UNIT-2

* Propositional logic is limited in its ability to represent complex relationships / detailed information.

* **Knowledge representation** :- methods used to encode info abt the world into a format that an AI system can understand & use it to make decisions

  - AI bridges b/w raw data and meaningful reasoning.

  - Predicates are a means of rep. Know.

* Predicate logic :-

  - First-order logic (FOL)

  - extends propositional logic (PPL) ⇒ represent rel. b/w objects & their properties.

  - Unlike PPL (T/F), it introduces predicates, variables, consts and quantifiers. → multiple objects & their interactions

  - AI uses this, to infer new info from existing facts, making it suitable for decision-making tasks.

ex:- John is the father of Mary ⇒ Peredicate Father (John, Mary)

→ **Components of PL** :

PVC

- predicates : returns T/F
  ex:- IsHungry (John)

- variables : placeholders for objs within domain. They allow us to represent general statements that apply to multiple objects.
  ex:- IsHungry (X)

- Consts : specific objects / entities in the domain
  ex:- John. in IsHungry (John)

→ Structure of Predicates:

- predicate symbol + arguments
- enhanced with quantifiers to specify the scope of the variables

- Predicate symbol : property / relationship being described
  - symbols are followed by arguments enclosed in parentheses.

- Arguments & Arity :
  - Specific objects that the predicate is applied to.
  - refers to the no. of arguements taken.

ex ← Married(x, y) → 2 args, arity = 2

- Quantifiers :-

→ Specifies the scope of variables

* Existential (∃)
  ↓
  there exists atleast 1 object satifies the given cond^n
  ∃x IsHuman(x) → atleast 1
  Negation :- no such that exists.
  ¬ → no one is human

* Universal (∀)
  ↓
  for all objects
  ∀x (IsHuman(x) → IsMortal(x))
  all human are mortal
  at least 1 exception
  ¬ → atleast 1 human is not

∧	→ conjuction	and
∨	→ disjunction	OR
⇒	→ implies	⇄ implication
¬	→ negation	Not
∀	→ for all	
∃	→ there exists.	

1. Simple Predicate: isHungry(John) — True if he is, False if he isn't

Appl^n in AI: In NLP based chatbots if user is hungry, it'd suggest hotels

2. Equality Predicates: E(x,y) ≡ (x = y) — x is equal to y; true if x = y

Appl^n in AI: AI based reasoning systems; robot warehouse, a robot may use this to determine if an object picked matches what is requested.

3. Mathematical predicate: X(a,b,c) ≡ (a+b+c = 0)

Appln: Optimization problems → check if constraints are satisfied / not.

4. Relationship Predicate :- M(x,y) ≡ x is married to y.

Appl^n :- family tree AI systems

5. Universal Quantification :- ∀x (IsHum(x) → IsMortal(y))

for all x, if x is human then x is Mortal.

Appl^n :- Knowledge based systems ⇒ medical diagnosis systems all virus can spread ofc.

6. Existential Quantification :- ∃x isHungry(x)

There exist at least one x, such that x is hungry.

Appl^n :- robot cleaning - there exist a task that req charging

7. Compound example with Multiple :- ∀x ∃y (Parent(x,y))

For all x / every person of x, there exists a person y such that x is parent of y.

Appln : Social network AI model to analyze relationships.

* Applns of predicate logic in AI :-

- Reasoning :- key regulate logical reasoning
  formulation of logical queries & inference new info from existing knowledge
  for prediction

X is parent of Y
if X is mum the
Y must be hum

- Database :- defines conds for querying & retrieving info
  SQL → filtering & searching records

- Expert systems :- designed to emulate the decision-making abilities of human expert in a specific dom
  - can capture rules, facts & heuristics

- NLP :- understands the semantics of sentences
  parses a sentence into Pred-Arg structures.

- ML :- used as features for training models

* Diff b/w PL vs PPL

PL	PPL
extends PPL by allowing vars & rel b/w objects	simple statements of T or F
uses predicates, var, const, quantifiers	uses atomic pp (simple statements)
more expressive [rel & poop]	less exp ; limited to T & F
Quantifiers [∃ & ∀]	none
Can rep complex rel	Can't
used in NLH, ML, SQL	decision-making systems
Complex	Simpler

Marcus was a man — man(Marcus)

Marcus was a pomp — pomp(Marcus)

Marcus was born in 40 A.D — born(Marcus, 40)

All men are mortal — $\forall x : man(x) \rightarrow mortal(x)$

All pomp died during volcano eruption in 79 A.D — $\forall x : pomp(x) \rightarrow died(x, 79)$

No mortal lives longer than 150 years — $\neg \exists x [mortal(x) \wedge livesLongerThan(x, 150)]$

$\forall x [mortal(x) \rightarrow \neg lives > 99]$

It is now 1991 — now = 1991

Alive means not dead. — $\forall x \forall y [Alive(x,y) \leftrightarrow \neg Dead(x,y)]$

$\forall x (Alive(x) \leftrightarrow \neg Dead(x))$

$\forall x \exists y \forall z : city(x) \wedge DogCatcher(y) \wedge Dog(z) \wedge lives(z,x) \Rightarrow bites(z,y)$

For every city, there exists a dogcatcher such that for every dog lives in city bites the dogcatcher.

$\forall n \exists a \forall t : Network(n) \wedge Admin(a) \wedge Threat(t) \wedge targets(t,n) \Rightarrow blocks(a,t)$

For every network, there exists an Admin such that for every threat that targets that network is blocked by admin

$\forall u \forall s : user(u) \wedge system(s) \wedge accesses(u,s) \Rightarrow authenticated(u) \wedge authorized(u,s)$

For all users, there exists systems such that the systems are auth by user and user is authorised to the system

↳ if a user accesses a systems, then user is auth & auth to access the sys

For every user and every device, if the device is stolen or compromised and it belongs to the user, then user revokes access for that device.

$\forall u \forall d : user(u) \wedge device(d) \wedge (stolen(d) \vee comp(v)) \Rightarrow revokeAccess(u,d)$

All users have Access

$\forall x : (u(x) \rightarrow A(x))$

There exists a threat that can exploit the system

$\exists x : (t(x) \rightarrow e(x,y))$

There ~~exists~~ is an incident concerning the server i.e, of ↑ severity

$\exists x : (I(x) \wedge c(x,y) \wedge \dots)$

All devices are monitored

$\forall x : (D(x) \rightarrow M(x))$

Every admin can manage policies

$\forall x : (A(x) \rightarrow P(x))$

there exists a breach that compromised data & is costly

$\forall x\,(Vuln(x) \rightarrow \neg Sec(x))$

$\exists P\,(Alert(P) \wedge Triggered\,By(P, Malware))$

$\exists K\,(Key(K) \wedge UsedIn(K, encrypt) \wedge \neg valid(K))$

* **Resolution:**

- a proof technique in predicate calculus, uses logical inference to determine the truth of statements.
- converting statements ⇒ clause form ⇒ +ing the negation to prove → deriving the contradiction
- Inference [new knowledge]

→ **Refutation based:-** proving statement by showing that its negation leads to contradiction.

Steps:- Conversion to clause form [predicate ⇒ conjuctive normal form]

Negation

Resolution : applied to pairs of clauses that contain complementary literals [a literal & negation ⇒ new clause (resolvent)]

Contradiction: process continues until empty clause is derived, proving the original statement.

ex:- If a user logs in from a blacklisted IP addr and the login attempt is unauth, then sys shld trigger an alert.

$$\forall x\,(Black(x) \wedge unauth(x) \rightarrow Alert(x))$$

Step1:- CNF:- $\neg(B(x) \wedge Auth(x)) \vee A(x) \rightarrow \neg B(x) \vee \neg Auth(x) \vee A(x)$

$\neg Black(x) \vee \neg unauth(x) \vee Alert(x)$

negation:- Alert didn't trigger

$\therefore \neg Black(x) \vee \neg unauth(x)$

$\neg Alert(x)$

$P \rightarrow Q \equiv \neg P \vee Q$

$\neg(\neg P \vee Q) \equiv P \wedge \neg Q$

$\Rightarrow \forall x\,(B(x) \wedge U(x) \rightarrow A(x))$

step 1:- negation

$p \rightarrow q \equiv \neg p \vee q$

$\neg(B(x) \wedge U(x)) \vee A(x)$

$\Rightarrow \neg B(x) \vee \neg U(x) \vee A(x)$ —— ①

step 2:- Literal negation $\neg(\neg p \vee q) \Rightarrow p \rightarrow \neg q$

$\Rightarrow \neg A(x)$ —— ②

$(B(x) \vee U(x)) \rightarrow \neg(A(x))$

step 3:- Resolution:-

literals from ① & ② cancel out e/o

$\neg B(x) \vee \neg U(x)$
$B(x) \vee U(x)$] → cancel out e/o

∴ empty clause

Contradicts → ∴ $\neg A(x)$ is false

statement is valid.

---

* structured representations organise knowledge in a way that captures relationships, hierarchies and dependencies among various entities/concepts

Types:-

Semantic networks: graph structures where nodes represent obj & edges rep relationships among them.

Semantic frames ontologies

frames:- Data structures for representing stereotypical situations various attributes/slots ⇒ can have default values or be filled w specific values

Ontologies:- formal rep of set of concepts within a domain, and rel b/w those concepts, serving to share common vocabularies for knowledge representation

* Inference: new knowledge from existing knowledge after logical rules is being applied.

Prodn based / frame based.

Prod^n based ⇒ set of rules to derive conclusions from set of known facts.

rule: If cond^n then action

A set of facts can be manipulated by the rules

Prod^n cycle: system's operation involves cycle where it checks the working memory against the rules & applies applicable rules to derive new facts / actions.

Frame based ⇒ utilises frames to rep stereotypical situations & allows inheritance to prop. from more general to more specific ones

Slots & fillers: Slots in frames can hold values which allows for organizing info hierarchically.

Defaults & inheritance: Default values can be overridden by specific instn.

more specific frames can inherit from more general frames.

K.R & inference are foundational to AI, enabling sys to emulate human understanding & reasoning.

structured rep like semantic networks, frames & ontologies → organize know.

systems based on prod^n & frames support process of inference

* Structured Representations of Knowledge in CYS.

→ Attack Taxonomies: MITRE ATTACK framework categorizes diff techniques, tactics & procedures that adversaries use in real life attacks. [understand potential threats]

→ Security Policies: - Stru. rep of S.P using a markup lang like XACML → define access control rules clearly enforce policies in IT

→ Threat Models: STRIDE Model (Spoofing, tampering, Repudiation, info. disclosure, DoDs, eleration of Privilege)

⇒ identify & cate sec. threats

→ Intrusion Detection :- compromised → unusual login patterns & other contextual info.

→ Threat Intelligence Aggregation : collect → use inference rules → recent malware activity → update defenses → patch vuln

* AI models in CYS :-

→ Anomaly Detection Algo : Isolation forest, One-class SVM → identify deviation from normal behave in network traffic.

→ NLP :- analyze threat reports, sec blogs to extract emerging threats.

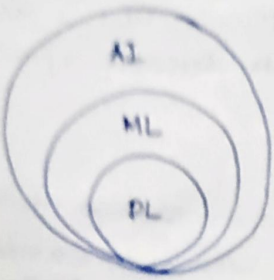
→ Generative Adversial Networks (GANs) → simulate attack patterns to generate synthetic data for training intrusion detection systems → helping to be ready against novel attacks.

→ Deep learning :- CNNs can classify traffic patters in static & dynamic analysis of execu.

→ Graph based Neural Net : rel b/w users, devices & transactions in a system. (Inde compromised accounts)

Attack Taxonomies
Sec. Policies
Intrusion detection
Threat models
Threat Intelligence Aggr.

UNIT-3



AI - Machines exhibit human-like intelligence

ML - Subset of AI ; sys learns from data

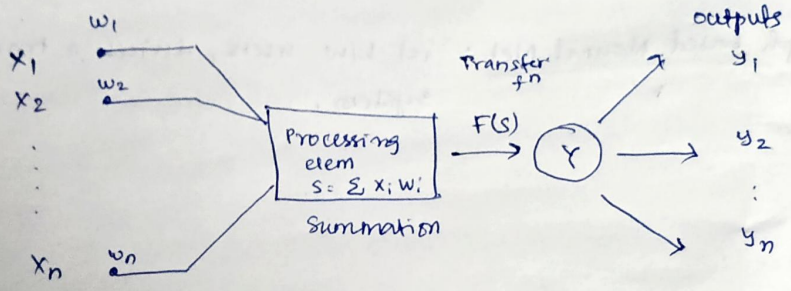
DL - Subset of ML, uses multi-layered NN

What is DL?

- uses multi-layered of non-linear processing units
- performs feature extraction & transⁿ hierachly
- learn via supervised / un-supervised
- build levels of abstraction

Why DL?

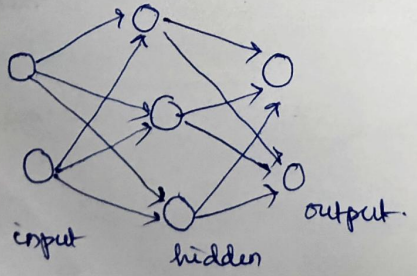
- Limitations of ML :-
  - poor w ↑ dim data
  - req manual feature extn
- Adv of DL
  - handles ↑ dim data
  - Auto feature extraction
  - computationally expensive but powerful.



ANN

* FFNN (Feedforward Neural Network) :-

- most fundamental & widely used
- single layer perceptron to



Input :- consists of neurons = recieve data
each neurons = feature of inpw data

output :- final output of network
no. of neurons = no. of classes in classification problem / no. of outputs in reg^n problem

hidden :- 1 or more are placed blw I & O
resp^le for learning complex patterns in data
each neuron = wt. sum of inputs ⊕ non-linear act^n f^n.

- dataset ↳ type of challenge determine the no. of neurons in final layer ↳ in layer.
- total ↳ error will be used to determine the no. of neurons in hidden layer ↳ no. of HL

- seq of inputs enter the layer ↳ are multiplied by the wts in the model.
- cal. input values are then summed tgt to form a total.
- each connection b/w neurons → associated wt that is adjusted during training to min the error in pred^n

- wt applied to each input to an artifical neuron
  1st input × wts
  then biased is applied } weighted sum
- wted sum ⇒ activation f^n.
- if sum of value > predetermined 'x' normally 0
  output = 1
  else = 0

NN can compare output of nodes → & rule
⇒ alter wts.

- train' & learn' process results ⇒ ∇ descent
- back propogation ⇒ updating wts in multi perceptron layer.

* CNN (convolutional Neural Network) :- max pooling layers

batch size = 64 ⇒ 64 images

- used for img processing, obj detection, videos / imgs.
- auto & adaptively learn spatial hierarchies of features by using conv. layers.

epoch ⇒ total no. of times

Gradient (∇) : how much f^n's output changes when input changes a little.
(Slope of f^n)

∇ descent : opt. algo that uses ∇'s to min errors in ML to train NN.

① Start at random pt
② Move in dir^n of steepest descent.
③ re compute the ∇ at new position
④ Repeat until f^n is close to 0

	Perceptron	Neuron
output	0 or 1	cont^n / non-linear
Activation f^n	Step f^n	Sigmoid, Softmax etc
Complexity	Simple, handles linearly separable problems	Complex, handles non-linear multi-dim problems
layers	Single layer (original)	Multi-layer (I, HL, O)
learning	Basic learning rule for linear problems	∀ descent w backpropagation
	Binary classification	Classification, regression & various deep learning tasks.

Components :-

inputs :- $x_1, x_2 \cdots x_n$ (several)

wts :- $w_1, w_2 \cdots w_n$ (each input is associated)

bias (b) → to shift decision boundary.

Activation f^n :- step f^n if it is above/below λ.

$$y = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} w_i x_i + b > 0 \\ 0 & \text{if } \sum_{i=1}^{n} w_i x_i + b \leq 0 \end{cases}$$

Binary output :- linearly seperable classification problems.

ex :- OR gate (+)

A	B	Y = A+B
0	0	0
0	1	1
1	0	1
1	1	1

let $w_1 = 0.6$

$w_2 = 0.6$

$\lambda = 1$

$\eta = 0.5$

⇒ A = 0, B = 0, target = 0

$w_i x_i = 0.6 \times 0 + 0.6 + 0 = 0 < \lambda$

$= 0.$ ✓

A=0, B=1

$\sum w_i x_i = 0 + 0.6 = 0.6 < \theta$

$= 0$ ✗

∴ we update the weights.

$$w_i = w_i + \eta (t - o) x_i$$

η = Learning rate

t = target

o – Actual output

$w_1 = 0.6 + 0.5 (1-0) \times 0$

$= 0.6$

$w_2 = 0.6 + 0.5 (1-0) \times 1 = 1.1$

A=0, B=0

$\sum w_i x_i = 0$ ✓

A=0 ; B=1

$\sum w_i x_i = 0 + 1.1 = 1.1 > 1$ ✓

A=1 ; B=0

$\sum w_i x_i = 0.6 < 1$ ✗

$= 0$

$w_i = w_i + \eta (t-o) x_i$

$w_1 = 0.6 + 0.5 (1-0) 1 = 1.1$

$w_2 = 1.1 + 0.5 (1-0) \times 0$

$= 1.1$.

A=0 ; B=0

$\sum x_i w_i = 0$ ✓

A=0 ; B=1

$\sum w_i x_i = 0 + 1.1 = 1.1 > 1$

$= 1$ ✓

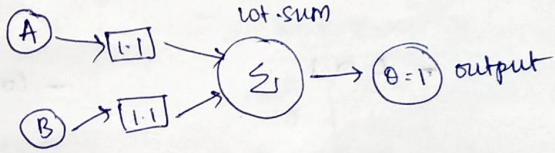
A=1 ; B=0

$\sum w_i x_i = 1.1 > 1 = 1$ ✓

A=1 ; B=1

$\sum w_i x_i = 2.2 > 1$

$= 1$ ✓



ex:- AND gate (•)

let $w_1 = 0.6$

$w_2 = 0.6$

$\eta = 0.5$

$\theta = 1$

A	B	Y = A·B
0	0	0
0	1	0
1	0	0
1	1	1

A=0 ; B=0

$\sum w_i x_i = 0$ ✓

A=0 ; B=1

$\sum w_i x_i = 0.6 < 1$

$= 0$ ✓

A=1 ; B=0

$\sum w_i x_i = 0.6 < 1$

$= 0$ ✓

A=1 ; B=1

$\sum w_i x_i = 0.6 + 0.6$

$= 1.2 > 1$

$= 1$ ✓

done

Limitations of linear Note & Perceptrons

- Linear Seperability :- can only solve if it is linearly sep.
  - can't solve more complex like XOR
- Single layer model limitation } → doesn't have hidden layers, limiting its expressiveness
- Only 2-class output
- Sensitivity to Noise data (outliers, change in output)

* Multi-Layer Perceptrons (MLP)

SLP	MLP
- One input, output	- I, Hidden, output
- Solves only linearly seperable problems	- both linear & non-linear
- Step fn (binary output)	- Non-linear fn's
- perceptron learning rule	- trained using backpropogation & gradient descent.
- 0 or 1	- Contin^n or multi-class
- Simple & limited	- Complex & powerful.

ex:- XOR gate.

$y = x_1\bar{x}_2 + \bar{x}_1 x_2$

$y = \bar{z}_1 + z_2$

where, $z_1 = x_1\bar{x}_2$

$z_2 = x_2\bar{x}_1$

$y = z_1 \odot z_2$ (XOR)

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

x_1	x_2	z_1
0	0	0
0	1	0
1	0	1
1	1	0

$w_{11} = w_{21} = 1$, $x = 1$, $\eta = 1.5$

$A=0, B=0$
$\sum w_i x_i = 0$ ✓

$A=0, B=1$
$\sum w_i x_i = 1$ ~~same~~

$\therefore w_{11} = 1 + 1.5(0-0)\times 0 = 1$
$w_{21} = 1 + 1.5(0-1)\times 1 = -0.5$

$x_1 \bar{x}_2$
$= x_1$ and $\bar{x}_2$

$A=0, B=0$
$\sum w_i x_i = 0$ ✓

$A=0, B=1$
$\sum w_i x_i = 0 + (-0.5) = -0.5 < 1$
$= 0$ ✓

$A=1, B=1$
$\sum x_i w_i = 1 + 0 = 1$ ✓

$A=1, B=1$
$\sum w_i x_i = 1 - 0.5$
$= 0.5 < 1$
$= 0$ ✓

new, $z_2 = \bar{x}_1 x_2$

x_1	x_2	z_2
0	0	0
0	1	1
1	0	0
1	1	0

$A=0, B=0$
$0 = 0$ ✓

$A=0, B=1$
$\sum = 0 + 1 = 1 = 1$ ✓

$A=1, B=0$
$\sum = 1 + 0 = 1 \neq 0$ ✗

$w_{12} = 1 + 1.5(0-1)1 = -0.5$
$w_{22} = 1 + 1.5(0-1)0 = 1$

$A=0, B=0$
$0 = 0$ ✓

$A=0, B=1$
$\sum = 0 + 1 = 1 = 1$ ✓

$A=1, B=0$
$\sum = +0.5 + 0 = -0.5 < 1$
$= 0 = 0$ ✓

$A=1, B=1$
$\sum = -0.5 + 1 = 0.5 < 1$
$= 0$
$= 0$ ✓

final function:-

$y = z_1 + z_2$

z_1	z_2	y
0	0	0
0	1	1
1	0	1
0	0	0

$A=0, B=0$
$\sum = 0 = 0$ ✓

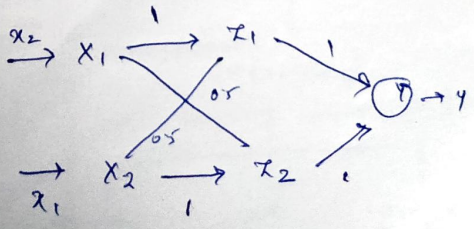
$A=0, B=1$
$\sum = 0 + 1 < 1$
$= 1$ ✓

$A=1, B=0$
$\sum = 1 + 0 = 1 = 1$
$= 1$ ✓

$A=0, B=0$
$\sum = 0 = 0$ ✓

$\therefore w_{11} = 1$ $\quad w_{21} = -0.5$
$w_{12} = -0.5$ $\quad w_{22} = 1$

$w_1 = 1$
$w_2 = 1$



ex:- XAND ⇒ $\overline{XOR}$ (XNOR)

$\overline{x_1\bar{x}_2 + \bar{x}_1 x_2} = (\overline{x_1\bar{x}_2})(\overline{\bar{x}_1 x_2})$

$= (\bar{x}_1 + x_2)(x_1 + \bar{x}_2)$

$w_0 = w_{12} = 1$

$x = 1$

$n = 1.5$

$+ \rightarrow \alpha$

$\cdot \rightarrow$ AND

$x_1 = \overline{x_1} + x_2$

x_1	x_2	I_1
0	0	1
0	1	1
1	0	0
1	1	1

A=0 , B=0

$\Sigma = 0 \neq 1$ ✗

$w_{11} = 1 + 1.5(1-1)\times 0 = 1$

$w_{12} = 1 + 1.5(1-1)\times 0 = 1$

∴ not defined.

ex:- NAND ⇒ $\overline{x_1 \cdot x_2}$

$= \overline{x_1} + \overline{x_2}$

x_1	x_2	I_1
0	0	1
0	1	1
1	0	1
1	1	0

A=0 | B=0 : $\Sigma = 0+1.5 = 1 = 1$ ✓

A=0, B=1 : $\Sigma = 1+1.5 = 1$ ✓

A=1 , B=0 : $\Sigma = 1+1.5 = 2.5 >$ ✓

A=1 B=1

$\Sigma = 1+1+1.5 = 1$ ✗

$w_1 = 1 + 1.5(0-1)\times 1 = -0.5$

$w_2 = 1 + 1.5(0-1)\times 1 = -0.5$

A=0, B=0 : ⇒ 0 ✓

A=0, B=1 : $\Sigma = -0.5+1.5 = 1$ ✓

~~$w_1 = -0.5 + 1.5(0-1)\times 0 = 0$~~

~~$w_2 = -0.5 + 1.5(0-1)\times 1 =$~~

A=1 , B=0 : $\Sigma = -0.5+1.5 = 1$ ✓

A=1 , B=1 : $\Sigma = -0.5 - 0.5 + 1.5 = 0.5 <$

$= 0 = 0$ ✓

∴ $w_1 = 0.5$, $w_2 = -0.5$

AI – John McCarthy
– science engi
comp progra
– developing
intelliga
– knowledg
searches → search
→ pattern
– Rear
– infer
– focu
– qual
– use
– pro
problem Sol
PAS

UNIT-1

AI - John McCarthy
- science engineering of making intelligent machine esp intelligent comp programs
- developing concepts, mechanisms, understanding biological intelligent behav.
- knowledge is collection of facts
- search
- pattern recog.
- Reasoning, pattern reco, learning
- inference
- focus on non-Algo exn reliance on heuristic search.
- Qualitative reasoning
- use of domain-specific knowledge
- sufficient answers

problem Solving :-

problem space : entire range of possible states, actions & transitions relevant to solving a specific problem.

Agents : Perceive environments & take actions to achieve goals

states : represent a specific config / situation within a prob.
8 — initial, current, operator, goal

PM

Search Algo

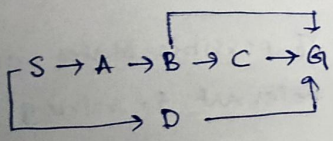
- un-informed (Blind)
  - DFS
  - BFS
  - uniform cost search
- other :- Bidir search, minmax
- informed (Heuristic)
  - greedy
  - A*
  - graph

* Uninformed Search
  - blind, explored systematically w/o addⁿ knowledge.
  - DFS :-
    → exploring deeply as possible before backtracking
  - BFS
    → all neighbouring nodes at current depth before moving to next level.
  - Uniform Cost Search : Similar to BFS, cost is calculated.

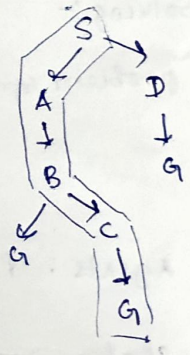
* Informed Search
  - Heuristic, to search more eff.
  - greedy BFS : estimate cost to reach the goal, selecting most promising path
  - A* search : combines cost of reaching node + H. est. (pathfind)
  - Genetic Algo : inspired by natural selection, these Algo evolve solns over generations to find optimal solns.

* Other :- Bidir, Minmax

DFS :-



depth ⇒ S → A → B → C → G.



BFS ⇒ S → D → G

	DFS	BFS
time complexity	$T(n) = 1 + \dots n^d = O(n^d)$	$T(n) = O(n^s)$
space	$S(n) = n \times d$	$S(n) = O(n^s)$
Completeness	complete if finite	if soln exists
Optimality	not optimal, cost spent is high.	optimal, if edges are of same length

* Heuristic → thumb rule / guiding principle i.e, used to make intelligent decisions / solve problems that are encountered during the problems.

optimization

(1) - decomposability
  - decomposable ex: chess, speech reco
  - non-decomposable. ex: Img reco, Auto vehicle navi.

(2)
- Ignorable : prev steps doesn't need in future
  ex: pathfinding

vs

- Irrecersible : involve actions that can't be undone.
  ex: autonomous driving.

(3)
- Deterministic : outcome of the problem is fully predictable based on inputs. (no need of prob. reason)
  ex: tic-tac-toe

vs

- Stochastic : need randomness, uncertainity, probabilistic decisions.
  ex: weather forecasting.

(4)
- static : environment doesn't change, AI process & give soln.

vs

- Dynamic : environment keeps changing as AI operates.

(5)
State (single) : soln is a specific state rather than a sequence of actions.

vs

path-based : determine sequence of paths.

(6)
Knowledge-based : prior knowledge & pre-defined rules.

vs

Data-driven : AI learns patterns from data.

(7)
Simple vs Complex

Goal oriented — optimization — good
— Classification — better
— Prediction. — future events

* Problem Solving Steps :-

Defining the problem [nature (I/O), expected output, constraints, available data]
↓
Data Collection & preparation
Collect relevant, feature, normalization
↓
Right Algo [search, ML (Sup, Unsup, Reinf), Knowledge-based]
↓
Model training and optimization [train, hyper parameter tuning, eval metrics]
↓
Deployment & Contin° monitoring

* problem representation :-

problem graphs
state-space rep.
pattern matching
constraint matching
Data Indexing
knowledge indexing

* Future Trends :-

Adv NN
GPT, speech

AI ethics & responsible AI
making critical desi--,
companies → bias detection
fairness auding

XAI (explainable)
insight of how AI
works

* A* Algorithm
— known for
— h(n) →
— eval
— Optim

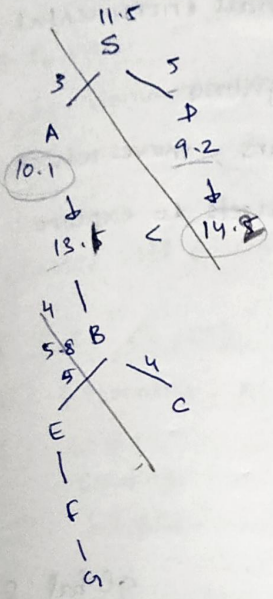
ex :-

Disad
me

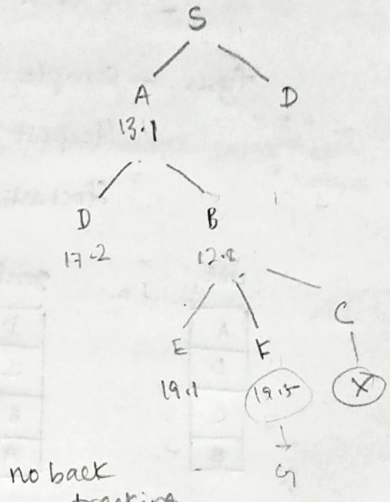
ex

* A* Algorithm
- known for balance b/w eff & optimality (outperform Dijkstra / BFA)
- h(n) → estimates cost from given node (n) to the goal.
- eval fn :- f(n) = g(n) + h(n)
- Optimality & completeness :- shortest path → est. cost

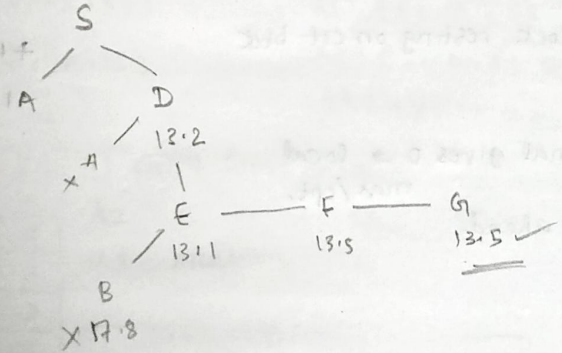
ex :-



⇒ path

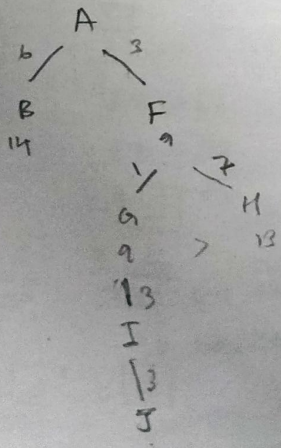


no back tracking



DisAdv :- memory usage.

ex :-

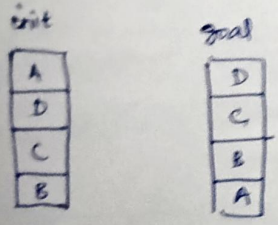


A → F → G → I → J.

* Hill Climbing :

- local search Algo by iteratively improving
- In good state a large set of inputs, hill climbing can find a really good soln it may not always give optimum max
- used where only max a real fn.
- It is a local search (small incremental changes)

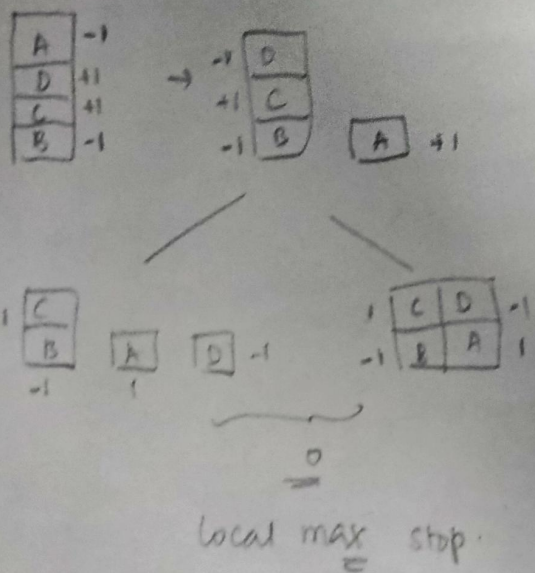
types - Simple - only one neighbour
Steepest - all neighbours, move which has good improv.
Stochastic - randomly selects to explore



local heuristic fns.

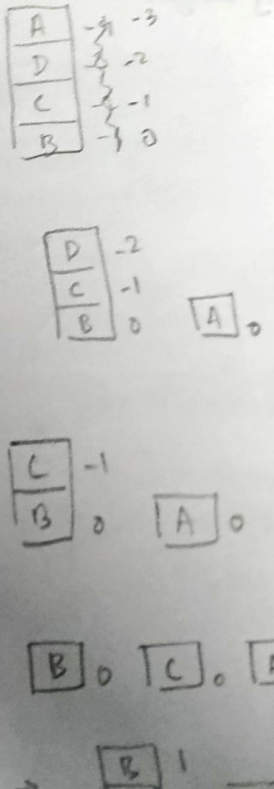
+1 for the block resting on crt block
-1 if not

→ if both optimal gives 0 ⇒ local max/opt.
can't go ahead.



global heuristic fn's

+1 for every all block crt
-1 for not.



* Constraint Satisfaction
- set of objects
X - set of var
D - set of
C - set of

ex :- no 2 class
X: 9
D:
C:

ex :- 3 Analy
diff

Soln :-

* Bac

ex :-

* Constraint satisfaction problems (CSP):-

- set of objects whose state must satisfy several constraints & condns.

X - set of var

D - set of domains of values for each var

C - set of constraints that specify the allowable combns of values

ex:- no 2 classes shld occur at same time.

X: {C1, C2, C3}

D: {Morning, Afternoon, Eve}

C: C1 ≠ C2
C2 ≠ C3
C3 ≠ C1

ex:- 3 Analysts {A1, A2, A3}
diff sys {S1, S2, S3}

A1: {S1 ↑ ; S2 : ↓ , S3 : Mid}
A2: {S1 Mid ; S2 : ↑ , S3 : ↓}
A3: {S1 : ↓ , S2 : Mid ; S3 : ↑}

C: A1 can't work on S2
A2 " S3
A3 must S1

Soln:- A: A1, A2, A3
S: S1, S2, S3

A1 ≠ S2
A2 ≠ S3
A3 ≠ S1

A3 → S1
A2 → S2
A1 → S3

* Backtracking :- Chks for constraints & vars values, if 'C' violated, it backtracks. to try diff value.

ex:- C1 → Mrng
C2 ≠ Mrng So, we backtrack for C2

* Constraint propogation :- Techniques that reduce the search space of CSP by updating domains of vars based on Constraints.

Arc consistency
MAC's Algo

ex :- $C_1 \rightarrow$ Ming then
C_2 domain can be reduced to {AN, eve}.

ex :- A, B, C ⇒ no 2 adj → same color.

Domain → {Red, Green, Blue}

Constraints → $A \neq B$
$A \neq C$
$B \neq C$

if A → red
then B → Blue
C → green
} any combo.

Arc Consistency :- every value of var x, there is a value of var Y, that satisfies the binary constraint.

* Probabilistic ; uncertainity can be modelled by prob.

Random var : they represent uncertain quantities

JPD : every prob comb^n.

Baye's theorem :- update the prob of hypothesis as more evidence becomes available.

ex :- P(P(x)) = 0.6
P(M(x)) = 0.4
S(P(x)) = 0.9 , S(P(M(x))) = 0.7

P(S(x)) = ? = 0.9 × 0.6 + 0.7 × 0.4
= 0.82 = 82%

AI Appl^{n}s

- Face & obj detection from footage
- intelligence extraction from text
- image tamper & deep fake detection
- Threat pred^n from
- Anamoly detection in user behav.

* Measure of Performance :-

Accuracy
Time Complexity
Space Complexity
Quality of Solutions