Population-based Automatic Fuzzy Neural Network for Online, Knowledge-based Learning

Mohammad Reza Keyvanpour[1], Hajar Homayouni[2], Samaneh Zolfaghari[3]

Receive :2016/04/20 Accepted: 2016/07/26

Abstract

In this paper, a novel fuzzy connectionist system for incremental online learning and knowledge discovery called Population-based Automatic Fuzzy Neural Network (PAFuNN) is demonstrated in detail. PAFuNNs evolve out of incremental learning. New connections and neurons are created based on a population of samples while operating the system which has the advantage of controlling the number of neurons involved and leads to the low complexity of the network. Learning Automata is implemented in order to optimize the network parameters including sensitivity and error thresholds to enhance the performance of the entire system. Afterward, the proposed method is compared with Evolving Fuzzy Neural Network (EFuNN) as a general online learning machine on two case study datasets consisting of gas furnace and iris data for prediction and classification tasks leading to the thorough analysis of the effects of selecting appropriate automata. Less complex, more accurate and robust results are obtained for the proposed method in comparison with the EFuNN.

Keywords: Evolving connectionist systems; Population-based Automatic fuzzy neural networks; On-line learning; Knowledge-based neural networks.

I. Introduction

Sophisticated methods and devices are required for building intelligent Information Systems (IS) which are capable of learning various types of knowledge through their incremental interaction with the environment [1]. The major requirements of the IS are: (1) learning rapidly from a large database, (2) adapting incrementally in an online way, (3) having an open structure, (4) having a long term memory, (5) interacting with the environment continually, (6) dealing with knowledge, and (7) representing space and time adequately. There have been miscellaneous fuzzy connectionist systems which have endeavored to address the above-mentioned seven issues. Nowadays, we can see an increased use of neural networks for pattern recognition, classification and optimization tasks [2]. Knowledge Based Neural Networks (KBNNs) [3] are pre-structured neural networks allowing learning from data, rule insertion, rule extraction, adaptation and reasoning which is a combination of fuzzy logic systems [4] and neural networks [5]. Fuzzy Neural Network (FUNN) [6] is a particular set of KBNNs in which structure can be interpreted as a set of fuzzy rules. EFuNNs [1] have the advantages of traditional KBNNs; nonetheless, they learn in a one-pass online mode, evolve utilizing local element tuning while their structure fluctuates as the system operates. Although EFuNNs are suitable for learning on-line incoming data rapidly, they are of high complexity in as much as there is no control over the number of the nodes added through the operation of the system. Creating the nodes in EFuNN which is expected to be optimized is based on the currently presented data resulting in a huge number of Rule Nodes. Some aggregating [7] and [8] as well as pruning [3] approaches are proposed for the reduction of the increasing number of nodes (Rule Nodes). Evolutionary methods [9] are also proposed for the optimization of EFuNN parameters. These methods are mostly slow in terms of the running time. On the other hand, in basic EFuNNs, thresholds are considered to be fixed and there is no efficient strategy for optimizing them, apart from a self-tuning approach in which thresholds are tuned locally based on novel samples [10].

The PAFuNN model presented here principally differs from all the fuzzy neural network models introduced so far despite the existing structural similarities. PAFuNNs have a five- layer structure similar to that of EFuNNs. Furthermore, it is appropriate for on-line knowledge discovery of

[1] Associate Professor at Computer Engineering Department, Alzahra University, Vanak Village Street, Tehran, Iran E-mail: Keyvanpour@alzahra.ac.ir

[2]PhD Candidate at Computer Engineering Department, University of Isfahan, Iran, E-mail: homayouni.hajar@eng.ui.ac.ir

[3]Msc Student at Computer Engineering Department, Alzahra University, Tehran, Iran, E-mail: s.zolfaghari.ir@ieee.org

large databases. In this method, neurons and connections are created based on a set of refuted samples unlike the EFuNNs in which each neuron is produced only on the basis of a single presented sample, which has the advantage of controlling the number of neurons so that they would not get too large. In other words, in EFuNNs, there is no control over the number of neurons added to the network throughout network learning leading to a complex-structured network. In a complex neural network, there is a high probability of over-fitting the network on input samples which is considered to be a problem that is more vital for noisy datasets in which the network learns noisy data absolutely efficiently which leads to high output error in testing the dataset. In the proposed method, after some samples are presented to the network and a definite number of Rule Nodes are created, if a sample does not match any of the existing Rule Nodes in the network, it will be stored and regularly, some Rule Nodes are produced according to a set of such samples.

The word "automatic" in the title of the proposed system is concerned with implementing learning automata so as to get adapted to the system parameters. Two fixed structured learning automata (FSLA) [11] are interconnected to the network in order to get adjusted to the sensitivity and error thresholds of the network to enhance the entire performance of the system and escape from the local minima. In raw EFuNN, the parameters are set to fixed values.

A comparative analysis between PAFuNN and EFuNN on two benchmarks proves the fact that PAFuNNs are comparable with EFuNNs in terms of the accuracy and robustness of the obtained results; nevertheless, they are faster, more controllable, and less complex.

The rest of this paper is organized as follows: In Section II, the PAFuNN is demonstrated and in Section III, the experimental consequences of applying the proposed algorithm on two case studies are analyzed. In the end, Section IV and V are associated with feedbacks.

II. Proposed Method

PAFuNNs have a five-layer structure similar to that of EFuNNs [1, 3, 7], and are appropriate for on-line knowledge discovery of mega-databases. The PAFuNNs have the advantages of the EFuNNs. Nonetheless, they possess two important distinctions so as to overcome their current challenges:

1- Creating neurons based on a set of refuted samples
2- Utilizing Fixed Structure Learning Automata in order to get adapted to the network parameters

They are both described in detail in the following.

1) Creating neurons based on a population of rejected examples

In order to control the entire number of the created nodes, the PAFuNN creates neurons on the basis of a set of refuted samples. To illustrate this, after some data samples are presented to the network and a definite number of Rule nodes are created, if a coming data does not match any of the existing Rule Nodes in the network, which means that the two aforementioned conditions in the EFuNN [3] are not satisfied for that input, it is stored and the next sample will be presented to the system. Otherwise, the network parameters are adjusted through hybrid supervised/unsupervised learning similar to the aforementioned EFuNN approach.

Regularly, at the end of some chunks of data presented to the network, Rule Nodes are created based on the set of the refuted stored samples utilizing the following algorithm instead of producing Rule Nodes for each single sample.

The last two parameters C(r) and Age(r) are appropriate for network pruning algorithms [1]. As the Algorithm 1 illustrates, the fuzzified stored samples are sorted first on basis of their fuzzy distance calculated via Eq. (1).

Algorithm 1. Neurons creation based on a population of rejected examples

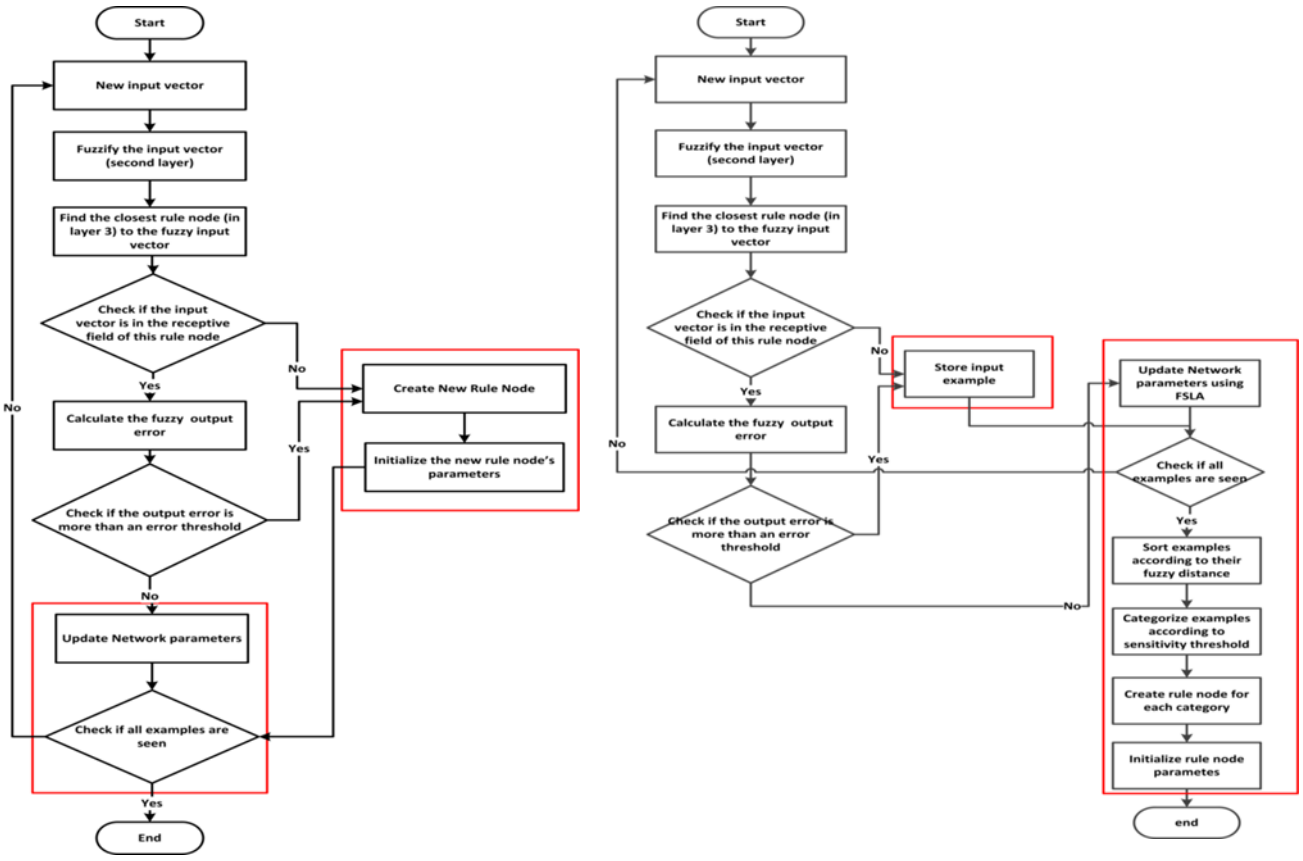
S=Set of fuzzified stored examples; // S={E_f | E_f= (X_f,Y_f)}
Sort S elements according to their fuzzy distances;
MD= maximum fuzzy distance between S elements;
*Number of categories: MD * sensitivity threshold;*
For i=1: Number of categories
For its subset S'={E_f'| E_f' =(X_f', Y_f')} of S
Create a new rule node r;
//Set its parameters:
W_1(r) = Mean of subset X_f';
W_2(r) = Mean of subset Y_f';
C(r)= Size (S') //number of samples pertain to r;
Age(r) = 0;

$$FD(E_{f1}, E_{f2}) = \frac{\|X_{f1} - X_{f2}\|}{\|X_{f1} + X_{f2}\|} \qquad (1)$$

These sets of samples should then be categorized into some subsets based on the sensitivity threshold inasmuch as this threshold is the radius of the input hyper sphere of a Rule Node and indicates the samples associated with the Rule Node. For each category, a Rule Node is created and its parameters are set as demonstrated in the algorithm.

2) Utilization of fixed structure learning automata for adaptation of network parameters

In this part fixed structure learning automata [12, 13, 14, 15] are utilized in order to adjust the PAFuNNs parameters to their best values and improve the network performance. Through interconnecting learning automata to the PAFuNN, parameters SThr (sensitivity threshold) and EThr (error threshold) are adjusted based on the output of the network for each data sample. The error threshold parameter EThr sets the error tolerance of the system and also defines the radius of the output cluster for each Rule Node. The sensitivity threshold parameter defines the minimum activation of the rule node r to a novel input vector x from a new sample (x, y) in order for the sample to be considered in its association with this Rule Node Two fixed structure learning automata are assigned to the rule layer and fuzzy output layer of PAFuNN so as to determine the sensitivity and error thresholds of those layers. Note that the PAFuNN is the environment for learning automata. The actions of the automata correspond to the values of the SThr and EThr parameters. The input (which is the response of the environment) of the first automaton is some function of the activation value of the selected Rule Node and that of the second automaton is a function of the fuzzy output error. The response of the environment to the first automaton is favorable if the activation value of the selected Rule Node (The Rule Node with the highest activation or in other words with the lowest FD value) is higher than the sensitivity threshold, and that of the environment to the second automaton is favorable if the fuzzy output error is lower than the error threshold value. Figure 1 shows the EFuNN and PAFuNN flow charts respectively. The red boxes are representing the differences



(a) (b)

Fig. 1 The EFuNN (a) and PAFuNN Flow Chart (b)

III. Experimental results and analysis

In this section, the proposed method is analyzed utilizing two standard datasets called Furnace and Iris for classification and prediction tasks.

1) Dataset

In order to conduct evaluation, two standard datasets which are explained as follows are utilized. Furnace dataset is one of the most prominent datasets which has been utilized by a majority of researchers in neuro-fuzzy engineering field [16, 17, 18, 19, 20]. The dataset consists of 292 consecutive values of methane during a time zone (t-4), and the carbon dioxide (CO_2) produced in a furnace throughout a specific period (t-1) as input variables comprising the produced CO_2 in a period of time (t) as an output variable.

Iris dataset is a prominent classification dataset [21, 22] which consists of 150 instances; 3 classes -setosa, versicolour and virginica and four attributes - X_1-sepal length, X_2-sepal width, X_3-petal length, and X_4-petal width.

2) Evaluation Technique

The evaluation technique is the method proposed in [1] in which the network is trained on the basis of each data pair of input-output vectors as they become accessible in an on-line mode. Then, the network is tested on the spot to anticipate

the following data items before the latter is accommodated (learned) in the system. The network is trained according to the first half of the data (on-line, one-pass training) in this way. Then, the evolved network is tested in an off-line mode throughout the second half of the data.

3) Evaluation criteria

To assess the feedbacks obtained through applying the proposed network to the two aforementioned tasks, various criteria are demonstrated in the following.

- Root Mean Square Error (RMSE) is a standard and prominent criterion for evaluating neural networks [23, 24]. The root mean square error is calculated at each data point Di from the input data stream as shown in Eq. 2.

$$RMSE(i) = sqrt\left(\frac{sum\{Err_t\}_{t=1,2,\dots,i}}{i}\right) \quad (2)$$

Where Errt= (dt –ot)^2 , dt is the desired output value and ot is the EFuNN output value produced for the *t*th input vector Dt.

- CPU Time is a criterion for measuring neural networks efficiency [25, 26] and consists of time spent on the training and testing phases of the network. The equation of CpuTime computing is presented in Eq. 3.

$$cpuTime = cpuTrainingTime + cpuTestingTime;\ where: (3)$$

$$cpuTrainigTime = time\ spent\ on\ cpu\ during\ training\ phase$$

$$cpuTestingTime = time\ spent\ on\ cpu\ during\ testing\ phase$$

- The Number of Rule Nodes is one of the best ways to indicate the complexity of the Fuzzy Neural Networks [1]. The Eq. 4 shown relation between complexity and Rule Nodes.

$$complexity \approx \#Rule\ Nodes$$

- Robustness is a proposed criterion which calculates the robustness of the network when confronting with a changing architecture. To achieve such a purpose, 20 per cent of the connection weights of the trained network are modified to prove how long it takes to get restored and what the features of the network will be then. Therefore, the robustness of the network is represented via parameters including CPU time, RMSE and the number of rules as represented in Eq.5.

$$robustness = \frac{1}{cpuTime + RMSE + \#ruleNodes}$$

4) Comparison with other networks

As the proposed method has taken its major concept from evolving fuzzy neural networks, the EFuNN and its progresses are utilized to be compared with the proposed method PAFuNN to accomplish the expected evaluation.

- Evolving Fuzzy Neural Network (EFuNN) proposed through [27] is appropriate for on-line knowledge discovery of mega-databases and was demonstrated in Section II.
- Self-tuning EFuNN (sEFuNN) proposed by [1] is one of the first improvements applied to EFuNNs. In this method, the aggregation and pruning of the Rule Nodes are applied to control the number of the Rule Nodes. Furthermore, the sensitivity threshold is updated utilizing Eq. 6.

$$S_j(t+1) = S_j(t) + FD\left(W_{1,j}(t+1), W_{1,j}(t)\right) \quad (6)$$

• Recurrent EFuNN (rEFuNN) is proposed in [28, 29] and has a recurrent structure. The recurrent structure in a rEFuNN emerges from regionally feeding the firing strength of a fuzzy rule back to itself. Due to the fact that this method is appropriate for extracting fuzzy rules in an on-line mode, it can be a proper method which can be compared with our proposed network.

5) Results

The PAFuNN comprises two significant distinctions with basic EFuNN as stated in the previous section. In this section, these two alterations are analyzed separately for better digestion and finally both changes are applied to generate the proposed PAFuNN. To illustrate this, the first improvement (Adding Connectionist based on a set of refuted samples) applied to EFuNN is called pEFuNN and the second (Utilizing the Fixed Structure Learning Automata to adapted the network parameters) is called aEFuNN. The results are analyzed through two experiments which are basically called time series prediction and classification.

a) The First Experiment: Time Series Prediction

In this section, the capability of PAFuNN in time series anticipation task is analyzed on the basis of furnace dataset. Table 1 demonstrates RMSE for training and testing phases, the number of the Rule Nodes, CPU Time for training/testing phase, and network robustness for EFuNN, sEFuNN, rEFuNN, pEFuNN, aEFuNN and PAFuNN. For better digestion, the same results are illustrated in Figure 2. Table 2 indicates the effect of various learning automata on different evaluating criteria. The same results are presented in Figure 3. Note that the resulted values are average over 10 distinct runs. Figure 4 demonstrates the process of the networks evolving out of the Gas furnace dataset. The real versus anticipated by the network values is presented when it is trained according to the first half of the gas-furnace data (on-line, one-pass training). The evolved network is then tested in an off-line mode throughout the second half of the data.

Table 1. Comparing EFuNN, sEFuNN, rEFuNN, pEFuNN, aEFuNN and PAFuNN on Gas furnace data set

Algorithm	RMSE Train	RMSE Test	#Rule Nodes	CPU Time (Train-Test)	Robustness
EFuNN	0.09	0.106	27	1.5-0.3	1/(3.2+0.079+27)=0.033
sEFuNN	0.054	0.101	19	1.9-0.4	1/(3.1+0.064+7)=0.098
rEFuNN	0.038	0.085	26	2-0.2	1/(4.1+0.05+26)=0.033
pEFuNN	0.102	0.100	13	0.9-0.3	1/(2.8+0.06+12)=0.067
aEFuNN (Krylov for SThr Krinsky for EThr)	0.09	0.07	13	1.3-0.2	1/(2+0.077+16)=0.055
PAFuNN	0.099	0.066	10	1.5-0.2	1/(5+0.058+5)=0.099

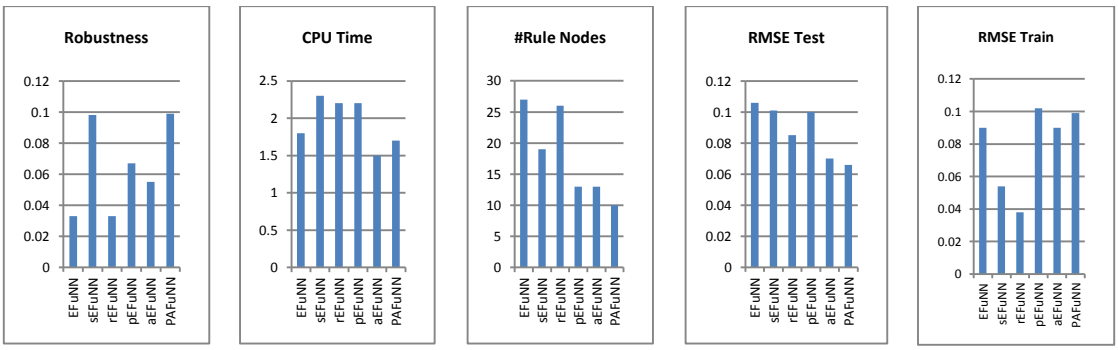


Fig. 2 Comparing EFuNN, sEFuNN, rEFuNN, pEFuNN, aEFuNN and PAFuNN on Gas furnace data set

According to the results, rEFuNN consists of small RMSE during test time phase; nevertheless, the number of the Rule Nodes for this network which indicates the network complexity is remarkable. Consequently, the CPU Time level is high for it. rEFuNN robustness has the lowest level among other approaches which signifies the fact that a modification in the network architecture would not result in an early appropriate repair. Although sEFuNN has a high level of robustness reaching approximately maximum, it comprises the highest CPU duration. On the other hand, the distinction between RMSE regarding the training period and testing time is too much which represents network over-fitting on the training data. The highest level of the Rule Nodes belongs to EFuNN which demonstrates network complexity. The reason is the lack of proper strategic method to control the number of the generated Rule Nodes for this method. Moreover, the highest level of the testing phase and the lowest level of the robustness are produced through this network. As it is obvious in the obtained results, the proposed PAFuNN is an appropriate way to extract rules from the input data inasmuch as it leads to a balance via all the distinct criteria. In general, the overall results illustrate the fact that PAFuNNs are more rapid, more controllable, and less complex; nonetheless, they are comparable with other networks in terms of the accuracy and robustness of the obtained results.

Table 2. The effects of choosing different learning automata in PAFuNN on Gas furnace dataset

Learning Automata	RMSE Train	RMSE Test	#Rule Nodes	CPU Time (Train-Test)	Robustness (CPU Time-RMSE-#Rule Nodes)
Tsetline for SThr Tsetline for EThr	0.11	0.133	13	1.0-0.2	1/(1.8+0.119+8)=0.100
Krylov for SThr Tsetline for EThr	0.067	0.105	19	1.4-0.2	1/(2.4+0.133+8)=0.094
Krylov for SThr Krylov for EThr	0.095	0.11	14	1.2-0.2	1/(2.6+0.087+12)=0.068
Tsetline for SThr Krylov for EThr	0.105	0.134	11	1.0-0.2	1/(2.3+0.102+12)=0.069
Krinsky for SThr Krylov for EThr	0.092	0.11	12	1.2-0.2	1/(2.6+0.080+13)=0.063
Krinsky for SThr Krinsky for EThr	0.097	0.127	14	1.2-0.2	1/(2.5+0.096+17)=0.051
Krinsky for SThr Tsetline for EThr	0.10	0.12	13	1.2-0.2	1/(2.0+0.127+8)=0.098
Tsetline for SThr Krinsky for EThr	0.11	0.11	20	1.1-0.2	1/(2.1+0.107+13)=0.065
Krylov for SThr Krinsky for EThr	0.103	0.130	13	1.2-0.3	1/(2.3+0.119+14)=0.060

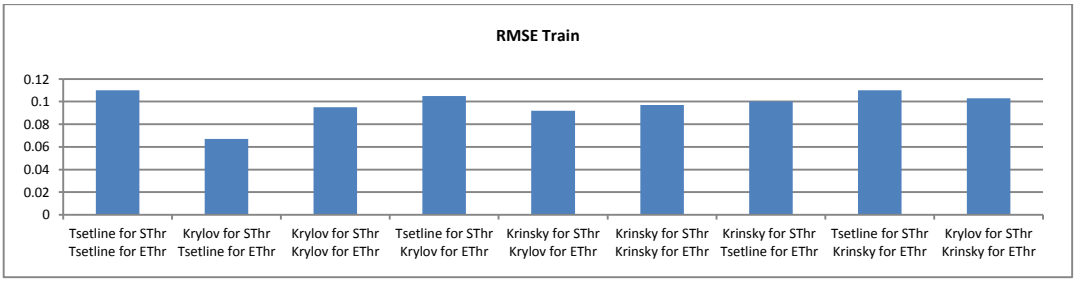
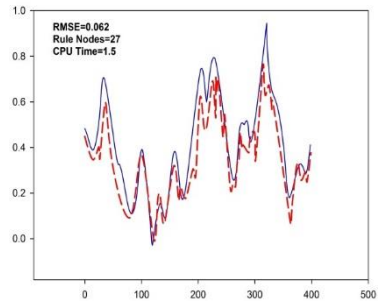
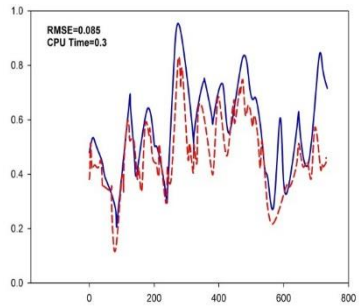


Fig. 3 The effects of choosing different learning automata in PAFuNN on furnace dataset

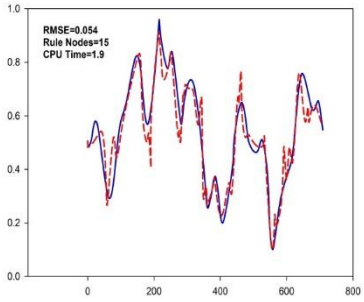
An appropriate selection of learning automata for both SThr and EThr parameters are effective in resulting in acceptable accuracy and complexity as Table 2 indicates. As the results show, utilizing Krylov learning automata for sensitivity threshold and Tsetline for Error threshold leads to the minimum level of RMSE throughout the training phase; however, the number of the Rule Nodes generated through this network and the CPU Time are too high. On the other hand, the distinction between RMSE in the testing and training phases is too high which shows the occurrence of over-fitting on the training data. Although selecting Tsetline learning automata for sensitivity threshold and Krylov for Error threshold leads to the minimum level of the number of the Rule Nodes, it possesses the highest level in RMSE for the testing phase. Utilizing Tsetline learning automata for sensitivity threshold and Tsetline for Error threshold leads to the most robust network; however, the RMSE of this network is high and approaches maximum level for both the training and testing phases. Although the testing RMSE is low in utilizing Tsetline learning automata for sensitivity threshold and Krinsky for Error threshold, the number of the Rule Nodes is the highest for it which means that both sensitivity and error thresholds are set at a small value for this network and a novel Rule Node will be generated for most of the individuals utilizing this network. Selecting Krinsky learning automata for sensitivity threshold and Krylov for Error threshold can be considered as the best choice



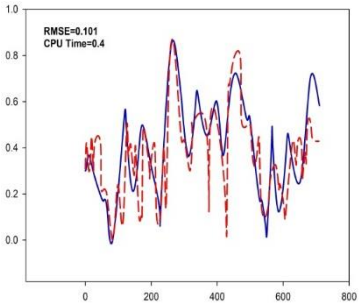
(a) The desired versus the predicted one step ahead value by EFuNN when it trained on the first half of the data



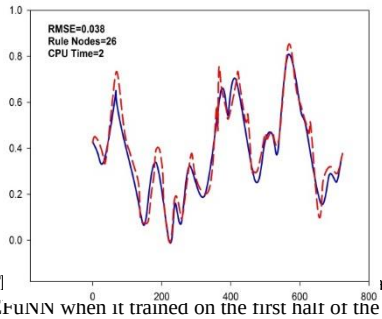
(b) The trained from EFuNN is tested on the second half of dataset



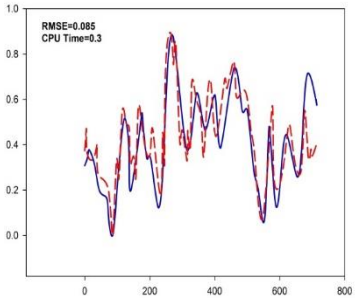
(c) The desired versus the predicted one step ahead value by sEFuNN when it trained on the first half of the data



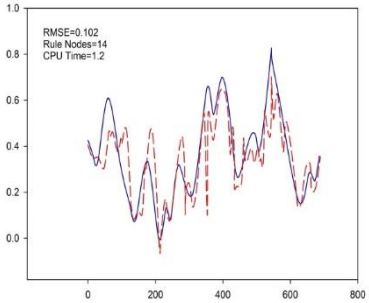
(d) The trained from sEFuNN is tested on the second half of dataset



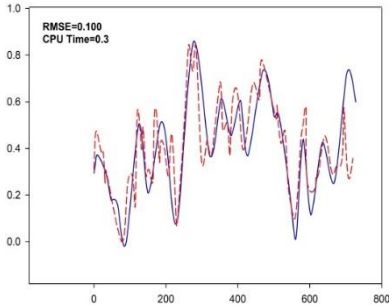
(e) Th d value by rEFuNN when it trained on the first half of the data set



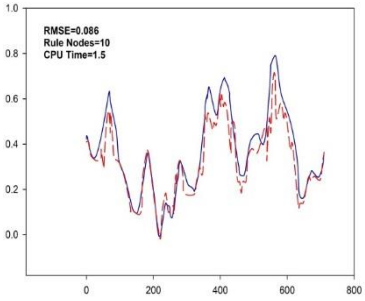
(f) The trained from rEFuNN is tested on the second half of dataset



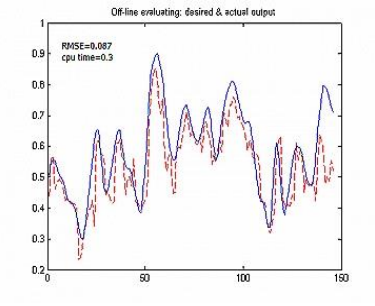
(g) The desired versus the predicted one step ahead value by pEFuNN when it trained on the first half of the data



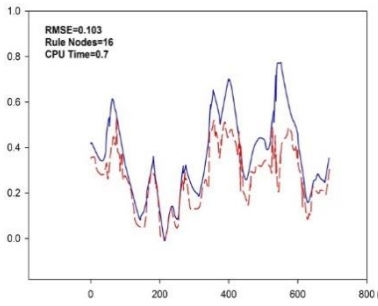
(h) The trained from pEFuNN is tested on the second half of dataset



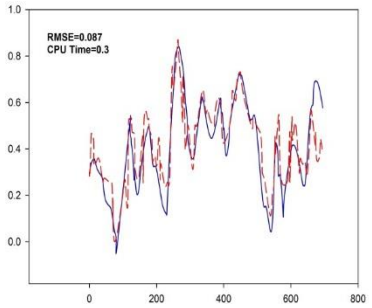
(i) The desired versus the predicted one step ahead value by aEFuNN when it trained on the first half of the data



(j) The trained from aEFuNN is tested on the second half of data set



(k) The desired versus the predicted one step ahead value by PAFuNN when it trained on the first half of the data



(l) The trained from PAFuNN is tested on the second half of dataset

Fig. 4 The process of EFuNN, sEFuNN, rEFuNN, pEFuNN, aEFuNN and PAFuNN evolving on the Gas furnace data set

Based on Figure 4, PAFuNN consists of more acceptable results in contrast to other networks especially in the testing phase. The reason why it is less accurate in the training phase compared with other networks is restoring refuted samples and generating Rule Nodes for them at the end of the training phase.

b) The Second Experiment: Classification

In this experiment, the capability of PAFuNN in classification task is assessed utilizing Iris dataset. Table 3 indicates the RMSE for the training and testing phases, the number of the Rule Nodes, the CPU Time throughout the training/testing phase, and the network robustness on basis of EFuNN, sEFuNN, rEFuNN, pEFuNN and aEFuNN. The same results are illustrated in Figure 5 for better digestion. Table 4 presents the effects of selecting various learning automata to adjust PAFuNN's thresholds. Figure 6 shows the same results. Note that the resulted values are average over 10 distinct runs. Figure 7 demonstrates the process of the networks evolving out of the Iris dataset. The real versus which is anticipated through the network values is presented when it is trained according to the first half of the Iris data (on-line, one-pass training). The evolved network is then tested in an off-line mode in the second half of the data.

Table 3. Comparing EFuNN, sEFuNN, rEFuNN, pEFuNN, aEFuNN and PAFuNN on Iris dataset

Algorithm	RMSE Train	RMSE Test	#Rule Nodes	CPU Time (Train-Test)	Robustness
EFuNN	0.099	0.488	8	1.3-0.2	1/(2.5+0.113+8)=0.094
sEFuNN	0.124	0.6	5	0.9-0.2	1/(2.0+0.196+19)=0.047
rEFuNN	NAN	NAN	11	6.0-0.4	1/(5.5+NAN+11)≈0
pEFuNN	0.106	0.100	4	1.1-0.2	1/(2.3+0.114+6)=0.118
aEFuNN (Krylov for SThr Krinsky for EThr)	0.145	0.075	6	0.7-0.2	1/(1.6+0.081+4)=0.176
PAFuNN	0.12	0.042	6	0.7-0.2	1/(1.8+0.085+5)=0.145

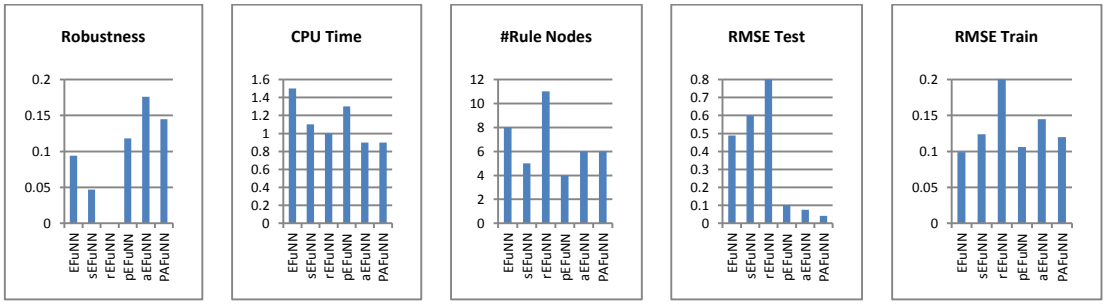
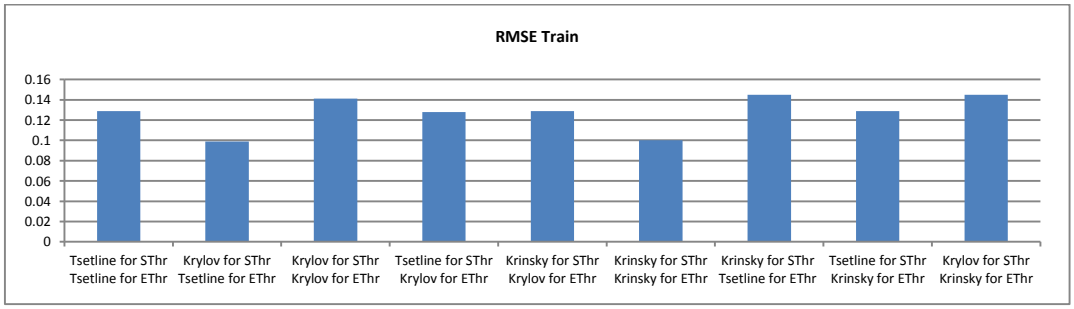


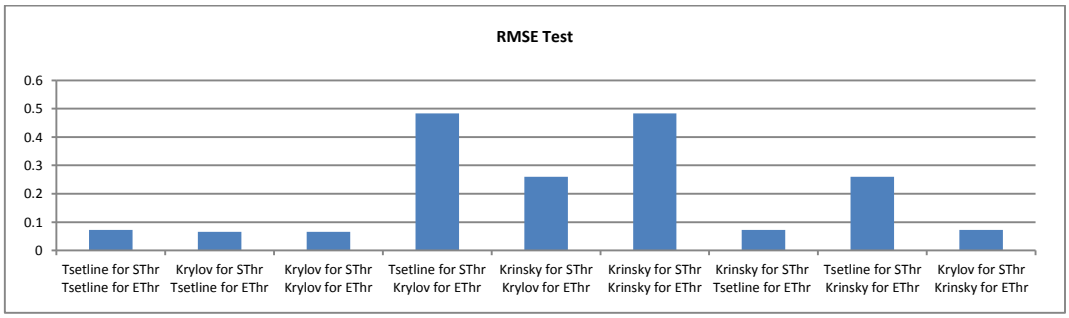
Fig. 5 Comparing EFuNN, sEFuNN, rEFuNN, pEFuNN, aEFuNN and PAFuNN on Iris data set

The results show that none of the EFuNN, sEFuNN and rEFuNN is appropriate for classification task based on Iris dataset. Over-fitting on the training data occurred to the first two networks. For rEFuNN, the RMSE is at a remarkably high level. Although pEFuNN has the lowest number of Rule Nodes, the RMSE is higher than aEFuNN and PAFuNN in the testing phase of this network. Furthermore, it is the least robust network in contrast with the two afore-mentioned ones. The most robust network is aEFuNN. Nevertheless, the testing RMSE in this network is higher than that of PAFuNN. The proposed PAFuNN has acceptable values for all its evaluating parameters in spite of having less complexity.

Table 4. The effects of choosing different learning automata in PAFuNN on Iris dataset

Learning Automata	RMSE Train	RMSE Test	#Rule Nodes	CPU Time (Train+Test)	Robustness
Tsetline for SThr Tsetline for EThr	0.129	0.0.072	5	1.4+0.4	1/(3.5+0.058+5)=0.11
Krylov for SThr Tsetline for EThr	0.099	0.066	10	1.5+0.2	1/(5+0.058+5)=0.099
Krylov for SThr Krylov for EThr	0.141	0.065	5	1.4+0.3	1/(2.0+0.060+5)=0.141
Tsetline for SThr Krylov for EThr	0.128	0.483	10	1.4+0.3	1/(2.6+0.106+5)=0.129
Krinsky for SThr Krylov for EThr	0.129	0.260	5	1.2+0.2	1/(2.5+0.144+5)=0.130
Krinsky for SThr Krinsky for EThr	0.100	0.483	10	0.9+0.3	1/(2.4+0.141+5)=0.132
Krinsky for SThr Tsetline for EThr	0.145	0.072	4	0.7+0.3	1/(1.8+0.075+4)=0.170
Tsetline for SThr Krinsky for EThr	0.129	0.260	5	1.2+0.3	1/(2.4+0.077+5)=0.133
Krylov for SThr Krinsky for EThr	0.145	0.072	4	0.7+0.3	1/(2.3+0.111+10)=0.080





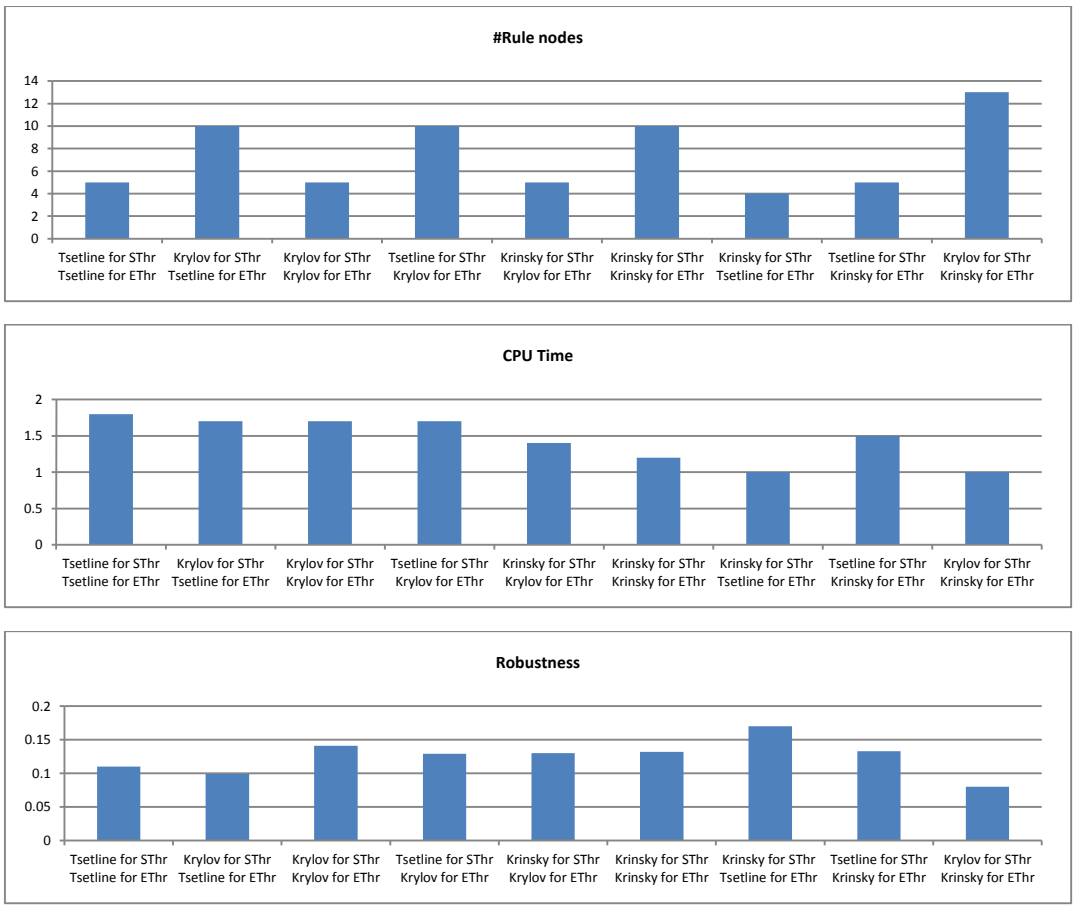
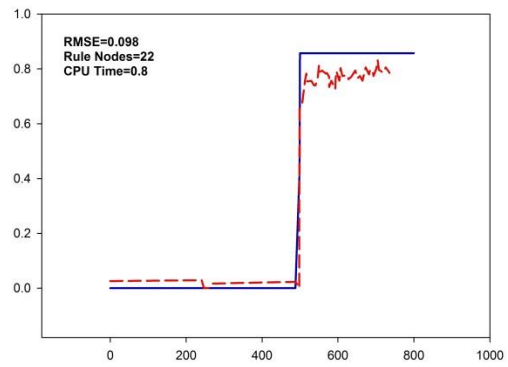
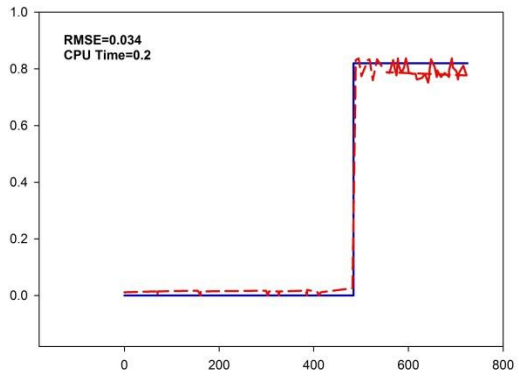


Fig. 6 The effects of choosing different learning automata in PAFuNN on Iris dataset

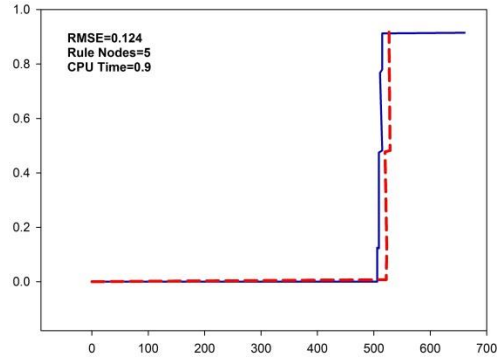
According to Table 4, a proper selection of learning automata for both SThr and EThr parameters is required so that they will result in acceptable accuracy and complexity. As the results show, utilizing Krylov learning automata for sensitivity threshold and Tsetline for Error threshold comprises the lowest amount of RMSE throughout the training time; however, it is the highest level regarding the number of the Rule Nodes. Although selecting Krylov learning automata for sensitivity threshold and Krinsky for Error threshold consists of the lowest number of Rule Nodes and CPU Time, it is the least robust network among others. Utilizing Krinsky learning automata for sensitivity threshold and Tsetline for Error threshold can be considered as the best option



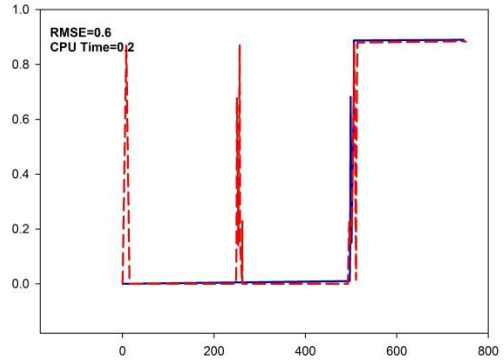
(a) The desired versus the predicted one step ahead value by EFuNN when it trained on the first half of the data set



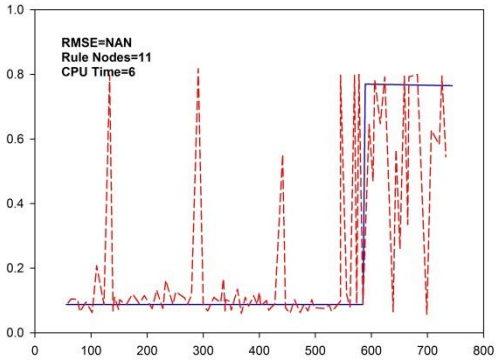
(b) The trained from EFuNN is tested on the second half of data set



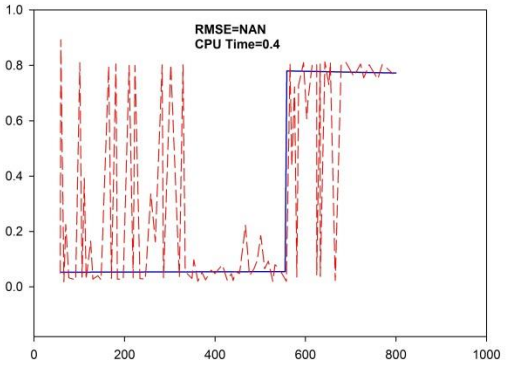
(c) The desired versus the predicted one step ahead value by sEFuNN when it trained on the first half of the data set



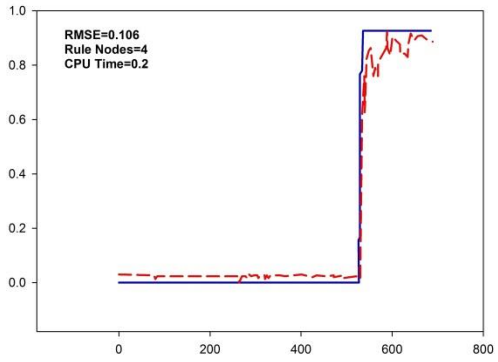
(d) The trained from sEFuNN is tested on the second half of data set



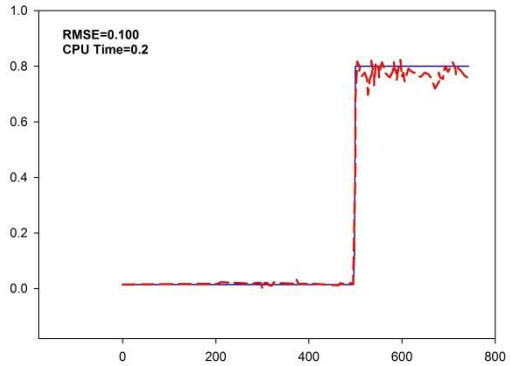
(e) The desired versus the predicted one step ahead value by rEFuNN when it trained on the first half of the data set



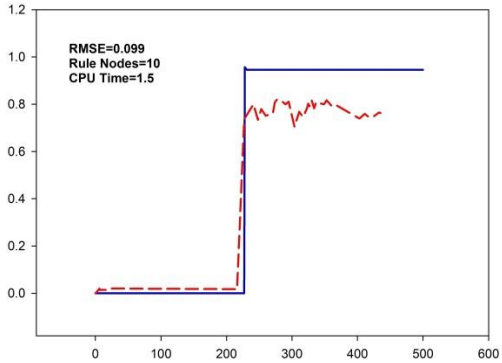
(f) The trained from rEFuNN is tested on the second half of data set



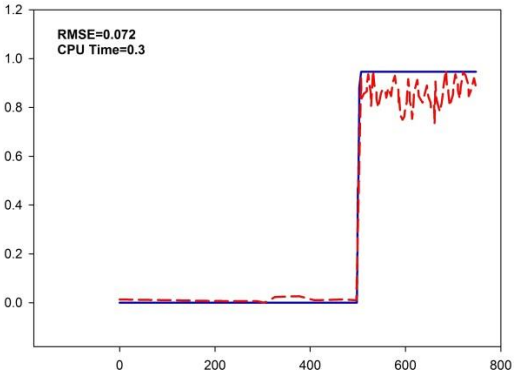
(g) The desired versus the predicted one step ahead value by pEFuNN when it trained on the first half of the data set



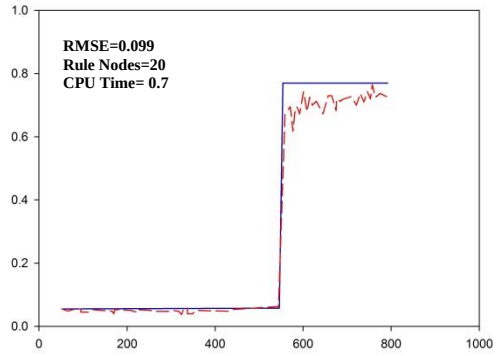
(h) The trained from pEFuNN is tested on the second half of data set



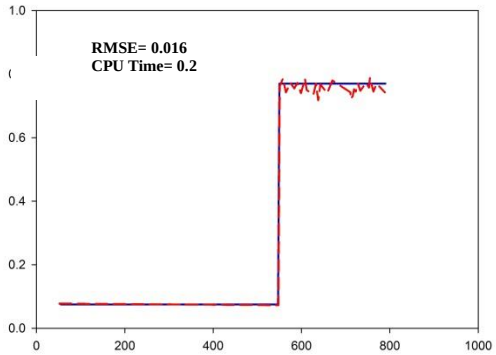
(i) The desired versus the predicted one step ahead value by aEFuNN when it trained on the first half of the data



(j) The trained from aEFuNN is tested on the second half of data set



(k) The desired versus the predicted one step ahead value by PAFuNN when it trained on the first half of the data



(l) The trained from PAFuNN is tested on the second half of data set

Fig. 7 The process of EFuNN, sEFuNN, rEFuNN, pEFuNN, aEFuNN and PAFuNN evolving on the Iris

As Figure 7 illustrates, PAFuNN generates more acceptable results in contrast to other networks especially in the testing phase. The reason why it has less accuracy in the training phase compared with other networks is restoring the refuted samples and producing Rule Nodes for them at the end of the training phase.

IV. Discussion

As illustrated through the paper, PAFuNNs are appropriate for on-line knowledge discovery of mega-databases. However, EFuNNs are appropriate for on-line learning; nevertheless, they have two important challenges which make them inapplicable for real world problems and large datasets which are as follows:

- There is no control over the number of the neurons added to the network throughout learning that results in a complex-structured network (a network with a large number of neurons and connections). In a complex neural network, there is a high probability of over - fitting the network on the input samples [30] which is a problem that is absolutely vital for noisy datasets.
- There are two thresholds in the EFuNNs which have direct impact on the network's accuracy and performance. An effective selection of these two parameters can improve the network performance considerably. In raw EFuNN, the parameters are set to fix values. In an enhancing self-tuning version, only the sensitivity threshold is updated according to network weights. Although this is more effective than the fix-valued threshold, this does not necessarily lead to the best value for the sensitivity threshold. On the other hand, no method is utilized to adjust the error threshold.

The PAFuNNs have the advantages of the EFuNNs. Nonetheless, they possess two important distinctions so as to overcome creating neurons based on a set of refuted samples and utilizing fixed structure learning automata in order to get adapted to the network challenges.

On the other hand, in the context of Neural Networks, a neural network is called as efficient if it can classify testing dataset with the minimum output error in an acceptable time. Considering "RMSE", and "CPU Time", we tried to analyze this property of the proposed network and illustrate that PAFuNNs can reach to the minimum RMSE in a short time. Furthermore, one of the most important challenges in Artificial Neural Networks is over-fitting networks on the training samples. One of the possible ways to avoid the problem is to prevent networks structures to be complicated during the training phase. By utilization of "Number of Rule Nodes" as a criterion for evaluating the complexity of network, we tried to demonstrate that our proposed network has lowered the probability of over-fitting. Moreover, the other important criterion for evaluating a neural network is its robustness for which we proposed a criterion that calculates how robust the network is facing with unpredictable changes (e.g. facing with noisy data that change network weights in a wrong direction).

Based on the results, PAFuNN consists of the lowest RMSE in the testing phase among other networks despite having a simple structure consisting of a lower number of neurons. This indicates that the implementation of the

population-based method decreases network complexity although it can increase its accuracy through reducing output errors. This occurs as a result of globally considering the input samples as close to a local view in order to add new neurons to the network which leads to the exclusion of ineffective nodes that are the causes of over-fitting on the training samples. This is obviously observed in comparing RMSE test and RMSE train for PAFuNN. Since utilizing the learning automata besides the population-based method steers the network toward its minimum output error value, it decreases the output error even more.

The results also demonstrate that our proposed method could successfully keep a balance in the discussed criteria. In other words, while it has declined error on testing samples in a short duration, it does not let the network to be complicated by controlling the number of neurons and as a result does not allow the occurrence of over-fitting.

V. Conclusion

In this paper, a novel connectionist model called PAFuNN is presented. PAFuNNs have a five-layer structure similar to that of EFuNNs, and are appropriate for on-line knowledge discovery of mega-databases. In this approach, after some samples are presented to the network and a definite number of Rule Nodes is created, if a sample does not match any of the existing Rule Nodes in the network, it will be stored, and regularly, some Rule Nodes are created based on a set of such samples unlike the EFuNN in which each neuron is created only according to a single-presented sample. Two fixed structure learning automata (FSLA) are interconnected to the network so as to adjust the sensitivity and error threshold parameters in order to enhance the entire performance of the system and minimize the network output error. A comparative study of PAFuNN and EFuNN on the basis of two benchmark datasets indicates that PAFuNNs are more rapid, more controllable, and less complex although they are comparable with EFuNNs in terms of the accuracy and robustness of the obtained results.

The proposed method is also applicable for new class of ECoS called evolving spiking neural networks (eSNN) [31,32,33] which evolve their structure and functionality in an on-line manner, from incoming information. The model uses trains of spikes as internal information representation rather than continuous variables. In other words, while the classical ECoS uses a simple sigmoid model of a neuron, the further developed evolving spiking neural network architecture uses a spiking neuron model for which similar ECoS principles and applications are applicable. eSNN architectures used both rank-order and time-based learning methods to account for spatio-temporal data [34]. In these networks, for every new input pattern, a new neuron that represents center of a cluster in the space of the synaptic weights is dynamically allocated and connected; accordingly, the population-based model presented in this paper can be used after each chunks of information to reduce the number of generated neurons and result to a less complex network. On the other hand, different parameters (i.e. the modulation factor, sensitivity profiles, and threshold θ) are utilized in eSNN which can be optimized using learning automata.

Our future works include using pruning and aggregation methods beside the proposed method to delete pointless Rule Nodes and improve the memory usage; furthermore the utilization of clustering methods such as K-means on the stored samples after each chunks of data, and using the cluster centers parameters for adding Rule Nodes to the network might cause a higher accuracy in the proposed method.

Acknowledgment

This work is supported by Research institute for ICT. The authors are grateful to anonymous referees of this paper for their constructive comments.

.

References

[1] Nikola Kasabov, "Evolving Fuzzy Neural Networks for Supervised/ Unsupervised On-line, Knowledge-based Learning" IEEE Trans. of Systems, Man and Cybernetics, vol.31, no.6, 2001, pp. 902- 918.

[2] Kenneth McGarry, Stefan Wermter, John MacIntyre, "Hybrid Neural Systems: From Simple Coupling to Fully integrated neural networks" Neural Computing Surveys, vol.2, no.1, 1999, pp. 62-93.

[3] Nikola Kasabov, "Evolving Fuzzy Neural Networks for On-line Knowledge Discovery" University of Otago, Information Science

Discussion Papers Series, 2001, http://hdl.handle.net/10523/1061 .

[4] A. Lotfi Zadeh, "Fuzzy Sets" Information and control, vol.8, no.3, 1965, pp. 338-353.

[5] Berndt Müller, Joachim Reinhardt, and Michael T. Strickland, "Neural Networks: An Introduction" Springer Science & Business Media, 2012.

[6] Nikola Kasabov, "Foundations of Neural Networks, Fuzzy Systems and Knowledge", Marcel Alencar, 1996.

[7] Xin Wang, "On-line time series prediction system—EFuNN-T" In Proc. Of the 5th Biannual Conf. on Artificial Neural Networks and Expert Systems, 2001, pp 82–86.

[8] Michael J. Watts, Nikola Kasabov, "Simple evolving connectionist systems and experiments on isolated phoneme recognition" In Proc. Of IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks, 2000, pp. 232–239.

[9] Michael J. Watts, "A decade of Kasabov's evolving connectionist systems: a review" IEEE Trans. on Systems, Man, and Cybernetics, Part C: Applications and Reviews, vol.39, no.3, 2009, pp.253- 269.

[10] Nikola Kasabov, "On-line learning, reasoning, rule extraction and aggregation in locally optimized evolving fuzzy neural networks" Neuro computing, vol. 41, no.1, 2001, pp. 25-45.

[11] Kumpati S. Narendra, Mandayam AL Thathachar, "Learning Automata: an introduction" Courier Corporation, 2012.

[12] Mohammad Reza Meybodi, Hamid Beigy, "New Learning Automata Based Algorithms for Adaptation of Backpropagation Algorithm Parameters" Neural Systems, vol. 12, no. 1, 2002, pp. 45-67.

[13] S. Mousavi, S., H. Rabiee, M. Moshref, A. Dabirmoghaddam, "Model Based Adaptive Mobility Prediction in Mobile Ad-Hoc Networks" In int. Conf. on Wireless Communications, Networking and Mobile Computing, 2007, pp. 1713- 1716.

[14] Samira Noferesti, Hamed Shah-Hosseini, "A Hybrid Algorithm for Solving Steiner Tree Problem" Computer Applications, vol.41, no.5, 2012, pp. 14-20.

[15] Hamid Beigy, Mohammad Reza Meybodi, Mohammad Bagher Menhaj, "Utilization of fixed structure learning automata for adaptation of learning rate in backpropagation algorithm" In Pakistan J. of applied sciences, vol.2, no.4, 2002, pp.437-443.

[16] Leonid Reznik, Vladimir Dimitrov, "Fuzzy systems design: social and engineering applications" Physica, vol. 17, 2013.

[17] Wael A. Farag, Victor H. Quintana, Germano Lambert-Torres, "A Genetic-based Neuro-Fuzzy Approach for Modeling and Control of Dynamical Systems" IEEE Trans. on neural networks, vol. 9, no.5, 1998, pp. 756-767.

[18] Jaesoo Kim, Nikola Kasabov, "HyFIS: adaptive neuro-fuzzy inference systems and their application to nonlinear dynamical systems" Neural Networks, vol.12, no.9, 1999, pp.1301-1319.

[19] Sung-Kwun Oh, Dong-Won Kim, Witold Pedrycz, "Hybrid fuzzy polynomial neural networks" Uncertainty, Fuzziness and Knowledge-Based Systems, vol.10, no.3, 2002, pp. 257 - 280.

[20] Krzysztof Simiński, "Neuro–rough–fuzzy approach for regression modelling from missing data" Int. J. Appl. Math. Comput. Sci., vol.22, no.2, 2012, pp. 461-476.

[21] Gary G Yen, Phayung Meesad, "An effective neuro-fuzzy paradigm for machinery condition health monitoring" IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics, vol.31, no.4, 2001, pp. 523-536.

[22] Krzysztof Simiński, "Analysis of new method of initialisation of neuro-fuzzy systems with support vector machines" Theorical and Applied Informatics, vol.24, no.3, 2012, pp. 243–254.

[23] Filipe Aires, Michel Schmitt, Alain Chedin, and Noëlle Scott, "The weight smoothing regularization of MLP for Jacobian stabilization" IEEE Trans. on Neural Networks, vol.10, no.6, 1999, pp. 1502-1510.

[24] Anna Esposito, Francesco Carlo Morabito, Advances in Neural Networks: Computational and Theoretical Issues, Edited by Simone Bassis, vol. 37, Springer, 2015.

[25] Dongjoo Park, Laurence R. Rilett, Gunhee Han, "Spectral basis neural networks for real-time travel time forecasting " transportation engineering, vol.125, no.6, 1999, pp.515-523.

[26] Vincent Vanhoucke, Andrew Senior, Mark Z. Mao, "Improving the speed of neural networks on CPUs" In Proc. Of Deep Learning and Unsupervised Feature Learning NIPS Workshop, vol. 1, 2011.

[27] Nikola Kasabov, Brendon Woodford, "Rule Insertion and Rule Extraction from Evolving Fuzzy Neural Networks: Algorithms and Applications for Building Adaptive, Intelligent Expert Systems"In Proc. Of IEEE Int. Fuzzy Systems Conf., vol.3, 1999, pp 1406-1411.

[28] Chia-Feng Juang, Ren-Bo Huang, Yang-Yin Lin, "A Recurrent Self-Evolving Interval Type-2 Fuzzy Neural Network for Dynamic System Processing" IEEE Trans. on Fuzzy Systems, vol.17, no.5, 2009, pp.1092-1105.

[29] Chia-Feng Juang, Yang-Yin Lin, Chiu-Chuan Tu, "A recurrent self-evolving fuzzy neural network with local feedbacks and its application to dynamic system processing" Fuzzy Sets and Systems, vol.161, no.19, 2010, pp. 2552-2568.

[30] Simon S. Haykin, Neural Networks and Learning Machines, Upper Saddle River: Pearson Education, 2009.

[31] Nikola Kasabov, Lubica Benuskova, Simei Gomes Wysoski, "A Computational Neurogenetic Model of a Spiking Neuron" In Proc. Of IEEE Int. Joint Conf. on Neural Networks, vol.1, 2005, pp.446-451.

[32] Stefan Schliebs, Nikola Kasabov, "Evolving spiking neural network—a survey" Evolving Systems, vol.4, no. 2, 2013, pp.87-98.

[33] Nikola Kasabov, Evolving Connectionist Systems: From Neuro-Fuzzy-, to Spiking-and Neuro-Genetic, Springer Handbook of Computational Intelligence, Springer Berlin Heidelberg, 2015, pp. 771-782.

[34] Nikola Kasabov, Kshitij Dhoble, Nuttapod Nuntalid, and Giacomo Indiveri, "Dynamic Evolving Spiking Neural Networks for On-line Spatio- and Spectro-Temporal Pattern Recognition" Neural Networks, vol.41, 2013, pp.188-201.