International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:1, No:9, 2007

Learning Flexible Neural Networks for Pattern Recognition

A. Mirzaaghazadeh, H. Motameni, M. Karshenas, and H. Nematzadeh

***Abstract*—**Learning the gradient of neuron's activity function like the weight of links causes a new specification which is flexibility. In flexible neural networks because of supervising and controlling the operation of neurons, all the burden of the learning is not dedicated to the weight of links, therefore in each period of learning of each neuron, in fact the gradient of their activity function, cooperate in order to achieve the goal of learning thus the number of learning will be decreased considerably.

Furthermore, learning neurons parameters immunes them against changing in their inputs and factors which cause such changing. Likewise initial selecting of weights, type of activity function, selecting the initial gradient of activity function and selecting a fixed amount which is multiplied by gradient of error to calculate the weight changes and gradient of activity function, has a direct affect in convergence of network for learning.

***Keywords*—**Back propagation, Flexible, Gradient, Learning, Neural network, Pattern recognition.

I. Introduction

VARIOUS researches have been done by different researchers to recognize patterns. [2,7,8,12,13,14] Most of these projects are based on having all the learning patterns before solving the classification problem to design the algorithm. But in this article we specifically used learning the gradient of activity function and other parameters to recognize the pattern.

The operation of neurons is related to each other .since the initial weight for links among neurons is usually selected randomly, predicting the behavior of each neuron is impossible individually, even if a certain network is taken stable against various instigations [11].

In most networks, the principle of learning a network is based on minimizing the gradient of error [9,10]. Therefore it is assumed that a network has a minimum error at the end of learning process [2] but it is not always happened like this. Sometimes because of the largeness of the domain of changes of the input network signal, the activity function of some neurons will be saturated and at last the output of these categories of neurons will be fixed in their border amount. It can make a same situation for the next layers of neurons. With continuing this situation, the network will be in a stable mode. In this case the output of neurons will be fixed and continuing learning is not useful because the network is trapped at a minimum position as a cure we can teach the neurons activity function gradient like links weight.

Among neurons activity functions sigmoid function (one_directed & two_directed) has the most application, therefore for studying the mathematical form of the network we provide our equations based on this function [2,4,6]. Designing a neural network which is used error back propagation algorithm is not only a science but also an experimental work. The reason is that many factors are engaged in designing a network which are the results of researcher's experiences however with considering some matters we can lead the back propagation algorithm to better performance [1,3,5].

II. The Structure of Neuron Model

The model of a network comprises analog cells like neuron. Fig. 1 shows an instance of these cells which are used in a network. This multi layer hierarchal network is made of lots of cell layers. In this network there are forward and backward links between cells. If this network is used for recognizing the pattern in this hierarchy, forward signals handle the process of recognizing pattern whereas backward signals handle the process of separating patterns and reminding. We can teach this network to recognize each set of patterns. Even being extra instigators or lack in patterns, this model can recognize it. It is not necessary that the complete reminding recognize manipulated shapes or the shapes that are changed in size or convert the imperfect parts to the main mode.

A. Mirzaaghazadeh and H. Motameni are with Department of Computer, Islamic Azad University, Sari Branch, Iran (e-mail: mir937n@yahoo.com, motameni@iausari.ac.ir).

M. Karshenas and H. Nematzadeh are with Department of Computer, Mazandaran University of Science & Technology, Iran (e-mail: mehrdad_karshenas2000@yahoo.com, hossein061@yahoo.com).

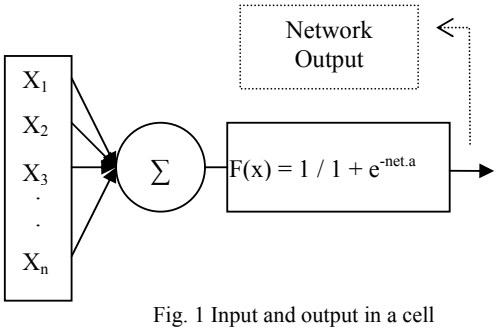

Fig. 1 Input and output in a cell

A. Activity Function

Activity function is a nonlinear function that when it is exerted to the pure input of neuron, its output determine the neuron .their domain is usually all the real numbers. Theoretically speaking there is no limitations on the pure amount of input. (Practically with limiting the weights we can limit the pure input simply and usually it is done like this, however they have almost unlimited domain). The range of activity function is usually limited. The scopes [0, 1] or [-1, 1] are the common scopes. The primary neural models that had *perceptron* used threshold simple function as an activity function. In this case if sum of inputs' weight is less than threshold the output of neuron is zero otherwise it will be one. If the activity function is derivable lots of benefits are attained that the threshold function doesn't have them. The most current models use sigmoid function. Sigmoid function is a continuous function with the domain of real numbers. And its derivation is always affirmative and its range is unlimited. The commonest types of sigmoid functions are USF and FUSF.

USF:

$$F(x) = 1 / 1 + e^{-net.a}$$

FUSF:

$$F(x) = 2|a| / 1 + e^{-2net.|a|}$$

The parameter a determines the gradient of the function in the way that the amounts which are smaller than a presents smoother mode of function with the passage area from the low limit of function to its upper limit and the amounts which are bigger than a makes this passage nearer to the stair function mode. One of the benefits of this function is that its derivation is simply calculable.

III. Back Propagation Algorithm

Multi layers networks from learning with the supervisor are used to solve various matters and they are successful. Learning the network is done through a common learning algorithm in many layers which is named error back propagation algorithm. This algorithm is based on error correction learning rule.

Basically the process of error back propagation has two passages through network layers: a forward passage and a backward passage. In a forward passage a pattern like an input vector is exerted to the network input neurons and its affect propagates in the network layer by layer. At last a set of outputs are fixed as a real response of a network. But through the backward stage the weights will be regulated based on the error correction rule. The error signal is produced through subtracting the real response from the desired (target) response. This error signal is regulated from the last layer into the network in a backward way to bring the network response nearer to the desired response. We define:

$e_j(n) = d_j(n) - y_j(n)$

$E(n) = 1/2 \sum e_j^2(n)$

$E_{av} = 1/N \sum_1^N E(n)$

Which n defines repetition number and N is the patterns number. So we have:

$net_j(n) = \sum_1^p w_{ji}(n)\, y_j(n)$

$y_j(n) = f_j(net_j(n))$

$\partial E(n) / \partial W_{ji}(n) = (\partial E(n) / \partial e_j(n))\, (\partial e_j(n) / \partial y_j(n))\, (\partial y_j(n) / \partial net_j(n))\, (\partial net_j(n) / \partial w_{ji}(n))$

$\Delta w_{ji}(n) = -\eta\, (\partial E(n) / \partial w_{ji}(n))$

$\Delta w(n) = -\eta_w\, (\partial E(n) / \partial w_{ij}(n))$

$\Delta a_j(n) = \eta_a\, (\partial E(n) / \partial a_j(n))$

$\partial E(n) / \partial a(n) = (\partial E(n) / \partial e_j(n))\, (\partial e_j(n) / \partial y_j(n))\, (\partial y_j(n) / \partial a_j(n)) = -e.f^{*}(a,x)$

IV. Affection of the Selected Weights in Number of Learning

We studied the affection of the selected weights in number of learning repetition through a computer program which is designed by ourselves in two following modes:

A. The affection of the selected weights' range in number of learning repetition while USF is selected as an activity function and only the weights of network is under the process of learning. In this learning E_{av} =0.001 and η=1 and network is made of just an input layer and an output layer. This affection is shown in Table I and Fig. 2.

TABLE I
The Affection of the Network Weight on the Number of Learning Repetition with USF

Weight Range	0 – 0. 1	0 – 0.01	0 – 0.001
Learning Repetition	1930	1830	1830

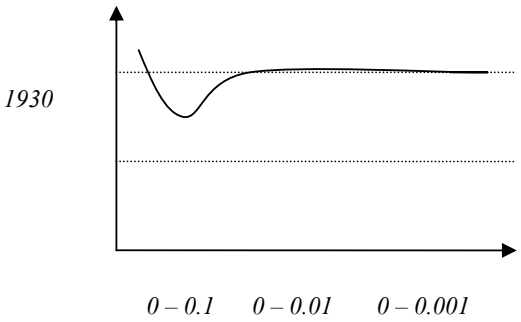

Fig. 2 The affection of the weights' range on the number of repetition with USF and learning through just weights

B. The affection of the selected weights in number of learning repetition while FUSF is selected as an activity function and parameters (w, a) have been taught. In this learning $E_{av} = 0.001$ and $\eta_1 = 1$ and $\eta_2 = 0.1$. This affection is shown in Table II and Fig. 3.

TABLE II
THE AFFECTION OF WEIGHTS' RANGE ON THE NUMBER OF LEARNING REPETITION WHILE THE GRADIENT OF THE ACTIVITY FUNCTION WITH THEIR WEIGHTS HAVE BEEN TAUGHT

Weight Range	***Learning repetition***
0 – 0.1	3360
0 – 0.01	970
0 – 0.001	870
0 – 0.0001	860
0 – 0.00001	860

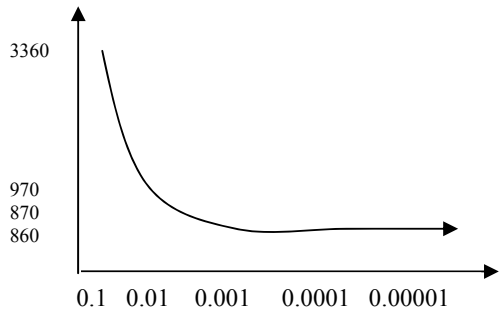

Fig. 3 The affection of weights' range on the number of learning repetition while the gradient of the activity function with their weights have been taught

V. The Affection of η_1 in the Number of Learning Repetition

We studied the affection of η_1 in the number of learning repetition through a computer program which is designed by ourselves in two following modes:

A. USF Activity Function

E = 0.001, W = [0, 0.001], a =1.

This affection is shown in Table III and Fig. 4.

TABLE III
THE AFFECTION OF η_1 OF NETWORK ON THE NUMBER OF LEARNING REPETITION WHILE ONLY THE WEIGHTS HAVE BEEN TAUGHT

η_1	1	2	3	4	5	6
Learning Repetition	1930	920	530	320	920	3570

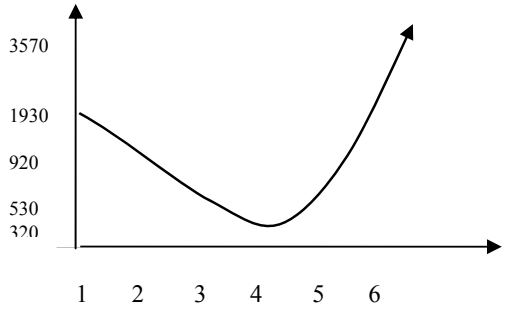

Fig. 4 The affection of η_1 of network on the number of learning repetition while only the weights have been taught

B. FUSF Activity Function

E = 0.001, W = [0, 0.001], η_2=0.1. This affection is shown in Table IV and Fig. 5.

TABLE IV
THE AFFECTION OF η_1 ON THE NUMBER OF LEARNING REPETITION WHILE THE GRADIENT OF ACTIVITY FUNCTION WITH THEIR WEIGHTS HAVE BEEN TAUGHT

η_1	1	2	3	4	5	6
Learning Repetition	870	380	260	220	330	750

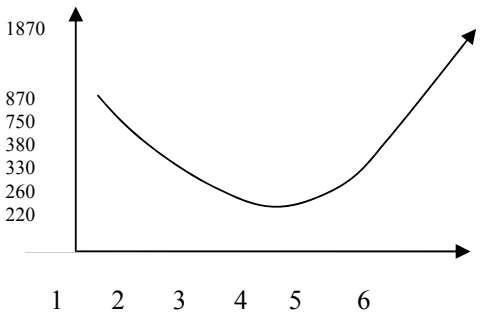

Fig. 5 The affection of η_1 on the number of learning repetition while the gradient of activity function with their weights has been taught

VI. Future Work

This article studied the affection of various parameters on the neuron model for learning the neural network. Learning the gradient of activity function with learning the weights makes the neural network to recognize the noisy patterns. This matter and the affection of the number of neural network's layers for recognizing noisy patterns will be studied in our future work.

VII. Conclusion

The operation of neurons is related to each other. Since the initial weight for links among neurons is usually selected randomly, predicting the behavior of each neuron is impossible individually. in designing the neural network which used back propagation , many factors are propounded which are the result of personal experiences of researchers however with considering some matters we can lead the back propagation algorithm to better performance. Initial selecting of weights, type of activity function, selecting the initial gradient of activity function and selecting a fixed amount which is multiplied by gradient of error to calculate the weight changes and gradient (slope) of activity function, has a direct affect in convergence of network for learning and recognizing the taught patterns after learning.

.

References

[1] A. Mirzaaghazadeh, H. Motameni, "Using Neural Network in Pattern Recognition", Proceeding of Iran Computer Conference, 2002.
[2] Kamarthi S.V., Pittner S. , Accelerating neural network traning using weight extrapolation, Neural networks, 9, 1999, pp. 1285-1299.
[3] A. Burak Goktepe, "Role of Learning Algorithm in Neural Network-Based Back calculation of Flexible Pavements", Journal of Computing in Civil Engineering, Volume 20, Issue 5, pp. 370-373 (September/October 2006).
[4] Manfred M Fisher, "Neural Networks: A General Framework for Non-Linear Function Approximation", Transactions in GIS, Volume 10 Page 521 – july 2006, doi:10.1111/j.1467-9671.2006.01010.x, Volume 10 Issue 4.
[5] V. Maiorov, "Approximation by neural networks and learning theory", Journal of Complexity, Volume 22, Issue 1, Februery 2006, Pages 102-117.
[6] Salvatore Cavalieri, "A novel learning algorithm which improves the partial fault tolerance of multilayer neural networks", Neural Networks, Volume 12, Issue 1, January 1999, Pages 91-106.
[7] Mohammad Teshnehlab and Keigo Watanabe (Eds.), "Intelligent control based on flexible neural networks", Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999, ISBN 0-7923 -5683-7, Automatica, Volume 38, Issue 3, March 2002, Pages 564-565.
[8] Edgar Rinast, HansDieter Weiss, "Neural network approach computer=assisted interpretation of ultrasound images of the gallbladder", Europen Journal of Radiology, Volume 17, Issue 3, November 1993, Pages 175-178.
[9] K. Economou and D. Lymberopoulos, "A new perspective in learning pattern generation for teaching neural networks", Neural Networks, Volume 12, Issue 4-5, June 1999, Pages 767-775.
[10] Eiji Mizutani and James W. Demmel, "On structure-exploiting trust-region regularized nonlinear least squares algorithms for neural-network learning", Neural Networks, Volume 16, Issue 5-6, June-July 2003, pages 745-753.
[11] R.Vicente Ruiz de angulo and Carme Torras, "Neural learning methods yielding functional invaiance", Theoretical Computer Science, Volume 320, Issue 1, 12 June 2004, Pages 111-121.
[12] Solanki Gautam, "Neural network and its application in pattern recognition", Seminar Report of Department of Computer Science and Engg. Indian Institue of Technology, Bombay, November 5, 2004.
[13] Alexander J. Faaborg, "Using neural networks to create an adaptive character recognition system", March 2002, http://web.media.mit.edu/~faaborg/research/cornell/hci_neuralnetwork_finalpaper.pdf
[14] O. Lezray, D. Fournier and H. Cardot, "Neural network induction graph for pattern recognition", Neurocomputing 57 (2004) 257-274.