Computational Intelligence
Lecture : Fuzzy-Neural

Arash Yeganeh Fallah
Naser sepehry

Department of Electrical Engineering
Amirkabir University of Technology

Fall 2010

Basic Structure of Fuzzy Neural Networks

Definition of Fuzzy Neurons

Fuzzy Neural Networks

Neural Network Representation of Fuzzy Relation Equations

A Fuzzy Neural Network Based on FN(V, Λ)

A Fuzzy δ Learning Algorithm

BP learning algorithm

BP learning algorithm of FAM's (method 1)

BP learning algorithm (method 2)

Basic Structure of Fuzzy Neural Networks (FNN)

Definition of Fuzzy Neurons

- The objective of FNN is to extend the capability of the neural networks to handle “vague” information than “crisp” information only.
- Classifying of FNN:

1- a fuzzy neuron with crisp signals used to evaluate fuzzy weights,

2- a fuzzy neuron with fuzzy signals which is combined with fuzzy weights,

3- a fuzzy neuron described by fuzzy logic equations.

A multifactorial function is actually a projective mapping from an m-array space to a one-array space, denoted by M_m.

$$M_m : [0,1]^m \longrightarrow [0,1], (x_1, \cdots, x_m) \longmapsto M_m(x_1, \cdots, x_m)$$

a natural partial ordering "≤ " is defined as follows

$$\left(\forall X, Y \in [0,1]^m\right)\left(X \leq Y \Longrightarrow (x_j \leq y_j, \quad j = 1, 2, \cdots, m)\right)$$

Basic operators (+, .) in multifactorial function change as bellow:

$$(\vee, \wedge), \ (\vee, \cdot), \ (+, \ \wedge), \ \text{etc}$$

Where V is MAX and Λ is MIN

A fuzzy neuron is regarded as a mapping *FN :*

$$FN : [0,1]^m \rightarrow [0,1]$$

$$X \longmapsto FN(X) \triangleq \varphi(M_m(X) - \theta),$$

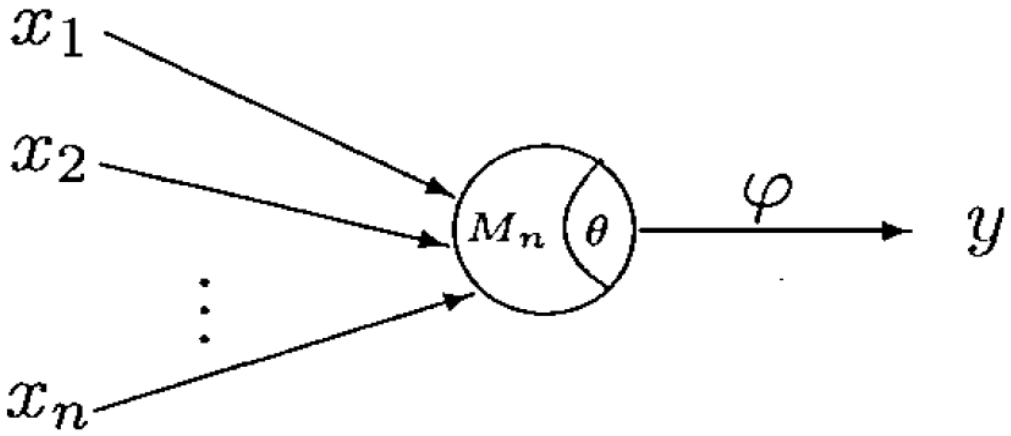
For

$\theta \in [0,1]$

$\varphi: \Re \rightarrow [0,1]$ *where* $\varphi(u) = 0, u \leq 0$

φ is a mapping or an activation function

This Figure illustrates the working mechanism of a fuzzy neuron.



Example:

The following mappings from $[0,1]^n$ to $[0,1]$ are all fuzzy neurons:

$$y = \varphi\Big(\sum_{i=1}^{n} w_i x_i - \theta\Big),$$

where $w_i \in [0,1]$ and $\sum_{i=1}^{n} w_i = 1$.

$$y = \varphi\Big(\bigvee_{i=1}^{n} (w_i x_i) - \theta\Big),$$

where $w_i \in [0,1]$ and $\bigvee_{i=1}^{n} w_i = 1$.

$$y = \varphi\Big(\bigvee_{i=1}^{n} (w_i \bigvee x_i) - \theta\Big),$$

where $w_i \in [0,1]$ and $\bigvee_{i=1}^{n} w_i \leq \bigvee_{i=1}^{n} x_i$.

Fuzzy Neural Networks

Neural Network Representation of Fuzzy Relation Equations

A typical kind of fuzzy relation equation:

$$\mathbf{X} \circ \mathbf{R} = \mathbf{B}$$

Where X is input vector

$$\mathbf{X} = (x_1, x_2, \cdots, x_n)$$

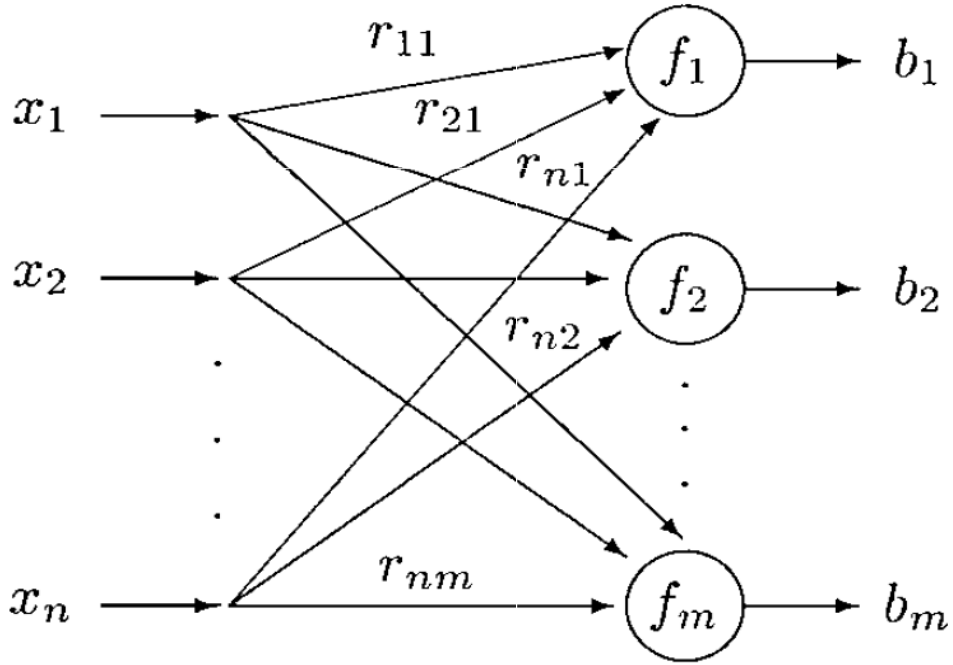
R is matrix of coefficients, B is constant matrix

$$\mathbf{R} = (r_{ij})_{n \times m} \qquad \mathbf{B} = (b_1, b_2, \cdots, b_m)$$

And operator “o” is defined as follow:

$$\bigvee_{i=1}^{n} (x_i \wedge r_{ij}) = b_j, \quad j = 1, 2, \cdots, m$$

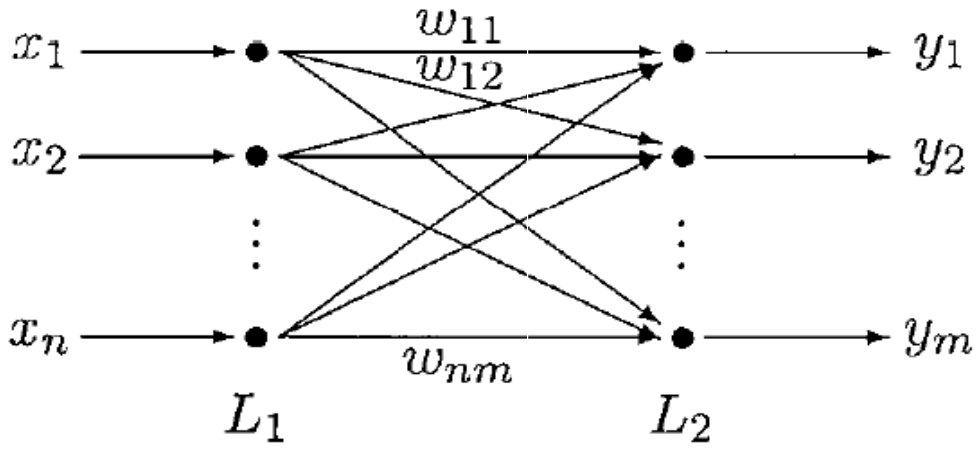
network representing fuzzy relation equations:



- where the activation functions of the neurons $f_1, f_2, \ldots, f_m$ are all taken as identity functions and the threshold values are zero
- Fuzzy relation equations can be solved using a fuzzy δ learning algorithm.

A Fuzzy Neural Network Based on FN(V, Λ)

The network is known to have **fuzzy associative memories (FAM)** ability based on FN(V, Λ).



The relation between input and output of this network is as follows:

$$\begin{cases} y_1 = (w_{11} \wedge x_1) \vee (w_{21} \wedge x_2) \vee \cdots \vee (w_{n1} \wedge x_n) \\ y_2 = (w_{12} \wedge x_1) \vee (w_{22} \wedge x_2) \vee \cdots \vee (w_{n2} \wedge x_n) \\ \quad \cdots \\ y_m = (w_{1m} \wedge x_1) \vee (w_{2m} \wedge x_2) \vee \cdots \vee (w_{nm} \wedge x_n) \end{cases}$$

Matrix form of FAM based on FN(∨, ∧) is as follow

$$\mathbf{Y} = \mathbf{X} \circ \mathbf{W}$$

where $\mathbf{Y} = (y_1, y_2, \cdots, y_m)$, $\mathbf{X} = (x_1, x_2, \cdots, x_n)$ and

$$\mathbf{W} = \begin{pmatrix} w_{11} & w_{12} & \cdots & w_{1m} \\ w_{21} & w_{22} & \cdots & w_{2m} \\ & \cdots & & \\ w_{n1} & w_{n2} & \cdots & w_{nm} \end{pmatrix}$$

For given a set of samples:

$$\{(\mathbf{a}_s, \mathbf{b}_s) | s = 1, 2, \cdots, p\},$$

where $\mathbf{a}_s = (a_{s1}, a_{s2}, \cdots, a_{sn})$,

$\mathbf{b}_s = (b_{s1}, b_{s2}, \cdots, b_{sm})$, $s = 1, 2, \cdots, p$,

Weight matrix $\mathbf{W}$ by means of the following system of fuzzy relation equations can be obtained:

$$\begin{cases} \mathbf{a}_1 \circ \mathbf{W} = \mathbf{b}_1 \\ \mathbf{a}_2 \circ \mathbf{W} = \mathbf{b}_2 \\ \quad \cdots \\ \mathbf{a}_p \circ \mathbf{W} = \mathbf{b}_p \end{cases} .$$

Morever, a_s and b_s can be calculated respectively as follow:

where $\mathbf{a}_s = (a_{s1}, a_{s2}, \cdots, a_{sn})$,

$\mathbf{b}_s = (b_{s1}, b_{s2}, \cdots, b_{sm})$, $s = 1, 2, \cdots, p$,

Weight matrix $\mathbf{W}$ by means of the following system of fuzzy relation equations can be obtained:

$$\begin{cases} \mathbf{a}_1 \circ \mathbf{W} = \mathbf{b}_1 \\ \mathbf{a}_2 \circ \mathbf{W} = \mathbf{b}_2 \\ \quad \cdots \\ \mathbf{a}_p \circ \mathbf{W} = \mathbf{b}_p \end{cases} .$$

Morever, a_s and b_s can be calculated respectively as follow:

$$\mathbf{A} = \begin{pmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \vdots \\ \mathbf{a}_p \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ & \cdots & & \\ a_{p1} & a_{p2} & \cdots & a_{pn} \end{pmatrix}$$

and,

$$\mathbf{B} = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_p \end{pmatrix} = \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1m} \\ b_{21} & b_{22} & \cdots & b_{2m} \\ & \cdots & & \\ b_{p1} & b_{p2} & \cdots & b_{pm} \end{pmatrix} \quad ;$$

Weight matrix $\mathbf{W}$ by means of the following system of fuzzy relation equations can be obtained:

$$\mathbf{A} \circ \mathbf{W} = \mathbf{B}.$$

A Fuzzy δ Learning Algorithm

Neural Network Representation of Fuzzy Relation Equations

Describing procedures for the fuzzy δ learning algorithm:

Step 1 Rdandomize w_{ij} initial valuse w_{ij}^0

$$w_{ij} = w_{ij}^0 \quad (i = 1,2,\dots,n, j = 1,2,\dots,m).$$

Often assigned with $(w_{ij}^0 = 1), \quad (\forall\, i, j)$

Step 2 Collect a pair of samples (a_s, b_s). Let $s = 1$.

Step 3 Calculate the outputs incurred by a_s. Let $k = 1$.

$$b'_{sj} = \bigvee_{i=1}^{n} (w_{ij} \wedge a_{si}), \quad j = 1,2,\dots,m$$

Step 4 Adjust weights. Let

$$\delta_{sj} = b'_{sj} - b_{sj}, \qquad j = 1,2,\dots,m.$$

Update weights, i.e., calculate $(k+1)th$ weights based on kth weights:

$$w_{ij}(k+1) = \begin{cases} w_{ij}(k) - \eta\delta_{sj}, & w_{ij}(t) \wedge a_{sj} > b_{sj} \\ w_{ij}(k) & otherwise, \end{cases}$$

Where $0 < \eta < 1$ is the learning rate.

Step 5 Looping. Go to Step 3 until the following condition holds:

$$(\forall\, ij)\left(w_{ij}(k) - w_{ij}(k+1) < \varepsilon\right),$$

where $\varepsilon > 0$ is small number for stopping the algorithm. set $k = k + 1$.

Step 6 repeat a new input. Let $s = s + 1$ and go to Step 2 until $s = p$

Example:

Given samples $\mathbf{a}_s, \mathbf{b}_s, \quad s = 1, 2, 3, 4$:

$$\begin{aligned}
\mathbf{a}_1 &= (0.3, 0.4, 0.5, 0.6), \quad \mathbf{b}_1 = (0.6, 0.4, 0.5), \\
\mathbf{a}_2 &= (0.7, 0.2, 1.0, 0.1), \quad \mathbf{b}_2 = (0.7, 0.7, 0.7), \\
\mathbf{a}_3 &= (0.4, 0.3, 0.9, 0.8), \quad \mathbf{b}_3 = (0.8, 0.4, 0.5), \\
\mathbf{a}_4 &= (0.2, 0.1, 0.2, 0.3), \quad \mathbf{b}_4 = (0.3, 0.3, 0.3).
\end{aligned}$$

So,

$$\mathbf{A} = \begin{pmatrix} 0.3 & 0.4 & 0.5 & 0.6 \\ 0.7 & 0.2 & 1.0 & 0.1 \\ 0.4 & 0.3 & 0.9 & 0.8 \\ 0.2 & 0.1 & 0.2 & 0.3 \end{pmatrix}, \mathbf{B} = \begin{pmatrix} 0.6 & 0.4 & 0.5 \\ 0.7 & 0.7 & 0.7 \\ 0.8 & 0.4 & 0.5 \\ 0.3 & 0.3 & 0.3 \end{pmatrix}$$

and

$$\begin{pmatrix} 0.3 & 0.4 & 0.5 & 0.6 \\ 0.7 & 0.2 & 1.0 & 0.1 \\ 0.4 & 0.3 & 0.9 & 0.8 \\ 0.2 & 0.1 & 0.2 & 0.3 \end{pmatrix} \circ \begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \\ w_{41} & w_{42} & w_{43} \end{pmatrix} = \begin{pmatrix} 0.6 & 0.4 & 0.5 \\ 0.7 & 0.7 & 0.7 \\ 0.8 & 0.4 & 0.5 \\ 0.3 & 0.3 & 0.3 \end{pmatrix}.$$

When $0.5 < \eta \leq 1$ and $\varepsilon = 0.0001$

at $k = 80$ we have stable $\mathbf{W}$ as follows:

$$\mathbf{W} = \begin{pmatrix} 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.0 \\ 0.7 & 0.4 & 0.5 \\ 1.0 & 0.4 & 0.5 \end{pmatrix}.$$

several tests show that most values in **W** are the same

Except for w_{31}, w_{33}, and w_{43}.

The following table details the difference.

η	k	w_{31}	w_{33}	w_{43}
0.5	63	0.700001	0.500001	0.400001
0.4	81	0.700001	0.500001	0.420001
0.3	110	0.700002	0.500002	0.440001
0.3	167	0.700003	0.500004	0.460003
0.1	324	0.700009	0.500009	0.480008

BP learning algorithm of FAM's

The back propagation (BP) algorithm for the connection weight matrix W of the FAM is presented as follow

$$y_j = \bigvee_{i=1}^{n} \{x_i \wedge w_{ij}\} \ (j = 1, \ldots, m).$$

Define $d-$variate functions $La,\ Sm : \mathbb{R}_+ \times [0,1]^d \longrightarrow [0,1]$ as

$$La(s; x_1, \ldots, x_d) = \frac{\sum_{i=1}^{d} x_i \cdot \exp\{s x_i\}}{\sum_{i=1}^{d} \exp\{s x_i\}}; \quad Sm(s; x_1, \ldots, x_d) = \frac{\sum_{i=1}^{d} x_i \cdot \exp\{-s x_i\}}{\sum_{i=1}^{d} \exp\{-s x_i\}}.$$

it is easy to show, $\forall x_1, ..., x_d \in [0,1]$, the following facts hold:

$$\forall s > 0,\ La(s; x_1, ..., x_d),\ \ Sm(s; x_1, ..., x_d) \in \left[x_1 \wedge \cdots \wedge x_d,\ x_1 \vee \cdots \vee x_d\right].$$

Lemma 1: suppose $d>1$, $s>0$, and $x_1, x_2, \ldots, x_d \in [0,1]$. Then the following estimation gets:

$$\left|\bigvee_{i=1}^{d}\{x_i\} - La(s; x_1, ..., x_d)\right| \le (d-1)\cdot \exp\{-s(x_{\max} - x_\delta)\};$$

$$\left|\bigwedge_{i=1}^{d}\{x_i\} - Sm(s; x_1, ..., x_d)\right| \le (d-1)\cdot \exp\{-s(x_\rho - x_{\min})\}.$$

Therefore,

$$\lim_{s\to+\infty} La(s; x_1, ..., x_d) = \bigvee_{i=1}^{d}\{x_i\},\ \lim_{s\to+\infty} Sm(s; x_1, ..., x_d) = \bigwedge_{i=1}^{d}\{x_i\}.$$

Lemma 2 the functions $La(s;x_1,\ldots, x_d)$ and $Sm(s;x_1,\ldots, x_d)$ are continuously differentiable on [0,1]^d. moreover,

for $j \in \{1,\ldots, d\}$,

$$\text{(i)}\ \frac{\partial La(s;x_1,\ldots,x_d)}{\partial x_j} = \frac{-\exp(sx_j)}{\left(\sum_{i=1}^{d}\exp(sx_i)\right)^2}\left\{\sum_{i=1}^{d}(sx_i - sx_j - 1)\exp(sx_i)\right\};$$

$$\text{(ii)}\ \frac{\partial Sm(s;x_1,\ldots,x_d)}{\partial x_j} = \frac{\exp(-sx_j)}{\left(\sum_{i=1}^{d}\exp(-sx_i)\right)^2}\left\{\sum_{i=1}^{d}(sx_i - sx_j + 1)\exp(-sx_i)\right\}.$$

By **Lemma 2**, It can be concluded that the following facts hold for the constant $a \in [0,1]$:

$$\begin{cases}\dfrac{\mathrm{d}La(s;x,a)}{\mathrm{d}x} = \dfrac{1}{(1+\exp(s(a-x)))^2}\{1-(sa-sx-1)\exp(s(a-x))\};\\ \dfrac{\mathrm{d}Sm(s;x,a)}{\mathrm{d}x} = \dfrac{1}{(1+\exp(-s(a-x)))^2}\{1+(sa-sx+1)\exp(-s(a-x))\}.\end{cases}$$

So for a given constant $a \in [0,1]$, it follows that

$$\lim_{s\to+\infty}\frac{\mathrm{d}La(s;x,a)}{\mathrm{d}x}=\begin{cases}\dfrac{\mathrm{d}(a\vee x)}{\mathrm{d}x}, & x\neq a,\\ \dfrac{1}{2}, & x=a;\end{cases}$$

$$\lim_{s\to+\infty}\frac{\mathrm{d}Sm(s;x,a)}{\mathrm{d}x}=\begin{cases}\dfrac{\mathrm{d}(a\wedge x)}{\mathrm{d}x}, & x\neq a,\\ \dfrac{1}{2}, & x=a.\end{cases}$$

BP learning algorithm

Suppose $\{(x_k, y_k) | k \in P\}$ is a fuzzy pattern pair family for training.

The input pattern x_k of FAM equations, the corresponding real output pattern will be:

$$\mathbf{o}_k = (o_1^k, \ldots, o_m^k) : \mathbf{o}_k = \mathbf{x}_k \circ W,$$

Where:

$$o_j^k = \bigvee_{i \in \mathrm{N}} \{x_i^k \wedge w_{ij}\} \ (k \in \mathrm{P}, j \in \mathrm{M}).$$

The error function $E(W)$ is defined as follows:

$$E(W) = \frac{1}{2} \sum_{k=1}^{p} \|\mathbf{o}_k - \mathbf{y}_k\|^2 = \frac{1}{2} \sum_{k=1}^{p} \sum_{j=1}^{m} (o_j^k - y_j^k)^2.$$

- *As* $E(W)$ is non-differentiable respect to w_{ij}, *the* **BP** algorithm can not be designed by using $E(W)$ *directly.*
- So the functions La and Sm *is used to r*eplaced the fuzzy operators V and Λ, respectively.
- By **Lemma 1**, when s is sufficiently large

- By **Lemma 1**, when s is sufficiently large

$$E(W) \approx e(W) \triangleq \frac{1}{2}\sum_{k=1}^{p}\sum_{j=1}^{m}\left(La\left(s; Sm(s; x_1^k, w_{1j}), \ldots, Sm(s; x_n^k, w_{nj})\right) - y_j^k\right)^2.$$

Theorem 1 Give the fuzzy pattern pair family $\{(x_k, y_k) | k \in P\}$.

Then e(W) is continuously differentiable with respect to w_{ij} for

i ∈ N, j ∈ M

$$\frac{\partial e(W)}{\partial w_{ij}} = \sum_{k=1}^{p} \frac{-\exp\left(s \cdot \triangle(i,k)\right)\Gamma(s)}{\left(\sum_{p=1}^{d}\exp\left(s \cdot \triangle(p,k)\right)\right)^2} \cdot \frac{1+(sx_i^k - sw_{ij}+1)\exp(-s(x_i^k - w_{ij}))}{(1+\exp(-s(x_i^k - w_{ij})))^2}.$$

Where

$$\Gamma(s) = \sum_{p=1}^{d}\left\{s \cdot \exp\left(s \cdot \triangle(p,k)\right) - \exp\left(s \cdot \triangle(i,k)\right) - 1\right\}\exp\left(s \cdot \triangle(p,k)\right),$$

and

$$\triangle(i,k) = Sm(s; x_i^k, w_{ij}).$$

Step of BP learning algorithm of FAM's :

***Step* 1.** Initialization. Put $w_{ij}(0) = 0$, and let $W(0) = (w_{ij}(0))_{n\times m}$, set $t = 1$.

Step 2. Denote $W(t) = (w_{ij}(t))_{n\times m}$.

Step 3. Iteration scheme. W(t) iterates with the following law:

$$\Omega = w_{ij}(t) - \delta \cdot \frac{\partial e(W(t))}{\partial w_{ij}(t)} + \alpha \cdot \triangle w_{ij}(t), \quad w_{ij}(t+1) = (\Omega \vee 0) \wedge 1.$$

Step 4. Stop condition. Discriminate $|e(W(t+1))| < \varepsilon$? If yes, output $w_{ij}(t+1)$; otherwise, let $t = t + 1$ go to Step 2.

Example:

Choose $\alpha = 0.05$, $\eta = 0.3$. Let $s = 100$. With 1000 iterations,

No.	Input pattern	Desired output	Real pattern
1	(0.64, 0.50, 0.70, 0.60)	(0.64, 0.70)	(0.6400, 0.7000)
2	(0.40, 0.45, 0.80, 0.65)	(0.65, 0.80)	(0.6500, 0.7867)
3	(0.75, 0.70, 0.35, 0.25)	(0.75, 0.50)	(0.7250, 0.5325)
4	(0.33, 0.67, 0.35, 0.50)	(0.67, 0.50)	(0.6700, 0.5000)
5	(0.65, 0.70, 0.90, 0.75)	(0.75, 0.80)	(0.7500, 0.7867)
6	(0.95, 0.30, 0.45, 0.60)	(0.80, 0.60)	(0.7250, 0.6000)
7	(0.80, 1.00, 0.85, 0.70)	(0.80, 0.80)	(0.7864, 0.7867)
8	(0.10, 0.50, 0.70, 0.65)	(0.65, 0.70)	(0.6500, 0.7000)
9	(0.70, 0.70, 0.25, 0.56)	(0.70, 0.56)	(0.7000, 0.5600)

BP learning algorithm

Modifications in weight space are performed according to:

$$\Delta w_{ij}(n+1) = \varepsilon E(n) + \alpha \Delta w_{ij}(n)$$

with learning error:

$$E(n) = \delta_j o_i$$

Output layer:

$$\delta_j = (o_j + \beta)(1 - o_j)(t_j - o_j)$$

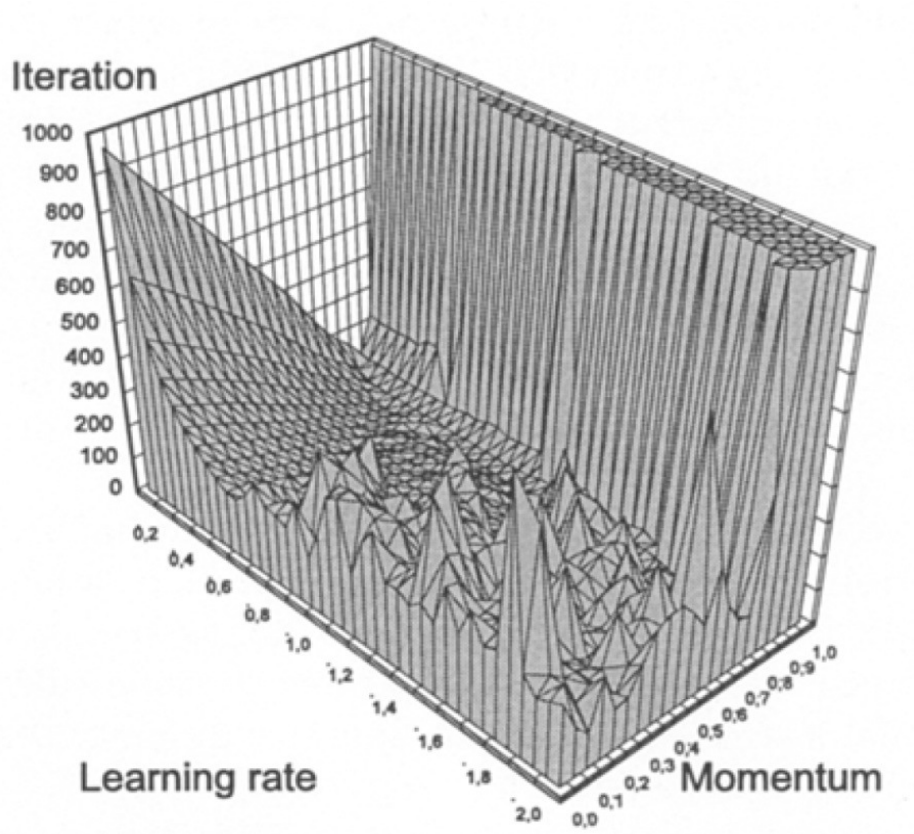
Hidden layer:

$$\delta_j = (o_j + \beta)(1 - o_j) \sum_j \delta_j w_{ij}$$

Where learning rate ε:, momentum α and offset β are constant values chosen by experience.

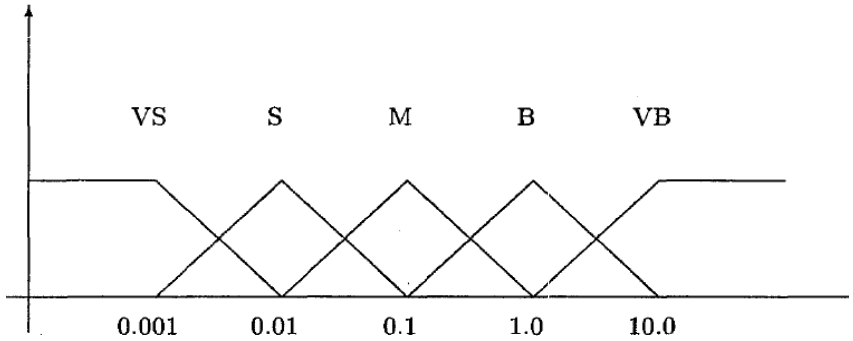
Example for Standard Backpropagation:

a set of 10 patterns (alphabetic letters from 'a' to 'j') has been presented for $0.1<\varepsilon<2.0$ and $0<\alpha<1$

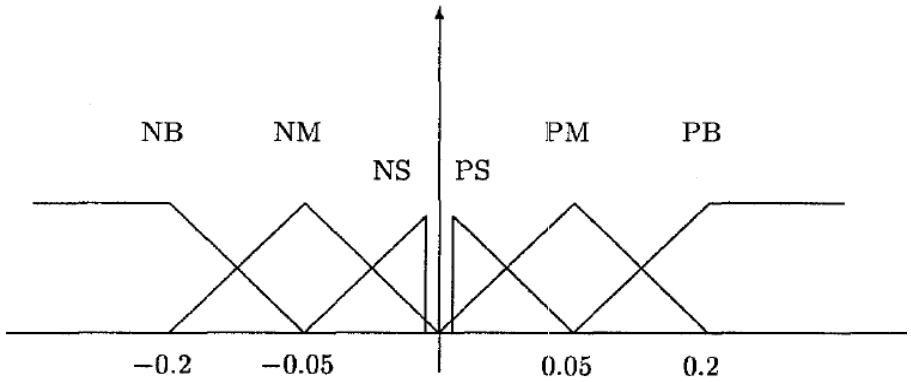


Fuzzy Adaption of Learning Parameters

Fuzzification of learning error *E(n)*

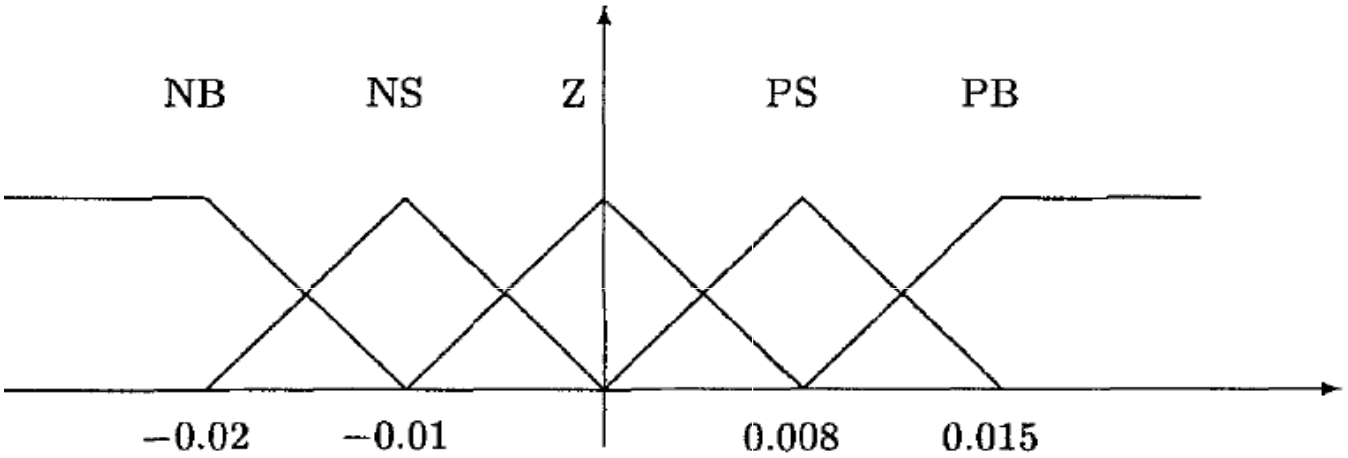


Fuzzification of error changes ΔE(n)

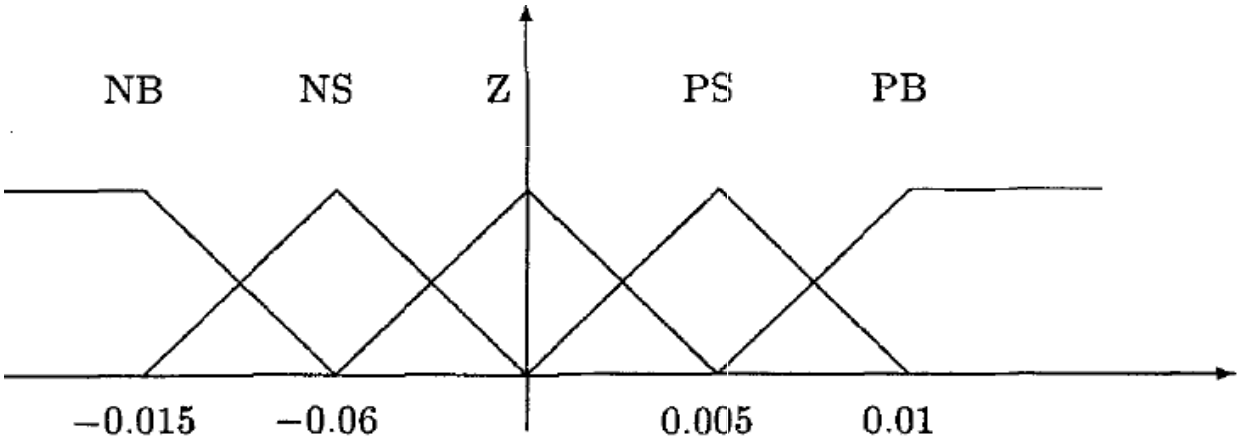


- Fuzzy set learning rate NB is restricted to -0.04 and PB is restricted to 0.02
- Fuzzy sets momentum NB and PB are restricted to -0.05 and 0.02

Fuzzified conclusions (adaption of learning rate)



Fuzzified conclusions (adaption of momentum)



The center of gravity defuzzification method has been proposed

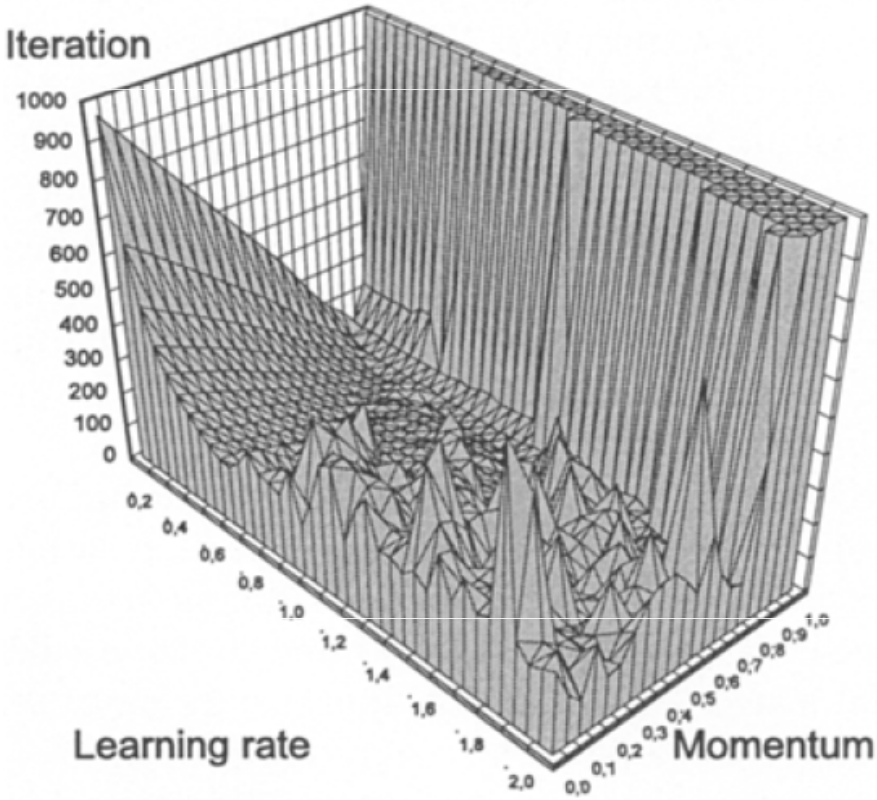
Decision rule base for learning rate ε

ΔE	Neg. Big	Neg. Medium	Neg. Small	Pos. Small	Pos. Medium	Pos. Big
$\|\|E\|\|$						
VERY SMALL	NB	NS	NS	Z	Z	PS
SMALL	NB	NS	Z	Z	Z	PS
MEDIUM	NB	NS	Z	Z	PS	PS
BIG	NB	NS	NS	PS	PS	PB
VERY BIG	NB	NB	NS	PS	PB	PB

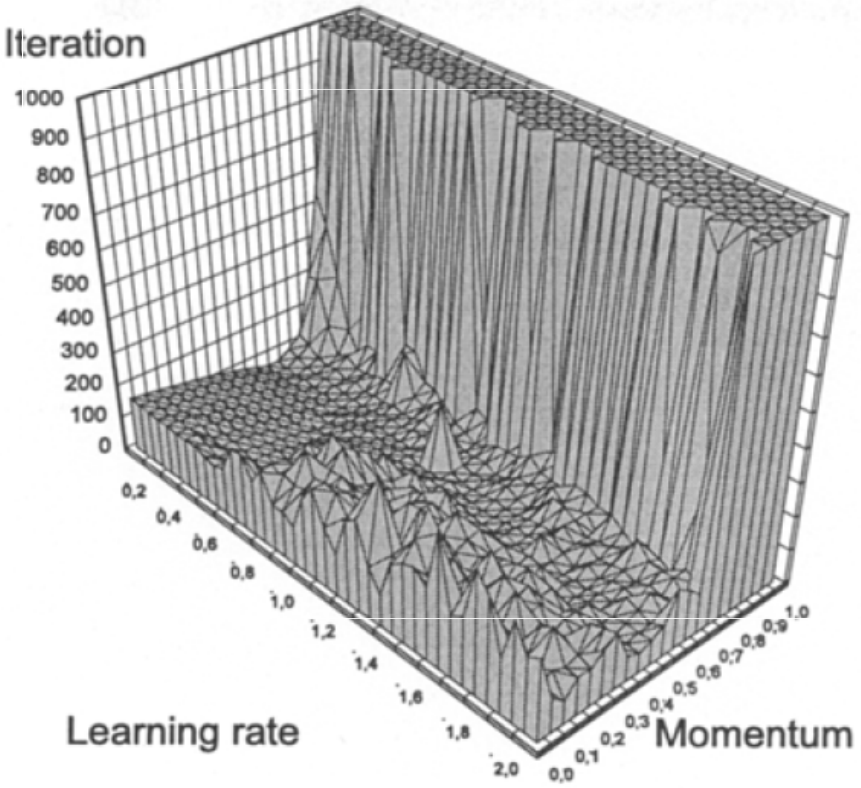
Decision rule base for momentum α

ΔE	Neg. Big	Neg. Medium	Neg. Small	Pos. Small	Pos. Medium	Pos. Big
$\|\|E\|\|$						
VERY SMALL	NB	NS	NS	Z	Z	PS
SMALL	NB	NS	NS	Z	Z	PS
MEDIUM	NB	NS	Z	Z	PS	PS
BIG	NB	NS	Z	Z	PS	PB
VERY BIG	NB	NB	NS	Z	PS	PB

Standard and fuzzy control of learning rate and momentum



Standard BP



Fuzzy BP

- Puyin Liu, Hongxing Li, FUZZY NEURAL NETWORK THEORY AND APLICATION, World Scientific, 2004
- Hongxing Li, C.L. Philip Chen, Han-Pang Huang, Fuzzy Neural Intelligent Systems, CRC Press, 2001
- Gerke .M, Hoyer .H, Fuzzy Backpropagation Training of Neural Networks, Lecture Notes in Computer Science, 1997, Volume 1226, Computational Intelligence Theory and Applications, Pages 416-427