

Hi, I'm Dr. Abhishek Thakur

I'm on a mission to help 100,000 students to achieve technical skills, to earn passive income using digital coaching.

Digital Marketer, Entrepreneur & A Mentor

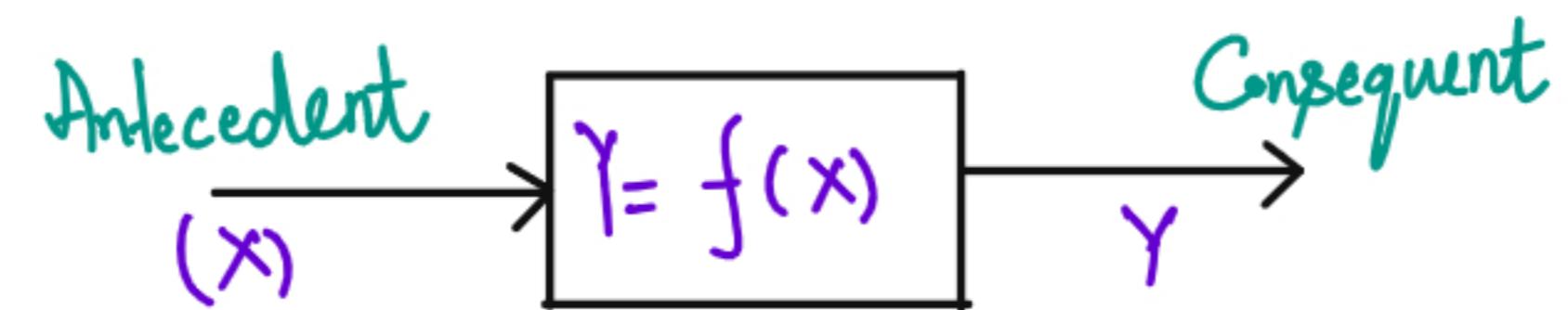


Thanks & Regard

Dr. Abhishek Thakur
Mobile: +91 9878388084
<https://atschool.in/>

https://www.youtube.com/channel/UC45IX_CQT_HE1TTq-us-PyA
<https://scholar.google.co.in/citations?hl=en&user=-Wjrcz4AAAAJ>
https://www.researchgate.net/profile/Abhishek_Thakur2
<https://www.linkedin.com/in/abhishek-thakur-00123321/>
<https://www.facebook.com/abhithakur25/>
<https://twitter.com/abhithakur25>

* Computing *



f : formal method | Algorithm | mapping function

- Precise Solution
- Unambiguous & Accurate
- Mathematical Model

* Hard Computing *

- * Precise Results
- * Control Actions
 - Unambiguous
 - formally defined
- * Numerical Problem
- * Searching & Sorting
- * Computational Geometry Problem

* Soft Computing *

- * Imprecision * Dynamic
- * Uncertainty * low Solution Cost
- * Do not require Mathematical Model



Hard Computing	Soft Computing
① Precision	① Imprecision
② Based on Binary Logic	② Based on Fuzzy Logic
③ No Approximation	③ Approximation
④ Exact Input Data	④ Noisy Data
⑤ Strictly Sequential	⑤ Allow parallel Computing
⑥ Certainty	⑥ Uncertainty

* FUZZY LOGIC *

Mathematical Language

fuzzy logic deals with Fuzzy Set

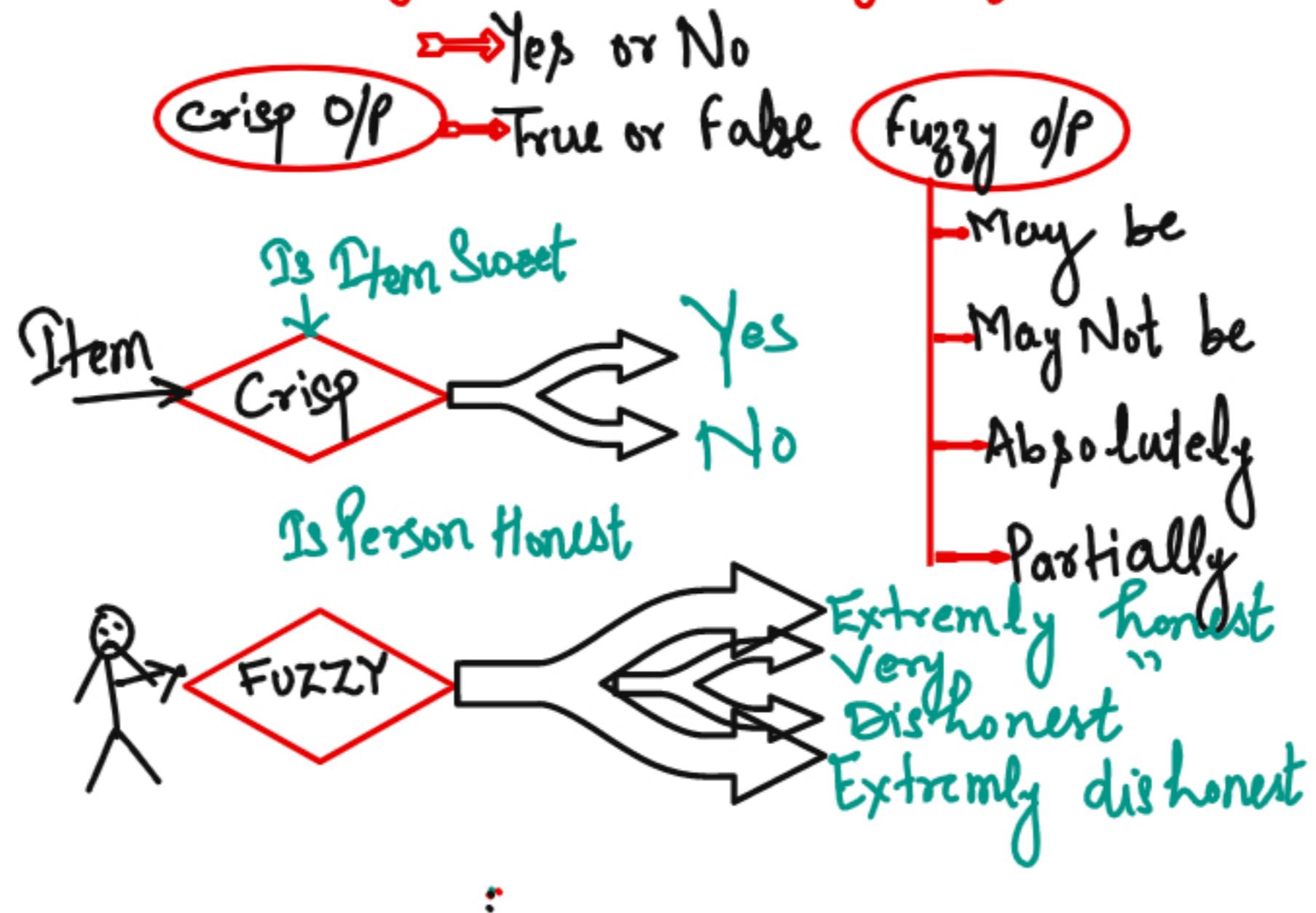
Relational
logic

Boolean
Logic

Predicate
logic

It deals with fuzzy Algebra

Crisp Logic Vs Fuzzy Logic

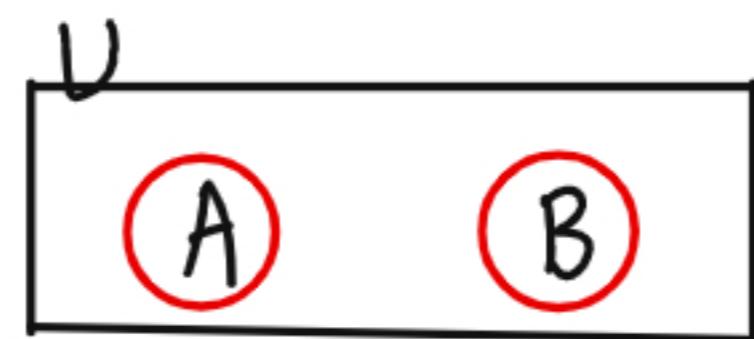


Crisp Set

U: All Students

A: In 10th Class

B: In 12th Class



* [Fuzzy Set] *

U: All Students

G: Good Students $G = \{G_i, \mu(G_i)\}$

S: Bad Students $\mu(\cdot)$ degree of Goodness

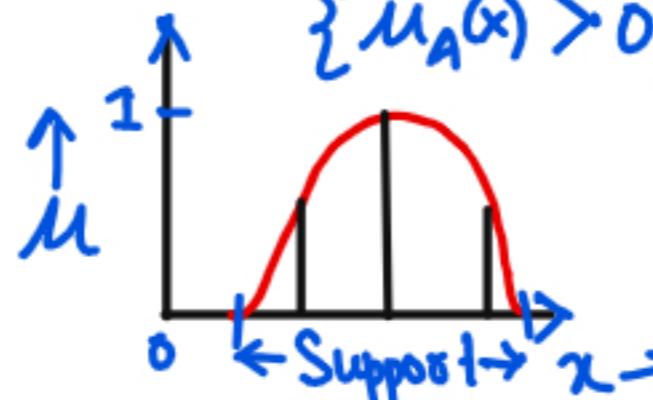
$G = \{(A, 0.9), (B, 0.7), (C, 0.1), (D, 0.3)\}$

$B = \{(A, 0.1), (B, 0.3), (C, 0.9), (D, 0.7)\}$

..

* Membership Function *

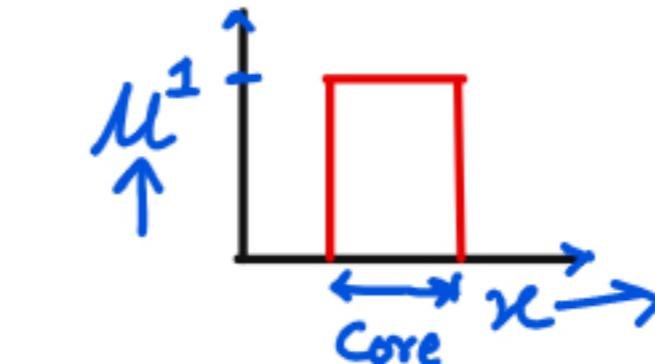
(I) Support :-
 Set of all points such that $\{\mu_A(x) > 0\}$



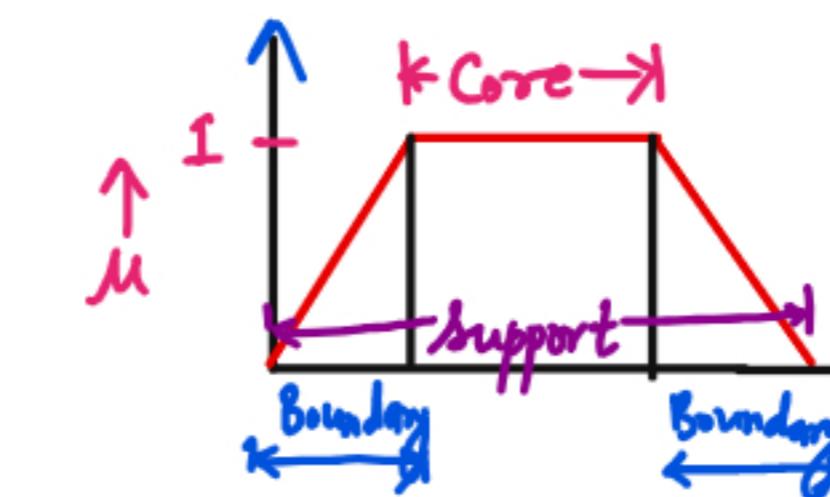
$$\text{Support}(A) = \{x \mid \mu_A(x) > 0\}$$

(II) Core :- $\mu_A(x) = 1$

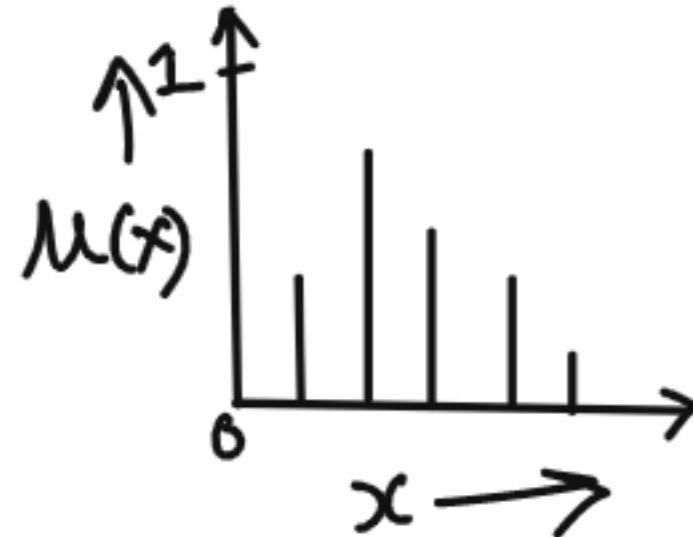
$$\text{Core}(A) = \{x \mid \mu_A(x) = 1\}$$



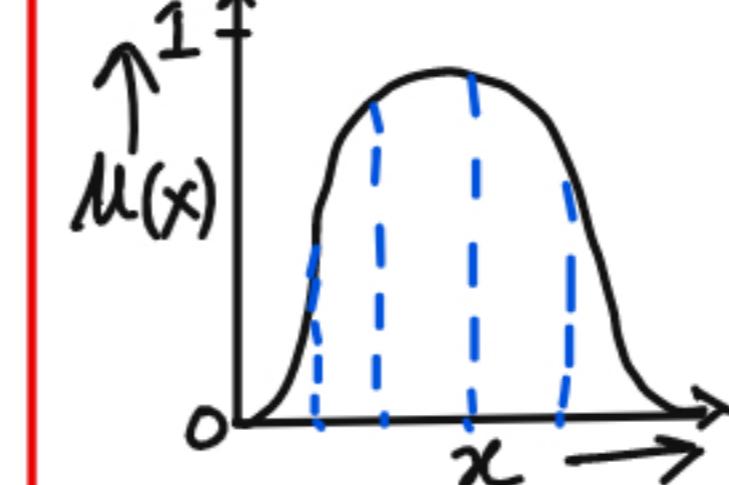
(III) Boundary ($1 > \mu_A(x) > 0$)



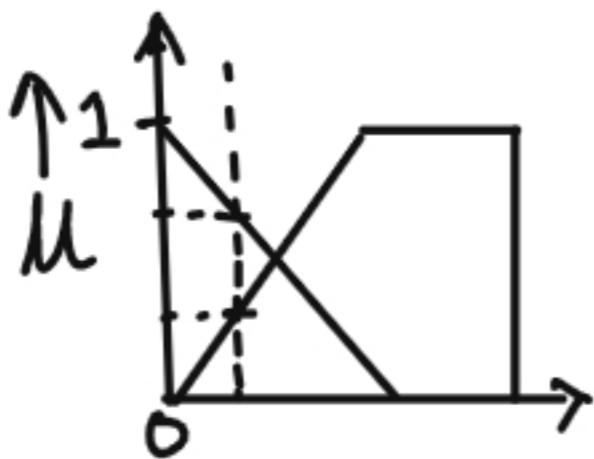
Discrete MF



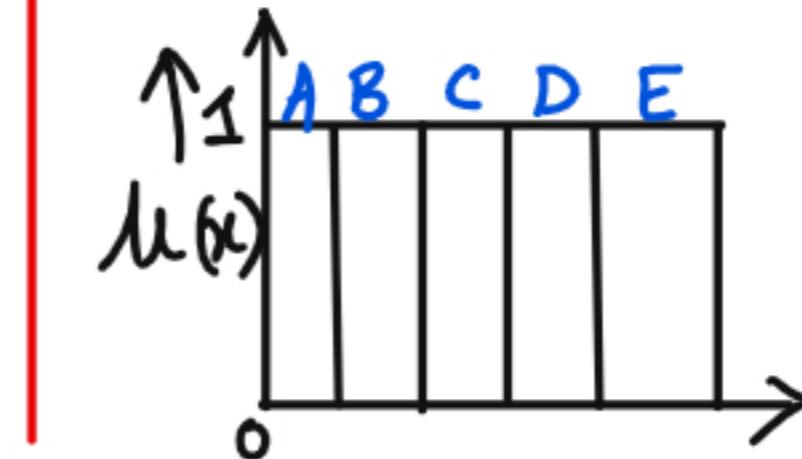
Continuous MF



Fuzzy Sets



Crisp Set



* Fuzzy Set Operation *

① UNION :- $(A \cup B)$

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

② Intersection :- $(A \cap B)$

$$\mu_{(A \cap B)}(x) = \min (\mu_A(x), \mu_B(x))$$

Example :- $A = \{(x_1, 0.6), (x_2, 0.7), (x_3, 0.4)\}$
 $B = \{(x_1, 0.3), (x_2, 0.2), (x_3, 0.5)\}$

$$(A \cup B) = \{(x_1,), (x_2,), (x_3,)\}$$

$$(A \cap B) = \{(x_1,), (x_2,), (x_3,)\}$$

3

③ Complement: A^c

$$\mu_{(A^c)}(x) = 1 - \mu_A(x)$$

Example :-

$$A = \{(x_1, 0.6), (x_2, 0.7), (x_3, 0.4)\}$$

$$A^c = \{(x_1,), (x_2,), (x_3,)\}$$

$$B = \{(x_1, 0.3), (x_2, 0.2), (x_3, 0.5)\}$$

$$B^c = \{(x_1,), (x_2,), (x_3,)\}$$

*

④ Vector Product : $(A \cdot B)$

$$\mu_{A \cdot B}(x) = \mu_A(x) \cdot \mu_B(x)$$

Example :

$$A = \{(x_1, 0.6), (x_2, 0.7), (x_3, 0.4)\}$$

$$B = \{(x_1, 0.3), (x_2, 0.2), (x_3, 0.5)\}$$

$$(A \cdot B) = \{(x_1, 0.18), (x_2,), (x_3,)\}$$

⑤ Cartesion Product: $(A \times B)$

$$\mu_{A \times B}(x, y) = \min(\mu_A(x), \mu_B(y))$$

Example:

$$A(X) = \{(x_1, 0.2), (x_2, 0.3), (x_3, 0.5)\}$$

$$B(Y) = \{(y_1, 0.8), (y_2, 0.6), (y_3, 0.3)\}$$

$$(A \times B) = \begin{array}{c|ccc} & y_1 & y_2 & y_3 \\ \hline x_1 & 0.2 & 0.2 & - \\ x_2 & - & - & - \\ x_3 & 0.3 & - & - \\ x_4 & - & - & - \\ \vdots & \vdots & \vdots & \vdots \end{array}$$

⑥ Scalar Product: $(\alpha \times A)$

$$\mu_{\alpha A}(x) = \alpha \times \mu_A(x)$$

⑦ Equality: $(A = B)$

$$\mu_A(x) = \mu_B(x)$$

⑧ Power: (A^α)

$$\mu_{A^\alpha}(x) = (\mu_A(x))^\alpha$$

.

⑨ Sum: $(A + B)$

$$\mu_{A+B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x) \cdot \mu_B(x)$$

⑩ Difference: $(A - B)$

$$\mu_{A-B}(x) = \mu_{A \cap B^c}(x)$$

⑪ Disjunctive Sum $(A \oplus B)$

$$\mu_{(A \oplus B)}(x) = (A^c \cap B) \cup (A \cap B^c)$$

:

Fuzzy Set Properties

① Commutative: $A \cap B = B \cap A$

$$A \cup B = B \cup A$$

② Associative: $A \cup (B \cup C) = (A \cup B) \cup C$

$$A \cap (B \cap C) = (A \cap B) \cap C$$

③ Distributive: $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$$

④ Idempotence:

$$A \cup A = A ; A \cap A = \emptyset$$

$$A \cup \emptyset = A ; A \cap \emptyset = \emptyset$$

⑤ Transitive:

If $A \subseteq B ; B \subseteq C$ then $A \subseteq C$

⑥ DeMorgan's Law:

$$(A \cap B)^c = A^c \cup B^c$$

$$(A \cup B)^c = A^c \cap B^c$$

Crisp Relation

$$A = \{1, 2, 3\} \quad B = \{4, 7, 8\}$$

$$A \times B = \{(1,4)(1,7)(1,8)(2,4)(2,7)(2,8)(3,4)(3,7)(3,8)\}$$

$$R = \{(1,4)(2,7)(3,8)\}$$

$$R = \begin{matrix} & \begin{matrix} 4 & 7 & 8 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} & \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{matrix}$$

$R_1 = \{(a,b) | a > b, (a,b) \in A \times B\}$
 $R_2 = \{(a,b) | a < b, (a,b) \in A \times B\}$

$R_2 = ?$

$$R_1 = \{(a,b) | a < b, (a,b) \in A \times B\}$$

$$R_1 = \{(1,4)(1,7)(1,8)(2,4)(2,7)(2,8)(3,7)(3,8)(3,4)\}$$

Operation on Crisp Relation

① Union : $(A \cup B)$ ② Intersection : $(A \cap B)$

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \quad B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A \cup B = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \quad A \cap B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Fuzzy Relation & Operation

$$A = \{(x_1, 0.6), (x_2, 0.2), (x_3, 0.3)\}$$

$$B = \{(y_1, 0.7), (y_2, 0.3), (y_3, 0.4)\}$$

$$\begin{aligned}\mu_R(x, y) &= \mu_{A \times B}(x, y) \\ &= \min [\mu_A(x), \mu_B(y)]\end{aligned}$$

$$R = A \times B = \begin{matrix} & y_1 & y_2 & y_3 \\ x_1 & 0.6 & 0.3 & 0.4 \\ x_2 & 0.2 & - & - \\ x_3 & 0.3 & - & - \end{matrix}$$

FUZZY IF THEN Rule

- Fuzzy Implication
- Fuzzy Rule
- Fuzzy Conditional Statement

If x is A then y is B

- x is A: Premise / antecedent
- y is B: Conclusion / consequence

Fuzzy Rule: R denoted as $R: A \rightarrow B$

Eg:- If Temp is High then Pressure is low

$$T_{HIGH} = \{(25, 0.1), (30, 0.2), (35, 0.3), (40, 0.6)\}$$

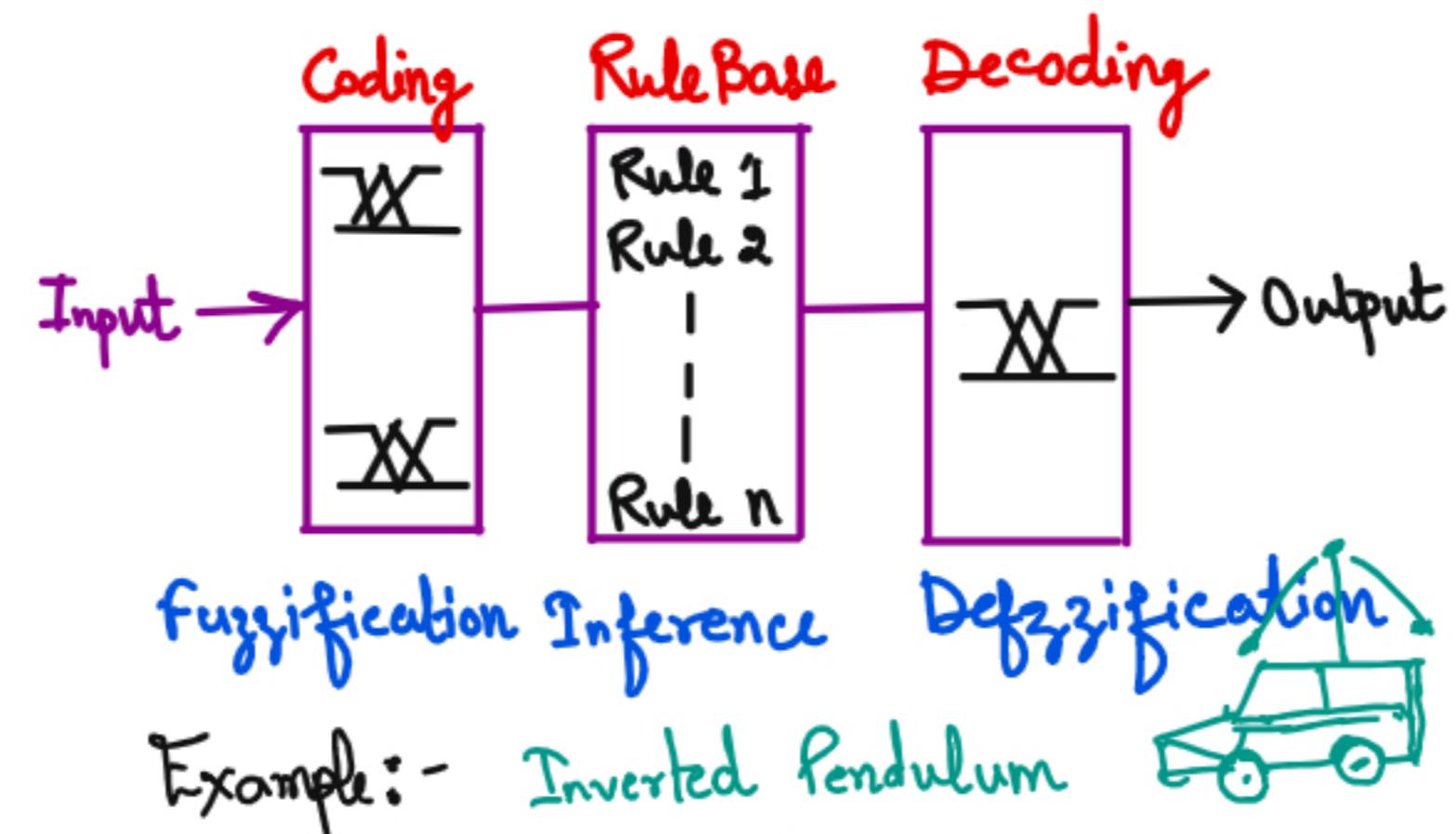
$$P_{LOW} = \{(2, 0.3), (5, 0.5), (6, 0.4)\}$$

If Temp is High then Pressure is Low

$$R: T_{HIGH} \rightarrow P_{LOW}$$

$$R = \begin{matrix} & \begin{matrix} 2 & 5 & 6 \end{matrix} \\ \begin{matrix} 25 \\ 30 \\ 35 \\ 40 \end{matrix} & \left[\begin{matrix} 0.1 & 0.1 & 0.1 \\ 0.2 & 0.2 & 0.2 \\ 0.3 & 0.5 & - \\ 0.3 & 0.5 & - \end{matrix} \right] \end{matrix}$$

R: A → B (Cartesian Product)
 AXB
 ↓
 min(μ_A, μ_B)



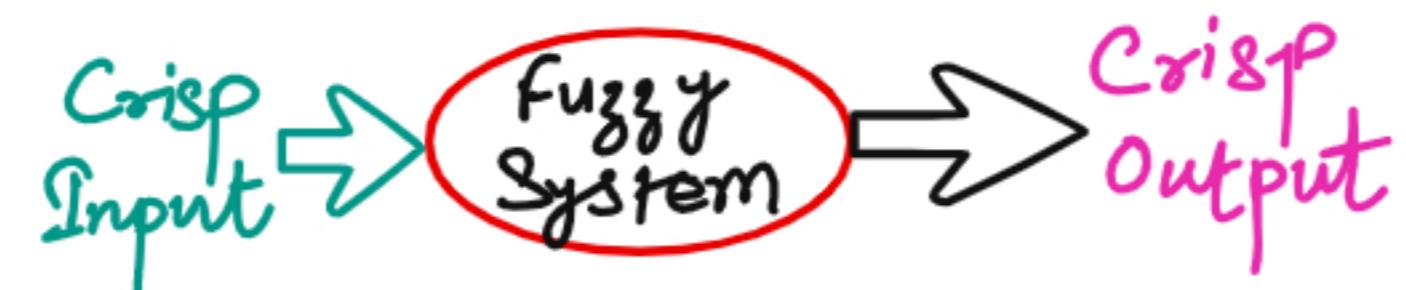
Input ① Error (difference)

② Angular Velocity

Output :- Current (mA)

Defuzzification

Fuzzy to Crisp conversion



1. Lambda Cut Method
2. Maxima methods
3. Weighted Sum Method
4. Centroid Methods

Lambda-cut method

Fuzzy Set $A \longrightarrow$ Crisp Set $A_\lambda (0 \leq \lambda \leq 1)$

$$A_\lambda = \{x \mid \mu_A(x) \geq \lambda\}$$

Example:

$$A = \{(x_1, 0.2), (x_2, 0.4), (x_3, 0.6)\}$$

$$\lambda = 0.3$$

$$A_{0.3} = \{(x_1, 0), (x_2, 1), (x_3, 1)\} = \{x_2, x_3\}$$

$$B = \{(y_1, 0.5), (y_2, 0.4), (y_3, 0.7)\}$$

$$B_{0.7}^{\lambda=0.7} = \{(y_1, 0), (y_2, 0), (y_3, 1)\} = \{y_3\}$$

Lamda-cut method (Fuzzy Relation)

$$R = \begin{bmatrix} 0.2 & 1 \\ 0.3 & 0.6 \end{bmatrix} \quad \lambda = 1, 0.5, 0.1, 0$$

$$R_1 = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \quad R_{0.5} = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix}$$

Perform Cartesion Product

$$R_{0.1} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \quad R_0 = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

R - fuzzy
Relation

R_> - Crisp
Relation

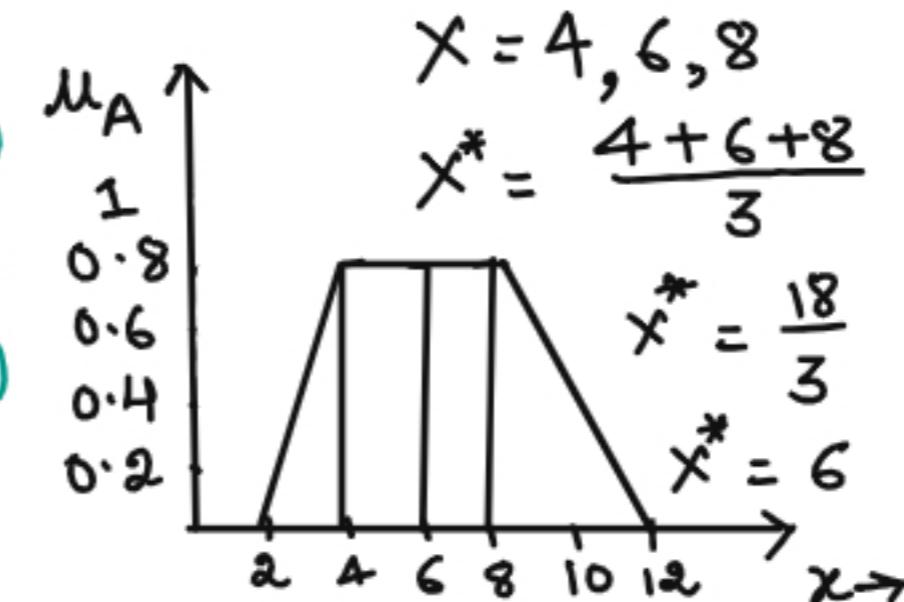
Maxima Method

1) First of Maxima (FOM $\Rightarrow x^* = 4$)

2) Last of Maxima (LOM $\Rightarrow x^* = 8$)

3) Mean of Maxima (MOM $\Rightarrow x^* = 6$)

$$x^* = \frac{\sum x_i \in M}{|M|} X_i$$



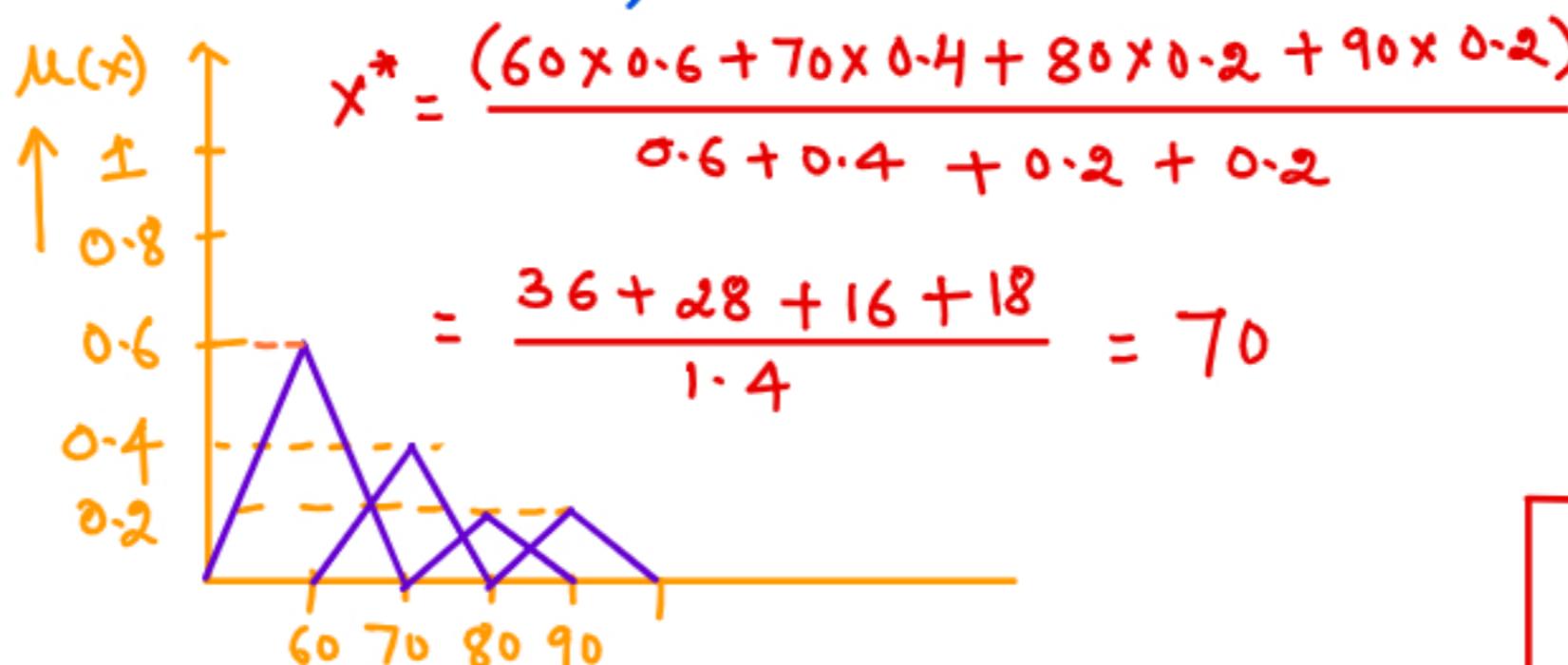
$M = \{x / \mu_A(x) = \text{height of fuzzy set}\}$

$|M| = \text{Cardinality of set } M$

$x^* = \text{Crisp Value}$

Weighted Average Method

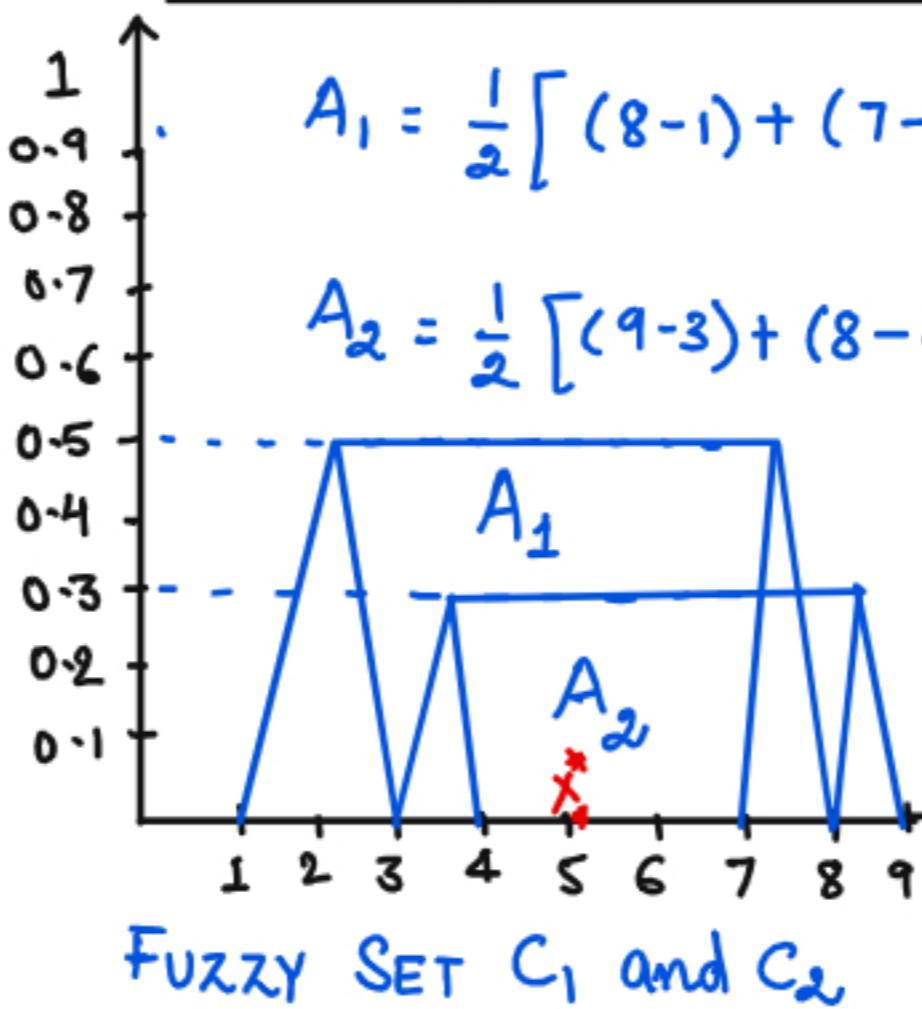
$$x^* = \frac{\sum \mu(x) \cdot x}{\sum \mu(x)}$$



$\begin{array}{r} 36 \\ 28 \\ 16 \\ 18 \\ \hline 98 \end{array}$	$\begin{array}{r} 2 \\ 14 \\ \hline 18 \end{array}$
------------------------------------------------------------------	-----------------------------------------------------

2

Center of Sum (cos)



$$x^* = \frac{\sum A x_c}{\sum A}$$

$$A_1 = \frac{1}{2} [(8-1) + (7-3)] \times 0.5 = \frac{1}{2} [7+4] \times 0.5 = \frac{5.5}{2} = 2.75$$

$$A_2 = \frac{1}{2} [(9-3) + (8-4)] \times 0.3 = \frac{1}{2} [6+4] \times 0.3 = \frac{3}{2} = 1.5$$

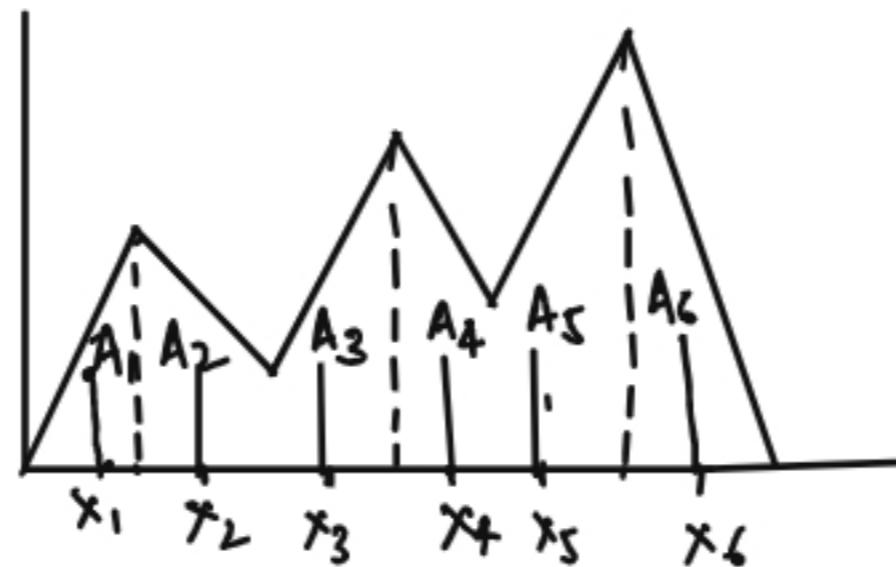
$$x^* = \frac{A_1 x_{c_1} + A_2 x_{c_2}}{A_1 + A_2}$$

$$= \frac{2.75 \times 5 + 1.5 \times 6}{2.75 + 1.5} = \frac{13.75 + 9}{4.25}$$

$$= \frac{22.75}{4.25} = 5.35$$

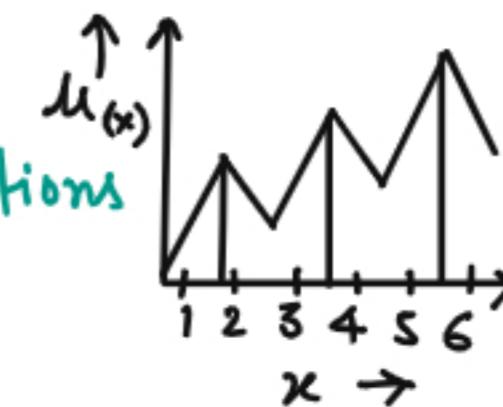
Centroid Method

Center of Gravity (COG) :- Divide into smaller portions



x_i^* - Center of Gravity

x^* - Crisp fuzzy set



$$x^* = \frac{\sum_{i=1}^n x_i A_i}{\sum_{i=1}^n A_i}$$

Truth Values & Tables in Fuzzy Logic

X is A

⇒ London is in UK
Subject Predicate

- Truth-table defines logic functions of 2 propositions

1) Conjunction (\wedge): $X \text{ AND } Y \rightarrow T_{(X \text{ AND } Y)} = T(x) \wedge T(y) = \min [T(x), T(y)]$

2) Disjunction (\vee): $X \text{ OR } Y \rightarrow T_{(X \text{ OR } Y)} = T(x) \vee T(y) = \max [T(x), T(y)]$

3) Implication (\rightarrow): If Then $Y \rightarrow T(\text{NOT } X) = 1 - T(x)$

4) Bidirectional (\leftrightarrow): $X \text{ If } Y \rightarrow T(x \leftrightarrow y) = \max [1 - T(x), \min [T(x), T(y)]]$

◦ Truth value of proposition in fuzzy

logic are in range $[0, 1]$

e.g.: - P : Ram is Boy

$$T(P) = 0.8$$

Fuzzy Proposition

P: Ram is Boy

$$T(P) = 0.0$$

$$T(P) = 0.2$$

$$T(P) = 0.8$$

$$T(P) = 1.0$$

Q: Ram is Intelligent

$$T(Q) = 0.6$$

\Rightarrow Ram is not Intelligent

$$T(\bar{Q}) = 1 - T(Q) = 1 - 0.6 = 0.4$$

\Rightarrow Ram is Boy and so is intelligent

$$\begin{aligned} T(P \wedge Q) &= \min(T(P) - T(Q)) \\ &= \min(0.8, 0.6) \\ &= 0.6 \end{aligned}$$

o Fuzzy Predicates: Shridhar is tall
 (tall, short, sick)

o Fuzzy Predicates modifier:

(Very, fairly, moderately, rather
 slightly)

- Fuzzy quantifiers: (Most, several, Many)
- Fuzzy qualifiers:
 - Based on Truth ($x \text{ is } t$)
 - Based on Probability ($x \text{ is } \lambda$)
 - Based on Possibility ($x \text{ is } \pi$)

Crisp Proposition deals with 0 and 1 OR
True and False only

Decomposition of Rules

1> Multiple Conjunctive antecedents

If x is $A_1, A_2, A_3, \dots, A_n$ Then Y is B_m

$$A_m = A_1 \cap A_2 \cap A_3 \cap \dots \cap A_n$$

$$\mu_{A_m}(x) = \min[\mu_{A_1}(x), \mu_{A_2}(x), \dots, \mu_{A_n}(x)]$$

\Rightarrow If A_m Then B_m

2> Multiple disjunctive antecedent

3> Conditional statements

⇒ If A_1 then B_1 else B_2

⇒ If A_1 then B_1

⇒ If Not A_1 then B_2

4> Nested If then Rules

If A_1 then [If A_2 then [If A_3 then B_1]]

⇒ If A_1 AND A_2 Then B_1

.

Aggregation of Fuzzy Rules

▷ Conjunctive System of Rules

- Rules to be jointly satisfied

- Using AND

- Using Intersection

$$y = y_1 \text{ AND } y_2 \text{ AND } y_3 \dots \text{ AND } y_n$$

$$y = y_1 \cap y_2 \cap y_3 \dots \cap y_n$$

$$\mu_y = \min [\mu_{y_1}(y), \mu_{y_2}(y), \mu_{y_3}(y), \dots, \mu_{y_n}(y)]$$

3) Disjunctive System of Rules

- o The satisfaction of at least one Rule
- o OR is used

$$y = y_1 \text{ or } y_2 \text{ or } y_3 \dots \text{ or } y_n$$

$$y = y_1 \cup y_2 \cup y_3 \dots \cup y_n$$

$$\mu_y(y) = \max[\mu_{y_1}(y), \mu_{y_2}(y), \mu_{y_3}(y), \dots, \mu_{y_n}(y)]$$

Takagi-Sugeno-Kang Fuzzy Rule Base System (TSK-FRBS)

Sugeno et al. suggested new model based on rules whose antecedent is composed of "Lingistic variables" and the consequent (output) is represented by a "function" of the input variables.

Ex:- $x_i \in X$ input variable, Y : output variable

If x_1 is \tilde{A}_1 and x_2 is \tilde{A}_2 and ... and x_n is \tilde{A}_n

$$\text{then } y = \beta_0 + \beta_1 x_1 + \dots + \beta_n x_n$$

consequent expression constitutes a linear combination of the variables involved in the antecedent.

$P = \{p_0, p_1, \dots, p_n\}$ are parameters

A_i = fuzzy set or a linguistic label that points to a particular member of a fuzzy partition of a linguistic variable

$$\text{Output} = \frac{\sum_{i=1}^m h_i y_i}{\sum_{i=1}^m h_i}$$

where m: number of rules present in the knowledge base
 y_i : output of each rule $1 \leq i \leq m$

$h_i : T(A_{i1}(x_1), \dots, A_{in}(x_n))$

The matching degree between the antecedent part of the i th rule and the current input to the system.

$$x_0 = (x_1, \dots, x_n)$$

TSK system do not need defuzzification being there output real numbers

It divides the input space in several fuzzy subspaces and defines a linear input output relationship in each one of the subspace.

Adv :- The system present a set of compact system equation that allows the parameters β_i to be estimated by mean of classical regression methods which facilitates the design process.

Dis: The structure of the rule consequents : difficult to be understood by human experts.

Singleton fuzzy rule system

Here, the rule consequent takes a single real valued number.

Eg: If x_1 is \hat{A}_1 and x_2 is \hat{A}_2 --- and x_n is \hat{A}_n
then y is y_0

Fuzzy Rule based Classifier

A fuzzy rule based classifier is an automatic classification system that uses fuzzy rules or knowledge representation tool.

Eg: If x_1 is A_1 and x_2 is A_2 and ... and x_n is A_n

Then Y is C_j (class label)

Here, Y is categorical variable, C is a class label

We use majority voting for the final result.

Example of Takagi and Sugeno's Approach

Rule: fuzzy antecedent then functional Consequent

i.e. Input (fuzzy) \rightarrow Output (functional)

e.g.: If (x_1 is \hat{A}_{i1}) and (x_2 is \hat{A}_{i2}) and ...
... and (x_{in} is \hat{A}_{in}) then

$$y_i = a_{i0} + a_{i1}x_1 + a_{i2}x_2 + \dots + a_{in}x_n$$

where $a_{i0}, a_{i1}, a_{i2}, \dots, a_{in}$ are coefficient

The weights of i th rule is

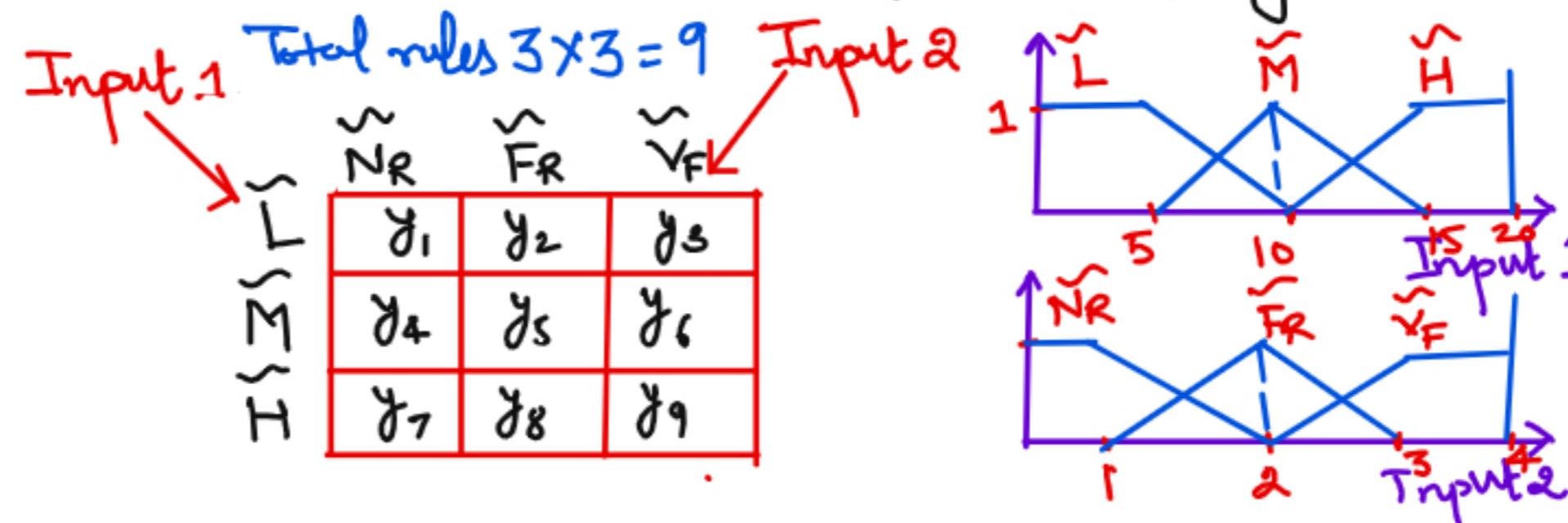
$$w_i = \mu_i \hat{A}_1(x_1) \times \mu_i \hat{A}_2(x_2) \times \dots \times \mu_i \hat{A}_n(x_n)$$

and

$$y = \frac{\sum_{i=1}^k w_i y_i}{\sum_{i=1}^k w_i}$$

where k is the total of number rules

Eg:- Let two Inputs : Input₁ and Input₂ be there
 and the corresponding linguistic states are
 for Input₁ : Low (\tilde{L}), Medium (\tilde{M}), High (\tilde{H})
 for Input₂ : Near (\tilde{NR}), Far (\tilde{FR}), Very far (\tilde{VF})



The output of the i th rule $1 \leq j, k \leq 3$

$$y_i = f(\text{Input}_1, \text{Input}_2)$$

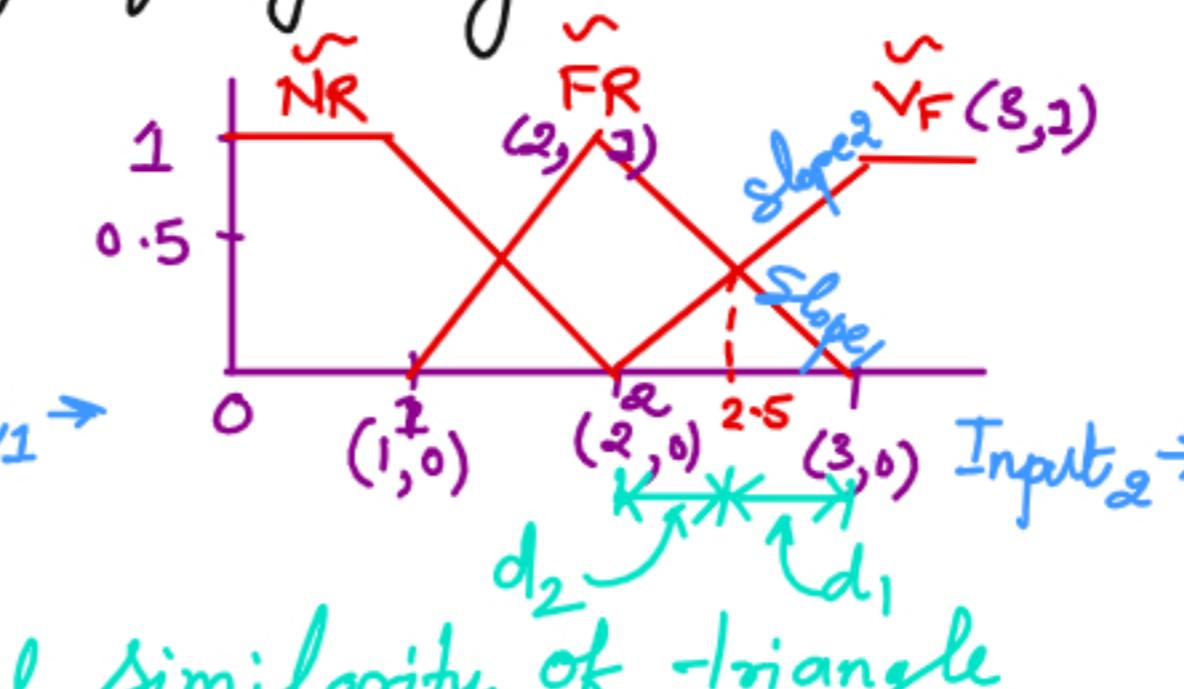
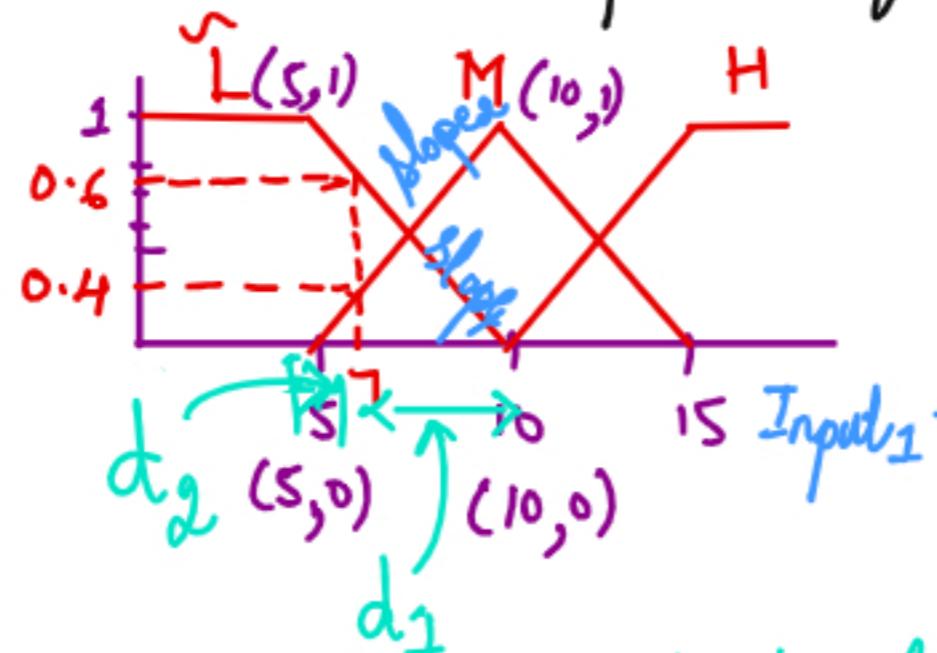
$$= a_{ij} \text{Input}_1 + b_{ik} \text{Input}_2$$

$\text{Input}_1 = \tilde{L}, \tilde{M}$, and \tilde{H} respectively

Coefficient of Input_1

$b_{i1} = 1, b_{i2} = 2, b_{i3} = 3$ for the inputs of (Input_2)
 \tilde{N}_R, \tilde{F}_R and \tilde{V}_F respectively.

If Input₁ = 7 and Input₂ = 2.5 then what will be the output of fuzzy logic control



Using the principal similarity of triangle

$$\text{Slope}_1 = \left| \frac{1-0}{5-10} \right| = \frac{1}{5}$$

$$d_1 = 10 - 7 = 3$$

$$\mu_L^*(\text{Input}_1) = \frac{1}{5} \times 3 = .6$$

$$\mu(x) = d \times \text{Slope}$$

$$\text{Slope}_2 = 7 - 5 = 2$$

$$\text{Slope}_2 = \left| \frac{1-0}{10-5} \right| = \frac{1}{5}$$

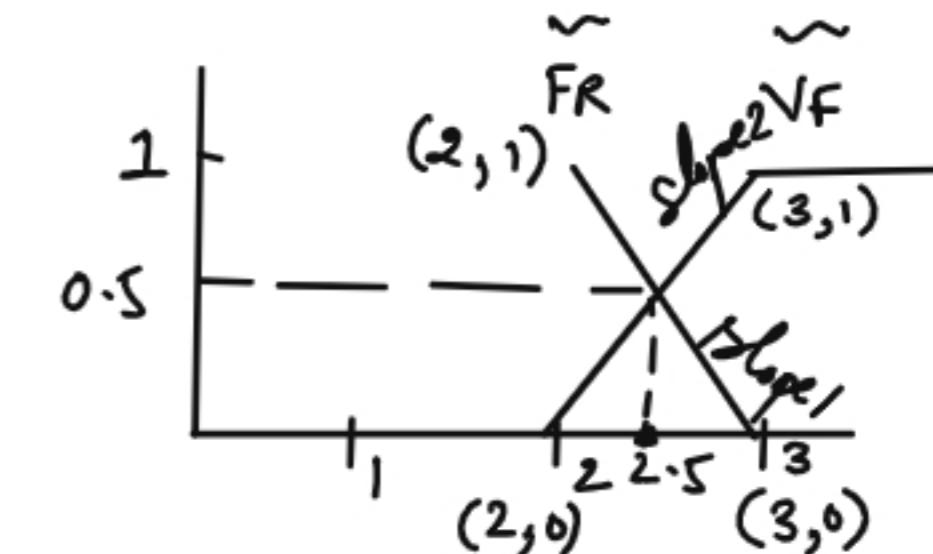
$$\mu_{\tilde{M}}(\text{Input}_1) = \frac{1}{5} \times 2 = 0.4$$

$$\text{Input}_2 = 2.5$$

$$\text{Slope}_1 = \left| \frac{1-0}{2-3} \right| = \frac{1}{1} = 1$$

$$d_1 = 3 - 2.5 = 0.5$$

$$\mu_{\tilde{F}_R}(\text{Input}_2) = 1 \times 0.5 = 0.5$$



$$\text{Slope}_2 = \left| \frac{1-0}{3-2} \right| = \frac{1}{1} = 1$$

$$d_2 = 2.5 - 2 = 0.5$$

$$\mu_{\tilde{V}_F}(\text{Input}_2) = 1 \times 0.5 = 0.5$$

$Input_2 = 2.5$	(0)	(0.5)	(0.5)
	NR	FR	VF
0.6 \tilde{L}	\checkmark_{R_1}	\checkmark_{R_2}	\checkmark_{R_3}
0.4 \tilde{M}	\times_{R_4}	\checkmark_{R_5}	\checkmark_{R_6}
0 \tilde{H}	\times_{R_7}	\times_{R_8}	\times_{R_9}

$Input_1 = 7$

Weights:

$$R_2: w_2 = \mu_{\tilde{L}}(Input_1) \times \mu_{\tilde{F}R}(Input_2) = 0.6 \times 0.5 = 0.3$$

$$R_3: w_3 = \mu_{\tilde{L}}(Input_1) \times \mu_{\tilde{V}F}(Input_2) = 0.6 \times 0.5 = 0.3$$

$$R_5: w_5 = \mu_{\tilde{M}}(Input_1) \times \mu_{\tilde{F}R}(Input_2) = 0.4 \times 0.5 = 0.2$$

$$R_6: w_6 = \mu_{\tilde{M}}(Input_1) \times \mu_{\tilde{V}F}(Input_2) = 0.4 \times 0.5 = 0.2$$

R_2 : Input₁ is \tilde{L} and Input₂ is $\tilde{F}R$

R_3 : Input₁ is \tilde{L} and Input₂ is $\tilde{V}F$

R_5 : Input₁ is \tilde{M} and Input₂ is $\tilde{F}R$

R_6 : Input₁ is \tilde{M} and Input₂ is $\tilde{V}F$

$$\begin{array}{ll}
 \text{L} = a_{i1} = 1 & \text{NR} = b_{i1} = 1 \\
 \text{M} = a_{i2} = 2 & \text{FR} = b_{i2} = 2 \\
 \text{H} = a_{i3} = 3 & \text{VF} = b_{i3} = 3
 \end{array}$$

$$\begin{array}{l}
 R_{21} \quad R_3 \\
 R_{51} \quad R_6 \\
 \text{Input}_1 = 7 \\
 \text{Input}_2 = 2.5
 \end{array}$$

Then y

$$y_2 = a_{21} \text{Input}_1 + b_{22} \cdot \text{Input}_2 = 1 \times 7 + 2 \times 2.5 = 12$$

$$y_3 = a_{31} \text{Input}_1 + b_{33} \cdot \text{Input}_2 = 1 \times 7 + 3 \times 2.5 = 14.5$$

$$y_5 = a_{52} \text{Input}_1 + b_{52} \cdot \text{Input}_2 = 2 \times 7 + 2 \times 2.5 = 19$$

$$y_6 = a_{62} \text{Input}_1 + b_{63} \cdot \text{Input}_2 = 2 \times 7 + 3 \times 2.5 = 21.5$$

The crisp output

$$\gamma = \frac{w_2 y_2 + w_3 y_3 + w_5 y_5 + w_6 y_6}{w_2 + w_3 + w_5 + w_6}$$

$$= \frac{0.3 \times 12 + 0.3 \times 14.5 + 0.2 \times 19 + 0.2 \times 21.5}{0.3 + 0.3 + 0.2 + 0.2}$$

$$= \frac{16.05}{1}$$

$$= 16.05$$

Mamdani Fuzzy model Sum with solved Example

Design a controller to determine the Wash Time of a domestic machine. Assume the input is dirt & grease on cloths. Use three descriptors for input variables and five descriptor for output variables. Derive the set of rules for controller action and defuzzification. The design should be supported by figure wherever possible. Show that if the cloths are solid to a larger degree the wash time will be more and vice versa.

Steps to Solve

Step 1: Identify input and output variables
and decide descriptor for the same.

Step 2:- Define membership functions for each
of input and output variables.

Step 3 :- Form a rule base

Step 4 :- Rule Evaluation

Step 5 :- Defuzzification

Step 1:- Identify Input and Output variable and decide descriptor.

- * Here inputs are 'dirt' and 'grease'
Assume there % age.
- * Output is 'Wash-time' measured in minutes

Descriptor for Input Variable

DIRT

SD - Small Dirt

MD - Medium Dirt

LD - Large Dirt

GREASC

NG - No Grease

MG - Medium Grease

LG - Large Grease

Wash Time

VS - Very Short

S - Short

M - Medium

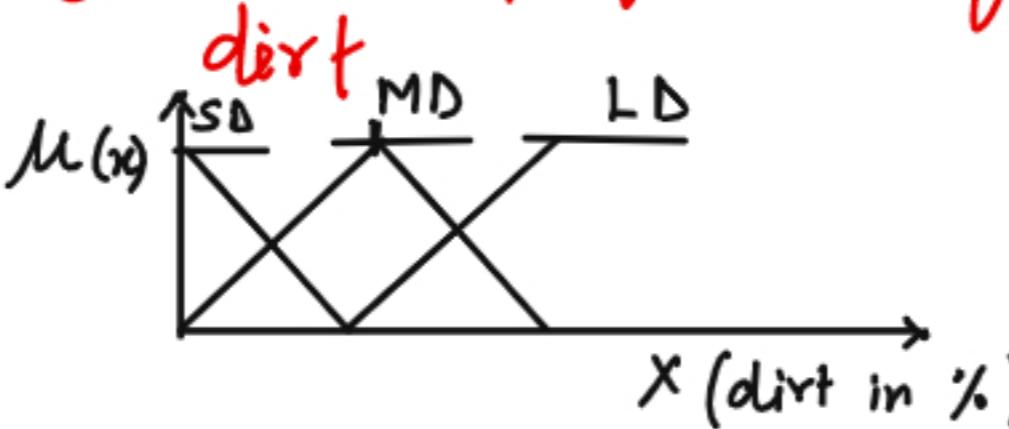
L - Large

VL - Very Large

Step 2:- Define Membership function for each of the input and output variable.

We use triangular MF

① Membership function for

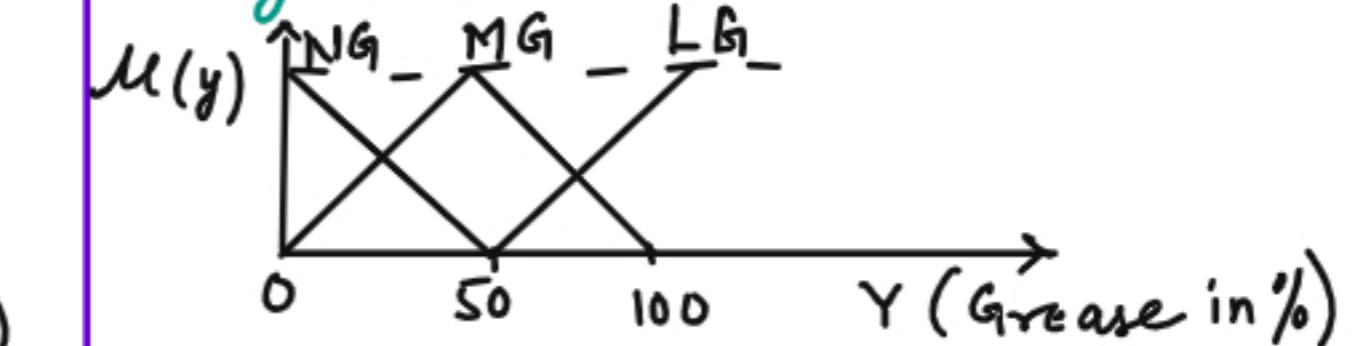


$$\mu_{SD}(x) = \frac{50-x}{50}, 0 \leq x \leq 50$$

$$\mu_{MD}(x) = \begin{cases} \frac{x}{50}, & 0 \leq x \leq 50 \\ \frac{100-x}{50}, & 50 \leq x \leq 100 \end{cases}$$

$$\mu_{LD}(x) = \frac{x-50}{50}, 50 \leq x \leq 100$$

② Membership function for grease

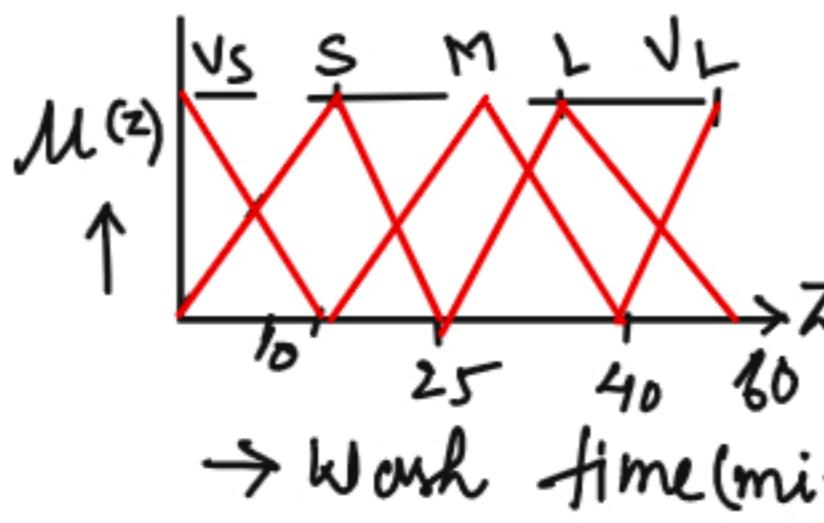


$$\mu_{NG}(y) = \frac{50-y}{50}, 0 \leq y \leq 50$$

$$\mu_{MG}(y) = \begin{cases} \frac{y}{50}, & 0 \leq y \leq 50 \\ \frac{100-y}{50}, & 50 \leq y \leq 100 \end{cases}$$

$$\mu_{LG}(y) = \frac{y-50}{50}, 50 \leq y \leq 100$$

3) Membership function for Wash Time



$$\mu_{VS}(z) = \frac{10-z}{10}, 0 \leq z \leq 10$$

$$\mu_S(z) = \begin{cases} \frac{z}{10}, & 0 \leq z \leq 10 \\ \frac{25-z}{15}, & 10 \leq z \leq 25 \end{cases}$$

$$\mu_M(z) = \begin{cases} \frac{z-10}{15}, & 10 \leq z \leq 25 \\ \frac{40-z}{15}, & 25 \leq z \leq 40 \end{cases}$$

$$\mu_L(z) = \begin{cases} \frac{z-25}{15}, & 25 \leq z \leq 40 \\ \frac{60-z}{15}, & 40 \leq z \leq 60 \end{cases}$$

$$\mu_{VL}(z) = \frac{z-40}{20}, 40 \leq z \leq 60$$

Step 3: Form a Rule Base

	x	NG	MG	LG
y	SD	S	M	L
	MD	S	M	L
	LD	M	L	NL

Evaluate $\mu_{MD}(x)$ and $\mu_{LD}(x)$
for $x=60$, we get

$$\mu_{MD}(60) = \frac{100 - 60}{50} = \frac{40}{50} = \frac{4}{5}$$

$$\mu_{LD}(60) = \frac{60 - 50}{50} = \frac{10}{50} = \frac{1}{5}$$

Step 4: Rule Evaluation

Assume Dist = 60%

Greace = 70%

$$\mu_{MD}(x) = \frac{100 - x}{50} \quad \mid \quad \mu_{LD}(x) = \frac{x - 50}{50}$$

Similarly Greace = 70% Maps 2 MFs

$$\mu_{MG}(y) = \frac{100 - y}{50} \quad \mid \quad \mu_{LG}(y) = \frac{y - 50}{50}$$

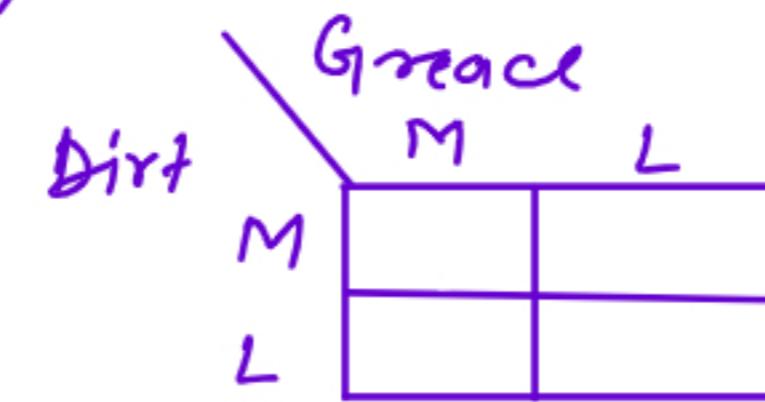
Evaluate $\mu_{MG}(y)$ & $\mu_{LG}(y)$
for $y = 70$ we get

$$\mu_{MG}(70) = \frac{100 - 70}{50} = \frac{30}{50} = \frac{3}{5}$$

$$\mu_{LG}(70) = \frac{70 - 50}{50} = \frac{20}{50} = \frac{2}{5}$$

The four equation leads to 4 rules we need to evaluate

- 1) Dirt is Medium and Greace is Medium
- 2) Dirt is Medium and Greace is Large
- 3) Dirt is Large and Greace is Medium
- 4) Dirt is Large and Greace is Large



A 2x2 grid diagram. The top-left cell contains the letter 'M'. The top-right cell contains the letter 'L'. The bottom-left cell contains the letter 'M'. The bottom-right cell contains the letter 'L'. A purple line labeled 'Dirt' points from the left towards the top-left cell. Another purple line labeled 'Greace' points from the top towards the top-left cell.

M	L
M	L

Since the antecedent is connected by the operator by **and** operator we use **min** operator to evaluate strength of each rule

Strength of Rule 1:- $D_M G_M$

$$S_1 = \min(\mu_{MD}(60), \mu_{MG}(70)) = \min\left(\frac{4}{5}, \frac{3}{5}\right) = \frac{3}{5}$$

Strength of Rule 2: $D_M G_L$

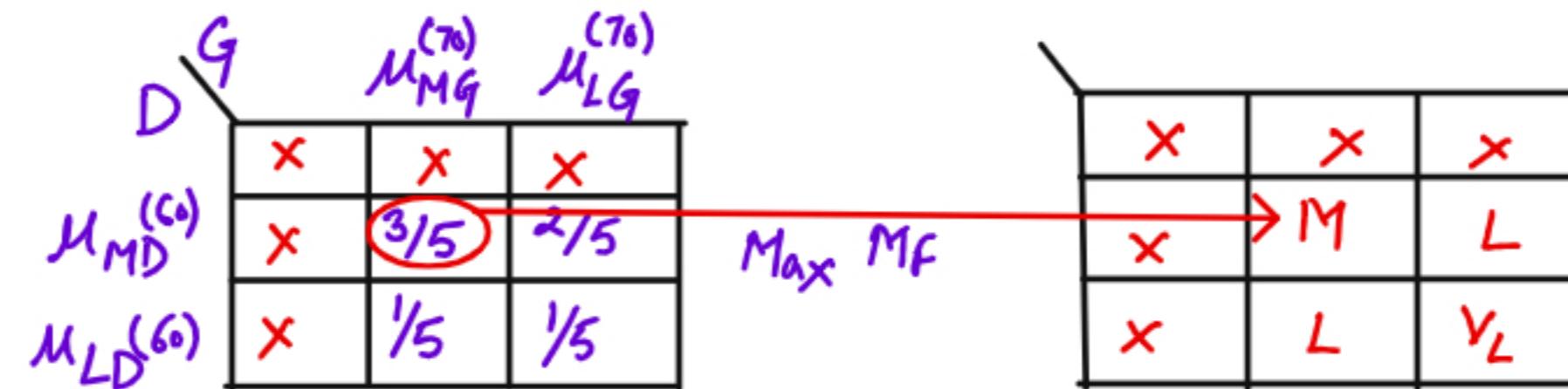
$$S_2 = \min(\mu_{MD}(60), \mu_{LG}(70)) = \min\left(\frac{4}{5}, \frac{2}{5}\right) = \frac{2}{5}$$

Strength of Rule 3: $D_L G_M$

$$S_3 = \min(\mu_{LD}(60), \mu_{MB}(70)) = \min\left(\frac{1}{5}, \frac{3}{5}\right) = \frac{1}{5}$$

Strength of Rule 4: $D_L G_L$

$$S_4 = \min(\mu_{LD}(60), \mu_{LG}(70)) = \min\left(\frac{1}{5}, \frac{2}{5}\right) = \frac{1}{5}$$



Step 5:- Defuzzification

Since we use 'Mean of Max' defuzzification technique

$$\begin{aligned} \text{Maximum Strength} &= \max(s_1, s_2, s_3, s_4) = \max\left(\frac{3}{5}, \frac{2}{5}, \frac{1}{5}, \frac{1}{5}\right) \\ &= 3/5 \text{ (This Correspond to Rule I)} \end{aligned}$$

Rule I: Dirt is Medium and Grace is Medium
has maximum strength (3/5)

To find out the final defuzzification value, we now take the average (mean) of $\mu_M(z)$

$$\mu_M(z) = \frac{z-10}{15} \text{ and } \mu_M(z) = \frac{40-z}{15}$$

$$\frac{3}{5} \nearrow z-10 \\ 5 \swarrow \frac{z-10}{15}$$

$$\frac{3}{5} \times 15 = z-10$$

$$z = 9 + 10$$

$$z = 19$$

$$\frac{3}{5} \nearrow 40-z \\ 5 \swarrow \frac{40-z}{15}$$

$$\frac{3}{5} \times 15 = 40-z$$

$$z = 40 - 9$$

$$z = 31$$

$$z^* = \frac{19+31}{2} = \frac{50}{2} = 25 \text{ min}$$

$$z^* = 25 \text{ min}$$

1. Find the power set and cardinality of the given set $X = \{2, 4, 6\}$. Also find cardinality of power set.

Solution: Since set X contains three elements, so its cardinal number is

$$n_X = 3$$

The power set of X is given by

$$\begin{aligned} P(X) = & \{\phi, \{2\}, \{4\}, \{6\}, \{2, 4\}, \\ & \{4, 6\}, \{2, 6\}, \{2, 4, 6\}\} \end{aligned}$$

The cardinality of power set $P(X)$, denoted by $n_{P(X)}$, is found as

$$n_{P(X)} = \overbrace{2^{n_X}}^{\circlearrowleft} = 2^3 = 8$$

2. Consider two given fuzzy sets

$$\underline{\mathcal{A}} = \left\{ \frac{1}{2} + \frac{0.3}{4} + \frac{0.5}{6} + \frac{0.2}{8} \right\}$$

$$\underline{\mathcal{B}} = \left\{ \frac{0.5}{2} + \frac{0.4}{4} + \frac{0.1}{6} + \frac{1}{8} \right\}$$

Perform union, intersection, difference and complement over fuzzy sets $\underline{\mathcal{A}}$ and $\underline{\mathcal{B}}$.

Solution: For the given fuzzy sets we have the following

(a) Union

$$\tilde{A} \cup \tilde{B} = \max\{\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(x)\}$$

$$= \left\{ \frac{1}{2} + \frac{0.4}{4} + \frac{0.5}{6} + \frac{1}{8} \right\}$$

$$\tilde{A} = 1 - \mu_{\tilde{A}}(x) = \left\{ \frac{0}{2} + \frac{0.7}{4} + \frac{0.5}{6} + \frac{0.8}{8} \right\}$$

$$\tilde{B} = 1 - \mu_{\tilde{B}}(x) = \left\{ \frac{0.5}{2} + \frac{0.6}{4} + \frac{0.9}{6} + \frac{0}{8} \right\}$$

(d) Difference

$$\tilde{A} \setminus \tilde{B} = \tilde{A} \cap \tilde{\bar{B}} = \left\{ \frac{0.5}{2} + \frac{0.3}{4} + \frac{0.5}{6} + \frac{0}{8} \right\}$$

$$\tilde{B} \setminus \tilde{A} = \tilde{B} \cap \tilde{\bar{A}} = \left\{ \frac{0}{2} + \frac{0.4}{4} + \frac{0.1}{6} + \frac{0.8}{8} \right\}$$

3. Given the two fuzzy sets

$$\tilde{B}_1 = \left\{ \frac{1}{1.0} + \frac{0.75}{1.5} + \frac{0.3}{2.0} + \frac{0.15}{2.5} + \frac{0}{3.0} \right\}$$

$$\tilde{B}_2 = \left\{ \frac{1}{1.0} + \frac{0.6}{1.5} + \frac{0.2}{2.0} + \frac{0.1}{2.5} + \frac{0}{3.0} \right\}$$

find the following:

- (a) $\tilde{B}_1 \cup \tilde{B}_2$; (b) $\tilde{B}_1 \cap \tilde{B}_2$; (c) $\overline{\tilde{B}_1}$;
- (d) $\overline{\tilde{B}_2}$; (e) $\tilde{B}_1 | \tilde{B}_2$; (f) $\overline{\tilde{B}_1 \cup \tilde{B}_2}$;
- (g) $\overline{\tilde{B}_1 \cap \tilde{B}_2}$; (h) $\tilde{B}_1 \cap \overline{\tilde{B}_1}$; (i) $\tilde{B}_1 \cup \overline{\tilde{B}_1}$;
- (j) $\tilde{B}_2 \cap \overline{\tilde{B}_2}$; (k) $\tilde{B}_2 \cup \overline{\tilde{B}_2}$

Solution: For the given fuzzy sets, we have the following:

$$(a) \underline{B}_1 \cup \underline{B}_2 = \left\{ \frac{1}{1.0} + \frac{0.75}{1.5} + \frac{0.3}{2.0} + \frac{0.15}{2.5} + \frac{0}{3.0} \right\}$$

$$\underline{B}_1 = \left\{ \frac{1}{1.0} + \frac{0.75}{1.5} + \frac{0.3}{2.0} + \frac{0.15}{2.5} + \frac{0}{3.0} \right\}$$

$$\underline{B}_2 = \left\{ \frac{1}{1.0} + \frac{0.6}{1.5} + \frac{0.2}{2.0} + \frac{0.1}{2.5} + \frac{0}{3.0} \right\}$$

$$(c) \overline{\underline{B}_1} = \left\{ \frac{0}{1.0} + \frac{0.25}{1.5} + \frac{0.7}{2.0} + \frac{0.85}{2.5} + \frac{1}{3.0} \right\}$$

$$(d) \overline{\underline{B}_2} = \left\{ \frac{0}{1.0} + \frac{0.4}{1.5} + \frac{0.8}{2.0} + \frac{0.9}{2.5} + \frac{1}{3.0} \right\}$$

$$(e) \underline{B}_1 \cap \overline{\underline{B}_2} = \underline{B}_1 \cap \left(\overline{\underline{B}_2} \right) \\ = \left\{ \frac{0}{1.0} + \frac{0.4}{1.5} + \frac{0.3}{2.0} + \frac{0.15}{2.5} + \frac{0}{3.0} \right\}$$

$$(f) \overline{\underline{B}_1 \cup \underline{B}_2} = \left\{ \frac{0}{1.0} + \frac{0.25}{1.5} + \frac{0.7}{2.0} + \frac{0.85}{2.5} + \frac{1}{3.0} \right\}$$

$$(g) \overline{\underline{B}_1 \cap \underline{B}_2} = \left\{ \frac{0}{1.0} + \frac{0.4}{1.5} + \frac{0.8}{2.0} + \frac{0.9}{2.5} + \frac{1}{3.0} \right\}$$

$$(h) \underline{B}_1 \cap \overline{\overline{\underline{B}_1}} = \left\{ \frac{0}{1.0} + \frac{0.25}{1.5} + \frac{0.3}{2.0} + \frac{0.15}{2.5} + \frac{0}{3.0} \right\}$$

$$(i) \underline{B}_1 \cup \overline{\overline{\underline{B}_1}} = \left\{ \frac{1}{1.0} + \frac{0.75}{1.5} + \frac{0.7}{2.0} + \frac{0.85}{2.5} + \frac{1}{3.0} \right\}$$

$$(j) \underline{B}_2 \cap \overline{\overline{\underline{B}_2}} = \left\{ \frac{0}{1.0} + \frac{0.4}{1.5} + \frac{0.2}{2.0} + \frac{0.1}{2.5} + \frac{0}{3.0} \right\}$$

$$(k) \underline{B}_2 \cup \overline{\overline{\underline{B}_2}} = \left\{ \frac{1}{1.0} + \frac{0.6}{1.5} + \frac{0.8}{2.0} + \frac{0.9}{2.5} + \frac{1}{3.0} \right\}$$

For aircraft simulator data the determination of certain changes in its operating conditions is made on the basis of hard break points in the mach region. We define two fuzzy sets \tilde{A} and \tilde{B} representing the condition of "near" a mach number of 0.65 and "in the region" of a mach number of 0.65, respectively, as follows

$\tilde{A} = \text{near mach } 0.65$

$$= \left\{ \frac{0}{0.64} + \frac{0.75}{0.645} + \frac{1}{0.65} + \frac{0.5}{0.655} + \frac{0}{0.66} \right\}$$

$\tilde{B} = \text{in the region of mach } 0.65$

$$= \left\{ \frac{0}{0.64} + \frac{0.25}{0.645} + \frac{0.75}{0.65} + \frac{1}{0.655} + \frac{0.5}{0.66} \right\}$$

For these two sets find the following:

- (a) $\tilde{A} \cup \tilde{B}$; (b) $\tilde{A} \cap \tilde{B}$; (c) $\overline{\tilde{A}}$;
- (d) $\overline{\tilde{B}}$; (e) $\overline{\tilde{A} \cup \tilde{B}}$; (f) $\overline{\tilde{A} \cap \tilde{B}}$

Solution: For the two given fuzzy sets we have the following:

(a) $\underline{A} \cup \underline{B}$

$$= \max\{\mu_{\underline{A}}(x), \mu_{\underline{B}}(x)\}$$

$$= \left\{ \frac{0}{0.64} + \frac{0.75}{0.645} + \frac{1}{0.65} + \frac{1}{0.655} + \frac{0.5}{0.66} \right\}$$

\underline{A} = near mach 0.65

$$= \left\{ \frac{0}{0.64} + \frac{0.75}{0.645} + \frac{1}{0.65} + \frac{0.5}{0.655} + \frac{0}{0.66} \right\}$$

\underline{B} = in the region of mach 0.65

$$= \left\{ \frac{0}{0.64} + \frac{0.25}{0.645} + \frac{0.75}{0.65} + \frac{1}{0.655} + \frac{0.5}{0.66} \right\}$$

(b) $\underline{A} \cap \underline{B}$

$$= \min\{\mu_{\underline{A}}(x), \mu_{\underline{B}}(x)\}$$

$$= \left\{ \frac{0}{0.64} + \frac{0.25}{0.645} + \frac{0.75}{0.65} + \frac{0.5}{0.655} + \frac{0}{0.66} \right\}$$

(c) $\bar{\underline{A}} = 1 - \mu_{\underline{A}}(x)$

$$= \left\{ \frac{1}{0.64} + \frac{0.25}{0.645} + \frac{0}{0.65} + \frac{0.5}{0.655} + \frac{1}{0.66} \right\}$$

(d) $\bar{\underline{B}} = 1 - \mu_{\underline{B}}(x)$

$$= \left\{ \frac{1}{0.64} + \frac{0.75}{0.645} + \frac{0.25}{0.65} + \frac{0}{0.655} + \frac{0.5}{0.66} \right\}$$

(e) $\bar{\underline{A}} \cup \bar{\underline{B}}$

$$= 1 - \max\{\mu_{\underline{A}}(x), \mu_{\underline{B}}(x)\}$$

$$= \left\{ \frac{1}{0.64} + \frac{0.25}{0.645} + \frac{0}{0.65} + \frac{0}{0.655} + \frac{0.5}{0.66} \right\}$$

(f) $\bar{\underline{A}} \cap \bar{\underline{B}}$

$$= 1 - \min\{\mu_{\underline{A}}(x), \mu_{\underline{B}}(x)\}$$

$$= \left\{ \frac{1}{0.64} + \frac{0.75}{0.645} + \frac{0.25}{0.65} + \frac{0.5}{0.655} + \frac{1}{0.66} \right\}$$

The discretized membership functions for a transistor and a resistor are given below:

$$\mu_T = \left\{ \frac{0}{0} + \frac{0.2}{1} + \frac{0.7}{2} + \frac{0.8}{3} + \frac{0.9}{4} + \frac{1}{5} \right\}$$

$$\mu_R = \left\{ \frac{0}{0} + \frac{0.1}{1} + \frac{0.3}{2} + \frac{0.2}{3} + \frac{0.4}{4} + \frac{0.5}{5} \right\}$$

Find the following: (a) Algebraic sum; (b) algebraic product; (c) bounded sum; (d) bounded difference.

The discretized membership functions for a transistor and a resistor are given below:

$$\mu_I = \left\{ 0 + \frac{0.2}{1} + \frac{0.7}{2} + \frac{0.8}{3} + \frac{0.9}{4} + \frac{1}{5} \right\}$$

$$\mu_R = \left\{ 0 + \frac{0.1}{1} + \frac{0.3}{2} + \frac{0.2}{3} + \frac{0.4}{4} + \frac{0.5}{5} \right\}$$

Find the following: (a) Algebraic sum; (b) algebraic product; (c) bounded sum; (d) bounded difference.

(a) Algebraic sum

$$\begin{aligned}\mu_{I+R}(x) &= [\mu_I(x) + \mu_R(x)] - [\mu_I(x) \cdot \mu_R(x)] \\ &= \left\{ 0 + \frac{0.3}{1} + \frac{1.0}{2} + \frac{1.0}{3} + \frac{1.3}{4} + \frac{1.5}{5} \right\} \\ &\quad - \left\{ 0 + \frac{0.02}{1} + \frac{0.21}{2} + \frac{0.16}{3} + \frac{0.36}{4} \right. \\ &\quad \left. + \frac{0.5}{5} \right\} \\ &= \left\{ 0 + \frac{0.28}{1} + \frac{0.79}{2} + \frac{0.84}{3} + \frac{0.94}{4} \right. \\ &\quad \left. + \frac{1}{5} \right\}\end{aligned}$$

(b) Algebraic product

$$\begin{aligned}\mu_{I \cdot R}(x) &= \mu_I(x) \cdot \mu_R(x) \\ &= \left\{ 0 + \frac{0.02}{1} + \frac{0.21}{2} + \frac{0.16}{3} + \frac{0.36}{4} + \frac{0.5}{5} \right\}\end{aligned}$$

(c) Bounded sum

$$\begin{aligned}\mu_{I \oplus R}(x) &= \min\{1, \mu_I(x) + \mu_R(x)\} \\ &= \min \left\{ 1, \left\{ 0 + \frac{0.3}{1} + \frac{1.0}{2} + \frac{1.0}{3} \right. \right. \\ &\quad \left. \left. + \frac{1.3}{4} + \frac{1.5}{5} \right\} \right\} \\ &= \left\{ 0 + \frac{0.3}{1} + \frac{1.0}{2} + \frac{1.0}{3} + \frac{1.0}{4} + \frac{1.0}{5} \right\}\end{aligned}$$

(d) Bounded difference

$$\begin{aligned}\mu_{I \ominus R}(x) &= \max\{0, \mu_I(x) - \mu_R(x)\} \\ &= \max \left\{ 0, \left\{ 0 + \frac{0.1}{1} + \frac{0.4}{2} + \frac{0.6}{3} \right. \right. \\ &\quad \left. \left. + \frac{0.5}{4} + \frac{0.5}{5} \right\} \right\} \\ &= \left\{ 0 + \frac{0.1}{1} + \frac{0.4}{2} + \frac{0.6}{3} + \frac{0.5}{4} + \frac{0.5}{5} \right\}\end{aligned}$$

Consider the following two fuzzy sets:

$$A = \left\{ \frac{0.3}{x_1} + \frac{0.7}{x_2} + \frac{1}{x_3} \right\}$$

and $B = \left\{ \frac{0.4}{y_1} + \frac{0.9}{y_2} \right\}$

Perform the Cartesian product over these given fuzzy sets.

Solution: The fuzzy Cartesian product performed over fuzzy sets \underline{A} and \underline{B} results in fuzzy relation \underline{R} given by $\underline{R} = \underline{A} \times \underline{B}$. Hence

$$\underline{R} = \begin{bmatrix} 0.3 & 0.3 \\ 0.4 & 0.7 \\ 0.4 & 0.9 \end{bmatrix} \sim \begin{array}{c|cc|c} & 1 & 2 & \\ \hline 1 & & & \\ 2 & & & \\ 3 & & & \end{array}$$

The calculation for \underline{R} is as follows:

$$\begin{aligned}\mu_{\underline{R}}(x_1, y_1) &= \min[\mu_{\underline{A}}(x_1), \mu_{\underline{B}}(y_1)] \\ &= \min(0.3, 0.4) = 0.3\end{aligned}$$

$$\begin{aligned}\mu_{\underline{R}}(x_1, y_2) &= \min[\mu_{\underline{A}}(x_1), \mu_{\underline{B}}(y_2)] \\ &= \min(0.3, 0.9) = 0.3\end{aligned}$$

$$\begin{aligned}\mu_{\underline{R}}(x_2, y_1) &= \min[\mu_{\underline{A}}(x_2), \mu_{\underline{B}}(y_1)] \\ &= \min(0.7, 0.4) = 0.4\end{aligned}$$

$$\begin{aligned}\mu_{\underline{R}}(x_2, y_2) &= \min[\mu_{\underline{A}}(x_2), \mu_{\underline{B}}(y_2)] \\ &= \min(0.7, 0.9) = 0.7\end{aligned}$$

$$\begin{aligned}\mu_{\underline{R}}(x_3, y_1) &= \min[\mu_{\underline{A}}(x_3), \mu_{\underline{B}}(y_1)] \\ &= \min(1, 0.4) = 0.4\end{aligned}$$

$$\begin{aligned}\mu_{\underline{R}}(x_3, y_2) &= \min[\mu_{\underline{A}}(x_3), \mu_{\underline{B}}(y_2)] \\ &= \min(1, 0.9) = 0.9\end{aligned}$$

Consider two relations

$$R = \begin{matrix} & -100 & -50 & 0 & 50 & 100 \\ 9 & [0.2 & 0.5 & 0.7 & 1 & 0.9] \\ 18 & [0.3 & 0.5 & 0.7 & 1 & 0.8] \\ 27 & [0.4 & 0.6 & 0.8 & 0.9 & 0.4] \\ 36 & [0.9 & 1 & 0.8 & 0.6 & 0.4] \end{matrix}$$

and

$$S = \begin{matrix} & 2 & 4 & 8 & 16 & 20 \\ -100 & [1 & 0.8 & 0.6 & 0.3 & 0.1] \\ -50 & [0.7 & 1 & 0.7 & 0.5 & 0.4] \\ 0 & [0.5 & 0.6 & 1 & 0.8 & 0.8] \\ 50 & [0.3 & 0.4 & 0.6 & 1 & 0.9] \\ 100 & [0.9 & 0.3 & 0.5 & 0.7 & 1] \end{matrix}$$

If R is a relationship between frequency and temperature and S represents a relation between temperature and reliability index of a circuit, obtain the relation between frequency and reliability index using (a) max-min composition and (b) max-product composition.

(a) Max-min composition is performed as follows.

$$T = R \circ S = \max\{\min[\mu_R(x, y), \mu_S(x, y)]\}.$$

$$\begin{array}{cccccc}
 & 2 & 4 & 8 & 16 & 20 \\
 = & 9 & \left[\begin{array}{ccccc} 0.9 & 0.6 & 0.7 & 1 & 0.9 \end{array} \right] \\
 & 18 & \left[\begin{array}{ccccc} 0.8 & 0.6 & 0.7 & 1 & 0.9 \end{array} \right] \\
 & 27 & \left[\begin{array}{ccccc} 0.6 & 0.6 & 0.8 & 0.9 & 0.9 \end{array} \right] \\
 & 36 & \left[\begin{array}{ccccc} 0.9 & 1 & 0.8 & 0.8 & 0.8 \end{array} \right]
 \end{array}$$

(b) Max-product composition is performed as follows.

$$T = R \circ S = \max\{\min[\mu_R(x, y) \times \mu_S(x, y)]\}$$

$$\begin{array}{cccccc}
 & 2 & 4 & 8 & 16 & 20 \\
 = & 9 & \left[\begin{array}{ccccc} 0.81 & 0.5 & 0.7 & 1.0 & 0.9 \end{array} \right] \\
 & 18 & \left[\begin{array}{ccccc} 0.72 & 0.5 & 0.7 & 1.0 & 0.9 \end{array} \right] \\
 & 27 & \left[\begin{array}{ccccc} 0.4 & 0.6 & 0.8 & 0.9 & 0.81 \end{array} \right] \\
 & 36 & \left[\begin{array}{ccccc} 0.9 & 1.0 & 0.8 & 0.64 & 0.64 \end{array} \right]
 \end{array}$$

Thus the relation between frequency and reliability index has been found using composition techniques.

Using the inference approach, find the membership values for the triangular shapes L_R , E , L_R , and T for a triangle with angles 45° , 55° and 80° .

Solution: Let the universe of discourse be

$$U = \{(X, Y, Z) : X = 80^\circ \geq Y = 55^\circ \geq Z = 45^\circ \text{ and } X + Y + Z = 80^\circ + 55^\circ + 45^\circ = 180^\circ\}$$

- Membership value of isosceles triangle, L :

$$\begin{aligned} \mu_L &= 1 - \frac{1}{60^\circ} \min(X - Y, Y - Z) \\ &= 1 - \frac{1}{60^\circ} \min(80^\circ - 55^\circ, 55^\circ - 45^\circ) \\ &= 1 - \frac{1}{60^\circ} \min(25^\circ, 10^\circ) \\ &= 1 - \frac{1}{60^\circ} \times 10^\circ \\ &= 1 - 0.1667 = 0.833 \end{aligned}$$

- Membership value of right-angle triangle, R :

$$\begin{aligned} \mu_R &= 1 - \frac{1}{90^\circ} |X - 90^\circ| = 1 - \frac{1}{90^\circ} |80^\circ - 90^\circ| \\ &= 1 - \frac{1}{90^\circ} \times 10^\circ = 0.889 \end{aligned}$$

- Membership value of equilateral triangle, E :

$$\begin{aligned} \mu_E &= 1 - \frac{1}{180^\circ} (X - Z) = 1 - \frac{1}{180^\circ} (80^\circ - 45^\circ) \\ &= 1 - \frac{1}{180^\circ} \times 35^\circ = 0.8056 \end{aligned}$$

- Membership value of isosceles and right-angle triangle, LR :

$$\begin{aligned} \mu_{LR} &= \min[\mu_L, \mu_R] = \min[0.833, 0.889] \\ &= 0.833 \end{aligned}$$

- Membership value of other triangles, T :

$$\begin{aligned} \mu_T &= \min[1 - \mu_L, 1 - \mu_E, 1 - \mu_R] \\ &= \min[0.167, 0.1944, 0.111] = 0.111 \end{aligned}$$

Thus the membership function is calculated for the triangular shapes.

For the given membership function as shown in Figure 1 below, determine the defuzzified output value by seven methods.

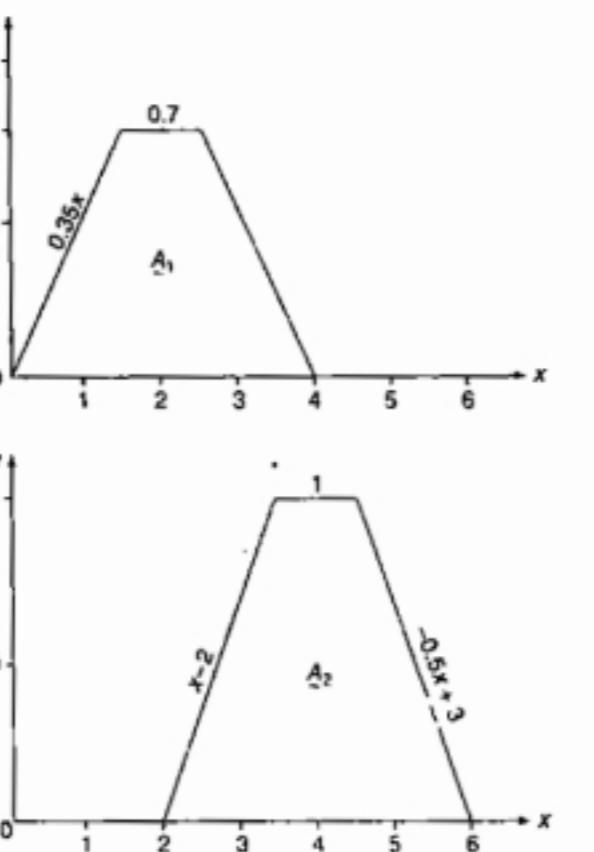


Figure 1 Membership functions.

- Centroid method

The two points are $(0, 0)$ and $(2, 0.7)$. The straight line is given by $(y - y_1) = m(x - x_1)$. Hence,

$$y - 0 = \frac{0.7}{2}(x - 0)$$

$$A_{11} \Rightarrow y = 0.35x$$

$$A_{12} \Rightarrow y = 0.7$$

$$A_{13} \Rightarrow \text{not necessary}$$

$$A_{21} \Rightarrow \text{the two points are } (2, 0), (3, 1)$$

$$y = x - 2$$

$$A_{22} \Rightarrow y = 1$$

$A_{23} \Rightarrow$ the two points are $(4, 1), (6, 0)$
we get $y = -0.5x + 3$

(A) From A_{12} we obtain $y = 0.7$.

(B) From A_{21} we obtain $y = x - 2$. On substituting the value $y = 0.7$ in (B), we obtain

$$x - 2 = 0.7 \Rightarrow x = 2.7$$

$$y = 0.7$$

The centroid method defuzzified output is

$$x^* = \int \frac{\mu_C(x)x dx}{\mu_C(x)}$$

$$\left[\int_0^2 0.35x^2 dx + \int_2^{2.7} 0.7x dx + \int_{2.7}^3 (x^2 - 2) dx \right]$$

$$+ \int_3^4 x dx + \int_4^6 (-0.5x^2 + 3x) dx \Big]$$

$$= \frac{\left[\int_0^2 0.35x^2 dx + \int_2^{2.7} 0.7x dx + \int_{2.7}^3 (x^2 - 2) dx \right]}{\left[\int_0^2 0.35x^2 dx + \int_2^{2.7} 0.7x dx + \int_{2.7}^3 (x^2 - 2) dx \right]}$$

$$+ \int_3^4 x dx + \int_4^6 (-0.5x^2 + 3x) dx \Big]$$

$$= \frac{10.78}{3.445} = 3.187$$

For the given membership function as shown in Figure 1 below, determine the defuzzified output value by seven methods.

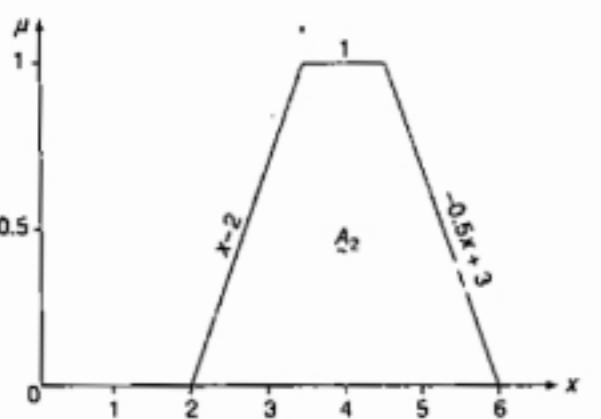
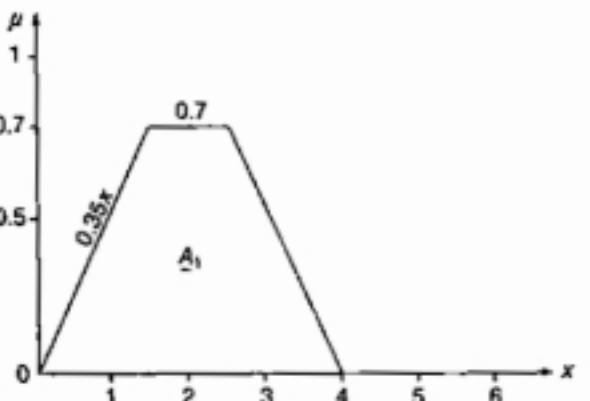


Figure 1 Membership functions.

- **Weighted average method:** The defuzzified value here is given by

$$x^* = \frac{2(0.7) + 4(1)}{0.7 + 1} = 3.176$$

- **Mean-max method:** The crisp output value here is given by

$$x^* = \frac{a + b}{2} = \frac{2.5 + 3.5}{2} = 3$$

- **Center of sums method:** The defuzzified value x^* is given by

$$x^* = \frac{\int x \sum_{i=1}^n \mu_{C_i}(x) dx}{\int \sum_{i=1}^n \mu_{C_i}(x) dx}$$

$$\begin{aligned} & \left[\int_0^6 [\frac{1}{2} \times 0.7 \times (3+2) \times 2 + \frac{1}{2} \times 1 \right. \\ & \quad \left. \times (2+4) \times 4] dx \right] \\ & = \frac{\left[\int_0^6 [\frac{1}{2} \times 0.7 \times (3+2) + \frac{1}{2} \times 1 \right. \\ & \quad \left. \times (2+4) \times 4] dx \right]}{\left[\int_0^6 (3.5 + 12) dx \right]} \\ & = \frac{0}{\int_0^6 (1.75 + 3) dx} = 2.84 \end{aligned}$$

For the given membership function as shown in Figure 1 below, determine the defuzzified output value by seven methods.

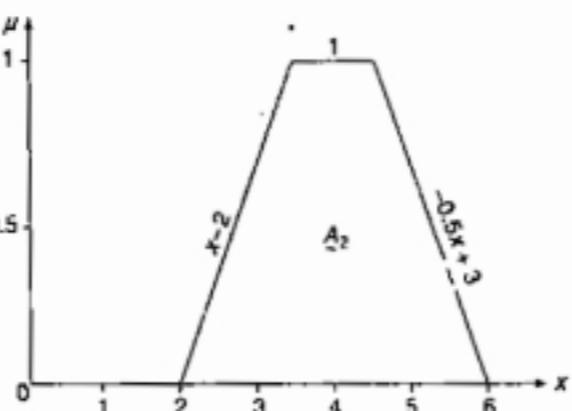
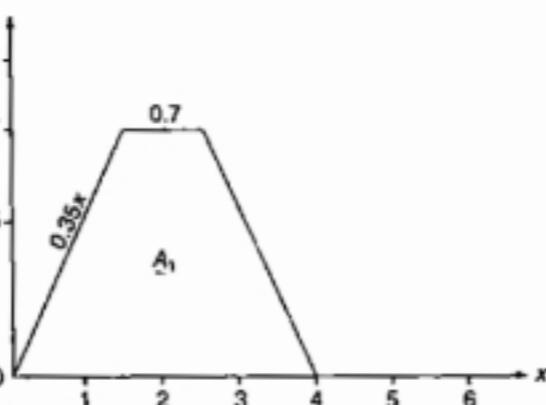


Figure 1 Membership functions.

Center of largest area:

$$\text{Area of I} = \frac{1}{2} \times 0.7 \times (2.7 + 0.7) = 1.19$$

$$\begin{aligned}\text{Area of II} &= \frac{1}{2} \times 1 \times (2 + 3) \times \frac{1}{2} \times 0.7 \\ &= 2.255\end{aligned}$$

Area of II is found to be larger; therefore the defuzzified output value is given by

$$\begin{aligned}x^* &= \frac{\int \mu_{G_i}(x)dx}{\int \mu_{G_i}(x)dx} \\ &= \frac{\left[\int_{2.7}^4 \frac{1}{2} \times 0.3 \times 0.3 \times 2.85 dx + \int_3^4 1 \times 1 \times 3.5 dx + \int_{\frac{3}{4}}^6 \frac{1}{2} \times 2 \times 1 dx \right]}{\left[\int_{2.7}^4 \frac{1}{2} \times 0.3 \times 0.3 \times 1 dx + \int_3^4 1 \times 1 dx + \int_{\frac{3}{4}}^6 \frac{1}{2} \times 2 \times 1 dx \right]} \\ &= \frac{\int_{2.7}^4 0.12825 dx + \int_3^4 3.5 dx + \int_{\frac{3}{4}}^6 5 dx}{\int_{2.7}^4 0.045 dx + \int_3^4 dx + \int_{\frac{3}{4}}^6 dx} = 4.49\end{aligned}$$

First of maxima: The defuzzified output value is

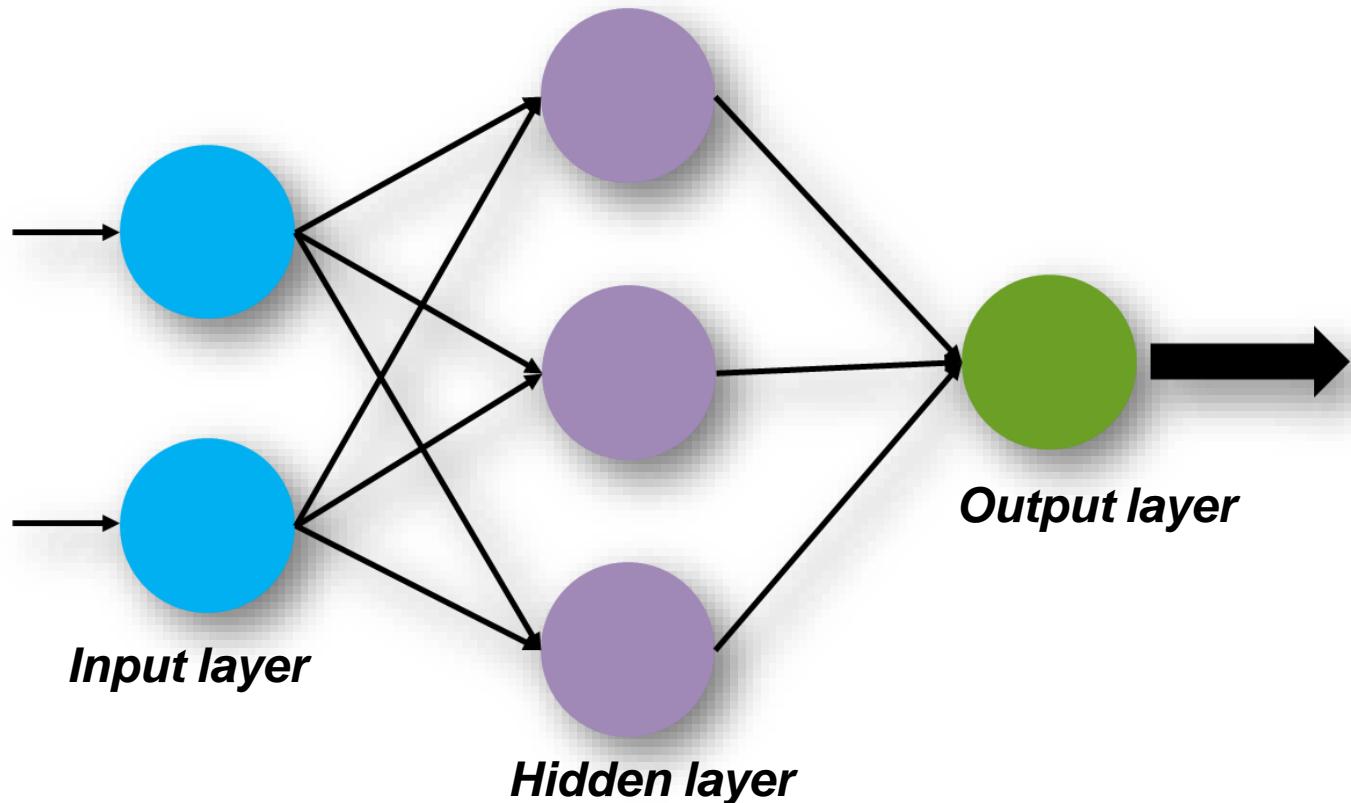
$$x^* = 3$$

Last of maxima: The defuzzified output value is

$$x^* = 4$$

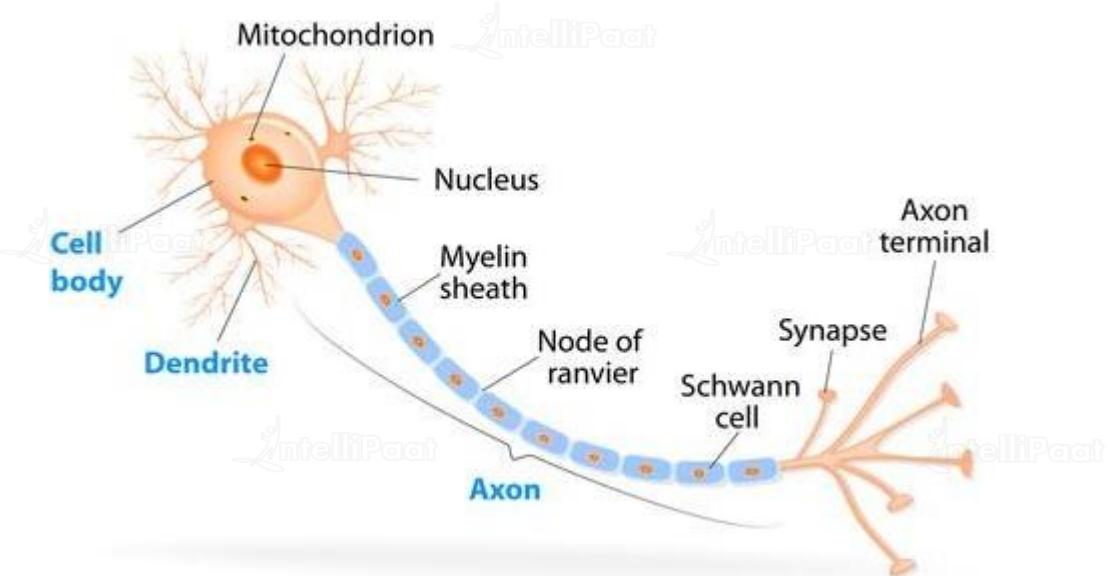
Topology of a Neural Network

Typically, artificial neural networks have a layered structure. The Input Layer picks up the input signals and passes them on to the next layer, also known as the 'Hidden' Layer (there may be more than one Hidden Layer in a neural network). Last comes the Output Layer that delivers the result



Neurons: How Do They Work?

A neural network is a computer simulation of the way biological neurons work within a human brain



Dendrites: These branch-like structures extending away from the cell body receive messages from other neurons and allow them travel to the cell body

Cell Body: It contains a nucleus, smooth and rough endoplasmic reticulum, Golgi apparatus, mitochondria, and other cellular components

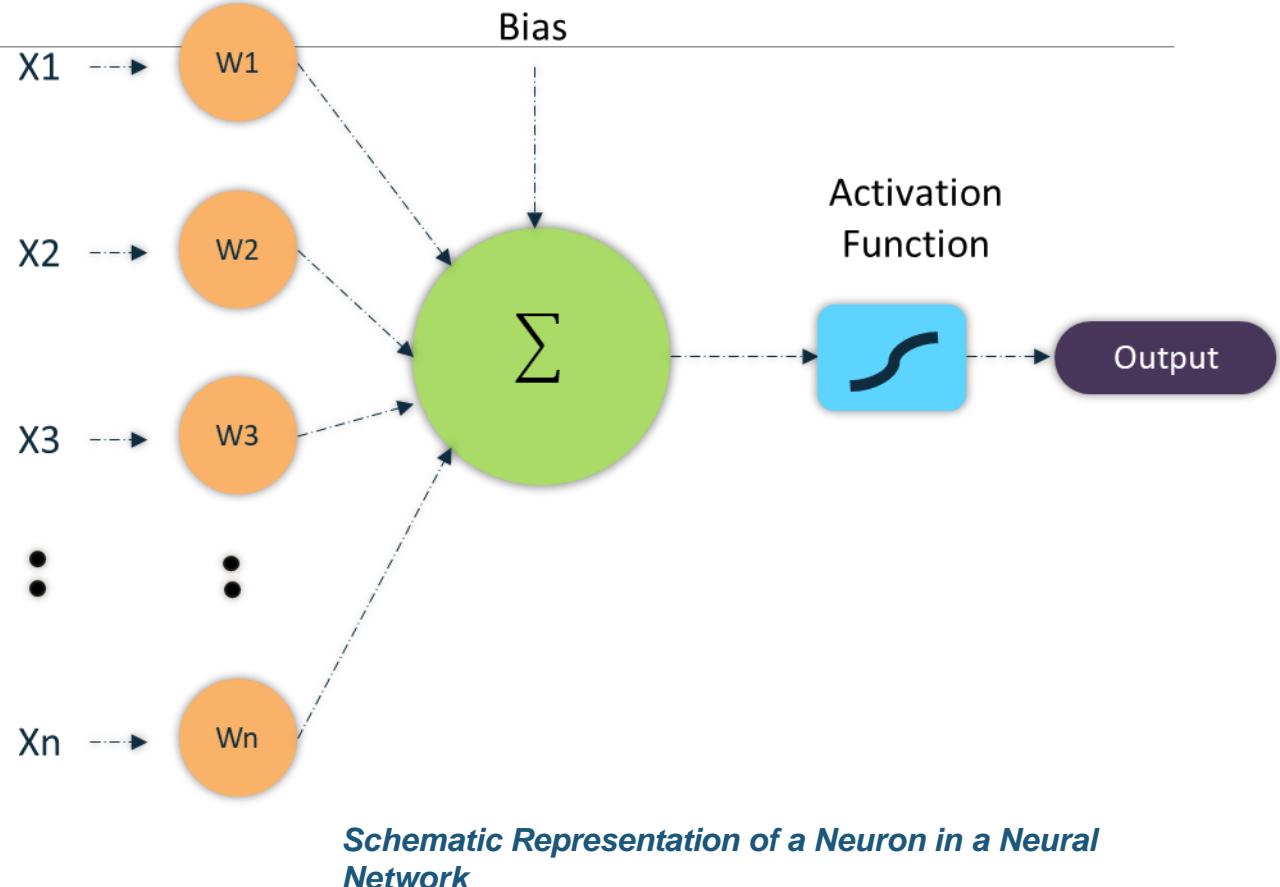
Axon: An axon carries an electrical impulse from the cell body to another neuron



Now, let us understand about artificial
neurons in detail!

Artificial Neurons

- The most fundamental unit of a deep neural network is called as an artificial neuron
- It takes an input, processes it, passes it through an activation function, and returns the output
- Such type of artificial neurons are called as *perceptrons*
- A perceptron is a linear model used for binary classification



Perceptron: How Does It Work?

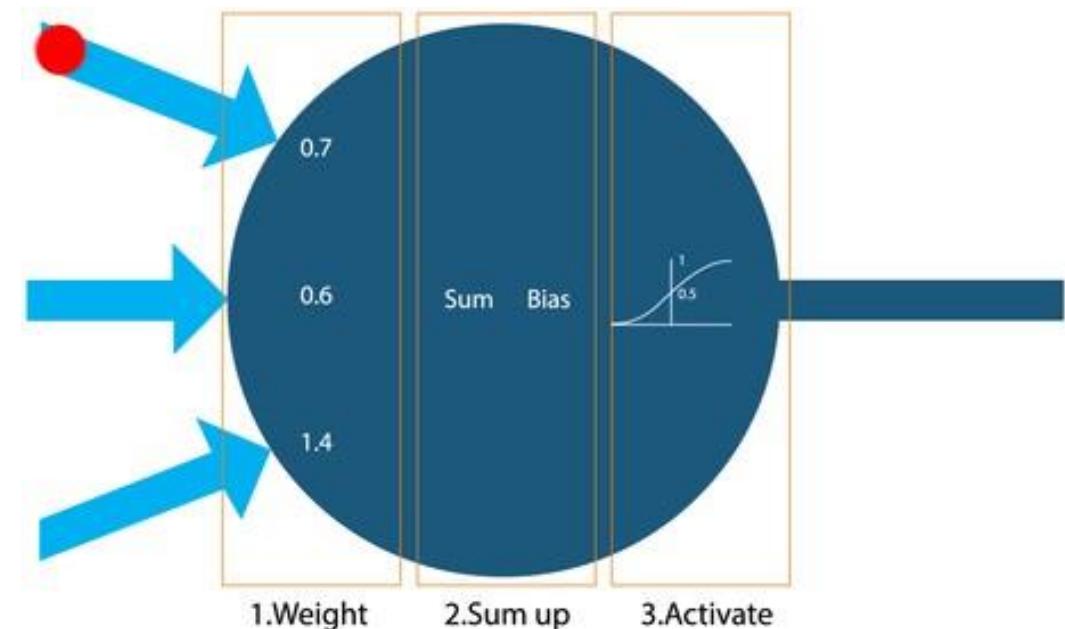
- The three arrows correspond to the three inputs coming into the network
- Values [0.7, 0.6, and 1.4] are weights assigned to the corresponding input
- Inputs get multiplied with their respective weights and their sum is taken
- Consider the three inputs as x_1 , x_2 , and x_3
- Let the three weights be w_1 , w_2 , and w_3

$$\text{Sum} = x_1w_1 + x_2w_2+x_3w_3$$

$$\text{Sum} = x_1(0.7) + x_2(0.6) + x_3(1.4)$$

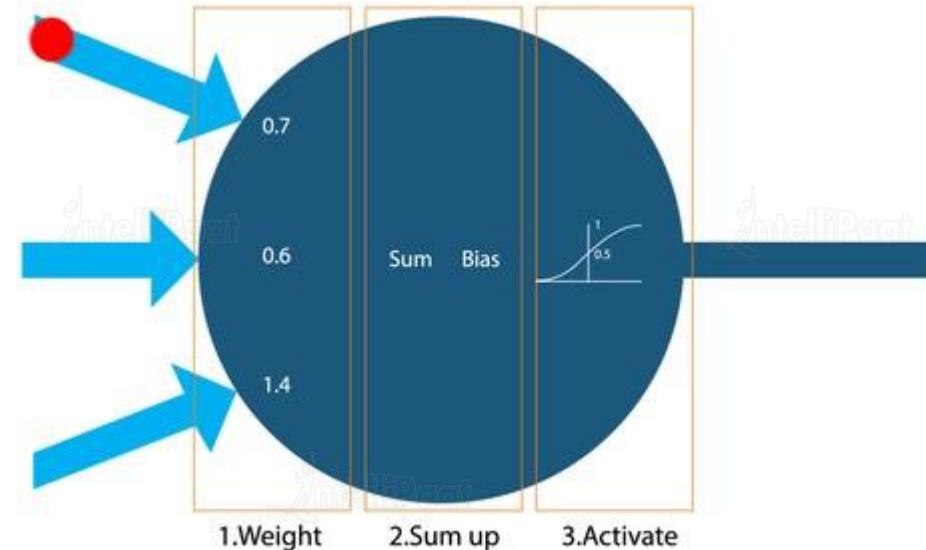
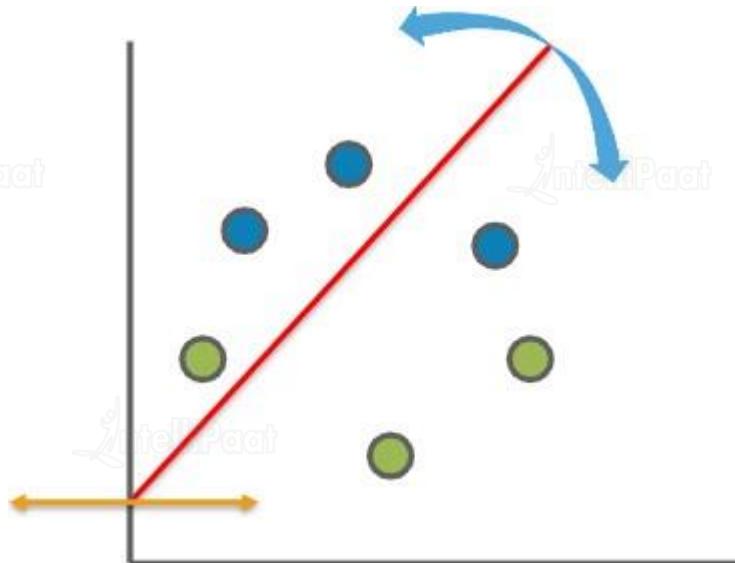
- An offset is added to this sum. This offset is called **Bias**
- It is just a constant number, say 1, which is added for scaling purposes

$$\text{New_Sum} = x_1(0.7) + x_2(0.6) + x_3(1.4) + \text{bias}$$

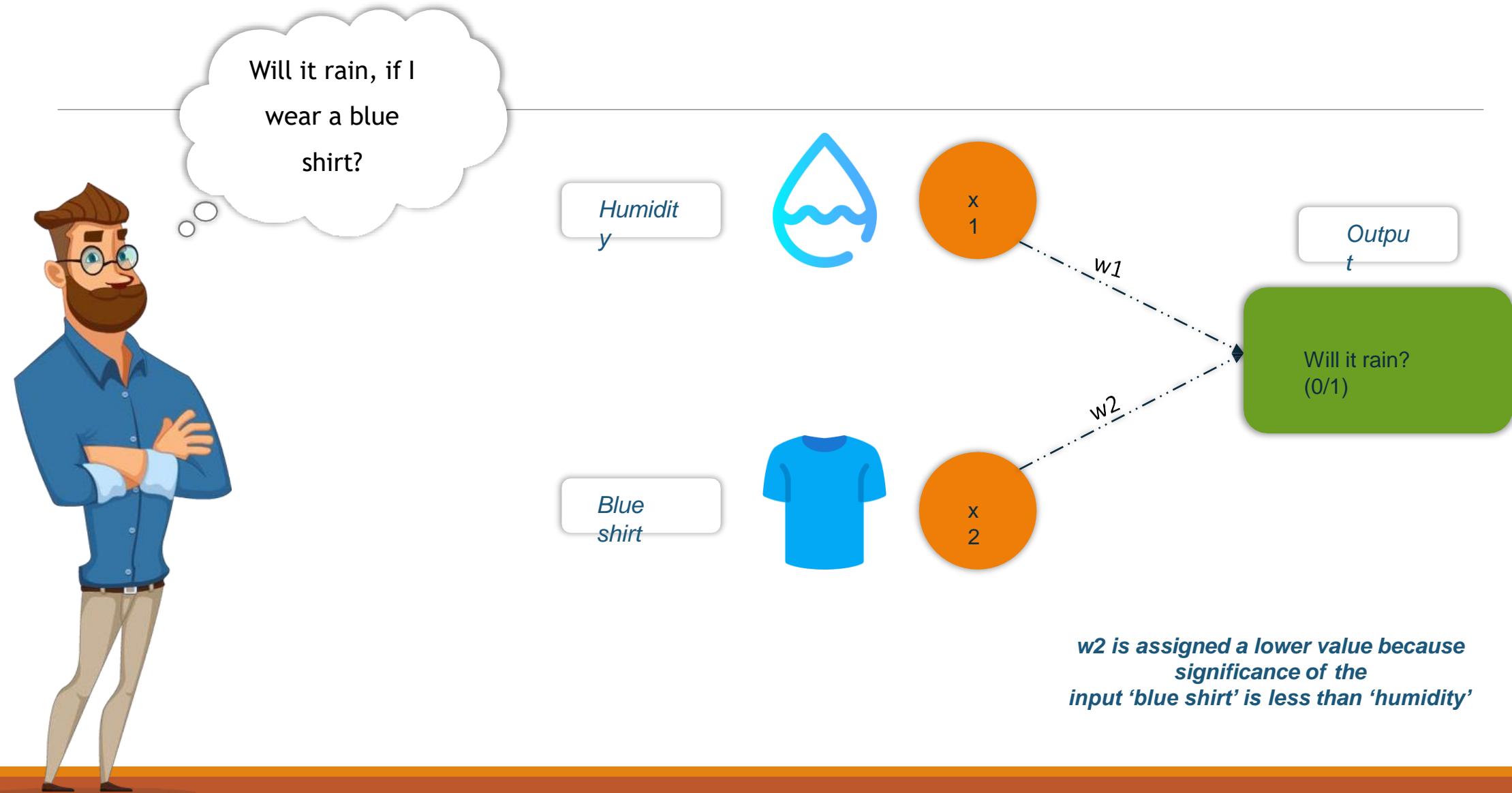


Why Do We Need Weights?

- Statistically, weights determine the relative importance of input
- Mathematically, they are just the slope of the line



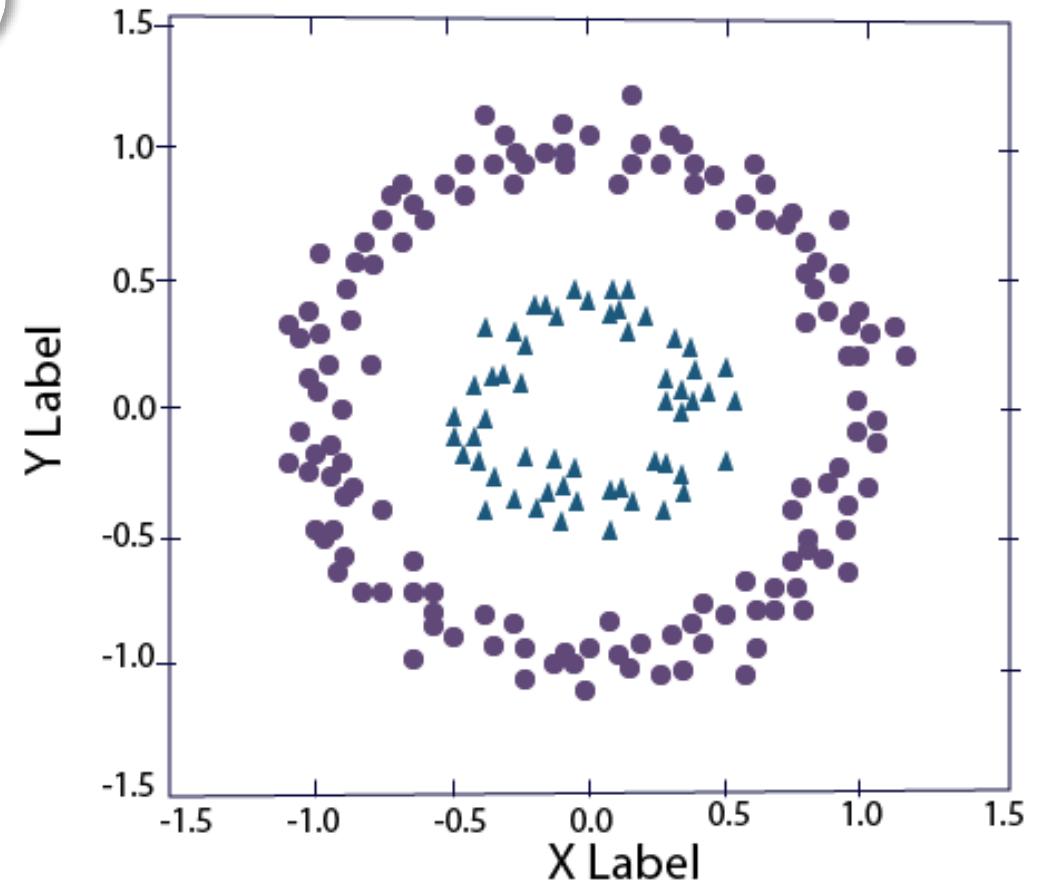
Why Do We Need Weights?



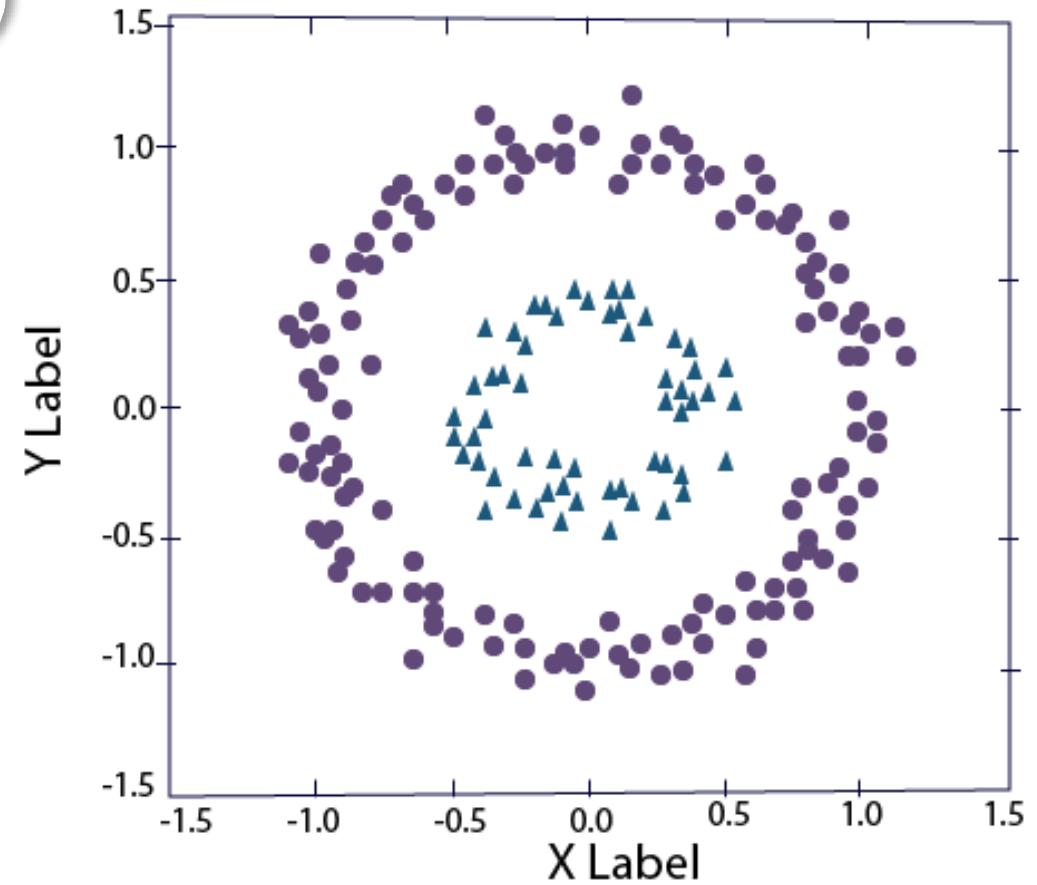
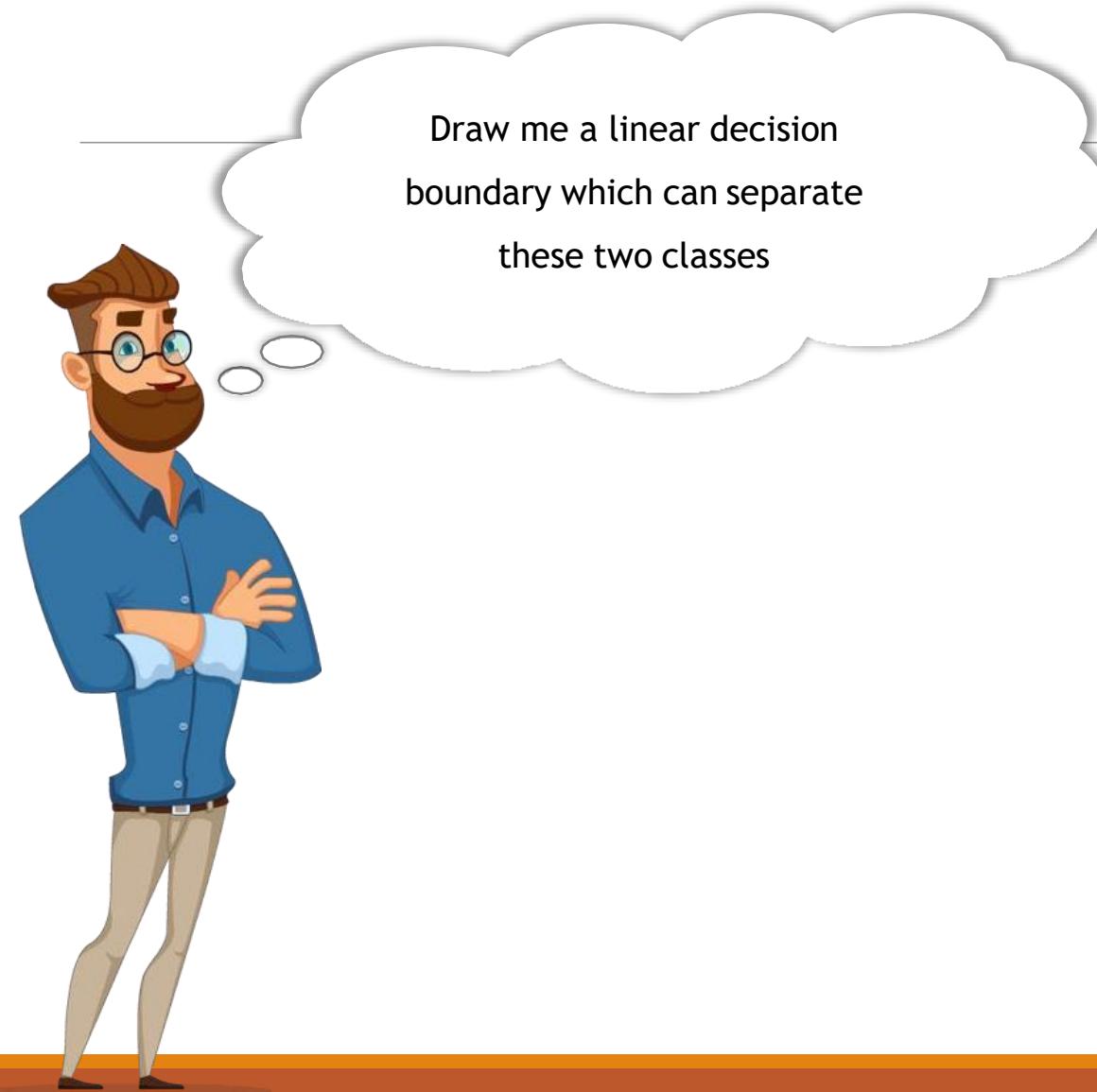
Why Do We Need Activation Functions?



We have two classes. One set is represented with triangles and the other with circles



Why Do We Need Activation Functions?

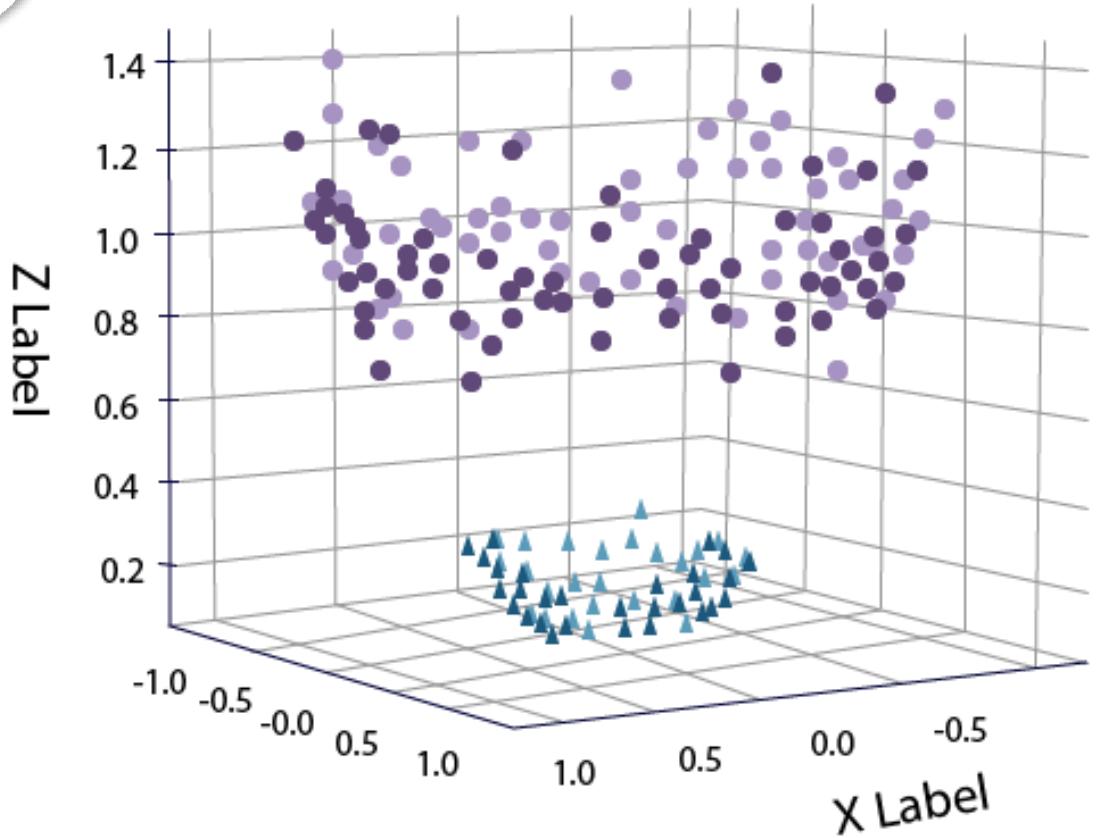


Activation Functions?



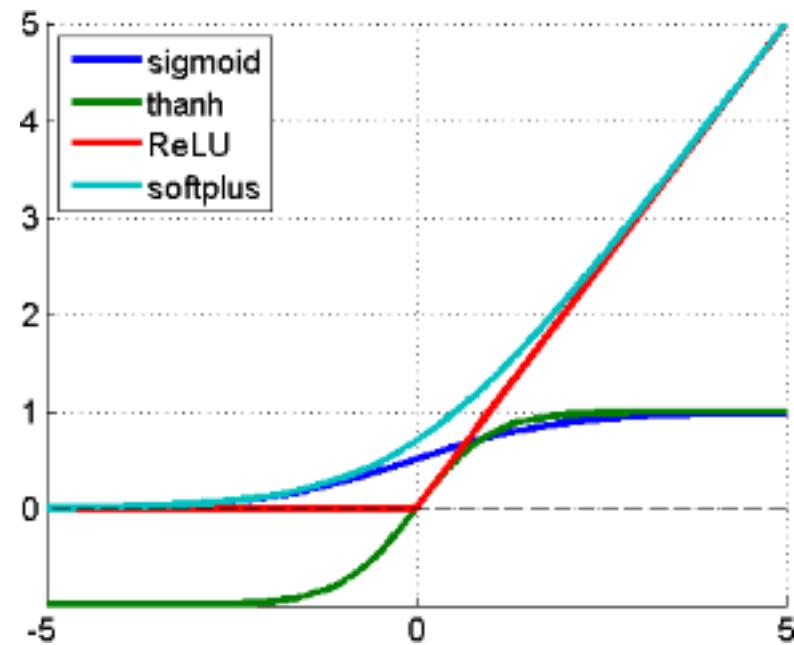
We will have to add a third dimension to create a linearly separable model which is easy to deal with

Data in \mathbb{R}^3 (Separable)



Activation Functions

- They are used to *convert an input signal* of a node in an artificial neural network to an *output signal*
- That output signal now is used as an input in the next layer in the stack
- Activation functions introduce *non-linear properties* to our network
- A neural network without an activation function is essentially just a *linear regression model*
- The activation function does non-linear transformation to the input making it capable to *learn and perform more complex tasks*



Types of Activation Functions

*Identit
y*

*Binary
Step*

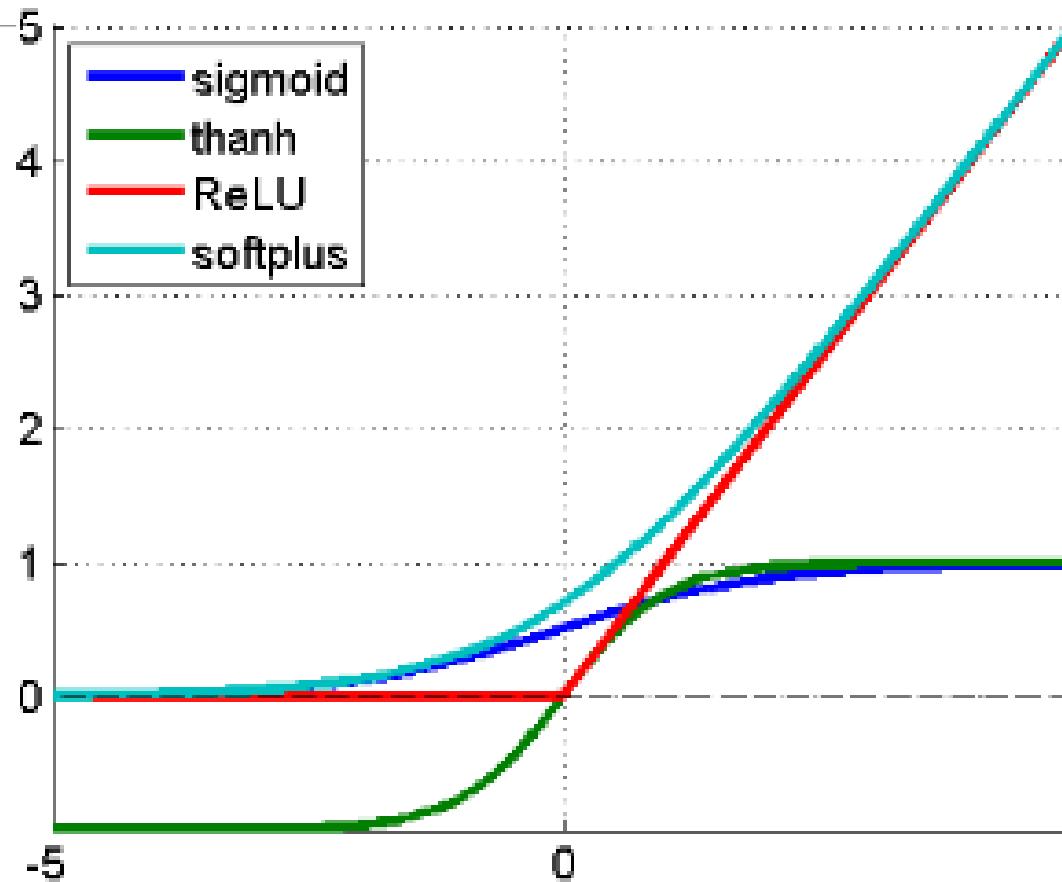
*Sigmoi
d*

*Tan
h*

ReLU

*Leaky
ReLU*

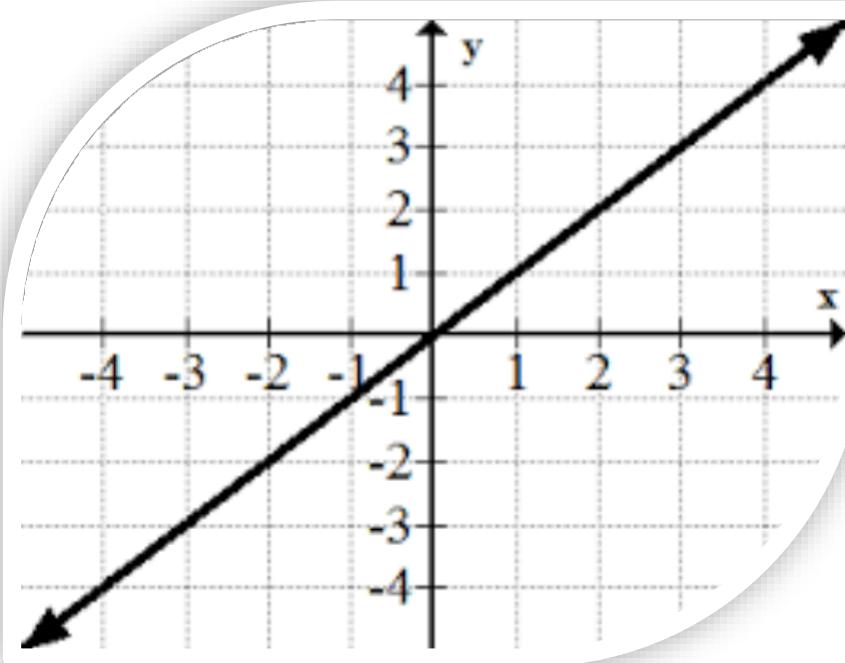
*Softma
x*



Identity Function

$$f(x) = x$$

- A straight line function where activation is proportional to input
 - No matter how many layers we have, if all of them are linear in nature, the final activation function of the last layer will be nothing but just a linear function of the input of the first layer
-
- We use a linear function to solve a linear regression problem
 - Range: $(-\infty, \infty)$

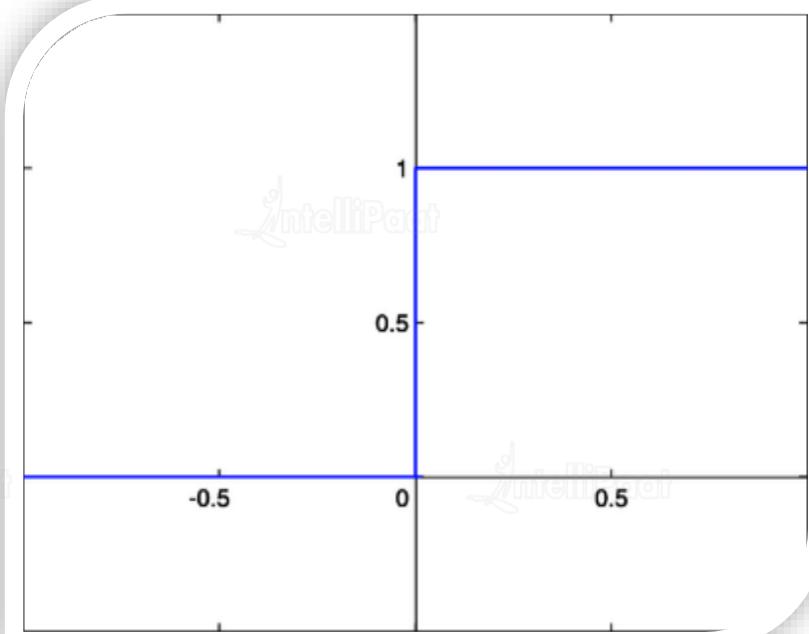


Binary Step Function

$$f(x) = \Phi$$

0 for $x < 0$
1 for $x = > 0$

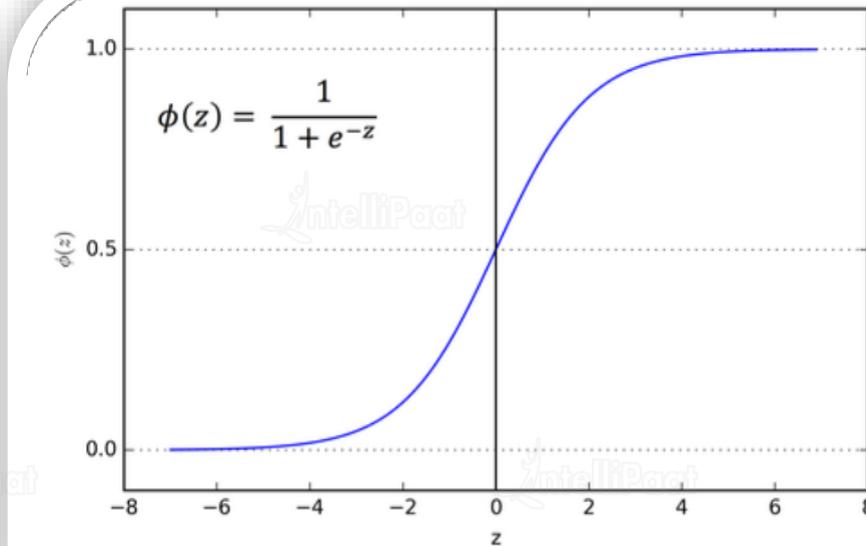
- It is also known as the Heaviside step function, or the unit step function, which is usually denoted by H or θ , is a discontinuous function
- Its value is 0 for the negative argument and 1 for the positive argument
- It depends on the threshold value we define
- We use the binary step function to solve a binary classification problem
- Range: $(0,1)$



Sigmoid Function

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

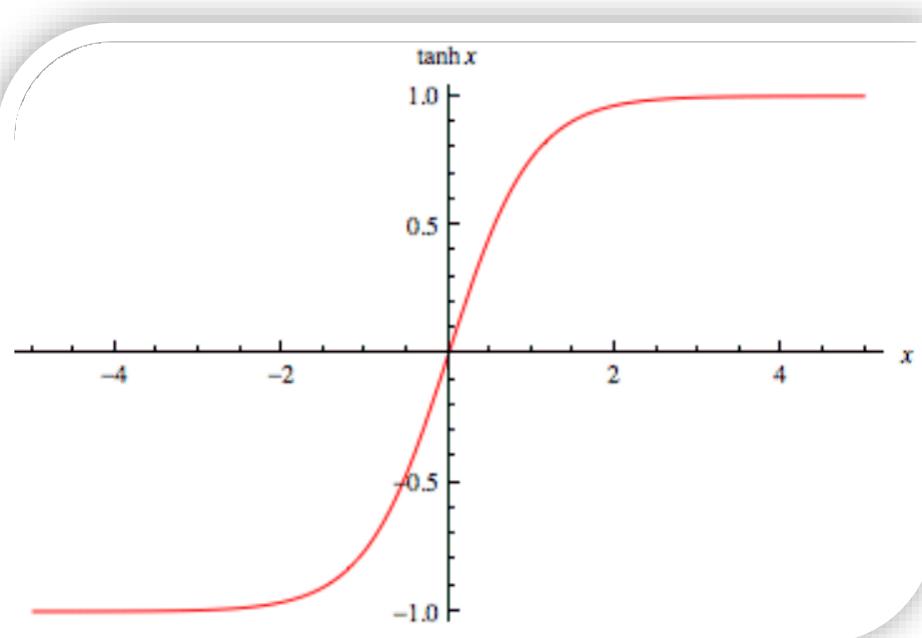
- The sigmoid function is an activation function where it scales values between 0 and 1 by applying a threshold
- When we apply the weighted sum in the place of x , the values are scaled in between 0 and 1
- Large negative numbers are scaled toward 0, and large positive numbers are scaled toward 1
- Range: (0,1)



Tanh Function

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

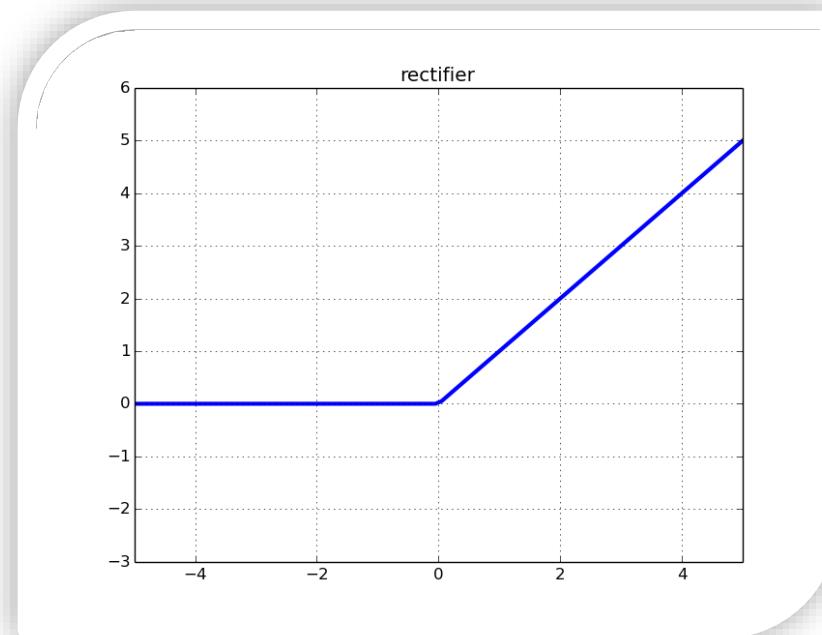
- It is a hyperbolic trigonometric function
- The Tanh activation works almost always better than sigmoid functions as optimization is easier in this method
- The advantage of Tanh is that it can deal more easily with negative numbers
- It is actually a mathematically shifted version of the sigmoid function
- Range: (-1,1)



ReLU Function

$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$

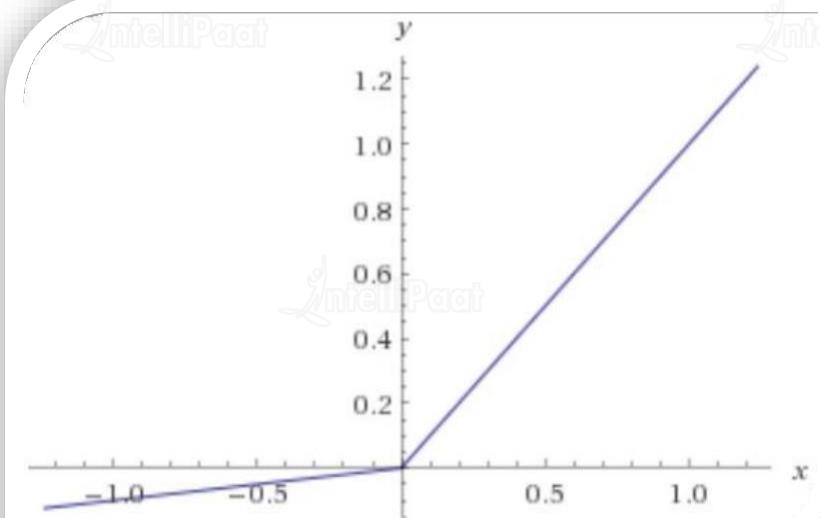
- ReLU stands for rectified linear unit
- It is the most widely used activation function
- It is primarily implemented in Hidden Layers of the neural network
- This function allows only the maximum values to pass during the front propagation as shown in the graph below
- Range: $(0, \infty)$



Leaky ReLU Function

$$f(x) = \begin{cases} 0.01x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$

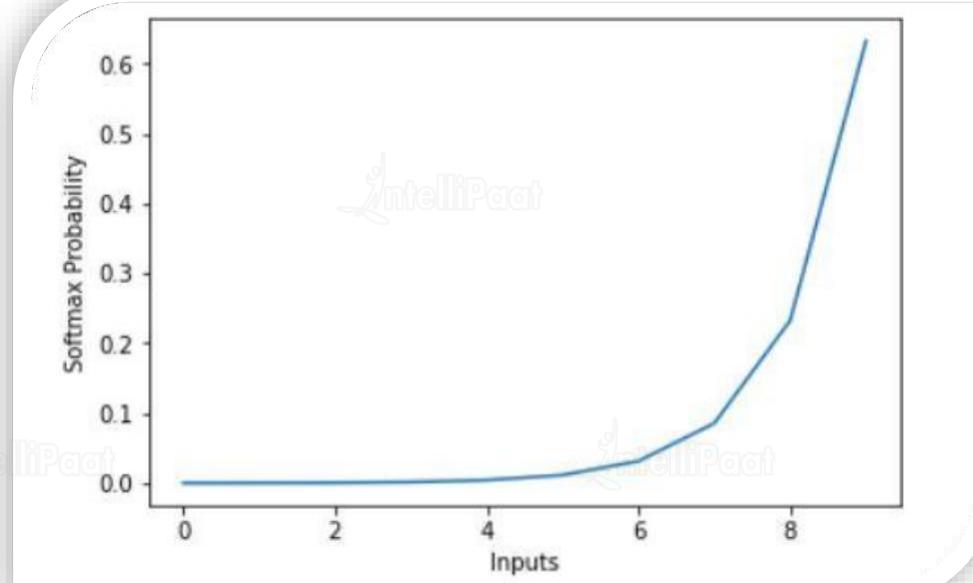
- Leaky ReLU allows a small negative value to pass during the back propagation if we have a dead ReLU problem
- This eventually activates the neuron and brings it down
- Range: $(-\infty, \infty)$



Softmax Function

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}}, j = 1, 2, \dots, K$$

- The Softmax function is used when we have multiple classes
- It is useful for finding out the class which has the max. probability
- The Softmax function is ideally used in the Output Layer of the classifier where we are actually trying to attain the probabilities to define the class of each input
- Range: (0,1)

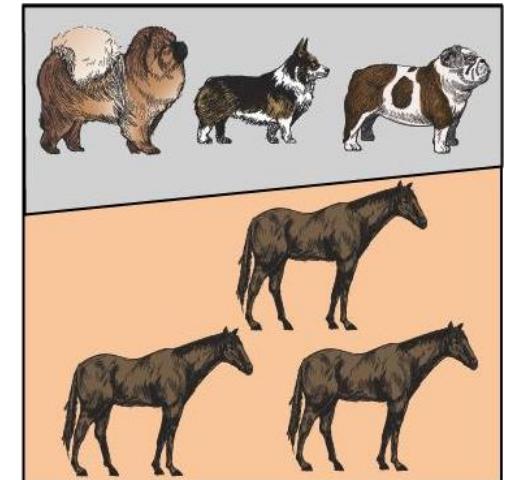
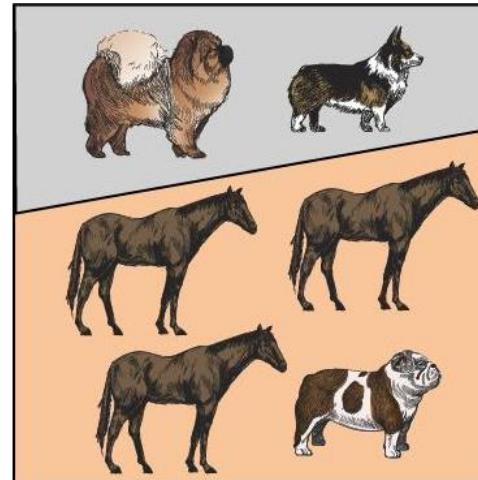
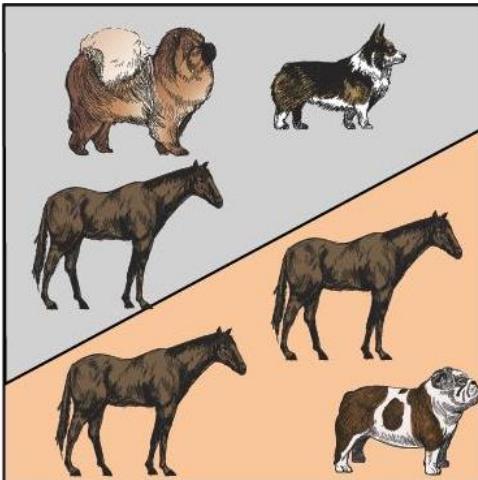
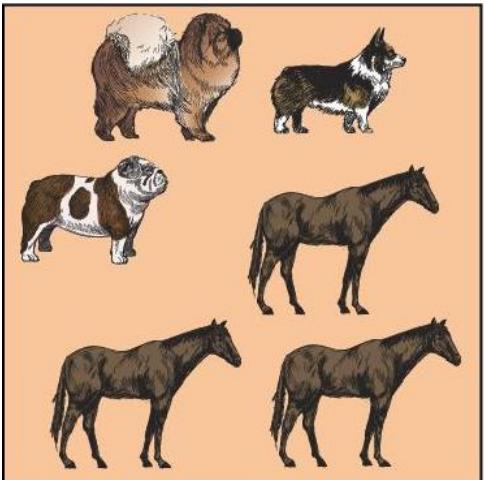




Got bored again? Let us get back to perceptrons and try to understand them in a better way!

Training a Perceptron

By training a perceptron, we try to find a line, plane, or some hyperplane which can accurately separate two classes by adjusting weights and biases

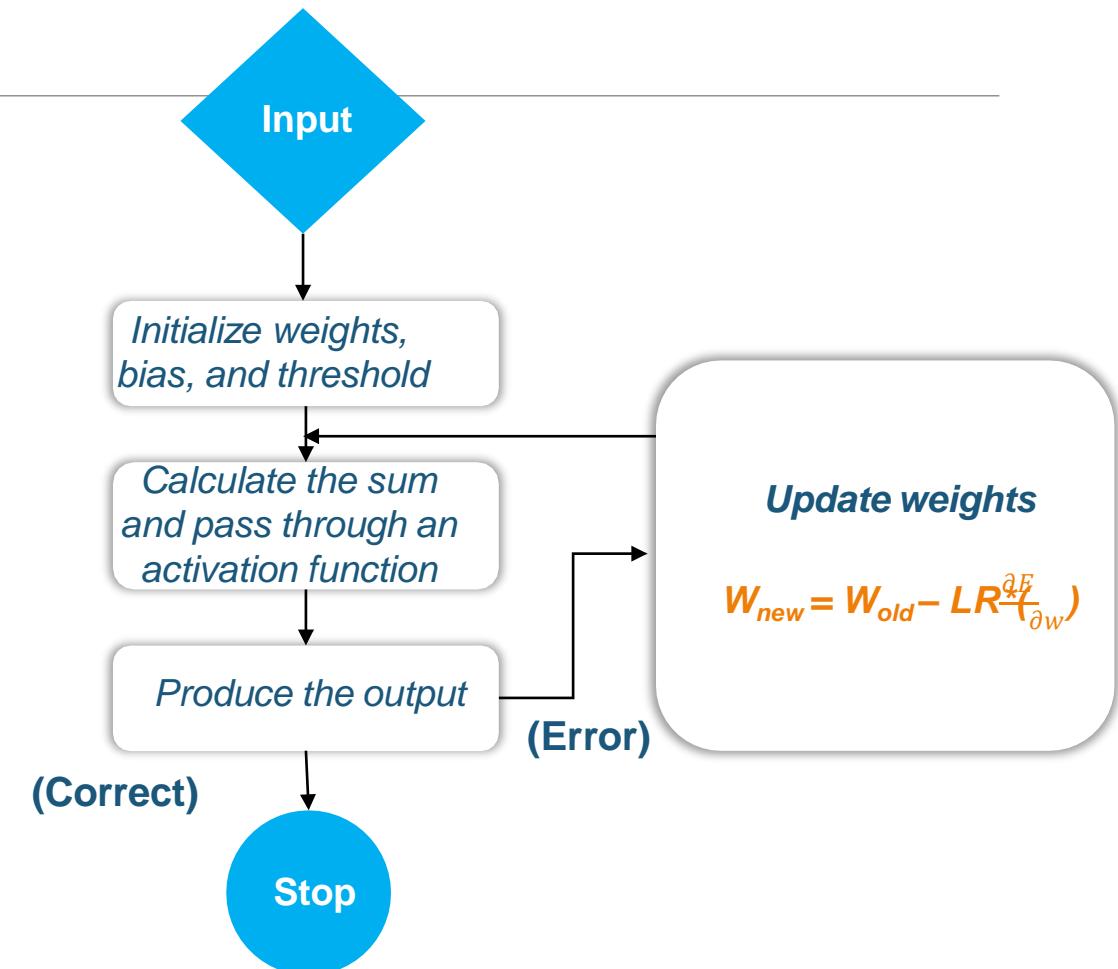
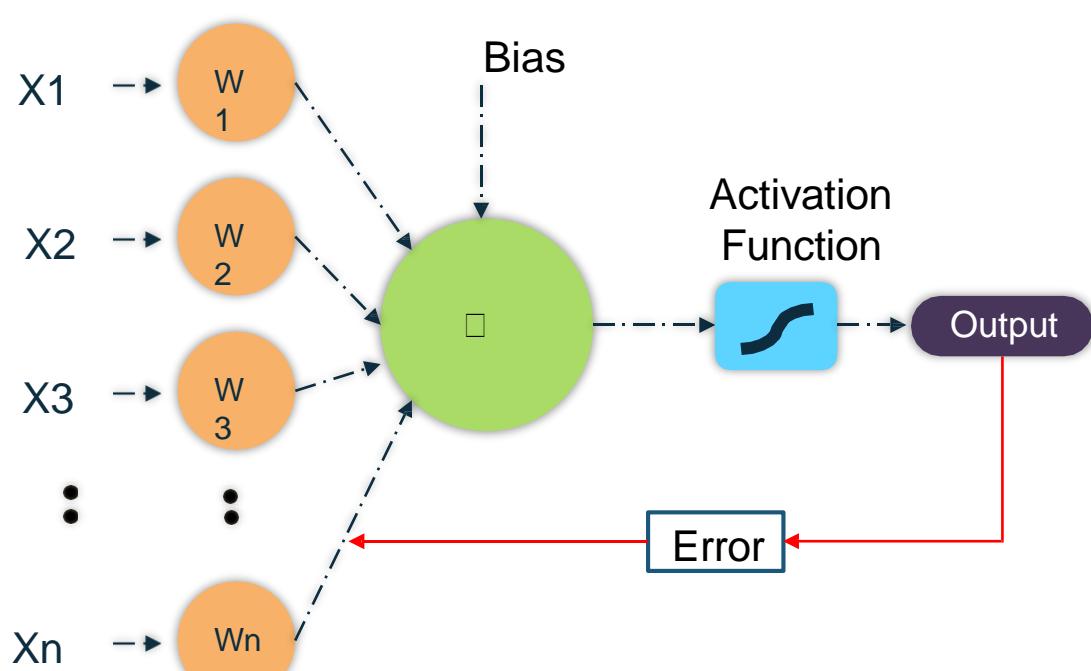


Error = 2

Error = 1

Error = 0

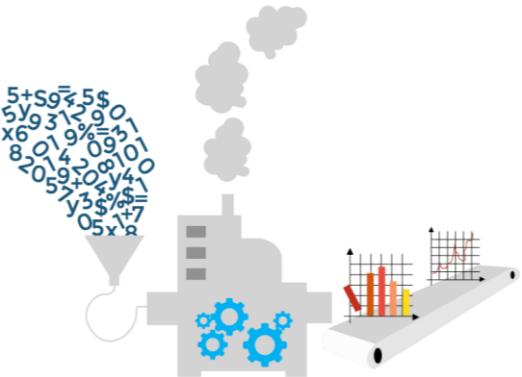
Perceptron Training Algorithm



Benefits of Using Artificial Neural Networks



Organic Learning



Non-linear Data Processing



Fault Tolerance



Self-repairing



Let us now move toward Deep
Learning frameworks!

Deep Learning Frameworks

These Deep Learning libraries help in implementing artificial neural networks



TensorFlow

Keras

PyTorch

DL4J

MXNet



TensorFlow is an open-source software library for high-performance numerical computations



Developed by Google

TensorFlow

Keras

PyTorch

DL4J

MXNet

Google Translate



Natural language
processing

Text classification

Forecasting

Tagging

TensorFlow

Keras

PyTorch

DL4J

MXNet



Tensor
Board



Used for visualizing TensorFlow computations and graphs

TensorFlow
Serving



Used for rapid deployment of new algorithms/experiments while retaining the same server architecture and APIs

TensorFlow

Keras

PyTorch

DL4J

MXNet



A high-Level API which can run on top of
TensorFlow, Theano, or CNTK

TensorFlow

Keras

PyTorch

DL4J

MXNet



A recurrent neural network

A convolutional neural network

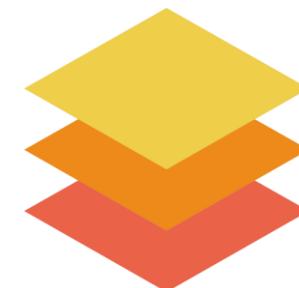
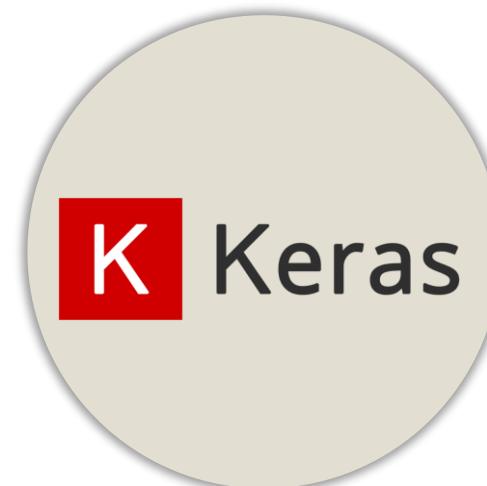
TensorFlow

Keras

PyTorch

DL4J

MXNet



Stacks layers on top

TensorFlow

Keras

PyTorch

DL4J

MXNet



A scientific computing framework developed by
Facebook

TensorFlow

Keras

PyTorch

DL4J

MXNet

PyTorch



'Pythonic' in nature

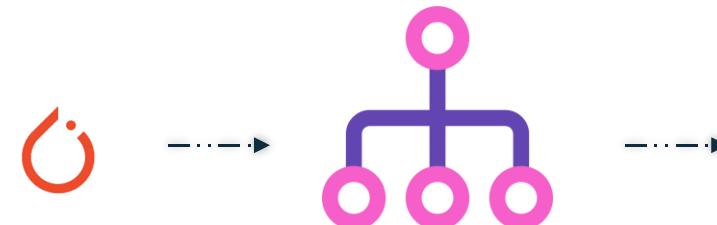
TensorFlow

Keras

PyTorch

DL4J

MXNet



Offers dynamic computational
graphs

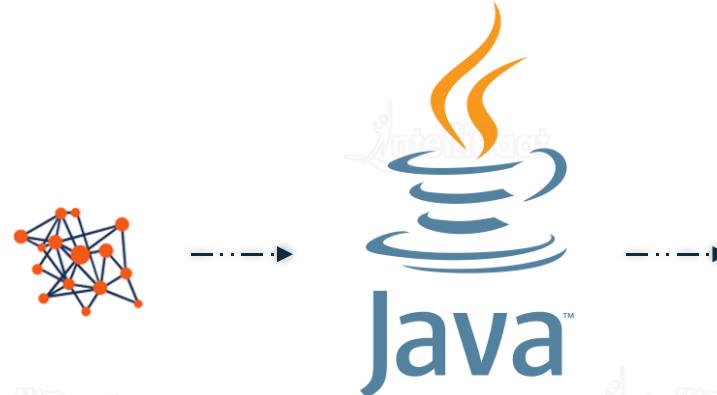
TensorFlow

Keras

PyTorch

DL4J

MXNet



A Deep Learning programming
library written for Java

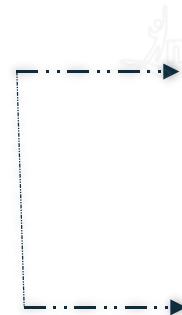
TensorFlow

Keras

PyTorch

DL4J

MXNet



TensorFlow

Keras

PyTorch

DL4J

MXNet



Image recognition

Fraud detection

Text mining

Parts of speech
tagging

Natural language
processing

TensorFlow

Keras

PyTorch

DL4J

MXNet

mxnet



Developed by Apache
Software Foundation

TensorFlow

Keras

PyTorch

DL4J

MXNet

mxnet

mx



R

Scala

julia

TensorFlow

Keras

PyTorch

DL4J

MXNet



Imaging

Speech
recognition

Forecasting

NLP

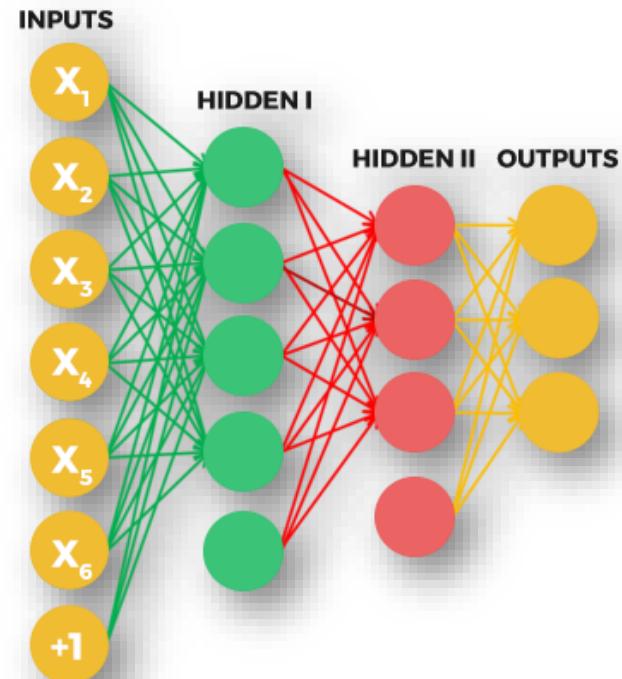
What Are Tensors?

A tensor is a multi-dimensional array in which data is stored

Tensor is given as an input to a neural network

5	1	8
3	5	4
6	9	0
4	7	2
5	9	3

Tensor



Tensor Rank

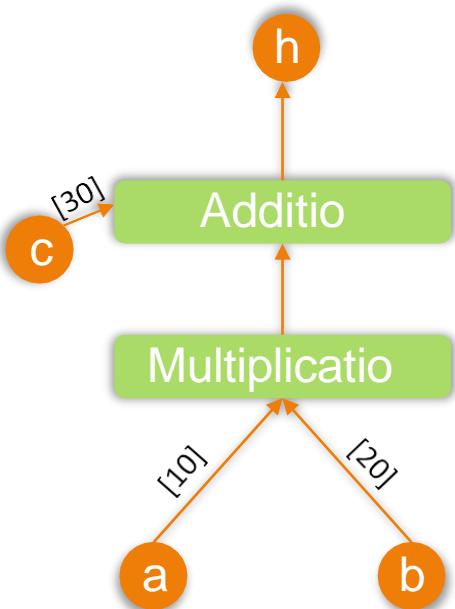
Tensor rank represents the dimension of the n -dimensional array

Rank	Math Entity	Example
0	Scalar (magnitude only)	$s = 483$
1	Vector (magnitude and direction)	$v = [1.1, 2.2, 3.3]$
2	Matrix (table of numbers)	$m = [1, 2, 3], [4, 5, 6], [7, 8, 9]$
3	3-Tensor (cube of numbers)	$t = [[[2], [4], [6]], [[8], [10], [12]], [[14], [16], [18]]]$
n	n -Tensor

Computational Graph

Computation is done in the form of a graph

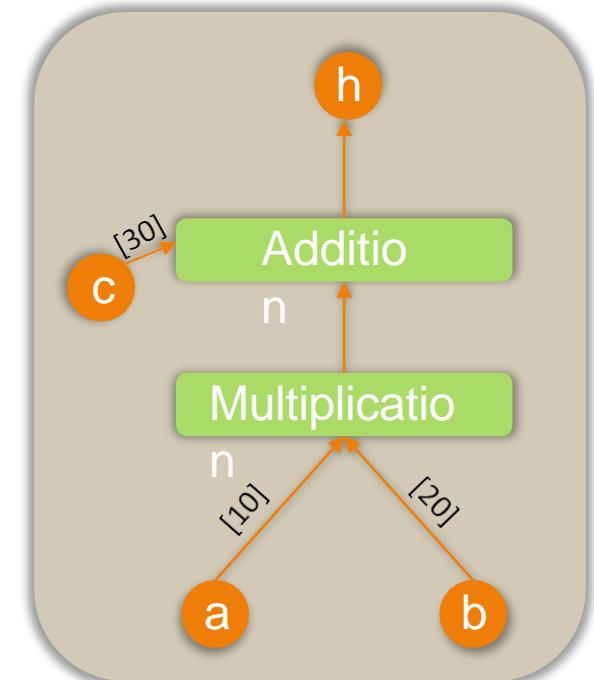
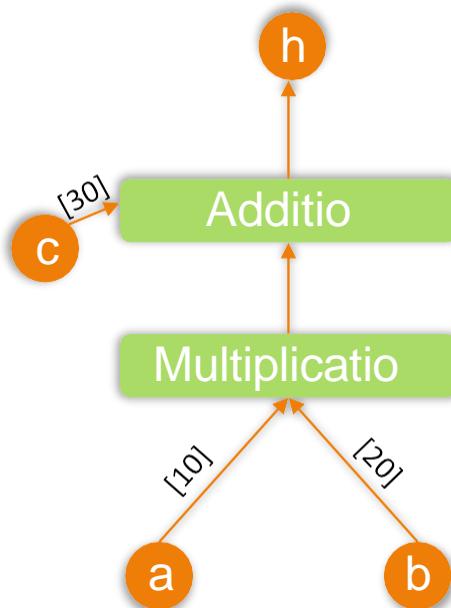
```
a = 10  
b = 20  
c = 30  
 $h = (a * b) + c$ 
```



Computational Graph

The computational graph is executed inside a session

```
a = 10  
b = 20  
c = 30  
 $h = (a * b) + c$ 
```

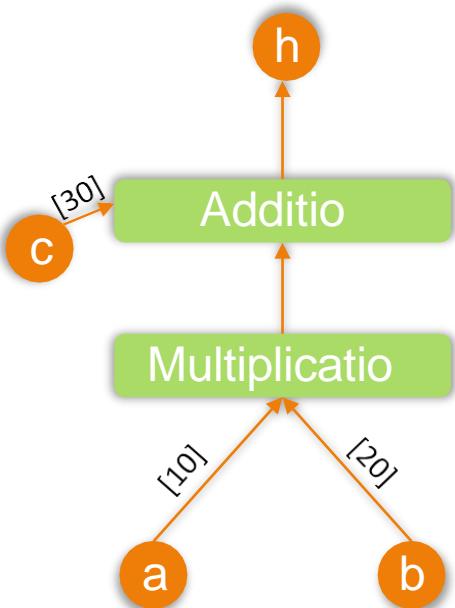


Session

Computational Graph

The computational graph is executed inside a session

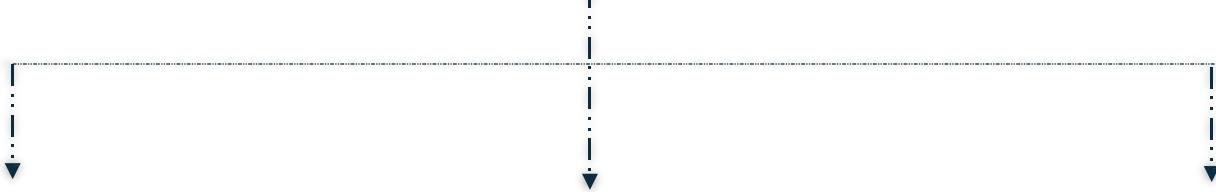
```
a = 10  
b = 20  
c = 30  
 $h = (a * b) + c$ 
```



Node -> Mathematical operation

Edge -> Tensor

Program Elements in TensorFlow



Constant

Placeholder

Variable

Constant

Constants are program elements whose values do not change

Placeholder

```
a=tf.constant(10)
```

```
b=tf.constant(20)
```

Variable

Constant

A placeholder is a program element to which we can assign data at a later time

Placeholder

```
x=tf.placeholder(tf.float32)
```

```
y=tf.placeholder(tf.string)
```

Variable

Constant

A variable is a program element which allows us to add new trainable parameters to the graph

Placeholder

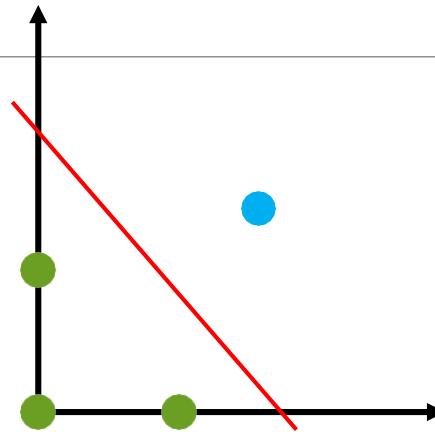
Variable

`W=tf.Variable([3],tf.float32)`

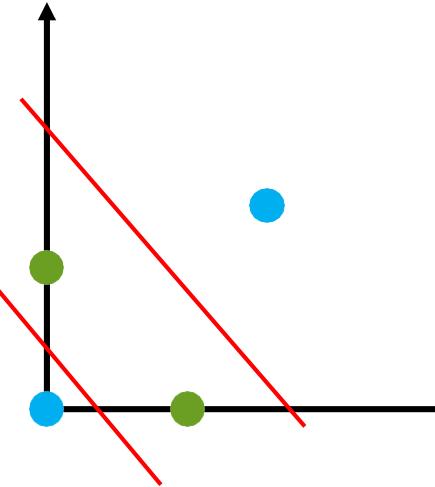
`b=tf.Variable([0.4],tf.float32)`

Limitations of a Single-layer Perceptron

- A single-layer perceptron can only learn linearly separable problems
- If the problem is not linearly separable, the learning process of a perceptron will never reach a point where all points are classified correctly
- Boolean 'AND' and 'OR' functions are linearly separable, whereas Boolean 'XOR' is not



Boolean 'AND': Linearly Separable



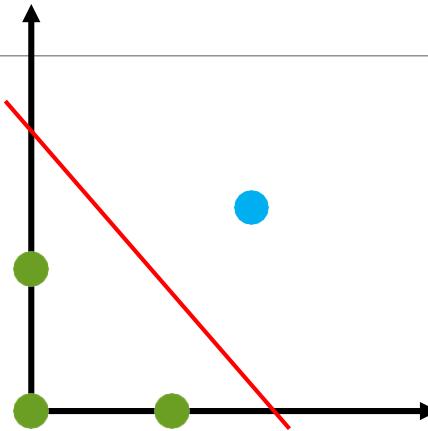
Boolean 'XOR': Non-Linearly Separable

Limitations of a Single-layer Perceptron

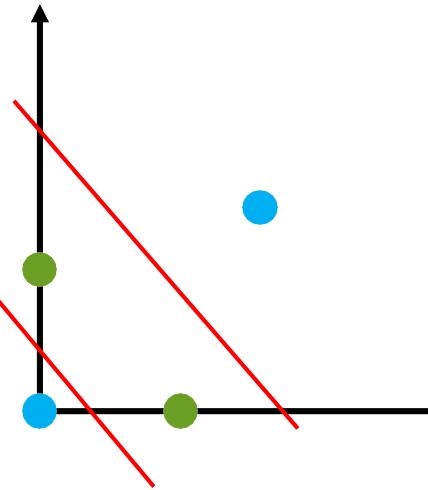
- A single-layer perceptron can only learn linearly separable problems
- If the problem is not linearly separable, the learning process of a perceptron will never reach a point where all points are classified correctly
- Boolean 'AND' and 'OR' functions are linearly separable, whereas Boolean 'XOR' is not



For solving this problem, we can use a multi-layer perceptron



Boolean 'AND': Linearly Separable



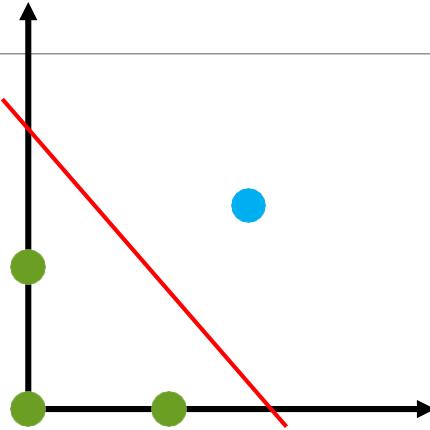
Boolean 'XOR': Non-Linearly Separable

Limitations of a Single-layer Perceptron

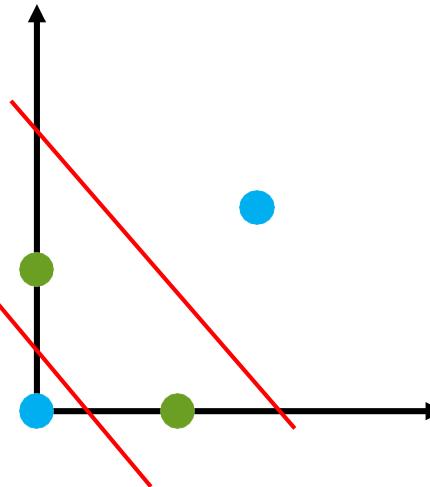


For solving this problem, we can use a multi-layer perceptron

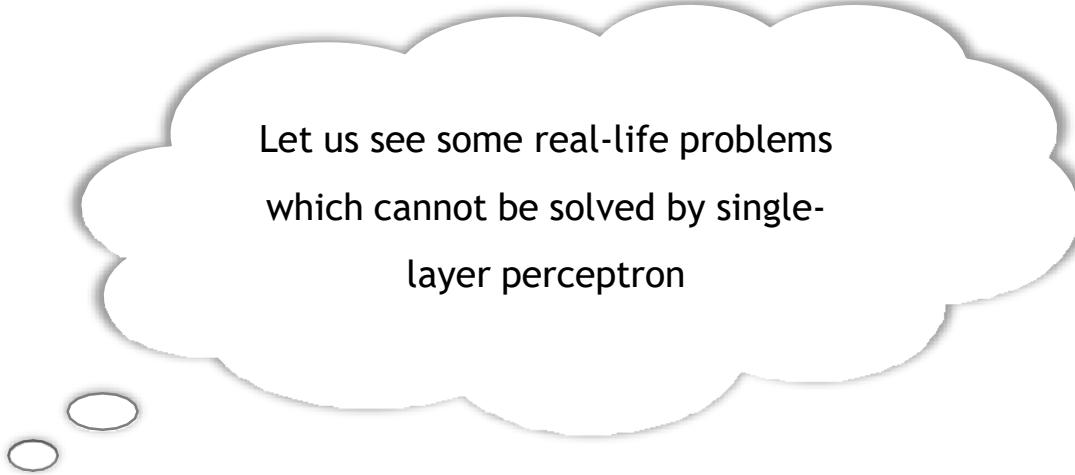
- A single-layer perceptron won't be able to solve complex problems such as *Image Classification*
- In such kinds of problems, the *dimensionality* and *complexity* of the classification is very high



Boolean 'AND': Linearly Separable



Boolean 'XOR': Non-Linearly Separable



Let us see some real-life problems
which cannot be solved by single-
layer perceptron

Use Case 1

Complex problems that involve a lot of parameters cannot be solved by a single-layer perceptron

- Consider a case where you own an e-commerce firm. You have planned to increase traffic on your site by providing a special discount on the products and services. Now, you want to create awareness among people regarding this end-season sale by marketing on different portals like:
 - Google ads
 - Personal emails
 - Sales advertisements on relevant sites
 - YouTube ads
 - Ads on different sites
 - linkedin
 - Blogs and so on
- This task is too complex for a human to analyze, as you can see that the number of parameters is quite high
- Let us try to solve it using Deep Learning

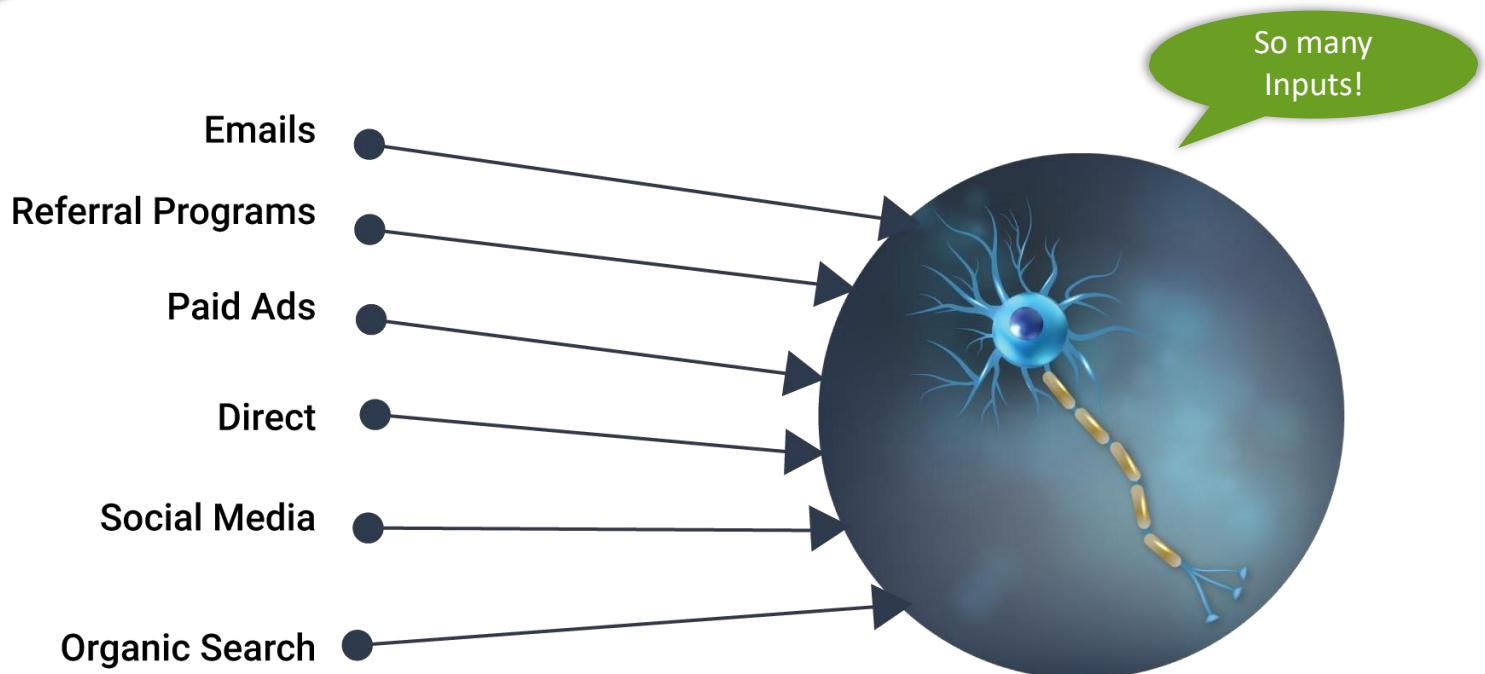
Use Case 1

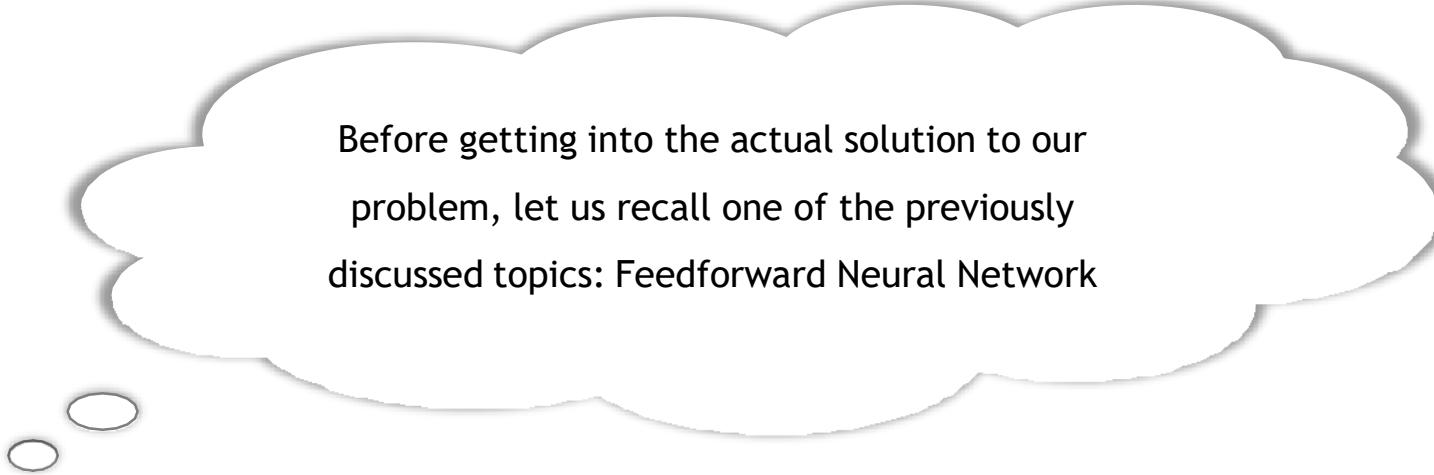


- You can either use just one platform for publicity or use a variety of them
- Each of them has its own advantages and disadvantages, but lots of factors would have to be considered
- The increased traffic on your portal or the number of sales that would happen is dependent on different categorical inputs, their sub-categories, and their parameters

Computing and calculating profit in terms of popularity and sales, from so many inputs and their sub-categories, is not possible just through one perceptron

So now, you know why a single perceptron cannot be used for complex non-linear problems



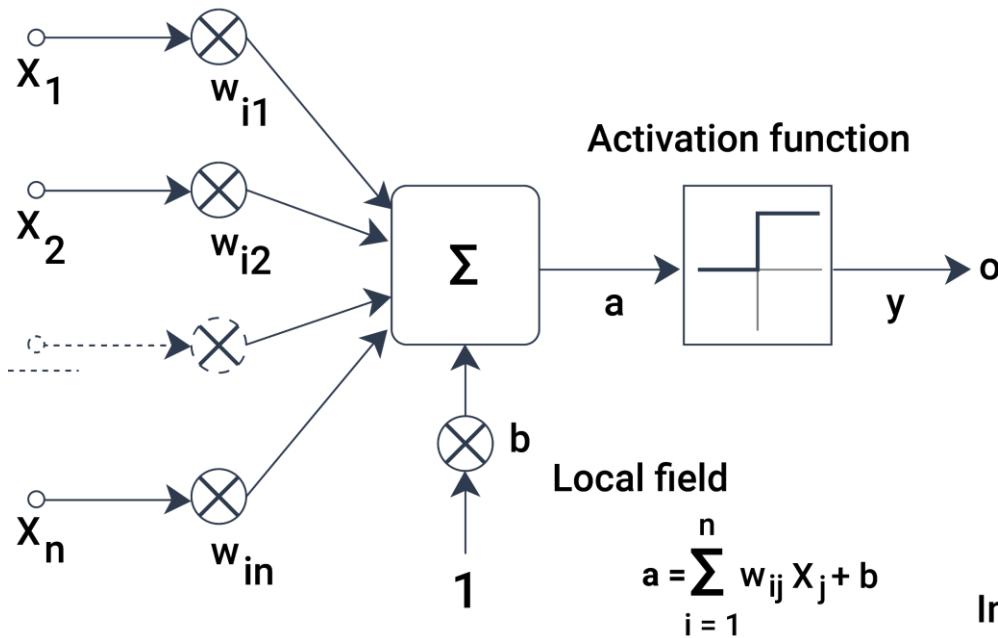


Before getting into the actual solution to our problem, let us recall one of the previously discussed topics: Feedforward Neural Network

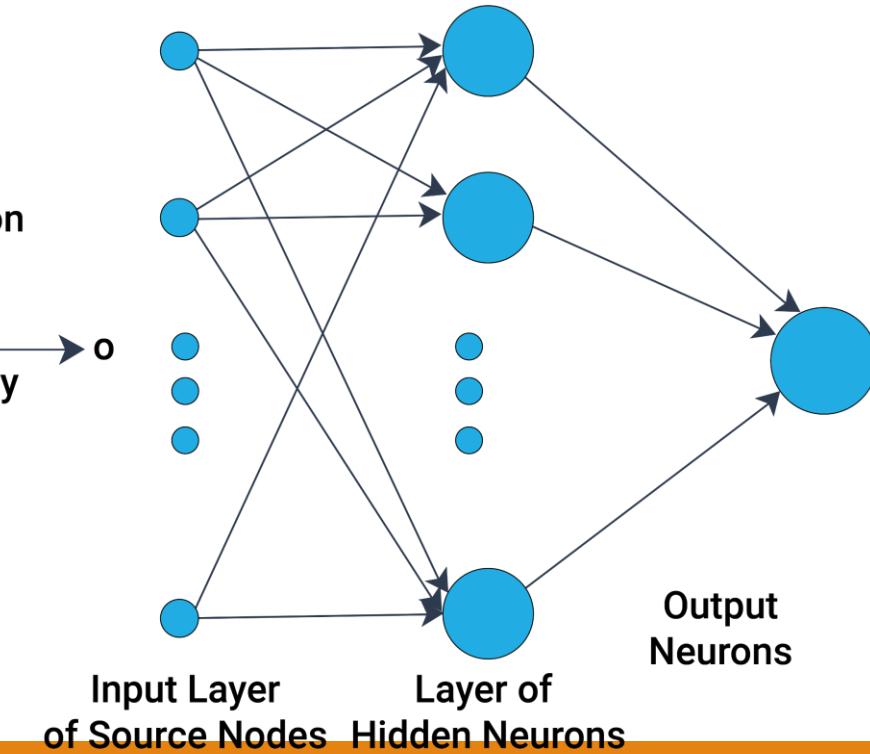
Feedforward Neural Network

Feedforward neural network is the most simple artificial neural network containing multiple nodes arranged in multiple layers.
Adjacent layer nodes have connections or edges where all connections are weighted

A. Neuron



B. Feedforward Network



Feedforward Neural Network

Feedforward neural network is the most simple artificial neural network containing multiple nodes arranged in multiple layers.
Adjacent layer nodes have connections or edges where all connections are weighted

“A feedforward neural network can contain two kinds of nodes”

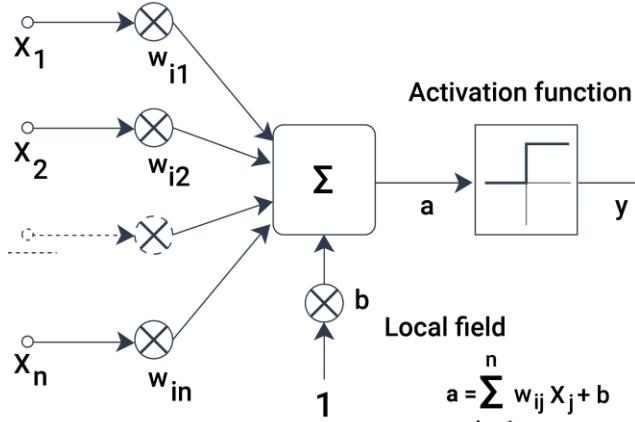
Monolayer

This is the simplest feedforward neural network that does not contain any hidden layers

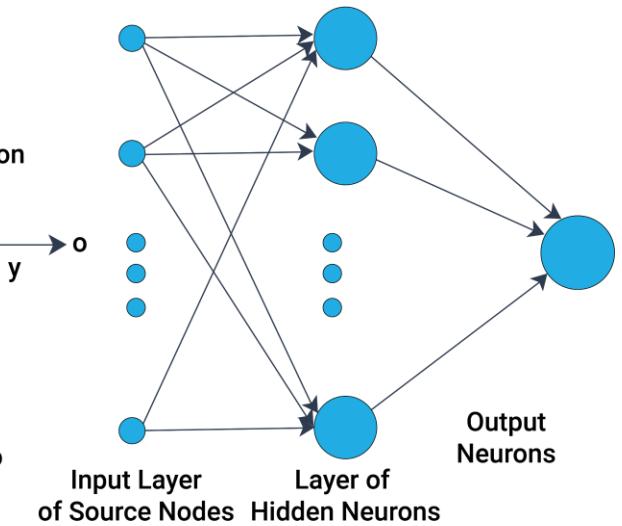
Multi-layer Perceptron

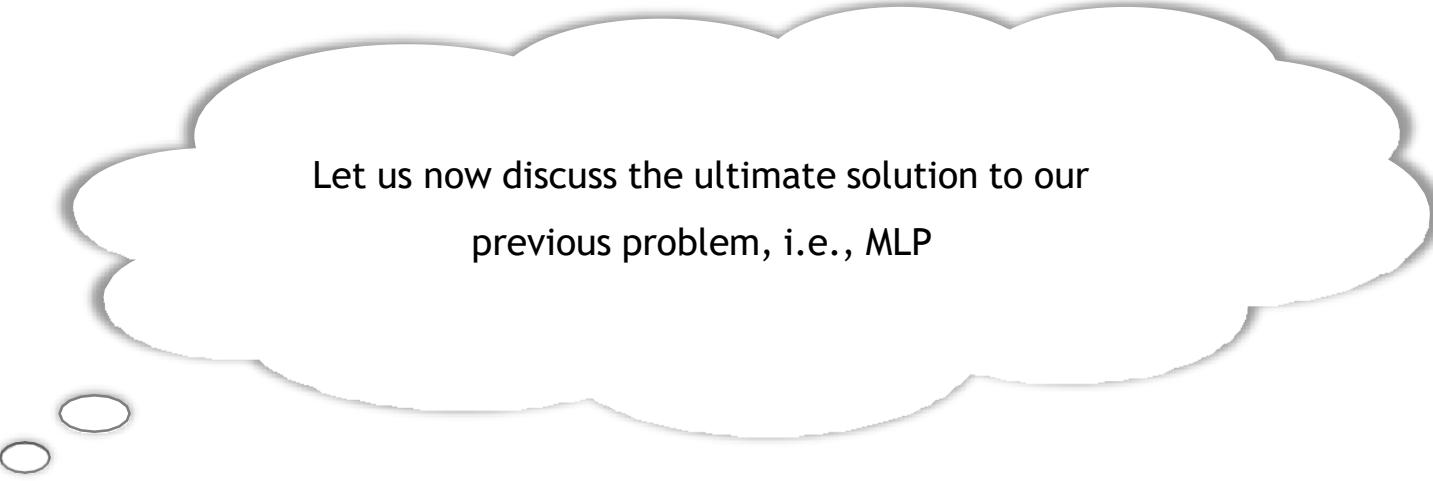
Multi-layer Perceptron (MLP) includes at least one hidden layer (except for one input layer and one output layer)

A. Neuron



B. Feedforward Network

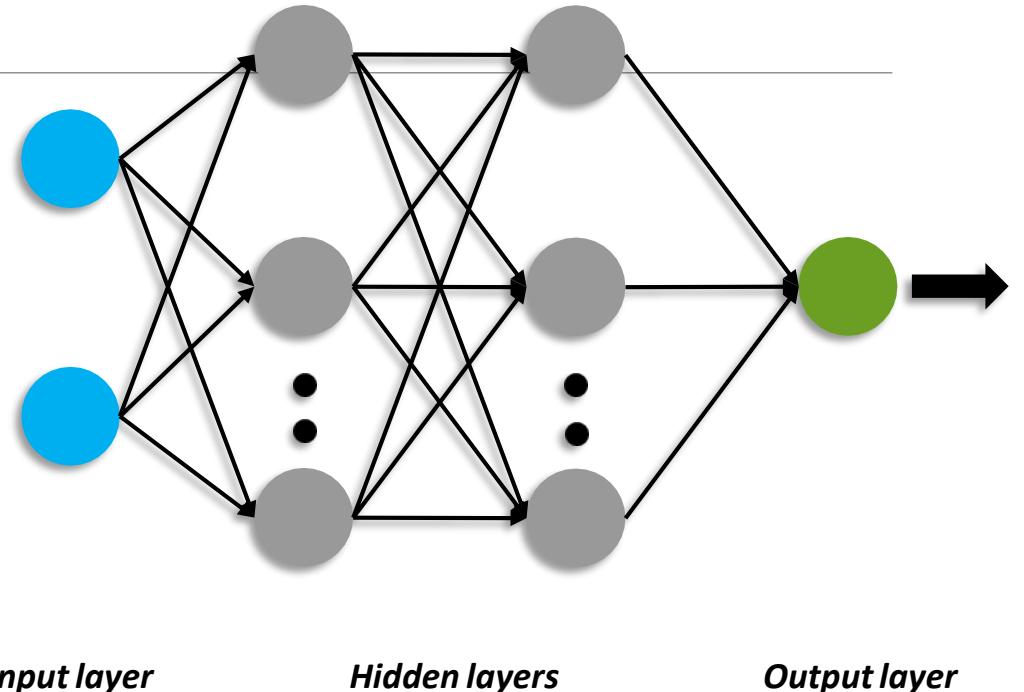




Let us now discuss the ultimate solution to our previous problem, i.e., MLP

Multi-layer Perceptron

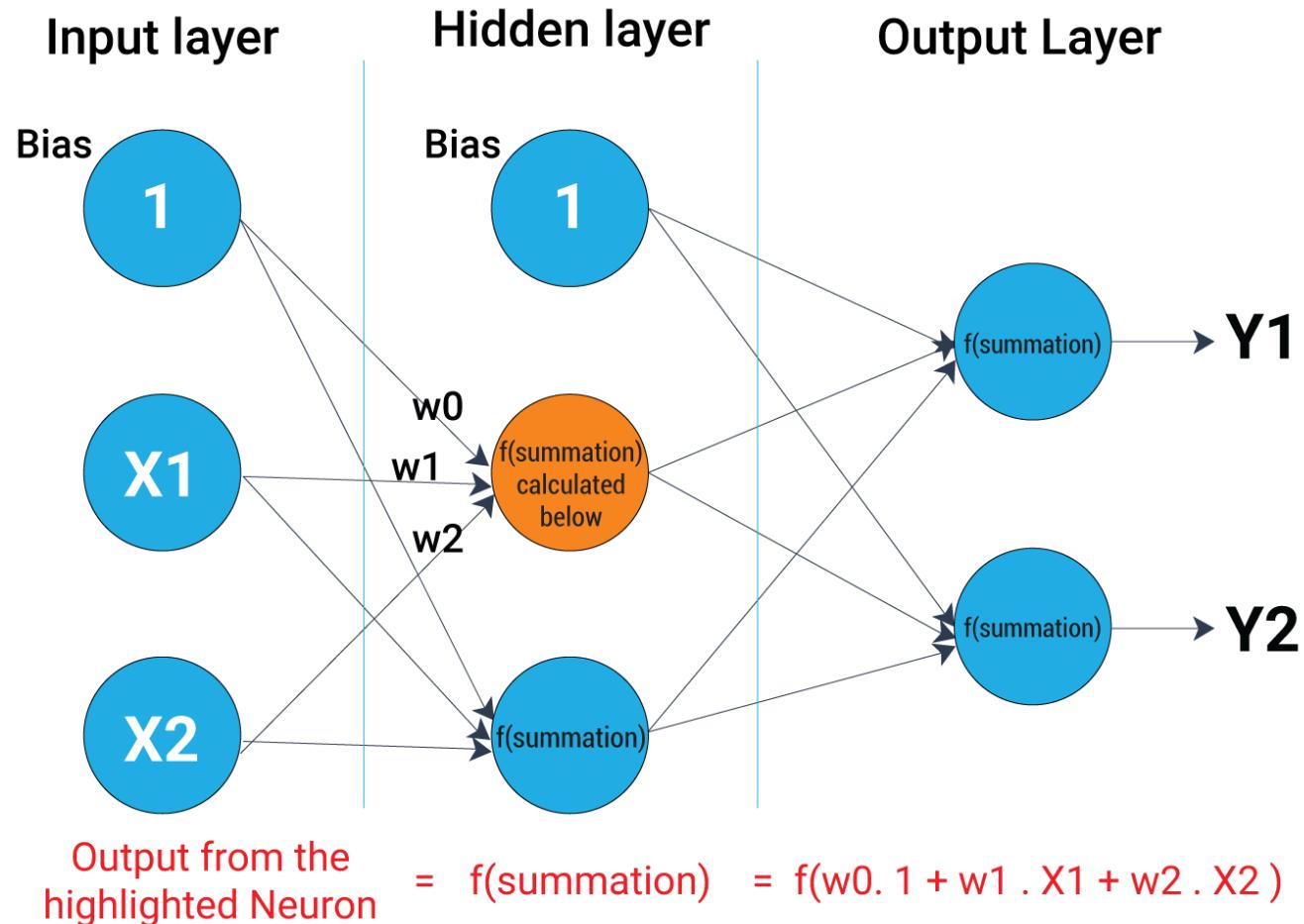
- A multi-layer perceptron (MLP) is a deep, artificial neural network
- It is composed of more than one perceptrons
- An MLP is comprised of:
 - An *input layer* to receive the signal
 - An *output layer* that makes a decision or prediction about the input
 - An *arbitrary number of hidden layers*
- Each node, apart from the input nodes, has a nonlinear activation function
- An MLP uses backpropagation as a supervised learning technique



MLP is widely used for solving problems that require supervised learning and research into computational neuroscience and parallel distributed processing. Such applications include speech recognition, image recognition, and machine translation

Multi-layer Perceptron

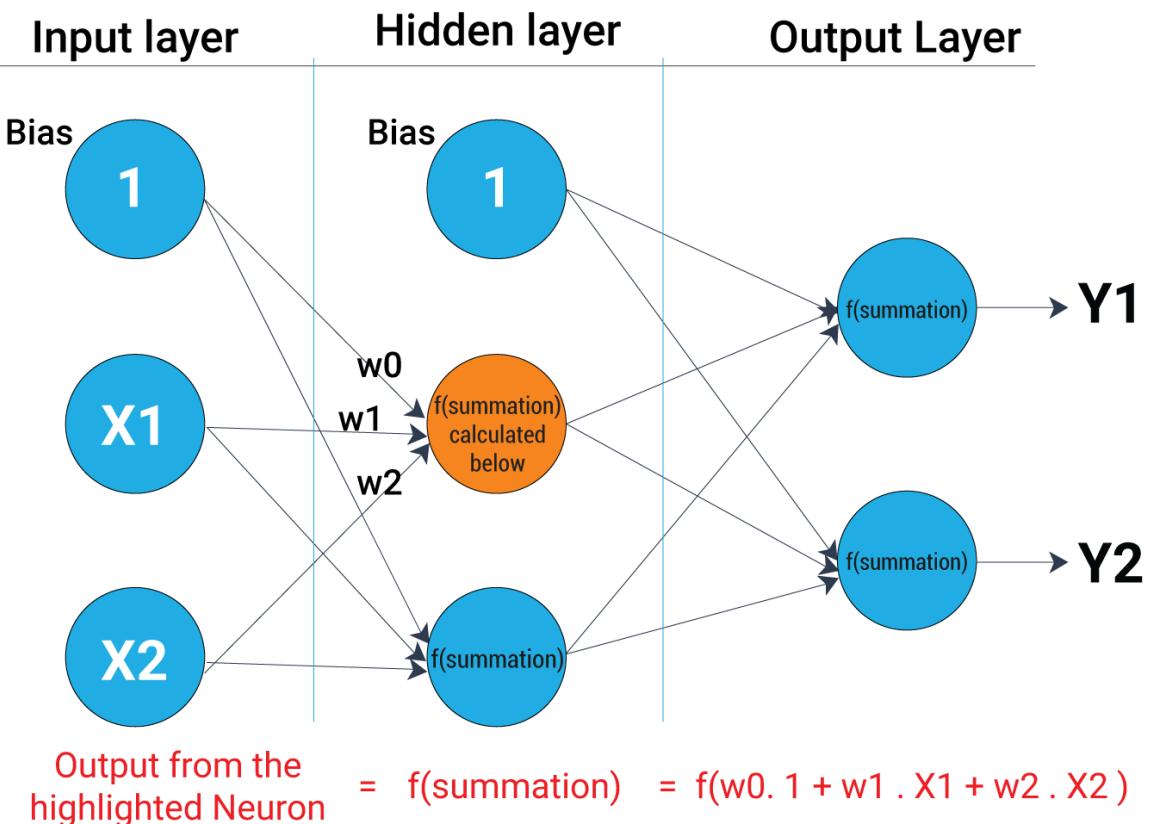
The figure shows a multi-layer perceptron with a single hidden layer. All connections have weights associated with them, but only three weights (w_0 , w_1 , and w_2) are shown in the figure



Multi-layer Perceptron

Input Layer:

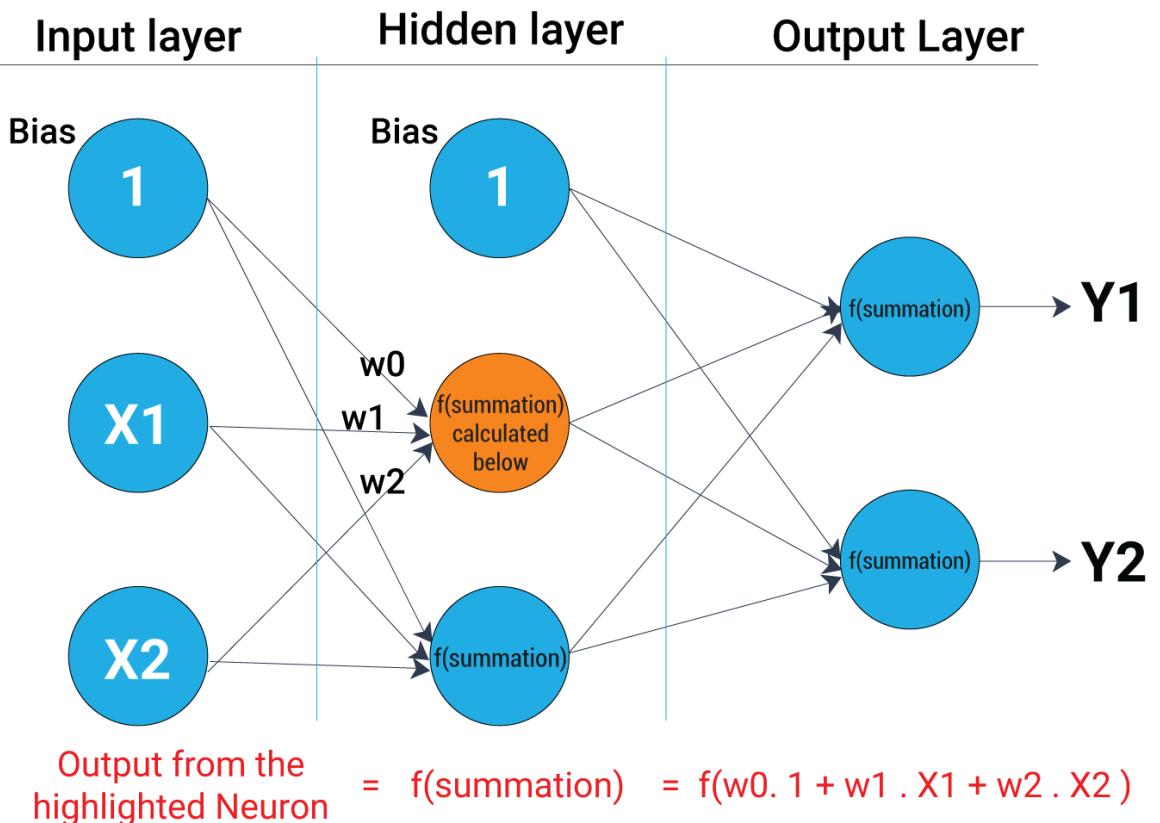
- It has three nodes
- Bias (offset) node has a value of 1
- The other two nodes take X_1 and X_2 as external inputs
- Outputs from nodes in the input layer are 1, X_1 , and X_2 , respectively, which are fed into the hidden layer



Multi-layer Perceptron

Hidden Layer:

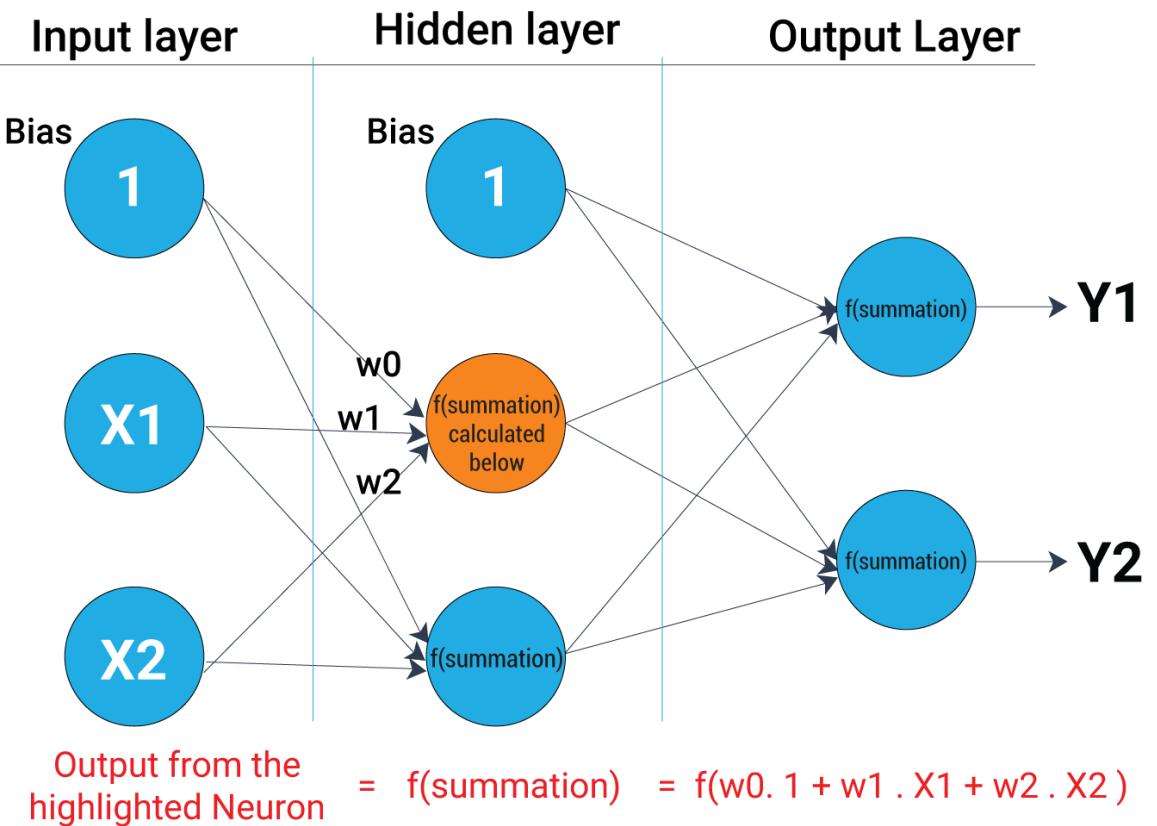
- It also has three nodes with the Bias node having an output of 1
- The output of the other two nodes in the hidden layer depends on the outputs from the input layer (1, X₁, and X₂) as well as the weights associated with the connections (edges)
- The figure shows the output calculation for one of the hidden nodes
- Similarly, the output from the other hidden node can be calculated
- Here, 'f' refers to the activation function. These outputs are then fed to the nodes in the output layer

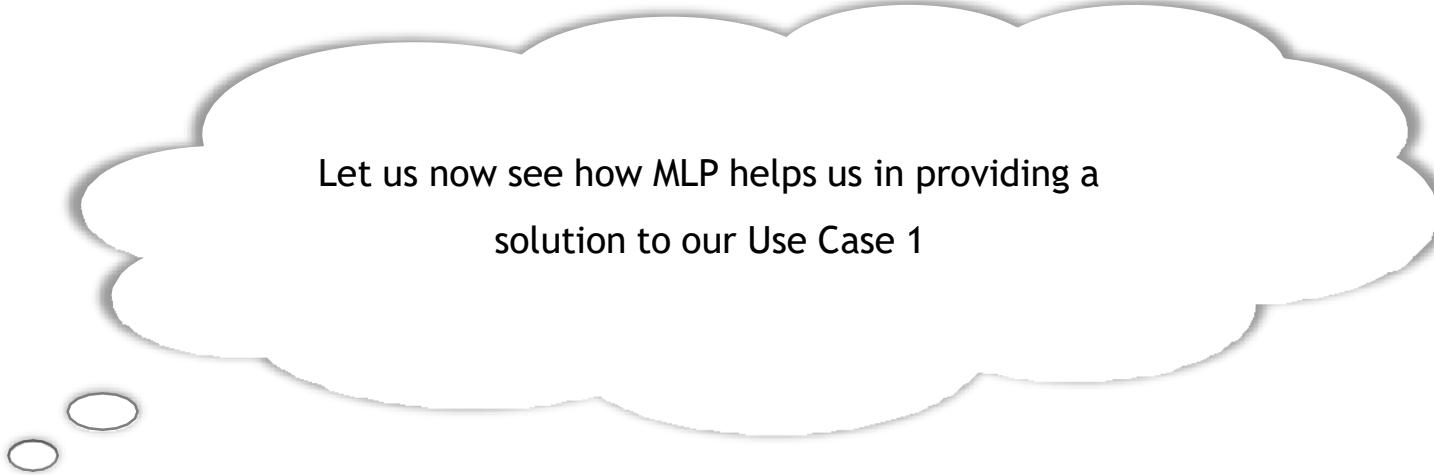


Multi-layer Perceptron

Output Layer:

- The output layer has two nodes which take inputs from the hidden layer and perform similar computations as shown for the highlighted hidden node
- Values calculated (Y_1 and Y_2) as a result of these computations act as outputs of the multi-layer perceptron

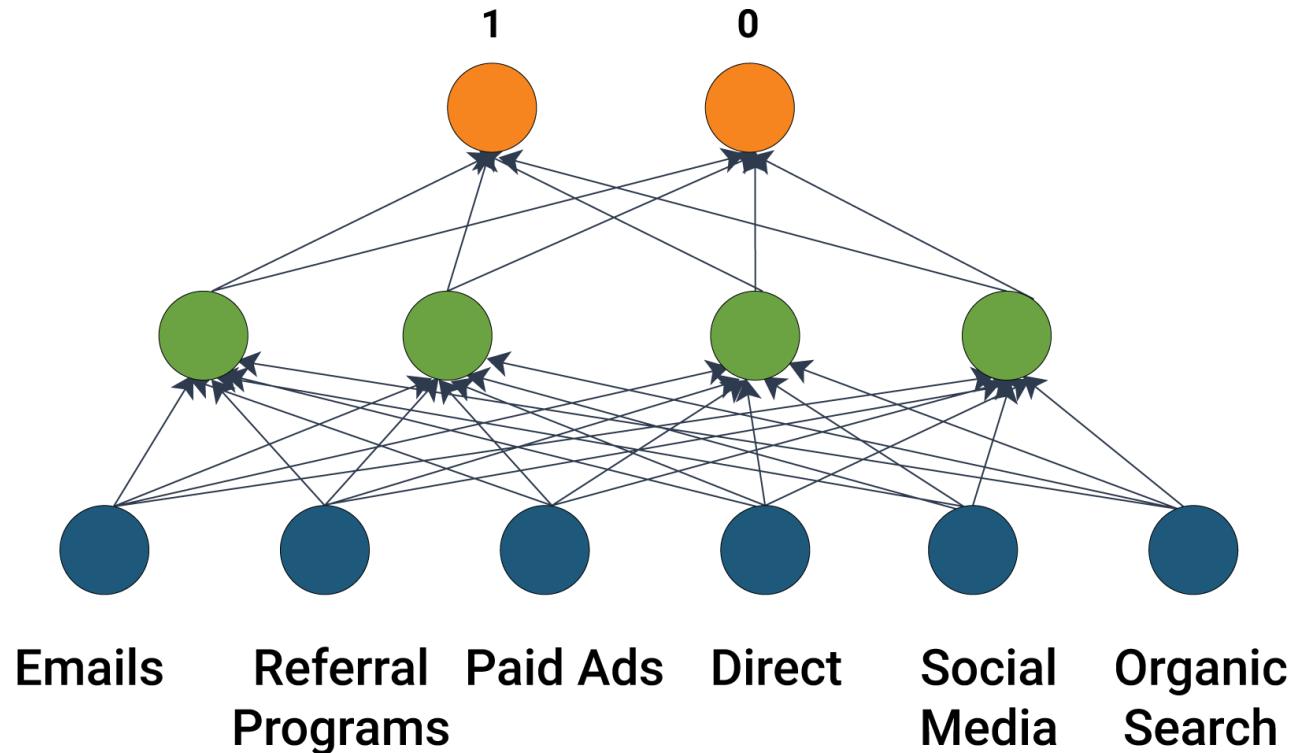


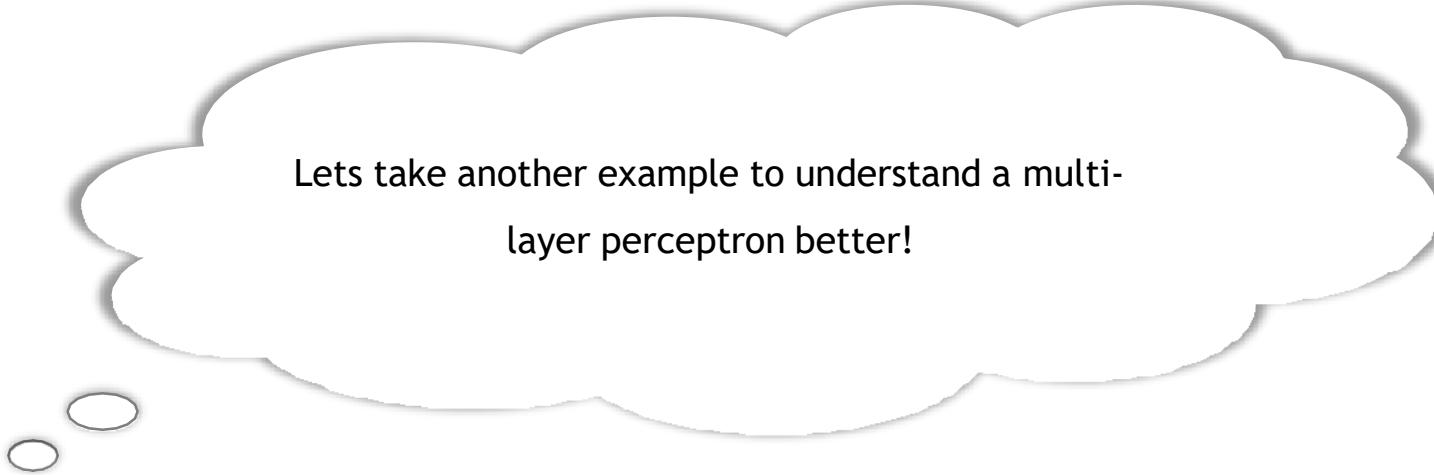


Let us now see how MLP helps us in providing a
solution to our Use Case 1

Use Case 1: Solution

-
- Every source behaves as an input to the neural network
 - Once all sources are fed into the system, the neural network calculates the output after the computation is done





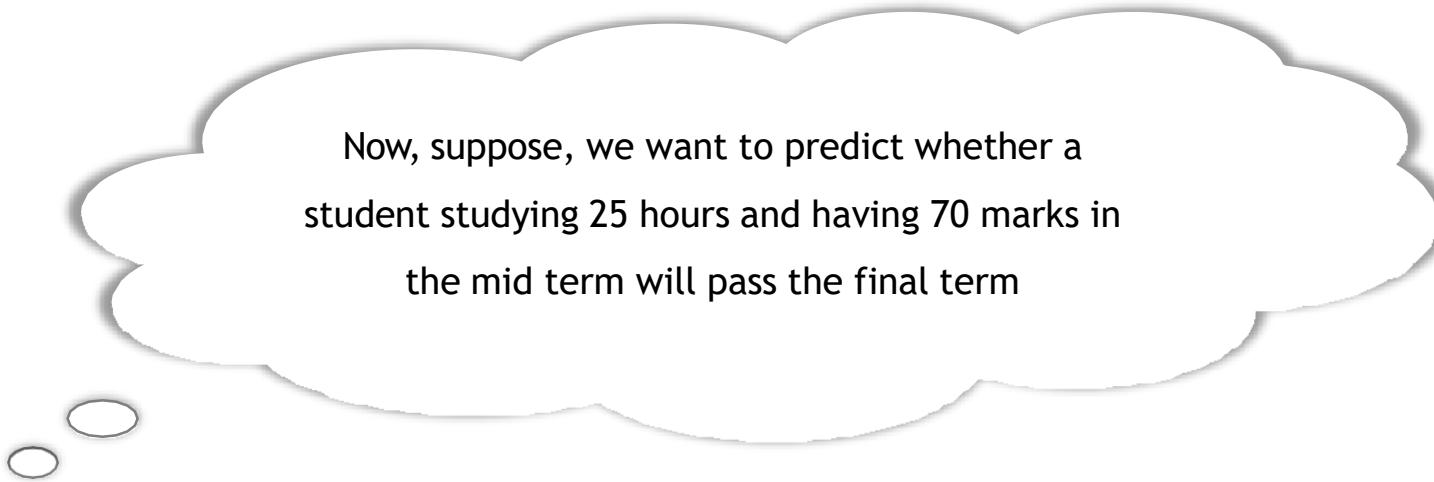
Lets take another example to understand a multi-layer perceptron better!

Use Case 2

Suppose, we have the following student-marks dataset

Hours Studied	Mid-term Marks	Final Results
35	67	1
12	75	0
16	89	1
45	56	1
10	90	0

- The two input columns show the number of hours each student has studied and the mid-term marks obtained by the student, respectively
- The Final Results column can have two values 1 or 0 indicating whether the student passed (1) in the final term or failed (0)

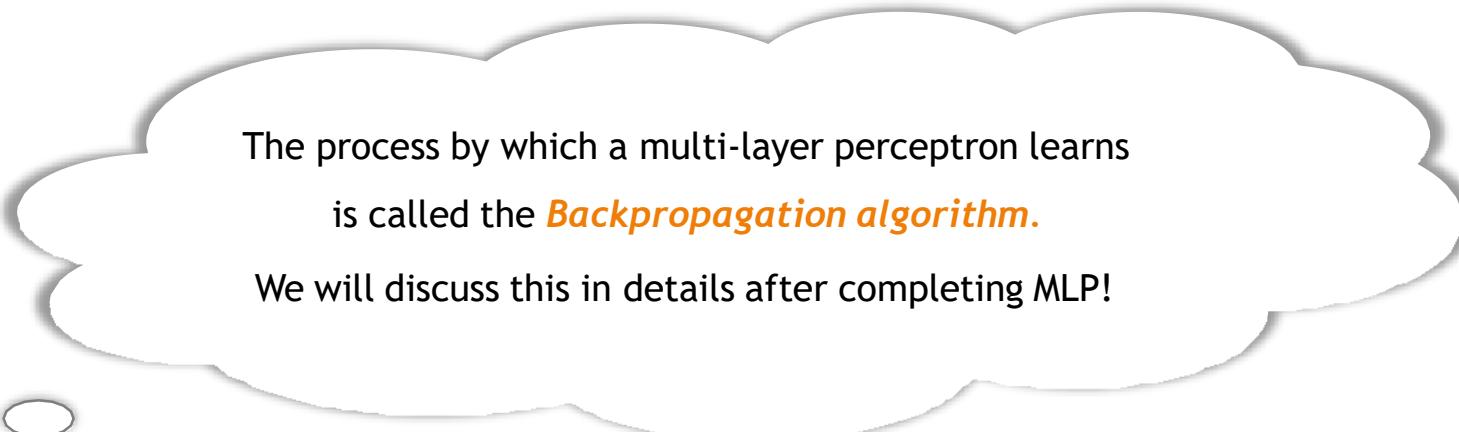


Now, suppose, we want to predict whether a student studying 25 hours and having 70 marks in the mid term will pass the final term

Use Case 2

Hours Studied	Mid-term Marks	Final Results
25	70	?

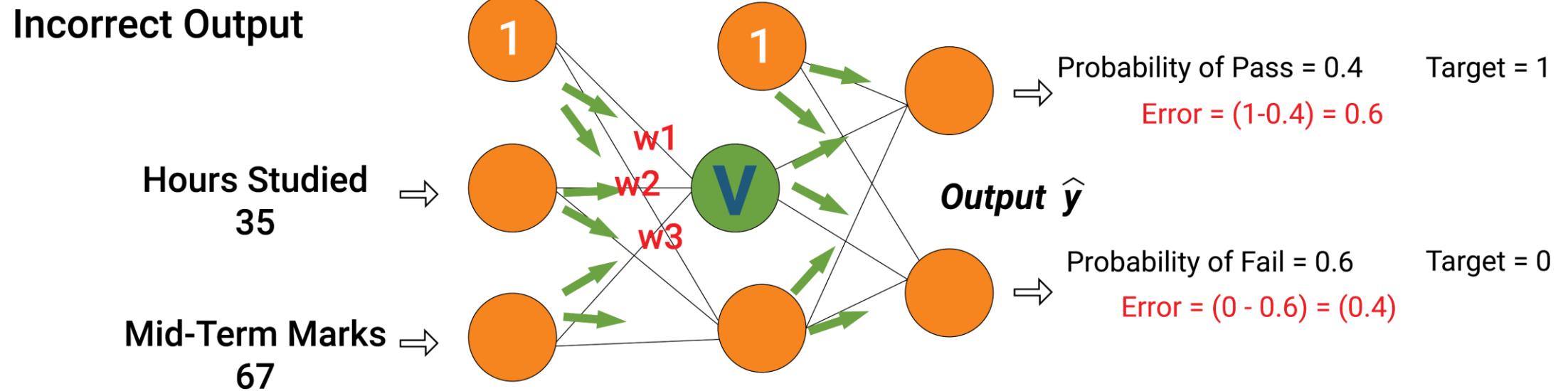
This is a binary classification problem where a multi-layer perceptron can learn from the given examples (the training data) and make an informed prediction when given a new data point. We will see now, how a multi-layer perceptron learns such relationships



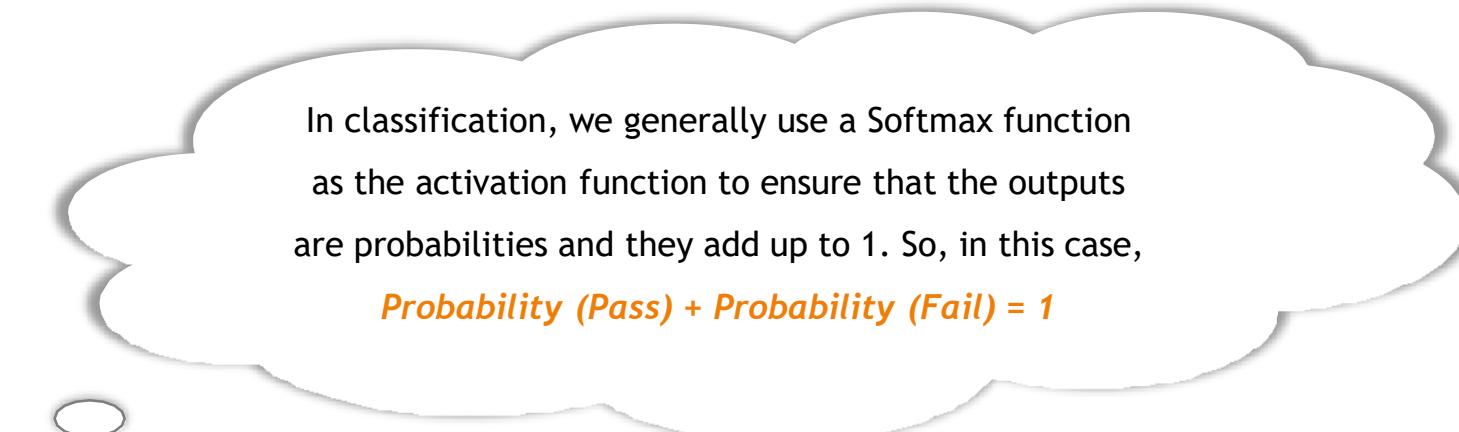
The process by which a multi-layer perceptron learns
is called the ***Backpropagation algorithm***.

We will discuss this in details after completing MLP!

Use Case 2: Solution



- The figure has two nodes in the input layer (apart from the Bias node) which take the inputs *Hours Studied* and *Mid-term Marks*
- It also has a hidden layer with two nodes (apart from the Bias node)
- The output layer has two nodes as well: the upper node outputs the *probability of 'Pass'* while the lower node outputs the *probability of 'Fail'*



In classification, we generally use a Softmax function as the activation function to ensure that the outputs are probabilities and they add up to 1. So, in this case,

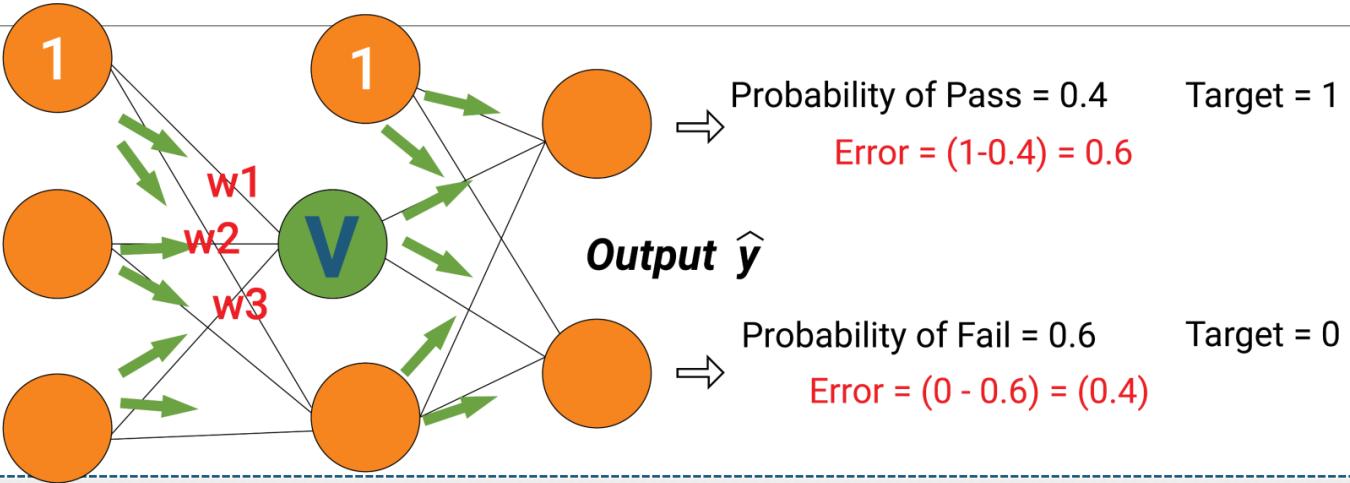
$$\text{Probability (Pass)} + \text{Probability (Fail)} = 1$$

Use Case 2: Solution

Incorrect Output

Hours Studied
35

Mid-Term Marks
67

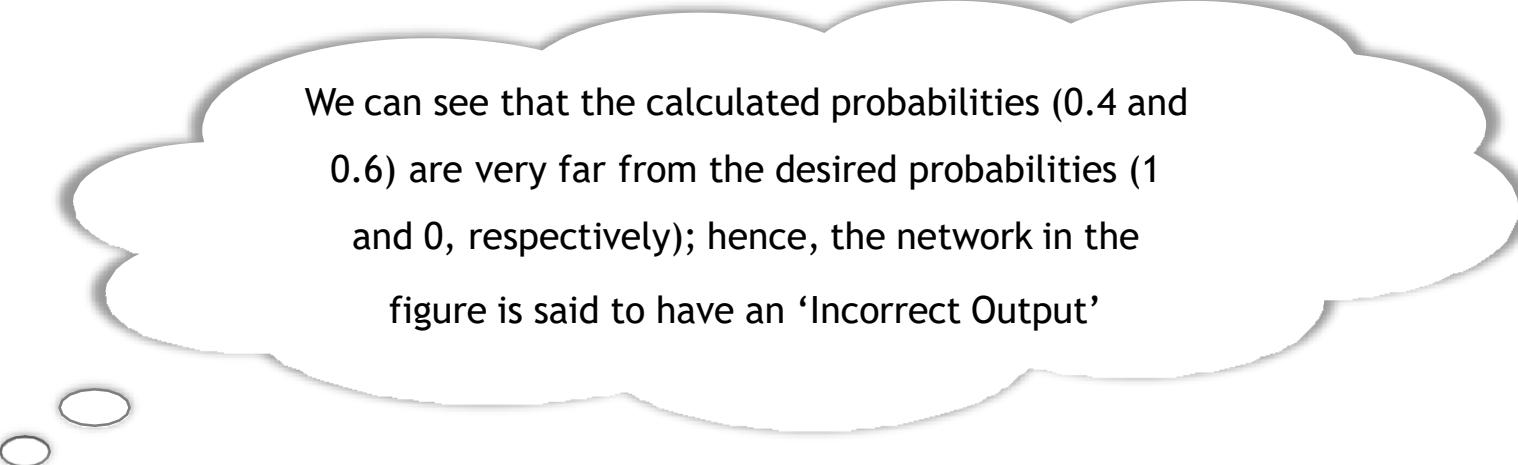


Step 1: *Forward Propagation*

- Let's consider the hidden layer node, marked V, in the figure
- Assume that the weights of the connections from the inputs to that node are w_1 , w_2 , and w_3 (as shown)
- The first training example as input:
 - Input to the network = [35, 67]
 - Desired output from the network (target) = [1, 0]
 - The output V from the node can be calculated as follows (where 'f' is an activation function): $V = f(1 \cdot w_1 + 35 \cdot w_2 + 67 \cdot w_3)$

Suppose, the output probabilities from the two nodes
in the output layer are 0.4 and 0.6, respectively
(since the weights are randomly assigned, outputs
will also be random)





We can see that the calculated probabilities (0.4 and 0.6) are very far from the desired probabilities (1 and 0, respectively); hence, the network in the figure is said to have an ‘Incorrect Output’

Use Case 2: Solution

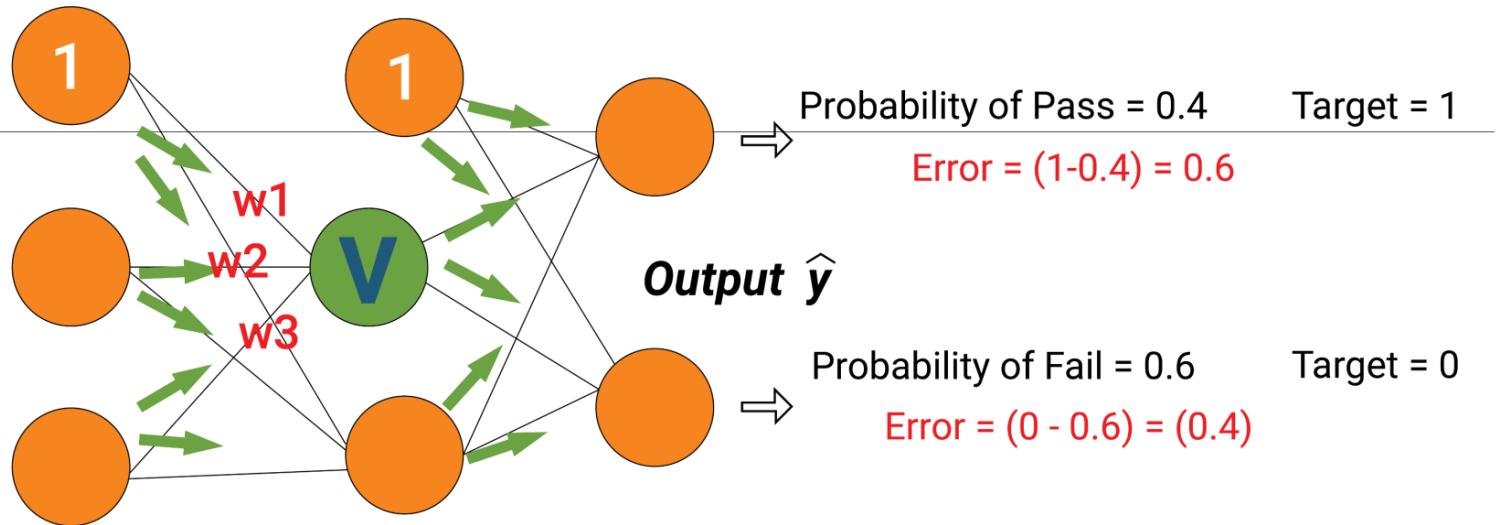
Incorrect Output

Hours Studied

35

Mid-Term Marks

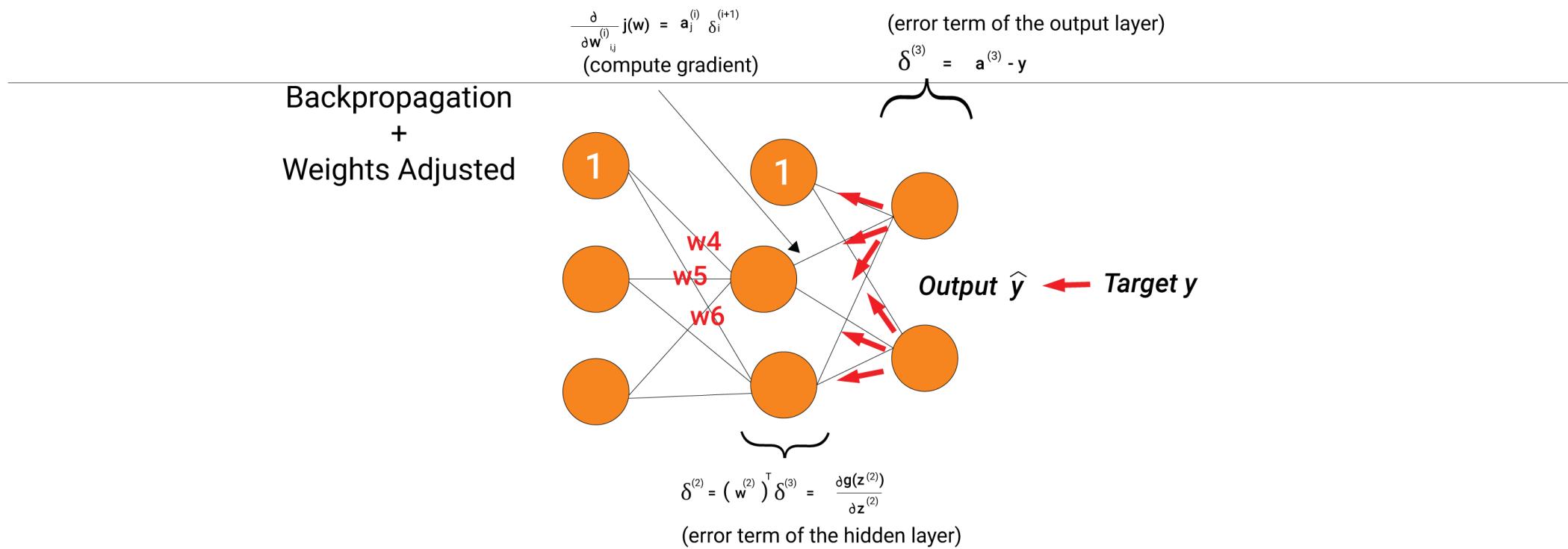
67



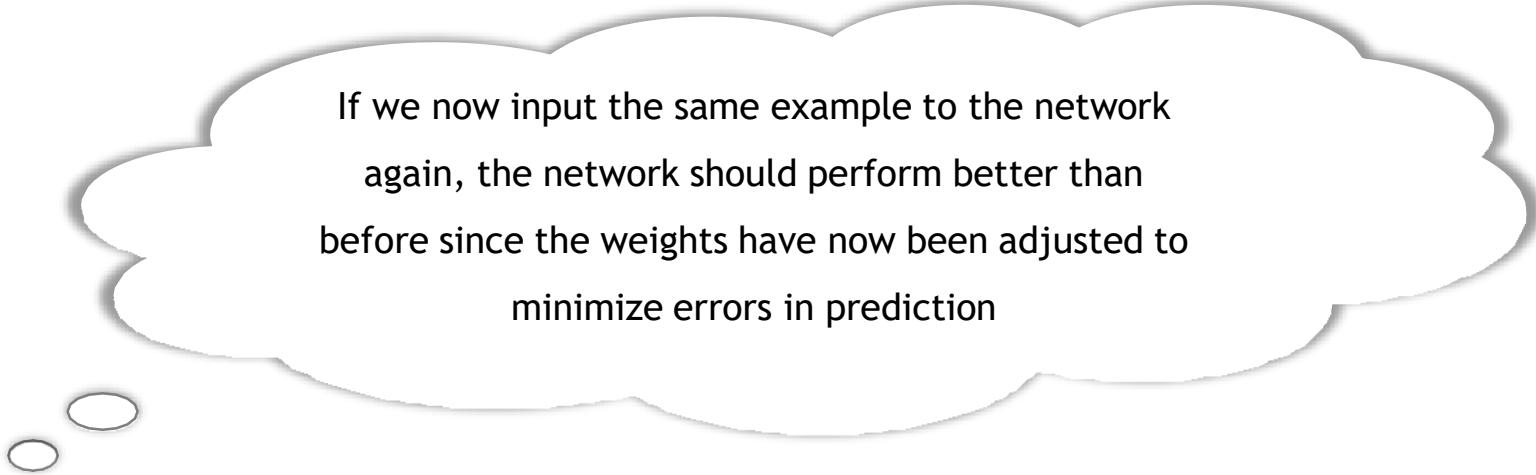
- **Step 2: Backpropagation and Weight Updates**

- We calculate the total errors at the output nodes and propagate these errors back through the network using backpropagation to calculate the gradients
- Then, we use an optimization method such as gradient descent to 'adjust' all weights in the network with an aim of reducing errors at the output layer
- This is shown in the next figure

Use Case 2: Solution

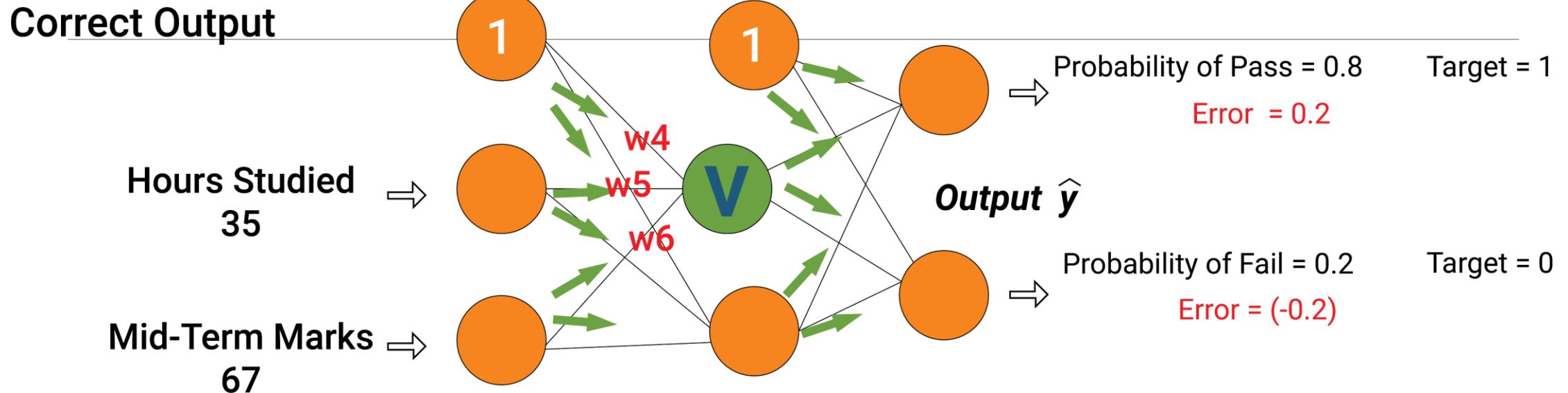


- **Step 2: Backpropagation and Weight Updates**
 - Suppose that the new weights associated with the node in consideration are w_4 , w_5 , and w_6 (after backpropagation and adjusting weights)



If we now input the same example to the network again, the network should perform better than before since the weights have now been adjusted to minimize errors in prediction

Use Case 2: Solution



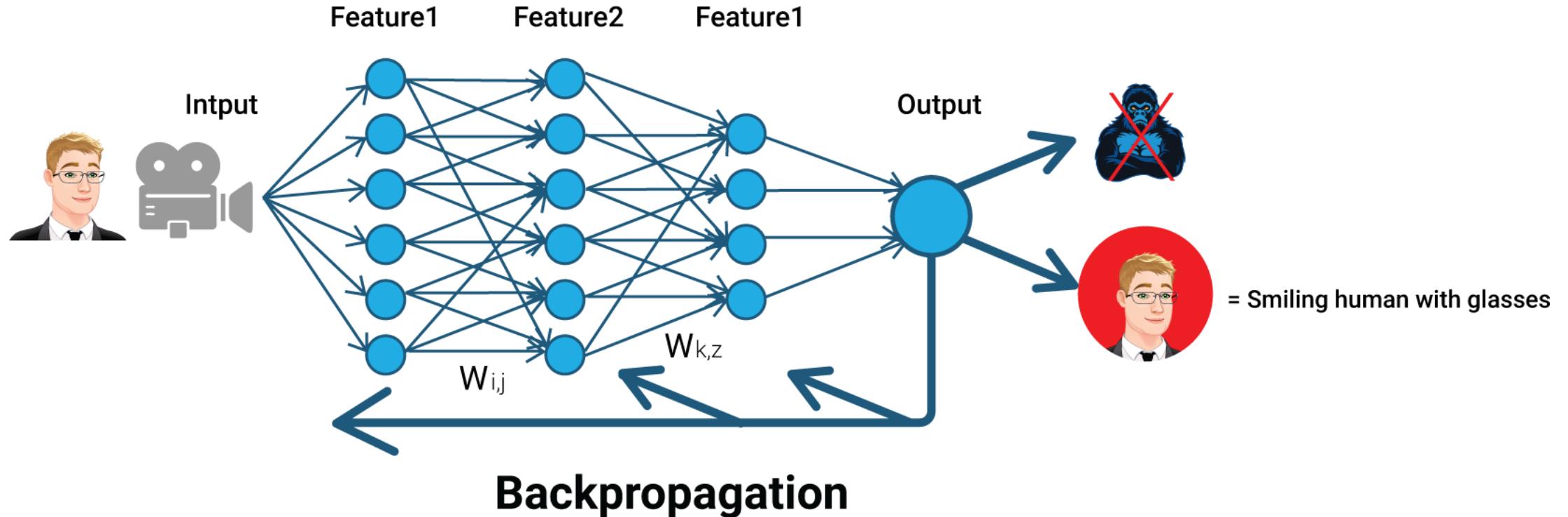
- As shown in Figure, errors at the output nodes have now reduced to [0.2, -0.2] as compared to [0.6, -0.4] earlier
- This means that our network has learned to correctly classify our first training example
- We repeat this process with all other training examples in our dataset. Then, our network will learn those examples as well

If we now want to predict whether a student studying 25 hours and having 70 marks in the mid term will pass the final term, we go through the forward propagation step and find the output probabilities for Pass and Fail

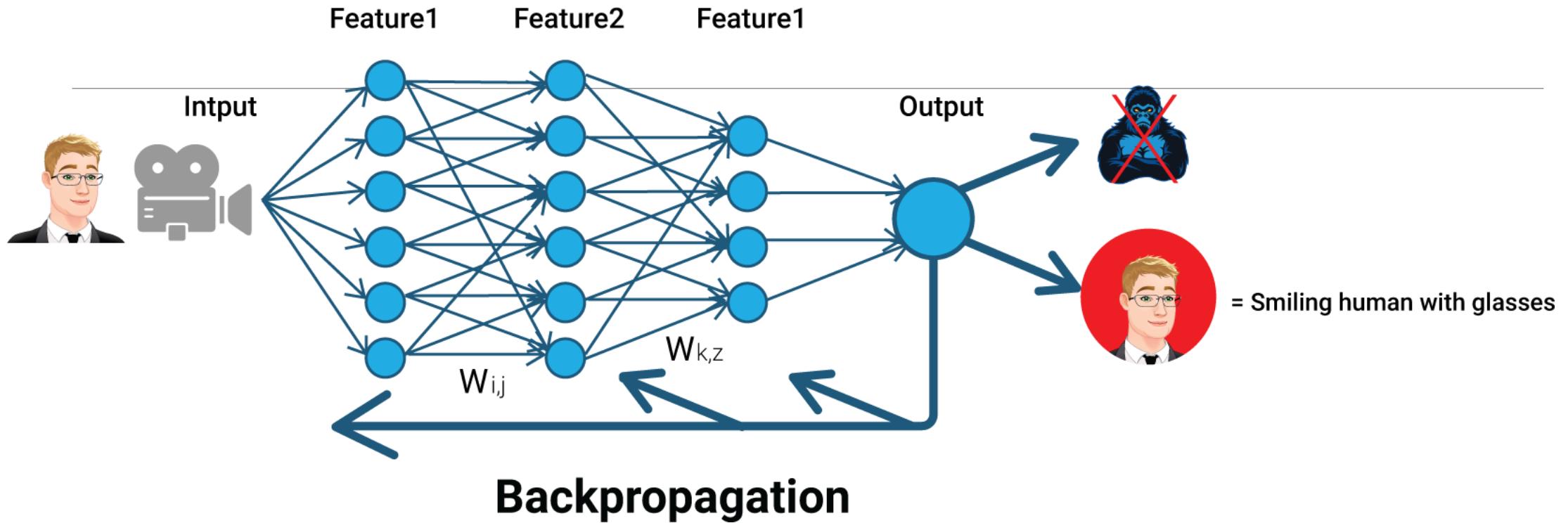


Backpropagation Algorithm

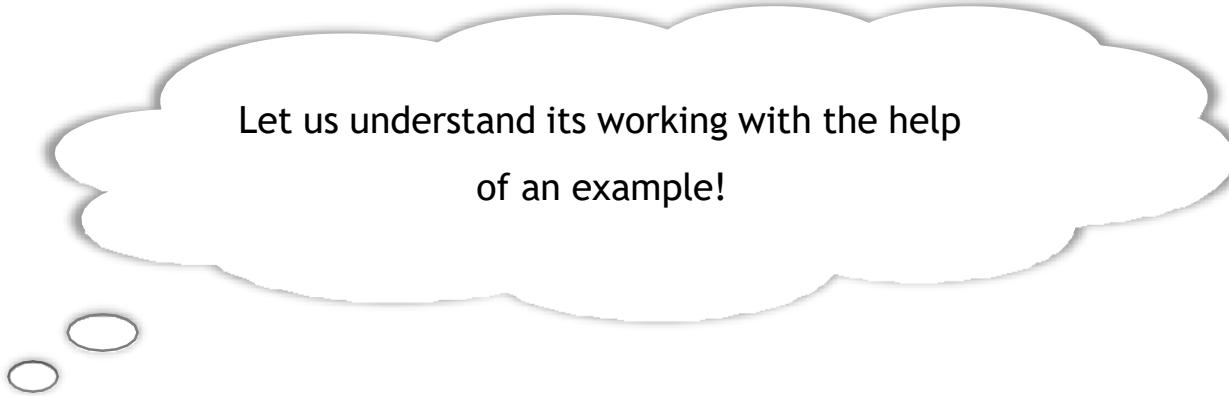
The backpropagation algorithm is a supervised learning method for multi-layer feedforward networks from the field of Artificial Neural Networks



Backpropagation Algorithm



- The principle of this approach is to model a given function by modifying internal weightings of input signals to produce an expected output signal
- The system is trained using a supervised learning method, where the error between the system's output and a known expected output is presented to the system and used to modify its internal state



Let us understand its working with the help
of an example!

Backpropagation Algorithm: How Does it Work?

Consider the following table

Input	Desired Output
0	0
1	2
2	4

Backpropagation Algorithm: How Does it Work?

Consider the initial value of weight as 3

Input	Desired Output	Model Output (W=3)
0	0	0
1	2	3
2	4	6

Backpropagation Algorithm: How Does it Work?

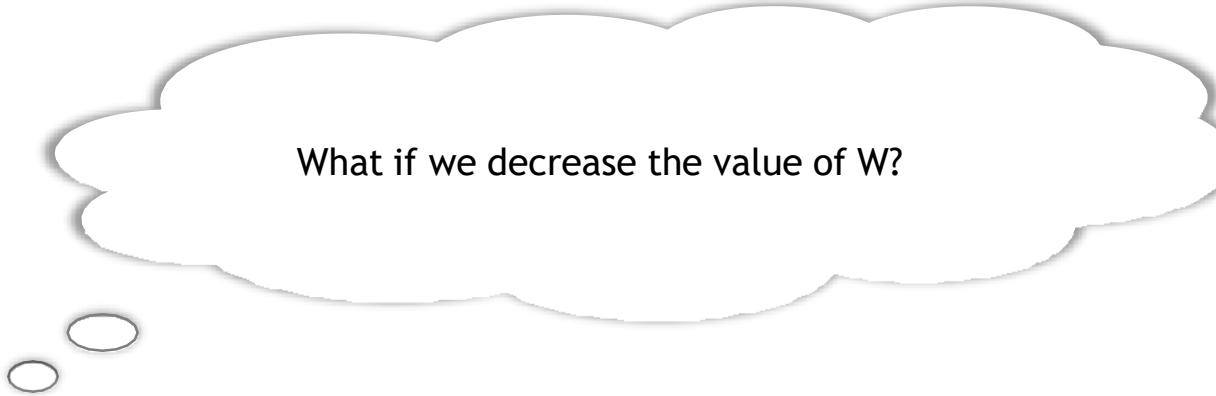
Observe the difference between the actual output and the desired output

Input	Desired Output	Model Output (W=3)	Absolute Error	
0	0	0	0	
1	2	3	1	
2	4	6	2	

Backpropagation Algorithm: How Does it Work?

Observe the error when changing the value of W to 4

Input	Desired Output	Model Output (W=3)	Absolute Error	Square Error	Model Output (W=4)	Square Error
0	0	0	0	0	0	0
1	2	3	1	1	4	4
2	4	6	2	4	8	16



What if we decrease the value of W?



Backpropagation Algorithm: How Does it Work?

Consider the value of weight as 2

Input	Desired Output	Model Output (W=3)	Absolute Error	Square Error	Model Output (W=2)	Square Error
0	0	0	0	0	0	0
1	2	3	1	1	3	1
2	4	6	2	4	4	0

Backpropagation Algorithm: How Does it Work?

Consider the value of weight as 2

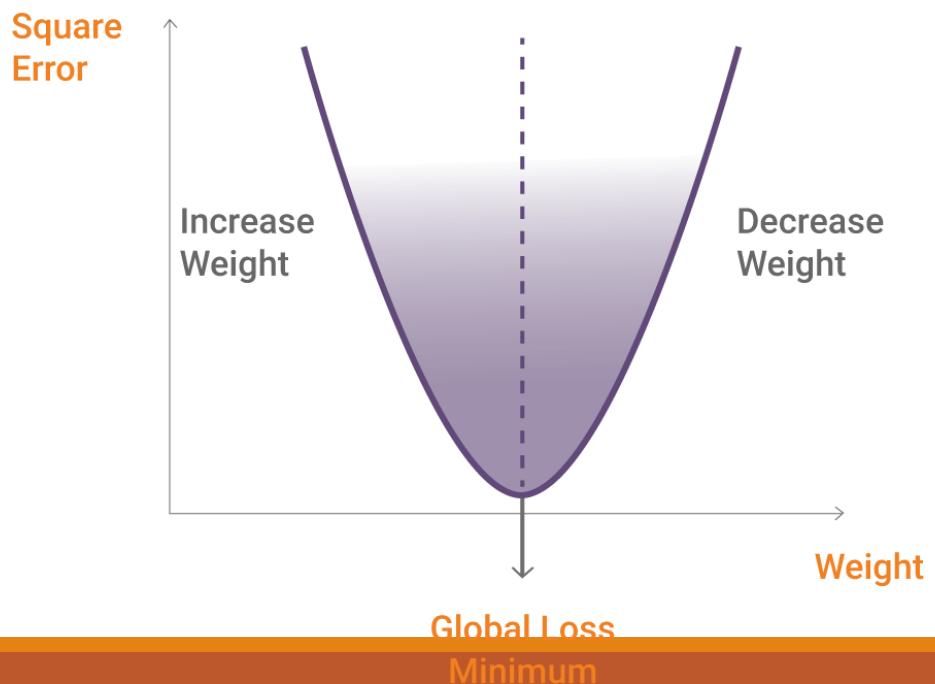
Input	Desired Output	Model Output (W=3)	Absolute Error	Square Error	Model Output (W=2)	Square Error
0	0	0	0	0	0	0
1	2	3	1	1	3	1
2	4	6	2	4	4	0

We see that when the weight is reduced, the error also decreases

Backpropagation Algorithm: How Does it Work?

Consider the following graph

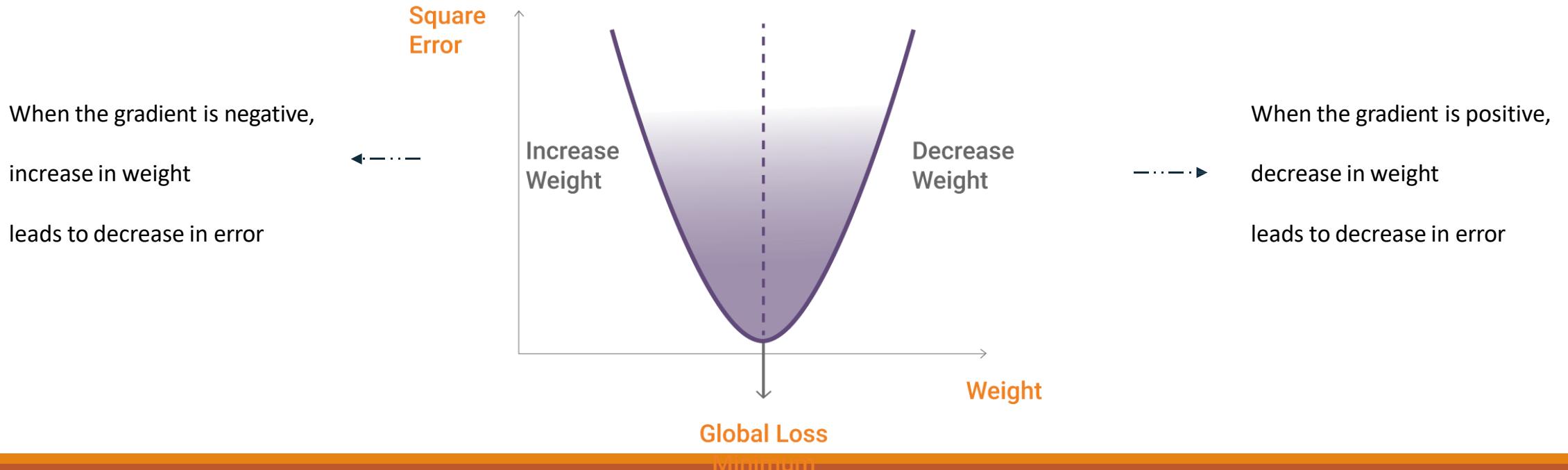
Backpropagation

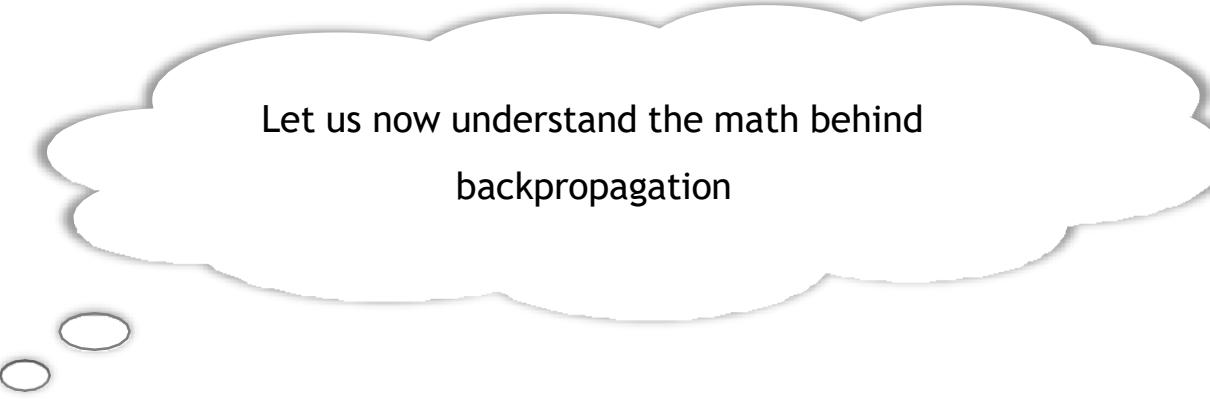


Backpropagation Algorithm: How Does it Work?

We need to reach the *Global Loss Minimum*. This is nothing but backpropagation

Backpropagation

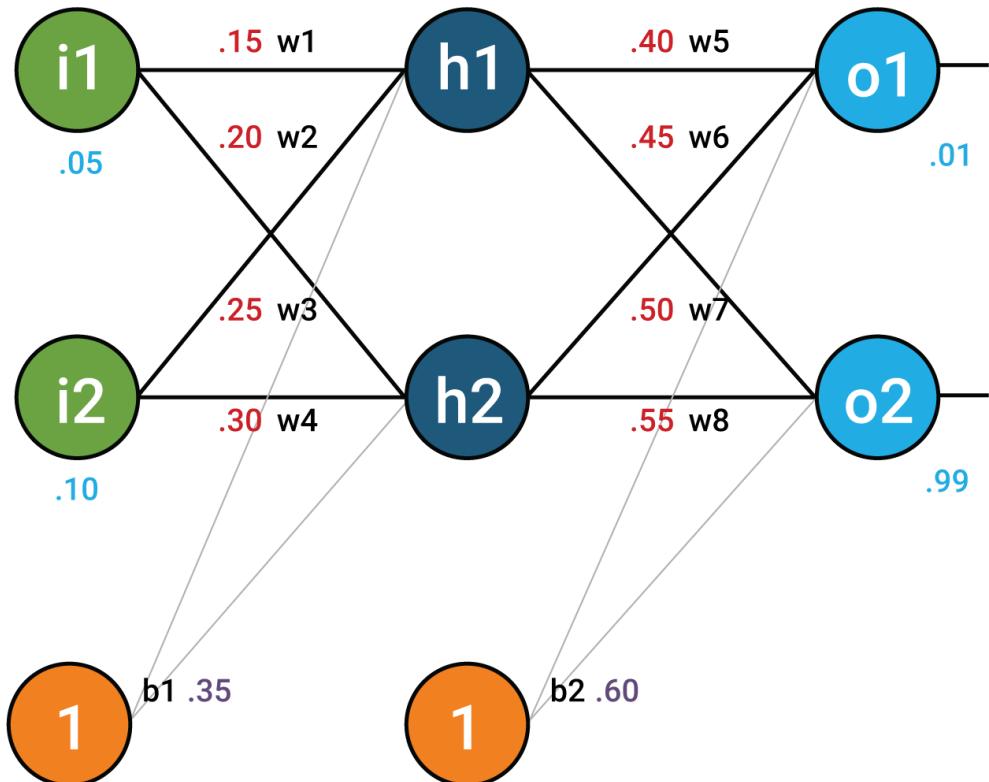




Let us now understand the math behind
backpropagation

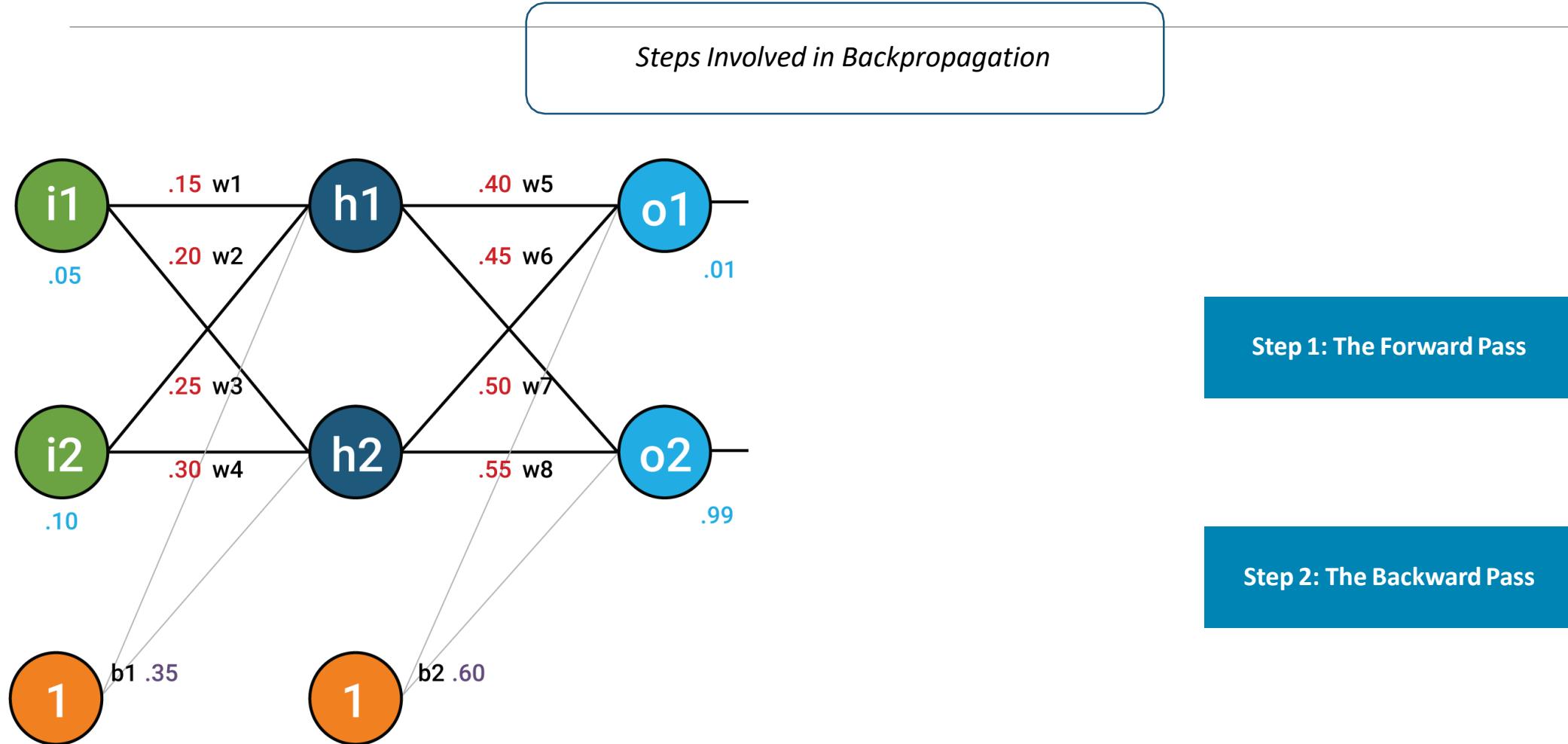
Backpropagation Algorithm: How Does it Work?

In order to have some numbers to work with, here are *initial weights, biases*, and *training inputs/outputs*



- The goal of backpropagation is to optimize the weights so that the neural network can learn how to correctly map arbitrary inputs to outputs
- We're going to work with a single training set: given inputs 0.05 and 0.10, we want the neural network to output 0.01 and 0.99

Backpropagation Algorithm: How Does it Work?



Backpropagation Algorithm: How Does it Work?

Step 1: The Forward Pass

The total net input for h1:

$$\begin{aligned} \text{net h1} &= w_1 * i_1 + w_2 * i_2 + b_1 * 1 \\ \text{net h1} &= 0.15 * 0.05 + 0.2 * 0.1 + 0.35 * 1 = 0.3775 \end{aligned}$$

The output for h1:

$$\text{out h1} = 1 / (1 + e^{-\text{net h1}}) = 1 / (1 + e^{-0.3775}) = 0.593269992$$

Carrying out the same process for h2:

$$\text{out h2} = 0.596884378$$

We repeat this process for the output layer neurons, using the output from the hidden layer neurons as their input

Backpropagation Algorithm: How Does it Work?

Step 1: The Forward Pass

The output for o1:

```
net o1 = w5 * out h1 + w6 * out h2 + b2 * 1  
net o1 = 0.4 * 0.593269992 + 0.45 * 0.596884378 + 0.6 * 1 = 1.105905967  
out o1 = 1/(1 + e-net o1) = 1/(1 + e-1.105905967) = 0.75136507
```

Carrying out the same process for o2:

```
out o2 = 0.772928465
```

Backpropagation Algorithm: How Does it Work?

Calculating the Total Error

We can now calculate the error for each output neuron using the squared error function and sum them to get the total error: $E_{total} = \sum 1/2 (target - output)^2$

The target output for o1 is 0.01, but the neural network output is 0.75136507; therefore, its error is:

$$E_{o1} = 1/2 (\text{target } o1 - \text{out } o1)^2 = 1/2 (0.01 - 0.75136507)^2 = 0.274811083$$

By repeating this process for o2 (remembering that the target is 0.99), we get:

$$E_{o2} = 0.023560026$$

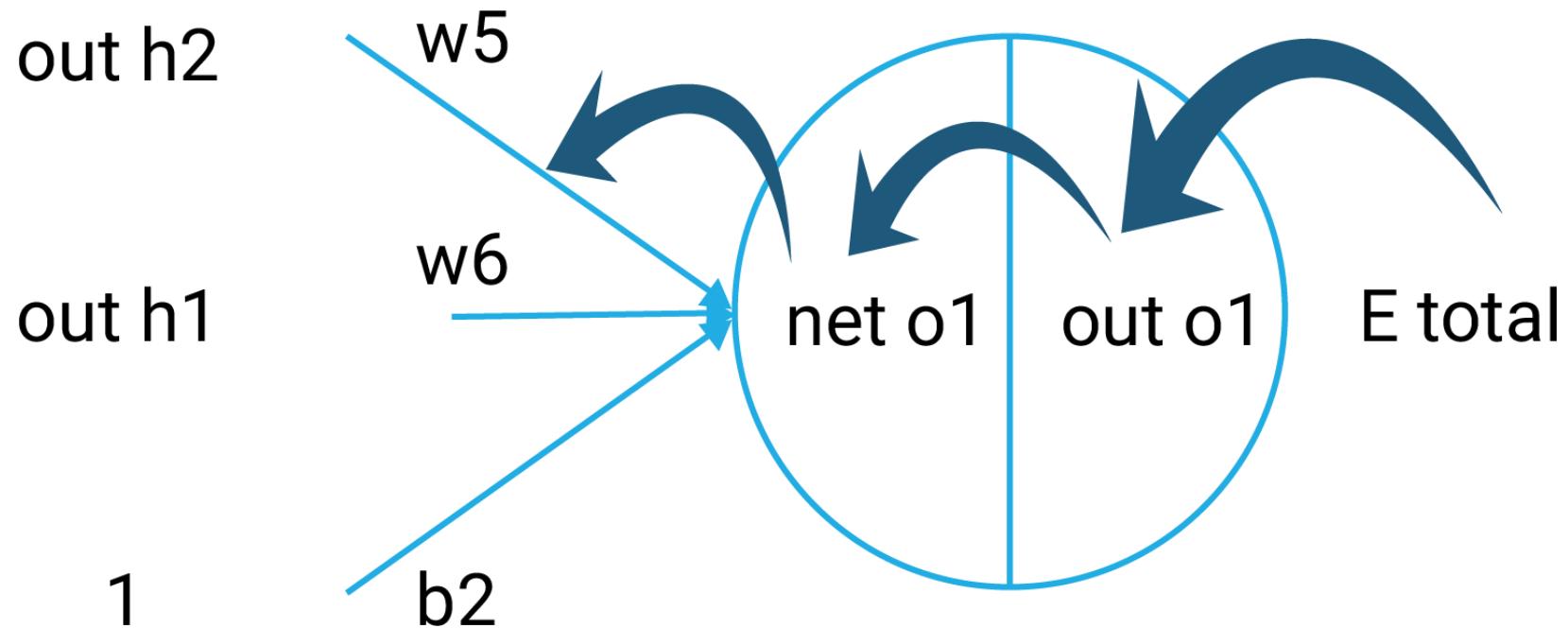
The total error for the neural network is the sum of these errors:

$$E_{total} = E_{o1} + E_{o2} = 0.274811083 + 0.023560026 = 0.298371109$$

Backpropagation Algorithm: How Does it Work?

Step 2: The Backward Pass

Our goal with backpropagation is to update each of the weights in the network so that the actual output is closer to the target output, thereby minimizing the error for each output neuron and the network as a whole



Backpropagation Algorithm: How Does it Work?

Step 2: The Backward Pass

Our goal with backpropagation is to update each of the weights in the network so that the actual output is closer to the target output, thereby minimizing the error for each output neuron and the network as a whole

Consider w_5 , we will calculate the rate of change of error w.r.t the change in weight w_5 :

$$(\partial E_{\text{total}}) / \partial w_5 = (\partial E_{\text{total}}) / (\partial \text{out } o_1) * (\partial \text{out } o_1) / (\partial \text{net } o_1) * (\partial \text{net } o_1) / \partial w_5$$

Since we are propagating backwards, the first thing we need to do is to calculate the change in total errors w.r.t. the outputs o_1 and o_2 :

$$E_{\text{total}} = (1/2) * (\text{target } o_1 - \text{out } o_1)^2 + (1/2) * (\text{target } o_2 - \text{out } o_2)^2$$

$$\begin{aligned} (\partial E_{\text{total}}) / (\partial \text{out } o_1) &= -(\text{target } o_1 - \text{out } o_1) = -(0.01 - 0.75136507) \\ &= 0.74136507 \end{aligned}$$

$$(\partial \text{net } o_1) / \partial w_5 = 1 * \text{out } h_1 * w_5^{(1-1)} + 0 + 0 = \text{out } h_1 = 0.593269992$$

Backpropagation Algorithm: How Does it Work?

Step 2: The Backward Pass

Putting all values together and calculating the updated weight value

let's put all values together:

$$(\partial E \text{ total}) / \partial w_5 = (\partial E \text{ total}) / (\partial \text{out } o_1) * (\partial \text{out } o_1) / (\partial \text{net } o_1) * (\partial \text{net } o_1) / \partial w_5 = 0.082167041$$

Calculate the updated value of w5:

$$w_5^+ = w_5 - n * (\partial E \text{ total}) / \partial w_5 = 0.4 - 0.5 * 0.082167041 = \mathbf{0.35891648}$$

We can repeat this process to get the new weights w6, w7, and w8

$$w_6^+ = 0.408666186$$

$$w_7^+ = 0.511301270$$

$$w_8^+ = 0.561370121$$

We perform the actual updates in the neural network after we have the new weights leading into the hidden layer neurons

Backpropagation Algorithm: How Does it Work?

Step 2: The Backward Pass Hidden Layer

We'll continue the backward pass by calculating new values for w_1, w_2, w_3 , and w_4

Start with w_1 :

$$\begin{aligned}(\partial E_{\text{total}}) / \partial w_1 &= (\partial E_{\text{total}}) / (\partial \text{out } h1) * (\partial \text{out } h1) / (\partial \text{net } h1) * (\partial \text{net } h1) / \partial w_1 \\(\partial E_{\text{total}}) / (\partial \text{out } h1) &= (\partial E_{\text{o1}}) / (\partial \text{out } h1) + (\partial E_{\text{o2}}) / (\partial \text{out } h1)\end{aligned}$$

We're going to use a similar process as we did for the output layer, but slightly different to account for the fact that the output of each hidden layer neuron contributes to the output. Thus, we need to take E_{o1} and E_{o2} into consideration

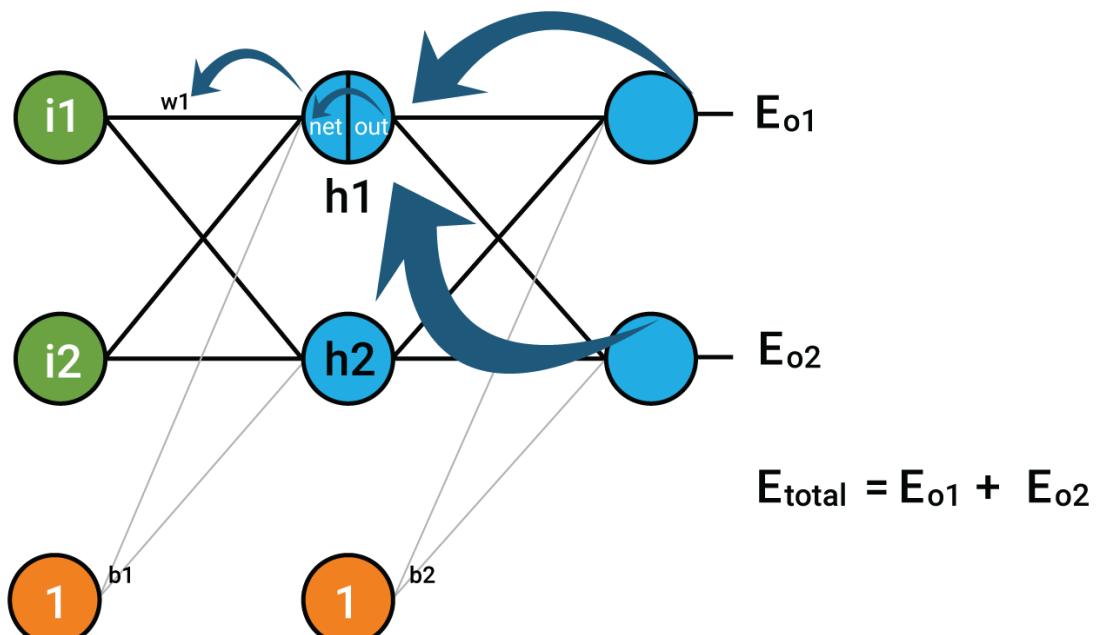
Backpropagation Algorithm: How Does it Work?

Step 2: The Backward Pass Hidden Layer

We can visualize it as:

$$\frac{\partial E_{\text{total}}}{\partial w_1} = \frac{\partial E_{\text{total}}}{\partial \text{out}_{h1}} * \frac{\partial \text{out}_{h1}}{\partial \text{net}_{h1}} * \frac{\partial \text{net}_{h1}}{\partial w_1}$$

$$\frac{\partial E_{\text{total}}}{\partial \text{out}_{h1}} = \frac{\partial E_{o1}}{\partial \text{out}_{h1}} + \frac{\partial E_{o2}}{\partial \text{out}_{h1}}$$



Backpropagation Algorithm: How Does it Work?

Step 2: The Backward Pass Hidden Layer

Starting with:

$$(\partial E_{\text{total}}) / (\partial \text{out h1}) = (\partial E_{\text{o1}}) / (\partial \text{out h1}) + (\partial E_{\text{o2}}) / (\partial \text{out h1})$$

$$(\partial E_{\text{o1}}) / (\partial \text{out h1}) = (\partial E_{\text{o1}}) / (\partial \text{net o1}) * (\partial \text{net o1}) / (\partial \text{out h1})$$

We can calculate $(\partial E_{\text{o1}}) / (\partial \text{net o1})$ using values calculated earlier:

$$\begin{aligned} (\partial E_{\text{o1}}) / (\partial \text{net o1}) &= (\partial E_{\text{o1}}) / (\partial \text{out h1}) * (\partial \text{out h1}) / (\partial \text{net o1}) \\ &= 0.74136507 * 0.186815602 = 0.138498562 \end{aligned}$$

$$\text{net o1} = w_5 * \text{out h1} + w_6 * \text{out h2} + b_2 * 1$$

$$(\partial \text{net o1}) / (\partial \text{out h1}) = w_5 = 0.40$$

Backpropagation Algorithm: How Does it Work?

Step 2: The Backward Pass Hidden Layer

Put the values in the equation:

$$\begin{aligned}(\partial E / \partial o_1) / (\partial o_1 / \partial h_1) &= (\partial E / \partial o_1) / (\partial \text{net}_1) * (\partial \text{net}_1 / \partial o_1) * (\partial o_1 / \partial h_1) \\&= 0.138498562 * 0.40 = 0.055399425\end{aligned}$$

Following the same process for $(\partial E / \partial o_2) / (\partial o_2 / \partial h_1)$, we get:

$$(\partial E / \partial o_2) / (\partial o_2 / \partial h_1) = -0.019049119$$

Backpropagation Algorithm: How Does it Work?

Step 2: The Backward Pass Hidden Layer

We can calculate:

$$\begin{aligned}(\partial E_{\text{total}}) / (\partial \text{out } h1) &= (\partial E_{o1}) / (\partial \text{out } h1) + (\partial E_{o2}) / (\partial \text{out } h1) \\&= 0.055399425 + (-0.019049119) = 0.036350306\end{aligned}$$

Now that we have $(\partial E_{\text{total}}) / (\partial \text{out } h1)$, we need to figure out $(\partial \text{out } h1) / (\partial \text{net } h1)$ and $(\partial \text{net } h1) / \partial w$ for each weight

$$\text{out } h1 = 1 / (1 + e^{-\text{net } h1})$$

$$(\partial \text{out } h1) / (\partial \text{net } h1) = \text{out } h1(1 - \text{out } h1) = 0.59326999(1 - 0.59326999) = 0.241300709$$

We calculate the partial derivative of the total net input to $h1$ with respect to w_1 the same as we did for the output neuron:

$$\text{net } h1 = w_1 * i_1 + w_3 * i_2 + b_1 * 1$$

$$(\partial \text{net } h1) / \partial w_1 = i_1 = 0.05$$

Backpropagation Algorithm: How Does it Work?

Step 2: The Backward Pass Hidden Layer

Put it all together:

$$\begin{aligned}(\partial E_{\text{total}}) / \partial w_1 &= (\partial E_{\text{total}}) / (\partial \text{out h1}) * (\partial \text{out h1}) / (\partial \text{net h1}) * (\partial \text{net h1}) / \partial w_1 \\(\partial E_{\text{total}}) / \partial w_1 &= 0.036350306 * 0.241300709 * 0.05 = 0.000438568\end{aligned}$$

We can now update w_1 :

$$w_1^+ = w_1 - n * (\partial E_{\text{total}}) / \partial w_1 = 0.15 - 0.5 * 0.000438568 = 0.149780716$$

Update other weights similarly:

$$w_2^+ = 0.19956143$$

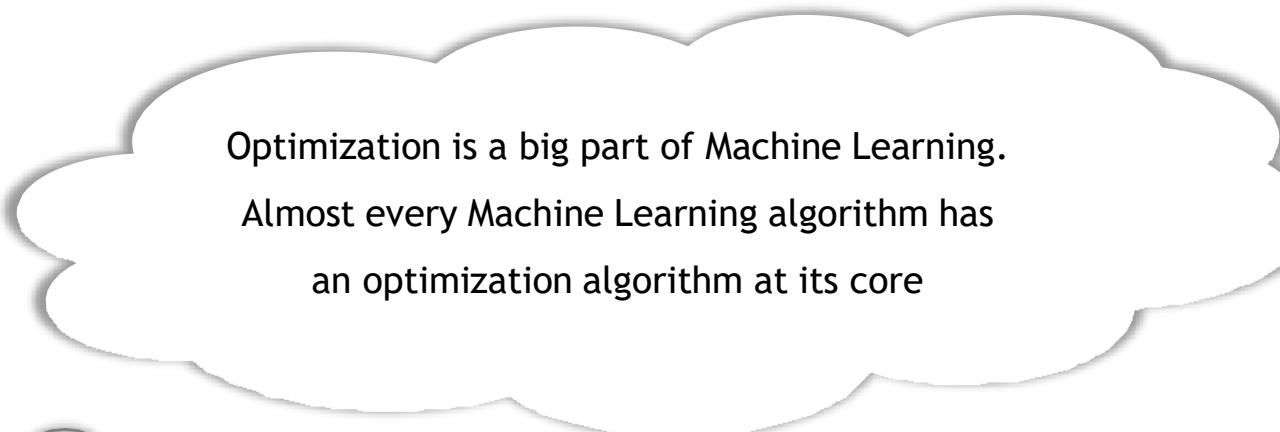
$$w_3^+ = 0.24975114$$

$$w_4^+ = 0.29950229$$

- When we fed forward 0.05 and 0.1 inputs originally, the error on the network was 0.298371109
- After this first round of backpropagation, the total error is now down to 0.291027924

It might not seem like much, but after repeating this process 10,000 times, for example, the error plummets to 0.0000351085. At this point, when we feed forward 0.05 and 0.1, the two output neurons generate 0.015912196 (vs. 0.01 target) and 0.984065734 (vs. 0.99 target)





Optimization is a big part of Machine Learning.
Almost every Machine Learning algorithm has
an optimization algorithm at its core



Gradient Descent

- Gradient descent is by far the most popular optimization strategy, used in Machine Learning and Deep Learning at the moment
 - It is used while training your model, can be combined with every algorithm, and is easy to understand and implement
-

Gradient measures how much the output of a function changes if you change the inputs a little bit

- You can also think of a gradient as the slope of a function. *The higher the gradient, the steeper the slope and the faster the model learns*

Gradient Descent

- Gradient descent is by far the most popular optimization strategy, used in Machine Learning and Deep Learning at the moment
- It is used while training your model, can be combined with every algorithm, and is easy to understand and implement

Gradient measures how much the output of a function changes if you change the inputs a little bit

- You can also think of a gradient as the slope of a function. *The higher the gradient, the steeper the slope and the faster the model learns*

How Does it Work?

Gradient Descent

- Gradient descent is by far the most popular optimization strategy, used in Machine Learning and Deep Learning at the moment
- It is used while training your model, can be combined with every algorithm, and is easy to understand and implement

Gradient measures how much the output of a function changes if you change the inputs a little bit

- You can also think of a gradient as the slope of a function. *The higher the gradient, the steeper the slope and the faster the model learns*

How Does it Work?

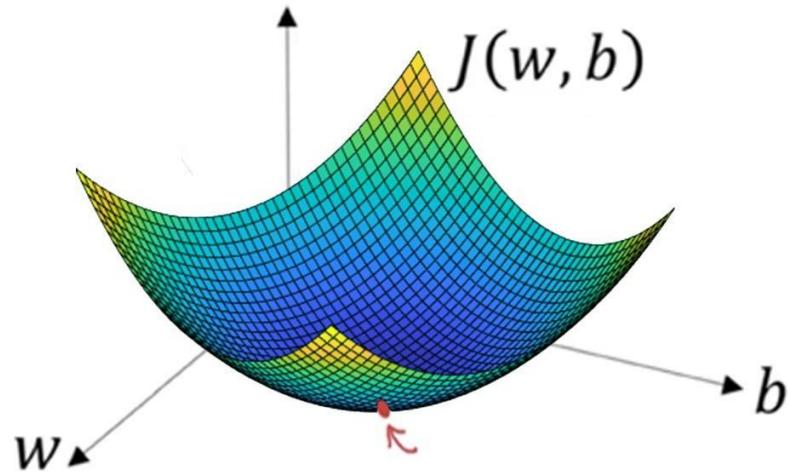
$$b = a - \gamma \nabla f(a)$$

- b = next value
- a = current value
- ‘ $-$ ’ refers to the minimization part of the gradient descent
- γ in the middle is the learning rate, and the gradient term $\nabla f(a)$ is simply the direction of the steepest descent

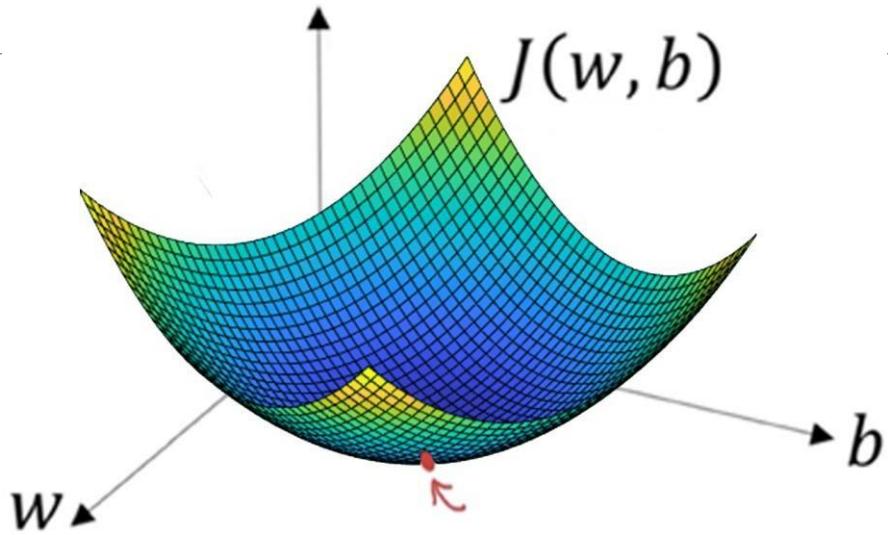
Gradient Descent

$$b = a - \gamma \nabla f(a)$$

- This formula basically tells you the next position where you need to go, which is the direction of the steepest descent
- Gradient descent can be thought of climbing down to the bottom of a valley, instead of climbing up a hill. This is because it is a *minimization algorithm* that minimizes a given function
- Consider the graph below where we need to find the *values of w and b* that correspond to the *minimum of the cost function* (marked with the red arrow)



Gradient Descent



- To start with finding the right values, we initialize the values of w and b with some random numbers, and gradient descent then starts at that point (somewhere around the top)
- Then, it takes one step after the other in the steepest downside direction (e.g., from top to bottom) till it reaches the point where the cost function is as small as possible

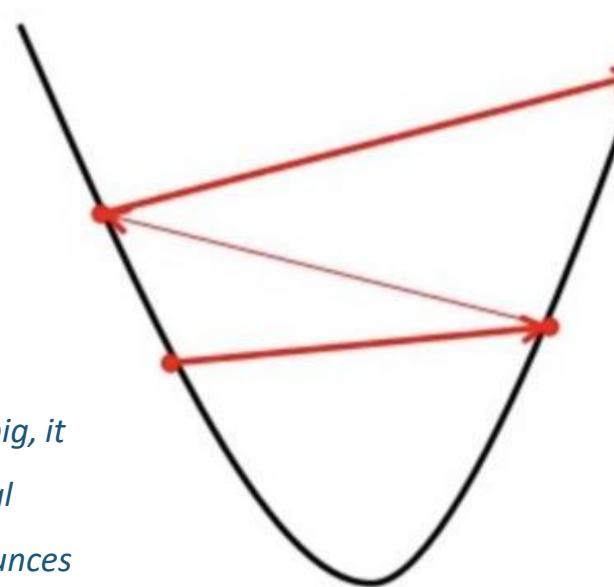
Importance of the Learning Rate

- Learning rate determines how fast or slow we will move toward the optimal weights
- In order for gradient descent to reach the local minimum, we have to set the learning rate to an appropriate value, which is neither too low nor too high

Big learning rate

Small learning rate

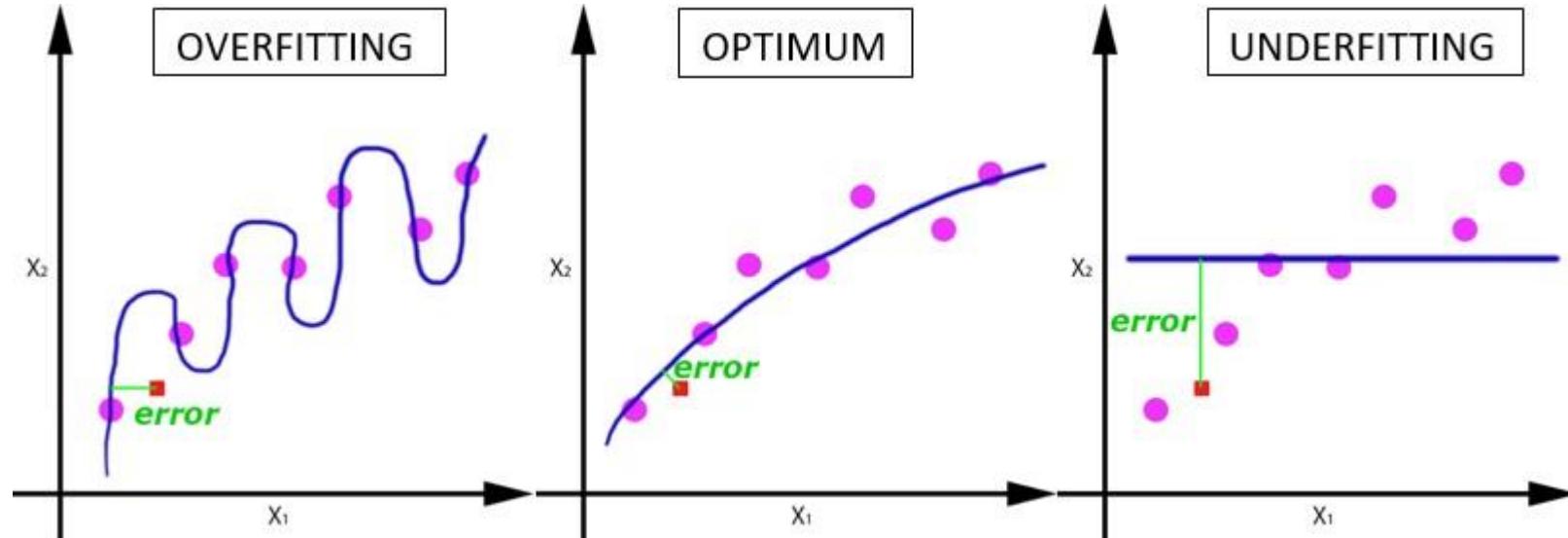
If the steps it takes are too big, it might not reach the local minimum because it just bounces back and forth between the convex function of gradient descent



If you set the learning rate to a very small value, gradient descent will eventually reach the local minimum, but it might take too much time

Understanding Epoch

- One epoch is when an ENTIRE dataset is passed forward and backward through the neural network only ONCE
- One epoch leads to underfitting of the curve in the graph



- As the number of epochs increases, more number of times the weights are changed in the neural network and the curve goes from **underfitting** to **optimal** to **overfitting**

Batches and Iterations

Batch Size

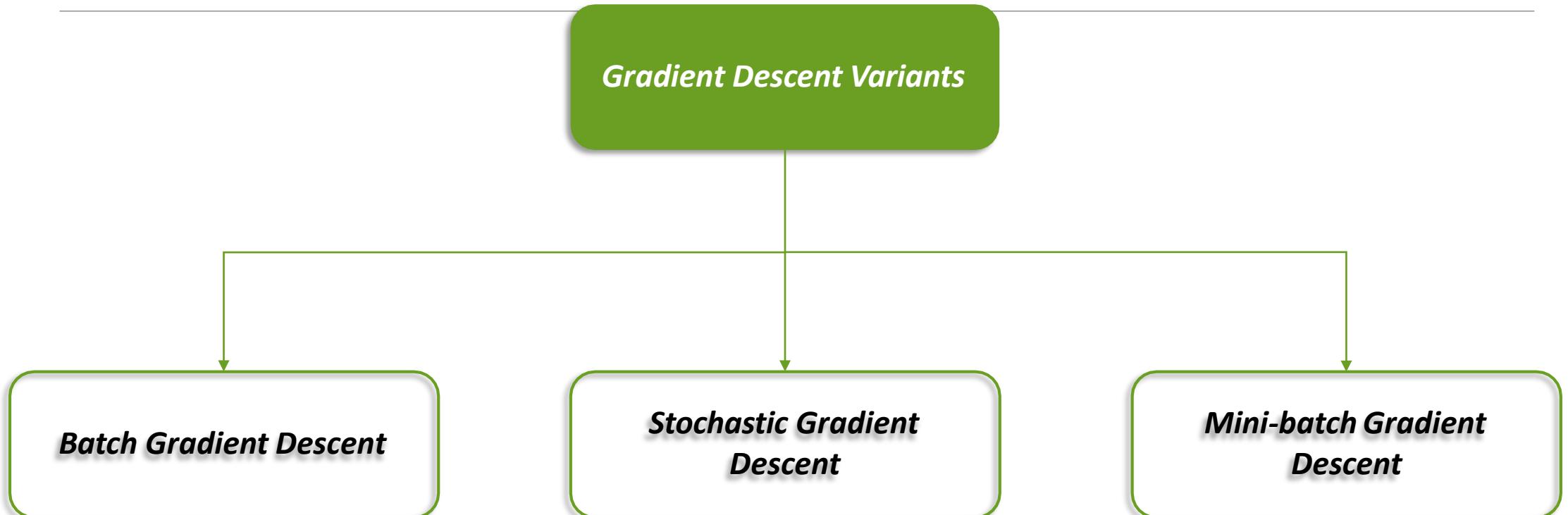
- Total number of training examples present in a single batch is referred to as the batch size
- Since we can't pass the entire dataset into the neural net at once, we divide the dataset into number of batches or sets or parts

Iterations

- Iteration is the number of batches needed to complete one epoch

Let's say, we have 2,000 training examples that we are going to use. We can divide the dataset of 2,000 examples into batches of 500, and then it will take four iterations to complete one epoch

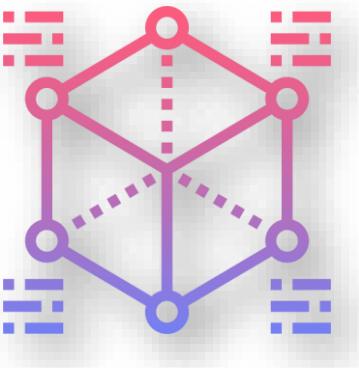
Gradient Descent Variants



Gradient Descent Variants



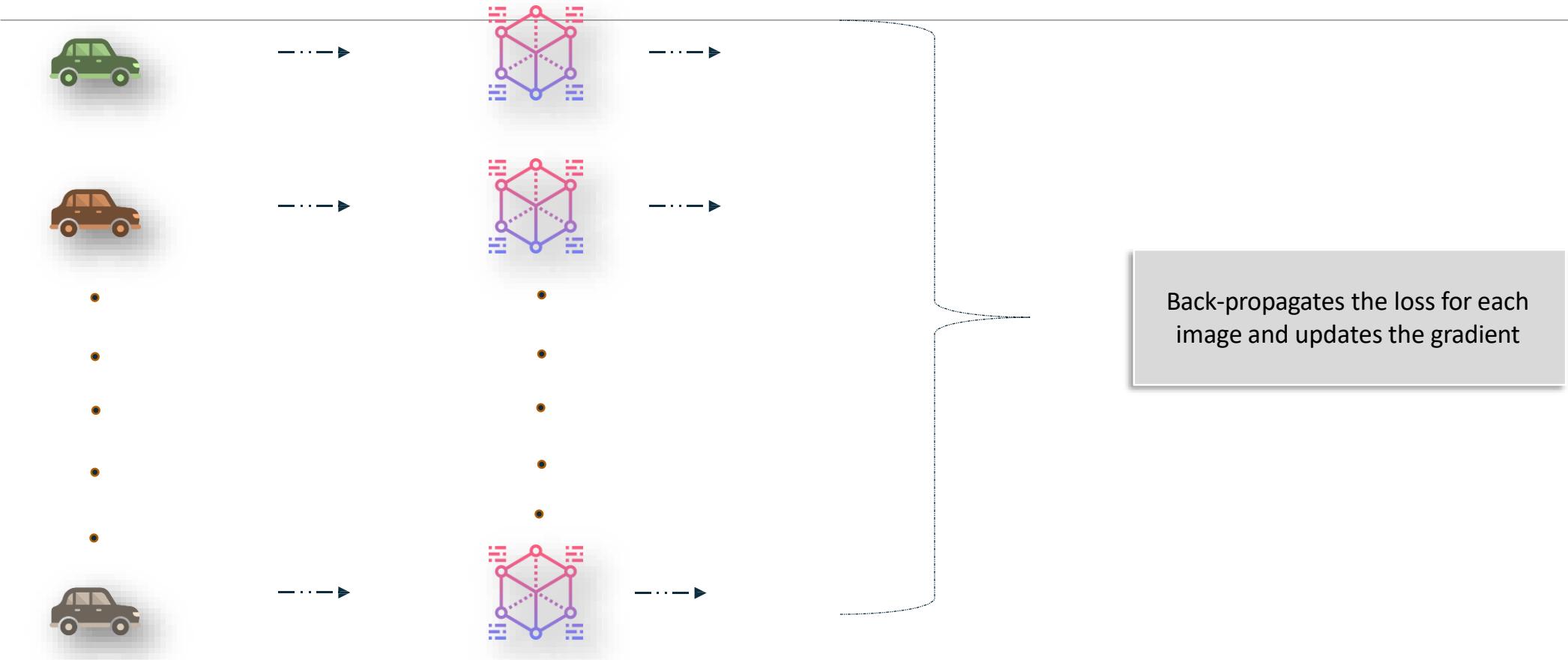
Batch Gradient Descent



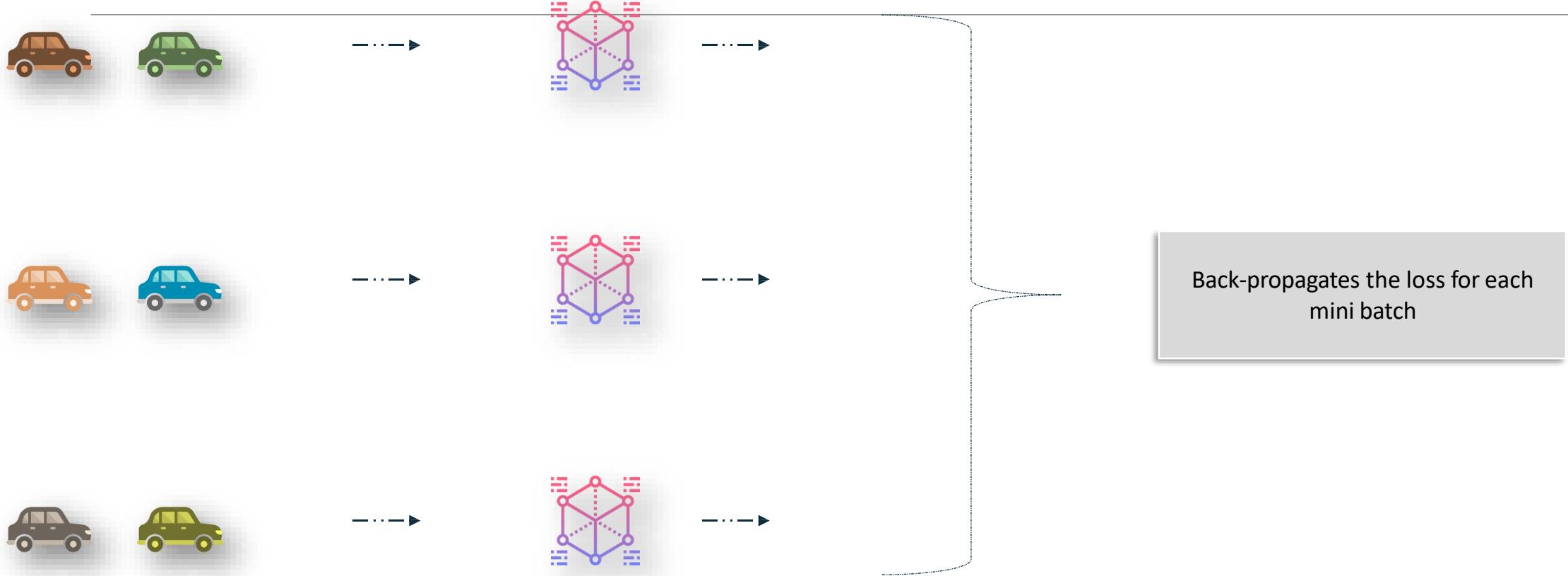
Back-propagates the loss for all 10 images at a time

Takes the entire dataset at a time

Stochastic Gradient Descent



Mini-Batch Gradient Descent

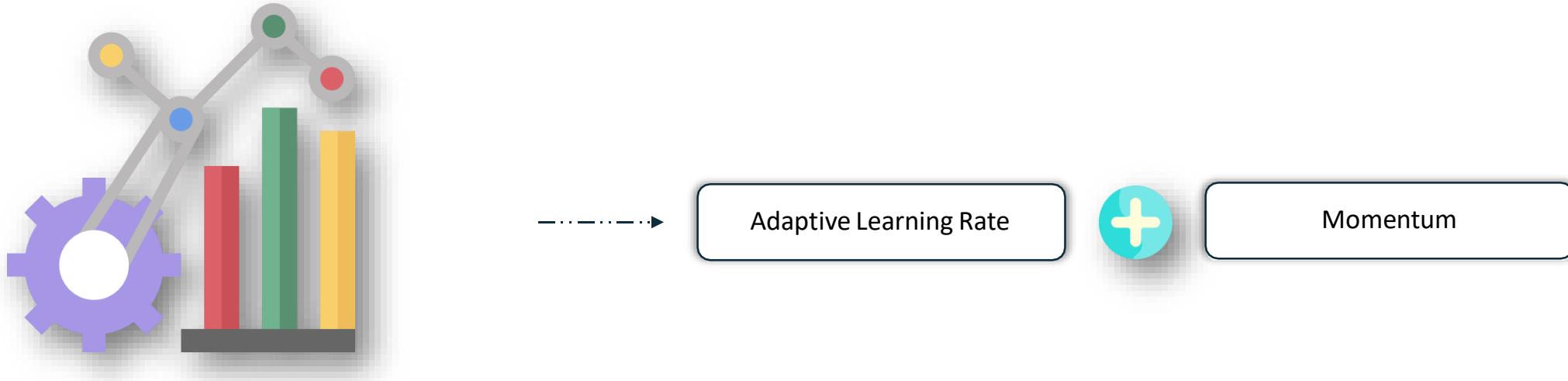


Tips for Gradient Descent

- **Plot Cost versus Time:** Collect and plot the cost values calculated by the algorithm for each iteration. The expectation for a well-performing gradient descent run is a decrease in cost at every iteration. If it does not decrease, try reducing your learning rate
- **Learning Rate:** The learning rate value is a small real value such as 0.1, 0.001, or 0.0001. Try different values for your problem and see which works best
- **Rescale Inputs:** The algorithm will reach the minimum cost faster if the shape of the cost function is not skewed and distorted. You can achieve this by rescaling all of the input variables (X) to the same range, such as [0, 1] or [-1, 1]

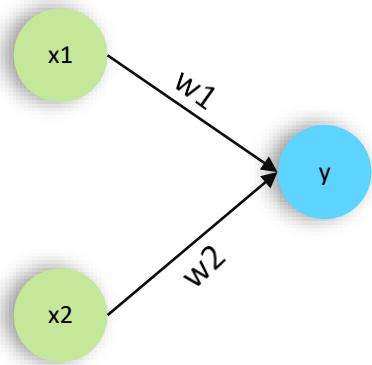
Adam Optimization Algorithm

- The Adaptive Moment Estimation or Adam optimization algorithm is a combination of gradient descent with momentum and RMSprop algorithms
- Adam is an adaptive learning rate method, which means that it computes individual learning rates for different parameters



Implementing a Simple Neural Network

Implementing a Simple Neural Network



Let $x_1 = 2$, $x_2 = 5$, and $y = 31$

Initializing the values of x_1 , x_2 , and y :

In [8]: `x1=2
x2=5
y=31`

Implementing a Simple Neural Network

Implementing forward propagation and backpropagation:

```
In [10]: lr=0.01  
w1=9  
w2=7  
  
for epoch in range(25):  
    y_pred=w1*x1 + w2*x2  
    error=(y-y_pred)**2  
    dEW1=2*(y-y_pred)*(-x1)  
    dEW2=2*(y-y_pred)*(-x2)  
  
    w1=w1-(lr*dEW1)  
    w2=w2-(lr*dEW2)  
    print("Value of w1 - ", w1, "Value of w2 - ", w2, "Error is - ", error)
```



```
Value of w1 -  8.12 Value of w2 -  4.8 Error is -  484  
Value of w1 -  7.750399999999999 Value of w2 -  3.876000000000003 Error is -  85.37759999999999  
Value of w1 -  7.595167999999999 Value of w2 -  3.487920000000004 Error is -  15.060608640000005  
Value of w1 -  7.529970559999999 Value of w2 -  3.324926400000003 Error is -  2.656691364096002  
Value of w1 -  7.5025876351999985 Value of w2 -  3.256469088000002 Error is -  0.46864035662653386  
Value of w1 -  7.491086806783999 Value of w2 -  3.227717016960007 Error is -  0.08266815890891807  
Value of w1 -  7.486256458849279 Value of w2 -  3.2156411471232005 Error is -  0.014582663231533663  
Value of w1 -  7.484227712716696 Value of w2 -  3.210569281791744 Error is -  0.002572381794042668  
Value of w1 -  7.483375639341812 Value of w2 -  3.208439098352533 Error is -  0.0004537681484689692  
Value of w1 -  7.483017768523224 Value of w2 -  3.2075444213080644 Error is -  8.00447013899338e-05  
Value of w1 -  7.482867462779753 Value of w2 -  3.2071686569493876 Error is -  1.4119885325192866e-05  
Value of w1 -  7.482804334367495 Value of w2 -  3.207010835918743 Error is -  2.490747771367834e-06  
Value of w1 -  7.482777820434347 Value of w2 -  3.2069445510858725 Error is -  4.3936790686730787e-07  
Value of w1 -  7.482766684582424 Value of w2 -  3.2069167114560666 Error is -  7.750449877198654e-08  
Value of w1 -  7.4827620075246175 Value of w2 -  3.2069050188115487 Error is -  1.3671793582514382e-08  
Value of w1 -  7.482760043160338 Value of w2 -  3.206900107900851 Error is -  2.41170438785733e-09  
Value of w1 -  7.482759218127341 Value of w2 -  3.2068980453183578 Error is -  4.254246540708817e-10  
Value of w1 -  7.482758871613482 Value of w2 -  3.206897179033711 Error is -  7.504490893870946e-11  
Value of w1 -  7.482758726077662 Value of w2 -  3.2068968151941593 Error is -  1.3237921953333858e-11  
Value of w1 -  7.482758664952617 Value of w2 -  3.2068966623815474 Error is -  2.3351694382142467e-12  
Value of w1 -  7.482758639280099 Value of w2 -  3.2068965982002506 Error is -  4.1192388488788035e-13  
Value of w1 -  7.48275862849764 Value of w2 -  3.2068965712441053 Error is -  7.266337750799056e-14  
Value of w1 -  7.482758623969008 Value of w2 -  3.206896559225243 Error is -  1.2817819245385772e-14  
Value of w1 -  7.482758622066982 Value of w2 -  3.206896555167461 Error is -  2.261062929716986e-15  
Value of w1 -  7.482758621268132 Value of w2 -  3.206896553170334 Error is -  3.988515178306028e-16
```

Multi-variable Equation

Multi-variable Equation

Loading the NumPy package:

1

```
In [36]: import numpy as np
```

Setting initial values for x1, x2, x3, and y:

2

```
In [21]: x1 = np.random.randint(3, 12, 10)
x2 = np.random.randint(9, 18, 10)
x3 = np.random.randint(12, 20, 10)
y = x1*7 + 5*x2 + 4*x3
```

Having a glance at x1, x2, x3, and y:

3

```
In [38]: x1,x2,x3
```



```
Out[40]: (array([8, 7, 3, 8, 4, 4, 4, 7, 3, 7, 7]),
           array([12, 13, 17, 15, 9, 14, 12, 15, 15, 10]),
           array([13, 12, 19, 12, 15, 19, 16, 15, 18, 13]))
```

Multi-variable Equation

Reshaping 'y':

4

```
In [41]: y = y.reshape(SHAPE,1)  
y
```



```
Out[41]: array([[168],  
[162],  
[182],  
[179],  
[133],  
[174],  
[173],  
[156],  
[196],  
[151]])
```

Transposing 'X':

6

```
In [41]: y = y.reshape(SHAPE,1)  
y
```



```
Out[33]: array([[ 8, 12, 13],  
[ 7, 13, 12],  
[ 3, 17, 19],  
[ 8, 15, 12],  
[ 4, 9, 15],  
[ 4, 14, 19],  
[ 7, 12, 16],  
[ 3, 15, 15],  
[ 7, 15, 18],  
[ 7, 10, 13]])
```

Creating a NumPy array from 'x1', 'x2', and 'x3':

5

```
In [32]: X = np.array([x1, x2, x3])
```

Multi-variable Equation

Setting the learning rate value and initializing random values:

In [34]: lr = 0.0001
w = np.array([np.random.randn(1),
 np.random.randn(1),
 np.random.randn(1)])

7

Implementing forward propagation and backpropagation:

In [35]: error_list = []
for i in range(50):
 y_pred = np.matmul(X, w)
 error = (y - y_pred)**2
 dEw1 = 2*np.matmul((y - y_pred).T, (-X[:,0].reshape(SHAPE, 1)))
 dEw2 = 2*np.matmul((y - y_pred).T, (-X[:,1].reshape(SHAPE, 1)))
 dEw3 = 2*np.matmul((y - y_pred).T, (-X[:,2].reshape(SHAPE, 1)))

 w[0][0] = w[0][0] - lr * dEw1
 w[1][0] = w[1][0] - lr * dEw2
 w[2][0] = w[2][0] - lr * dEw3

 error_list.append(error.mean())
print("value of w1 - ", w[0][0], "value of w2 - ", w[1][0], "value of w3 - ", w[2][0])
plt.plot(error_list)

8

Quiz

Quiz 1

A tensor is a single-dimensional array in which data is stored

A

True

B

False

Answer 1

A tensor is a single-dimensional array in which data is stored

A

True

B

False

Quiz 2

How many layers does a standard Neural Network has?

A 1

B 2

C 3

D 4 or more

Answer 2

How many layers does a standard Neural Network has?

A 1

B 2

C 3

D 4 or more

Quiz 3

Perceptron is other name of Neurons

A

Yes

B

No

Answer 3

Perceptron is other name of Neurons

A

Yes

B

No

Quiz 1

Single Layer Perceptron is easy to use than
Multi Layer Perceptron

A

True

B

False

Answer 1

Single Layer Perceptron is easier to use than
Multi Layer Perceptron

A

True

B

False

Quiz 2

The variants of Gradient Descent are...

- A Batch Gradient Descent
- B Adams Optimizer
- C Ada-Delta Optimizer
- D All of these

Answer 2

The variants of Gradient Descent are...

A

Batch Gradient Descent

B

Adams Optimizer

C

Ada-Delta Optimizer

D

All of these

Quiz 3

The backpropagation algorithm is a supervised learning method for multi-layer feedforward networks

A

Yes

B

No

Answer 3

The backpropagation algorithm is a supervised learning method for multi-layer feedforward networks

A

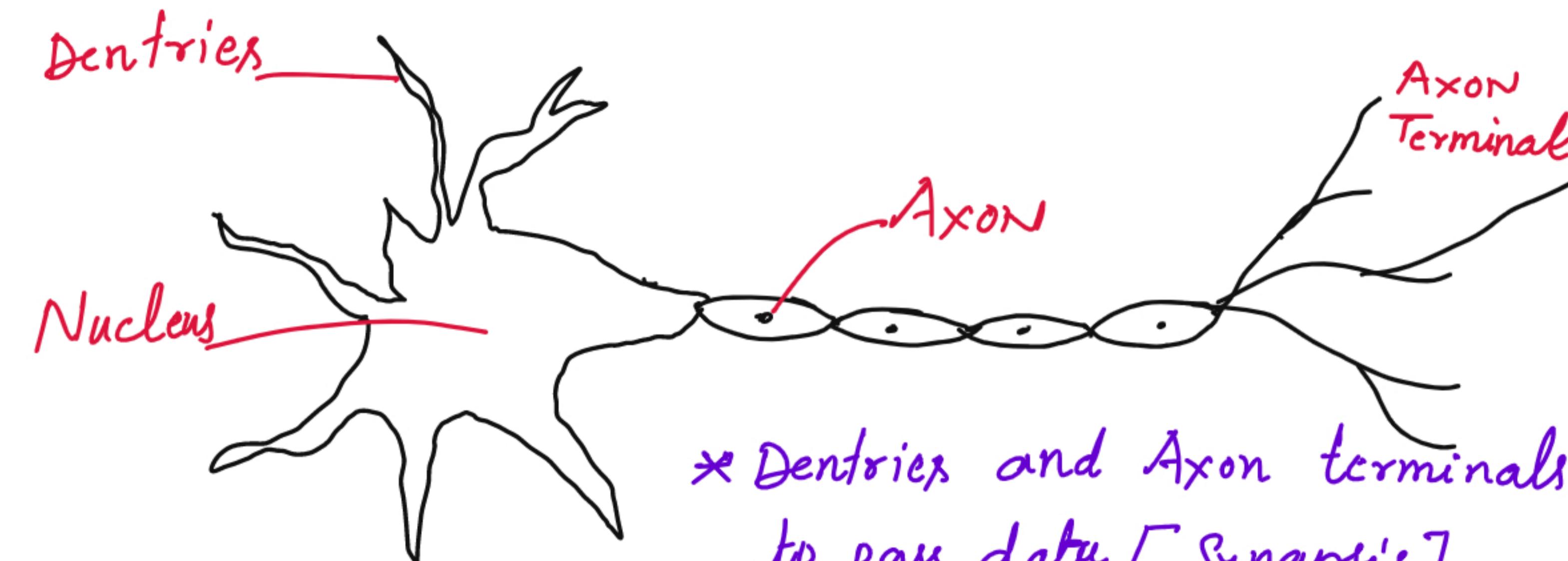
Yes

B

No

Introduction to Neural Network

- * It is a pattern analysis technique that can perform
 - * clustering
 - * classification
 - * Dimensionality Reduction
- * ANN is a network of neurons in which information is flowing in and out
- * It is inspired from biological neural network



* Dendrites and Axon terminals going to pass data [Synapsis]

Due to some motor / Brain Activity → Some electrical activity got initiated which in turn build up a potential that creates a potential difference (Pre-Synaptic potential difference)

This potential difference got converted into post S.P. (pre-synaptic potential difference) due to some chemical activity

↓
Since it is coming from multi neurons potential got aggregated

↓
finally when this value become more than a threshold

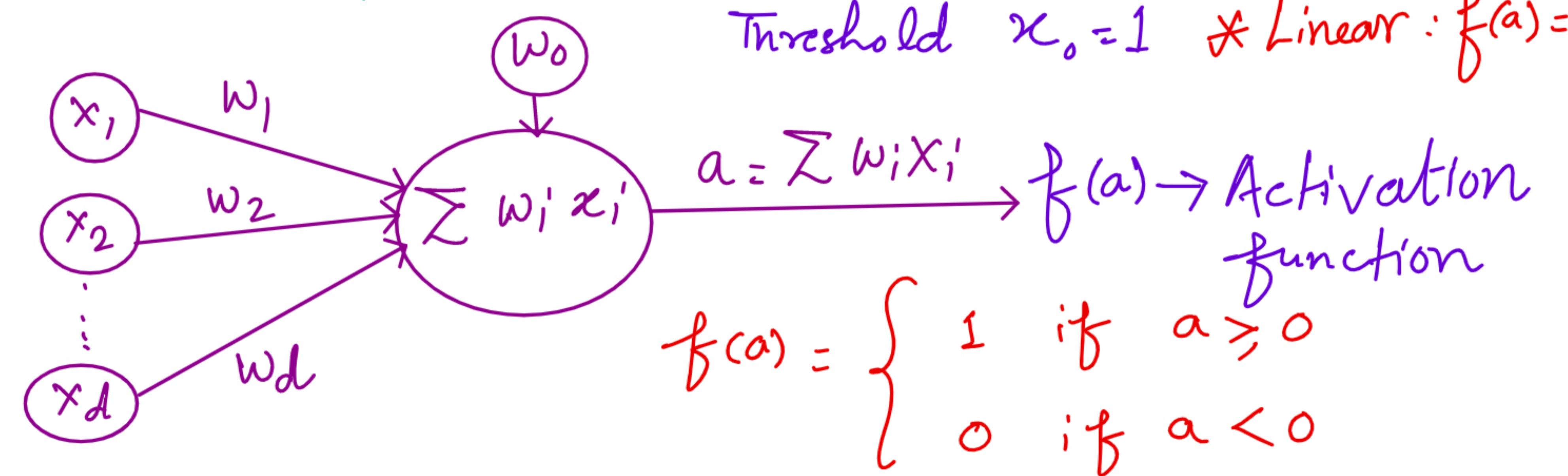
Neuron ↓
Fires (transfer infor. to next neuron)

Computational Neuron :-

d- input connection weight vector tries to approximate some complex electrical process.

If we us threshold logic neuron is called as

MC - culloch - Pitts Neuron



Key things in ANN

① Activation function

→ Threshold

→ Sigmoidal

→ Gaussian

→ Spiking

② Structure of Neuron

feed forward → Supervised

feed backward → Unsupervised

feed forward & backward → Competitive learning

④ Pattern Analysis's

→ Pattern classification

→ Pattern approximation

→ clustering

→ Storage & Retrieval

⑤ Types

→ Autoencoder → Restricted Boltzmann Machine

→ CNN-RCNN

→ RNN

Activation Function:-

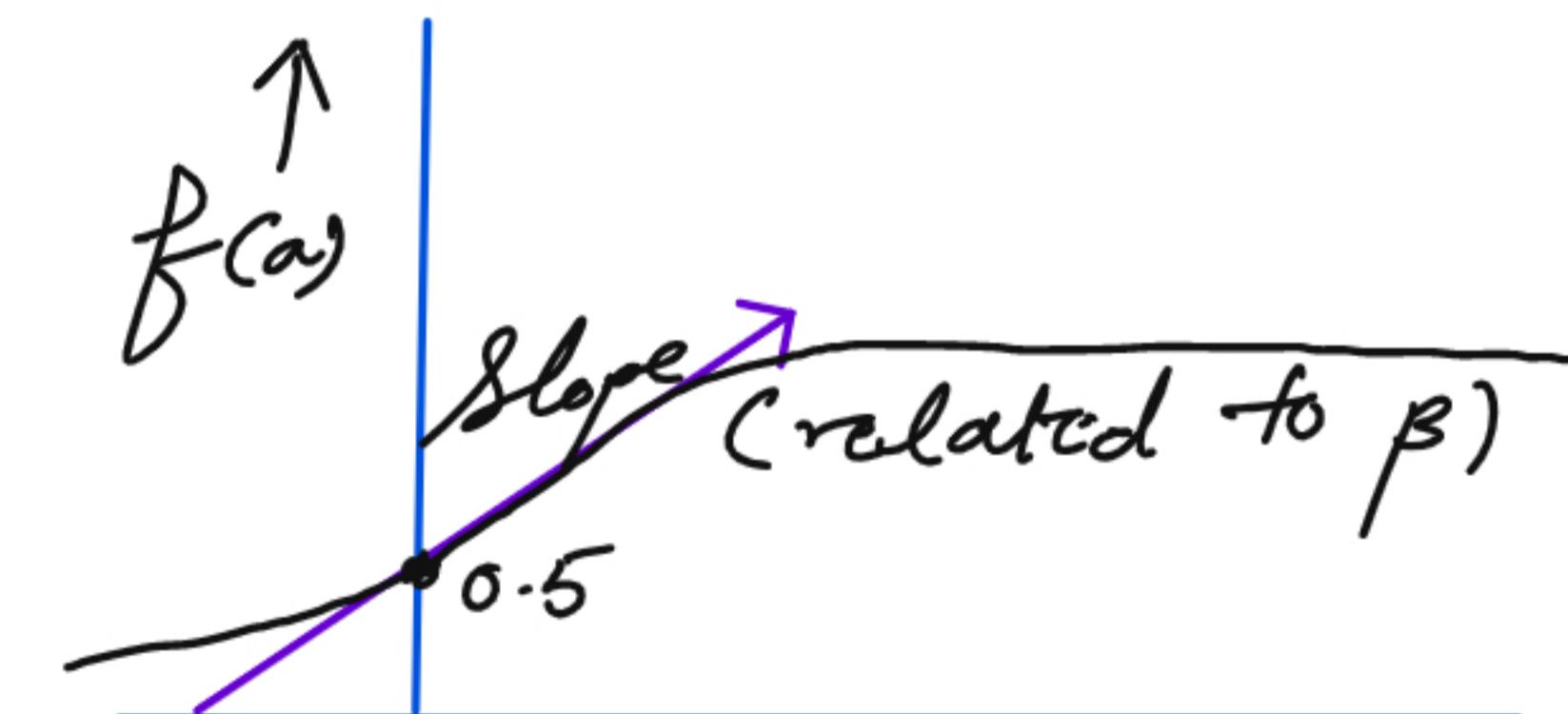
① Sigmoid Neural Activation Function

$$f(a) \in \{0 \text{ to } 1\}$$

Continuous as well as differentiable

$a \rightarrow f(a)$ in a non linear function

$$f(a) = \frac{1}{1 + e^{-\beta(a)}}$$



(11) Gaussian Activation Function

$$f(a) = e^{-\frac{(a-\mu)^2}{\sigma^2}}$$

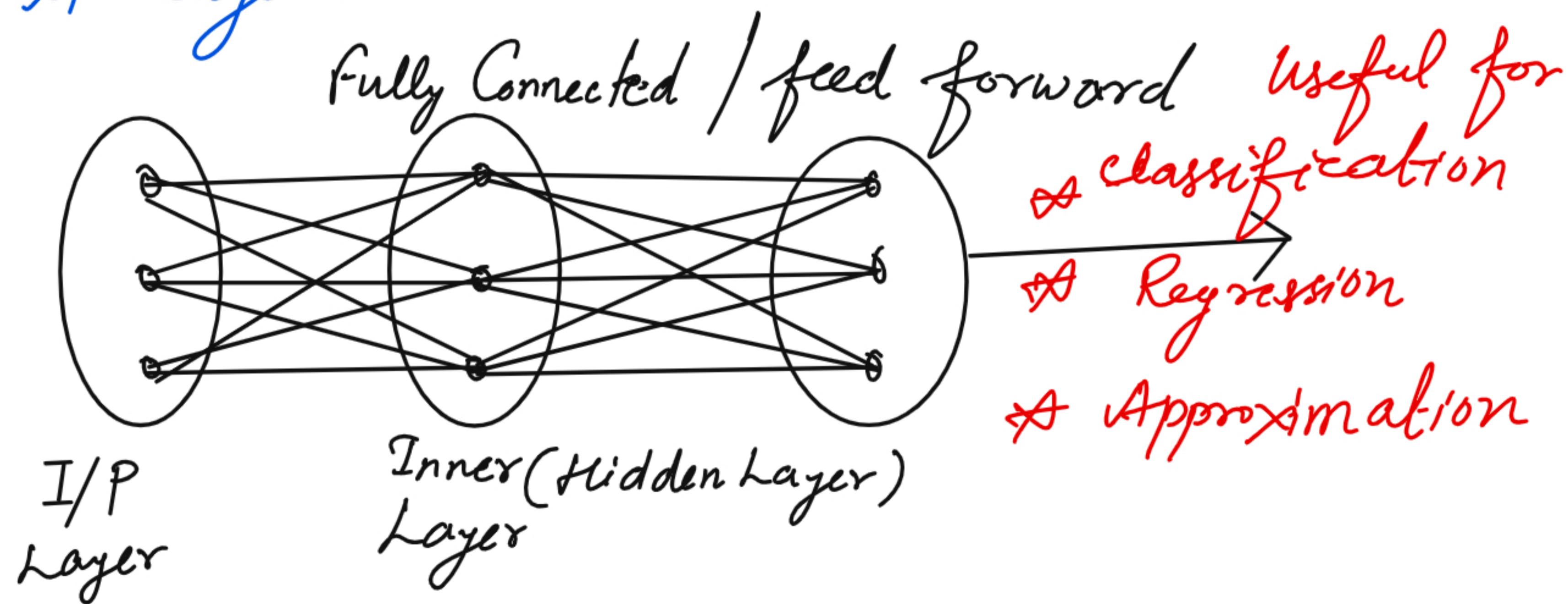
- Non linear , continuous , differentiable
- O/P got Squashed b/w {0, 1}
- used in Radial basis network
- Gives Spins \rightarrow Spiking neuron \rightarrow Pulse neural N/W
useful in VLSI

Collection of neuron can be seen as Neural N/W

Interconnection of Neuron

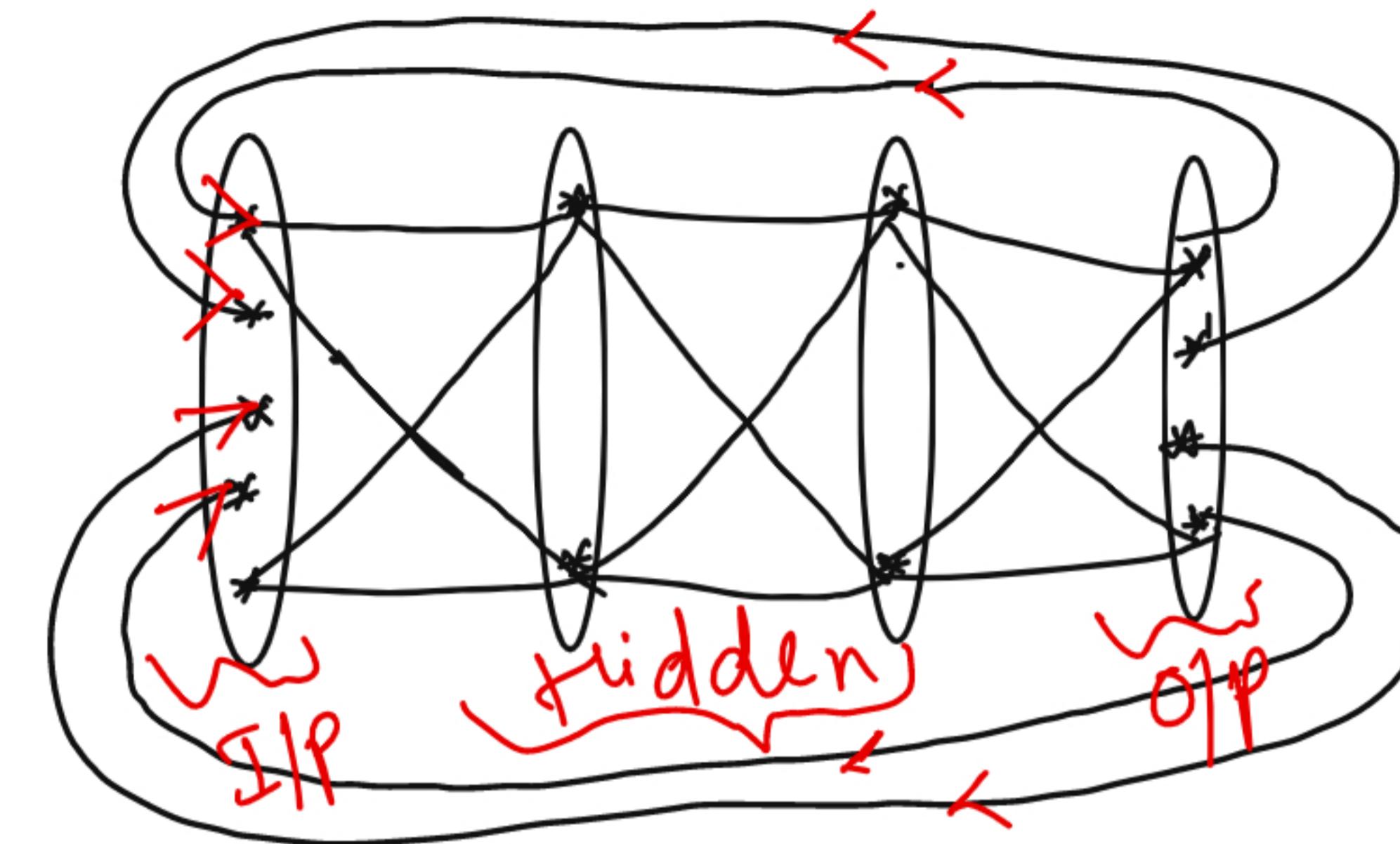
① Feed forward neural network

Layers of neuron are interconnected and connections are such that o/p of one layer are fed to next layer.



② Feed back NN :-

- * O/P has been feedback to I/P
- * O/P is not only depend upon I/P but on previous O/P
- * Generally not useful for classification
- * But useful in time series prediction pattern storage & retrieval
- * Very similar to sequential ckt. (flip flop)
as similar to associative memory.



feed backward Neural Network

Learning Strategy :-

- ① Error Correcting compute the error and minimize it.
 - Empirical risk minimization over the training data
 - Real problem is generalization.

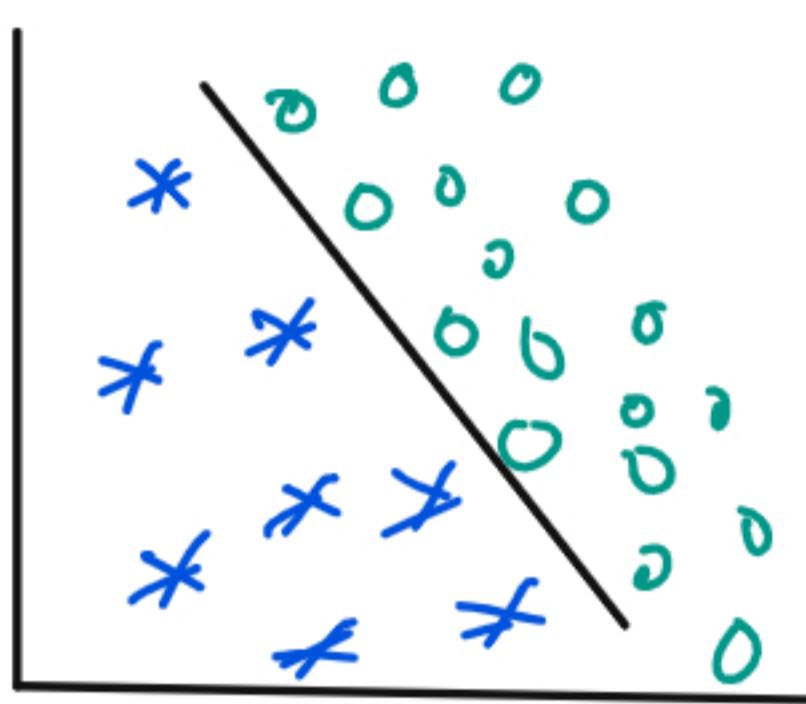
② Regularization and Statistical learning technique

Both focus on Structural risk minimization

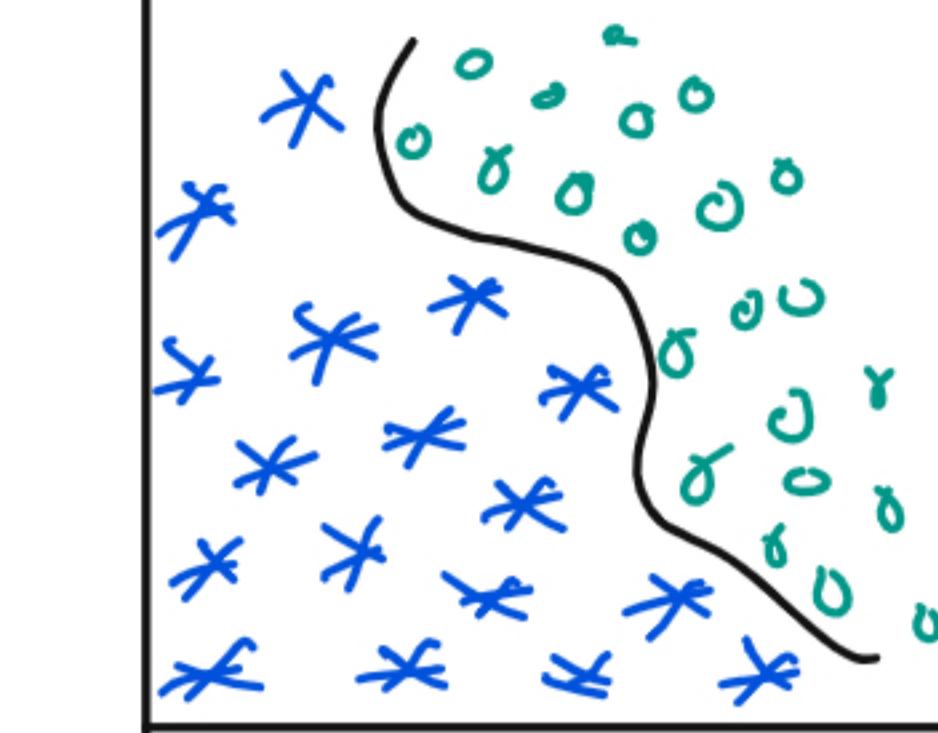
They focus to minimize the complexity
of the model

③ Competitive Learning :-

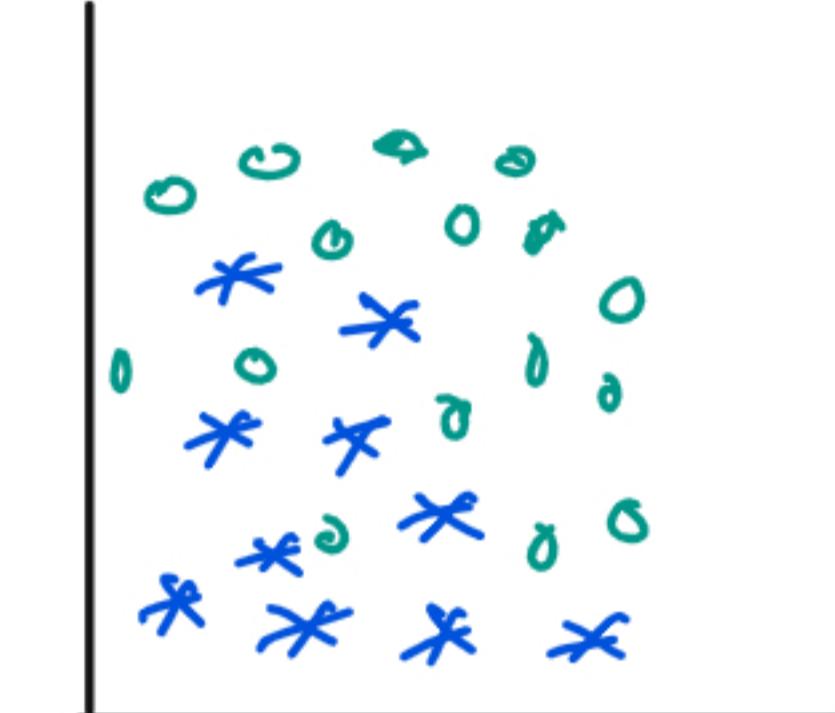
- Unsupervised learning using neighbourhood
- self organizing maps.



Linearly Separable



Nonlinear Sep.

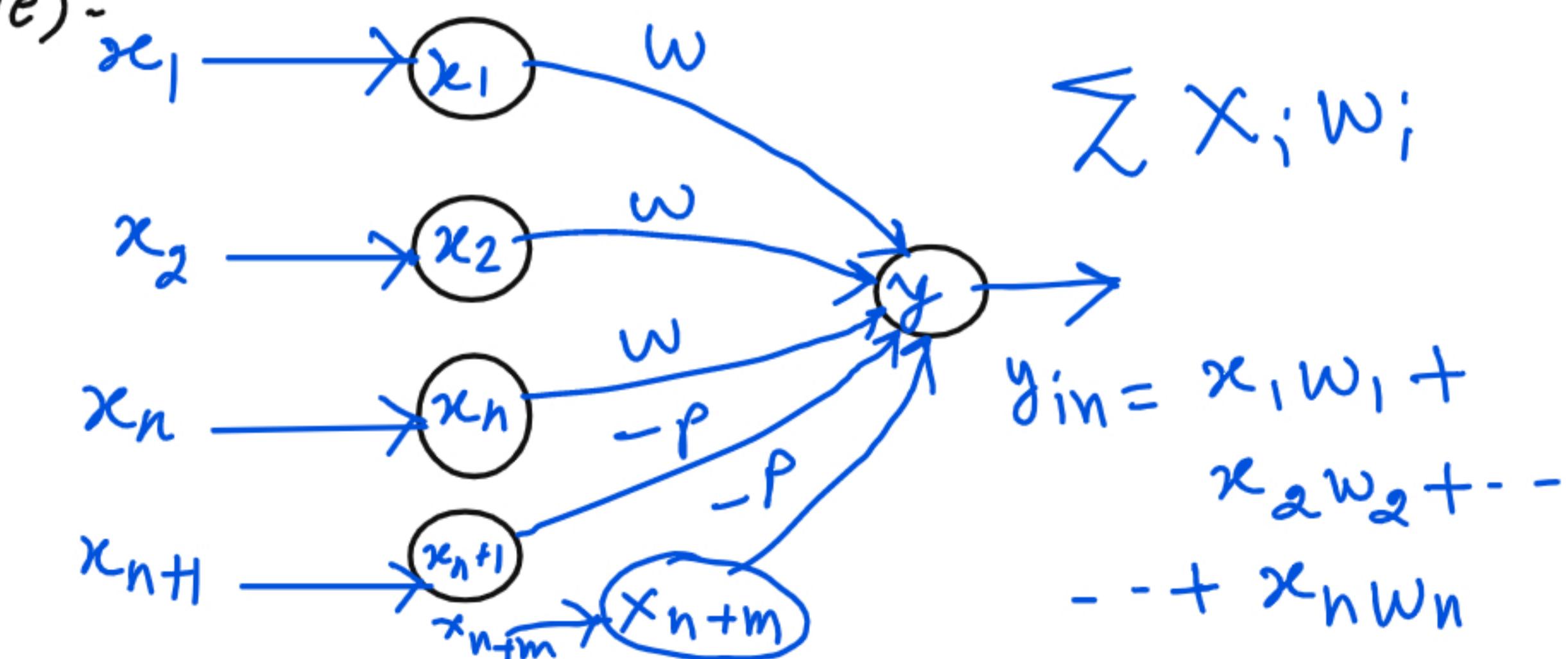


Overlapping

- * If data is linearly separable then single neuron can separate the two classes
- * For non-linear separable classes we require multiple neurons. This is piecewise stitched linear boundary
- * Multilayer neural network has to be used in order to realize such linear boundary aggregation.

McCulloch Pitt Neuron

Developed by Warren McCulloch and Walter Pitts in 1943. It is called as M-P neuron. There are two possible states in binary \rightarrow Neuron may fire (1) or may not fire (0). The weights associated with communication links may be excitatory (+ve) or inhibitory (-ve).



MP Neuron \rightarrow Activation function

$$y = f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} \geq \theta \quad \text{threshold} \\ 0 & \text{if } y_{in} < \theta \end{cases}$$

For inhibitory threshold with the activation function
should satisfy the following condition.

$$\theta > nw - p \rightarrow \text{Inhibitory weight}$$

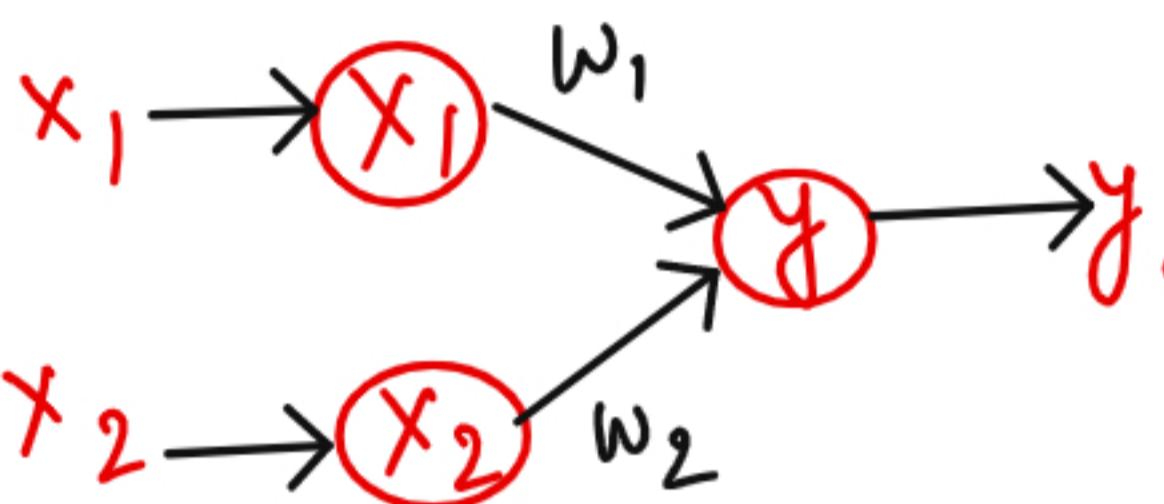


Exhibitoy wt.

- has no particular training algorithm

Implement AND NOT function using McCulloch-Pitt neuron

Binary data



x_1	x_2	y
0	0	0
0	1	0
1	0	1
1	1	0

AND NOT

00 0

01 0

10 0

11 1

Case I : Assume both weights are excitatory

$$w_1 = w_2 = 1$$

$$\theta \geq nw - P$$

$$\theta \geq 2(1) - 0$$

$\theta \geq 2$

$$y_{in} = x_1 w_1 + x_2 w_2$$

For I/P_S (0,0) $\Rightarrow y_{in} = 0 \times 1 + 0 \times 1 = 0$

$$(0,1) \Rightarrow y_{in} = 0 \times 1 + 1 \times 1 = 1$$

$$(1,0) \Rightarrow y_{in} = 1 \times 1 + 0 \times 1 = 1$$

$$(1,1) \Rightarrow y_{in} = 1 \times 1 + 1 \times 1 = 2$$

$$f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} \geq 2 \\ 0 & \text{if } y_{in} < 2 \end{cases}$$

It is not possible to fire the neuron with I/P(1,0)
only. Hence weights are not suitable

Case 2 :- Assume one weight as excitatory and other as inhibitory $w_1 = 1, w_2 = -1$

$$\text{For IIPs} : (0,0) \Rightarrow y_{in} = 0 \times 1 + 0 \times (-1) = 0$$

$$(0,1) \Rightarrow y_{in} = 0 \times 1 + 1 \times (-1) = -1$$

$$(1,0) \Rightarrow y_{in} = 1 \times 1 + 0 \times (-1) = 1$$

$$(1,1) \Rightarrow y_{in} = 1 \times 1 + 1 \times (-1) = 1 - 1 = 0$$

It is possible to fire neuron with IIP (1,0) only

$$\theta \geq nw - P$$

$$\theta \geq 2(1) - 1$$

$$\theta \geq 2 - 1$$

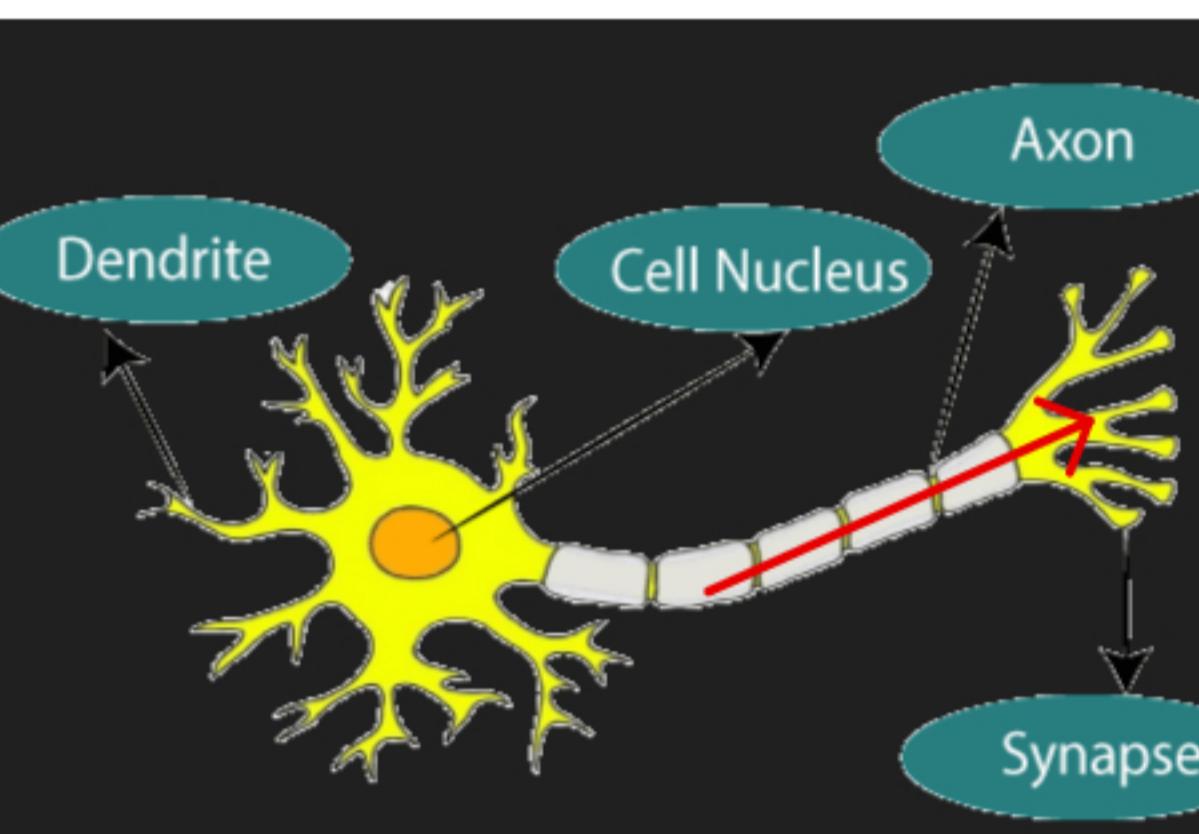
$$\theta \geq .1$$

$$y = f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} \geq 1 \\ 0 & \text{if } y_{in} < 1 \end{cases}$$



Hebb Network / Hebbian Learning Rule

In 1949 Donald Hebb → Learning → change in Synaptic Gap



When an axon of cell A is near enough to excite cell B repeatedly / Permanently firing it, then growth / metabolic change takes place in both of the cells.

Used in Unsupervised algorithm, pattern association, pattern categorization and pattern classification

Hebb's law :-

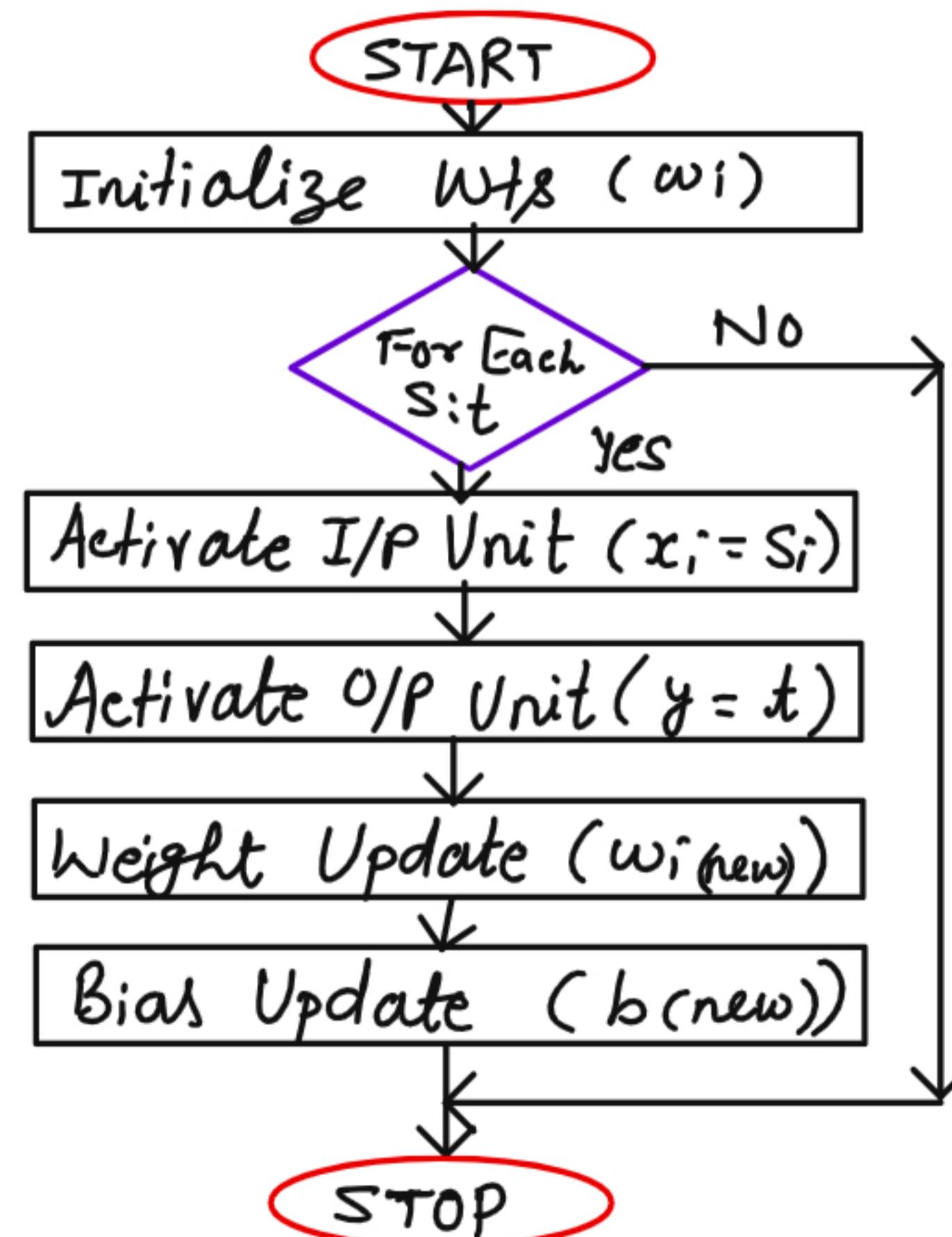
- 1) If 2 neurons fire simultaneously → weight ↑
- 2) If 2 neurons fire asynchronously → weight ↓

$$w_{i \text{ (new)}} = w_{i \text{ (old)}} + \alpha_i y$$

changes synaptic gap

$x_i \rightarrow \text{I/P vector}$, $y \rightarrow \text{O/P vector}$

Flowchart :-



Training Algorithm :- $w_i(\text{new}) = w_i(\text{old}) + x_i y$

$$b(\text{new}) = b(\text{old}) + y$$

1> Set all the weights to zero, $w_i = 0, i = 1, 2, \dots, n$

$$\text{bias} = b = 0$$

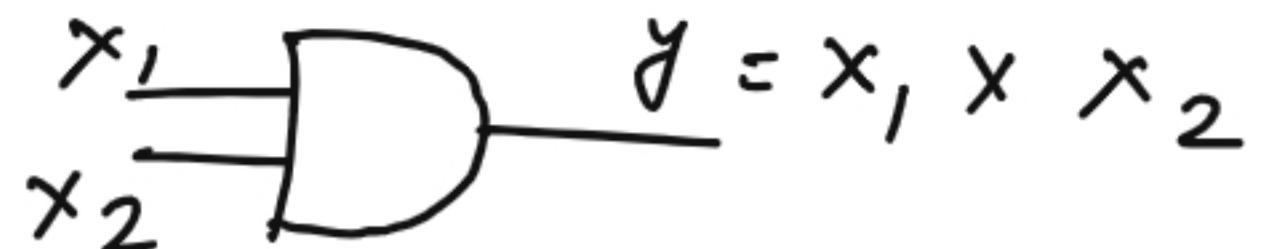
2> Repeat Step 3 to 5

3> Set the activation for I/P unit with $X_i = S_i ; i = 1, 2, \dots, n$

4> Set the corresponding o/p value $y = d$

5> Update the weights

And Gate :-



x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

Use bipolar Sigmoidal Activation function
 $(-1, 1)$ is the range for this

Input			Output	
	x_1	x_2	b	y
x_1	-1	-1	1	y_1
x_2	-1	1	1	y_2
x_3	1	-1	1	y_3
x_4	1	1	1	y_4

-1 → 0 } Assume
 1 → 1 } for
 bipo. Sigm.

Step I :- Set all the weights to zero & bias to zero

$$w = [0 \ 0 \ 0]^T, \quad b = 0$$

Step 2 : Set the activation for I/P unit with $x_i = s_i$
 $i = 1 \text{ to } 4$

$$x_1 = [-1 \quad -1 \quad 1]^T$$

$$x_2 = [-1 \quad 1 \quad 1]^T$$

$$x_3 = [1 \quad -1 \quad 1]^T$$

$$x_4 = [1 \quad 1 \quad 1]^T$$

Step 3 : 1st Iteration for $w_{\text{new}} = w_{\text{old}} + x_i \gamma$

$$= [0 \ 0 \ 0]^T + [-1 \ -1 \ 1] [-1]$$

$$= [0 \ 0 \ 0]^T + [1 \ 1 \ -1]$$

$$= [1 \ 1 \ -1]^T$$

IInd Iteration $w_{\text{new}} = [1 \ 1 \ -1]^T + [-1 \ 1 \ 1]^T [-1]$
 $= [1 \ 1 \ -1]^T + [1 \ -1 \ -1]^T$
 $= [2 \ 0 \ -2]^T$

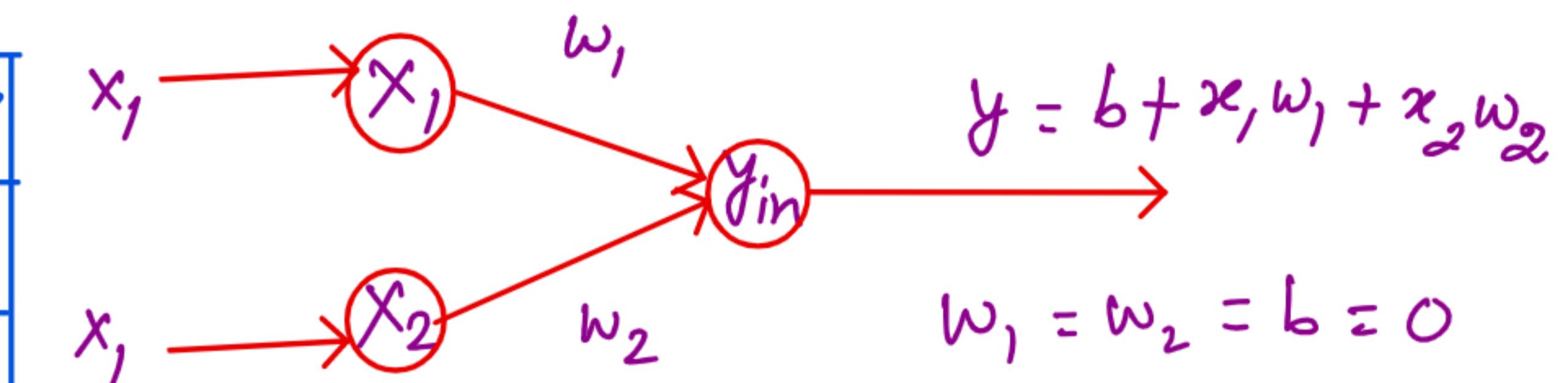
IIIrd Iteration $w_{\text{new}} = [2 \ 0 \ -2]^T + [1 \ -1 \ 1]^T [-1]$
 $= [2 \ 0 \ -2]^T + [-1 \ 1 \ -1]^T$
 $= [1 \ 1 \ -3]^T$

IVth Iteration $w_{\text{new}} = [1 \ 1 \ -3]^T + [1 \ 1 \ 1]^T [1]$
 $= [1 \ 1 \ -3]^T + [1 \ 1 \ 1]^T$
 $= [2 \ 2 \ -2]^T$

Design a Hebb network to implement logical AND fun.
Use bipolar input and targets.

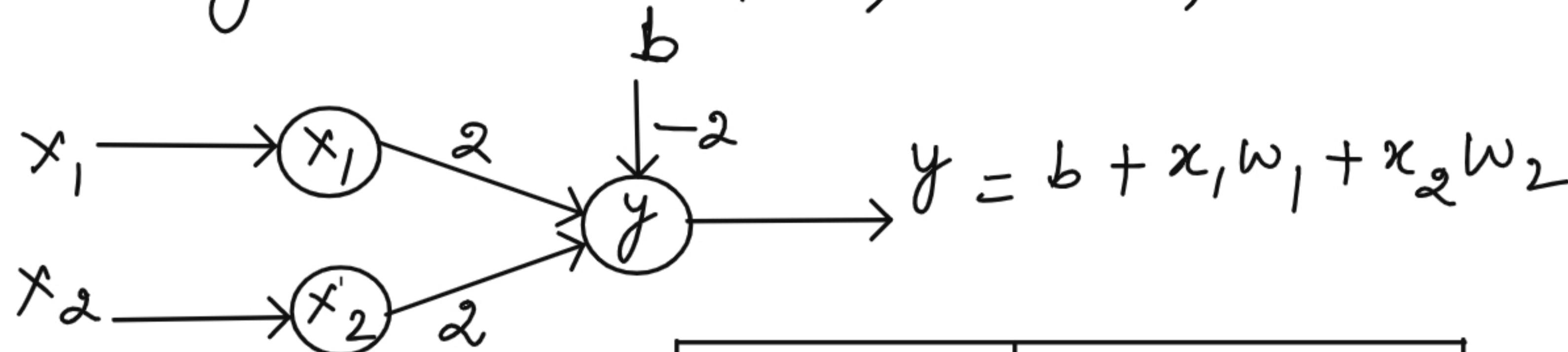
$$\underline{\text{AND}} \quad x_1 \cdot x_2 = y$$

Inputs		Targets
x_1	x_2	y
1	1	1
1	-1	-1
-1	1	-1
-1	-1	1



INPUTS				WEIGHT CHANGE			NEW WEIGHT		
x_1	x_2	b	y	Δw_1 x_1y	Δw_2 x_2y	Δb y	$w_1(\text{new})$ $w_1(\text{old}) + \Delta w_1$	$w_2(\text{new})$ $w_2(\text{old}) + \Delta w_2$	$b(\text{new})$ $b(\text{old}) + y$
1	1	1	1	1	1	1	0+1 = 1	0+1 = 1	0+1 = 1
1	-1	1	-1	-1	1	1	1+(-1) = 0	1+1 = 2	1+(-1) 0
-1	1	1	-1	1	-1	-1	0+1 = 1	2+(-1) = 1	0+(-1) = -1
-1	-1	1	-1	1	1	-1	1+1 = 2	1+1 = 2	-1+(-1) = -2

Final weights are :- $w_1 = 2, w_2 = 2, b = -2$



$$\begin{aligned} y &= -2 + (-1)(2) + (-1)(2) = \\ &= -2 - 2 - 2 = -6 \end{aligned}$$

$$\begin{aligned} y_2 &= -2 + (-1)(2) + (1)(2) = \\ &= -2 - 2 + 2 = -2 \end{aligned}$$

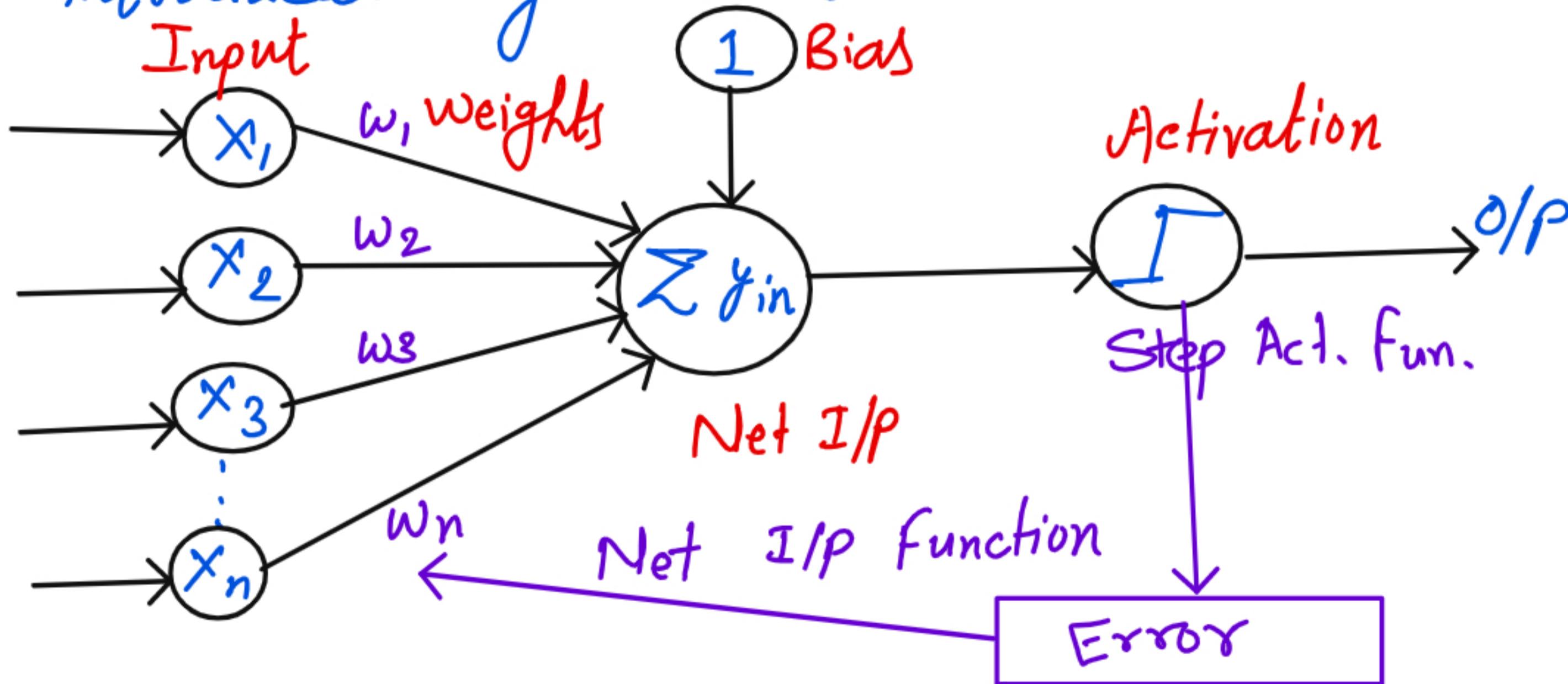
$$\begin{aligned} y_3 &= -2 + (1)(2) + (-1)(2) \\ &= -2 + 2 - 2 = -2 \end{aligned}$$

$$\begin{aligned} y_4 &= -2 + (1)(2) + (1)(2) \\ &= -2 + 2 + 2 = 2 \end{aligned}$$

x_1	x_2	x_1	x_2	y
0	0	-1	-1	-6
0	1	-1	1	-2
1	0	1	-1	-2
1	1	1	1	2

Perceptron Neural Network

Perceptron network come under single layer feed forward neural network. It is the simplest form of neural network to classify data into two classes. It is introduced by Frank Rosenblatt in 1957.



Two types of Perceptrons :-

- ① Single Layer
- ② Multi Layer - 2 or more layers & have greater processing power

The output of perception network

$$y_{in} = x_1 w_1 + x_2 w_2 + x_3 w_3 + \dots + x_n w_n + b$$

$$y = f(y_{in})$$

$f(y_{in})$ is activation function (Step)

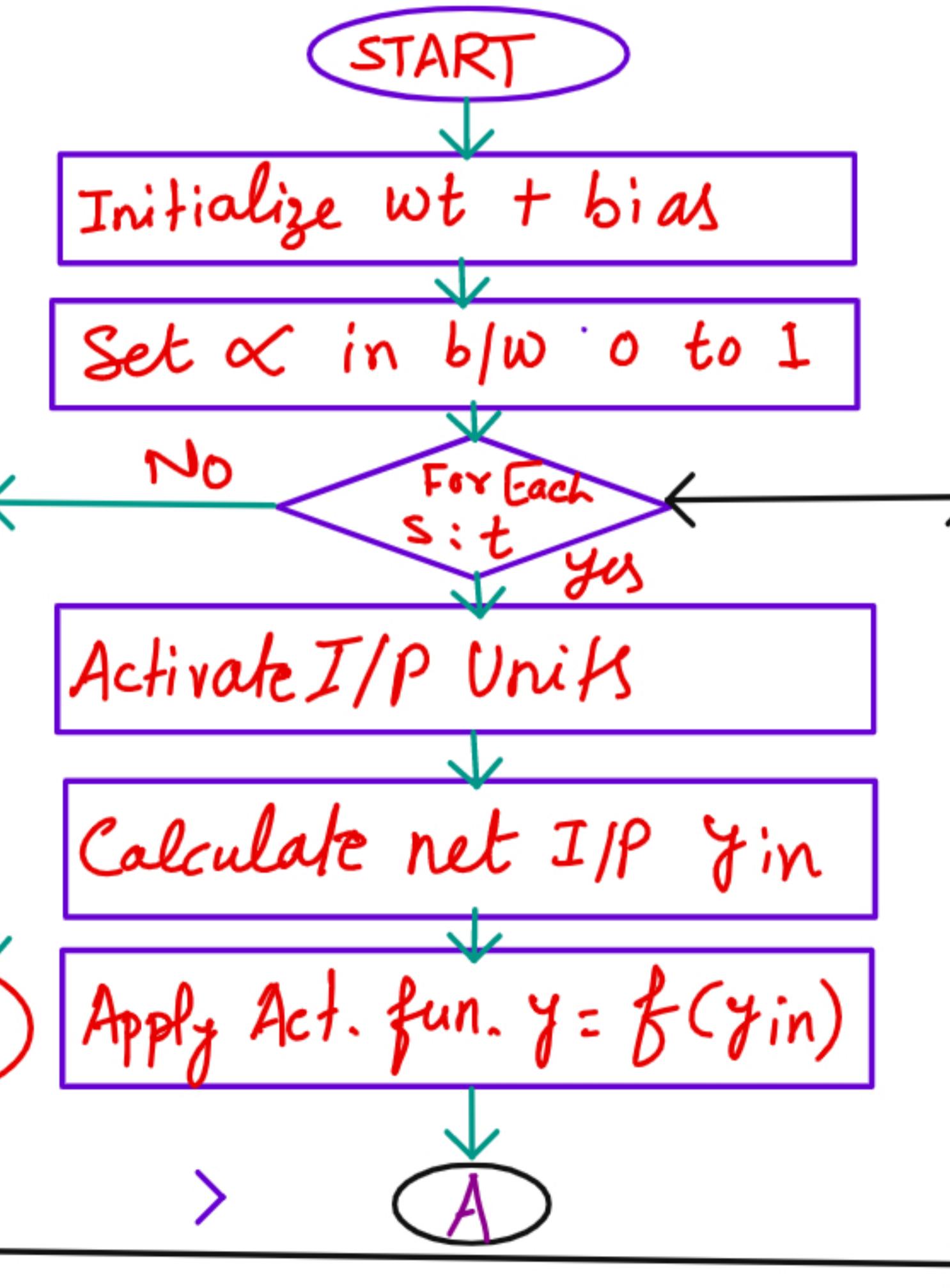
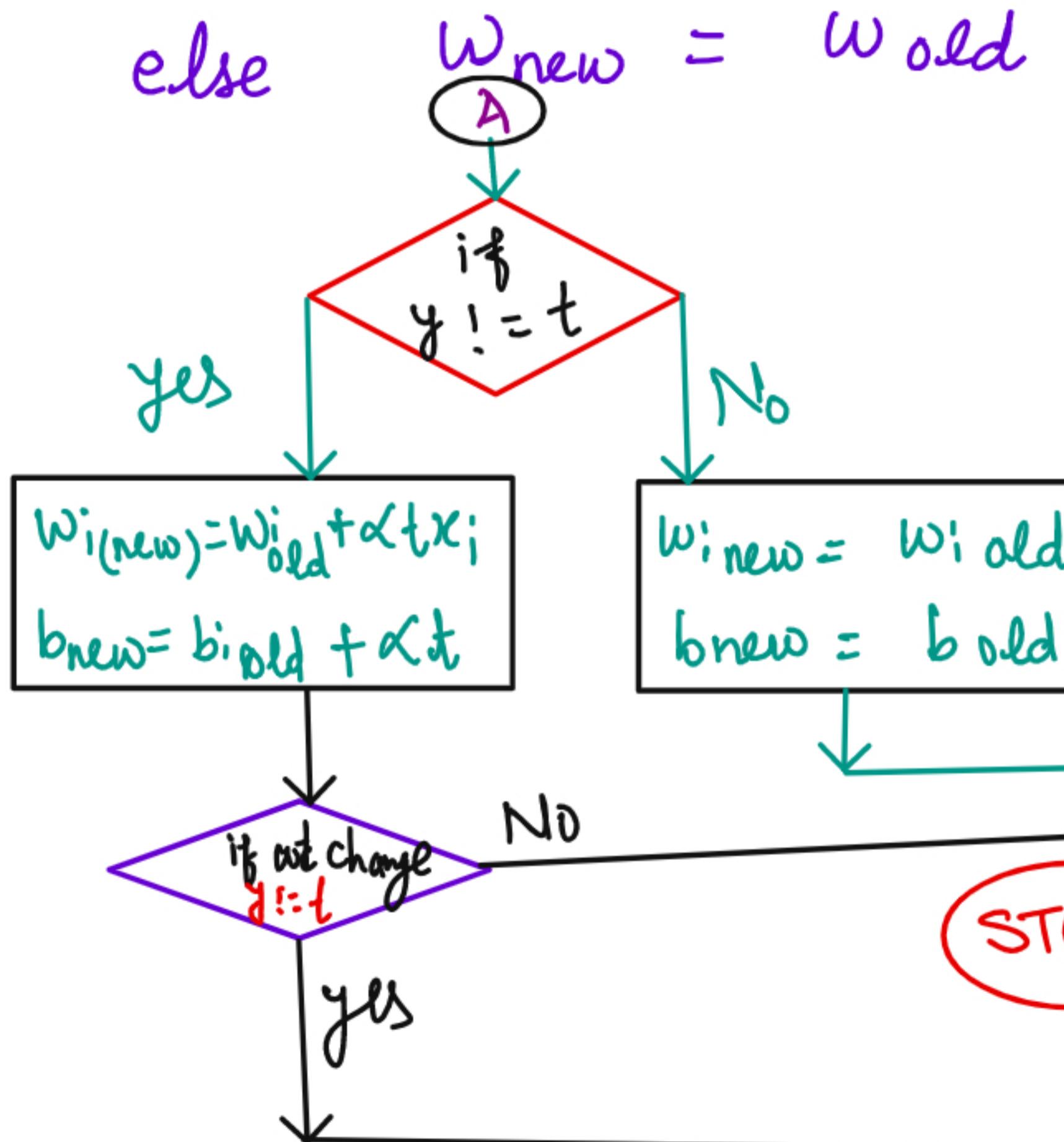
$$y = f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} > \theta \\ 0 & \text{if } -\theta \leq y_{in} < \theta \\ -1 & \text{if } y_{in} < -\theta \end{cases}$$

The weight updation

if $y \neq t$ then

$$w_{\text{new}} = w_{\text{old}} + \alpha t x \quad \downarrow \text{Learning rate}$$

else $w_{\text{new}} = w_{\text{old}}$



Training

Algorithm :-

Step 0:- Initialize wts and bias

$$\alpha = 1$$

Step 1:- Perform step 2-5 until final stopping condition is false

Step 2:- Perform step 3-4 for each training pair

Step 3:- Calculate the output

$$y_{in} = b + \sum_{i=1}^n x_i w_i$$

Apply activation function

$$y = f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} > \theta \\ 0 & \text{if } -\theta \leq y_{in} \leq \theta \\ -1 & \text{if } y_{in} < -\theta \end{cases}$$

Step 4: weight and bias adjustment

if $y \neq t$ then

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

$$b(\text{new}) = b(\text{old}) + \alpha t;$$

else

$$w_i(\text{new}) = w_i(\text{old})$$

$$b(\text{new}) = b(\text{old})$$

Testing Algorithm :-

Step 0 :- The initial weights to be used here are taken from training algorithm

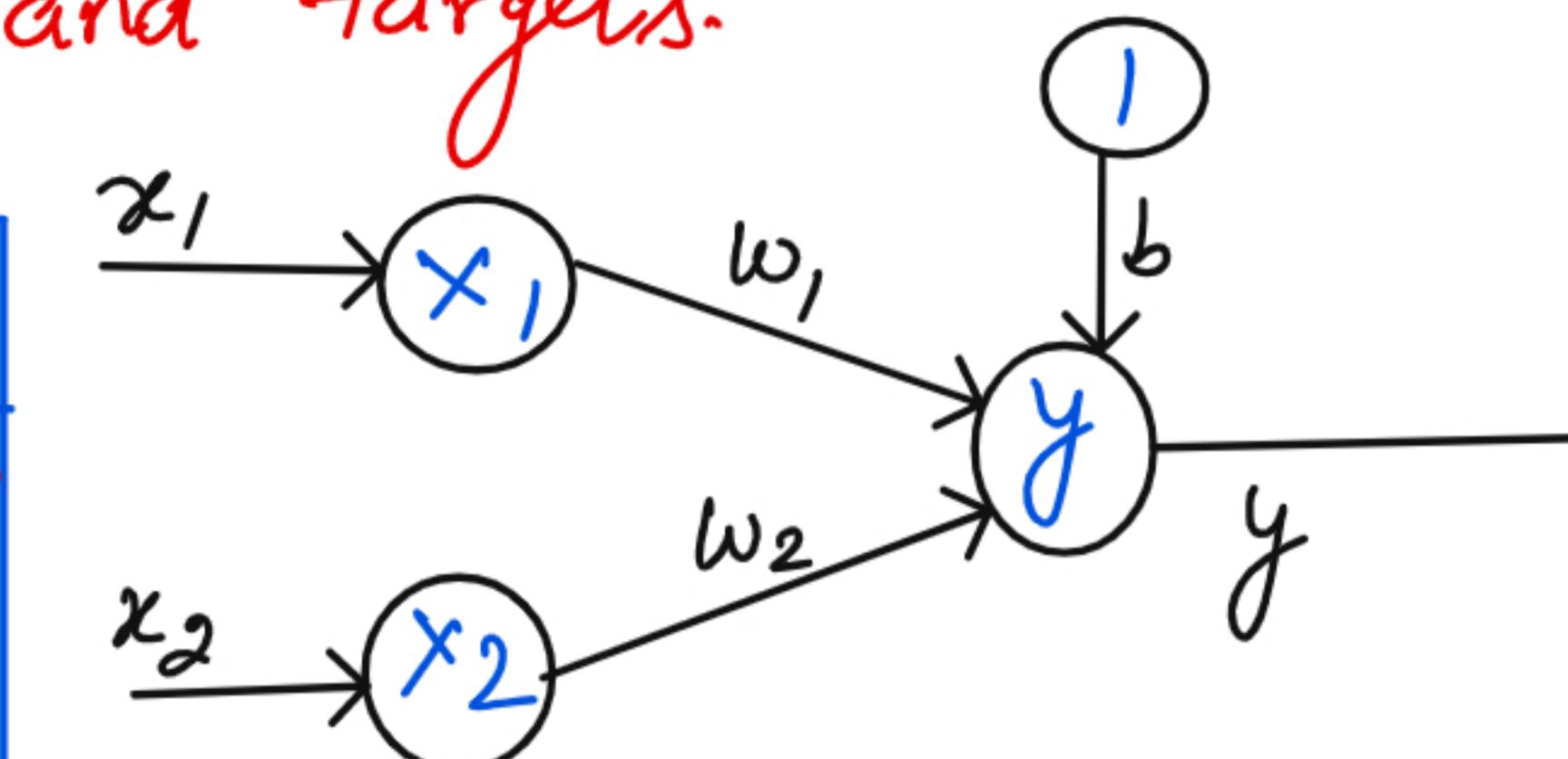
Step 1 :- For each training I/P vector x to be classified perform step 2-3

Step 2: Set activations $y_i = b + \sum_{i=1}^n x_i w_i$

$$y = f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} > \theta \\ 0 & \text{if } -\theta \leq y_{in} \leq \theta \\ -1 & \text{if } y_{in} < -\theta \end{cases}$$

Implement AND function using Perceptron network
for bipolar I/Ps and targets.

x_1	x_2	t
1	1	1
1	-1	-1
-1	1	-1
-1	-1	-1



$$y_{in} = x_1 w_1 + x_2 w_2 + b$$

$$\theta = 0$$

$$0 \rightarrow -1$$

$$1 \rightarrow 1$$

$$y = f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} > 0 \\ 0 & \text{if } y_{in} = 0 \\ -1 & \text{if } y_{in} < 0 \end{cases}$$

$$w_1(\text{new}) = w_1(\text{old}) + \alpha t x_1 \Delta w_1$$

$$w_2(\text{new}) = w_2(\text{old}) + \alpha t x_2 \Delta w_2$$

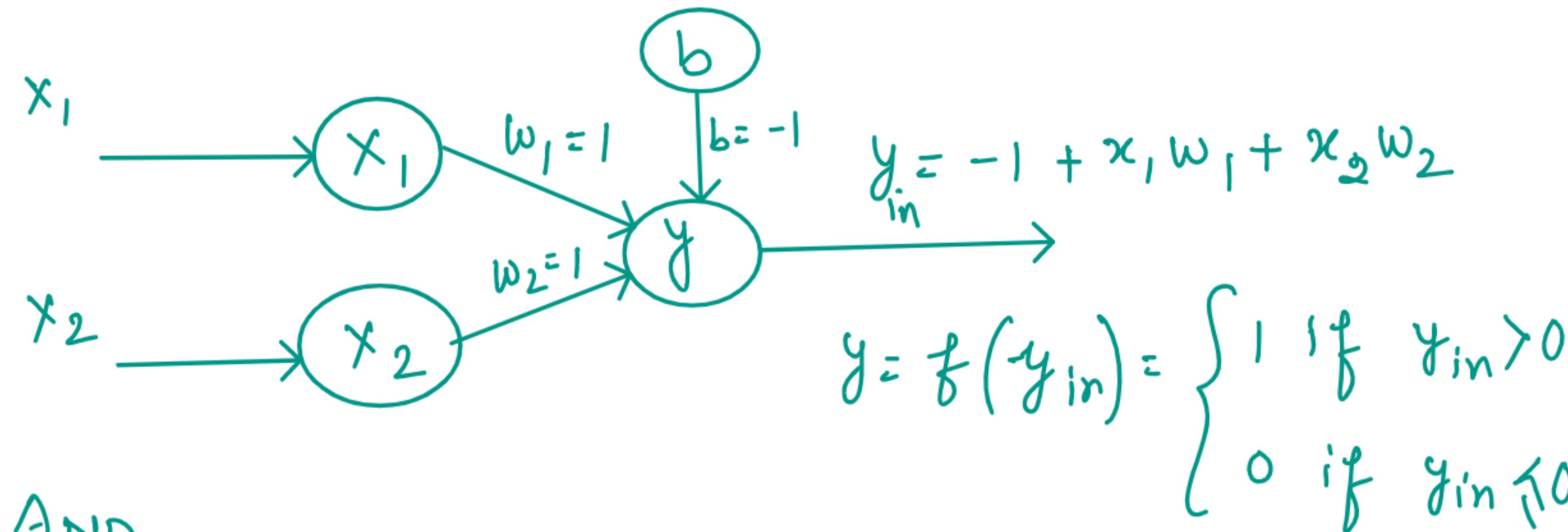
$$b(\text{new}) = b(\text{old}) + \alpha t \Delta b$$

Input			Target	Net Input	Cal o/p	Weight changes			New Weights		
x_1	x_2	b	(t)	y_{in}	y	Δw_1	Δw_2	Δb	w_1	w_2	b
Epoch - 1				$y_{in} = b + x_1 w_1 + x_2 w_2$		$\alpha \cdot t x_1$	$\alpha \cdot t x_2$	αt	$w_{old} + \Delta w_1$	$w_{old} + \Delta w_2$	$b_{old} + \Delta b$
1	1	1	1	$= 0 + 0 + 0$ $= 0$	0	$1 \times 1 x_1$ $= 1$	$1 \times 1 x_1$ $= 1$	1×1 $= 1$	$0 + 1$ $= 1$	$0 + 1$ $= 1$	$0 + 1$ $= 1$
1	-1	1	-1	$= 1 + 1 + (-1 \times 1)$ $= 1 + 1 - 1$ $= 1$	1	$1 \times (-1) x_1$ $= -1$	$1 \times (-1) x_2$ $= 1$	$1 \times (-1)$ $= -1$	$1 - 1$ $= 0$	$1 + 1$ $= 2$	$1 + (-1)$ $= 0$
-1	1	1	-1	$= 0 + (-1 \times 0)$ $+ (1 \times 2)$ $= 0 + 0 + 2$ $= 2$	1	$1 \times (-1)$ $\times (-1)$ $= 1$	$1 \times (-1)$ $\times (1)$ $= -1$	$1 \times (-1)$ $= -1$	$0 + 1$ $= 1$	$2 + (-1)$ $= 1$	$0 + (-1)$ $= -1$
-1	-1	1	-1	$= -1 + (-1) \times (1)$ $+ (-1) \times (1)$ $= -1 - 1 - 1$ $= -2$	-1	0	0	0	1	1	-1
Equal											

$w_1 = 1, w_2 = 1, b = -1$

Input			Target	Net Input	Cal O/P	Weight Change			New Weights		
x_1	x_2	b	t	y_{in}	y	Δw_1	Δw_2	Δb	w_1	w_2	b
Epoch-2				$= b + x_1 w_1 + x_2 w_2$		$\alpha t x_1$	$\alpha t x_2$	αt	$w_{old} + \Delta w_1$	$w_{old} + \Delta w_2$	$b_{old} + \Delta b$
1	1	1	1	$= -1 + x_1 + x_2 $ $= 1$	1 Equal	0	0	0	1	1	-1
1	-1	1	-1	$= -1 + (1 \times 1) + (-1 \times 1)$ $= -1 + 1 - 1$ $= -1$	-1 Equal	0	0	0	1	1	-1
-1	1	1	-1	$= -1 + (-1 \times 1) + (1 \times 1)$ $= -1 - 1 + 1$ $= -1$	-1 Equal	0	0	0	1	1	-1
-1	-1	1	-1	$= -1 + (-1 \times 1) + (-1 \times 1)$ $= -3$	-1 Equal	0	0	0	1	1	-1

.



AND

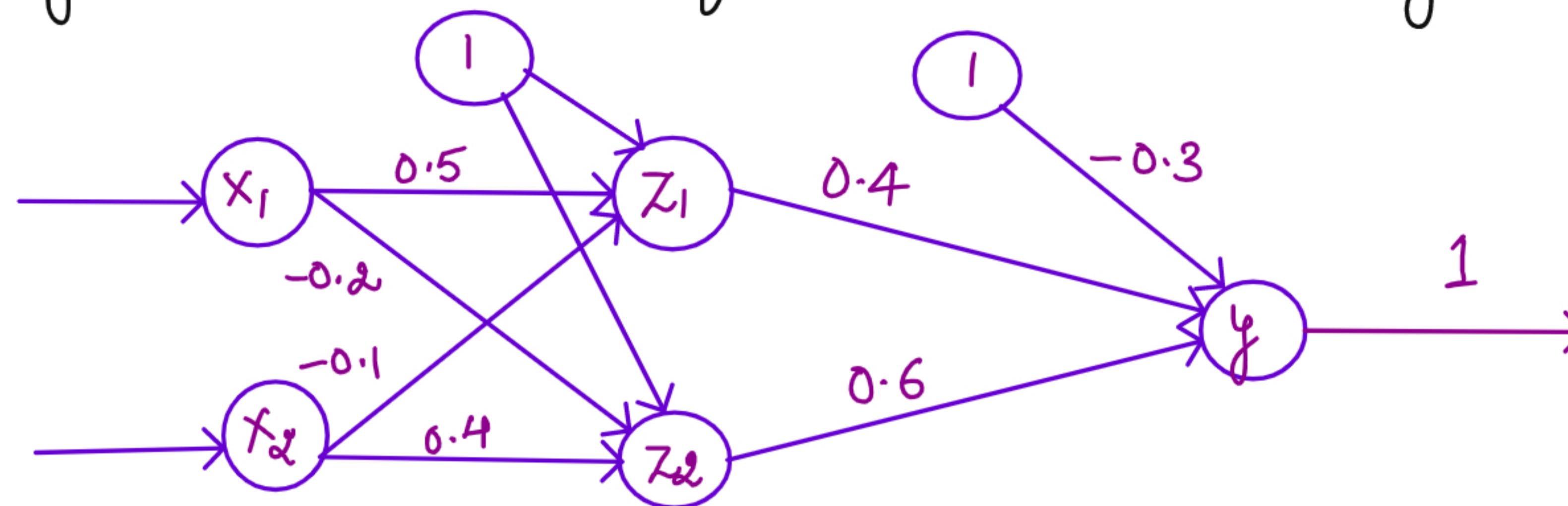
$$x_1 \quad x_2 \quad y \rightarrow y_{in} = -1 + (-1)x(1) + (-1)x(1) = -1 - 1 - 1 = -3 = 0$$

$$\begin{matrix} -1 & -1 & -1 \end{matrix} \rightarrow y_{in} = -1 + (-1)x(1) + (1)x(1) = -1 - 1 + 1 = -1 = 0$$

$$\begin{matrix} 1 & -1 & -1 \end{matrix} \rightarrow y_{in} = -1 + (1)x(1) + (-1)x(1) = -1 + 1 - 1 = -1 = 0$$

$$\begin{matrix} 1 & 1 & 1 \end{matrix} \rightarrow y_{in} = -1 + (1)x(1) + (1)x(1) = -1 + 1 + 1 = 1 = 1$$

For the input vector $[1, 0]$, train the MLP network using binary sigmoidal activation function with the target value 1.



$$\begin{aligned}
 \text{Summation of input at } z_1: z_{in1} &= b_1 + x_1 w_{11} + x_2 w_{21} \\
 &= 0.3 + 1 \times 0.5 + 0 \times (-0.1) \\
 &= 0.8
 \end{aligned}$$

$$\text{Output at } z_1 = f(z_{in1}) = \frac{1}{1 + e^{-0.8}} = 0.69$$

Summation of input at $z_2 = z_{in2} = b_2 + x_1 w_{12} + x_2 w_{22}$

$$= 0.5 + 1 \times (-0.2) + 0(0.4)$$

$$= 0.3$$

Output at $z_2 = f(z_{in2}) = \frac{1}{1 + e^{-0.3}} = 0.5744$

Summation of input at output neuron (Y) = $y_{in} = b + z_1 w_1 + z_2 w_2$

$$y_{in} = -0.3 + 0.69 \times (0.4) + 0.5744 \times (0.6) = 0.32064$$

output at $f = f(z_{in1}) = \frac{1}{1 + e^{-0.32064}} = 0.4205$

Calculating error at the output layer $\delta_k = (t_k - Y_k) f'(y_{in k})$

$$f'(y_{in k}) = f(y_{in k}) [1 - f(y_{in k})] = 0.4205 \times (1 - 0.4205) = 0.0244$$

Error can be computed as $\delta_1 = (1 - 0.4205)(0.8244) = 0.0141$.
Change in weights between the hidden and output layers

$$\Delta w_1 = \alpha \delta z_1 = (0.25)(0.0141)(0.6) = 0.00243$$

$$\Delta w_2 = \alpha \delta z_2 = (0.25)(0.0141)(0.5744) = 0.00202$$

$$\Delta w_0 = \alpha \delta = (0.25)(0.0141) = 0.00353$$

Computing error between hidden and input layer

$$\delta_j = S_{inj} f'(z_{inj}) = \sum \delta_k w_{jk}$$

$$S_{in1} = \delta_1 w_{11} = (0.0141)(0.4) = 0.00564$$

$$S_{in2} = \delta_2 w_{21} = (0.0141)(0.6) = 0.00846$$

$$\text{Error at } z_1 = s'(z_{in1}) = \int(z_{in})[1 - s(z_{in1})]$$
$$= (0.69)[1 - 0.69] = 0.2139$$

$$\delta_1 = \delta_{in1} s'(z_{in1}) = (0.00564)(0.2139) = 0.001206$$

$$\text{Error at } z_2 = s'(z_{in2}) = \int(z_{in2})[1 - s(z_{in2})]$$
$$= (0.5744)[1 - 0.5744] = 0.244$$

$$\delta_2 = \delta_{in2} s'(z_{in2}) = (0.00846)(0.244) = 0.00206$$

Change in weight between input and hidden layers -

$$\Delta V_{11} = \alpha \delta_1 x_1 = (0.25)(0.001206)(1) = 0.0003$$

$$\Delta V_{21} = \alpha \delta_1 x_2 = (0.25)(0.001206)(0) = 0$$

$$\Delta V_{01} = \alpha \delta_1 = (0.25)(0.001206) = 0.0003$$

$$\Delta V_{12} = \alpha \delta_2 x_1 = (0.25)(0.00206)(1) = 0.00051$$

$$\Delta V_{22} = \alpha \delta_2 x_2 = (0.25)(0.00206)(0) = 0$$

$$\Delta V_{02} = \alpha \delta_2 = (0.25)(0.00206) = 0.00051$$

Final Updated new weights:-

$$V_{11}(\text{new}) = V_{11}(\text{old}) + \Delta V_{11} = 0.5 + 0.0003 = 0.5003$$

$$V_{21}(\text{new}) = V_{21}(\text{old}) + \Delta V_{21} = -0.1 + 0 = -0.1$$

$$V_{01}(\text{new}) = V_{01}(\text{old}) + \Delta V_{01} = 0.3 + 0.0003 = 0.3003$$

$$V_{21}(\text{new}) = V_{21}(\text{old}) + \Delta V_{21} = -0.2 + 0.00051 = -0.1995$$

$$V_{22}(\text{new}) = V_{22}(\text{old}) + \Delta V_{22} = 0.4 + 0 = 0.4$$

$$V_{02}(\text{new}) = V_{02}(\text{old}) + \Delta V_{02} = 0.5 + 0.00051 = 0.50051$$

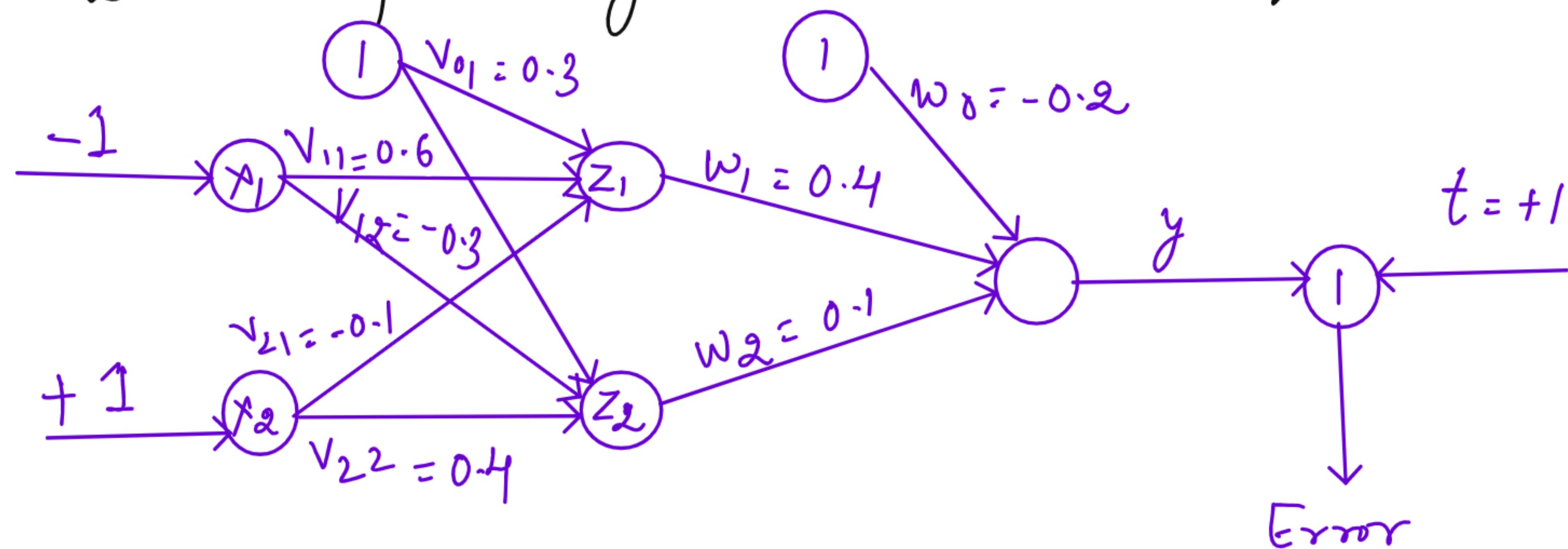
$$w_1(\text{new}) = w_1(\text{old}) + \Delta w_1 = 0.4 + 0.00243 = 0.40243$$

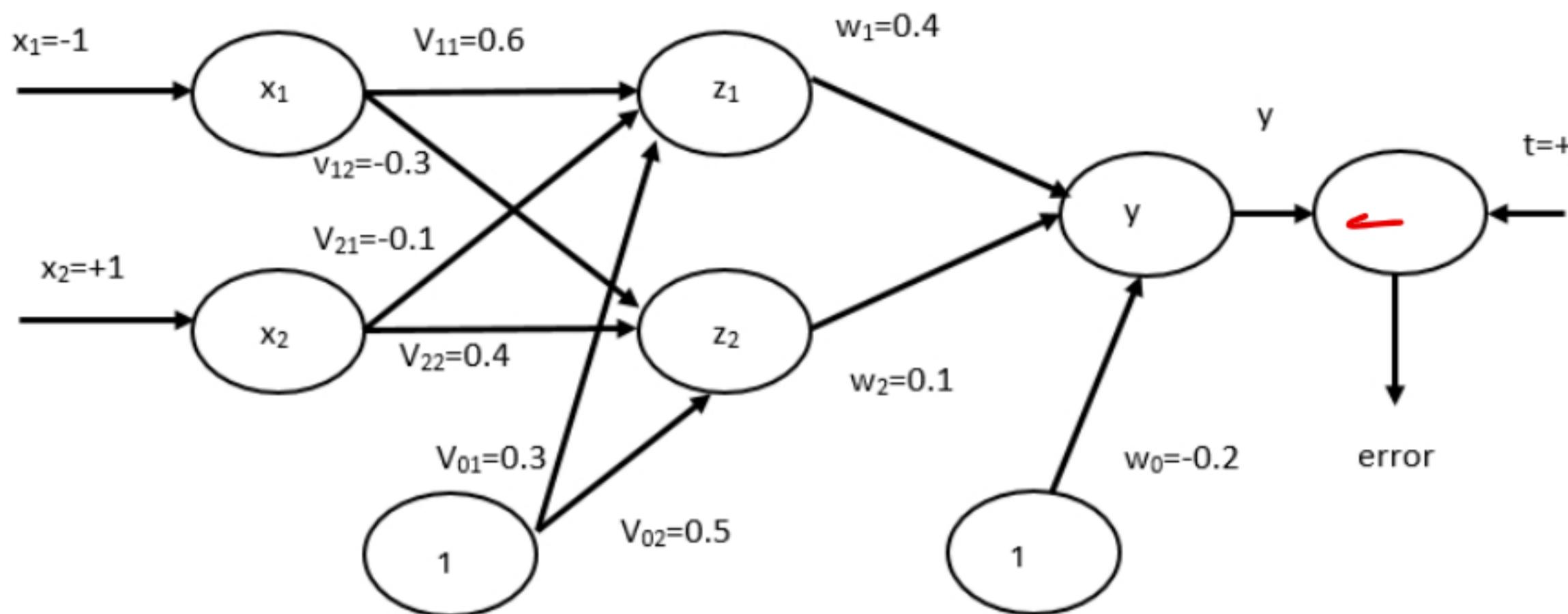
$$w_2(\text{new}) = w_2(\text{old}) + \Delta w_2 = 0.6 + 0.00202 = 0.60202$$

$$w_0(\text{new}) = w_0(\text{old}) + \Delta w_0 = -0.3 + 0.00353 = -0.29647$$

Thus the weights are updated for the given network. In the similar manner, the new weights are updated for all the input patterns. Repeat the same until the error is less than the threshold value.

find the new weights using back propagation algorithm for the network. The network is presented with the input patterns $[-1, 1]$ and the target output is $+1$. Use a learning rate 0.25 and bipolar sigmoidal activation function.





Initial Weights = $\{v_{01}, v_{11}, v_{21}\} = \{0.3, 0.6, -0.1\}$

$$\{v_{02}, v_{12}, v_{22}\} = \{0.5, -0.3, 0.4\}$$

$$\{w_0, w_1, w_2\} = \{-0.2, 0.4, 0.1\}$$

$$[x_1, x_2] = [-1, +1]$$

learning rate = 0.25

Bipolar Sigmoidal Ac. fun. = $f(x) = \frac{1-e^{-x}}{1+e^{-x}}$

target output = +1 , In Bipolar Sigmoidal activation function we have to find first derivative

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

If binary sigmoidal then $f(x) = \frac{1}{1+e^{-x}}$

STEP 1 :- Calculate at the hidden layer (z_1, z_2)

$$\begin{aligned} Z_{in_1} &= 0.3 \times 1 + (-1 \times 0.6) + (1 \times (-0.1)) \\ &= 0.3 - 0.6 - 0.1 = -0.4 \end{aligned}$$

$$\begin{aligned} Z_{in_2} &= 0.5 \times 1 + (-1 \times 0.3) + (1 \times 0.4) \\ &= 0.5 + 0.3 + 0.4 = 1.2 \end{aligned}$$

$$z_1 = f(z_{in1}) = \frac{1 - e^{-z_{in1}}}{1 + e^{-z_{in1}}} = \frac{1 - e^{(-0.4)}}{1 + e^{(-0.4)}} = \frac{1 - e^{0.4}}{1 + e^{0.4}}$$

$$z_1 = -0.1974$$

$$z_2 = f(z_{in2}) = \frac{1 - e^{-z_{in2}}}{1 + e^{-z_{in2}}} = \frac{1 - e^{(-1.2)}}{1 - e^{(-1.2)}} = 0.537$$

$$z_2 = 0.537$$

STEP 2 :- Calculate at output layer (y)

$$\begin{aligned} y_{in} &= z_1 w_1 + z_2 w_2 + 1 \times w_0 \\ &= -0.1974 \times 0.4 + 0.537 \times 0.1 + 1 \times (-0.2) \\ &= -0.22526 \end{aligned}$$

$$\gamma = f(y_{in}) = \frac{1 - e^{y_{in}}}{1 + e^{y_{in}}} = \frac{1 - e^{-(-0.22526)}}{1 + e^{(-0.22526)}} \\ = \frac{1 - e^{0.22526}}{1 + e^{0.22526}} = -0.1122$$

STEP 3 : - Compute error between output and hidden layer

$$\text{error}(\delta_{oh}) = (t - y) f'(y) = 1 - (-0.1122) \left[\frac{1}{2} (1+y)(1-y) \right] \\ = (1 + 0.1122) \left[\frac{1}{2} (1 + (-0.1122)) (1 - (-0.1122)) \right] \\ = (1.1122) (0.4937) \\ = 0.5491$$

Step 4 :- Change in weight between output and hidden layer

$$\Delta w_1 = \alpha \delta_{oh} z_1 = 0.25 \times 0.5491 \times (-0.1974) = -0.0271$$

$$\Delta w_2 = \alpha \delta_{oh} z_2 = 0.25 \times 0.5491 \times (0.537) = 0.0737$$

$$\Delta w_0 = \alpha \delta_{oh}^1 = 0.25 \times 0.5491 = 0.1373$$

STEP 5 :- Compute error between hidden layer and input layer

$$\begin{aligned}\text{error}(\delta_{ih}) &= \delta_{oh} \times w_1 \times f'(z_1) = 0.5491 \times 0.4 \times \frac{1}{2} [1-z_1][1+z_1] \\ &= 0.5491 \times 0.4 \times \frac{1}{2} [1-0.1974][1+0.1974] \\ &= 0.1056\end{aligned}$$

$$\begin{aligned}
 \text{error}(\delta_{ik}) &= \delta_{ik} \times w_2 \times f'(z_2) \\
 &= 0.5491 \times 0.1 \times \frac{1}{2} [(1-z_2)(1+z_2)] \\
 &= 0.5491 \times 0.1 \times \frac{1}{2} [(1-0.537)(1+0.537)] \\
 &= 0.0195
 \end{aligned}$$

Step 6 :- Change in weight between hidden layer
and Input layer

$$\begin{aligned}
 \Delta V_{01} &= \alpha \delta_{ik} z_1 x_1 = 0.25 \times 0.1056 \times 1 = 0.0264 \\
 \Delta V_{11} &= \alpha \delta_{ik} z_1 x_1 = 0.25 \times 0.1056 \times (-1) = -0.0264 \\
 \Delta V_{21} &= \alpha \delta_{ik} z_1 x_2 = 0.25 \times 0.1056 \times (1) = 0.0264 \\
 \Delta V_{02} &= \alpha \delta_{ik} z_2 x_1 = 0.25 \times 0.0195 \times 1 = 0.0049 \\
 \Delta V_{12} &= \alpha \delta_{ik} z_2 x_1 = 0.25 \times 0.0195 \times (-1) = -0.0049 \\
 \Delta V_{22} &= \alpha \delta_{ik} z_2 x_2 = 0.25 \times 0.0195 \times 1 = 0.0049
 \end{aligned}$$

Step 7 :- Computation of final weights

$$V_{01}(\text{new}) = V_{01}(\text{old}) + \Delta V_{01} = 0.3 + 0.0264 = 0.3264$$

$$V_{11}(\text{new}) = V_{11}(\text{old}) + \Delta V_{11} = 0.6 + (-0.0264) = 0.5736$$

$$V_{21}(\text{new}) = V_{21}(\text{old}) + \Delta V_{21} = -0.1 + 0.0264 = -0.0736$$

$$V_{02}(\text{new}) = V_{02}(\text{old}) + \Delta V_{02} = 0.5 + 0.0049 = 0.5049$$

$$V_{12}(\text{new}) = V_{12}(\text{old}) + \Delta V_{12} = -0.3 + (-0.0049) = -0.3049$$

$$V_{22}(\text{new}) = V_{22}(\text{old}) + \Delta V_{22} = 0.4 + 0.0049 = 0.4049$$

$$w_0(\text{new}) = w_0(\text{old}) + \Delta w_0 = -0.2 + 0.1373 = -0.0627$$

$$w_1(\text{new}) = w_1(\text{old}) + \Delta w_1 = 0.4 + (-0.027) = 0.3729$$

$$w_2(\text{new}) = w_2(\text{old}) + \Delta w_2 = 0.1 + 0.0737 = 0.1737$$

In order to have some numbers to work with, here are initial weights, biases, and training input and output.

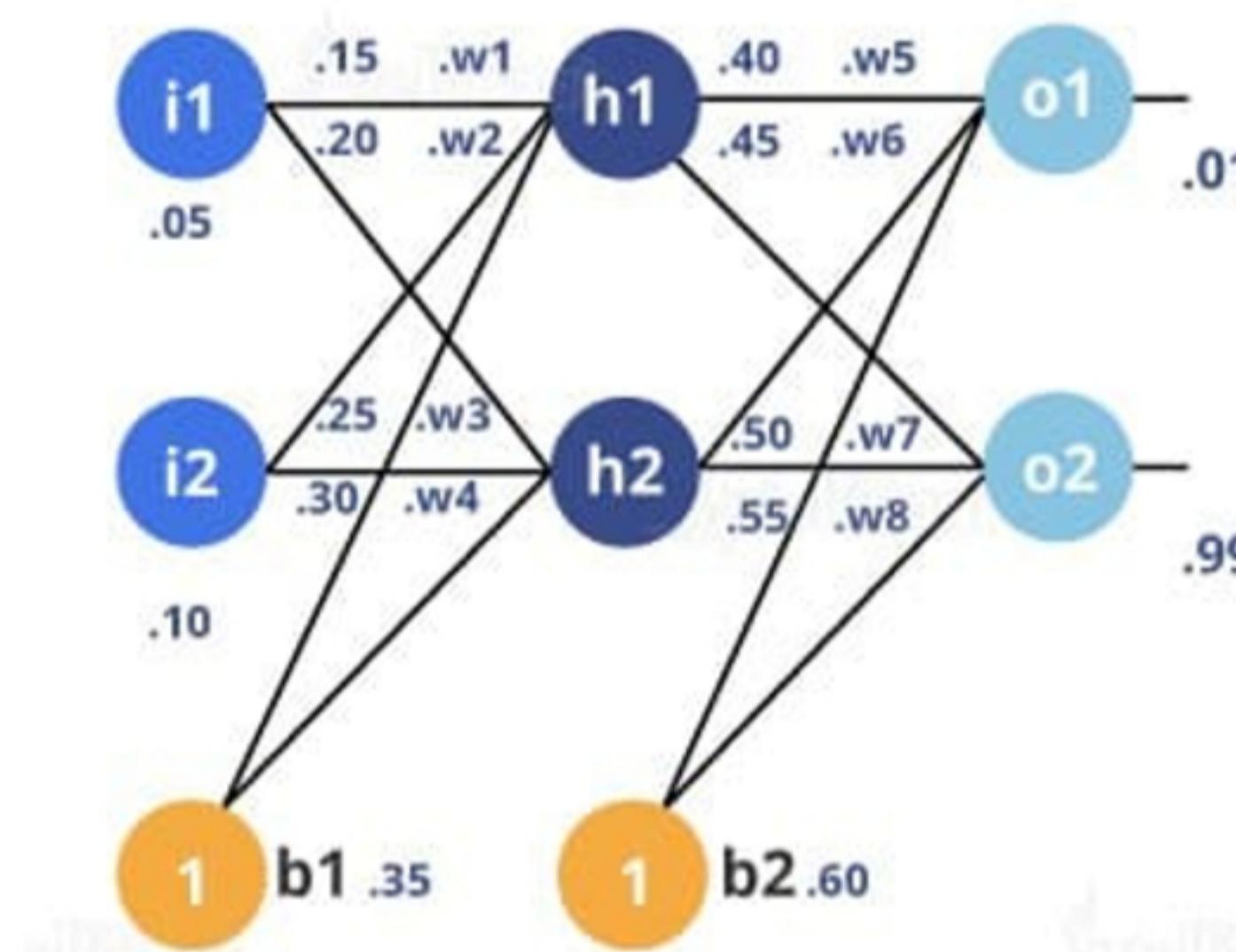
Inputs (i1): 0.05

Output (o1): 0.01

Inputs (i2): 0.10

Output (o2): 0.99

Step 1: The Forward Pass:



Calculate h1:

$$\text{net h1} = w_1 * i_1 + w_2 * i_2 + b_1 * 1 \dots \text{(Equation 1)}$$

$$\text{net h1} = 0.15 * 0.05 + 0.2 * 0.1 + 0.35 * 1 = 0.3775$$

The output for h1: The output for h1 is calculated by applying a sigmoid function to the net input of h1.

$$\text{out h1} = 1/(1 + e^{-\text{net h1}}) = 1/(1 + e^{-0.3775}) = 0.593269992 \dots \text{(Equation 2)}$$

Carrying out the same process for h2:

$$\text{out h2} = 0.596884378$$

The output for o1:

$$\text{net o1} = w_5 * \text{out h1} + w_6 * \text{out h2} + b_2 * 1 \dots \dots \dots \text{(Equation 3)}$$

$$\text{net o1} = 0.4 * 0.593269992 + 0.45 * 0.596884378 + 0.6 * 1 = 1.105905967$$

$$\text{out o1} = 1/(1 + e^{-\text{net o1}}) = 1/(1 + e^{-1.105905967}) = 0.75136507 \dots \dots \dots \text{(Equation 4)}$$

Carrying out the same process for o2:

$$\text{out o2} = 0.772928465$$

Calculating the Total error:

Total error: $E_{\text{total}} = \frac{1}{2}(\text{target} - \text{output})^2$

The target output for o1 is 0.01, but the neural network output is 0.75136507; therefore, its error is:

$$E_{o1} = \frac{1}{2}(\text{target } o1 - \text{out } o1)^2 = \frac{1}{2}(0.01 - 0.75136507)^2$$

$$E_{o1} = 0.27481108 \dots \text{ (Equation 5)}$$

By repeating this process for o2 (remembering that the target is 0.99), we get:

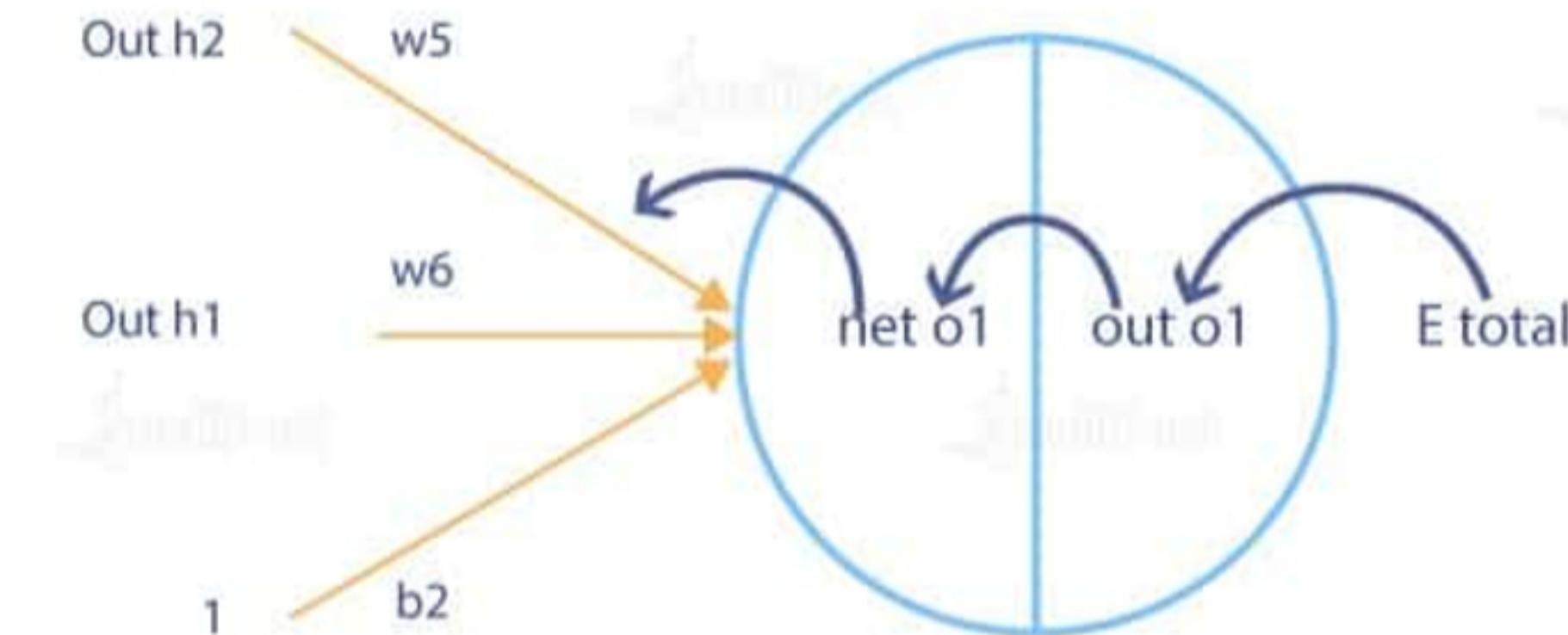
$$E_{o2} = 0.023560026$$

Then, the total error for the neural network is the sum of these errors:

$$E_{\text{total}} = E_{o1} + E_{o2} = 0.274811083 + 0.023560026 = 0.298371109$$

Step 2: Backward Propagation:

Our goal with back propagation algorithm is to update each weight in the network so that the actual output is closer to the target output, thereby minimizing the error for each output neuron and the network as a whole.



Consider w_5 ; we will calculate the rate of change of error w.r.t the change in the weight w_5 :

$$(\partial E_{\text{total}})/\partial w_5 = (\partial E_{\text{total}})/(\partial \text{out } o_1) * (\partial \text{out } o_1)/(\partial \text{net } o_1) * (\partial \text{net } o_1)/\partial w_5$$

Since we are propagating backward, the first thing we need to do is to calculate the change in total errors w.r.t the outputs o_1 and o_2 :

$$E_{\text{total}} = (1/2) * (\text{target } o_1 - \text{out } o_1)^2 + (1/2) * (\text{target } o_2 - \text{out } o_2)^2$$
$$(\partial E_{\text{total}})/(\partial \text{out } o_1) = -(\text{target } o_1 - \text{out } o_1) = -(0.01 - 0.75136507) = 0.74136507$$

Now, we will propagate further backward and calculate the change in the output o_1 w.r.t to its total net input:

$$\text{out } o_1 = 1/(1 + e^{-\text{net } o_1}) \quad (\text{from Equation 4})$$

$$(\partial \text{out } o_1)/(\partial \text{net } o_1) = \text{out } o_1(1 - \text{out } o_1) = 0.75136507(1 - 0.75136507) = 0.186815602$$

How much does the total net input of o_1 change w.r.t w_5 ?

$$\text{net o1} = w_5 * \text{out h1} + w_6 * \text{out h2} + b_2 * 1 \quad (\text{from Equation 3})$$

$$(\partial \text{net o1}) / \partial w_5 = 1 * \text{out h1} * w_5^{(1-1)} + 0 + 0 = \text{out h1} = 0.593269992$$

Putting all values together and calculating the updated weight value:

$$(\partial E \text{ total}) / \partial w_5 = (\partial E \text{ total}) / (\partial \text{out o1}) * (\partial \text{out o1}) / (\partial \text{net o1}) * (\partial \text{net o1}) / \partial w_5 = 0.082167041$$

Let's calculate the updated value of w_5 .

$$w_5^+ = w_5 - n * (\partial E \text{ total}) / \partial w_5 = 0.4 - 0.5 * 0.082167041 = 0.35891648 \text{ (This is gradient descent)}$$

We can repeat this process to get the new weights w_6 , w_7 , and w_8 .

$$w_6^+ = 0.408666186$$

$$w_7^+ = 0.511301270$$

$$w_8^+ = 0.561370121$$

We perform the actual updates in the neural network after the new weights lead into the hidden layer neurons.

We'll continue the backward pass by calculating new values for w1, w2, w3, and w4:

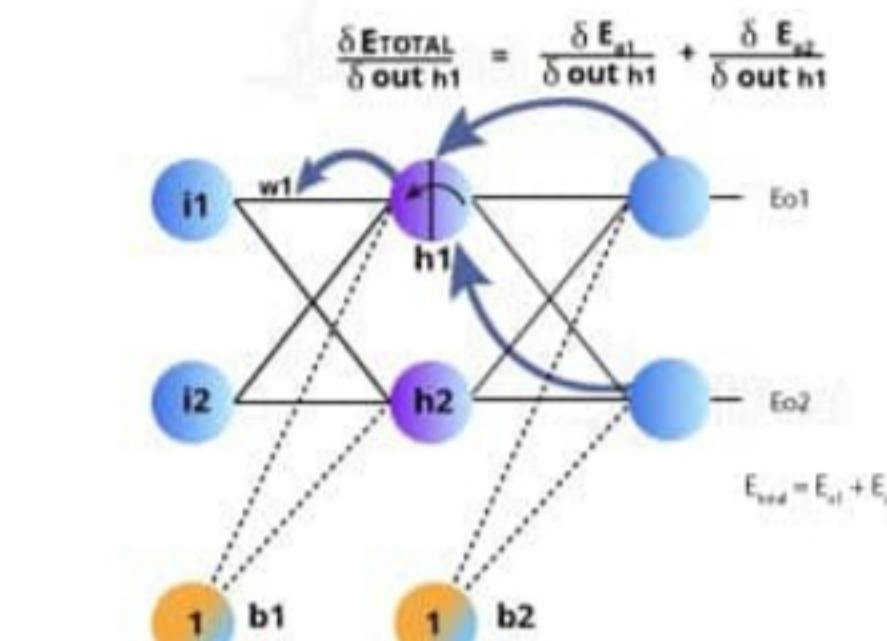
Starting with w1:

$$(\partial E_{\text{total}})/\partial w_1 = (\partial E_{\text{total}})/(\partial \text{out h1}) * (\partial \text{out h1})/(\partial \text{net h1}) * (\partial \text{net h1})/\partial w_1$$

$$(\partial E_{\text{total}})/(\partial \text{out h1}) = (\partial E_{o1})/(\partial \text{out h1}) + (\partial E_{o2})/(\partial \text{out h1})$$

We're going to use a similar process as we did for the output layer, but slightly different to account for the fact that the output of each hidden layer neuron contributes to the final output. Thus, we need to take E_{o1} and E_{o2} into consideration.

$$\frac{\delta E_{TOTAL}}{\delta w1} = \frac{\delta E_{TOTAL}}{\delta out h1} + \frac{\delta out h1}{\delta net h1} * \frac{net h1}{w1}$$



Starting with h1:

$$(\partial E_{total})/(\partial out h1) = (\partial E_{o1})/(\partial out h1) + (\partial E_{o2})/(\partial out h1)$$

$$(\partial E_{o1})/(\partial out h1) = (\partial E_{o1})/(\partial net o1) * (\partial net o1)/(\partial out h1)$$

We can calculate $(\partial E_{o1})/(\partial net o1)$ using the values calculated earlier.

$$(\partial E_{o1})/(\partial net o1) = (\partial E_{o1})/(\partial out h1) * (\partial out h1)/(\partial net o1)$$

$$= 0.74136507 * 0.186815602 = 0.138498562$$

$$net o1 = w_5 * out h1 + w_6 * out h2 + b_2 * 1$$

$$(\partial net o1)/(\partial out h1) = w_5 = 0.40$$

Let's put the values in the equation.

$$(\partial E_{o1})/(\partial out h1) = (\partial E_{o1})/(\partial net o1) * (\partial net o1)/(\partial out h1)$$

$$= 0.138498562 * 0.40 = 0.055399425$$

Following the same process for $(\partial E_{o2})/(\partial out h1)$, we get:

$$(\partial E_{o2})/(\partial out h1) = -0.019049119$$

$$\begin{aligned}
 (\partial E_{\text{total}})/(\partial \text{out } h_1) &= (\partial E_{o1})/(\partial \text{out } h_1) + (\partial E_{o2})/(\partial \text{out } h_1) \\
 &= 0.055399425 + (-0.019049119) = 0.036350306
 \end{aligned}$$

Now that we have $(\partial E_{\text{total}})/(\partial \text{out } h_1)$, we need to figure out $(\partial \text{out } h_1)/(\partial \text{net } h_1)$ and $(\partial \text{net } h_1)/\partial w$ for each weight

$$\text{out } h_1 = 1/(1 + e^{-\text{net } h_1})$$

$$(\partial \text{out } h_1)/(\partial \text{net } h_1) = \text{out } h_1(1 - \text{out } h_1) = 0.59326999(1 - 0.59326999) = 0.241300709$$

We will calculate the partial derivative of the total net input of h_1 w.r.t w_1 the same way as we did for the output neuron.

$$\text{net } h_1 = w_1 * i_1 + w_3 * i_2 + b_1 * 1$$

$$(\partial \text{net } h_1)/\partial w_1 = i_1 = 0.05$$

Let's put it all together.

$$(\partial E_{\text{total}})/\partial w_1 = (\partial E_{\text{total}})/(\partial \text{out } h_1) * (\partial \text{out } h_1)/(\partial \text{net } h_1) * (\partial \text{net } h_1)/\partial w_1$$

$$(\partial E_{\text{total}})/\partial w_1 = 0.036350306 * 0.241300709 * 0.05 = 0.000438568$$

We can now update w_1 .

$$w_1^+ = w_1 - n * (\partial E_{\text{total}})/\partial w_1 = 0.15 - 0.5 * 0.000438568 = 0.149780716$$

Let's update other weights similarly.

$$w_2^+ = 0.19956143$$

$$w_3^+ = 0.24975114$$

$$w_4^+ = 0.29950229$$

- *When we fed forward 0.05 and 0.1 inputs initially, the error on the network was 0.298371109.*
- *After the first round of backpropagation, the total error is now down to 0.291027924.*

It might not seem like much, but after repeating this process 10,000 times, the error plummets to 0.0000351085. When we feedforward 0.05 and 0.1, the two output neurons will generate 0.015912196 (vs. 0.01 target) and 0.984065734 (vs. 0.99 target).

13.1

History of Evolutionary Computing

The origin of the theory of evolution is the book *On the Origin of Species* written by Charles Darwin. The central concept lies in the process called “Survival of the fittest” used by the nature in its selection following the principle that all living creatures are descendants of older species and any variation occurs due to natural selection. The central concept mentioned above states that some individuals have a greater chance of reproduction due to some heritable differences and are said to have higher “fitness.” Fitness measures the success of an organism.

There are four requirements for evolutions to take place. These are fitness, variation, reproduction, and heredity. We define these concepts here.

1. **Fitness:** It measures the ability of an individual to survive and reproduce. The difference between the values of one organism to another is the higher number of offspring of one with the higher fitness value.
2. **Variation:** It is essential for evolution to take place. This is the series of changes occurring in species. Variations are of two types: inherited variation and environmental variation.
 - (a) **Inherited variation:** It is a genetically inherited character. Offsprings inherit half of their characters from each of their parents. Examples may be color of hair.
 - (b) **Environmental variation:** Climate and culture are the two factors of environmental variation, which affect organisms.
3. **Reproduction:** This is the process in which offsprings are generated from their parents. It is of two types: asexual and sexual.
 - (a) **Asexual reproduction:** It involves only one organism.
 - (b) **Sexual reproduction:** It requires two different organisms for its occurrence and is the most common method of reproduction.
4. **Heredity:** This is a process in which offsprings get some characteristics of parents. It helps in the variation in the population and hence may cause the development of new species.

13.1.1 Genetic Algorithms: History

During the late 1950s and early 1960s, some evolutionary biologists tried to develop algorithms to model aspects of natural evolution through programming in computers, which are now called genetic algorithms. Although it had evolutionary computing inherent, the biologists did not realize at that time that their strategy would be applicable to artificial problems. Evolution-inspired algorithms were independently developed by different researchers such as G. E. P. Box, G. J. Friedman, W. W. Bledsoe, and H. J. Bremermann. These algorithms were applied by them for function optimization and machine learning. This was around 1962 and was not pursued further immediately. The next step in the development was by I. Rechenberg in 1965, when he introduced a technique called *evolution strategy*. It had very little similarity to genetic algorithms. Most of the concepts used in genetic algorithms such as population, crossover, and mutation of parents to produce offspring were not there.

The technique of evolutionary programming was introduced by three scientists, L. J. Fogel, A. J. Owens, and M. J. Walsh in 1966, which is the next important step in this direction. The solutions were represented by finite state machines in their approach and as was done in the evolution strategy of Rechenberg, the process was to randomly mutate one of these simulated machines and keeping the better of the two. Both the areas of evolution strategy and the evolutionary programming technique are still being pursued for research even today. The most important aspect which was missed in both these approaches was recognition of the importance of crossover.

The three most important aspects of GA are as follows:

1. Definition of objective function.
2. Definition and implementation of genetic representation.
3. Definition and implementation of genetic operators.

Reproduction

Reproduction is usually the first operator applied on population. Chromosomes are selected from the population to be parents to crossover and produce offspring. According to Darwin's evolution theory of survival of the fittest, the best ones should survive and create new offspring. That is why reproduction operator is sometimes known as the selection operator. There exist a number of reproduction operators in GA literature but the essential idea in all of them is that the above average strings are picked from the current population and their multiple copies are inserted in the mating pool in a probabilistic manner. The various methods of selecting chromosomes for parents to crossover are as follows:

1. Roulette-wheel selection
2. Boltzmann selection
3. Tournament selection
4. Rank selection
5. Steady-state selection

Roulette-wheel selection: The commonly used reproduction operator is the proportionate reproductive operator where a string is selected from the mating pool with a probability proportional to the fitness. Thus, i th string in the population is selected with a probability proportional to F_i , where F_i is the fitness value for that string. Since the population size is usually kept fixed in a simple GA, the sum of the probabilities of each string being selected for the mating pool must be one. The probability of the i th selected string is

$$p_i = \frac{F_i}{\sum_{j=1}^n F_j}$$

where n is the population size.

One way to implement this selection scheme is to imagine a Roulette-wheel with its circumference for each string marked proportionate to string's fitness. The fitness of the population is calculated as Roulette-wheel is spun " n " times (in this example eight times), each time selecting an instance of the string chosen by the Roulette-wheel pointer. Since the circumference of the wheel is marked according to a string's fitness, the Roulette-wheel mechanism is expected to make F_i/F copies of the i th string of the mating pool.

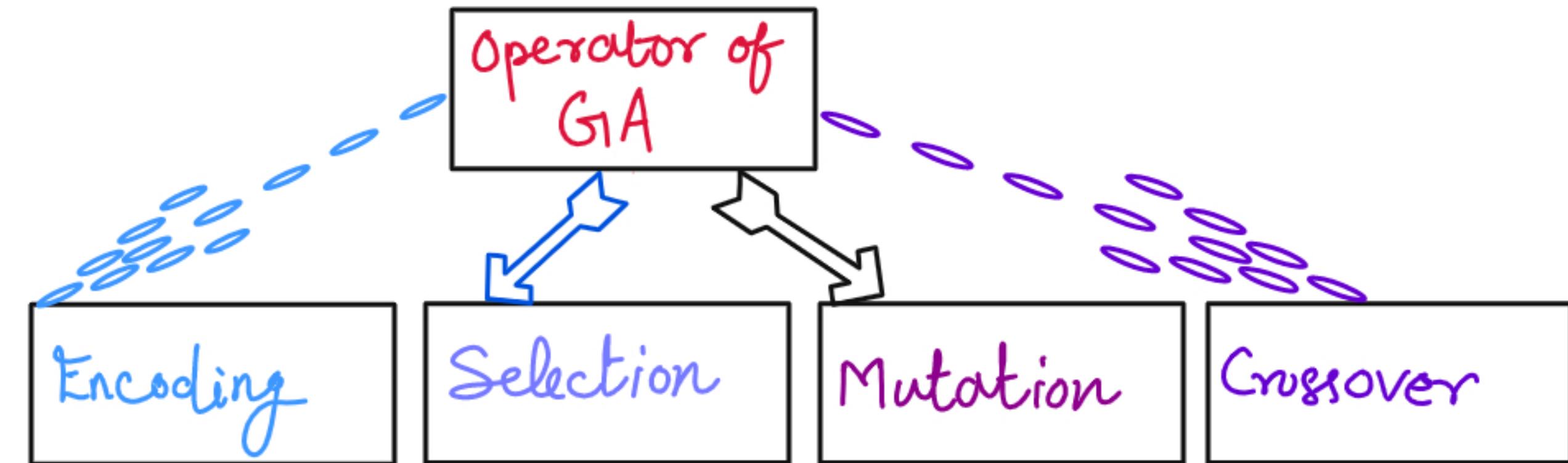
The average fitness is given by

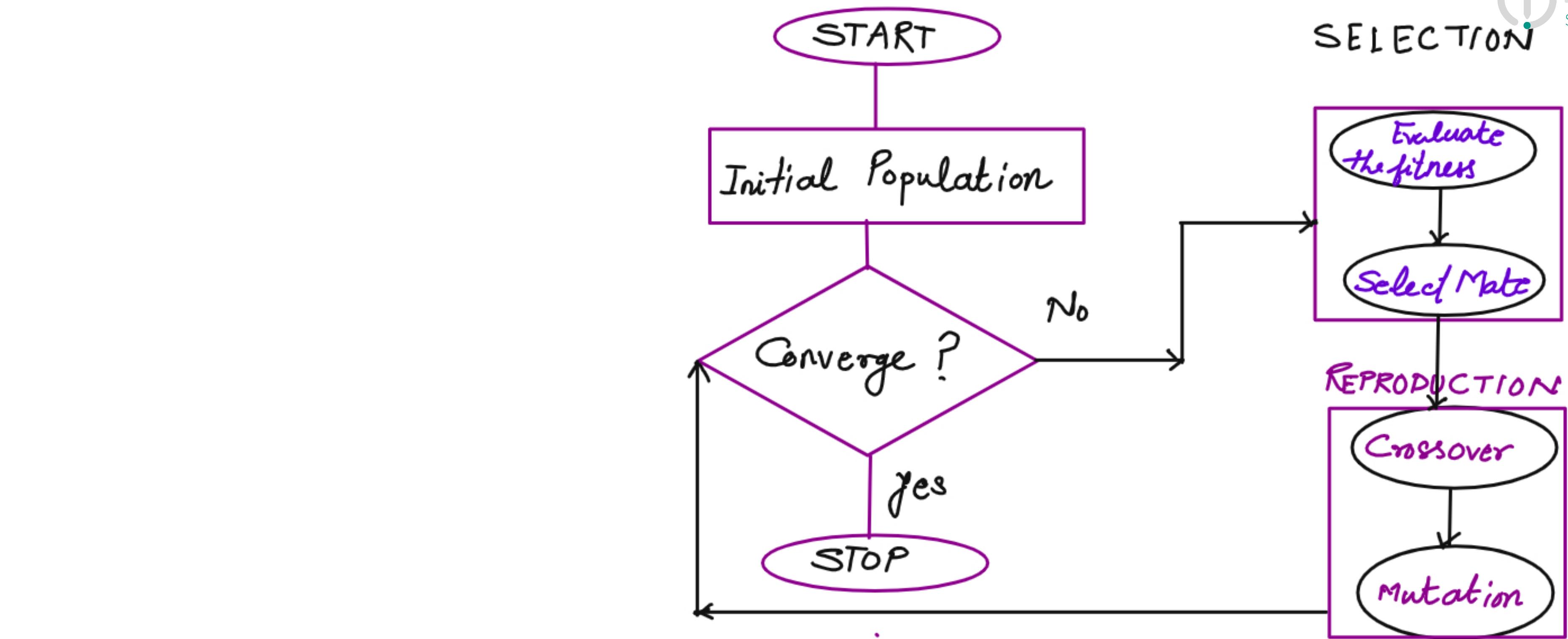
$$\bar{F} = \frac{1}{n} \left(\sum_{j=1}^n F_j \right)$$

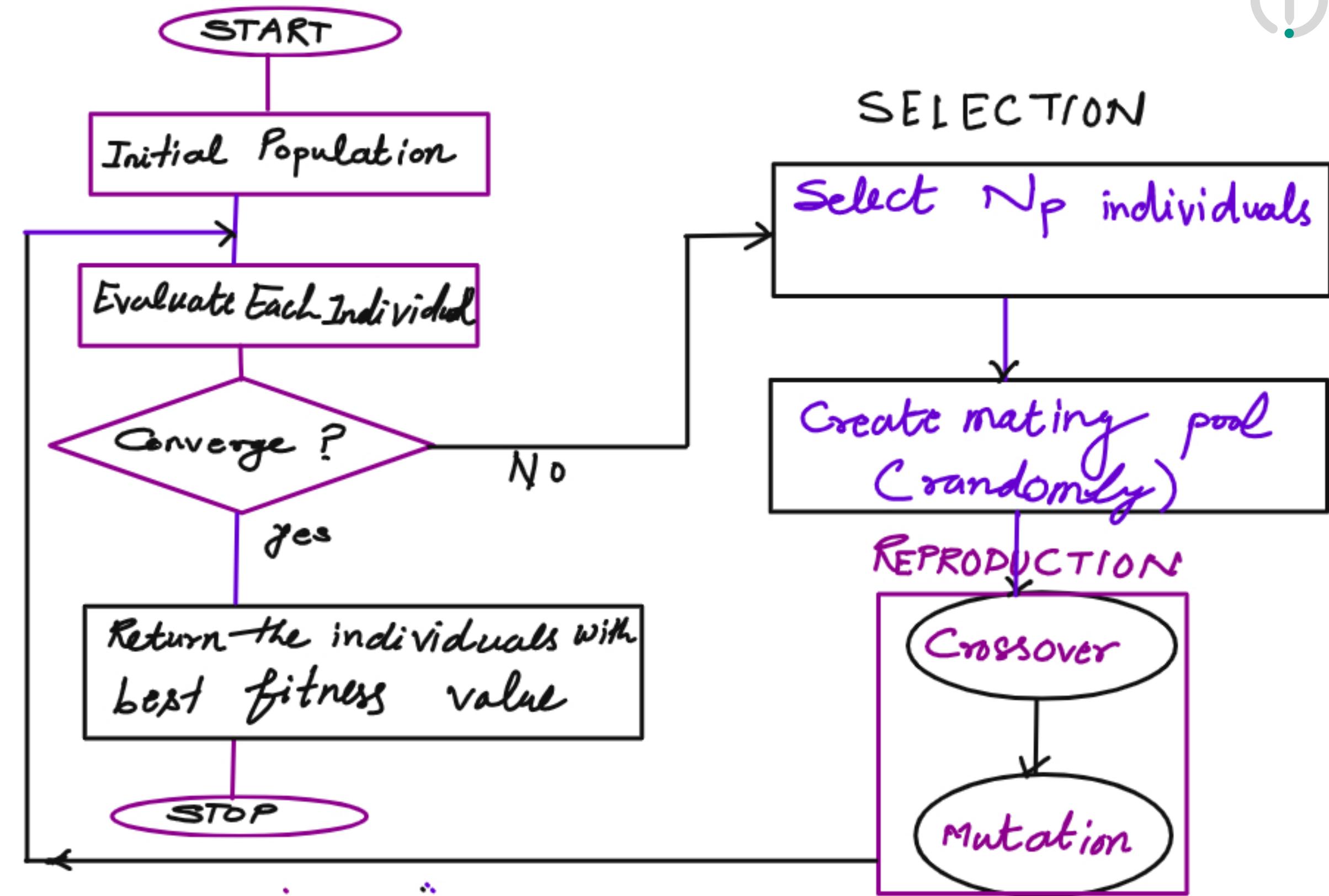
Genetic Algorithm



To generate high quality solution for optimization problem
Population and Individual







Genetic Algorithm

- I Genetic & Natural Selection to solve optimization problem
- II More Advanced
- III Use in field such as AI, ML
- IV Probabilistic Rules
- V Search on a population of points.

Traditional Algorithm

- I Step by step procedure to solve a given problem
- II Not as Advanced
- III Used in Programming, mathematics
- IV Fully deterministic rule
- V Search on single point.

Genetic Algorithm: Provide efficient, effective technique for optimization and machine learning algorithm.

Widely used today in business, scientific & engineering circles.

Genetic Operators: Manipulate chromosome / Solution

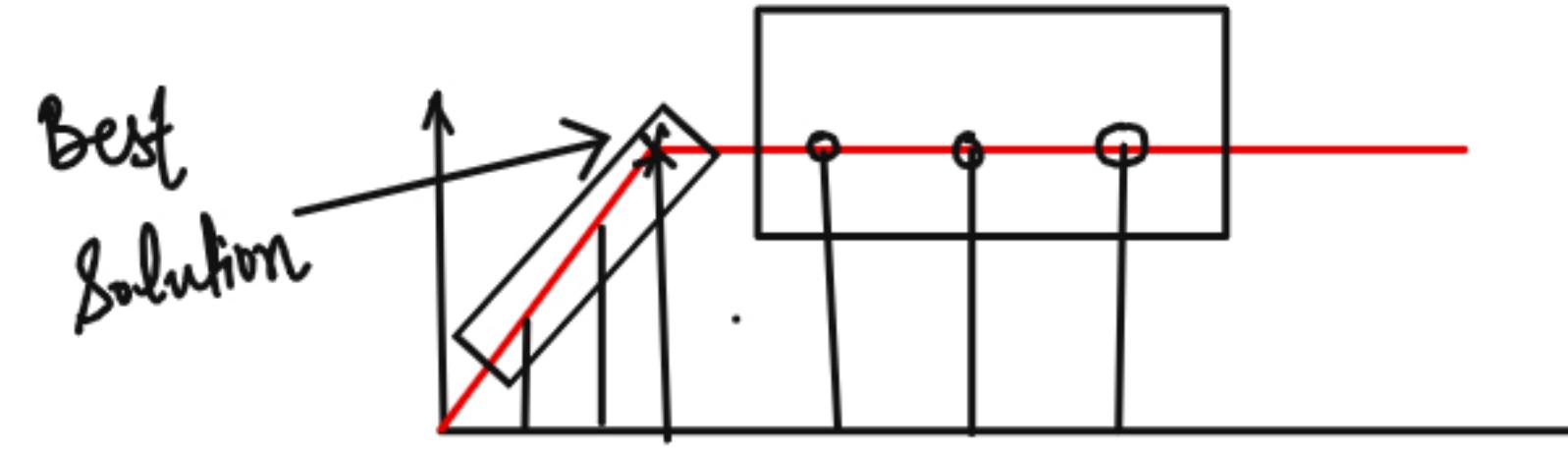
- Selection Operator
- Mutation
- Crossover

Selection Operator :- Process of selecting two or more parents from the population for crossing.

- Purpose of selection is to emphasize filter individuals in the population in hopes that the offspring have higher fitness.
- Methods: Roulette Wheel Selection, Boltzman Selection, tournament selection, rank Selection
Roulette Wheel Selection depending on % Contribution to the total population fitness String is selected for mating to form the next generation

Convergence Test / Termination Condition

1. Manual checking
2. Solution found that satisfies objective criteria
3. Fixed number of iteration
4. Budget limit reached
- 5.



Encoding

Binary Encoding :-

$$\textcircled{A} = 5 \quad (100)$$

$$\textcircled{B} = 10 \quad (110)$$

$$\textcircled{C} = 15 \quad (101)$$

$$M = 25$$



eg:- $f(x) = x^3; 0 \leq x \leq 255$

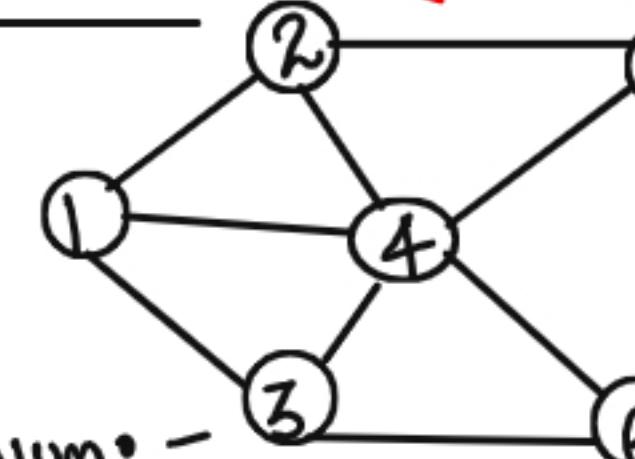
8 bit digit :- $100\ 101\ 011 \rightarrow 153$

$$010\ 101\ 101 \rightarrow 255$$

Fitness Evaluation :- Fitness function / Objective Function

Traveling Salesman (Minimize Sol)

Problem :-



Minimum:-

$$\text{Path 1 } 1 \ 2 \ 5 \ 6 \ 4 \ 3 \ 1 = 18$$

$$\text{Path 2 } 1 \ 2 \ 5 \ 4 \ 6 \ 3 \ 1 = 20$$

$$\boxed{\text{Path 3 } 1 \ 2 \ 4 \ 5 \ 6 \ 3 \ 1 = 12 \checkmark}$$

$$\text{Path 4 } 1 \ 2 \ 5 \ 6 \ 3 \ 4 \ 1 = 17$$

(Maximum Sol. Value)

A 5 (100)

B 10 (150)

C 15 (200)

M = 25
Condition

k_1	P	
A	100	5 100
AB	110	15 250
AC	101	20 300
BC	011	25 350
ABC	111	30 450 X

$\rightarrow M=25$

Crossover (Binary Coded GA)

Single Point:

P ₁	01 1 0 1 0
P ₂	11 0 1 0 0
O ₁	01 0 1 0 0
O ₂	11 1 0 1 0

Change bits after line

Two Points:

P ₁	01 1 0 1 0
P ₂	11 0 1 0 0
O ₁	01 1 0 0 1
O ₂	11 0 0 1 0

Change bits in between the line.

Multi Point Crossover:-

P ₁	0 1 1 0 1 0
P ₂	1 1 0 1 0 0

Change Alternate bits

O ₁	1 1 0 1 1 0
O ₂	0 1 1 0 0 0

Uniform Crossover :-

P_1 0 0 1 0 1 1

P_2 1 0 0 1 0 0

Tossing 0 0) 0 0 1 0

O_1 1 0 1 1 0 1

O_2 0 0 0 0 1 0

Crossover Mask \rightarrow

0	1	0	1	1	0
---	---	---	---	---	---

O_1 0 0 1 1 1 1

1 $\rightarrow P_1$

0 $\rightarrow P_2$

1 $\rightarrow P_2$

0 $\rightarrow P_1$

do yourself $\rightarrow O_2$

Mutation :-

offspring

0 1 1 0 0 1

MP (up)

0 1 0 0 0 1

M10

0 0 1 0 0 0

Random Selection of bits
offspring

M10

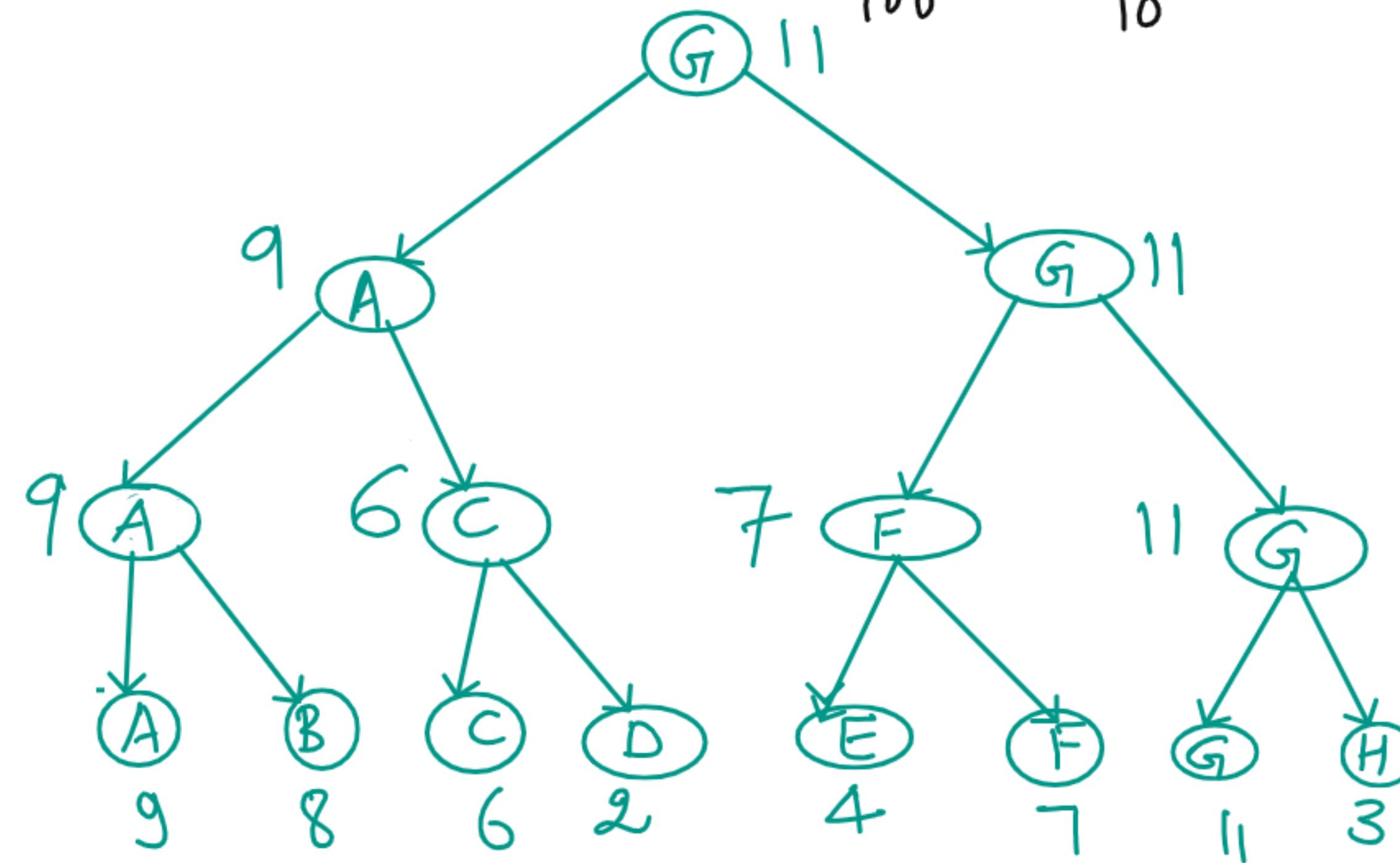
0 1 * 1 0 0 1
0 0 1 0 1 1

do yourself 0 1 * 0 0 1
Mp (up) 0 0 1 0 1 0

Random Selection

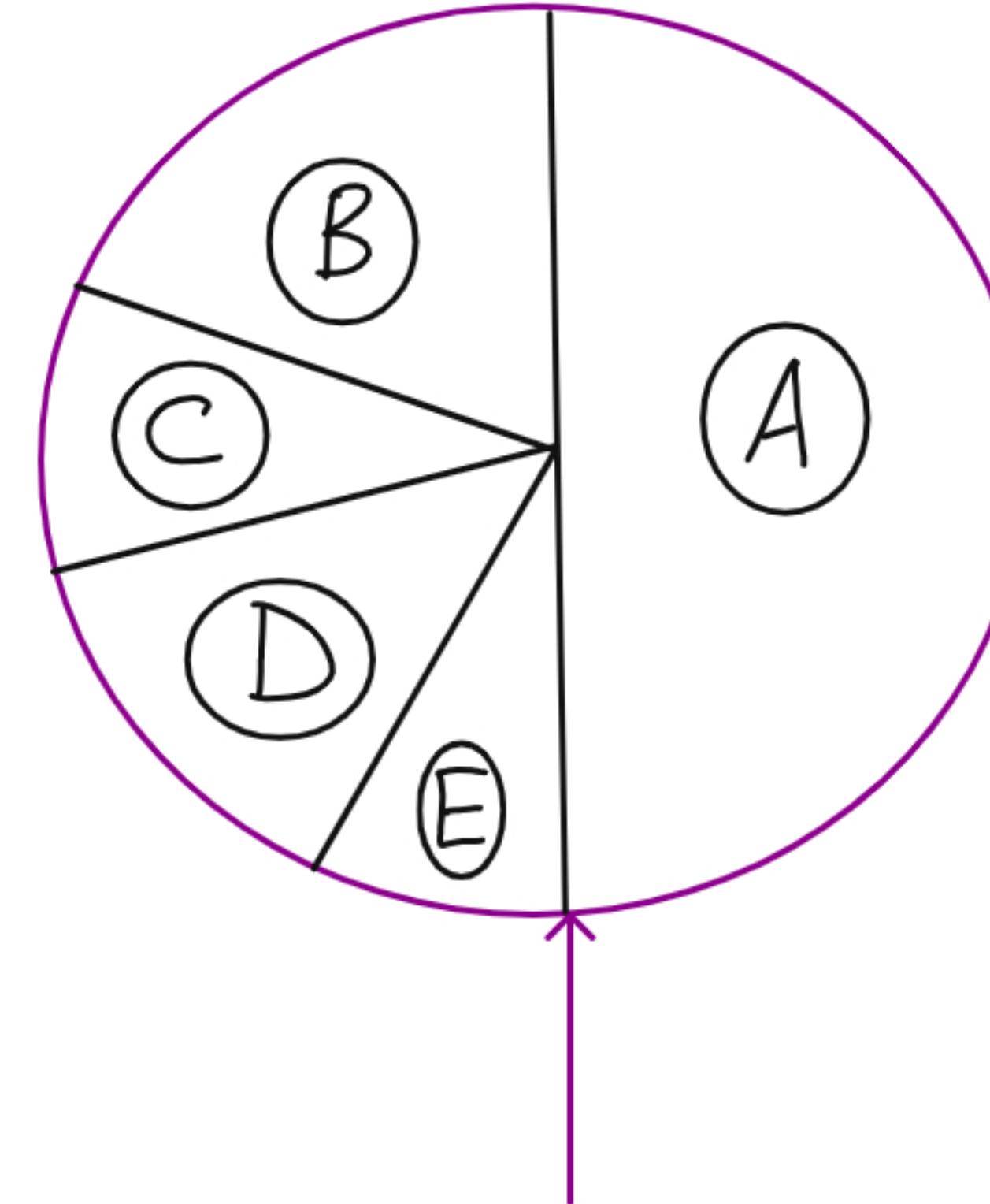
→ } do it

Tournament Selection :- $N \ll N_p$
 100 10



Roulette Wheel

C	F	
A	5	50%
B	2	20%
C	0.5	5 %
D	1.5	15 %
E	1	10 %



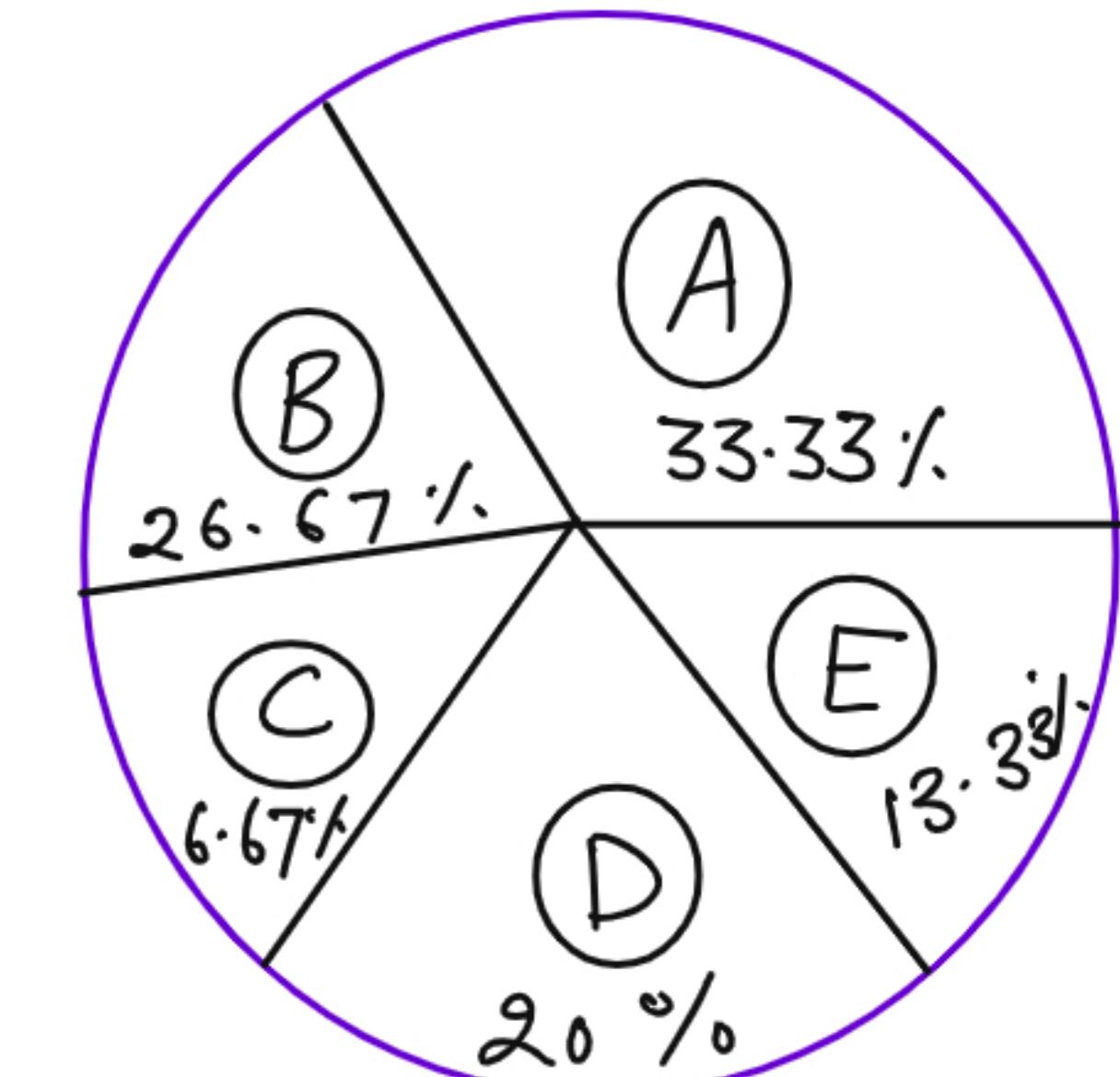
Rank based Selection

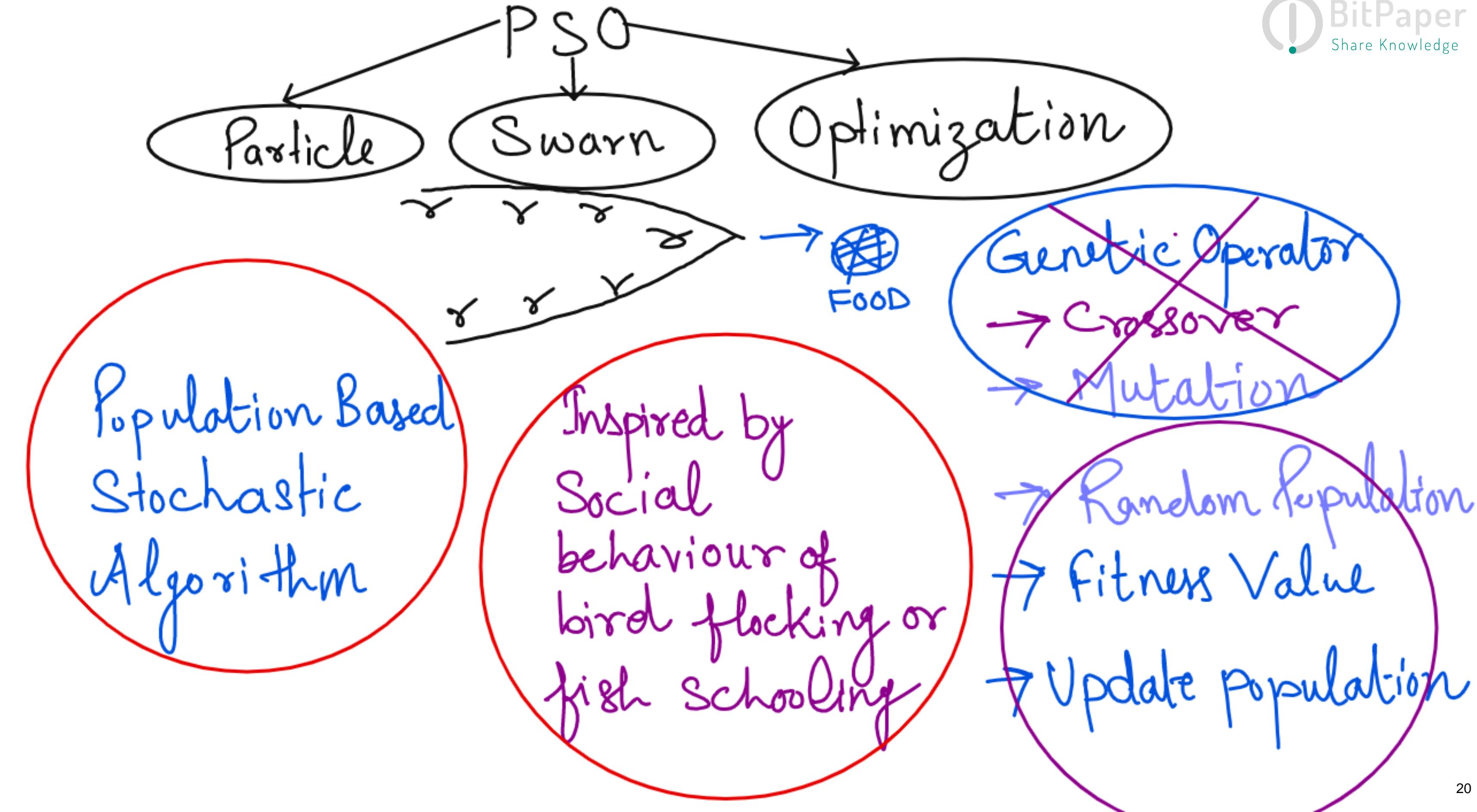
C	F
A	5
B	2
C	0.5
D	1.5
E	1

$$\frac{\text{Obt}}{\text{Total}} \times 100$$

5 = 33.33%
 4 = 26.67%
 1 = 6.67%
 3 = 20%
 2 = 13.33%

+ 15





For each particle Initialize particle END

Do FOR EACH PARTICLE Calculate fitness value If the fitness is better than the pbest in history Set Current value as the new pBest

END

Choose the particle with the best fitness value of all particles as the gbest

For each particle Calculate particle velocity ①

Update particle position ②

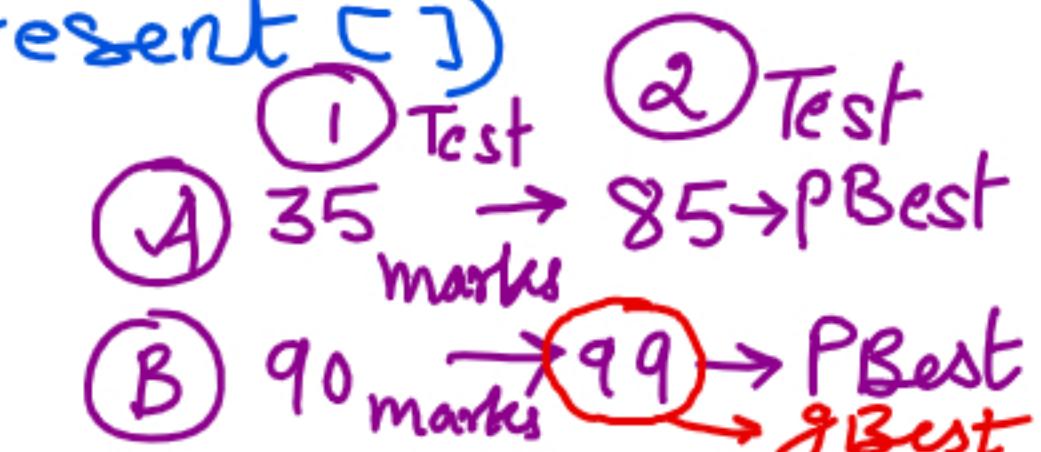
END

While maximum iteration or minimum error criteria is not attained.

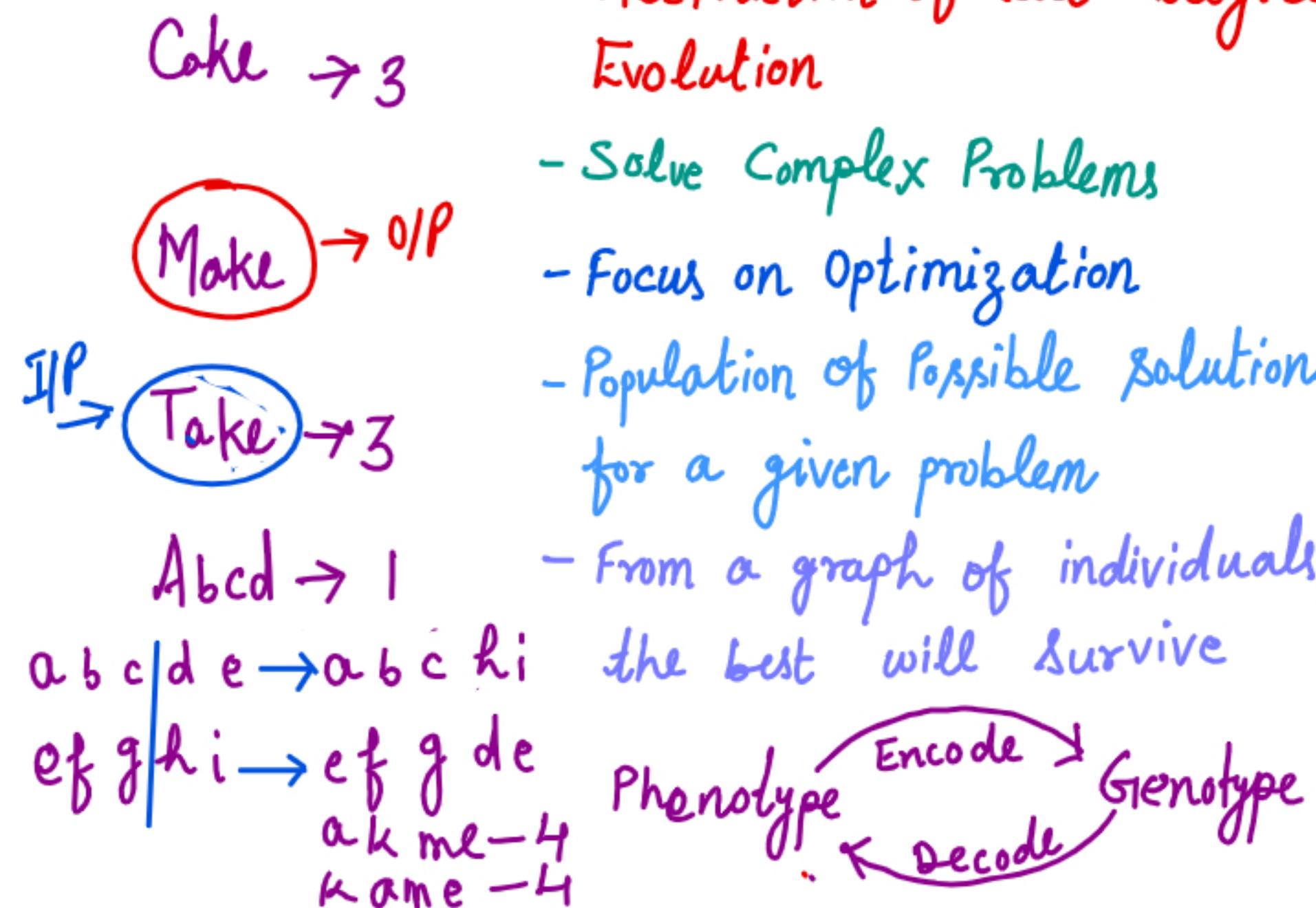
$$1) v[] = v[] + c_1 * \text{rand}() * (p\text{Best}[] - \text{present}[])$$

$$+ c_2 * \text{rand}() * (g\text{Best}[] - \text{present}[])$$

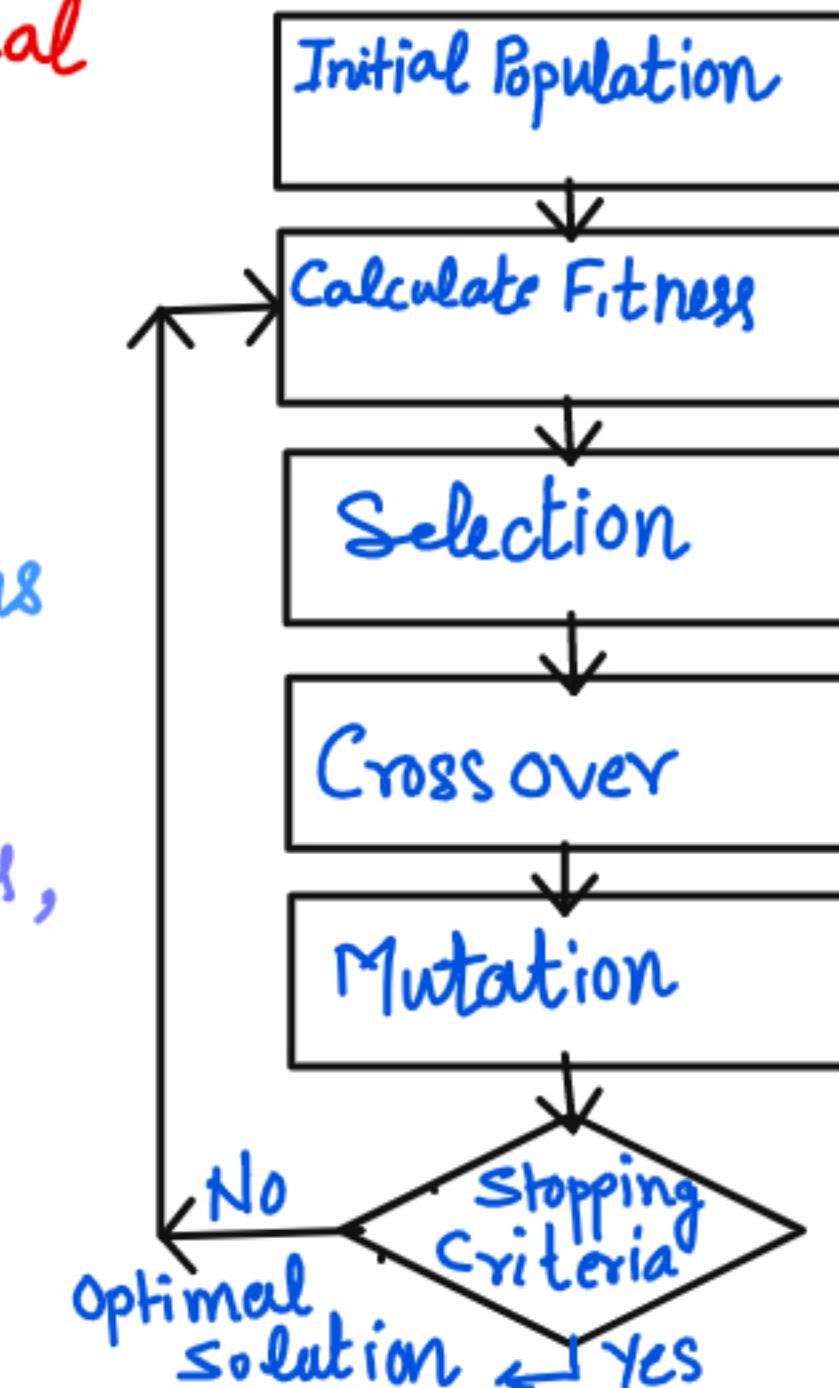
$$2) \text{present}[] = \text{present}[] + v[]$$



Genetic Algorithm (John Holland)



- Abstraction of Real Biological Evolution
- Solve Complex Problems
- Focus on Optimization
- Population of Possible Solutions for a given problem
- From a graph of individuals, the best will survive



Eg: Maximize the function $f(x) = x^2$ with x in interval $[0, 31]$ ie. $x = 0, 1, \dots, 30, 31$

Step 1: 1 01101₍₁₃₎, 11000₍₂₄₎
01000₍₈₎, 10011₍₁₉₎

Step 2: Decode into integer

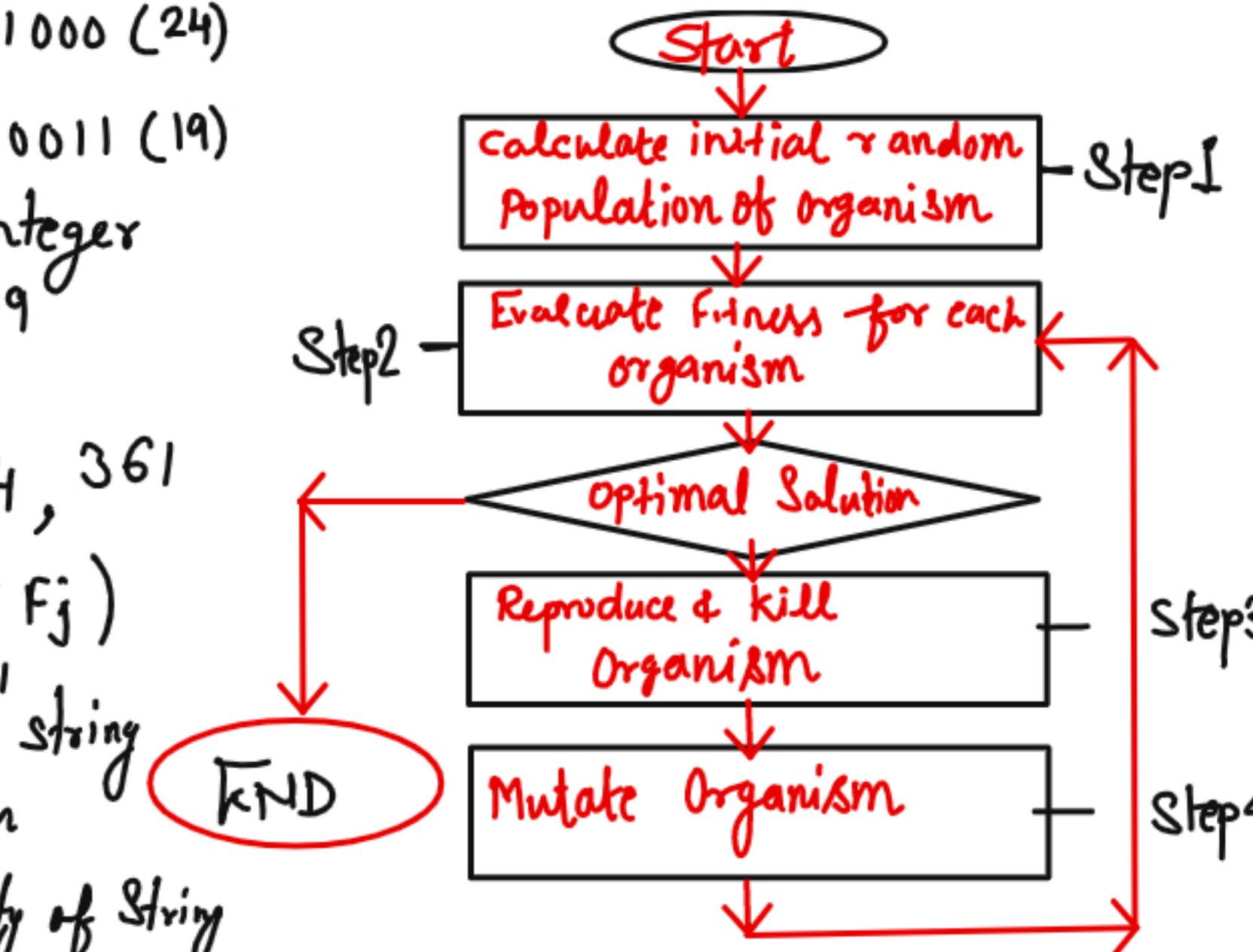
$$f(x) = x^2$$

$$169, 576, 64, 361$$

$$\text{Step 3: } P_i = f_i / \left(\sum_{j=1}^n f_j \right)$$

f_i = fitness for string
 i = Population

P_i = Probability of String



n = number of individuals in the population

$n_p i$ = is expected Concen.

String No	Initial Population	X Value	Fitness F_j $F(x) = x^2$	P_i	Expected Concen. $= N. Prob.$
1	01101	13	169	0.14	0.56
2	11000	24	576	0.49	1.97
3	01000	8	64	0.06	0.22
4	10011	19	361	0.31	1.23
Sum			1170	1.00	4.00
Average			293	0.25	1.00
Max			576	0.49	1.97

Crossover Operator can be of one point or two point scheme.
In one point crossover, selected pair of strings is cut at some random position and then segments are swapped to form new pair of strings.

In 2 point there will be 2 break points

eg:-

1001	11101
1011	01011

 off Spring

1001	01011
1011	11101

2 point :-

100	11	101	off Spring	100	01	101
101	01	011		101	11	011

String No.	Meeting Pool	Crossover Point	Offspring after crossover	X Value	fitness fun. $f(x) = x^2$
1	0110 1	4	0110 0	12	144
2	1100 0	4	1100 1	25	625
3	11 000	2	11 011	27	729
4	0011	2	00000	16	256
Σ_{nm}					1754
Avg.					439
					729

Mutation applied to each child individually after crossover. Bits are changed from 0 to 1 or from 1 to 0 at randomly chosen position of randomly selected string.

String No.	Offspring after Xover	Offspring after Mutat.	x value	fitness $f(x) = x^2$
1	01100	11100	28	784
2	11001	11001	25	625
3	11011	11011	21	729
4	10000	10100	20	400
Sum				2538
Avg				634.5
Max				784