A FUZZY NEURAL NETWORK AND ITS MATLAB SIMULATION

Alexander Nikov, Tzanko Georgiev
Technical University of Sofia
BG-1756 Sofia, Bulgaria
nikov@vmei.acad.bg, tzg@vmei.acad.bg

Abstract: A fuzzy neural network and its relevant fuzzy neuron and fuzzy learning algorithm are introduced. An object-oriented implementation of fuzzy neural network in MATLAB environment is realized. Simulations are carried out by SIMULINK. The performance of fuzzy neural network is experimentally compared with other neural networks trained by backpropagation algorithms. It shows better convergence speed. This confirms its applicability in learning large-sized neural networks of real-life applications like adaptive interactive systems, modelling of biotechnological processes, etc.

Keywords: neuro-fuzzy learning, object-oriented approach, MATLAB

1. Introduction

Currently grows the development and application of neuro-fuzzy systems (Nauck, 1997) integrating the learnability of neural networks with the transparency and interpretability of the fuzzy systems. This combination of both techniques finds successful application in many areas (Lin, 1997) like adaptive interactive systems (Nikov, 1999), modelling of biotechnological processes (Georgiev, 1997), etc. Here a fuzzy neural network is proposed. Its fuzzy neuron and a fuzzy learning algorithm are introduced. They are object-oriented implemented in MATLAB (Biran, 1999; Pärt-Enander, 1999).

2. Description of fuzzy neural network

A fuzzy set $\mathbf{P}$ is defined on the space $X$, as a set of ordered pairs $(p, \mu_p(x))$, where $\mu_p(x)$ denotes the value of membership function $\mu_p(x): 2^P \rightarrow [0,1]$ at a given point $p \in X$. The family of all fuzzy sets defined on space $X$ (universe) is denoted as $\mathcal{F}(X)$. Each point of $X$ is represented by vector $p$.

For fuzzy sets with membership function $\mu_p(x): 2^P \rightarrow [0,1]$ are defined the following operations:

- intersection: $\cap \left( \mu_1, \mu_2 \right) = \min \left( \frac{n}{k}, \mu_1, \mu_2 \right) = \wedge \left( \mu_1, \mu_2 \right) = \bigwedge \left( \mu_1^k, \mu_2^k \right)$;
- union: $\cup \left( \mu_1, \mu_2 \right) = \max \left( \frac{n}{k}, \mu_1, \mu_2 \right) = \vee \left( \mu_1, \mu_2 \right) = \bigvee \left( \mu_1^k, \mu_2^k \right)$;

where $\left( \frac{n}{k} \right)$ denotes all $k$-sized subsets of $n$-sized set (the final number of all subsets is $2^n$).

The basic operations are:

union: $\mathbf{A} \cup \mathbf{B}, \mu_{\mathbf{A} \cup \mathbf{B}}(x) = \bigvee (\mu_{\mathbf{A}}(x), \mu_{\mathbf{B}}(x))$
intersection: $\mathbf{A} \cap \mathbf{B}, \mu_{\mathbf{A} \cap \mathbf{B}}(x) = \bigwedge (\mu_{\mathbf{A}}(x), \mu_{\mathbf{B}}(x))$
Similar to a fuzzy set any fuzzy relation $R$ on the cartesian product $X_1 \times X_2 \times \ldots \times X_n$, i.e. $R \in \mathcal{I}(X_1 \times X_2 \times \ldots \times X_n)$ is defined by the membership function $\mu_R(x) : 2^{(X_1 \times X_2 \times \ldots \times X_n)} \rightarrow [0,1]$. The composition between the fuzzy set $X$ and the relation $R$ is defined as

$$X \circ R, \mu_{X \circ R}(y) = \bigoplus \mu_X(x) \otimes \mu_R(x,y).$$

A fuzzy neural network can be defined as:

$$X(i) = R_N(f_N(X(i-1)), f_N(W(i)))$$

(1)

$$W(i) = R_L(X(i), W(i), W(i-1))$$

(2)

where $X(i), X(i-1)$ are fuzzy sets of neural network state defined in the space $X$, i.e. $X(i), X(i-1) \in \mathcal{I}(X)$. $W(i) \in \mathcal{I}(W)$ are the weights on the $i$-th learning step. $R_N$ is a fuzzy relation describing the activation function and $R_L$ is a fuzzy relation describing the learning algorithm. Both fuzzy relations satisfy $R_N \in \mathcal{I}(X^2)$ and $R_L \in \mathcal{I}(X^2W^2)$.

### 2.1 Model of fuzzy neuron

Based on (1) the model of a fuzzy neuron at $k$-th position on the $l$-th layer of neural network can be described as follows:

$$X_k^{(l)}(i) = R_N(f_N(X_k^{(l)}(i-1)), f_N(W_k^{(l)}(i)))$$

(3)

The relation $R_N$ of fuzzy neuron is

$$R_N(X_k^{(l)}(i-1), W_k^{(l)}(i)) = \left\{ \bigotimes f_N(X_k^{(l)}(i-1)) \right\} \bigoplus \left\{ \bigotimes f_N(W_k^{(l)}(i)) \right\}$$

(4)

The fuzzy function

$$f_N(X) : X \rightarrow 2^X,$$

(5)

where $2^X$ is a notation of set on all subsets of set $X$.

The activation function of fuzzy neuron (3) is

$$F(X(i), X(i-1), W(i)) = \vee(1, R_N).$$

(6)

The output of neuron $a(i)$ is equal to activation function.

### 2.2 Fuzzy learning algorithm

The learning algorithm for $p$-th position on the $l$-th layer based on (2) is

$$W_k^{(l)} = R_L(X_k^{(l)}(i), W_k^{(l)}(i-1), W_k^{(l)}(i))$$

(7)

The learning rules are:
**Rule 1: Calculation of error $\delta$ between each neuron output $a$ and target value $t$.**

**Output layer**

\[ \delta_k = |t_k - a_k^0|, k = 1,2,\ldots,n_a. \]  

(8)

**Hidden layer**

\[ \delta_k = \bigwedge_{i=1}^{n_h} |t_k - a_k^0|, k = 1,2,\ldots,n_h, \]  

(9)

where operator $\Lambda$ is calculated in case of multiple targets.

**Rule 2: Calculation of weights update**

The update of weights is obtained as

\[ \Delta W = \eta \delta, \]  

(10)

where $\eta$ presents the learning rate.

**Rule 3: Calculation of weights**

Case 1: if $t = a_k^0$ then $W(i) = W(i-1)$.

Case 2: if $t > a_k^0$

- if $W(i-1) < t$ then $W(i) = 1 \land (W(i-1) + \Delta W)$
- elseif $W(i-1) \geq t$ then $W(i) = W(i-1)$.

Case 3: if $t < a_k^0$

- if $W(i-1) > t$ then $W(i) = 0 \lor (W(i-1) - \Delta W)$
- elseif $W(i-1) \leq t$ then $W(i) = W(i-1)$.

The fuzzy neural network is learned by training patterns with input values $X \in [0,1]$ and targets $t \in [0,1]$. The initial weights $W \in [0,1]$.

3. MATLAB object-oriented fuzzy neural network

MATLAB package (Biran, 1999; Pärt-Enander, 1999) is very appropriate for simulation of fuzzy neural network proposed. In the following an object-oriented presentation of fuzzy neuron, of leaning algorithm, and of fuzzy neural network composition is described. A simulation is carried out in the SIMULINK environment.

3.1 Encapsulation of fuzzy neuron and learning algorithm

Encapsulation presents a combination of a data structure with the functions (actions and methods) dedicated to data manipulating. A new class `FuzzyNeuron` and a class directory `@FuzzyNeuron` following MATLAB’s features are designed. The object `FuzzyNeuron` is presented by the fuzzy neuron model (3) and the learning algorithm (7) for $k$-th neuron at the $l$-th layer of fuzzy neural network. An M-file is used as a constructor of this object. The constructor creates `FuzzyNeuron` by initialising the data structures and by assigning the class tag. The object `FuzzyNeuron` is described by the following tree structures:

- `FuzzyNeuronDescription`
- `FuzzyNeuronSimulation`
- `FuzzyNeuronPosition`

The `FuzzyNeuronDescription` contains the following fields: PositionOfNeuron, InitialWeight, Target, LearningRate, StepStorages, OutputValues and OutputWeights. `FuzzyNeuronSimulation`
includes the standard SIMULINK structure for description of the simulation process. *FuzzyNeuronPosition* contains the following fields: Layer, PositionInLayer, PositionInDiagram, LibraryName, NumberOfInputs, BlockType, BlockName and DiagramName.

The object *FuzzyNeuron* is implemented in SIMULINK environment as a block shown in Fig. 1. It consists of two tarts. The first one presents a fuzzy neuron with its inputs and output. The second one presents the learning algorithm as a feedback of fuzzy neuron. The blocks labels are names of M-file procedures implemented by fuzzy neuron (3) and the learning algorithm (7). T and LR denote target values and learning rate respectively. It should be underlined that the proposed block diagram presents a non-linear closed-loop system.

![Fig.1. Fuzzy neuron block diagram](image)

### 3.2 Polymorphism

Polymorphism gives a name or a symbol to each action in the class hierarchy. To change the behaviours of MATLAB's operators and functions for the new object *FuzzyNeuron* are overloaded the following arithmetic operators: unary minus (-a); unary plus (+a); binary addition (a+b); binary subtraction (a-b); and horizontal concatenation ([a,b]). For example (cf. Fig. 2):

- -a (unary minus) – deletes an object from block diagram;
- +a (unary plus) – adds an object to block diagram;
- a+b (addition) – adds a line between objects;
- a-b (subtraction) – deletes a line between objects;
- [a,b] (horizontal concatenation) – makes a concatenation of neurons as a layer of fuzzy neural network.

### 4. Experimental comparison of fuzzy neural network with other neural networks

For simple illustration of the fuzzy neural network proposed and for its experimental comparison with other neural networks a neural network with single output neuron and a hidden layer is used (cf. Fig. 2). The network is learned by 10 training patterns. The fuzzy neural network is compared with the following two backpropagation algorithms: Conjugate gradient...
backpropagation algorithm Fletcher-Reeves Update and Steepest descent backpropagation algorithm (Hagen, 1996; MATLAB, 1997). The simulation process is carried out at error level (SquaredError) not greater than 0.001. The results on Fig. 3 show significant better convergence of fuzzy neural network in comparison with these two backpropagation algorithms. Experiments with other larger neural networks show similar results.

Fig. 2. Example fuzzy neural network with single output neuron and a hidden layer.

Fig. 3. Squared error for fuzzy neural network,
5. Conclusions

A fuzzy neural network is proposed. Its advantage is that implicit knowledge acquired from experts can be easily incorporated into fuzzy neural network, and that implicit knowledge can be learning from training samples to enhance the accuracy of the output. It is object-oriented implemented in MATLAB environment. A significant advantage of proposed object *FuzzyNeuron* is a union of fuzzy neuron description of learning algorithm. This ensures a flexible composition of any neural network and full using of SIMULINK capability (cf. Fig. 2).

Experimental comparison of fuzzy neural network showed significant better convergence speed over other two backpropagation algorithms. So this neural network is very appropriate to be applied in training of large-sized neural networks. This is very useful in creation of adaptive interfaces (Nikov, 1999) in the area of human-computer interaction. Also it can be applied in many other areas like modelling of biotechnological processes (Georgiev, 1997), etc.

In the further object-oriented presentation of fuzzy neural network the inheritance of objects should be added. This paper presents the first results in applying the fuzzy neural network proposed and should be viewed as the starting point of many forthcoming studies. Genetic algorithms can be incorporated to further enhance the performance of the fuzzy neural network.

6. References