Modus Ponens Fuzzy Network in Matlab

Dana Majerová, Jaromír Kukal

ICT Prague, Department of Computing and Control Engineering

Keywords: *fuzzy network, fuzzy logic function, Łukasiewicz algebra, Modus Ponens, inductive learning, Matlab.*

Abstract. *The Modus Ponens rule as the basic principle of logic motivates new fuzzy network architecture. The four layer fuzzy network architecture is defined in a neural network style. The fuzzy network consist of the input layer, the fuzzy logic function layer, the Modus Ponens layer and output layer. The network produces learnable fuzzy logic functions and its processing and learning rules are defined. The Modus Ponens Fuzzy Network is realized in MATLAB system.*

1 Introduction

The fuzzy logic function theory (see Novák, Perfilieva and Močkoř) [3]) enables to approximate Lipschitz continuous functions on hypercube $[0;1]^n$. Using the adjoint property and the Modus Ponens law, the lower and upper bounds of any given function is determined [1]. The compromise estimate is defined using Łukasiewicz square root function. The final four layer fuzzy network produces fuzzy logic functions on its output and is able to learn from the pattern set.

2 Modus Ponens Fuzzy Network

The four layer Modus Ponens Fuzzy Network (MPFN) consists of n-input nodes in the first layer, H-fuzzy logic function nodes in the second layer, $2m$-Modus Ponens nodes and m-output nodes. The MPFN is designed as vector fuzzy logic function in hierarchic neural network style. The MPFN scheme is on the Fig. 1.

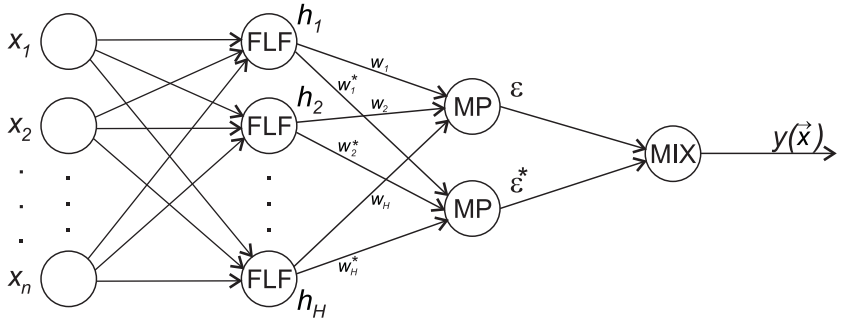

Figure 1: Modus Ponens Fuzzy Network for $m = 1$

3 Inductive learning rule

The learning process changes the weights from unit values down, while the system of fuzzy logic functions in the second layer is given before learning and then permanent. The learning process has two phases - the initialization phase is its first step.

INITIALIZATION PHASE:

$$w_{i,j} = 1, \ for \ 1 \le i \le m, \ 1 \le j \le H \quad (1)$$
$$w^*_{i,j} = 1, \ for \ 1 \le i \le m, \ 1 \le j \le H \quad (2)$$

Let $(\overrightarrow{x}_k, \overrightarrow{y}_k)$ be the k-th pattern. Let M be the number of patterns. Then the next step of learning is repeated for $1 \le k \le M$.

ONE PATTERN LEARNING:

For $1 \le i \le m, \ 1 \le j \le H$ do:

$$w_{i,j}^{NEW} = w_{i,j} \wedge (h_j(\overrightarrow{x}_k) \to y_{k,i}) \quad (3)$$
$$w^{*NEW}_{i,j} = w^*_{i,j} \wedge (h_j(\overrightarrow{x}_k) \to \neg y_{k,i}) \quad (4)$$
$$(5)$$

4 Signal processing rule

The MPFN processes the signal in three steps for any output $1 \le i \le m$:

$$a_i(\overrightarrow{x}) = \bigvee_{j=1}^{H} w_{i,j} \otimes h_j(\overrightarrow{x}) \quad (6)$$
$$b_i(\overrightarrow{x}) = \bigvee_{j=1}^{H} w^*_{i,j} \otimes h_j(\overrightarrow{x}) \quad (7)$$
$$y_i(\overrightarrow{x}) = (a_i(\overrightarrow{x}))^{1/2} \otimes (\neg b_i(\overrightarrow{x}))^{1/2} \quad (8)$$

5 Matlab realization of MPFN

The Modus Ponens fuzzy network was realized using two general functions. The first of them realizes the whole learning process. The LEARN.M function uses the input and output pattern matrices x, y with M rows and n or m columns respectively. The third parameter $flfname$ is a string containing the name of FLF base. The outputs are the matrices W, W^*.

```
function [w,wstar]=LEARN(x,y,flfname)
% MPFN (Modus Ponens Fuzzy Network) learning
% [w,wstar]=LEARN(x,y,flfname);
% x ... pattern input matrix x(m,ni)
% y ... pattern output matrix y(m,no)
% flfname ... name of user FLF base for hidden layer
%             ('Generate2N' recommended)
% w ... first MPFN weight matrix w(no,nh)
% wstar ... second MPFN weight matrix wstar(no,nh)

% testing evaluation for the dimesion of h
h=feval(flfname,x(1,:));
% dimension evaluation
```

```
ni=size(x,2);
no=size(y,2);
m=size(x,1);
nh=size(h,2);
% initial phase
w=ones(no,nh);
wstar=w;
% m learning steps
for k=1:m
    h=feval(flfname,x(k,:));
   for i=1:no
      for j=1:nh
         w(i,j)=min([w(i,j) 1-h(j)+y(k,i)]);
         wstar(i,j)=min([wstar(i,j) 2-h(j)-y(k,i)]);
      end
   end
end
```

The response of MPFN to any given input fuzzy vector is realized as the function RESPONSE.M. Its parameters are input vector x, the matrices W, W^* and the same $flfname$ string. The output response includes the output vector y, its *lower* and *upper* estimate.

```
function [y,lower,upper]=RESPONSE(x,w,wstar,flfname)
% MPFN response for given input vector
% [y,lower,upper]=RESPONSE(x,w,wstar,flfname);
% x ... input vector x(1,ni)
% w ... 1st MPFN weight matrix w(no,nh)
% wstar ... 2nd MPFN weight matrix wstar(no,nh)
% flfname ... name of user FLF base for hidden layer
%             ('Generate2N' recommended)
% y ... output vector y(1,no)
% lower ... lower bound lower(1,no)
% upper ... upper bound upper(1,no)

% hidden layer fuzzy value vector
h=feval(flfname,x);
% matrix dimensions
ni=size(x,2);
no=size(w,1);
nh=size(h,2);
lower=zeros(1,no);
upper=lower;
% lower bound estimate
for i=1:no
   lower(i)=0;
   for j=1:nh
      lower(i)=max([lower(i) w(i,j)+h(j)-1]);
   end
end
% upper bound estimate
for i=1:no
```

```
   upper(i)=0;
   for j=1:nh
      upper(i)=max([upper(i) wstar(i,j)+h(j)-1]);
   end
end
upper=1-upper;
% compromise output of MPFN
y=(lower+upper)/2; % try to recognize using
                   % of Lukasiewicz square roots
```

The MPFN library enables user oriented design of FLF base for the hidden layer preprocessing. There is an example of the simplest possible reasonable preprocessing.

```
function [h]=GENERATE2N(x)
% Simple MPFN base generator
% GENERATE2N generates vecttor h=[1 x1 ... xn ~x1 ...~xn]
% [h]=GENERATE2N(x);
% x ... given input fuzzy vector x(1,n)
% y ... hidden layer fuzzy vector y(1,2*n+1)
h=[1 x 1-x];
```

6 Conclusions

The MPFN network produces the fuzzy logic functions and contains learnable weights. The MPFN is realized in Matlab system with the FLF base containing the inputs and their negations.

References

[1] Kukal J.(2000). *Fuzzy Envelope and Modus Ponens Fuzzy Network*, Mendel 2000, Brno.

[2] Łukasiewicz J.(1970). *Selected Works*, Amsterdam.

[3] Novák V., Perfilieva I., Močkoř J. (1999). *Mathematical Principles of Fuzzy Logic*, Kluwer Academic Publishers, Boston.

Dana Majerová, Jaromír Kukal
ICT Prague, Department of Computing and Control Engineering
Technická 5, 166 28 Prague 6 Dejvice
Phone: 420-2-2435 4174, fax: 420-2-2435 5053
E-mails: Dana.Majerova@vscht.cz, Jaromir.Kukal@vscht.cz