A pseudo entropy based self-organizing neural network for nonlinear system

Xin Feng, Jiangming Kan*

***Abstract*—**The structure of feed forward neural networks strongly influences their nonlinear function approximation results. This paper proposes a self-organizing neural network that can automatically adjust the number of hidden layers and alter the neurons of each hidden layer in accordance with training data. To ascertain the optimal neural network structure, the pseudo entropy of each hidden layer determines the number of neurons it contains, and the reduced mean square error of the entire neural network determines the number of hidden layers. A tuning optimization algorithm tunes the parameter weights and biases. Experimental results show that the proposed neural network outperforms the state-of-the art feed forward neural network and the proposed self-organizing algorithm is very effective for nonlinear function approximation.

***Keywords*—**self-organizing network; training optimization; pseudo entropy; nonlinear system

I. Introduction

Hyperparameter optimization is a developing topic in neural training research that has practical applications. In this field, a focus of concern is network architecture, particularly the optimal number of neurons and layers. The unwieldly neuro-evolution algorithm, the earliest hyperparameter optimization method [1] [2], carries obvious disadvantages in that it is inefficient and lacks flexibility.

Some researchers have recently adopted bionics methods and bioinformatics parameters for structure alteration. Although their experiments show progress in architecture adjustment, their application scenario is prohibitively simplistic [3]. In addition, several papers have addressed neuronal connectivity. Although such approaches accelerate training, the optimization of their results is difficult [4]. Cross-entropy is typically employed as a measure of error that influences the adjustment of weights and biases during training [5]. However, cross-entropy cannot represent information or ordered degree as architecture references.

According to universal approximation theory [6], a backpropagation(BP) neural network can approximate any nonlinear function; however, the architecture of a neural network greatly influences its model training process. Taken together with the aforementioned concerns, the architecture of a neural network growth and optimization algorithm must be more flexible and efficient.

This work is supported by the National Natural Science Foundation of China (Grant No. 31570713) and the Beijing municipal construction project special fund.

Xin Feng, Jiangming Kan are with the Beijing Forestry University, Beijing 100083, China. (corresponding author phone: 010-62336137-706; e-mail: kanjm@bjfu.edu.cn).

We propose a novel type of feed forward neural network, termed the growth entropy-neural network (GEN), with an efficient growth algorithm based on entropy (GAE). Both of these methods are based on the BP neural network with forward and backward processes. GEN is advantageous in several aspects. First, information entropy is introduced to improve neuronal development and the number of layers. Entropy is a reflection of information quantity. From an abstract point of view, the hidden layers of a BP neural network gradually reduce the weight influence of useless features in input data by mapping them into higher or lower dimensions via the number of hidden neurons and emphasizing essential features that lead to improved results.

Second, GAE addresses neural network architecture design, including both optimal neurons and the number of layers. The key adaptive growth process of GAE does not focus on pruning or connectivity; rather, it observes the information entropy change in each layer. Experimentally, entropy is very high if certain hyperparameters such as neurons and layers are unsuitable, leading to unsatisfactory results.

Third, the model convergence of a typical feed forward neural network is generally associated with low speed during training. Hence, optimization becomes mandatory. One reason for this sluggish convergence process is that backpropagation is prohibitively costly time-wise [7]. Thus, to retrieve the global minimum in a gradient descent process and overcome the weaknesses of BP neural networks, the tuning optimization (TO) method is proposed for acceleration and accuracy in this paper's experiments.

The structure of this paper is as follows. In the 2nd section, the main methodology of GEN is described. In the 3rd section, several experiments are described using the GEN model using different criteria to evaluate its performance. Additionally, the performance of GEN is compared with similar models. The 4th section gives the conclusion of the paper.

II. Growth Entropy-Neural Network

In this paper, the GEN model based on FNN is proposed. Compared with several recent models, the unique contributions of GEN are as follows:

(1) Automatically adjusting both the architecture and weights is key to the training process.

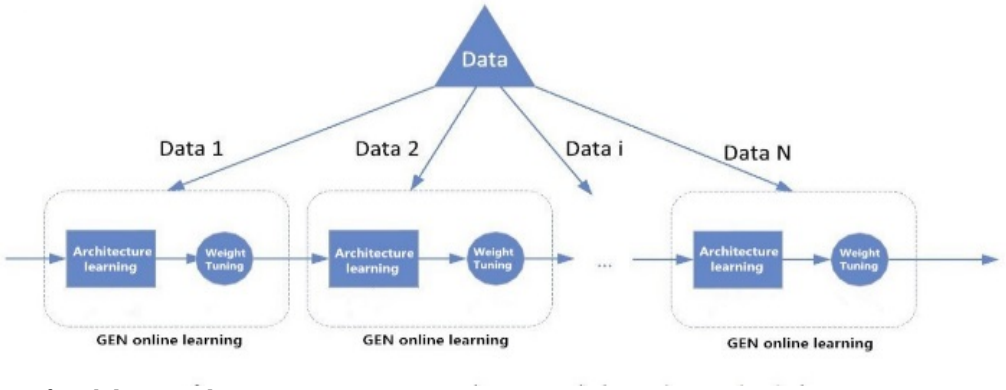

Fig. 1 Actual GEN procedure

(2) Using the information entropy of each layer to determine the neurons of the hidden layer and the reduced mean square error of the entire neural network to determine the number of hidden layers via GAE.

(3) Training with the TO acceleration algorithm.

A. Growth Algorithm based on Entropy (GAE)

In this paper, we propose a novel FNN that uses GAE to train both the parameters and architecture of a neural network. By introducing information entropy as a growth criterion to nonlinear function approximation, each layer is rendered adaptive for making net changes in architecture and mapping the input data into different dimensions.

During the training process, the input data are unordered compared with the required output. The hidden layers and neurons map the input into more ordered results layer by layer. From its definition, information entropy is a reflection of the ordered degree and quantity of information of each layer [4].

B. Neuronal growth phase

The architecture growth process includes the development of neurons and layers starting from an initial or previously experienced architecture. This growth process is designed for online learning [8], and its macroscopic process is shown in Fig. 1 above.

The definition of information entropy put forward by Shannon is a measure of information and disorder:

$$H(x) = -k\sum_{i=1}^{n} p_i \log p_i (i = 1,2,3...n) \quad (1)$$

Recently, researchers have proposed specific methods and related standards that use entropy for neural network pruning [9] [10] and made definite progress. In this paper, we use pseudo-entropy to represent the order degree for optimization as follows:

First, to calculate the pseudo entropy of each node, we require a method to calculate possibility. The output of each node is shown as follows:

$$O_i = Sigmoid(\sum_{k=1}^{n_j} W_{ki} X_k) \quad (2)$$

where O_i is the output value of the ith neuron in the jth hidden layer, W_{ki} is the weight between the kth input and the ith hidden neuron and X_k is the kth input value. The sigmoid function is: $sigmoid(x) = \left(1 + e^{-x}\right)^{-1}$.

A pseudo possibility definition for each node is proposed as:

$$p_i = O_i / \sum_{k=1}^{n_j} O_k \quad (3)$$

where n_j is the number of neurons in the jth layer. Thus, by regarding each layer as a system, each hidden layer's pseudo entropy is defined as:

$$H(x) = -\sum_{i=1}^{n_j} p_i \log p_i \quad (4)$$

If a layer's value of pseudo entropy H(x) is smaller than the threshold, E_0, the number of layer neurons should be raised by 1 because the ordered degree should be limited to a descending trend in hidden layer training.

C. Layer change phase

The training process of a BP neural network should be dynamic; thus, layer changes require a proper criterion. The layer increasing behavior of GEN is tied to the reduced mean square error (RMSE) and its adjustment trend. RMSE is defined as:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (y_i - y'_i)^2} \quad (5)$$

$$RMSE_0 \le RMSE \quad (6)$$

$$\frac{RMSE(t) - RMSE(t-1)}{RMSE(t-1)} \le \frac{RMSE(t-1) - RMSE(t-2)}{RMSE(t-2)} \quad (7)$$

where n denotes the amount of training data, and y_i and y'_i represent the predicted and target values, respectively. $RMSE_0$ is the threshold for RMSE, which performs poorly with the initial architecture. Thus, this represents a key step influencing the growth of layers at the beginning by Eq.(6). In addition, Eq.(7) represents a trend of RMSE decline; thus, when it is satisfied, the number of layers should be decreased by 1.

Pseudo entropy can be useful for explaining detailed significance during tuning. Each layer maps the last layer's input data into a higher or lower dimension to reduce the entropy and, compared with the prerequisite data, makes the input data more ordered until mapping helps the input become more closely approximated to the results.

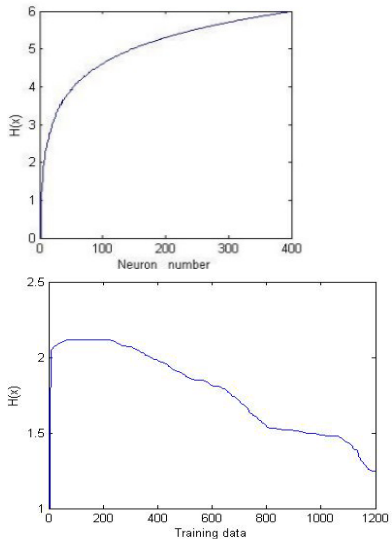

Fig. 2 Change of pseudo entropy under different situations

As a result, the entropy should decrease progressively. If not, the layer's mapping process is redundant, so it is imperative to delete abnormal layers. By observing the changes in Fig. 2, the correlation conditions of pseudo entropy, we see that the weight training process causes a layer's pseudo entropy to decrease while neuronal growth causes it to increase. The key steps of GEN are introduced in Table 1.

Table 1
The main sequence of GEN

%Initialize the architecture and weights as a normal distribution at the start
For input x(t) do
% Read the input
For all hidden layers **do**
% Neuronal growth
While entropy H(i)≤ $\boldsymbol{E_0}$ **do**
Hidden layer i's neurons plus 1, and the connecting weights between the new neuron and neurons in the other layers are all set to 0;
End
End
If RMSE≥ $RMSE_0$
The layers of neural network plus 1;
Else if the entire layers' pseudo entropy set decreases progressively
Delete the last layer, change the connectivity and set the new weights according to the normal distribution;
Else if meets the Eq.(7)
The layers of neural network plus 1;
End
Update the weights with the TO algorithm;
End

Remark 1: GAE is a strategy for neural network architecture optimization during training. The key point of GAE is to use pseudo entropy and RMSE as criteria to optimize network architecture. From previous experience, the peak value of the hidden layers is limited to 5 [11].

D. Tuning Optimization (TO)

The drawback of BP neural networks is that their calculation and weight optimization processes are extremely costly. Although the nets are fully-connected in this research field, GEN applies a new method for weight tuning optimization that simplifies this process. Recently, new methods have been proposed to optimize the calculation process. Some experiments use relative mutual information(RMI) [4] to achieve connectivity [12]; however, their calculation and weight tuning algorithms must be optimized.

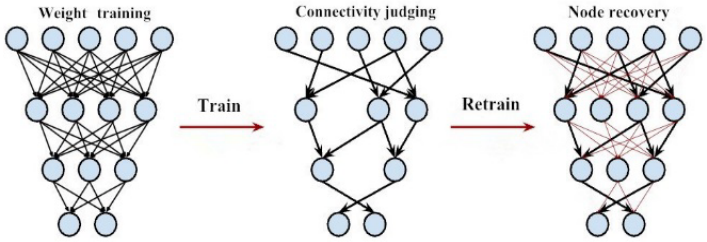

Fig. 3 The main process of the TO algorithm

As shown in Fig. 3, each connection in GEN signifies a feedforward and backward calculation, so it is important that nodes with low influence get the target value. GEN's tuning optimization (TO) method consists of three components: weight training, connectivity judging and node recovery, which are all determined by the weights' absolute values.

Table 2
Algorithm: Tuning Optimization Process

Weights Initialization: W(t) ~N(0,1)

Weights Training:
Normal backpropagation weight training for preset epochs
Get the weight set

Connectivity judging:
For all connectivity **do**
If |W(t)|< W_0
Delete the connectivity
Else if W_0≤|W(t)|<W_1
Save and ignore the connectivity while weight training in this phase
Else
Normal BP weight training
End
End

Nodes Recovery:
Recover all ignored connectivity
Train the weights normally

By analyzing the distribution of weights, regarding weights as important criteria for connectivity to adjust the neural network architecture, we can accelerate the training process, improve accuracy and avoid saddle points [13]. The TO algorithm overcomes the drawbacks of the dropout method and prevents the overfitting problem. W_0=0.05 and W_1=0.3 are the presets for all the experiments in this paper.

III. Experiments

To show the advantages of GEN, this section details three experiments for checking the efficiency of the GAE and TO algorithms. All the experiments are achieved in a MATLAB R2014b environment with a 3.4 GHZ i5-7500 CPU and 8 G of DDR4 RAM. Compared with similar algorithms for self-organizing architecture, the performance of GEN displays particular efficiency on certain complex problems. Examples 3.1-3.2 solve problems of classic nonlinear function approximation with different dimensions. Example 3.3 focuses

on a nonlinear system modeling problem. The performance measure uses mean squared error (MSE) and average percentage error (APE), which are defined as:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - y'_i)^2 \quad (8)$$

$$APE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - y'_i}{y'_i}\right| \times 100\% \quad (9)$$

where n is the input size and y_i and y'_i represent the true and prediction values, respectively.

A. 1-dimensional function input

To evaluate the efficiency of GEN, the following approximated function is given:

$$y = 0.2[1+0.1(x-7)^2]\cos 2x + 0.5e^{-2x}\sin(2x - 0.1\pi) \quad (10)$$

where x is randomly defined by a uniform distribution U[0,1]. This function is also used in [4] and [14] to demonstrate and examine the FNN's operational efficiency.

Table 3

Comparison of different self-organizing algorithms as described in section 3.1

Algorithm	Number of hidden neurons			CPU run time(s)	MSE	APE
	H1	H2	H3			
GEN	6	5	-	1.29	0.0014	0.0156
AANN	6	-	-	1.41	0.0113	0.0211
AGPNN	7	-	-	17.25	0.0341	0.0620
AMGA	6	-	-	30.12	0.0544	0.0989
CFNN	8	-	-	10.45	0.0114	0.0207
CNNDA	6	4	-	22.12	0.0124	0.0225

-These results are either unlisted in the original paper or non-significant.

The sizes of the training and test sets are both 200 in this experiment. By Kolmogorov theory [15], we define the initial architecture of GEN as follows: one input neuron, 1 output neuron and H1 with 2 neurons. The experimental results are compared with AANN (automatic axon-neural network) [4], AMGA (adaptive merging and growing algorithm) [16], CFNN (constructive feedforward neural network) [17], AGPNN (adaptive growing and pruning neural network) [18] and CNNDA (cascade neural network algorithm) [19]. The evaluation criteria for the experimental measurements are CPU run time, MSE and APE. A detailed comparison is shown in Table 3. Prior to training, the preset parameters are $RMSE_0$=0.005 and E_0=1.5.

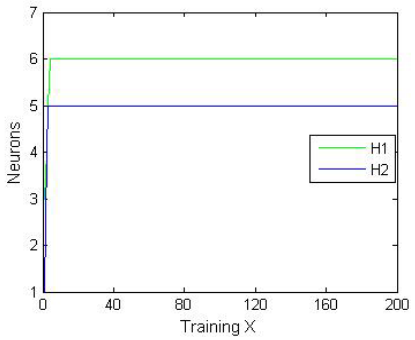

Fig. 4 Neuronal dynamic changes during training

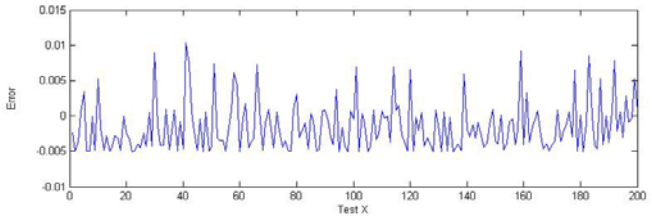

Fig. 5 Absolute error

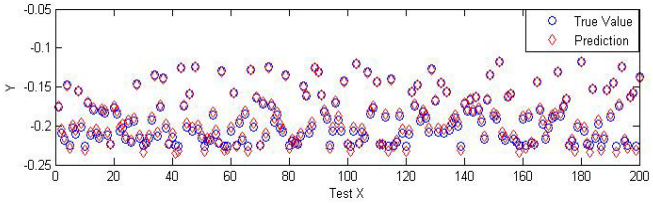

Fig. 6 Prediction results

In Fig. 4, the dynamic progression of GEN gradually becomes steady with two hidden layers. The final stable architecture of GEN is 1-6-5-1. Figs. 5-6 illustrate the error and sound performance of GEN in this experiment. Moreover, these results also show the superiority of the TO algorithm. In this experiment, GEN uses an additional layer to achieve higher accuracy, leading to an MSE ten times lower than the other algorithms, which is a clear improvement. However, most of the training time is saved via the cooperation of the two algorithms. GEN runs more quickly and efficiently than the majority of similar self-organizing neural network algorithms.

B. 2-dimensional function input

A more complicated example used to test the proposed FNN is the complicated interaction function (CIF), a 2-D function frequently used to test FNNs [16]:

$$y = 1.9(1.35 + e^{x_1}\sin(13(x_1 - 0.6)^2)\,e^{-x_2}\sin(7x_2)) \quad (11)$$

where x1 and x2 are randomly sampled by the normal distribution x1, x2 ~N[0,1] and the sizes of both the training and testing data are 400. The initial architecture of GEN is 2-3-1, which is identical to that of AANN, AMGA and CNNDA. The dynamic training process of GEN and its neuronal changes are shown in Fig. 7. In addition, its prediction and error results are shown in Figs.8-9. This effective demonstration shows outstanding performance on a 3-D model with surface fitting, which requires distinct analysis.

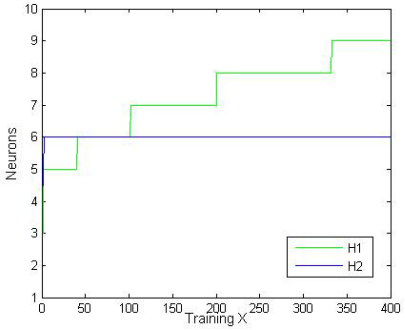

Fig.7 Dynamic neuronal changes during training

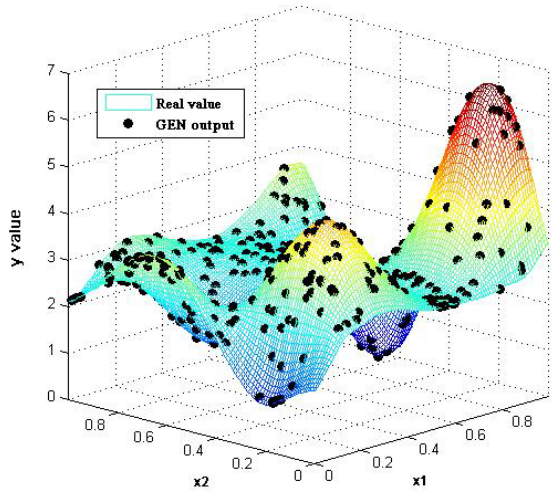

Fig.8 GEN approximation with a 2-D function

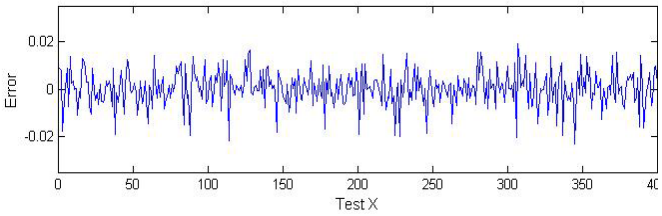

Fig.9 Absolute error

A results comparison is shown in Table 4 using the main standards of outcome measurements for FNNs. The final architecture of GEN is 2-9-6-1. This helps the model achieve relatively fast training and high accuracy. In Table 4, it is easily noted that, although the architecture is more complicated, it achieves lower error and run times, which is a reflection of GEN's improved efficiency. The preset parameters are $RMSE_0$=0.005 and E_0=1.5.

Table 4

Comparison of different self-organizing algorithms as described in section 3.2

Algorithm	Number of hidden neurons			CPU run time(s)	MSE	APE
	H1	H2	H3			
GEN	9	6	-	2.93	0.0102	0.0249
AANN	9	6	-	3.14	0.0113	0.0379
AGPNN	14	-	-	23.22	0.0210	0.0704
AMGA	10	-	-	39.12	0.0456	0.1529
CFNN	16	-	-	27.13	0.0156	0.0523
CNNDA	13	5	-	20.41	0.0121	0.0406

-These results are either unlisted in the original paper or non-significant.

C. Nonlinear dynamic system modeling

Another experiment used to test the efficiency of a neural network is classic nonlinear dynamic system modeling, which is referenced in many studies in the literature [4] [20] [21] for testing FNN performance. The function and model are described below:

$$y(t+1)=\frac{y(t)y(t-1)[y(t)+2.5]}{1+y^2(t)+y^2(t-1)}+u(t) \quad (12)$$

The model is identified in series-parallel mode defined as [22]:

$$y(t+1)=f(y(t),y(t-1),u(t)) \quad (13)$$

where y(0)=0, y(1)=0 and u(t)=sun(2πt/25).

In this experiment, the training set size is 500, sampled by the function above, and the domain of t is [1,500]. [501,700] is regarded as the test data set. (y(t), y(t-1), u(t)) are the inputs, including 3 elements, and the output is y(t+1). In addition, $RMSE_0$ =0.005 and E_0 =1.5. In this experiment, the initial architecture of GEN is 3-3-1. Fig.10 shows the neuronal changes that occur while GAE is running. The neurons tend to be steady with a final architecture of 3-10-6-1.

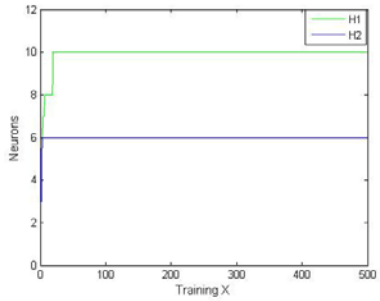

Fig.10 Neuronal changes during training

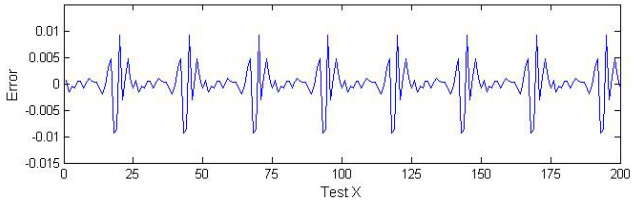

Fig.11 Absolute error

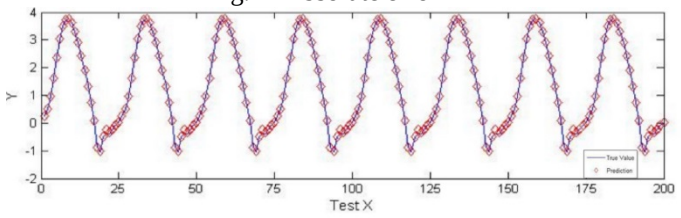

Fig.12 Prediction results

In Figs. 11-12, GEN exhibits solid performance and low error. A comparison with AANN, SOFNNGA, AMGA, CNNDA and SOFNN-ACA[21] is listed in Table 5, where the advantages of GEN are readily evident. Not only the architecture but also the training efficiency is improved with GEN. It reduces much time and error than most these recent similar methods. Particularly with this type of complex nonlinear system, GEN signifies huge potential with respect to the training processes of other neural network training methods, markedly for the optimization of deep neural network training.

Table 5

Comparison of different self-organizing algorithms as described in section 3.3

Algorithm	Number of hidden neurons	CPU run time(s)	MSE	APE

	H1	H2	H3			
GEN	10	6	-	11.16	7.26e-04	3.13e-03
AANN	10	-	-	13.31	1.08e-03	3.90e-03
SOFNNGA	8	-	-	76.23	2.30e-03	5.10e-03
AMGA	12	-	-	58.28	1.34e-02	6.07e-02
CNNDA	10	6	-	62.12	2.10e-03	-
SOFNN-ACA	6	-	-	16.24	1.05e-02	3.90e-03

-These results are either unlisted in the original paper or non-significant.

To summarize the nonlinear regression function approximation presented in sections 3.1-3.2, GEN's self-organizing strategy achieves relatively optimal architecture and improved efficiency on weight training with GAE, which avoids saddle points, leads to lower run time and has a better convergence [23], MSE and APE. Namely, the proposed GEN has upgraded performance in nonlinear function approximation compared with analogous self-organizing FNN models.

IV. Conclusion

This paper proposes a new self-organizing feed forward neural network called GEN that is based on entropy as a criterion for simultaneously adjusting both network architecture and weights during training. To make GEN function more efficiently, we also use a training algorithm called TO, which considers the significance of each connection to decide the training process and type. We demonstrate the application of GEN in different scenarios and its advantages and performance in comparison with several similar self-organizing algorithms.

The GEN can play an import role in nonlinear system modeling for its efficient and adaptive algorithm in accordance with training, which leads it to a wide application scene. Such as time series analysis and the deep learning model compression in deep learning. The more important is that the GAE algorithm can be transferred as the structure generation algorithm for other artificial neural network models like convolutional neural network(CNN) or deep belief network(DBN).

In summary, GEN is comparable to state-of-the-art self-organizing algorithms and the strategy's merits and novelties are summarized as follows:

(1) The adaptive structural algorithm is different from the former online strategies for it introduces a pseudo entropy as a measure of the order degree in each hidden layer to dynamically decide the amount of hidden neurons.

(2) To develop the efficiency of dynamic learning process, this paper uses GAE algorithm, which regards weights as the connection importance, as an optimizing method which helps the network costs less sources and higher accuracy.

(3) This efficient neural network model has great potential for future research in more diverse or complex environments such as fully-connected layer compressed computation in deep neural networks and other synthetic models.

References

[1] Wierstra D, Gomez F J. Modeling systems with internal state using evolino[C]// Genetic and Evolutionary Computation Conference, GECCO 2005, Proceedings, Washington Dc, Usa, June. DBLP, 2005:1795-1802.

[2] Han H G, Qiao J F. Adaptive computation algorithm for RBF neural network[J]. IEEE Transactions on Neural Networks & Learning Systems, 2012, 23(2):342-347.

[3] Tsodyks M, Gilbert C. Neural networks and perceptual learning[J]. Nature, 2004, 431(7010):775-781.

[4] Han H G, Wang L D, Qiao J F. Efficient self-organizing multilayer neural network for nonlinear system modeling[J]. Neural Networks, 2013, 43(7):22-32.

[5] Kline D M, Berardi V L. Revisiting squared-error and cross-entropy functions for training neural network classifiers[J]. Neural Computing & Applications, 2005, 14(4):310-318.

[6] Scarselli F, Tsoi A C. Universal Approximation Using Feedforward Neural Networks: A Survey of Some Existing Methods, and Some New Results[M]. Elsevier Science Ltd, 1998, 11(1):15.

[7] Ferrari S, Jensenius M. A Constrained Optimization Approach to Preserving Prior Knowledge During Incremental Training[J]. IEEE Transactions on Neural Networks, 2008, 19(6):996-1009.

[8] Chao J, Li X, Kang W, et al. Adaptive control of nonlinear system using online error minimum neural networks☆[J]. Isa Transactions, 2016, 65:125.

[9] Geok, See Ng and Abdul Rahman, Abdul Wahab and Shi, Daming. Entropy learning and relevance criteria for neural network pruning[J]. International Journal of Neural Systems, 2003, 13(5):291-305.

[10] Abid S, Chtourou M, Djemel M. Pseudo-Entropy Based Pruning Algorithm for Feed forward Neural Networks[J]. Australian Journal of Basic & Applied Sciences, 2013.

[11] Briggs F, Callaway E M. Laminar Patterns of Local Excitatory Input to Layer 5 Neurons in Macaque Primary Visual Cortex[J]. Cerebral Cortex, 2005, 15(5):479.

[12] Wan L, Zeiler M, Zhang S, et al. Regularization of neural networks using dropconnect[C]// International Conference on Machine Learning, 2013:1058-1066.

[13] Han S, Pool J, Narang S, et al. DSD: Dense-Sparse-Dense Training for Deep Neural Networks[C]// International Conference on Learning Representations, 2016.

[14] Narasimha P L, Delashmit W H, Manry M T, et al. An integrated growing-pruning method for feedforward network training[J]. Neurocomputing, 2008, 71(13-15):2831-2847.

[15] Eswaran K, Singh V. Some Theorems for Feed Forward Neural Networks[J]. International Journal of Computer Applications, 2015, 130.

[16] Islam M M, Sattar M A, Amin M F, et al. A new adaptive merging and growing algorithm for designing artificial neural networks[J]. IEEE Transactions on Systems Man & Cybernetics Part B, 2009, 39(3):705-722.

[17] Ma L, Khorasani K. Constructive feedforward neural networks using Hermite polynomial activation functions[J]. IEEE Transactions on Neural Networks, 2005, 16(4):821.

[18] Hsu C F. Adaptive growing-and-pruning neural network control for a linear piezoelectric ceramic motor[J]. Engineering Applications of Artificial Intelligence, 2008, 21(8):1153-1163.

[19] Islam M M, Murase K. A new algorithm to design compact two-hidden-layer artificial neural networks[J]. Neural Networks, 2001, 14(9):1265-1278.

[20] Zhang C, Wu W, Chen X H, et al. Convergence of BP algorithm for product unit neural networks with exponential weights[J]. Neurocomputing, 2008, 72(1–3):513-520.

[21] H Han, XL Wu, JF Qiao. Nonlinear systems modeling based on self-organizing fuzzy-neural-network with adaptive computation algorithm[J]. IEEE Trans Cybern, 2014, 44 (4):554-564

[22] H Han, L Zhang, X Wu, J Qiao. An Efficient Second-Order Algorithm for Self-Organizing Fuzzy Neural Networks[J]. IEEE Transactions on Cybernetics, 2017, PP (99) :1-13

[23] G Rajchakit, R Saravanakumar, CK Ahn, HR Karimi. Improved exponential convergence result for generalized neural networks including interval time-varying delayed signals[J]. Neural Networks,

2017, 86 :10-17

Xin Feng born in 1993, got Bachelor's degree in Zhengzhou University, Zhengzhou, China, at 2013~2017. At present, he is a postgraduate at Beijing Forestry University, Beijing, China. His research is concentrating on deep learning and computer vision.

Jiangming Kan born in 1976, Ph.D. degree, professor at Beijing Forestry. His research interests include computer vision and intelligent control.