CI Endsem solution and evaluation scheme, Autumn 2024

Q1.

a) Human expertise, biologically inspired computing models, new optimization techniques, numerical computation, new application domains, model-free learning, intensive computataion, fault tolerance, goal-driven characteristics, real-world applications (any 5)

b) Discrete unordered –
Let X = {San Francisco, Boston, Los Angeles} be the set of cities one may choose to live in. The fuzzy set C= "desirable city to live in" may be described as follows:
C= {(San Francisco, 0.9), (Boston, 0.8), (Los Angeles, 0.6)}.

Discrete ordered –
Let X = {0, 1, 2, 3, 4, 5, 6} be the set of numbers of children a family may choose to have. Then the fuzzy set A= "sensible number of children in a family" may be described as follows:
A= {(0,0.1), (1, 0.3), (2, 0.7), (3, 1), (4, 0.7), (5, 0.3), (6, 0.1)}

Continuous –
Let X= R+ be the set of possible ages for human beings. Then the fuzzy set B= "about 50 years old" may be expressed as

$$B = \{(x, \mu_B(x) | x \in X\},$$

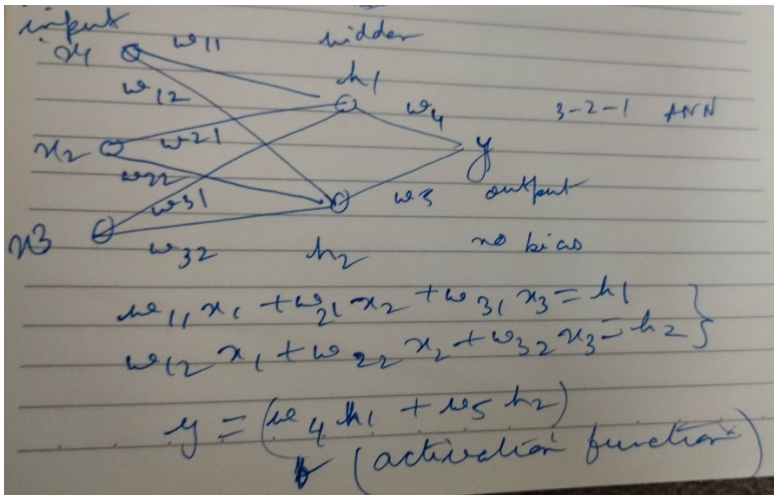
where

$$\mu_B(x) = \frac{1}{1 + \left(\frac{x-50}{10}\right)^4}.$$

c) Yager's complement is given by $N_w(a) = (1-a^w)^{1/w}$
A´ = {1/1 + 0.9/2 + 1/3 + 0.7/4 + 0.6/5 + 1/6 + 0.4/7 + 1/8 + 0.3/9}

d) Learning rate of an ANN model - The learning rate of an ANN model is a hyperparameter that controls the size of the steps taken to minimize the loss function.

e)



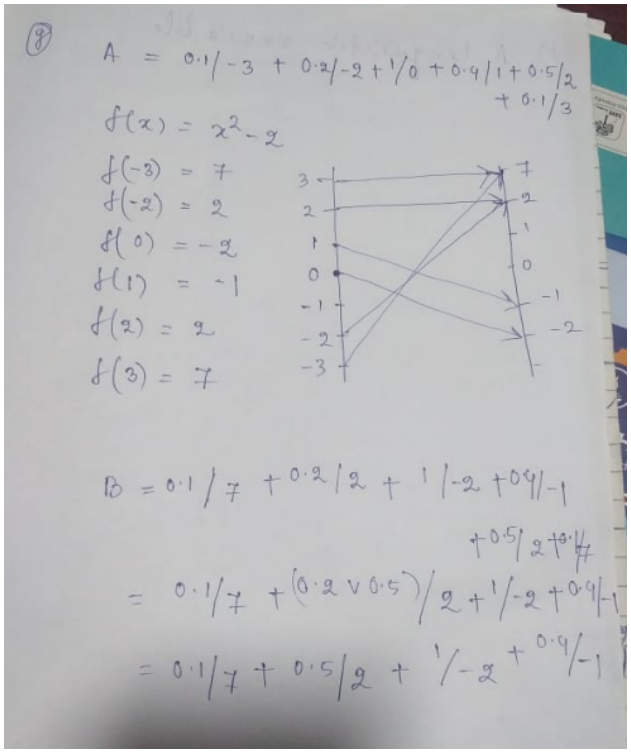
f) Support (A) is all elements with membership grade greater than zero.
So Support (A) ={x1, x2, x3, x4, x5, x6, x7}

Core (A) is membership grade is 1.
So Core (A) = {x5}
alpha cut A is membership grade greater than or equal to 0.3
so {x2,x3,x4,x5,x6}

Strong alpha core means strictly greater than 0.3

So { x3,x4,x5,x6}

g)



h) A **local minimum** is a point in the solution space of an optimization problem where the objective function value is lower than at any nearby points.
A **global minimum**, on the other hand, is a point in the solution space where the objective function value is lower than or equal to the value at any other point in the entire space. It represents the absolute lowest point of the objective function.

i)

Aspect	**PSO**	**GA**
Inspiration	Social behavior of birds/fish	Natural selection and genetics
Population Entities	Particles	Chromosomes
Movement	Based on velocity and best positions	Based on crossover and mutation

Aspect	PSO	GA
Information Sharing	Personal best and global best positions	Genetic material exchange through crossover
Memory	Particles remember personal best positions	No memory of individual best positions
Diversity Mechanism	Limited, mainly through swarm behavior	Maintained through crossover and mutation
Parameters	Fewer parameters to tune	More parameters to tune
Convergence Speed	Generally faster but can get stuck	Generally slower but more robust

j)

A linguistic variable is a variable whose values are not numbers but words or sentences in natural or artificial language. Each linguistic value is characterized by a fuzzy set. A linguistic variable is typically defined as a quintuple (five-tuple) (X,T(X),U,G,M).

1. X: The name of the linguistic variable.
2. T(X): The set of linguistic terms (values) that the variable can take.
3. U: The universe of discourse, which is the range of all possible values that the linguistic variable can assume.
4. G: A syntactic rule or grammar that generates the terms in T(X).
5. M: A semantic rule that associates each linguistic term with its meaning, represented as a fuzzy set in the universe of discourse U.

Example: Linguistic Variable "Temperature"

Let's define the linguistic variable "Temperature" as a quintuple.

X: Temperature.

T(X): {Very Cold, Cold, Warm, Hot, Very Hot}.

U: The universe of discourse could be the range of temperature values in degrees Celsius, say [−30,50][-30, 50][−30,50].

G: The syntactic rule could be the natural language grammar that generates the terms "Very Cold," "Cold," "Warm," "Hot," and "Very Hot."

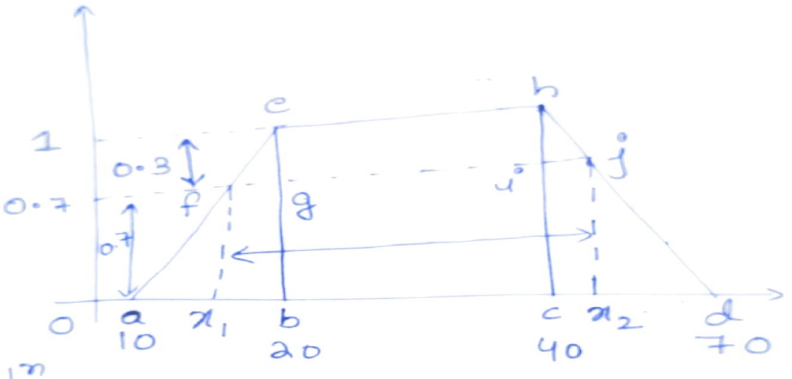
M: The semantic rule would associate each term in T(X) with a fuzzy set.

Q2.

a)

Q2a · $A_{0.7}$

1. Trapezoid (x; 10, 20, 40, 70)
2. Gaussian (x; 10, 3)



Soln

$ab = 10$

$bc = 40 - 20 = 20$

$dc = 70 - 40 = 30$

Step 1: Δefg & Δaeb

$$\frac{fg}{eg} = \frac{ab}{be}$$

$$\frac{fg}{0.7} = \frac{10}{1}$$

$$fg = 3$$

Step 2: Δhij & Δhcd

$$\frac{ij}{hi} = \frac{cd}{hc}$$

$$\frac{ij}{0.3} = \frac{30}{1}$$

$$ij = 30 \times 0.3$$

$$ij = 9$$

Step 3:

$$A_{0.7} = fg + bc + ij$$
$$= 3 + 20 + 9$$
$$= 23 + 9$$
$$= 32 \text{ ans.}$$

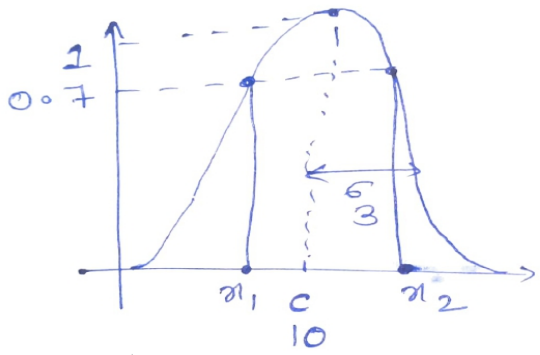
Width $A_{0.7}$ = 32 ans

bandwidth = $|x_2 - x_1|$

= $|9-3|$

= 6

2. Gaussian$(x; 10, 3)$



Soln. :-

$[\text{width } A_\alpha] = 2\sqrt{-2\ln\alpha}$

width $A_{0.7} = 2\sqrt{-2\ln 0.7}$

bandwidth = $e^{-1/2\left[\frac{\text{width}(A_\alpha)}{26}\right]^2}$

$= e^{-1/2\left[\frac{2\sqrt{-2\ln(0.7)}}{26}\right]^2}$

$= e^{-1/2\left[\frac{\sqrt{-2\ln 0.7}}{6}\right]^2}$

$= e^{-1/2\left[\frac{\sqrt{2\times -0.3566}}{3}\right]^2}$

$= e^{-1/2\left[\frac{-0.7132}{9}\right]}$

$= e^{\frac{+0.7132}{18}}$

$= e^{0.03962}$

$= 1.040415$ ans.

b)

Input: p = -1, t=1

Weight and Biases:

$$w^1(0) = -1, b^1(0) = 1$$
$$w^2(0) = -2, b^2(0) = 1$$

Learning rate (α) = 1

Activation Function: Tan Sigmoid / Bipolar Activation Function with Slope Param 2

$$f(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$

Forward Calculation:

Hidden Layer Calculation:

$$n^1 = w^1 p + b^1 = (-1)(-1)+1=2$$

$$a^1 = f(n^1) = \frac{1 - e^{-2*2}}{1 + e^{-2*2}} \approx 0.964$$

Output Layer Calculation:

$$n^2 = w^2 a^1 + b^2 = (-2)(0.964)+1= -0.928$$
$$a^2 = f(n^2) = \frac{1 - e^{-2*-0.928}}{1 + e^{-2*-0.928}} \approx -0.730$$

Error Calculation:

$$e = t - a^2 = 1 - (-0.730) = 1.730$$

Backward Pass:

Output Layer Local Gradient

The derivative of f(x) is $f'(x) = [1 - f(x)][1 + f(x)]$

$$\delta^2 = (t - a^2) f'(n^2) = 1.730 * [1 - (-0.928)] * [1 + (-0.928)] = 0.240$$

Hidden Layer Local Gradient

$$\delta^1 = \delta^2 w^2 f'(n^1) = 0.240 * -2 * [1 - (2)] * [1 + (2)] = 1.44$$

Update Weights and Biases:

$$w^2(1) = -2 + \alpha * \delta^2 * a^1 = -2 + 1 * 0.240 * 0.964 = -1.76$$
$$b^2(1) = 1 + \alpha * \delta^2 = 1 + 1 * 0.240 = 1.24$$
$$w^1(1) = -1 + \alpha * \delta^1 * p = -1 + 1 * 1.44 * -1 = -2.44$$
$$b^1(1) = 1 + \alpha * \delta^1 = 1 + 1 * 1.44 = 2.44$$

Q3.

a)

Q3 (a): P-1

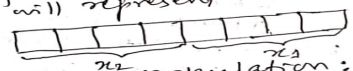
Apply Genetic Algorithm for the following optimization problem (Go for at least two generations):

Maximize $f(x_1, x_2) = 15x_2 + 10x_1^2$ where x_1 and x_2 are the integers defined in the range $0 \le (x_1, x_2) \le 15$

Ans:-

Step 1 :- Encoding of chromosome or parent.:

- We shall go for binary encoding of chromosome.
- Since (x_1, x_2) is the solution variable and since range of x_1 as well as x_2 varies from 0 to 15, each variable can be encoded using 4 bits.
- So we shall have a chromosome of 8 bit one where first 4 bits will represent x_2 and next 4 bits will represent x_1.



Step 2 :- Initial population:

We shall randomly select four chromosomes or parents as the members of initial population.

P1	1 0 1 1 0 1 1 1
P2	0 1 0 1 1 0 0 1
P3	1 0 1 0 1 0 1 0
P4	1 1 0 0 0 1 1 0

Step 3 : Fitness calculation:

In this problem, the fitness function can be chosen as $f(x_1, x_2) = 15x_2 + 10x_1^2$

i) Fitness value of parent P1 :-

$x_2 = 1011 = 11$ and $x_1 = 0111 = 7$

Fitness value $= 15 \times 11 + 10 \times 7^2 = 165 + 490 = \underline{655}$

ii) Fitness value of parent P2 :-

$x_2 = 0101 = 5$ and $x_1 = 1001 = 9$

Fitness value $= 15 \times 5 + 10 \times 9^2 = 75 + 810 = \underline{885}$ (P50)

P-2

iii) Fitness value for Parent P3:

$x_2 = 1010 = 10$ and $x_1 = 1010 = 10$

Fitness value $= 15 \times 10 + 10 \times 10^2 = 150 + 1000 = \underline{1150}$

iv) Fitness value for Parent P4:

$x_2 = 1000 = 8$ and $x_1 = 1100 = 12$

Fitness value $= 15 \times 8 + 10 \times 12^2 = 120 + 1440 = \underline{1560}$

Step 4 :- Selection of parents on the basis of fitness contribution in the population pool:

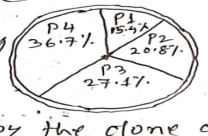
Parent #	Binary chromosome	(x_2, x_1)	Fitness value	% contribution of fitness	REMARK
P1	1011 0111	(11, 7)	655	15.4%	×
P2	0101 1001	(5, 9)	885	20.8%	✓
P3	1010 1010	(10, 10)	1150	27.1%	✓
P4	1000 1100	(8, 12)	1560	36.7%	✓

TOTAL: 4250 → 100%

MAX: 15.60 → 36.7%

MEAN: 1062.5 → 25%

- We shall use Biased Roulette wheel selection approach.
- It is clear from this approach that Parent 1 (P1) is contributing lowest fitness compared to other. So we can reject P1 from the population pool and instead we can replace it by the clone of Parent 3 (P3) so that 4 parents remain the pool for forthcoming genetic operations.



Selected parents for 1st generation Processing:-

P'1	1 0 1 1 0 1 1 0
P'2	0 1 0 1 1 0 0 1
P'3	1 0 1 0 1 0 1 0
P'4	1 0 0 0 1 1 0 0

1st Generation Processing

Step 5 : Crossover :

Parent #	Binary chromosome	(x_2, x_1)	Fitness value	% contribution	Crossover	Offsprings	
P_1'	10101010	(10, 10)	1150	24.2%	1010 | 1010	10111001	C_1
P_2'	01001001	(5, 9)	885	18.6%	0100 | 1001	01001010	C_2
P_3'	10101010	(10, 10)	1150	24.2%	1010 | 1010	10101100	C_3
P_4'	10001100	(8, 12)	1566	33.0%	1000 | 1100	10001010	C_4

TOTAL: 4745 → 100%
MAX: 1560 → 33%
MEAN: 1186.25 → 25%

Let us choose, crossover point 3 (i.e CP3) for parents P_1' & P_2' and crossover point 5 (i.e CP5) for parents P_3' and P_4'

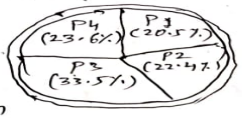
Step 6 : selection of next generation of parents on the basis of fitness contribution

- Children generated above due to crossover operation become the parents of next generation.

Children #	Parent #	Binary chromosome	(x_2, x_1)	Fitness value	% contribution	Remark
C1	P1	10111001	(11, 9)	975	20.5%	✓
C2	P2	01001010	(4, 10)	1060	22.4%	✓
C3	P3	10101100	(10, 12)	1590	33.5%	✓
C4	P4	10001010	(8, 10)	1120	23.6%	✓

TOTAL: 4745 → 100%
MAX: 1590 → 33.5%
MEAN: 1186.25 → 25%

- We shall use Biased Roulette wheel selection approach.
- It is clear from this approach that all the four parents can be selected without any rejection. These four parents will undergo 2nd generation genetic operation.



2nd Generation Processing

Step 7 : Crossover :

- Let us choose crossover point 4 (i.e CP4) for all the two pairs of parents.

Parents for 2nd Generation

P_1^2	10111001
P_2^2	01001010
P_3^2	10101100
P_4^2	10001010

Parent #	Binary chromosome	(x_2, x_1)	Fitness value	% contribution of fitness	Crossover	Offsprings	
P_1^2	10111001	(11, 9)	975	20.5%	1011 | 1001	10111010	C_1
P_2^2	01001010	(4, 10)	1060	22.4%	0100 | 1010	01001001	C_2
P_3^2	10101100	(10, 12)	1590	33.5%	1010 | 1100	10101010	C_3
P_4^2	10001010	(8, 10)	1120	23.6%	1000 | 1010	10001100	C_4

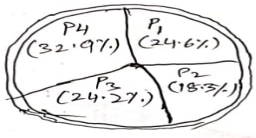
Step 8 : selection of next generation of parents on the basis of fitness contribution :

Children #	Parent #	Binary chromosome	(x_2, x_1)	Fitness value	% contribution of fitness	Remark
C1	P1	10111010	(11, 10)	1165	24.6%	✓
C2	P2	01001001	(4, 9)	870	18.3%	✓
C3	P3	10101010	(10, 10)	1150	24.2%	✓
C4	P4	10001100	(8, 12)	1560	32.9%	✓

TOTAL: 4745 → 100%
MAX: 1560 → 32.9%
MEAN: 1186.25 → 25%

~~Step 9 : selection of next generation of parents on the basis of fitness contribution~~ :

- We shall use Biased Roulette Wheel selection approach
- It is clear from this approach that all the four parents can be selected without any rejection. These four parents will undergo 3rd generation genetic operation.



$P_1^3 = 10111010$ $P_2^3 = 10101010$
$P_3^3 = 01001001$ $P_4^3 = 10001100$

- As per question, we have done for at least 2 generations. So we can stop here.

b)
(i) Single rule with single antecedent

Single Rule with Single Antecedent

$$\begin{aligned}\mu_{B'}(y) &= [\vee_x(\mu_{A'}(x) \wedge \mu_A(x)] \wedge \mu_B(y) \\ &= w \wedge \mu_B(y).\end{aligned}$$

In other words, first we find the degree of match w as the maximum of $\mu_{A'}(x) \wedge \mu_A(x)$ then the MF of the resulting B' is equal to the MF of B clipped by w
Intuitively, w represents a measure of degree of belief for the antecedent part of a rule; this measure gets propagated by the if-then rules and the resulting degree of belief or MF for the consequent part (B') should be no greater than w.

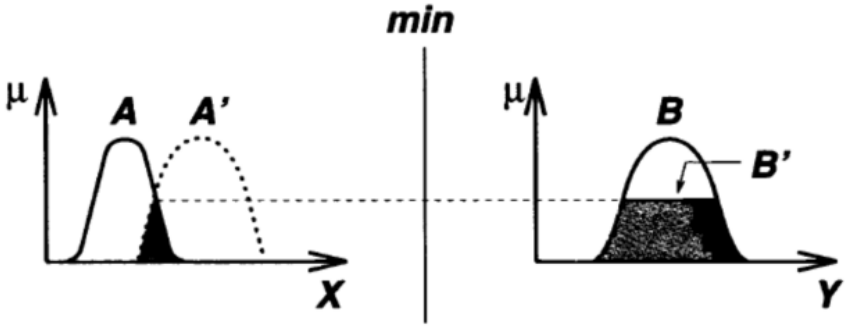


Figure 3.12. *Graphic interpretation of GMP using Mamdani's fuzzy implication and the max-min composition.*

(ii) Single rule with multiple antecedents

Single Rule with Multiple Antecedents

A fuzzy if-then rule with two antecedents is usually written as "if x is A and y is B then z is C." The corresponding problem for GMP is expressed as

premise 1 (fact):	x is A' and y is B',
premise 2 (rule):	if x is A and y is B then z is C,
consequence (conclusion):	z is C'.

The fuzzy rule in premise 2 can be put into the simpler form "$A \times B \rightarrow C$." Intuitively, this fuzzy rule can be transformed into a ternary fuzzy relation R_m

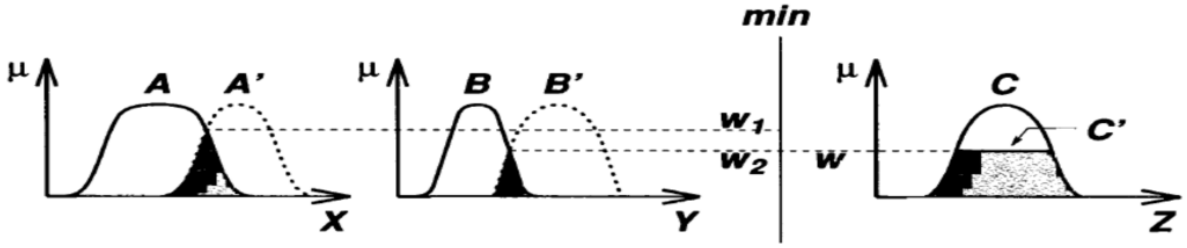


Figure 3.13. *Approximate reasoning for multiple antecedents.*

based on Mamdani's fuzzy implication function, as follows:

$$R_m(A,B,C) = (A \times B) \times C = \int_{X\times Y\times Z} \mu_A(x) \wedge \mu_B(y) \wedge \mu_C(z)/(x,y,z).$$

The resulting C' is expressed as

$$C' = (A' \times B') \circ (A \times B \to C).$$

Thus

$$\begin{aligned} \mu_{C'}(z) &= \vee_{x,y}[\mu_{A'}(x) \wedge \mu_{B'}(y)] \wedge [\mu_A(x) \wedge \mu_B(y) \wedge \mu_C(z)] \\ &= \vee_{x,y}\{[\mu_{A'}(x) \wedge \mu_{B'}(y) \wedge \mu_A(x) \wedge \mu_B(y)]\} \wedge \mu_C(z) \\ &= \{\underbrace{\vee_x[\mu_{A'}(x) \wedge \mu_A(x)]}_{w_1}\} \wedge \{\underbrace{\vee_y[\mu_{B'}(y) \wedge \mu_B(y)]}_{w_2}\} \wedge \mu_C(z) \\ &= \underbrace{(w_1 \wedge w_2)}_{\substack{\text{firing} \\ \text{strength}}} \wedge \mu_C(z), \end{aligned} \tag{3.25}$$

where w_1 and w_2 are the maxima of the MFs of $A \cap A'$ and $B \cap B'$, respectively.

(iii) Multiple rules with multiple antecedents

Multiple Rules with Multiple Antecedents

The interpretation of multiple rules is usually taken as the union of the fuzzy relations corresponding to the fuzzy rules. Therefore, for a GMP problem written as

premise 1 (fact):	x is A' and y is B',
premise 2 (rule 1):	if x is A_1 and y is B_1 then z is C_1,
premise 3 (rule 2):	if x is A_2 and y is B_2 then z is C_2,
consequence (conclusion):	z is C',

we can employ the fuzzy reasoning shown in Figure 3.14 as an inference procedure to derive the resulting output fuzzy set C'.

To verify this inference procedure, let $R_1 = A_1 \times B_1 \to C_1$ and $R_2 = A_2 \times B_2 \to C_2$. Since the max-min composition operator $\circ$ is distributive over the $\cup$ operator, it follows that

$$\begin{aligned} C' &= (A' \times B') \circ (R_1 \cup R_2) \\ &= [(A' \times B') \circ R_1] \cup [(A' \times B') \circ R_2] \\ &= C'_1 \cup C'_2, \end{aligned} \tag{3.28}$$

where C'_1 and C'_2 are the inferred fuzzy sets for rules 1 and 2, respectively. Figure 3.14 shows graphically the operation of fuzzy reasoning for multiple rules with multiple antecedents.

Q4.

a)

XOR function has the following truth table:

Input (x1, x2)	Output (y)
(0, 0)	0
(0, 1)	1
(1, 0)	1
(1, 1)	0

XOR is non-linearly separable, meaning a simple perceptron cannot model it. RBF networks, with their ability to handle non-linearity, can model XOR effectively.

1. **Problem Setup - XOR Truth Table**
    1. Inputs (x_1, x_2) → Output (y)
    2. (0,0) → 0
    3. (0,1) → 1
    4. (1,0) → 1
    5. (1,1) → 0
2. **RBF Network Structure**
    1. Input layer: 2 neurons (x_1, x_2)
    2. Hidden layer: 2 RBF neurons (minimum needed for XOR)
    3. Output layer: 1 neuron with bias
    4. The output will be: $y = w_1\varphi_1(x) + w_2\varphi_2(x) + b$ where φ_i are the RBF functions and w_i are weights

    RBF Function Definition

    Using Gaussian RBF: $\varphi_i(x) = \exp(-||x-c_i||^2/2\sigma^2)$

    where:
    1. x is the input vector (x_1, x_2)
    2. c_i is the center vector for the i-th RBF neuron
    3. σ is the width parameter
    4. $||\cdot||$ denotes Euclidean distance

    Center Selection
    1. Need centers that can separate the XOR patterns
    2. Optimal placement:
        1. $c_1 = (0.25, 0.25)$ - closer to (0,0)
        2. $c_2 = (0.75, 0.75)$ - closer to (1,1)
    3. This placement creates higher activation for points closer to centers

    Width Parameter Selection

    Choose $\sigma \approx 0.5$

    This ensures:
    1. Sufficient overlap between RBFs
    2. Good coverage of input space
    3. Not too narrow (overfitting)
    4. Not too wide (underfitting)
3. **Mathematical Formulation**

    For each input pattern x: $\varphi_1(x) = \exp(-||x-c_1||^2/2\sigma^2)$ $\varphi_2(x) = \exp(-||x-c_2||^2/2\sigma^2)$ $y = w_1\varphi_1(x) + w_2\varphi_2(x) + b$
4. **System of Equations** For the four XOR patterns:
    1. $w_1\varphi_1(0,0) + w_2\varphi_2(0,0) + b = 0$
    2. $w_1\varphi_1(0,1) + w_2\varphi_2(0,1) + b = 1$
    3. $w_1\varphi_1(1,0) + w_2\varphi_2(1,0) + b = 1$
    4. $w_1\varphi_1(1,1) + w_2\varphi_2(1,1) + b = 0$
5. **Solution for Weights**
    1. Write in matrix form: $\Phi w = y$
    2. where Φ is the matrix of RBF outputs plus bias column
    3. $w = [w_1, w_2, b]^T$
    4. $y = [0, 1, 1, 0]^T$
    5. Solve using pseudo-inverse: $w = (\Phi^T\Phi)^{-1}\Phi^T y$
6. **Decision Boundary**

1. The decision boundary occurs where network output = 0.5
2. This creates a complex boundary that separates:
    1. (0,0) and (1,1) [output ≈ 0]
    2. (0,1) and (1,0) [output ≈ 1]

b)

Q4.b)

(i) Support

The **support** of a fuzzy set is the set of all elements in the universe of discourse where the membership function has a value greater than zero.

If A is a fuzzy set with membership function $\mu_A(x)$, then the **support** of A is:

$\text{Support}(A)=\{x \in X \mid \mu_A(x)>0\}$

(ii) Core

The **core** of a fuzzy set is the set of all elements in the universe of discourse where the membership function is equal to 1.

If A is a fuzzy set with membership function $\mu_A(x)$, then the **Core** of A is:

$\text{Core}(A)=\{x \in X \mid \mu_A(x)=1\}$

(iii) Normality

A fuzzy set is **normal** if its membership function reaches the maximum value of 1 for at least one element in the universe of discourse.

- **Condition for Normality:**
  A fuzzy set A is **normal** if: $\exists\ x \in X$ such that $\mu_A(x)=1$.

(iv) Convexity

A fuzzy set is **convex** if, for any two elements in the universe of discourse, the membership function of every point on the straight line connecting them is greater than or equal to the minimum membership value of the two points.

- **Mathematical Expression:**
  A is convex if:
  $\mu_A(\lambda_{x1}+(1-\lambda)_{x2}) \geq \min(\mu_A(x_1),\mu_A(x_2))$,for all $x_1,x_2 \in X$,$\lambda \in [0,1]$.

(v) Bandwidth – Refers to the distance between two crossover points.

Q5.

a)

(i) Triangular MF –

A triangular MF is specified by three parameters $\{a, b, c\}$ as follows:

$$\text{triangle}(x; a, b, c) = \begin{cases} 0, & x \leq a. \\ \frac{x-a}{b-a}, & a \leq x \leq b. \\ \frac{c-x}{c-b}, & b \leq x \leq c. \\ 0, & c \leq x. \end{cases}$$

(ii) Trapezoidal MF –

A **trapezoidal MF** is specified by four parameters $\{a, b, c, d\}$ as follows:

$$\text{trapezoid}(x; a, b, c, d) = \begin{cases} 0, & x \le a. \\ \frac{x-a}{b-a}, & a \le x \le b. \\ 1, & b \le x \le c. \\ \frac{d-x}{d-c}, & c \le x \le d. \\ 0, & d \le x. \end{cases}$$

(iii) Gaussian MF –

A **Gaussian MF** is specified by two parameters $\{c, \sigma\}$:

$$\text{gaussian}(x; c, \sigma) = e^{-\frac{1}{2}\left(\frac{x-c}{\sigma}\right)^2}.$$

(iv) Generalized Bell MF –

A **generalized bell MF** (or **bell MF**) is specified by three parameters $\{a, b, c\}$:

$$\text{bell}(x; a, b, c) = \frac{1}{1 + \left|\frac{x-c}{a}\right|^{2b}}, \tag{2.23}$$

(v) Sigmoid MF –

A **sigmoidal MF** is defined by

$$\text{sig}(x; a, c) = \frac{1}{1 + \exp[-a(x-c)]},$$

where a controls the slope at the crossover point $x = c$.

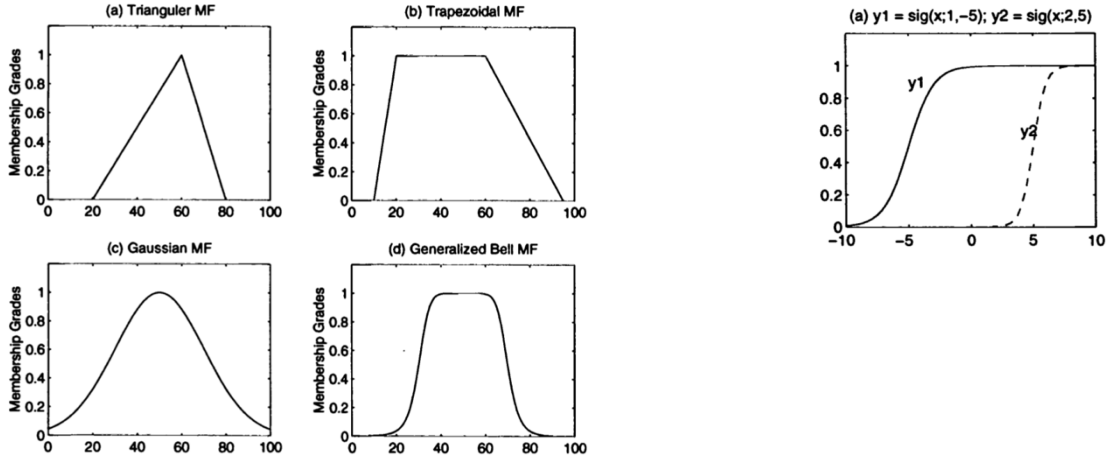


Figure 2.7. *Examples of four classes of parameterized MFs: (a) triangle*$(x; 20, 60, 80)$*; (b) trapezoid*$(x; 10, 20, 60, 95)$*; (c) gaussian*$(x; 50, 20)$*; (d) bell*$(x; 20, 4, 50)$. (MATLAB file: `disp_mf.m`)

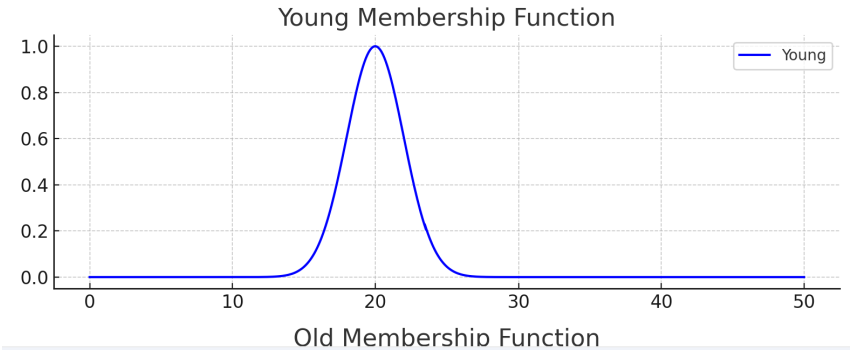
b)

(i) Bell Membership functions for "young" and "old" age as follows:

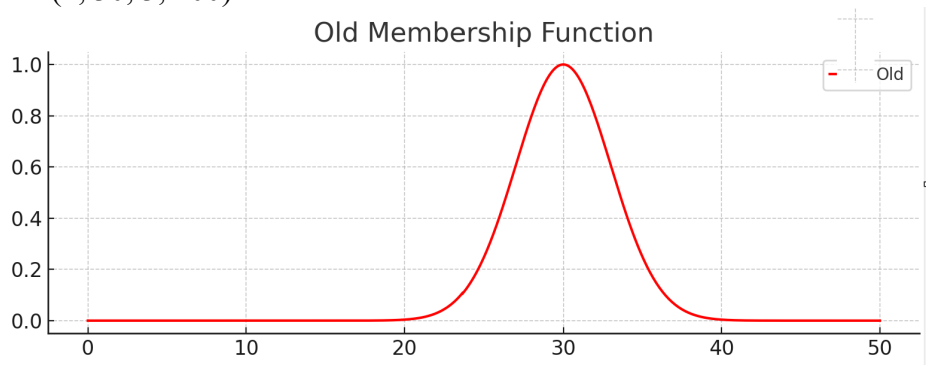
Young Age bell Membership Function:

bell(x;20,2,1)

here center (c) = 20, width (w) = 2, and height (h) = 1



(ii) Old Age Membership Function:
bell(x, 30, 3, 100)



Composite Linguistic Terms

(i) Not Very Young but More or Less Young

- 'Not Very Young' is obtained by negating the membership function for 'young':

$$\mu_{\text{not A}}(x) = 1 - \mu_A(x)$$

- 'More or Less Young' is represented by a wider bell-shaped function:

$$\mu_{\text{not very young AND not very old}}(x) = \min(\mu_{\text{not very young}}(x), \mu_{\text{not very old}}(x))$$

(ii) Old but Not Too Old

- "Old": Use the original "old" membership function:

$$\mu_{\text{old}}(x) = \text{bell}(x; 30, 3, 100)$$

- “Not very Old”:

$$\mu_{\text{not very old}}(x) = 1 - \mu_{\text{old}}(x)$$

- Combine "old" and "not too old" using a fuzzy intersection (minimum):

$$\mu_{\text{old but not too old}}(x) = \min(\mu_{\text{old}}(x), \mu_{\text{not too old}}(x))$$

(iii) Not Very Young and Not Very Old

- "Not Very Young":

$$\mu_{\text{not very young}}(x) = 1 - \mu_{\text{young}}(x)$$

- "Not Very Old":

$$\mu_{\text{not very old}}(x) = 1 - \mu_{\text{old}}(x)$$

- Combine using fuzzy intersection (minimum):

$$\mu_{\text{not very young and not very old}}(x) = \min(\mu_{\text{not very young}}(x), \mu_{\text{not very old}}(x))$$

Q6.

a)

To implement an OR gate using an Adaptive Linear Neuron (Adaline), we need to follow these steps:

Define the OR gate truth table: The OR gate has the following truth table:

Input 1	Input 2	Output
0	0	0
0	1	1
1	0	1
1	1	1

Initialize weights, bias, and learning rate:

1. Weights w1=0.2 , w2=0.3
2. Bias b=0.1
3. Learning rate (η) should be chosen, e.g., η=0.1

- **Iterate through the training patterns and perform the following steps for each design −** Compute the weighted sum of inputs using the current weights.
- Apply the linear activation function to the weighted sum.
- Calculate the error between the predicted output and the desired output.
- Update the weights using the delta rule, which involves multiplying the input values by the error and adjusting the weights accordingly.
- Repeat these steps for a certain number of iterations or until the network converges.

Here are the computations for one epoch of training the Adaline for the OR gate:

Input x1,x2x_1, x_2	Target tt	Net Input zz	Error t−zt-z	Updated Weights [w1,w2][w_1, w_2]	Updated Bias bb
[0,0][0, 0]	0	0.1	-0.1	[0.2, 0.3]	0.09
[0,1][0, 1]	1	0.39	0.61	[0.2, 0.361]	0.151
[1,0][1, 0]	1	0.351	0.649	[0.2649, 0.361]	0.2159
[1,1][1, 1]	1	0.8418	0.1582	[0.28072, 0.37682]	0.2317

After one epoch:

- **Final Weights:** [0.28072,0.37682][0.28072, 0.37682]
- **Final Bias:** 0.23170.2317

The input like following is also correct but same algorithm must be followed.

Input 1	Input 2	Output
-1	-1	-1
-1	1	1
1	-1	1
1	1	1

b)

6.(b)(i) $\bar{A} = \frac{0.1}{x_1} + \frac{0.3}{x_2} + \frac{0.6}{x_3} + \frac{0.8}{x_4} + \frac{1}{x_5} + \frac{0.7}{x_6} + \frac{0.4}{x_7} + \frac{0.2}{x_8}$,

Sugeno's complement $N_s(a) = \frac{1-a}{1+sa}$, and $s = 2$.

So, $N_2(0.1) = \frac{1-0.1}{1+2\times 0.1} = 0.75$, $N_2(0.3) = \frac{1-0.3}{1+2\times 0.3} = 0.438$,

$N_2(0.6) = \frac{1-0.6}{1+2\times 0.6} = 0.182$, $N_2(0.8) = \frac{1-0.8}{1+2\times 0.8} = 0.077$,

$N_2(1) = \frac{1-1}{1+2\times 1} = 0$, $N_2(0.7) = \frac{1-0.7}{1+2\times 0.7} = 0.125$,

$N_2(0.4) = \frac{1-0.4}{1+2\times 0.4} = 0.333$, $N_2(0.2) = \frac{1-0.2}{1+2\times 0.2} = 0.571$.

$\therefore \bar{A}_S^e = \frac{0.75}{x_1} + \frac{0.438}{x_2} + \frac{0.182}{x_3} + \frac{0.077}{x_4} + \frac{0}{x_5} + \frac{0.125}{x_6} + \frac{0.333}{x_7} + \frac{0.571}{x_8}$.

(ii) Yager's complement $N_w(a) = (1-a^w)^{1/w}$ where $w > 0$ ——— (1).

Now, $N_w(N_w(a))$

$= \{1 - [N_w(a)]^w\}^{1/w}$ [using equn (1)]

$= \{1 - [(1-a^w)^{1/w}]^w\}^{1/w}$ [using equn (1)]

$= (1 - 1 + a^w)^{1/w}$

$= a$

Thus, Yager's complement operator satisfies Involution property.

(Sugeno's complement – 2 marks, Yager's complement proofs – 3 marks)

Q7.

a)

Max-min composition

max(0.6, 0.1, 0.3)	max(0.2, 0.5, 0.3)	max(0.4, 0.5, 0.3)
max(0.5, 0.1, 0.4)	max(0.2, 0.6, 0.7)	Max(0.4, 0.6, 0.5)

0.6	0.5	0.5
0.5	0.7	0.6

(2 marks)

Max-product composition

max(0.42, 0.05, 0.12)	max(0.12, 0.30, 0.30)	max(0.24, 0.30, 0.15)
max(0.35, 0.06, 0.28)	max(0.10, 0.36, 0.70)	Max(0.20, 0.36, 0.35)

0.42	0.30	0.30
0.35	0.70	0.36

(3 marks)

b)

The purpose of the Particle Swarm Optimization (PSO) algorithm is to find the optimal solution to a given problem by simulating the social behavior of a group of birds, fishes or insects, known as a swarm, searching for food. PSO is a powerful optimization technique. Some common applications of PSO include optimizing the design of structures, machines, and systems in Engineering design, tuning the parameters in Control systems, portfolio optimization and risk management in Finance, path planning and motion control in Robotics, and many more. Overall, PSO is a valuable tool for solving complex optimization problems, and its popularity continues to grow due to its simplicity, efficiency, and versatility.

The Particle Swarm Optimization (PSO) algorithm relies on two primary equations to update the position and velocity of each particle after identifying the best values:

- Velocity Update Equation:

$$v_{ik}(t+1) = w * v_{ik}(t) + c1 * r1 * (pbest_{ik} - x_{ik}(t)) + c2 * r2 * (gbest_k - x_{ik}(t))$$

- Position Update Equation:

$$x_{ik}(t+1) = x_{ik}(t) + v_{ik}(t+1)$$

The meaning of each terms used in the above equations are as follows:

$v_{ik}(t)$, $v_{ik}(t+1)$: Velocity of the i-th particle at time step t and t+1, respectively

w: Inertia weight, the parameter that controls the impact of previous velocity

c1: Cognitive coefficient, an acceleration coefficients that tunes the influence of personal best in the velocity update

c2: Social coefficient, an acceleration coefficients that tunes the influence of global best in the velocity update

r1, r2: Random numbers between 0 and 1

$pbest_{ik}$: Personal best position of the i-th particle.

$gbest_k$: Global best position found by the swarm

$x_{ik}(t)$, $x_{ik}(t+1)$: Position of the i-th particle at time step t and t+1, respectively

The equations mentioned above govern the movement of each particle in the swarm. The particles adjust their velocities based on their personal best experiences (pbest) and the best experience of the entire swarm (gbest). This collective behavior guides the swarm towards optimal solutions.

The steps involved in the Particle Swarm Optimization (PSO) algorithm are as listed below:

Initialization:

1. Set parameters: Define the algorithm parameters, such as the number of particles, maximum number of iterations, inertia weight (w), cognitive coefficient (c1), and social coefficient (c2).
2. Initialize the swarm: Create a population of particles, each with a random position and velocity within the search space.

Iteration:

1. Evaluate fitness: Calculate the fitness value of each particle based on the objective function.
2. Update personal best: If the current fitness of a particle is better than its previous best, update its personal best position (pbest).
3. Update global best: Determine the particle with the best fitness in the entire swarm and update the global best position (gbest).
4. Update velocity: Calculate the new velocity of each particle using the velocity update equation, considering its current velocity, personal best, and global best.
5. Update position: Update the position of each particle using the position update equation based on its current position and new velocity.

Termination:

1. Check termination criteria: If the maximum number of iterations is reached or a satisfactory solution is found, terminate the algorithm. Otherwise, go back to step 2 of iteration.

Output:

The final global best position represents the optimal solution found by the PSO algorithm.

(1 mark for purpose of PSO, 0.5 each for two equations of PSO, 1 mark for description of parameters of PSO equations, 2 marks for steps in PSO algorithm.)