Learning Algorithm for the Enhanced Fuzzy Perceptron

TH Goh, PZ Wang, HC Lui
Institute of Systems Science
National University of Singapore
Heng Mui Keng Terrace
Kent Ridge, S0511
Singapore
Email : ISSGTH @ NUSVM, thgoh@iss.nus.sg
December 1991

Abstract

In this paper, we propose a new network based on the enhanced fuzzy perceptron and its corresponding learning algorithm using the generalised delta rule[1]. This permits, for the first time, automatic extraction of fuzzy rules from historical data. Several test cases including the exclusive OR problem are reported.

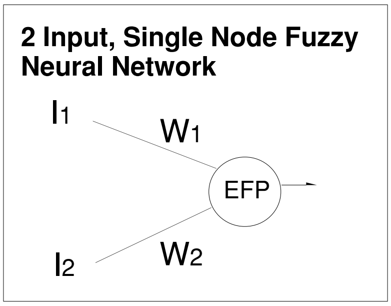
Introduction

In an effort to fuze neural networks with fuzzy logic, Teh [2][4] proposed the fuzzy perceptron as the building block for fuzzy neural networks. The fuzzy perceptron was able to perform many basic fuzzy logic operations such as the AND, OR, 2ndMax, N-Max, and NOT. The original perceptron, due to its use of the unit step function was, highly sensitive to changes in the weights, difficult to implement and could not learn from past data. In this paper we propose the enhanced fuzzy perceptron (furon), a modification to the fuzzy perception that replaces the unit step function with a continuous function. This modification results in greater stability and increased functionality, and allows for a new learning algorithm based on the generalised delta rule. Since the fuzzy operation of each furon may be determined by its weights,the output of a feedforward network of furons can be determined by chaining these operations together. By training such a network on historical data using the proposed learning algorithm, one can then extract the fuzzy rules from the trained network. First we propose the learning algorithm for the two input fuzzy perceptron. We then extend the learning algorithm to the two layer network. Clearly the algorithm may be extended to arbitrary layers and inputs. The learning algorithm was tested on several toy problems including the XOR problem . In all cases, the furon network was able to converge quickly and the fuzzy relations interpreted. Since the learning algorithm is based on gradient descent, it was tolerant to noise in the training data and this was also shown to be true in the XOR case when some noise was introduced.

Learning Algorithm for the Two Input Fuzzy Perceptron

For each input, repeat steps one and two until error is minimised.

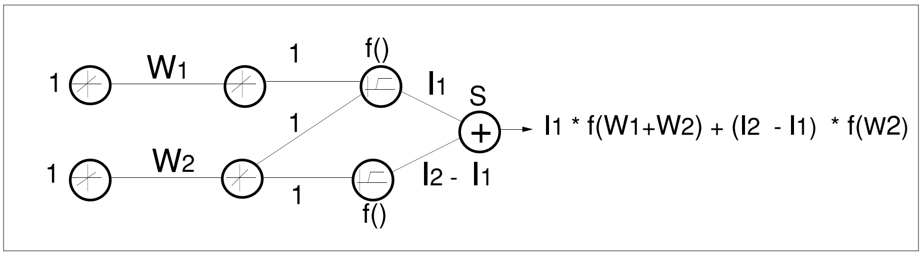
Step One: Sort the inputs of the perceptron such that $I_1 \le I_2$.



The output of the network can be expressed,

$$\text{Output} = I_1 * f(W_1+W_2) + (I_2 - I_1) * f(W_2)$$

This expression may be viewed as a conventional neural network as



Step Two : Applying the generalised delta rule to the above sorted network, we derive the incremental changes for the weights W_1 and ΔW_2,

$$\Delta W_1 = \varepsilon * I_1 * f'(W_1+W_2)$$
$$\Delta W_2 = \varepsilon * I_1 * f'(W_1+W_2) + \varepsilon * (I_2 - I_1) * f'(W_2)$$

Learning Algorithm for the Two Layer Fuzzy Neural Network

For each input, repeat steps one and two until error is minimised.

Step One : Do a forward pass and sort the network such that $I_1 \leq I_2$ and $O_1 \leq O_2$.

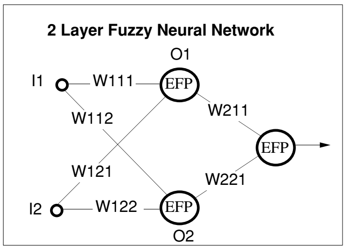


Figure 3.

Step Two : Apply the generalised delta rule to the resultant network.

Update weights W_{211} and W_{221} using

$$\Delta W_{211} = \varepsilon * O_1 * f'(W_{211}+W_{221})$$
$$\Delta W_{221} = \varepsilon * O_1 * f'(W_{211}+W_{221}) + \varepsilon * (O_2 - O_1) * f'(W_{221})$$

where ε = target - output

To update weights W_{111}, W_{112}, W_{121} and W_{122}, consider the output equation of the output node,

$$\text{Output} = (f(W_{211}+W_{221}) - f(W_{221})) * O_1 + f(W_{221}) * O_2$$

Applying the generalised delta rule to derive the incremental changes to O_1 and O_2,

$$\Delta O_1 = \varepsilon * (f(W_{211}+W_{221}) - f(W_{221}))$$

and $$\Delta O_2 = \varepsilon * f(W_{221})$$

Backpropagating ΔO_1 and ΔO_2 as the error and applying the generalised delta rule, we derive the incremental changes to W_{111} and W_{121},

$$\Delta W_{111} = \Delta O_1 * I_1 * f'(W_{111}+W_{121})$$
$$\Delta W_{121} = \Delta O_2 * I_1 * f'(W_{111}+W_{121}) + \Delta O_2 * (I_2 - I_1) * f'(W_{121})$$

and W_{112} and W_{122} using,

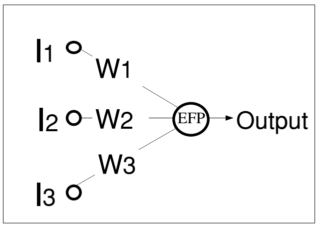
$$\Delta W_{112} = \Delta O_1 * I_1 * f'(W_{112}+W_{122})$$
$$\Delta W_{122} = \Delta O_2 * I_1 * f'(W_{112}+W_{122}) + \Delta O_2 * (I_2 - I_1) * f'(W_{121})$$

Test Results

Single Furon with 3 inputs

The learning algorithm was first tested on a single furon with three inputs.

MIN Test



In this test the output was set at MIN(I_1,I_2,I_3) where I_1,I_2,I_3 were randomly generated. After less than 20 epochs, the network was able to converge to $W_1 = W_2 = W_3 = 0.33$. These weights corresponds to those of he fuzzy MIN furon as expected.

MAX Test

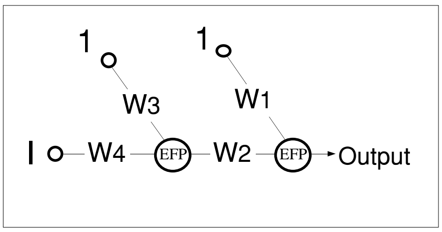
In this test, the output was set to MAX(I_1,I_2,I_3) where I_1,I_2,I_3 were randomly generated. After less than 20 epochs, the network was able to converge to $W_1 = W_2 = W_3= 1$. Again these weights corresponds to those of the fuzzy MAX furon.

2nd MAX Test

In this test, the output was set to 2ndMAX(I_1,I_2,I_3) where I_1,I_2,I_3 were randomly generated. After less than 20 epochs, the network was able to converge to $W_1 = W_2 = W_3= 0.5$ and they corresponds to the fuzzy 2nd MAX furon as expected..

Two layer Network with Single Input and Bias

In this test, there were two furons (fuzzy perceptron) each with two inputs connected as shown in the diagram at the right. Both furons have one of their inputs as "1", serving as a bias. Several tests were done. In each test the output was set to a fuzzy function of the input I. The following functions were tested.



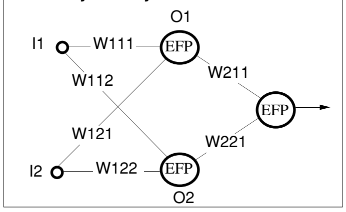
Output	W1	W2	W3	W4
1 - I	0.5	0.5	1	-0.63
0.3(1 - I)	0.7	-0.25	0.65	0.34
1 - 0.8I	0.6	0.4	1	-0.5
1 - 0.3 I	0.82	0.4	1	-0.41
0.9 - 0.5I	0.64	0.36	0.93	-0.35
0.7 + 0.3I	0.79	0.26	0.65	0.35
0.3 + 0.7I	0.60	0.41	0.57	0.43

In all cases, the network was able to converge in less than 50 epochs. In the 1 - I case the output furon is the direct through furon which lets the input propagates unchanged and the input furon performs the (1 - I) or fuzzy NOT operation. Hence the output of the network network is the fuzzy NOT operation. In the 1 - 0.8I case, the input furon does the (1 - I) operation. The output of the output furon, may be computed as (0.6 - 0.5)/0.5 * (1 - input) + input. Where input in this case is (1 - I). Hence the output of the network may be determined as 1 - 0.8I as expected.

Two Layer Network with Two Inputs

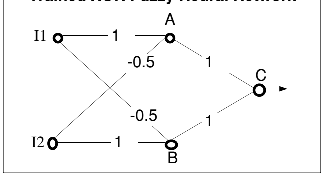
XOR Test Case

In this test, the network was trained on the XOR truth table



2 Layer Fuzzy Neural Network

I_1	I_2	XOR
0	0	0
0	1	1
1	0	1
1	1	0



Trained XOR Fuzzy Neural Network

After less than 100 epochs, the network was able to converge as shown in the diagram. In this case, A performs the <u>I1 - I2</u> operation, B the <u>I2 - I1</u> operation and C the fuzzy OR operation on the outputs of furons A and B.

Fuzzy XOR Test Case

In this test, the network was trained on the following truth tables:-

1	I_2	XOR
0	0	0.01
0	1	0.97
0.98	0	1
1	1	0

1	I_2	XOR
0	0	0
0	0.1	0.1
0.1	0	0.1
1	1	0

I_1	I_2	XOR
0	0	0
0	0.9	0.9
0.9	0	0.9
1	1	0

In the first case, noise is introduced to both inputs and output. After a few thousand epochs, the network was able to converge to the same weights as the preceding XOR test case.

Further Work

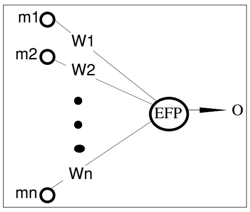
The original implementation was on a spreadsheet using a Macintosh personal computer. Although this allowed us to monitor the progress of the learning algorithm very easily, it becomes increasing difficult to implement complex test cases. Presently we are working on implementing the algorithm using C on a workstation. We will then test the algorithm on more complex data.

Conclusions

We have presented the enhanced fuzzy perceptron which has greater stability and functionality compared to the fuzzy perceptron, and a new learning algorithm for a network of enhanced fuzzy perceptron that for the first time, allows automatic extraction of fuzzy relations from data. The algorithm converges quickly and was able to learn the classical XOR problem.

Appendix A: Propagation Algorithm of the Fuzzy Perceptron

For a fuzzy perceptron with incoming links A_i ... A_n with weights W_i ... W_n and membership degrees m_i to m_n,



Step 1 : Rearrange Ai according to the ascending order of the membership degree m_i and add an item m_{jo} at the beginning of this sequence.

$0 = m_{jo} \leq m_{j1} \leq m_{j2} \leq m_{j3} \leq m_{j4} \leq \ldots \leq m_{jn} \leq 1$

Compute the consecutive difference between the items of the sequence

$p_i = m_{ji} - m_{j(i-1)} \quad (i = 1, \ldots ,n)$

Step 2 : Write down the weightage sequence corresponding to the membership degree value of the rearranged nodes. For each i (i = 1, . . . , n) compute

$b_i = \sum W_{jn} \quad (n = 1, \ldots, i)$ and

$f_i = F(b_i)$ where $F(w) = 0 \quad w < 1$
$F(w) = 1 \quad w \geq 1$

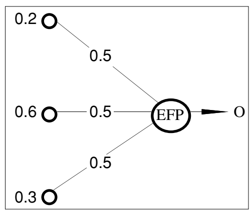
F(w) is the weight transfer function.

Step 3 : Multiply f_i by p_i and take the sum to get the output of the fuzzy perceptron.

$O = \sum f_i * p_i$

Example :

Consider the following network,



Step 1	0	0.2	0.3	0.6
	p	0.2	0.1	0.3
Step 2	w	0.5	0.5	0.5
	b	1.5	1.0	0.5
	f	1	1	0
Step 3	O =	1 * 0.2 +	1 * 0.1 +	0 * 0.3 = 0.3

Hence output of this network is 0.3.

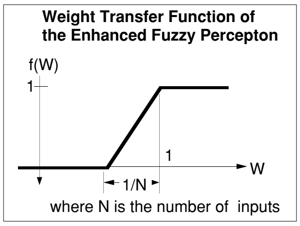
Appendix B: The Enhanced Fuzzy Perceptron

Weight Transfer Function

In the Enhanced Fuzzy Perceptron, the weight transfer function is modified from the unit step function to a ramp function as shown.

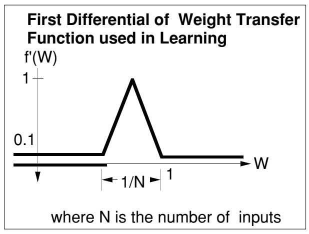
$$
\begin{array}{lllll}
F(w) & = & 1 & & w \geq 1 \\
F(w) & = & 0 & & w \leq (1 - 1/N) \\
F(w) & = & N(w + \frac{1}{N} - 1) & & (1 - 1/N) < w < 1
\end{array}
$$

where N is the number of input links.



First Differential of Weight Transfer Function

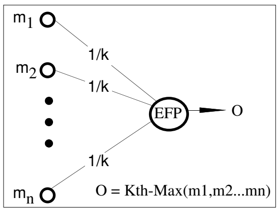
Also for the purpose of learning, for better convergence[5], the first differential, F'(w) is modified as shown.

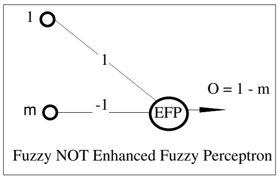


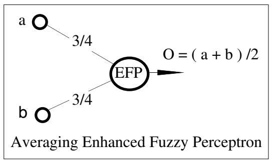
Propagation Algorithm

The propagation algorithm remains the same as that of the fuzzy perceptron.

Examples of Single Enhanced Fuzzy Perceptron Networks







References

1. Yoh-Han Pau, "Adaptive Pattern Recognition and Neural Networks",Addison-Wesley,Chapter 5,1990.

2. LS Hsu,SC Chan,KF Loe,PZ Wang,HH Teh,"Fuzzy Perceptrons",unpublished,1990.

3. D.E. Rumelhart, JL McClelland and the PDP Research Group,"Parallel Distributed Processing", Vol.1, MIT press, 1986.

4. HH Teh, BT Loh, HC Lui and AH Tan, "Connectionist Expert System with AdaptiveLearning Capability", IEEE Transaction in Knowledge and Data Engineering, Vol 3, No. 2, June 1991.

5. Scott E. Fahlman, "An Empirical Study of Learning Speed in Back-Propagation Networks", Carnegie-Mellon University Technical Report CMU-CS-88-162, June 1988.