International Journal of Grid and High Performance Computing
Volume 14 • Issue 1

Nonlinear System Identification Based on an Online SCFNN With Applications in IoTs

Ye Lin, Zhejiang Industry and Trade Vocational College, China*
Yea-Shuan Huang, Chung Hua University, Taiwan
Rui-Chang Lin, Guangzhou Panyu Polytechnic, China

ABSTRACT

In this paper, an online self-constructing fuzzy neural network (SCFNN) is proposed to solve four kinds of nonlinear dynamic system identification (NDSI) problems in the internet of things (IoTs). The SCFNN is capable of constructing a simple network without the need for knowledge of the NDSI. Thus, carefully setting conditions for the increased demands for fuzzy rules will make the architecture of the constructed SCFNN fairly simple. The applications of neural networks in IoTs are discussed. The authors also propose a new identification model for NDSI. Through an experimental example, it is proved that online learning can arrange membership functions in a more appropriate vector space. The performance of the online SCFNN is compared with both MLP and RBF through four extensive simulations. The comparison terms are convergence rate, training root mean square error (RMSE), test RMSE, and prediction accuracy (PA). The simulation results show that SCFNN is superior to MLP and RBF in NDSI problems.

KEYWORDS

Internet of Things (IoTs), Multi-Layer Perceptron (MLP), Nonlinear Dynamic System Identification (NDSI), Online Learning, Radial Basis Function (RBF), Self-Construction Fuzzy Neural Network (SCFNN)

1. INTRODUCTION

In recent years, the concept of Internet of things put forward on the basis of the Internet has been widely concerned, and the applications of intelligent Internet of things are increasing, such as smart community, smart home, smart car, intelligent green house, etc. (Xia et al., 2012). These applications mainly receive and analyze data in real time through sensors and other devices and feedback the results to the control system, so as to achieve the intelligent effect, and the actual system modeling is an important part. System identification theory for linear systems has been well-established and many applications of system identification have been reported. On the other hand, system identification theory for nonlinear systems has not been established so systematically, because of its extremely wide scope (Adachi et al., 2004). Neural network provides a new way for

DOI: 10.4018/IJGHPC.316153 *Corresponding Author

the modeling of unknown nonlinear dynamic systems. As long as the input and output of a dynamic system in the Internet of things are measurable, it can be identified online by neural network. In recent years, there has been many research on modeling, prediction, and control of the Internet of things using neural networks. Patra. J. C. proposed a neural network-based interface framework to automatically compensate for the nonlinear influence of the environmental temperature and the nonlinear-response characteristics of a capacitive pressure sensor to provide correct read out (2005). In the study of Manonmani, a neural network was used to model and control sufficient growth conditions of a greenhouse system (GHS) resulting in high cross yield, advanced production period, better quality, and less use of protective chemicals (2016). In the study of Hamid Taghavifar, the potential of a supervised artificial neural network (ANN) approach was assessed to diagnose the energy consumption and environmental indexes of application production in the learning location (2015).

The research of neural network in nonlinear dynamic system identification is as follows. In the past decade, due to the ability of learning on the basis of appropriate error function optimization and the good performance of approximation of nonlinear function (Antsaklis, 1992), ANN based on different examples (MLP, RBF, etc.) have been widely used as powerful learning tools for complex system identification and control tasks. In 1990, Narendra and Artasarathy proposed effective identification and control of nonlinear dynamic systems using MLP (1990). Because of its simple structure, RBF (Haykin, 2008; Pislaru & Shebani, 2014) is considered as an alternative to MLP. At the same time, in order to obtain better performance, some algorithms (such as BP) are used to train the role of network-based neural networks in complex power system/plant modeling and control (Narendra & Parthasarathy, 1990; Park & Sandberg, 1991; Parlos et al., 1994). Although studies have shown that these networks with corresponding algorithms can effectively identify and control complex process dynamics, they can achieve better performance (Zhao & Zhang, 2009) on the premise of increasing computational complexity. In general, the training of MLP and RBF is usually based on the back propagation (BP) or gradient descent (GD) training algorithm with fixed learning rate, but it is difficult to find the optimal learning rate in BP or GD algorithm. Basically, if the selected learning rate is very small, then the convergence speed of the network is very slow, and it takes a long time to converge. On the other hand, the high learning rate will lead to unstable learning process and network dispersion (Cao & Lin, 2008; Yoo et al., 2006). There are also other novel designs of FNN, such as self-organizing fuzzy neural networks (SOFNN) (Han et al., 2017) and self-organizing deep belief networks (SODBN) (Qiao et al., 2018), which are capable of constructing a simple fuzzy network without expert knowledge. A new growing and pruning algorithm was proposed in (Han et al., 2010) which is named as self-organizing radial basis function (SORBF) for RBFNN.

However, the existing ANN training algorithms mainly use the so-called offline mode. There are many problems with offline training. Firstly, if the characteristics of the system/plant are time-varying, the input and output data collected by a single time cannot accurately capture the information of the device. Second, if system/plant characteristics do not change over time, random sampling is not the most convincing. In short, off-line training cannot effectively process sample data reflecting changes in system/plant characteristics. In order to improve the role of neural networks in the identification of nonlinear systems, it is necessary to further study the online training of neural network. This means that the collection of sample data, the construction of network and the training of network are all carried out simultaneously, so that there is no time isolation.

In this article, a new NDSI identification model is proposed, and membership functions of hidden nodes generated by on-line and off-line training neural networks are analyzed. The authors find that the distribution of online learning membership functions can effectively cover the vector space of input nodes, while the distribution of offline learning membership functions can't effectively cover the vector space of input nodes. Through simulation, it is found that the online training neural network has better recognition accuracy than the off-line training. The authors also conduct extensive

simulation to compare the performance of the proposed SCFNN with MLP and RBF in convergence speed, root mean square error and prediction accuracy. The simulation results show that SCFNN is superior to MLP and RBF in NDSI problems.

The rest of this paper is organized as follows. Section 2 briefly introduces SCFNN. Section 3 introduces four kinds of non-linear dynamic system models and proposes a novel non-linear dynamic system model. In this section, the difference between on-line training and off-line training is also discussed through an NDSI example. Section 4 gives the simulation results of four NDSI examples by using SCFNN, MLP and RBF neural networks respectively. Finally, Section 5 concludes this work.

2. A BRIEF DESCRIPTION TO SCFNN AND IOT

2.1. SCFNN

The structure of SCFNN is shown in Fig. 1. In order to limit the length of this article, the authors neglected the detailed learning processes of SCFNN. Readers interested in SCFNN can refer to (Lin et al., 2001; Weng et al., 2005).

In general, SCFNN is capable of constructing a simple FNN without the need for knowledge of the universe of discourse of each input variable. This is because the SCFNN can self-adjust the position of the fuzzy partition of the input space, and there is no need to estimate the number and position of input states in advance. Thus, carefully setting conditions for the increased demand for fuzzy rules makes the architecture of the constructed SCFNN fairly simple.

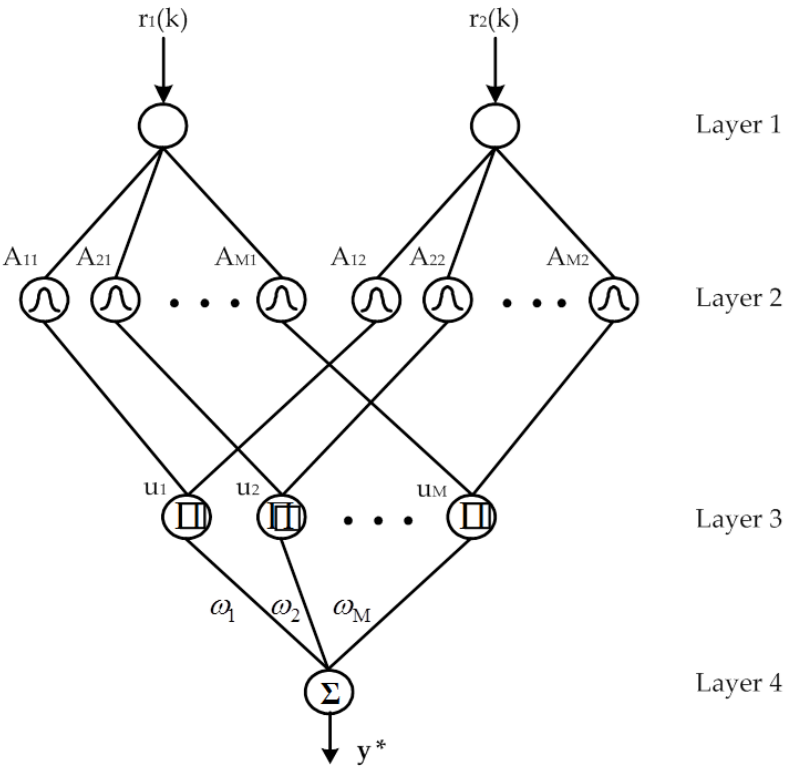
Two types of learning algorithms, namely, structure learning and parameter learning algorithms, are needed to construct SCFNN. Based on the concept of minimizing the number of generation rules, structure learning is used to find appropriate input space fuzzy partitions and fuzzy logic rules. The central task of the parameter learning is to obtain the adaptive rules that can be used to adjust the parameters of the network based on a given set of input-output pairs. If the parameters of the network are considered as elements of a parameter vector, then the learning process involves determining the vector which minimizes a given energy function.

The gradient of the energy function with respect to the vector is computed, and the vector is adjusted along the negative gradient. This method is generally referred to as the back-propagation (BP) learning rule because the gradient vector is calculated in the direction opposite to the flow of the output of each node.

2.2. IOT

Internet of things refers to the ubiquitous end devices and facilities, including sensors, industrial systems, building control systems and home intelligent facilities with internal intelligence. Externally enabled, for example, intelligent objects or animals or intelligent dust such as RFID pasted assets, individuals and vehicles with wireless terminals, etc., realize interconnection through long-distance and short-distance communication networks of various wireless wires, provide safe, controllable and personalized real-time online monitoring, dispatching, command, remote control, remote maintenance and other management and service functions. Intelligent control of objects, the Internet of things based on cloud computing platform and intelligent network can make decisions according to the data obtained by sensor network, and change the behavior of objects for control and feedback, for example, adjusting the brightness of street lights according to the intensity of light and automatically adjusting the interval of traffic lights according to the flow of vehicles. The Internet of things has a wide range of applications. Facing complex and diverse non-linear systems, fuzzy neural networks have superior discrimination, recognition and control capabilities, making them suitable for becoming part of the Internet of things.

Figure 1. Schematic structure of SCFNN



3. COMPARISON OF SCFNN OPERATING IN NDSI ONLINE AND OFFLINE TRAININGS

3.1. Traditional Nonlinear Dynamical System Model

Many identification schemes can only work if the identified system has a good approximate mathematical model. For any such scheme, the quality of the system model obtained by the identification program must meet certain standards. Therefore, the system identification program must be carefully selected. The identification process includes selecting the model structure of a given system and approximating its order, then building the same structure for the FNN model to approximate the unknown dynamics of a given system. There are various models of nonlinear systems, and the characteristics of different models are quite different. Each model can only be effective for a specific type of system, and it cannot be universally applied. At present, there are four classical nonlinear models: Hammerstein model, Wiener model, Hammerstein Wiener model, and NARMAX model. A gross nonlinear system in IoTs may belong to one of the four models introduced by the following nonlinear difference equations (Kumar et al., 2017; Narendra & Parthasarathy, 1992):

- **Model 1:**

$$y_p\left(n+1\right)=\sum_{i=0}^{k-1}a_j y_p\left(n-i\right)+g\left[u\left(n\right),u\left(n-1\right)\ldots u\left(n-m+1\right)\right] \tag{1}$$

- **Model 2:**

$$y_p\left(n+1\right)=f\left[y_p\left(n\right),y_p\left(n-1\right)\ldots y_p\left(n-k+1\right)\right]+\sum_{i=0}^{m-1}\beta_i u\left(n-i\right) \tag{2}$$

- **Model 3:**

$$y_p\left(n+1\right)=f\left[y_p\left(n\right),y_p\left(n-1\right)\ldots y_p\left(n-k+1\right)\right]+g\left[u\left(n\right),u\left(n-1\right)\ldots u\left(n-m+1\right)\right] \tag{3}$$

- **Model 4:**

$$y_p\left(n+1\right)=f\left[y_p\left(n\right),y_p\left(n-1\right)\ldots y_p\left(n-k+1\right),u\left(n\right),u\left(n-1\right)\ldots u\left(n-m+1\right)\right] \tag{4}$$

The models shown above are used to represent discrete-time systems. Symbol *n* represents the order of the system, and *m* £ *n*. Also f and g represents the nonlinear functions. Refer to Fig. 2 and Fig. 3 for the meaning of the remaining symbols. All four models represent the power systems, where (*n*+1)th time plant output depends on the present value of plant output and external input and its past value (Kumar et al., 2017; Narendra & Parthasarathy, 1990). The identification process requires knowledge of the following two aspects:

1. The model to which the given system belongs.
2. The estimate of plant's order (value of *n*).

FNN will be used to approximate the nonlinearity in factories. The problem of identification is to establish an appropriate parametric identification model, and then to adjust the relevant parameters according to the instantaneous mean square error between the output of the system and the output of the identification model. The structure of the selected recognition model is the same as the mathematical structure of plants. But there are a few things, as shown in the following, must be kept in mind to ensure that the identification process results in the convergence of the identification model parameters to their expected values (Kumar et al., 2017; Wei & Liu, 2015).

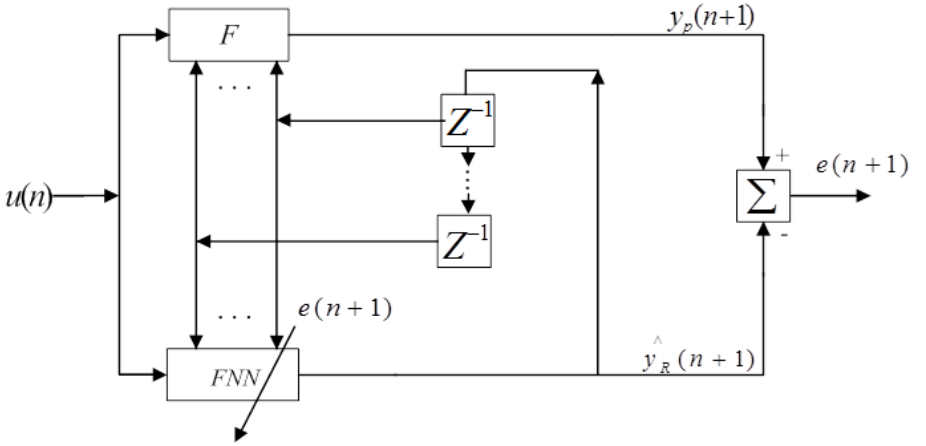
3.2. A New Series-Parallel Identification Model

In this work, the authors designed a new identification model (Fig. 2). In this model both the plant and the FNN use the current and previous output values of FNN along with external input *u*(*n*) to compute the next output. So what the neural network learns is the right input-output mapping relationship. Using this new identification model, both the input and desired output sequence of learning samples cannot be prepared in advance. They can only be gradually acquired with the increase of time t in the learning process. So online training is needed in the proposed new model.

3.3. Behavior Understanding of the Proposed Series-Parallel Model With one NDSI Example

A system belongs to model 1 mentioned in section 3.1 is learned by online and offline SCFNN, the governing equation of the system is given by (Kumar et al., 2017; Narendra & Parthasarathy, 1990)

Figure 2. A new series-parallel identification model



$$y(n+1)=0.3y(n)+0.6y(n-1)+\frac{0.6}{1+u(n)^2} \tag{5}$$

where the output at time (n+1) is a linear function of past output at times (n) and (n–1) plus a nonlinear function of the input at time (n). The reference input $u(n)$ to the system is selected as $u(n) = sin(2\pi n/100)$. To show the robustness of the proposed structure to the variations in the amplitude and frequency of the input, an input with 50% reduction in the frequency (within the 400–600 time steps) and 100% increase in the frequency (within the 600–800 time steps) is applied to the neural network. The waveform and distribution chart of $u(n)$ and $y(n)$ in this example is shown in Fig. 3 and Fig. 4 respectively. The distribution of $u(n)$ and $y(n)$ can be observed intuitively by sorting the values of $u(n)$ and $y(n)$. The output values of $u(n)$ are approximately evenly distributed between -1 and +1. The value of $y(n)$ ranges from 0 to 5.5, and most of them are located between 3 and 5.

Refer to Fig. 3 and Fig. 4. After sorting the values of $u(n)$ and $y(n)$, the distribution can be observed intuitively. All output values of $u(n)$ are evenly distributed between -1 and +1. The output value of $y(n)$ is between 0 and 5.5, and most of it is between 3 and 5. In the process of simulation, the example is learned by both offline and online training methods, and the parameters of SCFNN are adjusted to produce four and six hidden nodes. The membership functions of the hidden nodes of SCFNN that complete the learning are shown in Fig. 5 and Fig. 6.

When the number of hidden nodes is 4, the membership function vector space of $u(n)$ input nodes is uniformly distributed between –1 and 1, and $y(n)$ and $y(n$-1) membership function vector spaces are distributed within –1 and 5 in both training modes. The advantage of membership function distribution of hidden nodes generated by online training cannot be seen.

However, when the number of hidden nodes is increased to 6, the SCFNN obtained by offline training repeats the two additional hidden nodes near the existed node membership function of 4 hidden nodes. Especially the hidden nodes inputted by $y(n)$ and $y(n$-1) put all the additional two node membership functions between 0 and 3 of the input value distribution, which can only slightly improve the performance of the original network. The SCFNN obtained by online training places the two hidden nodes added by $y(n)$ and $y(n$-1) input hidden nodes between 3 and 5 with a large number of input values. In addition, for the hidden nodes of $u(n)$ input nodes, the transfer function obtained by online training is more uniform between -1 and 1, which is consistent with the distribution of the

Figure 3. The waveform and distribution of u(n)

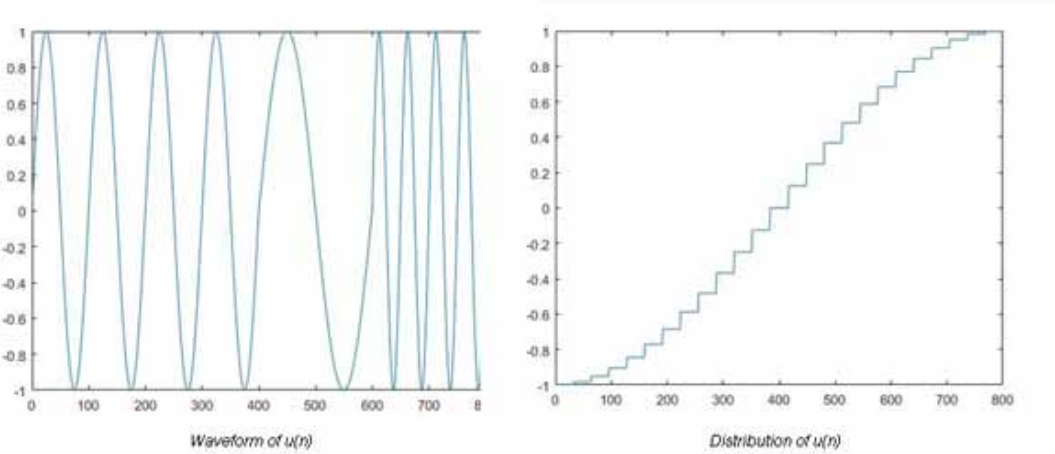
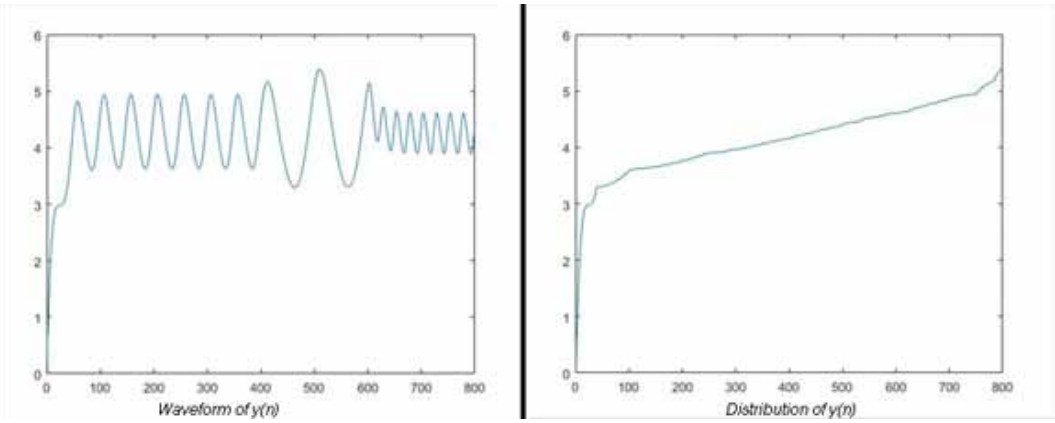


Figure 4. The waveform and distribution of y(n)



actual input value of $u(n)$. Therefore, the input membership function distribution obtained by online training is more reasonable than that obtained by offline training.

Table 1 shows the root mean square error (RMSE) of both training and test data. As can be seen from this table, the off-line training of SCFNN has achieved a very low training-data RMSE, but the RMSE of its test-data is far larger than that of the training-data. This is because there are too few samples for off-line training, and it is easy to fall into the trap of local optimum. Although the RMSE of the training-data of online training SCFNN is much larger than that of offline training SCFNN, it achieves better accuracy in testing because online training SCFNN places the additional hidden layer nodes in a more reasonable vector space.

4. SIMULATION RESULTS OF SCFNN, RBFNN AND MLPNN IN NDSI

The objective of this section is to illustrate the performance and capabilities comparison of the online training SCFNN, MLP and RBF for identification of four nonlinear systems models introduced in

Figure 5. the membership function of u(n), y(n) and y(n-1) in offline and online training (4 hidden nodes)

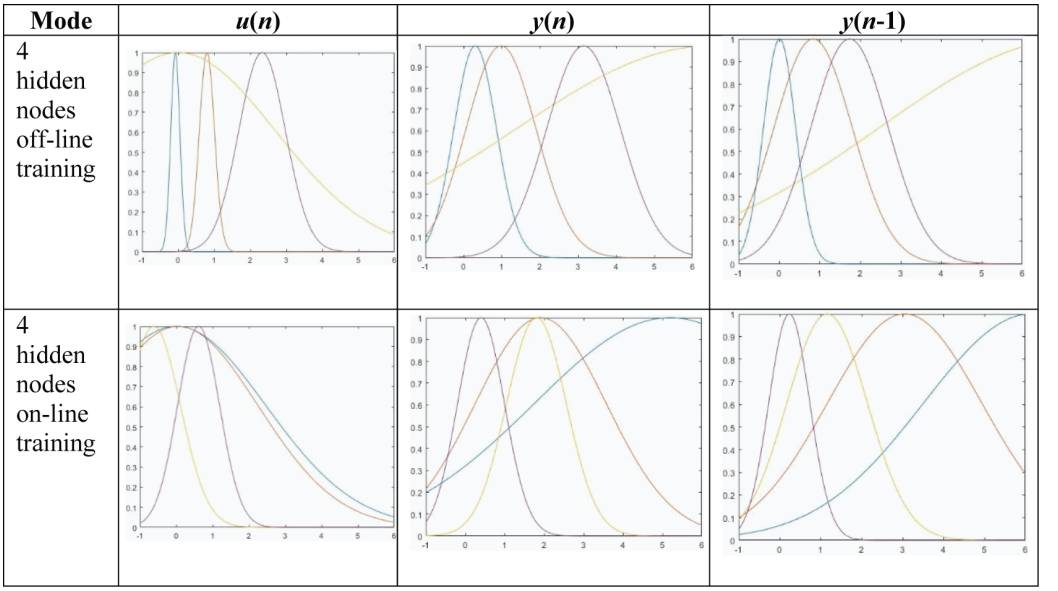


Figure 6. the membership function of u(n), y(n) and y(n-1) in offline and online training (6 hidden nodes)

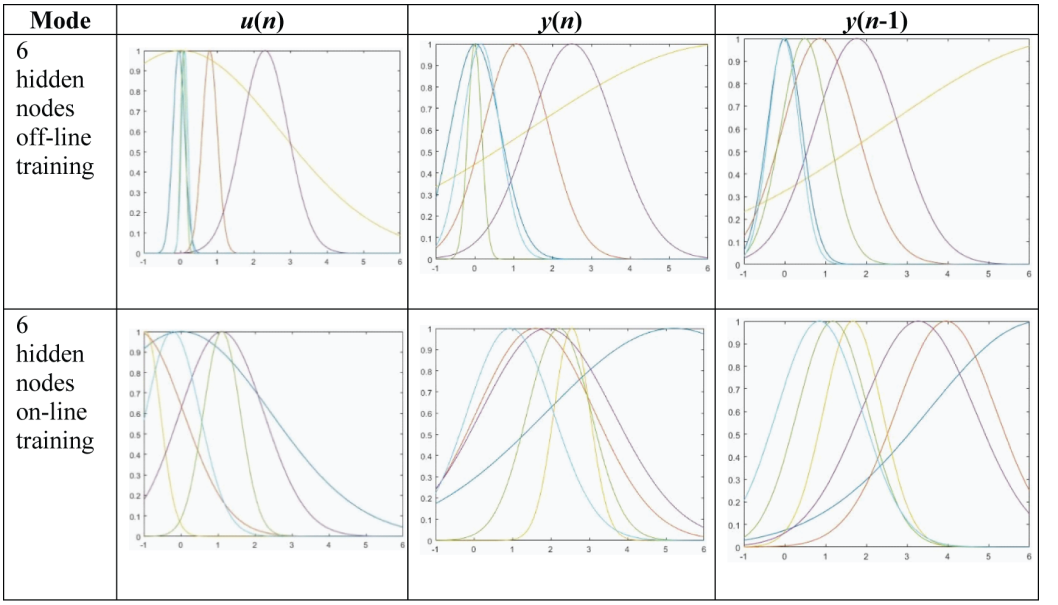


Table 1. RMSE comparison in online and offline training

Training mode	Training rmse	Testing rmse
Offline training	0.0138	0.1206
Online training	0.0502	0.0907

Sec. 3.1. In order to fairly compare the performance, the authors set the same number of hidden nodes to these three neural networks to make them in the same complexity. The reference input $u(n)$ to all the identifiers must be selected to be "persistently exciting". For identification of these systems the persistent excitation of the input guarantees the convergence of the identifier parameters to their true values. The amplitude and the frequency of the reference inputs are selected experimentally as recommended in (Moghanloo et al., 2015; Narendra & Parthasarathy, 1990).

In order to evaluate the performance of SCFNN, MLP and RBF in NDSI, the authors first simulate SCFNN, get the number of SCFNN fuzzy rules, and then set the same hidden nodes in MLP and RBF to simulate. The four examples simulated in this paper are all trained and predicted online. E_{RMSE} in equation (6) and E_{PA} in equation (7) are two prediction performance evaluation items, which are defined as

$$E_{RMSE} = \left(\frac{1}{N-1} \sum_{n=1}^{N} (\hat{y}(n) - y(n))^2 \right)^{1/2} \tag{6}$$

$$E_{PA} = \frac{\sum_{n=1}^{N} \left(\hat{y}(n) - \hat{y}_m \right) \left(y(n) - y_m \right)}{(\mathrm{N}-1) \sigma_{\hat{y}} \sigma_y} \tag{7}$$

Here $y(n)$ is the real output of the system output, and $\hat{y}(n)$ is the prediction value of the neural network. Similarly y_m and σ_y are mean and standard deviation of $y(n)$ respectively, and $\hat{y}_m$ and $\sigma_{\hat{y}}$ are mean and standard deviation of $\hat{y}(n)$ respectively. E_{RMSE} represents the observation value average deviation degree. E_{PA} represents the relationship of average prediction and observation value, where E_{PA}Î[-1,1]. When the prediction error is 0, E_{PA}=1. In summary, E_{RMSE} indicates the absolute deviation between prediction and observation value, and E_{PA} reacts the similarity between prediction and observation data (Xi & Wang, 2012).

Each example has a convergence comparison curve, three output and error curves and a numerical comparison table. Without exception, SCF converges faster than MLP and RBF. From Table 2 to Table 5, it can be seen that SCF is superior to the residual MLP and RBF in both training and testing stages. The root mean square error of the test period is higher than that of the training period, which is a reasonable phenomenon and an inevitable result of online learning and testing.

4.1. First Example

In the first example, the governing equation of the system is given by (Narendra & Parthasarathy, 1990; Xi & Wang, 2012), that is

$$y(n+1) = 0.3y(n) + 0.6y(n-1) + \frac{0.6}{1 + u(n)^2} \tag{8}$$

Note that SCFNN, RBFNN and MLPNN are trained online. The authors set them the equal complexity by setting the same number of hidden nodes in RBF and MLP. Fig. 7 depicts the convergence comparison of SCF, MLP and RBF, which shows that SCFNN converges faster. Fig. 8-Fig. 10 show the corresponding simulation results of SCF, MLP and RBF, respectively. As shown in these figures, SCFNN is more robust than the other two networks with respect to the variations in the amplitude as well as the frequency of the input. Table 2 shows the simulation results, which indicates SCFNN obviously outperforms RBFNN and MLPNN with respect to different metrics.

Figure 7. The learning convergence curves of SCFNN,MLPNN and RBFNN for the first example

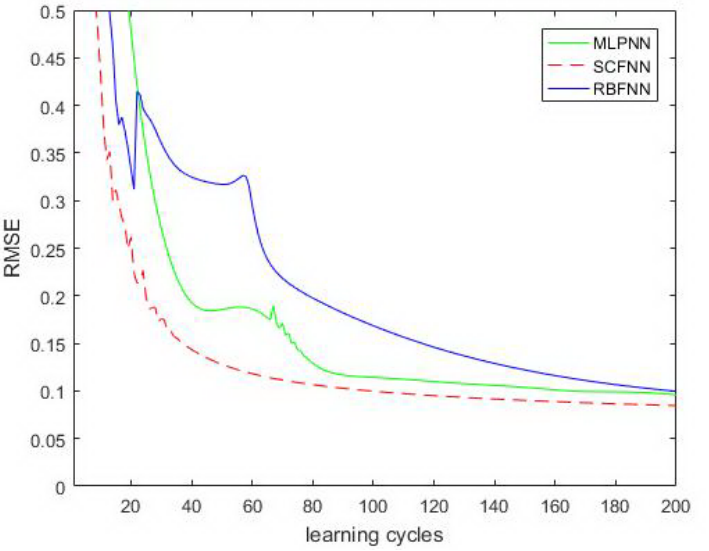
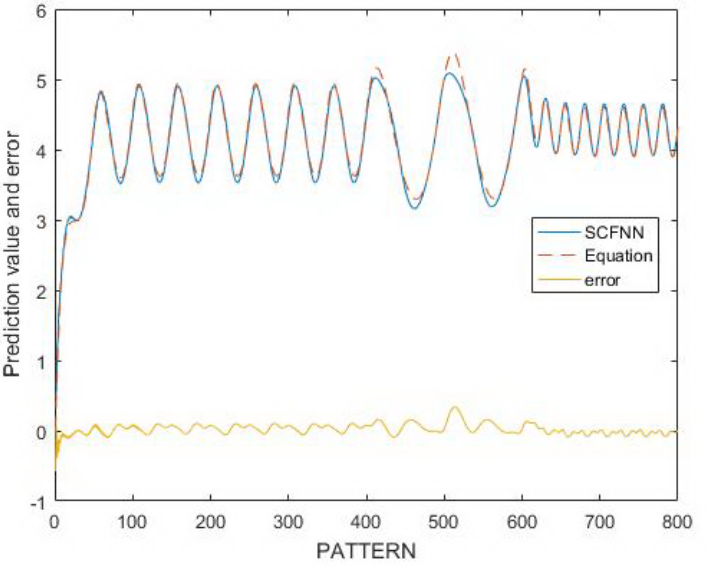


Figure 8. The outputs of SCFNN and PLANT, and their errors



4.2. Second Example

In the second example, the governing equation of the system is given by (Narendra & Parthasarathy, 1990; Xi & Wang, 2012), that is

$$y(n+1)=\frac{y(n)y(n-1)+\left(y(n-1)+2.5\right)}{1+y(n)^{2}+y(n-1)^{2}}+u(n) \tag{9}$$

Figure 9. The outputs of RBFNN and PLANT, and their errors

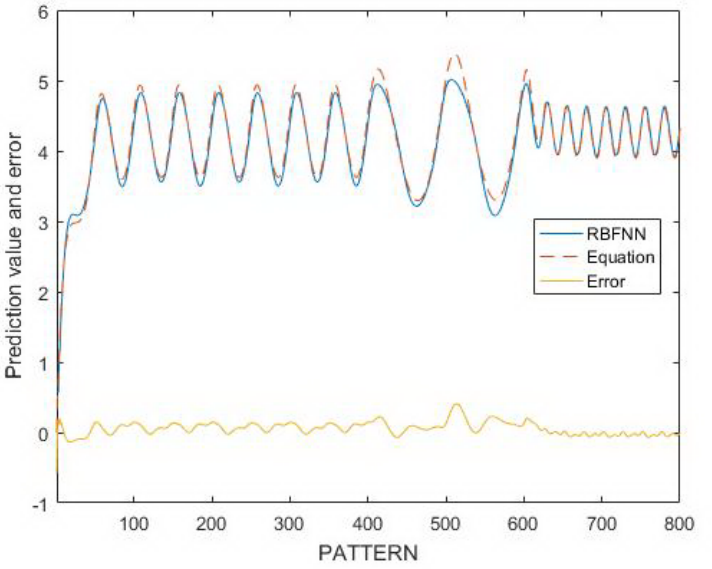


Figure 10. The outputs of MLPNN and PLANT, and their errors

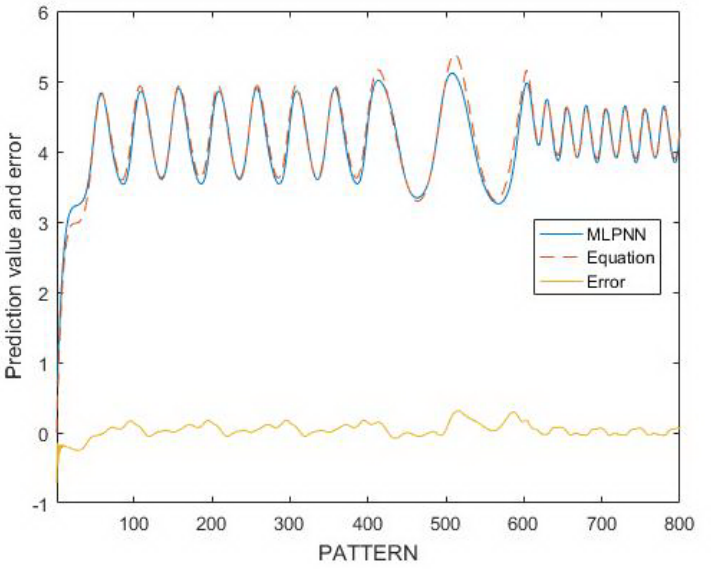


Table 2. Numerical comparison of SCFNN, MLP and RBF for the first example

Neural Network	Fuzzy rules (Perceptrons)	Training Ermse	Testing Ermse	Prediction Accuracy
SCF	16	0.0518	0.0834	0.9927
RBF	16	0.0915	0.1084	0.9907
MLP	16	0.0587	0.1132	0.9902

where the output at time (n+1) is a nonlinear function of the outputs at times (n) and (n –1) plus a linear function of the input at time (n). The reference input $u(n)$ is selected as $u(n) = sin(2\pi n/25)$. An input with 50% reduction in the amplitude (within the 100–200 time steps), 100% increase in the amplitude (within the 200–300 time steps), 50% reduction in the frequency (within the 300–400 time steps), and 100%increase in the frequency (within the 400–500 time steps) is applied to the neural network in order to show the robustness of the proposed structure to variations in the input amplitude and frequency.

The training and validation processes of SCFNN, RBFNN and MLPNN in NDSI are same with the first example. Fig. 11 displays the convergence comparison of SCFNN, MLP and RBF. Fig. 12-Fig. 146 display the corresponding simulation results of SCFNN, MLP and RBF respectively. Table 3 shows the simulation results numerically with respect to various metrics.

4.3. Third Example

In the third example, the governing equation of the system is given by (Narendra & Parthasarathy, 1990; Xi & Wang, 2012), that is

$$y\left(n+1\right) = \frac{0.2y\left(n\right) + 0.6y\left(n-1\right)}{1+y\left(n\right)^2} + sin\left(u\left(n\right)\right) \tag{10}$$

where the output at time (n+1) is a nonlinear function of the output at time (n) and (n−1) plus a nonlinear function of the input at time (n). The reference input applied to the system is $u(n)$= $sin(2\pi n)/10) + sin(2\pi n/25)$. An input with 50% reduction in the amplitude (within the 400–600 time steps), and 100%increase in the frequency (within the 600–800 time steps) is applied to the neural network in order to show the robustness of the proposed structure to variations in the input amplitude and frequency.

The training and validation processes of SCFNN, RBFNN and MLPNN in NDSI are same with the first example. Fig. 15 displays the convergence comparison of SCFNN, MLP and RBF. Fig. 16-

Figure 11. The learning convergence curves of SCFNN, MLPNN and RBFNN for the second example

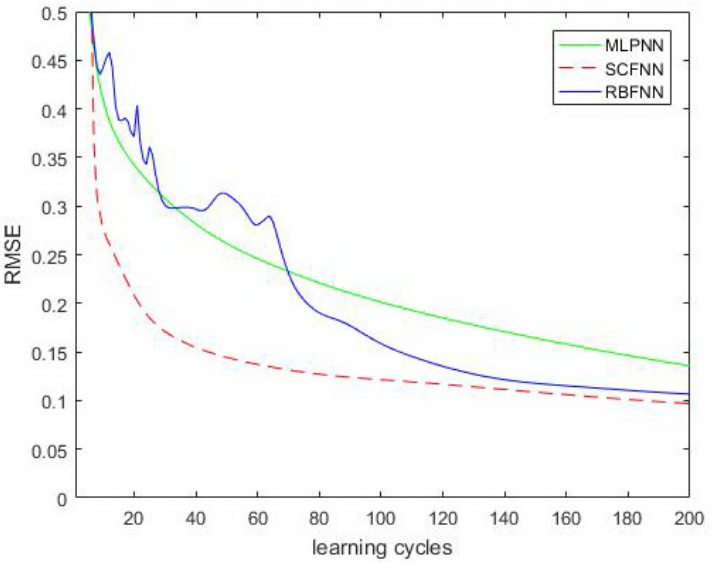


Figure 12. The outputs of SCFNN and PLANT, and their errors

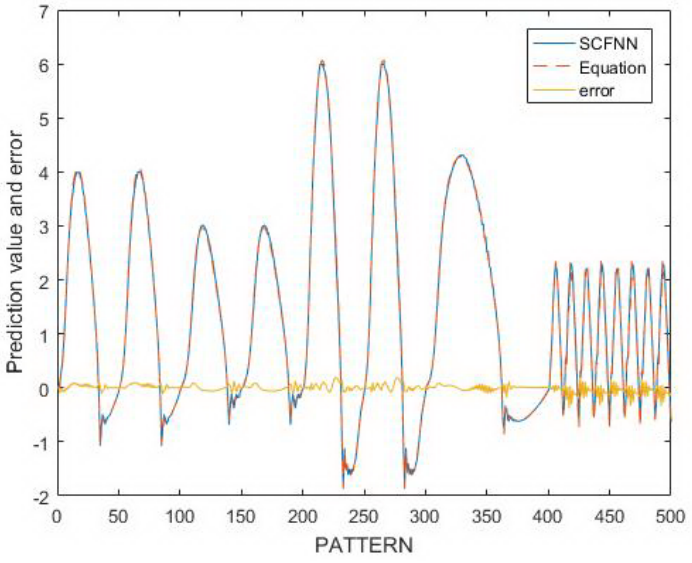


Figure 13. The outputs of RBFNN and PLANT, and their errors

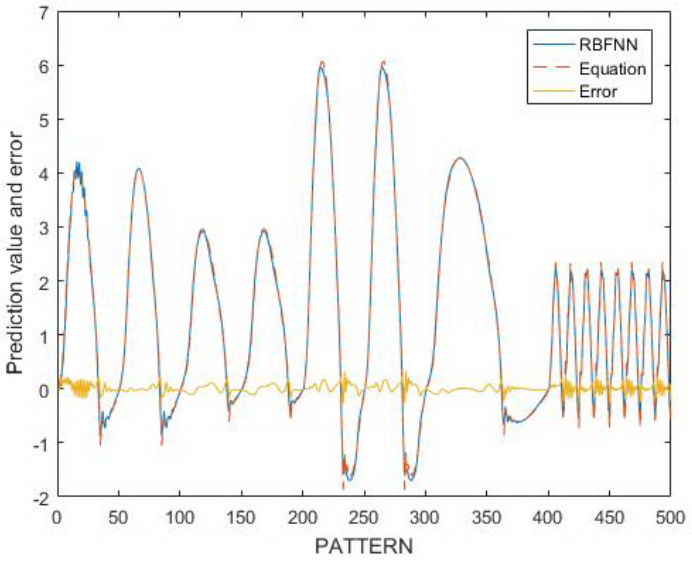


Fig. 18 display the corresponding simulation results of SCFNN, MLP and RBF respectively. Table 4 shows the simulation results numerically.

4.4. Fourth Example

In the final example, the governing equation of the system is given by (Narendra & Parthasarathy, 1990; Xi & Wang, 2012), that is

Figure 14. The outputs of MLPNN and PLANT, and their errors

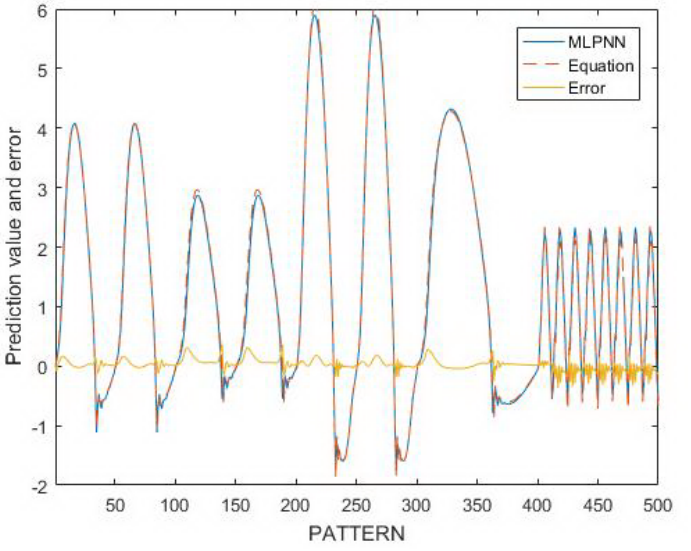


Table 3. Numerical comparison of SCFNN, MLP and RBF the second example

Neural Network	Fuzzy rules (Perceptrons)	Training Ermse	Testing Ermse	Prediction Accuracy
SCF	40	0.0770	0.0771	0.9992
RBF	40	0.0831	0.0910	0.9988
MLP	40	0.0812	0.1082	0.9984

Figure 15. The learning convergence curves of SCFNN,MLPNN and RBFNN for the third example

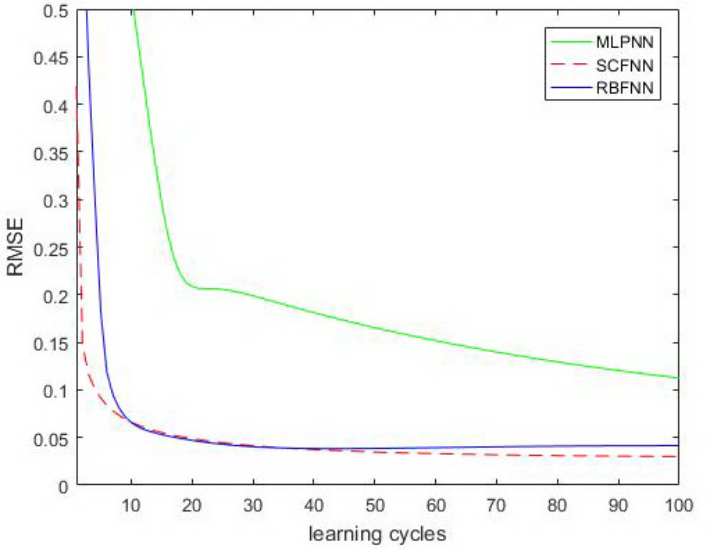


Figure 16. The outputs of SCFNN and PLANT, and their errors

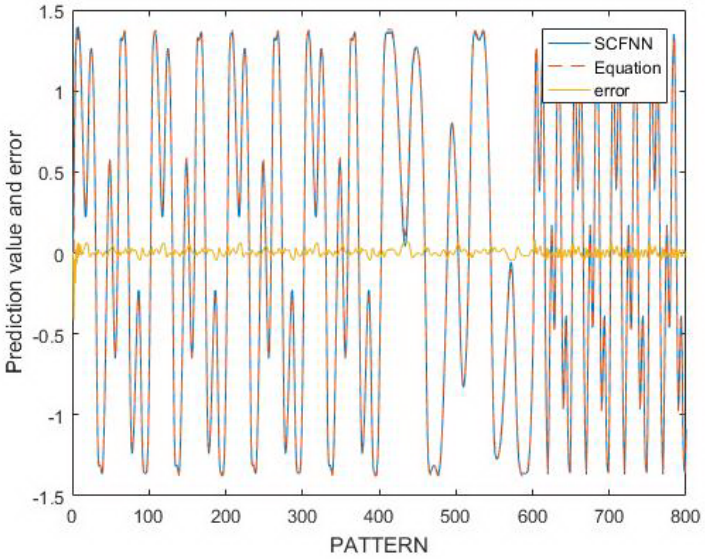
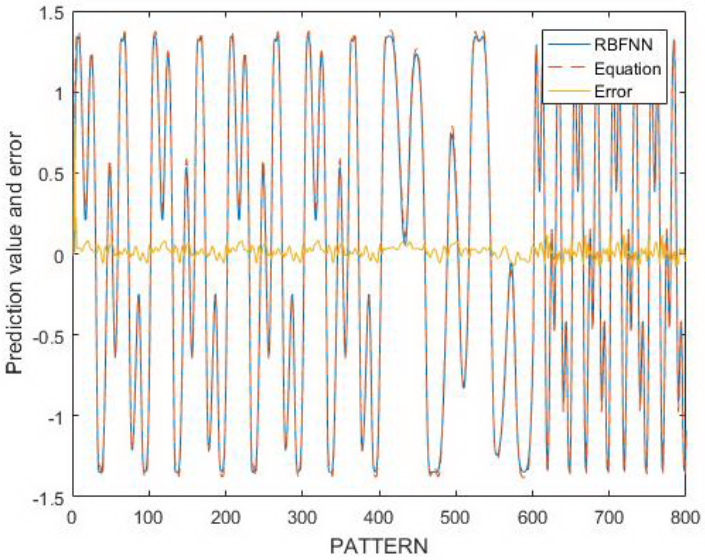


Figure 17. The outputs of RBFNN and PLANT, and their errors



$$y\left(n+1\right)=\frac{y\left(n\right)+u\left(n\right)}{1+y\left(n\right)^{2}} \tag{11}$$

Figure 18. The outputs of MLPNN and PLANT, and their errors

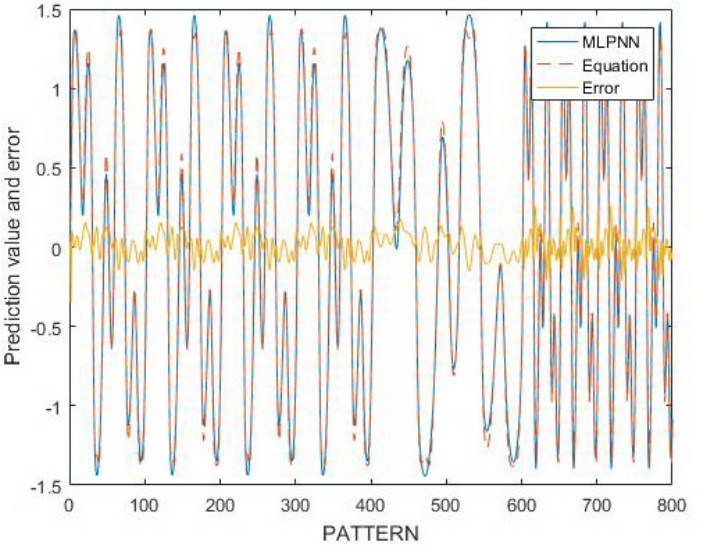


Table 4 Numerical comparison of SCFNN, MLP and RBF for the third example

Neural Network	Fuzzy rules (Perceptrons)	Training Ermse	Testing Ermse	Prediction Accuracy
SCF	12	0.0284	0.0290	0.9995
RBF	12	0.0367	0.0482	0.9989
MLP	12	0.0777	0.0818	0.9966

where the output at time $(n+1)$ is a nonlinear function of the outputs at times (n) and the inputs at times (n). The reference input is $u(n) = sin(2\pi n/50)$. An input with 50%reduction in the amplitude (within the 250–500 time steps), 100% increase in the amplitude (within the 500–750 time steps), 50% reduction in the frequency (within the 750–1,000 time steps), and 100% increase in the frequency (within the 1,000–1250 time steps) is applied to the neural network in order to show the robustness of the proposed structure to variations in the input amplitude and frequency.

The training and validation processes of SCFNN, RBFNN and MLPNN in NDSI are same with the first example. Fig. 19 displays the convergence comparison of SCFNN, MLP and RBF. Fig. 20-Fig. 22 display simulation results of SCFNN, MLP and RBF respectively. Table 5 shows the simulation results numerically.

5. CONCLUSION

In this paper, an online SCFNN for NDSI with applications in IoTs is proposed. Setting the conditions carefully to increase the demand for fuzzy rules will make the architecture of the SCFM constructed quite simple. The paper also propose a new NDSI online learning model, and the advantages of online learning over offline learning of neural networks in NDSI are analyzed. Furthermore, this paper applies online MLP and online RBF in the same identification of four models of nonlinear systems. Simulation results indicate that SCFNN, MLP and RBF learning in steepest descent method

Figure 19. The learning convergence curves of SCFNN,MLPNN and RBFNN for the fourth example

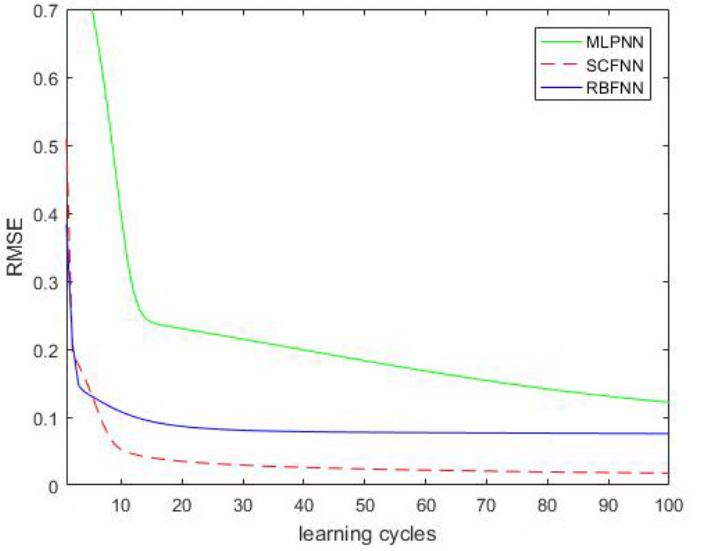
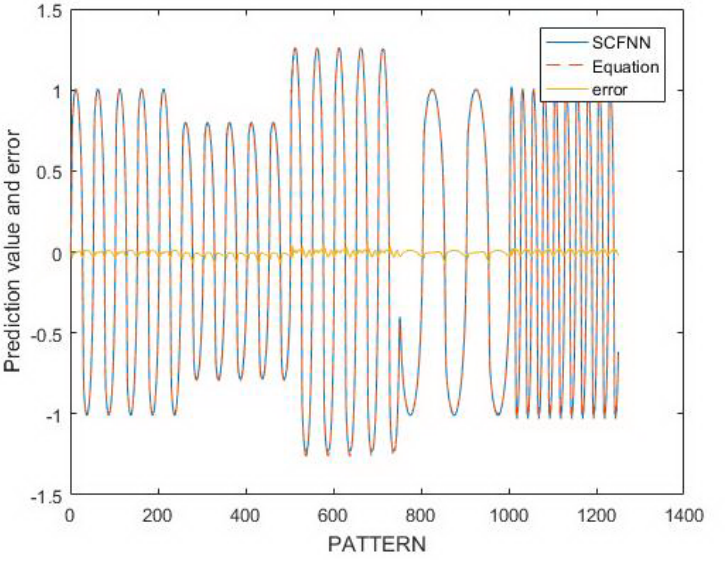


Figure 20. The outputs of SCFNN and PLANT, and their errors



are effective in identifying the input–output mappings of different classes of nonlinear systems. SCFNN constructs a compact FNN by structure learning, and adjusts the parameters of membership functions and link weights by parameter learning. No matter in convergence speed, training RMSE, testing RMSE and prediction accuracy, the proposed SCFNN has better performance in comparison with MLP and RBF. In the future, the authors will try to further apply the proposed online SCFNN to real systems in IoTs and evaluate its performance on them.

Figure 21. The outputs of RBFNN and PLANT, and their errors

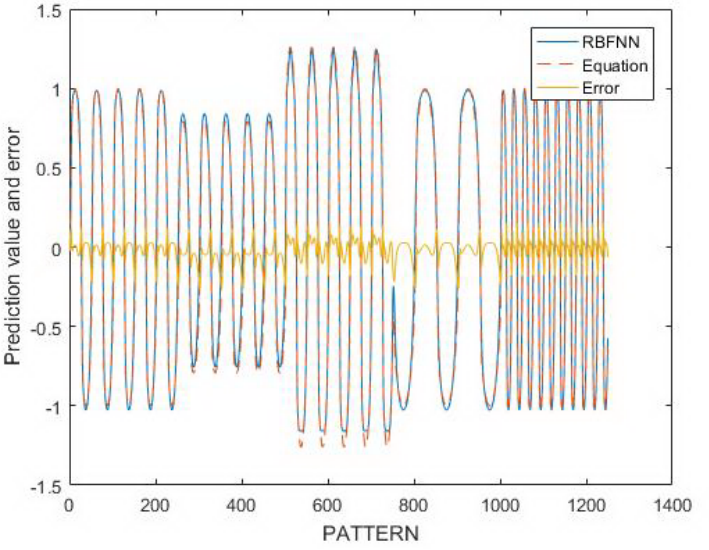


Figure 22. The outputs of MLPNN and PLANT, and their errors

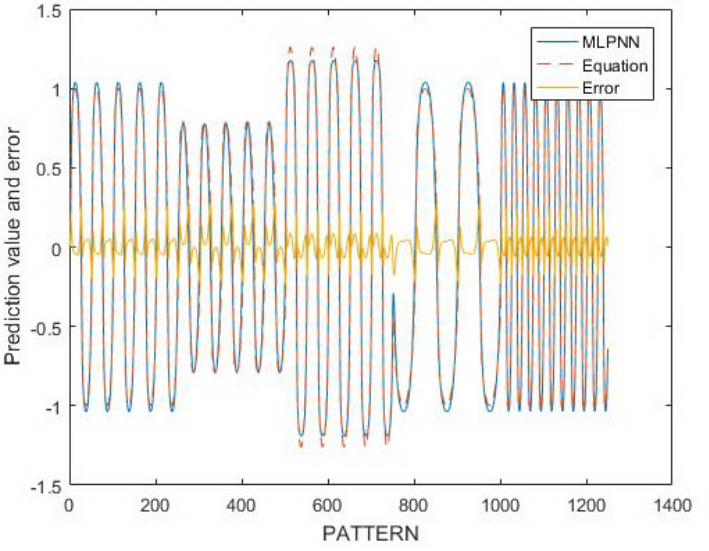


Table 5. Numerical comparison of SCFNN, MLP and RBF for the fourth example

Neural network	Fuzzy rules (Perceptrons)	Training Ermse	Testing Ermse	Prediction Accuracy
SCF	12	0.0130	0.0179	0.9998
RBF	12	0.0729	0.0772	0.9962
MLP	12	0.0918	0.0921	0.9941

ACKNOWLEDGMENT

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

REFERENCES

Adachi, S., Okada, Y., & Maciejowski, J. M. (2004). System identification in the presence of nonlinear sensors. *IFAC Proceedings Volumes*, *37*(12), 185-190.

Antsaklis, P. J. (1992). Special issue on neural networks in control systems. *IEEE Control System Magazine*, *12*.

Cao, J., & Liang, J. (2004). Boundedness and stability for Cohen–Grossberg neural network with time-varying delays. *Journal of Mathematical Analysis and Applications*, *296*(2), 665–685. doi:10.1016/j.jmaa.2004.04.039

Cao, J., & Lin, X. (2008). Application of the diagonal recurrent wavelet neural network to solar irradiation forecast assisted with fuzzy technique. *Engineering Applications of Artificial Intelligence*, *21*(8), 1255–1263. doi:10.1016/j.engappai.2008.02.003

Han, H., Chen, Q., & Qiao, J. (2010). Research on an online self-organizing radial basis function neural network. *Neural Computing & Applications*, *19*(5), 667–676. doi:10.1007/s00521-009-0323-6 PMID:20651904

Han, H., Lin, Z., & Qiao, J. (2017). Modeling of nonlinear systems using the self-organizing fuzzy neural network with adaptive gradient algorithm. *Neurocomputing*, *266*, 566–578. doi:10.1016/j.neucom.2017.05.065

Haykin, S. (2008). Neural Networks: A Comprehensive Foundation, 3rd ed., Pearson, 71-80.

Kumar, R., Srivastava, S., & Gupta, J. R. P. (2017). Modeling and adaptive control of nonlinear dynamical systems using radial basis function network. *Soft Computing*, *21*(15), 4447–4463. doi:10.1007/s00500-016-2447-9

Lin, F. J., Lin, C. H., & Shen, P. H. (2001). Self-constructing fuzzy neural network speed controller for permanent-magnet synchronous motor drive. *IEEE Transactions on Fuzzy Systems*, *9*(5), 751–759. doi:10.1109/91.963761

Manonmani, A., Thyagarajan, T., Elango, M., & Sutha, S. (2016). Modelling and control of greenhouse system using neural networks. *Transactions of the Institute of Measurement and Control*, *40*(3), 918–929. doi:10.1177/0142331216670235

Moghanloo, F. N., Yazdizadeh, A., & Fomani, A. P. J. (2015). A New Modified Elman Neural Network with Stable Learning Algorithms for Identification of Nonlinear Systems. Computer and Information Science, 171-193.

Narendra, K. S., & Parthasarathy, K. (1990). Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, *1*(1), 4–27. doi:10.1109/72.80202 PMID:18282820

Narendra, K. S., & Parthasarathy, K. (1992). Neural networks and dynamical systems. *International Journal of Approximate Reasoning*, *6*(2), 109–131. doi:10.1016/0888-613X(92)90014-Q

Park, J., & Sandberg, I. W. (1991). Universal approximation using radial-basis-function networks. *Neural Computation*, *3*(2), 246–257. doi:10.1162/neco.1991.3.2.246 PMID:31167308

Parlos, A. G., Chong, K. T., & Atiya, A. F. (1994). Application of the recurrent multilayer perceptron in modeling complex process dynamics. *IEEE Transactions on Neural Networks*, *5*(2), 255–266. doi:10.1109/72.279189 PMID:18267795

Patra, J. C., Ang, E. L., Das, A., & Chaudhari, N. S. (2005). Auto-compensation of nonlinear influence of environmental parameters on the sensor characteristics using neural networks. *ISA Transactions*, *44*(2), 165–176. doi:10.1016/S0019-0578(07)60175-X PMID:15868856

Pislaru, C., & Shebani, A. (2014). Identification of Nonlinear Systems Using Radial Basis Function Neural Network. *International Journal of Computer, Information Systems and Control Engineering*, *8*(9), 1528–1533.

Qiao, J., Wang, G., Li, X., & Li, W. (2018). A self-organizing deep belief network for nonlinear system modeling. *Applied Soft Computing*, *65*, 170–183. doi:10.1016/j.asoc.2018.01.019

Taghavifar, H., & Mardani, A. (2015). Prognostication of energy consumption and greenhouse gas (ghg) emissions analysis of apple production in west azarbayjan of iran using artificial neural network. *Journal of Cleaner Production*, *87*, 159–167. doi:10.1016/j.jclepro.2014.10.054

Wei, Q., & Liu, D. (2015). Neural-network-based adaptive optimal tracking control scheme for discrete-time nonlinear systems with approximation errors. *Neurocomputing*, *149*, 106–115. doi:10.1016/j.neucom.2013.09.069

Weng, W. D., Lin, R. C., & Hsueh, C. T. (2005). The Design of an SCFNN Based Nonlinear Channel Equalizer. *Journal of Information Science and Engineering*, *21*(4), 695–709.

Xi, J., & Wang, H. (2012). Modeling and prediction of multivariate series based on an improved RBF network. *Information and Control*, *41*(2), 159–164.

Xia, F., Yang, L. T., Wang, L., & Vinel, A. (2012). Internet of things. *International Journal of Communication Systems*, *25*(9), 1101–1102. doi:10.1002/dac.2417

Yoo, S. J., Park, J. B., & Choi, Y. H. (2006). Adaptive dynamic surface control of flexible-joint robots using self-recurrent wavelet neural networks. *IEEE Transactions on Systems, Man, and Cybernetics. Part B, Cybernetics*, *36*(6), 1342–1355. doi:10.1109/TSMCB.2006.875869 PMID:17186810

Zhao, H., & Zhang, J. (2009). Nonlinear dynamic sensor identification using pipelined functional link artificial recurrent neural network. *Neurocomputing*, *72*(13-15), 3046–3054. doi:10.1016/j.neucom.2009.04.001

Ye Lin received his M.S. degree in Software Engineering from Huazhong University of Science and Technology 2012. He is presently a lecturer with the School of Zhejiang Industry and Trade Vocational College, China. His teaching and research interests are in the areas of microprocessor applications, image and signal processing, and artificial neural network.

Yea-Shuan Huang graduated from the Computer Science Department of Concordia University, Canada, in 1994. He retired from Industrial Technology Research Institute (Taiwan) in 2006 and at the same year became an associate professor of Computer Science & Information Engineering Department in Chung-Hua University. His research interests are in the areas of face recognition, gesture analysis, biometrics authentication, OCR, image analysis, computer vision, and pattern recognition. In 2010 and 2011, his received the third place and the second place of the Utechzone Machine Vision Prize on performing face detection, face recognition, gender recognition and age recognition. He has received about 50 patents and performed more than 10 technology transfers to industrial companies and research institutes.

Rui-Chang Lin received his Ph.D. from the Institute of Engineering Science and technology of Yunlin University of science and technology in 2005. Now he is a vice professor of mechanical and electrical College of Guangzhou Panyu Polytechnic. Teaching and research fields include neural network application, FPGA system design, power supply and distribution technology.