Unit-5

Machine learning:

Machine learning is a sub field of Artificial Intelligence that provides computer the ability to learn and improve its learning from experience without being written rules explicitly. The machine start learning with observations or data, in order to look for patterns in data and make better decisions in the future based on the examples that it has been provided.

Applications:

→ Computer Vision Processing
→ Natural Language Processing
→ Forecasting (e.g; stock market trends)
→ Games
→ Expert Systems.

The difference between traditional programming and machine learning can be illustrated with following figures:

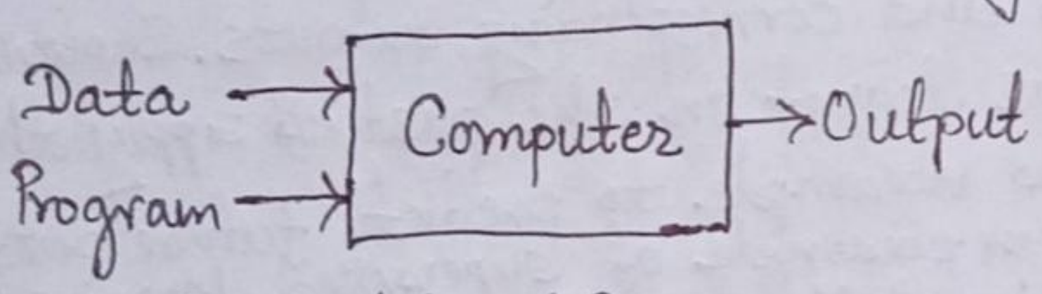


fig: Traditional Programming.

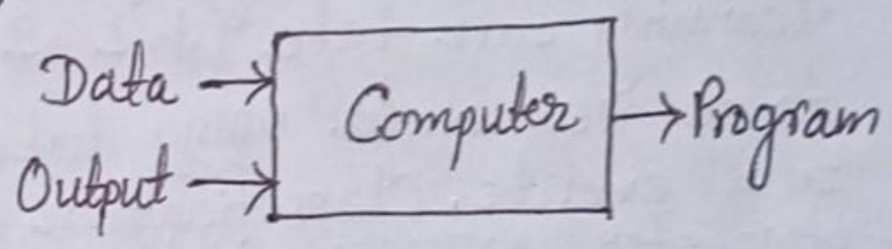
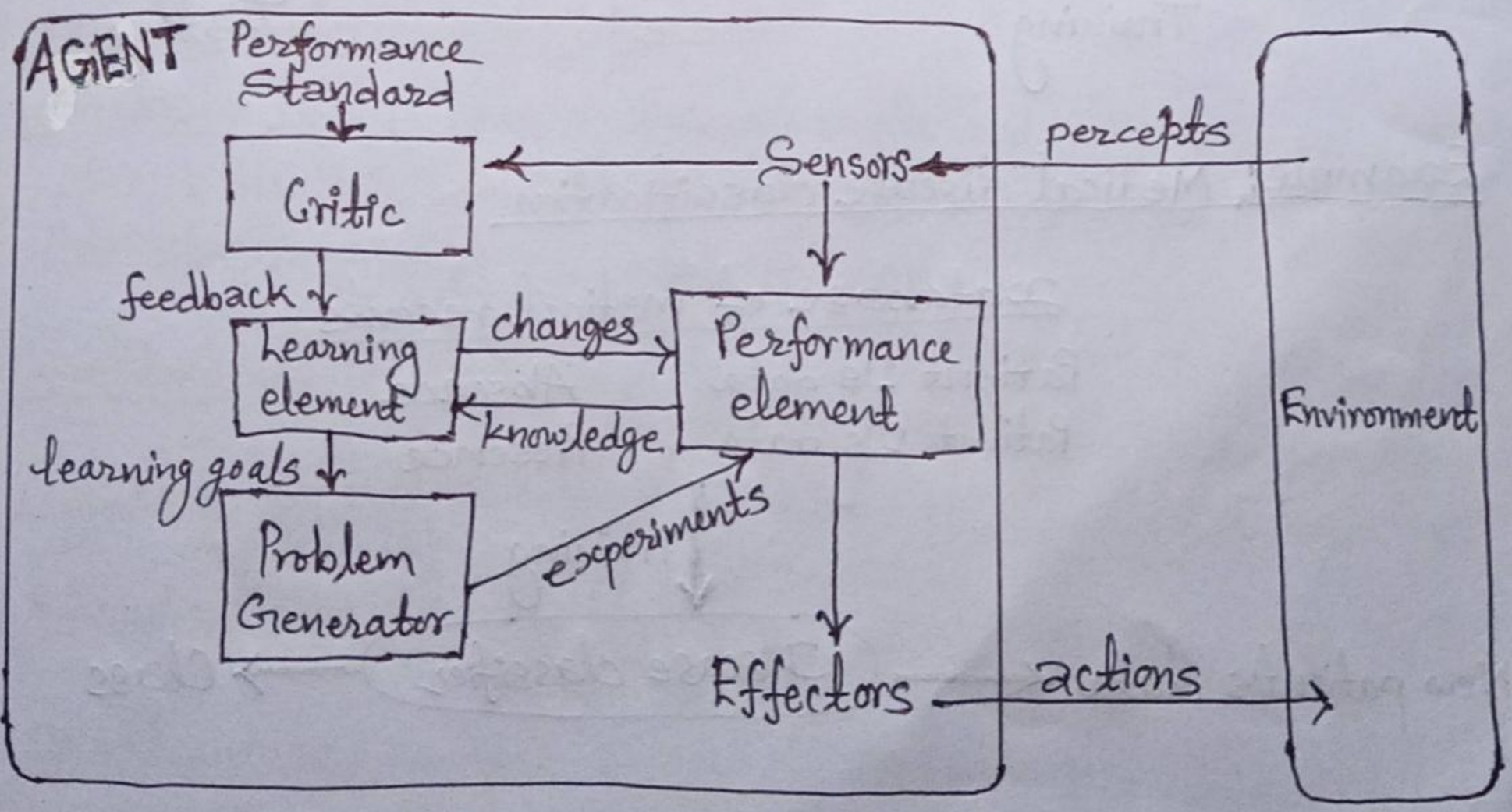


fig: Machine learning

⊛. Machine learning architecture or framework (Concept of Learning):



Learning agent consists of following components:

Scanned by TapScanner

i) Learning element → It receives and processes the input obtained from environment.

ii) Knowledge base → It is a collection of knowledge. It updates knowledge periodically if agent is able to acquire new knowledge.

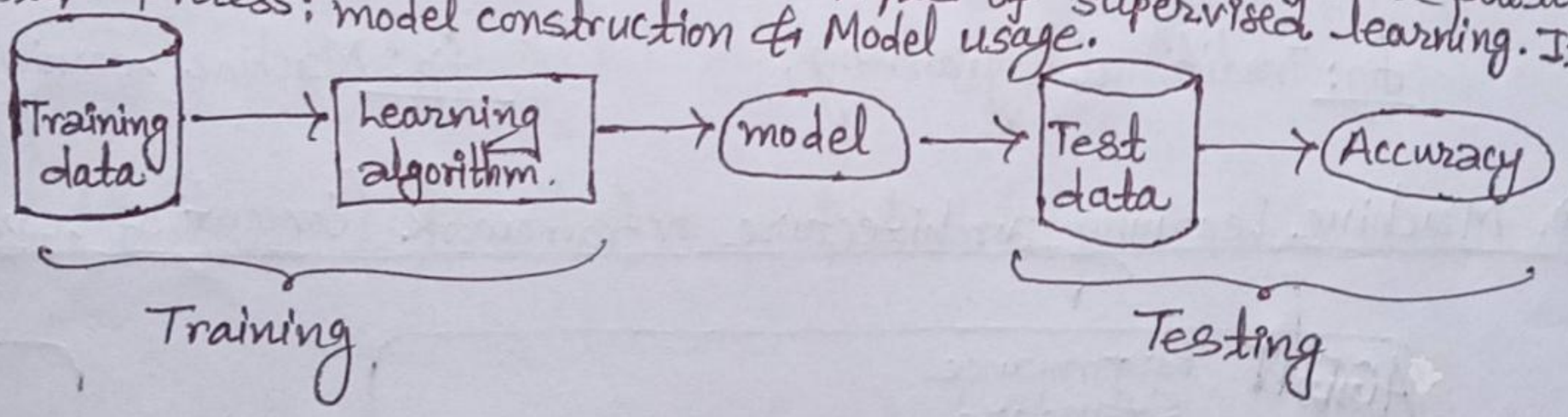
iii) Performance element → It uses the updated knowledge base to perform some tasks or solves some problems and produces the corresponding output.

iv) Feedback element → It receives two inputs, one from learning element and one from standard system. It is used to determine what should be done to produce the correct output.

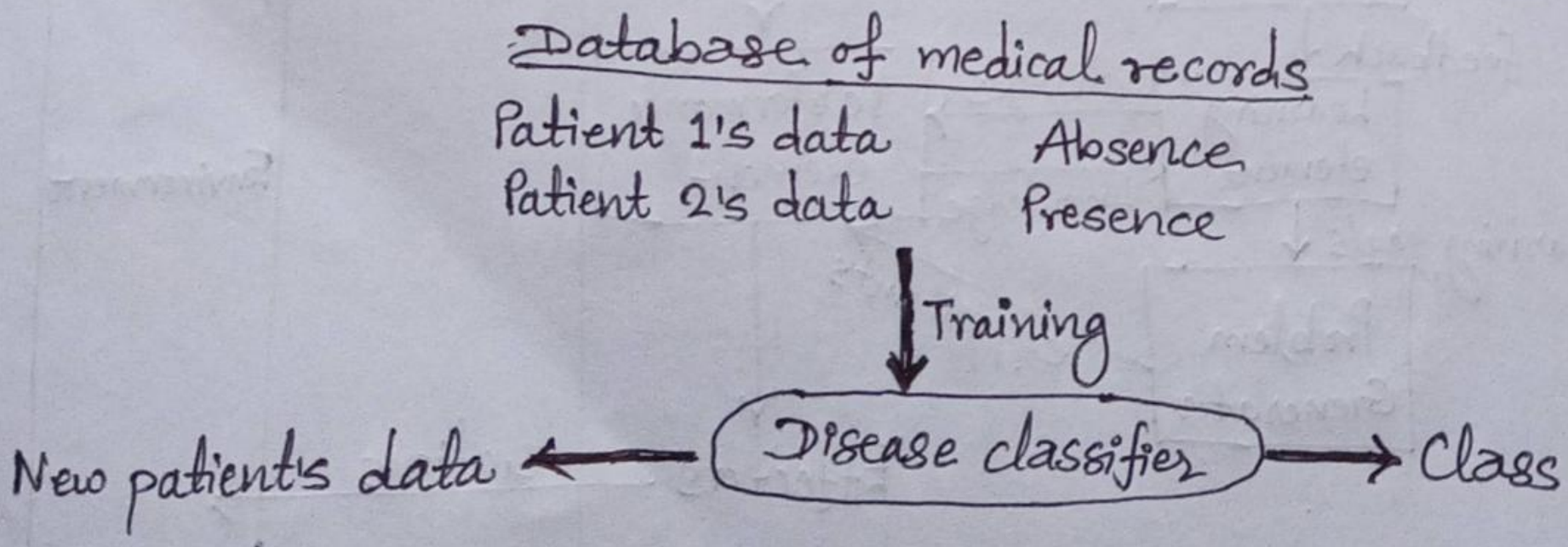
v) Standard system → It is a well programmed system that is able to produce the correct output.

⊛. Types of Machine Learning:

i) Supervised learning → The system is supplied with a set of training examples consisting of inputs and corresponding outputs. Supervised learning can take what it has learnt in the past & apply that to a new data using labelled example to predict future patterns and events. Classification is an example of supervised learning. It is two-step process: model construction & Model usage.



Example: Medical disease classification.



Scanned by TapScanner

ii) Unsupervised learning → It is also called predictive learning. In this learning information is used to train is neither classified nor labelled. It studies how system can infer a function to describe a hidden structure unlabelled data. This is similar to how babies learn easily in their life.

Examples: Clustering of data into similar group, Predict new target market etc.

iii) Semi-supervised learning → It falls somewhere in between supervised and unsupervised learning, since they use both labelled and unlabelled data for training - typically a small amount of labelled data and a large amount of unlabelled data. This method are able to improve learning accuracy.

Example:- Classification of target market after searching information on GOOGLE.

iv) Reinforcement learning → It is a type of dynamic learning that trains agent using reward & punishment. In this learning, agent learns by interacting with environment & consists of three components: Agent (learner), Environment (Agent interacts with) and Action (what agent can do).

Example: FB Newsfeed, Youtube recomendations etc.

Types of Reinforcement: (less imp)

i) Positive → It increases the strength and the frequency of the behaviour. It has a positive effect on the behaviour.

Advantages: → Maximizes performance
→ Sustain change for long period of time.

Disadvantage: Overhead of states can diminish the results.

ii) Negative → Strengthening of a behaviour because a negative condition is stopped or avoided.

Advantage: → Increases behaviour

Disadvantage: Provides enough to meet up the minimum behaviour.

Scanned by TapScanner

⊛. Statical-based Learning: Navie Bayes Model:-

Basics: ← (for understanding and solving problems in easier way)

i) Probability of event X i.e, $P(X) = \frac{\text{No. of times X occurs}}{\text{Total no. of events in sample space}}$

ii) **Joint Probability:** Consider two events A & B that occurs:

$P(A \wedge B) = P(A) * P(B)$ if the events are independent.

$P(A \wedge B) = P(A/B) * P(B)$

iii) **Bayes Theorem:** $P(A/B) = \frac{P(B/A)\,(PA)}{P(B)}$

(write from here if asked in exam)

Navie Baye's Learning / Classifier:

In this model class variable C (which is to be predicted) is the root & the attribute variable x_i are the leaves. The model is called navie because that attributes are conditionally independent of each other in the given class. To calculate Navie Bayes' Learning we use Baye's theorem, which helps to calculate conditional dependent probability.

$$P(C/x_1 \ldots x_n) = \alpha\ P(x) \prod_{i=1}^{n} P(x_i/C).$$

This model is used to calculate the mostly likelihood parameters or attributes of a given class.

Applications of Navie Baye's Model:

→ Real time prediction.
→ Text classification.
→ Recommendation system.

Example:- From the given data set decide whether to play golf or not in today's conditions:

today = (Sunny, Hot, Normal, False).

Scanned by TapScanner

S.N.	Outlook	Tempreature	Humidity	Wind	Play Golf
1.	Rainy	Hot	High	False	No
2.	Rainy	Hot	High	True	No
3.	Overcast	Hot	High	False	Yes
4.	Sunny	Mild	High	False	Yes
5.	Sunny	Cool	Normal	False	Yes
6.	Sunny	Cool	Normal	True	No
7.	Overcast	Cool	Normal	True	Yes
8.	Rainy	Mild	High	False	No.
9.	Rainy	Cool	Normal	False	Yes.
10.	Sunny	Mild	Normal	False	Yes
11.	Rainy	Mild	Normal	True	Yes
12.	Overcast	Mild	High	True	Yes.
13.	Overcast	Hot	Normal	False	Yes.
14.	Sunny	Mild	High	False	No.

Solution:

For Outlook:

	Yes	No	P(yes)	P(No)
Sunny	3	2	3/9	2/5
Overcast	4	0	4/9	0
Rainy	2	3	2/9	3/5
Total	9	5		

For tempreature:

	Yes	No	P(Yes)	P(No)
Hot	2	2	2/9	2/5
Cool	3	1	3/9	1/5
Mild	4	2	4/9	2/5
Total	9	5		

Scanned by TapScanner

For Humidity:

	Yes	No	P(Yes)	P(No)
High	3	4	3/9	4/5
Normal	6	1	6/9	1/5
Total	9	5		

For Wind:

	Yes	No	P(Yes)	P(No)
True	3	2	3/9	2/5
False	6	3	6/9	3/5
Total	9	5		

Since today's conditions are Sunny, Hot, Normal, False only according to the question

Now,

$$P(Yes/Today) = P(Sunny/Yes) \times P(Hot/Yes) \times P(Normal/Yes) \times P(False/Yes) \times P(Yes)$$

$$= \frac{3}{9} \times \frac{2}{9} \times \frac{6}{9} \times \frac{6}{9} \times \frac{9}{14}$$

Total no. of Yes / Total Yes + Total No

$$= 0.0212$$

$$P(No/Today) = P(Sunny/No) \times P(Hot/No) \times P(Normal/No) \times P(False/No) \times P(No)$$

$$= \frac{2}{5} \times \frac{2}{5} \times \frac{1}{5} \times \frac{3}{5} \times \frac{5}{14}$$

$$= 0.0069.$$

Since, sum of probabilities must be equal to 1. So, we have to normalize.

i.e, $$P(Yes/Today) = \frac{P(Yes/Today)}{P(Yes/Today) + P(No/Today)}$$

$$= \frac{0.0212}{0.0212 + 0.0069}$$

$$= 0.7544$$

& $$P(No/Today) = \frac{P(No/Today)}{P(No/Today) + P(Yes/Today)} = \frac{0.0069}{0.0069 + 0.0212} = 0.2456$$

$\because P(Yes/Today) > P(No/Today)$. So, they can play golf in today's conditions.

Scanned by TapScanner

⊛. Learning by Genetic Algorithm (GA):

Genetic Algorithms (GAs) are adaptive heuristic search algorithm that belongs to the larger part of evolutionary algorithms. It is based on the idea of neural selection & genetics. The historical data are provided to find out better solution or simply GA's are used to generate high quality solutions for optimization and search problem.

Genetic Algorithms (GA's) simulate the process of natural selection which means those species who can adopt to change their environment are able to survive & reproduce next generation (fittest survival). Each generation consists of a population of individuals and each individual represents a point in search space & possible solutions which are represented as string of characters/integers/floats/bits.

Operations in Genetic Algorithm:

i) Reproduction/Selection → This operator provides the performance to the individuals with good fitness score and allow them to pass their genes to the successive generator. To select the parents the techniques are round wheel, tournament selection, rank selection etc.

ii) Crossover → This represents mating between two individuals. Two individual are selected using selection operators and crossover sites are choosen randomly then genes at the crossover sites are exchanged thus, it creates completely new offsprings.

Example:

0	1	2	3	4	5	6	7	8	9

5	8	9	4	2	3	5	7	5	8

→

0	1	2	3	4	5	7	5	8

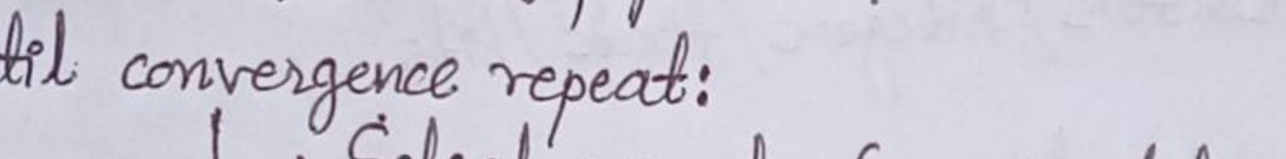
Child after crossover.

The popular crossovers are one-point crossover, multipoint crossover & uniform crossover.

Scanned by TapScanner

iii) Mutation → The key idea is to insert random genes in offspring to maintain the diversity in population to avoid the premature convergence.

Example:



Before mutation → After mutation.

iv) Fitness function → It is the function which takes a candidate solution to the problem as input & produce output, determines how good or fit candidate is.

Genetic Algorithm:

→ Randomly initialize population.
→ Determine fitness of population.
→ Until convergence repeat:
  - → Select parents from population.
  - → Crossover and generate new population.
  - → Perform mutation on new population.
  - → Calculate fitness for new population.

Applications of GA's:

→ Recurrance Neural Networks.
→ Mutation Testing
→ Learning Fuzzy rule.

Scanned by TapScanner

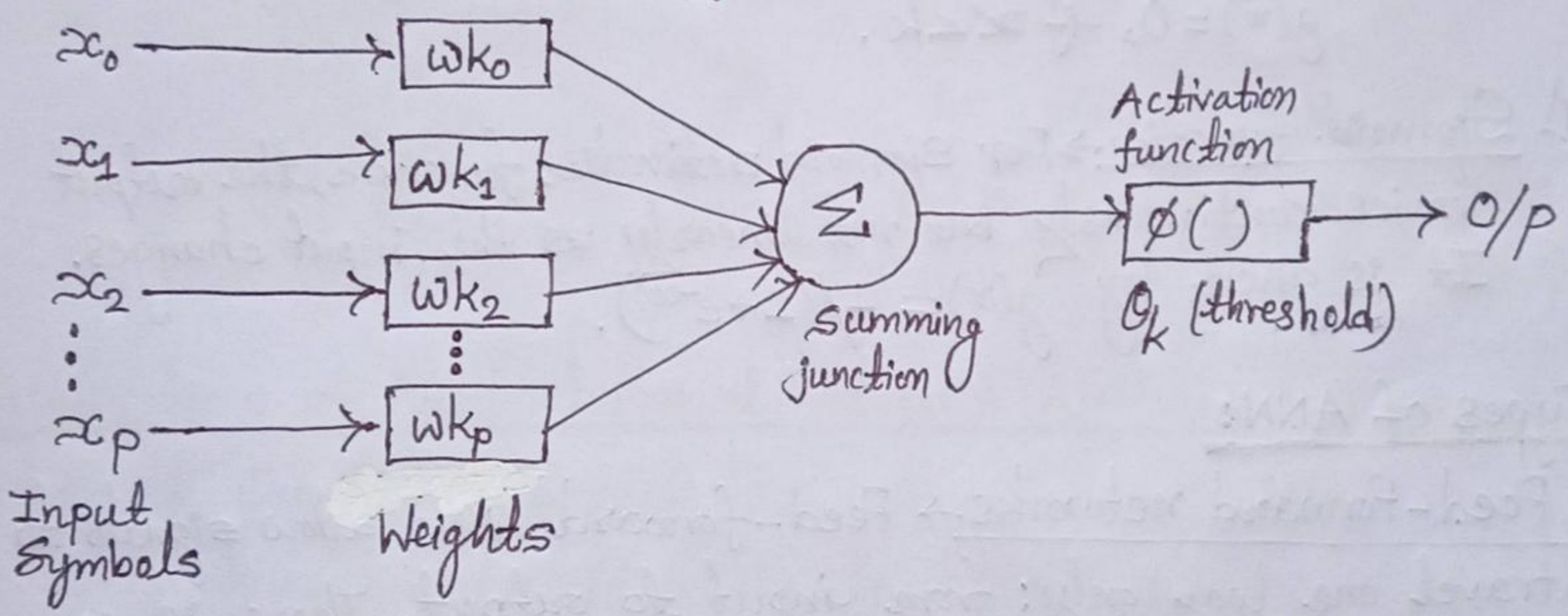
⊛. Learning with Neural Networks:

Artificial Neural Network (ANN):-

ANN is the non-linear parallely distributed & highly connected network having capacity of adaptivity, self-organization, fault tolerence & very large scale integration (VLSI) implementation of neurons. Neural networks are composed of simple elements operating in parallel. These elements are inspired by biological normal system. The network function is determined by connecting neurons in different levels & we can train them to perform some task by assigning some values to the connection called weight.

The goal of ANN is to solve problems in the same way that human mind does. ANN are widely used in computer vision, speech recognization, medical diagnosis, face recognization, signature verification etc.

⊛ Mathematical Model of ANN:



Inputs: Numerical values $x_1, x_2, x_3 \ldots x_p$ are the input in neurons.

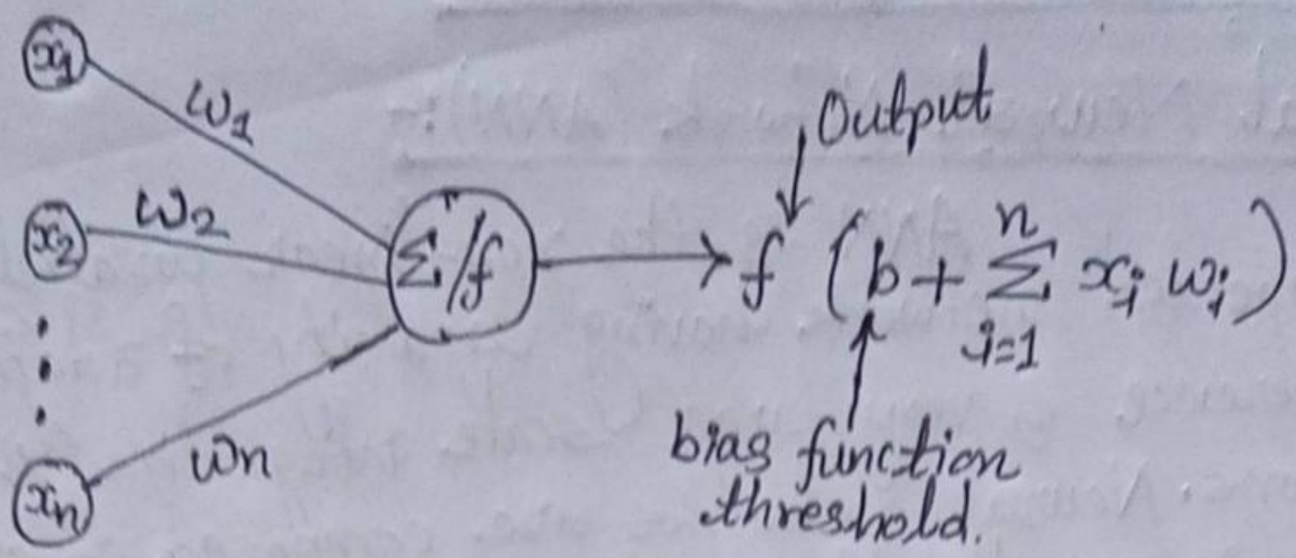
Bias weight: Weight of all neurons determined by threshold (i.e, predefined value to make thing correct).

Activation function: The process of deciding output after aggregating input is called aggregation function. Each unit first computes a weighted sum of its inputs:

$in_i = \sum_{k=0}^{n} w_{k,i} \cdot x_k$, then it applies activation function g to this sum to derive the output: $O_i = g(in_i) = g\left(\sum_{k=0}^{n} w_{k,i} \cdot x_k\right)$.

Scanned by TapScanner

where, g is activation function.



⊛ Types of Activation Functions:

i) Linear function → For linear activation functions, the output activity is proportional to the total weighted output. This function is given by $g(x) = kx + c$, where k & x are constant.

ii) Threshold function → For threshold functions, the output are set at one of the two levels, depending on whether the total input is greater than or less than some threshold value.

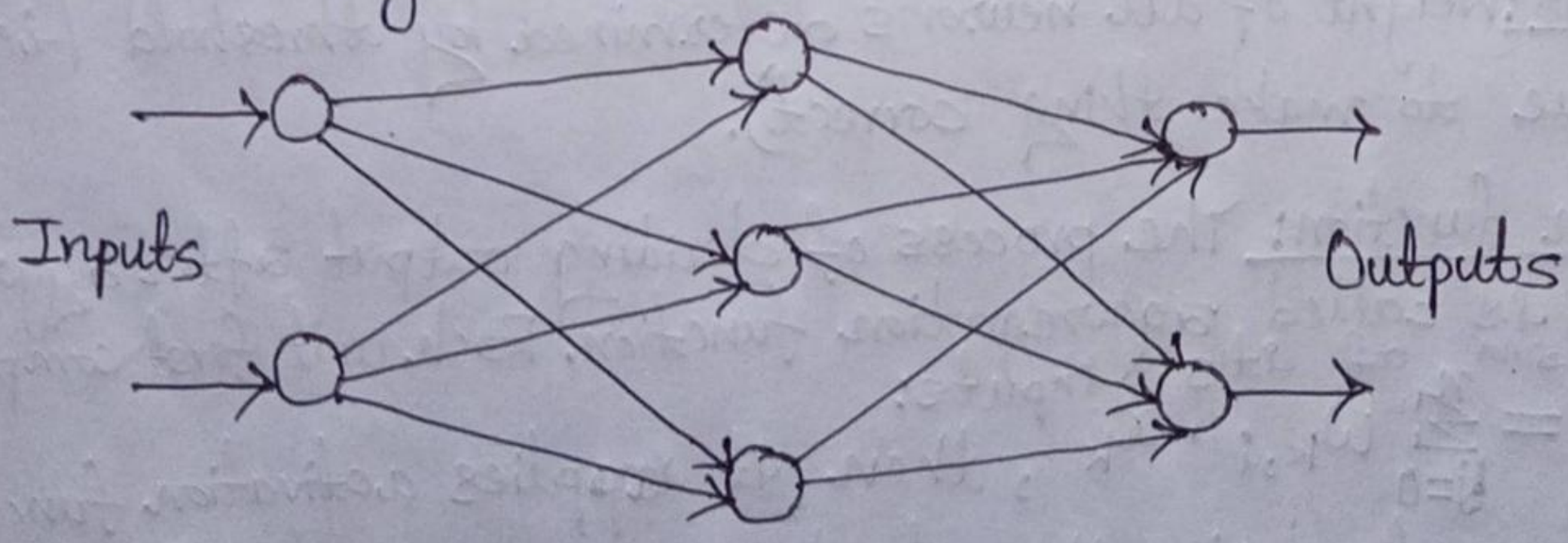
$g(x) = 1$, if $x >= k$

$g(x) = 0$, if $x < k$.

iii) Sigmoid function → For Sigmoid activation functions, the output varies continuously but not linearly as the input changes. It is given by $g(x) = 1/(1+e^{-x})$.

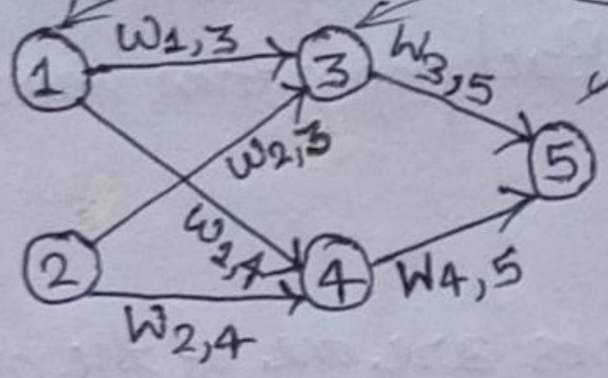
⊛ Types of ANN:

i) Feed-Forward networks → Feed-forward ANNs allow signals to travel one way only: from input to output. There is no feedback (loops). i.e, the output of any layer does not affect the same layer. This types of ANNs are extensively used in pattern recognization.



Scanned by TapScanner

Example:



first layer a_1
second layer a_2
third layer a_3

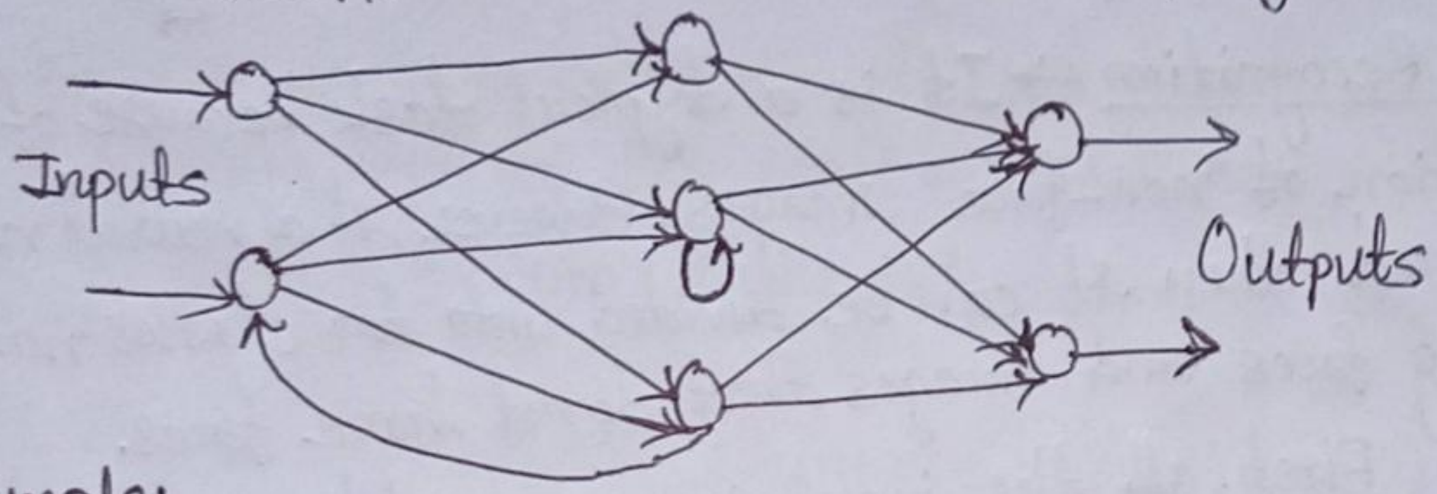
output मी एतील दिले

$$g_3 = g(w_{1,3} a_1 + w_{2,3} a_2)$$
$$g_4 = g(w_{1,4} a_1 + w_{2,4} a_2)$$
$$g_5 = g(w_{4,5} a_2 + w_{3,5} a_3)$$

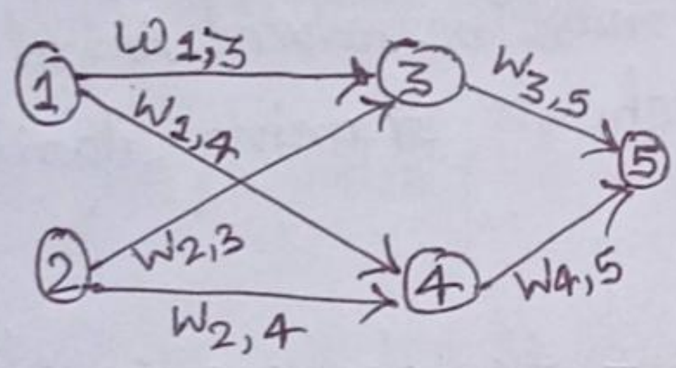
Note: If fig. contains 2 layer i.e, a_1, a_2 then it is single layer. If contains more than two it is multilayer.

Examples described here are multilayer since contains more than 2 i.e, 3 layers

ii) Feedback networks (Recurrent networks): → Feedback networks can have signals travelling in both directions by introducing loops in the network.



Example:



Here,

$$g5 = g(w_{3,5} a_3 + w_{4,5} a_4)$$
$$= g(w_{3,5}\, g(w_{1,3} a_1 + w_{2,3} a_2) + g(w_{1,4} a_1 + w_{2,4} a_2)).$$

⊛ Applications of Artificial Neural Networks (ANNs):

i) Speech Recognition → Speech occupies a prominent role in human-human interaction. Therefore, it is natural for people to expect speech interfaces with computers. ANN has made great progress in this field. Following ANNs have been used for speech recognition:-

→ Multilayer networks
→ Multilayer networks with recurrent connections.
→ Kohonen self-organizing feature map.

ii) Character Recognition → It is an interesting problem which falls under the general area of Pattern Recognition. Many neural networks have been developed for automatic recognition of handwritten characters, either letters or digits. Following are the

Scanned by TapScanner

Some ANNs which have been used for character recognition:

→ Multilayer neural networks such as Backpropagation neural networks.

→ Neocognitron.

iii) Signature Verification Application → Signatures are one of the most useful ways to authorize and authenticate a person in legal transactions. Signature verification technique is a non-vision based technique. The trained neural network will classify the signature as being genuine or forged under the verification stage.

iv) Human Face Recognition → It is a typical task because of the characterization of "non-face" images. However, if a neural network is well trained, then it can be divided into two classes namely images having faces and images that do not have faces.

First, all the input images must be processed. Then, the dimensionality of that image must be reduced. And at last it must be classified using neural network training algorithm.

⊛ Learning by Training ANN:

Learning means to adopt the changes in itself when there is change in environment. In complex system of ANN, after learning it changes its internal structure based on the information passing through it.

Learning is important in ANN because of some changes occuring in environment then ANN must change its inputs, outputs, activation function & Weight. To change the internal structure of ANN there are various methods (i.e, learning rules) which are as follows;

1) Hebbian learning rule: Hebbian learning rule is introduced by Dnald Hebbian in 1949. It is also called unsupervised feed forward learning. This rule basically states that, if neuron i is near enough to excite neuron j and repeatedly participates in its activation, the synaptic connection between these two neurons is strengthened (i.e, made stronger) and neuron j becomes more sensitive to

Scanned by TapScanner

from neuron i.

Hebb's law can be represented in the form of two rules:

i) If two neurons on either side of a connection are actived synchronously, then the weight of that connection is increased.

ii) If two neurons on either side of a connection are actived asynchronously, then the weight of that connection is decreased.

Hebb's Algorithm:

Step1: Initialize all weights and bias to 0.

Step2: Given a training input s, with its target output t, set the activations of the input units: $x_i = s_i$

Step3: Set the activation of the output unit to the target value: $y = t$

Step4: Adjust the weights: $w_i(new) = w_i(old) + x_i y$

Step5: Adjust the bias (just like the weights): $b(new) = b(old) + y$

Problem: Construct a Hebb Net which performs like an AND function.

Solution:

Let us construct truth table of AND gate as follows:-

b always 1 हुन्छ / x_1 र x_2 दुवै 1 भए मात्र y 1 हुन्छ अन्यत्र -1. AND Gate जस्तै same but 0 लाई -1 ले represent गरेको मात्र

INPUT				TARGET	
	x_1	x_2	b		y
X_1	-1	-1	1	Y_1	-1
X_2	-1	1	1	Y_2	-1
X_3	1	-1	1	Y_3	-1
X_4	1	1	1	Y_4	1

There are 4 training samples, so there will be 4 iterations.

Step1: Set weighs and bias to zero, $W = [0\ 0\ 0]$ and $b = 0$.

Step2: Set the activations of input units $x_i = s_i$ for $i = 1$ to 4.

$X_1 = [-1\ -1\ 1]$

$X_2 = [-1\ 1\ 1]$

$X_3 = [1\ -1\ 1]$

$X_4 = [1\ 1\ 1]$

i.e. $X_1 = [x_1\ x_2\ b]$

Scanned by TapScanner

Step3: Output value is set to $y=t$.

Step4: Adjust (or Modify) weights using Hebbian Rule:

1st iteration: $W(new) = w(old) + x_1 y_1$

$= [0\ 0\ 0] + [-1\ -1\ 1] . [-1]$

$= [1\ 1\ -1]$

2nd iteration: $w(new) = w(old) + x_2 y_2$

(for 2nd iteration, final wt. of 1st iteration will be used & so on...)

$= [1\ 1\ -1] + [-1\ 1\ 1] . [-1]$

$= [2\ 0\ -2]$

3rd iteration: $w(new) = w(old) + x_3 y_3$

$= [2\ 0\ -2] + [1\ -1\ 1] . [-1]$

$= [1\ 1\ -3]$

4th iteration: $w(new) = w(old) + x_4 y_4$

$= [1\ 1\ -3] + [1\ 1\ 1] . [1]$

$= [2\ 2\ -2]$.

So, the final weight matrix is $[2\ 2\ -2]$.

($[x_1, x_2, b]$ form)

Testing the Network: ← Step 5

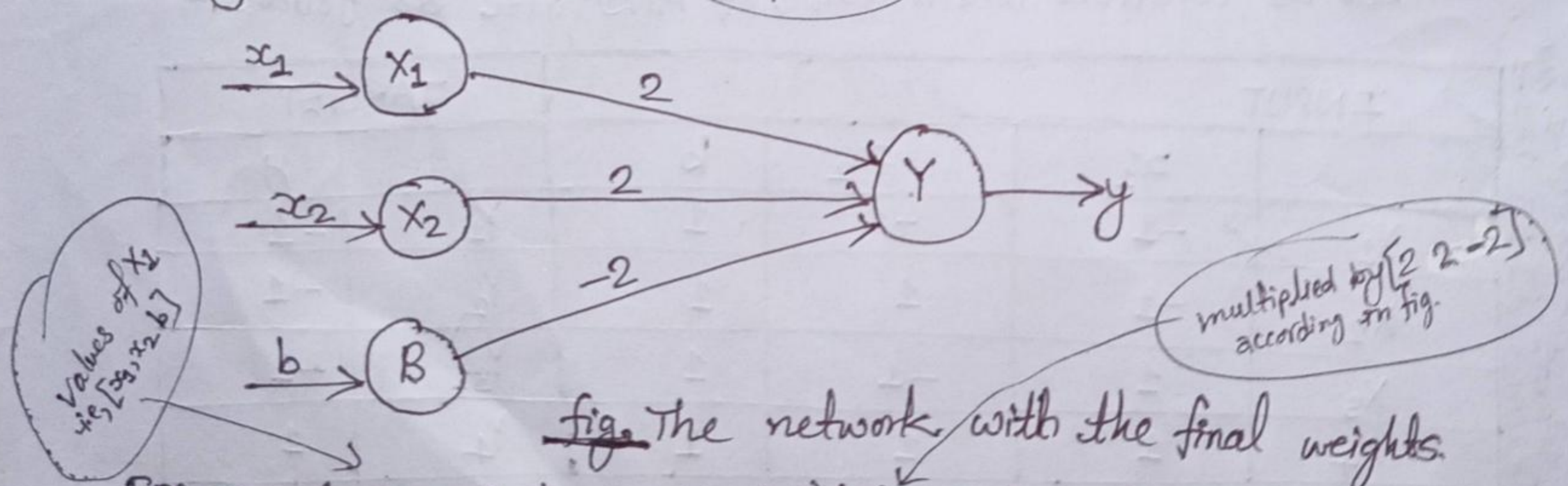


fig. The network with the final weights.

For $x_1=-1, x_2=-1, b=1$, $Y=(-1)(2)+(-1)(2)+(1)(-2)=-6$

For $x_1=-1, x_2=1, b=1$, $Y=(-1)(2)+(1)(2)+(1)(-2)=-2$

For $x_1=1, x_2=-1, b=1$, $Y=(1)(2)+(-1)(2)+(1)(-2)=-2$

For $x_1=1, x_2=1, b=1$, $Y=(1)(2)+(1)(2)+(1)(-2)=2$

(i.e, first 3 values negative and final only positive)

The results are all compatible with the original table.

Decision Boundry:

(from fig)

We have $2x_1+2x_2-2b=y$

Now replacing y with 0, we get, $2x_1+2x_2-2b=0$

Since, bias, $b=1$, so, $2x_1+2x_2-2(1)=0$ i.e, $2(x_1+x_2)=2$

Scanned by TapScanner

From $2(x_1+x_2)=2$, we get final equation, $x_2=-x_1+1$.

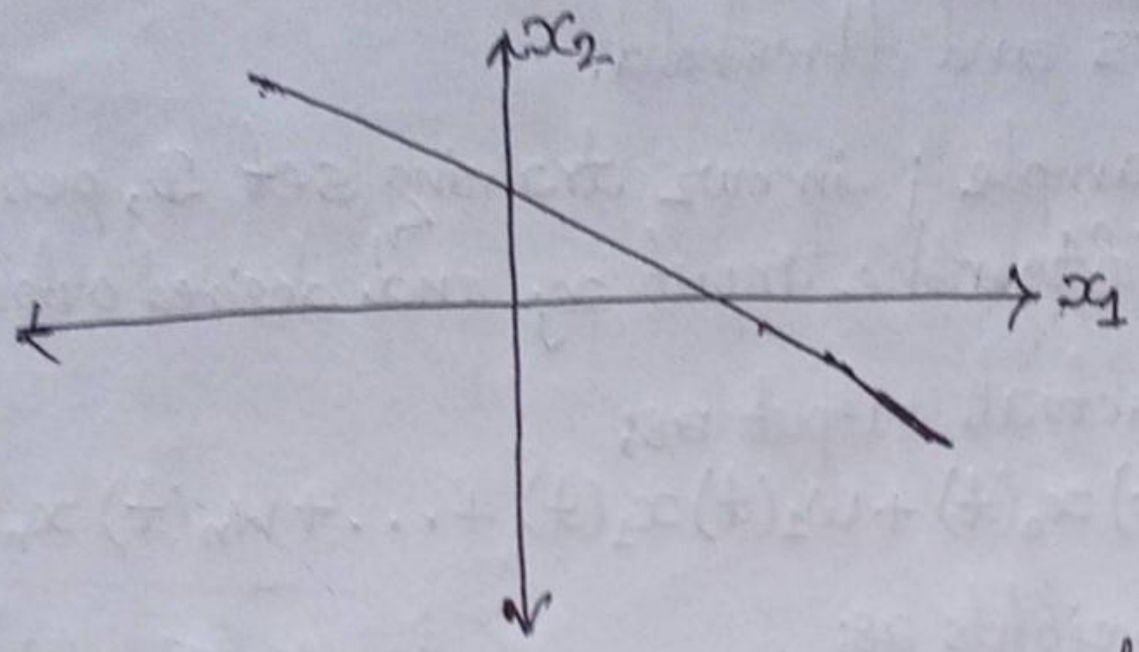


fig. Decision Boundry of AND Function.

less imp but once have a look similar to algorithm only use of { to n braces do

Pseudocode for Hebb's Net (Supervised)

```
Initialization; for i=1 to n : {b=0, w_i=0}.
For each of the training samples s: to do
{ /*s is the input pattern, t the target output of the sample */
   for i=1 to n {x_i=s_i}   /*set to input units*/
                y=t         /*set y to the target*/
   for i=1 to n{
                w_i=w_i+x_i*y
                } /*update weight*/
   b=b+x_i*y /*update bias*/
}
```

2) Perceptron Learning:

Perceptron learning is used in supervised learning rule to classify the data in two classes. It consist of a single neuron with arbitary value 1 or 0 depending on threshold & also consist of bias weight. Training patterns are presented to the networks inputs; the output is computed.

Variables used:

* $y=f(x)$ denotes the output for an input vector x.
* $D=(x_1,d_1),(x_2,d_2),\ldots,(x_n,d_n)$ is the training set of n samples.

where, $x_j \rightarrow$ input vector
$d_j \rightarrow$ desired output for x_j.

Scanned by TapScanner

Algorithm:

Step1: Initialize weights and threshold.

Step2: For each example j in our training set D, perform following steps 3 & 4 over the input x_j and desired output d_j.

Step3: Calculate the actual output as;

$$y(t) = g[w_0(t)x_0(t) + w_1(t)x_1(t) + \ldots + w_n(t)x_n(t)]$$

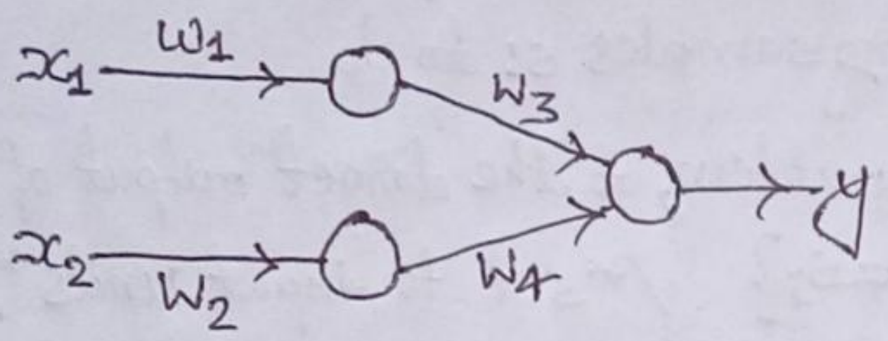
i.e, $y(t) = g[\sum_i w_i(t) \cdot x_i(t)]$

Step4: Update the weights as;

$$w_i(t+1) = w_i(t) + \alpha [d(t) - y(t)] x_i(t)$$

where, $0 <= \alpha <= 1$ (learning rate)

Example:



Given, desired output $d(t)$.

Actual output is computed as;

$$y(t) = g(y_1 w_3(t) + y_2 w_4(t))$$

where, $y_1 = g(x_1 w_1(t))$

$y_2 = g(x_2 w_2(t))$

Now update weight as;

$$w_1(t+1) = w_1(t) + \alpha [d(t) - y(t)] x_1(t)$$
$$w_2(t+1) = w_2(t) + \alpha [d(t) - y(t)] x_2(t)$$
$$w_3(t+1) = w_3(t) + \alpha [d(t) - y(t)] y_1(t)$$
$$w_4(t+1) = w_4(t) + \alpha [d(t) - y(t)] y_2(t)$$

After updating weights, we recompute $y(t)$. If the desired solution is obtained then stop otherwise we will iterate again.

Scanned by TapScanner

3) Back-propagation learning:-

It is a supervised learning method, and is an implementation of Delta rule. It requires a teacher that can calculate the desired output for any given input. It is most useful for feed-forward networks. Back propagation requires that the activation function used by the artificial neurons (or nodes) is differentiable.

Algorithm:

1. Randomly choose the initial weight.
2. Compute output (y) from neural network:

$$y = g(\sum x_i w_i) \text{ where, } g(x) = \frac{1}{1+e^{-x}} \text{ is sigmoid function.}$$

3. Compute error $\delta = t - y$; where t is targated output and y is actual output.
4. Propagate this δ error back to layer of neural network and compute error at each layer as;

$$\delta_i = \sum w_{ij} \delta_j$$

5. Update the weights as;

$$w_{ij}(new) = w_{ij}(old) + \alpha \delta_j \frac{d(y_i)}{de} x_i$$

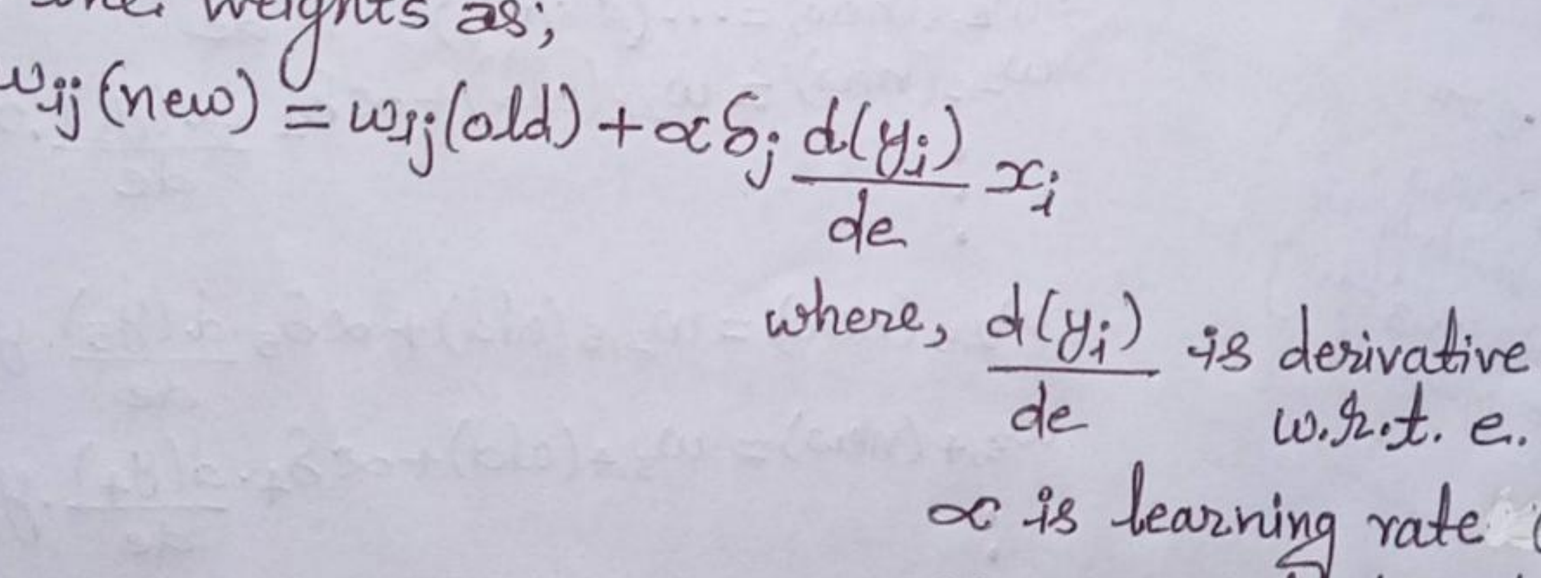
where, $\frac{d(y_i)}{de}$ is derivative of y_i w.r.t. e.

α is learning rate $0 \leq \alpha \leq 1$

x_i is input to node.

6. Repeat step 2 to 5 until y converges to t.

Example 1:



Scanned by TapScanner

$$y = g(y_4 w_{4,6} + y_5 w_{5,6})$$

where, $y_4 = g(y_1 w_{1,4} + y_2 w_{2,4} + y_3 w_{3,4})$

$y_5 = g(y_1 w_{1,5} + y_2 w_{2,5} + y_3 w_{3,5})$

where,

$y_1 = g(x_1 w_{1,1} + x_2 w_{2,1})$

$y_2 = g(x_1 w_{1,2} + x_2 w_{2,2})$

$y_3 = g(x_1 w_{1,3} + x_2 w_{2,3})$

Given target t, Compute $\delta = t - y$.

Now calculate backpropagate error as;

$\delta_4 = w_{4,6} * \delta$

$\delta_5 = w_{5,6} * \delta$

$\delta_1 = w_{1,4}\delta_4 + w_{1,5}\delta_5$

$\delta_2 = w_{2,4}\delta_4 + w_{2,5}\cdot\delta_5$.

$\delta_3 = w_{3,4}\cdot\delta_4 + w_{3,5}\cdot\delta_5$.

if confusion once look and compare with algorithm.

Now update weight as;

$$w_{11}(new) = w_{1,1}(old) + \alpha\,\delta_1 \cdot \frac{d(y_1)}{de} \cdot x_1.$$

$$w_{1,2}(new) = w_{1,2}(old) + \alpha\,\delta_2 \frac{d(y_2)}{de} \cdot x_1$$

$w_{1,3}$ हुटेको ε_1 वा As $w_{1,1}$, $w_{1,2}$ similarly we can do $w_{1,3}$ easily do yourself

$w_{1,3}(new) = \ldots$ (similarly)

$$w_{2,1}(new) = w_{2,1}(old) + \alpha\,\delta_1 \cdot \frac{d(y_1)}{de} \cdot x_2$$

$$\vdots$$

$$w_{2,5}(new) = w_{2,5}(old) + \alpha\,\delta_5 \cdot \frac{d(y_5)}{de} \cdot y_2$$

$$w_{3,4}(new) = w_{3,4}(old) + \alpha\,\delta_4 \cdot \frac{d(y_4)}{de} \cdot y_3$$

$$\vdots$$

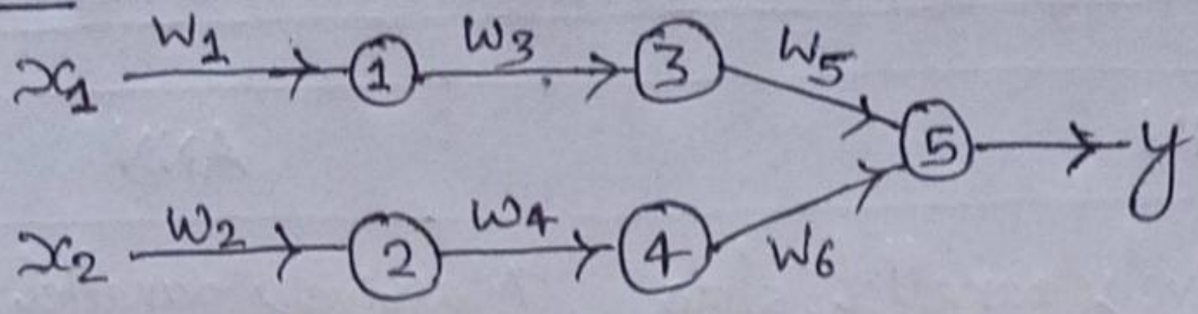
$$w_{4,6}(new) = w_{4,6}(old) + \alpha \cdot \delta \cdot \frac{dy}{de} \cdot y_4$$

$$w_{5,6}(new) = w_{5,6}(old) + \alpha \cdot \delta \cdot \frac{dy}{de} \cdot y_5$$

After updating weights., we recompute y. Then we compute $\delta = t - y$. If the value of y converges to 't' then the value is fixed up otherwise we will iterate again.

Scanned by TapScanner

Example 2:



$$y = g(y_3 w_5 + y_4 w_6)$$

where, $y_3 = g(y_1 w_3)$

$y_4 = g(y_2 w_4)$

where, $y_1 = g(x_1 w_1)$

$y_2 = g(x_2 w_2)$

given target t, Compute $\delta = t - y$.

Now compute backpropagate error as;

$$\delta_3 = w_5 \cdot \delta$$

$$\delta_4 = w_6 \cdot \delta$$

$$\delta_1 = w_3 \cdot \delta_3$$

$$\delta_2 = w_4 \cdot \delta_4$$

Now update weight as;

$$w_5(new) = w_5(old) + \alpha \cdot \delta \cdot \frac{dy}{de} \cdot y_3$$

$$w_6(new) = w_6(old) + \alpha \cdot \delta \cdot \frac{dy}{de} \cdot y_4$$

$$w_3(new) = w_3(old) + \alpha \cdot \delta_3 \cdot \frac{d(y_3)}{de} \cdot y_1$$

$$w_4(new) = w_4(old) + \alpha \cdot \delta_4 \cdot \frac{d(y_4)}{de} \cdot y_2$$

$$w_1(new) = w_1(old) + \alpha \cdot \delta_1 \cdot \frac{d(y_1)}{de} \cdot x_1$$

$$w_2(new) = w_2(old) + \alpha \cdot \delta_2 \cdot \frac{d(y_2)}{de} \cdot x_2$$

After updating weights, we recompute y. Then we compute $\delta = t - y$. If the value of y converges to t then the value is fixed up otherwise we will iterate again.

Scanned by TapScanner

⊛ Biological Neural Networks (BNN) vs. Artificial Neural Networks (ANN):

Criteria	BNN	ANN
i) Processing	Massively parallel, slow but superior than ANN	Massively parallel, fast but inferior than BNN
ii) Size	10^{11} neurons and 10^{15} interconnections	10^2 to 10^4 nodes (mainly depends on type of application and network designer).
iii) learning	They can tolerate ambiguity.	Very precise, structured and formatted data is required to tolerate ambiguity
iv) Fault tolerance	Performance degrades with even partial damage.	It is capable of robust performance, hence has the potential to be fault tolerent.
v) Storage Capacity	Stores the information in the synapse.	Stores the information in continious memory locations.

⊛ Supervised Learning vs. Unsupervised Learning:

Supervised learning	Unsupervised learning
i) Supervised learning algorithms are ~~used~~ trained using labeled data.	i) Unsupervised learning algorithms are trained using unlabeled data.
ii) Supervised learning model predicts the output.	ii) Unsupervised learning model finds the hidden patterns in data.
iii) In supervised learning, input data is provided to the model along with the output.	iii) In ~~s~~unsupervised learning, only input data is provided to the model.
iv) Supervised learning needs supervision to train the model.	iv) Unsupervised learning does not need any supervision to train the model.
v) Supervised learning model produces an accurate result.	v) Unsupervised learning model may give less accurate result as compared to supervised learning.

Scanned by TapScanner