Identifying Optimal Path

→ Optimal path → lowest cost or Traversing the shortest destan
→ Method for identifying optimal path → British Museum procedure
→ Techniques of identifying optimal path
* A*
* Uniform cost search
* Greedy search

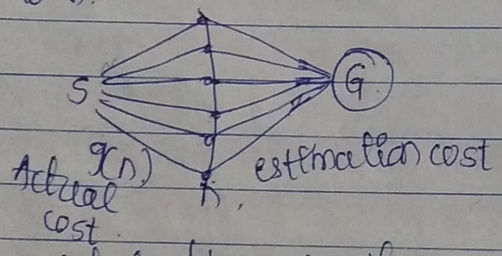
A* Algorithms

A* algorithms operates in the same manner as best first search. but it uses the following function evaluate nodes

$f(n) = g(n) + h(n)$

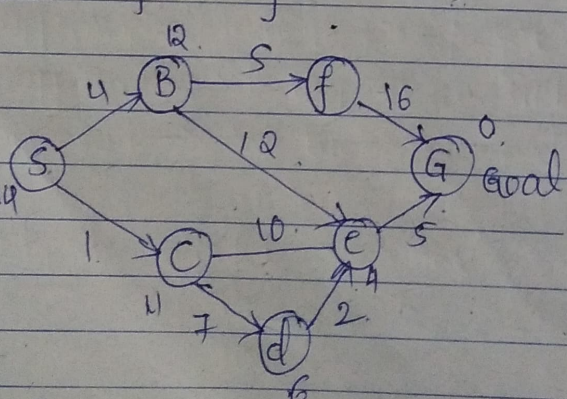
Actual cost from start node to n

Estimation cost from n to goal node.



* A* is Admissible algorithm

It guarantees that it will give the optimal solution starts with $f(n) = g(n) + h(n)$



$f(s) = 0 + 14 = 14$

S → B. S → C.
4 → 12. 3 + 11. Less cost path
16 = 14 S → C is (14)

SC → e = 3+10+4 = 17 less cost
SC → d 3+7+6 = 16 ie 16

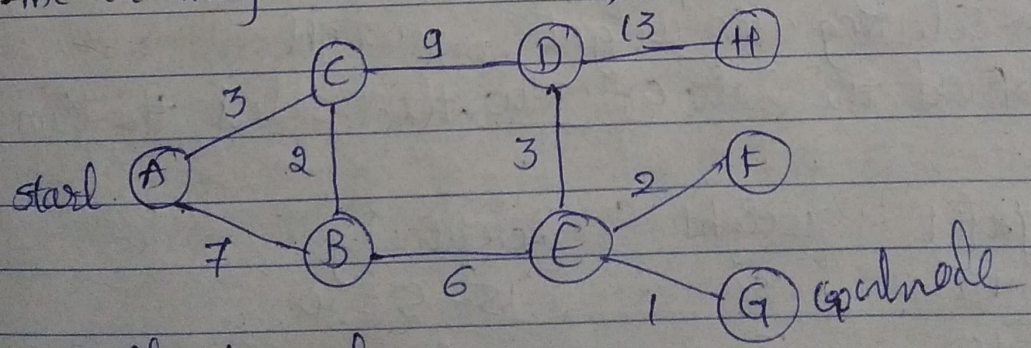
scd → e = e = 3+7+2+4 = 16
scde → G = 3+7+2+5+0
12+5+0 = 17 optimal cost to reach goal

SB → f

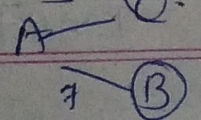
Uniform cost search. {Backtracking is there}
→ Used for traversing
It is used for traversing weighted tree of graph, it is used when different cost available in each edge. The goal of this is to find a path to the goal node which has lowest cumulative cost.
* It is used in where optimal cost is in demand..
* It is implemented by priority queue, it gives maximum priority to lowest cumulative

Advantages → Every state the path with the least cost is chosen {optimal solution}

Disadvantage → Does not care about no of steps involved in searching & only concerned about path cost, sometime it may struct in infinite.



Path to goal

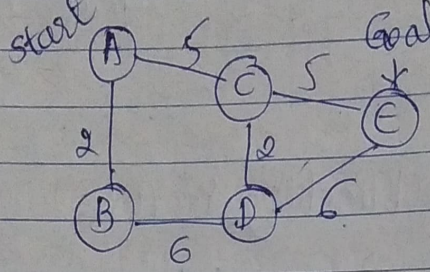


A → 3 C → 2 B → 6 E → 1 G = 12

$A \xrightarrow{3} C \xrightarrow{8} B \xrightarrow{6} E \xrightarrow{1} G \xrightarrow{2} \cancel{C} \xrightarrow{3} D \xrightarrow{12} H$

Greedy Search

→ It selects the path that has lowest heuristic value or estimated ~~value~~ distance to goal. heuristic value.



Goal
A → 10
B → 8
D → 3
C → 5 C < B
E → 0. E < D

$A \rightarrow B \rightarrow E$

$X = \sum_{i=1}^{n} w_i x_i$

$Y = \begin{cases} +1 & \text{for } x > t \\ 0 & \text{for } x \leq t \end{cases}$

Knapsack Problem

Fractional knapsack problem.

object	1	2	3	4	5	6	7	
profit(P)	5	10	15	7	8	9	4	w = 15
weight(w)	1	3	5	4	1	3	2	n = 7
	5	3.3	3	1.75	8	3	2	max P/w

I approach - Selecting max profit
II approach → Selecting min weight
III approach → Find the ratio of P/w, then select the item

I)

object	profit	weight	Remaining
3	15	5	15 - 5 = 10
2	10	3	7
6	9	3	4
5	8	1	3
4	$\frac{7 \times 3}{4} = 5.25$	3	3 - 3 = 0

I = 47.25

17/16 33/11

Approach II

obj	profit	weight	Remaining
1	5	1	15-1=14
5	8	1	13
7	4	2	11
2	10	3	8
6	9	3	5
4	7	4	1
3	$\frac{15}{5} \times 3$	1	1-1=0

II = 46

III

object	profit	weight	Remaining weight
5	8	1	15-1=14
1	5	1	13
2	10	3	10
3	15	5	5
6	9	3	2
7	4	2	0
	=51		

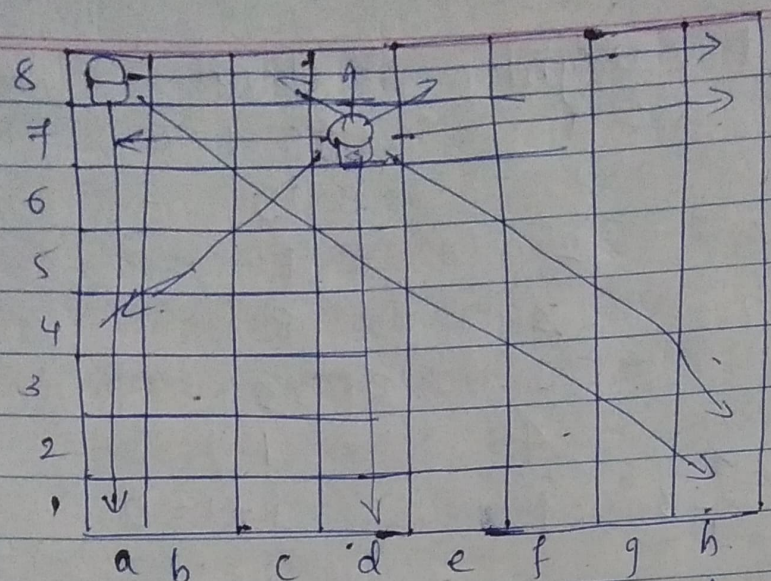
P/W = 5 3.33 3.175 8 3 2

profit = 5!

Constraint Satisfaction Search

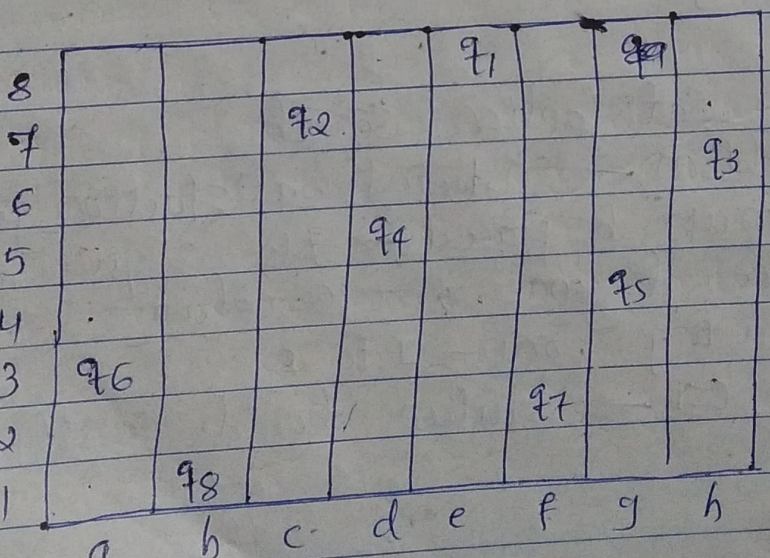
→ 8 queens problem → Constraint satisfaction problem
→ 8 queens must be placed → No 2 queens are on same diagonal row or columns
→ columns → a to g, row → 1 to 8
→ This is called CSP → solution should satisfy constraints

Let x



Goal node in the tree should satisfy the constraints that no two ~~Dode~~ queens are same node & diagonal. Analyze the every possible configuration. until one was found that matched the constraint
8 ~~queens~~ problems. would be used to depth first search on a search tree.
The first branch from the root node would represent the 1st choice of square for a ~~Queens~~, the next branch represent choice where to place the second ~~queen~~

1st level have branching factor of 64.
next level have lower branching factor

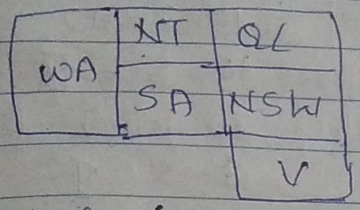


Forward checking

→ Learning this FCP by map coloring problem.

Domains {R, G, B}

variables {WA, NT, SA, QL, NSW, V}



Implicit: {(WA ≠ NT), (WA ≠ SA), (NT ≠ SA), NT ≠ QL, CSA ≠ QL), (QL ≠ NSW), (SA ≠ NSW), (NSW ≠ V)}

Initial domain	WA RGB	NT RGB	SA RGB	QL RGB	NSW RGB	V
WA = B	B	RG	RG	RGB	RGB	RGB
NT = G	B	G	R	RB	BGB	RGB
SA = R	B	G	R	B	GB	GB
QL = B	B	G	R	B	G	GB
NSW = G	B	G	R	B	G	B
V	B	G	R	B	G	B

Forward checking where assigning a variable that remove the conflicting values for all the connected variables with

16/05/2022

object	1	2	3	4	5	6	7
profit(P)	6	12	16	8	10	5	1
weight(w)	1	2	5	4	2	3	1

W = 18

object	Profit	weight	P/w	R-w
1	6	1	6.1	17
2	12	2	6.2	17-2=15
3	16	5	3.2	15
4	8	4	2	
5	10	2	5 3	
6	5	3	2 6.6	
7	1	1	1	

object	profit	weight	Remaining weight
1	6	1	18 - 1 = 17
2	12	2	17 - 2 = 15
3 5	10	2	15 - 2 = 13
4 3	16	5	13 - 5 = 8
5 4	8	4	8 - 4 = 4
6	5	3	4 - 3 = 1
7	1	1	1 - 1 = 0

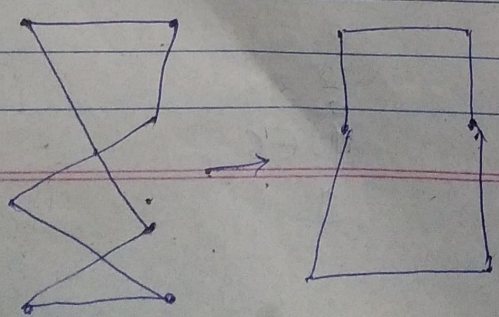
profit = 58

Local Search & metaheuristics

→ Starting from some initial configuration (randomly)
→ Making changes to the configuration until state is reached
→ This is also called metaheuristics with no better state
→ Problem of allocating teachers to classrooms randomly & make some changes.

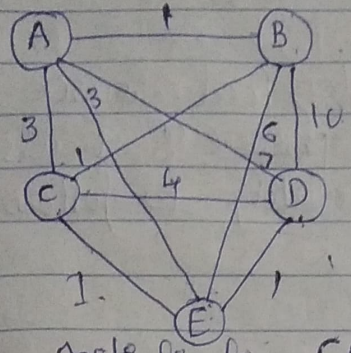
Exchanging Heuristics

EH moves from one state to another by exchanging one or more variables by giving them different value
'k' exchange heuristics used to solve the travelling salesman problem. If we use two exchange we remove two edges from the two and substitute for two other edges.
If this produces a valid two shorter than previous one then we continue the same or go back to the previous two.



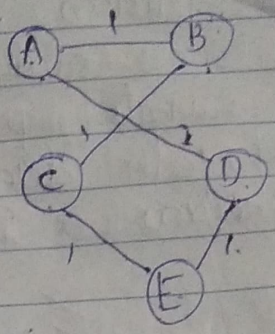
Tabu Search

Tabu Search for TSP



Random Initialization (ACEDBA)
total cost = 16

II



optimal soln

A DECBA Total cost = 5

III A → E → C → D → B → A
total cost = 19

23/05/2022

Ant Colony optimization

→ Ant deposit pheromon on ground to mark some path to be followed by other members.

→ eg: sugar candy.

→ Ants navigate from nest to food source.

→ shortest path discovered via pheromon trails.

→ more pheromon → increases probability to follow the path

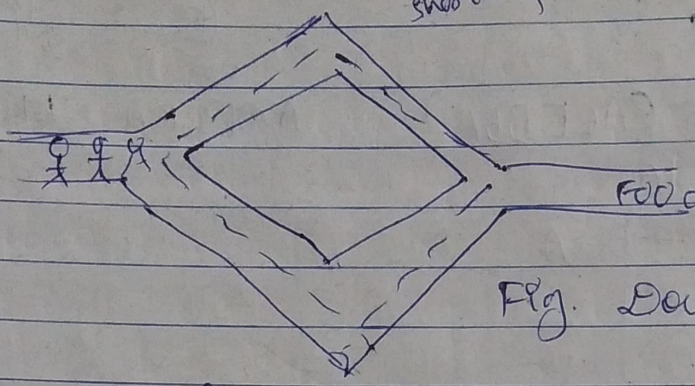


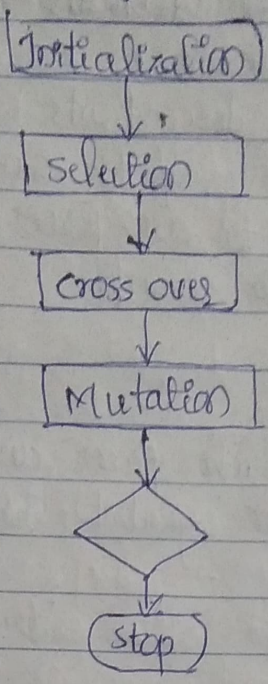
Fig. Double bridge

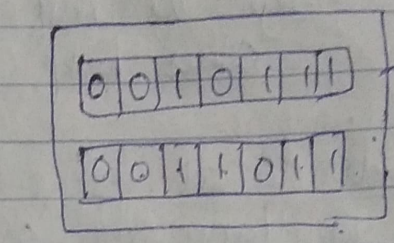
* simulated Annealing

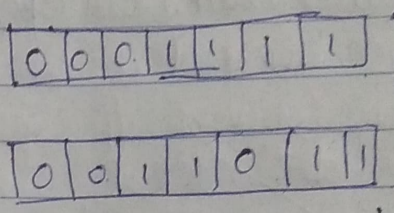
uses concept

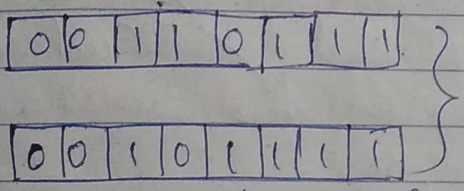
Genetic Algorithm

→ Initiale









mutation → making random changes to particular genes.

A population of possible the solution is generated & fitness level of value for each chromosome is determined.

This fitness is used to determine the given chromosome will survive to the next generation.

Reproduction is done by applying cross over to two chromosomes, features of each chrome are combined together & mutation is applied for making random changes to particular genes.

Parallel Search

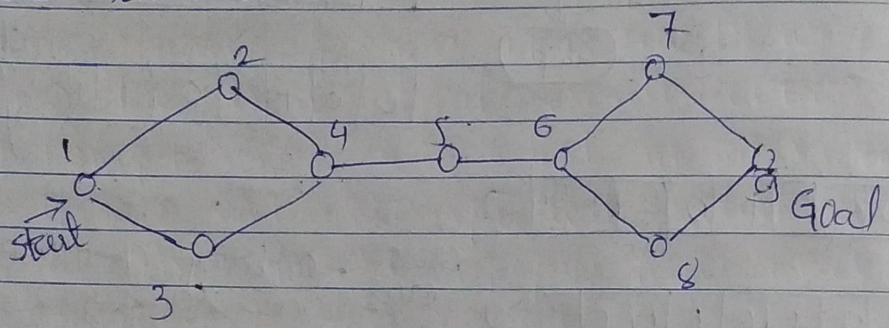
→ Task → Broken into no. of subtasks → can seen in parallel on separate processor.

→ 2 processors → Divide the descendants of root node in half → ~~Assign~~ one half to second.

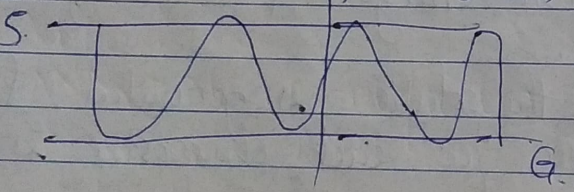
Assign one half to one processor other half to second

Bidirectional Search

Two simultaneously search from an initial node to goal & backward from goal to initial stopping when two meet.



If uses breadth first search



Nondeterministic Search

→ Running a nondeterministic search → new paths are added to queue at random positions.

```
Function random()
{
    queue = [];
    start = root_node;
    while(true)
    {
        if is_goal(state)
            then return Success.
```

```
else
add_randomly_to_queue(successors(state));
    if queue == []
        then report failure
    state = queue[0];
    remove_first_item_from(queue);
  }
}
```

Non chronological Backtracking

→ It is alternative to chronological backtracking

→ chronogical Backtracking

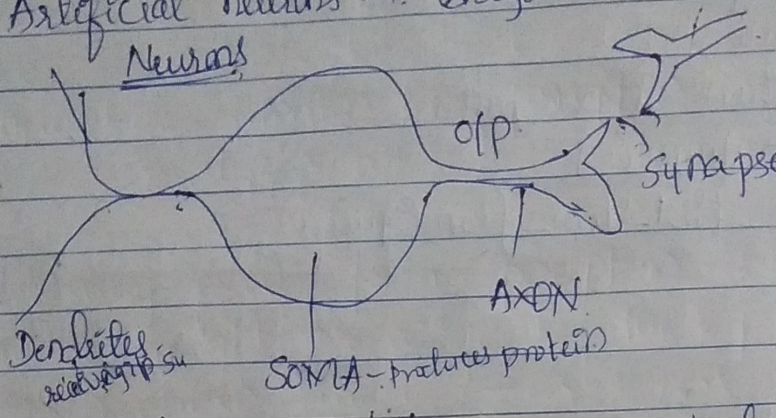
When dead end in a tree is found move back up the search tree to the last point in a tree where decision has to be made. and choose the next option at this junction.

It is useful in solving constraints satisfaction problem where backtracking can be applied by going back to the previous choice that caused contradiction to fail.

Unit-3

Neural Networks

→ Biological neurons → human brain
Artificial neurons → artificial neural networks



→ Human brain – Ten billion neurons → Each connected to other neurons – synapses

→ Neuron receives o/p from other neurons using dendrites when this input signal exceeds certain threshold the neuron fires, a chemical reaction occurs it is known as action potential which is send as to the Axon toward synapse that connect the neuron to dendrite of other neurons.

→ Artificial neurons
→ ANN → No. of artificial neurons
→ Neurons in ANN have fewer connections than biological neurons.
→ Each neuron in neural network receives no. of inputs
→ Activation function applied to these input values → Results in activation level of neuron

a) Step function
b) sigmoid function
c) linear function.

Behaviour of neuron

$$x = \sum_{i=1} w_i x_i$$

$$Y = \begin{cases} +1 & \text{for } x > t \\ 0 & \text{for } x \leq t \end{cases}$$

— Activation level

w_i - weight

Most commonly used function is the i/p to the neurons are summed (multiplied by a weig this sum is compared with threshold 't'. If sum is greater than threshold then the neuron fires & has activation level of +1. Otherwise it is inactive & has activation level of 0.

x - weighted sum of n input to neuron
x_1 to x_n → Each i/p multiplied by corresponding weight w_n

$w_1 = 0.8$ $w_2 = 0.4$ t
the i/p x_1 & x_2 neuron $x_1 = 0.7$ & $x_2 = 0.9$
Calculate the sum weight

$x = w_1x_1 + w_2x_2$
$0.8 \times 0.7 + 0.4 \times 0.9$
$0.56 + 0.36$
$x = 0.92$

Perceptron

→ Perceptron → can have many no of inputs
→ can be used for image classification or recognition tasks
→ Perceptron uses step functions returns +1 if weighted sum of input(x) > threshold(t) & 0 if x less than or equal to t.

$$x = \sum_{i=1} w_i x_i$$

$$y = \begin{cases} +1 & \text{if } x > t \\ 0 & \text{if } x \leq t \end{cases}$$

step(x):

$$step(x) = \begin{cases} +1 & \text{for } x > t \\ 0 & \text{for } x \leq t \end{cases}$$

activation function for perceptron.

$$Y = step\left(\sum_{i=0}^{n} w_i x_i\right)$$

Learning process for a perceptron is as follows random weights are assign to the inputs. Weights will be chosen between -0.5 & $+0.5$.
An item of training data is presented to perceptron & its output classified. If the o/p is incorrect the weights are adjusted.

If the perceptron incorrectly classifies a +ve piece of training data is -ve then the weights need to modify to increase the o/p from set of i/p.

$$w_i \leftarrow w_i + (a \times x_i \times e)$$

where is e is error.
a - Learning rate

epoch

Logical OR function for 2 inputs, threshold ($t=0$) & learning rate of 0.2

Epoch	x_1	x_2	Expected Y	Actual Y	Error	w_1	w_2
1	0	0	0	0	0	-0.2	0.4
1	0	1	1	1	0	-0.2	0.4
1	1	0	1	0	1	0	0.4
1	1	1	1	1	0	0	0.4
2	0	0	0	0	0	0	0.4
2	0	1	1	1	0	0	0.4
2	1	0	1	0	1	0.2	0.4
2	1	1	1	1	0	0.2	0.4
3	0	0	0	0	0	0.2	0.4
3	0	1	1	1	0	0.2	0.4
3	1	0	1	1	0	0.2	0.4
3	1	1	1	1	0	0.2	0.4

$w_1 = 0.2$

$w_2 = 0.4$

Training data $x_1 = 0$ $x_2 = 0$.

our expected output $x_1 \vee x_2 = 0$.

apply our formula

$Y = \text{step}\left(\sum_{i=0}^{n} w_i x_i\right)$

$= \text{step}(0 \cdot x_0.2) + (0 \times 0.4)$

$= 0$.

o/p Y is expected error @ is 0, so weights don't change.

$x_1 = 0$ $x_2 = 1$ $w_1 = -0.2$

$0x - 0.2 + 1 \times 0.4$.

$= 0.4$

$\text{step}(0.4) = 1 > 0$

For $x_1 = 1$ & $x_2 = 0$.

$Y = \text{step}(1 \times 0) + (0 \times 0.4)$

$= \text{step}(0)$

$= 0$

This is incorrect value; weights are adjusted

$w_i \leftarrow w_i + (a \times x_i \times e)$

$w_1 = 0.0 + (0.2 \times 1 \times 1)$

$= 0.2$ a x_0 e.

$w_2 = 0.4 + (0.2 \times 0 \times 1)$

$= 0.4$

$x_1 = 1$ $x_2 = 1$

$0 + 0.4 \times 1$

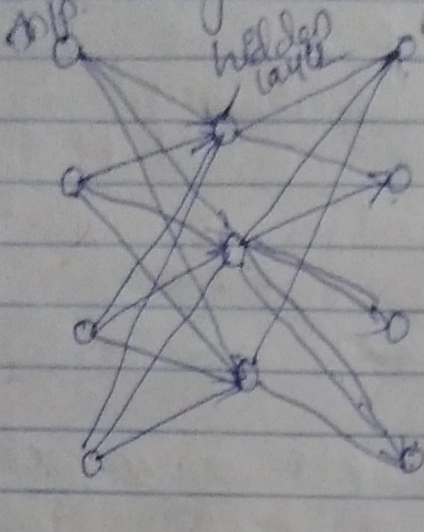
$0 + 0.4$

$0.4 = 1$.

22/05/2020

Multilayer Neural Networks

→ Neural network consist of no of neurons connected together arranged in layers



Three layer feed forward neural network

Back propagation

Each neuron has weights associated with its input & greater numbers of weights to be adjusted when an error is made with piece of training data. So how to assign credit to various weights used in backed propagation

Multilayer back propagation.

uses sigmoid function.

It is defined as $\sigma(x) = \frac{1}{1+e^{-x}}$

Recurrent Networks

→ Feed forward → Data passes from i/p to o/p & not vice versa

→ It does not have memory

→ Recurrent networks have connections that go backward from o/p node to i/p nodes.

→ Has memory

→ Used in stock market, weather prediction.

Unsupervised learning networks

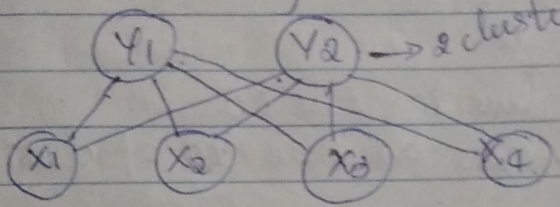
- Kohen maps
- Called as self organising feature map.
- Trained using unsupervised learning.
- Used for clustering data.

Algorithm

Step 0: Initialize weights w_{ij}, random values may be assign, initialize the learning rate

Step 1: Complete the square of Euclidean distance for each $j = 1$ to m

$$D_j = \sum_{i=1}^{n} \sum_{j=1}^{m} (x_i - w_{ij})^2$$



→ 2 clusters

m - no of clusters

n - no of i/p vectors

m = 2

n =

Step 2: Find the winning index j so that D_j is minimum.

Step 3: For all units j and for all i ~~terbu~~ calculate new weights

$$w_{ij}^{new} = w_{ij}^{old} + \alpha (x_i - w_{ij})$$

new — old — learning — i/p vector — old

Step 4: Update the learning rate α using the formula

$\alpha(t+1)$ — t - iteration

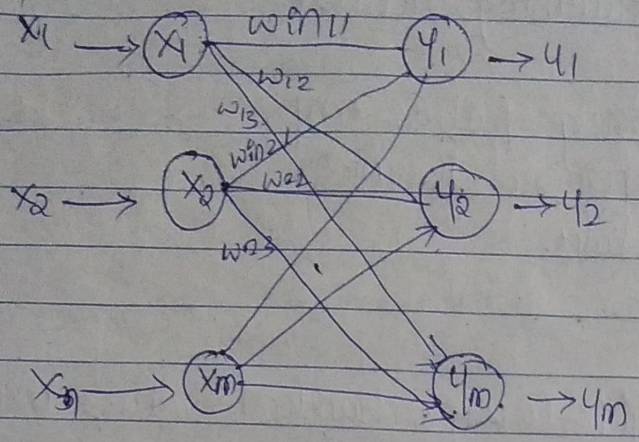
Step 5: After number of epochs test for stopping condition of the network.

$w_{01} = 0.1239$

$w_{02} = 0.1234$

$w_{03} = 0.1233$

$w_{04} = 0.1235$



Construct ~~(EASE)~~ on to clusters for 4 given vectors & no of clusters are formed at 2.

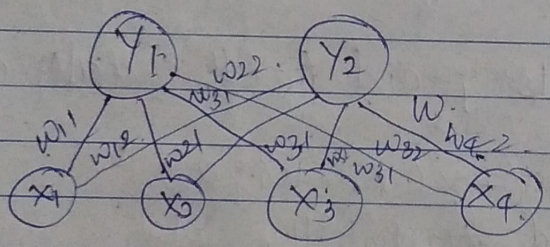
x ∃ [0 0 1 1]
2) [1 0 0 0]
3) [0 1 1 0]
4) [0 0 0 1]

Assume initialize learning rate 0.5

Sol: No of i/p vectors are n=4 no of clusters n=2

i = 1 to 4.

j = 1 to 2.



Here we have 4 values to 2 clusters Y_1 & Y_2 and we calculate distance. whichever is minimum for that update the weights & repeat the process.

$$w_{ij} = \begin{bmatrix} 0.2 & 0.9 \\ 0.4 & 0.7 \\ 0.6 & 0.5 \\ 0.8 & 0.3 \end{bmatrix}$$

w_{i1} weights going for cluster

w_{i2} weight.

[0.0 0.8 0.2 0.2 0.65 0.0

First i/p $x = [0\ 0\ 1\ 1]$

$D(j) = \sum_{i=1}^{n} \sum_{j=1}^{m} (x_i - w_{ij})^2$

$D(1) = (x_1 - w_{11})^2 + (x_2 - w_{21})^2 + (x_3 - w_{31})^2 + (x_4 - w_{41})^2$

$= (0 - 0.2)^2 + (0 - 0.4)^2 + (1 - 0.6)^2 + (1 - 0.8)^2$

$= 0.4 + 0.16 +$

0.4

$D_2 = (x_1 - w_{12})^2 + (x_2 - w_{22})^2 + (x_3 - w_{32})^2 + (x_4 - w_{42})^2$

$= 0 - 0.9)^2 + (0 - 0.7)^2 + (1 - 0.5)^2 + (1 - 0.3)^2$

$= 2.04$

$D_1 < D_2$ winning cluster is $j = 1$ i.e. $Y = 1$

Update weights of winning cluster

$j = 1,\ Y = 1$

$w_{ij} = w_{ij} + \alpha (x_i - w_{ij})$

$w_{i1} = w_{i1} + \alpha (x_i - w_{i1})$

$w_{11} = w_{11} + \alpha (x_1 - w_{11})$

$= 0.2 + 0.5 (0 - 0.2)$

$= 0.1 \quad (x_1 \rightarrow Y_1$

$w_{21} = w_{21} + \alpha (x_2 - w_{21})$

$= 0.4 + 0.5 (0 - 0.4)$

$= 0.2$

$w_{31} = w_{31} + \alpha (x_3 - w_{31})$

$= 0.6 + 0.5 (1 - 0.6)$

$= 0.8$

$w_{41} = w_{41} + \alpha (x_4 - w_{41})$

$= 0.8 + 0.5 (1 - 0.8)$

$= 0.9$

So now updated weight

$w_{ij} = \begin{bmatrix} 0.1 & 0.9 \\ 0.2 & 0.7 \\ 0.5 & 0.5 \\ 0.9 & 0.3 \end{bmatrix}$

Second i/p vect [1 0 0 0]

$D(1) = (x_1 - w_{11})^2 + (x_2 - w_{21})^2 + (x_3 - w_{31})^2 + (x_4 - w$

$= (1 - 0.1)^2 + (0 - 0.2)^2 + (0 - 0.8)^2 + (0 -$

$= 2.3$

$D(2) = (x_1 - w_{12})^2 + (x_2 - w_{22})^2 + (x_3 - w_{32})^2 + (x_4 - w$

$= (1 - 0.9) + (0 - 0.7)^2 + (0 - 0.5)^2 + (0 - 0.$

$= 0.84$

$D(2) < D(1)$

$j = 2 \quad y = 2$

update the weights of winning cluster

$w_{ij}^o = w_{ij}^o + \alpha (x_i - w_{ij}^o)$

$w_{12}^o = w_{12}^o + \alpha (x_1^o - w_{12}^o)$

$w_{12} = w_{12} + 0.5 (x_1 - w_{12})$

$= 0.9 + 0.5 (1 - 0.9)$

$= 0.95$

$w_{22} = w_{22} + 0.5 (x_2 - w_{22})$

$= 0.7 + 0.5 (0 - 0.7)$

$= 0.35$

$w_{32} = w_{32} + 0.5 (x_3 - w_{32})$

$= 0.5 + 0.5 (0 - 0.5)$

$= 0.25$

$w_{42} = w_{42} + 0.5 (x_4 - w_{42})$

$= 0.3 + 0.5 (0 - 0.3)$

$= 0.15$

so now updated weights are

$w_{ij} = \begin{bmatrix} 0.1 & 0.95 \\ 0.2 & 0.35 \\ 0.8 & 0.25 \\ 0.9 & 0.15 \end{bmatrix}$

3rd i/p $(x = 0.0\,0\,0\,1)$

$D1 = (x_1 - w_{11})^2 + (x_2 - w_{21})^2 + (x_3 - w_{31})^2 + (x_4 - w$

$= (0 - 0.1)^2 + (0 - 0.2)^2 + (0.8 - 0.8)^2 + (1 - 0.9)$

$D_1 = 1.5$

$D_2 = (x_1 - w_{12})^2 + (x_2 - w_{22})^2 + (x_3 - w_{32})^2 + (x_4 - w_{42})^2$

$= 1.91$

$D(1) < D(2)$ $\quad j=1 \quad 4=1$

update the weights of winning cluster

$w_{11} = 0.05$ D(1)

$w_{ij}^{new} = w_{ij}^{old} + \alpha (x_i - w_{ij}^{old})$

$w_{i1}^{new} = w_{i1}^{old} + \alpha (x_i - w_{i1}^{old})$

$w_{11} = w_{11} + 0.5 (x_1 - w_{11})$

$= 0.1 + 0.5 (0 - 0.1)$

$= 0.05$

w_{21}

$= 0.6$

w_{31}

$= 0.9$

$w_{41} = 0.45$

$$w_{ij} = \begin{bmatrix} 0.05 & 0.95 \\ 0.6 & 0.35 \\ 0.9 & 0.25 \\ 0.45 & 0.15 \end{bmatrix}$$

4th i/p 0 0 0 1

$D(1) =$

$= 1.475$

$D(2) = 2.065$

$D(2) = 1.81$

$D(1) < D(2)$ $\quad j=1, \quad 4=1$

update weights.

$w_{11} = 0.025$ $w_{21} = 0.3$ $w_{31} = 0.45$ $w_{41} = 0.$

1 st

$$\alpha(t+1) = 0.5 \times \alpha(t)$$
$$\alpha(0+1) = 0.5 \times \alpha(0.5)$$
$$= 0.25$$

Hopfield Networks (feedback)

→ Form of Recurrent feedback network.

→ Uses sign activation function

$$\text{sign}(x) = \begin{cases} +1 & \text{for } x > 0 \\ -1 & \text{for } x < 0 \end{cases}$$

weight of network represented by matrix, W

$$W = \sum_{i=1}^{N} x_i x_i^t - NI$$

$$x_1 = \begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix} \quad x_1^t \; [1 \; -1 \; 1] \quad I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Eg: 5 nodes & 3 training input

$$x_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \quad x_2 = \begin{bmatrix} -1 \\ -1 \\ -1 \\ -1 \\ -1 \end{bmatrix} \quad x_3 = \begin{bmatrix} 1 \\ -1 \\ 1 \\ 1 \\ -1 \end{bmatrix}$$

x_1, x_2, x_3 → are traing i/p

$$W = \sum_{i=3}^{3} x_i x_i^t - 3I$$

$$= x_1 x^t - 3I$$

$$\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} [1 \; 1 \; 1 \; 1 \; 1] - 3 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$