Hindawi
Complexity
Volume 2019, Article ID 2379584, 17 pages
https://doi.org/10.1155/2019/2379584

Research Article

Distributed Gaussian Granular Neural Networks Ensemble for Prediction Intervals Construction

Chunyang Sheng,[1] Haixia Wang,[2] Xiao Lu ,[2] Zhiguo Zhang,[2] Wei Cui ,[1] and Yuxia Li[2]

[1]*College of Electrical Engineering and Automation, Shandong University of Science and Technology, Qingdao 266590, China*
[2]*Key Laboratory for Robot & Intelligent Technology of Shandong Province, Shandong University of Science and Technology, Qingdao 266590, China*

Correspondence should be addressed to Xiao Lu; luxiao98@163.com

Received 25 February 2019; Revised 22 May 2019; Accepted 12 June 2019; Published 3 July 2019

Academic Editor: Sergio Gómez

To overcome the weakness of generic neural networks (NNs) ensemble for prediction intervals (PIs) construction, a novel Map-Reduce framework-based distributed NN ensemble consisting of several local Gaussian granular NN (GGNNs) is proposed in this study. Each local network is weighted according to its contribution to the ensemble model. The weighted coefficient is estimated by evaluating the performance of the constructed PIs from each local network. A new evaluation principle is reported with the consideration of the predicting indices. To estimate the modelling uncertainty and the data noise simultaneously, the Gaussian granular is introduced to the numeric NNs. The constructed PIs can then be calculated by the variance of output distribution of each local NN, i.e., the summation of the model uncertainty variance and the data noise variance. To verify the effectiveness of the proposed model, a series of prediction experiments, including two classical time series with additive noise and two industrial time series, are carried out here. The results indicate that the proposed distributed GGNNs ensemble exhibits a good performance for PIs construction.

1. Introduction

It is important for a predictor to be able to estimate the reliability of the prediction results. However, the point-oriented prediction can only provide the estimated values without any indication of their reliability [1]. Compared to the point-oriented methods, PIs construction can be employed to quantify the modelling uncertainty and the data noise. It becomes more popular in the field of data-based prediction [1, 2]. The models adopted for prediction mainly include fuzzy model [3], support vector machine (SVM) [4, 5], and neural networks (NNs) [6–8], in which NNs-based models are the most commonly used for PIs construction.

The variance-estimation based numeric NNs, a class of typical PIs construction methods, have been proposed in the literatures, such as the delta [7], mean variance estimation (MVE) [9], and Bayesian techniques [10]. The delta is applied to the NN modelling for PIs construction [11] with the assumption that the target variance is a constant for all the samples. However, such assumption deviates from real-world conditions, while the MVE method estimates the target variance by using a dedicated NN on an assumption that prediction errors are normally distributed around the true mean of the targets. As such, the PIs can be constructed if the parameters of the distribution are known [1, 9]. For the Bayesian technique based NNs, Gaussian probability distribution is considered over the weight values of NN, and the corresponding outputs can be approximated by a Gaussian distribution [12, 13]. The essence of the Bayesian NN can be regarded as a kind of Gaussian granular NN, in which the Gaussian probability density function (PDF) represents an information granularity. In such a way, the variance of the Gaussian PDF is the summation of the variances of the modelling uncertainty and the data noise, where the PIs can be built up from the variance of the output Gaussian PDF [14].

In addition, the interval-valued NNs modelling is another class of PIs construction method, in which the bounds of the intervals are considered. A multilayer perceptron (MLP) is proposed in [15] to learn the relationship between the interval-valued inputs and outputs, and the trained MLP

comes with interval-valued weights and biases. Another interval-valued network is also reported in [16], where the neurons connections are denoted by a series of interval values, while the modelling inputs are numerical values. Besides, considering how to utilize the interval information directly, a lower upper bound estimation (LUBE) based NN with two outputs is proposed in [17] for estimating the PI bounds, in which the two outputs are the upper and lower bounds of the PIs, respectively. The training objective of NN in [17] is a function of two evaluation indices of PIs, i.e., the PIs coverage probability and the mean width of PIs which can be calculated based on the PI bounds.

It is apparent that both classes of PIs construction mentioned above are based on the NN individual. When coping with large-scale or strongly nonlinear problems, such methods may exhibit weak modelling capacities due to their limited learning ability. Besides NN individual, modular NNs [18] and NNs ensemble [19] are also applied for classification, recognition, and prediction. For instance, a kind of modular NNs is proposed in [18] and applied for human recognition in [20–22], which improves the recognition rates compared with other existing methods. An NNs ensemble is reported in [23] with Type-1 and Type-2 fuzzy integration for time series prediction, the parameters of which are optimized by using the particle swarm optimization (PSO) algorithm. In addition, a network ensemble structure is designed in [24, 25] for PIs construction compared to the network individual-based methodology. It is still a challenging task to organize an effective NN ensemble despite their popularity, because of three basic limitations of the ensemble structure. (1) The performance of an NN ensemble depends on each local network. If some local networks are biased, the accuracy of the ensemble modelling will be severely deteriorated [26, 27]. Moreover, it is rather hard for all local networks to guarantee their unbiased features. (2) The modelling uncertainty variance and the data noise variance are estimated for PIs construction [1, 2]. Another independent NN is required for the noise variance estimation. It may be more effective if these variances are estimated simultaneously. (3) Although an NNs ensemble consists of a number of NN individuals, it is not suitable for distributed computing. In such a way, the traditional ensemble approaches demand a high computational load and is troublesome for the real-time applications [28].

Granular computing is originally proposed by L. Zadeh [30, 31]. A granule may be interpreted as an interval, a set, a probability density function, or a distribution. Combining with granular computing, the shortcomings of the NNs ensemble can be properly overcome. The NNs ensemble can achieve better performance in many fields, such as time series prediction, human recognition, and pattern recognition. However, at present, there is still a lack of some relative research on the granular NNs ensemble-based PIs.

In this study, a Map-Reduce based distributed computing structure composed of a number of local Gaussian granular NNs (GGNNs) is proposed in this study for PIs construction. To avoid the ensemble being performed badly due to biased local networks, each local network of the proposed ensemble is specified by a penalty coefficient to weight their contributions to the ensemble. The coefficient can be estimated by evaluating the performance of PIs constructed by the local networks, where a new evaluation principle is designed here with the consideration of the interval width, the coverage probability, and the accuracy. To simultaneously estimate the uncertainties due to modelling and data noises, GGNNs are constructed as the local networks, and the neurons connections and the outputs are represented by Gaussian distributions. The prediction variance of the local GGNN has two components that are used to quantify the variance of modelling uncertainty and the data noise, respectively. In addition, a Map-Reduce programming framework is developed here in order to enhance the computing efficiency. To verify the effectiveness of the proposed model, a number of prediction tasks, including the noisy classical prediction problems and industrial data, are considered here. The experimental results indicate that the proposed distributed GGNNs exhibit a good performance for PIs construction.

The rest of this paper is organized as follows. First, the Gaussian PDF based granularity and the Gaussian granular based reasoning are introduced. The distributed GGNN is also proposed. Second, the parameter estimation of the distributed GGNNs is described, including the intrinsic parameters of local networks and the contribution coefficients of local networks. A Map-Reduce framework is developed for the distributed processing as well. Third, the quality of the proposed ensemble is experimentally verified. Finally, the conclusions are given.

2. Structure of the Distributed GGNNs

As for the drawbacks of the conventional ensemble structure analysed in the introduction, the contribution of this study lies in a distributed NNs ensemble, which consists of several local GGNNs whose connections are described by Gaussian PDFs. In the proposed ensemble, it does not depend on the bootstrap method to estimate the variance of the predicted targets. Here, the PIs are constructed based on a set of variances of the Gaussian probability densities of the outputs of the local NNs.

2.1. Gaussian PDF Based Granularity. Let X and Y be two independent Gaussian random variables, $X \sim \mathcal{N}(\mu_X, \sigma_X^2)$ and $Y \sim \mathcal{N}(\mu_Y, \sigma_Y^2)$. A linear combination $aX + bY$ is also Gaussian [23]. That is,

$$aX \pm bY \sim \mathcal{N}\left(a\mu_X \pm b\mu_Y, a^2\sigma_X^2 + b^2\sigma_Y^2\right) \tag{1}$$

where a and b are linear coefficients for random variables.

The connecting weights of one network can then be described by Gaussian PDF. We consider two neural networks (echo state network (ESN) [24] and extreme learning machine (ELM) [25]), whose mapping relationships between the unknown weights and the outputs are linear. If the output weights are represented by the Gaussian PDF $W_i^{out} \sim \mathcal{N}(\mu_{W_i}, \sigma_{W_i}^2)$, the network outputs can be represented by a linear combination of several Gaussian PDF. There is no

doubt that the network outputs should follow a Gaussian distribution, denoted by $y \sim \mathcal{N}(\mu_y, \sigma_y^2)$, and

$$\begin{aligned} \mu_y &= \sum_{i=1}^{W} H_i \mu_{W_i}, \\ \sigma_y^2 &= \sum_{i=1}^{W} H_i^2 \sigma_{W_i}^2 \end{aligned} \tag{2}$$

where H_i are some linear coefficients.

In the perspective of information granularity, the Gaussian PDF can be viewed as a form of granularity for granular NNs construction. Nevertheless, the input and internal connections are usually unknown for most of the neural networks, and the mapping relationships of most networks are unlikely linear all the time. If the network weights are represented by Gaussian granules, it is hard to guarantee the output distributions to follow Gaussian distribution. Fortunately, this problem can be addressed with the help of Bayesian and Laplace approximate reasoning.

2.2. Gaussian-Granularity Based Approximate Reasoning. Without loss of generality, one can assume a neural network with the input $\mathbf{u}$ and output t, and the relationship between input and output can be described as

$$t = y(\mathbf{w}, \mathbf{u}) + \varepsilon = \sum_{j=1}^{h} \left[w_j f \left(\sum_{i=0}^{m} w_{ji} u_i \right) \right] + \varepsilon \tag{3}$$

where ε is the white Gaussian noise with mean zero, denoted by $p(\varepsilon) = g(\varepsilon; 0, \sigma_\varepsilon^2)$, and m and h are the numbers of the input and hidden units, respectively.

Since the distribution information of the network weights $\mathbf{w}$ is unknown, the posterior of $\mathbf{w}$ needs to be first estimated by using the training dataset D. Considering the Bayesian rules, the posterior distribution $p(\mathbf{w} \mid D)$ can be formulated by

$$p(\mathbf{w} \mid D) = \frac{[p(D \mid \mathbf{w})\, p(\mathbf{w})]}{p(D)} \tag{4}$$

where $p(D)$ is a normalizing constant, $p(\mathbf{w})$ is the prior distribution, and $p(D \mid \mathbf{w})$ is the likelihood function used to quantify the similar degree between the network outputs and the observed outputs given the weights $\mathbf{w}$.

$$\begin{aligned} p(D \mid \mathbf{w}) &= \prod_{k=1}^{n} p(\varepsilon_k \mid \mathbf{w}) \\ &= \frac{1}{Z_D(\beta)} \exp\left(-\frac{\beta}{2} \sum_{k=1}^{n} \left(y(\mathbf{w}, \mathbf{u}_k) - t_k \right)^2 \right) \end{aligned} \tag{5}$$

where $Z_D(\beta) = (2\pi/\beta)^{n/2}$ is a normalizing constant. Here, β is a hyperparameter related to the variance of the data noise, denoted by $\beta = 1/\sigma_\varepsilon^2$, and n is the number of training samples.

When the distribution information of $\mathbf{w}$ is unknown, the prior distribution of $\mathbf{w}$ can be assumed as a Gaussian distribution with zero mean [26]; i.e.,

$$p(\mathbf{w} \mid \alpha) = \frac{\exp\left(-\alpha \cdot \|\mathbf{w}\|^2 / 2 \right)}{Z_W(\alpha)} \tag{6}$$

where α is a hyperparameter related to the prior distribution of $\mathbf{w}$. $Z_W(\alpha) = (2\pi/\alpha)^{W/2}$ is a normalizing constant. W is the dimension of the weights $\mathbf{w}$.

Substituting the prior and the likelihood into (4), the posterior distribution of $\mathbf{w}$ can be written by using the Laplace approximation.

$$\begin{aligned} &p(\mathbf{w} \mid D) \\ &= \frac{1}{Z_M} \exp\left(-\frac{\beta}{2} \sum_{k=1}^{n} \left[y(\mathbf{w}, \mathbf{u}_k) - t_k \right]^2 - \frac{\alpha}{2} \|\mathbf{w}\|^2 \right) \end{aligned} \tag{7}$$

Using the Bayesian rule and the conditional independence between t and $\mathbf{u}$, the output distribution can be written as (8) given a new input $\mathbf{u}$.

$$p(t \mid \mathbf{u}, D) = \int p(t \mid \mathbf{u}, \mathbf{w})\, p(\mathbf{w} \mid D)\, d\mathbf{w} \tag{8}$$

Now considering the distribution $p(t|\mathbf{u}, \mathbf{w})$, given that the noise ε obeys a Gaussian distribution with zero mean, $p(t \mid \mathbf{u}, \mathbf{w})$ can be rewritten as

$$p(t \mid \mathbf{u}, \mathbf{w}) = p\left([t - y(\mathbf{w}, \mathbf{u})] \mid \mathbf{w} \right) = p(\varepsilon \mid \mathbf{w}) \tag{9}$$

$$p(\varepsilon \mid \mathbf{w}) \propto \exp\left(-\beta \cdot \frac{[y(\mathbf{w}, \mathbf{u}) - t]^2}{2} \right) \tag{10}$$

Introducing (7) and (9) into (8), the output can also be approximated by a Gaussian distribution based on the Laplace approximation.

$$p(t \mid \mathbf{u}, D) = \frac{1}{(2\pi\sigma_t)^{1/2}} \exp\left(-\frac{[t - y(\mathbf{w}_{MP}, \mathbf{u})]^2}{2\sigma_t^2} \right) \tag{11}$$

where $\mathbf{w}_{MP}$ is the optimal weights and σ_t^2 is the prediction variance. To construct the PIs, it is necessary to estimate the prediction variance related to the two hyperparameters α and β as well as the optimal weights $\mathbf{w}_{MP}$.

2.3. Distributed GGNNs Based PIs Construction. In this study, a novel distributed GGNNs ensemble composes of a number of nonrelated local neural networks is proposed. Its architecture is illustrated in Figure 1. Once the distribution information of the local output is obtained, the local network can be used to construct the local PIs. As such, the performance of the PIs constructed by the local network can be evaluated, and the contribution coefficient of each local network can also be computed. One can then obtain the output distribution of the distributed networks by reconciling those of the local networks.

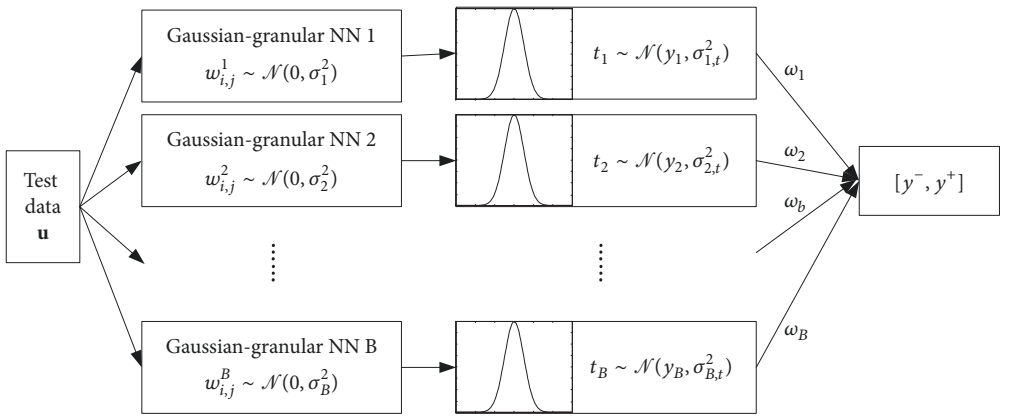

Figure 1: Architecture of the distributed GGNNs.

Given the numeric input u, if the weights of the bth local network obey the Gaussian distribution, i.e., $w_{i,j}^b \sim \mathcal{N}(0, \sigma_b^2)$, the corresponding output can be approximated by a Gaussian distribution $t_b \sim \mathcal{N}(y_b, \sigma_{b,t}^2)$. The distribution of t_b can be written as

$$p(t_b) = g\left(t_b; y_b, \sigma_{b,t}^2\right) \tag{12}$$

The distribution information of the output t_b is obtained, and $100(1-\alpha)\%$ PIs can be constructed.

$$\begin{aligned} y_b^- &= y_b - z^{1-(\alpha/2)} \cdot \sigma_{b,t}, \\ y_b^+ &= y_b + z^{1-(\alpha/2)} \cdot \sigma_{b,t} \end{aligned} \tag{13}$$

where $z^{1-(\alpha/2)}$ is the $1-(\alpha/2)$-quantile of a standard normal distribution function with zero mean and unit variance. Introducing the contribution coefficient ω_b of the bth local network, the output PDF of the ensemble can be written as

$$\begin{aligned} y^- &= \sum_{b=1}^{B} \omega_b y_b^- = \sum_{b=1}^{B} \left(\omega_b y_b - z^{1-(\alpha/2)} \cdot \omega_b \cdot \sigma_{b,t}\right) \\ y^+ &= \sum_{b=1}^{B} \omega_b y_b^+ = \sum_{b=1}^{B} \left(\omega_b y_b + z^{1-(\alpha/2)} \cdot \omega_b \cdot \sigma_{b,t}\right) \end{aligned} \tag{14}$$

where $0 < \omega_b < 1$ and $\omega_1 + \omega_2 + \cdots + \omega_B = 1$.

The architecture of the distributed GGNNs has the following advantages. (1) The proposed architecture has a stronger learning ability than the neural network individual. Sometimes, the network individual might achieve a good performance of PIs, yet the performance is not very stable. However, the contribution of each local network in the proposed architecture is weighted through evaluating the individual performances. A more stable model can be achieved based on the proposed architecture. (2) Unlike the generic NNs ensemble, the relationship among the local networks of the proposed architecture is relatively independent. Therefore, the proposed architecture is suitable for distributed computing, and the computational efficiency of the ensemble can be significantly improved.

3. Parameters Estimation of the Distributed GGNNs

The parameters estimation process of the proposed NN ensemble is briefly illustrated in Figure 2. For a training dataset S with n_o data samples, this study first adopts a bootstrap method to resample the training dataset S, and then a bootstrap dataset S_b with n_b data samples are obtained, where $n_b \ll n_o$. Meanwhile, a corresponding dataset R_b is also generated by excluding the bootstrap samples from the original dataset. Repeating the same process B times, B bootstrap datasets can be obtained. In this study, the parameters estimation process of the proposed model consists of two parts. First, B local GGNNs are trained on the basis of the B bootstrap datasets, respectively. Second, the contribution coefficients are estimated by employing the B remaining datasets.

3.1. Parameters Estimation of Local GGNN. Given that the optimal weights are used for point-oriented prediction and the output variance are used for PIs construction, they have to be estimated for each of local GGNN. Since the optimal weights can be determined when the maximum of the posterior PDF occurs, one can optimize (7) to find the optimal weights of the bth local GGNN. Also, given that Z_M is a normalizing constant in (7), the optimal problem is equivalent to minimize the function $M(\mathbf{w}^b)$ according to the feature of the exponential function [10]. Then, we have

$$\begin{aligned} M\left(\mathbf{w}^b\right) &= \beta_b E_D + \alpha_b E_W \\ &= \frac{\beta_b}{2} \sum_{k=1}^{n_b} \left[y\left(\mathbf{w}^b, \mathbf{u}_k^b\right) - t_k\right]^2 + \frac{\alpha_b}{2} \left\|\mathbf{w}^b\right\|^2 \end{aligned} \tag{15}$$

To minimize $M(\mathbf{w}^b)$, (15) can be linearly approximated by Taylor expanding with respect to $\mathbf{w}_{MP}^b$. That is,

$$\begin{aligned} M\left(\mathbf{w}^b\right) &= M\left(\mathbf{w}_{MP}^b\right) \\ &\quad + \frac{1}{2}\left(\mathbf{w}^b - \mathbf{w}_{MP}^b\right)^T \mathbf{A}_b \left(\mathbf{w}^b - \mathbf{w}_{MP}^b\right) \end{aligned} \tag{16}$$

where $\mathbf{A}_b = \beta_b \nabla\nabla E_D^{MP} + \alpha_b \mathbf{I}$ is the Hessian of $M(\mathbf{w}^b)$.

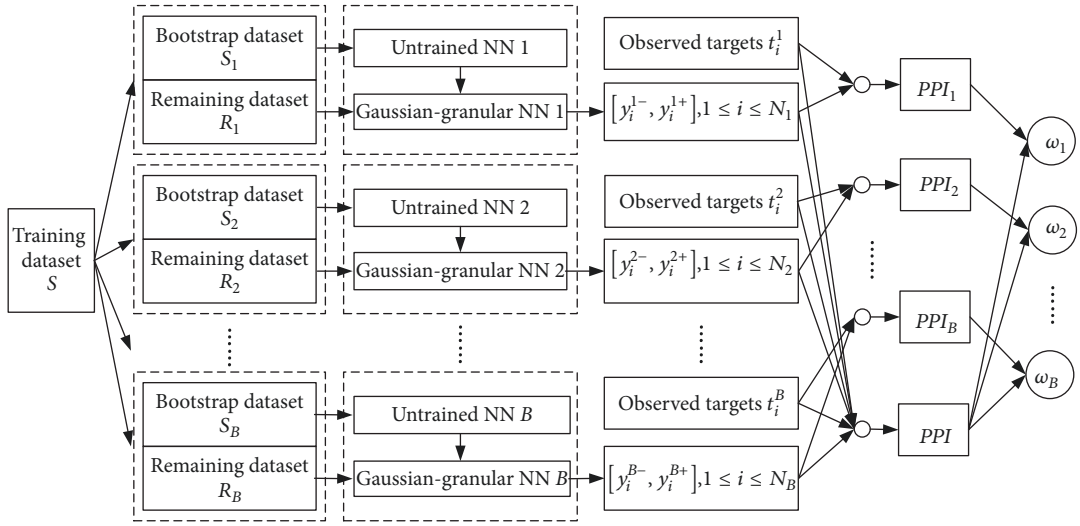

Figure 2: Training process of the proposed distributed networks.

As such, the prediction variance in (11) can be written as $\sigma_{t,b}^2 = 1/\beta_b + \mathbf{g}_b{}^T \mathbf{A}_b{}^{-1} \mathbf{g}_b$ and $\mathbf{g}_b = \nabla_{\mathbf{w}} y \mid \mathbf{w}_{MP}$. Since the variance is greatly related to α_b and β_b, it is necessary to simultaneously optimize the values of the hyperparameters and the network weights. According to [26], there exist relationships between $\mathbf{w}^b$ and the hyperparameters α_b and β_b, respectively.

$$\begin{aligned} \alpha_b^{new} &= \frac{\gamma_b}{2E_w}, \\ \beta_b^{new} &= \frac{(n-\gamma_b)}{2E_D} \end{aligned} \tag{17}$$

where $\gamma_b = \sum_{i=1}^{W} \lambda_i/(\lambda_i + \alpha_b)$, and λ_i are the eigenvalues of the Hessian matrix. Particularly, if the Hessian matrix $\mathbf{A}_b$ in (16) is singular, the eigenvalues should be approximated by using the singular value decomposition (SVD) algorithm. Since the prior knowledge is introduced for the Bayesian method, the dominator of γ_b is the sum of λ_i and α_b. Even λ_i might be equal to zero, the value of γ_b is still effective. Based on some experiments, the iterative strategy for the update of the hyperparameters is indeed feasible.

Based on the inference process from (3) to (11), α_b and β_b are two hyperparameters related to the distributions of the weights and the output, respectively. To optimize the cost function of (15), the extended Kalman filter (EKF) [27] is adopted here to optimize the weights of local networks, in which α_b and β_b can be viewed as the process noise and the measurement noise, respectively. We summarize the steps of the weights estimation based on the EKF as follows.

Step 1. Choose the initial values for α_b and β_b. Initialize the weights $\mathbf{w}_0^b$ and the corresponding variance $\mathbf{P}_{\mathbf{w}_0}$ by using the values drawn from the prior Gaussian distribution. Define the external loop number N_{loop}.

Step 2. For $k \in \{1, 2, \cdots, \infty\}$, the time update equations of the EKF are

$$\widehat{\mathbf{w}}_{k|k-1}^b = \mathbf{w}_{k-1|k-1}^b \tag{18}$$

$$\mathbf{P}_{\mathbf{w}_{k|k-1}}^b = \mathbf{A}_{k-1} \mathbf{P}_{\mathbf{w}_{k-1|k-1}}^b \mathbf{A}_{k-1}^T + \mathbf{R}_{\mathbf{v}}^b \tag{19}$$

where $\mathbf{A}_{k-1}$ is the identity matrix with the same dimensionality as the network weights and the variance of the process noise is defined as $(\mathbf{R}_{\mathbf{v}}^b)^{-1} = (\alpha_b/2) \cdot \mathbf{A}_{k-1}$.

Step 3. The measurement update equations are

$$\mathbf{K}_k^b = \mathbf{P}_{\mathbf{w}_{k|k-1}}^b \left(\mathbf{H}_k^b\right)^T \left(\mathbf{H}_k^b \mathbf{P}_{\mathbf{w}_{k|k-1}}^b \left(\mathbf{H}_k^b\right)^T + \mathbf{R}_n^b\right) \tag{20}$$

$$\widehat{\mathbf{w}}_{k|k}^b = \widehat{\mathbf{w}}_{k|k-1}^b + \mathbf{K}_k^b \left[\mathbf{t}_k - \mathbf{y}\left(\widehat{\mathbf{w}}_{k|k-1}^b, \mathbf{u}_k\right)\right] \tag{21}$$

$$\mathbf{P}_{\mathbf{w}_{k|k}}^b = \left(\mathbf{I} - \mathbf{K}_k^b \mathbf{H}_k^b\right) \mathbf{P}_{\mathbf{w}_{k|k-1}}^b \tag{22}$$

where $\mathbf{H}_k^b = \partial y(\mathbf{w}, \mathbf{u})/\partial \mathbf{w}|_{\widehat{\mathbf{w}}_{k|k}^b}$, $(\mathbf{R}_n^b)^{-1} = (\beta_b/2) \cdot \mathbf{B}_k$, and $\mathbf{B}_k$ is the identity matrix with the same dimensionality as the training samples.

Step 4. Update the hyperparameters α_b and β_b by (17).

Step 5. Iterate to Step 2 until the external loop N_{loop} reaches.

3.2. Coefficient of Local Gaussian-Granular NNs. Considering one bootstrap dataset S_b with n_b data samples, i.e., at least $n_0 - n_b$ samples in R_b, S_b will be used to train the bth local network, while R_b will be used to determine its contribution to the ensemble by evaluating the performance of PIs constructed by it. Generally, the performance of the PIs

can be quantified by the PIs coverage probability (PICP) and the mean PIs width (MPIW).

$$PICP = \sum_{i=1}^{N} \frac{c_i}{N} \tag{23}$$

$$MPIW = \sum_{i=1}^{N} \frac{\left(\widehat{y}_i^{+} - \widehat{y}_i^{-}\right)}{N} \tag{24}$$

where N is the number of the testing data samples. c_i is a binary-state variable. If the inequality $y_i^- \le t_i \le y_i^+$ is valid, then $c_i = 1$, else $c_i = 0$. Particularly, a coverage width-based criterion (CWC) for PIs evaluation that is a function of PICP and NMPIW is proposed in [1]. However, the CWC may not be suitable according to some experiments conducted in [17]. A very large value of CWC will be obtained when a not very large PICP is counted with a relatively low MPIW due to the amplification effect of the exponential function. On the contrary, the index CWC cannot achieve a large value, when the constructed PIs are much wider and the PICP is large.

The evaluation of the quality of the constructed PIs is a multiple objective optimization problem. It is difficult to define a universal and effective criterion for all prediction problems. Even for the same prediction problem, the expectations of PICP, MPIW, and MAE may vary under different application backgrounds. A more concise and effective evaluated strategy is designed as follows:

$$\begin{aligned} PPI &= \eta \cdot MPIW + \mu \cdot MAE + \gamma \\ &\quad \cdot (MPIW + MAE)(1 - PICP) \\ &= \left[\eta + \gamma \cdot (1 - PICP)\right] MPIW \\ &\quad + \left[\mu + \gamma \cdot (1 - PICP)\right] MAE \end{aligned} \tag{25}$$

where the constants η, μ, and γ are scale factors that subject to $\eta + \mu + \gamma = 1$. η, μ, and γ serve to qualify the amount of attention paid to MPIW, MAE, and PICP, respectively. The three parameters can be determined according to the application requirements. It is noteworthy that the evaluation index of accuracy named Mean Absolute Error (MAE) is also considered here since the point-oriented prediction is the median of the constructed PIs. The definition of *MAE* is given by

$$MAE = \sum_{i=1}^{N} \frac{\left|y_i - t_i\right|}{N} \tag{26}$$

where y_i is the predicted value and t_i is the observed value.

The detailed analysis of (25) is provided here. First, the values of the *MPIW* and the *MAE* are usually expected to be small; meanwhile, the *PICP* is expected to be large. According to (24) and (26), the measure indices *MPIW* and *MAE* with the same measurement dimension can be merged under the premise that the practical significance is not considered.

$$\begin{aligned} RMPIW &= MPIW + MAE \\ &= \sum_{i=1}^{N} \frac{\left[\left(y_i^{+} - y_i^{-}\right) + \left|y_i - t_i\right|\right]}{N} \end{aligned} \tag{27}$$

Second, the *PICP* is related to *MPIW* and *MAE* that is difficult to be quantified by any learning method. It is true that there is an inverse function relationship between *PICP* and *RMPIW*. Notably, *PICP* is in percentage. In order to establish the link between the two indices, it is only required to multiply (1-*PICP*) by *RMPIW*.

According to section (11), the output distribution of a trained GGNN can be written as follows given an input $\mathbf{u}_i^b$ of the remaining dataset R_b:

$$p\left(t_i^b \mid \mathbf{u}_i^b, S_b\right) = \frac{1}{\left(2\pi\sigma_{b,t}^2\right)^{1/2}} \exp\left[\frac{-\left(t_i^b - y_i^b\right)^2}{2\sigma_{b,t}^2}\right] \tag{28}$$

where y_i^b is the network output defined as $y_i^b = \widehat{\mathbf{W}}_b^{out}(\mathbf{u}_i^b, \mathbf{x}_i^b)$.

Then, the PIs can be constructed by the network output and the prediction variance in (28), formulated by

$$\begin{aligned} y_i^{b-} &= y_i^b - z^{1-(\alpha/2)}\sigma_{b,t}, \\ y_i^{b+} &= y_i^b + z^{1-(\alpha/2)}\sigma_{b,t} \end{aligned} \tag{29}$$

As such, the value of $PICP_b$ and $RMPIW_b$ can be calculated, and the value of the synthesized index PPI_b can also be obtained. To determine the contribution of each local network, the performance of the distributed networks can be presumably defined by the sum performance of the local networks.

$$PPI_s = \sum_{b \in B} PPI\left(RMPIW_b, PICP_b\right) \tag{30}$$

where $PICP_b$ is the PIs coverage probability of the bth local network and $RMPIW_b$ is the sum of the corresponding mean PIs width and root mean square error.

To make sure the contribution of the optimal local network can be enlarged and the contribution of the poor one can be minimized, an intermediate variable $\lambda_b = (PPI_b/PPI)^{\eta}$ is defined, where η is a scale factor. So the contribution of the bth local network to the distributed networks can be defined as

$$\omega_b = \frac{\lambda_b}{\sum_{b \in B} \lambda_b} \tag{31}$$

Here, η can be set from 5 to 20. The value of η controls the amount of attention paid to the optimal local network. The larger the value of η is, the larger the contribution coefficient ω of the optimal network is.

3.3. Map-Reduce Based Distributed Model. The advantage of the proposed NN ensemble is that its distributed structure allows an improved learning ability. If the performance of one local network is poor, that of the ensemble can still be guaranteed. However, there is a disadvantage of the proposed model that the computational efficiency is relatively low. Map-Reduce programming framework provides an efficient model that can be used to handle large datasets [28]. Map-Reduce can greatly improve the operation time [30].

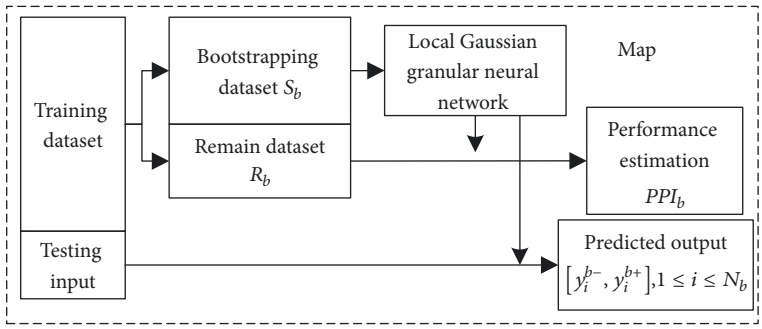

Figure 3: The processing procedure of the Map task.

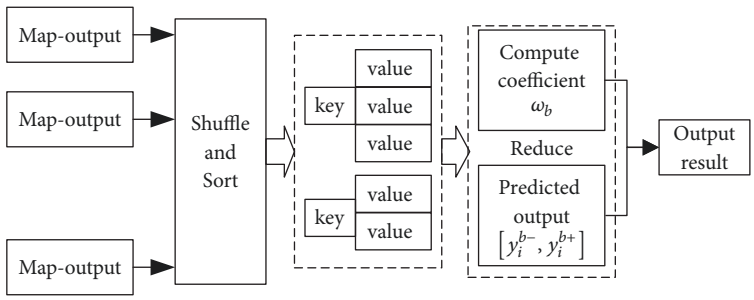

Figure 4: The architecture of the Map-Reduce programming model.

In this section, the proposed model is organized into a Map-Reduce programming architecture. The Map task is responsible for three important works, as shown in Figure 3. First, the local neural network is built and learned in each Map task based on the bootstrap dataset; second, the performance of each local network is evaluated based on the remaining dataset; and third, the prediction of each network is also completed given the testing input. The input of Map task is a pair of key-value, where the key labels the serial number of the local network and the value denotes the corresponding dataset for the local network including the training dataset and the testing one. The output of Map task is also a pair of key-value, where the output value is the evaluated performance and the predicted output of the local network.

The Reduce task is responsible for two parts, as illustrated in Figure 4. The first one computes the contribution coefficients. The second one generates the integrated output $[y_i^-, y_i^+]$, $1 \le i \le N$ of the distributed model with $[y_i^{b-}, y_i^{b+}]$, $1 \le i \le N_b$ of each local network. The input of the Reduce task is a pair of key-value from the output of the Map function, and its output is the final predicted output.

4. Experimental Results and Analysis

To verify the effectiveness of the proposed distributed model, four prediction tasks are considered. The first two ones are two classical prediction problems, involving the differential defined Rossler chaotic time series and the Mackey Glass chaotic time series. In order to exhibit the ability of practical application, we also consider two kinds of industrial validation, coming from a traffic time series provided by the website http://www.neural-forecasting-competition.com/datasets.htm and a real-world problem that involves real-time flow prediction for blast furnace gas in steel industry. The experiments are designed as follows that involves the parameters estimation, the PIs construction, and the performances evaluation. Since the generic NNs ensemble will cost a large computational load, the experiments based on MATLAB tools are carried out by using the ESNs ensemble whose parameters estimation is relatively efficient. And the generic NNs ensemble and the ESNs ensemble are employed to do a set of comparative experiments for a complete comparative study.

4.1. Rossler Chaotic Time Series. Rossler chaotic system, a kind of chaos phenomena of the lowest dimension equation, is proposed by Rossler in 1976 [32], which is described by three coupled first-order differential equations.

$$\begin{aligned} \frac{dx}{dt} &= -y - z, \\ \frac{dy}{dt} &= x + ay, \\ \frac{dz}{dt} &= b + z(x - c) \end{aligned} \tag{32}$$

Time series of all variables (x, y, z) are obtained from solving the above differential equations via 4^{th}-order Runge-Kutta method. The integral step of 4^{th}-order Runge-Kutta

Table 1: The validation result of the number of local networks.

B	*PICP*	*MPIW*	*MAE*	*PPI*	*T(s)*
1	0.5667	0.4785	0.3695	0.4131	5.24
2	0.6000	0.4490	0.2338	0.3079	10.33
3	0.8833	0.5074	0.1817	0.1893	15.38
4	0.8500	0.5012	0.1756	0.1986	21.14
5	0.8000	0.5196	0.1807	0.1904	25.63
6	0.8667	0.5145	0.1691	0.1880	30.73
7	0.8667	0.4927	0.1611	0.1855	35.82
8	0.9333	0.4856	0.1385	0.1590	41.05
9	0.9333	0.4981	0.1365	0.1535	46.35
10	0.9333	0.4935	0.1328	0.1528	50.94
11	*0.9667*	*0.5023*	*0.1356*	*0.1509*	*57.44*
12	0.9500	0.5284	0.1485	0.1560	62.65
13	0.9667	0.5143	0.1680	0.1533	67.53
14	0.9000	0.4976	0.1516	0.1558	72.88
15	0.9333	0.5061	0.1528	0.1576	79.61
16	0.9167	0.5101	0.1459	0.1660	84.01
17	0.9333	0.4956	0.1576	0.1617	90.44
18	0.9167	0.5064	0.1574	0.1588	94.85
19	0.9000	0.5135	0.1371	0.1524	101.9
20	0.9167	0.5035	0.1405	0.1630	109.1

method is 0.15, the previous 1000-step values are abandoned, and the following 9000-step values of all variables are adopted as the sample data where 1-5000 values are used for training and the remaining 4000-step values are taken to test and validate the performance of PIs. In this study, [a, b, c] are set as $[0.398, 2, 4]$, and the initial values of $[x, y, z]$ are set as [-1, 0, 1].

Take the x-variable for example, and the processes of constructing PIs for y-variable and z-variable are so similar that they are not given here. The desired noisy Rossler chaotic time series is obtained by adding a Gaussian white noise with the variance 0.01 to the original values of the time series. This study selects 2000 data samples from the training dataset by using the bootstrap resampling to build the bth local networks, whereas 500 data samples are randomly selected from the training dataset excluding the bootstrap samples to evaluate the bth local network. 60 data samples are selected from the testing dataset to verify the proposed model. In this study, the parameters are set for convenient calculation as follows. The number of the input units is set as 60 and the dimension of the dynamic reservoir is set as 200.

Table 1 lists the bootstrap validation results with respect to the different number of local networks for noisy Rossler prediction problem. In Table 1, three indices for evaluating the performance of the model are considered, where PICP, MPIW, and MAE pay attention to the coverage probability, the interval width, and the prediction accuracy, respectively. The penultimate column gives the validation results of a comprehensive index PPI that consider the PICP, MPIW, and MAE with weights 2/9, 1/9, and 2/3, respectively. The weights of PICP, MPIW, and MAE for PPI can be set by using a trial and error method. From Table 1, when the number of local networks equals 8, the performance of the ensemble is obviously better than that of the ensemble with less than 8 local networks. When the number of local networks equals 11, the performance of the ensemble is the best for the noisy Rossler problem.

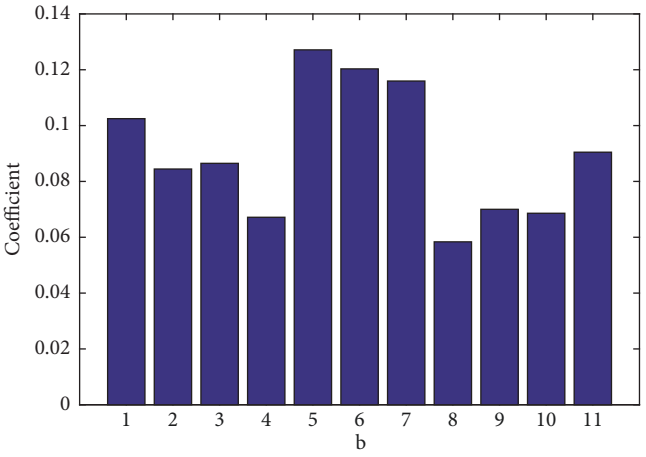

Figure 5: The histogram of the contribution coefficient of each local network.

Figure 5 shows the contribution coefficients of 11 local networks, where the scale factor η of (31) is set as 20. From this figure, the performance of each local network is significantly good and the contribution to the distributed networks shows no significant difference. The prediction results for the noisy Rossler problem based on the proposed distributed model with 11 local networks are shown in Figure 6. From

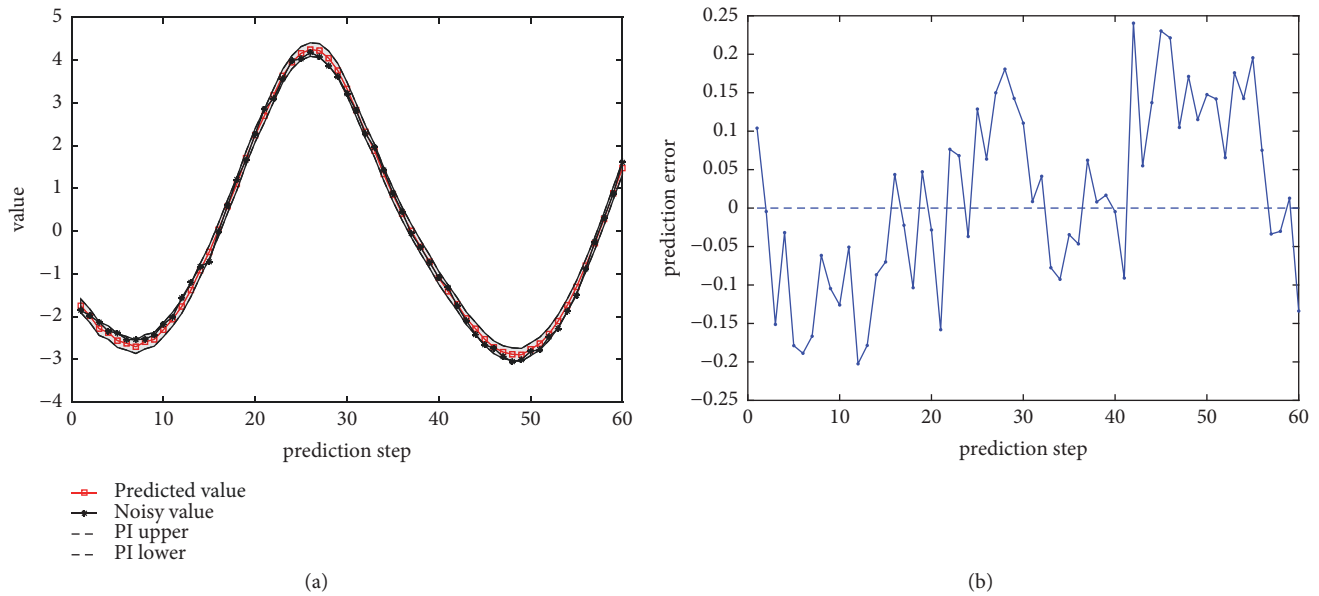

Figure 6: Prediction results for the noisy MSO problem: (a) the constructed PIs; (b) the corresponding predicted error.

Figure 6(a), the constructed PIs cover the observed value with a lower width, and the smaller predicted error in Figure 6(b) illustrates the prediction accuracy of the proposed model for the noisy Rossler time series.

4.2. Mackey Glass Chaotic Time Series. Mackey Glass system [33] is a time-delay differential system formulated as

$$\frac{dx(t)}{dt} = \frac{ax(t-\tau)}{1+x(t-\tau)^{10}} - bx(t) \tag{33}$$

where the parameters are typically set as $a = 0.2$, $b = -0.1$, and $\tau = 17$. To sample the time series, we numerically integrate (33) by using the fourth-order Runge-Kutta method with a sampling period of 2s and an initial condition $x(0) = 1.2$. The desired noisy Mackey Glass time series is obtained by adding a Gaussian white noise with the variance 0.01 to the original Mackey Glass time series. The original dataset sampled from the noisy time series consists of 2000 data samples, where 1500 data samples are used for training and the others for testing. We select 1000 data samples from the training dataset by using the bootstrap resampling method to build the bth local network, whereas 500 data samples are randomly selected from the training dataset excluding the bootstrap samples to evaluate the bth local network. 60 data samples are selected from the testing dataset to verify the proposed model. In addition, the parameters of the local network are estimated based on the training data. The number of the input units is set as 60 and the dimension of the dynamic reservoir is set as 200.

Mackey Glass time series is also generated from a chaotic system, yet the range of values is narrower than that of the Rossler time series. Compared to Table 1, the forecasting performance of the proposed ensemble becomes much better; especially, the values of PICP are significantly improved. The penultimate column in Table 2 gives the validation results of PPI with the consideration of the PICP, MPIW, and MAE whose weights are 2/9, 1/9, and 2/3, respectively. The weights of PICP, MPIW, and MAE for PPI can be set by using a trial and error method. From this table, the values of PPI of different ensembles are not greatly varied with the increase of the number of the local networks. However, it is apparently that when the number of local networks is larger than 9, the performance of the ensemble is significantly improved. And when the number of local networks equals 16, the performance of the distributed ensemble is the best for the noisy Mackey Glass problem.

Figure 7 shows the contribution coefficients of 8 local networks where the scale factor η of (31) is set as 20. The prediction results for the noisy Mackey Glass problem based on the proposed distributed model with 8 local networks are shown in Figure 8. From Figure 8(a), the constructed PIs with the confidence level 95%, in which the noisy value can almost be covered by the PIs, and the predicted error in Figure 8(b) illustrate the high accuracy of the proposed ensemble for the noisy Mackey Glass problem.

4.3. Application to Prediction of Traffic Time Series. In this subsection, the data are provided by the NN GC1 Forecasting Competition that can be found on website. The samples are constructed on the basis of the data of Tournament 1, 1.F. The original dataset consists of 1500 data samples, where 1000 data samples are used for training and the others for testing. We select 500 data samples from the training dataset by using the bootstrap resampling method to build the bth local network, whereas 100 data samples are randomly selected from the training dataset excluding the bootstrap samples to evaluate the bth local network. 60 data samples are selected from the testing dataset to verify the proposed model. In addition, the parameters of the local network are estimated based on the training data. In this study, the number of the input units is

Table 2: The validation result of the number of local networks.

B	*PICP*	*MPIW*	*MAE*	*PPI*	*T(s)*
1	0.9167	0.4261	0.1191	0.1338	5.46
2	0.9333	0.4168	0.1093	0.1282	11.34
3	0.9667	0.4164	0.1039	0.1201	15.96
4	0.9500	0.4283	0.1096	0.1248	20.53
5	0.9389	0.4045	0.1059	0.1239	25.19
6	0.9500	0.4180	0.1033	0.1217	33.04
7	0.9333	0.4155	0.1127	0.1291	37.82
8	0.9444	0.4175	0.1154	0.1292	42.37
9	0.9666	0.4325	0.1027	0.1140	47.32
10	0.9667	0.4212	0.1077	0.1173	52.54
11	0.9667	0.4153	0.1060	0.1153	56.25
12	0.9445	0.4174	0.1062	0.1178	61.78
13	0.9555	0.4124	0.1056	0.1141	67.58
14	0.9667	0.4174	0.1010	0.1146	73.41
15	0.9500	0.4229	0.1092	0.1167	79.41
16	*0.9722*	*0.4151*	*0.1129*	*0.1116*	*82.42*
17	0.9667	0.4250	0.1008	0.1159	87.53
18	0.9667	0.4098	0.0967	0.1122	92.82
19	0.9722	0.4148	0.1073	0.1131	99.04
20	0.9833	0.4128	0.1056	0.1141	102.8

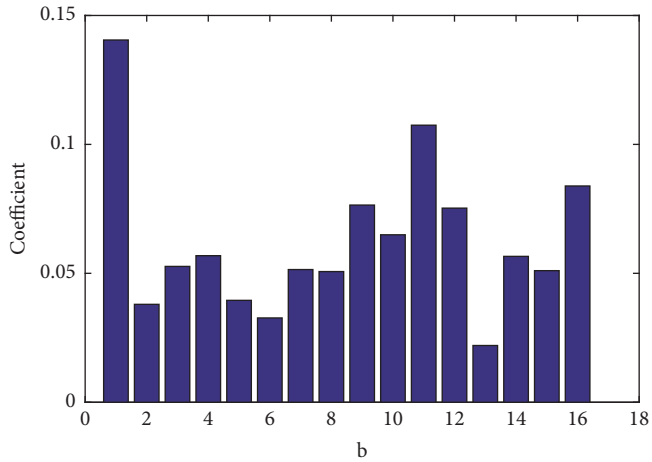

Figure 7: The histogram of the contribution coefficient of each local network.

set as 50 and the dimension of the dynamic reservoir is set as 20.

The bootstrap validation results of the number of local networks are shown in Table 3. First, although the PIs are the widest when there is only one local network, the corresponding PICP is not the highest. Due to the increase of the prediction difficulty, there exists a large difference in the prediction performance between two networks. Sometimes, the prediction performance of one local network is bad for the traffic time series. When the number of local networks increases, the performance of the proposed distributed networks can be improved because the good performance is amplified and meanwhile the poor performance is punished in this study. The above inference is verified to a certain extent in Figure 9. In modeling the distributed networks, there exist some local neural networks with poor performances as the 1th, 3th, 4th, 5th, 7th, 9th, 10th, and 11th local networks shown in Figure 9. Only the 2th, 6th, 8th, and 12th local networks make a great contribution to the output of the distributed networks, especially the second local network. The penultimate column in Table 3 gives the validation results of PPI with the consideration of PICP, MPIW, and MAE whose weights are 1/3, 1/3, and 1/3, respectively. The weights of PICP, MPIW, and MAE for PPI can be set by using a trial and error method. From Table 3, when the number of local networks equals 12, the distributed networks perform best.

The prediction results based on the proposed distributed model with 12 local networks is shown in Figure 10. From

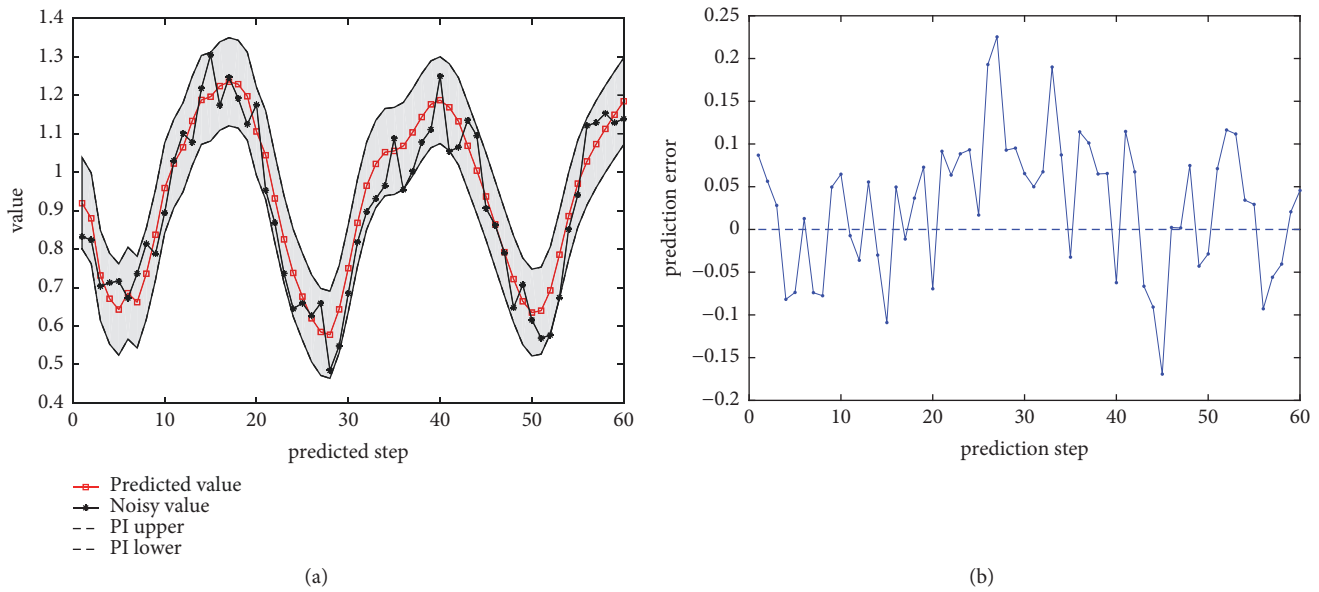

Figure 8: Prediction results for noisy Mackey Glass problem: (a) the constructed PIs; (b) the corresponding predicted error.

Table 3: The validation result of the number of local networks.

B	*PICP*	*MPIW*	*MAE*	*PPI*	*T(s)*
1	0.6533	7509.348	4199.105	9627.972	0.53
2	0.7317	3679.751	2496.305	2640.191	0.93
3	0.73	3313.649	2247.185	2402.14	1.12
4	0.77	3569.069	1966.393	2270.854	1.34
5	0.7717	3715.51	2051.193	2343.564	1.69
6	0.7234	3697.235	1892.151	2296.226	1.89
7	0.755	3356.219	2052.896	2257.913	2.21
8	0.8133	4079.909	1988.189	2384.771	2.45
9	0.745	3211.481	2317.851	2427.852	2.69
10	0.7733	3000.334	1799.689	1960.604	2.95
11	0.8317	3579.286	1759.333	2068.051	3.28
12	*0.7833*	*3046.309*	*1752.181*	*1943.746*	*3.54*
13	0.8133	3543.992	1815.015	2062.386	3.79
14	0.785	3122.935	1740.262	1961.455	4.08
15	0.8083	3430.32	1906.759	2085.079	4.34
16	0.7866	3024.173	1817.909	1987.508	5.64
17	0.7916	3217.54	1885.096	2084.68	5.75
18	0.8071	3325.56	1746.16	2030.345	5.95
19	0.795	3258.99	1786.836	2039.716	6.32
20	0.8242	3181.69	1850.961	2073.198	6.61

Figure 10(a), the constructed PIs with the confidence level 95%, in which the noisy value can almost be covered by the PIs, and the predicted error in Figure 10(b) conforms the accuracy of the proposed model. The prediction results in Figure 10 are different from those of the first two prediction problems. The numerical range of the value is large and the dynamic characteristics of the data are more complex. Since the uncertainties of the data and model are larger, the PIs are much wider. Figure 10 illustrates that the proposed model is effective for constructing the PIs of the traffic time series.

4.4. Application to the Consumption Flow of Blast Furnace Gas. In this section, we present a real-world industrial application problem by using the proposed distributed model. Steel industry is usually accompanied with high energy consumption and environmental pollution, in which the byproduct gas is one of the useful energy resources. There is a need to predict the gas consumption flow for energy scheduling in a steel plant. The experimental data, with a sample interval of 1 min, come from a steel plant in China and cover the consumption flow of BFG by the #1, #2 Coke Oven in September 2015. The

Table 4: The validation result of the number of local networks.

B	*PICP*	*MPIW*	*MAE*	*PPI*	*T (s)*
1	0.8333	4.4338	1.8133	1.8809	2.46
2	0.8333	4.4931	1.6129	1.8561	4.19
3	0.8333	4.4535	1.6979	1.8618	5.61
4	0.8333	4.4429	1.5997	1.8365	6.62
5	0.8333	4.4614	1.6273	1.8488	8.06
6	0.8500	4.3609	1.5767	1.7381	10.05
7	0.8500	4.4735	1.6104	1.7814	11.28
8	0.8667	4.4594	1.5856	1.7045	12.68
9	*0.8667*	*4.2224*	*1.6232*	*1.7032*	*14.31*
10	0.8500	4.4399	1.6018	1.7688	15.84
11	0.8500	4.4073	1.6018	1.7583	17.54
12	0.8500	4.4396	1.6307	1.7409	19.21
13	0.8500	4.3062	1.6412	1.7340	20.98
14	0.8500	4.3559	1.6400	1.7842	22.57
15	0.8500	4.3597	1.6188	1.7739	24.21
16	0.8500	4.3136	1.6419	1.7366	25.36
17	0.8500	4.4593	1.6439	1.7839	27.40
18	0.8500	4.4745	1.6396	1.7879	29.15
19	0.8500	4.4033	1.6404	1.7651	31.28
20	0.8500	4.4270	1.6377	1.7722	32.13

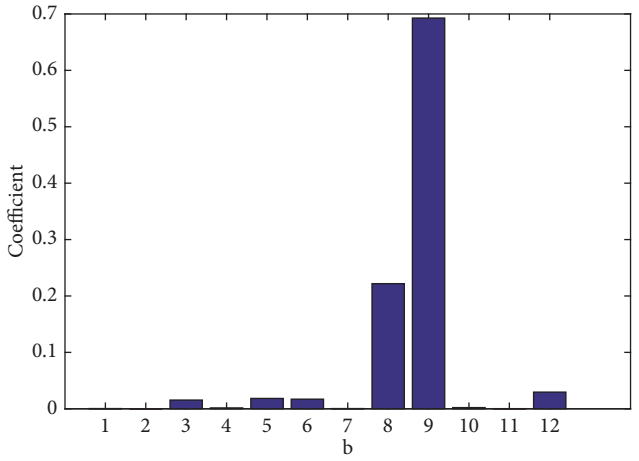

Figure 9: The histogram of the contribution coefficient of each local network.

original dataset consists of 4000 data samples, where 3000 data samples are used for training and the others for testing. We select 1000 data samples from the training dataset by using the bootstrap resampling method to build the bth local network, whereas 100 data samples are randomly selected from the training dataset excluding the bootstrap samples to evaluate the bth local network. 60 data samples are selected from the testing dataset to verify the proposed model. Again, the number of the input units is set as 50 and the dimension of the dynamic reservoir is set as 100.

Table 4 shows the bootstrap validation results of the number of local networks for the consumption flow of BFG. From Table 4, when the number of local networks equals to 9, the performance of the distributed networks is the best for the consumption flow. Figure 11 shows the contribution coefficients of 9 local networks, where the scale factor η of (31) is set as 20. The prediction results for the consumption flow prediction problem based on the proposed distributed model with 9 local networks are shown in Figure 12(a). The constructed PIs with a confidence level of 95%, in which the observed value can almost be covered by the PIs, and the predicted error in Figure 12(b) illustrate the high accuracy of the proposed model for the consumption flow prediction problem.

4.5. Statistical Analysis and Distributed Computation. To further evaluate the performance, a series of statistical experimental results are reported in Table 5, where NN-based conformal prediction [29], the MVE-based NN individual [1], the Bayesian NN individual [14], and the networks ensemble in [22] are conducted for comparison. To guarantee the full indication of the statistical experiments, the PICP, MPIW, MAE, PPI, and computing time are employed here for statistics. The mean values of the PICP, MPIW, MAE, PPI, and computing time for 50 times are given in Table 5. In addition, to verify the performance of the proposed method, the ESN individuals-based ensemble and the MLP individuals-based ensemble that is the generic NNs ensemble are considered for the comparative experiments.

From Table 5, we observe that the proposed distributed NNs ensemble shows the best performance with the overall consideration of *PICP*, *MPIW,* and *MAE*. Although conformal prediction has a stable and good performance for the coverage probability of PIs, it comes with the tradeoff in MPIW. Although ESN individual has a lower computational

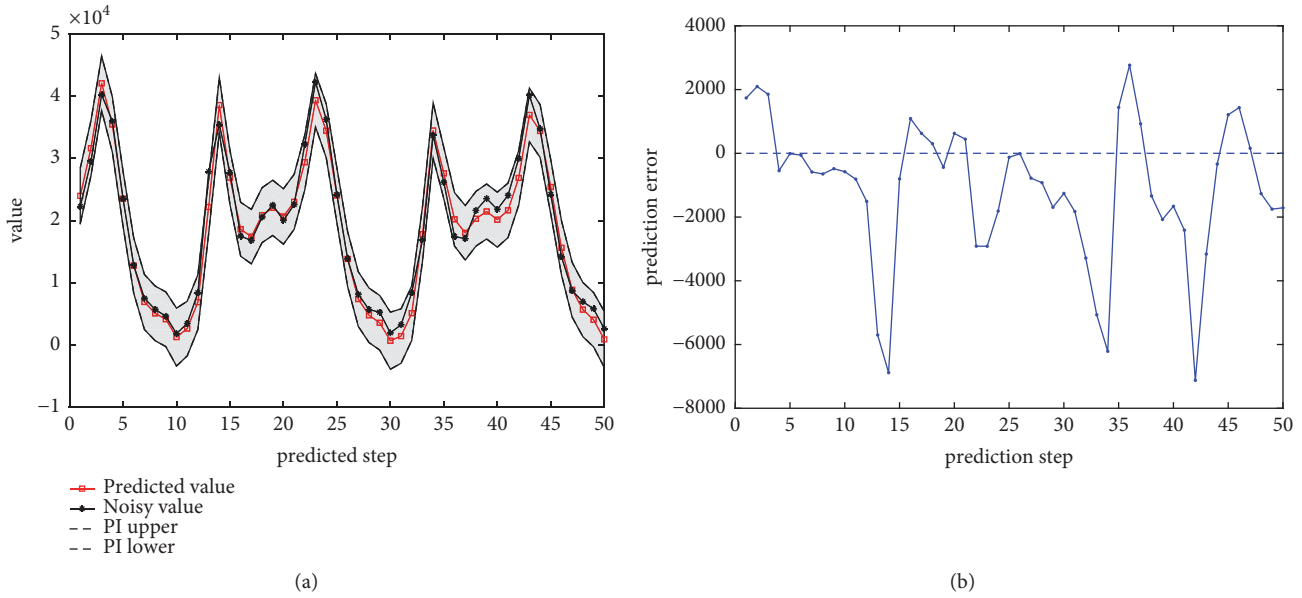

Figure 10: Prediction results for the data of Tournament 1: (a) the constructed PIs; (b) the corresponding prediction error.

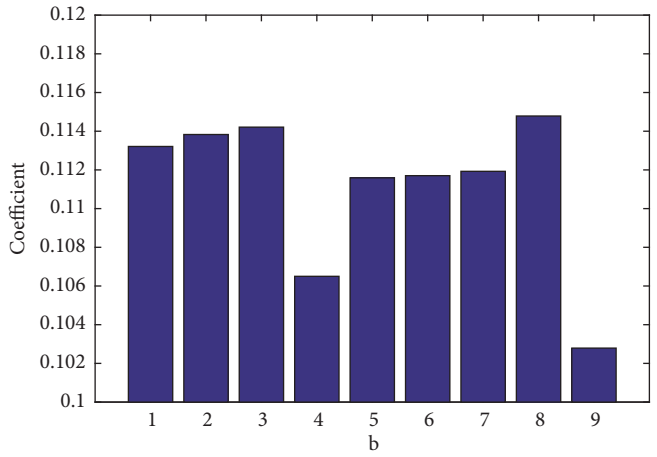

Figure 11: The histogram of the contribution coefficient of each local network.

load and a higher accuracy, it is often accompanied with the instable performance since the coefficient matrix of unknown parameters might be ambiguous, which makes the distributed ESNs ensemble do not show the performance as well as the generic NNs ensemble. To improve the accuracy of the ESNs ensemble, an effective parameters estimation strategy is required, but with a large computational load. By contraries, MLP individual has a stable and excellent prediction performance but with a higher computational cost. Thus, the proposed model inhibits a better prediction performance in total if the computational efficiency can be improved.

It is well known that the distributed NNs ensemble implemented in serial mode costs more computing time than the NN individual. In Table 5, the computational time is monitored on the MATLAB platform. If the algorithm is implemented on another platform in series mode, it might cost a higher computational load. Fortunately, the proposed method can be implemented in parallel mode in the JAVA environment with the Map-Reduce programming model that can effectively improve the computational efficiency. The experiments are conducted based on the Baidu open cloud platform. The master node consists of Intel Xeon 2.00-GHz 4 Core, 32-GB memory, and 400G Hard disk, and the slave nodes are the same as the master one. The experimental results are shown in Figure 13 and Table 6. From Figures 13(a) and 13(b), when the distributed model consists of more local networks, the computational efficiency can be improved with an increase in the number of the local networks. The measurement unit of the value in Table 6 is minute. The parameters of the distributed networks are set as the above statements. From Table 6, we can see that the computing time of the distributed model can be greatly reduced with the increase of the number in the slave nodes. When the number

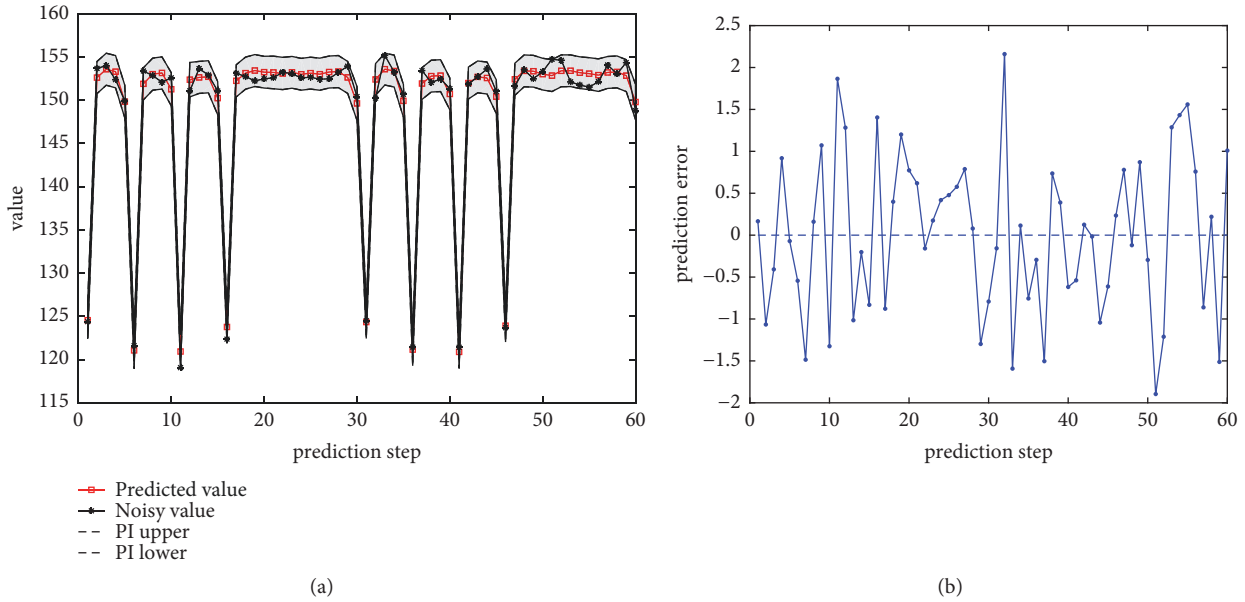

Figure 12: Prediction results for the consumption flow of BFG: (a) the constructed PIs; (b) the corresponding prediction error.

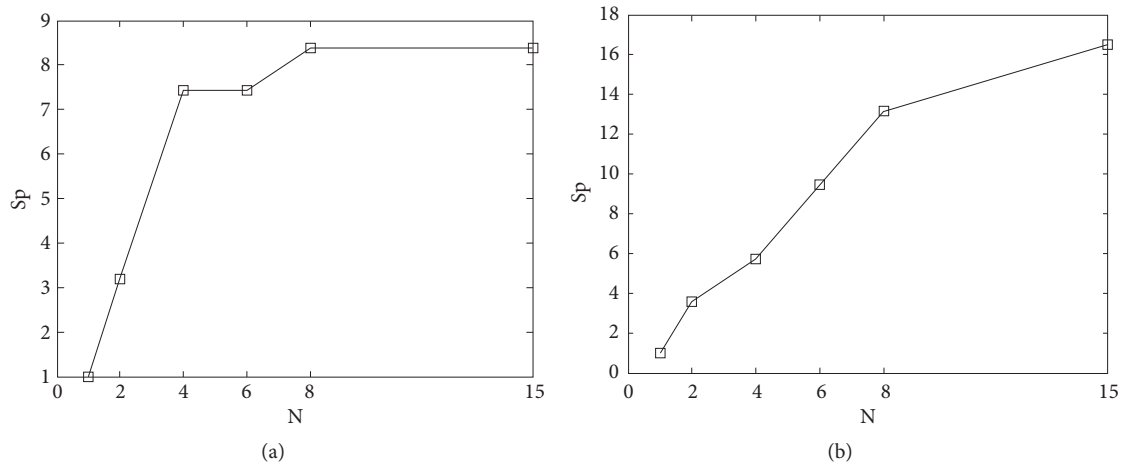

Figure 13: The speedup chart of the Map-Reduce programming model: (a) the ensemble with 8 local NNs; (b) the ensemble with 15 local NNs.

of the local GGNNs equals 8, the computational efficiency can be obviously improved when using one master and four slaves. As for the distributed model with 15 local GGNNs, the computational efficiency can be improved with one master and eight slaves. For the two groups of experiments, the shortest computing time of each experiment only requires 8 minutes.

5. Conclusions

In this paper, a novel distributed GGNNs ensemble is proposed for PIs construction. The distributed model has B cascaded GGNNs and the output of the distributed model is the weighted sum of the output of the local neural networks. The advantages of the proposed method involve three important respects. (1) The distributed model has a stronger learning ability than the neural network individual. (2) The performance of the proposed model is more stable. The effect due to local network with poor performance can be reduced when computing the coefficient of the local one contributed to the distributed model, and on the contrary, the good performance of the local networks can be amplified. The proposed distributed model is different from the generic NNs ensemble by taking the contribution of the network individual into consideration. (3) The relationship among the local networks of the proposed architecture is relatively independent. Therefore, the proposed architecture is suitable for implementing on a distributed programming platform named Map-Reduce architecture to improve the computational efficiency. Four prediction problems are used in our experiments to illustrate the effectiveness of the proposed method for PIs construction. Compared to three network

Table 5: Statistical analysis of the comparative experiments.

Problems	Model	*PICP*	*MPIW*	*MAE*	PPI_M	PPI_{SD}	*T(s)*
Noisy Rossler problem	Conformal prediction in [29]	0.9500	0.5758	0.1742	0.1947	0.0319	0.7623
	MVE-based NN individual	0.8333	0.8917	2.6027	1.4426	0.22	*0.5387*
	Bayesian NN	0.9500	1.0688	0.3243	1.0688	0.1216	0.7615
	NNs ensemble in [22]	*0.9833*	1.2238	0.1843	1.2238	0.3881	50.48
	Distributed ESNs ensemble	0.9667	0.6723	0.1506	0.1732	0.0077	57.44
	Distributed NNs ensemble	0.9667	*0.5023*	*0.1356*	*0.1509*	*0.0067*	1474.14
Noisy Mackey Glass	Conformal prediction in [29]	0.9500	0.4538	0.1149	0.1162	0.0297	0.7623
	MVE-based NN individual	0.9077	0.6482	0.4811	0.3417	0.1921	*0.5387*
	Bayesian NN	0.9310	0.5084	0.1179	0.1412	0.0385	0.7615
	NNs ensemble in [22]	*0.9828*	0.5363	0.1178	0.1388	0.0501	50.48
	Distributed ESNs ensemble	0.9527	0.4323	0.1145	0.1211	0.0125	82.42
	Distributed NNs ensemble	0.9722	*0.4151*	*0.1129*	*0.1116*	*0.0113*	1035.49
Traffic time series	Conformal prediction in [29]	0.9500	12487.00	4135.78	5394.57	295.1779	0.7623
	MVE-based NN individual	*0.9962*	21674.00	30325.00	17267.00	1282.2291	*0.5387*
	Bayesian NN	0.8958	12057	4182.77	5789.36	307.6441	0.7615
	NNs ensemble in [22]	0.7615	6525.37	3694.27	4986.57	374.1657	50.48
	Distributed ESNs ensemble	0.7952	4674.72	2705.11	2493.67	168.1368	3.54
	Distributed NNs ensemble	0.7833	*3046.309*	*1752.181*	*1943.746*	*129.8434*	1108.09
BFG consumption flow	Conformal prediction in [29]	*0.9500*	7.6245	1.6385	1.7123	0.1265	0.7623
	MVE-based NN individual	0.8077	5.2250	1.9899	2.1357	0.2401	*0.5387*
	Bayesian NN	0.9167	5.0133	1.7477	1.7046	0.1073	0.7615
	NNs ensemble in [22]	0.8563	4.5125	1.7989	1.9625	0.1758	50.48
	Distributed ESNs ensemble	0.8500	4.3422	1.7285	1.7039	0.0708	14.31
	Distributed NNs ensemble	0.8667	*4.2224*	*1.6232*	*1.7032*	*0.0674*	828.01

Table 6: Comparison of the computational efficiency.

B	Serial	2 slaves	4 slaves	6 slaves	8 slaves	15 slaves
8	67	21	9	9	8	8
15	132	37	23	14	10	8

individual-based methods and the generic NNs ensemble, the proposed model is shown to have the best performance. In addition, it shows an improved computational efficiency by the aid of the Map-Reduce programming framework.

In this study, the contribution of individual to the ensemble is quantified by evaluating the performance of the individual with (25) proposed in this paper. However, to implement the distributed computation of the proposed ensemble, the cooperation among different individuals is not given enough attention. The learning process of the individuals of an ensemble is relatively independent. In the

future, we will pay more attention to the research on the cooperative learning of the individuals in an ensemble to achieve a stronger learning ability. For instance, the fuzzy integration or some constructive algorithms will be considered in the future. Meanwhile, we are expected to focus on the distributed implementation of the cooperative learning method in the future and obtain some valuable results. In addition, model-free-based PIs will be considered, which can be viewed as another future work. If the PIs are applied for industrial process, the computational load is one significant index to be solved. For the model-based PIs, especially the ensemble or modular NNs-based PIs, it costs much more computational load despite high accuracy, which affects the application of the PIs.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (61773245, 61603068, and 61806113), Shandong Province Natural Science Foundation (ZR2018ZC0436, ZR2018PF011, and ZR2018BF020), Key Research and Development Program of Shandong Province (2018GGX101053), China Postdoctoral Science Foundation (2016T90640), Scientific Research Foundation of Shandong University of Science and Technology for Recruited Talents (2017RCJJ061, 2017RCJJ062, and 2017RCJJ063), and Taishan Scholarship Construction Engineering.

References

[1] A. Khosravi, S. Nahavandi, D. Creighton, and A. F. Atiya, "Comprehensive review of neural network-based prediction intervals and new advances," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 22, no. 9, pp. 1341–1356, 2011.

[2] T. Heskes, "Practical confidence and prediction intervals," in *Advances in Neural Information Processing Systems 9*, pp. 176–182, MIT Press, 1997.

[3] D. Sáez, F. Ávila, D. Olivares, C. Cañizares, and L. Marín, "Fuzzy prediction interval models for forecasting renewable resources and loads in microgrids," *IEEE Transactions on Smart Grid*, vol. 6, no. 2, pp. 548–556, 2015.

[4] K. Seok, C. Hwang, and D. Cho, "Prediction intervals for support vector machine regression," *Communications in Statistics—Theory and Methods*, vol. 31, no. 10, pp. 1887–1898, 2002.

[5] K. De Brabanter, J. De Brabanter, J. A. K. Suykens, and B. De Moor, "Approximate confidence and prediction intervals for least squares support vector regression," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 22, no. 1, pp. 110–120, 2011.

[6] G. Chryssolouris, M. Lee, and A. Ramsey, "Confidence interval prediction for neural network models," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 7, no. 1, pp. 229–232, 1996.

[7] J. T. Hwang and A. A. Ding, "Prediction intervals for artificial neural networks," *Journal of the American Statistical Association*, vol. 92, no. 438, pp. 748–757, 1997.

[8] G. Papadopoulos, P. Edwards, and A. Murray, "Confidence estimation methods for neural networks: a practical comparison," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 12, no. 6, pp. 1278–1287, 2001.

[9] D. A. Nix and A. S. Weigend, "Estimating the mean and variance of the target probability distribution," in *Proceedings of the International Conference on Neural Networks (ICNN' 95)*, vol. 1, pp. 55–60, Orlando, Fla, USA, 1994.

[10] C. M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, New York, NY, USA, 1995.

[11] R. Tibshirani, "A comparison of some error estimates for neural network models," *Neural Computation*, vol. 8, no. 1, pp. 152–163, 1996.

[12] P. W. Goldberg, C. K. I. Williams, and C. M. Bishop, "Regression with input-dependent noise a Gaussian process treatment," in *Proceedings of the 11th Annual Conference on Neural Information Processing Systems, NIPS 1997*, pp. 493–499, USA, December 1997.

[13] W. Wright, "Bayesian approach to neural-network modeling with input uncertainty," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 10, no. 6, pp. 1261–1270, 1999.

[14] C. M. Bishop and C. S. Qazaz, "Regression with input-dependet noise: A bayesian treatment," in *Proceedings of the 10th Annual Conference on Neural Information Processing Systems, NIPS 1996*, vol. 9, pp. 347–353, USA, December 1996.

[15] M. G. C. A. Cimino, B. Lazzerini, F. Marcelloni, and W. Pedrycz, "Genetic interval neural networks for granular data regression," *Information Sciences*, vol. 257, pp. 313–330, 2014.

[16] M. Song and W. Pedrycz, "Granular neural networks: concepts and development schemes," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 4, pp. 542–553, 2013.

[17] A. Khosravi, S. Nahavandi, D. Creighton, and A. F. Atiya, "Lower upper bound estimation method for construction of neural network-based prediction intervals," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 22, no. 3, pp. 337–346, 2011.

[18] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 10, pp. 993–1001, 1990.

[19] R. Tuvsgurani, "A comparison of some error estimates for neural network models," *Neural Computation*, vol. 8, pp. 152–163, 1996.

[20] M. Islam, X. Yao, and K. Murase, "A constructive algorithm for training cooperative neural network ensembles," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 14, no. 4, pp. 820–834, 2003.

[21] Y. Liu and X. Yao, "Simultaneous training of negatively correlated neural networks in an ensemble," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 29, no. 6, pp. 716–725, 1999.

[22] C. Sheng, J. Zhao, W. Wang et al., "Prediction intervals for a noisy nonlinear time series based on a bootstrapping reservoir computing network ensemble," *IEEE Transactions on Neural*

Networks and Learning Systems, vol. 24, no. 7, pp. 1036–1048, 2013.

[23] M. I. Ribeiro, *Gaussian Probability Density Functions: Properties and Error Characterization*, Institute for Systems and Robotics, Lisbon, Portugal, 2004.

[24] H. Jaeger and H. Haas, "Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication," *Science*, vol. 304, no. 5667, pp. 78–80, 2004.

[25] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1–3, pp. 489–501, 2006.

[26] C. M. Bishop, *Neural Networks for Pattern Recognition*, Clarendon Press, Oxford, UK, 1995.

[27] H. Simon, *Kalman Filtering and Neural Networks*, Wiley, New York, NY, USA, 2001.

[28] I. Triguero, D. Peralta, J. Bacardit, S. García, and F. Herrera, "MRPR: a MapReduce solution for prototype reduction in big data classification," *Neurocomputing*, vol. 150, pp. 331–345, 2015.

[29] H. Papadopoulos and H. Haralambous, "Reliable prediction intervals with regression neural networks," *Neural Networks*, vol. 24, no. 8, pp. 842–851, 2011.

[30] N. K. Alham, M. Li, Y. Liu, and M. Qi, "A MapReduce-based distributed SVM ensemble for scalable image classification and annotation," *Computers & Mathematics with Applications*, vol. 66, no. 10, pp. 1920–1934, 2013.

[31] Y. Xue, L. Yang, and S. Haykin, "Decoupled echo state networks with lateral inhibition," *Neural Networks*, vol. 20, no. 3, pp. 365–376, 2007.

[32] O. E. Rössler, "An equation for continuous chaos," *Physics Letters A*, vol. 57, no. 5, pp. 397-398, 1976.

[33] M. C. Mackey and L. Glass, "Oscillation and chaos in physiological control systems," *Science*, vol. 197, no. 4300, pp. 287–289, 1977.