A fast identification algorithm with outliers under box-cox transformation-based annealing robust radial basis function networks

Pi-Yun Chen [1], Chia-Ju Wu[2], Chia-Nan Ko [3] and Jin-Tsong Jeng[4]

[1]Graduate School of Engineering Science and Technology, National Yunlin University of Science and Technology
Douliou, Yunlin 640, Taiwan
[2]Department of Electrical Engineering,National Yunlin University of Science and Technology
Douliou, Yunlin 640, Taiwan
[3]Department of Automation Engineering, Nan Kai University of Technology
Tsaotun, Nantou 542, Taiwan
[4]Department of Computer Science and Information Engineering, National Formosa University
Huwei, Yunlin 632, Taiwan
(email:[1]g9310820@yuntech.edu.tw;[2]WUCJ@yuntech.edu.tw;[3]t105@nkut.edu.tw;[4]tsong@nfu.edu.tw)

Abstract: In this paper, a Box–Cox transformation-based annealing robust radial basis function networks (ARRBNFs) is proposed for the identification algorithm with outliers. Firstly, a fixed Box–Cox transformation-based ARRBFNs model with support vector regression (SVR) is derived to determine the initial structure. Secondly, the results of the SVR are used as initial structure in the fixed Box–Cox transformation-based ARRBFNs for the identification algorithm with outliers. At the same time, an annealing robust learning algorithm (ARLA) is used as the learning algorithm for the fixed Box–Cox transformation-based ARRBFNs, and applied to adjust the parameters and weights. Hence, the fixed Box–Cox transformation-based ARRBFNs with ARLA have fast convergence speed for the identification algorithm with outliers. Finally, the proposed algorithm and its efficacy are demonstrated with an illustrative example in comparison with Box–Cox transformation-based radial basis function networks.

Keywords: identification algorithm, Box–Cox transformation, outliers, annealing robust radial basis function networks

I. INTRODUCTION

The standard radial basis function networks (RBFNs) consists of three layers; namely, the input layer, the hidden layer, and the output layer. Due to the RBFNs structural simplicity, it has been widely used for nonlinear function approximation [1] and system identification [2]. For the different applications the learning algorithms of the RBFN are designed with different optimization criteria. One of the main applications that the RBFNs has been applied to system dynamic modeling [2]. Besides, it is well known that the well conditioning of a model base is critically important in order to obtain good parameter estimators for the identification problem. Hence, identification algorithm has many important applications including nonlinear control systems, robotic systems, etc.

In general, the Box-Cox transformation on the system output is one of the major statistical techniques Box and Cox [3] to reduce heteroscedasticity when the distribution of the dependent variable is unknown. Hong [4] proposed that a RBFNs model base is derived based on a rank revealing orthogonal matrix triangularization; namely, QR decomposition Hong and Pan [5]. Besides, the identification algorithm uses Gauss-Newton method to derive the Box-Cox transformation parameter. For a large data set, using the QR decomposition increases computational expense for model structure. On the other hand, for the scientific and engineering applications, the obtained training data are always subject to outliers. The intuitive definition of an outlier Hawkins [6] is "*an observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism.*" However, outliers may occur due to various reasons, such as erroneous measurements or noisy data from the tail of noise distribution functions. When the outliers are exists, there still exist some problems in the algorithm of Hong [4].

In this paper, a fast identification algorithm with outliers is introduced for the Box-Cox transformation-based ARRBFNs. Firstly, the support vector regression (SVR) is derived to determine the initial structure. Because of a SVR approach is equivalent to solving a linear constrained quadratic programming problem under a fixed structure of SVR, the number of hidden nodes, the initial parameters and the initial weights of the ARRBFNs are easy obtained via the SVR approach. Secondly, an annealing robust learning algorithm (ARLA) is used as the learning algorithm for the fixed Box–Cox transformation-based ARRBFNs, and applied to adjust the parameters and weights.

II. BOX-COX TRANSFORMATION-BASED ARRBFNS

Given a data set $\{(\vec{x}_i, y_i), i = 1,2,\cdots,N\}$, where $\vec{x}_i$ is the system input vector and y_i is the positive system output. N is the number of data samples. The objective of Box-Cox transformation is usually to make residuals more homogeneous in regression, or transform data to be normally distributed. The well known Box-Cox version of power transformation Box and Cox [4] is formed as

$$z(y,\lambda)=\begin{cases}\dfrac{y^{\lambda}-1}{\lambda\tilde{y}^{\lambda-1}}, & \text{if } \lambda\neq 0\\ \tilde{y}\log y, & \text{if } \lambda=0\end{cases}, \quad (1)$$

where λ is the transformation parameter and $\tilde{y}=\sqrt[N]{\prod_{i=1}^{N} y_i}$ is the geometric mean of the system output. For a given λ, an RBFN with a single output can be represent as Hong [4]

$$z(y,\lambda)= f(\vec{x},W)+e=\sum_{j=1}^{L} w_j G_j(\vec{x})+b+e, \quad (2)$$

where $f(\vec{x})$ is the output of the Box-Cox transformation-based ARRBFNs Chuange et al [7], e is model error, $W=[w_1,w_2,\cdots,w_L]^T$ is the synaptic weight vector, and L is the number of hidden layer in the Box-Cox transformation-based ARRBFNs. The radial basis functions G_j are chosen as Gaussian functions that it can be express in the form

$$G_j(\vec{x})=\exp\left\{\frac{-\left\|\vec{x}-m_j\right\|^2}{2\sigma_j^2}\right\}, \quad (3)$$

where m_j and σ_j are the center and width of Gaussian functions, respectively. Hence, the $f(\cdot)$ can be rewritten as

$$f(\vec{x},W)=\sum_{j=1}^{L} w_j \exp\left\{\frac{-\left\|\vec{x}-m_j\right\|^2}{2\sigma_j^2}\right\}+b, \quad (4)$$

Note that

$$\lim_{\lambda\to 0} z(y,\lambda)=\lim_{\lambda\to 0}\left[\frac{y^{\lambda}-1}{\lambda\tilde{y}^{\lambda-1}}\right]=\tilde{y}\log y \quad (5)$$

and the inverse of Box-Cox transformation upon $f(\vec{x},W)$ for given $\lambda\neq 0$ and W is

$$\hat{y}=z^{-1}\left(f(\vec{x},W)\right)=\sqrt[\lambda]{1+\lambda\tilde{y}^{\lambda-1} f(\vec{x},W)}. \quad (6)$$

That is, when $\lambda=0$, $\hat{y}=\exp\left(\dfrac{f(\vec{x},W)}{\tilde{y}}\right)$ in the proposed approach. The proposed structure is shown in Fig. 1.

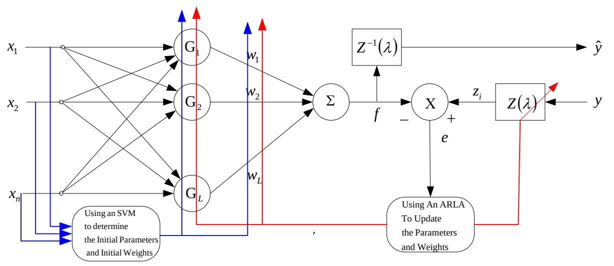

Fig.1. Box-Cox transformation-based ARRBFNs.

2.1 The initial structure of Box-Cox transformation-based ARRBFNs by the SVR approach

An SVR approach is used to approximate an unknown function from a set of samples $\{(\vec{x}_i,z_i), i=1,2,\cdots,N\}$, where the system output y_i is replaced by the normalized transformed response z_i. Assuming that a set of basis functions $\{g_k(\vec{x}), k=1,2,\cdots,m\}$ is given, there exists a family of functions that can be expressed as a linear expansion of the basis function. Then, the problem of function approximation transforms into that finding the parameters of the following basis function linear expansion:

$$f(\vec{x},\vec{\theta})=\sum_{k=1}^{m}\theta_k g_k(\vec{x})+b \quad (7)$$

where $\vec{\theta}\in(\theta_1,\ldots,\theta_m)$ is a parameter vector to be identified and b is a constant. Then, the solution for the problem is to find f that minimizes

$$R(\vec{\theta})=\frac{1}{n}\sum_{i=1}^{n} L_{\varepsilon}\left(y_i - f(\vec{x}_i,\vec{\theta})\right), \quad (8)$$

subject to the constraint

$$\left\|\vec{\theta}\right\|^2 < C, \quad (9)$$

where $L_{\varepsilon}(\cdot)$ is the ε-insensitive loss function and defined as

$$L_{\varepsilon}(e)=\begin{cases}0, & \text{for } |e|\leq\varepsilon,\\ |e|-\varepsilon, & \text{otherwise,}\end{cases} \quad (10)$$

for some previously chosen nonnegative number ε. In (9), the constraint is imposed to trade off the complexity of the solution.

By using the Lagrange multiplier method, it can be shown Vapnik [8] that the minimization of (8) leads to the following dual optimization problem, minimize

$$\begin{aligned}Q(\alpha,\alpha^*)=\varepsilon\sum_{r=1}^{N}(\alpha_r+\alpha_r^*)-\sum_{r=1}^{N} y_r(\alpha_r^*-\alpha_r)\\ +\frac{1}{2}\sum_{r,s=1}^{N}(\alpha_r^*-\alpha_r)(\alpha_s^*-\alpha_s)\left[\sum_{k=1}^{m} g_k(\vec{x}_r)g_k(\vec{x}_s)\right],\end{aligned} \quad (11)$$

subject to the constraint

$$\sum_{r=1}^{N}\alpha_r^*=\sum_{r=1}^{N}\alpha_r, \quad 0<\alpha_r,\alpha_r^*<C \quad \text{for } r=1,\ldots,N. \quad (12)$$

In (11), the inner product of basis functions $g_k(\vec{x})$ is replaced via the kernel function

$$K(\vec{x}_r,\vec{x}_s)=\sum_{k=1}^{m} g_k(\vec{x}_r)g_k(\vec{x}_s). \quad (13)$$

The kernel function determines the smoothness properties of solutions and should reflect a prior knowledge of the data. In the literature, the polynomials, B-spline and Gaussian kernel function often used Vapnik et al [9]. Hence the optimization of (11) is rewritten as

$$\begin{aligned}Q(\alpha,\alpha^*)=\varepsilon\sum_{r=1}^{N}(\alpha_r+\alpha_r^*)-\sum_{r=1}^{N} y_r(\alpha_r^*-\alpha_r)\\ +\frac{1}{2}\sum_{r,s=1}^{N}(\alpha_r^*-\alpha_r)(\alpha_s^*-\alpha_s)K(\vec{x}_r,\vec{x}_s),\end{aligned} \quad (14)$$

It was shown in Vapnik et al [9] that the solution of the SVR approach is in the form of the following linear expansion of kernel functions (i.e. the parameter θ_i in

(8) can be represented as $\sum_{i=1}^{m}(\alpha_k^* - \alpha_k)g(\vec{x}_i)$),

$$f(\vec{x},\alpha,\alpha^*) = \sum_{k=1}^{m}(\alpha_k^* - \alpha_k)K(\vec{x}_r,\vec{x}_s) + b \,. \tag{15}$$

Note that only some of $(\alpha_k^* - \alpha_k)$'s are not zeros and the corresponding vectors $\vec{x}_k$'s are called SVs. In this paper, the Gaussian function is used as the kernel function. Hence, (15) can be rewritten as

$$f(\vec{x},\vec{\upsilon}) = \sum_{k=1}^{SV}\upsilon_k \exp\left\{\frac{-\left\|\vec{x}-\vec{x}_k\right\|^2}{2\sigma^2}\right\} + b\,, \tag{16}$$

where SV is the number of SVs, $\upsilon_k = (\alpha_k^* - \alpha_k) \neq 0$ and $\vec{x}_k$ are SVs. On comparing (16) with (4), the SV, k, υ_k and $\vec{x}_k$ in (16) can be regarded as the L, j, w_j and m_j in (4), respectively. That is, based on Eqs. (4) and (16), the initial weight w_j, the number of hidden node L, and the parameters of the proposed neural network in Fig. 1 can be determined via an SVR method.

2.2 The annealing robust learning algorithm of Box-Cox transformation-based ARRBFNs

In the Box-Cox transformation-based ARRBFNs, an ARLA is proposed as a learning algorithm. An important feature of the ARLA that adopts the annealing concept in the cost function of robust back-propagation learning algorithm is proposed in Chuang et al [10]. Hence, the ARLA can overcome the existing problems in robust back-propagation learning algorithm. A cost function for the ARLA is defined here as

$$E(\vec{x}_i,t) = \frac{1}{N}\sum_{i=1}^{N}\rho\left[e_i(\vec{x}_i,t);\beta(t)\right], \tag{17}$$

where

$$e_i(\vec{x}_i,t) = z_i(y,\lambda) - \sum_{j=1}^{N} w_j \exp\left\{\frac{-\left\|\vec{x}_i - m_j\right\|^2}{2\sigma_j^2}\right\}, \tag{18}$$

t is the epoch number, $e_i(\vec{x}_i,t)$ is the error between the ith Box-Cox transformation of desired output and the ith output of the proposed approach at epoch t, $\beta(t)$ is a deterministic annealing schedule acting like the cut-off points and $\rho(\cdot)$ is a logistic loss function and defined as

$$\rho[e_i;\beta] = \frac{\beta}{2}\ln\left[1+\left(\frac{e_i^2}{\beta}\right)\right]. \tag{19}$$

Based on the gradient-descent kind of learning algorithms, the synaptic weights w_j, centers m_j, width σ_j of Gaussian function and λ_j of Box-Cox transformation parameter are updated as

$$\Delta w_j = -\eta\frac{\partial E}{\partial w_j} = -\eta\sum_{i=1}^{N}\varphi(e_i;\beta)\frac{\partial e_i}{\partial w_j}\,, \tag{20}$$

$$\Delta m_j = -\eta\frac{\partial E}{\partial m_j} = -\eta\sum_{i=1}^{N}\varphi(e_i;\beta)\frac{\partial e_i}{\partial m_j}\,, \tag{21}$$

$$\Delta\sigma_j = -\eta\frac{\partial E}{\partial \sigma_j} = -\eta\sum_{i=1}^{N}\varphi(e_i;\beta)\frac{\partial e_i}{\partial \sigma_j}\,, \tag{22}$$

$$\Delta\lambda_j = \frac{\partial z(y,\lambda_j)}{\partial\lambda_j}\,, \tag{23}$$

$$\varphi(e_i;\beta) = \frac{\partial\rho(e_i;\beta)}{\partial e_i} = \frac{e_i}{1+e_i^2/\beta(t)}\,, \tag{24}$$

where η is a learning constant and $\varphi(\cdot)$ is usually called the influence function. When outliers exist, they have a major impact on the approximated results. Such an impact can be understood through the analysis of the influence function. The learning algorithm of the proposed approach is summarized as follows:

Step 1: Using Box-Cox transformation by (1) to form the transformed output.

Step 2: Initialize the Box-Cox transformation-based ARRBFNs structure using an SVR approach that is described by (16) with the given Gaussian kernel functions, the ε-insensitive function and the constant C.

Step 3: Compute the Box-Cox transformation z_i and its error by (18) for all training data.

Step 4: Update the synaptic weights w_j, the centers m_j, the width σ_j of Gaussian function and the λ_j of the Box-Cox transformation parameter are iteratively updated by (20)~(24), respectively. In this process, the influence of the outliers is detected and discriminated.

Step 5: Determine the values of the annealing schedule $\beta(t) = k/t$ for each epoch, where k is set as $2\cdot\max\left\{\left|e_i\right|_{initial}\right\}$.

Step 6: Compute the robust cost function E defined by (17).

Step 7: If the termination conditions are not satisfied, then go to *Step* 3; otherwise terminate the learning process.

Step 8: The inverse of the Box-Cox transformation (6) is applied to the Box-Cox transformation-based ARRBFNs model output as the system output predictions $\hat{y}$.

III.SIMULATION RESULTS

The simulations were conducted in the *Matlab* environment. The root mean square error (RMSE) of the testing is used to measure the performance of the learned networks. The RMSE is defined as

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}{N}}\,, \tag{25}$$

where y_i is the desire value at x_i and $\hat{y}_i$ is the output of the Box-Cox transformation-based ARRBFNs.

Example. The *sinc* function is considered and defined as

$$y(x) = \left(\frac{\sin x}{x} + \xi\right) + 5\,,\quad -10 \le x \le 10\,. \tag{26}$$

Because the output y must be positive in the Box-Cox transformation, the output of *sinc* function must shifts up 5. Ten hundred training data $y(x)$ were generated by using uniformly distributed random $x \in [-10, 10]$. Besides, the noise is a normal disturbance with $N(0,1)$ and one hundred artificial outliers are added to the *sinc* function. Firstly, an initial structure of the Box-Cox transformation-based ARRBFNs is obtained by an SVR approach. The parameters in the SVR are set as $C = 3$, Gaussian kernel function with $\sigma = 0.3$ and $\varepsilon = 0.05$. The initial structure of the Box-Cox transformation-based ARRBFNs with the hidden nodes (i.e. the number of SVs) are obtained as 149. Secondly, the parameters of the Box-Cox transformation-based ARRBFNs are adjusted by the ARLA. After 100 epochs using the ARLA, the testing RMSE of the Box-Cox transformation-based ARRBFNs is 0.0263, as shown in Fig. 2. For a comparison study, a Box–Cox transformation-based radial basis function networks Hong [4] was constructed for the same data, but the testing RMSE is 0.4704, as shown in Fig. 3. Besides, the results of comparison with different training data and artificial outliers are shown in Table 1. From the simulation results, the proposed robust learning algorithms could indeed improve the learning performance as the training data contain outliers.

Table 1: The results with different training data and artificial outliers are shown.

The number of training data	The number of artificial outliers	RMSE	
		Box-Cox transformation-based RRBFNs	Box-Cox transformation-based RBFNs Hong [8]
100	10	0.0549	0.4721
1000	100	0.0263 (Fig.2)	0.4704 (Fig.3)
5000	500	0.0270	0.5131
8000	800	0.0254	Out of memory
10000	1000	0.0434	Out of memory

IV. CONCLUSIONS

In this paper, we proposed a fast identification algorithm with outliers, namely the fixed Box-Cox transformation-based ARRBFNs. Using an SVR approach determines the number of hidden nodes, the initial parameters of the kernel, and the initial weights of the proposed neural networks. At the same time, an ARLA is applied to adjust the parameters and weights. Finally, from the simulation results show that the fixed Box–Cox transformation-based ARRBFNs with ARLA have fast convergence speed for the identification algorithm with outliers.

ACKNOWLEDGMENTS

This work was supported in part by the National Science Council, Taiwan, R.O.C., under grants NSC96-2628-E-224-007-MY2.

REFERENCE

[1] Hong X (2007), Modified radial basis function neural network using output transformation, IET Control Theory & Applications 1(1): 1-8.

[2] Hong X, Harris CJ (2003), Experimental design and model construction algorithms for radial basis function networks, Int. J. Syst. Sci. 34(14-15): 733-745.

[3] Box GEP, Cox DR (1964), An analysis of transformation (with discussion), Journal of the Royal Statistical Society, Series B, 26(2):211-252.

[4] Hong X (2006), A fast identification algorithm for Box-Cox transformation based radial basis function neural network, IEEE Trans. Neural Networks 17(4):1064-1069.

[5] Hong YP, Pan CT (1992), Rank-revealing QR factorizations and the singular value decomposition, Math. Comput. 58(192):213-232.

[6] Hawkins DM (1980), Identification of Outliers, Chapman and Hall.

[7] Chuange CC, Jeng JT, Lin PT (2004), Annealing robust radial basis function networks for function approximation with outliers, Neurocomputing, 56(1):123-139.

[8] Vapnik V (1995), The Nature of Statistical Learning Theory, Springer, Berlin.

[9] Vapnik V, Golowich S, Smola AJ (1997), Support vector method for function approximation, regression estimation, and signal processing, Neural Inf. Process. Syst. 9:281-287.

[10] Chuang CC, Su SF, Hsiao CC (2000), The annealing robust backpropagation (BP) learning algorithm, IEEE Trans. Neural Networks 11(5):1067-1077.

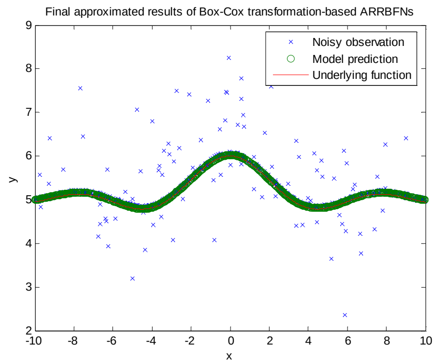

Fig.2. The final output of the proposed approach after 100 epochs uses the ARLA for Example (RMSE=0.0263).

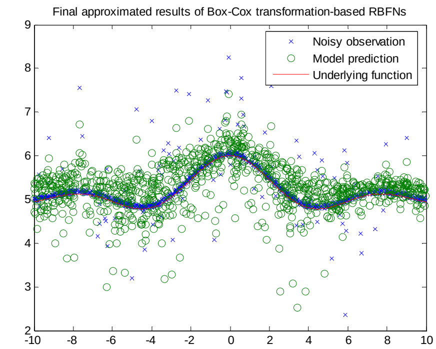

Fig.3. The final output of the Box-Cox transformation-based RBFNs for Example (RMSE =0.4704).