A Dual Neural Network Based On Confidence Intervals For Fuzzy Random Regression Problems

Hang Yu
Centre for Artificial Intelligence
Faculty of Engineering and Information Technology
University of Technology Sydney
Sydney, Australia
Hang.Yu@student.uts.edu.au

Jie Lu
Centre for Artificial Intelligence
Faculty of Engineering and Information Technology
University of Technology Sydney
Sydney, Australia
Jie.Lu@uts.edu.au

Guangquan Zhang
Centre for Artificial Intelligence
Faculty of Engineering and Information Technology
University of Technology Sydney
Sydney, Australia
Guangquan.Zhang@uts.edu.au

Dianshuang Wu
Centre for Artificial Intelligence
Faculty of Engineering and Information Technology
University of Technology Sydney
Sydney, Australia
Dianshuang.Wu@uts.edu.au

***Abstract*—Uncertainty in dependent variables or independent variables is typically caused by randomness or fuzziness. But randomness and fuzziness are more and more often appearing simultaneously in independent variables or dependent variables, giving rise to the concept of a fuzzy random variable. Regression analysis is a statistical measure to model the relationship between a dependent variable and one or more independent variables. However, the standard regression algorithms cannot handle the fuzzy random variables, so we propose a dual neural network algorithm based on confidence intervals for fuzzy random regression problems in this paper. The algorithm relies on the expectations of, and variances in, fuzzy random variables to construct the confidence intervals for fuzzy random input-output data. A dual neural network then identifies the sides of the interval output data; one network identifies the upper side, another network identifies the lower side, while a dual v-support vector regression algorithm concurrently constructs the initial structure of the dual neural network. Lastly, a dynamic genetic backpropagation algorithm tunes the parameters of the dual neural network to improve performance. Experiment results demonstrate the validity and applicability of the proposed dual neural network algorithm based on confidence intervals.**

Keywords*— *fuzzy random variable, regression analysis, neural networks, support vector regression, genetic backpropagation

I. Introduction

Regression analysis is a statistical measure used in finance, investing and other disciplines that attempts to model the relationship between a dependent variable and one or more independent variables [1]. Algorithms that can facilitate regression analysis are known as regression algorithms. Many standard regression algorithms request the available information of variables is precise [2], but in many real-world applications, the available information of variables is often uncertain. Hence to model the uncertain variables in regression algorithm, random variables and fuzzy variables were proposed from different theories [3]. In statistics, a random variable is a variable whose possible values are numerical outcomes of a random phenomenon [4]. A fuzzy variable is a generalization of a real number in the sense that it does not refer to one single value but rather to a connected set of possible values, where each possible value has its own weight between 0 and 1 [5].

However, randomness and fuzziness can appear simultaneously in a dependent or independent variables. For example, consider a set of expert evaluations for an product. Assume there are 100 samples of the agricultural product and five experts evaluate the samples on the basis of ten attributes. Each expert grades each sample according to his experience and expertise. When different experts give different grades, the differences among the grades can be treated as a random variable, and each individual grade also can be treated as a fuzzy variable, such as a grade is given as about 5. Thus, the grades have both randomness and fuzziness information at the same time. The concept of fuzzy random variables (FRV) [6] was proposed to deal with such variations in grade types, and the example considered here is a fuzzy random regression problem [7]. Many researchers have since studied the mathematical properties of a fuzzy random variable [8-9], but few works have laid out a blueprint for designing an algorithm that solves fuzzy random regression problems [10].

To fill this gap in the literature, this paper presents a dual neutral network algorithm based on confidence intervals (CI-DNN) is proposed for fuzzy random regression problems. First, the confidence intervals in fuzzy random input-output data are constructed based on the expectations of and variances in fuzzy random variables. Then, the upper and lower sides of the interval output data are identified using a dual neural network. One identifies the upper side; the other identifies the lower side. A dual *v*-support vector regression algorithm concurrently maps out the initial structure of the dual neural network, which a dynamic genetic backpropagation algorithm uses to tune the parameters of the dual neural network to improve performance.

The contributions of this paper are twofold: (1) a strategy to better deal with the fuzzy random variables in regression model; (2) a new regression algorithm which include a dual neural network, and use a novel support vector regression algorithm to construct the initial structure of the dual neural network, and a novel backpropagation algorithm to tune the initial values.

The rest of this paper is organized as follows. Section II sets out the preliminaries of fuzzy random variables. Section III discusses the use of CI-DNN algorithm to solve fuzzy random regression problems. The experimental settings and results are presented in Section IV, along with a discussion of the findings. Section V provides the concluding remarks.

II. PRELIMINARIES

A. *FRV: Expected Value and Variance*

Given a universe Γ, let *Pos* be a possibility measure that is defined on the power set $P(\Gamma)$ of Γ. Let $\Re$ be the set of real numbers. A function $Y:\Gamma \rightarrow \Re$ is said to be a fuzzy variable defined by $\mu_Y(t) = Pos\{Y = t\}$. $t \in \Re$ is the possibility of event $\{Y = t\}$. The possibility *Pos*, necessity *Nec*, and credibility *Cr* of event $\{Y \le r\}$ for the fuzzy variable *Y*, with possibility distribution μ_Y are:

$$Pos\{Y \le r\} = \sup_{t \le r} \mu_Y(t)$$
$$Nec\{Y \le r\} = 1 - \sup_{t > r} \mu_Y(t)$$
$$Cr\{Y \le r\} = \frac{1}{2}\left(1 + \sup_{t \le r} \mu_Y(t) - \sup_{t > r} \mu_Y(t)\right) \tag{1}$$

The expected value of a fuzzy variable based on the above credibility measure follows in *Definition* 1.

Definition 1 (see [11]): Let *Y* be a fuzzy variable. The expected value of *Y* is defined as

$$E[Y] = \int_0^\infty Cr\{Y \ge r\}dr - \int_{-\infty}^0 Cr\{Y \le r\}dr \tag{2}$$

provided that the two integrals are finite.

Definition 2 (see[12]): Suppose that $(\Omega, \Sigma, \Pr)$ is a probability space and F_v is a collection of fuzzy variables defined on possibility space $(\Gamma, P(\Gamma), Pos)$. A fuzzy random variable is a mapping $X:\Omega \rightarrow F_v$ such that for any Borel subset *B* of $\Re$, $Pos\{X(\omega) \in B\}$ is a measurable function of ω.

Definition 3 (see [12]): Let *X* be a fuzzy random variable defined on a probability space $(\Omega, \Sigma, \Pr)$. Then, the expected value of *X* is defined as

$$E[\xi] = \int_\Omega \left[\int_0^\infty Cr\{\xi(\omega) \ge r\}dr - \int_{-\infty}^0 Cr\{\xi(\omega) \le r\}dr\right]\Pr(d\omega) \tag{3}$$

Definition 4 (see [12): Let *X* be a fuzzy random variable defined on a probability space $(\Omega, \Sigma, \Pr)$ with the expected value *e*. Then, the variance of *X* is defined as

$$Var[X] = E\left[(X - e)^2\right] \tag{4}$$

where *e* = *E*[X] is given by Definition 3.

B. *FRV With Confidence Intervals*

The format of the data is shown in Table I, where the input data X_{ik} and the output data Y_i for all *i* = 1, . . . , N, and *k* = 1, . . . , *N*, are fuzzy random variables, which are defined as [13]:

$$Y_i = \bigcup_{t=1}^{M_{Y_i}} \left\{\left(Y_i^t, Y_i^{t,l}, Y_i^{t,r}\right)_T, p_i^t\right\} \tag{5}$$

$$X_{ik} = \bigcup_{t=1}^{M_{X_{ik}}} \left\{\left(X_{ik}^t, X_{ik}^{t,l}, X_{ik}^{t,r}\right)_T, q_{ik}^t\right\} \tag{6}$$

respectively. This means that all values are given as fuzzy numbers with probabilities, where fuzzy variables $(Y_i^t, Y_i^{t,l}, Y_i^{t,r})_T$ and $(X_{ik}^t, X_{ik}^{t,l}, X_{ik}^{t,r})_T$ are associated with probability p_i^t and q_{ik}^t, for i = 1,2, . . . , *N*, k = 1,2, . . . , *N*, and t = 1,2, . . . , M_{Y_i} or t = 1,2, . . . , $M_{X_{ik}}$, respectively.

A confidence interval is defined based on the expectation of and variance in a fuzzy random variable, and it for each fuzzy random variable is expressed as :

$$I[e_X, \sigma_X] \triangleq \left[E(X) - \sqrt{Var(X)}, E(X) + \sqrt{Var(X)}\right] \tag{7}$$

given a one-sigma confidence interval (1 × σ), and hence called a one-sigma confidence interval [13]. Two-sigma and three-sigma confidence intervals can be similarly defined. All of these confidence intervals are called σ-confidence intervals. Table II shows the data for one-sigma confidence intervals.

III. AN CONFIDENCE-INTERVAL-BASED DUAL V-SUPPORT VECTOR REGRESSION NETWORKS

Based on the above σ-confidence intervals, both the input and output data translate into interval data. Thus, a fuzzy random regression problem is translated into an interval regression problem [14-15], i.e., where the interval input and output data are used to train a regression model. For solving an interval regression problem, the CI-DNN algorithm proposed in this paper.

In the CI-DNN algorithm, a dual neural network are employed to identify the sides of the interval output data; one neural network identifies the upper side, other neural network identifies the lower side. The initial structures of the dual neural network are obtained by a dual *v*-support vector regression algorithm. Then a dynamic genetic backpropagation algorithm is employed to tune the parameters of the dual neutral network under the initial structure. The detailed description of the dual neural network algorithm based on confidence intervals is presented in the following subsections.

TABLE I. FUZZY RANDOM INPUT-OUTPUT DATA

	Output	Inputs
i	Y	X_1, X_2, … X_t, …, X_N
1	$\{(Y_1^t, Y_1^{t,l}, Y_1^{t,r})_T, p_1^t\}$	$\{(X_{11}^t, X_{11}^{t,l}, X_{11}^{t,r}), q_{11}^t\}, \ldots, \{(X_{1N}^t, X_{1N}^{t,l}, X_{1N}^{t,r}), q_{1N}^t\}$
2	$\{(Y_2^t, Y_2^{t,l}, Y_2^{t,r})_T, p_2^t\}$	$\{(X_{21}^t, X_{21}^{t,l}, X_{21}^{t,r}), q_{21}^t\}, \ldots, \{(X_{2N}^t, X_{2N}^{t,l}, X_{2N}^{t,r}), q_{2N}^t\}$
⋮	⋮	…
N	$\{(Y_N^t, Y_N^{t,l}, Y_N^{t,r})_T, p_N^t\}$	$\{(X_{N1}^t, X_{N1}^{t,l}, X_{N1}^{t,r}), q_{N1}^t\}, \ldots, \{(X_{NN}^t, X_{NN}^{t,l}, X_{NN}^{t,r}), q_{NN}^t\}$

TABLE II. INPUT-OUTPUT DATA WITH CONFIDENCE INTERVAL

Sample	Output	Inputs
i	$I[e_Y, \sigma_Y]$	$I[e_{X_1}, \sigma_{X_1}]$, … , $I[e_{X_N}, \sigma_{X_N}]$
1	$I[e_{Y1}, \sigma_{Y1}]$	$I[e_{X_{11}}, \sigma_{X_{11}}]$, … , $I[e_{X_{1N}}, \sigma_{X_{1N}}]$
2	$I[e_{Y2}, \sigma_{Y2}]$	$I[e_{X_{21}}, \sigma_{X_{21}}]$, … , $I[e_{X_{2N}}, \sigma_{X_{2N}}]$
⋮	⋮	
N	$I[e_{YN}, \sigma_{YN}]$	$I[e_{X_{N1}}, \sigma_{X_{N1}}]$, … , $I[e_{X_{NN}}, \sigma_{X_{NN}}]$

A. *Initial structure of CI-TVSVRNs*

How to build the initial structure is an important issue of the neutral networks algorithm [16]. In our algorithm, we use the support vector regression (SVR) algorithm to build the initial structure of the dual neural network, because using the SVR algorithm to build the initial structures of the networks is equivalent to solve a quadratic programming problem [17]. Besides, the number of hidden nodes and adjustable parameters are easily obtained given the fixed structure of SVR [18]. However, the standard SVR algorithms cannot handle the interval data [19]. Thus, we developed a novel SVR algorithm that can handle the interval data, called a dual *v*-support vector regression (DVSVR) algorithm.

Once the intervals in the input-output data $\left([\underline{X}, \overline{X}], [\underline{Y}, \overline{Y}]\right)$ have been determined, the DVSVR algorithm concurrently calculates the lower and upper sides of the interval output data $f^+(x)$ determines the upper side of the interval output data; $f^-(x)$ determines the lower side as follows:

$$f^+(x) = \left\langle w^+ \cdot \overline{x_i} \right\rangle + b^+ \tag{8}$$

$$f^-(x) = \left\langle w^- \cdot \underline{x_i} \right\rangle + b^- \tag{9}$$

The functions $f^+(x)$ and $f^-(x)$ are derived by solving the following pair of quadratic programming problems:

$$\min \frac{1}{2}\|w^+\|^2 + C_1\left(v_1 b^+ + \frac{1}{N}\sum_{i=1}^{N}\zeta_{1i}\right)$$
$$\text{s.t. } \left\langle w^+ \cdot \overline{x_i} \right\rangle + b^+ \geq \overline{y_i} - \xi_{1i} \text{ and } \xi_{1i} \geq 0 \text{ for } i = 1, \ldots, N. \tag{10}$$

and

$$\min \frac{1}{2}\|w^-\|^2 + C_2\left(v_2 b^- + \frac{1}{N}\sum_{i=1}^{N}\zeta_{2i}\right)$$
$$\text{s.t. } \left\langle w^- \cdot \underline{x_i} \right\rangle + b^- \geq \underline{y_i} + \xi_{2i} \text{ and } \xi_{2i} \geq 0 \text{ for } i = 1, \ldots, N. \tag{11}$$

where $\|w^{+/-}\|^2$ is the regularization term, C_1, $C_2 \geq 0$ are the regularization parameters and ξ_{1i}, ξ_{2i} are the slack variables. Parameter $v_{1,2} \in (0, 1)$ controls the trade-off between the minimization of $b^{+/-}$ and the minimization of errors.

The dual problem of (10) and (11) can be obtained as:

$$\max -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} a_{1i}a_{1j}\left\langle \overline{x_i} \cdot \overline{x_j} \right\rangle + \sum_{i=1}^{N} a_{1i}\overline{y_i}$$
$$\text{s.t. } \sum_{i=1}^{N} a_{1i} = C_1 v_1 \text{ and } a_{1i} \in \left[0, {C_1}/{N}\right], \; i = 1, \ldots, N. \tag{12}$$

and

$$\max -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} a_{2i}a_{2j}\left\langle \underline{x_i} \cdot \underline{x_j} \right\rangle - \sum_{i=1}^{N} a_{2i}\underline{y_i}$$
$$\text{s.t. } \sum_{i=1}^{N} a_{2i} = C_2 v_2 \text{ and } a_{2i} \in \left[0, {C_2}/{N}\right], \; i = 1, \ldots, N. \tag{13}$$

After solving the problem (12) and (13), the weight vector can be obtained:

$$w^+ = \sum_{i=1}^{N} a_{1i}\overline{x_i} \tag{14}$$

$$w^- = -\sum_{i=1}^{N} a_{2i}\underline{x_i} \tag{15}$$

The parameter b^+ and b^- can be determined from the KKT (Karush–Kuhn–Tucker) conditions

$$a_{1i}\left[\left\langle w^+ \cdot \overline{x_i} \right\rangle + b^+ - \overline{y_i} + \xi_{1i}\right] = 0 \tag{16}$$

$$\left(\frac{C_1}{N} - a_{1i}\right)\xi_{1i} = 0 \tag{17}$$

and

$$a_{2i}\left[\underline{y_i} - \left\langle w^- \cdot \underline{x_i} \right\rangle - b^- + \xi_{2i}\right] = 0 \tag{18}$$

$$\left(\frac{C_2}{N} - a_{2i}\right)\xi_{2i} = 0 \tag{19}$$

from equations (17) and (19), some $a_{1i} \in (0, C_1/N), \xi_{1i} = 0$ and $a_{2i} \in (0, C_2/N)$, $\xi_{2i} = 0$ can be obtained. Moreover, the second factors in (16) and (18) are zero. Hence, b^+ and b^- can be calculated by:

$$\max\left\{b^{+}=\overline{y_i}-\left\langle w^{+}\cdot\overline{x_i}\right\rangle\right\} \text{ for some } a_{1i}\in\left(0,C_1/N\right) \quad (21)$$

and

$$\min\left\{b^{-}=\underline{y_i}-\left\langle w^{-}\cdot\underline{x_i}\right\rangle\right\} \text{ for some } a_{2i}\in\left(0,C_2/N\right) \quad (22)$$

Finally, the upper bound function $f^{+}(x)$ and the lower bound function $f^{-}(x)$ of the interval output data are built as:

$$f^{+}(x)=\sum_{i=1}^{N}\alpha_{1i}\left\langle\overline{x}\cdot\overline{x_i}\right\rangle+b^{+} \quad (24)$$

and

$$f^{-}(x)=-\sum_{i=1}^{N}a_{2i}\left\langle\underline{x}\cdot\underline{x_i}\right\rangle+b^{-} \quad (25)$$

The training samples with α_{1i}, $\alpha_{2i}>0$ are termed support vectors, because only these points determined the final regression function, and according to the KKT conditions given by (16), (17), (18), and (19), only training samples that are outside the insensitive zone, and lying on the upper or lower side function are captured as support vectors, so the number of support vectors is very few. Hence, the prediction speed of the proposed dual *v*-support vector regression algorithm is fast.

B. Extension to Nonlinear Regression Model

The regression model estimated by equations (10) and (11) is a linear function in $\boldsymbol{R}^n$. In order to extend it to nonlinear regression model, the concept of kernel function is employed, which utilizes a nonlinear transform $\Phi:\boldsymbol{R}^n\rightarrow\boldsymbol{F}$ to map the data points into a higher dimensional feature space $\boldsymbol{F}$ and estimates a linear regression model in the feature space $\boldsymbol{F}$. After mapping all data points into the feature space $\boldsymbol{F}$, the formulation of the DVSVR algorithm would only depend on the data through inner products in $\boldsymbol{F}$, i.e., on the functions of the form $\left\langle\Phi(x_i)\cdot\Phi(x_j)\right\rangle$. Hence, it suffices to know and use the kernel function $K(x_i,x_j)=\left\langle\Phi(x_i)\cdot\Phi(x_j)\right\rangle$ instead of $\Phi(\bullet)$ explicitly. Replacing $\left\langle x_i\cdot x_j\right\rangle$ with $K(x_i,x_j)$ in the problems (12) and (13), the following dual quadratic programming problems can be obtained:

$$\max -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}a_{1i}a_{1j}K\left(\overline{x_i}\cdot\overline{x_j}\right)+\sum_{i=1}^{N}a_{1i}\overline{y_i}$$
$$\text{s.t.} \sum_{i=1}^{N}a_{1i}=C_1v_1 \text{ and } a_{1i}\in\left[0,{C_1}/{N}\right],\ i=1,\ldots,N. \quad (26)$$

and

$$\max -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}a_{2i}a_{2j}K\left(\underline{x_i}\cdot\underline{x_j}\right)-\sum_{i=1}^{N}a_{2i}\underline{y_i}$$
$$\text{s.t.} \sum_{i=1}^{N}a_{2i}=C_2v_2 \text{ and } a_{2i}\in\left[0,{C_2}/{N}\right],\ i=1,\ldots,N. \quad (27)$$

The upper and lower bound functions respectively are:

$$f^{+}(x)=\sum_{i=1}^{N}\alpha_{1i}K\left(\overline{x},\overline{x_i}\right)+b^{+} \quad (28)$$

and

$$f^{-}(x)=-\sum_{i=1}^{N}a_{2i}K\left(\underline{x},\underline{x_i}\right)+b^{-} \quad (29)$$

C. Learning algorithms of CI-DNN

In outlined above, CI-DNN derives the initial structures for the dual neural network through a DVSVR algorithm based on the initial values $\{w^{+}, b^{+}\}$ and for $\{w^{-}, b^{-}\}$. However, it is well-known that performance in a SVR algorithm can be improved by properly tuning the network parameters with a learning schemes [20]. Hence, it is desirable to apply a learning algorithm to tune the parameters of two neural networks under the initial structure obtained by the DVSVR algorithm to improve the performance.

The genetic algorithm can search the objective function in the parallel space, and check a number of possible solutions at one time. It can search the global optimal or suboptimal solution in the solution space, and it has an excellent ability of global searching. Hence, through the use of genetic algorithms to optimize original weights and biases of neural network, a better searching space in solution space could be obtained, but the learning speed of genetic algorithm is a problem [21]. Therefore, we propose a dynamic genetic backpropagation algorithm (DGBP) to tune the parameters of two neutral networks under the initial structure.

The DGBP algorithm is similar to genetic back propagation algorithm [20]:

- Data normalization: all data are normalized to the values between [-1-1].
- Importing data: importing 2/3 input and output data as training datasets, and the rest of the input and output data as testing datasets.
- Configuration parameters of the network: Then selecting the basic structure parameters of network according to the target problem. Due to only two output values, two output neurons are there in output layer, and the number of input nodes will be equal to the number of features. The number of hidden-layer is one. Neurons within the hidden layer relies on the fixed structure of the dual v-support vector regression model, and in our algorithm, we use two dual v-support vector regression models to identify the two output data.
- Establishing and running the genetic algorithms: setting up initial population size. It will be kept between 20

and 30. Then setting up other operating parameters, the number of generations Ngen should range between 200 and 800, and the chromosome will also be encoded as real values. The values of crossover is taken as 1, and the crossover rate ranges between 0.5-0.8. Besides, in the proposed algorithm, flip the gene value with a random value within range of [-1,1], which is same as the range considered for assigning initial weights and mutation rate is kept 0.01.

- Information transfer: information passed from the input layer to the hidden-layer(s) and then to the output layer through the weights, biases and the DVSVR algorithm to obtain the output prediction results.
- Training the network, or call the error back propagation stage: calculating the errors between the neural network outputs and the desired ones. If the errors were less than the predefined threshold 0.01, stopped training and got the final network architecture. Otherwise, a dynamic backpropagation algorithm adjusted weights and biases along with the reverse path of the information transfer. The learning rate of backpropagation algorithm be taken in the range of [0.5-0.8], and the number of epochs is kept in range of [300-500]. Repeating this process until the network was able to predict the given outputs within desired tolerance 0.01.

Therefore, the difference between DGBP algorithm and traditional genetic backpropagation algorithm is that when we are calculating the errors between the neural network outputs and the desired ones, we use a dynamic weight to represent the degree of error in the cost function rather than a constant value. Thus, for upper side network, the cost function is defined as:

$$E^{+}(t)=\frac{1}{2}\sum_{i=1}^{N}E_i^{+}=\frac{1}{2}\sum_{i=1}^{N}\mu_i\left[e_i(t)\right]^2 \tag{30}$$

$$e_i(t)=\overline{y_i}-f^{+}(x_i) \tag{31}$$

where t is the epoch number, $\overline{y_i}$ is the high value of the interval output, $e_i(t)$ is the error between $\overline{y_i}$ and $f^{+}(x_k)$ (the output of upper side network at epoch t), and the dynamic weighting parameter μ_i is defined as:

$$\mu_i=\begin{cases}1 & \text{if } \overline{y_i}>f^{+}(x_i)\\ \mu(t) & \text{if } y_i\le f^{+}(x_i)\end{cases} \tag{32}$$

where $\mu(t)$ is a small positive value. To accelerate the learning speed, a decreasing function is used for $\mu(t)$ rather than a constant. The decreasing function, as defined in [22], is:

$$\mu(t)=\frac{1}{\left[1+(t/2000)\right]^3} \tag{33}$$

A gradient-descent kind of learning algorithm is then directly used to update initial values $\{w^{+}, b^{+}\}$. These updated rules are:

$$\Delta w_i^{+}=-\eta\frac{\partial E_i^{+}}{\partial w_i^{+}}=\eta\mu_i e_i K\left(\overline{x},\overline{X}\right) \tag{34}$$

$$\Delta b_i^{+}=-\eta\frac{\partial E_i^{+}}{\partial b_i^{+}}=\eta\mu_i e_i \tag{35}$$

where η is a learning constant. Similarly, to train the lower side network, the dynamic weighting parameter in the cost function as (32) is replaced by:

$$\mu_i=\begin{cases}\mu(t) & \text{if } \underline{y_i}\ge f^{-}(x_i)\\ 1 & \text{if } \underline{y_i}<f^{-}(x_i)\end{cases} \tag{36}$$

where $\underline{y_i}$ is the low value of the interval output. the initial values $\{w^{-}, b^{-}\}$ in the lower side network are also updated by (34) and (35).

The Fig.1 demonstrate the paradigm of CI-DNN. Moreover, since the CI-DNN have good initial structures through the DVSVR algorithm due to its optimization nature, the CI-DNN with the DGBP algorithm have a very fast convergent speed.

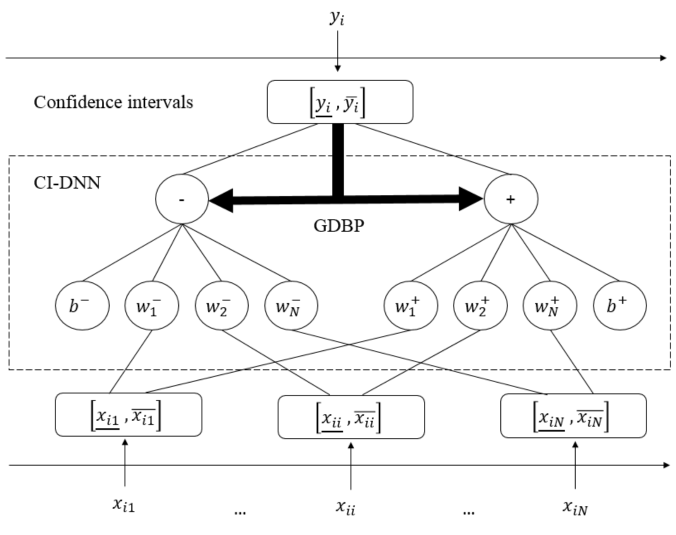

y_i
Confidence intervals
$[\underline{y_i}, \overline{y_i}]$
CI-DNN
-
+
GDBP
b^-
w_1^-
w_2^-
w_N^-
w_1^+
w_2^+
w_N^+
b^+
$[\underline{x_{i1}}, \overline{x_{i1}}]$
$[\underline{x_{ii}}, \overline{x_{ii}}]$
$[\underline{x_{iN}}, \overline{x_{iN}}]$
x_{i1}
...
x_{ii}
...
x_{iN}

Fig. 1. The structure of CI-DNN

IV. Experiments

In this section, we use several artificial examples to validate the effectiveness and performance of the proposed CI-DNN algorithm. All examples are implemented with a radial basis function (RBF) kernel (37) in MATLAB version 2016a running on a Windows 7 PC with an Intel Core i5 processor (2.40 GHz) and 8-GB RAM. For simplicity, the model parameters were set as $C_1 = C_2 = C$ and $v_1 = v_2 = v$. The values for the parameter C were selected from $\{10^i \mid i = 0, 1, \ldots, 6\}$;

from {0.01, 0.02, ..., 0.4} for parameter q; and from {0.01, 0.02, ..., 0.6} for parameter v.

$$K(u,v)=\exp\left\{-q\|u-v\|^2\right\} \tag{37}$$

A. Examples

In this section, we use two artificial examples to illustrate our proposed CI-DNN algorithm, we firstly estimate a linear function. The training data sets firstly were generated by:

$$\begin{cases} y_k = 0.02x_k + 1 \\ x_k = k,\ k = 1,2,\ldots 50 \end{cases} \tag{38}$$

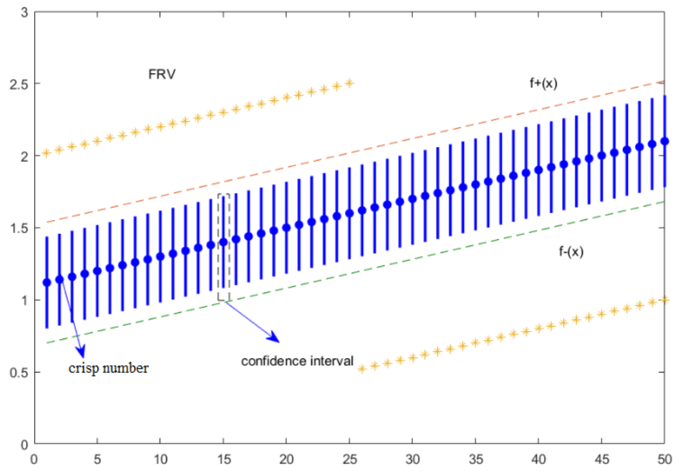
We then design a method to transform these crisp numbers into fuzzy random variables. To replicate randomness, we used a noise variable e_k to perturb x_k, (i.e., $x_k = k + e_k$) with a 50 percent probability that the crisp number would become 1, and a 50 percent probability that it would become -1. To replicate fuzziness, we used the triangular fuzzy numbers (m,α,β) to represent the fuzziness, where $\alpha=\beta=0.5$. The same method was used to handle the dependent variables y_k. This process resulted in the fuzzy random input-output training set shown in Table III.

TABLE III. FUZZY RANDOM INPUT-OUTPUT DATA

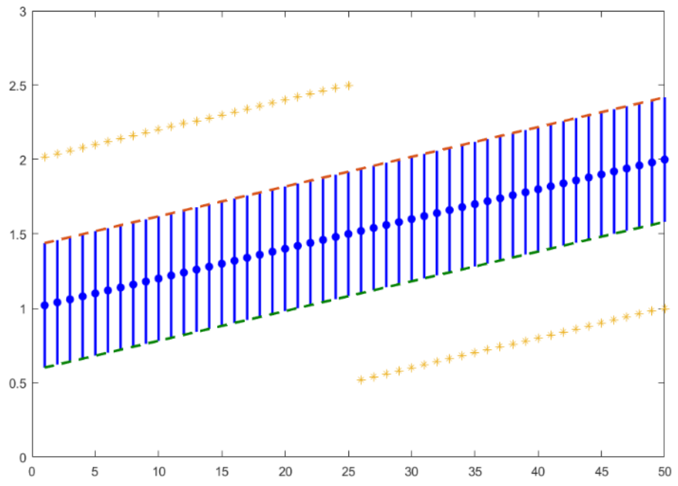
Sample	Output	Inputs
1	{(3.5, 4, 4.5),0.5}, {(5.5, 6, 6.5),0.5}	{(-0.5, 0, 0.5),0.5}, {(1.5, 2, 2.5),0.5}
2	{(9.5, 10, 10.5),0.5}, {(11.5, 12,12.5),0.5}	{(0.5, 1, 1.5),0.5}, {(2.5, 3, 3.5),0.5}
⋮	⋮	
50	{(99.5, 100, 100.5),0.5}, {(101.5, 102, 102.5),0.5}	{(48.5, 49, 49.5),0.5}, {(50.5, 51, 51.5),0.5}

We then use the data in the Table III as training data sets to train our proposed CI-DNN algorithm. From above section, we know in the CI-DNN algorithm, we use the DVSVR algorithm to produce two functions in the first step, these two functions identify the upper side and lower side of the confidence intervals, and we use the DGBP algorithm in the next step to tune the parameters in the DVSVR algorithm. Fig.2 shows the results of the two steps.

In the Fig.2, the crisp numbers (blue solid dots) were produced by the function (38), the fuzzy random variables (yellow asterisks) based on the crisp numbers were produced by the method described previously, and the confidence intervals (vertical blue lines) of the fuzzy random variables were produced by function (7) and the confidence intervals always include the true values. In the Fig.2.(a), the CI-DNN algorithm produces two functions f^{+}(x) and f(x) by the DVSVR algorithm to identify the upper side and lower side of the confidence intervals, but the two functions cannot to fit the confidence intervals suitably. The reason is the parameters in the DVSVR algorithm not be set suitably.



(a). Using an DVSVR algorithm to produce two functions



(b). Using an DGBP algorithm to turn the parameters in the DVSVR algorithm

Fig. 2. The results of linear regression

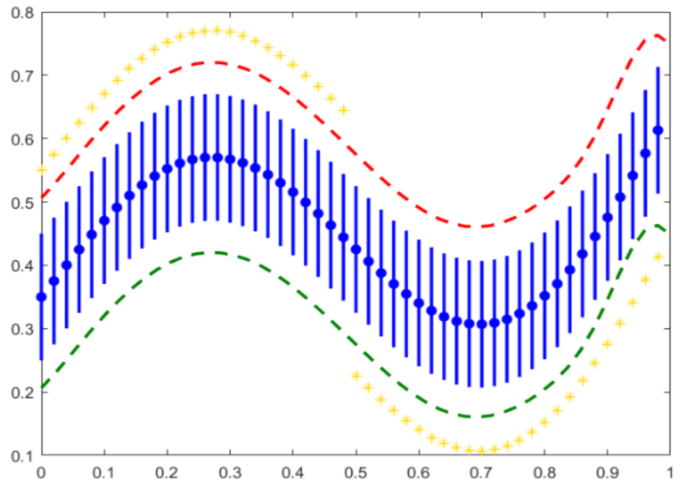
But in the Fig.(2).(b), we can see the CI-DNN tightly fits the confidence intervals, so it shows that the DGBP algorithm improves the CI-DNN algorithm's performance. In other word, the parameters in the DVSVR algorithm were turned suitably by the DGBP algorithm.

Next, we estimated a nonlinear function. The training data sets were generated by:

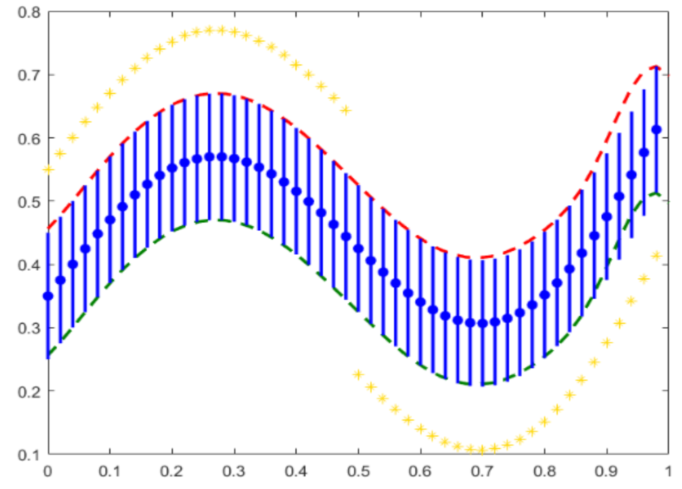
$$\begin{cases} y_k = 0.2\sin(2\pi x_k) + 0.2x_k^2 + 0.3 + \left(0.1x_k^2 + 0.05\right) \\ x_k = 0.02\left(k-1\right),\ k = 1,2,\ldots 50 \end{cases} \tag{39}$$

The method described previously was again used to transform crisp numbers into fuzzy random variables. However, in this case, e_k = 0.2 or -0.2. We also use two figures to illustrate the CI-DNN algorithm. The results of using an DVSVR algorithm to produce two functions are shown in Fig.3.(a), and the results of using the DGBP algorithm to adjust two functions shown in the Fig.3.(b). From the Fig.3, we can get the same results in the Fig.2.

Therefore, we can get three conclusions: (1) whether the relationship between the fuzzy random variables is linear or nonlinear, the confidence intervals are good variables to replace the fuzzy random variables, (2) the fuzzy random regression problems can be translated into the interval regression problems. (3) the CI-DNN algorithm can solve the interval regression problems. In the next sections, we illustrate CI-DNN algorithm not only can solve interval regression problems, but also can efficiently solve the interval regression problems.



(a). Using an DVSVR algorithm produces two functions



(b). Using an DGBP algorithm to turn the parameters in the DVSVR algorithm

Fig. 3. The results of nonlinear regression

B. *DVSVR Algorithm*

Due to the initial structure of the CI-DNN algorithm is constructed by the DVSVR algorithm, so if the DVSVR algorithm is an effective algorithm to solve the interval regression problem, the CI-DNN algorithm will be an efficient algorithm. Based on the reason, we compare the DVSVR algorithm with TSVR algorithm [22] and ITSVR algorithm [23] in the second experiment. These two algorithms are also designed for interval regression problem. Table IV shows the comparison result based on the training data sets were generated by the sinc function $y = \sin c(x),\ x \in [-3\pi, 3\pi]$.

TABLE IV. COMPARISON RESULT OF SINC FUNCTION

Algorithm	**RMSE**	**Time (s)**
TSVR	0.1633	0.0530
ITSVR	0.1095	0.0520
DVSVR	0.0722	0.0510

We used 10-fold cross-validation to test the perfomence, Table IV lists the average prediction results of the three algorithms, including the RMSE [23], Time, From Table IV, we can find that our algorithm obtains the better prediction performance than other two algorithms.

TABLE V. COMPARISON RESULT OF POLYNOMIAL FUNCTION

Algorithm	**RMSE**	**Time (s)**
TSVR	4.2200	0.0345
ITSVR	3.9856	0.0290
DVSVR	3.1976	0.0282

We then also use another training data sets to test the DVSVR algorithm. The second data sets is the ‘‘order wise’’ polynomial problem with fuzzy data points [24]. Table V also shows the comparison result in terms of RMSE and Time (seconds) and the results in Table V also show our algorithm obtains the better performance than other two algorithms, so the DVSVR algorithm is an efficient and applicability algorithm.

C. *DGBP Algorithm*

Due to in CI-DNN algorithm, we use the DGBP algorithm to turn the parameters in the DVSVR algorithm, so if the DGBP algorithm is an effective algorithm to turn the parameters in the DVSVR algorithm, the CI-DNN algorithm will be a efficient algorithm to solve the interval regression problem. Based on the reason, in the last experiment, we test the performance of the DGBP algorithm. We compare the performance of the DVSVR algorithm based on using the traditional backpropagation algorithm [25], the genetic backpropagation algorithm [25], and the DGBP algorithm. The training data sets is used from [23]. It includes 41 training points. Table VI shows the comparison result in terms of RMSE and Time.

TABLE VI. COMPARISON RESULT (ISVIRN AND CI-DNN)

Algorithm	**RMSE**	**Time (s)**
DVSVR+BP	0.1015	0.1630
DVSVR+GB	0.1000	0.1561
DVSVR+DGBP (CI-DNN)	0.1001	0.1272

The results in Table VI show that the DGBP algorithm improves the DVSVR algorithm’s performance and reduce the training speed than other algorithms, so the DGBP algorithm is an efficient and applicability algorithm.

V. Conclusion And Future Works

This paper presents an algorithm, called CI-DNN, to fill the gap in effective algorithms that can deal with fuzzy random regression problems. The CI-DNN algorithm is robust to uncertainty where randomness and fuzziness exist at the same time. Besides, the experimental results indicate that the CI-DNN algorithm brings four advantages: (1) the learning speed of CI-DNN is fast; (2) the CI-DNN also has good generalization performance; (3) the number of hidden nodes and adjustable parameters of the CI-DNN are easily obtained; and (4) the CI-DNN has a very fast convergent speed. Nowadays, technology advanced make it possible to access fast and infinite data, such as data streams. Solving the regression problem of data streams, an incremental regression algorithm need to be proposed, but the CI-DNN algorithm is a batch learning algorithm, so in the future research works, we are plan to extend the CI-DNN algorithm to an incremental algorithm.

VI. Acknowledgement

The work presented in this paper was supported by Australian Research Council (ARC) under discovery grant DP170101632.

References

[1] H. Zuo, G. Zhang, W. Pedrycz, V. Behbood, and J. Lu, "Granular Fuzzy Regression Domain Adaptation in Takagi-Sugeno Fuzzy Models," IEEE Trans. Fuzzy Syst., vol. 6706, no. c, pp. 1–1, 2017.

[2] W. J. Lee, H. Y. Jung, J. H. Yoon, and S. H. Choi, "The statistical inferences of fuzzy regression based on bootstrap techniques," Soft Comput., vol. 19, no. 4, pp. 883–890, 2015.

[3] R. Gao and D. A. Ralescu, "Convergence in Distribution for Uncertain Random Variables," IEEE Trans. Fuzzy Syst., vol. 6706, no. c, pp. 1–1, 2017.

[4] C. M. Grinstead and J. L. Snell, "Introduction to Probability," Swart. Coll., pp. 1–520, 2007.

[5] J. G. Dijkman, H. Van Haeringen, S. J. De Lange, and L. Zadeh, "Fuzzy Numbers," J. Math. Anal. Appl., vol. 92, pp. 301–341, 1983.

[6] H. Kwakernaak, "Fuzzy random variables-I. definitions and theorems," Inf. Sci. (Ny)., vol. 15, no. 1, pp. 1–29, 1978.

[7] W. Näther, "Regression with fuzzy random data," Comput. Stat. Data Anal., vol. 51, no. 1, pp. 235–252, 2006.

[8] G. Hesamian and M. Shams, "Parametric testing statistical hypotheses for fuzzy random variables," Soft Comput., vol. 20, no. 4, pp. 1537–1548, 2016.

[9] K. Yao and J. Gao, "Law of Large Numbers for Uncertain Random Variables," IEEE Trans. Fuzzy Syst., vol. 24, no. 3, pp. 615–621, Jun. 2016.

[10] G. Hesamian and M. G. Akbari, "Nonparametric Kernel Estimation Based on Fuzzy Random Variables," IEEE Trans. Fuzzy Syst., vol. 25, no. 1, pp. 84–99, Feb. 2017..

[11] B. Liu and Y. K. Liu, "Expected value of fuzzy variable and fuzzy expected value models," IEEE Trans. Fuzzy Syst., vol. 10, no. 4, pp. 445–450, 2002.

[12] Y. K. Liu and B. Liu, "Fuzzy random variables: A scalar expected value operator," Fuzzy Optim. Decis. Mak., vol. 2, no. 2, pp. 143–160, 2003.

[13] J. Watada and W. Pedrycz, "Building Confidence-Interval-Based Fuzzy Random Regression Models," IEEE Trans. Fuzzy Syst., vol. 17, no. 6, pp. 1273–1283, 2009.

[14] M. Pratama, J. Lu, and G. Zhang, "Evolving Type-2 Fuzzy Classifier," IEEE Trans. Fuzzy Syst., vol. 24, no. 3, pp. 574–589, 2016.

[15] K. C. Hung and K. P. Lin, "Long-term business cycle forecasting through a potential intuitionistic fuzzy least-squares support vector regression approach," Inf. Sci. (Ny)., vol. 224, pp. 37–48, 2013.

[16] C. Juang, S. Member, and Y. Tsao, "A Self-Evolving Interval Type-2 Fuzzy Neural Network With Online Structure," IEEE Trans. Fuzzy Syst., vol. 16, no. 6, pp. 1411–1424, 2008.

[17] X. Yang, G. Zhang, J. Lu, and J. Ma, "A kernel fuzzy c-Means clustering-based fuzzy support vector machine algorithm for classification problems with outliers or noises," IEEE Trans. Fuzzy Syst., vol. 19, no. 1, pp. 105–115, 2011.

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

[19] A. J. Smola, B. Sch, and B. Schölkopf, "A Tutorial on Support Vector Regression," Stat. Comput., vol. 14, no. 3, pp. 199–222, 2004.

[20] H. Wang and D. Xu, "Parameter Selection Method for Support Vector Regression Based on Adaptive Fusion of the Mixed Kernel Function," J. Control Sci. Eng., vol. 2017, pp. 1–12, 2017.

[21] H.-X. Huang, J.-C. Li, and C.-L. Xiao, "A proposed iteration optimization approach integrating backpropagation neural network with genetic algorithm," Expert Syst. Appl., vol. 42, no. 1, pp. 146–155, 2015.

[22] X. Peng, "TSVR: An efficient Twin Support Vector Machine for regression," Neural Networks, vol. 23, no. 3, pp. 365–372, 2010.

[23] X. Peng, D. Chen, L. Kong, and D. Xu, "Interval twin support vector regression algorithm for interval input-output data," Int. J. Mach. Learn. Cybern., vol. 6, no. 5, pp. 719–732, 2015.

[24] P. D'Urso and T. Gastaldi, "An 'orderwise' polynomial regression procedure for fuzzy data," Fuzzy Sets Syst., vol. 130, no. 1, pp. 1–19, 2002.

[25] M. N. H. Siddique and M. O. Tokhi, "Training neural networks: backpropagation vs. genetic algorithms," IJCNN01 Int. Jt. Conf. Neural Networks Proc. Cat No01CH37222, vol. 4, pp. 2673–2678, 2001.