A New Neural Network With Fuzzy Technique: Disease Diagnosis, A Case Study.

Ferdous Ahmed Sohel
Dept. of Computer Science and Engineering
International Islamic University Chittagong – Dhaka Campus
Dhanmondi, Dhaka, Bangladesh
Email: ferasohel27@yahoo.com

ABSTRACT

Neural networks and Fuzzy systems [1], [2], [3], [4] are being used in the field of artificial intelligence (AI). A hybrid neuro-fuzzy[5], [6] technique can made various implementations easier. Backpropagation [7], [8], [9] is a powerful algorithm in neural networks and also Kohonen [10] Neural Network has self-organizing power. In this paper I've used a neuro-fuzzy technique to recognize a system, which can detect some common diseases from their physical symptoms. I've used a combination of feedforward part of the backpropagation algorithm and Kohonen self-organizing neural networks to train the networks.

1. INTRODUCTION

Simple diseases say, *fever, cold, malaria, mumps, flu, measles* etc have some very common external symptoms like *temperature, runny nose, sneezing, headache, swollen of the salivary glands, cough, vomiting, rash on the skin* etc. These symptoms vary according to the disease. So we can use here a fuzzification technique here. Say, temperature, which is normally measured in degree Celsius. It may be 98.4^{o}F to 105 ^{o}F. We can differentiate this temperature into several categories (HIGH, MEDIUM, LOW), moreover there may be some modifiers (VERY, NOT). From these linguistic variables we can have fuzzy values using *if then else* rules. To detect these diseases we can use a neural network with fuzzy systems. Here I have proposed a neural network, which is a combination of back propagation and Kohonen neural network. The rest of the paper contains a few words about Fuzzy neural network in section two. Section 3 contains the proposed algorithm and its features. Section 4 contains experimental results along with the comparisons with results of similar algorithms and section 5 contains conclusion.

2. FUZZY AND NEURAL NETWORKS

There are AND and OR fuzzy neurons [11]. A three layer neural network can be built out of AND and OR neurons [11], where the hidden layer and output layer may be formed by using AND and OR neurons respectively and vice versa. But these sorts of neural networks are not strong enough in the situations where there are many output neurons (actually they are well fitted where there is only one output neuron).

Other neural network architectures are also possible – a three layer neural network can be used to simulate a situation when *n* fuzzy inputs are applied to *m* fuzzy inference rules [5]. But these are also providing single output.

Fuzzy systems may be viewed as a layered feedforward network [12], [13] and also backpropagation learning algorithm can be used to match input output pairs [12], [13]. The system uses the classical four step fuzzy control process of (1) scaling and fuzzification of the crisp input, (2) development of a fuzzy rule base, (3) fuzzy inference using the fuzzy rule base and (4) rescaling and defuzzification to give a crisp result or recommended action.

Kohonen [10] neurons are self-organizing, which means that these have the ability to learn without being given the output for an input pattern. They modify the weight just basing on the characteristics of the input patterns. Actually they work in a winner takes all fashion (Let's assume that the weight vectors are randomly distributed and then determine how close each neuron's weight vector is to the inputs vector. The neurons then compete for the privilege of learning. The neuron with the largest dot product of the input vector and a weight component is declared the winner (in case of ties one winner is randomly chosen), this neuron is the only neuron that will be allowed to generate an output signal; all other neuron outputs will be set to 0.). The Kohonen network has the potential for real time application learning, but as the network is smaller, the less accurate the network is. There may be some wild swing over the weights if the inputs and the weights are not well randomly distributed.

Counter propagation network [14], is a combination of Kohonen self-organizing network and Grossberg outstar network. This combination yields properties not available in either alone. It's a powerful neural network, which can classify patterns even if all the inputs are not complete. Also it has the ability to safe a lot of computing time but it suffers form the flaw of not much accurate.

3. THE PROPOSED ALGORITHM

The Proposed Neural network: I have proposed a neural net that will combine the features of both Feedforward part of the backpropagation algorithm and power of Kohonen neurons. The neural network has three layers. First layer is the input layer. The inputs are the fuzzy values. The second layer is a layer of neurons like in figure 1. The third layer is Kohonen layer (a Kohonen neuron is like figure 2). The network is highly connected (bilateral connection). The output is the output from the Kohonen layer. The complete network is shown in figure 3. Each output represents one of diseases. From the values in the

output layer (actually from the highest value) we can make decision about the disease.

Training of the Network: The following steps are taken while training the network:

i) Weight initialization.

ii) In the forward pass the output of the hidden layer is calculated.

iii) All these values are then normalized. These normalized values are taken as the input to the Kohonen layer.

iv) The output of the Kohonen layer is recorded. The neuron with the highest output value in the Kohonen layer is treated as the *winner neuron*. This neuron has a value 1 and all other neurons have the value 0. In case of ties all the winners are assigned 1.

v) In the backward pass the weights are adjusted. In the Kohonen layer only the winning neurons weights are adjusted. And in the hidden layer Backpropagation technique is used. The error term for this layer is calculated from the recorded output value and after winning value of the Kohonen output neuron.

For the Kohonen layer the weight adjustment is

$W_{new} = W_{old} + \Delta W$

Where $\Delta W = \eta[X - W_{old}]$

Here η is the learning rate constant with values [0.2 to 0.9], typically 0.7.

For the hidden layer weight adjustment is

$W_{new} = W_{old} + \Delta W$

Where $\Delta W = -\eta \sum \delta . Y$

Where $\delta = \varepsilon f'(v)$

Where $\varepsilon = \sum_k \delta_k . w_k$

Where f is the activation function and f′ is its first derivative.

vi) The network then can be trained until its Kohonen inputs and weight converge (are almost same).

Input Fuzzification:

I have collected the symptoms as statistical data. These data are then fuzzified and then posted to the input neurons. Some examples of fuzzification are listed in table 1.

Symptoms	Statistical information	Fuzzy Value
Temperature	98◦ F	0.2
	100◦ F	0.5
	102◦ F	0.6
	103◦ F	0.7
	104◦ F	0.85
	105◦ F	0.95
Cough	Normal	0.1
	Medium	0.5
	Severe	1

Table 1: Input fuzzification

Weight initialization: All weights are random numbers. The starting weights for the Kohonen layer should be small. And these weights should be normalized. For the hidden layers the weights are comparatively higher values.

Normal operation of Kohonen Layer: In its simplest form, the Kohonen layer functions in a winner take all fashion; that is for a given input vector one and only one neuron of the Kohonen neurons outputs a logical one, all other output a zero.

Associated with each Kohonen neuron is a set of weights connecting it to each input. The output is simply the summation of weighted inputs.

$n_j = \sum_i x_i . w_{ij}$

Or in vector notation

N = XW, Where N is the vector of Kohonen layer network outputs.

The Kohonen neuron with the largest N value is the "winner" and its output value is set to one; all others are set to 0.

Normal operations of the hidden layer: In the forward pass summation of weighted inputs is taken as, $nj = \sum_i x_i . w_{ij}$ then this value for each neuron is taken as the parameter of the activation function. The output of the activation function is normalized as follows,

$x_i = x_i / \sqrt{(x_1^2 + x_2^2 + \ldots \ldots + x_n^2)}$.

The normalized value is input to the Kohonen layer.

Backpropagation trains the hidden layer by propagating the adjusted error back through the network. The error term ε is calculated in the following manner,

ε = Recorded value – Winning value (of Kohonen layer).

Number of Neurons: The number of neurons in the input layer is equal to the number of collected physical symptoms. The number of output neurons is equal to the number of diseases in the experiment. And the number of hidden neurons is equal to the average of input and output neurons.

4. EXPERIMENTAL RESULTS

Diseases: *Fever, Yellow Fever, Normal Cold, Malaria, Mumps, Measles, Flu, Chicken Pox, and Whooping Cough.*

Symptoms: *Temperature, Runny Nose, Cough, Rash, Blood Eyes, Intolerance to Light, Stiff Joints, Sore Throats, Painful Suck, Salivary Gland Swollen, Body Ache, Weakness, Vomiting, Itching Skin, Headache.*

So I have formed a neural network with 15 input neurons, 9 output neurons and 12 hidden neurons. I have taken random numbers as the weights. And for training I have collected data of about 10000 patients in the relative fields from some clinics, hospitals, diagnostics centers and some physicians. I have implemented three algorithms for the network: back propagation algorithm, Kohonen algorithm and my proposed algorithm. In all thee cases fuzzy values from the fuzzification engine have been provided in the input layers. Also I have used 200 sets of input each time for classification (Testing). The results on the basis of convergence and classification rates are summarized in table 2.

(Activation function for back propagation and the proposed algorithm is $1/(1+e^{-i})$, I have used Intel (P 2 600 MHz with 64 MB RAM) processor.)

Val of η	# Of patterns as inputs	BP		Kohonen		Proposed algo.	
		T sec	R %	T sec	R %	T sec	R %
0.2	3000	6	60	2	70	5	77
	5000	9	63	3	71	7	77
	7000	14	68	4	72	10	78
	8000	16	75	5	70	12	76
	9000	17	82	5	74	13	77
	10000	18	85	6	75	15	77
0.3	3000	6	70	2	70	5	77
	5000	9	72	3	72	7	77
	7000	14	77	4	73	10	77
	8000	16	81	5	70	12	78
	9000	17	84	5	74	13	77
	10000	18	85	6	75	15	77
0.4	3000	6	81	2	73	5	78
	5000	9	84	3	71	7	79
	7000	14	85	4	72	10	79
	8000	16	86	5	70	12	79
	9000	17	86	5	74	13	79
	10000	18	86	6	75	15	80
0.5	3000	6	85	2	70	5	80
	5000	9	86	3	73	7	81
	7000	14	86	4	74	10	81
	8000	16	87	5	75	12	81
	9000	17	87	5	75	13	81
	10000	18	87	6	87	15	80
0.6	3000	6	84	2	70	5	81
	5000	9	85	3	72	7	81
	7000	14	85	4	73	10	81
	8000	16	86	5	74	12	81
	9000	17	86	5	77	13	80
	10000	18	87	6	80	15	81
0.7	3000	6	84	2	70	5	82
	5000	9	85	3	76	7	83
	7000	14	85	4	76	10	82
	8000	16	86	5	77	12	82
	9000	17	86	5	77	13	83
	10000	18	86	6	80	15	82
0.8	3000	6	78	2	72	5	81
	5000	9	81	3	70	7	81
	7000	14	85	4	74	10	80
	8000	16	84	5	77	12	81
	9000	17	84	5	75	13	81
	10000	18	85	6	75	15	81
0.9	3000	6	70	2	70	5	80
	5000	9	82	3	69	7	80
	7000	14	82	4	72	10	81
	8000	16	85	5	71	12	81
	9000	17	85	5	73	13	82
	10000	18	80	6	74	15	81

Table 2: Experimental Results

From table 2 we may have the following observations.

- At lower values of η the back propagation algorithm (BP) misclassifies a lot of testing elements. So we can say that at lower learning rates the BP takes long time to converge. However, the Kohonen neural network is very fast in calculations, but it is very much weak in classification, its performance never exceeds the 70s.
- The proposed algorithm is a bit slower than the Kohonen neural network but its classification rate is much better than that of the Kohonen neural network. If we compare the proposed algorithm with the BP, we can see that at the lower values of η the BP performs a bit poorly. It indicates that BP takes more training to converge. But the proposed algorithm takes much less steps to converge. From the view point of classification the completely trained BP performs better than the proposed algorithm. Yet, the performance of the proposed algorithm is not so poor as that of the Kohonen network.
- Moreover, both BP and Kohonen neural networks have disadvantages of their own. For example, BP suffers from the problems of local maxima, instability and infinite learning time. Performance of Kohonen neural network is questionable when the network is small. Also its performance depends on the initial weights. The proposed algorithm no such acute problems.
- The classification rate of the proposed algorithm remains almost constant at any values of η. It indicates the quick convergence and stability of the proposed algorithm.

5. CONCLUSION

The proposed algorithm is a simple but robust algorithm to classify patterns. From the performance analysis we can conclude that the proposed algorithm is no less then the backpropagation algorithm and the Kohonen neural networks from the view point of converging time, converging rate and correctly classification rate.

REFERENCES

[1]: Zadeh, L. A., "Fuzzy Sets," *Information and Control,* Vol. 8, pp. 338-353, 1965.

[2]: Zadeh, L. A., "Outline of a New Approach of Complex Systems and Decision Processes," in *IEEE Transactions on Systems, Man and Cybernetics,* SMC – 3, pp. 28- 44, 1973.

[3]: Zadeh, L. A., "The Concept of a Linguistic Variable and its Application to AAPPROXIMATE Reasoning," *Information Sciences,* Vol. 8, pp. 199- 249, 1975.

[4]: Zadeh, L. A., "Fuzzy Sets as a basis of possibility," in *Fuzzy sets and systems,* Vol. 1, pp. 3-28, 1978.

[5]: Gupta, M. M., and Knopf, G. K., "Fuzzy Neural Network Approach to Control Systems," in *Analysis and Management of Uncertainty: Theory and Applications,* B. M. Ayyub, M. M.

Gupta, and L. N. Kanal, eds., pp. 183-197, Elsevier, Amsterdam, 1992.

[6]: Pedrycz, W., "Fuzzy controls and fuzzy systems," *second (extended edition)*, John Wiley and sons, New York, 1993.

[7]: Rumelhart, D. E., Hinton, G. R., Williams, R. J., "Learning Internal Representation by Error Propagation," in *Parallel Distributed Processing,* Vol. 1, D. E. Rumelhart, and J. L. McClelland, eds., MIT Press, Cambridge, MA, 1986.

[8]: Werbos, P., "Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences," Ph. D. Dissertation, Harvard University, Boston, MA, 1974.

[9]: Parker, D., Learning Logic, "Invention Report," S81-64, File 1, *Office of Technology Licensing, Stanford University*, Palo Alto, CA, 1972.

[10]: Kohonen, T., "Self-organizing and Associative Memory," 2nd ed., Springer- Verlag, New York, 1988.

[11]: Pedrycz, W., and Rocha, A. F., "Fuzzy –Set Based Models of Neurons and Knowledge Based Networks," *IEEE Transactions on Fuzzy Systems*, Vol. 1, No. 4, pp. 254-256, 1993.

[12]: Wang, L. -X., "Adaptive Fuzzy Systems and Control," Prentice-Hall, Englewood Cliffs, NJ, 1994.

[13]: Wang, L. X., and Mandel, J. M. "Backpropagation Fuzzy systems as Nonlinear System Dynamic System Identifiers," in *IEEE International Conference on Fuzzy Systems,* San Diego, CA, 1992, pp. 1409-1418.

[14]: Hecht-Nielson R., "Neuro Computing," Addision-Weslsy, Reading, MA, 1990.

FIGURES

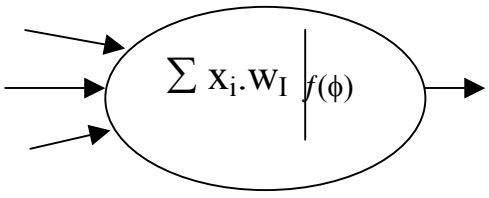

Figure 1: Neurons for the hidden layer

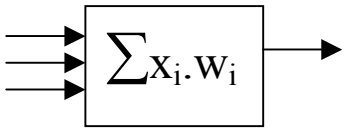

Figure 2: Neurons for the Kohonen layer

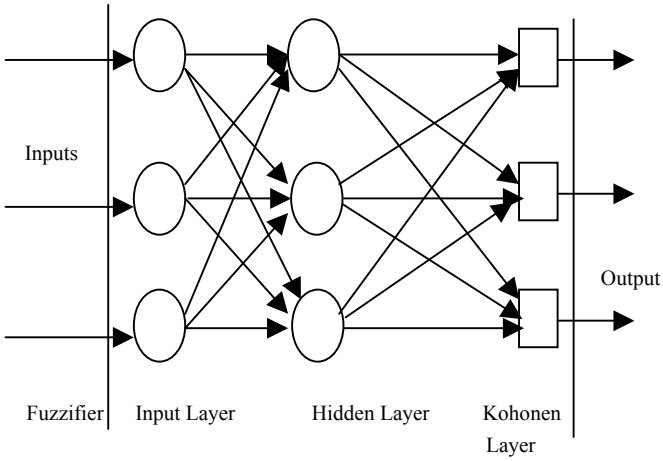

Figure 3: A Complete Network (Prototype)