Hybrid Systems

Integration of Neural Network, Fuzzy Logic & Genetic Algorithm

Soft Computing

Hybrid systems, topic : Integration of neural networks, fuzzy logic and genetic algorithms; Hybrid systems - sequential, auxiliar, and embedded; Neuro-Fuzzy hybrid - integration of NN and FL; Neuro-Genetic hybrids - integration of GAs and NNs ; Fuzzy-Genetic hybrids - integration of FL and GAs. Genetic Algorithms Based Back Propagation Networks : hybridization of BPN and GAs; Genetic algorithms based techniques for determining weights in a BPN - coding, weight extraction, fitness function algorithm, reproduction of offspring, selection of parent chromosomes, convergence. Fuzzy back propagation networks : LR-type fuzzy numbers, operations on LR-type fuzzy numbers; Fuzzy neuron; Architecture of fuzzy BP. Fuzzy associative memories : example of FAM Model of washing machine - variables, operations, representation, defuzzification. Simplified fuzzy ARTMAP : supervised ARTMAP system, comparing ARTMAP with back-propagation networks.

Hybrid Systems

Integration of Neural Network, Fuzzy Logic & Genetic Algorithm

Soft Computing

Topics

(Lectures 41, 42   2 hours)

Slides

Hybrid Systems

Integration of NN FL GA

What is Hybridization ?

- Hybrid systems employ more than one technology to solve a problem.

- Hybridization of technologies can have pitfalls and therefore need to be done with care.

- If one technology can solve a problem then a hybrid technology ought to be used only if its application results in a better solution.

- Hybrid systems have been classified as :

  - *Sequential hybrid system*: the technologies are used in pipelining fashion;
  - *Auxiliary hybrid system*: the one technology calls the other technology as subroutine;
  - *Embedded hybrid system* : the technologies participating appear to be fused totally.

- Hybridization of fuzzy logic, neural networks, genetic algorithms has led to creation of a perspective scientific trend known as soft computing.
  - *Neural networks* mimic our ability to adapt to circumstances and learn from past experience,
  - *Fuzzy logic* addresses the imprecision or vagueness in input and output,
  - *Genetic algorithms* are inspired by biological evolution, can systemize random search and reach to optimum characteristics.

- Each of these technologies have provided efficient solution to wide range of problems belonging to different domains. However, each of these technologies has advantages and disadvantages. It is therefore appropriate that Hybridization of these three technologies are done so as to over come the weakness of one with the strength of other.

1. Introduction :

Hybridization - Integration of NN , FL , and GA

Fuzzy logic, Neural networks and Genetic algorithms are soft computing methods which are inspired by biological computational processes and nature's problem solving strategies.

Neural Networks (NNs) are highly simplified model of human nervous system which mimic our ability to adapt to circumstances and learn from past experience. Neural Networks systems are represented by different architectures like single and multilayer feed forward network. The networks offers back proposition generalization, associative memory and adaptive resonance theory.

Fuzzy logic addresses the imprecision or vagueness in input and output description of the system. The sets have no crisp boundaries and provide a gradual transition among the members and non-members of the set elements.

Genetic algorithms are inspired by biological evolution, can systemize random search and reach to optimum characteristics.

Each of these technologies have provided efficient solution to wide range of problems belonging to different domains. However, each of these technologies suffer from advantages and disadvantages.

It is therefore appropriate that Hybridization of these three technologies are done so as to over come the weakness of one with the strength of other.

04

1.1 Hybrid Systems

Hybrid systems employ more than one technology to solve a problem. Hybridization of technologies can have pitfalls and therefore need to be done with care. If one technology can solve a problem then a hybrid technology ought to be used only if its application results in a better solution. Hybrid systems have been classified as *Sequential , Auxiliary and Embedded*.

In *Sequential hybrid system*, the technologies are used in pipelining fashion.

In *Auxiliary hybrid system*, one technology calls the other technology as subroutine.

In *Embedded* hybrid system, the technologies participating appear to be fused totally.

05

● Sequential Hybrid System

In Sequential hybrid system, the technologies are used in pipelining fashion. Thus, one technology's output becomes another technology's input and it goes on. However, this is one of the weakest form of hybridization since an integrated combination of technologies is not present.

Example: A Genetic algorithm preprocessor obtains the optimal parameters for different instances of a problem and hands over the preprocessed data to a neural network for further processing.

06

● **Auxiliary Hybrid System**

In Auxiliary hybrid system, one technology calls the other technology as subroutine to process or manipulate information needed. The second technology processes the information provided by the first and hands it over for further use. This type of hybridization is better than the sequential hybrids.

Example : A neuron-genetic system in which a neural network employs a genetic algorithm to optimize its structural parameters that defines its architecture.

07

● **Embedded Hybrid System**

In Embedded hybrid system, the technologies participating are integrated in such a manner that they appear intertwined. The fusion is so complete that it would appear that no technology can be used without the others for solving the problem.

Example : A NN-FL hybrid system may have an NN which receives fuzzy inputs, processes it and extracts fuzzy outputs as well.

08

1.2 Neural Networks, Fuzzy Logic, and Genetic Algorithms Hybrids

Neural Networks, Fuzzy Logic, and Genetic Algorithms are three distinct technologies.

Each of these technologies has advantages and disadvantages.   It   is therefore appropriate that hybridization of these three technologies are done so as to over come the weakness of one with the strength of other.

09

● **Neuro-Fuzzy Hybrid**

Neural Networks and Fuzzy logic represents two distinct methodologies to deal with uncertainty. Each of these has its own merits and demerits.

Neural Networks :

– Merits : Neural Networks, can model complex nonlinear relationships and are appropriately suited for classification phenomenon into predetermined classes.

– Demerits : Neural Network's output, precision is often limited to least squares errors; the training time required is quite large; the training data has to be chosen over entire range where the variables are expected to change.

Fuzzy logic :

– Merits : Fuzzy logic system, addresses the imprecision of inputs and outputs defined by fuzzy sets and allow greater flexibility in formulating detail system description.

Integration of NN and FL, called Neuro-Fuzzy systems, have the potential to extend the capabilities of the systems beyond either of these two technologies applied individually. The integrated systems have turned out to be useful in :

– accomplishing mathematical relationships among many variables in a complex dynamic process,

– performing mapping with some degree of imprecision, and

– controlling nonlinear systems to an extent not possible with conventional linear control systems.

There are two ways to do hybridization :

– One, is to provide NNs with fuzzy capabilities, there by increasing the network's expressiveness and flexibility to adapt to uncertain environments.

– Second, is to apply neuronal learning capabilities to fuzzy systems so that the fuzzy systems become more adaptive to changing environments. This method is called NN driven fuzzy reasoning.

● **Neuro-Genetic Hybrids**

The Neural Networks and Genetic Algorithms represents two distinct methodologies.

Neural Networks :   can learn various tasks from examples, classify phenomena and model nonlinear relationships.

Genetic Algorithms : have offered themselves as potential candidates for the optimization of parameters of NN.

Integration of GAs and NNs has turned out to be useful.

– Genetically evolved nets have reported comparable results against their conventional counterparts.

– The gradient descent learning algorithms have reported   difficulties   in leaning the topology of the networks whose weights they optimize.

– GA based algorithms   have provided   encouraging   results   especially with regard to face recognition, animal control, and others.

– Genetic algorithms   encode the parameters of NNs as a string of properties of the network, i.e. chromosomes. A large population of chromosomes   representing many possible parameters sets, for the given NN, is generated.

– GA-NN is also known as GANN have the ability to locate the neighborhood of the optimal solution quicker than other conventional search strategies.

– The drawbacks of GANN algorithms are : large amount of memory required to handle and manipulate chromosomes for a given network; the question is whether this problem scales as the size of the networks become large.

11

● Fuzzy-Genetic Hybrids

Fuzzy systems have been integrated with GAs.

The fuzzy systems like NNs (feed forward) are universal approximator in the sense that they exhibit the capability to approximate general nonlinear functions to any desired degree of accuracy.

The adjustments of system parameters called for in the process, so that the system output matches the training data, have been tackled using GAs. Several parameters which a fuzzy system is involved with like input/output variables and the membership function that define the fuzzy systems, have been optimized using GAs.

12

1.3 Typical Hybrid Systems

The Systems considered are listed below.

1. Genetic algorithm based back propagation network

   (Neuro Genetic Hybrid)

2. Fuzzy back propagation network

   (Neuro – Fuzzy Hybrid with Multilayer Feed forward Network as the host architecture)

3. Simplified Fuzzy ARTMAP

   (Neuro – Fuzzy Hybrid with Recurrent Network as the host architecture)

4. Fuzzy Associative Memory

   (Neuro – Fuzzy Hybrid with single layer Feed forward architecture)

5. Fuzzy logic controlled Genetic algorithm

   (Fuzzy – Genetic Hybrid)

13

2. Genetic Algorithm (GA) based Back Propagation Network (BPN)

Neural networks (NNs) are the adaptive system that changes its structure based on external or internal information that flows through the network. Neural network solve problems by self-learning and self-organizing.

Back Propagation Network (BPN) is a method of training multi-layer neural networks. Here learning occurs during this training phase.

The steps involved are:

− The pattern of activation arriving at the output layer is compared with the correct output pattern to calculate an error signal.

− The error signal is then back-propagated from output to input for adjusting the weights in each layer of the BPN.

− The Back-Propagation searches on the error surface using gradient descent method to minimize error $E = 1/2 \Sigma (T_j - O_j)^2$ where T_j is target output and O_j is the calculated output by the network.

Limitations of BPN :

− BPN can recognize patterns similar to those they have learnt, but do not have the ability to recognize new patterns.

− BPN must be sufficiently trained to extract enough general features applicable to both seen and unseen; over training to network may have undesired effects.

14

Genetic Algorithms (GAs) are adaptive search and optimization algorithms, mimic the principles of nature.

– GAs are different form traditional search and

– Optimization exhibit simplicity, ease of operation, minimal requirements, and global perspective.

Hybridization of BPN and GAs

– The BPN determines its weight based on gradient search technique and therefore it may encounter a local minima problem.

– GAs do not guarantee to find global optimum solution, but are good in finding quickly good acceptable solution.

– Therefore, hybridization of BPN and GAs are expected to provide many advantages compare to what they alone can.

The GA based techniques for determining weights in a BPN are explained next.

15

2.1 GA based techniques for determining weights in a BPN

Genetic algorithms work with population of individual strings.

The steps involved in GAs are:

– each individual string represent a possible solution of the problem considered,

– each individual string is assigned a fitness value,

– high fit individuals participate in reproduction, yields new strings as offspring and they share some features with each parents,

– low fit individuals are kept out from reproduction and so die,

– a whole new population of possible solutions to the problem is generated by selecting high fit individuals from current generation,

– this new generation contains characteristics which are better than their ancestors,

– processing this way after many generation, the entire population inherits the best and fit solution.

However, before a GA is executed :

– a suitable **coding** for the problem is devised,

– a **fitness function** is formulated,

– parents have to be **selected** for reproduction and crossover to generate offspring.

All these aspects of GAs for determining weights of BPN are illustrated in next few slides.

● **Coding**

Assume a BPN configuration **ℓ - m − n** where

  − **ℓ** is input , **m** is hidden and **n** is output neurons.

  − number of weights to be determined are **(ℓ + n) m.**

  − each weight (gene) is a real number.

  − assume number of digits (gene length) in weight are **d** .

  − a string **S** represents weight matrices of input-hidden and the hidden-output layers in a linear form arranged as row-major or column-major selected.

  − population size is the randomly generated initial population of **p** chromosomes.

Example :

Consider a BPN configuration **ℓ - m − n** where **ℓ = 2** is input , **m = 2** is hidden and **n = 2** is output neuron.

Input neuron Hidden neurons output neurons
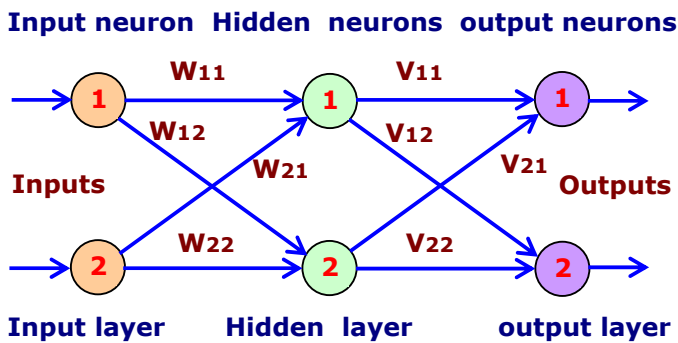

Fig. BPN with 2 − 2 - 2

  − number of weights is **(ℓ + n) m**
     = (2 + 2) . 2 = 8

  − each weight is real number and assume number   of   digits in weight are   **d = 5**

  − string **S** representing chromosome of weights is **8 x 5 = 40** in length

  − Choose a population size **p = 40** ie   choose **40** chromosome**s**

Gene ←k=0 →	*Gene* ←k=1 →	*Gene* ←k=2 →	*Gene* ←k=3 →	*Gene* ←k=4 →	*Gene* ←k=5 →	*Gene* ←k=6 →	*Gene* ←k=7 →
84321	46234	78901	32104	42689	63421	46421	87640

←———————————— Chromosome ————————————→

32478	76510	02461	84753	64321	14261	87654	12367

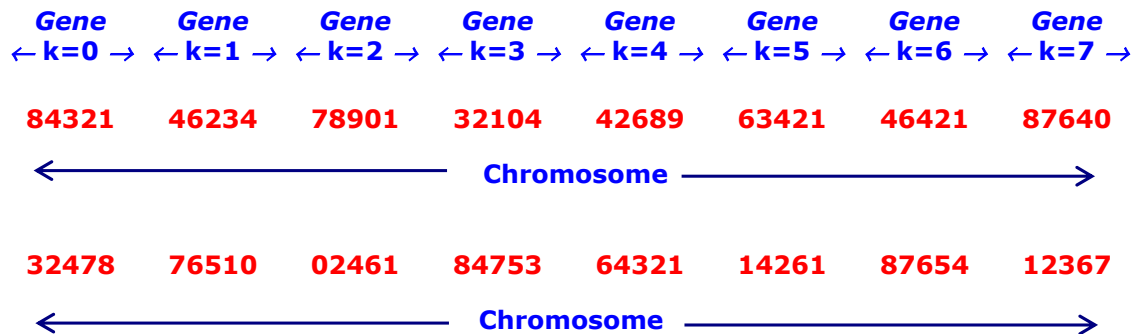←———————————— Chromosome ————————————→

Fig.   Some randomly generated chromosome made of 8 genes representing 8 weights for BPN

• **Weight Extraction**

Extract weights from each chromosomes, later to determine the fitness values.

Let $x_1, x_2, \ldots x_d, \ldots x_L$ represent a chromosome and

Let $x_{kd+1}, x_{kd+2}, \ldots x_{(k+1)d}$ represent k^{th} gene $(k \geq 0)$ in the chromosomes.

The actual weight w_k is given by

$$w_k = \begin{cases} + \dfrac{x_{kd+2}\, 10^{d-2} + x_{kd+3}\, 10^{d-3} + \ldots + x_{(k+1)d}}{10^{d-2}}, & \text{if } 5 \leq x_{kd+1} \leq 9 \\[3mm] - \dfrac{x_{kd+2}\, 10^{d-2} + x_{kd+3}\, 10^{d-3} + \ldots + x_{(k+1)d}}{10^{d-2}}, & \text{if } 0 \leq x_{kd+1} < 5 \end{cases}$$

Example : *[Ref Fig. BPN previous slide]*

The Chromosomes are stated in the Fig. The weights extracted from all the eight genes are :

- **Gene 0 : 84321** ,

  Here we have, **k = 0 , d = 5** ,   and x_{kd+1} is x_1 such that

  $5 \leq x_1 = 8 \leq 9$.   Hence, the weight extracted is

  $$W_0 = + \frac{4 \times 10^3 + 3 \times 10^2 + 2 \times 10 + 1}{10^3} = +4.321$$

- **Gene 1 : 46234** ,

  Here we   have, **k = 1 , d = 5** ,   and x_{kd+1} is x_6 such that

  $0 \leq x_6 = 4 \leq 5$.   Hence, the weight extracted is

  $$W_1 = - \frac{6 \times 10^3 + 2 \times 10^2 + 3 \times 10 + 4}{10^3} = - 6.234$$

- Similarly for the remaining genes

  Gene 2 : 78901   yields   W2 = + 8.901

  Gene 3 : 32104   yields   W3 = − 2.104

  Gene 4 : 42689   yields   W4 = − 2.689

  Gene 5 : 63421   yields   W5 = + 3.421

  Gene 6 : 46421   yields   W6 = − 6.421

  Gene 7 : 87640   yields   W7 = + 7.640

18

Fitness Function :

A fitness is devised for each problem.

Example :

The matrix on the right, represents a set of input I and output T for problem P to be solved.

$$\begin{cases} (I_{11}, I21) \ (T_{11}, T21) \\ (I_{12}, I22) \ (T_{12}, T22) \\ (I_{13}, I23) \ (T_{13}, T23) \end{cases}$$

Generate initial population P_0 of size $p = 40$.

Let $C^0_1, C^0_1, \ldots, C^0_{40}$ represent the 40 chromosomes.

Let $\overline{w}^0_1, \overline{w}^0_2, \ldots \overline{w}^0_{40}$ be the weight sets extracted, using the Eq. in the previous slides, from each of the chromosome C^0_i, $i = 1, 2, \ldots, 40$.

Let $\overline{o}^0_1, \overline{o}^0_2, \overline{o}^0_3$ be the calculated outputs of BPN.

Compute root mean square error :

$$E_1 = (T_{11} - O_{11})^2 + (T_{21} - O_{21})^2,$$
$$E_2 = (T_{12} - O_{12})^2 + (T_{22} - O_{22})^2$$
$$E_3 = (T_{13} - O_{13})^2 + (T_{23} - O_{23})^2$$

The root mean square of error is

$$E = [(E_1 + E_2 + E_3) / 3]^{1/2}$$

Compute Fitness F_1 :

The fitness F_1 for the chromosome C^0_1 is given by

$$F_1 = 1 / E.$$

Similarly, find the fitness F_2 for the chromosome C^0_2 and so on the fitness F_n for the chromosome C^0_n

19

[continued from previous slide fitness fuction]

Algorithm

{

Let ($\overline{I_i}$, $\overline{T_i}$) **, i = 1 , 2 , . . . , N** represents the input-output pairs of the problem to be solved by BPN with configuration **ℓ - m – n** ; where

$\overline{I}$ = **(I$_{1i}$, I$_{2i}$,, ..., I$_{ℓ i}$)** and

$\overline{T_i}$ = **(T$_{1i}$, T$_{2i}$,, ..., T$_{n i}$)**

For each chromosome **C$_i$, i = 1 , 2 , . . . , p** belonging to current the population **P$_i$** whose size is **p**

{

Extract weights $\overline{w}_i$ form **C$_i$** using Eq. 2.1 in previous slide;

Keeping $\overline{w}_i$ as a fixed weight, train the BPN for the **N** input instances;

Calculate error E$_i$ for each of the input instances using the formula below

E$_i$ = $\sum\limits_{j}$ (T$_{ji}$ – O$_{ji}$)^2 where $\overline{O}_i$ is the output vector calculated by BPN;

Find the root mean square E of the errors **E$_i$, i = 1 , 2 , . . . , N**

i.e. **E = (($\sum\limits_{i}$ E$_i$) / N)$^{1/2}$**

Calculate the Fitness value **F$_i$** for each of the individual string of the population as **F$_i$ = 1 / E**

}

Output **F$_i$** for each **C$_i$, i = 1 , 2 , . . . , p** ;

}

20

[continued from previous slide - Fitness Function]

Thus the Fitness values F_i for all chromosomes in the initial population are computed. The population size is $p = 40$, so F_i, $i = 1$, 2, $.., 40$ are computed.

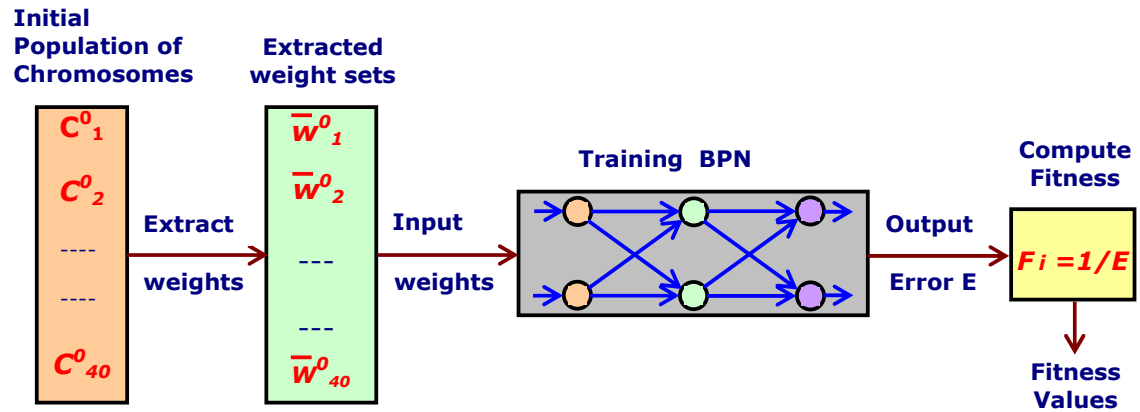A schematic for the computation of fitness values is illustrated below.

Fig. Computation of Fitness values for the population

21

● **Reproduction of Offspring**

Before the parent chromosomes reproduce offspring :

First, form a mating pool by excluding that chromosome C_ℓ with least fitness F^{min} and then replacing it with a duplicate copy of C_k with highest fitness F^{max} ;

i.e., the best fit individuals have multiple copies while worst fit individuals die off.

Having formed the mating pool, select parent pair at random. Chromosomes of respective pairs are combined using crossover operator. Fig. below shows :

  – two parent chromosomes **Pa** and **Pb**,

  – the two point crossover,

  – exchange of gene segments by the parent pairs, and

  – the offspring **Oa** and **Ob** are produced.
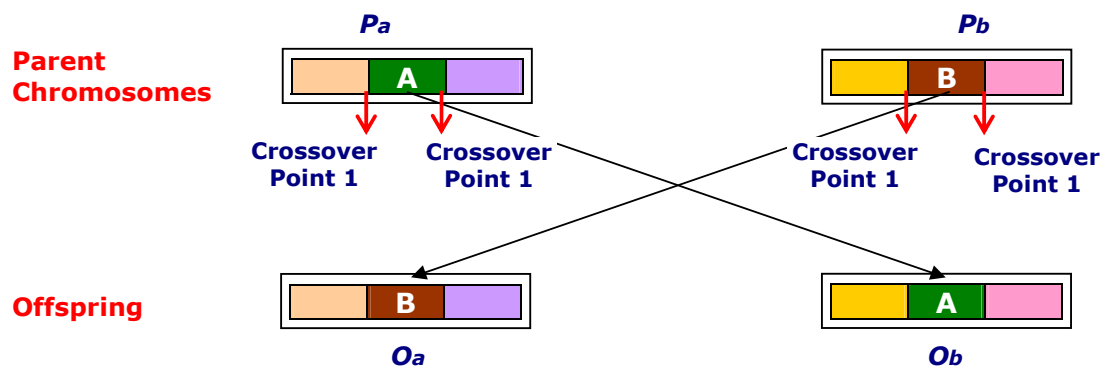
Fig. Two – point crossover operator

[continued from previous slide - Reproduction]

Example :

– Consider the initial population of chromosomes P_0 generated, with their fitness value F_i, *where* $i = 1, 2, .., 40$,

– Let $F^{max} = F_k$ be maximum and $F^{min} = F_\ell$ be minimum fitness value for $1 \leq \ell, k \leq 40$ where $\ell \neq k$

– Replace all chromosomes having fitness value F^{min} with copies of chromosomes having fitness value F^{max}

Fig. below illustrates the Initial population of chromosomes and the formation of the mating pool.
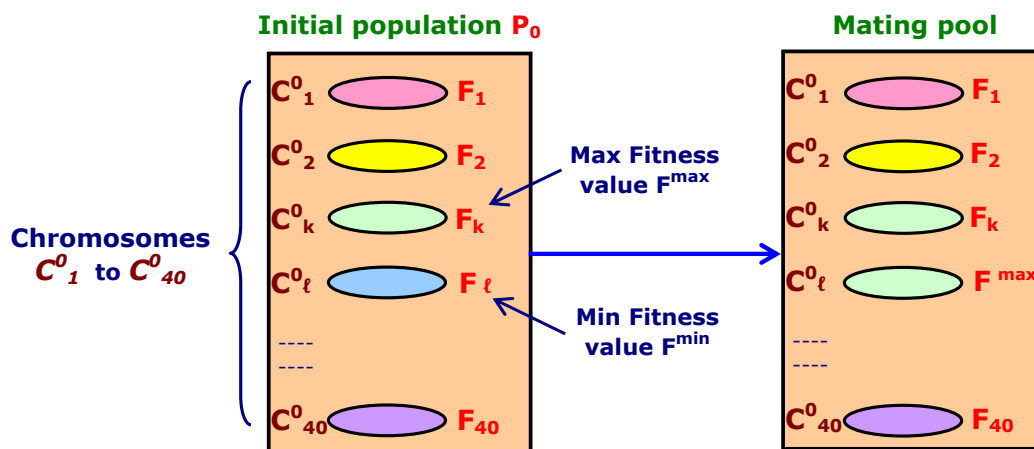
Fig. Formation of Mating pool
F^{min} **is replaced by** F^{max}

23

Selection of Parent Chromosomes

The previous slide illustrated Reproduction of the Offspring.

Here, sample *"Selection Of Parents"* for the *"Two Points Crossover"* operator to produce Offspring Chromosomes are illustrated.
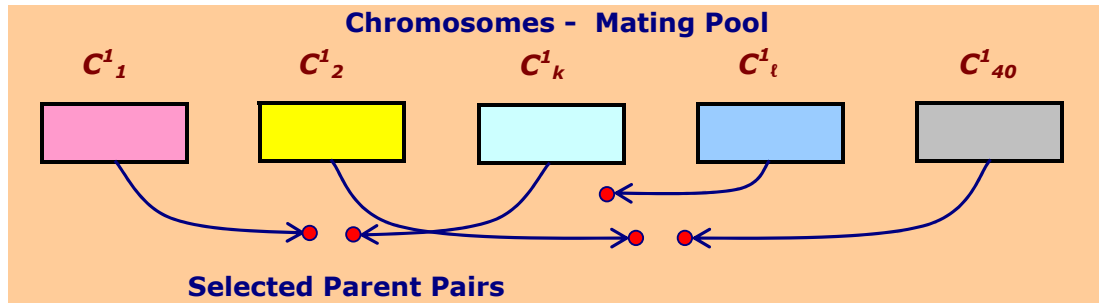
Chromosomes - Mating Pool

C^1_1 C^1_2 C^1_k C^1_ℓ C^1_{40}

Selected Parent Pairs

Fig. Random Selection of Parent Chromosomes

The Crossover Points of the Chromosomes are randomly chosen for each parent pairs as shown in the Fig. below.
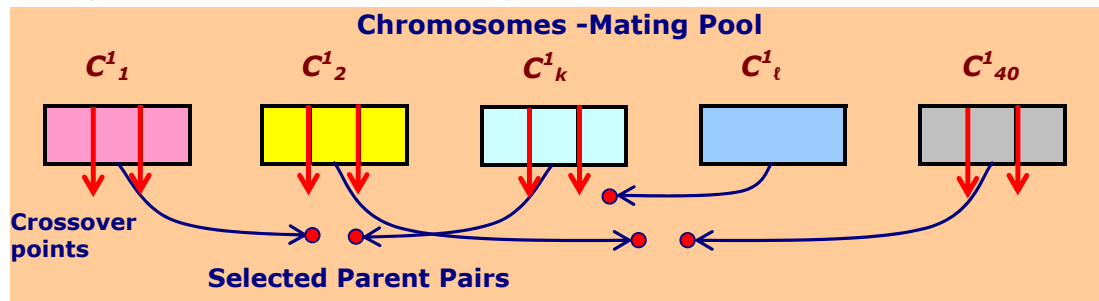
Chromosomes -Mating Pool

C^1_1 C^1_2 C^1_k C^1_ℓ C^1_{40}

Crossover points

Selected Parent Pairs

Fig. Randomly chosen Crossover points of Parent Chromosomes

The Genes are exchanged for *Mutation* as shown in the Fig. below.

Chromosomes -Mating Pool

C^1_1 C^1_2 C^1_k C^1_ℓ C^1_{40}

C^1_1 C^1_2 C^1_k C^1_ℓ C^1_{40}
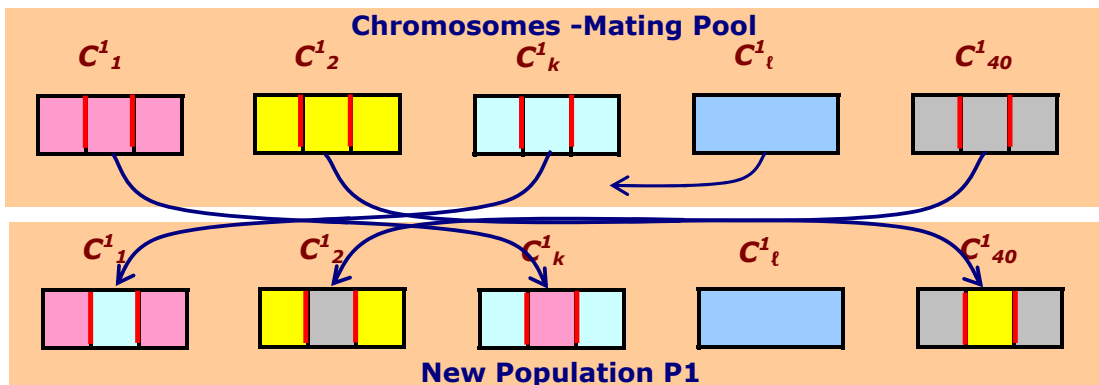
New Population P1

Fig. New population P_1 after application of two point Crossover operator

Thus new population P_1 is created comprising **40** Chromosomes which are the Offspring of the earlier population generation P_0 .

24

● **Convergence**

For any problem, if GA is correctly implemented, the population evolves over successive generations with fitness value increasing towards the global optimum.

Convergence is the progression towards increasing uniformity.

A population is said to have converged when 95% of the individuals constituting the population share the same fitness value.

Example :

Let a population P_1 undergoes the process of selection, reproduction, and crossover.

- the fitness values for the chromosomes in P_1 are computed.
- the best individuals replicated and the reproduction carried out using two-point crossover operators form the next generation P_2 of the chromosomes.
- the process of generation proceeds until at one stage 95% of the chromosomes in the population P_i converge to the same fitness value.
- at that stage, the weights extracted from the population P_i are the final weights to be used by BPN.

25

3. Fuzzy Back Propagation Network

Neural Networks and Fuzzy logic (NN-FL) represents two distinct methodologies and the integration of NN and FL is called Neuro-Fuzzy systems.

Back Propagation Network (BPN) is a method of training multi-layer neural networks where learning occurs during this training phase.

Fuzzy Back Propagation Network (Fuzzy-BPN) is a hybrid architecture. It is, Hybridization of BPN by incorporating fuzzy logic.

Fuzzy-BPN architecture, maps   fuzzy   inputs   to crisp outputs.   Here, the Neurons uses LR-type fuzzy numbers.

The   Fuzzy-Neuron   structure,   the architecture of   fuzzy BP,   its   learning mechanism and algorithms are illustrated in next few slides.

26

3.1 LR-type Fuzzy Numbers

The LR-type fuzzy number are special type of representation of fuzzy numbers. They introduce functions called **L** and **R**.

- **Definition**

A fuzzy member $\tilde{M}$ is of **L-R** type if and only if

$$\mu_{\tilde{M}}(x) = \begin{cases} L\left[\dfrac{m-x}{\alpha}\right] & \text{for } x \leq m, \ \alpha > 0 \\[2ex] R\left[\dfrac{m-x}{\beta}\right] & \text{for } x \leq m, \ \beta > 0 \end{cases}$$

where   **L**   is a left reference

       R   is a right reference,

    m ,   is called mean of $\tilde{M}$ is a real number,

   α , **β**   are left and right spreads respectively.

    $\mu_{\tilde{M}}$ is the membership function of fuzzy member $\tilde{M}$

The functions **L** and **R** are defined as follows:

$$L\left[\frac{m-x}{\alpha}\right] = \max\left(0, 1 - \left[\left|\frac{m-x}{\alpha}\right|\right]\right)$$

$$R\left[\frac{m-x}{\alpha}\right] = \max\left(0, 1 - \left[\left|\frac{m-x}{\alpha}\right|\right]\right)$$

LR-type fuzzy number $\tilde{M}$ can be represented as **(m, α, β)**$_{LR}$ shown below.
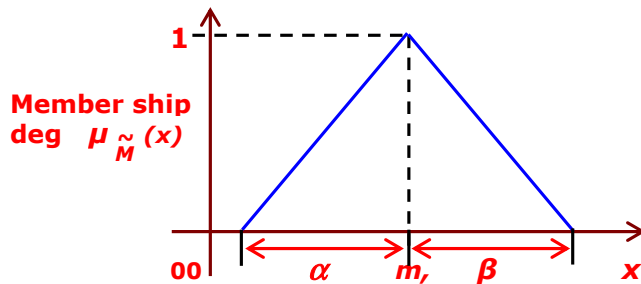

Fig. A triangular fuzzy number (m, α, β).

Note : If **α** and **β** are both zero, then **L-R** type function indicates a crisp value. The choice of **L** and **R** functions is specific to problem.

● **Operations on LR-type Fuzzy Numbers**

Let $\tilde{M}$ = $(m, \alpha, \beta)_{LR}$ and $\tilde{N}$ = $(n, \gamma, \delta)_{LR}$ be   two L R-type fuzzy

numbers.   The basic operations are

- **Addition**

  $(m, \alpha, \beta)_{LR} \oplus (n, \gamma, \delta)_{LR} = (m + n, \alpha + \gamma, \beta + \delta)_{LR}$

- **Substraction**

  $(m, \alpha, \beta)_{LR} \ominus (n, \gamma, \delta)_{LR} = (m - n, \alpha + \delta, \beta + \gamma)_{LR}$

- **Multiplicaion**

  $(m, \alpha, \beta)_{LR} \otimes (n, \gamma, \delta)_{LR} = (mn, m\gamma + n\alpha, m\delta + n\beta)_{LR}$ for m≥0 , n≥0

  $(m, \alpha, \beta)_{LR} \otimes (n, \gamma, \delta)_{LR} = (mn, m\alpha - m\delta, n\beta - m\gamma)_{RL}$ for m<0 , n≥0

  $(m, \alpha, \beta)_{LR} \otimes (n, \gamma, \delta)_{LR} = (mn, - n\beta - m\delta, -n\alpha - m\gamma)_{LR}$ for m<0 , n<0

- **Scalar Multiplicaion**

  $\lambda*(m, \alpha, \beta)_{LR} = (\lambda m, \lambda\alpha, \lambda\beta)_{LR}$, $\forall \lambda \geq 0$, $\lambda \in R$

  $\lambda*(m, \alpha, \beta)_{LR} = (\lambda m, -\lambda\alpha, -\lambda\beta)_{RL}$, $\forall \lambda < 0$, $\lambda \in R$

28

● **Fuzzy Neuron**

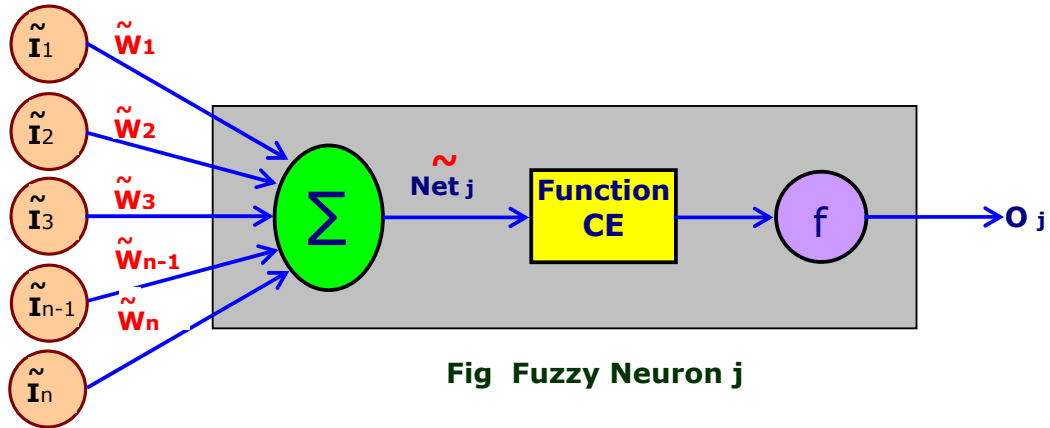The fuzzy neuron is the basic element of Fuzzy BP network. Fig. below shows the architecture of the fuzzy neuron.

Fig Fuzzy Neuron j

Given input vector $\tilde{I} = \tilde{I}_1 , \tilde{I}_2 , \cdot\cdot \tilde{I}_n$

and weight vector $\tilde{W} = \tilde{w}_1 , \tilde{w}_2 , \cdot\cdot \tilde{w}_n$

The fuzzy neuron computes the crisp output given by

$$O = f(NET) = f (CE (\sum_{i=1}^{n} \tilde{W}_i . \tilde{I}_i)) \text{ where } \tilde{I}_0 = (1, 0, 0) \text{ is the bias.}$$

Here, the fuzzy weighted summation is given by

$$\tilde{net} = \sum_{i=0}^{n} \tilde{w}_i \cdot \tilde{I}_i \text{ is first computed and}$$

$$NET = CE (\tilde{net}) \text{ is computed next}$$

The function **CE** is the centroid of triangular fuzzy number, that has **m** as mean and α , **β** as left and right spreads explained before, can be treated as defuzzification operation, which maps fuzzy weighted summation to crisp value.

If $\tilde{net} = (\tilde{net}_m , \tilde{net}\alpha , \tilde{net}\beta)$ is the fuzzy weighted summation

Then function **CE** is given by

$$CE (\tilde{net}) = CE (\tilde{net}_m , \tilde{net}\alpha , \tilde{net}\beta) = \tilde{net}_m + 1/3 (\tilde{net}\beta - \tilde{net}\alpha) = NET$$

The function **f** is a sigmoidal function that performs nonlinear mapping between the input and output. The function **f** is obtained as :

$$f(NET) = 1 / (1 + \exp (- NET)) = O \text{ is final crisp output value.}$$

[continued from previous slide – Fuzzy Neuron]

Note :

In the fuzzy neuron, both input vector $\tilde{I}_n$ and weight vector $\tilde{w}_n$ are represented by triangular LR-type fuzzy numbers.

For input vector $\tilde{I}$ = $\tilde{I}_1$, $\tilde{I}_2$, $\cdot\cdot$ $\tilde{I}_n$ e input component $\tilde{I}_i$ represented by the LR-type fuzzy number $\tilde{I}_{mi}$ $\tilde{I}_{\alpha i}$ $\tilde{I}_{\beta i}$

Similarly, for the weight vector $\tilde{W}$ = $\tilde{w}_1$, $\tilde{w}_2$, $\cdot\cdot$ $\tilde{w}_n$ weight vector component $\tilde{w}_i$ is represented as $\tilde{w}_{mi}$, $\tilde{w}_{\alpha i}$, $\tilde{w}_{\beta i}$

30

● **Architecture of Fuzzy BP**

Fuzzy Back Propagation Network (BP) is a 3-layered feed forward architecture. The 3 layers are: input layer, hidden layer and output layer. Considering a configuration of **ℓ-input** neurons, **m-hidden** neurons and **n-output** neurons, the architecture of Fuzzy BP is shown below.
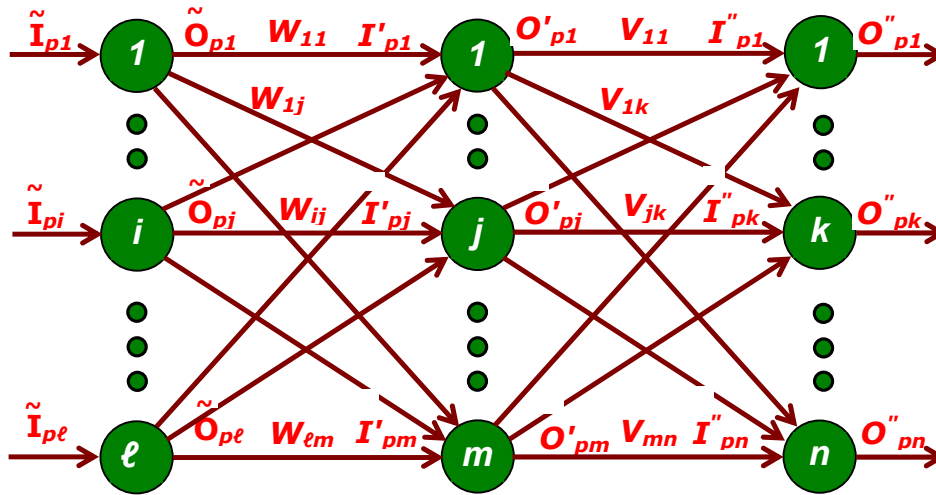

Fig. Three layer Fuzzy BP architecture.

Let $\tilde{I}_p = \tilde{I}_{p1}, \tilde{I}_{p2}, \cdots \tilde{I}_{p\ell}$ for **p = 1, 2, . . , N,** be the **p**[th] pattern among **N** input patterns that Fuzzy BP needs to be trained.

Here, $\tilde{I}_{pi}$ indicates the **i**[th] component of input pattern **p** and is an LR-type triangular fuzzy number, i.e., $\tilde{I}_p = \tilde{I}_{p1}, \tilde{I}_{p2}, \cdots I_{p\beta\ell}$

- Let $\tilde{O}_{pi}$ be the output value of **i**[th] input neuron.
- Let O'_{pj} and O'_{pk} are **j**[th] and **k**[th] crisp defuzzification outputs of the hidden and output layer neurons respectively.

- Let **W**$_{ij}$ is the fuzzy connection weight between **i**[th] input node and **j**[th] hidden node.

- Let **V**$_{jk}$ is the fuzzy connection weight between **j**[th] hidden node and **k**[th] output node.

[Continued in next slide]

31

[continued from previous slide – Architecture of Fuzzy BP]

The computations   carried out by each layer are as follows:

Input neurons:

$$\tilde{O}_{pi} \; = \; \tilde{I}_{pi} \; , \; i = 1 , 2 , \ldots , \ell \; .$$

Hidden neurons:

$$O'_{pj} \; = \; f (NET_{pj}) \; , \; i = 1 , 2 , \ldots , m \; .$$

$$\text{where} \quad NET_{pj} \; = \; C \, E \, (\sum_{i=0}^{\ell} W_{ij} \; O'_{pi})$$

Out neurons:

$$O''_{pk} \; = \; f (NET_{pk}) \; , \; i = 1 , 2 , \ldots , n \; . ,$$

$$\text{where} \quad NET_{pk} \; = \; C \, E \, (\sum_{j=0}^{m} V_{jk} \; O'_{pj})$$

32

Fuzzy Associative Memory

A fuzzy logic system contains the sets used to categorize input data (i.e., fuzzification), the decision rules that are applied to each set, and then a way of generating an output from the rule results (i.e., defuzzification).

In the fuzzification stage, a data point is assigned a degree of membership (DOM) determined by a membership function. The member-ship function is often a triangular function centered at a given point. The Defuzzification is the name for a procedure to produce a real (non-fuzzy) output .

Associative Memory is a type of memory with a generalized addressing method. The address is not the same as the data location, as in traditional memory. An associative memory system stores mappings of specific input representations to specific output representations. Associative memory allows a fuzzy rule base to be stored. The inputs are the degrees of membership, and the outputs are the fuzzy system's output.

Fuzzy Associative Memory (FAM) consists of a single-layer feed-forward fuzzy neural network that stores fuzzy rules **"If x is X_k then y is Y_k"** by means of a fuzzy associative matrix.

FAM has many applications; one such application is modeling the operations of **washing machine**.

33

● **Example : Washing Machine (FAM Model)**

For a washing machine , the input/output variables are :

Output variable : **washing time (T)** , depends upon two input variables.

Input variables are : **weight of clothes (X)** and **stream of water (Y).**

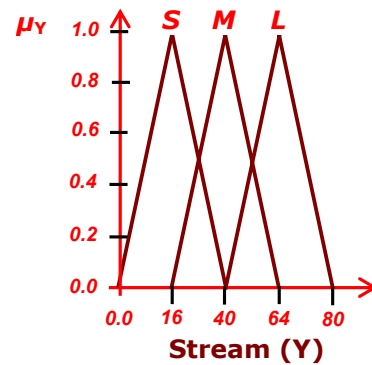These variables have three different degree of variations as :

small (S), **medium (M)**, and **large (L)** .

These three variables **X , Y,** and **T**, are defined below showing their membership functions μ_X , μ_Y and μ_T .
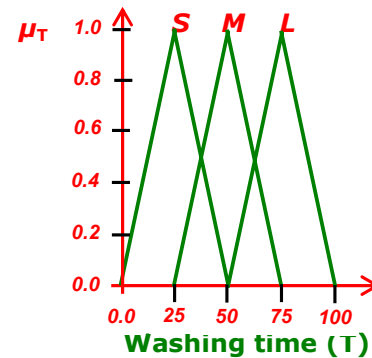
- **Clothes weight** is **X**,
    - range is from **0** to **10** and
    - the unit is kilogram (k.g).
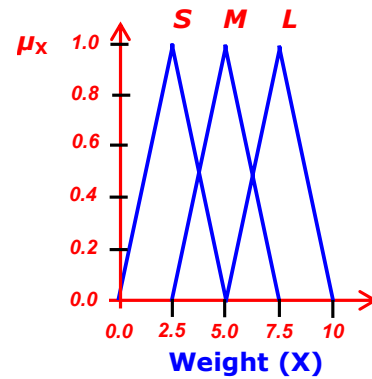
- **Stream of water** is **Y**
    - range is from **0** to **80** and
    - the unit is liter per minute (liters/min)

- **Washing time** is **T**
    - range is from **0** to **100** and
    - the unit is minutes (min.)

34

[continued from previous slide – Model of Washing Machine]

The problem indicates, that there are two inputs and one-output variables. The inference engineer is constructed based on fuzzy rule :

" If < input variable > AND < input variable >

THEN < output variable >"

According to the above fuzzy rule, the Fuzzy Associative Memory (FSM) of **X, Y,** and **T** variables are listed in the Table below.

Washing time (T)		Weight (X)		
		S	**M**	**L**
	S	**M**	**L**	**L**
Stream (Y)	**M**	**S**	**M**	**L**
	L	**S**	**S**	**L**

Table 1. Fuzzy associative memory (FSM) of Washing Machine

- **Operations :** To wash the clothes
  - Turn on the power,
  - The machine automatically detects the weight of   the   clothes as **(X) = 3.2** K.g. ,
  - The machine adjusts the water stream **(Y)** to **32** liter/min.,

35

■ **Fuzzy Representation :**

The fuzzy sets representation, while **X = 3.2 Kg** and **Y = 32 liter/min.**, according to the membership functions, are as follows:

The fuzzy set of $X_{3.2\ Kg}$ = { 0.8/S, 0.2/M, 0/L }

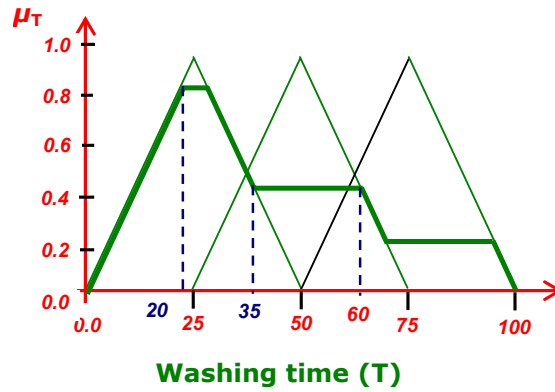The fuzzy set of $Y_{32\ liters/min.}$ = { 0.4/S, 0.8/M, 0/L }

Fig. Simulated Fuzzy set representation of washing machine

36

■ **Defuzzification**

The real washing time is defuzzied by the **Center of gravity (COG)** defuzzification formula. The washing time is calculated as :

$$Z_{COG} = \sum_{j=1}^{n} \mu_c(Z_j) \, Z_j \; / \; \sum_{j=1}^{n} \mu_c(Z_j) \quad \text{where}$$

j = 1, . . . , n , is the number of quantization levels of the output,

Z_j is the control output at the quantization level *j*,

$\mu_c(Z_j)$ represents its membership value in the output fuzzy set.

Referring to Fig in the previous slide and the formula for COG, we get the fuzzy set of the washing time as *w = { 0.8/20,   0.4/35,   0.2/60 }* The calculated washing time using COG formula *T = 41.025 min.*

37

5. Simplified Fuzzy ARTMAP

ART is a neural network topology whose dynamics are based on Adaptive Resonance Theory (ART). ART networks follow both supervised and unsupervised algorithms.

- The Unsupervised ARTs are similar to many iterative clustering algorithms where "nearest" and "closer" are modified slightly by introducing the concept of "**resonance**". *Resonance* is just a matter of being within a certain threshold of a second similarity measure.

- The Supervised ART algorithms that are named with the suffix "MAP", as ***ARTMAP***. Here the algorithms cluster both the inputs and targets and associate two sets of clusters.

The basic ART system is an unsupervised learning model.

The ART systems have many variations : ART1, ART2, Fuzzy ART, ARTMAP.

ART1: The simplest variety of ART networks, accepting only binary inputs.

ART2 : It extends network capabilities to support continuous inputs.

ARTMAP : Also known as Predictive ART. It combines two slightly modified ART-1 or ART-2 units into a supervised learning structure. Here, the first unit takes the input data and the second unit takes the correct output data, then used to make the minimum possible adjustment of the vigilance parameter in the first unit in order to make the correct classification.

The Fuzzy ARTMAP model is fuzzy logic based computations incorporated in the ARTMAP model.

Fuzzy ARTMAP is neural network architecture for conducting supervised learning in a multidimensional setting. When Fuzzy ARTMAP is used on a learning problem, it is trained till it correctly classifies all training data. This feature causes Fuzzy ARTMAP to 'over-fit' some data sets, especially those in which the underlying pattern has to overlap. To avoid the problem of 'over-fitting' we must allow for error in the training process.

● **Supervised ARTMAP System**

ARTMAP is also known as predictive ART. The Fig. below shows a supervised ARTMAP system. Here, two ART modules are linked by an inter-ART module called the Map Field. The Map Field forms predictive associations between categories of the ART modules and realizes a match tracking rule. If ARTa and ARTb are disconnected then each module would be of self-organize category, groupings their respective input sets.
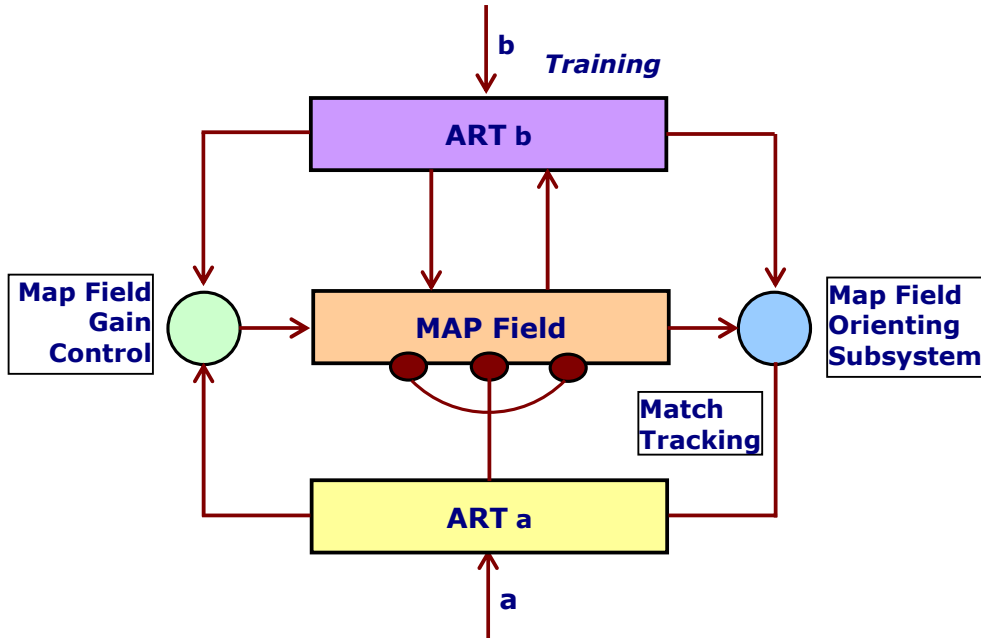

Fig.   Supervised ARTMAP system

In supervised mode, the mappings are learned between input vectors *a* and *b*. A familiar example of supervised neural networks are feed-forward networks with back-propagation of errors.

39

● **Comparing ARTMAP with Back-Propagation Networks**

ARTMAP networks are self-stabilizing, while in BP networks the new information gradually washes away old information. A consequence of this is that a BP network has separate training and performance phases while ARTMAP systems perform and learn at the same time

– ARTMAP networks are designed to work in real-time, while BP networks are typically designed to work off-line, at least during their training phase.

– ARTMAP systems can learn both in a fast as well as in a slow match configuration, while, the BP networks can only learn in slow mismatch configuration. This means that an ARTMAP system learns, or adapts its weights, only when the input matches an established category, while BP networks learn when the input does not match an established category.

– In BP networks there is always a danger of the system getting trapped in a local minimum while this is impossible for ART systems. However, the systems based on ART modules learning may depend upon the ordering of the input patterns.

40

6. References : Textbooks

1. *"Neural Network, Fuzzy Logic, and Genetic Algorithms - Synthesis and Applications", by S. Rajasekaran and G.A. Vijayalaksmi Pai, (2005), Prentice Hall, Chapter 10-15, page 297-435.*

2. *"Soft Computing and Intelligent Systems - Theory and Application", by Naresh K. Sinha and Madan M. Gupta (2000), Academic Press, Chapter 1-25, page 1-625.*

3. *"Soft Computing and Intelligent Systems Design - Theory, Tools and Applications", by Fakhreddine karray and Clarence de Silva (2004), Addison Wesley, chapter 7, page 337-361.*

4. *"Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence" by J. S. R. Jang, C. T. Sun, and E. Mizutani, (1996), Prentice Hall, Chapter 17-21, page 453-567.*

5. *"Fuzzy Logic: Intelligence, Control, and Information", by John Yen, Reza Langari, (1999), Prentice Hall, Chapter 15-17, page 425-500.*

6. *"Fuzzy Logic and Neuro Fuzzy Applications Explained", by Constantin Von Altrock, (1995), Prentice Hall, Chapter 4, page 63-79.*

7. *Related documents from open source, mainly internet. An exhaustive list is being prepared for inclusion at a later date.*

41