Machine Learning

Learning

learning is acquiring new or modifying existing knowledge, behaviours, skills and may involve synthesizing different types of information.

Why learning?

→ Huge computational power is available.
→ To understand and improve efficiency of human learning.
→ To discover new things or structure that is unknown to humans.
→ To fill in skeletal or incomplete specification about a domain.

Machine learning

Machine learning is a branch of AI that provides computer the ability to learn and improve its learning from experience without being written rules explicitly.
→ Machine learning focuses on prediction based on known properties learned from the training data.

Machine learning & AI

AI refers to a very large field of research that encompasses a number of techniques aimed at developing computers that can learn and solve problem whereas ML is the field of AI concerned with learning from data on its own.

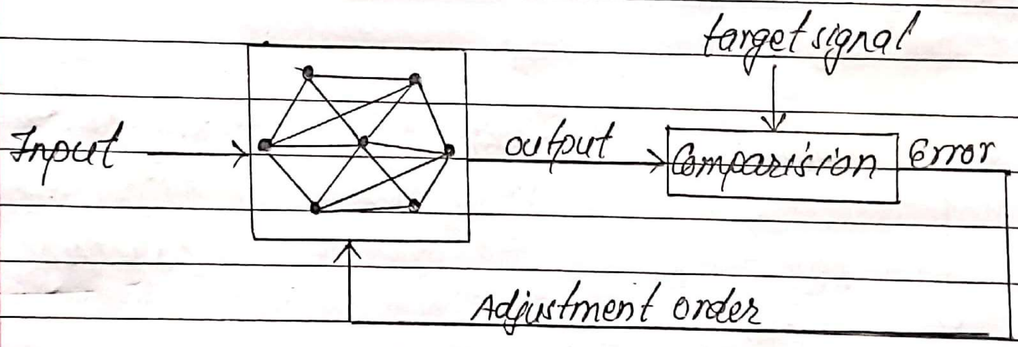
Types of Learning

1) Supervised Learning

The system is supplied with a set of training examples consisting of inputs and corresponding outputs, and is required to discover the relation or mapping betn them.

In supervised learning, the machine is presented with inputs together with the target outputs. Then, the machine network tries to produce an output as close as possible to the target signal by adjusting the values of internal weights.

e.g.
Error correction method



2) Unsupervised learning

The system is supplied with a set of training examples consisting only of inputs and is required to discover for itself what appropriate outputs should be.
e.g. self organizing Map

→ In unsupervised learning, there is no hint at all about the correct output.

→ It consists of getting data without any label and try to group data depending on their features. – clustering.

5) Rote learning

Rote learning is a memorization technique based on repetition.
→ Rote learning techniques avoids understanding the inner complexities but focuses on memorizing the materials so that it can be recalled by learner exactly the way it read or heard.

Vimp

Learning Framework / Architecture

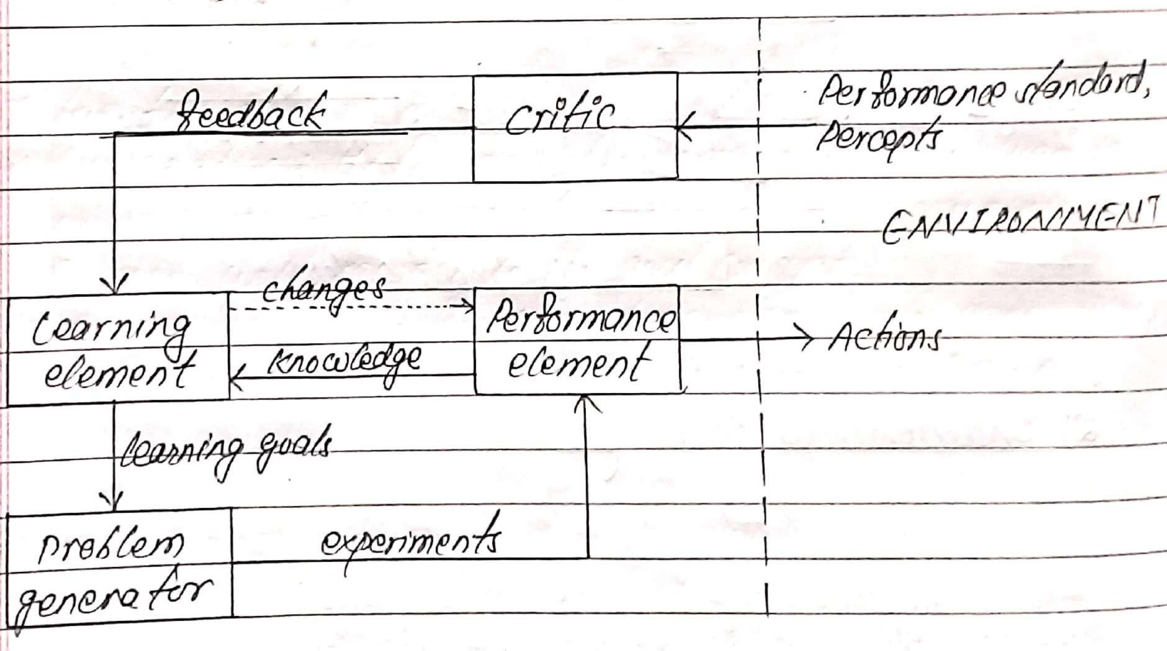


fig: Architecture of a learning system / learning framework

It consists of the following components:

1) Learning element:
It is responsible for making improvements. It uses knowledge about the agent and feedback on its actions to improve performance.

2) Performance element:
It is responsible for selecting external actions. The performance element takes in percepts and decides on actions.

3) Critic (feedback element):
It tells learning element how agent is doing by comparing with the fixed standard of performance. The feedback is used to determine what should be done in order to produce the correct output.

4) Problem generator:
It is responsible for suggesting actions that will generate new experience that helps the system to train further.

Example:
Automated taxi on city road.

- Performance element : Consists of knowledge & procedures for driving actions. e.g. turning, accelerating, breaking are the performance elements on road.
- Learning element : It formulates goals. e.g. learn rules for breaking, accelerating, learn geography of the city.
- Critic : observes world and passes information to learning element. e.g. quick right turn accross three lanes of traffic, observes reaction of other drivers.
- problem generator : Try south city road.

Explanation Based learning

An explanation based learning (EBL) system accepts an example (i.e. a training example) and explains what it learns from the example. The EBL system takes only the relevant aspects of the training. This explanation is translated into particular form that a problem solving program can understand. The explanation is generalized so that it can be used to solve other problems.

EBL accepts four inputs:

1. A training example : what the learning sees in the world.
2. A goal concept : the set of all possible conclusion
3. An operational criterion: a description of which concepts are usuable.
4. A domain theory : a set of rules that describes relationship betn objects and actions in a domain.

It has two steps:

i) Explanation : In this step, the domain theory is used to prune away all unimportant aspects of the training example with respect to the goal concept.

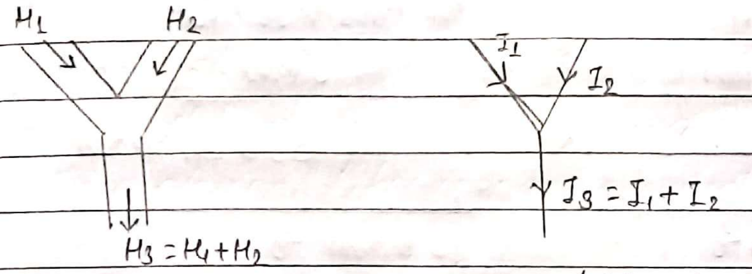
ii) Generalisation : In this step, the explanation is generalized as far possible while still describing the goal concept.

Learning by Analogy (case base learning)

Learning by analogy means acquiring new knowledge about an input entity by transferring it from a known similar entity.

This technique transforms the solution of problems in one domain to the solutions of the problems in other domain

by discovering analogous states and operators in the two domains
e.g.
Infer by analogy the hydraulic's laws that are similar to kirchoff's law.



Hydraulic law Kirchoff's law

Two methods of analogy problem solvings are :

1) Transformational Analogy :

It focuses on final solution. It look for a previous similar situation and copy it to the new situations making suitable substitution when appropriate.

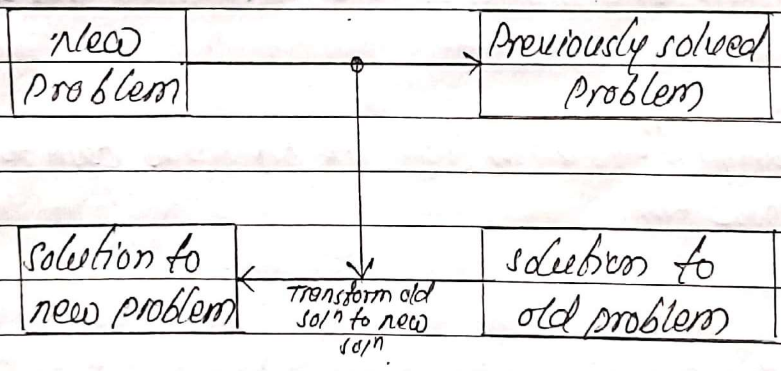


fig: Transformational analogy.

2) Derivational Analogy :

It focuses on process of problem solving.
- Transformational analogy doesn't look how problem was solved, it only looks at the final soln. But in derivational analogy,

it replays a previous derivation and modify it if necessary.
- If initial assumptions and steps are same copy it then, else alternative need to be found.

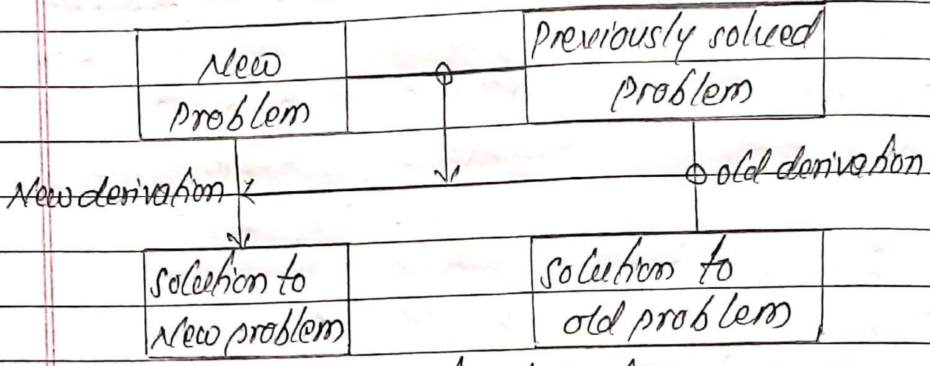


fig: Derivational Analogy.

Learning by analogy consist of following steps / Components:

1. Retrieve:
Given a target problem, retrieve cases from memory that are relevant to solving it. A case consists of a problem, its solution, and process of how the solution was derived.

2. Reuse:
Map the solution from the previous case to the target problem.

3. Revise:
If necessary, revise the already existing solution from previous case for new problem.

4. Retain:
After the solution has been successfully adapted to the ~~tar~~ target problem, save the solution as a new case in memory.

Learning by Evolution (Genetic Algorithm)

Genetic algorithm is a search-based optimization technique based on the principles of Genetics and natural selection.
A genetic algorithm applies the principles of evolution found in nature to the problem of finding an optimal solution to a solver problem. In a genetic algorithm, the problem is encoded in a series of bit strings that are manipulated by the algorithm.

Five phases are considered in a genetic algorithm:

1. Initial population
2. Fitness function
3. Selection
4. Crossover
5. Mutation

$\left.\begin{array}{l}\text{3. Selection}\\ \text{4. Crossover}\\ \text{5. Mutation}\end{array}\right\} \rightarrow$ Operators in genetic algorithm

1. Initial population:
Population is a subset of solutions (set of individuals) in the current generation. Each subset is characterized by a set of parameters (variables) known as Genes. Genes are joined into a string to form a chromosome that becomes a solution.

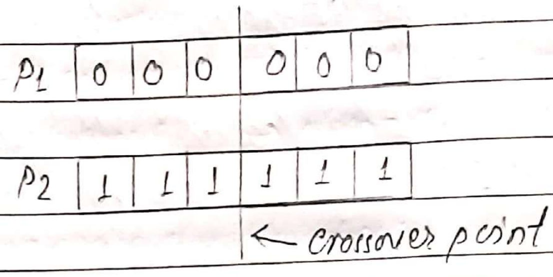
2. Fitness function:
The fitness function determines how fit an individual is (the ability of an individual to compete with other individuals). It gives a fitness score to each individual.

3. Selection:
The idea of selection phase is to select the individuals with good fitness scores and allow them to pass their genes to the next generation. Two pairs of individuals (parents) are selected based on their fitness scores. (high fitness → more chance to be selected)

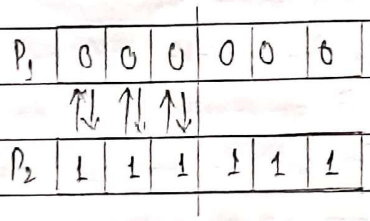
4. Crossover:

This represents mating betn individuals. For each pair of parents to be mated, a crossover point is choosen at random.

E.g.



Offsprings are created by exchanging the genes of parents among themselves until the crossover point is reached.



The new offsprings:

P3	1	1	1	0	0	0
P4	0	0	0	1	1	1

5. Mutation:

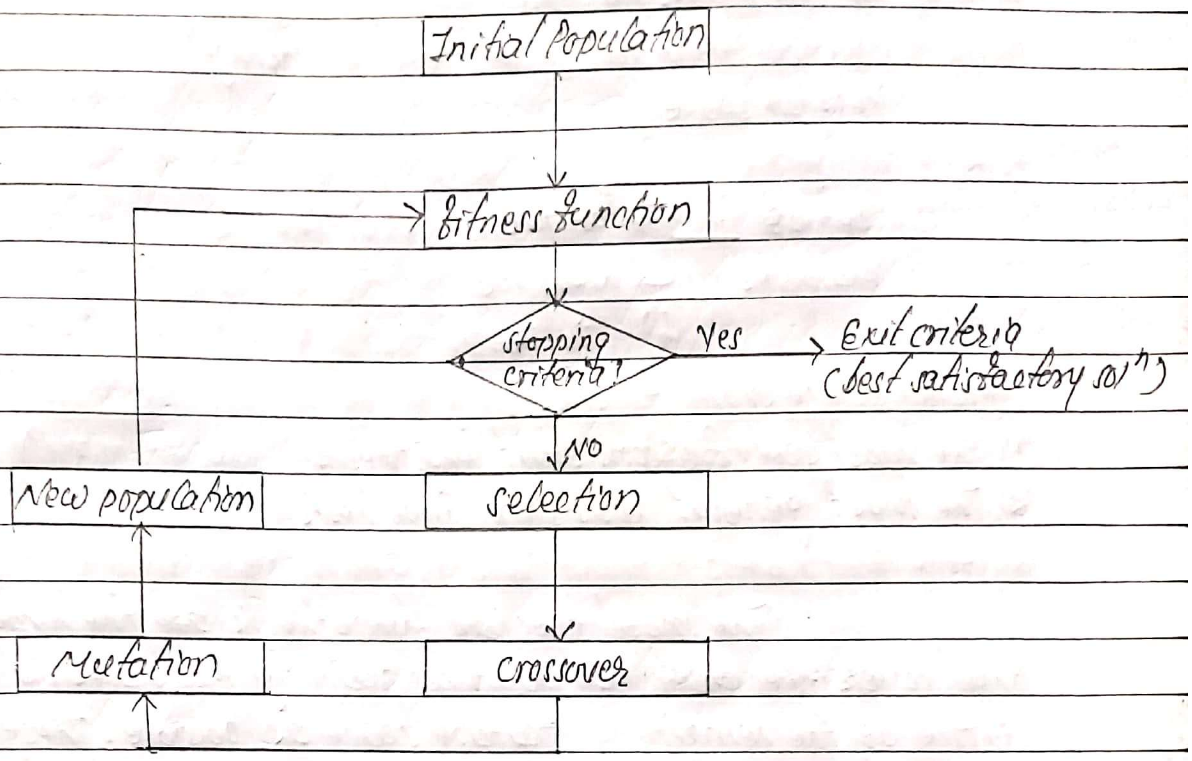
Mutation is applied to each child individually after crossover. It randomly alters some genes with a small probability

E.g.

Mutation of P3

1	1	0	1	1	0

Flowchart:



Learning through examples (Induction / Inductive Learning)

Learning by example is a general learning strategy where a concept is learned by drawing inductive inferences from a set of facts.

- Inductive learning requires a certain number of training examples to achieve a given level of generalization accuracy.
- It generalize from observed training examples by identifying features that empirically distinguish positive from negative examples.

The learning by example consists of the idea of version space.
A version space is a hierarchical representation of knowledge that enables you to keep track of all the useful information

supplied by a sequence of learning examples without remembering any of the examples.

$$\begin{tikzcd}
\text{Version space} \arrow[d] & \\
\text{set of Examples} \arrow[r] \arrow[dr] & \text{Positive} \\
& \text{Negative}
\end{tikzcd}$$

Version space characteristics:

1) The most specific consistent hypothesis
2) The most general consistent hypothesis

The most specific hypothesis are the hypothesis that cover the observed positive training examples, and as little of the remaining feature space as possible. These are hypotheses, ~~where~~ which if reduced any further would exclude a positive training example, and hence become inconsistent.

The most general hypothesis are those which cover the observed positive training examples, but also cover as much of the remaining feature space without including any negative training examples. These are hypothesis which if enlarged any further would include a negative training examples, and hence become inconsistent.

Candidate Elimination Algorithm

↳ used to construct generalization tree & specialization tree.

- Nodes in the generalization tree are connected to a model that matches everything in its subtree.
- Nodes in the specialization tree are connected to a model

that matches only one thing in its subtree.

1. Initialize G, the set of maximally general hypothesis (Null).
2. Initialize S, the set of maximally specific hypothesis, to contain one element: the 1st positive example.
3. Accept new training examples

If it is positive:
 i) Generalize all the specific models to match positive example.
 ii) Prune away all the general models that fails to match the positive example.

If it is negative:
 i) specialize all the general model to prevent match with negative example.
 ii) Prune away all the specific models that match the negative example.

If S and G are both singleton sets:
 → If they are identical, output their value and halt.
 → If they are different, the training cases were inconsistent. Output this result and halt.
 → Else continue accepting new training examples.

e.g.
Learning of Japanese Economy car.
Attributes: (Country, Manufacture, color, Decade, Type)

Country	Manufacturer	Color	Decade	Type	Example type
Japan	Honda	Blue	1980	Economy	+ve
Japan	Toyota	Green	1970	sports	-ve
Japan	Toyota	Blue	1990	Economy	+ve
USA	Chrysler	Red	1980	Economy	-ve
Japan	Honda	white	1980	Economy	+ve

Initialize

$G = (x_1, x_2, x_3, x_4, x_5)$

$S =$ (Japan, Honda, Blue, 1980, Economy)

1. +ve Example

The generalization-specialization tree is

$$\begin{tikzcd} G \; (x_1, x_2, x_3, x_4, x_5) \\ S \; (\text{Japan, Honda, Blue, 1980, Economy}) \end{tikzcd}$$

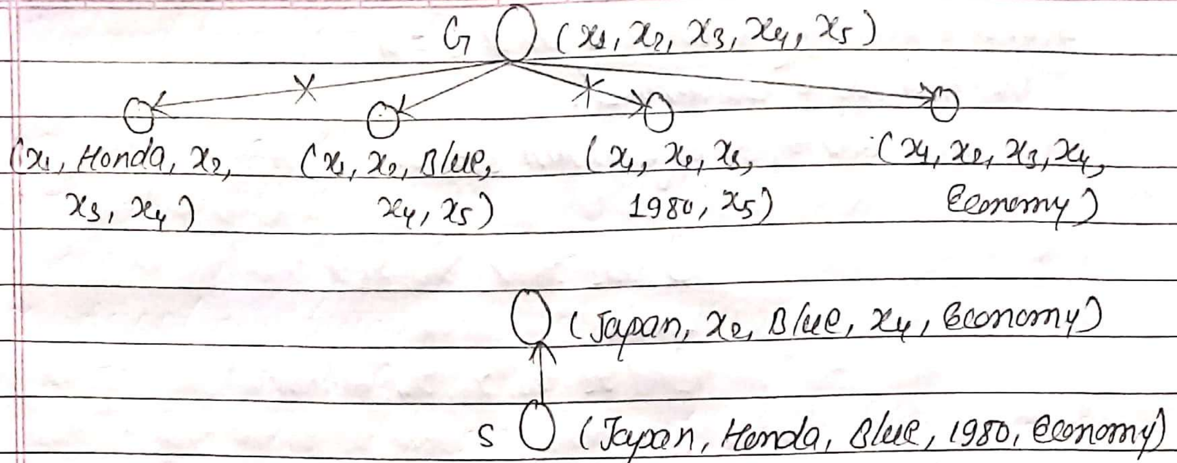
2. -ve Example : (Japan, Toyota, Green, 1970, sports)

Specialize G to exclude -ve example & prune elements in S if it matches -ve examples.

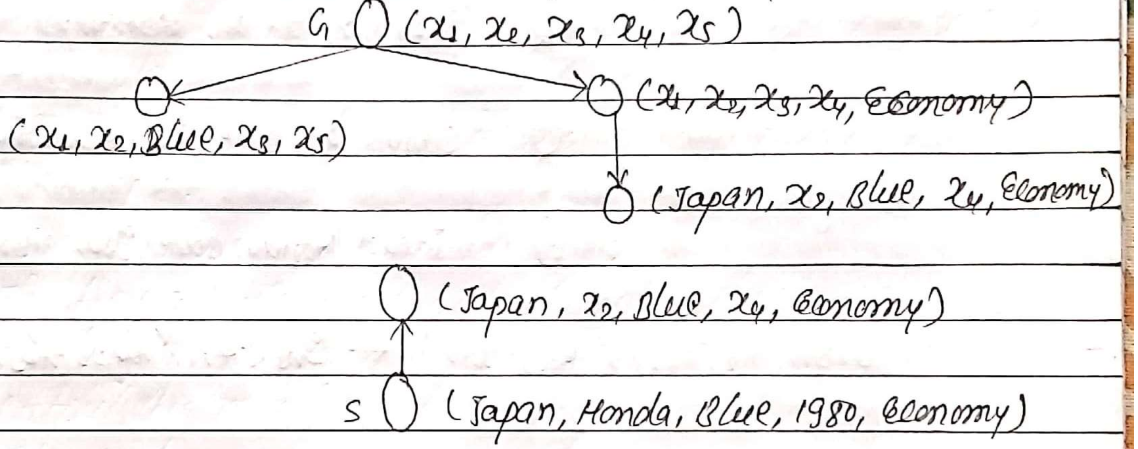
$$\begin{tikzcd} & & G \; (x_1, x_2, x_3, x_4, x_5) \arrow[dll] \arrow[dl] \arrow[d] \arrow[dr] & \\ (x_1, \text{Honda}, x_3, x_4, x_5) & (x_1, x_2, \text{Blue}, x_4, x_5) & (x_1, x_2, x_3, 1980, x_5) & (x_1, x_2, x_3, x_4, \text{Economy}) \\ & & S \; (\text{Japan, Honda, Blue, 1980, Economy}) & \end{tikzcd}$$

3. +ve example : (Japan, Toyota, Blue, 1990, Economy)

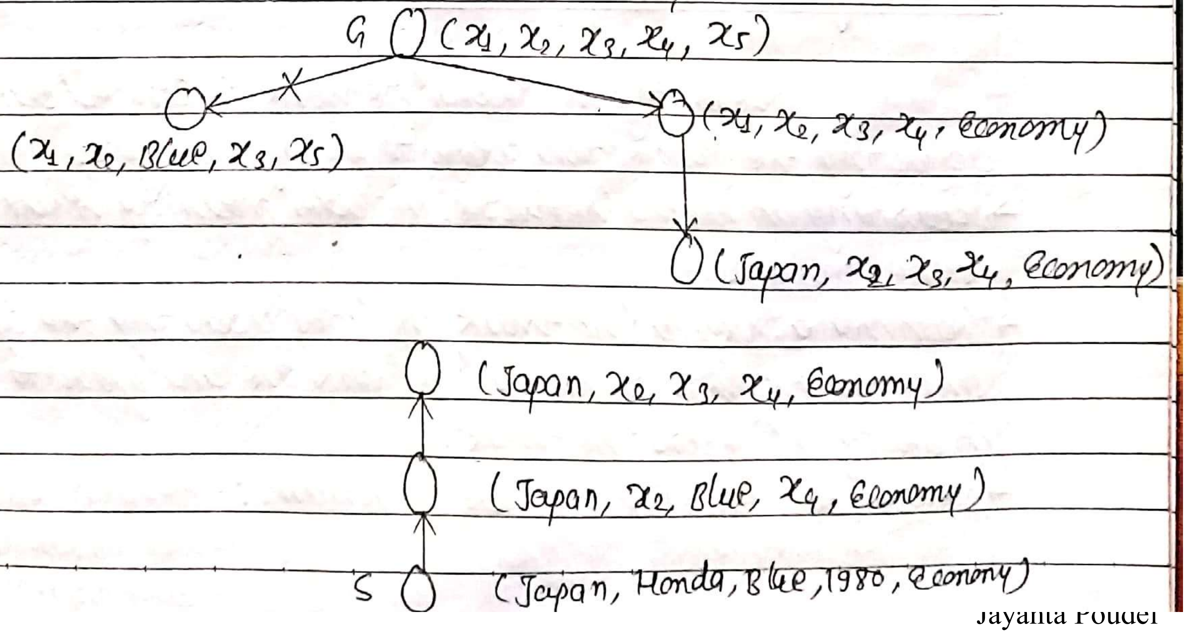
Generalize S to include +ve example and prune away elements in G that fail to include +ve example.


G (x1, x2, x3, x4, x5)
X
X
(x1, Honda, x2, x3, x4)
(x1, x2, Blue, x4, x5)
(x1, x2, x3, 1980, x5)
(x4, x2, x3, x4, Economy)
(Japan, x2, Blue, x4, Economy)
S (Japan, Honda, Blue, 1980, Economy)

4. -ve example: (USA, Chrysler, Red, 1980, Economy)
Specialize G to exclude -ve example (but stay consistent with S)


G (x1, x2, x3, x4, x5)
(x1, x2, Blue, x3, x5)
~~(x1, x2, x3, x4, Economy)~~
(Japan, x2, Blue, x4, Economy)
(Japan, x2, Blue, x4, Economy)
S (Japan, Honda, Blue, 1980, Economy)

5. +ve Example: (Japan, Honda, white, 1980, Economy)
Generalize S to include +ve example.


G (x1, x2, x3, x4, x5)
X
(x1, x2, Blue, x3, x5)
(x1, x2, x3, x4, Economy)
(Japan, x2, x3, x4, Economy)
(Japan, x2, x3, x4, Economy)
(Japan, x2, Blue, x4, Economy)
S (Japan, Honda, Blue, 1980, Economy)

G and S are singleton sets and S = G
Version space converged.
Now more data, so algorithm stops.

$$\begin{tikzcd}
\bigcirc\ (x_1, x_2, x_3, x_4, x_5) \arrow[d] \\
\bigcirc\ (x_1, x_2, x_3, x_4, \text{Economy}) \arrow[d] \\
\bigcirc\ (\text{Japan}, x_2, x_3, x_4, \text{Economy}) \arrow[d, Leftrightarrow] \\
\bigcirc\ (\text{Japan}, x_2, x_3, x_4, \text{Economy}) \\
\bigcirc\ (\text{Japan}, x_2, \text{Blue}, x_4, \text{Economy}) \arrow[u] \\
\bigcirc\ (\text{Japan}, \text{Honda}, \text{Blue}, 1980, \text{Economy}) \arrow[u]
\end{tikzcd}$$

∴ (Japan, x_2, x_3, x_4, Economy) ← Consistent hypothesis.

Learning in Neural Network

Learning in neural network is carried out by adjusting the connection weights among neurons.

→ Learning in neural network includes training ANN to solve some problems

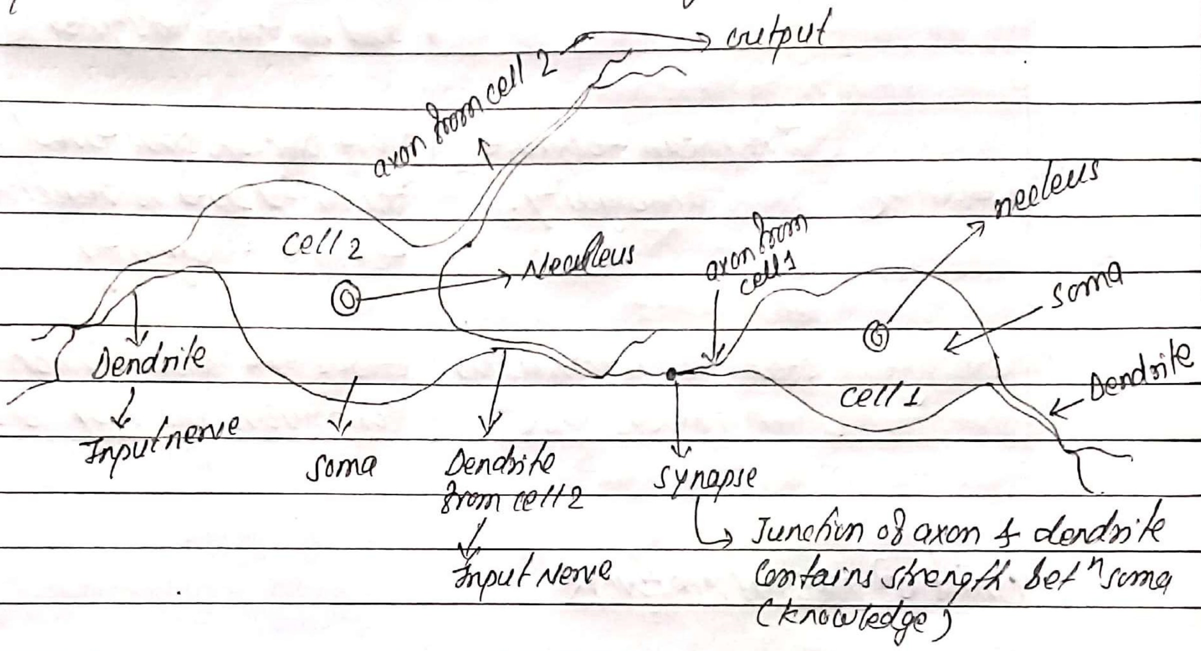
→ Training a neural network includes updating the weights, since the weights in neural network are analogous to synapse in biological neural network.

→ Neural net learning methods / Algorithms : - Hebbian learning
- Perceptron learning
- Backpropagation

Neural Network

Biological Neural Network (BNN)

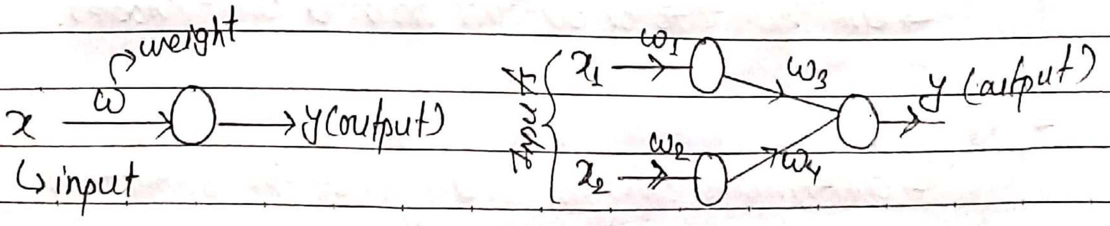
A biological neural network refers to any group of connected biological nerve cells that process information. Biological neuron has dendrites to receive signals, a cell body (soma) to process them, and an axon to send signals out to other neurons.



Artificial Neural Network (ANN)

An artificial neural network is a composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problem.

- It is based on the structure and functions of BNN.
- ANN has a number of input channels, a processing stages and one output.



BNN VS ANN

	BNN	ANN
Processing	Massively parallel slow but superior than ANN	Massively parallel, fast but inferior than BNN
Size	10^{11} neurons to 10^{15} interconnections.	10^2 to 10^4 nodes (mainly depends on the type of application and network designer)
Learning	They can tolerate ambiguity.	very precise, structured and formatted data is required to tolerate ambiguity
storage	stores the information in the synapse	stores the information in continuous memory location

Units of Neural Network

→ Nodes (units):
↳ represents a cell of neural network.

→ Links:
↳ are directed arrows that show propagation of information from one node to another node.

→ Activation:
↳ are inputs to or outputs from a unit (node).

→ Weight:
↳ determines strength and sign of the connection.

→ Activation function:
↳ a function which is used to derive output activation from the input activations to a given node.

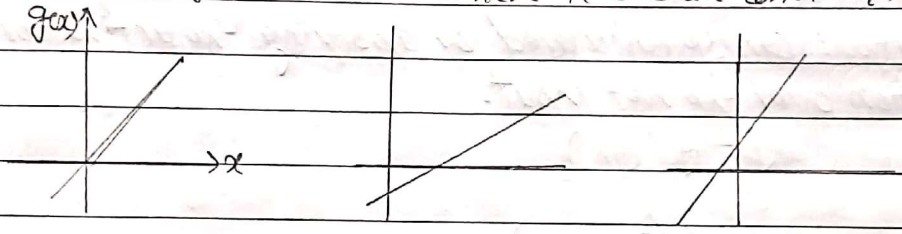
→ Bias weight: → Threshold value
↳ is used to set the threshold for a unit. Unit is activated when the weighted sum of real inputs exceeds the bias weight.

Types of Activation function

1) Linear function:

The output activity is proportional to the total weighted output.

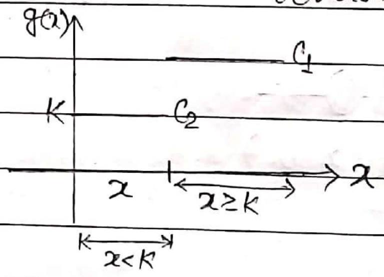
$g(x) = kx + c$ where k & c are constant.



2) Threshold function / step function:

The output are set at one of two levels.

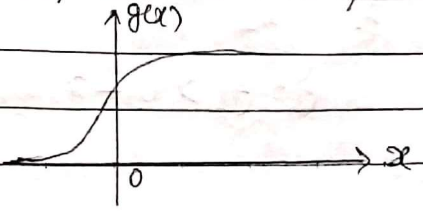
$g(x) = C_1$ if $x \geq k$
$= C_2$ if $x < k$



3) Sigmoid function:

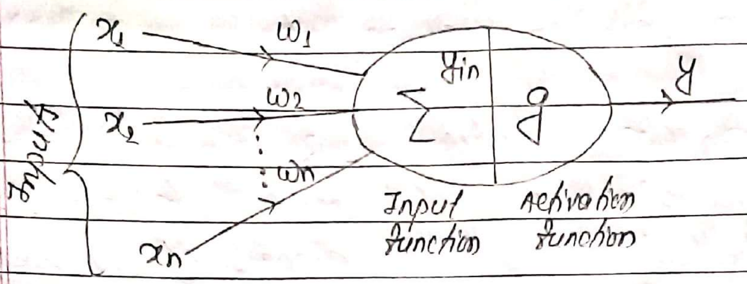
The output varies continuously but not linearly as the input changes.

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



Imp:

Mathematical model of ANN



For the above general model of ANN, the net input can be calculated as follows:

$$y_{in} = x_1 w_1 + x_2 w_2 + \cdots\cdots + x_n w_n$$

i.e.

$$\text{Net input } y_{in} = \sum_{i=1}^{n} x_i w_i$$

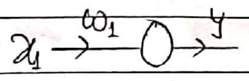
The output can be calculated by applying the activation function over the net input.

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

$$\text{i.e. } y = g\left(\sum_{i=1}^{n} x_i w_i\right)$$

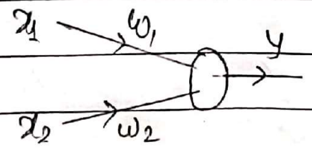
e.g.

①

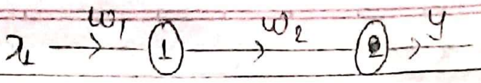


$$y = g(x_1 w_1)$$

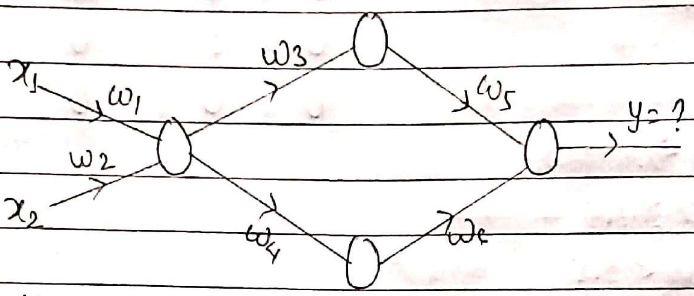
②



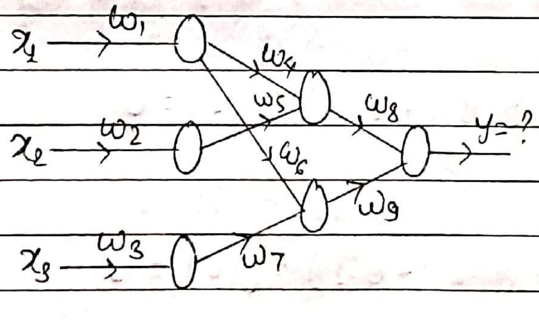
$$y = g(x_1 w_1 + x_2 w_2)$$

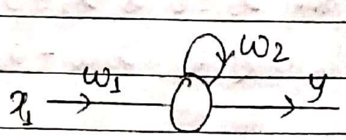
③ 

$y = g(g(x_1 w_1) \cdot w_2)$ or $y_1 = g(x_1 w_1)$
$y = g(y_1 w_2)$

④ 

$y_1 = g(x_1 w_1 + x_2 w_2)$
$y_2 = g(y_1 w_3)$
$y_3 = g(y_1 w_4)$
$y = g(y_2 w_5 + y_3 w_6)$

⑤ 

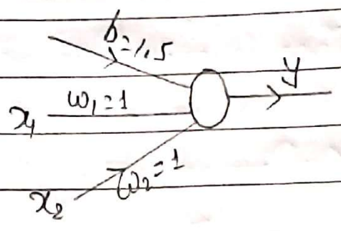
⑥ 

$y_1 = g(x_1 w_1)$
$y_2 = g(x_2 w_2)$
$y_3 = g(x_3 w_3)$
$y_4 = g(y_1 w_4 + y_2 w_5)$
$y = g(y_1 w_6 + y_3 w_7)$
$\therefore y = g(y_4 w_8 + y_5 w_9)$

$y = g(x_1 w_1 + g(x_1 w_1) \cdot w_2)$

Realizing Logic gate by NN

AND



x_1	x_2	x_1 AND x_2
1	1	1
1	0	0
0	1	0
0	0	0

suppose a step function,

$g(x) = 1$ if $x \geq 1.5$
$= 0$ if $x < 1.5$

for $x_1 = 1$ & $x_2 = 1$
$y = g(x_1w_1 + x_2w_2) = g(1\cdot1 + 1\cdot1) = g(2) = 1$

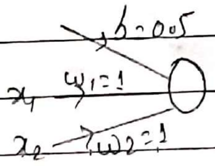
for $x_1 = 1$ & $x_2 = 0$
$y = g(x_1w_1 + x_2w_2) = g(1\cdot1 + 0\cdot1) = g(1) = 0$

for $x_1 = 0$ & $x_2 = 1$
$y = g(x_1w_1 + x_2w_2) = g(0\cdot1 + 1\cdot1) = g(1) = 0$

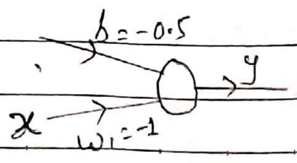
for $x_1 = 0$ & $x_2 = 0$
$y = g(x_1w_1 + x_2w_2) = g(0\cdot1 + 0\cdot1) = g(0) = 0$

Try for OR & NOT

OR



NOT

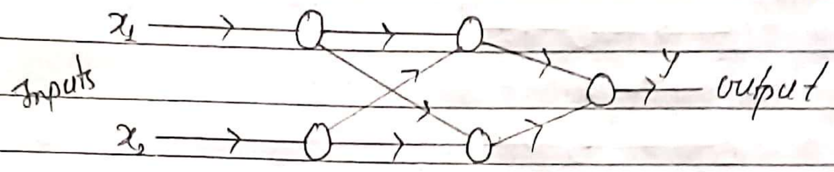


Network structure / Types of ANN

1. Feed-forward networks

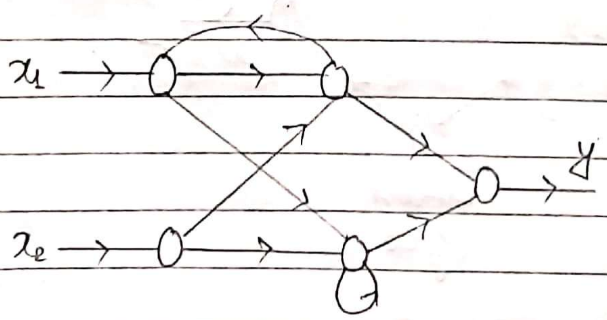
→ feed-forward ANNs allow signals to travel one way only from input to output. There is no feedback (loop). They are extensively used in pattern recognition.

(unidirectional)



2. Feedback networks

→ feedback networks can have signals travelling in both directions by introducing loops in the network.
→ feedback networks are dynamic; their state is changing continuously until they reach an equilibrium point. They remain at the equilibrium point until the input changes and a new equilibrium needs to be found.

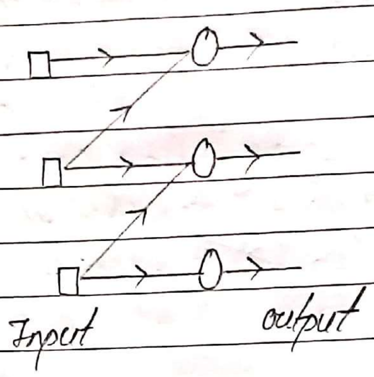


Types of feed forward NN

1) single-layer NN
2) Multilayer NN

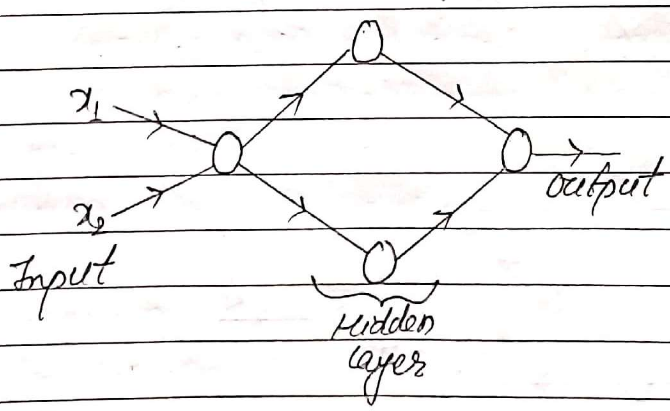
1) Single-layer NN

→ A neural network in which all the inputs connected directly to the outputs is called single layer NN.
→ Each output unit is independent of the others.



2) Multilayer NN

→ The neural network which contains input layer, output layer and some hidden layer also is called multi-layer NN.
→ Layers of the network are normally fully connected.



Imp.

Hebbian learning

Hebbian learning is a general principles that states that the synaptic efficacy between two neurons should increase if the two neurons are simultaneously active and decrease if not.

Hebb's postulate of learning:

" When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic changes take place in one or both cells such that A's efficiency as one of the cells firing B is increased".

Hebb's Algorithm

1. Initialize all weights to 0.
2. Given a training input s with its target output t, set the activation of the input unit : $x_i = s_i$
3. Update the weight as : $w_i(new) = w_i(old) + x_i t$
4. Update the bias weight (if any) : $b(new) = b(old) + t$.

Example.

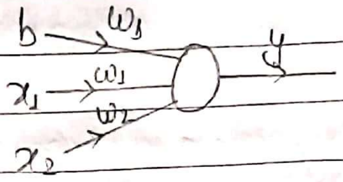
Consider an AND gate problem

Given s =

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

Consider the activation function is step function :

$$g(x) = 0 \quad \text{if } x < b$$
$$= 1 \quad \text{if } x \geq b$$



Initially,

$\omega_1 = 0, \omega_2 = 0$ & $\omega_b = 0$

1st iteration

$S_1 = (1, 1, 1)$ with $t = 1$

$\omega_1(new) = \omega_1(old) + x_1 t = 0 + 1 \cdot 1 = 1$

$\omega_2(new) = \omega_2(old) + x_2 t = 0 + 1 \cdot 1 = 1$

$\omega_b(new) = \omega_b(old) + t = 0 + 1 = 1$

2nd iteration

$S_2 = (1, -1, 1)$ with $t = -1$

$\omega_1(new) = 1 + 1(-1) = 0$

$\omega_2(new) = 1 + (-1)(-1) = 2$

$\omega_b(new) = 1 + (-1) = 0$

3rd iteration

$S_3 = (-1, 1, 1)$ with $t = -1$

$\omega_1(new) = 0 + (-1)(-1) = 1$

$\omega_2(new) = 2 + 1(-1) = 1$

$\omega_b(new) = 0 + (-1) = -1$

4th iteration

$S_4 = (-1, -1, -1)$ with $t = -1$

$\omega_1(new) = 1 + (-1)(-1) = 2$

$\omega_2(new) = 1 + (-1)(-1) = 2$

$\omega_b(new) = -1 + (-1) = -2$

for input $x_1 = 1, x_2 = 1$ & $b = 1$

$$y = g(x_1 w_1 + x_2 w_2 + b \cdot w_b)$$
$$= g(1 \cdot 2 + 1 \cdot 2 + 1 \cdot (-2))$$
$$= g(2)$$
$$= 1$$

for input $x_1 = 1, x_2 = -1$ & $b = 1$

$$y = g(x_1 w_1 + x_2 w_2 + b \cdot w_b)$$
$$= g(1 \cdot 2 + (-1) \cdot 2 + 1 \cdot (-2))$$
$$= g(-2)$$
$$= 0$$

for input $x_1 = -1, x_2 = 1$ & $b = 1$

$$y = g(x_1 w_1 + x_2 w_2 + b \cdot w_b)$$
$$= g((-1) \cdot 2 + 1 \cdot 2 + 1 \cdot (-2))$$
$$= g(-2)$$
$$= 0$$

for input $x_1 = -1, x_2 = -1$ & $b = 1$

$$y = g(x_1 w_1 + x_2 w_2 + b \cdot w_b)$$
$$= g((-1) \cdot 2 + (-1) \cdot 2 + 1 (-2))$$
$$= g(-6)$$
$$= 0.$$

Imp.

Perceptron learning

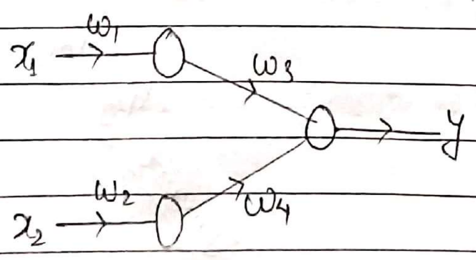
Perceptron is a single artificial neuron that computes the weighted input with the help of the threshold activation function or step function.
- The algorithm for perceptron is based on supervised learning procedure.

Algorithm

1. Initialize weight ω_i and threshold (if any) to random.
2. Present input x_i and desired output $d(t)$.
3. Calculate actual output:
   $y(t) = g\,[\sum x_i \omega_i(t)]$ at any timestamp t.
4. Update weight as
   $\omega_i(t+1) = \omega_i(t) + \alpha\,[d(t) - y(t)]\,x_i(t)$
   where,
   $\omega_i(t+1)$ is new updated weight at time t.
   α is learning rate constant $0 \le \alpha \le 1$
   $d(t)$ is targeted output.
   $y(t)$ is actual output at timestamp t.
   $x_i(t)$ is input x_i at time t.

5. Repeat step 3 & 4 until the desired solution is obtained.

E.g.



$\alpha = 0.5$ ↓ moderate rate

Given, desired output $d(t)$.
Actual output is computed as;

$$y(t) = g(y_1 \omega_3(t) + y_2 \omega_4(t))$$

where,

$$y_1 = g(x_1 \omega_1(t))$$
$$y_2 = g(x_2 \omega_2(t))$$

Now update weight as:

$$\omega_1(t+1) = \omega_1(t) + \alpha [d(t) - y(t)] x_1(t)$$
$$\omega_2(t+1) = \omega_2(t) + \alpha [d(t) - y(t)] x_2(t)$$
$$\omega_3(t+1) = \omega_3(t) + \alpha [d(t) - y(t)] y_1(t)$$
$$\omega_4(t+1) = \omega_4(t) + \alpha [d(t) - y(t)] y_2(t)$$

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

Delta Rule

Delta rule states that for a neuron j with activation function $g(x)$; the delta rule for j's ith weight ω_{ji} is given by;

$$\Delta \omega_{ji} = \alpha (t_j - y_j) g'(h_j) x_i$$

where,

α is learning rate.
$g(x)$ is the neuron's activation function.
t_i is the target output.
h_j is the weighted sum of the neuron's input.
y_j is the actual path.
x_i is the ith input.

It holds that $h_j = \sum x_i \omega_{ji}$ and $y_j = g'(h_j)$

V.Imp

Backpropagation

It is a supervised learning method, and is an implementation of the Delta rule. It requires a teacher that can calculate the desired output for any given input.

→ It is most useful for feed-forward networks.

→ Backpropagation requires that the activation function used by the artificial neurons (or nodes) is differentiable.

Algorithm

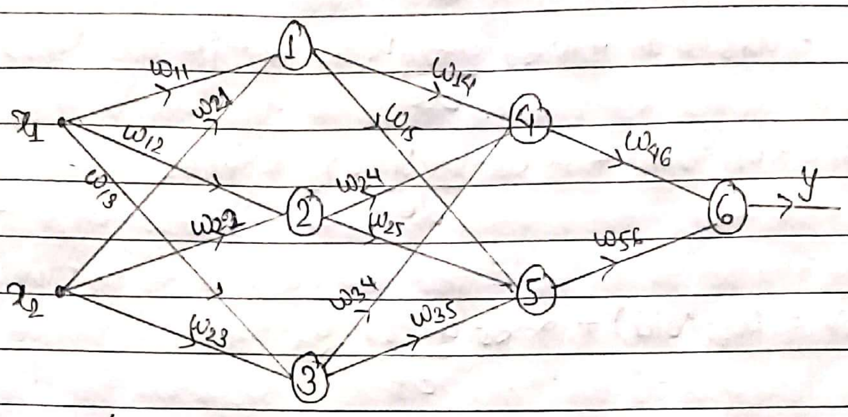
1. Randomly choose the initial weight.
2. Compute output (y) from neural network.
   $y = g(\sum x_i w_i)$ where $g(x) = \frac{1}{1+e^{-x}}$ is sigmoid function.
3. Compute error $\delta = t - y$; where t is targeted output and y is actual output.
4. Propagate this δ error back to layer of neural network 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.to e.
   α is learning rate. $0 \le \alpha \le 1$
   x_i is input to node.

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

E.g.

1)



~~Given targe~~

$y = g(y_4 w_{46} + y_5 w_{56})$

where,

$y_4 = g(y_1 w_{14} + y_2 w_{24} + y_3 w_{34})$

$y_5 = g(y_1 w_{15} + y_2 w_{25} + y_3 w_{35})$

where,

$y_1 = g(x_1 w_{11} + x_2 w_{21})$

$y_2 = g(x_1 w_{12} + x_2 w_{22})$

$y_3 = g(x_1 w_{13} + x_2 w_{23})$

Given target t.

Compute

$\delta = t - y$

Now backpropagate error as;

$\delta_4 = w_{46} \times \delta$

$\delta_5 = w_{56} \cdot \delta$

$\delta_1 = w_{14}\, \delta_4 + w_{15} \cdot \delta_5$

$\delta_2 = w_{24} \cdot \delta_4 + w_{25} \cdot \delta_5$

$\delta_3 = w_{34} \cdot \delta_4 + w_{35} \cdot \delta_5$

Now update weight as;

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

$$W_{12}(new) = W_{12}(old) + \alpha \delta_2 \frac{d(y_2)}{de} \cdot x_1$$

$$W_{13}(new) = W_{13}(old) + \alpha \delta_3 \frac{d(y_3)}{de} \cdot x_1$$

$$W_{21}(new) = W_{21}(old) + \alpha \delta_1 \frac{d(y_1)}{de} \cdot x_2$$

⋮

$$W_{25}(new) = W_{25}(old) + \alpha \cdot \delta_5 \frac{d(y_5)}{de} \cdot y_2$$

$$W_{34}(new) = W_{34}(old) + \alpha \cdot \delta_4 \cdot \frac{d(y_4)}{de} \cdot y_3$$

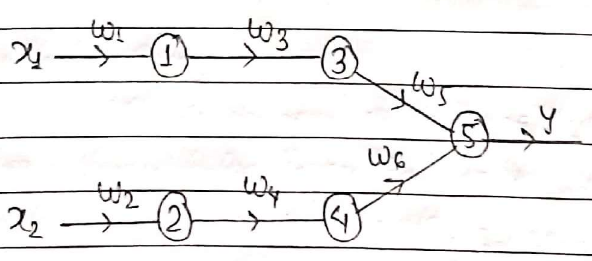
⋮

$$W_{46}(new) = W_{46}(old) + \alpha \cdot \delta \cdot \frac{dy}{de} \cdot y_4$$

$$W_{56}(new) = W_{56}(old) + \alpha \cdot \delta \cdot \frac{dy}{de} \cdot y_5$$

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

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 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;

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

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

$$w_3(new) = w_3(old) + \alpha . \delta_3 . \frac{dy_3}{de} . y_1$$

$$w_4(new) = w_4(old) + \alpha . \delta_4 . \frac{dy_4}{de} . y_2$$

$$w_1(new) = w_1(old) + \alpha . \delta_1 . \frac{dy_1}{de} . x_1$$

$$w_2(new) = w_2(old) + \alpha . \delta_2 . \frac{dy_2}{de} . x_2$$

After updating weights, we recompute y. Then we compute $\delta = t - y$.

If the value of y is converges to t then the value is fixed up otherwise we will iterate again.