# SOLUTION OF AoSC
## MAKE UP

Solution 1a.

To store a bipolar vector pair, the weight matrix is

$$w_{ij} = \sum_{p=1}^{P} s_i(p)\, t_j(p)$$

If the outer product rule is used, then

$$W = \sum_{P} s^T(p)\, t(p)$$

for 1st pair

$$S = [1 \;\; -1 \;\; -1 \;\; -1] \qquad t = [-1 \;\; 1]$$

$$s^T(1)\, t(1) = \begin{bmatrix} 1 \\ -1 \\ -1 \\ -1 \end{bmatrix} [-1 \;\; 1] = \begin{bmatrix} -1 & 1 \\ 1 & -1 \\ 1 & -1 \\ 1 & -1 \end{bmatrix}$$

Similarly find for 2nd, 3rd and 4th pair

The final weight matrix is = $\begin{bmatrix} -4 & 4 \\ -2 & 2 \\ 2 & -2 \\ 4 & -4 \end{bmatrix}$

Test the performance with one mistaken data

Let the test vector be $x = [0 \;\; 1 \;\; 0 \;\; -1]$ with changes made in two components of second input vector $[1 \;\; 1 \;\; -1 \;\; -1]$

$$[y_{in1} \;\; y_{in2}] = [0 \;\; 1 \;\; 0 \;\; -1] \begin{bmatrix} -4 & 4 \\ -2 & 2 \\ 2 & -2 \\ 4 & -4 \end{bmatrix} = [-6 \;\; 6]$$

Applying activations to compute the output, we get

$$[y_1 \ y_2] = [-1 \ 1]$$

Thus, the network has recognized the missing data

Solution 1b:

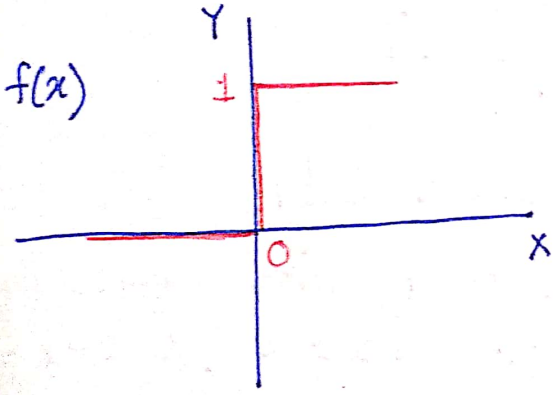
Activation function is an important part of an artificial neural network. They basically decide whether a neuron should be activated or not. Thus it bounds the value of the net input. The activation function is a non-linear transformation that we do over the input before sending it to the next layer of neurons or finalizing it as output.

Types of activation function:

1) Step function

$f(x) = 1$, if $x >= 0$

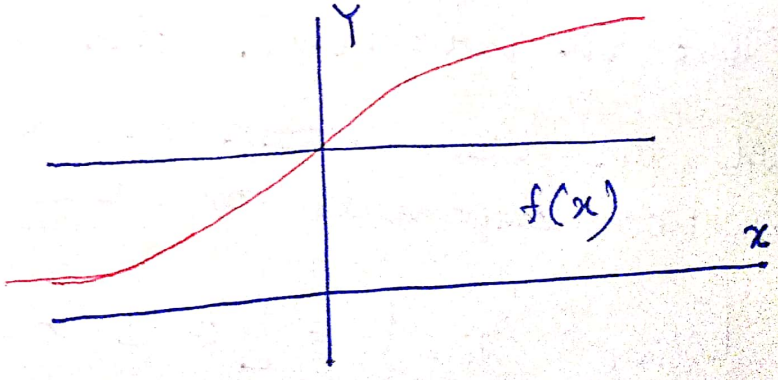
$f(x) = 0$, if $x < 0$




2) Sigmoid function

Sigmoid function is a widely used activation function. It is defined as

$$\frac{1}{1 + e^{-x}}$$

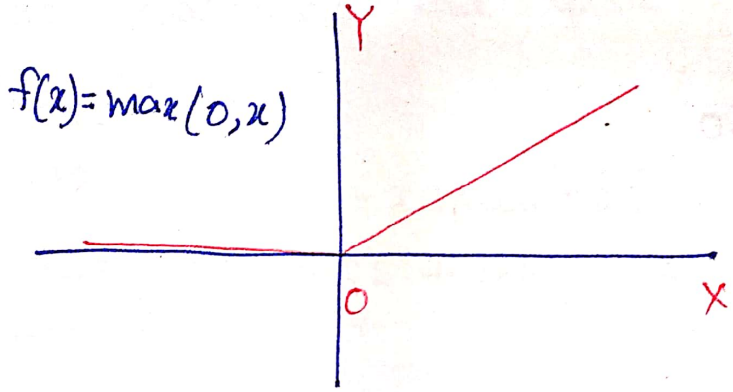


This is a smooth function and is continuously differentiable. The biggest advantage that it has over step and linear function is that it is non-linear.

3. ReLU:

The ReLU function is the Rectified linear Unit. It is most widely used activation function. It is defined as:

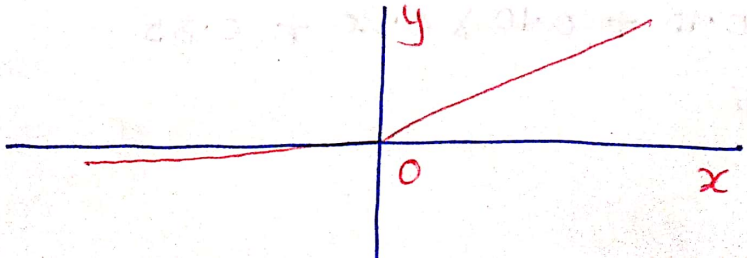
$$f(x) = \max(0, x)$$




4. Leaky ReLU:

Leaky ReLU function is nothing but an improved version of the ReLU function. Instead of defining the ReLU function as 0 for $x$ less than 0, we define it as a small linear component of $x$. It can be defined as:

$$f(x) = ax, \quad x < 0$$
$$f(x) = x, \quad \text{otherwise}$$



Solution 2a:

Values—

$x1 = 0.05$

$x2 = 0.10$

Initial weight

| | |
|---|---|
| $W1 = 0.15$ | $W5 = 0.40$ |
| $W2 = 0.20$ | $W6 = 0.45$ |
| $W3 = 0.25$ | $W7 = 0.50$ |
| $W4 = 0.30$ | $W8 = 0.55$ |

Bias Values

$b1 = 0.35$ $b2 = 0.60$

Target values

$T1 = 0.01$

$T2 = 0.99$

1) Now, we first calculate the values of H1 and H2 by a forward pass.

Forward Pass—

To find the value of H1 we first multiply the input value from the weights as

$$H1 = x1 \times w_1 + x2 \times w_2 + b1$$

$$H1 = 0.05 \times 0.15 + 0.10 \times 0.20 + 0.35$$

$$H1 = 0.3775$$

To calculate the final result of H1, we perform the sigmoid function as

$$H1_{final} = \frac{1}{1 + \frac{1}{e^{1/H1}}}$$

$$H1_{final} = \frac{1}{1 + e^{1/0.3775}}$$

$$H1\ final = 0.593269992$$

We will calculate the value of H2 in the same way as H1

$$H2 = x1 \times w_3 + x2 \times w_4 + b$$

$$H2 = 0.05 \times 0.25 + 0.10 \times 0.30 + 0.35$$

$$H2 = 0.3925$$

To calculate the final result of H1, perform sigmoid fn. as —

$$H_{2\text{-}final} = \frac{1}{1 + \frac{1}{e^{1/H2}}} = 0.596884378$$

Now calculate the values of y1 & y2 in the same way as we calculate the H1 & H2

To find the value of y1, we first multiply the input value i.e the outcome of H1 & H2 from the weights as

$$y1 = H1 \times w_5 + H2 \times w_6 + b2$$

$$y_1 = 1.10590597$$

$$y_{1final} = \frac{1}{1 + e^{1/y1}} = 0.75136507$$

Calculate y2 to in same way

$$y2_{final} = 0.772928465$$

$$E_{total} = \sum \frac{1}{2}(target - output)^2$$

So the total error is

$$= \frac{1}{2}(t1 - y1_{final})^2 + \frac{1}{2}(T2 - y2_{final})^2$$

$$= \frac{1}{2}(0.01 - 0.75136507)^2 + \frac{1}{2}(0.99 - 0.772928)^2$$

$$E_{total} = 0.29837111$$

Now, we will back propagate this error to update the weights using backward pass -

Backward pass at the output layer

To update the weight, we calculate the error corresponding to each weight with the help of a total error. The error on wt. w is calculated by differentiating total error with respect to w.

$$Error = \frac{\delta E_{total}}{\delta w}$$

$$Error_{w5} = \frac{\delta E_{total}}{\delta w_5} \quad \text{---(1)}$$

$$E_{total} = \frac{1}{2}(T1 - y1_{final})^2 + \frac{1}{2}(T2 - y2_{final})^2 \quad \text{---(2)}$$

$$\frac{\delta E_{total}}{\delta w_5} = \frac{\delta E_{total}}{\delta y1_{final}} \times \frac{\delta y1_{final}}{\delta y1} \times \frac{\delta y1}{\delta w_5} \quad \text{---(3)}$$

Now calculate each term one by one to differentiate $E_{total}$ with respect to $w5$ as

$$\frac{\delta E_{total}}{\delta y1_{final}} = \frac{\delta\left(\frac{1}{2}(T1 - y1_{final})^2 + \frac{1}{2}(T2 - y2_{final})^2\right)}{\delta y1_{final}}$$

$$= 2 \times \frac{1}{2} \times (T1 - y1_{final})^{2-1} \times (-1) + 0$$

$$= -(T1 - y1_{final})$$

$$= -(0.01 - 0.75136507)$$

$$\frac{\delta E_{total}}{\delta y1_{final}} = 0.74136507 \quad \text{---(4)}$$

$$y1_{final} = \frac{1}{1 + e^{-y1}} \quad \text{---(5)}$$

$$\frac{\delta y1_{final}}{\delta y1} = \frac{\delta\left(\frac{1}{1+e^{-y1}}\right)}{\delta y1} = \frac{e^{-y1}}{(1+e^{-y1})^2}$$

$$= e^{-y1} \times (y1_{final})^2 \quad \text{---(6)}$$

$$y1_{final} = \frac{1}{1 + e^{-y1}}$$

$$e^{-y1} = \frac{1 - y1_{final}}{y1_{final}} \quad \text{——} (7)$$

Putting the value of $e^{-y}$ in eqn. (5)

$$= \frac{1 - y1_{final}}{y1_{final}} \times (y1_{final})^2$$

$$= y1_{final}\,(1 - y1_{final})$$

$$= 0.75136507 \times (1 - 0.75136507)$$

$$\frac{Sy1_{final}}{Sy1} = 0.186815602 \quad \text{——} (8)$$

$$y1 = H1_{final} \times w5 + H2_{final} \times w6 + b2 \quad \text{——} (9)$$

$$\frac{Sy1}{Sw5} = \frac{S(H1_{final} \times w5 + H2_{final} \times w6 + b2)}{Sw5}$$

$$= H1_{final}$$

$$\frac{Sy1}{Sw5} = 0.596884378 \quad \text{——} (10)$$

So, we put the values of $\frac{SE_{total}}{Sy1_{final}}$, $\frac{dy_{1final}}{Sy1}$ & $\frac{Sy1}{Sw5}$ in eqn. (3) to find the final result

$$\frac{SE_{total}}{Sw5} = \frac{SE_{total}}{Sy1_{total}} \times \frac{Sy1_{final}}{Sy1} \times \frac{Sy1}{Sw5}$$

$$= 0.75136507 \times 0.186815602 \times 0.593269992$$

$$Error_{w5} = \frac{SE_{total}}{Sw5} = 0.082167040 7 \quad \text{——} (11)$$

Now, we calculate the updated weight $w5_{new}$ with the help of following formula

$$W_{5new} = W_5 - \eta \times \frac{\delta E_{total}}{\delta w_5}$$ Here $\eta$ = learning rate $= 0.5$

$$= 0.4 - 0.5 \times 0.0821670407$$

$$W5_{new} = 0.35891648 \quad \text{—— (12)}$$

In the same way, calculate $W6_{new}$, $W7_{new}$, $W8_{new}$

$$W5_{new} = 0.35891648$$

$$W6_{new} = 0.408666186$$

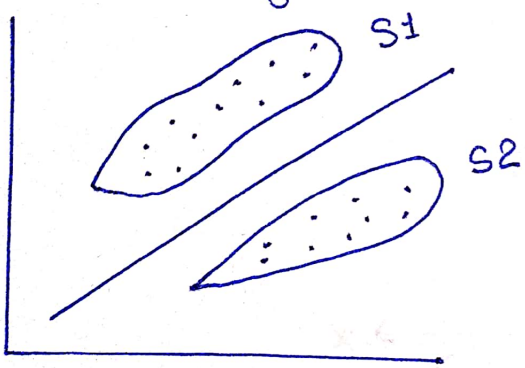
$$W7_{new} = 0.511301270$$

$$W8_{new} = 0.561370121$$

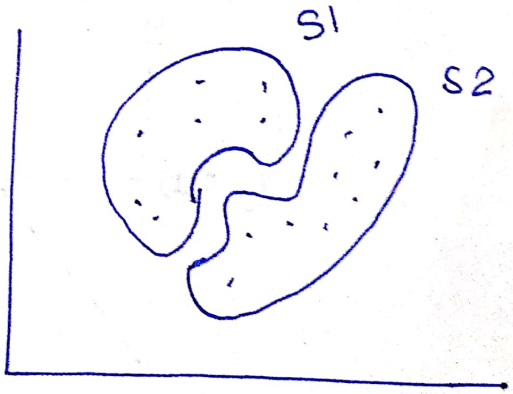
Solution 2b—

Perceptron cannot handle, a task which are not linearly separable if the if the sets can be separated by a straight line.

Generalizing, a set of points in n-dimensional space are linearly separable if there is a hyperplane of (n-1) dimensions that separates the sets. Fig. a below linearly separable patterns and non-linearly separable patterns.




(a) Linearly separable Patterns




(b) Non-linearly separable Patterns

The perceptron cannot find weights for classification problems that are not linearly separable. Consider this type of problems and

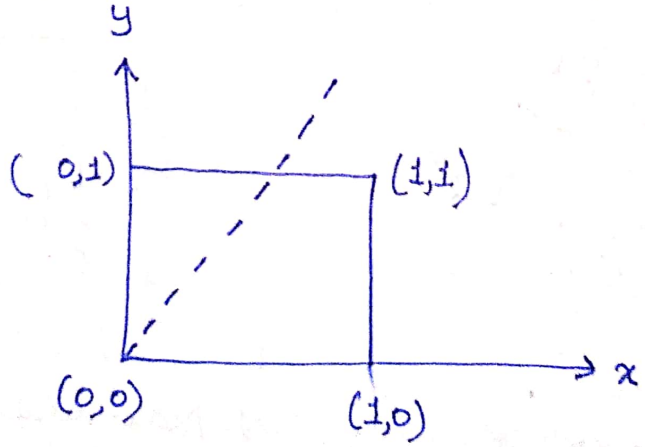
## XOR Problem

XOR is a logical operation as described by truth table

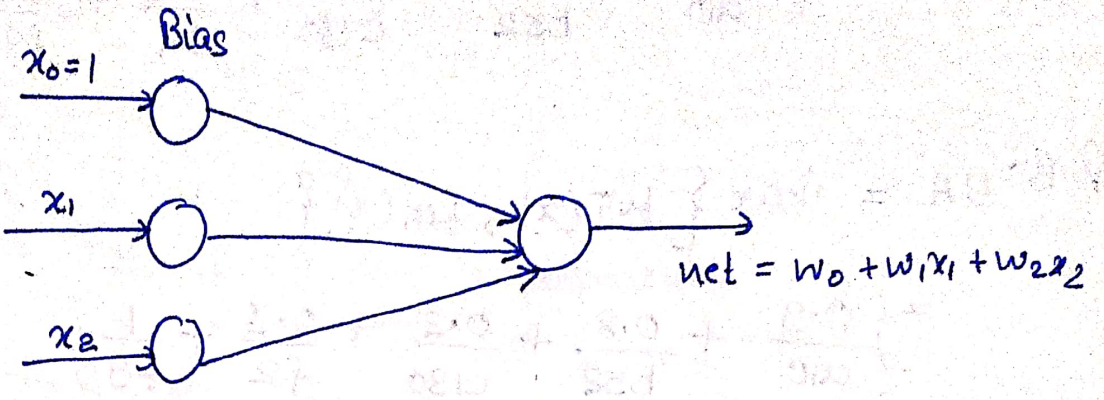
| Inputs | Inputs | Outputs | |
|---|---|---|---|
| 0 | 0 | 0 | → Even parity |
| 1 | 1 | 0 | → Even parity |
| 0 | 1 | 1 | → odd parity |
| 1 | 0 | 1 | → odd parity |

The problem for the ANN is to classify the inputs as odd parity or even parity. Here, odd parity means odd number of 1 bits in the inputs & even parity refers to even no. of 1 bits in the inputs.

This is impossible, since as is evident from figure below the perceptron is unable to find a line separating even Parity input patterns from odd parity inputs patterns



Consider a perceptron network with two inputs $x_1$ and $x_2$ and bias $x_0 = 1$. The weighted sum of inputs




$$net = w_0 + w_1 x_1 + w_2 x_2$$

represents the eqn. of a straight line

The straight line act as a decision boundry separing the points into classes $C_1$ & $C_2$, above and below the line respectively.


This is what the perceptron aims to do for a problem when it is able to obtain their weights.

Solution 3a

a) $A \cup B = \max \{ \mu_A(x), \mu_B(x) \}$

$$= \left\{ \frac{0.3}{a10} + \frac{0.4}{b52} + \frac{0.8}{c130} + \frac{0.7}{f2} + \frac{1}{f9} \right\}$$

b) $(A \cap B)^c = 1 - \min \{ \mu_A(x), \mu_B(x) \}$

$$= \left\{ \frac{0.9}{a10} + \frac{0.8}{b52} + \frac{0.8}{c130} + \frac{0.9}{f2} + \frac{1}{f9} \right\}$$

c) $(A \cup B)^c = 1 - \max\{\mu_A(x), \mu_B(x)\}$

$$= \left\{ \frac{0.7}{a10} + \frac{0.6}{b52} + \frac{0.2}{c130} + \frac{0.3}{f2} + \frac{0}{f9} \right\}$$

d) $B^c \cup A = \max\{\mu_{\bar{B}}(x), \mu_A(x)\}$

$$= \left\{ \frac{0.9}{a10} + \frac{0.8}{b52} + \frac{0.2}{c130} + \frac{0.3}{f2} + \frac{1}{f9} \right\}$$

e) $A^c \cap B = \min\{\mu_{\bar{A}}(x), \mu_B(x)\}$

$$= \left\{ \frac{0.1}{a10} + \frac{0.2}{b52} + \frac{0.8}{c130} + \frac{0.7}{f2} + \frac{0}{f9} \right\}$$

Solution 3b

To determine (i) compute $\tilde{R} = (\tilde{A} \times \tilde{B}) \cup (\bar{\tilde{A}} \times Y)$ where

$$\mu_{\tilde{R}}(x,y) = \max(\min(\mu_{\tilde{A}}(x), \mu_B(y)), 1 - \mu_{\tilde{A}}(x))$$

$\tilde{A} \times \tilde{B}$ =

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| a | 0 | 0 | 0 | 0 |
| b | 0.2 | 0.8 | 0.8 | 0 |
| c | 0.2 | 0.6 | 0.6 | 0 |
| d | 0.2 | 1 | 0.8 | 0 |

$\bar{\tilde{A}} \times Y$ =

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| a | 1 | 1 | 1 | 1 |
| b | 0.2 | 0.2 | 0.2 | 0.2 |
| c | 0.4 | 0.4 | 0.4 | 0.4 |
| d | 0 | 0 | 0 | 0 |

Here, Y the universe of discourse could be viewed as $\{(1,1)\ (2,1)\ (3,1)\ (4,1)\}$ a fuzzy set all of whose elements $x$ have $\mu(x) = 1$

Therefore

$$\vec{R} = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0.2 & 0.8 & 0.8 & 0.2 \\ 0.4 & 0.6 & 0.6 & 0.4 \\ 0.2 & 0.1 & 0.8 & 0 \end{bmatrix} \end{array}$$

represents

If $x$ is $\vec{A}$ Then $y$ is $\vec{B}$

ii)

$\vec{R} = (\vec{A} \times \vec{B}) \cup (\overline{\vec{A}} \times \vec{C})$ where

$\mu_{\vec{R}}(x,y) = \max(\min(\mu_{\vec{A}}(x), \mu_{\vec{B}}(y)), \min(1 - \mu_{\vec{A}}(x), \mu_{\vec{C}}(y)))$

$$\vec{A} \times \vec{B} = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0.2 & 0.8 & 0.8 & 0 \\ 0.2 & 0.6 & 0.6 & 0 \\ 0.2 & 1 & 0.8 & 0 \end{bmatrix} \end{array}$$

$$\overline{\vec{A}} \times \vec{C} = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 0.4 & 1 & 0.8 \\ 0 & 0.2 & 0.2 & 0.2 \\ 0 & 0.4 & 0.4 & 0.4 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

$\vec{R} = \max((\vec{A} \times \vec{B}), (\overline{\vec{A}} \times \vec{C}))$ gives

$\vec{R} =$

$$\underset{\sim}{R} = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 0.4 & 1 & 0.8 \\ 0.2 & 0.8 & 0.8 & 0.2 \\ 0.2 & 0.6 & 0.6 & 0.4 \\ 0.2 & 1 & 0.8 & 0 \end{bmatrix} \end{array}$$

The above $\underset{\sim}{R}$ represents IF $x$ is $\tilde{A}$ THEN $y$ is $\tilde{B}$ EISE $y$ is $\tilde{C}$.

## Solution 4a

$$\underset{\sim}{R} = \begin{array}{c} \\ x_1 \\ x_2 \end{array} \begin{array}{c} \begin{array}{cc} & \end{array} \\ \begin{bmatrix} 0.6 & 0.3 \\ 0.2 & 0.9 \end{bmatrix} \end{array} \qquad \underset{\sim}{S} = \begin{array}{c} \\ y_1 \\ y_2 \end{array} \begin{array}{c} \begin{array}{ccc} z_1 & z_2 & z_3 \end{array} \\ \begin{bmatrix} 1 & 0.5 & 0.3 \\ 0.8 & 0.4 & 0.7 \end{bmatrix} \end{array}$$

a) Max – min composition

$$\underset{\sim}{T} = \underset{\sim}{R} \circ \underset{\sim}{S} = \begin{array}{c} \\ x_1 \\ x_2 \end{array} \begin{array}{c} \begin{array}{ccc} z_1 & z_2 & z_3 \end{array} \\ \begin{bmatrix} 0.6 & 0.5 & 0.3 \\ 0.8 & 0.4 & 0.7 \end{bmatrix} \end{array}$$

$$\mu_T(x_1, z_1) = \max \{ \min [\mu_R(x_1, y_1), \mu_S(y_1, z_1)]$$

$$= \max \{ \min (0.6, 1), \min (0.3, 0.8) \}$$

$$= 0.6$$

$(x_1, z_2) = 0.5$

$(x_1, z_3) = 0.3$

$\mu_{I_4}(x_2, z_1) = 0.8$

$\mu_{I_4}(x_2, z_2) = 0.4$

$\mu_{I_4}(x_2, z_3) = 0.7$

b) Max- product

$$I_4 = R_4 \bullet S_4$$

$$\mu_{I_4}(x_1, z_1) = \max\{[\mu_{R_4}(x_1, y_1) \bullet \mu_{S_4}(y_1, z_1)], [\mu_{R_4}(x_1, y_2) \bullet \mu_{S_4}(y_2, z_1)]\}$$

$$= \max(0.6, 0.24)$$

$$= 0.6$$

Similarly

$\mu_{I_4}(x_1, z_2) =$ ~~max~~ $[(0.6 \times 0.5), (0.3$ ~~$0.3$~~

$\mu_{I_4}(x_1, z_3) = 0.21$

$\mu_{I_4}(x_2, z_1) = 0.72$

$\mu_{I_4}(x_2, z_2) = 0.36$

$\mu_{I_4}(x_2, z_3) = 0.63$

$$I_4 = R_4 \bullet S_4 = \begin{matrix} \\ x_1 \\ x_2 \end{matrix} \begin{matrix} \begin{matrix} z_1 & z_2 & z_3 \end{matrix} \\ \begin{bmatrix} 0.6 & 0.3 & 0.21 \\ 0.72 & 0.36 & 0.63 \end{bmatrix} \end{matrix}$$

Solution 4b –

Membership value of isoceles triangle, $I_4$:

$$\mu_{I_4} = 1 - \frac{1}{60^\circ} \min(x-y, y-z)$$

$$= 1 - \frac{1}{60^\circ} \min(80^\circ - 60, 60^\circ - 40)$$

$$= 1 - \frac{1}{60^\circ} \min(20^\circ, 20^\circ)$$

$$= 1 - \frac{1}{60^\circ} \times 20^\circ = 0.667$$

Membership value of right-angle triangle, $R_4$:

$$\mu_{R_4} = 1 - \frac{1}{90^\circ} |x - 90^\circ| = 1 - \frac{1}{90^\circ} |80^\circ - 90^\circ|$$

$$= 1 - \frac{1}{90^\circ} \times 10^\circ = 0.889$$

Membership value of other triangles, $I_4$:

$$\mu_{I_4} = \min [1 - \mu_{I_4}, 1 - \mu_{R_4}]$$

$$= \min [1 - 0.667, 1 - 0.889]$$

$$= \min [0.333, 0.111] = 0.111$$

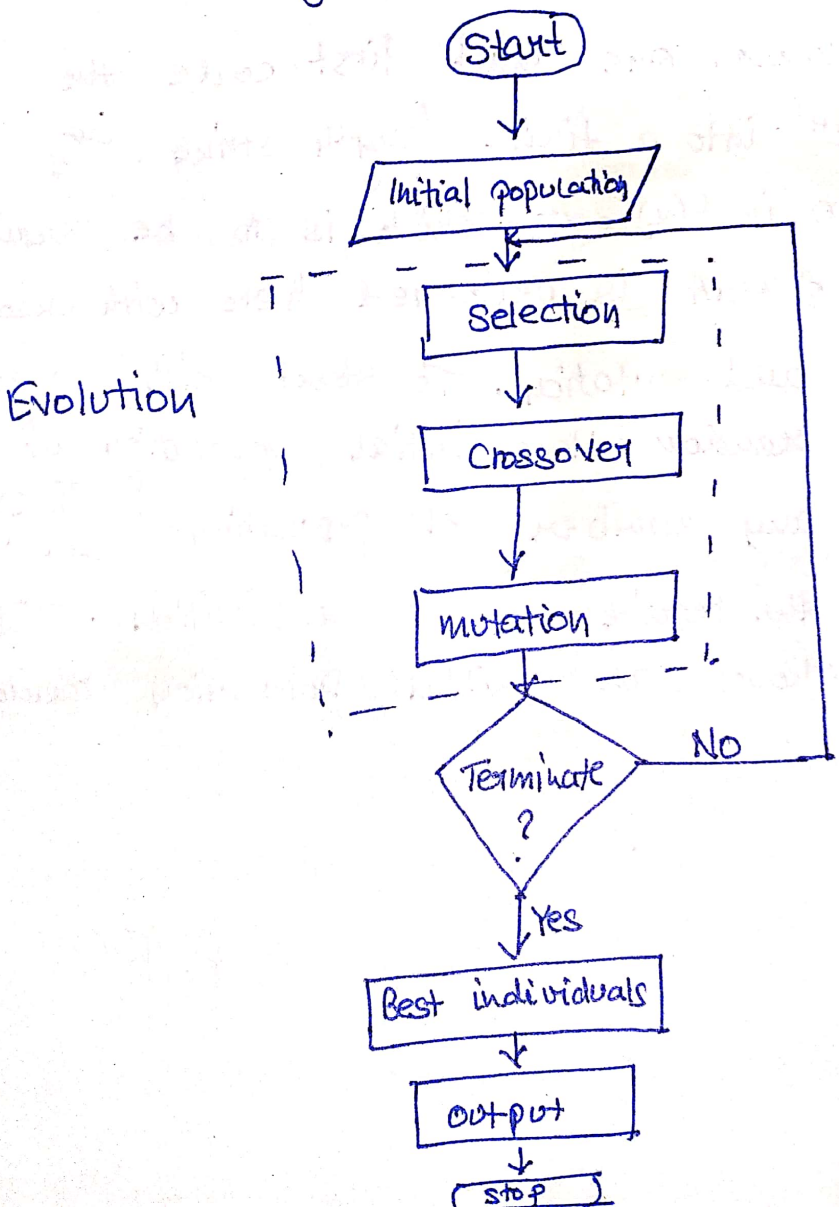
## Solution 5a

The general GA is as follows:

Step 1: Create a random initial state → An initial population is created from a random selection of solutions.

Step 2: Evaluate fitness — A value for fitness is assigned to each solution (chromosome) depending on how close it actually is to solving the problem.

Step 3: Reproduce (and children mutate): Those chromosomes with a higher fitness value are more likely to reproduce offspring. The offspring is a product of the father and mother, whose composition consists of a combination of genes from the two (this process is known as "crossingover")



Step 4 → Next Generation — If the new generation contains a solution that produces an output that is close enough or equal to the desired answer then the problem has been solved. If this is not the case then the new generation will go through the same process as their parents did. This will continue until a solution is reached.

## Solution 5b

### Step 1

For using GA approach, one must first code the decision variable "x" into a finite length string. The objective function here is $f(x) = x^2$ which is to be maximized. A single generation of a GA is performed here with encoding, selection, crossover and mutation. To start with, select initial population at random. Here initial population of size 4 is chosen, but any number of populations can be selected based on the requirement of application. Table on next page shows an initial population randomly selected

P.T.O

| String no. | Initial population (randomly selected) | $x$ value | Fitness $f(x)=x^2$ | Prob$_i$ | Percentage probability (%) | Expected count | Actual count |
|---|---|---|---|---|---|---|---|
| 1 | 01100 | 12 | 144 | 0.1247 | 12.47 | 0.4987 | 1 |
| 2 | 11001 | 25 | 625 | 0.5411 | 54.11 | 2.1645 | 2 |
| 3 | 00101 | 5 | 25 | 0.0216 | 2.16 | 0.0866 | 0 |
| 4 | 10011 | 19 | 361 | 0.3126 | 31.26 | 1.2502 | 1 |
| Sum | | | 1155 | 1.0000 | 100 | 4.0000 | 4 |
| Average | | | 288.75 | 0.2500 | 25 | 1.0000 | 1 |
| Maximum | | | 625 | 0.5411 | 54.11 | 2.1645 | 2 |

Step 2: Obtain the decoded $x$ values for the initial population generated. Consider string 1,

$$\begin{aligned} 01100 &= 0*2^4 + 1*2^3 + 1*2^2 + 0*2^1 + 0*2^0 \\ &= 0 + 8 + 4 + 0 + 0 \\ &= 12 \end{aligned}$$

Thus for all the four strings the decoded values are obtained.

Step 3: Calculate the fitness or objective function. This is obtained by simply squaring the "$x$" value, since the given function is $f(x) = x^2$. When $x = 12$, the fitness value is

$$f(x) = x^2 = (12)^2 = 144$$

For $x = 25$, $f(x) = x^2 = (25)^2 = 625$

and so on, until the entire population is computed.

Step 4: Compute the probability of selection,

$$\text{Prob}_i = \frac{f(x)_i}{\sum_{i=1}^{n} f(x)_i} \tag{15.15}$$

where $n$ is the number of populations; $f(x)$ is the fitness value corresponding to a particular individual in the population;

$\Sigma f(x)$ is the summation of all the fitness value of the entire population.

Considering string 1,

$$\text{Fitness } f(x) = 144$$
$$\Sigma f(x) = 1155$$

The probability that string 1 occurs is given by

$$P_1 = 144/1155 = 0.1247$$



Scanned with OKEN Scanner

percentage probability is obtained as

$$0.1247 * 100 = 12.47\%$$

The same operation is done for all the strings. It should be noted that summation of probability select is 1.

The next step is to calculate the expected count, which is calculated as

$$\text{Expected count} = \frac{f(x)_i}{[\text{Avg}\, f(x)]_i} \tag{15.16}$$

where

$$(\text{Avg}\, f(x))_i = \left[\frac{\sum_{i=1}^{n} f(x)_i}{n}\right]$$

For string 1,

$$\text{Expected count} = \text{Fitness/Average} = 144/288.75 = 0.4987$$

We then compute the expected count for the entire population. The expected count gives an idea of which population can be selected for further processing in the mating pool.

6: Now the actual count is to be obtained to select the individuals who would participate in the crossover cycle using Roulette wheel selection. The Roulette wheel is formed as shown Figure 15-33.

The entire Roulette wheel covers 100% and the probabilities of selection as calculated in step 4 for the entire populations are used as indicators to fit into the Roulette wheel. Now the wheel may be spun and the number of occurrences of population is noted to get actual count.

1. String 1 occupies 12.47%, so there is a chance for it to occur at least once. Hence its actual count may be 1.
2. With string 2 occupying 54.11% of the Roulette wheel, it has a fair chance of being selected twice. Thus its actual count can be considered as 2.
3. On the other hand, string 3 has the least probability percentage of 2.16%, so their occurrence for next cycle is very poor. As a result, it actual count is 0.

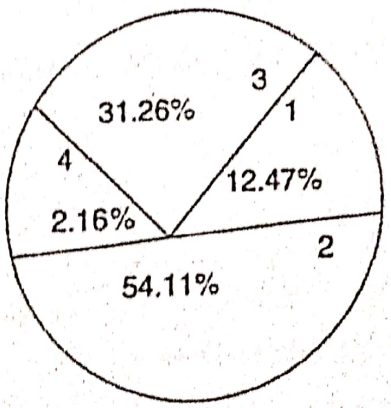



**Figure 15-33** Selection using Roulette wheel.

| String no. | Mating Pool | Crossover point | Offspring after crossover | $x$ value | Fitness value $f(x)=x^2$ |
|---|---|---|---|---|---|
| 1 | 0 1 1 0 0 | 4 | 0 1 1 0 1 | 13 | 169 |
| 2 | 1 1 0 0 1 | 4 | 1 1 0 0 0 | 24 | 576 |
| 3 | 1 1 0 0 1 | 2 | 1 1 0 1 1 | 27 | 729 |
| 4 | 1 0 0 1 1 | 2 | 1 0 0 0 1 | 17 | 289 |
| Sum | | | | | 1763 |
| Average | | | | | 440.75 |
| Maximum | | | | | 729 |

4. String 4 with 31.26% has at least one chance for occurring while Roulette wheel is spun, thi its actual count is 1.

The above values of actual count are tabulated as shown is Table 15-5.

Step 7: Now, write the mating pool based upon the actual count as shown in Table 15-5.

The actual count of string no. 1 is 1, hence it occurs once in the mating pool. The actual count of string no. 2 is 2, hence it occurs twice in the mating pool. Since the actual count of string no. 3 is 0, it does not occur in the mating pool. Similarly, the actual count of string no. 4 being 1, it occurs once in the mating pool. Based on this, the mating pool is formed.

Step 8: Crossover operation is performed to produce new offspring (children). The crossover point is specified and based on the crossover point, single-point crossover is performed and new offspring is produced. The parents are

| | |
|---|---|
| Parent 1 | 0 1 1 0 0 |
| Parent 2 | 1 1 0 0 1 |

The offspring is produced as

| | |
|---|---|
| Offspring 1 | 0 1 1 0 1 |
| Offspring 2 | 1 1 0 0 0 |

In a similar manner, crossover is performed for the next strings.

Step 9: After crossover operations, new offspring are produced and "$x$" values are decoded and fitness i calculated.

Step 10: In this step, mutation operation is performed to produce new offspring after crossover operation As discussed in Section 15.9.4.1 mutation-flipping operation is performed and new offspring are produced. Table 15-6 shows the new offspring after mutation. Once the offspring are obtainec after mutation, they are decoded to $x$ value and the fitness values are computed.

This completes one generation. The mutation is performed on a bit-bit by basis. The crossover probabili and mutation probability were assumed to be 1.0 and 0.001, respectively. Once selection, crossover a

| String no. | Offspring after crossover | Mutation chromosomes for flipping | Offspring after mutation | $x$ value | Fitness $f(x)=x^2$ |
|---|---|---|---|---|---|
| 1 | 0 1 1 0 1 | 1 0 0 0 0 | 1 1 1 0 1 | 29 | 841 |
| 2 | 1 1 0 0 0 | 0 0 0 0 0 | 1 1 0 0 0 | 24 | 576 |
| 3 | 1 1 0 1 1 | 0 0 0 0 0 | 1 1 0 1 1 | 27 | 729 |
| 4 | 1 0 0 0 1 | 0 0 1 0 0 | 1 0 1 0 0 | 20 | 400 |
| Sum | | | | | 2546 |
| Average | | | | | 636.5 |
| Maximum | | | | | 841 |

mutation are performed, the new population is now ready to be tested. This is performed by decoding th new strings created by the simple GA after mutation and calculates the fitness function values from the values thus decoded. The results for successive cycles of simulation are shown in Tables 15-4 and 15-6.

From the tables, it can be observed how GAs combine high-performance notions to achieve better performance. In the tables, it can be noted how maximal and average performances have improved in the new population. The population average fitness has improved from 288.75 to 636.5 in one generation. The maximum fitness has increased from 625 to 841 during the same period. Although random processes make this best solution, its improvement can also be seen successively. The best string of the initial population (1 1 0 0 1) receives two chances for its existence because of its high, above-average performance. When this combines at random with the next highest string (1 0 0 1 1) and is crossed at crossover point 2 (as shown in Table 15-5), one of the resulting strings (1 1 0 1 1) proves to be a very best solution indeed. Thus after mutation at random, a new offspring (1 1 1 0 1) is produced which is an excellent choice.

This example has shown one generation of a simple GA.



Scanned with OKEN Scanner