Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 970931, 8 pages
http://dx.doi.org/10.1155/2014/970931

Research Article

Multilayer Perceptron for Robust Nonlinear Interval Regression Analysis Using Genetic Algorithms

Yi-Chung Hu

Department of Business Administration, Chung Yuan Christian University, Chung Li 32023, Taiwan

Correspondence should be addressed to Yi-Chung Hu; ychu@cycu.edu.tw

Received 5 May 2014; Accepted 12 June 2014; Published 29 June 2014

Academic Editor: Chih-Chou Chiu

On the basis of fuzzy regression, computational models in intelligence such as neural networks have the capability to be applied to nonlinear interval regression analysis for dealing with uncertain and imprecise data. When training data are not contaminated by outliers, computational models perform well by including almost all given training data in the data interval. Nevertheless, since training data are often corrupted by outliers, robust learning algorithms employed to resist outliers for interval regression analysis have been an interesting area of research. Several approaches involving computational intelligence are effective for resisting outliers, but the required parameters for these approaches are related to whether the collected data contain outliers or not. Since it seems difficult to prespecify the degree of contamination beforehand, this paper uses multilayer perceptron to construct the robust nonlinear interval regression model using the genetic algorithm. Outliers beyond or beneath the data interval will impose slight effect on the determination of data interval. Simulation results demonstrate that the proposed method performs well for contaminated datasets.

1. Introduction

In many practical applications, since the available information is often derived from uncertain assessments, real intervals can be employed to represent uncertain and imprecise observations [1]. Interval regression analysis, which provides interval estimation of individual dependent variables, is an important tool for dealing with uncertain data [1–3]. Interval regression analysis was developed on the basis of an important tool, namely, fuzzy regression analysis introduced by Tanaka et al. [4], whose objective is to build a model that contains all observed output data in terms of fuzzy numbers [4, 5].

Among computational models in intelligence, neural-network-based approaches have been employed to deal with nonlinear fuzzy or interval regression analysis (e.g., [6–10]) to facilitate the usefulness of fuzzy regression analysis. It is known that the neural-network-based approaches can overcome the difficulty in nonlinear fuzzy regression with LP, which is to choose a nonlinear model from an infinite number of alternatives [2]. When training data are not contaminated by outliers, which can be simply interpreted as data with a large deviation from its designated location [3, 11], these methods perform well and the estimated data interval includes almost all given training data in the data interval. Nevertheless, since training data are often corrupted by outliers, data interval obtained by these methods can be influenced by outliers. It is noted that statistical data preprocessing is helpful to detect outliers, but the robust interval regression analysis can not only resist outliers but also provide interval estimation of individual dependent variables.

To overcome the corruption arising from outliers, robust learning approaches involving computational intelligence (e.g., neural networks and support vector machines) have been employed to determine the upper and lower bounds of data interval. For instance, Huang et al. [3] employed two MLPs to determine nonlinear interval models using a new cost function, in which the cost function introduced in [6] and the robust BP algorithm for function approximation [12] were taken into account. Jeng et al. [2] employed two radial basis function networks to determine the upper and lower bounds after applying the support vector regression approach to determine the initial structure and parameter values of neural networks. Moreover, Hwang et al. [1] proposed

a robust method by combining the possibility estimation formulation integrating the property of central tendency with the principle of support vector interval regression.

It is found that, to determine the maximum and minimum limitations of interval regression, Jeng et al. [2] estimated the trimmed standard deviation of residuals excluding the highest and lowest α% of the number of training data. α can be specified as zero and 5–10 without and with outliers contained in the data, respectively. To avoid the estimated upper and lower bounds going beyond the maximum and minimum limitations, the values of a constant in the robust learning algorithms, introduced in [1, 2], are required to be carefully specified for both the uncontaminated and the contaminated data, to modify the desired output of each pattern. Moreover, in [3, 13], the upper limits on the percentage of outliers beyond and beneath the true interval model are prespecified for determining the degree of influence of each training pattern. Although the above-mentioned robust learning algorithms are robust against outliers, the required parameters for these algorithms are related to whether the collected data contain outliers or not. However, since the collected data are more or less contaminated, it is not easy to prespecify the degree of contamination. This motivates us to develop robust method without considering the degree of contamination.

On the basis of the effectiveness of using two MLPs to independently identify the upper and lower bounds of data interval for interval regression analysis, this paper aims to propose robust learning algorithms with the weighting schemes in [6] for MLP by using the genetic algorithm (GA) to identify outliers automatically for determining the robust nonlinear interval regression model, since the GA is a powerful search and optimization method [14, 15]. The proposed method has the feature that outliers beyond or beneath the data interval will impose slight effect on the determination of upper and lower bounds. In practice, only parameter specifications for the GA are required for the proposed learning approach. To sum up, the main contribution of this paper is to use MLP to construct a robust nonlinear interval model by making use of the GA.

The rest of this paper is organized as follows. The functional-link net with functional-expansion model for approximation is introduced in Section 2. Section 3 introduces the MLP-based approach proposed by Ishibuchi and Tanaka [6] for nonlinear interval regression analysis. Section 3 describes the proposed robust learning algorithms in detail. In Section 4, in order to examine the effectiveness and applicability of the proposed method for determining a nonlinear interval regression model, several examples and real data are taken into account. From the experimental results, it is seen that the nonlinear interval models obtained by the proposed learning algorithms can include almost all regular data. This paper is concluded in Section 5.

2. Multilayer Perceptron for Nonlinear Interval Regression Analysis

Since the weighting schemes proposed by Ishibuchi and Tanaka [6] are incorporated into the proposed robust learning algorithms, the MLP-based approach employed to determine the estimated data interval is described in this section. MLP and the MLP-based approach are briefly reviewed in Sections 2.1 and 2.2, respectively.

2.1. Multilayer Perceptron. MLP is usually used as a tool of approximation of functions like regression [16]. A three-layer perceptron with n input nodes and a single hidden layer is taken into account. Let us denote the given nonfuzzy input-output pairs by $(\mathbf{x}_p, y_p)$, $p = 1, 2, \ldots, m$, where $\mathbf{x}_p = (x_{p1}, x_{p2}, \ldots, x_{pn})$ and y_p are the input vector and the corresponding desired output value, respectively. The sigmoid function, whose output ranges from 0 to 1, is commonly used as the transfer function for each hidden node and output node.

When $\mathbf{x}_p$ is presented to MLP, the output value from the jth hidden node is computed as

$$o_{pj} = f_s\left(\sum_{i=1}^{n} w_{ij} x_{pi} + \theta_j\right), \tag{1}$$

where f_s represents the sigmoid function, θ_j is the bias to the jth hidden node, and w_{ij} is the connection weight from the ith input node to the jth hidden node. Then, the final output value from the output node is computed as

$$o_p = f_s\left(\sum_{j=1}^{k} w_j o_{pj} + \theta\right), \tag{2}$$

where k is the number of hidden nodes, θ is the bias to the output node, and w_j is the connection weight from the jth hidden node to the output node. Thus, there are $(n + 2)k + 1$ synaptic connections. The following cost function can be employed to train MLP:

$$E = \frac{1}{2}\sum_{i=1}^{m}\left(y_p - o_p\right)^2, \tag{3}$$

where m represents the number of training patterns. It should be noted that since it was shown that MLP with a single hidden layer and any fixed continuous sigmoid function is sufficient to approximate any continuous function [17], a three-layer model is taken into account for our study.

2.2. MLP-Based Approach. Ishibuchi and Tanaka [6] employed two feed-forward MLPs, MLP^* and MLP_*, to determine effectively a nonlinear interval model. Each of the two networks has only one hidden layer. Let $g^*(\mathbf{x})$ and $g_*(\mathbf{x})$ denote the output functions realized by MLP^* and MLP_*, respectively. In practice, $g^*(\mathbf{x})$ and $g_*(\mathbf{x})$ represent the upper and lower bounds of a nonlinear interval model, respectively.

A nonlinear optimization problem is formulated to determine the nonlinear interval regression model as follows:

$$\begin{aligned} \text{Minimize} \quad & \left(g^*(\mathbf{x}_1) - g_*(\mathbf{x}_1)\right) + \left(g^*(\mathbf{x}_2) - g_*(\mathbf{x}_2)\right) \\ & + \cdots + \left(g^*(\mathbf{x}_m) - g_*(\mathbf{x}_m)\right) \\ \text{subject to} \quad & g_*(\mathbf{x}_p) \le y_p \le g^*(\mathbf{x}_p), \quad p = 1, 2, \ldots, m, \end{aligned} \tag{4}$$

where $(g^*(\mathbf{x}_p) - g_*(\mathbf{x}_p))$ represents the width of the estimated data interval for $\mathbf{x}_p$. The objective of the above formulation is to determine the nonlinear interval model with the least sum of widths of the predicted intervals for the respective inputs subject to the estimated data interval determined by the two MLPs including all the given input-output pairs.

Instead of deriving a learning algorithm directly from the above nonlinear optimization problem, Ishibuchi and Tanaka derived learning algorithms for $g^*(\mathbf{x})$ and $g_*(\mathbf{x})$ by the following cost function with weighting scheme ω_p:

$$E = \sum_{p=1}^{m} \frac{1}{2} \omega_p \left(y_p - g^* \left(\mathbf{x}_p \right) \right)^2, \tag{5}$$

where ω_p is a small positive value in the interval (0, 1). To determine the upper bound $g^*(\mathbf{x})$, ω_p is defined as

$$\omega_p = \begin{cases} 1, & \text{if } y_p > g^* \left(\mathbf{x}_p \right), \\ \omega, & \text{if } y_p \le g^* \left(\mathbf{x}_p \right). \end{cases} \tag{6}$$

As for determining the lower bound $g_*(\mathbf{x})$, ω_p is defined as

$$\omega_p = \begin{cases} 1, & \text{if } y_p < g_* \left(\mathbf{x}_p \right), \\ \omega, & \text{if } y_p \ge g_* \left(\mathbf{x}_p \right). \end{cases} \tag{7}$$

It can be seen that the cost function can be multiplied by a small positive value in the interval (0, 1) (i.e., ω_p) depending on whether the desired output of an input pattern is less than or greater than the network-estimated value. Weights updating rules in the MLP-based approach can be derived by gradient descent from the above objective function with weighting scheme. It is obvious that the two learning algorithms for determining $g^*(\mathbf{x})$ and $g_*(\mathbf{x})$ are the same except for the weighting schemes. Ishibuchi and Tanaka suggested that a smaller value of $\omega_p^{(r)}$ could lead to better satisfaction of the constraint condition.

Nevertheless, the nonlinear interval model derived by the MLP-based approach can include all training data with outliers [3]. That is, the nonlinear interval model obtained by the MLP-based approach is sensitive to contaminated training data. To overcome the above problem, Huang et al. [3] pointed out that we should only pay attention to the quality of the training data whose desired outputs are greater than the actual outputs for determining $g^*(\mathbf{x})$ and that of the training data whose desired outputs are less than the actual output for determining $g_*(\mathbf{x})$. In practice, those more suspected outliers beyond or beneath the estimated data interval impose slighter effect on the determination of data interval.

3. The Proposed Robust Learning Algorithms

According to the objective function with the weighting schemes introduced in [6], Section 3.1 demonstrates the formulation of optimization problems. Moreover, the fitness functions of the GA are described. Subsequently, the coding scheme and genetic operations are described in detail in Sections 3.2 and 3.3, respectively. Finally, the proposed robust learning algorithms for determining the nonlinear interval model are presented.

3.1. Problem Formulation. Two neural networks, MLP* and MLP$_*$, are still employed to determine the upper and lower bounds of data interval. To determine the upper bound of the nonlinear interval regression model (i.e., $g^*(\mathbf{x})$), the single-objective optimization problem of constructing MLP* is formulated as

$$\begin{aligned} &\text{Minimize} \quad E\,(\text{MLP}) \\ &\text{subject to} \quad y_p \le g^* \left(\mathbf{x}_p \right), \quad p = 1, 2, \ldots, m. \end{aligned} \tag{8}$$

In a similar manner, the single-objective optimization problem of constructing MLP$_*$ for the lower bound (i.e., $g_*(\mathbf{x})$) is formulated as

$$\begin{aligned} &\text{Minimize} \quad E\,(\text{MLP}) \\ &\text{subject to} \quad g_* \left(\mathbf{x}_p \right) \le y_p, \quad p = 1, 2, \ldots, m, \end{aligned} \tag{9}$$

where E(MLP) is defined as

$$E\,(\text{MLP}) = \sum_{p=1}^{m} \frac{1}{2} \omega_p \Psi_p \left(y_p - f^* \left(\mathbf{x}_p \right) \right)^2. \tag{10}$$

It can be seen that Ψ_p is incorporated into the above-mentioned cost function E. In the determination of $g^*(\mathbf{x})$ at the tth generation during evolution, Ψ_p for $\mathbf{x}_p$ is set to be of a very small positive value, say Ψ, if $\mathbf{x}_p$ beyond $g^*(\mathbf{x})$ is an outlier. On the contrary, Ψ_p is set to be 1. In a similar manner for determining $g_*(\mathbf{x})$, Ψ_p for $\mathbf{x}_p$ is set to be of a very small positive value if $\mathbf{x}_p$ beneath $g^*(\mathbf{x})$ is an outlier. Otherwise, Ψ_p is set to be 1. In other words, outliers beyond or beneath the data interval can impose a slight effect on the determination of data interval.

The single-objective GA, which is a general-purpose optimization technique, can be applied to the above problems by introducing a reward to the fitness function when the constraint condition is satisfied:

$$\text{fitness}\,(\text{MLP}) = w_e \cdot \frac{1}{1 + E\,(\text{MLP})} + w_{\text{num}} \cdot R\,(\text{MLP}), \tag{11}$$

where w_e and w_{num} are constant positive weights of the objective and reward, respectively, and R(MLP) is the ratio of input-output pairs that satisfy the constraint condition:

$$R\,(\text{MLP}) = \frac{1}{m} \sum_{p=1}^{m} r_p, \tag{12}$$

where r_p is defined for determining $g^*(\mathbf{x})$ as

$$r_p = \begin{cases} 0, & \text{if } y_p > g^* \left(\mathbf{x}_p \right), \\ 1, & \text{if } y_p \le g^* \left(\mathbf{x}_p \right), \end{cases} \tag{13}$$

whereas for determining $g_*(\mathbf{x})$, r_p is defined as

$$r_p = \begin{cases} 1, & \text{if } y_p > g_* \left(\mathbf{x}_p \right), \\ 0, & \text{if } y_p \le g_* \left(\mathbf{x}_p \right). \end{cases} \tag{14}$$

The reward R(MLP) is incorporated into the fitness function (i.e., fitness(MLP)) in order to facilitate all regular data beyond or beneath the estimated data interval.

3.2. Coding. As in [13], the transformation of the network output with respect to the pth pattern, say o_p, to the domain interval of the desired output is performed as follows:

$$o_p = o_p\,(ma - mi) + mi, \quad p = 1, 2, \ldots, m, \tag{15}$$

where ma and mi equal $(\max\{y_p \mid p = 1, 2, \ldots, m\} + u)$ and $(\min\{y_p \mid p = 1, 2, \ldots, m\} - l)$, respectively. Those unknown parameters including u, l, $(n+2)k+1$ connection weights and bias of a MLP are automatically determined by the GA. Thus, there are $(n+2)k+3$ substrings in a binary string. To identify outliers, a substring consisting of m bits is employed to determine the set of outliers, and each bit can indicate whether the corresponding training pattern is an outlier or not. In practice, the bit of 0 indicates that the corresponding training pattern is not an outlier; otherwise, that pattern is an outlier. In an initial population, each bit in the string is randomly assigned as either 1 or 0, with the probability of 0.5. A substring except for determining outliers and u, l can be directly decoded as a real value ranging from -5 to 5. Such a range should be acceptable when the sigmoid function is used as the activation function, since it is known that a node tends to suffer from the premature saturation resulting from infinitely large positive or negative weights.

The determination of the length of a string depends mainly on their domain lengths and the corresponding required precision [18]. That is, if the domain of a variable has the length of τ_1 where $2^{\tau_3-1} < \tau_1 10^{\tau_2} < 2^{\tau_3}$, and the corresponding required precision is τ_2 decimal places, then τ_3 bits are required to code such a variable.

3.3. Genetic Operations. Let N_{pop} denote the population size. When the fitness of each chromosome in the current population is obtained, genetic operators including selection, crossover, and mutation [14, 15, 19] are employed to determine the newly generated N_{pop} strings in the next population. Using the binary tournament selection with replacement, two strings are randomly selected from the current population, and the one with the maximum fitness can be placed in the mating pool. This process can be repeated N_{pop} times until there are N_{pop} strings in the mating pool. In other words, $0.5N_{\text{pop}}$ pairs of chromosomes can be randomly selected from the current population.

After tournament selection, crossover and mutation are applied to a selected parent to reproduce children by altering the chromosomal makeup of two parents. In practice, the one-point crossover operation with the crossover probability Pr_c is used for exchanging partial information between two substrings in the selected pair of strings, and two new strings are generated to replace their parent strings. Each crossover point in a substring is chosen randomly. That is, we use a substring-wise one-point crossover operation where the total number of crossover points is the same as the number of substrings in each string. The mutation operation with the mutation probability Pr_m is performed on each bit of strings generated by the crossover operation.

3.4. Learning Algorithm Implementation. The MLP robust learning algorithm for determining the upper bound of the nonlinear interval regression model (i.e., $g^*(\mathbf{x})$) is the same as that for determining the lower bound (i.e., $g_*(\mathbf{x})$), except for the weighting scheme, R(MLP), and Ψ_p in the fitness function. The learning of each of the two MLPs is independent of each other. That is, two MLPs are independently employed to determine the upper and lower bounds of data interval. The learning algorithm for determining $g^*(\mathbf{x})$ is written in the following.

Algorithm 1 (a robust MLP learning algorithm for determining the upper bound of a nonlinear interval model).

Input

(a) Population size: N_{pop};
(b) Total number of generations: N_{con};
(c) Number of elite chromosomes: N_{del};
(d) Crossover rate: Pr_c;
(e) Mutation rate: Pr_m;
(f) Relative weights in the fitness function: w_e and w_{num};
(g) A set of training patterns.

Output. The upper bound of a nonlinear interval model.

Method

Step 1: Initialization. A population containing N_{pop} binary strings is generated randomly.

Step 2: Transform Network-Estimated Values. For each pattern, transform the output of MLP into a value ranging from $(\max\{y_p \mid p = 1, 2, \ldots, m\}+u)$ to $(\min\{y_p \mid p = 1, 2, \ldots, m\}-l)$ by (15).

Step 3: Compute Fitness Values. Compute the fitness value of each string in the current population by (11).

Step 4: Termination Test. N_{con} is used as the stopping condition. If the stopping condition is not satisfied, then proceed to the next step. That is, the genetic operations are iterated again to generate the new strings in the next population.

Step 5: Generate New Strings. Genetic operators are employed to generate the N_{pop} new strings in the next population from the current population.

Step 6: Perform Elitist Strategy. N_{del} strings are randomly removed from the newly generated N_{pop} strings. Then, add N_{del} best strings in the current population to form the next one.

The best string among the successive generations is taken as the desired solution.

4. Computer Simulations

Several data sets are employed to examine the effectiveness of the proposed method for determining a nonlinear interval

regression model. Each data set is involved in learning a function of one variable (i.e., $n = 1$). Since there is no best set of GA parameter specifications, according to the principles introduced in [18], the prespecified parameter specifications of the proposed robust learning algorithms for each simulation are described as follows.

(1) $N_{\text{pop}} = 50$: the most common population size varies from 50 to 500 individuals. Hence, 50 individuals is an acceptable minimum size. In an initial population, each bit in a binary string is randomly assigned as either 1 or 0, with the probability of 0.5.

(2) $N_{\text{con}} = 5000$: the stopping condition is specified according to the available computation time. However, a sufficient evolution of the GA is necessary.

(3) $N_{\text{del}} = 2$: to avoid generating too much perturbation in the next generation, a small number of elite chromosomes are taken into account.

(4) Substring length: (i) since a connection weight ranges from −5 to 5 (i.e., $\tau_1 = 10$), and the required precision is three decimal places (i.e., $\tau_2 = 3$), 14 bits (i.e., $\tau_3 = 14$) are required to code each connection weight. (ii) Both u and l are set to be small real values in $[0, 3]$ (i.e., $\tau_1 = 5$), and the required precision is also three decimal places. Therefore, 13 bits are required to code both u and l.

(5) $\text{Pr}_c = 0.95$ and $\text{Pr}_m = 0.001$: since a Pr_c with a larger value allows the exploration of more of the solution space, a larger Pr_c is usually taken into account. Furthermore, in order not to generate excessive perturbation, Pr_m should be specified as a lower value.

(6) $w_e = 2$, $w_{\text{num}} = 1$: it is considered that the minimization of the cost function is the primary objective of the regression analysis; a larger value is thus set to be w_e.

(7) $\omega = 0.2$, $\Psi = 10^{-5}$: as mentioned above, ω is suggested to be specified as a small value. Ψ is set to be a very small positive value approaching zero.

Besides, as in [6], a MLP with five hidden nodes (i.e., $k = 5$) is taken into account. Therefore, there are 18 substrings in a binary string.

First, uncontaminated data are employed to verify the effectiveness of the proposed learning algorithms. The uncontaminated training data of the first example generated by

$$\begin{aligned} x_p &= 0.02\,(p-1), \quad p = 1, 2, \ldots, 51, \\ y_p &= 0.2\sin\left(2\pi x_p\right) + 0.2x_p^2 + 0.3 \\ &\quad + \left(0.1x_p^2 + 0.05\right)\text{rnd}\,[-1, 1] \end{aligned} \tag{16}$$

are taken into account, where rnd[−1, 1] denotes a real number generated in the interval [−1, 1] at random. The simulation result obtained using the proposed learning algorithms is depicted in Figure 1. As can be seen, 51 training data are approximately included in the data interval. For

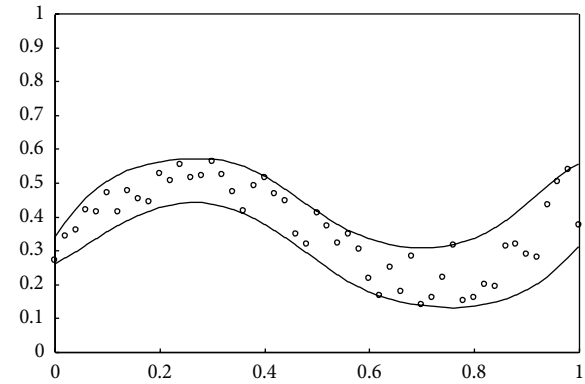

Figure 1: Simulation result for the first example without outliers.

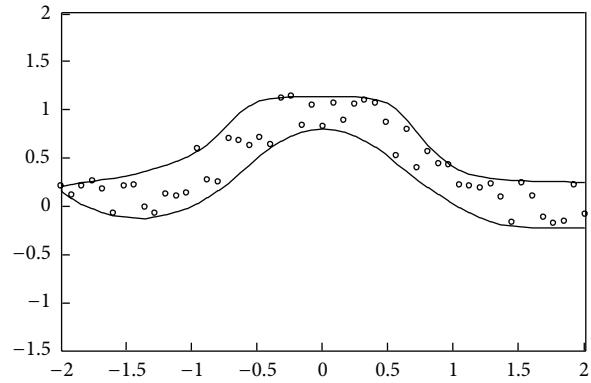

Figure 2: Simulation result for the second example without outliers.

the second example, a simple function for generating the uncontaminated training data is defined as

$$\begin{aligned} x_p &= 0.08\,(p-1) - 2, \quad p = 1, 2, \ldots, 51, \\ y_p &= \exp\left(-x_p^2\right) + 0.5\text{rnd}\,[-1, 1]\,. \end{aligned} \tag{17}$$

The simulation result obtained using the proposed method for the above uncontaminated data is depicted in Figure 2. Moreover, as can be seen, the data interval includes almost all given training data. Subsequently, the simulation results for the real data set of studies on National Institute of Standards and Technology (NIST), which involve ultrasonic calibration consisting of 54 observations [20] and quantum defects in iodine atoms consisting of 25 observations [21], are further shown in Figures 3 and 4, respectively. The ultrasonic calibration data, whose response variable is ultrasonic response and whose predictor variable is metal distance, are often used to illustrate the construction of a nonlinear regression model. The quantum defects in data of iodine atoms, whose response variable is the number of quantum defects and whose predictor variable is the excited energy state, can also be employed to construct a nonlinear least squares regression model. It can be seen that the proposed learning algorithms work well for these two real data sets.

Contaminated data are further employed to examine the data intervals obtained by the proposed learning algorithms.

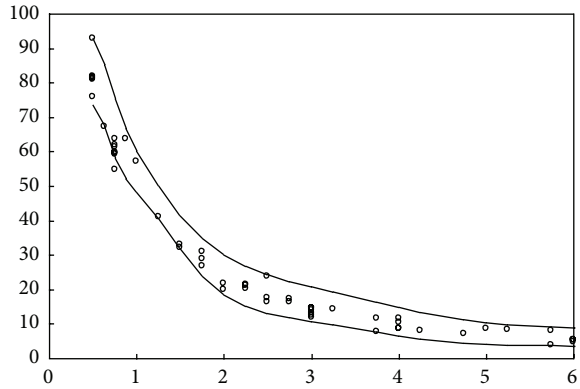

Figure 3: Simulation result for the ultrasonic calibration data.

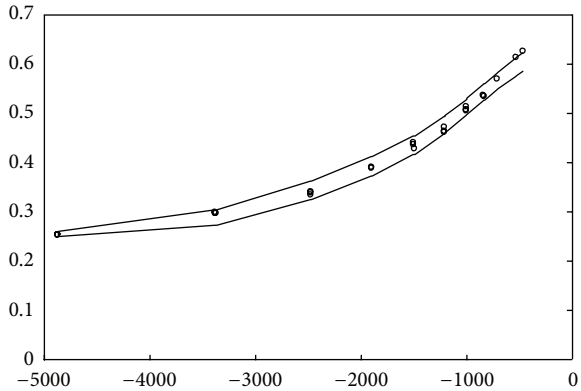

Figure 4: Simulation result for the data of quantum defects in iodine atoms.

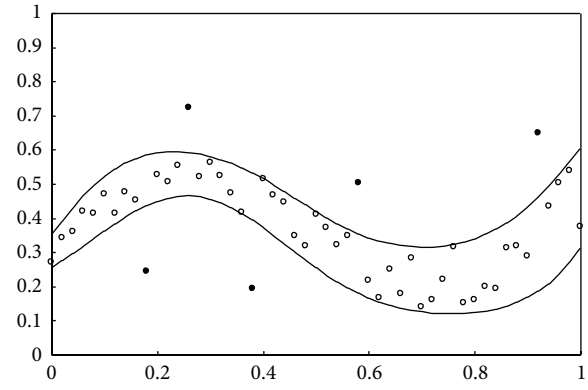

Figure 5: Simulation result for the first example with outliers.

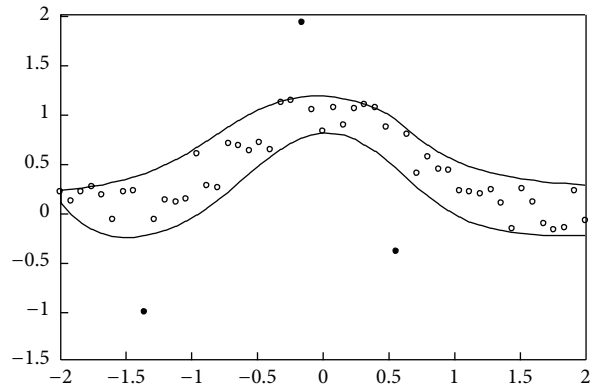

Figure 6: Simulation result for the second example with outliers.

For the first example, 5 out of the same 51 input-output pairs (i.e., 46 pieces of regular data) are randomly selected as outliers. The simulation result shown in Figure 5 indicates that the proposed learning algorithms can resist outliers. Those regular data are approximately included in the robust nonlinear interval model. Furthermore, comparison with Figure 1 shows that both results are approximately identical. For the second example, 3 out of the same 51 input-output pairs are randomly selected as outliers. Figure 6 shows that the estimated upper and lower bounds obtained by the proposed learning algorithms are not influenced by outliers. It can be seen that the simulation results shown in Figures 2 and 6 are similar. Using the traditional MLP-based approach, the results shown in [3] are not depicted in Figures 5 and 6 to simplify the presentation. From [3], it can be seen that the nonlinear interval model obtained by the MLP-based approach is sensitive to contaminated training data.

5. Conclusions

As mentioned above, since the available information is often derived from uncertain assessments, it is reasonable to use real intervals to deal with imprecise observations. Except for the traditional regression function determined merely by minimizing the least squared error, computational models in intelligence have been employed to determine the nonlinear interval regression model. There is no doubt that since the available data often contain outliers, the development of robust algorithms is necessary. Since the collected data are more or less contaminated, it is not easy to estimate the degree of contamination without performing statistical data preprocessing. In comparison with computational models in intelligence presented in [1–3, 13], the proposed robust learning algorithms have the advantage of avoiding considering the degree of contamination of the collected data.

This paper proposes MLP learning algorithms with the weighting schemes in [6] for determining the robust nonlinear interval regression model. Outliers, which are identified by the GA, beyond or beneath the data interval will impose a slight effect on the determination of data interval. As seen from the experimental results, it is seen that the nonlinear interval models obtained by the proposed learning algorithms can include almost all regular data. That is, the proposed learning algorithms are robust against outliers for contaminated data. Thus, it seems that the incorporation of the ratio of training data that are included in the interval model into the fitness function can facilitate the inclusion of regular data in the robust nonlinear interval model.

The nonlinear interval models shown in the previous section are satisfactory for both uncontaminated and contaminated data, whereas customized parameter tuning in

computer simulations is not particularly considered for the proposed learning algorithms. The same parameter specifications in the GA are applied to each experiment. For this, it seems that the proposed learning algorithms are not sensitive to GA parameter specifications. The experimental results show that common setting of the GA parameter specifications for the proposed approach is acceptable.

Previously, several literatures with respect to the robust interval regression model have been published. For instance, Fagundes et al. [22] dealt with cases that have interval-valued outliers in the input data set; Chuang and Lee [23] used data preprocessing to filter out outliers in the training data and then a regression model could be constructed by using the filtered data to train the support vector regression networks; D'Urso et al. [24] proposed a robust fuzzy linear regression model based on the least median squares-weighted least squares estimation procedure for the highly skewed data; Huang [25] proposed a reduced support vector machine approach in evaluating interval regression models with non-fuzzy inputs and interval output, but the models seem not to pay more attention to outlier resistance. In comparison with these methods, the proposed approach uses MLP to construct the robust interval regression models with crisp inputs and crisp outputs by using the GA elaborately. The use of data preprocessing to detect outliers and the consideration of interval-valued data set remain to be studied in future work.

Conflict of Interests

The author declares that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The author would like to thank the anonymous referees for their valuable comments. This research is partially supported by the National Science Council of Taiwan under Grant NSC 102-2410-H-033-039-MY2.

References

[1] C. Hwang, D. H. Hong, and K. H. Seok, "Support vector interval regression machine for crisp input and output data," *Fuzzy Sets and Systems*, vol. 157, no. 8, pp. 1114–1125, 2006.

[2] J. T. Jeng, C. C. Chuang, and S. F. Su, "Support vector interval regression networks for interval regression analysis," *Fuzzy Sets and Systems*, vol. 138, no. 2, pp. 283–300, 2003.

[3] L. Huang, B. Zhang, and Q. Huang, "Robust interval regression analysis using neural networks," *Fuzzy Sets and Systems*, vol. 97, no. 3, pp. 337–347, 1998.

[4] H. Tanaka, S. Uejima, and K. Asai, "Linear regression analysis with fuzzy model," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 12, no. 6, pp. 903–907, 1982.

[5] J. Watada and Y. Yabuuchi, "Fuzzy robust regression analysis," in *Proceedings of the 3rd IEEE Conference on Fuzzy Systems*, pp. 1370–1376, Beijing, China, June 1994.

[6] H. Ishibuchi and H. Tanaka, "Fuzzy regression analysis using neural networks," *Fuzzy Sets and Systems*, vol. 50, no. 3, pp. 257–265, 1992.

[7] H. Ishibuchi, H. Tanaka, and H. Okada, "An architecture of neural networks with interval weights and its application to fuzzy regression analysis," *Fuzzy Sets and Systems*, vol. 57, no. 1, pp. 27–39, 1993.

[8] H. Ishibuchi and M. Nii, "Fuzzy regression using asymmetric fuzzy coefficients and fuzzified neural networks," *Fuzzy Sets and Systems*, vol. 119, no. 2, pp. 273–290, 2001.

[9] C. B. Cheng and E. S. Lee, "Fuzzy regression with radial basis function network," *Fuzzy Sets and Systems*, vol. 119, no. 2, pp. 291–301, 2001.

[10] K. Kwon, H. Ishibuchi, and H. Tanaka, "Neural networks with interval weights for nonlinear mappings of interval vectors," *IEICE Transactions on Information and Systems*, vol. E77-D, no. 4, pp. 409–417, 1994.

[11] H. Ishibuchi and H. Tanaka, "Several formulations of interval regression analysis," in *Proceedings of the Sino-Japan Joint Meeting on Fuzzy Sets and Systems*, Beijing, China, 1990.

[12] D. S. Chen and R. C. Jain, "Robust back propagation learning algorithm for function approximation," *IEEE Transactions on Neural Networks*, vol. 5, no. 3, pp. 467–479, 1994.

[13] Y. C. Hu, "Functional-link nets with genetic-algorithm-based learning for robust nonlinear interval regression analysis," *Neurocomputing*, vol. 72, no. 7–9, pp. 1808–1816, 2009.

[14] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, Pa, USA, 1989.

[15] J. S. R. Jang, C. T. Sun, and E. Mizutani, *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*, Prentice Hall, Upper Saddle River, NJ, USA, 1997.

[16] K. A. Smith and J. N. D. Gupta, "Neural networks in business: techniques and applications for the operations researcher," *Computers and Operations Research*, vol. 27, no. 11-12, pp. 1023–1044, 2000.

[17] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals, and Systems*, vol. 2, no. 4, pp. 303–314, 1989.

[18] A. Osyczka, *Evolutionary Algorithms for Single and Multicriteria Design Optimization*, Physica, New York, NY, USA, 2002.

[19] K. F. Man, K. S. Tang, and S. Kwong, *Genetic Algorithms: Concepts and Designs*, Springer, London, UK, 1999.

[20] D. Chwirut, *Ultrasonic Reference Block Study Data*, National Institute of Standards and Technology (NIST), US Department of Commerce, Gaithersburg, Md, USA, 1979.

[21] L. Roszman, *Quantum defects for sulfur I atom*, National Institute of Standards and Technology (NIST), US Department of Commerce, 1979, http://www.itl.nist.gov/div898/strd/nls/data/roszman1.shtml.

[22] R. A. A. Fagundes, R. M. C. R. de Souza, and F. J. A. Cysneiros, "Robust regression with application to symbolic interval data," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 1, pp. 564–573, 2013.

[23] C. Chuang and Z. Lee, "Hybrid robust support vector machines for regression with outliers," *Applied Soft Computing Journal*, vol. 11, no. 1, pp. 64–72, 2011.

[24] P. D'Urso, R. Massari, and A. Santoro, "Robust fuzzy regression analysis," *Information Sciences*, vol. 181, no. 19, pp. 4154–4174, 2011.

[25] C. Huang, "A reduced support vector machine approach for interval regression analysis," *Information Sciences*, vol. 217, pp. 56–64, 2012.