

A Noisy Data Regression Model Based on General Regression Neural Networks

Shih-Chun Shao Wen-Hui Chen

Graduate Institute of Automation Technology
National Taipei University of Technology
Taipei, Taiwan
t8618007@ntut.edu.tw whchen@ntut.edu.tw

Jun-Horng Chen

Department of Communication Engineering
Oriental Institute of Technology
New Taipei, Taiwan
jhchen@mail.oit.edu.tw

Abstract—Analysis of noisy data gathered from measurement devices is challenging in the power grid. In this study, an effective noisy data regression approach based on general regression neural networks (GRNN) is employed to deal with the problem for remote terminal units (RTU) in power SCADA systems. Experimental results show the proposed model is able to handle noisy data for practical applications, and has good performance in removing the unintended changes to the original data.

Keywords— *general regression neural networks; genetic algorithms; remote terminal units; power SCADA systems*

I. INTRODUCTION

Supervisory control and data acquisition (SCADA) systems are indispensable in the operation of modern power systems. A SCADA system is a centralized control and monitoring system that typically consists of a master station, communication networks, and remote terminal units (RTUs). In power SCADA systems, all the field data are collected by RTUs, and transmit to the host computer in the control center based on some protocol through communication networks. Data acquisition at the RTU includes meter readings and equipment status reports. In the data transmission process, there will be a chance that parts of the collected data are corrupted due to noise interference. Past research indicated the processing of the noisy data at RTU is necessary in SCADA systems [1]-[2].

In the data transmission process, there are different data processing stages between field sites and the control center, including raw data acquisition, transformation, conversion, and transmission in various devices. Spikes or power surges can cause data corruption in the data transmission process, which makes the acquired data become unreliable. For data integrity, it is essential to process the raw data in the substation level before transmitting them to the host computer in the control center. How to handle the corrupted data for improving information accuracy becomes essential as it can have an impact on the system operation and can influence dispatchers during the decision-making process.

To take the data corruption issue into consideration, the converted data in the RTU needs to be processed using data processing techniques before converting back to practical engineering values in the host computer. For improving the data accuracy, in [2], the authors applied fuzzy logic theory to

deal with the problem of data uncertainty. In the application of fuzzy logic theory, the selection of membership functions is a key issue, and is time-consuming by try-and-error process. As inappropriate membership functions can cause bad results. The authors in [3] applied genetic algorithms to adjust the required parameters of the membership function to improve the estimation results. However, it is not easy to obtain the optimal membership function using few sample data.

In this study, a computational approach based on general regression neural networks (GRNN) was proposed to construct a systematic fitting model for RTU noisy data processing. The proposed approach is employed to examine the analogue data collected by RTUs as a data preprocessing step to reduce errors existing in the conversion process before running the required data analysis.

The rest of the paper is organized as follows. Section II introduces the proposed GRNN model for dealing with the noisy data. Section III provides an example to verify the effectiveness of the proposed GRNN model. Conclusions are finally drawn in section IV.

II. METHODOLOGY

A. GRNN Architecture

The GRNN is a powerful regression tool proposed in [4] as an alternative neural network based learning paradigm. The network training speed of GRNN is extremely fast as it adopts a one-pass learning algorithm with a highly parallel structure. In addition, the parameter that needs to be determined in GRNN is much fewer than the widely used back-propagation neural networks (BPNN). Due to the simplicity of network structure and ease of implementation, GRNN has been applied to a variety of applications including noisy data removal [5], function approximation [6], active vibration control [7], and fault diagnosis [8].

In GRNN, each training sample is operated as a kernel during the training process. The regression surface is established by using Parzen window estimator. The estimation of GRNN is based on non-parametric regression analysis to make the best fit of the observed data. As such, GRNN does not require prior knowledge of the regression function. These

features are useful for obtaining a better estimation of noisy data in the problem of RTU data processing.

Assume that the underlying scalar function to be approximated is denoted as $y=f(\mathbf{x})$ where $\mathbf{x} \in R^m$ is an independent variable and $y \in R^1$ is a dependent variable. Regression in the GRNN is carried out by calculating the expected conditional mean of y given $\mathbf{x}$ as shown in (1). The Parzen probability density estimator $g(\mathbf{x}, y)$ with Gaussian function defined in (2) is based on sample values $\mathbf{x}_i$ and y_i of the random variable $\mathbf{x}$ and y , where n is the number of training samples and m is the dimension of $\mathbf{x}$.

$$E[y | \mathbf{x}] = \frac{\int_{-\infty}^{\infty} yg(\mathbf{x}, y)dy}{\int_{-\infty}^{\infty} g(\mathbf{x}, y)dy} \quad (1)$$

$$g(\mathbf{x}, y) = \frac{1}{(2\pi)^{(m+1)/2} \sigma^{(m+1)}} \cdot \frac{1}{n} \sum_{i=1}^n \exp\left[-\frac{(\mathbf{x} - \mathbf{x}_i)^T (\mathbf{x} - \mathbf{x}_i)}{2\sigma^2}\right] \cdot \exp\left[-\frac{(y - y_i)^2}{2\sigma^2}\right] \quad (2)$$

The probability estimator $g(\mathbf{x}, y)$ assigns the sample probability of width σ for each sample $\mathbf{x}_i$ and y_i , and the probability estimate is the sum of those sample probabilities. The non-negative constant σ called smoothing parameter or spread constant is the only parameter that needs to be determined in the GRNN-based approach, and is related to how much the neural network will generalize. Therefore, the performance of the GRNN is affected by the adjustment of spread constants.

When the spread constant is made large, the estimation is forced to be smooth but it does not represent the training samples very precisely. On the other hand, a small value of spread constant allows the estimation that represents the training samples very well but it has many wiggles in the fitting curve. Therefore, a tradeoff between smoothness and minimal error has to be made between precision and smoothness.

Substituting the joint probability estimator in (2) into the conditional mean in (1) yields the desired estimated value as follows,

$$\hat{y}(\mathbf{x}) = \frac{\sum_{i=1}^n y_i \exp(-\frac{D_i^2}{2\sigma^2})}{\sum_{i=1}^n \exp(-\frac{D_i^2}{2\sigma^2})} \quad (3)$$

where D_i is defined as

$$D_i^2 = (\mathbf{x} - \mathbf{x}_i)^T (\mathbf{x} - \mathbf{x}_i) \quad (4)$$

The distance, D_i , between the training sample and the point of estimation is used as a measure of how well each training sample can represent the position of estimation. With (3) it is possible to estimate values of the fitting function based on training samples.

B. Noisy Data Regression Using GRNNs

The use of GRNN to build a regression model for noisy data from RTUs is described in this subsection. A feed-forward neural network, as shown in Fig. 1, is used to estimate the regression variable y from a measurement vector $\mathbf{x}$.

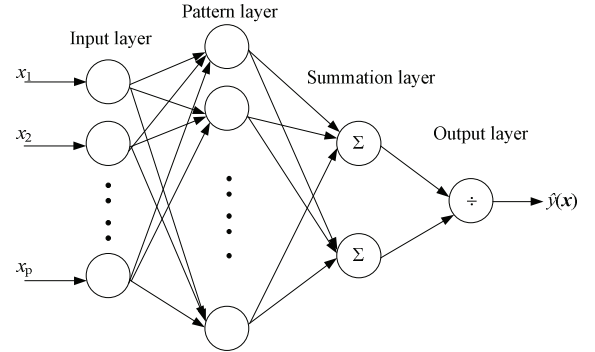


Fig. 1 Schematic diagram of the GRNN architecture.

There are four different types of layers in the structure of the GRNN model: input layer, pattern layer, summation layer, and the output layer. The input neurons in the input layer are merely distribution units, which provide all of the measurement variables to the second layer, the pattern layer. The pattern layer is a latent regression layer. Each neuron in the pattern layer is corresponding to a training sample or one cluster center. The calculations performed in each pattern neuron are the normal distribution centered at each training sample.

The summation layer performs a dot product between a weight vector and a vector composed of the signals from the pattern layer. It includes two units. One is the denominator summation unit, and the other is the numerator summation unit. The first unit adds up the weight values coming from each of the hidden neurons. The numerator summation unit adds up the weight values multiplied by the actual target value for each hidden neuron.

The output layer generates the desired estimation of y , denoted by $\hat{y}$. It divides the value accumulated in the numerator summation unit by the value in the denominator summation unit, and uses the result as the estimated value. This estimated value is an exponentially weighted average of all observed values y_i .

The training of GRNN does not need to determine the connection weights between the neurons in different layers but to adjust the spread constant to make the estimation model best fit the sample data. In other words, the generalization ability of GRNN is controlled by this spread constant. Traditionally, the value of spread constant of GRNN is usually selected by a try-and-error process. In the GRNN-GA model,

the genetic algorithm is used to optimize the spread constant of GRNN.

Genetic algorithms (GA) are population-based search algorithms based on the mechanics of natural selection and genetics. GA is able to globally exploring a solution space searching for potential solutions, and has been applied to solve many optimization problems [9]-[11].

The process of GA begins with a randomly selected population of chromosomes. Each chromosome string contains possible solution to the spread constant of GRNN. GA is used for searching an appropriate spread constant that can minimize the mean square errors between the target values and training samples. The fitness function f is defined as follows,

$$f = \frac{\sum_{i=1}^n [y_i - \hat{y}(x_i)]^2}{n} \quad (5)$$

where n is the number of samples.

The optimum solution can be obtained by minimizing the fitness function defined in (5). The fitness function is used to evaluate the quality of each chromosome. The lower fitness value chromosome at each searching step will be chosen to be the parent of the next generation, while the higher fitness one will be discarded. In creating the new offspring, the selected parent chromosomes will get through the reproduction process by performing GA operations, such as crossover and mutation as described in [12]. The process continues updating newer and more suitable chromosome until a predefined stopping condition is satisfied.

III. EXPERIMENTAL RESULTS

The performance of the proposed GRNN model was evaluated by a set of RTU data and is described in this section. The optimal spread constant of the GRNN model is determined according to the mean square error (MSE) criterion using GA searching technique.

The program codes in this experiment are written in MATLAB language for all the simulations. The performance evaluation used in this study is the mean absolute percentage error (MAPE) in accuracy. This value is calculated by first taking the absolute deviation between the actual value y_i and the estimated value $\hat{y}_i$. Then the total ratio of deviation with its actual value is calculated. The percentage of the average of this ratio is the mean absolute percentage error. The process of calculating the MAPE is shown in (6).

$$\text{MAPE} = \frac{\sum_{i=1}^n (|y_i - \hat{y}_i|/y_i)}{n} \times 100\% \quad (6)$$

where n is the total number of test data.

To examine the performance of the proposed GRNN model, a comparison with an existing method was given. The test data

was adopted from [3]. Suppose that a bus voltage recorded from a substation varies from 180 kV to 260 kV (nominal 220 kV with 18% variations). The voltage values are converted to the value in pulse counts, ranging from 19800 to 28600 (110 counts per kV). Table I lists the test examples and their corresponding calibrated values. In Table I, the measurands in the second column are measurements that contain errors, which are transformed to the pulse output via A/D converters in terms of counts. Actual values in the fourth column are calibrated quantities without errors.

TABLE I
TEST EXAMPLES AND THEIR CORRESPONDING CALIBRATED VALUES

No. of samples	Measurand (kV)	Counts	Actual values (kV)
1	180	19800	180.911
2	190	20900	190.0015
3	200	22000	200.00
4	210	23100	210.0005
5	220	24200	220.00
6	230	25300	230.0005
7	240	26400	240.2563
8	250	27500	250.00
9	260	28600	259.6281

In this experiment, the parameters of GA are initialized with a population of 40 chromosomes, and the crossover probability and mutation probability are taken as 0.85 and 0.01, respectively. Fig. 2 depicts the relation between the number of iterations and the corresponding spread constants. The MSE is plotted versus the number of iterations in Fig. 3. The optimal spread constant was set at 0.6647 to leverage the smooth of fitting function and the mean square error.

The performance evaluation of the Fuzzy-GA (FGA) model proposed in [3] and the proposed GRNN model is listed in Table II. From the results, it can be seen that the performance of the proposed model outperforms that of the FGA model. These results justify the proposed approach is able to approximate the fitting function more accurate than that in [3].

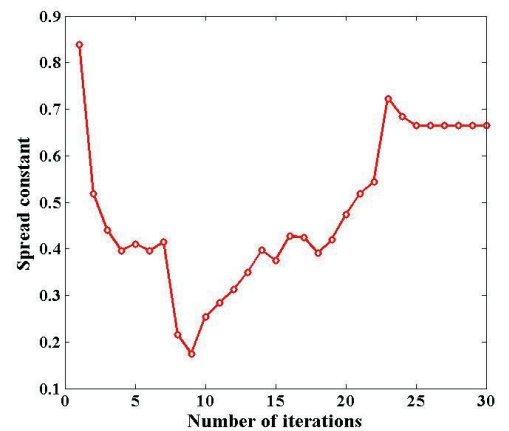


Fig. 2 The iteration-spread constant graph.

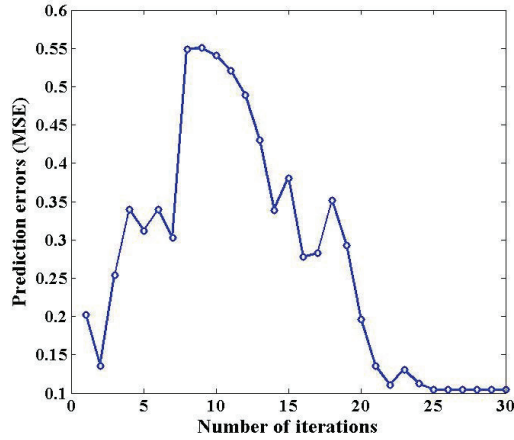


Fig. 3 The MSE-spread constant graph.

TABLE II
PERFORMANCE EVALUATION BETWEEN GRNN AND FGA MODELS

Counts	Actual value (kV)	FGA model (kV)	The proposed GRNN model (kV)
19800	180.911	181.822	182.5051
20900	190.0015	190.003	190.1614
22000	200.00	200.000	200.0015
23100	210.0005	210.001	210.0004
24200	220.00	220.000	220.0005
25300	230.0005	230.001	230.0379
26400	240.2563	240.512	240.1800
27500	250.00	250.00	249.9565
28600	259.6281	249.243	257.9410
	MAPE	0.5124	0.1868

IV. CONCLUSION

This paper has presented an efficient noisy data regression model for the RTU application using GRNN approach. The main merits of the proposed method include its simple computational framework, precision in nonlinear approximation, and fast learning speed. Although we only provide the voltage parameter for demonstration, the proposed approach can also be applied to other analogue quantities in the measurement of remote terminal units.

ACKNOWLEDGMENT

This work was partially supported by the projects sponsored by the National Science Council, Taiwan, under the Grant No. 98-2221-E-027-072 and 99-2218-E-002-005.

REFERENCES

- [1] A. G. Bruce, "Reliability analysis of electric utility SCADA systems," *IEEE Trans. Power Systems*, vol. 13, no. 3, pp. 844-849, Aug. 1998.

- [2] P. Kumar, V. K. Chandna, and M. S. Thomas, "Intelligent algorithm for preprocessing multiple data at RTU," *IEEE Trans. Power Systems*, vol. 18, no. 4, pp.1566-1572, Nov. 2003.
- [3] P. Kumar, V. K. Chandna, and M. S. Thomas, "Fuzzy-genetic algorithm for preprocessing data at the RTU," *IEEE trans. Power Systems*, vol. 19, no.2, pp.718-723, May 2004.
- [4] D. F. Specht, "A general regression neural network," *IEEE Trans. Neural networks*, vol. 2, no. 6, pp. 568-576, Nov. 1991.
- [5] W. M. Lee, C. P. Lim, K. K. Yuen, and S. M. Lo, "A hybrid neural network model for noisy data regression," *IEEE Trans. Systems, Man, and Cybernetics-Part B*, vol. 34, no. 2, pp. 951-960, April 2004.
- [6] J. Y. Goulermas, P. Liatsis, X. J. Zeng, and P. Cook, "Density-driven generalized regression neural networks (DD-GRNN) for function approximation," *IEEE Trans. Neural networks*, vol. 18, no. 6, pp.1683-1696, Nov. 2007.
- [7] A. Madkour, M. A. Hossain, K. P. Dahal, and H. Yu, "Intelligent learning algorithms for active vibration control," *IEEE Trans. Systems, Man, and Cybernetics-Part C*, vol. 37, no. 5, pp.1022-1033, Sept. 2007.
- [8] B. G. Hyun and K. Nam, "Faults diagnoses of rotating machines by using neural nets: GRNN and BPN," in *Proc. IEEE 21st Int. IECON Conf. Industrial Electronics, Control, Instrumentation*, vol. 2, 1995, pp.1456-1461.
- [9] Sangameswar Venkatraman and Gary Yen, "A genetic framework for constrained optimization using genetic algorithm," *IEEE Trans. Evolutionary Computation*, vol. 9, no.4, pp. 424-435, August 2005.
- [10] Zne-Jung Lee, Shun-Feng Su, and Chou-Yuan Lee, "A genetic algorithm with domain knowledge for weapon-target assignment problems," *Journal of the Chinese Institute of Engineers*, vol. 25, no. 3, pp. 287-295, 2002.
- [11] Shu-Chien Huang and Chu-Fu Wang, "Genetic algorithms for approximation of digital curves with line segments and circular arcs," *Journal of the Chinese Institute of Engineers*, Vol. 32, no. 4, pp. 437-444, 2009.
- [12] D. E. Goldberg, *Genetic algorithms in search, optimization, and machine learning*, Addison-Wesley Inc., 1989.