x}_N) \\ \vdots & & \vdots \\ K_1(\vec{x}_1) & \cdots & K_n(\vec{x}_N) \end{bmatrix} \tag{14}$$

and

$$L = \lambda(\vec{x}_i) \tag{15}$$

As the kernels in (13) are multivariate basis functions given by (9) and the SV are the respective centers, the two-layer neural network given by (13) can be rewritten as a RBF network. In matrix form, (13) becomes

$$Y = \psi \begin{bmatrix} \theta'_1 \\ \vdots \\ \theta'_n \end{bmatrix} + \begin{bmatrix} \theta'_{n+1} \\ \vdots \\ \theta'_{n+1} \end{bmatrix}, \tag{16}$$

where $Y = [y_1, \cdots, y_N]^T$. Since $\sum_{j=1}^{n} K_j(\vec{x}_i) / \lambda(\vec{x}_i) = 1$, then

$$L^{-1}\psi \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}. \tag{17}$$

Eq.(17) can be rewritten as

$$Y = \psi \begin{bmatrix} \theta'_1 \\ \vdots \\ \theta'_n \end{bmatrix} + L^{-1}\psi \begin{bmatrix} \theta'_{n+1} \\ \vdots \\ \theta'_{n+1} \end{bmatrix} = L^{-1}\psi \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}, \tag{18}$$

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

$$\psi\begin{bmatrix}\theta_1\\ \vdots\\ \theta_n\end{bmatrix} = L\psi\begin{bmatrix}\theta'_1\\ \vdots\\ \theta'_n\end{bmatrix} + \psi\begin{bmatrix}\theta'_{n+1}\\ \vdots\\ \theta'_{n+1}\end{bmatrix}, \qquad (19)$$

Rearranging (20), the weights $\theta_1,\cdots,\theta_n$ can be expressed in terms of $\{\theta'_1,\cdots,\theta'_{n+1}\}$ as

$$\begin{bmatrix}\theta_1\\ \vdots\\ \theta_n\end{bmatrix} = (\psi^T\psi)^{-1}\psi^T L\psi\begin{bmatrix}\theta'_1\\ \vdots\\ \theta'_n\end{bmatrix} + \begin{bmatrix}\theta'_{n+1}\\ \vdots\\ \theta'_{n+1}\end{bmatrix}, \qquad (20)$$

From (21), the following network is obtained.

$$y_i = \frac{\sum_{j=1}^{n}\theta_j K_i(\vec{x}_i)}{\sum_{j=1}^{n} K_i(\vec{x}_i)} = \sum_{j=1}^{n}\theta_j N_j(\vec{x}_i), \qquad (21)$$

Where $N_j(\vec{x}_i) = K_j(\vec{x}_i) / \lambda(\vec{x}_i)$ is the normalized basis function. For convenience, the network given by (22) is referred to as the support vector neural network inference system (SVNNIS). The architecture of the SVNN is shown in Figure 1.

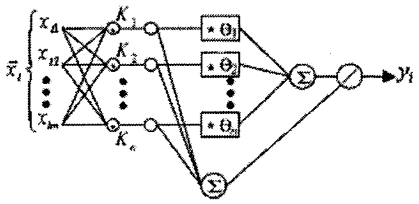


Figure 1. The architecture of the SVNNIS

In the second phase, the SV of the SVVNNIS are obtained by minimizing (5), the weights $\{\theta_1,\cdots,\theta_n\}$ are to be adjusted via the recursive least square method with forgetting factor algorithm. Rewriting (22) in matrix form gives

$$y_i = B^T(\vec{x}_i)\theta \qquad (22)$$

where $B(\vec{x}_i)=[N_1(\vec{x}_i),\cdots,N_n(\vec{x}_i)]^T$ and $\theta=[\theta_1,\cdots,\theta_n]^T$. The cost function is

$$E(t)=\sum_{j=1}^{n}\alpha^{n-1}(y_i-\hat{y}_i(t))^2, \tag{23}$$

$$\hat{y}_i(t)=B^T(\vec{x}_i)\hat{\theta} \tag{24}$$

The linear least-squares estimate of the weights $\hat{\theta}$ is

$$\hat{\theta}=[\phi^T\phi]^{-1}\phi^TY, \tag{25}$$

Where

$$\phi=[N_1(\vec{x}_i),\cdots,N_n(\vec{x}_i)] \tag{26}$$

and

$$Y=[y_1,\cdots,y_n]^T \tag{27}$$

The learning algorithm is Where t is the epoch number, $\alpha\in(0,1)$ is a forgetting factor, $\hat{y}_i$ is the estimate of the y_i given by

$$\hat{\theta}(t+1)=\hat{\theta}(t)+\eta(t+1)\varepsilon^T(t+1), \tag{28}$$

Where

$$\eta(t+1)=\frac{\alpha^{-1}P(t)\vec{x}_i(t+1)}{1+\alpha^{-1}\vec{x}_i^{\,T}(t+1)P(t)\vec{x}_i(t+1)},$$
$$\varepsilon(t+1)=y_i-\hat{y}_i(t+1),$$
$$P(t+1)=\alpha^{-1}P(t)-\alpha^{-1}\eta(t+1)\vec{x}_i(t+1)P(t),$$
$$P(0)=I.$$

The SVNNIS learning algorithm is summarized as follows.
Step1) For a given set of training data $\{(\vec{x}_i,y_i),i=1,2,\cdots,N\},\vec{x}_i\in R^n,y_i\in R;$ a set of testing data $\{(\vec{x}_k,y_k),k=1,2,\cdots,N\},\vec{x}_k\in R^n,y_k\in R;$.
Step 2) Initialize the SVNNIS structure with the given kernel function, the loss function, the threshold ε_R used for the determining the termination condition of the learning algorithm.
Step 3) The support vectors are obtained by the sequential minimal optimization (SMO) algorithm, so we can construct a SVNNIS.
Step 4) For each training pattern, compute the estimated result by (8) and its error.

Step 5) Update the weight vector $\hat{\theta}$ incrementally by (28).

Step 6) Compute the error defined by (23).

Step 7) If the termination conditions are not satisfied, then go to step 4; otherwise, termin ate the learning process.

4. SIMULATION RESULTS

The simulations were conducted in the Matlab environment. The SVR toolbox provided by Gunn [11]. In this study, the results of various cases with different loss functions (the ε - insensitive function with 0, 0.01), different σ s, and different C s are presented for illustration. The forgetting factor α used in the simulation is 0.95. To analysis the performance of the proposed SVNNIS, the modeling error is defined by as Root Mean Square Error (RMSE):

$$E = \sqrt{\frac{\sum_{k=1}^{N} (y_k - \hat{y}_k)}{N}} \qquad (29)$$

where N is the number of data, y_k and $\hat{y}_k$ are the system and the model output.

In this paper, the functions is defined as

$$y = x^{2/3} \quad with \quad x \in [-2,2]. \qquad (30)$$

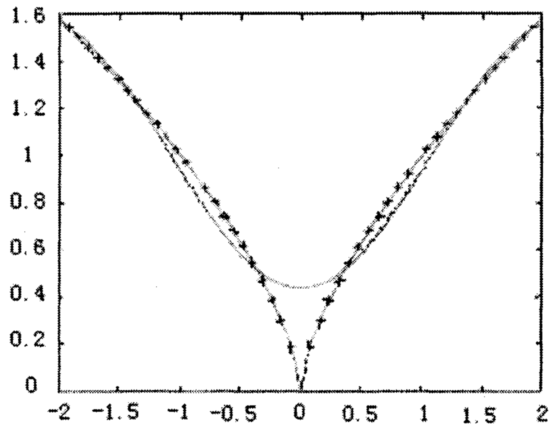
The test data sets with 200 patterns are also generated for both examples. The testing RMSEs of the SVR are obtained for various loss functions with different parameters sets $\{\sigma, C\}$. Detailed results are shown in [12]. From the simulations it can found that the parameter sets for the best performance in different loss functions are different. For example, in the case of using the ε -insensitive function with ε =0.1 as the loss function, when $\{\sigma, C\}$={0.5,5}, the performance is the best and the testing RMSE is 0.0743. For this case, after the SVNNIS learning, the RMSE becomes 0.0484. The testing RMSEs of SVR and of SVNNIS with ε =0.1

Table 1.

	C=0.5	**C=1**	**C=5**	**C=10**	**C=50**	**C=100**
$\tau = 0.5$	0.09153	0.08548	0.07438	0.09071	0.09478	0.09736
	0.03358	**0.07348**	**0.0484**	**0.07546**	**0.09355**	**0.9668**
$\tau = 0.7$	0.04112	0.02381	0.02582	0.02901	0.03919	0.04545
	0.01874	**0.01475**	**0.01220**	**0.01603**	**0.02688**	**0.03334**
$\tau = 1$	0.01801	0.01515	0.01513	0.01609	0.01842	0.01941
	0.00205	**0.00300**	**0.00638**	**0.00793**	**0.01091**	**0.01207**
$\tau = 2$	0.01506	0.01073	0.00779	0.00784	0.00892	0.00946
	0.00051	**0.00078**	**0.00217**	**0.00301**	**0.00506**	**0.00579**
$\tau = 3$	0.04518	0.02276	0.00986	0.00815	0.00675	0.00658
	0.01218	**0.00804**	**0.00365**	**0.00278**	**0.00202**	**0.00189**

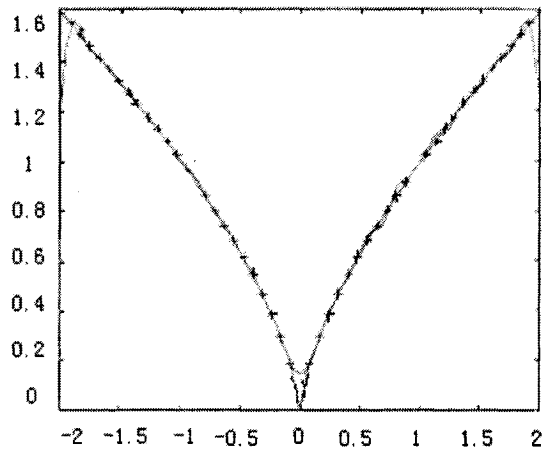
The learned results of SVR and SVNNIS are respectively displayed in

Figure 2(a) and (b) for illustration. They are all the cases of using the ε-insensitive function with ε =0 and $\{\sigma, C\}$={2,100}. These results match with the concept discussed in[13]. It can be found that the proposed SVNNIS can reduce the overfitting phenomena.



(a)

Figure 2. (a). Result obtained by SVR for function y=$x^{2/3}$



(b)

Figure 2. (b). Result obtained by SVNNIS for function y=$x^{2/3}$

5. CONCLUSIONS

In this paper, a new support vector neural network inference system was proposed to enhance the generalization capability of the SVR approaches. The basis idea of the approach is to adopt the structure similar with that of the SVR, while the output of the SVNNIS is unbiased compared with the SVR, the weights can be updated by the recursive least square method with forgetting factor. The advantages of this system are their good generalization capabilities. The simulation result illustrates the effectiveness of the proposed SVNNIS.

REFERENCES

1. C. Cortes and V. Vapnik, "Support vector networks," Machine Learning, vol. 20, pp. 273–297, 1995.
2. V. Vapnik, The Nature of Statistical Learning Theory. New York: Springer-Verlag, 1995.
3. S. Mukherjee, E. Osuna, and F. Girosi, "Nonlinear prediction of chaotic time series using a support vector machine," in Proc. NNSP, 1997, pp.24–26.
4. H. Drucker et al., "Support vector regression machines," in Neural Information Processing Systems. Cambridge, MA: MIT Press, 1997, vol. 9.
5. V. Vapnik, S. Golowich, and A. J. Smola, "Support vector method for function approximation, regression estimation, and signal processing,"in Neural Information Processing Systems. Cambridge, MA: MIT Press, 1997, vol. 9.
6. A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," Royal Holloway College, London, U.K., Neuro COLT Tech. Rep.TR-1998-030, 1998.
7. A. J. Smola, B. Schölkopf, and K. R. Müller, "General cost functions for support vector regression," presented at the ACNN, Australian Congr.Neural Networks, 1998.
8. A. J. Smola, "Regression estimation with support vector learning machines,"thesis, Technical Univ. Munchen, Munich, Germany,1998.
9. Jongcheol Kim*, sangchul Won* "New Fuzzy Inference System using a Support Vector Machine" Proceeding of the 41st IEEE Conference on Decision and Control,12 2002
10. Harris, C.J., Wu, Z.Q., Gan, Q., 1999. Neurofuzzy state estimators and their applications. In: Sinha, N.K., Gupta, M.M. (Eds.), Soft Computing and Intelligent Systems. Academic Press, New York, pp. 377 - 402.
11. S. R. Gunn. (1999) Support vector regression-Matlab toolbox. Univ. Southampton, Southampton, U.K.[Online]. Available: http://kernel-machines.org.
12. C.C. Chuang, "Robust modeling for function approximation under outliers," Ph.D.dissertation, Dept. Electr. Eng., Nat. Taiwan Univ. Sci. Technol., Taipei, 2000.
13. F. E. H. Tay and L. Cao, "Application of support vector machines in financial time series forecasting," in Int. J. Manage. Sci., 2001, pp.309–317.