ARTIFICIAL INTELLIGENCE

- Toy problems are simplified versions of the real world problems.

- Traits which can be incorporated in computer from human beings:
  1. Decision Making, Problem Solving.
  2. Natural Language.
  3. Vision.
  4. Logical Reasoning.
  5. Learning.

- Units:
  1. Basics of AI.
  2. Decision Making.
  3. Logical Reasoning.
  4. Reasoning with uncertainty.
  5. Machine Learning.
  6. Expert Systems.

** Two Water Jugs Problem: Optimal Solution.

1. Two Jugs of 5 litres and 3 litres size.
2. Tap → unlimited water supply.
3. Ground → unlimited water can fall on ground.
4. Requires exactly 4 litres of water.

Soln: 1. Initialize Jugs with 0 litres.

(0, 0)

↓ 1. Fill 5 litre jug fully.

(5, 0)

↓ 2. Fill 3 litre jug from 5 litre jug.

(2, 3)

↓ 3. Throw the 3 litre jug water.

(2, 0)

↓ 4. Transfer remaining 2 litre water to 3 litre jug.

(0, 2)

↓ 5. Fill 5 litre jug fully again.

(5, 2)

↓ 6. Transfer 1 litre to fill the 3 litre jug fully.

(4, 3).

2. (0,0) → (0,3) → (3,0) → (3,3) → (5,1) → (0,1) → (1,0) → (1,3) → (4,0).

** Three Water Jug Problem: Optimal Solution.

1. 3 Jugs with capacity 8, 5, 3 respectively.
2. No tap available & 8 litre jug is fully filled.
3. No water can be thrown onto the ground.
4. 4 litre of water required.

Soln:

(8, 0, 0)

↓ 1

(3, 5, 0)

↓ 2

~~(0, 5, 3)~~ (3, 2, 3)

↓ 3

(6, 2, 0)

↓ 4

(6, 0, 2)

↓ 5

(1, 5, 2)

↓ 6.

(1, 4, 3).

** 4-Queens Problem:

** 8-Queens Problem:

Has no optimal solution.

1	2	3	4	5	6	7	8.
Q_1							
				Q_2			
							Q_3
					Q_4		
		Q_5					
						Q_6	
	Q_7						
			Q_8				

- Difference between Water Jug Problem and N-Queen Problem is that, Water Jug problem has optimal solution but N-Queen Problems solutions are equally good.

5. 8- Puzzle Problem. }
6. 15- Puzzle Problem. } Has optimized solution.

7. Tower of Hanoi.
8. Lion Goat Grass Problem.

no. of discs.		no. of moves.
2	$\rightarrow$	3.
3	$\rightarrow$	8.
4	$\rightarrow$	15.
5	$\rightarrow$	31.
⋮		⋮
$\boxed{n}$	$\rightarrow$	$\boxed{2^n - 1}$

8. Lion Goat Grass Problem:

Bank 1.	River.	Bank 2.
(L, Go, G, B)		(0, 0, 0, 0)
(L, 0, G, 0)		(0, Go, 0, B)
(L, 0, 0, 0)		(0, 0, G, B)
(L, Go, 0, B)		(0, 0, G, 0)
(0, Go, 0, 0)		(L, 0, G, B)
(0, Go, 0, B)		(L, 0, G, 0)
(0, 0, 0, 0)		

9. Missionaries and Cannibals:

Rules:

1. No of Missionaries or ~~cannibals~~ can be greater than No. of Cannibals but not vice versa.
2. Atmost 2 people can go at a time in the boat.

Soln:

Side A (M, C, B)		Side B (M, C, B)
(3, 3, 1)		(0, 0, 0)
(2, 2, 0)	$\leftarrow$	(1, 1, 1)
(3, 2, 1)		(0, 1, 0)
(2, 1, 0)		(1, 2, 1)
(3, 1, 1)		(0, 2, 0)
(2, 1, 0)		(1, 2, 1)
(2, 2, 1)		(1, 1, 0)
(1, 1, 0)		(2, 2, 1)
(1, 2, 1)		(2, 1, 0)
(0, 1, 0)		(3, 2, 1)
(0, 2, 1)		(3, 1, 0)
(0, 0, 0)		(3, 3, 1).

11 steps.

10. **Jealous Husband Problem:**

Husband : H_1, H_2, H_3.

Wives : W_1, W_2, W_3.

No wife cannot be alone with other husbands.

Soln: $H_1 H_2 H_3 W_1 W_2 W_3 B \rightarrow$

$H_1 H_2$.

$H_2 H_3 W_1 W_2$

$H_1 H_2 H_3 W_1 W_2 W_3 B \longrightarrow W_1 W_2$.

$H_1 H_2 H_3 W_1 W_3$. $\longleftarrow W_2$.

~~$H_1 H_2 H_3 W_1 W_3 \longrightarrow H_1 H_2 W_2$~~

~~$H_1 H_3 W_1 W_3$. $\longleftarrow H_2 W_2$.~~

1) $H_1 H_2 H_3$ B $\xrightarrow{H_1 W_1}$ $H_1 W_1$.
$W_1 W_2 W_3$

2) $H_1 H_2 H_3$ $\xleftarrow{H_1}$ $H_1 W_1 B$.
$W_2 W_3$

3). $H_1 H_2 H_3 B$ $\xrightarrow{W_2 W_3}$ $W_1 W_2 W_3$.

4). $H_1 H_2 H_3$ $\xleftarrow{W_1}$ $W_2 W_3 B$.

5). $H_1 W_1 B$. $\xrightarrow{H_2 H_3}$ $W_2 W_3 H_2 H_3$.

6). $H_1 W_1 H_2 W_2$ $\xleftarrow{H_2 W_2}$ $H_3 W_3 B$.

7). $W_1 W_2 B \xrightarrow{H_1 H_2}$ $H_1 H_2 H_3 W_3$ ~~B~~.

8). $W_1 W_2 W_3$ $\xleftarrow{W_3}$ $H_1 H_2 H_3 B$.

9). $W_3 B \xrightarrow{W_1 W_2}$ $H_1 W_1 H_2 W_2 H_3$ ~~W~~

10). $W_1 W_3$ $\xleftarrow{W_1}$ $H_1 H_2 W_2 H_3 B$.

11) ~~W~~ B. $\xrightarrow{W_1 W_3}$ $H_1 W_1 H_2 W_2 H_3 W_3$ ~~B~~

11. **Counter-feit ~~Coin~~ Problem:**

Out of 12 ~~pt~~ coins, one is counterfeit. You cannot figure just by looking. Either the weight of counterfeit coin is less or more. We can weigh only 3 times and figure out whether the weight is less or more than others

12. Conjuncture — a statement which doesnt has a to proof or disproof.

Ex: Knuth's Conjucture:

$$\left\lfloor \sqrt{\sqrt{\sqrt{\sqrt{\sqrt{(4!)!}}}}} \right\rfloor = 5$$

4 is fixed, we can change the sequence of operations to get any positive number.

** Solving Problems using Systems:

- Ordere tuple → (x, y)
  ↳ set of co-ordinates where order is important.

1. One tuple represents a state of the problem.
2. Set of all possible states are known as state space.
3. Special states of state space are initial state and final state. (Goal state).
4. There may be atmost and atleast one initial state.
5. There can be more than one goal state.
6. States can be represented using different ways.

Ex: $S_1 = (8,0,0)$ Child of S_4.

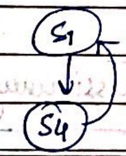
$S_2 = (6,2,0)$ Not a child of S_1

$S_3 = (5,3,0)$ — " —

$S_4 = (3,5,0)$ Child of S_1.

Also S_1 is child of S_4 creates a cycle.

★ • State space will be represented with tree/graph



- Path is an alternating sequence of nodes and edges.
- Solution is not always a path in every situation. Ex: 3-water Jug Problem — where path is needed, 8 queen — where only final state is required.
- Therefore solution is a special path including the initial state, intermediate nodes, traversing edges and goal state.

  Solution is either a goal state or a path of intermediate nodes.

★ Missionaries and Cannibals:

M C B

(3,3,1)

CC, MM, M, MC, C

states → (3,1,0) (1,3,0) (2,3,0) (2,2,0) (3,2,0)

violates the constraint.

state 1 ⟶ | move / operator / transition | ⟶ state 2.

★ • Operators / Rules:

M → One missionary goes to other side of river

MM → Two ——— " ———

C → One Cannibal goes to other side of river.

CC → Two ——— " ———

MC → One Missionary and one Cannibal ——— " ———

★ 3-Water Jug Problem:

- Operators:

1. Pour water from 8 litre Jug to 5 litre Jug.
2. Pour water from 5 litre Jug to 3 litre Jug.
3. Pour water from 5 litre Jug to 8 litre Jug.
4. Pour water from 3 litre Jug to 5 litre Jug.
5. Pour water from 8 litre Jug to 3 litre Jug.
6. Pour water from 3 litre Jug to 8 litre Jug.

★ 2-Water Jug Problem:

- Operators:

1. Fill the 3-litre Jug fully from tap.
2. Fill the 5-litre Jug fully from tap.
3. Pour the water from 5 litre Jug to 3 litre Jug.
4. Pour the water from 3 litre Jug to 5 litre Jug.
5. Empty the 5 litre Jug on the ground.
6. Empty the 3 litre Jug on the ground.

★ 4-Queen Problem:

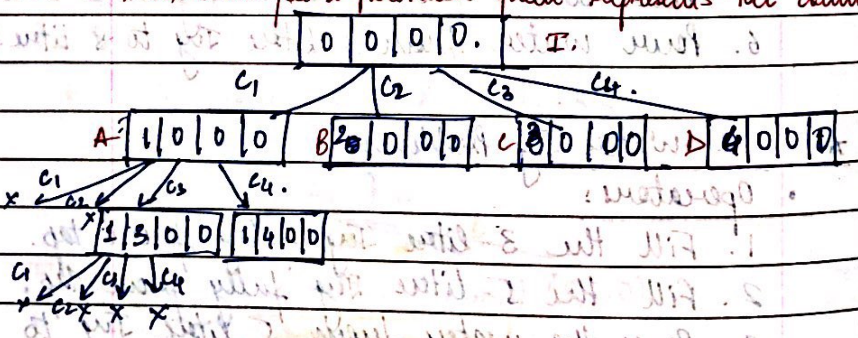
	q_1		
			Q_2
Q_3			
		Q_4	

Assumption: Put the i^{th} queen in i^{th} row always.

| 0 | 0 | 0 | 0 |

4 value = 4 column value.

* box value for a particular queen represents the column no.



• The computer will decide if the solution is achieved if the array contains 4 values other than 0. (Goal State).

★ 8-Puzzle Problem:

· Representation: Matrix of 3x3.

Ex: $\begin{bmatrix} 4 & 3 & 2 \\ 1 & 0 & 6 \\ 5 & 7 & 8 \end{bmatrix}$ 3x3.

0 ≡ empty tile.

State space size $= 9! = 9\times8\times7\times6\times5\times4\times3\times2\times1$.

Fix initial tile out of 9, second with out of 8 and so on.

Initial state: can be anything, wherever we start from.

Goal State: Can vary and depends on how we define.

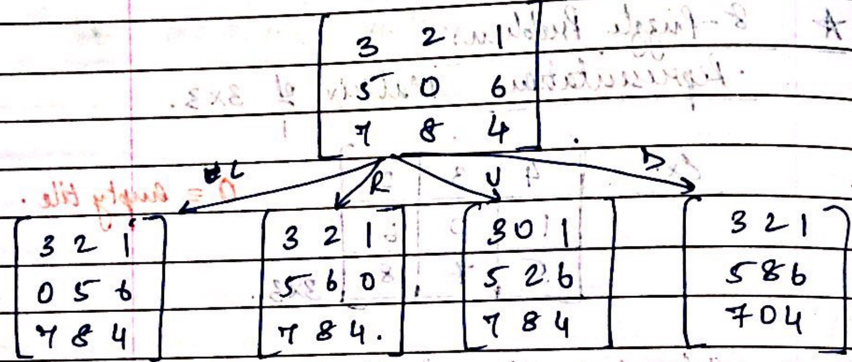
Ex. $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{bmatrix}$, $\begin{bmatrix} 1 & 2 & 3 \\ 8 & 0 & 4 \\ 7 & 6 & 5 \end{bmatrix}$, etc.

Both initial and goal states are input to the program.

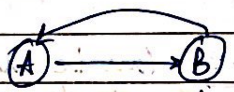
★ Operations:

L – Move empty space left.
R – Move empty space Right.
U – Move empty space Up.
D – Move empty space down.

Natural number → countable infinity.
Real Number → Uncountable infinity.



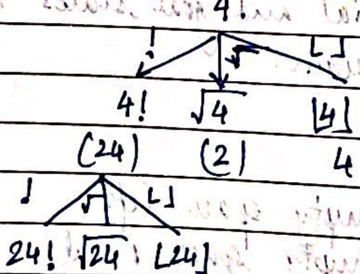
For a state, it can have atmost 4, 3, 2 children.



A can be parent and B can be parent of A.

** Knuth's conjucture:

It has ~~next~~ finite number of operations / operators but the ~~space~~ state ~~stage~~ space tree size is infinite.



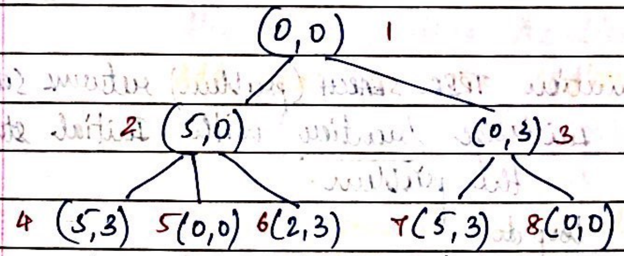
- Search of next Operator.
  → Which next operator to apply on a state to get close to the goal state.

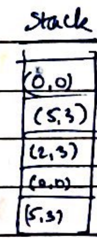
Soln: 1. Control Strategy.
2. Searching Algorithms.

Ex: 2-Water Jug Problem: State space is represented with graph.

Operators:

1. Fill the 5 litre Jug from Tap.
2. — " — 3 — " —
3. Pour water from 5 litre Jug on ground.
4. — " — 3 — " —
5. Pour water from 5 litre Jug to 3 litre Jug.
6. — " — 3 litre Jug to 5 — " — .





Traverse and perform operation in Breadth First Search Traversal.

*** Measuring Problem Solving Performance:

1. **Completeness:**
   It is an algorithm that guarantees to find solution when there is one.

2. **Optimality:**
   With respect to Algorithm. Optimal Algorithm and optimal solution has different meaning

3. **Time Complexity:**
   How long does it take to find out a solution?

4. **Space Complexity:**
   How much memory is needed to find the solution.

- function TREE_SEARCH (problem) returns solution or failure

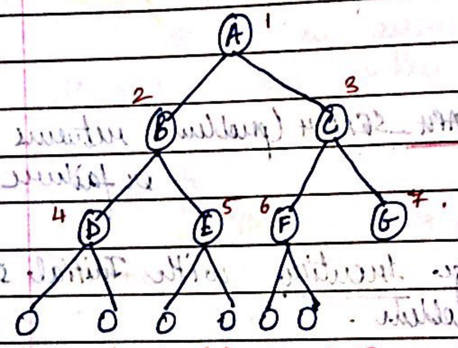
{ Initialize frontier with Initial state of the problem.
  loop do
    If frontier is empty then return failure
    Choose a leaf node and remove it from the frontier.
    If node contains goal state, returns the corresponding solution.
    Expand chosen node, adding the resulting nodes into frontier.
}.

- function GRAPH_SEARCH (problem) returns solution or failure.

{
  Initialize frontier with Initial state of the problem.
  Initialize Explored set to be empty
  loop do
    If frontier is empty then return failure.
    Choose a leaf node and remove it from the frontier.
    If node contains goal state, returns the corresponding solution.
    Add node to the explored set.
    Expand chosen node, adding the resulting nodes in to frontier. Only If not present in frontier or explored set.
}.

If frontier is queue → FIFO → then BFS.
If frontier is stack → LIFO → then DFS.
If frontier is ~~Circled~~ Priority Queue → then same other implementation.

* Breadth First Search Algorithm using GRAPH-SEARCH.



Current Node	Frontier	Explored
A	C, B	

Current Node	Frontier	Explored
B	E, D, C	A

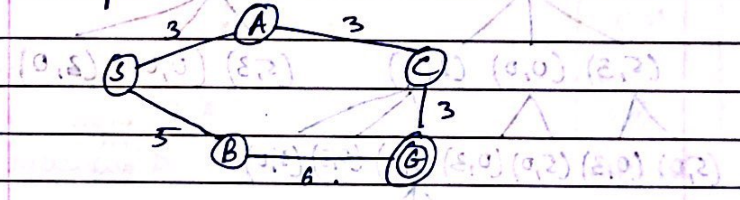
Current Node	Frontier	Explored
C	G, F, E, D	B, A

- In BFS, the ~~go~~ state is determined to be a goal state upon generation.
- In General Algorithm, the state is first pushed in frontier and upon processing checked for goal state.

- Branching factor is the number of child nodes a state the can generate.

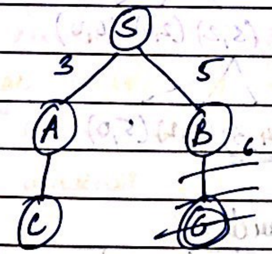
- An infinite state space tree can have finite branching factor. Ex: Knuth's conjecture with only 3 operations.

1. Complete ? → Yes.
2. Optimal ? → No.



Apply BFS, $Q =$ | ~~S~~ | ~~A~~ | ~~B~~ | C | G |

V = S, A, B, C, G



= 11 But optimal is 9

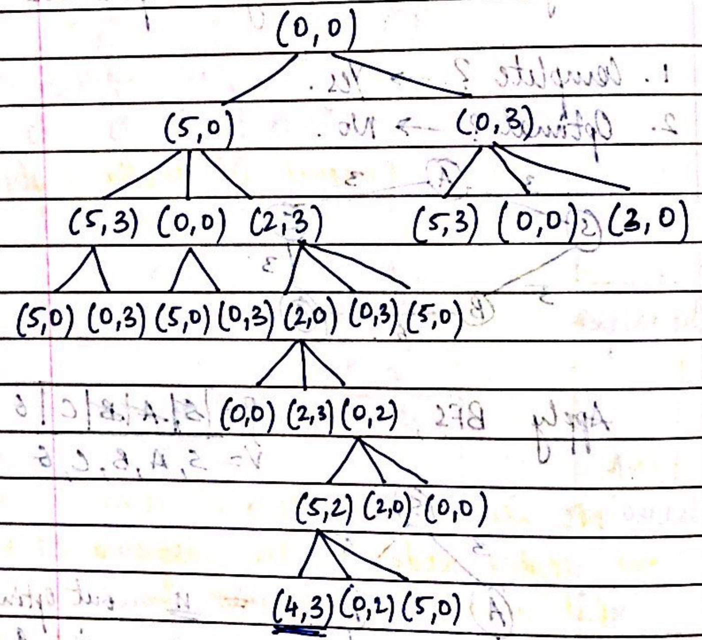
Current Node	Frontier	Explored
S.	B, A, ~~S~~	

Current Node	Frontier	Explored
A	C, B	S

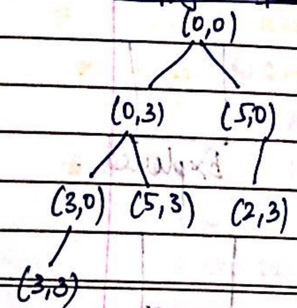
Goal Found

Current Node B. | G | A S

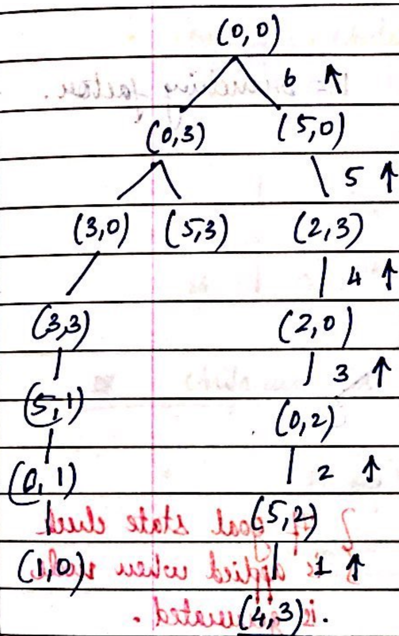
★ BFS on 2 Water Jug Problem:



★ Apply Graph Search:



Fr.	Ex
3,3	5,3
—	3,0
—	5,0
—	0,3
—	0,0



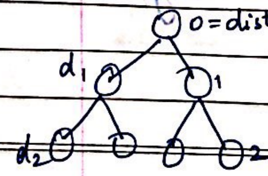
Explored
(0,1)
(0,2)
(5,1)
(2,0)
(3,3)
(2,3)
(5,3)
(3,0)
(5,0)
(0,3)
(0,0)

Explored.

- Note:

BFS always gives optimal solution when the edge weights of the graph are same. But when edge weights are different BFS doesnot guarantee optimal solution.

★ If Path cost is non-decreasing function of depth of the node then BFS is optimal.



Path cost $= f(d)$

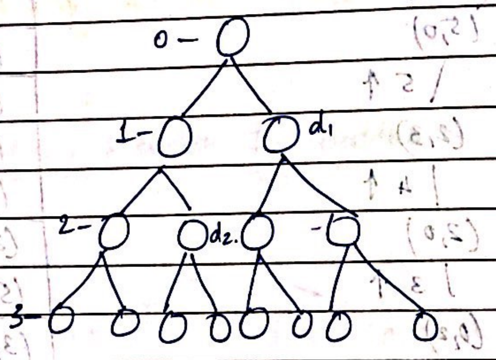
f is non-decreasing

$d_2 > d_1$

then $f(d_2) \geq f(d_1)$.

3. Time Complexity:

b = branching factor.



$1 + b + b^2 + b^3 + \dots + b^d$ } If goal state check is applied when node is generated.

$\therefore O(b^d)$

$1 + b + b^2 + b^3 + \dots + b^{d+1}$ } If goal state check is applied when node is processed.

$\therefore O(b^{d+1})$

4. Space Complexity:

$O(b^d)$ — Combination of Explored and Frontier.

** Brute Force Analysis :-

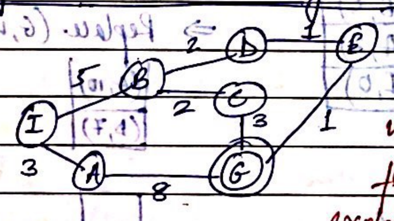
Ex: Time required to process 1 node $= 10^{-6}$

memory for 1 node = 100 bytes.

$b = 10$.

Depth	Nodes	Time	Memory
2	$111 \cong 10^2$	0.111 msec	111000 bytes = 11.1 KB
4	$11111 = 10^4$	11.111 msec	1.11 MB.
⋮	⋮	⋮	⋮
14	$\cong 10^{14}$	3.5 years	990 TB.
16	$\cong 10^{16}$.	350 years	Exabyte.

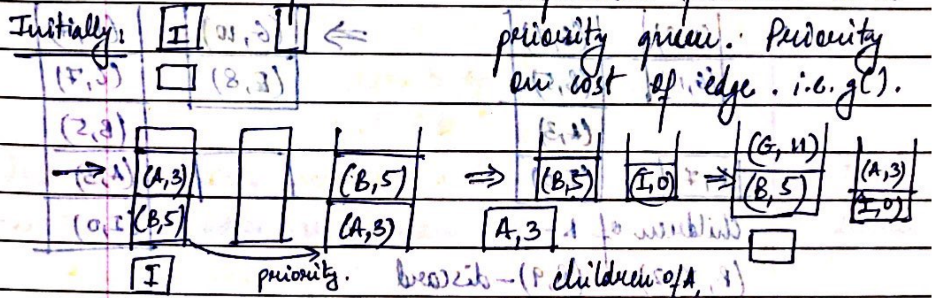
■ Uniform-Cost Search Algorithm:



Goal check is applied when node is taken from frontier for exploration and not during generation of that node.

Frontier Explored set

Implement frontier as a priority queue. Priority on cost of edge. i.e. g( ).



Explore I

$g(A) = 3$. g = distance of a node from start node.

$g(B) = 5$

children of A → (I, 6) → i.e. I to A, A to I 3 & 3 = 6

→ (G, 11)

If a node exists, check its value & keep lower one.

Algo - cost → cost between two nodes.

(D,7)		(G,11)	(B,5)
(C,7)	(A,3)	(D,7)	(A,3)
(G,11)	(I,0)	(C,7)	(I,0)

⇒ $\boxed{B,5}$

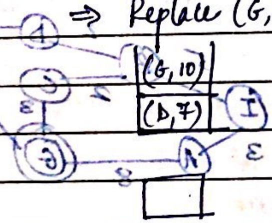
Children of B :

(I,10) — discarded as (I,0) is lower limit.

(C,7), (D,7).

Uniform - cost Search Algorithm:

⇒ (G,11) (D,7) | (B,5) (A,3) (I,0) ⇒ Replace (G,11) by (G,10) | (G,10) (D,7) | (C,7) (B,5) (A,3) (I,0)



Score is applied when node is taken from frontier for exploration not ... of that node

$\boxed{C,7}$

Children of C,

(B,9) — discard (B,5) < (B,9)

(G,10) — Keep, (G,10) < (G,11)

Frontier Explored set

Frontier	Explored set
(E,8)	(C,7)
(G,10)	(B,5)
	(A,3)
	(I,0)

⇒ (G,10) (E,8) | (D,7) (C,7) (B,5) (A,3) (I,0)

$\boxed{D,7}$

Children of D – E, A

(B, 7+2) = (B,9) – discard

(E,8) → put in frontier.

g = distance of a node from start node.

If a node exists check its value & keep lower one.

$\boxed{G,10}$	(D,7)	
	(C,7)	
$\boxed{E,8}$	(B,5)	
	(A,3)	
	(I,0)	

⇒ Replace (G,10) by (G,9)

Frontier	Explored set
(G,9)	(E,8)
	(D,7)
	(C,7)
	(B,5)
	(A,3)
	(I,0)

Children of E,

(D,9) — discard, (D,7) < (D,9)

(G,9) — Keep, (G,9) < (G,10).

⇒ | (E,8) (D,7) (C,7) (B,5) (A,3) (I,0)

$\boxed{G,9}$ G → is goal state.

Goal found, no further exploration.

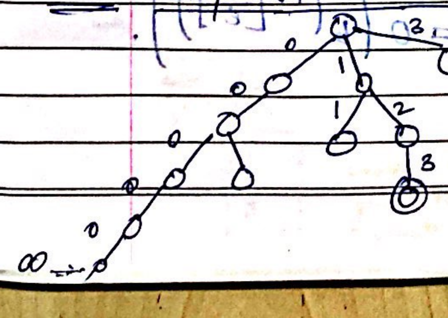
Backtrack to get the solution.

Solution: start.

G ← E ← D ← B ← I

Loop ★ Explored set is implemented using Hash Table to have minimum time complexity for search but before exploration.

★★ Uniform Cost Search Algo Analysis:-



→ In this example, Brute force will find the solution.

step-cost → cost between two nodes.

→ But, for Uniform Cost Algorithm it will get stuck as the cost of infinite path is all zeroes.

∴ The weight of the edge should be a very small when $\varepsilon > 0$ and step cost $\geq \varepsilon$ (non-negative).

①. ⇒ Uniform Cost Search Algorithm is Complete with above condition.

②. Uniform cost search is Optimal. ⇐

③. Time Complexity :—

Let C^* = optimal cost.

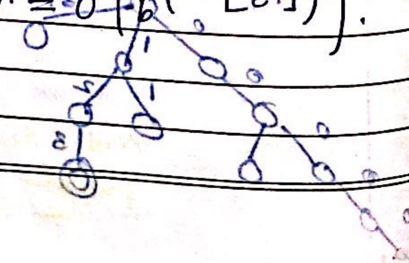
ε = minimum cost of an edge, $\varepsilon > 0$.

No. of edges traversed of weight ε to reach $C^* = C^*/\varepsilon$.

Time Complexity $= O\left(b^{\left(1+\left\lfloor \frac{C^*}{\varepsilon} \right\rfloor\right)}\right)$

$O\left(b^{\left(1+\left\lfloor \frac{C^*}{\varepsilon} \right\rfloor\right)}\right)$ → Because we check goal Node not after generation but before exploration.

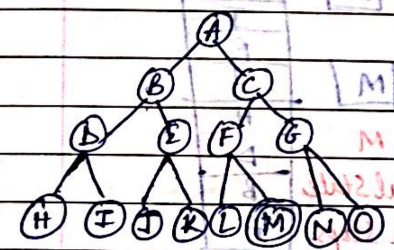
Space Complexity $= O\left(b^{\left(1+\left\lfloor \frac{C^*}{\varepsilon} \right\rfloor\right)}\right)$. ★★



Node :— State + id + pid (parent id).

Ex: pid → $\boxed{0 \;\; A \;\; 1}$ → id.

Depth First Search — A.I.:



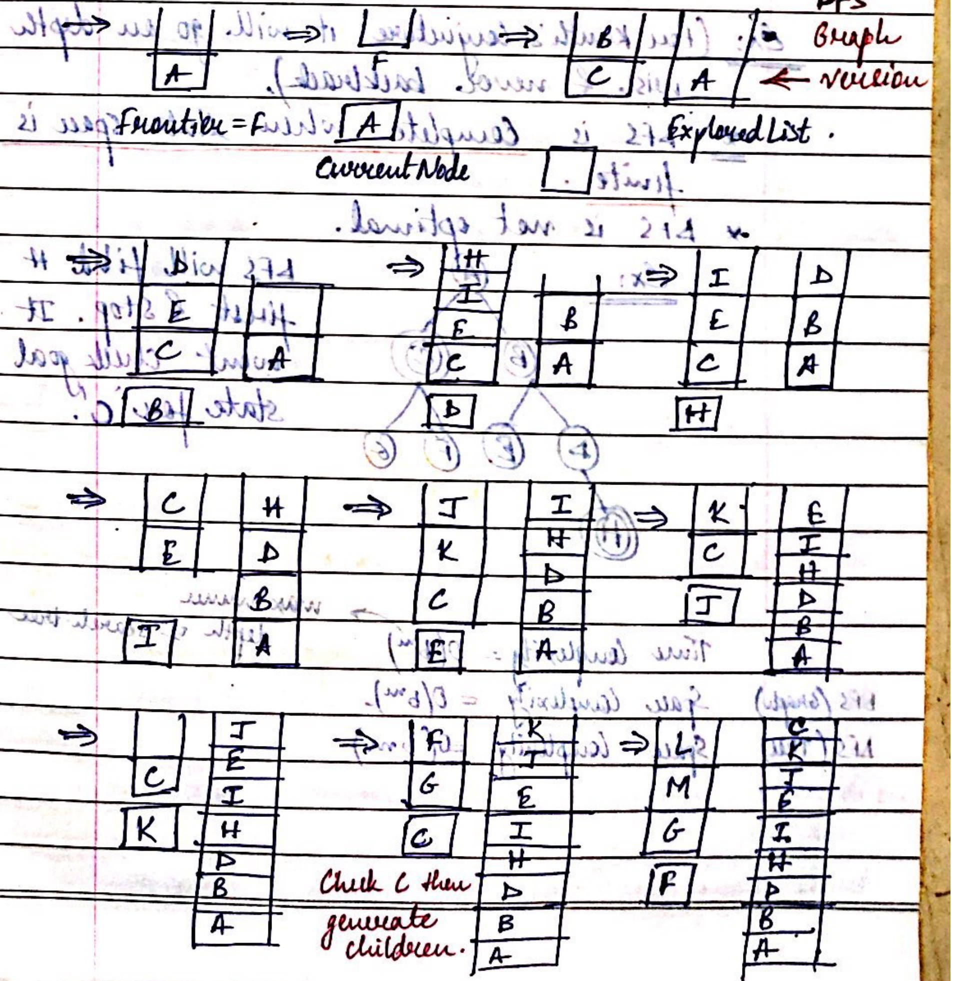
Frontier = LIFO (stack).

No extra efforts required for backtracking.

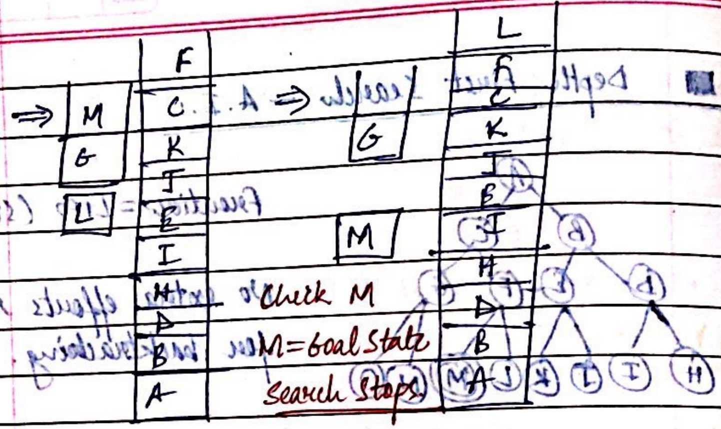
DFS Graph version

Frontier = $\boxed{A}$ Explored List.

Current Node $\boxed{}$



Check C then generate children.

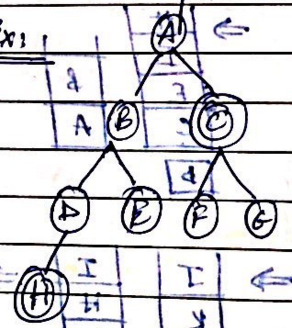


$\Rightarrow$ Check M. M = Goal State. Search Stops.

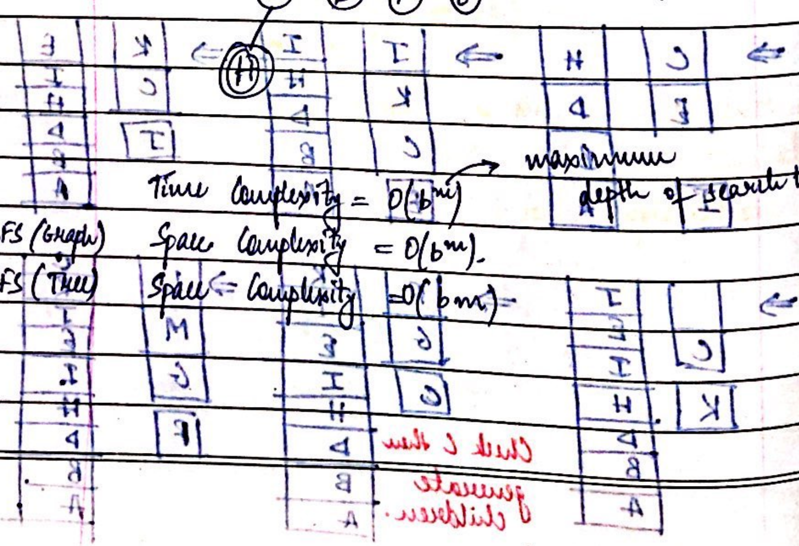
DFS Graph version — Ex: (for Knuth's conjecture) It will go on depth wise & never backtrack.

* DFS is Complete when state space is finite.
* DFS is not optimal.

Ex:



DFS will find H first & stop. It won't check goal state for C.



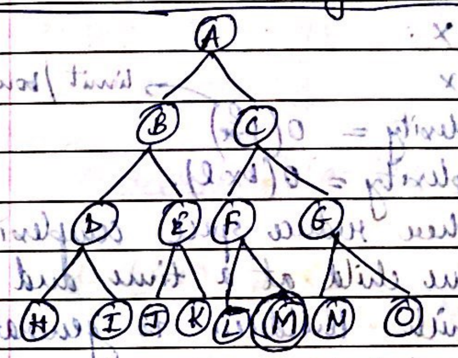
Time Complexity $= O(b^m)$ → maximum depth of search tree

DFS (Graph) Space Complexity $= O(b^m)$.

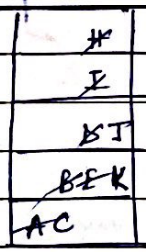
DFS (Tree) Space Complexity $= O(bm)$

2. Uniform cost search (Graph version).
3. DFS (Both version).

• DFS tree version on Binary Tree:



Frontier



$\Rightarrow O(b \cdot m)$.

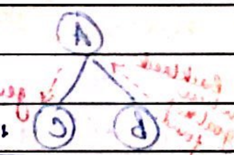
Suppose, $b = 10$, $m = 16$

BFS $\Rightarrow b^m = 10^{16} = 1$ exabyte.

DFS. $\Rightarrow b \times m = 10 \times 16 = 160$ bytes.

Current Node: ~~A B D H~~ ~~I~~ E

★• Depth Limited Search: where we make a bound to the some limit is applied regarding known solution depth. If the depth limit is reached then backtrack and find other path.



• Iterative Deepening Search: where we go on increasing the depth until we find solution and apply algorithm from start.

- Complete: ✗
- Optimal: ✗
- Time Complexity = $O(b^l)$ ⟶ limit / bound.
- Space Complexity = $O(b \times l)$.

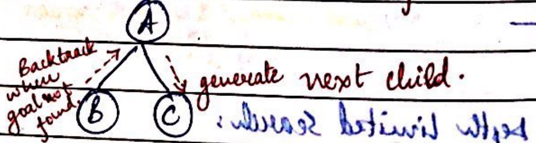
To further reduce space complexity, generate one child at a time and keep track which children to generate next.

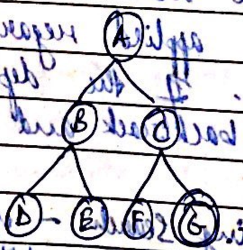
$\therefore O(m)$ will be space complexity (m = depth of tree).

- **Iterative Deepening Search:**

$l=0$ (A). — increase depth when goal not found.

$l=1$.





---

whatever studied till here Algorithms are
↳ Blind Search Algorithms.
↳ Non-Heuristic.

3. **Space Complexity:** $O(b \times d)$ (b = branching factor, d = depth of tree).

4. **Time Complexity:**

A node is explored $b^k \times (d-k)$ times.

$\therefore O(b^d) = b(d-1) + b^2(d-2) + \dots + b^d(1)$.

If $b=10$, $d=5$.

Number of Nodes generated in BFS is,

$N(BFS) = 1+10+100+1000+10000+\cancel{100000}$
$= 1,11,111.$

Substitute in formula,

$= 5 + 10^1(4) + 10^2(3) + 10^3 \times 2 + 10^4 \times 1.$
$= 12,345$

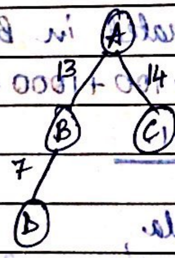
When, $12,345 / 11,111 = 10\%$ more than normal DFS.

Graph Version $= O(b^m)$ · Tree Version – No.

Criterion	Breadth First Search	Uniform Cost	Depth First	Depth Limited	Iterative Deepening
Complete	Yes[a]	Yes[a,c]	* Yes – Graph version	No	Yes[a]
Optimal	Yes[b]	Yes	No	No	Yes
Space	$O(b^d)$	$O(b^{(1+\lfloor C^*/\epsilon \rfloor)})$	$O(bm)$ T.V.	$O(bl)$	$O(b \times d)$
Time	$O(b^d)$	$O(b^{(1+\lfloor C^*/\epsilon \rfloor)})$	$O(b^m)$	$O(b^l)$	$O(b^d)$

a: complete if b (branching factor) is finite.
b: if cost is non-decreasing function of depth.
c: is complete if step cost $\geq \epsilon$ (non-neg).

- All of the above searches are uninformed searches meaning these are generalized algorithms & not for any particular data / information or question. Also called as Blind Search Algorithms or Non-Heuristic Search Algorithm.



$g(B) = 13$

$g(C) = 14$

$g(D) = 13+7 = \underline{20}$.

★ DFS Reviews:

1. DFS Tree Version: [Graph version = $O(b^m)$]

(i) We don't put a check for repeated nodes. So a problem like 8 puzzle problem, the algo will go on applying operators & going depth wise. ∴ Incomplete.

(ii) Not Optimal.

(iii) Time Complexity $= O(b^m)$, b = branching factor, m = depth of goal node (shallowest)

(iv) Space Complexity $= O(bm)$ — if we only store frontier.

Space Complexity can be reduced to $O(m)$ if we generate only one child at a time for a node.

In this we need to keep track of the next child to be generated for a node.

- DFS Graph Version:

(i) Complete for finite state space → we need to store frontier as well as explored nodes. ∴ All nodes are stored to get the goal node.

(ii) Not Optimal

(iii) Time Complexity $= O(b^m)$

(iv) Space Complexity $= O(b^m)$

★★★ Heuristic Function: [Always keep underestimated value]

It is an estimate that gives value of a node from a destination / goal node.

$h(G) = 0$ → Goal Node.

h: State Space → Positive integers + {0}.

Ex: 8 Puzzle Problem, Distance between cities.

1	2	3
5	4	6
7	0	8

I = (above)

A =

1	2	3
5	4	6
0	7	8

B =

1	2	3
5	0	6
7	4	8

C =

1	2	3
5	4	6
7	8	0

Define a Heuristic function for a node of a 8 puzzle problem.

$h(N)$ = No. of mismatch tiles with respect to Goal state.

Now,

$H(I) =$

1	2	3
5	4	6
7	0	8

1	2	3
4	5	6
7	8	0

$= 0+0+0+1+1+0+0+1+1$. Overestimated value as solution comes in 3.

$= 4.$

But when not considering 0 as a tile as we want to calculate underestimated value

$= 0+0+0+1+1+0+0+1+0$

$= \underline{3}$

- The value should be non-overestimating, either exact estimating or underestimating.
- Greatest non-overestimation is best heuristic function.

Worst	Branching factor of 8-Puzzle Problem = 4
Average	" = 3.

Average depth of solution node = $\underline{\underline{22}}$

- 8 - Puzzle Problem:

Avg $b = 3$.

Avg goal node depth = 22.

Using BFS - Tree version $\Rightarrow 3^{22} = 31,381,059,603$

$\approx 3.1 \times 10^{10}$ nodes will be generated

State space of 8 puzzle problem = 9!

Using BFS ~~Tree~~ Graph Version — 9! nodes generated almost $\Rightarrow 9! = 3,62,880$ (Approx).

- This is because graph version takes care of repeated nodes.

$G_1 =$

1	2	3
4	5	6
7	8	

$G_2 =$

1	2	3
8		4
7	6	5

Parity = 0 / even.

$1 \to 0$ $5 \to 0$

$2 \to 0$ $6 \to 0$

$3 \to 0$ $7 \to 0$

$4 \to 0$ $8 \to 0$

Odd Parity.

$1 \to 0$ $4 \to 0$

$2 \to 0$ $7 \to 2$

$3 \to 0$ $6 \to 1$

$8 \to 4$ $5 \to 0$.

$4+2+1 = \underline{7}$.

- If given G has odd parity, then all the possible states generated from it will have odd parity and vice versa.

∴ State space gets halved.

State space for BFS Graph $= \frac{9!}{2} = 181,440$.

This is still huge.

Hence, we try to find a Heuristic Solution.

Heuristic Function – 2

$H_2(A)$ = Sum of Manhattan distance from Goal State.

$H_1(A)$ = No. of Mismatched tiles.

Ex: $H_2(A)$. A =

2	1	8
4	5	7
3	6	

1 → 1   5 → 0

2 → 1   6 → 2

3 → 4   7 → 3

4 → 0   8 → 3.

$H_2(A) = 1+1+4+0+0+2+3+3 = 14$

$H_1(A) = 2$.

$\boxed{H_2(G) = 0}$, G = Goal Node.

$\boxed{\text{H2 is always better than H1. Greatest Non-Overestimation is best Heuristic function.}}$

~~***~~ GREEDY BEST FIRST SEARCH ALGORITHM:

*** Greedy Best First Search Algorithm:

G =

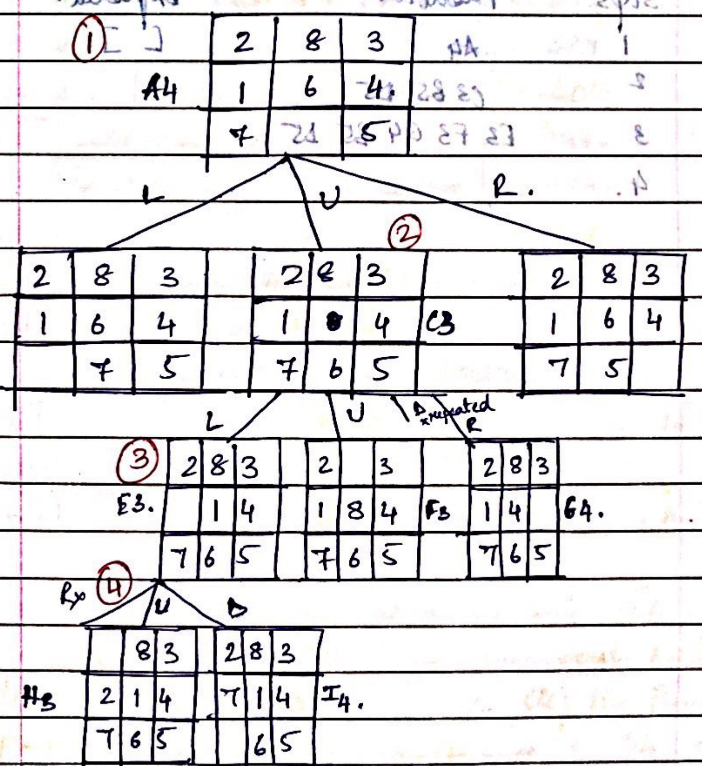
1	2	3
8		4
7	6	5

Initial Node := A =

2	8	3
1	6	4
7		5

defined as A4 → Heuristic Function(1)

• Operations: Left (L), Right (R), Up (U), Down (D)



	8	3
2	1	4
7	6	5

R D

8		3
2	1	4
7	6	5

J3.

Steps	Frontier	Explored
1	A4	[ ]
2	C3 B5 D5	A4
3	E3 F3 G4 B5 D5	
4.		

node name. A 366 → H() value. Frontier – Priority queue.

A Greedy Best First Search (Graph Version).

→ Distance from Goal Node.

City		Distance	City	Distance
Arad	A	366	L	244
Bucharest.	B	0	M	241
	C	160	N	234
	D	242	O	380
	E	161	P	100
	F	176	R	193
	G	77	S	253.
	H	151	T	329
	I	226	U	80
			Y	199
			Z	374.

Frontier: [A 366] ① Explored

[S T Z]. [A].

140 ②

[Z374] [S253] [T329]

[T Z O F R]. [A, S].

99 ③

[O380] [F176] [R193]

[T Z O F R B]. [A, S, F].

211

[B0]

E

We check for Goal state when we remove [A, S, F, B]

140+99+211 = 450 Goal Node Found

Complete. ∴ Not Optimal. ∴ Stop the Process.

if state space is finite.

From Another Path A – S – R – P – B = 418.

Frontier - Priority queue. [illegible] value.

- Tree version of Greedy Best First Search is not complete as it can enter an infinite loop and therefore not optimal.

Space & Time Complexity $= O(b^n)$ → no. of nodes in level.

~~A*~~ A* Search Algorithm:

- So now, we propose a new Heuristic function to get an optimal solution.

distance from initial node. → $\boxed{f(N) = g(N) + h(N)}$ (g: Uniform cost search; h: function applied in previous example.)

Frontier

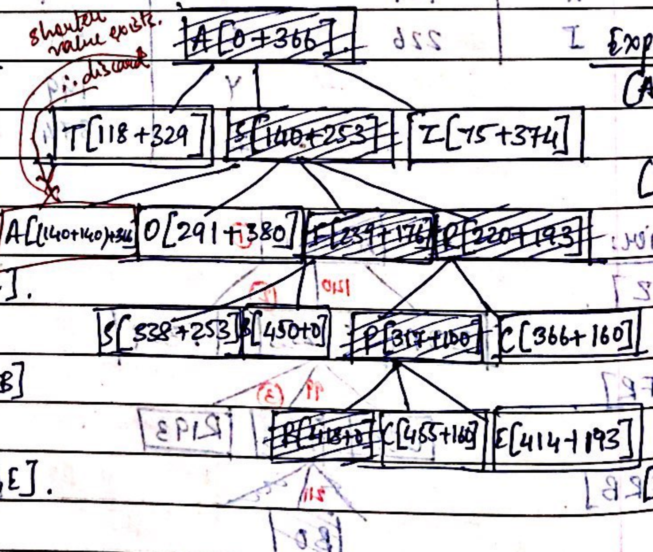
[S, T, Z]

[T, Z, O, F, R]

[T, Z, O, F, R, P, C].

[T, Z, O, F, R, P, C, B]

[T, Z, O, F, R, P, C, E].



Explored.

[A]

[A, S].

[A, S, R]

[A, S, R, F]

[A, S, R, F, P]

We check for Goal state when we remove the node from frontier for processing.

↳ A* is a special case of BFS. If $h(N)=0$ then it is Best Breadth First Search.

↳ A* is Greedy Best First Search if $G(N)=0$. → Special case of A*

↳ A* is Uniform Cost Search if $H(N)=0$.

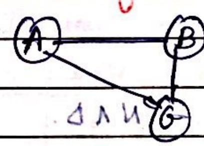
- **Admissible Heuristic Function:**

It is Admissible if the Heuristic function is Non-Overestimating.

- A* Tree version — the solution will be optimal if heuristic function is admissible.

- A* Graph version — Solution will be optimal if Heuristic function is consistent.

Consistent Heuristic Function:



$\Delta\ H(A) \leq d(A,B) + H(B)$

d → distance of A to B.

Prolog & Lisp.

UNIT – 3

Predicate and Propositional Logic.

- Classical A.I.:
  - ↳ Designed on the basis of symbols not numbers.
  - ↳ Called Expert systems.
  - ↳ Built using languages Prolog & Lisp.
  - ↳ Not much intelligence achieved.

- Today's AI (2020):
  - Designed using Numbers
  - & some use hybrid approach using numbers & symbols.

- Propositional logic: has only 2 values – True & False

  It is Night – N
  It is Dark – D
  It is Night & dark – $N \wedge D$.
  I like Apple – A.

- Connectives:

① $\wedge$, conjunction.

A	B	$A \wedge B$
F	F	F
F	T	F
T	F	F
T	T	T

② $\vee$, Disjunction.

A	B	$A \vee B$
F	F	F
F	T	T
T	F	T
T	T	T

③ $\sim$, $\neg$, Negation.

A	$\sim A$
T	F
F	T

④ $\rightarrow$, Implies. If A is true then B is true.

A	B	$A \rightarrow B$	$\neg A \vee B$
F	F	T	T
F	T	T	T
T	F	F	F
T	T	T	T

Ex: If you work hard, you will get pass.

WH	P	$WH \rightarrow P$	$\neg WH \vee P$
F	F	T	T
F	T	T	T
T	F	F	F
T	T	T	T

⑤ $\Longleftrightarrow$, Double Implication (iff → if and only if.)
A implies and implied by B.

WH	P	$WH \leftrightarrow P$	$(WH \rightarrow P) \wedge (P \rightarrow WH)$
F	F	T	T
F	T	F	F
T	F	F	F
T	T	T	T

A.I → Learning, Logical based
M.L. → Logical based.

$A \leftrightarrow B \equiv (\neg A \vee B) \wedge (\neg B \vee A)$.

- **Precedence:**
  Higher to lower,
  $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$

- **Literals:**
  Positive → A.
  Negative → $\neg A$.

- **Tautology:**

A	$A \vee (\neg A)$
F	T
T	T

Always True.

- **Contradiction:**

A	$A \wedge (\neg A)$
F	F
T	F

Always False.

- $\boxed{\perp \text{ (falsum)}}$ (contradiction) ⟺

★ **Deduction:**
$\{A_1, A_2, \ldots, A_n\} \rightarrow C$

From the set of proposition ($\{A_1, A_2, \ldots, A_n\}$) we have deduced C.

$\boxed{\begin{array}{l} A \wedge B \Rightarrow \text{True (suppose).} \\ A \text{ will always be true.} \end{array}}$ Deduction.

★ **Inference Rules:**

$A \vdash B$.

OR $\dfrac{A}{B}$

1. $\wedge$ Introduction (AND-Introduction).
   A: Sachin is Hard Working.
   B: Sachin is intelligent.
   } given.

$\dfrac{A, B}{A \wedge B}$.

2. AND Elimination:
   Ex: A: Sachin is Hard Working.
   B: Sachin is intelligent.

$\dfrac{A \wedge B}{A}$, also, $\dfrac{A \wedge B}{B}$ → deductions.

3. OR-Introduction:

$\dfrac{A}{A \vee B}$ → If one is true, the whole is true.

4. Implication - Elimination:
   Ex: $A \rightarrow B$ A: You worked hard
   $\Rightarrow A \rightarrow B$ B: You got passed.

Literal = atomic symbol or negation of an atomic symbol.

5. Reduction Ad Absurdum: (Proof by Contradiction)

Given: A.

Assume: ¬A

From ¬A lead to a ⊥ (contradiction).

⇒ assumption is wrong
⇒ A is true. } deduction

6. → Introduction:

Given: A.

and we get C while deduction

i.e, A

C

A → C.

7. ¬¬ Elimination:

$$\frac{\neg\neg A}{A}$$

** Resolution in Propositional Logic:

Given a statement, convert into Conjunctive Normal Form. (CNF).

$A_1 \wedge A_2 \wedge A_3 \wedge \ldots \wedge A_n$.

where, $A_i = B_1 \vee B_2 \vee \ldots B_m$.

where, each B_i is a Literal.

- To convert into CNF form we use some equivalences

1. $A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A)$.
2. $A \rightarrow B \equiv \neg A \vee B$.
3. $\neg(A \wedge B) \equiv (\neg A) \vee (\neg B)$. } Demorgan's Law.
4. $\neg(A \vee B) \equiv (\neg A) \wedge (\neg B)$
5. $\neg\neg A = A$.
6. $A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$.
7. $A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$.

$\wedge, \vee \equiv$ Conjuctive, $A \wedge B \equiv B \wedge A$.

$\wedge, \vee \equiv$ Associative, $A \wedge (B \wedge C) = (A \wedge B) \wedge C$.

Ex: 1 Convert into CNF form:

$(A \rightarrow B) \rightarrow C$

$\equiv (\neg A \vee B) \rightarrow C$

$\equiv \neg(\neg A \vee B) \vee C$.

$\equiv (A \wedge \neg B) \vee C$.

$= \cancel{(A \wedge \neg B) \vee (C)}$.

$\equiv \cancel{A} \wedge (\neg B \vee C)$

$\equiv \boxed{(A \vee C) \wedge (\neg B \vee C)}$.

Ex2: Convert into CNF form:

$A \leftrightarrow (B \wedge C)$.

$\equiv (A \rightarrow (B \wedge C)) \wedge ((B \wedge C) \rightarrow A)$

$\equiv (\neg A \vee (B \wedge C)) \wedge (\neg(B \wedge C) \vee A)$.

$\equiv ((\neg A \vee B) \wedge (\neg A \vee C)) \wedge ((\neg B) \vee (\neg C) \vee A)$.

Also represented as,

$\{(\neg A, B), (\neg A, C), (\neg B, A, \neg C)\}$.

★ The Resolution Rule:

Ex 1: Given: $(A \vee B)$ and $(\neg B \vee C)$ = Deduce: $A \vee C$.

$A \vee B \equiv \neg A \rightarrow C$ — A = work hard.

$\neg B \vee C \equiv B \rightarrow C$. — B = fail.

C = sad.

$\neg A \rightarrow B$ and $B \rightarrow C$

$\therefore \neg A \rightarrow C$

i.e., $\underline{A \vee C}$

class-1 class-2

$\boxed{(A, B)}$, $\boxed{(\neg B, C)}$ ⟶ 2 class resolved into 1

(A, C)

Ex 2: $\{(A, B, C), D, (\neg A, D, E), (\neg D, F)\}$.

$\{(\cancel{A}, B, C, \cancel{\neg A}, D, E), D, (\neg D, F)\}$.

$\{(B, C, D, E), (\cancel{D}, \cancel{\neg D}, F)\}$.

$\{(B, C, D, E), F\}$.

OR

$\{(B, C, D, E), D, (\neg D, F)\}$

$\{(B, C, E, F), D\}$.

Ex 3: - Resolve: $\{(A \vee B) \wedge (\neg A \vee C) \wedge (\neg B \vee C)\}$ $\boxed{\models}$ C ⟶ resolvant.

$\Rightarrow \{(A, B), (\neg A, C), (\neg B, C)\}$.

$\Rightarrow \{(B, C), (\neg B, C)\}$

$\Rightarrow \{C\}$

~~AA~~ ← Resolution Refutation (Proof by Contradiction).

$\{(\neg A, B), (\neg A, \neg B, C), A, \neg C\} \models \perp$.

$\Rightarrow \{(\neg A, C), A, \neg C\}$.

we get $\Rightarrow \{C, \neg C\}$.

$\Rightarrow \perp$ (failure) as this is $\wedge$ operation.

But,

$\{(\neg A, B), (\neg A, \neg B, C), A, \neg C\} \not\models \neg A$.

$\Rightarrow \{(\neg A, B), \neg B\}$

$\Rightarrow \{\underline{\neg A}\}$.

- If we get falsehood, we say that the class is $\underline{\underline{\text{inconsistent}}}$.

Q. Consider the following sentences:-

1. If it rains and I do not have an umbrella then I will get wet.
2. It is raining and I do not have umbrella.

<u>To Prove:</u> Therefore, I will get wet.

<u>Proof:-</u>

Convert the staments to propositional logic

$R \to$ Rains ; $U \to$ Umbrella ; $W \to$ wet.

1. $R \wedge (\neg U) \to W$.
2. $R \wedge (\neg U)$
3. $\therefore W$.

We have to prove that "I will get wet", i.e., W. as it's $\wedge$ operation.

Now let us assume that "I will not get wet", i.e., $\neg W$.

which implies;

1. $R \wedge (\neg U) \to W$
2. $R \wedge (\neg U)$
3. $\neg W$.

1. $[R \wedge (\neg U)] \to W$.

$\Rightarrow \neg(R \wedge \neg U) \vee W$.

$\Rightarrow \neg R \vee U \vee W$.

2. $R \wedge (\neg U)$

$\Rightarrow R \wedge (\neg U)$

3. $\neg W$.

$\therefore (\neg R \vee U \vee W) \wedge (R) \wedge (\neg U) \wedge (\neg W)$.

$\Rightarrow \{(\neg R, U, W), R, \neg U, \neg W\}$.

$\Rightarrow \{(U, W), \neg U, \neg W\}$.

$\Rightarrow \underline{\bot}$

- This proves that our assumption is wrong, i.e., $\neg W$ is not true.

$\Rightarrow W$ is true.

$\therefore$ "I will get wet" is true.

<u>Hence Proved</u> (there's no way to get false value)

<u>Q</u>

1. $A \to B$.
2. $B \to C$
3. $C \to D$
4. $D \to E \vee F$

$\therefore A \to F$.

$\longrightarrow$ Let us assume that $\neg(A \to F)$

1. $A \to B \equiv \neg A \vee B$.
2. $B \to C \equiv \neg B \vee C$.

3. $C \to D \equiv \neg C \vee D$.

4. $D \to \cancel{E \wedge F}\ E \vee F \equiv \neg D \vee (E \vee F)$.

$$\neg(A \to F) = \neg(\neg A \vee F) = (A \vee \neg F).$$

$\therefore (\neg A \vee B) \wedge (\neg B \vee C) \wedge (\neg C \vee D) \wedge (\neg D \vee E \vee F) \wedge (A \vee \neg F)$

$\Rightarrow \{(\neg A, B), (\neg B, C), (\neg C, D), (\neg D, E, F), (A), \neg F)\}$.

$\Rightarrow \{(\neg A, C), (\neg C, E, F), (A), \neg F)\}$.

$\Rightarrow \{(\neg A, E, F), (A) \neg F)\}$

$\Rightarrow \{(E)\}$.

Here we dont get falsum. (There's no way to get falsum which)

$\therefore$ We cannot prove $A \to F$, i.e., it is a wrong proposition.

Q. 1. $A \to B$.
2. $B \to C$
3. $C \to D$
4. $D \to E \wedge F$.

$\therefore\ A \to F$.

Assume $\neg(A \to F)$ is true,

$\neg(A \to F) \equiv \neg(\neg A \vee F) \equiv (A \wedge \neg F)$

$\to (\neg A \vee B)\ (\neg B \vee C)\ (\neg C \vee D)\ (\neg D \vee E) \wedge$
$(\neg D \vee F), (A, \neg F) \longrightarrow \neg(\neg A \vee F)$.

did not use this clause

$\Rightarrow \{(\neg A, B), (\neg B, C), (\neg C, D), \boxed{(\neg D, E)}, (\neg D, F), A, \neg F\}$

$\neg D$

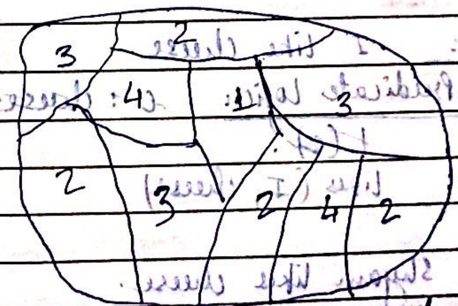
$C \qquad \neg C$

$\perp$

Here we get falsum, but we didnt use the clause $(\neg D, E)$, but then also $A \to F$ is true.

Hence Proved.

$\bigstar$



$\hookrightarrow$ outer region.

This is called 4 colour conjecture.

It states that any given map can be coloured with a minimum of (4) Four colour.

Ex: $R \wedge \neg U \rightarrow W$

$R \wedge \neg U_{SHYAM} \rightarrow W_{SHYAM}$.

$R \wedge \neg U_{HALE} \rightarrow U_{HALE}$.

- Generalized Form:

If it is raining and anyone who do not have umbrella got wet.

- Predicate Logic:

$R \wedge \neg U(x) \rightarrow W(x)$.

i.e.,

$(\forall x)\ R \wedge \neg U(x) \rightarrow W(x)$.

- Propositional logic is a special case of Predicate Logic.

- Ex: I like cheese.

Predicate Logic: $L(c)$.

Likes (I, cheese)

Ex: Shyam likes cheese.

PL: Likes (Shyam, cheese) ordered arguments.

Predicate (verb generally)

- Universal Quantifier. $\forall$ - For All
- Existential Quantifier. $\exists$ - There Exists.

Ex: Cheese likes Shyam.

PL: Likes (Cheese, Shyam).

Ex: Cheese liked by Shyam.

PL: Likedby (Cheese, Shyam).

Ex: Everyone likes cheese.

PL: ① ~~Likes (x, cheese)~~ $\forall x \in$ Person.

② $(\forall x)$ Person $(x) \rightarrow$ Likes $(x$, cheese$)$.

Ex: There is someone who have written LET_US_C.

PL: ① $(\exists x)$ Person$(x) \rightarrow$ ~~written~~ write $(x$, LET_US_C$)$.

② $(\exists x)$ write $(x$, LET_US_C$)$.

Ex: If everyone likes cheese, then there is someone who likes cheese.

$(\forall x)$ Likes $(x$, cheese$) \rightarrow (\exists x)$ Likes $(x$, cheese$)$

Ex: If someone likes cheese, then everyone likes cheese.

$(\exists x)$ likes $(x$, cheese$) \rightarrow (\forall x)$ likes $(x$, cheese$)$

Ex: Everyone likes something

$(\forall x)(\exists y)$ likes (x, y)

$\boxed{\text{likes } (x, \exists y)}$ wrong notation

** Parameter of Predicate can be — variables, Constants, Functions.

Predicates can have 0 or more parameters or arguments.

Ex: Everyone likes Shyam.
PL: $(\forall x)$ Likes(x, Shyam).

Ex: Everyone likes themselves.
PL: $(\forall x)$ Likes(x, x)
Shyam likes Shyam, A likes A, etc.

Ex: Someone likes everyone.
PL: $(\exists x)(\forall y)$ Likes(x, y).

Ex: Everyone likes cheese.
PL: $(\forall x)$ Likes(x, cheese)
Person(x) | variable

Ex: There is some person who doesn't like cheese.
Form 1: ① $\neg[(\forall x)\ \text{Person}(x) \rightarrow \text{Likes}(x, \text{cheese})]$
Form 2: ② $(\exists x)\ \text{Person}(x) \rightarrow \neg\ \text{Likes}(x, \text{cheese})$.

$$\boxed{\begin{aligned}\neg(\forall x)\ P(x) &\equiv (\exists x)\ \neg P(x)\\ \neg(\exists x)\ P(x) &\equiv (\forall x)\ \neg P(x)\end{aligned}}$$

$\neg(\exists x\ \text{Likes}(x, \text{cheese})) \equiv (\forall x)\ \neg \text{Likes}(x, \text{cheese})$.

Ex: John's Mother likes cheese.
PL: Likes(mother(John), cheese).
function ~~constant~~
wrong notation

★ Terminology:
1. Bound variable 2. Free variable.

If a variable is either Universal Quantifier or an Existential quantifier, then it is a Bound variable.

Ex: $\forall x\ (P(x, y))$
x is a bound variable,
y is a free variable

If a variable is neither Universal Quantifier nor an Existential Quantifier, then it is a Free variable.

Scope of Quantifier is important.
1. $(\forall x\ (P(x, y))) \rightarrow Z(x)$
2. $\forall x\ ((P(x, y)) \rightarrow Z(x))$
same rules apply for OR ($\vee$).
both mean different things.

① $\forall x\ P(x, y)$
② $\forall x\ P(x, z)$
③ $\forall y\ P(y, z)$

① ≡ ② ≡ ③ (as x, y, z are dummy variables

★ PRENEX NORMAL FORM:

- English Language Statement

  ↓

  Predicate Logic

  ↓

  Prenex Normal Form (CNF + Quantifiers on left side).

★ Additional Rules: – Used to bring quantifiers to left side.

1. $\neg(\forall x)\, A(x) \equiv (\exists x)\, (\neg A(x))$.
2. $\neg(\exists x)\, A(x) \equiv (\forall x)\, (\neg A(x))$.
3. $(\forall x)\, A(x) \wedge B \equiv (\forall x)\, (A(x) \wedge B)$
4. $(\forall x)\, A(x) \vee B \equiv (\forall x)\, (A(x) \vee B)$
5. $(\exists x)\, (A(x)) \wedge B \equiv (\exists x)\, (A(x) \wedge B)$
6. $(\exists x)\, (A(x)) \vee B \equiv (\exists x)\, (A(x) \vee B)$

Scope of quantifier is important. * B does not depend on x

7. $(\forall x)\, A(x) \wedge (\forall y)\, B(y) \equiv (\forall x)(\forall y)(A(x) \wedge B(y))$
8. $(\forall x)\, A(x) \wedge (\exists y)\, B(y) \equiv (\forall x)(\exists y)\, (A(x) \wedge B(y))$
9. $(\exists x)\, A(x) \wedge (\forall y)\, B(y) \equiv (\exists x)(\forall y)\, (A(x) \wedge B(y))$
10. $(\exists x)\, A(x) \wedge (\exists y)\, (B(y)) \equiv (\exists x)(\exists y)\, (A(x) \wedge B(y))$.

Same rules apply for OR ($\vee$).

both mean different things.

Q. Convert to Prenex Normal Form:

Ex: 1. $(\forall x)\, (E(x) \rightarrow G(x)) \rightarrow (\exists y)\, (E(y) \wedge G(y))$.

$\equiv (\forall x)\, (\neg E(x) \vee G(x)) \rightarrow (\exists y)\, (E(y) \wedge G(y))$

$\equiv \neg((\forall x)\, (\neg E(x) \vee G(x))) \vee ((\exists y)(E(y) \wedge G(y)))$

$\equiv (\exists x)\, \neg(\neg E(x) \vee G(x)) \vee (\forall x)\, \neg(E(y) \wedge G(y))$ ✗

This does not depend on y.

$\equiv \boxed{(\exists x)\, (E(x) \wedge \neg G(x))} \vee ((\exists y)\, (E(y) \wedge G(y)))$

$\therefore (\exists y)$ can be brought to the front.

$\equiv (\exists x)(\exists y)\, [(E(x) \wedge \neg G(x)) \vee (E(y) \wedge G(y))]$

$\equiv (\exists x)(\exists y)\, [(E(x) \vee E(y)) \wedge (E(x) \vee G(y)) \wedge (\neg G(x) \vee E(y)) \wedge (\neg G(x) \vee G(y))]$

- Step 2: Getting Rid of Existential Quantifiers ($\exists$) from Prenex Normal Form – also known as $\boxed{\text{Skolemization}}$

$\exists x\ P(x) \Rightarrow P(c)$ where c is that x, that satisfies $(\exists x)$.

There may be more than one instances of x.

$(\exists x)\, \{P(x)\} \neq P(c)$

but, $(\exists x)\, \{P(x)\}$ ✓

Condition: 'c' shouldn't be already present in the given sentence.

Ex: $(\exists x)$ Work done (Shyam $\vee$ x)

Doing Skolemization:–

$\Rightarrow$ Work done (Shyam $\vee$ b).

(b is constant and instance of x).

- Skolemize the sentence:

Ex: Everyone likes someone.

$\Rightarrow (\forall x)(\exists y)\ Likes(x,y)$.

Skolemize,

$(\forall x)\ Likes(x,c)$. ✗ means everyone likes Skolem constant only 1 person.

$(\forall x)\ Likes(x, \underbrace{f(x)}_{\text{Skolem Function}})$ ✓

Ex: There is someone who likes everyone.

$\Rightarrow (\exists x)(\forall y)\ Likes(x,y)$.

Skolemized: $(\forall y)\ Likes(c,y)$. Skolem constant.

Ex: $(\forall x)(\forall y)(\exists z)\ P(x,y,z)$.

$\Rightarrow (\forall x)(\forall y)\ P(x,y,f(x,y))$

Ex: $(\forall w)(\forall x)(\exists y)(\forall z)\ (P(x) \wedge Q(w,y,z))$

$\Rightarrow (\forall w)(\forall x)(\forall z)\ (P(x) \wedge Q(w, f(x,w), z))$

Steps:

English statement
↓ Step 1
Predicate Logic
↓ Step 2
Prenex Normal Form
↓ Step 3
Skolem Normal Form (Skolemization)
↓ Step 4
Resolution.

- Now that Skolem Normal Form contains only Universal quantifiers (variables) we can just remove these quantifiers.

Ex: $(\forall w)(\forall x)(\forall z)\ (P(x) \wedge Q(w, f(w,x), z))$

$\Downarrow$

$P(x) \wedge Q(w, f(w,x), z)$

- Steps:
  2. Prenex Normal Form
  3. Skolem Normal Form
  4. Clause Form.
  5. Unification / substitution

Ex: $A \vee B \quad \neg A \vee C$

Ex: $(\forall x)\ A(x) \vee B(x) \quad \neg A(y) \vee C(y)$.

Unification or substitution: $\theta = \{x/y\}$ $B\{y\} \vee C\{y\}$.

Ex 1: Everyone who loves all animals is loved by someone.

$(\forall x)(\forall y)\ loves(x,y) \rightarrow$

$$\left[(\forall x)(\forall y)\left(\text{Animal}(y) \rightarrow \text{Loves}(x,y)\right)\right] \longrightarrow (\exists z)\ \text{Loves}(z,x)\Big]$$

<u>Convert to Prenex Normal Form:</u>

$$= [(\forall x)[(\forall y)(\neg\text{Animal}(y) \vee \text{Loves}(x,y))) \rightarrow (\exists z)\ \text{Loves}(z,x)]].$$

$$= [(\forall x)\neg[(\forall y)(\neg\text{Animal}(y) \vee \text{Loves}(x,y))) \vee (\exists z)\ \text{Loves}(z,x)]].$$

$$= (\forall x)\ (\exists y)\ \neg(\neg\text{Animal}(y) \vee \text{Loves}(x,y)) \vee (\exists z)\ \text{Loves}(z,x)).$$

$$= (\forall x)(\exists y)\ (\text{Animal}(y) \wedge \neg\text{Loves}(x,y) \vee (\exists z)\ \text{Loves}(z,x))$$

$(A \wedge B) \vee C = (A \vee C) \wedge (B \vee C)$.

$$= (\forall x)(\exists y)(\exists z)\left((\text{Animal}(y) \vee \text{Loves}(z,x)) \wedge (\neg\text{Loves}(x,y) \vee \text{Loves}(z,x))\right)$$

$\theta = \{x|y, z|\ldots\}$ substitution

<u>Convert the sentence to Skolemized Form:</u>

$$= (\forall x)\left(\text{Animal}(f(x)) \vee \text{Loves}(g(x), x)\right) \wedge \left(\neg\text{Loves}(x, f(x)) \vee \text{Loves}(g(x), x)\right).$$

Universal Quantifier $\longrightarrow$ use Implication ($\rightarrow$)

Existential Quantifier $\rightarrow$ use AND ($\wedge$).

- <u>Convert into Clause Form:</u>

$$\underbrace{[\text{Animal}(f(x)) \vee \text{Loves}(g(x), x)]}_{A} \wedge \underbrace{[\neg\text{Loves}(x, f(x)) \vee \text{Loves}(g(x), x)]}_{B}.$$

<u>Ex 2:</u> Anyone who kills an animal is loved by no one.

<u>Soln:</u> ~~$(\forall y)$~~

$\forall x$ kills an animal.

$(\exists y)\ (\text{Animal}(y) \wedge \text{Kills}(x,y))$

Hence x is loved by no one.

$(\forall z)\ \neg\text{Loves}(z,x)$.

Sentence is:

$$(\forall x)\left[(\exists y)\ \text{Animal}(y) \wedge \text{Kills}(x,y) \longrightarrow (\forall z)\ \neg\text{Loves}(z,x)\right].$$

<u>Ex 3:</u> Jack loves all animals.

$(\forall y)\ \text{Animal}(y) \longrightarrow \text{Loves}(\text{Jack}, y)$.

<u>Ex 3b:</u> Jack loves some animals.

$(\exists y)\ \text{Animal}(y) \wedge \text{Loves}(\text{Jack}, y)$.

<u>Ex 4:</u> Either Jack or curiosity killed the cat who is named Tuna.

Soln: $\text{Kills}(\text{Jack}, \text{Tuna}) \vee \text{Kills}(\text{curiosity}, \text{Tuna})$

OR $\text{Kills}(\text{Jack}, \text{Cat}(\text{Tuna})) \vee \text{Kills}(\text{curiosity}, \text{Cat}(\text{Tuna}))$

↳ Predicates (Prenex Normal form)

~~Ex 5: Did curiosity killed the cat?~~

Ex 5: Every cat is an animal.

$(\forall x)\ Cat(x) \rightarrow Animal(x).$

Ex 6: Tuna is cat.

$Cat(Tuna).$

Ex 7: Did curiosity killed the cat?

Assume: Curiosity did not kill the cat.

$7:\ \neg Kills(curiosity, Tuna).$

① ∧ ② ∧ ③ ∧ ④ ∧ ⑤ ∧ ⑥ ∧ ⑦

but first convert all these into clause form.

Ex 2: to Prenex Normal Form:

$(\forall x)[(\exists y)\ Animal(y) \wedge Kills(x,y)) \rightarrow (\forall z)\ \neg Loves(z,x)]$

$\Rightarrow (\forall x)[\neg((\exists y)\ Animal(y) \wedge Kills(x,y)) \vee (\forall z) \wedge \neg Loves(z,x)]$

$\Rightarrow (\forall x)[(\forall y)\ \neg Animal(y) \vee \neg Kills(x,y) \vee (\forall z)\ \neg Loves(z,x)].$

$\Rightarrow (\forall x)(\forall y)(\forall z)[\neg Animal(y) \vee \neg Kills(x,y) \vee \neg Loves(z,x)].$

↳ Predicates (Prenex Normal Form)

$\Rightarrow (\forall x)(\forall y)(\forall z)[\neg Animal(y), \neg Kills(x,y), \neg Loves(z,x)]$

↳ skolemized.

$\Rightarrow \boxed{\neg Animal(y), \neg Kills(x,y), \neg Loves(z,x)}$

↳ Clause Form.

C

Ex 3: Convert to Clause Form:

$(\forall y)\ Animal(y) \rightarrow Loves(Jack, y).$

$\Rightarrow (\forall y)\ \neg Animal(y) \vee Loves(Jack, y)$

$\Rightarrow (\forall y)(\neg Animal(y) \vee Loves(Jack, y)).$

↳ PNF, skolem.

$\Rightarrow \boxed{(\neg Animal(y), Loves(Jack, y))}.$

D

Ex 4: $\boxed{Kills(Jack, Tuna), Kills(curiosity, Tuna)}$

↳ PNF, skolem, Clause.

Ex 5: $(\forall x)\ Cat(x) \rightarrow Animal(x).$

$\Rightarrow \neg Cat(x) \vee Animal(x)$

$\Rightarrow \boxed{\neg Cat(x), Animal(x)}$

F

Ex 6: $Cat(Tuna)$

$\Rightarrow \boxed{Cat(Tuna)}$ G

Ex 7: $\neg Kill(curiosity, Tuna)$

$\Rightarrow \boxed{\neg Kills(curiosity, Tuna)}$ 7H.

- **Unification:**

(A), (B), (C), (D), (E), (F), (G), (TH).

$\theta = (x/Tuna)$

$\boxed{Animal(Tuna)}$ I .

(A) (B) (C) (D) (E) (I) (TH)

$\boxed{Kills(Jack, Tuna)}$ J

(A) (B) (C) (D) (I) (J)

$\theta = \{y/Tuna\}$

$\boxed{(\neg Kills(x, Tuna), \neg Loves(z, x))}$ K

(A) (B) (D) (J) (K)

$\theta = x/Jack$

$\boxed{\neg Loves(z, Jack)}$ L.

(A) (B) (D) (L)

$\theta = \{x/Jack\}$ $\{f(a)/x\}$

$\theta = \{x/f(x)\}$.

$\neg Animal(Jack)$

Ex 2: 1. Anyone whom Marry loves is a star.

(A) $(\forall x)\ Loves(Marry, x) \rightarrow F\_Star(x)$.

2. Any student who does not pass does not play.

(B) $(\forall x)\left([Student(x) \land \neg Pass(x)] \rightarrow \neg Play(x)\right)$.

(C) $\boxed{Student(John)}$

3. John is a Student.

(D) 4. Any student who does not study does not pass.

$(\forall x)\left([Student(x) \land \neg Study(x)] \rightarrow \neg Pass(x)\right)$.

5. Anyone who does not play, is a not a ~~football~~ star.

(E) $(\forall x)\left(\neg Play(x) \rightarrow \neg F\_Star(x)\right)$.

(F) 6. Conclusion / Therefore / To Prove / Prove that:

If John does not study then Marry does not love John.

(F) $\neg Study(John) \rightarrow \neg Love(Marry, John)$.

$\neg(\neg Study(John) \rightarrow \neg Love(Marry, John))$.

P.T.O.

1. $(\forall x)\ \neg Loves(Marry, x) \vee F\_Star(x)$

$\Rightarrow \boxed{\neg Loves(Marry, x),\ F\_Star(x)}$ (A)

2. $(\forall x)\ [\neg(Student(x) \wedge \neg Pass(x)) \vee \neg Play(x)]$

$\Rightarrow (\forall x)\ (\neg Student(x) \vee Pass(x) \vee \neg Play(x))$

$\Rightarrow \boxed{(\neg Student(x),\ Pass(x),\ \neg Play(x))}$ (B)

3. $\boxed{Student(John)}$ (C)

4. $(\forall x)\ (\neg(Student(x) \wedge \neg Study(x)) \vee \neg Pass(x))$

$\Rightarrow (\forall x)\ (\neg Student(x) \vee Study(x) \vee \neg Pass(x))$

$\Rightarrow \boxed{(\neg Student(x),\ Study(x),\ \neg Pass(x))}$. (D)

5. $(\forall x)\ (\neg Play(x) \rightarrow \neg F\_Star(x))$.

$\Rightarrow (\forall x)\ (\neg(\neg Play(x)) \vee \neg F\_Star(x))$.

$\Rightarrow (\forall x)\ (Play(x) \vee \neg F\_Star(x))$.

$\Rightarrow \boxed{(Play(x),\ \neg F\_Star(x))}$. (E)

6. $\neg(\neg(\neg Study(John)) \vee \neg Love(Marry, John))$

$\Rightarrow \boxed{\neg(Study(John) \wedge \neg Love(Marry, John))}$ (TF)

$\Rightarrow \boxed{\neg(Study(John)}$ (TF)

$\Rightarrow \boxed{Love(Marry, John)}$ (TG).

Step 1: Unify (C) & (B).

~~$\theta = \{\}$~~

Student(John)　　(¬Student(x), Pass(x), ¬Play(x))

(M)　　$\theta = \{x/John\}$.

$\boxed{Pass(John),\ \neg Play(John)}$. (H)

Step 2: Unify (C) & (D)

Student(John)　　(¬Student(x), Study(x), ¬Pass(x))

$\theta = \{x/John\}$.

$\boxed{Study(John),\ \neg Pass(John)}$ (I)

Step 3: Unify (H) & (I)

Pass(John), ¬Play(John)　　Study(John), ¬Pass(John)

$\boxed{\neg Play(John),\ Study(John)}$ (J)

Step 4: Unify (J) & (E)

¬Play(John), Study(John)　　Play(x), ¬F_Star(x).

$\theta = \{x/John\}$.

$\boxed{Study(John),\ \neg F\_Star(John)}$ (K)

Step5: Unify (K) & (A).

Study(John), ¬F_Star(John)　　¬Loves(Marry, x), F_Star(x)

$\theta = \{x/John\}$.

$\boxed{Study(John),\ \neg Loves(Marry, x)}$ (L)

Step 6: Unify (L) & (F)

Study(John), ¬Loves(Marry, x)   ¬Study(John)

¬Loves(Marry, x)   (M)

Step 7: Unify (M) & (G)

¬Loves(Marry, x)   (G) Loves(Marry, John)

$\theta = \{x/John\}$,

$\perp$

Failure.

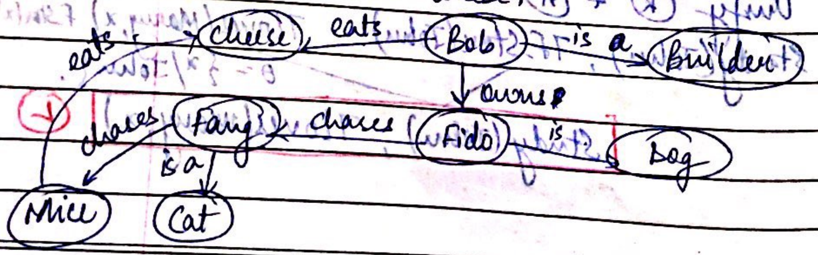
SEMANTIC NETS:

Consider the following knowledge base.

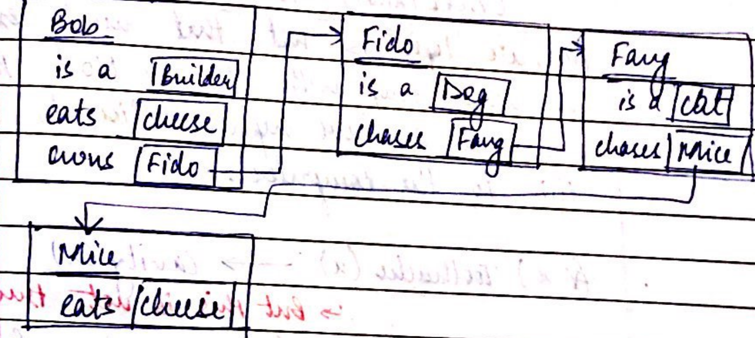
1. Bob is a builder.

(I) Bob $\xrightarrow{is}$ Builder

2. Bob eats cheese
3. Bob owns Fido
4. Fido is a dog.
5. Fido chases Fang.
6. Fang is a cat.
7. Fang chases Mice
8. Mice eats cheese.



• Frames:



→ but these are not true always.

degree of belief.

[set of all possible world.]

sample space.

UNIT — 4.

UNCERTAINTY IN REASONING

- Predicate Logic is not that much expressive. It deals only with Yes or No; Therefore we need another representation to represent this to the computer.

- $(\forall x)$ Toothache $(x) \longrightarrow$ Cavity (x)

  ↳ But this is not true always.

- So now to represent, use the Probability with Predicates.

  ↳ Toothache $\rightarrow$ cavity (.1)

  ↳ Toothache $\wedge$ Age $\leq 25 \longrightarrow$ cavity (.01).

- Probability Distribution:

  Normal Distribution   Gaussian Distribution.

2-12. $\frac{1}{11}$ Cards are assigned with numbers (sum of 2 dice).   2-72 .. $\frac{1}{36}$ Ex, T (25,52, 61, 16, 34, 43)

Probablistic Reasoning / Bayesian Reasoning:

Ex: Raining (.6)

degree of belief.

[set of all possible world.]

sample space.

- Mutually Exclusive:

  The intersection is null.

- Exhaustive:

  All possible values of a sample space.

Ex: $S = \{1, 2, 3, 4, 5\}$ — for a dice.

Mutually Exclusive — ✓

Exhaustive — X. (Because 6 is not in the set).

Ex: $S = \{(1,1), (1,2), (1,3), \ldots, (6,5), (6,6)\}$ Two dice

w is an element of sample space.

*. Event is a subset of sample space.

- When we assign a value to an element of sample space,

  then $0 \leq P(w_i) \leq 1$,

  $$\sum_{w_i \in S} P(w_i) = 1.$$

- Unconditional Probability: (Prior).

- Conditional Probability: $P(a|b)$. Posterior Probability

  Probability of a when condition b

Ex: $P(\text{double} \mid \text{Dice1} = 5) = \frac{1}{6}$.

As Dice 1 has fixed value, & the probability of (5,5) is $\frac{1}{6}$.

★ • Random Variable is a variable whose value we assign.

Cavity → true ≡ cavity.

Random variable. ✗ value ~~surrender~~ variable.

Toothache false ≡ ¬cavity

Catch.

toothache, ¬toothache

catch, ¬catch

Ex: P(Cavity | Toothache). ✗

P(Cavity = cavity | Toothache = toothache)

can be true or false as well

↓ ≡

P(cavity | toothache).

Ex: P(Tose) = <.5, .5>
Head Tail

Ex: P(cavity | toothache) = 0.6

i.e. if one has toothache, probability of cavity is 0.6.

Ex: P(cavity | toothache ∧ ¬cavity) = 0.

• How to define Conditional Probability in terms of Unconditional Probability.

(i) $P(a|b) = \frac{P(a \wedge b)}{P(b)}$, when $P(b) \neq 0$. → ①

$P(a \wedge b) = P(a|b)\,P(b)$ → ② Joint Probability

(ii) $P(b|a) = \frac{P(a \wedge b)}{P(a)}$ → ③

$P(a \wedge b) = P(b|a)\,P(a)$ → ④.

$$\boxed{P(b|a) = \frac{P(a|b)\,P(b)}{P(a)}}$$

• Baye's Formula

• Baye's Theorem.

★★ INFERENCE USING FULL JOINT DISTRIBUTION:

Ex: Toothache, Cavity, Catch are given Random Variable.

(all are Predicates / Binary). ∴ Possible values are $2^3 = 8$. (000, ..., 111)

3 Dimensional Representation

P.T.O.

	toothache		¬ toothache	
	catch	¬ catch	catch	¬ catch
cavity	0.108	0.012	0.072	0.008
¬cavity	0.016	0.064	0.144	0.576

→ $P(\neg toothache \wedge catch \wedge \neg cavity)$

1. $P(cavity \vee toothache) = ?$

$\Rightarrow P(cavity) + P(toothache) - P(cavity \wedge toothache)$

$\Rightarrow 0.108 + 0.012 + 0.072 + 0.008 + 0.108 + 0.012 + 0.016 + 0.064 - 0.108 - 0.012$

$\Rightarrow \underline{\underline{0.28}}$

$$\boxed{P(cavity \vee toothache) = P(c \vee t) + P(\neg c \vee t) + P(c \vee \neg t)}$$

2. $P(cavity) = ?$

$\Rightarrow 0.108 + 0.012 + 0.072 + 0.008$

$\Rightarrow \underline{0.2}$.

Marginal Probability

$$\boxed{P(cavity) = \sum_{\substack{x \in Toothache \\ y \in Catch}} P(cavity, x, y)}$$

double summation

Ex: $P(cavity \mid toothache) = ?$

$\longrightarrow P(cavity \mid toothache) = \dfrac{P(cavity \wedge toothache)}{P(toothache)}$

$= \dfrac{0.108 + 0.012}{0.108 + 0.012 + 0.016 + 0.064}$

$= \dfrac{0.12}{0.2} = \boxed{0.6}$

Ex: $P(\neg cavity \mid toothache) = ?$

$\longrightarrow P(\neg cavity \mid toothache) = \dfrac{P(\neg cavity \wedge toothache)}{P(toothache)}$

$= \dfrac{0.016 + 0.064}{0.108 + 0.012 + 0.016 + 0.064}$

$= \dfrac{0.08}{0.2} = \boxed{0.4}$

$P(a \mid b) = \dfrac{P(a \wedge b)}{P(b)}$, $P(\neg a \mid b) = \dfrac{P(\neg a \wedge b)}{P(b)}$.

$P(a \mid b) + P(\neg a \mid b) = 1$

$\dfrac{P(a \wedge b) + P(\neg a \wedge b)}{P(b)} = 1$.

$$\boxed{P(b) = P(a \wedge b) + P(\neg a \wedge b)}$$

(Sir's method,

let $\alpha = \dfrac{1}{P(toothache)}$

$P(cavity \mid toothache) = \alpha' P(cavity \wedge toothache)$.

$P(\neg cavity \mid toothache) = \alpha P(\neg cavity \wedge toothache)$

$\alpha [P(cavity \wedge toothache) + P(\neg cavity \wedge toothache)] = \underline{\underline{1}}$.

toothache

toothache catch cavity

P.T.O.

$\alpha = \dfrac{1}{P(cavity \wedge toothache) + P(\neg cavity \wedge toothache)}$

$P(toothache) = \dfrac{1}{\alpha} = P(cavity \wedge toothache) + P(\neg cavity \wedge toothache)$

$\boxed{0.2}$

★ Note:

As no. of Random Variables increases (n) and each has a value of m, (m=2 (Binary))
⇒ m^n data to be represented, which is not feasible!!!

Ex: toothache, cavity, catch, weather (sunny, foggy, cloudy, rainy). Binary.
4 values.

2. Table size $= 4 \times 2 \times 2 \times 2 = 4 \times 2^3$.

Ex: P(toothache, catch, cavity, cloudy)
⟶ ⇒ P(cloudy | toothache, catch, cavity) × P(toothache, catch, cavity)

Logically these doesnt depend on weather.

P(cloudy | toothache, catch, cavity) = P(cloudy)

Table size required $= 2^3 + 4 = 12$.

I

toothache
catch
cavity

weather

Ex: P(cavity, toothache, catch, rainy) = P(rainy | cavity, toothache, catch) * P(cavity, toothache, catch).

⇒ P(rainy) * P(cavity, toothache, catch).

weather does not depend cavity, toothache, catch.

∴ P(Cavity, Toothache, Catch, Weather) = P(weather) * P(Cavity, Toothache, Catch)
32. 4 8.

Toothache, Cavity, Catch, Weather.
⇓
Toothache
Cavity
Catch

Weather

Independent of each other.

But it is not always possible to reduce or to divide the things in 2 or more classes / sets.

- By Baye's Theorem:

$P(a|b) = \dfrac{P(b|a)\, P(a)}{P(b)}$; $P(b) \neq 0$.

P.T.O.

Ex: $P(\text{disease} \mid \text{symptoms}) = \dfrac{P(\text{symptoms} \mid \text{disease}) * P(\text{disease})}{P(\text{symptoms})}$

Doctor is interested ~~in~~ identifying disease when symptoms are there. Other things like symptoms when disease is there is known to him.

$$P(\text{cause} \mid \text{effect}) = \frac{P(\text{effect} \mid \text{cause}) * P(\text{cause})}{P(\text{effect})}.$$

Ex: After your yearly checkup, the doctor has bad news & good news. The bad news is that you tested positive for a serious disease and test returns correct positive results only in 98% of the cases, the test returns correct negative results only in 97% of the cases in which disease is not present. The good news is it is rare disease striking only 1 out of 10000. What are the chances that you have disease.

$\rightarrow P(\text{disease} \mid \text{positive\_res}) = ?$

$P(\neg\text{disease} \mid \text{positive\_res}) = ?$ By Baye's.

$P(a \mid b) = \dfrac{P(b \mid a)\, P(a)}{P(b)}$; $P(b) \neq 0$.

$$P(\text{disease} \mid \text{positive\_res}) = \frac{P(\text{positive\_res} \mid \text{disease}) * P(\text{disease})}{P(\text{positive\_res})}.$$

Let $\alpha = \dfrac{1}{P(\text{positive\_res})}$

P.T.O.

$P(\text{negative\_res} \mid \text{disease}) = 0.02$.

$P(\text{positive\_res} \mid \neg\text{disease}) = 1 - 0.97 = \boxed{0.03}$

$P(\text{disease} \mid \text{positive\_res}) = \alpha\, P(\text{positive\_res} \mid \text{disease}) * P(\text{disease}) \rightarrow (1)$.

$P(\neg\text{disease} \mid \text{positive\_res}) = \alpha\, P(\text{positive\_res} \mid \neg\text{disease}) \times P(\neg\text{disease}) \rightarrow (2)$.

$(1) + (2) \Rightarrow$

$$1 = \alpha\left[\left(0.98 * \frac{1}{10000}\right) + \left(\boxed{0.03} * \left(1 - \frac{1}{10{,}000}\right)\right)\right]$$

$$1 = \alpha\,(0.98 \times 10^{-4} + 0.03 * 0.9999)$$

$$\alpha = \frac{10^4}{0.98 + 0.03 * 0.9999}$$

$P(\text{disease} \mid \text{positive\_res}) = \alpha * .98 * 0.0001 = \boxed{0.000098\,\alpha}$

$P(\neg\text{disease} \mid \text{positive\_res}) = \alpha * 0.03 * 0.9999 = \boxed{0.029997\,\alpha}$.

$\alpha = \dfrac{1}{0.000098 + 0.029997}$

$$P(\text{disease} \mid \text{positive\_res}) = \frac{0.000098}{0.000098 + 0.029997} = \boxed{0.004}$$

$$P(\neg\text{disease} \mid \text{positive\_res}) = \frac{0.029997}{0.000098 + 0.029997} = \boxed{0.996}$$

Conditionally Independent

Suppose $P(\text{Catch}, \text{Toothache} \mid \text{Cavity})$

now if catch does not depend on toothache

$$P(\text{Catch} \wedge \text{Toothache} \mid \text{Cavity}) = P(\text{Catch} \mid \text{Cavity}) \times P(\text{Toothache} \mid \text{Cavity})$$

Catch & Toothache are $\boxed{\text{Conditionally Independent}}$ when Cavity is given.

Catch & Toothache are Conditionally Independent but; Catch & Weather are not Absolutely Independent

$P(\text{Catch} \wedge \text{Toothache}) \neq P(\text{Catch}) \times P(\text{Toothache})$.

Catch & Toothache are not Absolutely Independent

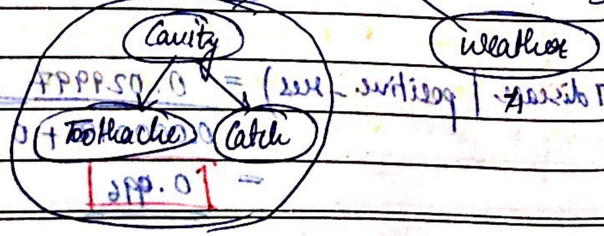
$P(\text{Catch} \wedge \text{Weather}) = P(\text{Catch}) \times P(\text{Weather})$

Catch & Weather are ~~not~~ Absolutely Independent

$P(\text{Catch} \wedge \text{Toothache}) = P(\text{Catch} \mid \text{Toothache}) \times P(\text{Toothache})$

$P(\text{Catch}, \text{Cavity}, \text{Toothache}, \text{Weather})$

32



$P(A \mid B)$

$P(\text{Toothache}, \text{Catch}, \text{Cavity}) = P(\text{Toothache}, \text{Catch} \mid \text{Cavity})\ P(\text{Cavity})$ — A, B

$P(B)$

$= P(\text{Toothache} \mid \text{Cavity}) * P(\text{Catch} \mid \text{Cavity})$ — 2

$* P(\text{Cavity})$ — 1

Calculation needed.

$P(\text{Toothache} \mid \text{Cavity})$ = (A) $P(\text{toothache} \mid \text{cavity})$

(B) $P(\neg\text{toothache} \mid \text{cavity}) = 1 - A$

(C) $P(\text{toothache} \mid \neg\text{cavity})$

(D) $P(\neg\text{toothache} \mid \neg\text{cavity})$ ↳ $1 - C$

we only really need $\boxed{A, C}$ = 2 calculations.

↳ 5 calculations

Originally, we need $2^3 = 8$ i.e 7 comparisons. as 8^{th} one can be done by $1 - 7$'s...

Naive Baye's Algo

Ex: $P(\text{Spam} \mid \text{money}, \text{how}, \text{you}) = \dfrac{P(\text{money}, \text{how}, \text{you} \mid \text{Spam}) * P(\text{Spam})}{P(\text{money}, \text{how}, \text{you})}$ — (1)

Now if Spam is given we consider that (money, how, you) are conditionally independent

(2) $= \dfrac{P(\text{money} \mid \text{spam}) * P(\text{how} \mid \text{spam}) * P(\text{you} \mid \text{spam}) \times P(\text{spam})}{P(\text{money}, \text{how}, \text{you})}$.

Ex:

Example No.	Colour	Type	Origin	Stolen?
1	Red	Sports	Domestic	Yes
2	Yellow	Sports	Domestic	No
3	Yellow	Sports	Imported	Yes
4	Yellow	SUV	Imported	Yes
5	Yellow	SUV	Domestic	No
6	Red	SUV	Imported	No
7	Red	Sports	Imported	Yes

Using above data, classify a Red Domestic SUV by using Naive Bayes Classifier.

$P(yes \mid Red, SUV, Domestic) = ?$

$P(No \mid Red, SUV, Domestic) = ?$

↳ 2 calculations

$P(yes \mid Red, SUV, Domestic) = \alpha P(Red, SUV, Domestic \mid yes) \times P(yes)$.

$= \alpha P(Red \mid yes) \times P(SUV \mid yes) \times P(Domestic \mid yes) \times P(yes)$

$= \alpha \frac{2}{4} \times \frac{1}{4} \times \frac{1}{4} \times \frac{4}{7}$

$= \alpha \frac{1}{56} \rightarrow (1)$

$P(No \mid Red, SUV, Domestic) = \alpha P(Red, SUV, Domestic \mid No) \times P(No)$

$= \alpha P(Red \mid No) \times P(SUV \mid No) \times P(Domestic \mid No) \times P(No)$

$= \alpha \frac{1}{3} \times \frac{2}{3} \times \frac{2}{3} \times \frac{3}{7} = \alpha \frac{4}{63} \rightarrow (2)$

From (1) + (2) $\Rightarrow \alpha = \frac{1}{1/56 + 4/63}$

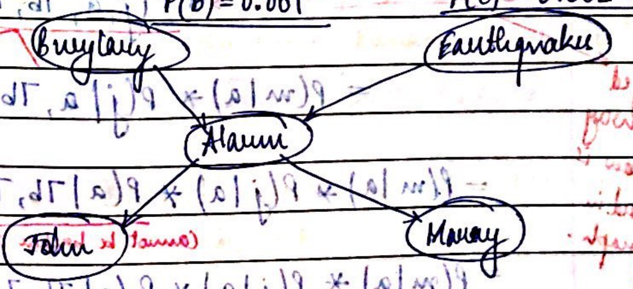
$P(yes) = \frac{1/56}{1/56 + 4/63} = 0.219$

$P(No) = \frac{4/63}{1/56 + 4/63} = 0.781$.

Ex: You have a new Burglary alarm installed at Home. It is fairly reliable at detecting a burglary but also responds to occasional minor earthquakes. You also have 2 neighbours John & Mary and who have promised to call you at work when they hear alarm. John nearly always calls when he hears the alarm but some-times confuses the telephone ringing with alarm, Mary on the other hand likes rather loud music and misses alarm.

Soln: $P(b) = 0.001$ $\quad$ $P(e) = 0.002$

Bayesian Network (Directed Graph between Random Variables with Probability values)



$P(a \mid b, e)$ — a = Alarm Rings
$P(a \mid \neg b, e)$ — b = Burglary.
$P(a \mid b, \neg e)$ — e = Earthquake.
$P(a \mid \neg b, \neg e)$ — j = John
m = Mary.

B	E	P(a)	A	P(J)	A	P(M)
T	T	0.95	T	0.9	T	0.7
T	F	0.94	F	0.05	F	0.01
F	T	0.29				
F	F	0.001				

Ex: You have a ... at home. It ... at detecting ...

$P(\neg a \mid b, e) = 1 - 0.95 = 0.05$

2 neighbours John & Mary ...

What is the probability that alarm has sounded when neither Burglary nor Earthquake has occurred & both John & Mary have called.

$\rightarrow P(a, \neg b, \neg e, j, m)$

Bayesian formula: $P(a,b) = P(a \mid b) * P(b)$

$P(m, j, a, \neg b, \neg e) = P(m \mid j, a, \neg b, \neg e) \cdot P(j, a, \neg b, \neg e)$. — Mary is only independent

arranged in a way that flow is upward in graph.

$= P(m \mid a) * P(j \mid a, \neg b, \neg e) * P(a, \neg b, \neg e)$

$= P(m \mid a) * P(j \mid a) * P(a \mid \neg b, \neg e) * P(\neg b, \neg e)$

cannot be broken as it depends on b & e

$= P(m \mid a) * P(j \mid a) * P(a \mid \neg b, \neg e) * P(\neg b) * P(\neg e)$ — independent

$= 0.7 * 0.9 * 0.001 \times 0.999 \times .998$

$= 6.2811 \times 10^{-4} = 0.000628.$

3. What is the probability that alarm has sounded, when John & Mary has called & Burglary & earthquake has occurred.

$\rightarrow P(a, b, e, j, m) = ?$

$P(m, j, a, b, e) = P(m \mid j, a, b, e) * P(j, a, b, e)$

$= P(m \mid a) * P(j \mid a, b, e) * P(a, b, e)$

$= P(m \mid a) * P(j \mid a) * P(a \mid b, e) * P(b, e).$

$= P(m \mid a) * P(j \mid a) * P(a \mid b, e) * P(b) * P(e)$

$= 0.7 * 0.9 * 0.95 * 0.001 * 0.002.$

$= \boxed{1.197 \times 10^{-6}}$

Probability that Burglary has occurred, John & Mary has called.

$\rightarrow P(j, m, b) = ?$ (Marginal Probability)

where info about some Random Variables is not given, i.e., e & a, so we take all possible combinations for these Random variables.

$$P(j,m,b) = \sum_{e \in E} \sum_{a \in A} P(j, m, a, b, e).$$

$$= P(j,m,a,b,e) + P(j,m,\neg a,b,e) + P(j,m,a,b,\neg e) + P(j,m,\neg a,b,\neg e).$$

$P(Toss = head) = 0.5$

$P(Toss = tail) = 0.5$

$P(C = head \mid Toss = head) = 1$

$P(C = tail \mid Toss = head) = 0$

$P(C = head \mid Toss = tail) = 0.2$

$P(C = tail \mid Toss = tail) = 0.8$

$P(C = head) \Rightarrow P(C = head, Toss = head) + P(C = head, Toss = tail)$

$\Rightarrow P(C = head \mid Toss = head) * P(Toss = head) + P(C = head \mid Toss = tail) * P(Toss = tail)$

$P(C=head) = 1 * 1/2 + 0.2 * 0.5 = \boxed{0.6}$

$\longrightarrow [P(j|a) * P(m|a) * P(a|b,e) * P(b) * P(e)] +$
$[P(j|\neg a) * P(m|\neg a) * P(\neg a|b,e) * P(b) * P(e)] +$
$[P(j|a) * P(m|a) * P(a|b,\neg e) * P(b) * P(\neg e)] +$
$[P(j|\neg a) * P(m|\neg a) * P(\neg a|b,\neg e) * P(b) * P(\neg e)]$

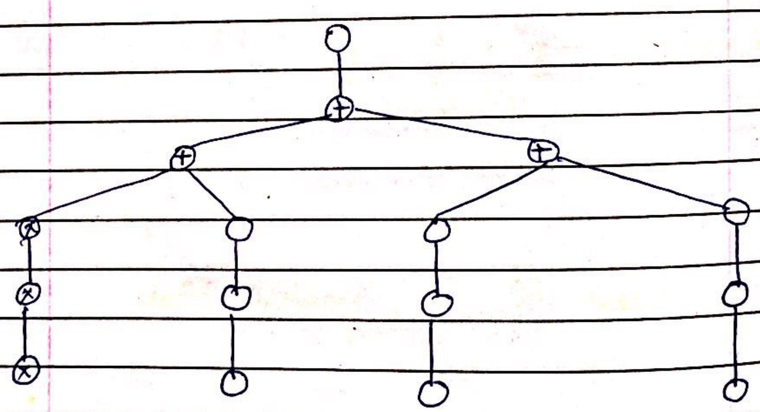
$\Rightarrow [0.9 * 0.7 * 0.95 * 0.001 * 0.002] +$

- Expression Tree:

$P(b, j, m) = P(b) \sum_e P(e) \sum_a P(j|a) * P(m|a) * P(a|b,e)$.

$\Rightarrow P(b) \left[P(e) \sum_a P(j|a) * P(m|a) * P(a|b,e) \right]$
$\quad + \left[P(\neg e) \sum_a P(j|a) * P(m|a) * P[a|b,\neg e] \right]$

$\Rightarrow P(b) \left[P(e) \left[P(j|a) * P(m|a) * P(a|b,e) \right] + \left[P(j|\neg a) * P(m|\neg a) * (\neg a|b,e) \right] \right]$
$\quad + \left[P(\neg e) \left[P(j|a) * P(m|a) * P(a|b,\neg e) \right] + \left[P(j|\neg a) * P(m|\neg a) * (\neg a|b,e) \right] \right]$.



5. Probability of b when a, j, ¬m is given.

$\Rightarrow P(b|a,j,\neg m) = ?$

$P(b|a,j,\neg m) = \dfrac{P(b,a,j,\neg m)}{P(a,j,\neg m)} = \alpha P(b,a,j,\neg m) \longrightarrow (1)$

$P(\neg b|a,j,\neg m) = \alpha P(\neg b,a,j,\neg m) \longrightarrow (2)$.

$P(b,a,j,\neg m) = P(j,\neg m,a,b,e) + P(j,\neg m,a,b,\neg e)$
$= P(j|a) * P(\neg m|a) * P(a|b,e) * P(b) * P(e)$.
$+ P(j|a) * P(\neg m|a) * P(a|b,\neg e) * P(b) * P(\neg e)$

$= 0.9 * 1-0.7 * 0.95 * 0.001 * 0.002$.
$+ 0.9 * 1-0.7 * 0.94 * 0.001 * 0.998$.

$= 0.000253$.

$P(\neg b,a,j,\neg m) = P(j,\neg m,a,\neg b,e) + P(j,\neg m,a,\neg b,\neg e)$
$= P(j|a) * P(\neg m|a) * P(a|\neg b,e) * P(\neg b) * P(e)$
$+ P(j|a) * P(\neg m|a) * P(a|\neg b,\neg e) * P(\neg b) * P(\neg e)$.

$= (0.9 * 0.3 \times 0.999) * [0.29 * 0.002 + 0.001 * 0.998]$

$= 0.000425$

$(1)+(2) = 1$.

$\alpha = \dfrac{1}{P(b,a,j,\neg m) + P(\neg b,a,j,\neg m)}$.

$P(b|a,j,\neg m) = \alpha$

UNIT – 5.

MACHINE LEARNING

- Supervised Learning:
- Unsupervised Learning:

Traditional Model:

Input $\rightarrow$ [Algorithm] $\rightarrow$ y output.

$y = f(x)$

ML Model: (Supervised Learning — I/O both are provided)

Classification.

Input, Output $\rightarrow$ [Machine Learning] $\rightarrow$ Algorithm $\rightarrow$ Input $\rightarrow$ [Algo] $\rightarrow$ Output.

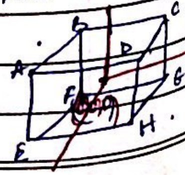
① Model Training. ② Model Testing.

- Unsupervised Learning:

Where clustering is done based on the similar features, where only 'Input' is provided.

Ex: Cube of length = 2, center of the cube is (0,0,0)

~~A = (0,2,2).~~ ~~B = (~~



How many different classifications are required for 8 points.

$P_0(-1,-1,-1)$, $P_1(-1,-1,1)$, $P_2(-1,1,-1)$, $P_3(-1,1,1)$

$P_4(1,-1,-1)$, $P_5(1,-1,1)$, $P_6(1,1,-1)$, $P_7(1,1,1)$.

Random Classification.

$C_1 = \{P_3, P_5, P_6, P_7\}$, $C_2 = \{P_0, P_1, P_2, P_4\}$.

Problem-1

Ex: When given an input of 3 points, the computer should determine in which class that belongs.

Soln: In above example, C_1 holds all points having maximum 1 –ve co-ordinate, and others in C_2.

1st way: Keep a counter of either positive or negative values. In either way when counter reaches to '2' then we have classified either way.

2nd way: $sum = x_1 + x_2 + x_3$
if $sum > 0$
then C_1.
else C_2.

3rd way: Input x_1, x_2, x_3.
$sum = x_1 + x_2 + x_3$
$y = sign(sum)$
if $y > 0$ then C_1
else C_2.

How many different classification are required for 8 points.

Ex: Problem 2: $C_1 = \{P_7\}$
$C_2 = \{$ Rest of the points $\}$. Random classification

Soln: $Sum = x_1 + x_2 + x_3 - 2$

$y = sign(sum)$

Problem 1 Ex: When given are 3 points the computer should determine which class that belongs.

Ex: Problem 3: In above example, $C_2 = \{P_7\}$ having maximum coordinate.

$C_1 = \{$ Rest of the points $\}$ and

Soln: $Sum = w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 * 1$.

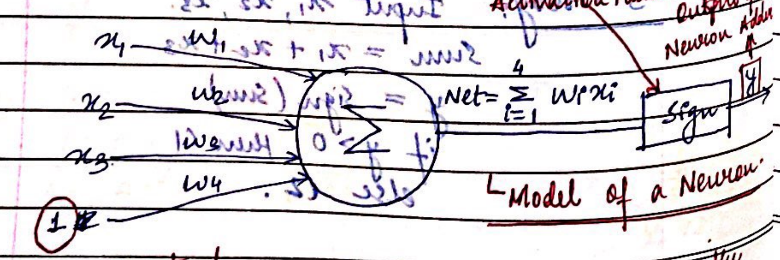
Generalized Solution for all 3 problems:

$y = sign(sum)$ either negative or values

$W = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}$ Problem1-soln

Problem2-soln

$\begin{bmatrix} -1 \\ -1 \\ -1 \\ 2 \end{bmatrix}$ Problem3-soln



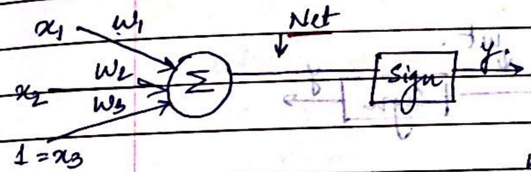
└Model of a Neuron.

Biased Input / Dummy Input for maintaining the condition.

AND CLASSIFICATION:

x_2	x_1	t.
-1	-1	-1
-1	1	+1
1	-1	+1
1	1	1.

→ target / desired O/P.



$W_1 = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}$ $W_2 = \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix}$ $1 \times -1 + 1 \times -1 + -1 \times 1 = -3$

trial & error method.

so $Net = w_1 x_1 + w_2 x_2 + w_3 x_3$

$\therefore Net = x_1 + x_2 - 1$

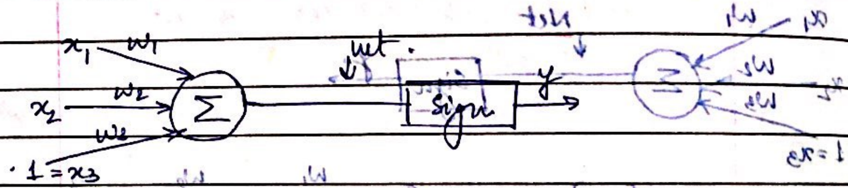
x_1	x_2	$Net = x_1 + x_2 - 1$	$y = sign(Net)$	t.
-1	-1	-3	-1	-1
-1	1	-1	-1	-1
1	-1	-1	-1	-1
1	1	1	1	1.

$\forall y_i = t_i$ ∴ Proper classification & correct solution.

i.e., actual O/P = desired O/P.

*** OR CLASSIFICATION: ***

x_2	x_1
−1	−1
−1	1
1	−1
1	1



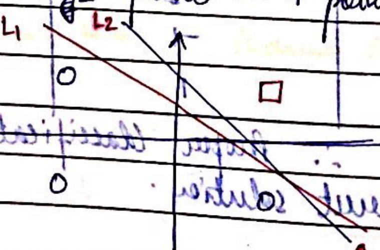
$w_1 = 1,\ w_2 = 1,\ w_3 = 1$ — trial & error method.

$$Net = w_1 x_1 + w_2 x_2 + w_3 x_3$$
$$= x_1 + x_2 + 1$$

∴ we can similarly check the output as in previous examples.

When only two classes formed — Binary Classification

2D — $x_1 x_2$ plane — 4 points are input.



AND Classification

x_2	x_1	t
−1	−1	−1
−1	1	1
1	−1	1
1	1	1

w_3 acts as constant

$y = mx + c$

- We are interested in finding a line in $x_1 x_2$ plane, such that it divides the points in two classes.
- Multiple solutions exist as multiple such lines can be drawn.

∴ Infinite many solutions.

Considering w_1, w_2, w_3 from L_1.

$L_1 : y = mx + c.$

$m = -1,\ c = 1.$

$y = -x + 1$

$y + x - 1 = 0$

$x_2 + x_1 - 1 = 0$

$\boxed{x_1 + x_2 - 1 = 0} \rightarrow ①.$

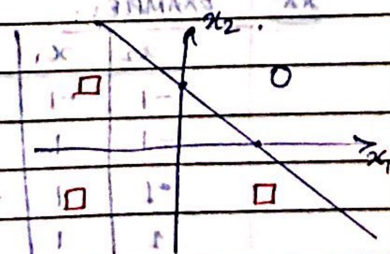
Generalized equation = $w_1 x_1 + w_2 x_2 + w_3 x_3 = 0$

∴ By comparing:

$w_1 = 1,\ w_2 = 1,\ w_3 = -1.$

*** NAND CLASSIFICATION:

x_2	x_1	t
−1	−1	1
−1	1	1
1	−1	1
1	1	−1



$L_1 : y = mx + c$, $m = -1$, $c = 1$

$y = -x + 1$.

$y + x - 1 = 0$.

$x_2 + x_1 - 1 = 0$.

$x_1 + x_2 - 1 = 0$.

$w_1 = 1$, $w_2 = 1$, $w_3 = -1$ ∴ but doesn't match with truth table.

∴ check line orientation

Plot any point. less line in eqn,

put(0,0) $x_2 + x_1 - 1 \Rightarrow 0 + 0 - 1 = -1 < 0$.

but we want 1 the value on less than side of the line L_1.

This is called orientation line.

∴ Multiply by -1, which changes the orientation of line.

∴ $-x_1 - x_2 + 1 = 0$

∴ $w_1 = -1$, $w_2 = -1$, $w_3 = 1$.

** EXAMPLE: **

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

$L_1 = x_1 = 0$, ∴ $c = 0$. $1 \times x_1 + 0 \times x_2 + 0 \times 1 = 0$.

$1 \times x_2 + 0 \times x_1 + 0 \times 1 = 0$.

$$w_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad w_1 = 1,\ w_2 = 0,\ w_3 = 0.$$

Put (1,1) in equation to check orientation.

$1 \times 1 + 0 \times 1 + 0 \times 1 \geq 0$ ✓ should give 1.

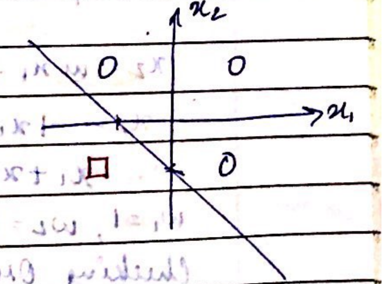
& the answer is 1.

∴ Orientation is correct.

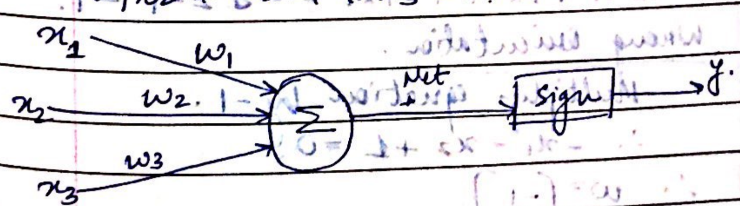
Also solution is $[2, 0, -1]$.

★ NOR Classification:

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



$w_1 x_2 + w_2 x_1 + w_3 = 0$

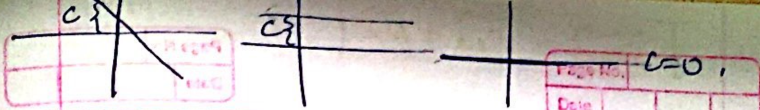


$y = mx + c$. $m = -1$, $c = -1$.

$y = -1x - 1$.

$x + y + 1 = 0 \Rightarrow x_1 + x_2 + 1 = 0$.

$w_1 = 1$, $w_2 = 1$, $w_3 = 1$.



Checking Orientation:-

$w_1x_1 + w_2x_2 + w_3x_3 \geq 0$

$1\times -1 + 1\times -1 + 1\times 1 = -1$

Wrong orientation.

$\therefore$ Multiply equation by -1.

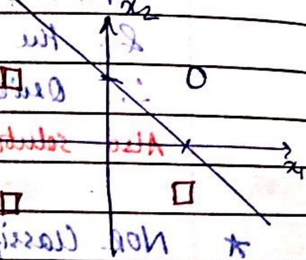
$\therefore -x_1 - x_2 - 1 = 0$

$w = [-1, -1, -1]^T$. Put $(1,1)$

$0 < 1\times 0 + 1\times 0 + 1\times 1$ ✓

Ex:

x_2	x_1	t
-1	-1	$\square$
-1	1	+1
1	-1	+1
1	1	-1



Also solution is [1, +1, -1]

Non classification ★

$x_2 = m x_1 + c$, $m = -1$, $c = 1$

$x_2 = -1x_1 + c\,1$

$\therefore x_1 + x_2 - 1 = 0$.

$w_1 = 1$, $w_2 = 1$, $w_3 = -1$.

Checking Orientation:

$1\times 1 + 1\times 1 + -1\times 1 = 2 - 1 = 1$.

Wrong Orientation.

Multiply equation by -1.

$\therefore -x_1 - x_2 + 1 = 0$.

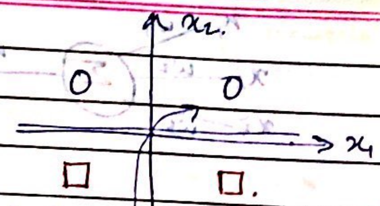
$\therefore w = \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix}$.

$y = -1x - 1$.

$x + y + 1 = 0$

x_2	x_1	t
-1	-1	1 $\square$
-1	1	1 $\square$
1	-1	-1 o
1	1	-1 o

Ex:



Slope $\frac{x_2}{y} = 0$, $c = 0$

$\therefore 0 * x_1 + 1 * x_2 + 0 * = 0$.

Put $(1,1)$ $0*1 + 1*1 + 0\times 1 = 0$

$= 1 > 0$, but it should be -ve. $\times(-1)$

$\Rightarrow$ Change orientation. Multiply by -1.

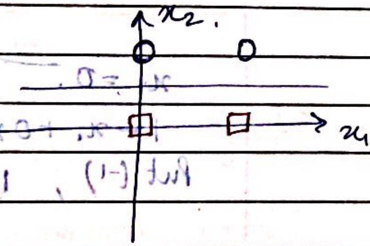
$-x_2 = 0$

$w_1 = 0$, $w_2 = -1$, $w_3 = 0$.

Biased Input:

Ex:

x_2	x_1	t
0	0	1 $\square$
0	1	1 $\square$



$x_2 = m x_1 + c$, $m = 0$, $c = 0.5$.

$x_2 = 0 + 0.5$

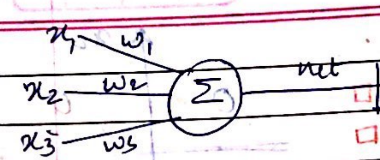
$= 1/2$

$2x_2 - 1 = 0$.

$w_1 = 0$, $w_2 = 2$, $w_3 = -1$.

Put $(0,0)$ $0\times 0 + 1\times 0 + -1\times 1 = 0 - 1 = -1$.

if net ≥ 0 then y = 1 else y = 0.



Change Orientation:-
Multiply by −1.

$\therefore w = \begin{bmatrix} 0 \\ -2 \\ 1 \end{bmatrix}$

Ex: Not Gate

x_1	y
−1	1
1	−1

$x_1 w_1 + x_2 w_2 = 0$. Biased Input.

$x_1 = 0$.

$1 * x_1 + 0 * 1 = 0$.

Put (−1), $1 * (-1) + 0 = -1 + 0 = -1$.

∴ should be +ve.

⇒ Change Orientation:

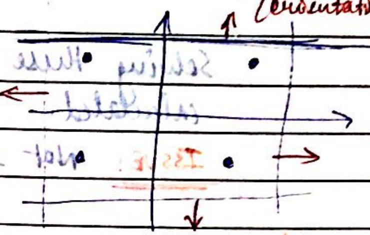
⇒ $-1 \times x_1 - 0 \times 1 = 0$.

$w_1 = -1$, $w_2 = 0$

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

Put (0,0) $0 \times 0 + 0 \times 1 - 1 \times 1$

x_2	x_1	y.
−1	1	−1
1	−1	−1
1	1	−1



∴ $x_2 = 1.5$ $0 \times x_1 + 1 * x_2 - 1.5$

$w_1 x_1 + w_2 x_2 + w_3 x_3 = 0$

$0 * x_1 + 1 \times x_2 - 1.5 = 0$

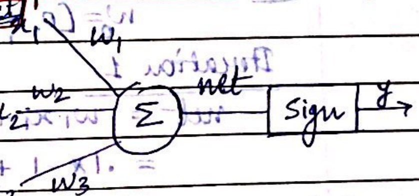
$0 * x_1 + 2x_2 - 3 = 0$

put (1,1) $2 - 3 = -1 < 0$. ✓

$w = [0 \ \ 2 \ \ -3]^T$ or $w = [0 \ \ 1 \ \ -1.5]^T$

Ex: AND GATE: How computer can learn this?

x_2	x_1	y [target]
−1	−1	−1
−1		−1



$\text{net} = w_1 x_1 + w_2 x_2 + w_3$.

Approach 1: Form Inequalities.

$-w_1 - w_2 + w_3 < 0 \rightarrow (1)$

$w_1 - w_2 + w_3 < 0 \rightarrow (2)$

$-w_1 + w_2 + w_3 < 0 \rightarrow (3)$

$[w_1 + w_2 + w_3] > 0 \rightarrow (4)$.

Sign func:
if net < 0
y = −1
else
y = 1.

$$\boxed{\vec{W_{new}} = \vec{W_{old}} + \vec{\Delta W}}$$

- Solving these for inequalities, weights can be calculated.
- **ISSUE:** Not scalable.

RANDOM LEARNING (Learning Algorithm)

Approach 2: Random Weights Generation

— Calculate w_1, w_2, w_3 randomly and ask machine to check. If incorrect re-generate weights.

ISSUE: 1. Not scalable, Number of iterations might increase to very large value.

2. Non-Systematic Approach.

Approach 3: Start with some random value and anything will converge to result.

Ex: Iteration 0 → Randomly generate w_0

$w_0 = [0.1, 0.1, 0.1]$

Iteration 1

$net = w_1x_1 + w_2x_2 + w_3$.

$= .1 * -1 + .1 * -1 + .1 = -0.1$.

$y = sign(net) = sign(-0.1) = -1$.

$$\boxed{\vec{W_{new}} = \vec{W_{old}} + \vec{\Delta W}}$$

$\vec{W_1} = \vec{W_0} + \vec{\Delta W}$

Case 1 if $y = t$ then $\vec{\Delta W} = \vec{0} = [0\,0\,0]$

$\vec{W_1} = \vec{W_0} = [.1, .1, .1]^T$

Iteration 2: $w_1 = [0.1 + 0.1 + 0.1]$.

for $(1, -1)$. $net = 0.1 \times 1 + 0.1 * -1 + 0.1 = 0.1$.

$x_1 \quad x_2$

$y = sign(net) = 1 \neq$ target

$\therefore$ Calculate relation.

$net_{new} = \vec{W_{new}} \cdot \vec{x} = (\vec{W_{old}} + \vec{\Delta w}) \cdot \vec{x}$.

$= \vec{W_{old}} \cdot \vec{x} + \vec{\Delta W} \cdot \vec{x}$.

$net_{new} = net_{old} + \vec{\Delta w} \cdot \vec{x}$

$net_{new} < net_{old}$

Here $net_{old} = 0.1$.

we wanted $y = -ve$

we need $net_{new} < net_{old}$.

$\therefore$ we need $\vec{\Delta w} \cdot \vec{x}$ to give (move towards -ve side).

. [Dot product ⇒ $\cos\theta$, $\cos\theta$ is +ve for $\theta = 90°$ to, $\theta = 180°$ (2nd Quadrant).

$\cos\theta$ will give highest negative value for 180°

$\therefore \vec{\Delta w} = -\vec{x}$

$\vec{\Delta w} \cdot \vec{x} = -\vec{x} \cdot \vec{x}$

$\vec{W_{new}} = \vec{W_{old}} + \vec{\Delta w} = [.1\ .1\ .1] + [-1\ 1\ -1]$. multiplied by -1.

$w_2 = [0.1\ \ 0.1\ \ 0.1] + \vec{\Delta w}$ OR

$\vec{\Delta w} = -[1\ -1\ 1]$.

$w_2 = [0.1\ \ 0.1\ \ 0.1] - [1\ -1\ 1]$.

$w_2 = [-0.9\ \ 1.1\ \ -0.9]$.

One iteration cover one data point. Here

P.T.O.

case 2 if $y=1$ & $t=-1$
then $\Delta w = -\vec{x}$
case 3 if $y=-1$ & $t=1$,
then $\Delta w = \vec{x}$

Iteration 3: $w_2 = [-0.9,\ 1.1,\ -0.9]$.

for $(-1, 1)$ [x_1, x_2], $net = -0.9 * (-1) + 1.1 * 1 + (-0.9)*1$
$= 1.1$

$y = sign(1.1) = 1$.
$y \neq$ target.
Again change w.
$w_3 = w_2 + \Delta w$
we need $-ve$.
$\Delta w = -x$
$w_3 = w_2 + (-x)$
$= [-0.9\ \ 1.1\ \ -0.9] - [-1\ \ 1\ \ 1]$.
$w_3 = [0.1\ \ 0.1\ \ -1.9]$.

Iteration 4: $w_3 = [0.1,\ 0.1,\ -1.9]$.
for $(1,1)$, $net = 0.1 * 1 + 0.1 * 1 + (-0.9)*1$
$= -1.7$
$y = Sign(-1.7) = -1$.
$y \neq$ target.
we wanted +ve,
$\Rightarrow$ move towards positive
$\therefore \vec{\Delta w} = +\vec{x}$ $(\theta = 0°,\ \cos\theta = 1$ (max) max cos value.$)$
$\vec{w_4} = \vec{w_3} + \vec{\Delta w}$
$= \vec{w_3} + \vec{x}$
$= [0.1\ \ 0.1\ \ -1.9] + [1\ \ 1\ \ 1]$
$= [1.1\ \ 1.1\ \ -0.9]$.

One iteration covers one data point. Here

There are total 4 data points.
$\therefore$ No. of iterations $= 4$.
$\Rightarrow$ One epoch consists of 4 iterations.

Now start second epoch, if for all data points, i.e., if for all iterations $\vec{w}$ give correct values, then stop after this epoch.

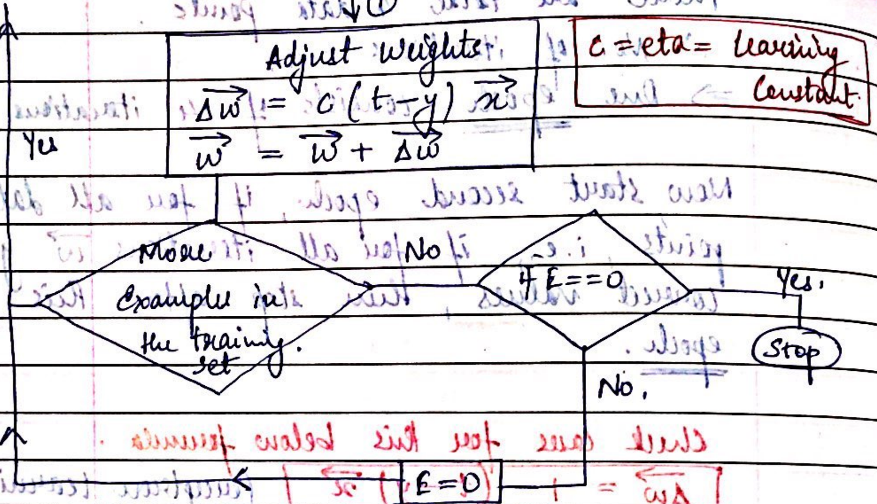
Check case for this below formula.

$$\boxed{\vec{\Delta w} = \frac{1}{2}(t - y)\,\vec{x}}$$ Perceptron Learning Algorithm.

Case 1: $\frac{1}{2}(1-1)\,\vec{x} = [0\,0\,0]^T$.
case 2: $\frac{1}{2}(-1-1)\,\vec{x} = -\vec{x}$
case 3: $\frac{1}{2}(1-(-1))\,\vec{x} = \vec{x}$

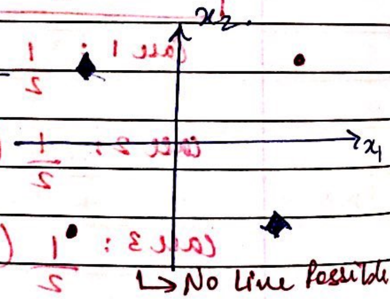
Flowchart:

Initialize weights w
↓
Submit Inputs X & target output t, $E = 0$
↓
Calculate $net = \vec{w}.\vec{x}$
$y = sign(net)$
↓
$E \leftarrow E + 1/2\ \|t - y\|^2$
↓
(1)
Done for an Epoch.



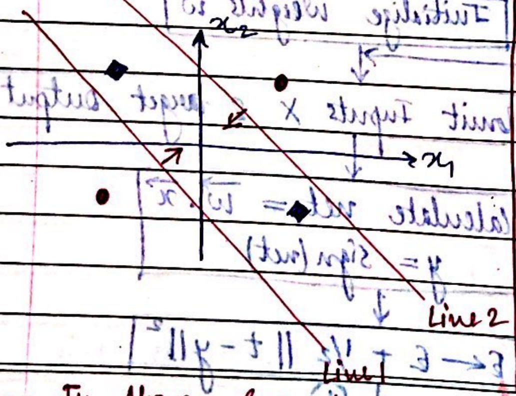
Ex:	XOR	
x_2	x_1	
-1	-1	-1
1	-1	1
-1	1	1
1	1	-1

LINEARLY NON-SEPARABLE PROBLEM



↳ No Line Possible

— Rest of the above problems are LINEARLY SEPARABLE PROBLEMS:



XOR & XNOR are two problems that are Linearly Non-Separable and requires 'Neural Network' to be solved.

— In Above Example we have only need a Single Neuron.

Ex: Train OR gate classification using Perceptron Learning Algorithm with Initial weights as $w^0 = [.1 \; -0.2 \; .3]^T$. Find weights after 1 Epoch. Learning constant $= 1$, $c = 1$

Solu:

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

Iteration 1:

$net = w_1 x_1 + w_2 x_2 + w_3 x_3$.

$= .1 \times -1 + (-0.2) \times -1 + 1 \times .3$

$= -0.1 + 0.2 + 0.3$

$= 0.4$

$y = sign\,(0.4) = 1.$

$y \neq t.$

$\vec{W_{new}} = \vec{W_{old}} + \vec{\Delta w}$

$\vec{\Delta w} = c\,(t-y)\,\vec{x}$

$= 1\,(-1-1) * \vec{x}$

$= -2 * \vec{x}$

$\vec{W_{new}} = \vec{W_{old}} + \vec{\Delta w}$

$= [.1 \; -0.2 \; .3] + [-2 \; -2 \; 2].$

$= [-1.9 \; -2.2 \; 2.3]$

$E = E + 1/2 \,\|t-y\|^2$

$= 0 + 1/2\,(-1-1)^2 = 0 + 1/2\,(4) = 2$

P.T.O.

Iteration 2: $w = [2.1 \quad 1.8 \quad -1.7]$

$net = w_1 x_1 + w_2 x_2 + w_3 x_3$

$= 2.1(1) + 1.8(-1) + (-1.7) * 1.$

$= 2.1 - 1.8 - 1.7.$

$= -1.4 \qquad y = sign(-1.4) = -1.$

$y \neq t, \quad t = 1.$

$\overrightarrow{W_{new}} = \overrightarrow{W_{old}} + \overrightarrow{\Delta w}$

$\overrightarrow{\Delta w} = c.(t-y).\overrightarrow{x^T}$

$= 1.(1-(-1)).\overrightarrow{x}$

$= 2.\overrightarrow{x}$

$\overrightarrow{W_{new}} = [2.1 \quad 1.8 \quad -1.7] + [2 \quad -2 \quad 2].$

$= [4.1 \quad -0.2 \quad 0.3].$

$E = 2 + \frac{1}{2} \|1-(-1)\|^2 = 2+2 = 4.$

Iteration 3: $w = [4.1 \quad -0.2 \quad 0.3]^T$

$net = w_1 x_1 + w_2 x_2 + w_3 x_3.$

$= 4.1(-1) + (-0.2)*1 + 0.3*1.$

$= -4.1 - 0.2 + 0.3$

$= -4.0.$

$y = sign(-4.0) = -1$

$y \neq t. \quad t = 1.$

$\overrightarrow{W_{new}} = \overrightarrow{W_{old}} + \overrightarrow{\Delta w}$

$\overrightarrow{\Delta w} = c.(t-y).\overrightarrow{x}$

$= 1.(1-(-1)).\overrightarrow{x}$

$= 2.\overrightarrow{x} = [-2 \quad 2 \quad 2].$

$\overrightarrow{W_{new}} = [4.1 \quad -0.2 \quad 0.3] + [-2 \quad 2 \quad 2]$

$= [2.1 \quad 1.8 \quad 2.3].$

$E = 4 + 1/2 \|1-(-1)\|^2 = 4+2 = \underline{6}.$

Iteration 4: $w = [2.1 \quad \cancel{-0.2} \; 1.8 \quad \cancel{0.3} \; 2.3]$

$net = w_1 x_1 + w_2 x_2 + w_3 x_3.$

$= 2.1(1) + 1.8*1 + 2.3*1.$

$= 2.1 + 1.8 + 2.3$

$= 6.2$

$y = sign(net) = 1.$

$t = y. \quad \therefore \Delta w = 0.$

$E = 6 + 1/2(1-1)^2 = 6.$

$\overrightarrow{w_4} = \overrightarrow{w_3} = [2.1 \quad 1.8 \quad 2.3]^T.$

$E = 0$

Epoch 2: * **Iteration 1:** $w^4 = [2.1 \quad 1.8 \quad 2.3]^T.$

$net = w_1 x_1 + w_2 x_2 + w_3 x_3$

$= 2.1(-1) + 1.8(-1) + 2.3(1).$

$= -3.9 + 2.3 = -1.6$

$y = sign(net) = -1. \quad t = -1.$

$y = t$

$E = 0 + 1/2(-1-(-1))^2 = 0.$

Iteration 2: $w^5 = [2.1 \quad 1.8 \quad 2.3]^T$

$net = w_1 x_1 + w_2 x_2 + w_3 x_3$

$= 2.1(1) + 1.8(-1) + 2.3(1)$

$= 2.1 - 1.8 + 2.3$

$= 2.6$

$y = sign(net) = 1, \quad t = 1.$

$y = t$ ✓. $\therefore \Delta w = 0$

$E = 0 + 1/2[(1-1)^2]$

and so on for Iteration 3 & 4.

** DECISION TREES :-

- Information Content :-

    If a statement contains an event whose probability of happening is less, such statement contains more information as compared to a statement that contains event whose probability is high.

    $\Rightarrow$ information $\propto \frac{1}{P(E)}$.

Information for E_1 + Information for E_2 = $P(E_1) * P(E_2)$

$f(P(E_1)) + f(P(E_2)) = f(P(E_1) * P(E_2))$.

$\therefore$ Defining Information,

information $= \log_2 (1/P(E))$

Mean $\frac{1+2+3+4+5+6}{6} = \frac{21}{6} = 3.5$.

Mean of rolling dice $= \Sigma x \times P(x)$

$= 1 \times P(1) + 2 * P(2) + 3 P(3) + 4 P(4) + 5 P(5) + 6 P(6)$

$= \frac{1+2+3+4+5+6}{6} = 3.5$.

For 2 Dice :-

Mean $= 2 * 1/36 + 3 * 2/36 + 4 * 3/36 + 5 * 4/36 + 6 * 5/36 + 7 * 6/36 + 8 * 5/36 + 9 * 4/36 + 10 * 3/36 + 11 * 2/36 + 12 * 1/36$.

$= \boxed{7}$

For Maximum probability, beat the drum the least.

Normal Mean $\Rightarrow 2 * 1/36 + 3 * 2/36 + 4 * 3/36 + 5 * 4/36 + 6 * 5/36 + 7 * 6/36 + 8 * 5/36 + 9 * 4/36 + 10 * 3/36 + 11 * 2/36 + 12 * 1/36 = \boxed{7}$

Compressed Mean $\Rightarrow 11 * 1/36 + 9 * 2/36 + 7 * 3/36 + 5 * 4/36 + 3 * 5/36 + 1 * 6/36 + 2 * 5/36 + 4 * 4/36 + 6 * 3/36 + 8 * 2/36 + 10 * 1/36$

$= \boxed{4.47}$

Hence, Mean has been reduced.

** Error Based learning / Information Based learning.

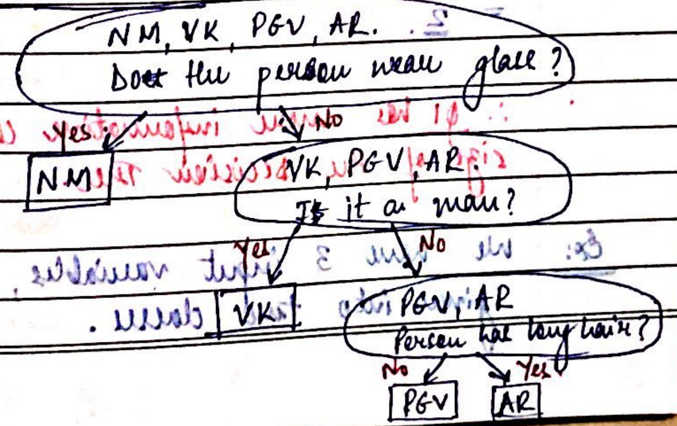
Narendra Modi (NM), Virat Kohli (VK), Priyanka Gandhi Vadra (PGV), Aishwarya Rai (AR)

Q1 Is the person a man? Q2 Person wear glasses?

Man	Long Hair	Glasses	Name.
Yes	No	Yes	NM
Yes	No	No	VK
No	No	No	PGV
No	Yes	No	AR.

Q3 Person has long hair?

- Decision Trees



NM   VK   PGV   AR

$= 1*1/4 + 2*1/4 + 3*1/4 + 3*1/4$ ← got results after how many question

$= 2.25$

OR

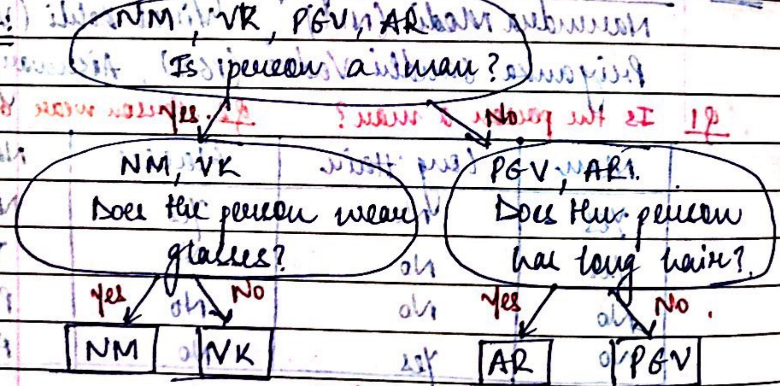
$\frac{1+2+3+3}{4} = 2.25$

When 2 NM cards are there,

$= 1*2/5 + 2*1/5 + 3*1/5 + 3*1/5$

$= 2$

Ex2: NM, VK, PGV, AR



$= 2*1/4 + 2*1/4 + 2*1/4 + 2*1/4$

$= 2.$

∴ Q1 has more informative causing lesser size of the Decision Tree.

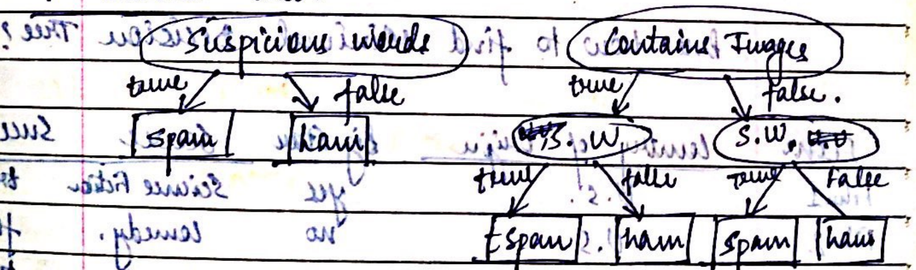
Ex: We have 3 input variables, classify the given into two classes.

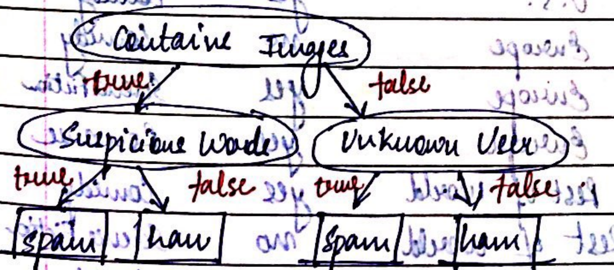
~~Ex3:~~ ~~NM, VK, PGV, AR.~~

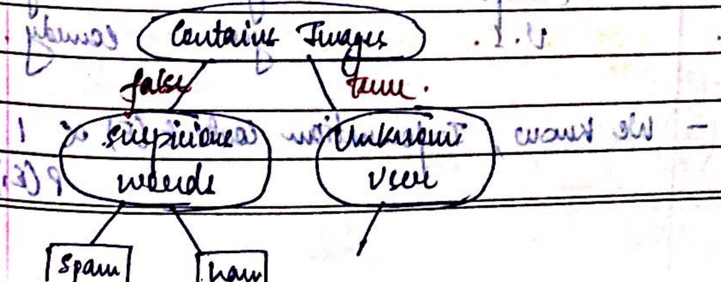
ID.	Suspicious Words	Unknown User	Contains Images	Class.
1.	true	false	true	spam
2.	true	true	false	spam
3.	true	true	false	spam
4.	false	true	true	ham
5.	false	false	false	ham
6.	false	false	false	ham.
7.	true	true	true	? spam

$1/2 * 2/6 = 1/6 = 0.1667$

• Decision Tree:







$\boxed{(CI = true) \wedge (SW = true) \vee (CI = false \wedge VV = true)}$.

DNF

$2^{2^{10}} = 2^{1024} = 10^{308}$ — for 10 variables, possible Decision trees.

- We prefer the smallest Decision Tree. Why?

⇒ Because No. of steps & probes required are minimal.

- But how to find Minimal Decision Tree?

Film	Country of Origin	Big Star	Genre	Success
Film 1	U.S.	yes	Science Fiction	true
Film 2	U.S.	no	Comedy.	false
Film 3	U.S.	yes	Comedy	true
Film 4	Europe	no	Comedy	true
Film 5	Europe	yes	Science Fiction	false
Film 6	Europe	yes	Romance	false
Film 7	Rest of World	yes	Comedy	false
Film 8	Rest of World	no	Science Fiction	false
Film 9	Europe	yes	Comedy	true
Film 10.	U.S.	yes	Comedy.	true

- We know, Information Content $(E_i) \propto \frac{1}{P(E_i)}$

- For 2 Events:

$$\text{Information Content}(E_1) + \text{Information Content}(E_2) \propto \left(\frac{1}{P(E_1)} \times \frac{1}{P(E_2)}\right).$$

$$- \text{Info. Content}(E_1) + \text{Info. Content}(E_2).$$

$$= \log_2\left(\frac{1}{P(E_1)} \cdot \frac{1}{P(E_2)}\right)$$

$$- \text{Info. Content} = \log_2\left(\frac{1}{P(E)}\right) = \boxed{-\log_2(P(E))}$$

Average Information Content (Entropy):

$$\boxed{- \sum_{i=1} P(E_i) \log_2 (P(E_i)).}$$

Probability × Information Content

ENTROPY

Ex: A Coin Toss,

$$\text{Entropy} = S = \sum P(E_i) \log_2 (P(E_i)) \quad H = \text{Head}. \quad T = \text{Tail}.$$

$$= -\frac{1}{2} \log_2\left(\frac{1}{2}\right) - \frac{1}{2} \log_2\left(\frac{1}{2}\right)$$

$$= \frac{1}{2} + \frac{1}{2} = \underline{1}.$$

Ex: 3 cards of Head & 1 card of Tail.

$$\text{Entropy} = - P(H) \log_2 P(H) - P(T) \log_2 (P(T))$$

$$= -\frac{3}{4} \log_2\left(\frac{3}{4}\right) - \frac{1}{4} \log_2\left(\frac{1}{4}\right)$$

$$= \underline{\underline{0.8112}}.$$

Ex: 4 Cards of Head & 0 cards of Tail.

$$= -\frac{4}{4}\log_2(1) - \frac{0}{4}\log_2(0)$$

$$= 0 - 0 = 0 . \qquad \boxed{\log_2(0) = 0}$$

Ex: Calculate the entropy of $\boxed{\text{success}}$ (Table Film).

$\Rightarrow$ Entropy $= -P(5/10)\log_2(5/10) - P(5/10)\log_2(5/10)$

$= 1/2 + 1/2 = \boxed{1}$

$\hookrightarrow P(\text{success})\log_2(P(\text{success})) - P(\text{fail})\log_2(P(\text{fail}))$

★ : Information Gain : (IG)

$$IG(CO) = H(\text{success}) - \frac{4}{10} \times H(US) - \frac{4}{10} \times H(EU) - \frac{2}{10} H(RW).$$

CO = Country of Origin
RW = Rest of World
EU = Europe.
US = United States.

Probability × Information content

ENTROPY

$$H(US) = -\frac{3}{4}\log_2\left(\frac{3}{4}\right) - \frac{1}{4}\log_2\left(\frac{1}{4}\right) = 0.8112.$$

Success for US films

$$H(EU) = -\frac{2}{4}\log_2\left(\frac{2}{4}\right) - \frac{2}{4}\log_2\left(\frac{2}{4}\right) = 1.$$

$$H(RW) = -\frac{0}{4}\log_2\left(\frac{0}{4}\right) - \frac{4}{4}\log_2\left(\frac{4}{4}\right) = 0.$$

$$IG(CO) = 1 - \frac{4}{10}(0.8112) - \frac{4}{10}(1) - \frac{2}{10}(0) = \boxed{0.2755}$$

Ex: $IG(BS) = H(\text{success}) - \frac{7}{10} H(\text{yes}) - \frac{3}{10} H(\text{No})$ & BS = Big Star

$$H(\text{yes}) = -\frac{4}{7}\log_2\left(\frac{4}{7}\right) - \frac{3}{7}\log_2\left(\frac{3}{7}\right) = 0.9852$$

Success for Big Star.

$$H(\text{No}) = -\frac{1}{3}\log_2\left(\frac{1}{3}\right) - \frac{2}{3}\log_2\left(\frac{2}{3}\right) = 0.91829$$

Success for Not Big Star.

$$IG(BS) = 1 - \frac{7}{10}(0.9852) - \frac{3}{10}(0.9182)$$

$$= 0.034859 \cong \boxed{0.0349.}$$

Ex: $IG(\text{Genre}) = H(\text{success}) - \frac{3}{10} H(SF) - \frac{6}{10} H(C) - \frac{1}{10} H(R)$

$$H(SF) = -\frac{1}{3}\log_2\left(\frac{1}{3}\right) - \frac{2}{3}\log_2\left(\frac{2}{3}\right) = 0.91829.$$

$$H(C) = -\frac{4}{6}\log_2\left(\frac{4}{6}\right) - \frac{2}{6}\log_2\left(\frac{2}{6}\right) = 0.91829.$$

$$H(R) = -\frac{0}{1}\log_2\left(\frac{0}{1}\right) - \frac{1}{1}\log_2\left(\frac{1}{1}\right) = 0.$$

ID3 Algorithm

$$IG(\text{Genre}) = 1 - \frac{3}{10} \times 0.9182 - \frac{6}{10} \times 0.9182 - \frac{1}{10} \times 0.$$

$$= \boxed{0.173539}$$

The attribute with greatest information gain will be root of the decision tree.

$IG(CO)$ is maximum, Hence CO is root node of the decision tree.

Country of Origin → US, EU, RW

US: Genre?

Now, to get the next node, calculate IG of Genre & Big Star using only 4 US Rows.

UNIT – 6

Approaches to build Intelligent Systems.
- Rule Based.
- Statistical Machine Learning System
- Deep Learning System
- Hybrid.

~~Ex: Lift~~

1. Rule Based System:
   Ex: Lift in a 3 storey building.

★ Conflict Resolution Algorithms:
1. Priority based
2. Longest – Matching strategy.
3. Most recently matching rules.

Algorithm

- Components of Rule Based System:
  - a database of rules (knowledge base)
  - a database of facts.
  - an interpreter, or inference engine.

  which are used to derive conclusion from rules and facts.

★ Forward Chaining: (Data driven reasoning).
- Uses deductions to reach a conclusion from a set of antecedents is forward chaining.

★ Backward Chaining:
tries to find a logical path from the conclusion to a set of antecedents to verify from the database.

Ex: Rules:

Priority ↓

Rule 1. $A \wedge B \rightarrow C$
2. $A \rightarrow D$
3. $C \wedge D \rightarrow E$
4. $B \wedge E \wedge F \rightarrow G$
5. $A \wedge E \rightarrow H$
6. $D \wedge B \wedge H \rightarrow$

Facts: 1. A
2. B
3. F

To Prove: H

	Facts.	Rules Triggered	Rule Fired.
	~~A,B,F~~		
1.	A, B, F.	1, 2.	1
2.	A, B, C, F	2	2.
3.	A, B, C, D, F	3	3.
4.	A, B, C, D, E, F	4, 5,	4.
5.	A, B, C, D, E, F, G	5	5
	A, B, C, D, E, F, G, H	6.	STOP.

- **Backward Chaining:**

	Facts	Goals	Matching Rules
1.	A, B, F	H	5.
2.	A, B, F	E	3.
	As A, B are Facts, once we get E, we will get H.		
3.	A, B, F	C, D	1
4.	A, B, C, F	D	2.
	A, B, C, D, F		STOP

★ **Comparisons between Forward Chaining & Backward Chaining.**

★ **Rule Based Expert System:**

- **People involved in Expert System:**
  1. Domain Expert. – domain knowledge to KE
  2. Knowledge Engineer – interact DE, people and convert into rules & meta rules & knowledge for conflict Resolution.
  3. End-User.

★★★ **Problem Characteristics:**

1. Is the problem Decomposable?
   ↳ Problem of Integration.
   - Yes.

   ↳ Block World Problem
   - No

2. Can Solution steps can be Ignored or undone?
   ↳ Mathematical Theorem.
   - Ignorable (Lema).

   ↳ 8 – Puzzle
   - Recoverable.

   ↳ Chess.
   - Not Recoverable.

3. Is the Universe Predictable?
   ↳ 8 – puzzle.
   - Yes.

   ↳ Bridge with playing cards.

4. Is a Good Solution Relative or Absolute?
   ↳ Any Path Problem (Solving 8-queen)
   - Absolute

   ↳ Best Path Problem (TSP).
   - Relative.

5. Solution / Path:
   ↳ 8 – Queen
   - Solution.

   ↳ 2 - Water Jug.
   - Path.

7. Is lot of knowledge base required?