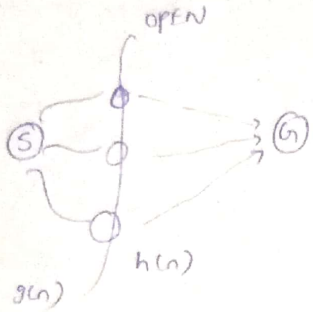
A Star Search:- (If finds optimal path, If heuristic is admissible)

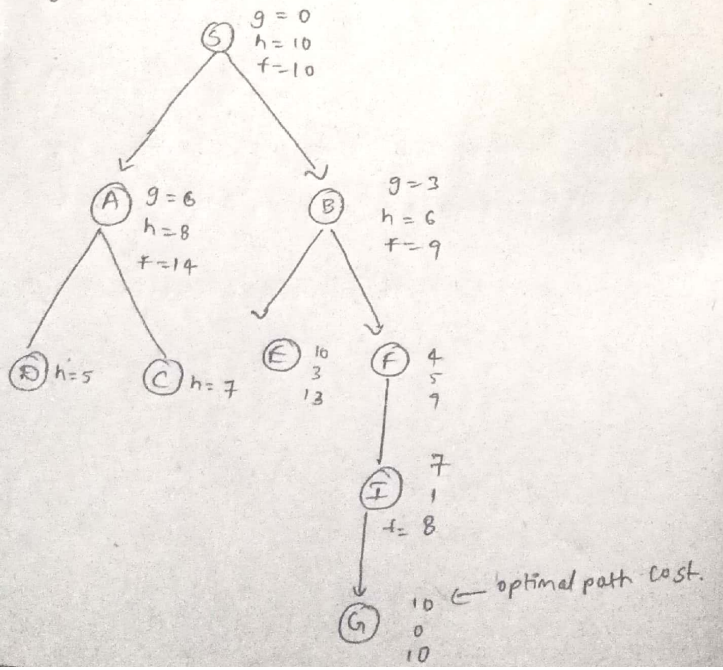


g(n) = Actual cost from s to n

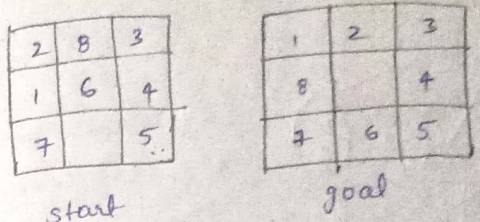
h(n) = estimated cost from n to g

f(n) = A star cost = g(n) + h(n)

Eg:-

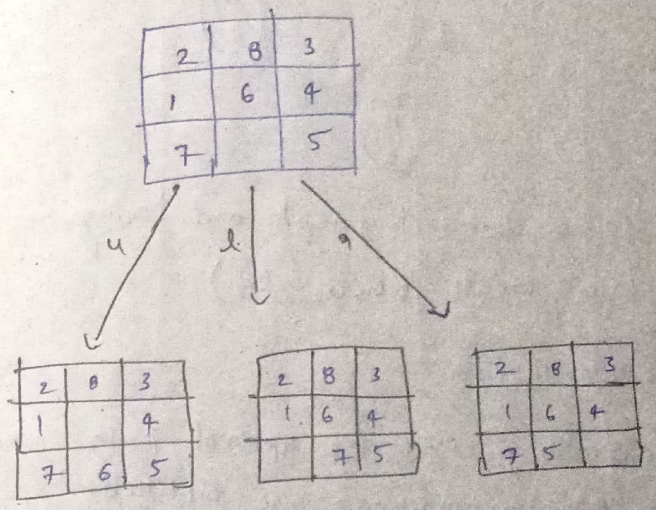


Eg:-

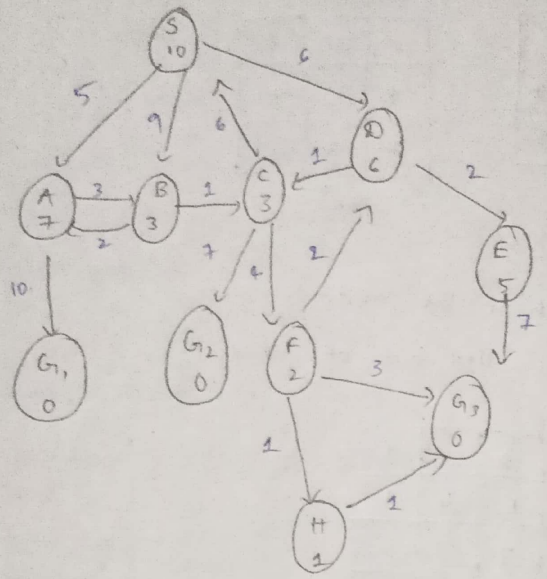


g(n) = depth of the node

h(n) = no. of tiles out of place.



Eg:-



→ It is a directed graph and having multiple Goals (G_1, G_2, G_3)

Admissible Heuristic :-

A* will always find the optimal path if the following conditions are fulfilled:-

1) Branching factor is finite.

2) For all edges the edge cost must be positive ($>\epsilon$).

(If negative cost is there at edge, it will never be reduced the cost.)

heuristic is less than the Actual cost.

→ Heuristic search is underestimating the Actual search cost.

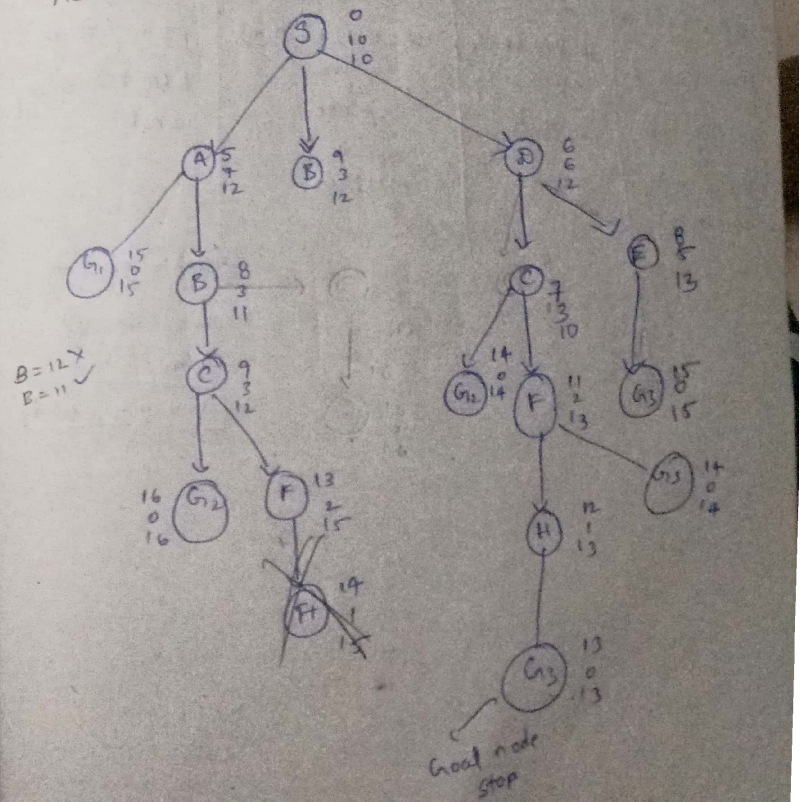
3) The Heuristic function always underestimate the actual cost to reach goal.

$$h(n) \leq h^*(n)$$

($h^*(n)$ = actual cost)

$h^*(n)$ = actual cost.

The above 3 characteristics are under Admissible heuristic.



CLOSED

S, 10

A, 12

B, 11

C, 12

optimal path : G - H - F - C - D - S - nil

cost = 13.

Three cases

if m ∉ OPEN and m ∉ closed	if m ∈ OPEN	if m ∈ CLOSED
/* new node */	if g(n) + k(n,m) < g(m)	if g(n) + k(n,m) < g(m)
parent(m) = n	update	update
g(m) = g(n) + k(n,m)		update children.
f(m) = g(m) + h(m)		

Dictionary:

n → m

parent(m) = n.

k(n, m) = Edge cost

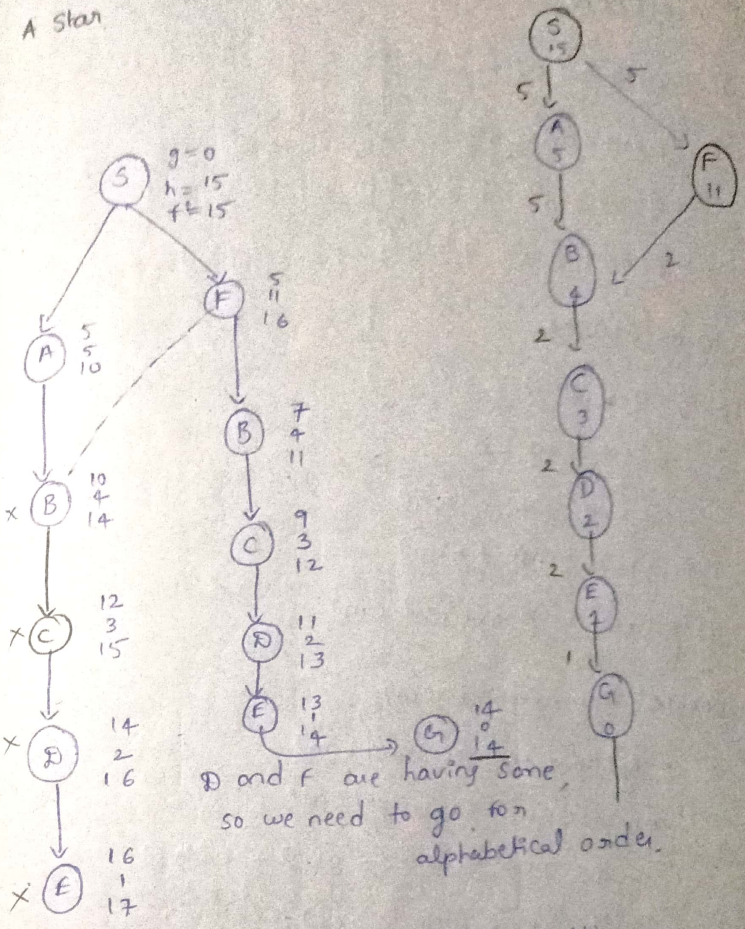
g(m) = g(n) + k(n, m)

h(m)

f(m) = g(m) + h(m)

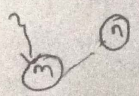
9/10/2023

A Star



D and F are having same, so we need to go for alphabetical order.

→ Finally, we got the optimal path.



propagation function

1) m ∉ open and m ∈ CLOSED

/* new node */

parent(m) = n

g(m) = g(n) + c(n,m)

f(m) = g(m) + h(m)

2) m ∈ OPEN

if g(m) > g(n) + c(n,m)

parent(m) = n

g(m) = g(n) + c(n,m)

f(m) = g(m) + h(m)

3) m ∈ CLOSED

if g(m) > g(n) + c(n,m)

parent(m) = n

g(m) = g(n) + c(n,m)

f(m) = g(m) + h(m)

propagate Improvement(m)

propagate Improvement(m)

neighbour ← move Gen(m)

for each s ∈ neighbours

do new Gvalue ← g(m) + c(m,s)

if newGvalue < g(s):

then parent(s) ← m

g(s) ← g(m) + c(m,s)

f(s) ← g(s) + h(s)

if s ∈ CLOSED

then

propagate Improvement(s)

A star :

OPEN ← [start]

g(start) ← 0

f(start) ← g(start) + h(start)

parent(start) ← nil

CLOSED ← [ ]

while OPEN is not empty

do n ← remove node n from OPEN with lowest f value.

add n to CLOSED

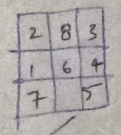
neighbours ← move Gen(n)

for m in neighbours:

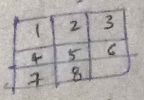
if case 1 :

elif case 2:

elif case 3:



return failure



Assignment

→ solve 8-puzzle problem using A* algorithm.

class Node: hvalue

state = [[2,8,3], [1,6,4], [7,'_',8]]

goal = [[1,2,3], [4,5,6], [7,8,'_']]

gvalue = 0 // parent.gvalue + 1

hvalue = //calculate

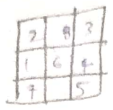
fvalue = gvalue + hvalue

Now, required methods are:

goal test ()

movGen ()

// returns new node for each children



[[2,8,3], [1,_,4], [7,6,5]]

[[2,8,3], [1,6,4], [_,7,5]]

[[2,8,3], [1,6,4], [7,5,_]]

class puzzle:

- OPEN = [ ]
- CLOSED = [ ]
- parent = { }
- printNode ()
- propagateImprovement ()
- Solve () → // Algorithm.
- takeInput ()

(2,1) + (-1,0) = (1,1) — 1st step

(1,0) = (3,1)

(0,-1) = (2,0)

(0,1) = (2,2)

(2,1) (2,2) → Third step.

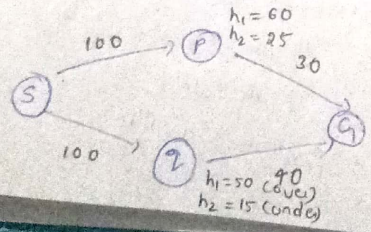
o/p:

All states should be printed before going to goal states.

→ That means, every step is to be printed.

Heuristic:

— underestimate or overestimate.

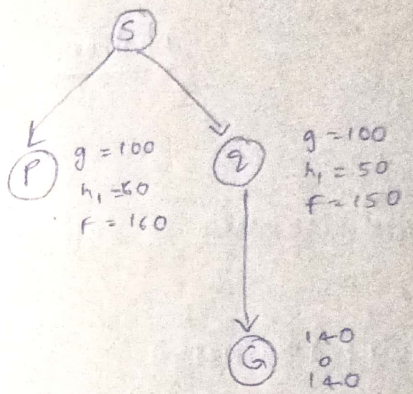


h_1 : overestimate

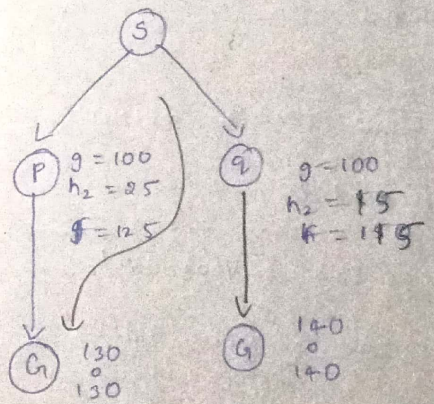
h_2 : underestimate.

→ If we use, overestimate,

overestimate:



underestimate



→ underestimate allows to check the other paths also but not overestimate.

Conclusion :

Heuristic must underestimate the actual cost.

↳ Then only the algorithm finds the optimal path.

11/10/2023

Game playing

- Multi agent scenario
  - competing against each other.
- Game Theory
  ↳ A Beautiful mind (John Nash)

Example of competing scenario and payoff

- prisoner's Dilemma

→ Thinking fast & slow (Decision making)

Two criminals are arrested.

→ Both confess ⇒ Both 1 years jail

→ Both Deny ⇒ Both 2 years jail

→ If one deny and other confess

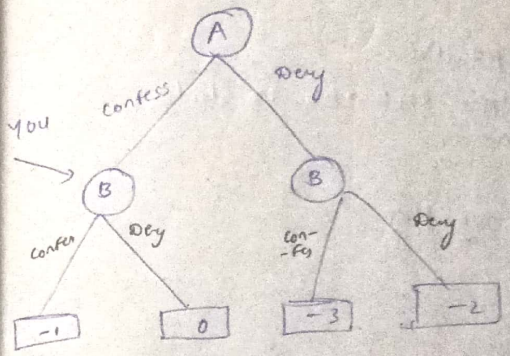
⇒ confess → 1 years

⇒ deny → 0 years.

pay off matrix

	B: confess	B: Deny
A: confess	(-1, -1)	(-3, 0)
A: Deny	(0, -3)	(-2, -2)

Game Tree :



→ Deny is the best decision, $\left(\frac{0+(-2)}{2} = \frac{-2}{2} = -1\right)$

→ confess is not the best decision, $\left(\frac{-1+(-3)}{2} = -\frac{4}{2} = -2\right)$

→ Assume that other player always act selfishly.

- Rational choice : Deny.

Types of Games :

Based on payoff

1) Zero - sum game

(⇒ stock-market trade making is zero - sum game)

Here,

Total payoff is zero

Eg: cricket

2) positive - sum game

- Total payoff is positive

Eg: Two student helping each other in study.

3) Negative - sum game

- Total payoff is negative.

Eg: price war.

→ Based on no. of players

→ Two player

→ multi player

Based on uncertainity

- complete information

- Eg - Tictactoe

- Incomple information

- Eg: card game.

Board Game

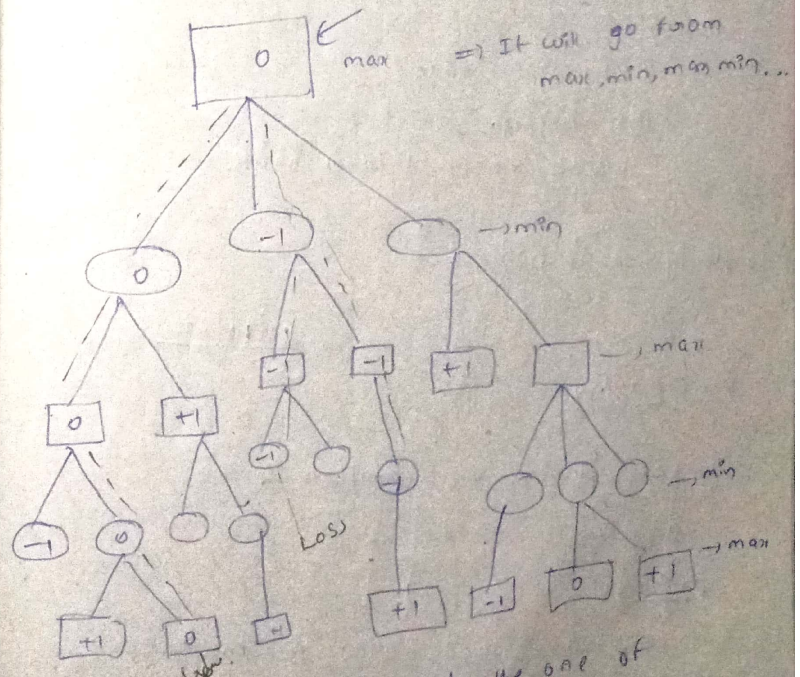
- Two player
- zero sum
- complete Information
- Alternative move
- Deterministic
  - no chance element.

A Game Tree : (seach space)

- max - ▭ (You)
- min - ○

⇒ It is in a layered manner

⇒ It will go from max, min, max, min...



→ Leaf nodes will represents the one of the possibility.

They are:

+1 :- max win

-1 :- max loss

0 :- draw

minimax backup rule :-

→ If 'O' place perfectly, the draw will happens.

⇒ if J is max node
then backup highest value from children nodes.

→ if J is min node
then backup Lowest value from children nodes.

Complexity = No. of nodes =

13/10/2023

Game playing

In Game playing, the problem is the complexity.

problem :

complexity = No. of nodes in Game tree.

Tic tac toe - $9 \times 8 \times 7 \cdots = 9!$

→ It is a big game.

chess :- Average 35 moves.

At every step, we have possibilities. In an average, 35 moves are available in chess.

→ Average game length is 50.

→ No. of nodes for a chess game is :-

$\frac{35}{50}$ i.e., $35^{50} \simeq 10^{120}$

→ In a second, the no. of nodes that can processed $= 10^{12}$ (1 trillion). (Assumption)

→ Total time $= \frac{10^{120}}{10^{12}}$ seconds are required.

i.e., $\boxed{10^{108} \text{ seconds}}$

31536000 (Total no. of seconds in a year)

$\frac{10^{108}}{10^7} \rightarrow$ years; 10^{99} years.

$10^{12} \rightarrow$ 1 billion year.

→ For practically, we do not implement the minimax backup rule.

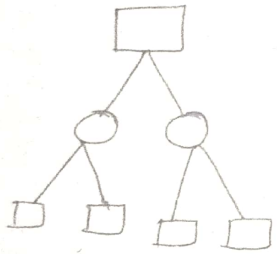
K-play Look ahead

In this, we are having,

→ k-steps looking ahead

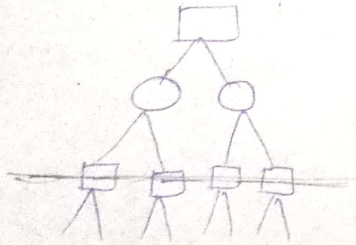
→ Evaluating

→ Taking decision.

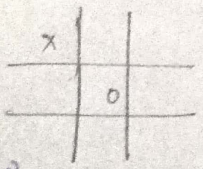


It is a 2-play look ahead.

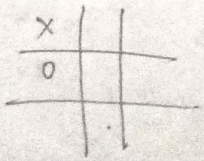
→ To Evaluate this,



→ we look into this step only and apply heuristic to evaluate.



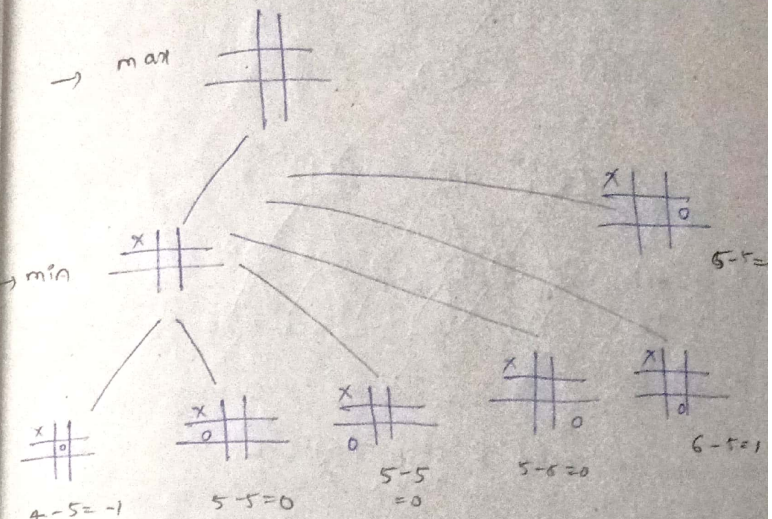
4-5 = -1

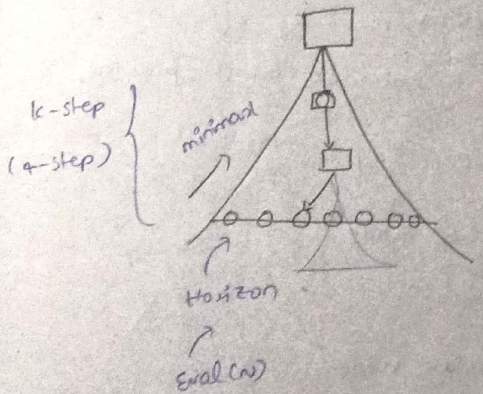


5-5=0

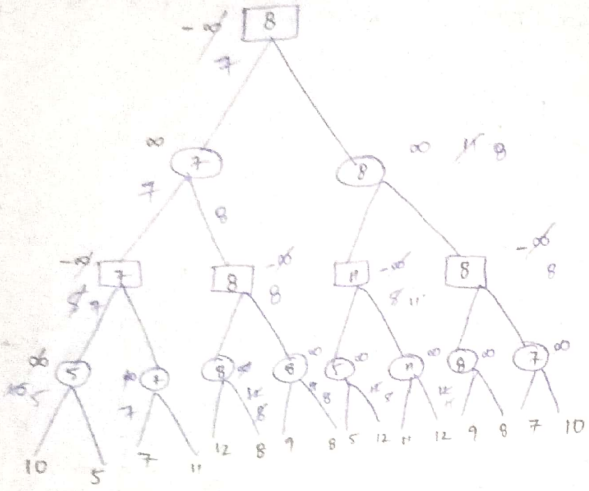
Eval (N) ⇒ no. of x_1, c_2, d available for 'X'

⇒ no. of x_1, c_2, d available for 'O'





Example:



Here, we are seeing 4-step process.

The Above min & max will tell the to move right / left direction.

Here we need DFS to calculate Game-playing.

In this, we have 3 possibilities,

if N is terminal node

return Eval (N)

if N is a max node

val = $-\infty$ → for each children of N

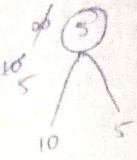
val ← max (val, minmax(c))

if N is a min node

val = ∞

for each children c of N

val ← min (val, minmax(c))



current value is = ∞

value is = 10 $10 < \infty$

we need to update

Again

current value = 10 $5 < 10$

we need to update

1) k-play Look ahead

2) pruning (cutting the unnecessary moves)

→ pruning is the General Technique in A I.

(α-β pruning)

Alpha - Beta pruning

α - will be updated only in max node

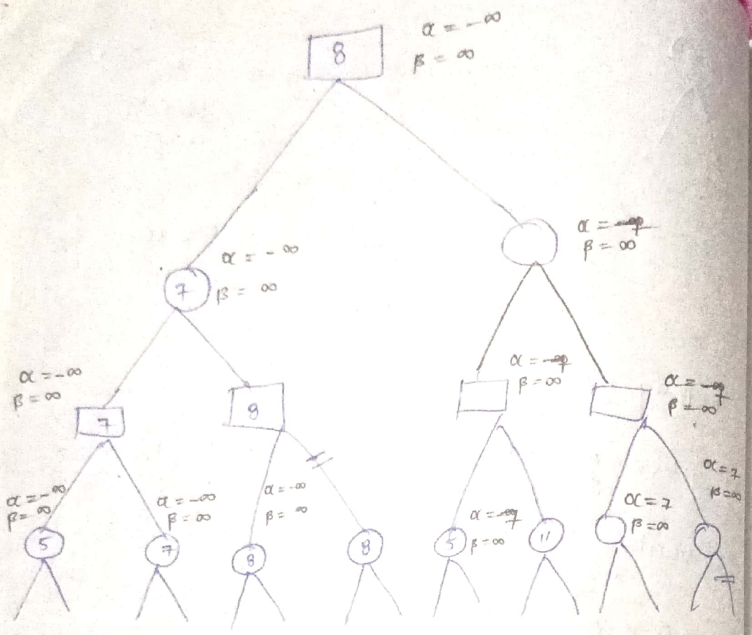
β - only in min node

α = $-\infty$

β = ∞

if ($\alpha \geq \beta$)

prune



1) How many nodes will be pruned ?

4 nodes

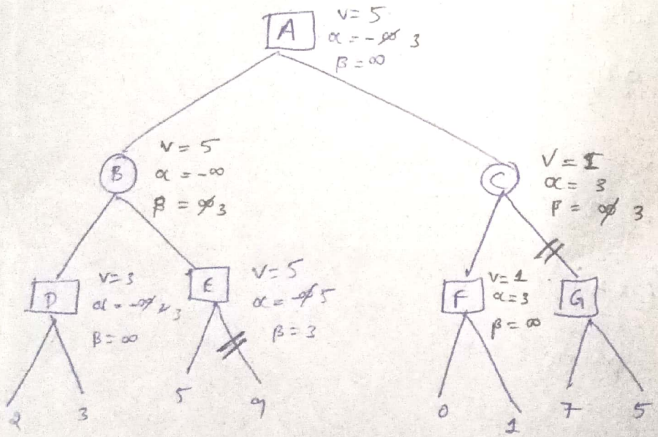
2) what is the final α and β value at root node ?

α = 3
β = ∞

3) which decision more will be taken by max at node A ?

A to B.

minimax Backup with Alpha Beta Pruning :-



Heuristic :- 200 (k − k')
+20 (q − q')
+10 (c − c')

if α ≥ β
prune

updated α = max
updated β = min

if max node
return min(α, β)

if max node
return max(α, β)

α = Large
β = Large

Alpha-Beta (N, α, β)

if N is a terminal node
return Eval (N)
else if N is a max node
for each child node c of N:
α ← max (α, Alpha-Beta (c, α, β))
if α ≥ β then
return β
return α
else if N is a min node

for each children c of N:
β ← min (β, Alpha-Beta (c, α, β))
if α ≥ β then return α

return β

classical AI

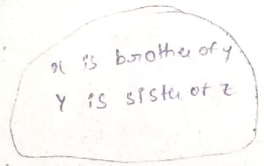
Two types of agents

- Goal Based Agent
- Game playing Agent
- Logical Agent
- Leaning Agent (Machine Learning)

Logical Agent :

→ propositional Logic

•goal
a is brother of z



Knowledge Base

→ Inference Rules
- modus ponen
- modus TOllen

Here,
Higher order Logic is there

i.e, First order Logic
→ variable
→ ∃ ∀ → for all.
↓
There Exists

→ second order Logic

Leaning Agent :

→ It Learns from experience
↳ supervised
→ unsupervised.

Syllabus

Algorithms
→ prediction – Linear Regression
– classification
↳ Logistic Regression
– Bayes classifier
– Artificial Neural network (Basic)

– Evaluation metrics
– Accuracy
– precision
– Recall
– F1 score.

– classical AI
→ Goal Based Agent
→ Game playing Agent
→ Logical Agent

– modern AI
– Leaning Agent
↳ Machine Learning
↳ Data/Experience.

1) Supervised Learning

It uses Labelled data.

2) Unsupervised Learning

It use unlabelled data.

Supervised :-

- prediction
  - ↳ Linear Regression
- classification

:- Linear Regression :-

↳ simple
- multiple
- polynomial

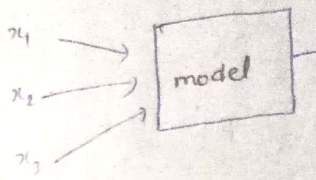
→ It is used to identify the something.

Eg:- House price prediction

(y) depended variable	Independed variable (x_1 x_2 x_3)
price	→ Area (x_1)
	→ Distance from city (x_2)
	→ no.of floors (x_3)

→ It is like a program.

⇒ This function we call it as model.



Input

output

$\hat{y}$ → predicted value.

y	x_1	x_2	x_3

$\hat{y} = f(x_1, x_2, x_3)$

$\hat{y} = w_1x_1 + w_2x_2 + w_3x_3$

→ It is a Linear. (power is '1')
(or) degree

$\hat{y} = w_1x_1^2 + w_2x_2^3 + w_3x_3$

→ It is non-linear (power is 2, 3, 1)

$y = \alpha + \beta x$

Estimate the α & β

$\hat{\alpha}$ and $\hat{\beta}$ (prediction)

$\hat{y} = \hat{\alpha} + \hat{\beta}x$

↳ Estimated model.

↳ feature

Data

x	y
0	1
2	3
4	8
5	11
3	7

$\hat{\alpha}$ and $\hat{\beta}$ are the parameters of the estimated model.

→ will have data and model.

we need to find α and β parameters.

$$\hat{\alpha} = \bar{y} - \hat{\beta}\bar{x}$$

$$\hat{\beta} = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2}$$

$$\hat{y} = \hat{\alpha} + \hat{\beta} x$$

↓ Intercept ↘ slope

$\Rightarrow$ $\hat{\alpha}$ & $\hat{\beta}$ are controllers

Best Fit Line :-

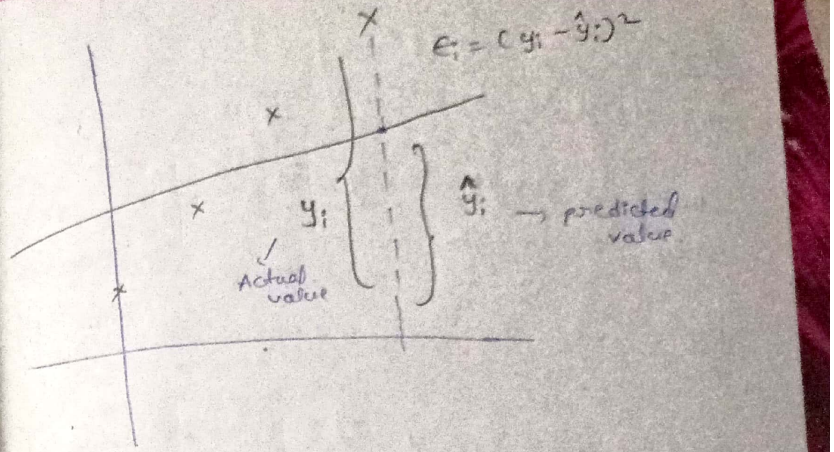
↳ Error

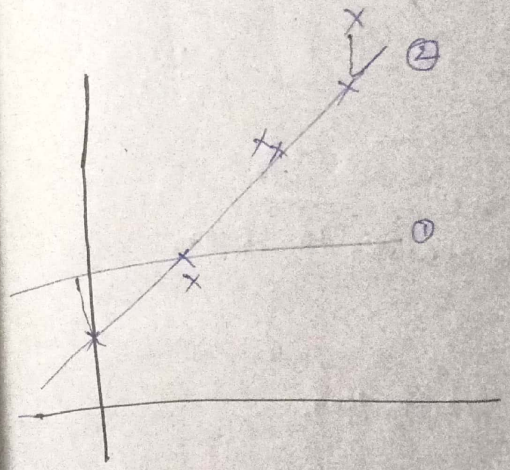
↳ minimum Error

Error = Actual value − predicted value

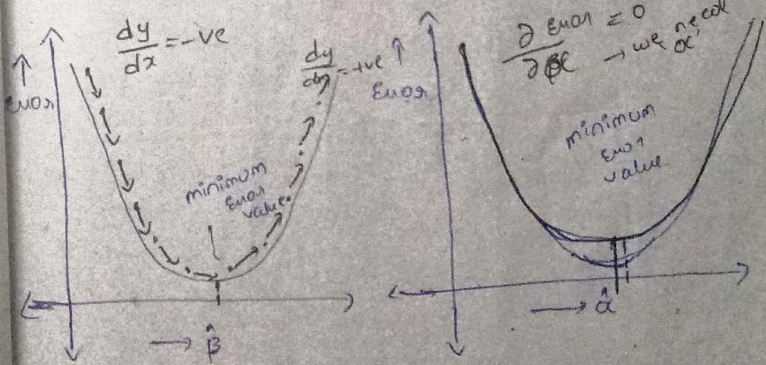
$$SSE = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

SSE :- Sum of squared Error.





→ Line with minimum error.

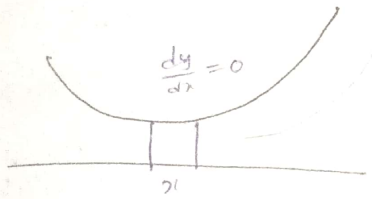


slope:

It is change in x-axis & y-axis. Ratio of

→ Ratio of change in y-axis vs change in x-axis.

$$\frac{dy}{dx} = -ve$$



It is the technique which is used to measure the Error.

$\frac{dy}{dx}$ = derivative.

→ It is change in the function.

Estimating α and β

$\hat{\alpha}$ s.t $\hat{\beta}$ s.t

$\frac{\partial SSE}{\partial \hat{\alpha}} = 0$ $\frac{\partial SSE}{\partial \hat{\beta}} = 0$

$$SSE = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

$$SSE = \sum_{i=1}^{n} (y_i - \hat{\alpha} - \hat{\beta} x_i)^2$$

⇒ first derivative

$$\frac{\partial SSE}{\partial \hat{\alpha}} = 0$$

$$\Rightarrow 2 \sum_{i=1}^{n} (y_i - \hat{\alpha} - \hat{\beta} x_i)(-1) = 0$$

$$\Rightarrow \sum_{i=1}^{n} (y_i - \hat{\alpha} - \hat{\beta} x_i) = 0$$

$$\Rightarrow \sum_{i=1}^{n} y_i - \sum_{i=1}^{n} \hat{\alpha} - \sum_{i=1}^{n} \hat{\beta} x_i = 0$$

$$\Rightarrow \sum_{i=1}^{n} y_i = n\hat{\alpha} + \hat{\beta} \sum_{i=1}^{n} x_i$$

$$\Rightarrow \frac{\sum_{i=1}^{n} y_i}{n} = \hat{\alpha} + \frac{\sum_{i=1}^{n} x_i}{n}$$

$$\Rightarrow \bar{y} = \hat{\alpha} + \hat{\beta} \cdot \bar{x}$$

$$\Rightarrow \boxed{\hat{\alpha} = \bar{y} - \hat{\beta} \bar{x}}$$

$$SSE = \sum_{i=1}^{n} (y_i - (\bar{y} - \hat{\beta}\bar{x}) - \hat{\beta}x_i)^2$$

$$= \sum_{i=1}^{n} [(y_i - \bar{y}) - \hat{\beta}(x_i - \bar{x})]^2$$

$$\frac{\partial SSE}{\partial \hat{\beta}} = 2 \sum_{i=1}^{n} [(y_i - \bar{y}) - \hat{\beta}(x_i - \bar{x})] - (x_i - \bar{x}) = 0$$

$$\Rightarrow \sum_{i=1}^{n} [(y_i - \bar{y}) - \hat{\beta}(x_i - \bar{x})](x_i - \bar{x}) = 0$$

$$\Rightarrow \sum_{i=1}^{n} (y_i - \bar{y})(x_i - \bar{x}) - \hat{\beta} \sum_{i=1}^{n} (x_i - \bar{x})^2 = 0$$

$$\Rightarrow \hat{\beta} = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2}$$

Setting the derivative to '0' and finding the minimum error.

That minimum error is $\hat{\alpha}$ & $\hat{\beta}$.

Neural network technique will follows the above technique.

Learning Agent

06/11/2023

prediction - Regression

	train data							test data	
x	1	2	3	4	5	6	7	8	9
y	9	8	10	12	11	13	14	16	17

→ Which of the following two models is a better model?

(a) $y = 2x + 6$ (Better)

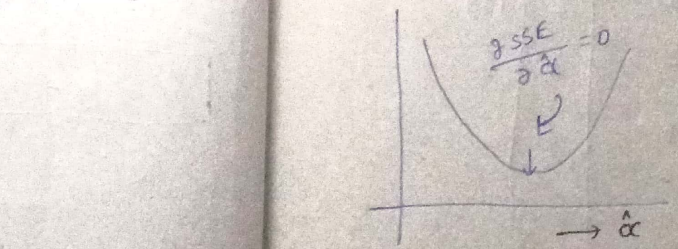
(b) $y = -2x + 6$

- Find the parameters of the best fitting model.
- Find the error on the test set by best fitting model.

Sol: $\hat{y} = \hat{\alpha} + \hat{\beta}x$

$\hat{\alpha}, \hat{\beta}$ are parameters.

min error = SSE



$$\hat{\alpha} = \bar{y} - \hat{\beta}\bar{x}$$

$$\hat{\beta} = \frac{\Sigma (x_i - \bar{x})(y_i - \bar{y})}{\Sigma (x_i - \bar{x})^2}$$

$$\hat{\beta} = \frac{n\Sigma x_i y_i - \Sigma x_i \Sigma y_i}{n\Sigma x_i^2 - (\Sigma x_i)^2}$$

If we have
Data, Regression is imp.

model:

$\hat{y} = f(x)$

parameter	learning
(α, β)	$\hat{\alpha}$

Error.

X	Y	$\hat{y} = 2x+6$	$\hat{y} = -2x+6$	ε_1	ε_2
1	9	8	4	1	25
2	8	10	2	4	36
3	10	12	0	4	100
4	12	14	-2	4	4
5	11	16	-4	25	
6	13	18	-6		
7	14				
8	16				
9	17				

Error $= (y_i - \hat{y}_i)^2$

→ $y = 2x+6$ is better than $y = -2x+6$.

X	Y	X^2	X·Y
1	9	1	9
2	8	4	16
3	10	9	30
4	12	16	48
5	11	25	55
6	13	36	78
7	14	49	91
28	77	140	334
~~9~~	~~17~~	~~81~~	

$\bar{x} = \frac{28}{7} = 4$

$\bar{y} = \frac{77}{7} = 11$

$\hat{\alpha} = 11 - 0.929 \times 4$

$\hat{\alpha} = 7.284$

$\hat{\beta} = \frac{7 \times 334 - 28 \times 77}{7 \times 140 - 28 \times 28}$

$\hat{\beta} = 0.929$

$\hat{y} = 7.284 + 0.929x$

From on test data

x	y	$\hat{y}$
8	16	14.716
9	17	15.645

Test Error:

$$(16 - 14.716)^2 + (17 - 15.645)^2$$

=

MLR:

X_1	X_2	Y
1	4	1
2	5	6
3	8	8
4	2	12

Here, there are two features, i.e., X_1 & X_2.

X_1, X_2 : input.

Y : output

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2$$

parameters : $\hat{\beta}_0, \hat{\beta}_1, \hat{\beta}_2$

$$\begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \\ \hat{\beta}_2 \end{bmatrix} = (X^T X) X^T Y$$

$3 \times 3 \quad 3 \times 4 \quad 4 \times 1$

$3 \times 4 \quad 4 \times 1$

3×1

β vector

$$X = \begin{bmatrix} 1 & 1 & 4 \\ 1 & 2 & 5 \\ 1 & 3 & 8 \\ 1 & 4 & 2 \end{bmatrix}_{4 \times 3}$$

$$Y = \begin{bmatrix} 1 \\ 6 \\ 8 \\ 12 \end{bmatrix}_{4 \times 1}$$

using R^2 and Adjusted R^2, we can

model Evaluation

R^2 and Adjusted R^2

- prediction - area | price
- classification -

age	salary	loan approved

y → continuous ⇒ prediction

y → Discrete ⇒ classification

$\hat{y} = f(X_1, X_2)$

classification :-

→ Email classification

→ URL classification
  └ safe/not safe.

→ News classification

→ Image classification

→ Headline classification.

→

Supervised :-

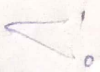
Having Y label.

Unsupervised :-

does not having Y label.

Evaluation of classification algorithms :-

→ Error

→ Cancer prediction 

Actual	predicted	
yes	yes	→ True positive.
yes	no	→ Errors. → false negative
no	yes	→ Errors. → Falsepositive
no	no	→ True negative.

And we arange the above Table.

confusion matrix :- → predicted

actual	Yes	NO
Yes	TP	FN Type I
NO	FP Type II	TN

Type I :-
FN (False Negative)

Type II :-
FP (False positive)

→ False Negative is more dangerous.