Soft computing

# THE DEVELOPMENT OF MEMBERSHIP FUNCTIONS



Soft computing

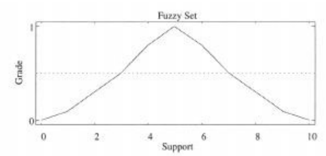
## DEVELOPMENT OF MEMBERSHIP FUNCTIONS

- There are possible more ways to assign membership values or function to fuzzy variables than there are to assign probability density functions to random variables [Dubois and Prade, 1980]
- Just as there are an several ways to characterize fuzziness, there are an several number of ways to graphically depict the membership functions that describe this fuzziness.

Soft computing

## DEVELOPMENT OF MEMBERSHIP FUNCTIONS

- **Since the membership function *essentially* embodies all fuzziness for a particular fuzzy set, its *description* is the essence of a fuzzy property or operation.**




Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

Soft computing

## MEMBERSHIP VALUE ASSIGNMENT

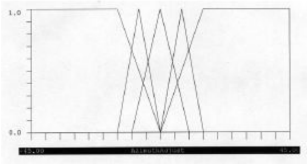
- **Intuition**
- **Inference**
- **Rank ordering**
- **Angular fuzzy sets**
- **Neural networks**
- **Genetic algorithms**
- **Inductive reasoning**

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

Soft computing

# INTUTITION

- Derived from the capacity of humans to develop membership functions through their own innate intelligence and understanding.
- Involves contextual and semantic knowledge about an issue; it can also involve linguistic truth values about this knowledge.

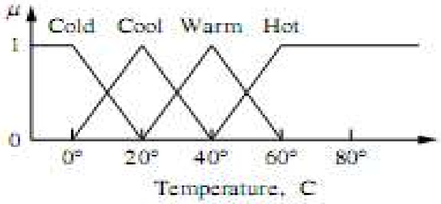


Fuzzy Logic with Engineering Applications: Timothy J. Ross

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

Soft computing

# INTUITION




- For example, each curve is a membership function corresponding to various fuzzy variables, such as very cold, cold, normal, hot, and very hot.

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

Soft computing

## INTUITION

- In numerous examples we shall see that the precise shapes of these curves are not so important in their utility.
- Rather, it is the approximate placement of the curves on the universe of discourse, the number of curves (partitions) used, and the overlapping character that are the most important ideas.



Soft computing

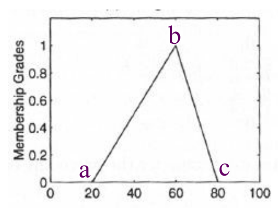
## TYPES OF MEMBERSHIP FUNCTIONS

- The most commonly used in practice are
  - Triangles
  - Trapezoids
  - Bell curves
  - Gaussian, and
  - Sigmoidal

Fuzzy Logic:Intelligence, Control, and Information, J. Yen and R. Langari, PrenticeHall

Soft computing

# TRIANGULAR MF




**Specified by three parameters {a,b,c} as follows:**

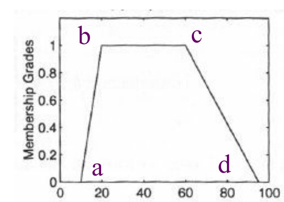
$$triangle(x:a,b,c)=\begin{cases}0 & x<a\\(x-a)/(b-a) & a\le x\le b\\(c-x)/(c-b) & b\le x\le c\\0 & x>c\end{cases}$$

Fuzzy Logic:Intelligence, Control, and Information, J. Yen and R. Langari, PrenticeHall

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

Soft computing

# TRAPEZOIDAL MF



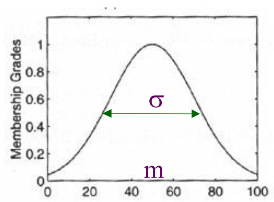

**Specified by four parameters {a,b,c,d} as follows:**

$$trapezoidal(x:a,b,c,d)=\begin{cases}0 & x<a\\(x-a)/(b-a) & a\le x<b\\1 & b\le x<c\\(d-x)/(d-c) & c\le x\le d\\0 & x\ge d\end{cases}$$

Fuzzy Logic:Intelligence, Control, and Information, J. Yen and R. Langari, PrenticeHall

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

# GAUSSIAN MF




**Specified by two parameters {m,σ} as follows:**

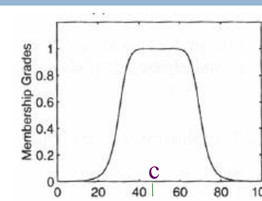
$$gaussian(x : m,\sigma) = \exp\left\{-\frac{(x-m)^2}{\sigma^2}\right\}$$

**Where m and σ denote the center and width of the function, respectively**
**A small σ will generate a "thin"MF, while a big σ will lead to a "flat"MF.**

Fuzzy Logic:Intelligence, Control, and Information, J. Yen and R. Langari, PrenticeHall

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

# BELL-SHAPED MF




**Specified by three parameters {a,b,c} as follows:**

$$bell(x : a,b,c) = \frac{1}{1+\left|\frac{x-c}{a}\right|^{2b}}$$

**Where the parameter b is usually positive and we can adjust c and a to vary the center and width of the function and then use b to control the slopes.**
***this membership function is is direct generalization of Cauchy distribution**

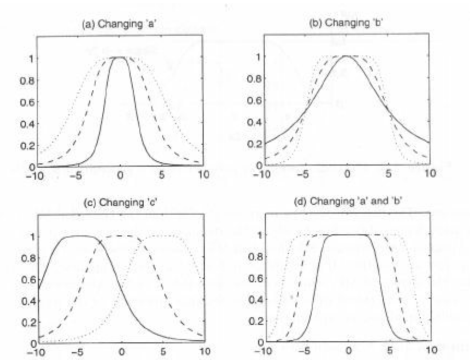
Fuzzy Logic:Intelligence, Control, and Information, J. Yen and R. Langari, PrenticeHall

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

Soft computing

## BELL-SHAPED MF

$$bell(x : a, b, c) = \frac{1}{1 + \left|\frac{x-c}{a}\right|^{2b}}$$

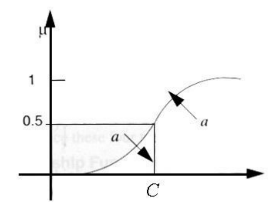



Fuzzy Logic:Intelligence, Control, and Information, J. Yen and R. Langari, PrenticeHall

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

Soft computing

## SIGMOIDAL MF




**Specified by two parameters {a, c} as follows:**

$$Sigmoidal(x : a, c) = \frac{1}{1 + e^{-a(x-c)}}$$

**Where c is the center of the function and a control the slope.**

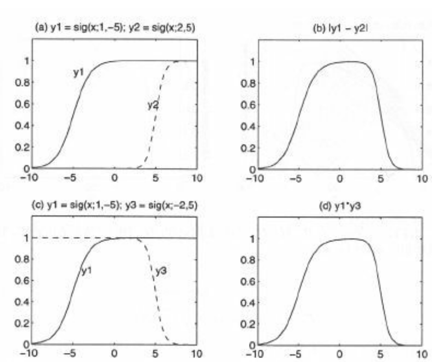
Fuzzy Logic:Intelligence, Control, and Information, J. Yen and R. Langari, PrenticeHall

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

Soft computing

# SIGMOIDAL MF

$$Sigmoidal(x : a,c) = \frac{1}{1+e^{-a(x-c)}}$$




Fuzzy Logic:Intelligence, Control, and Information, J. Yen and R. Langari, PrenticeHall

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

Soft computing

# HEDGES: A MODIFIER TO A FUZZY SET

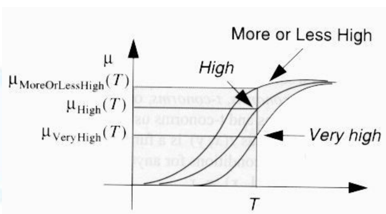
- **Hedge modifies the meaning of the original set to create a compound fuzzy set**
  - **Example:**
    - **Very (Concentration)**
    - **More or Less(Dilation)**

Fuzzy Logic:Intelligence, Control, and Information, J. Yen and R. Langari, PrenticeHall

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

Soft computing

## HEDGES: VERY & MORE OR LESS




*Very* :

$$\mu_{veryA}(x) = [\mu_A(x)]^2$$

*MoreorLess* :

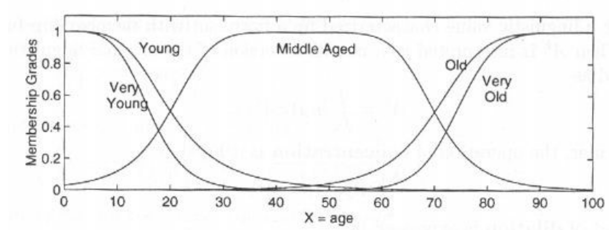
$$\mu_{MoreOrLessA}(x) = \sqrt{\mu_A(x)}$$

Fuzzy Logic:Intelligence, Control, and Information, J. Yen and R. Langari, PrenticeHall

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

Soft computing

## HEDGES: VERY

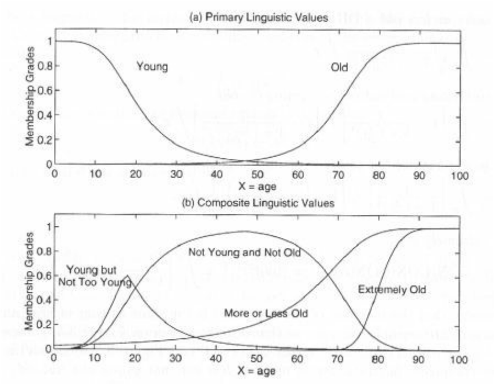



Fuzzy Logic:Intelligence, Control, and Information, J. Yen and R. Langari, PrenticeHall

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

Soft computing

## HEDGES: VERYVERYVERY (EXTREME)




Fuzzy Logic:Intelligence, Control, and Information, J. Yen and R. Langari, PrenticeHall

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

Soft computing

## INFERENCE

- **Use knowledge to perform deductive reasoning, i.e . we wish to deduce or infer a conclusion, given a body of facts and knowledge.**

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

Soft computing

## INFERENCE : EXAMPLE

- **In the identification of a triangle**
  - **Let A, B, C be the inner angles of a triangle**
    - **Where $A \geq B \geq C$**
  - **Let U be the universe of triangles, i.e.,**
    - **$U = \{(A,B,C) \mid A \geq B \geq C \geq 0;\ A+B+C = 180^\circ\}$**
  - **Let 's define a number of geometric shapes**
    - **I — Approximate isosceles triangle**
    - **R — Approximate right triangle**
    - **IR — Approximate isosceles and right triangle**
    - **E — Approximate equilateral triangle**
    - **T — Other triangles**

| | |
|---|---|
| $\underset{\sim}{I}$ | Approximate isosceles triangle |
| $\underset{\sim}{R}$ | Approximate right triangle |
| $\underset{\sim}{IR}$ | Approximate isosceles *and* right triangle |
| $\underset{\sim}{E}$ | Approximate equilateral triangle |
| $\underset{\sim}{T}$ | Other triangles |

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

Soft computing

## INFERENCE : EXAMPLE

- **We can infer membership values for all of these triangle types through the method of inference, because we possess knowledge about geometry that helps us to make the membership assignments.**

- **For Isosceles,**
  **$\mu_i (A,B,C) = 1 - 1/60^* \min(A-B, B-C)$**
  - **If A=B OR B=C THEN $\mu_i (A,B,C) = 1$;**
  - **If $A=120^\circ$, $B=60^\circ$, and $C=0^\circ$ THEN $\mu_i (A,B,C) = 0$.**

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

## INFERENCE : EXAMPLE

- **For right triangle,**
  **$\mu_R$ (A,B,C) = 1- 1/90* |A-90˚|**
  - **If A=90˚ THEN $\mu_i$ (A,B,C) = 1;**
  - **If A=180˚ THEN $\mu_i$ (A,B,C) = 0.**
- **For isosceles and right triangle**
  - **IR = min (I, R)**
  **$\mu_{IR}$ (A,B,C) = min[$\mu_I$ (A,B,C), $\mu_R$ (A,B,C)]**
  **= 1 - max[1/60min(A-B, B-C), 1/90|A-90|]**

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

## INFERENCE : EXAMPLE

- For equilateral triangle
  - $\mu_E$ (A,B,C) = 1 - 1/180* (A-C)
  - – When A = B = C then $\mu_E$ (A,B,C) = 1, A = 180 then $\mu_E$ (A,B,C) = 0
- For all other triangles
  - – T = (I.R.E)' = I'.R'.E'
    - = min {1 - $\mu_I$ (A,B,C) , 1 - $\mu_R$ (A,B,C) , 1 - $\mu_E$ (A,B,C)

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

Soft computing

## INFERENCE : EXAMPLE

– **Define a specific triangle:**
  - $A = 85^\circ \geq B = 50^\circ \geq C = 45^\circ$

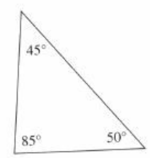
$\mu_R = 0.94$

$\mu_I = 0.916$

$\mu_{IR} = 0.916$

$\mu_E = 0.7$

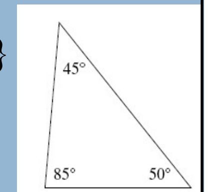
$\mu_T = 0.05$




Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

**Example 6.1 Define a specific triangle, as shown in Fig., with these three order angles:**

$$\{X : A = 85 \geq B = 50 \geq C = 45, where A + B + C = 180\}$$




Answer

$$\mu_{\tilde{R}}(A,B,C) = \mu_{\tilde{R}}(x) = 1 - \frac{1}{90^\circ}\min|A - 90^\circ| = 1 - \frac{1}{90^\circ}|85^\circ - 90^\circ|$$

$$= 1 - \frac{1}{90^\circ}|-5^\circ| = 1 - \frac{1}{90^\circ} \times 5^\circ = 1 - 0.056 = 0.94$$

Highest Membership in R

$$\mu_{\tilde{I}}(A,B,C) = \mu_{\tilde{I}}(x) = 1 - \frac{1}{60^\circ}\min(A - B, B - C) = 1 - \frac{1}{60^\circ}\min(85^\circ - 50^\circ, 50^\circ - 45^\circ)$$

$$= 1 - \frac{1}{60^\circ}\min(35^\circ, 5^\circ) = 1 - \frac{1}{60^\circ}(5^\circ) = 1 - 0.083 = 0.9167$$

High Membership in I and IR

$$\mu_{\tilde{IR}}(A,B,C) = \mu_{\tilde{IR}}(x) = \min\{\mu_{\tilde{I}}(A,B,C), \mu_{\tilde{R}}(A,B,C)\} = \min[0.916, 0.94] = 0.916$$

$$\mu_{\tilde{E}}(A,B,C) = \mu_{\tilde{E}}(x) = 1 - \frac{1}{180^\circ}(A - C) = 1 - \frac{1}{180^\circ}(85^\circ - 45^\circ) = 0.78$$

$$\mu_{\tilde{T}}(A,B,C) = \mu_{\tilde{T}}(x) = \min\{1 - \mu_{\tilde{I}}(A,B,C), 1 - \mu_{\tilde{E}}(A,B,C), 1 - \mu_{\tilde{R}}(A,B,C)\} = \min\{1 - 0.916, 1 - 0.78, 1 - 0.94\} =$$

$$= \min\{0.084, 0.022, 0.05\} = 0.05$$

## RANK ORDERING

- Assessing preferences by a single individual, a committee, a poll, and other opinion methods can be used to assign membership values to a fuzzy variable.
- Preference is determined by pairwise comparisons, and these determine the ordering of the membership.



Soft computing

## RANK ORDERING

- Example6.2 .Suppose 1000 people respond to a questionnarie about their pairwise preferences among 5 colors , X={red, orange, yellow, green, blue}. Define a fuzzy set as A on the universe of colors "best color."
- Table 6.1 is a summery. In this table, for example, out of 1000 people 517 preferred the color red to the color orange, 841 preferred the color orange to yellow, etc.
- The total number of responses is 10,000 (10 comparisons). If the sum of the preferences of each color (row sum) is normalized to the total number of responses, a rank ordering can be determined in the last two columns.

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

Soft computing

## EXAMPLE IN RANK ORDERING

TABLE 6.1 Number who preferred

| | Red | Orange | Yellow | Green | Blue | Total | Percentage | Rank order |
|---|---|---|---|---|---|---|---|---|
| Red | - | 517 | 525 | 545 | 661 | 2248 | 22.5 | 2 |
| Orange | 483 | - | 841 | 477 | 576 | 2377 | 23.8 | 1 |
| Yellow | 475 | 259 | - | 534 | 614 | 1782 | 17.8 | 4 |
| Green | 455 | 523 | 466 | - | 643 | 2087 | 20.9 | 3 |
| Blue | 339 | 424 | 386 | 357 | - | 1506 | 15 | 5 |
| Total | | | | | | 10,000 | | |

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

Soft computing

## NEURAL NETWORK

It is a technique that seeks to build an intelligent program using models that simulate the working network of the neurons in the human brain.

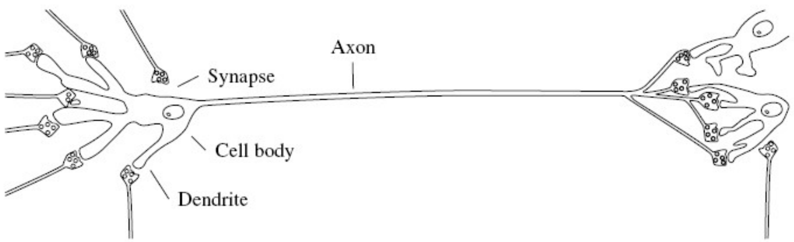



Figure. A simple schematic of a human neuron

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

Soft computing

## HUMAN NEURON

A neuron is made up of several protrusions called dendrites and a long branch called the axon.

A neuron is joined to other neurons through the dendrites. The dendrites of different neurons meet to form (synapses ), the areas where messages pass.

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

Soft computing

## HUMAN NEURON

The neuron receive the impulses via the synapses. If the total of the impulses received exceeds a certain threshold value, then the neuron sends an impulse down the axon where the axon is connected to other neurons through more synapses. The synapses may be excitatory or inhibitory in nature.

An excitatory synapse adds to the total of the impulses reaching the neuron, whereas an inhibitory neuron reduces the total of the impulses reaching the neuron.

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

Soft computing

## EXAMPLE

- If the percentage preference of these colors is plotted to a normalized scale on the universe of colors in an ascending order on the color universe, the membership function for the "best color" shown in Fig.6.3 would result. Alternately, the membership function could be formed based on the rank order Table 6.1)

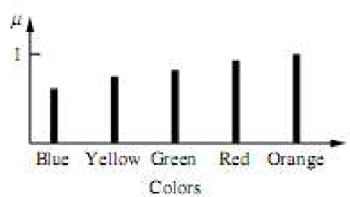



Figure 6.3. Membership function for best color.

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

Soft computing

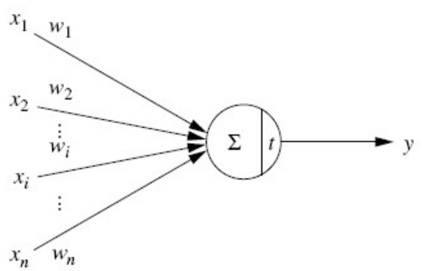
## NEURON

- In a global sense, a neuron receives a set of input pulses and sends out another pulse that is a function of the input pulses.

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

Soft computing

## A THRESHOLD ELEMENT AS AN ANALOG TO A NEURON




$$y = F(\sum w_i x_i - t)$$

$$F(s) = \frac{1}{1+e^{-s}}$$

$x_i$ signal input *($i$ = 1, 2, . . . , $n$)*

$w_i$ weight associated with the signal input $x_i$

*t* threshold level prescribed by user

*F (s)* is a nonlinear function, e.g., a sigmoid function

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

Soft computing

## A SIMPLE NEURAL NETWORK FOR A SYSTEM

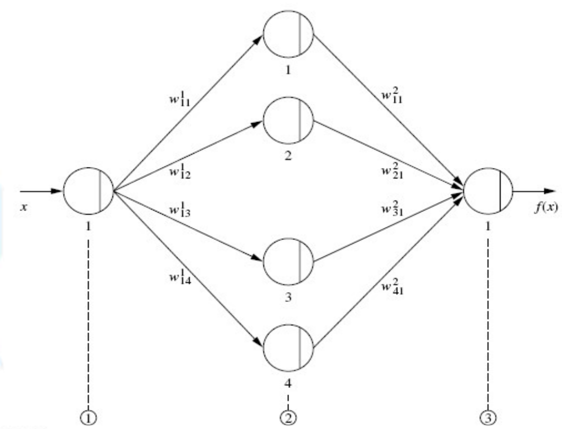



**FIGURE 6.6**
A simple $1 \times 4 \times 1$ neural network, where $w^i_{jk}$ represents the weight associated with the path connecting the $j$th element of the $i$th layer to the $k$th element of the $(i+1)$th layer.

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

Soft computing

## NEURAL NETWORK

- Neural systems solve problems by adapting to the nature of the data (signals) they receive.
- One of the ways to accomplish this is to use a training data set and a checking data set of input and output data/signals *(x, y) (for a multiple-input, multiple-output* system using a neural network, we may use input–output sets comprised of vectors

$$(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n).$$

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

Soft computing

## A SIMPLE NEURAL NETWORK FOR A SYSTEM

- We start with a random assignment of weights $w^i_{jk}$ *to the* paths joining the elements in the different layers (Fig. 6.6).
- Then an input *x from the* training data set is passed through the neural network.
- The neural network computes a value ($f(x)_{output}$), *which is compared with the actual value* ($f(x)_{actual} = y$). *The error measure E is computed from these two output values as*

$$E = f(x)_{actual} - f(x)_{output}$$

*(This is the error measure associated with the last layer of the neural network)*

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

Soft computing

## A SIMPLE NEURAL NETWORK FOR A SYSTEM

- To distribute this error to the elements in the hidden layers by using a technique called back- propagation.

✓ Let $E_j$ be the error associated with the $j$th element.

✓ Let $w_{nj}$ be the weight associated with the line from element $n$ to element $j$ .

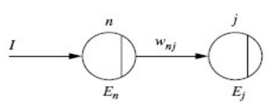
✓ Let $I$ be the input to unit $n$.

- The error for element $n$ is computed as

$$E_n = F'(I) w_{nj} E_j$$

where, for $F(I) = 1/(1 + e{-}I)$, the sigmoid function, we have

$$F'(I) = F(I)(1 - F(I))$$




Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

Figure. Distribution of error to different elements

Soft computing

## NEXT

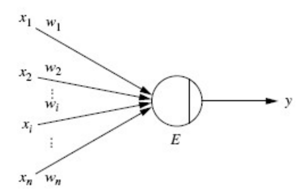
- the different weights $w^i_{jk}$ connecting different elements in the network are corrected so that they can approximate the final output more closely.
- For updating the weights, the error measure on the elements is used to update the weights on the lines joining the elements.
- For an element with an error $E$ associated with it, the associated weights may be updated as

$$w_i\,(\text{new}) = w_i\,(\text{old}) + \alpha E x_i$$

$\alpha$ = learning constant

$E$ = associated error measure

$x_i$ = input to the element




Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

Figure. A threshold element with an error E associated with it

Soft computing

## CONTINUE

- The input value $x_i$ is passed through the neural network again, and the errors, if any, are computed again. This technique is iterated until the error value of the final output is within some user-prescribed limits.
- The neural network then uses the next set of input–output data. This method is continued for all data in the training data set. This technique simulates the nonlinear relation between the input–output data sets.
- Finally a checking data set is used to verify how well the neural network can simulate the nonlinear relationship.

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

Soft computing

## DRAWBACKS OF NEURAL NETWORKS

- The lack of intuitive knowledge in the learning process is one of the major drawbacks of neural networks for use in cognitive learning.

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

## GENERATION OF MEMBERSHIP FUNCTIONS USING A NEURAL NETWORK

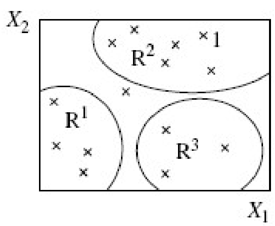
•*Select* a number of input data values and divide them into a training data set and a checking data set.

•The training data set is used to train the neural network. Its coordinate values of the different data points considered.



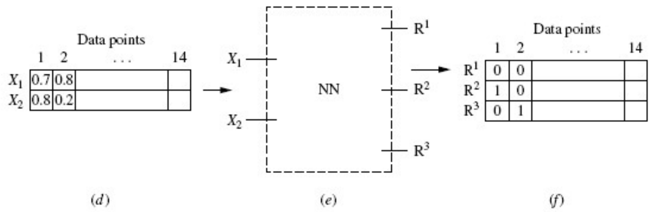
## GENERATION OF MEMBERSHIP FUNCTIONS USING A NEURAL NETWORK

- The data points are first divided (Fig. *a*) by conventional clustering techniques as classes, $R^1$, $R^2$, and $R^3$



Soft computing

## GENERATION OF MEMBERSHIP FUNCTIONS USING A NEURAL NETWORK




Figures. Using a neural network to determine membership functions

•Consider data point 1, $x_1$ = 0.7 and $x_2$ = 0.8 (Fig *d*). As this is in region $R^2$, we assign to it a complete membership of one in class $R^2$ and zero membership in classes $R^1$ and $R^3$ (Fig. *f*).

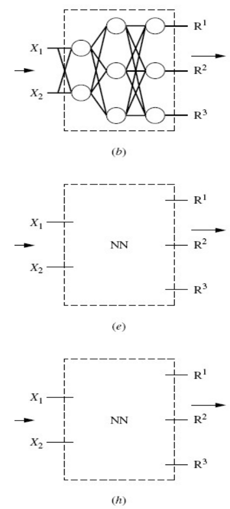
•Similarly, the other data points are assigned membership values of unity for the classes they belong to initially.

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

Soft computing

## GENERATION OF MEMBERSHIP FUNCTIONS USING A NEURAL NETWORK

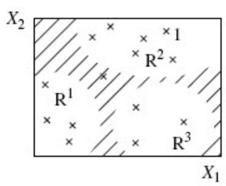
- A neural network is created (Figs. *b*, *e*, *h*) that uses the data point 1 and the corresponding membership values in different classes for training itself to simulate the relationship between coordinate locations and the membership values.




Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

Soft computing

## GENERATION OF MEMBERSHIP FUNCTIONS USING A NEURAL NETWORK




(*c*)

Figure *c* represents the output of the neural network, which classifies data points into one of the three regions.

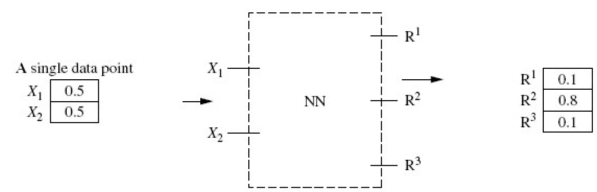
The neural network then uses the next set of data values (e.g., point 2) and membership values to train itself further as seen in Fig. *d*.

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

Soft computing

## GENERATION OF MEMBERSHIP FUNCTIONS USING A NEURAL NETWORK

- This repetitive process is continued until the neural network can simulate the entire set of input–output (coordinate location–membership value) values.
- The performance of the neural network is then checked using the checking data set.
- Once the neural network is ready, its final version (Fig. *h*)can be used to determine the membership values (function) of any input data (Fig. *g*) in the different regions (Fig. *i*).




Source: Fuzzy Logic with Engineering Applic...

(*g*) (*h*) (*i*)

Soft computing

## GENERATION OF MEMBERSHIP FUNCTIONS USING A NEURAL NETWORK

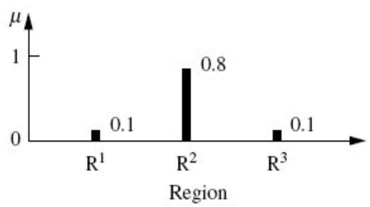



**FIGURE 6.10**
Membership function for data point $(X_1, X_2 = (0.5, 0.5))$.

Notice that the points shown in the table in Fig. 6.9*i* are actually the membership values in each region for the data point shown in Fig. 6.9*g*. These could be plotted as a membership function, as shown in Fig. 6.10.

A complete mapping of the membership of different data points in the different fuzzy classes can be derived to determine the overlap of the different classes (the hatched portion in Fig. 6.9*c shows the overlap of the three* fuzzy classes).



Soft computing

## EXAMPLE 6.3

**TABLE 6.2**
Variables describing the data points to be used as a training data set

| Data point | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| $x_1$ | 0.05 | 0.09 | 0.12 | 0.15 | 0.20 | 0.75 | 0.80 | 0.82 | 0.90 | 0.95 |
| $x_2$ | 0.02 | 0.11 | 0.20 | 0.22 | 0.25 | 0.75 | 0.83 | 0.80 | 0.89 | 0.89 |

**TABLE 6.3**
Variables describing the data points to be used as a checking data set

| Data point | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|
| $x_1$ | 0.09 | 0.10 | 0.14 | 0.18 | 0.22 | 0.77 | 0.79 | 0.84 | 0.94 | 0.98 |
| $x_2$ | 0.04 | 0.10 | 0.21 | 0.24 | 0.28 | 0.78 | 0.81 | 0.82 | 0.93 | 0.99 |

We have placed these data points in two fuzzy classes, R1 and R2, using a clustering technique. We would like to form a neural network that can determine the membership values of any data point in the two classes.

Soft computing

## EXAMPLE 6.3

The membership values in Table 6.4 are to be used to train and check the performance of the neural network.

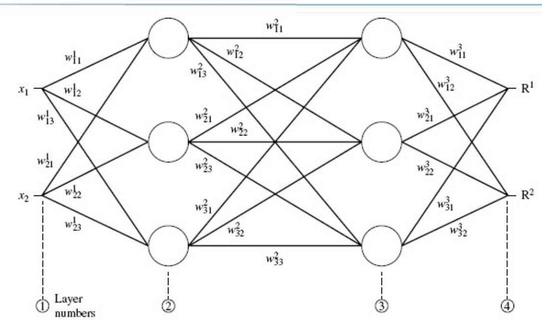
**TABLE 6.4**
Membership values of the data points in the training and checking data sets to be used for training and checking the performance of the neural network

| Data points | 1 & 11 | 2 & 12 | 3 & 13 | 4 & 14 | 5 & 15 | 6 & 16 |
|---|---|---|---|---|---|---|
| $R_1$ | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 0.0 |
| $R_2$ | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |
| | **7 & 17** | **8 & 18** | **9 & 19** | **10 & 20** | | |
| | 0.0 | 0.0 | 0.0 | 0.0 | | |
| | 1.0 | 1.0 | 1.0 | 1.0 | | |



Soft computing

## EXAMPLE 6.3




- We select a 2 × 3 × 3 × 2 neural network to simulate the relationship between the data points and their membership in the two fuzzy sets, R1 and R2 .
- The coordinates $x_1$ and $x_2$ for each data point are used as the input values, and the corresponding membership values in the two fuzzy classes for each data point are the output values .

Soft computing

## EXAMPLE 6.3

- Table 6.5 shows the initial quasi-random values that have been assigned to the different weights connecting the paths between the elements in the layers in the network shown in Fig. 6.11.

**TABLE 6.5**
The initial quasi-random values that have been assigned to the different weights connecting the paths between the elements in the layers in the network of Fig. 6.11

| | | |
|---|---|---|
| $w_{11}^1 = 0.5$ | $w_{11}^2 = 0.10$ | $w_{11}^3 = 0.30$ |
| $w_{12}^1 = 0.4$ | $w_{12}^2 = 0.55$ | $w_{12}^3 = 0.35$ |
| $w_{13}^1 = 0.1$ | $w_{13}^2 = 0.35$ | $w_{21}^3 = 0.35$ |
| $w_{21}^1 = 0.2$ | $w_{21}^2 = 0.20$ | $w_{22}^3 = 0.25$ |
| $w_{22}^1 = 0.6$ | $w_{22}^2 = 0.45$ | $w_{31}^3 = 0.45$ |
| $w_{23}^1 = 0.2$ | $w_{23}^2 = 0.35$ | $w_{32}^3 = 0.30$ |
| | $w_{31}^2 = 0.25$ | |
| | $w_{32}^2 = 0.15$ | |
| | $w_{33}^2 = 0.60$ | |

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

Soft computing

## EQUATIONS USED FOR OUTPUT AND ERROR

$$O = \frac{1}{1 + \exp[-(\sum x_i w_i - t)]}$$

$O$ = output of the threshold element computed using the sigmoidal function
$x_i$ = inputs to the threshold element $(i = 1, 2, \ldots, n)$
$w_i$ = weights attached to the inputs
$t$ = threshold for the element

*Determining errors:*

$$R_1 : E_1^4 = O_{1\ \text{actual}}^4 - O_1^4 = 1.0 - 0.666334 = 0.333666$$

$$R_2 : E_2^4 = O_{2\ \text{actual}}^4 - O_2^4 = 0.0 - 0.635793 = -0.635793$$

Distribute this error to the other nodes (elements) in the network

$$E_n = O_n(1 - O_n)\sum_j w_{nj} E_j$$

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

## EQUATIONS USED FOR UPDATED WEIGHTS

- Update the weights associated with these elements so that the network approximates the output more closely.
- To update the weights we use the following equation.

$$w^{i}_{jk}(new) = w^{i}_{jk}(old) + \alpha E^{i+1}_{k} x_{jk}$$

- Now that all the weights in the neural network have been updated, the input data point *(x1 = 0.05, x2 = 0.02) is again passed through the neural network. The errors in approximating* the output are computed again and redistributed as before.
- This process is continued until the errors are within acceptable limits.

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

## CONTINUED

- Next, the second data point (*x1 = 0.09, x2 = 0.11,*Table 6.2) and the corresponding membership values (R1 = 1,*R2 = 0, Table 6.4) are used to* train the network.
- This process is continued until all the data points in the *training data set*(Table 6.2) are used.
- The performance of the neural network (how closely it can predict the value of the membership of the data point) is then checked using the data points in the *checking* data set (Table 6.3).

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

Soft computing

## NEURAL NETWORK

- Once the neural network is trained and verified to be performing satisfactorily, it can be used to find the membership of any other data points in the two fuzzy classes.
- A complete mapping of the membership of different data points in the different fuzzy classes can be derived to determine the overlap of the different classes ($R_1$ and $R_2$).

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

Soft computing

## GENETIC ALGORITHMS

In a genetic algorithm, the parameter set of the problem is coded as a finite string of bits. For example, given a set of two-dimensional data (*(x, y)* data points), we want to fit a linear curve (straight line) through the data.

To get a linear fit, we encode the parameter set for a line *(y = C1x + C2)* by creating independent bit strings for the two unknown constants *C*1 and *C*2 (parameter set describing the line) and then join them (concatenate the strings). The bit strings are combinations of zeros and ones, which represent the value of a number in binary form. An *n*-bit string can accommodate all integers up to the value 2*n* − 1. For example, the number 7 requires a 3-bit string, that is, 23 − 1 = 7, and the bit string would look like "111," where the first unit digit is in the 22 place (=4), the second unit digit is in the 21 place (=2), and the last unit digit is in the 20 place (=1); hence, 4 + 2 + 1 = 7. The number 10 would look like "1010," that is, 23 + 21 = 10, from a 4-bit string. This bit string may be mapped to the value of a parameter, say *Ci* , *i* = 1, 2, by the mapping

$$C_i = C_{\min} + \frac{b}{2^L - 1}(C_{\max_i} - C_{\min_i}),$$

*where "b" is the number in decimal form that is being represented in binary form (e.g., 152 may be represented in binary form as 10011000), L is the length of the bit string (i.e., the number of bits in each string), and Cmax and Cmin are user-defined constants between which C1 and C2 vary linearly. The parameters C1 and C2 depend on the problem.*

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

Soft computing

**TABLE 6.6**
Data set through which a line fit is required.

| Data number | $x$ | $y'$ |
|---|---|---|
| 1 | 1.0 | 1.0 |
| 2 | 2.0 | 2.0 |
| 3 | 4.0 | 4.0 |
| 4 | 6.0 | 6.0 |

Let Let us us consider the data set in Table 6.6. For performing a line *(y = C1x + C2)* fit, as mentioned earlier, we first encode the parameter set *(C1, C2)* in the form of bit strings. Bit strings are created with random assignment of ones and zeros at different bit locations. We start with an initial population of four strings (Table 6.7a, column 2). The strings are 12 bits in length. The first 6 bits encode the parameter *C*1, and the next 6 bits encode the parameter *C*2. Table 6.7a, columns 3 and 5, shows the decimal equivalent of their binary coding.

- These binary values for *C*1 and *C*2 are then mapped into values relevant to the problem using Equation (6.15). We assume that the minimum value to which we would expect *C*1 or *C*2 to go would be −2 and the maximum would be 5 (these are arbitrary values – any other values could just as easily have been chosen). Therefore, for Equation (6.15), *C*min*i* = −2 and *C*max*i* = 5. Using these values, we compute *C*1 and *C*2 (Table 6.7*a*, columns 4 and 6). The values shown in Table 6.7*a*, columns 7, 8, 9, and 10, are the values computed using the equation *y* = *C*1*x* + *C*2, using the values of *C*1 and *C*2 from columns 4 and 6, respectively, for different values of *x* as given in Table 6.6. These computed values for the *y* are compared with the correct values (Table 6.6), and the square of the errors in estimating the *y* is calculated for each string. This summation is subtracted from a large number (400 in this problem) (Table 6.7*a*, column 11) to convert the problem
- into a maximization problem



Soft computing

## MEMBERSHIP VALUE ASSIGNMENT

**TABLE 6.7a**
First iteration using a genetic algorithm, Example 6.4.

| (1) String number | (2) String | (3) $C_1$ (binary) | (4) $C_1$ | (5) $C_2$ (binary) | (6) $C_2$ | (7) $y_1$ | (8) $y_2$ | (9) $y_3$ | (10) $y_4$ | (11) $f(x) = 400 - \Sigma(y_i - y_i')^2$ | (12) Expected count = $f/f_{av}$ | (13) Actual count |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 000111 010100 | 7 | −1.22 | 20 | 0.22 | −1.00 | −2.22 | −4.66 | −7.11 | 147.49 | 0.48 | 0 |
| 2 | 010010 001100 | 18 | 0.00 | 12 | −0.67 | −0.67 | −0.67 | −0.67 | −0.67 | 332.22 | 1.08 | 1 |
| 3 | 010101 101010 | 21 | 0.33 | 42 | 2.67 | 3.00 | 3.33 | 5.00 | 4.67 | 391.44 | 1.27 | 2 |
| 4 | 100100 001001 | 36 | 2.00 | 9 | −1.00 | 1.00 | 3.00 | 3.67 | 11.00 | 358.00 | 1.17 | 1 |
| | | | | | | | | | | Sum 1229.15 | | |
| | | | | | | | | | | Average 307.29 | | |
| | | | | | | | | | | Maximum 391.44 | | |

Soft computing

## MEMBERSHIP VALUE ASSIGNMENT

**TABLE 6.7b**
Second iteration using a genetic algorithm, Example 6.4.

| (1) Selected strings | (2) New strings | (3) $C_1$ | (4) $C_1$ (binary) | (5) $C_2$ | (6) $C_2$ (binary) | (7) $y_1$ | (8) $y_2$ | (9) $y_3$ | (10) $y_4$ | | (11) $f(x) = 400 - \Sigma(y_i - y_i')^2$ | (12) Expected count = $f/f_{av}$ | (13) Actual count |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0101\|01 101010 | 010110 001100 | 22 | 0.44 | 12 | −0.67 | −0.22 | 0.22 | 1.11 | 2.00 | | 375.78 | 1.15 | 1 |
| 0100\|10 001100 | 010001 101010 | 17 | −0.11 | 42 | 2.67 | 2.56 | 2.44 | 2.22 | 2.00 | | 380.78 | 1.17 | 2 |
| 010101 101\|010 | 010101 101001 | 21 | 0.33 | 41 | 2.56 | 2.89 | 3.22 | 3.89 | 4.56 | | 292.06 | 0.90 | 1 |
| 100100 001\|001 | 100100 001010 | 36 | 2.0 | 10 | −0.89 | 1.11 | 3.11 | 7.11 | 11.11 | | 255.73 | 0.78 | 0 |
| | | | | | | | | | | Sum | 1304.35 | | |
| | | | | | | | | | | Average | 326.09 | | |
| | | | | | | | | | | Maximum | 380.78 | | |



Soft computing

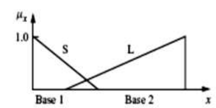



**FIGURE 6.13**
Membership functions for the input (and output) variables are assumed to be right triangles.

**TABLE 6.8**
Data for a single-input, single-output system.

| | | | | | |
|---|---|---|---|---|---|
| $x$ | 1 | 2 | 3 | 4 | 5 |
| $y$ | 1 | 4 | 9 | 16 | 25 |

**TABLE 6.9**
Functional mapping for the system.

| | | |
|---|---|---|
| $x$ | S | L |
| $y$ | S | VL |

- In Table 6.9 we see that each of the variables *x* and *y* makes use of two fuzzy classes (*x* uses S (small) and L (large); *y* uses L (large) and VL (very large)). The functional mapping tells us that a *small x* maps to a *small y*, and a *large x* maps to a *very large y*.
- We assume that the range of the variable *x* is [0, 5] and that of *y* is [0, 25]. We assume that each membership function has the shape of a right triangle, as shown in Figure 6.13.
- The membership function on the right side of Figure 6.13 is constrained to have the right-angle wedge at the upper limit of the range of the fuzzy variable.
- The membership function on the left side is constrained to have the right-angle wedge on the lower limit of the range of the fuzzy variable. It is intuitively obvious that under the foregoing constraints the only thing needed to describe the shape and position of the membership function fully is the length of the base of the right-triangle membership functions.
- We use this fact in encoding the membership functions as bit strings.

Soft computing

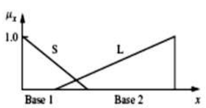



**FIGURE 6.13**
Membership functions for the input (and output) variables are assumed to be right triangles.

**TABLE 6.8**
Data for a single-input, single-output system.

| | | | | | |
|---|---|---|---|---|---|
| $x$ | 1 | 2 | 3 | 4 | 5 |
| $y$ | 1 | 4 | 9 | 16 | 25 |

**TABLE 6.9**
Functional mapping for the system.

| | | |
|---|---|---|
| $x$ | S | L |
| $y$ | S | VL |

- The unknown variables in this problem are the lengths of the bases of the four membership functions ($x$(S, L) and $y$(S, VL)). We use 6-bit binary strings to define the base of each of the membership functions.
  - The binary values are later mapped to decimal values using Equation (6.15).
    $$C_i = C_{\min} + \frac{b}{2^L - 1}(C_{\max_i} - C_{\min_i}).$$
  - These strings are then concatenated to give us a 24-bit (6 × 4) string.
- As shown in Table 6.10a, column 1, we start with an initial population of four strings. These are decoded to the binary values of the variables as shown in Table 6.10a, columns 2, 3, 4, and 5. The binary values are mapped to decimal values for the fuzzy variables using Equation (6.15) (Table 6.10a, columns 6, 7, 8, and 9). For the fuzzy variable $x$ *(range* $x$ = 0, 5*)*
- we use $C$min = 0 and $C$max = 5 for both the membership functions S (Small) and L (Large). For the fuzzy variable $y$ *(range* $y$ = 0, 25*)* we use $C$min = 0 and $C$max = 25.

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

Soft computing

**TABLE 6.10a**
First iteration using a genetic algorithm for determining optimal membership functions.

| String number | (1) String | (2) Base 1 (binary) | (3) Base 2 (binary) | (4) Base 3 (binary) | (5) Base 4 (binary) | (6) Base 1 | (7) Base 2 | (8) Base 3 | (9) Base 4 | (10) $y'$ ($x=1$) | (11) $y'$ ($x=2$) | (12) $y'$ ($x=3$) | (13) $y'$ ($x=4$) | (14) $y'$ ($x=5$) | (15) $1000-\Sigma(y_i-y_i')^2$ | (16) Expected count = $f/f_{av}$ | (17) Actual count |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 000111 010100 010110 110011 | 7 | 20 | 22 | 51 | 0.56 | 1.59 | 8.73 | 20.24 | 0 | 0 | 0 | 12.25 | 25 | 887.94 | 1.24 | 1 |
| 2 | 010010 001100 101100 100110 | 18 | 12 | 44 | 38 | 1.43 | 0.95 | 17.46 | 15.08 | 12.22 | 0 | 0 | 0 | 25 | 521.11 | 0.73 | 0 |
| 3 | 010101 101010 001101 101000 | 21 | 42 | 13 | 40 | 1.67 | 3.33 | 5.16 | 15.87 | 3.1 | 10.72 | 15.48 | 20.24 | 25 | 890.46 | 1.25 | 2 |
| 4 | 100100 001001 101100 100011 | 36 | 9 | 44 | 35 | 2.86 | 0.71 | 17.46 | 13.89 | 6.98 | 12.22 | 0 | 0 | 25 | 559.67 | 0.78 | 1 |
| | | | | | | | | | | | | | | Sum | 2859.18 | | |
| | | | | | | | | | | | | | | Average | 714.80 | | |
| | | | | | | | | | | | | | | Maximum | 890.46 | | |

$$C_i = C_{\min} + \frac{b}{2^L - 1}(C_{\max_i} - C_{\min_i}).$$

Here $C$min = 0 and $C$max = 5 for both the membership functions S (Small) and L (Large). For the fuzzy variable $y$ *(range* $y$ = 0, 25*)* $C$min = 0 and $C$max = 25.

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

Soft computing

**TABLE 6.10b**
Second iteration using a genetic algorithm for determining optimal membership functions.

| (1) Selected strings | (2) New Strings | (3) Base 1 (binary) | (4) Base 2 (binary) | (5) Base 3 (binary) | (6) Base 4 (binary) | (7) Base 1 | (8) Base 2 | (9) Base 3 | (10) Base 4 | (11) $y'$ $(x=1)$ | (12) $y'$ $(x=2)$ | (13) $y'$ $(x=3)$ | (14) $y'$ $(x=4)$ | (15) $y'$ $(x=5)$ | (16) $1000-\Sigma(y_i-y_i')^2$ | (17) Expected count = $f/f_{av}$ | (18) Actual count |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 000111 0101\|00 010110 110011 | 000111 010110 001101 101000 | 7 | 22 | 13 | 40 | 0.56 | 1.75 | 5.16 | 15.87 | 0 | 0 | 0 | 15.93 | 25 | 902.00 | 1.10 | 1 |
| 010101 1010\|10 001101 101000 | 010101 101000 010110 110011 | 21 | 40 | 22 | 51 | 1.67 | 3.17 | 8.73 | 20.24 | 5.24 | 5.85 | 12.23 | 18.62 | 25 | 961.30 | 1.18 | 2 |
| 010101 101010 001101 101000 | 010101 101010 001101 10\|0011 | 21 | 42 | 13 | 35 | 1.67 | 3.33 | 5.16 | 13.89 | 3.1 | 12.51 | 16.68 | 20.84 | 25 | 840.78 | 1.03 | 1 |
| 100100 001001 101100 100011 | 100100 001001 101100 10\|1000 | 36 | 9 | 44 | 40 | 2.86 | 0.71 | 17.46 | 15.87 | 6.11 | 12.22 | 0 | 0 | 25 | 569.32 | 0.70 | 0 |
| | | | | | | | | | | | | | | Sum | 3 273.40 | | |
| | | | | | | | | | | | | | | Average | 818.35 | | |
| | | | | | | | | | | | | | | Maximum | 961.30 | | |

$$C_i = C_{\min} + \frac{b}{2^L - 1}(C_{\max_i} - C_{\min_i}).$$

Here *C*min = 0 and *C*max = 5 for both the membership functions S (Small) and L (Large). For the fuzzy variable *y (*range *y* = 0, 25*)* *C*min = 0 and *C*max = 25.



Soft computing

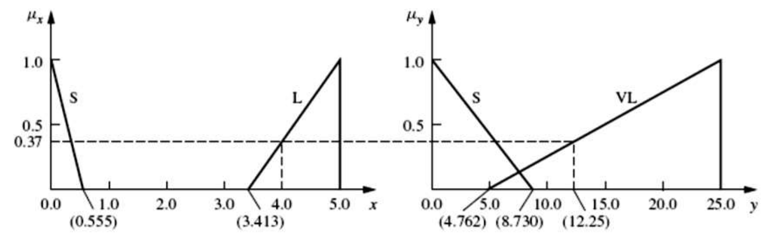



**FIGURE 6.14**
Physical representation of the first string in Table 4.12 and the graphical determination of $y$ for a given $x$.

Soft computing

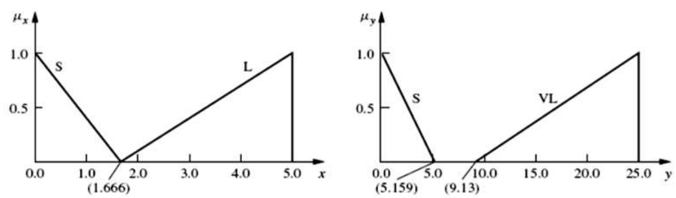



**FIGURE 6.15**
Physical mapping of the best string in the first generation of strings in the genetic algorithm.

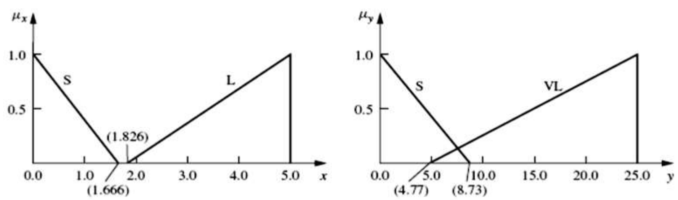



**FIGURE 6.16**
Physical mapping of the best string in the second generation of strings in the genetic algorithm.

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

Soft computing

## INDUCTIVE REASONING

- *Inductive reasoning*, derives a general consensus from the particular (derives the generic from the specific). The induction is performed by the **entropy minimization principle**, which clusters most optimally the parameters corresponding to the output classes (De Luca and Termini, 1972).
- This method is based on an ideal scheme that describes the input and output relationships for a well-established database, that is, the method generates membership functions based solely on the data provided.
  - The method can be quite useful for complex systems where the data are abundant and static.
  - In situations where the data are dynamic, the method may not be useful, since the membership functions will continually change with time

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

Soft computing

## INDUCTIVE REASONING

- Three laws of induction (Christensen, 1980):
  - Given a set of irreducible outcomes of an experiment, the induced probabilities are those probabilities consistent with all available information that maximize the entropy of the set.
  - The induced probability of a set of independent observations is proportional to the probability density of the induced probability of a single observation.
    - appropriate for calculating the mean probability of each step of separation (or partitioning).
  - The induced rule is that rule consistent with all available information of which the entropy is minimum
    - appropriate for classification (or, for our purposes, membership function development)



Soft computing

## INDUCTIVE REASONING

- A key goal of entropy minimization analysis
  - to determine the quantity of information in a given data set.
- The entropy of a probability distribution is a measure of the uncertainty of the distribution (Yager and Filev, 1994).
  - This information measure compares the contents of data to a prior probability for the same data.
  - The higher the prior estimate of the probability for an outcome to occur, the lower will be the information gained by observing it to occur.
- The entropy on a set of possible outcomes of a trial where one and only one outcome is true is defined by the summation of probability and the logarithm of the probability for all outcomes.
  - the entropy is the expected value of information

Soft computing

## INDUCTIVE REASONING

- For a simple one-dimensional (one uncertain variable) case,
  - let us assume that the probability of the $i$ th sample $w_i$ to be true is $\{p(w_i)\}$. If we actually observe the sample $w_i$ in the future and discover that it is true, then we gain the following information, $I(w_i)$:

    $I(w_i) = -k \ln p(w_i)$ ,... (6.16)

    where $k$ is a normalizing parameter.
  - If we discover that it is false, we still gain this information:

    $I(w_i) = -k \ln[1 - p(w_i)]$ ...(6.17)
  - Then the entropy of the inner product of all the samples ($N$) is

    $S = -k \sum_{i=1}^{N} [p_i \ln p_i + (1 - pi) \ln(1 - p_i)]$, .... (6.18)

    where $p_i = p(w_i)$. The minus sign before parameter $k$ in Equation (6.18) ensures that $S \geq 0$, because $\ln x \leq 0$ for $0 \leq x \leq 1$.

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

Soft computing

## INDUCTIVE REASONING

- The third law of induction, which is typical in pattern classification, says that the entropy of a rule should be minimized.
  - Minimum entropy ($S$) is associated with all the $pi$ being as close to ones or zeros as possible, which in turn implies that they have a very high probability of either happening or not happening, respectively.
  - Note in Equation (6.18) that if $pi = 1$ then $S = 0$. This result makes sense since $pi$ is the probability measure of whether a value belongs to a partition or not

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

Soft computing

## INDUCTIVE REASONING

- **Membership function generation**

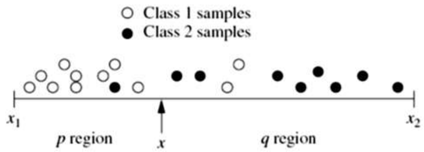



**FIGURE 6.17**
Illustration of threshold value idea.

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

Soft computing

## INDUCTIVE REASONING

- A brief review of the threshold value calculation using the induction principle for a two-class problem.
  - First, we assume that we are seeking a threshold value for a sample in the range between $x1$ and $x2$.
    - Considering this sample alone, we write an entropy equation for the regions [$x1$, $x$] and [$x$, $x2$].
    - We denote the first region $p$ and the second region $q$, as is shown in Figure 6.17.
  - By moving an imaginary threshold value $x$ between $x1$ and $x2$, we calculate entropy for each value of $x$

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

Soft computing

## INDUCTIVE REASONING

An entropy with each value of $x$ in the region $x_1$ and $x_2$ is expressed by Christensen (1980) as

$$S(x) = p(x)S_p(x) + q(x)S_q(x), \tag{6.19}$$

where

$$S_p(x) = -[p_1(x) \ln p_1(x) + p_2(x) \ln p_2(x)], \tag{6.20}$$

$$S_q(x) = -[q_1(x) \ln q_1(x) + q_2(x) \ln q_2(x)], \tag{6.21}$$

where

$p_k(x)$ and $q_k(x)$ = conditional probabilities that the class $k$ sample is in the region $[x_1, x_1 + x]$ and $[x_1 + x, x_2]$, respectively

$p(x)$ and $q(x)$ = probabilities that all samples are in the region $[x_1, x_1 + x]$ and $[x_1 + x, x_2]$, respectively

$p(x) + q(x) = 1.$

A value of $x$ that gives the minimum entropy is the optimum threshold value. We calculate entropy estimates of $p_k(x)$, $q_k(x)$, $p(x)$, and $q(x)$, as follows (Christensen, 1980):

$$p_k(x) = \frac{n_k(x) + 1}{n(x) + 1}, \tag{6.22}$$

Soft computing

## INDUCTIVE REASONING

$$q_k(x) = \frac{N_k(x) + 1}{N(x) + 1}, \tag{6.23}$$

$$p(x) = \frac{n(x)}{n}, \tag{6.24}$$

$$q(x) = 1 - p(x), \tag{6.25}$$

where

$n_k(x)$ = number of class $k$ samples located in $[x_l, x_l + x]$

$n(x)$ = the total number of samples located in $[x_l, x_l + x]$

$N_k(x)$ = number of class $k$ samples located in $[x_l + x, x_2]$

$N(x)$ = the total number of samples located in $[x_l + x, x_2]$

$n$ = total number of samples in $[x_1, x_2]$

$l$ = a general length along the interval $[x_1, x_2]$.

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

Soft computing

## INDUCTIVE REASONING

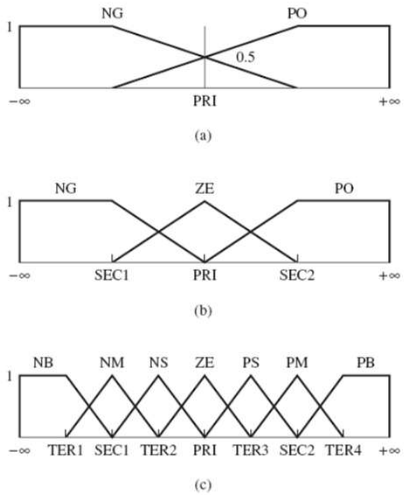



**FIGURE 6.18**
Repeated partitions and corresponding fuzzy set labels: (a) the first partition, (b) the second partition

Source:1

Soft computing

## EXAMPLE

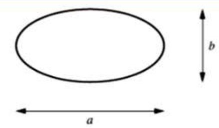



**FIGURE 6.19**
Geometry of an ellipse.

- Example 6.6. The shape of an ellipse may be characterized by the ratio of the length of two chords a and b, as shown in Figure 6.19 (a similar problem was originally posed in Chapter 1; see Figure 6.3).
- Let x = a/b; then as the ratio a/b→∞, the shape of the ellipse tends to a horizontal line, whereas as a/b → 0, the shape tends to a vertical line. For a/b = 1 the shape is a circle.
- Given a set of a/b values that have been classified into two classes (class division is not necessarily based on the value of x alone; other properties like line thickness, shading of the ellipse, etc., may also be criteria), divide the variable x = a/b into fuzzy partitions, as illustrated in Table 6.11.
- First we determine the entropy for different values of x. The value of x is selected as approximately the midvalue between any two adjacent values. Equations (6.19)–(6.25) are then used to compute p1, p2, q1, q2, p(x), q(x), Sp(x), Sq(x), and S ; and the results are displayed in Table 6.12. The value of x that gives the minimum value of the entropy (S )

Source: Fuzzy Logic with Engineering Applications: Timothy J. Ross

**TABLE 6.11**
Segmentation of $x$ into two arbitrary classes (from raw data).

| $x = a/b$ | 0 | 0.1 | 0.15 | 0.2 | 0.2 | 0.5 | 0.9 | 1.1 | 1.9 | 5 | 50 | 100 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Class | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 2 | 2 | 2 | 2 |

**TABLE 6.12**
Calculations for selection of partition point PRI.

| $x$ | **0.7** | **1.0** | **1.5** | **3.45** |
|---|---|---|---|---|
| $p_1$ | $\frac{5+1}{6+1}=\frac{6}{7}$ | $\frac{6+1}{7+1}=\frac{7}{8}$ | $\frac{7+1}{8+1}=\frac{8}{9}$ | $\frac{7+1}{9+1}=\frac{8}{10}$ |
| $p_2$ | $\frac{1+1}{6+1}=\frac{2}{7}$ | $\frac{1+1}{7+1}=\frac{2}{8}$ | $\frac{1+1}{8+1}=\frac{2}{9}$ | $\frac{2+1}{9+1}=\frac{3}{10}$ |
| $q_1$ | $\frac{2+1}{6+1}=\frac{3}{7}$ | $\frac{1+1}{5+1}=\frac{2}{6}$ | $\frac{0+1}{4+1}=\frac{1}{5}$ | $\frac{0+1}{3+1}=\frac{1}{4}$ |
| $q_2$ | $\frac{4+1}{6+1}=\frac{5}{7}$ | $\frac{4+1}{5+1}=\frac{5}{6}$ | $\frac{4+1}{4+1}=1.0$ | $\frac{3+1}{3+1}=1.0$ |
| $p(x)$ | $\frac{6}{12}$ | $\frac{7}{12}$ | $\frac{8}{12}$ | $\frac{9}{12}$ |
| $q(x)$ | $\frac{6}{12}$ | $\frac{5}{12}$ | $\frac{4}{12}$ | $\frac{3}{12}$ |
| $S_p(x)$ | 0.49 | 0.463 | 0.439 | 0.54 |
| $S_q(x)$ | 0.603 | 0.518 | 0.32 | 0.347 |
| $S$ | 0.547 | 0.486 | 0.4 ✓ | 0.49 |



Soft computing

## INDUCTIVE REASONING

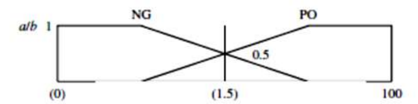



**FIGURE 6.20**
Partitioning of the variable $x = a/b$ into positive (PO) and negative (NG) partitions.

**TABLE 6.13**
Calculations to determine secondary threshold value: NG side.

| $x$ | **0.175** | **0.35** | **0.7** |
|---|---|---|---|
| $p_1$ | $\frac{3+1}{3+1}=1.0$ | $\frac{5+1}{5+1}=1.0$ | $\frac{5+1}{6+1}=\frac{6}{7}$ |
| $p_2$ | $\frac{0+1}{3+1}=\frac{1}{4}$ | $\frac{0+1}{5+1}=\frac{1}{6}$ | $\frac{1+1}{6+1}=\frac{2}{7}$ |
| $q_1$ | $\frac{4+1}{5+1}=\frac{5}{6}$ | $\frac{2+1}{3+1}=\frac{3}{4}$ | $\frac{2+1}{2+1}=1.0$ |
| $q_2$ | $\frac{1+1}{5+1}=\frac{2}{6}$ | $\frac{1+1}{3+1}=\frac{2}{4}$ | $\frac{0+1}{2+1}=\frac{1}{3}$ |
| $p(x)$ | $\frac{3}{8}$ | $\frac{5}{8}$ | $\frac{6}{8}$ |
| $q(x)$ | $\frac{5}{8}$ | $\frac{3}{8}$ | $\frac{2}{8}$ |
| $S_p(x)$ | 0.347 | 0.299 | 0.49 |
| $S_q(x)$ | 0.518 | 0.562 | 0.366 |
| $S$ | 0.454 | 0.398 ✓ | 0.459 |

Soft computing

# INDUCTIVE REASONING

**TABLE 6.14**
Calculations to determine secondary threshold value: PO side.

| $x$ | 27.5 |
|---|---|
| $p_1$ | $\frac{0+1}{2+1} = \frac{1}{3}$ |
| $p_2$ | $\frac{2+1}{2+1} = 1.0$ |
| $q_1$ | $\frac{0+1}{2+1} = \frac{1}{3}$ |
| $q_2$ | $\frac{2+1}{2+1} = 1.0$ |
| $p(x)$ | $\frac{2}{4}$ |
| $q(x)$ | $\frac{2}{4}$ |
| $S_p(x)$ | 0.366 |
| $S_q(x)$ | 0.366 |
| $S$ | 0.366 $\checkmark$ |

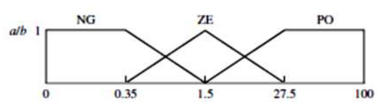



**FIGURE 6.21**
Secondary partitioning for Example 6.6.

Source: Fuzzy Logic with Engineer