day / date:

What is AI?

- Artificial — perception, understanding, predict, manipulate
  → produced by human art or effort, rather than originating naturally.
- Intelligence
  → ability to acquire knowledge and use it

* study intelligent part concerned with humans
* represent those action using computer.

	Thought	Behaviour
	System that think like humans (2)	System that act like humans (1)
	System that think rationally (3)	System that act rationally (4)
	Human	Rational

(1) Acting Humanly: Turing Test Approach

- a computer passes a test, if a human interrogator after posing some written questions, cannot tell whether the written response come from a person or a computer.
- natural language processing → for communication with human
- knowledge representation → to store what it knows or hears
- automated reasoning → use the stored information to answer questions and draw new conclusions.
- machine learning → adapt to new circumstances and detect patterns.

(2) Thinking Humanly: Cognitive Modeling Approach

- need to get inside the actual workings of human minds
- introspection → trying to catch our own thoughts.
- psychological experiments → observing a person in action
- brain imaging → observing brain in action.

(3) Thinking rationally: "Laws of thought" approach

- logic can't express everything
- logical approach often not feasible in terms of computation time, loops forever.

(4) Acting rationally. Rational agent approach

- agent is something that acts; rational agent is the one that acts so as to achieve the best outcome or when there is uncertainty, the best expected outcome.
- rational agents are general
- one way to act rationally is to reason logically to the conclusion that a given action will achieve one's goals and then to act on that conclusion.

Agents

- It is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators
- percept → agents perceptual inputs at any given instant.
- percept sequence → complete history of everything the agent has ever perceived.
- agent function → that maps any given percept sequence to an action.
- agent program → agent function for an artificial agent will be implemented by agent program.

Vaccum cleaner World

- a world with 2 locations, where vaccum cleaner can either move left, right, suck or do nothing.
- performance measure will be the amount of dirt cleaned up in single eight hour shift.

Task Environment

- A rational agent selects actions that maximize its expected utility.
- Task environment are essentially the problem to which rational agents are the solution.

P - Performance

E - Environment

A - Actuators

S - Sensors

Properties of Environment

FULLY OBSERVABLE VS PARTIALLY OBSERVABLE

- If an agents sensor give it access to the complete state of the environment at each point in time, then the task environment is fully observable.
- when an agent can determine the state of system at all times it is fully observable.
- an environment might be partially observable because of noisy and inaccurate sensors or missing parts.

SINGLE AGENT VS MULTIAGENT

- environment may contain other agents which may be of same or different kind as that of the agent.
- e.g. chess, playing soccer is multiagent
- e.g. solving a crossword is single-agent.

COMPETITIVE VS COOPERATIVE

- if B's behaviour is best described as maximizing a performance measure whose value depends on agent A's behaviour e.g. chess.
- avoiding collisions maximizes the performance measure of all agents so a partially cooperative multiagent environment; but partially competitive when one car can occupy a parking space

day / date:

DETERMINISTIC vs STOCHASTIC

- If the next state of the environment is completely determined by the current state and the action executed by the agent, then environment is deterministic.
- If environment is partially observable, it could appear to be stochastic
- an environment is uncertain if it is not fully observable or not deterministic.

EPISODIC vs SEQUENTIAL

- in an episodic task environment, the agents experience is divided into atomic episodes.
- In each episode, the agent receives a percept and then performs a single action, the next episode does not depend on the actions taken in previous episodes.
- in sequential, current decision could affect all future decisions

STATIC vs DYNAMIC

- if an environment can change while an agent is deliberating, then environment is dynamic
- in static environment, agent need not keep looking at the world while it is deciding on an action
- if environment does not change with time, but agents performance score does, then semidynamic
- crosswords/chess are static, taxi driving is dynamic

DISCRETE vs CONTINUOUS

- if there are limited number of distinct, clearly defined, states of environment, it is discrete e.g. chess
- taxi-driving is continuous

KNOWN vs UNKNOWN

- in a known environment, outcomes for all actions are given
- for unknown, agent will have to learn how it works to make good decisions
- known environment can be partially observable, and unknown can be fully observable

Structure of Agents

Agent = architecture + program

the machinery that an agent executes on	an implementation of an agent function.

- agent program takes current percept as input, agent function takes the entire percept history.

Types of Agents

1. Simple Reflex Agent

- agents select actions on the basis of the current percept. e.g vaccum cleaner checks if dirt is present in that specific room or not.
- based on condition - action rule e.g. as the driver of the taxi, you see that the car in front brakes and brake lights are on, then you should initiate breaking
- some processing is done on the visual input to establish the connection/condition. meaning if-else statements
- their environment is fully observable.

2 Model Based Reflex Agent

- agent should maintain some internal state that depends on the percept history and reflects some of the unobserved aspects of the current state.
- need information about how the world evolves independently of the agent (a), and how the agents own actions affect the world (b).
- model → how the world works.

3 Goal Based Agent

- Knowing something about the current state is always not enough, the agent might have some sort of goals and there are many directions to choose from.
- more flexible because the knowledge that support its decision is represented explicitly and can be modified.

4. Utility Based Agent

- a more general performance measure should allow a comparision of different world states according to exactly how happy they would make the agent.
- a rational utility based agent chooses the action that maximizes the expected utility of the action outcome.
- has to model and keep track of environment

Learning Agents (c)

- performance element → selecting external action, agent function.
- learning element → makes improvement by observance performing
- critic → feedback on how the agent is doing and how it should be modified
- problem generator → suggest other possible courses of actions.

day / date:

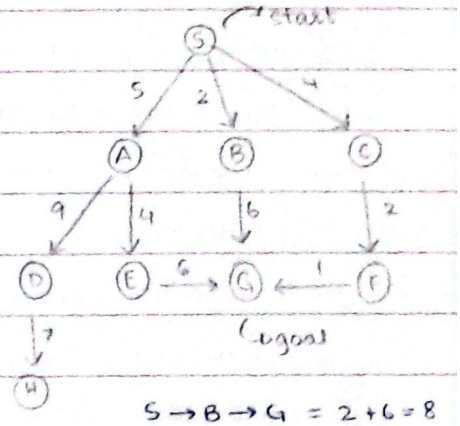
Uninformed Search Strategies

- generate successors and distinguish a goal state from a non-goal state

1. Breadth-first search

- use of FIFO queue
- expand all nodes at depth (i) before expanding nodes at depth (i+1)

expanded node	frontier list
	S
S	A, B, C
A	B, C, D, E
B	C, D, E, G
C	D, E, G, F
D	E, G, F, H
E	G, F, H, G
G → goal	F, H, G

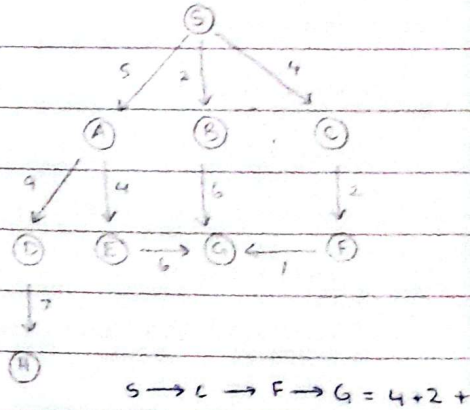


S → B → G = 2+6 = 8

2. Uniform Cost Search

- expansion with lowest cost path
- fringe = queue ordered by path cost
- optimal and complete

expanded node	frontier list
	S
S	B:2, C:4, A:5
B	C:4, A:5, G:6+2
C	A:5, F:4+2, G:6+2
A	F:4+2, G:8, E:5+4, D:14
F	G:4+2+1, G:8, E:9, D:14
G → goal	G:8, E:9, D:14



S → C → F → G = 4+2+

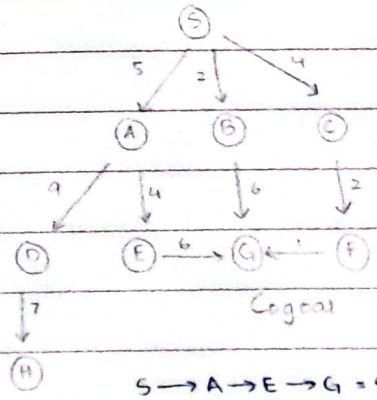
C^* : cost of optimal solution.

day / date:

3. Depth First Search

- expand deepest node
- fringe is a LIFO stack
- not optimal as it finds leftmost solution, not complete → $O(b^m)$ (m: max depth)

expanded node	frontier list
	S
S	A, B, C
A	D, E, B, C
D	H, E, B, C
H → no exp	E, B, C
E	G, B, C
G → goal	B, C

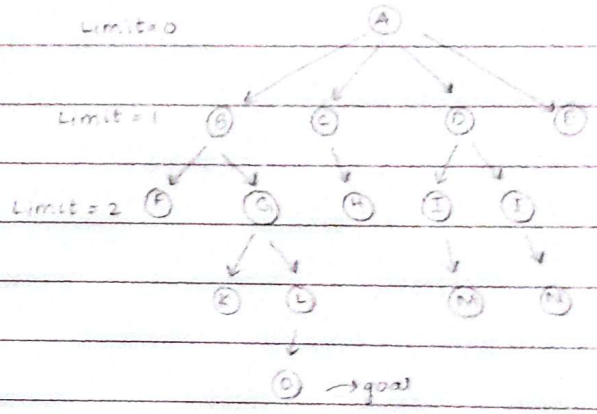


S → A → E → G = 5 + 4 + 6 = 15

4. Depth limited search

- searching not permitted beyond the depth bound
- incomplete search algorithm if goal beneath depth bound
- complete if goal state at depth less than L, but not optimal

expanded node	frontier list
	A
A	B, C, D, E
B	F, G, C, D, E
F → no exp	G, C, D, E
G → no exp	C, D, E
C	H, D, E
H → no exp	D, E
D	I, J, E
I → no exp	J, E
J → no exp	E
E → no exp	{ }

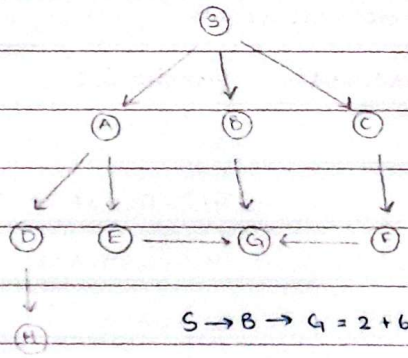


day / date:

5. Iterative Deepening search

- gradually increases limit from 0, until goal is found.
- finds best depth limit
- combines BFS and DFS
- optimal when path cost is a nondecreasing function of the depth of the node.

	expanded node	frontier list
		S
D=1	S	A, B, C
D=1	A	B, C
D=1	B	C
D=1	C	{ }
D=2	S	A, B, C
D=2	A	D, E, B, C
D=2	D → no exp	E, B, C
D=2	E → no exp	B, C
D=2	B	G, C
D=2	G → goal	C



S → B → G = 2 + 6 = 8

6. Bi-directional search

day / date:

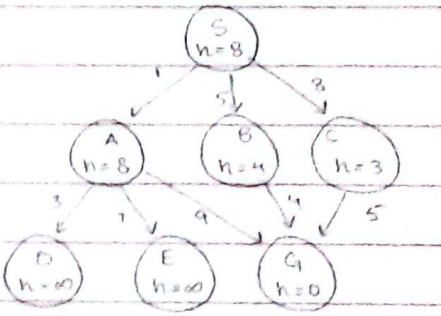
Informed (Heuristic) Search

- heuristics are informed guesses
- estimates the goodness of node n, how close node n is to a goal
  and the estimated cost of minimal cost (cheapest) path from node n to a goal

1. Greedy Best First Search

- $f(n) = h(n)$, sorting nodes in the frontier by increasing values of f
- ignore edge weights, cost is calculated by actual cost

expand node	frontier list
	S
S	C:3, B:4, A:8
C	G:0, B:4, A:8
G → goal	B:4, A:8



S → C → G = 8 + 3 + 0 = 13

- not optimal, incomplete

2. A* Search

- avoid expanding paths that are already expensive
- evaluation → $f(n) = g(n) + h(n)$

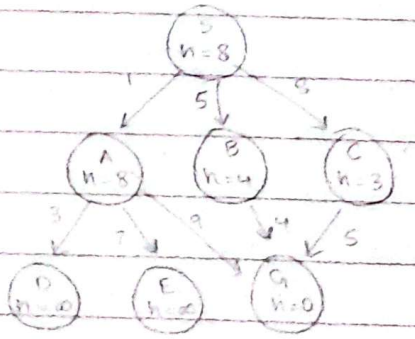
estimated cost of the cheapest path through n to goal

cost so far to reach n from start node.

estimated cost of cheapest path from n to goal

- is complete and optimal
- when $h(n) \leq h^*(n)$ holds true for all n, h is called admissible heuristic function.
- $h^*(n)$ is the actual cost of the minimum cost path from n to a goal

n	g(n)	h(n)	f(n)	$h^*(n)$	
S	S → S = 0	8	8	9	→ S → B → G
A	S → A = 1	8	9	9	→ A → G
B	S → B = 5	4	9	4	→ B → G
C	S → C = 8	3	11	5	→ C → G
D	S → D = 1+3	8+∞	4+∞	∞	→ D → G
E	S → E = 1+7	8+∞	8+∞	∞	→ E → G
G	S → G = 10/9/13	0	10/9/13	0	→ G → G



day / date:

- checking admissibility since $h(n) \leq h^*(n)$ for all n, which is true in this case.
- for consistency it should satisfy following condition for every node n

★ what if this is not true, what will be the next step? or will we just stop over here

$$h(n) \leq c(n, n') + h(n')$$

- $h(n)$: estimated cost of node n to the goal
- $c(n, n')$: actual cost of moving from $n \to n'$
- $h(n')$: estimated cost from n' to goal

- does not predict shorter distance than reality
- estimated cost is less or equal to real cost + estimate of next node.
- A* will not revisit node with lower cost if consistent

edge $(n \to n')$	$h(n)$	$c(n,n')$	$h(n')$	$c(n,n')+h(n')$	consistency
$S \to A$	8	1	8	9	true
$S \to B$	8	5	4	9	true
$S \to C$	8	8	3	11	true
$A \to D$	8	3	∞	$3+\infty$	true
$A \to E$	8	7	∞	$7+\infty$	true
$B \to G$	4	4	0	4	true
$C \to G$	3	5	0	5	true
$A \to G$	8	9	0	9	true

- $h(n)$: heuristic of where you are standing right now
- $c(n,n')$: actual cost between 2 nodes
- $h(n')$: heuristic of where you want to go.

expand node	frontier list	
	S : 8	
S	A : 1+8 , B : 5+4 , C : 8+3	$S \to B \to G = 5+4 = 9$
A	B : 9 , G : 1+9 , C : 8+3 , D, E	
B	G : 5+4+0 , G : 10 , C : 11 , D, E	→ if 2 goal, then discard maximum
G → goal	C : 11 , D : ∞ , E : ∞	

day / date:

Genetic Algorithm

1. select encoding type
2. choose population size
3. randomly choose initial population
4. select parental chromosomes
5. crossover and mutation.
6. evaluation of offspring.
7. repeat until stopping criteria is met.

- selection
  - → roulette wheel: probability of selection
  - → tournament: few individuals are selected randomly.
- crossover
  - → single point:
  - → twopoint . segment between these points are swapped.
  - → uniform both parents are randomly swapped.
- mutation.
- calculate f(x), total, average.
- calculate probability count from f(x) / total
- calculate expected count from f(x) / average
  - ↳ round off for actual count.

day / date:

Most constrained variable / Minimum remaining value

- choosing variable with the fewest legal values → less option (values) in domain.
- if all variables have same number of legal values, then use degree heuristic
  → highest number of boundaries

Least constraining value

- prefers the value that rules out the fewest choice for the neighbouring variables.

Forward checking

- whenever a variable X is assigned, the forward checking process looks at each unassigned variable Y that is connected to X by a constraint and deletes from Y's domain any value that is inconsistent with the value chosen for X
- MRV is an obvious partner for forward checking, so next to choose is the one with move boundaries.
- If partial assignment is inconsistent with the constraints of problem, then backtrack.

Arc consistency

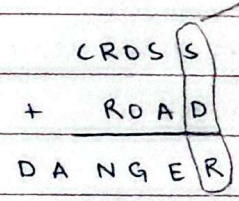
$X \rightarrow Y$: for every value of x of X, there is some allowed y.

Backtracking

- depth first search that chooses values for one variable at a time and backtracks when a variable has no legal values left to assign.

- unary constraints involve a single variable
- binary constraints involve pairs of variables.
- higher order constraints involve 3 or more variables.

- for cryptarithmetic, identify all unique characters.
- assign digits to those letters. If any duplicates backtrack and unassign.



Co based on assumptions.

day / date:

Intro to ML

- ML allows computer to learn from data and improve their performance over time without being explicitly programmed for each task.
- classic example of recognizing spam emails after being trained on a dataset of labeled emails.

AI vs ML vs DL

- AI : creating machine that can mimic human. — robot performing tasks
- ML : subset of AI , learns from data. — recommendation system
- DL : subset of ML , uses deep neural networks to learn from large amount of data — image recognition e.g. google photos

AI vs ML

AI : rules written to handle tasks

ML : system learns from experience (data)

study problem → train ML → evaluate → analyze errors

A program learns by improving its performance on a task(T), using experience (E), performance evaluated by a measure (P).

T : predicting whether customer will buy a product based on their website history

E : dataset containing historical customer behaviour.

P : accuray, how many correct predictions

Quantitative vs Qualitative

Qualitative → nominal, ordinal

Quantitative → discrete, continuous.

Structured vs Unstructured

Structured

↳ highly-organized , difficult to collect , limited insights

Unstructured

↳ no predefined format , easy to collect

day / date:

- supervised vs unsupervised learning vs reinforcement

  ↳ reinforcement: teaching an agent to make decision by rewarding it for correct actions

How data is used?

training data: set of labeled data used to train the model.

validation data: fine-tune model and adjust parameters

test data: evaluate models performance on new, unseen data

Feature extraction

: transforming raw data into a set of features that are useful for machine learning.

Model Development

dataset → preprocess	→ training → train → ML testing → test validation → tune → Model →	New case ↓ Final model ↓ predicted class.
extraction	Training validation.	

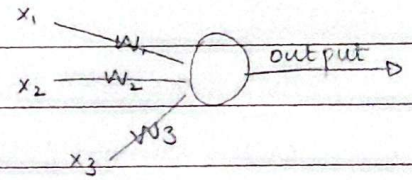
day / date:

ARTIFICIAL INTELLIGENCE

ARTIFICIAL NEURAL NETWORKS

- crude way of trying to simulate the human brain.
- human brain has ~10 billion neurons, connected to thousand others.
- neuron
  - → cell body
  - → dendrites (input signal)
  - → axons (output)
- each input into the neuron has its own weight associated with it.

$$z = x_1w_1 + x_2w_2 + x_3w_3 + \dots x_nw_n$$



- an activation function is then applied to the weighted sum plus a bias.

COMPONENTS

Inputs → set of values, features which we need to predict an output

Weights → quantify the influence of each input on the output. It minimizes prediction errors.

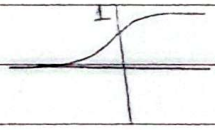
Bias → allows model to shift activation function.

Summation → bind weights and inputs together.

Activation → introduces non-linearity in the model.

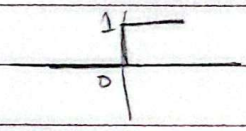
Node → combination of summation and activation.

$$\text{Sigmoid function} = \frac{1}{1 + e^{-x}}$$

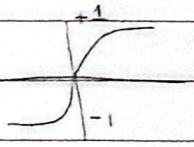


$$\text{Linear function} = x$$

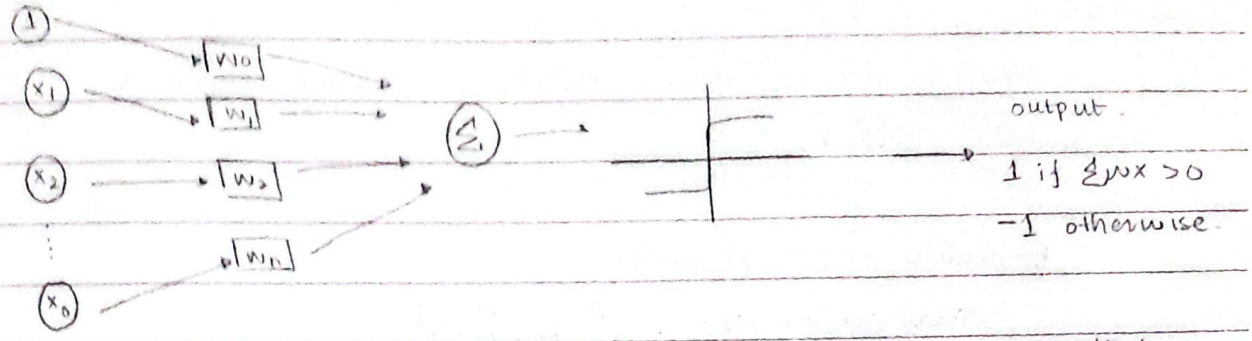
$$\text{step function} = \begin{cases} 1 \text{ if } x \geq 0 \\ 0 \text{ if } x < 0 \end{cases}$$



$$\text{Hyperbolic Tangent} = \frac{(e^x - e^{-x})}{e^x + e^{-x}}$$



PERCEPTRON



inputs | weights | summation & bias | activation function | output

$$\hat{y} = g(w_0 + \Sigma x_i w_i)$$

output — AF — bias

IMPLEMENTING 'OR' WITH PERCEPTRON

$w_0 = -0.3$, $w_1 = w_2 = 0.5$

$$Y = \begin{cases} 1 & \text{if } \Sigma wx > 0 \\ -1 & \text{otherwise} \end{cases}$$

x1	x2	Σwx	Y
0	0	-0.3	-1
0	1	0.2	1
1	0	0.2	1
1	1	0.7	1

$w_0 + w_1 x_1 + w_2 x_2$

- $-0.3 + (0.5)(0) + (0.5)(0) = -0.3$
- $-0.3 + (0.5)(0) + (0.5)(1) = 0.2$
- $-0.3 + (0.5)(1) + (0.5)(0) = 0.2$
- $-0.3 + (0.5)(1) + (0.5)(1) = 0.7$

IMPLEMENTING 'AND' WITH PERCEPTRON

$w_0 = -0.8$ $w_1 = w_2 = 0.5$

x1	x2	Σwx	Y
0	0	-0.8	-1
0	1	-0.3	-1
1	0	-0.3	-1
1	1	0.2	0

day / date:

PERCEPTRON LEARNING

→ Loss Function

- How much error network is making in predicting the class labels

$$\mathcal{L}(\hat{y}, y)$$

($\hat{y}$: predicted, y: actual)

- for multiple training examples

$$\text{Total Loss} = J(w) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\hat{y}^i, y^i)$$

also called cost function

$$J(w) = \frac{1}{m} \sum (\hat{y}^i - y^i)^2$$

- optimizing loss

$$w^* = \text{argmin}_w \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\hat{y}^i, y^i)$$

$$w^* = \text{argmin}_w J(w)$$

GRADIENT DESCENT

- find minimum value of J
- algorithm

1. Initialize weights randomly $\sim N(0, \sigma^2)$
2. Loop until convergence
3. Compute gradient $\frac{\partial J(w)}{\partial w}$
4. Update weights $w \leftarrow w - \eta \frac{\partial J(w)}{\partial w}$
5. Return weights

day / date:

. Types of Gradient Descent

1. Batch Gradient Descent

$$w_i := w_i + \eta \sum_{j=1}^{N} (t^j - o^j) x_i^j$$

2. Stochastic Gradient

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

3. Mini-Batch Gradient

$$w_i := w_i + \frac{\eta}{n} \sum_{j=nk}^{(k+1)n} (t^j - o^j) x_i^j$$

t = target output

o = predicted output

n = batch size

$k = (1, \frac{N}{n})$ number of batches

- batch gradient updates weight once per epoch
- stochastic gradient updates weight after each training example.
- mini-batch gradient updates weight using small batches of example.

day / date:

BREADTH FIRST SEARCH (BFS)

- use of First-in-First Out queue.
- search time takes $O(b^s)$ s: tiers
- s must be finite, so it is complete.
- it is optimal only if costs are all 1
- takes too much memory and space.

UNIFORM COST SEARCH (UCS)

- only cares about cost and not number of steps.
- it is optimal
- it is complete
- worst case $O(b^{c^*/\varepsilon})$

DEPTH FIRST SEARCH (DFS)

- if s is finite, takes time $O(b^s)$
- fringe space $O(bm)$
- s can be finite, so if cycles are prevented then only it is complete.
- it is not optimal as it searches leftmost

DEPTH LIMITED SEARCH

- it is incomplete.
- works well if we know what the depth of the solution is.
- it is not optimal
- time complexity $O(b^L)$
- space complexity $O(bL)$

ITERATIVE DEEPENING SEARCH

- memory $O(bd)$
- combines DFS and BFS
- complete when branching factor is finite and optimal when pathcost is a non-decreasing function.
- preferred when large search space and depth unknown.
- it is complete.
- time complexity $O(b^d)$

day / date:

BIDIRECTIONAL SEARCH

- it is complete, if BFS used in both ways
- it is optimal if BFS used for search and paths have uniform cost.
- time and space complexity $O(b^{d/2})$

SINGLE-LAYER PERCEPTRON - AND GATE

INPUTS		DESIRED OUTPUT	INITIAL WEIGHTS		ACTUAL OUTPUT	ERROR	FINAL WEIGHTS	
x_1	x_2	t	w_1	w_2	o	$t-o$	w_1	w_2
0	0	0	0.3	-0.1	0	0	0.3	-0.1
0	1	0	0.3	-0.1	0	0	0.3	-0.1
1	0	0	0.3	-0.1	1	-1	0.2	-0.1
1	1	1	0.2	-0.1	0	1	0.3	0.0
0	0	0	0.3	0.0	0	0	0.3	0.0
0	1	0	0.3	0.0	0	0	0.3	0.0
1	0	0	0.3	0.0	1	-1	0.2	0.0
1	1	1	0.2	0.0	1	0	0.2	0.0

1. Given:

   bias = -0.2

   Learningrate = $\eta = 0.1$

   Initial weights $w1 = 0.3$; $w2 = -0.1$

   stochastic gradient → $-0.2 + w_1x_1 + w_2x_2$

2. Summation (Σ)

$(0,0) \rightarrow -0.2 + w_1x_1 + w_2x_2 = 0$

$-0.2 + (0.3)(0) + (-0.1)(0) = 0$

$f(x) = -0.2$

3. Activation

$$\text{actual output} = \begin{cases} 1 & f(x) \geq 0 \\ 0 & f(x) < 0 \end{cases}$$

actual output = 0

4. Error

desired - actual = 0 - 0

error = 0

5. Update weights

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

- $w_1 = 0.3 + (0.1)(0)(0) = 0.3$
- $w_2 = -0.1 + (0.1)(0)(0) = -0.1$

(0,1) → initial weights $w_1 = 0.3$ $w2 = -0.1$

2. $-0.2 + (0.3)(0) + (-0.1)(1) = -0.3$
3. actual output = 0
4. error = 0 - 0 = 0
5. $w_1 = 0.3 + (0.1)(0)(0) = 0.3$

   $w_2 = -0.1 + (0.1)(0)(1) = -0.1$

(1,0) → initial weights $w_1 = 0.3$ $w2 = -0.1$

2. $-0.2 + (0.3)(1) + (-0.1)(0) = 0.1$
3. actual output = 1
4. error = 0 - 1 = -1
5. $w_1 = 0.3 + (0.1)(-1)(1) = 0.2$

   $w_2 = -0.1 + (0.1)(-1)(0) = -0.1$

(1,1) → initial weights $w_1 = 0.2$ $w2 = -0.1$

2. $-0.2 + (0.2)(1) + (-0.1)(1) = -0.1$
3. actual output = 0
4. error = 1 - 0 = 1
5. $w_1 = 0.2 + (0.1)(1)(1) = 0.3$

   $w_2 = -0.1 + (0.1)(1)(1) = 0.0$

(0,0) → initial weights $w1 = 0.3$ $w2 = 0.0$

2. $-0.2 + (0.3)(0) + (0)(0) = -0.2$
3. actual output = 0
4. error = 0 - 0 = 0

   $w_1 = 0.3 + (0.1)(0)(0) = 0.3$

   $w_2 = -0.0 + (0.1)(0)(0) = -0.0$

(0,1) → initial weights $w1 = 0.3$ $w2 = 0.0$

2. $-0.2 + (0.3)(0) + (0)(1) = -0.2$
3. actual output = 0
4. error = 0 - 0
5. $w_1 = 0.3 + (0.1)(0)(0) = 0.3$

   $w_2 = 0.0 + (0.1)(0)(1) = 0$

(1,0) → initial weights $w1 = 0.3$ $w2 = 0.0$

2. $-0.2 + (0.3)(1) + (0)(0) = 0.1$
3. actual output = 1
4. error = 0 - 1 = -1
5. $w_1 = 0.3 + (0.1)(-1)(1) = 0.2$

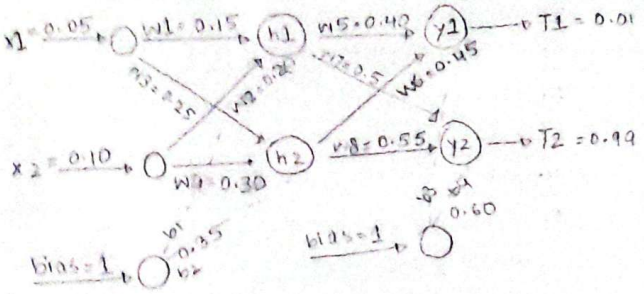
   $w_2 = 0.0 + (0.1)(-1)(0) = 0$

(1,1) → initial weights $w1 = 0.2$ $w2 = 0.0$

2. $-0.2 + (0.2)(1) + (0.0)(0) = 0$
3. actual output = 1
4. error = 1 - 1 = 0
5. $w_1 = 0.2 + (0.1)(0)(1) = 0.2$

   $w_2 = 0.0 + (0.1)(0)(1) = 0.0$

continue until desired output = actual output



1. calculate the outputs from h1 and h2 using logistic activation function. (SF)

sigmoid function = $\frac{1}{1+e^{-x}}$

- output H1 = $x1*w1 + x2*w2 + b*1$
  = (0.05)(0.15) + (0.10)(0.20) + (0.35)(1)
  = 0.3775

$$SF = \frac{1}{1+e^{-0.3775}} = 0.59327$$

- output H2 = $x2*w2 + x_1w_3 + b*1$
  = (0.10)(0.30) + (0.05)(0.25) + (0.35)(1)
  = 0.3925

$$SF = \frac{1}{1+e^{-0.3925}} = 0.59689$$

2. calculate the outputs from y1 and y2 using sigmoid function.

- output y1 = $out_{h1}*w5 + out_{h2}*w6 + b*1$
  = (0.59372)(0.40) + (0.59689)(0.45) + 0.6
  = 1.10591

$$SF = \frac{1}{1+e^{-1.10591}} = 0.75137$$

- output y2 = $out_{h2}*w8 + out_{h1}*w7 + b*1$
  = (0.59689)(0.55) + (0.59327)(0.5) + (0.60)
  = 1.22492

$$SF = \frac{1}{1+e^{-1.22492}} = 0.77293$$

3. calculate error as y1 and y2 are not same as target

$$E = \frac{1}{2}(T_1 - out_{y1})^2 + \frac{1}{2}(T_2 - out_{y2})^2$$

$$E = \frac{1}{2}(0.01 - 0.75137)^2 + \frac{1}{2}(0.99 - 0.77293)^2$$

$$E = 0.29837$$

4. we begin backward pass. Adjust the weights leading to o1 and o2 i.e. y1 and y2

- calculate error with respect to y1

$\delta_{y1} = (T_1 - out_{y1}) \times (out_{y1}) \times (1 - out_{y1})$
= (0.01 - 0.75137) × (0.75137) × (1 - 0.75137)
= -0.13850

- update associated weights w5, w6, bias

$w5 = w5 + \eta \times \delta_{y1} \times out_{h1}$
= 0.40 + 0.5 × (-0.13850) × 0.59327
= 0.35892

$w6 = w6 + \eta \times \delta_{y1} \times out_{h2}$
= 0.45 + 0.5 × (-0.13850) × (0.59689)
= 0.40867

$b3 = b3 + \eta \times \delta_{y1} \times 1$
= 0.60 + 0.5 × (-0.13850) × 1
= 0.53075

- calculate error with respect to y2

$\delta_{y_2} = (T_2 - out_{y_2}) \times (out_{y_2}) \times (1 - out_{y_2})$
= (0.99 - 0.77293) × (0.77293) × (1 - 0.77293)
= 0.03810

- update associated weights w7, w8, bias

$w7 = w7 + \eta \times \delta_{y_2} \times out_{h1}$
= 0.5 + 0.5 × 0.03810 × 0.59327
= 0.51130

$w8 = w8 + \eta \times \delta_{y_2} \times out_{h2}$
= 0.55 + 0.5 × 0.03810 × 0.59689
= 0.56137

$b4 = b4 + \eta \times \delta_{y_2} \times 1$
= 0.60 + 0.50 × 0.03810 × 1
= 0.61905

5. Adjust the weights leading to H1 and H2

- calculate error with respect to H1

$$\delta_{h1} = (\delta_{y1} \times w5 + \delta_{y2} \times w7) \times out_{h1} \times (1 - out_{h1})$$
$$= (-0.13850)(0.40) + (0.03810)(0.5) \times (0.59327) \times (1 - 0.59327)$$
$$= -0.00877$$

- update associated weights w1, w2, b4

$$w1 = w1 + \eta \times \delta_{h1} \times x1$$
$$= 0.15 + 0.5 \times (-0.00877) \times 0.05$$
$$= 0.14978$$

$$w2 = w2 + \eta \times \delta_{h1} \times x2$$
$$= 0.20 + 0.50 \times (-0.00877) \times (0.10)$$
$$= 0.19956$$

$$b1 = b1 + \eta \times \delta_{h1} \times 1$$
$$= 0.35 + 0.5 \times (-0.00877) \times 1$$
$$= 0.34562$$

- calculate error with respect to H2

$$\delta_{h2} = (\delta_{y1} \times w6 + \delta_{y2} \times w8) \times out_{h2} \times (1 - out_{h2})$$
$$= (-0.13850)(0.45) + (0.03810)(0.55) \times (0.59689) \times (1 - 0.59689)$$
$$= -0.00995$$

- update associated weights w3, w4, b2

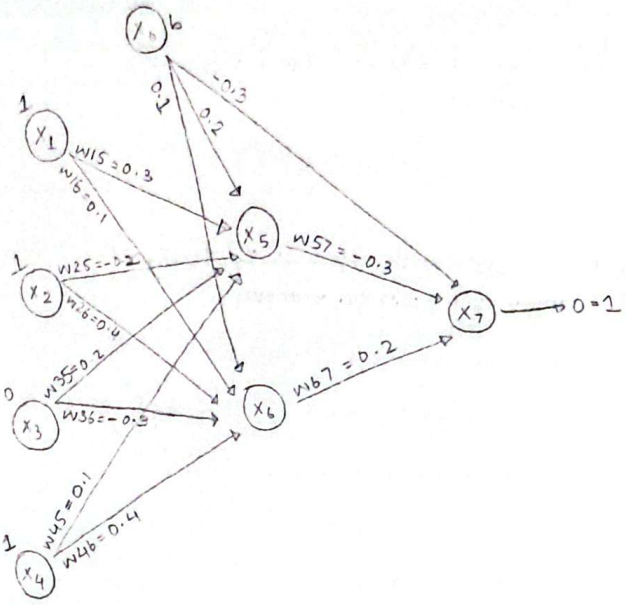
$$w3 = w3 + \eta \times \delta_{h2} \times x1$$
$$= 0.25 + 0.5 \times (-0.00995) \times (0.05)$$
$$= 0.24975$$

$$w4 = w4 + \eta \times \delta_{h2} \times x2$$
$$= 0.30 + 0.5 \times (-0.00995) \times (0.10)$$
$$= 0.29950$$

$$b2 = b2 + \eta \times \delta_{h2} \times 1$$
$$= 0.35 + 0.50 \times (-0.00995) \times 1$$
$$= 0.34503$$

6. Repeat until root mean square of output errors is minimized.

EXAMPLE #2



Learning rate = 0.8

1. calculate output at x_5 and x_6

- output x_5 = (x1)(w15) + (x2)(w25) + (x3)(w35) + (x4)(w45) + (1)(b5)
$$= (1)(0.3) + (1)(-0.2) + (0)(0.2) + (1)(0.1)$$
$$= 0.15 + 0.2 = 0.4$$

- output x_6 = (x1)(w16) + (x2)(w26) + (x3)(w36) + (x4)(w46) + (1)(b6)
$$= (1)(0.1) + (1)(0.4) + (0)(-0.30) + (1)(0.4)$$
$$= 1.4 + 0.1 = 1$$

$$SF_5 = \frac{1}{1 + e^{0.4}} = 0.5987$$

$$SF_6 = \frac{1}{1 + e^{-1}} = 0.7311$$

2. calculate output at x_7

- output x7 = (x5)(w57) + (x6)(w67) + (1)(b7)
$$= (0.5987)(-0.3) + (0.7311)(0.2) + (-0.3)$$
$$= -0.3334 \longrightarrow O_7 = 0.5825$$

3. calculate error

$$Error = O_{desired} - O_{estimated}$$
$$= 1 - \frac{1}{1 + e^{-0.3334}}$$
$$= 0.417$$

4. for output layer i.e x_7

$$error = O_7 (1 - O_7)(error)$$

for hidden layer i.e. x_5 / x_6

$$error_j = O_j (1 - O_j)(w_{jk} \times error_k)$$

$\delta_7 = (0.5825)(1-0.5825)(0.417)$
$= 0.1014$

$\delta_5 = (0.5987)(1-0.5987)(-0.3)(0.1014)$
$= -0.0073$

$\delta_6 = (0.7311)(1-0.7311)(0.2)(0.1014)$
$= 0.0040$

5. update weights with respect to X_5, X_6, X_7

$w_{ij} = w_{ij} + \eta \times error_j \times output_i$

- $w_{57} = -0.3 \times 0.8 \times 0.1014 \times 0.5987 = -0.2514$
- $w_{67} = 0.2 + 0.8 \times 0.1014 \times 0.7311 = 0.259$
- $w_{15} = 0.3 + 0.8 \times (-0.0073)(1) = 0.29416$
- $w_{16} = 0.1 + 0.8 \times (0.0040)(1) = 0.1032$
- $w_{25} = (-0.2) + 0.8 \times (-0.0073)(1) = -0.2058$
- $w_{26} = 0.4 + 0.8 \times (0.0040)(1) = 0.4032$
- $w_{35} = 0.2 + 0.8 \times (-0.0073)(0) = 0.19416$ ~~0.19416~~
- $w_{36} = (-0.3) + 0.8 \times (0.0040)(0) = -0.3$
- $w_{45} = (0.1) + 0.8 \times (-0.0073)(1) = 0.09416$
- $w_{46} = 0.4 + 0.8 \times (0.0040)(1) = 0.4032$

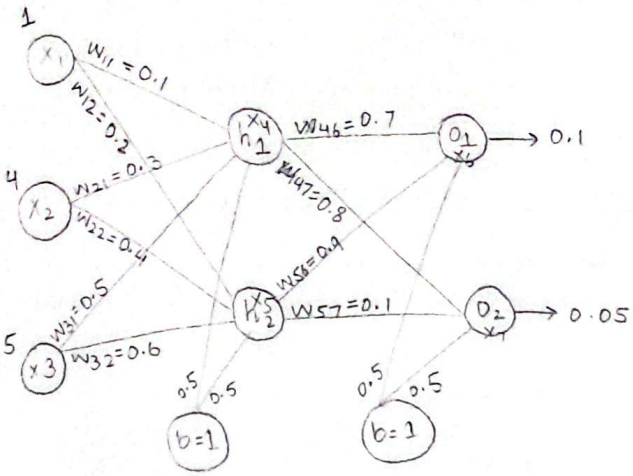
6. update biasness

$b_5 = b_5 + \eta \times error_5$
$= 0.2 + 1.8 \times (-0.0073) = 0.19416$

$b_6 = 0.1 + 0.8 \times error_6$
$= 0.1032$

$b_7 = -0.3 + 0.8 \times 0.1014$
$= -0.2188$

EXAMPLE # 3



- calculate output at hidden layers h1, h2

output $h_1 = (w_{11})(x_1) + (w_{21})(x_2) + (w_{31})(x_3)$
$+ (1)(0.5)$
$= (0.1)(1) + (0.3)(4) + (0.5)(5) + (1)(0.5)$
$= 4.3$

output $h_2 = (w_{12})(x_1) + (w_{22})(x_2) + (w_{32})(x_3)$
$+ (1)(0.5)$
$= (0.2)(1) + (0.4)(4) + (0.6)(5)$
$+ (1)(0.5)$
$= 5.3$

- calculate output at O_1 and O_2 using SF

output $O_1 = (w_{46})(h_1) + (w_{56})(h_2) + (1)(0.5)$
$= (0.7)(4.3) + (0.9)(5.3) + (0.5)$
$= 8.28$

$$SF = \frac{1}{1+e^{-8.28}} = 0.9997$$

output $O_2 = (w_{57})(h_2) + (w_{47})(h_1) + (1)(0.5)$
$= (0.1)(5.3) + (0.8)(4.3) + 0.5$
$= 4.47$

$$SF = \frac{1}{1+e^{-4.47}} = 0.9887$$

- calculate error according to target

$$E = \frac{1}{2}(T_1 - O_1)^2 + \frac{1}{2}(T_2 - O_2)^2$$

$$E = \frac{1}{2}(0.1 - 0.9997)^2 + \frac{1}{2}(0.05 - 0.9887)^2$$

$$E = 0.845$$

MULTI-...

- calculate error at x_4, x_5, x_6, x_7 i.e h_1, h_2, o_1, o_2

$$\delta_{b_1} = (T_1 - out_{h_1})(out_{h_1})(1 - out_{h_1})$$
$$= (0.1 - 0.999)(0.999)(1 - 0.999)$$
$$= -0.0002698$$

$$\delta_{b_2} = (T_2 - out_{o_2})(out_{o_2})(1 - out_{o_2})$$
$$= (0.05 - 0.9887)(0.9887)(1 - 0.9887)$$
$$= -0.01049$$

$$\delta_{h_1} = (4.3)(1-4.3)(-0.0002698)(0.7) +$$
$$= (-0.01049)(0.8) = 0.1188$$

$$\delta_{h_2} = (5.3)(1-5.3)(0.9)(-0.0002698) + (0.1)(-0.01049)$$
$$= 0.02944$$

- update weights b_1, w_4, w_{10}

$$w_4 = w_{22}$$
$$= 0.4 + 0.01 \times$$
$$= 0.400$$

$$w_{10} = w_{57}$$
$$= 0.1 + 0.01 \times (-0.01049)(5.3)$$
$$= 0.09944$$

$$b_1 = 0.5 + 0.01 *$$
$$= 0.504$$

* when going downward in branch, throw exact alpha beta values

* when going upwards in branch, throw node value and update according to min-max value.