AI intelligence

Artificial ⟶ Something created by humans.

Intelligence ⟶ ① The ability to solve novel problems.

② The ability to think like humans.

The ability to act- like humans.

act → subconscious actions as well.

③ Ability to think rationally. (unbiased)

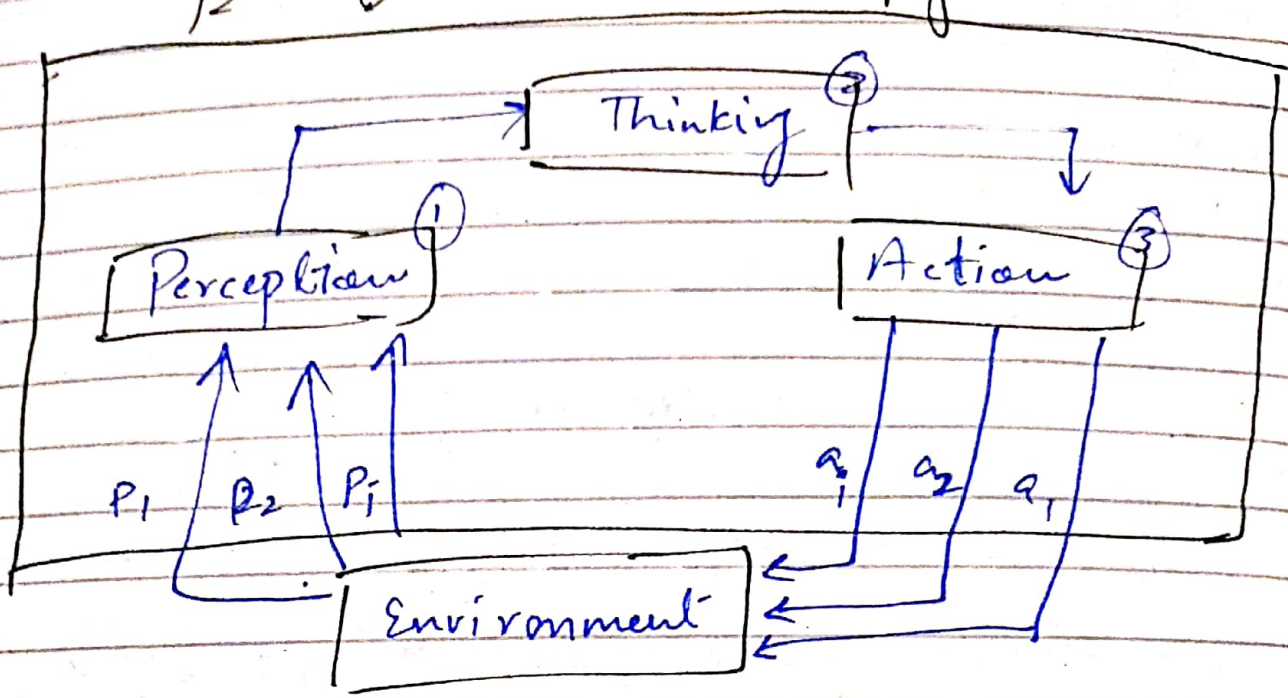
Ability to act rationally.

Rational:- A rational decision is the best possible outcome given the available facts which is unbiased.

facts → performance measure.

AI:- Is to build & understand intelligent entities or agents.

Model of an agent:-



1- Perception:- Receiving input from environment through sensors & maintaining an ~~initial~~ internal state. (sometimes)

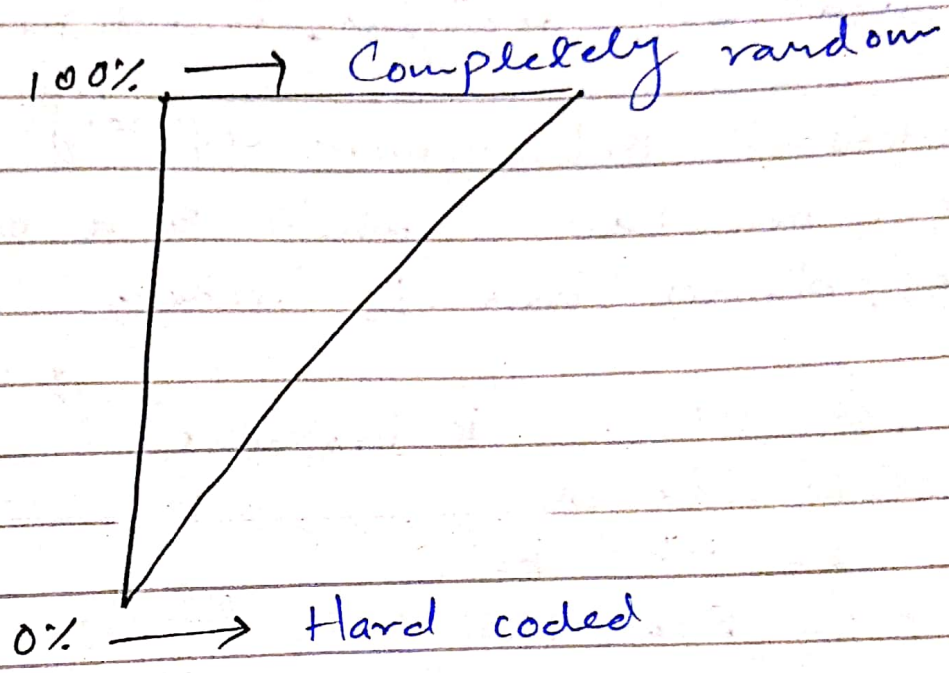
↳ arrangement of inputs (for now).

2- Thinking:- It is the mapping of perception to action.

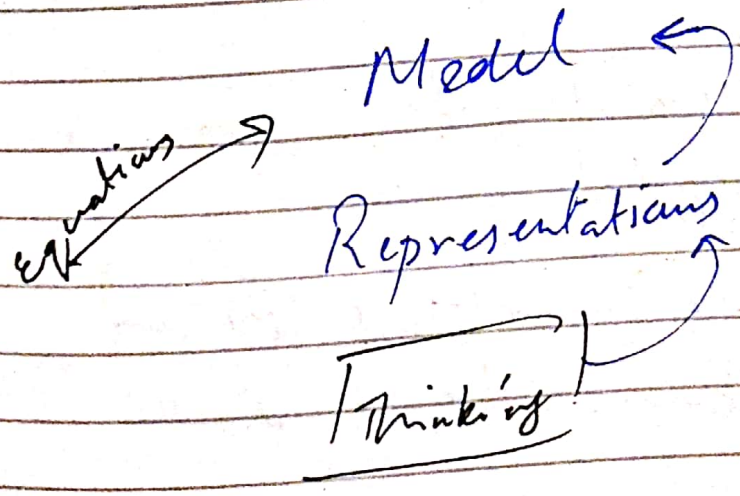
3- Action:- The change by the agent on a environment through actuators.
the

Example: light switch is an agent but we know that it's not intelligent. Hence to prove it wrong we say that there must be some ~~percent~~ degree of learning and autonomy.

Autonomy :- It is the extent/degree to which an agent's action are decided by exploring the environment rather than using the designer's provided knowledge.



Learning :- Denotes to changes in the agent's model after experiencing the environment.



. - - - - - - - .

Artificial Intelligence...

Def 1:- (Simple)

AI is an effort for the reproduction of human-like reasoning & intelligent behaviour through computational methods, machines, robots etc.

Intelligent Behaviour:- Responding to questions

Eg: The box has a white & a black ball
The darker ball is broken.

Q:- What is broken? → asked from machine

1- The box
2- White ball
3- Black ball

Reasoning:-

e.g. f1:- All men are mortal
f2:- Ali is a man
Conc:- Ali is mortal.

→ AI has 4 types of softwares/robots
e.g. of playchess (More focused on actions)

Act like humans	Think like humans
Act rationally	Think rationally

→ How to determine the extent/degree of learning/autonomy?

pandas Series Operations vs. numpy arrays.

1) Content → Act like humans. Model the human actions.

Machine Turing Test → pass if machine fools the interv-

Eliza → Program simulating a psychotherapist.

e.g.

Eliza Q: what is your name?
Ans: My name is Ali.

Human Q: What is my name?
Ans: Your name is Ali.

Syntactic pattern.

"My"

→ Was able to fool over 30%.

CAPTCHA → also a turing test

Eugene Goo.. → Ukrainian boy chat bot.

Turing limitations include → judges

Verbal dodge → Answer the question by dodging the actual question & responding in the same context.

→ Machines lack understanding.

A better Turing Test:-
→ MCQs. to avoid verbal dialogues.

Symantic web:- Relate different concepts (Most difficult).

2) Think like humans:- Modelling of thoughts that are human-like
eg:- Be able to process symbols/ symbolic knowledge (def, theorems etc).

Cognitive Science Approaches
→ Reverse Engineer.

Humans have neurons that are connected to 10,000 other neurons on avg. Hence, an incredible amount/degree of parallelism.
Problems: emotions/biases etc.

3) Think Rationally:- Idealized way of thinking. (logic/ list/ reasoning) approach.

Problems:- computational complexity, etc.

Binary facts-

f_1	f_2	C
0	0	c_1
0	1	c_2
1	0	c_3
1	1	c_4

2^4 conclusions.

Encoding this with

2. Describing real world (Not every fact will be binary). / not 100%. true etc.

3. Rational reasoning without logic e.g. (reflex) *

4. Dealing with uncertainty

---

4) Act Rationally (easier way) :-

→ Act optimally to achieve goals / Maximize your utility.

Summary :- playing cricket.

1. actions like human.
2. Model/Info processing is human brain inspired
3. Logical thinking
4. Performance Measure.

What AI system cannot do?

→ understanding natural language.

→ surfing web.

→ Comment on arbitrary scene.

→ Equal learn a language themself.

# of atoms in Universe = 10^{80}.

Chess $M_{max} = 10^{120}$

Go game = 10^{250}

→ Beating Go game player.

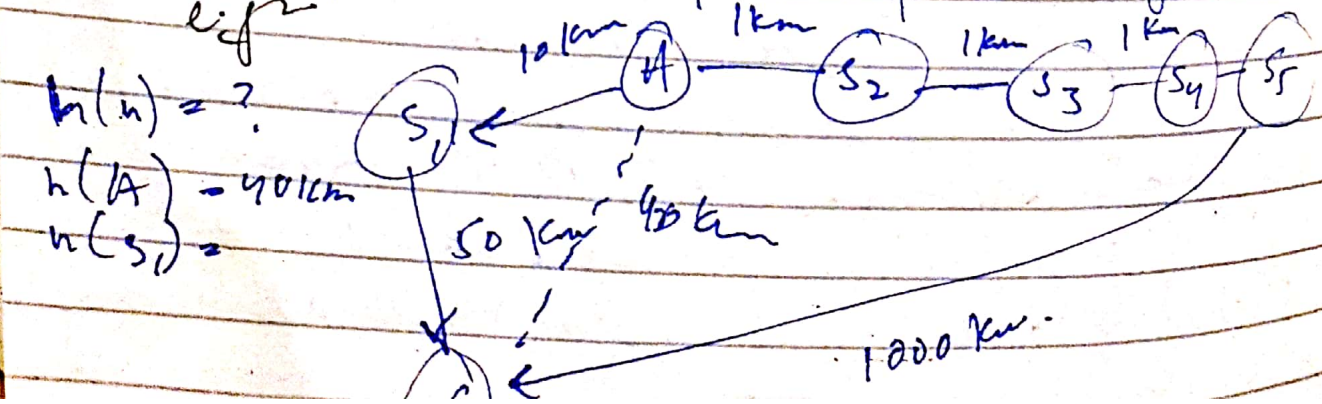
→ Construct plans in dynamic real-time domains.

→ Refocus attention in complex environments.

→ Perform life-long learning.

// End of chapter.

Heuristic → Give additional info. to find time to speed up the algorithm



Chapter # 02

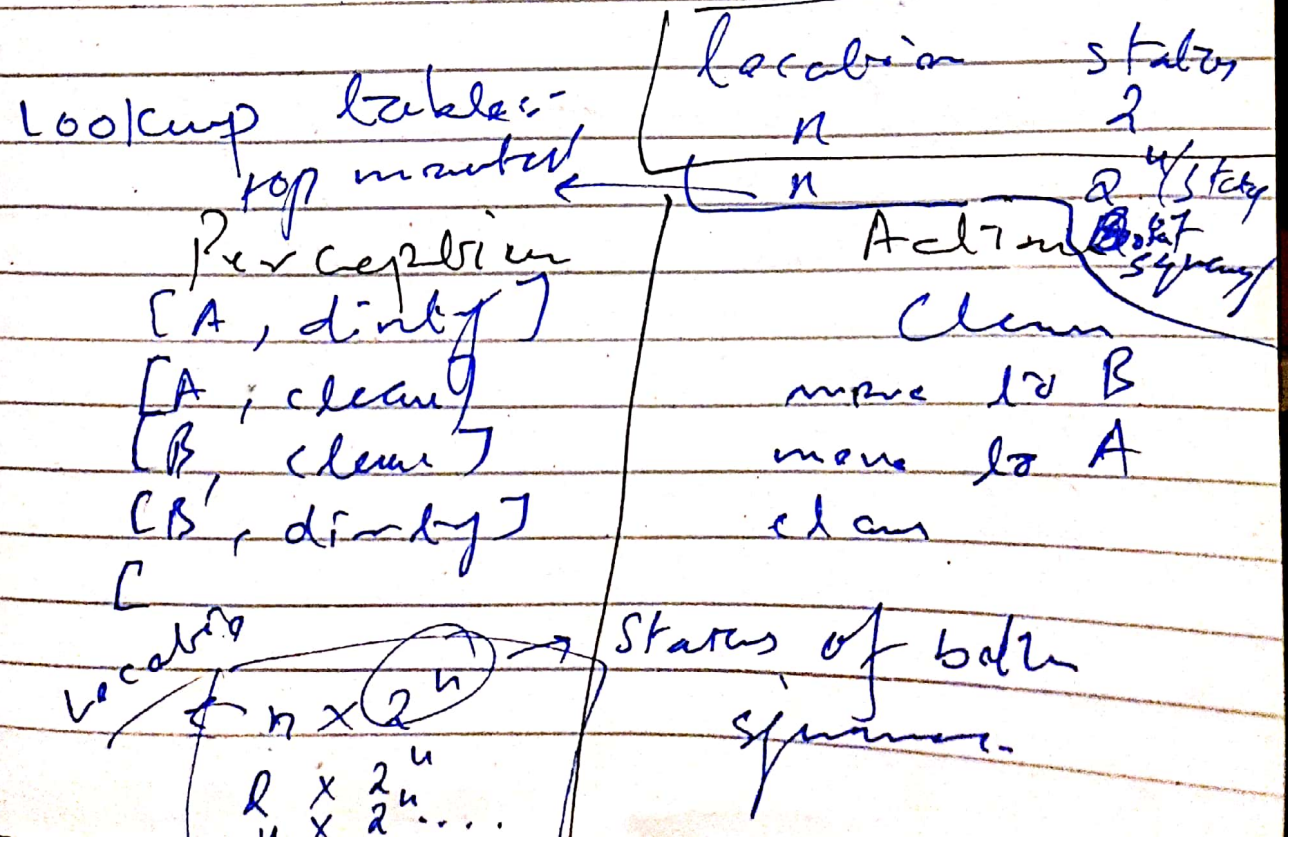
Agent:- (Another Def):-

1. Situated in some environment.
2. Autonomous (sense environment & related to it).
3. Flexible
   a. Reactive (Responsive)
   b. Proactive
   c. Social

Agent Function & Program

→ maps from percept-histories into actions.

e.g.:- Vaccum Cleaner World



Agent & environments:-

Rational Agents:-

Rationality:-

1. Performance Measure
2. Prior knowledge of environment
3. Actions that can be performed
4. Agent's percept sequence to date.

Percept Sequence:- Series of perceptions.

* To design a rational agent we need to specify its PEAS. (Performance measure, Environment, Actions, Sensors).

Rational agent → Selection of action to maximize performance measure for each percept sequence.

Rationality vs. Omniscience:-

Omnipotent:- All powerful.

Omniscience:- → Perfect agent (only a concept)
An agent is omniscient that sees everything.
eg:- Knowing all 10^{120} states for chess.

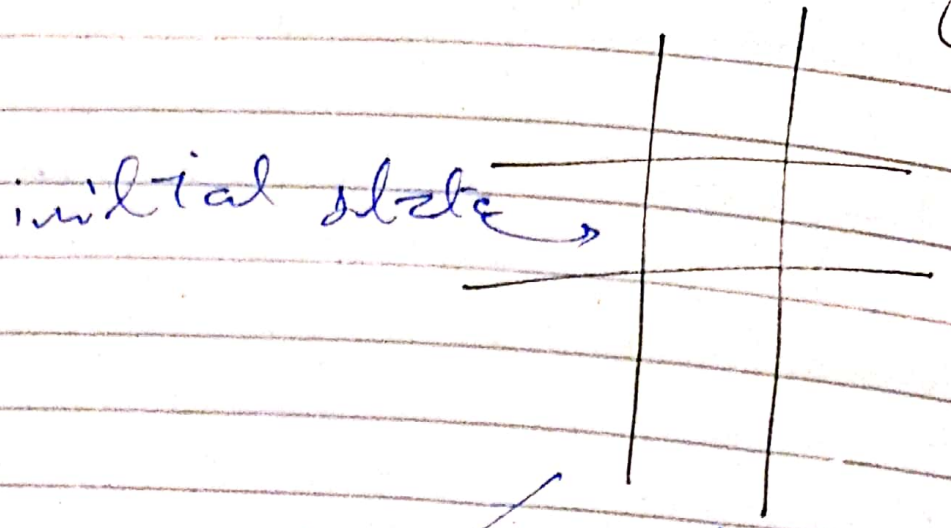
Vs Rationality:-
eg:- In cards game, don't know others' cards.
→ Best action in given situation (Not perfect).

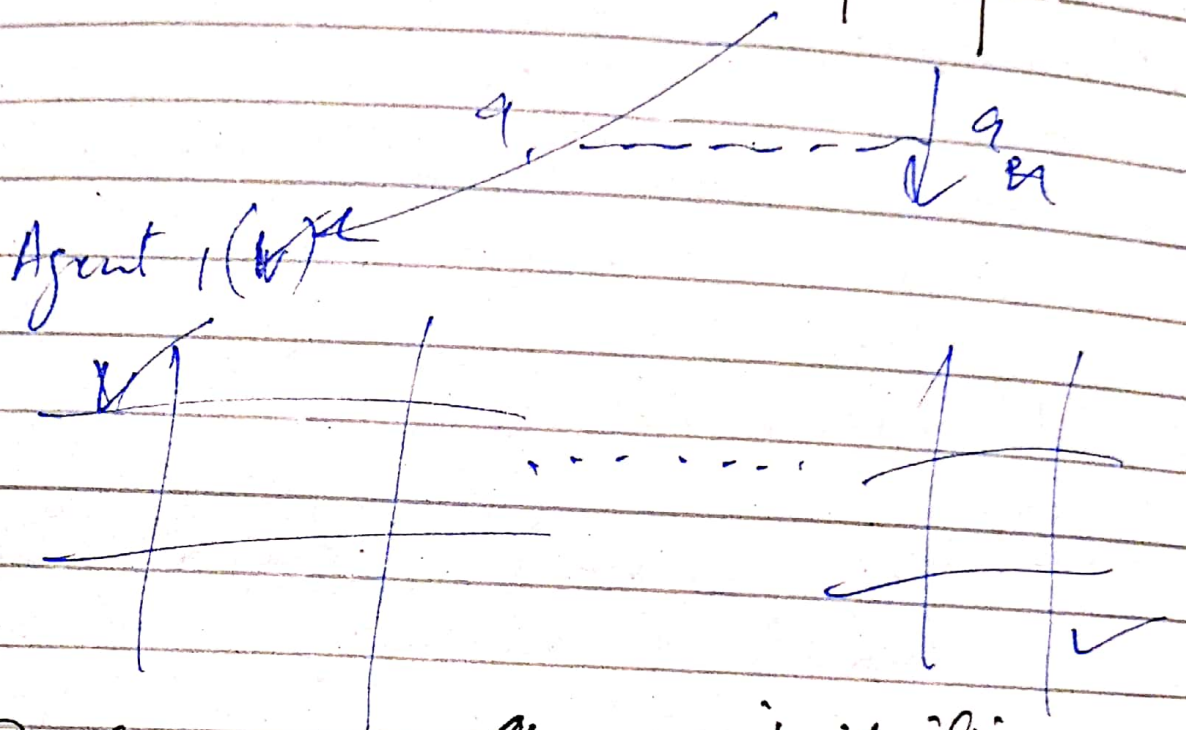
Specifying Task Environments.

→ Task environment problems where rational agents as solutions
e.g:- Automated Taxi
↳ Must contain PEAS.

Refined Tic Tac Toe Agent

eg





Performance Measure → ① Maximize winning, only legal moves etc.

② possibilities

Environment:-

→ The environment will be the board / ~~state~~ game playarea.

Actuators: Hand with a pen.

Sensors:- Camera / Angle sensors /

State:- An abstract representation of the environment by retaining all relevant information.

eg:- In the case of tic tac toe the abstract representation of the initial state can be a 3X3 2D Array which is empty.

Types of Environments:-

(1) Fully observable vs partially observable -

Fully observable (All state variables) :- If an agent's sensor provide all relevant details necessary for taking a rational decision. Then the environment is fully observable.

(2) Deterministic vs. stochastic
(3) Episodic vs. sequential
(4) Static vs. dynamic
(5) Discrete vs. continuous
(6) Single agent vs. multi-agent.
(7) Known vs. unknown.

Stochastic Vs Deterministic

→ you're at a bus stop & the arrival of passengers at the stop is stochastic. (Random for you) It will can follow some distribution but it's still random.

→ If the next state of the environment if it can be fully determined by choosing an action - it's called deterministic.

Eg:- When you perform an action in tic tac toe you'll know the next state. (deterministic)

Episodic Vs. Sequential:-

→ If choice of action depends only the current perception only.

Eg:- spam filter of email.

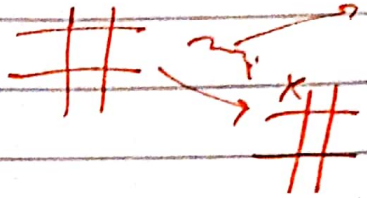
Vs. pac man is sequential.

→ Tic tac toe is episodic as

Static vs. Dynamic :-

→ The environment (relevant details) does not change while the agent is thinking/ to take an action.

e.g.



→ Opposite of static is dynamic

e.g. autonomous cars.

* → Semi-dynamic → static environment but there is a time factor / e.g. blitz chess.
↳ that changes the agent's performance.

Discrete vs. Continuous :-

→ finite values

Infinite values.

When an agent perceives. The number of possibilities for those state variables is finite vs. infinite is continuous.

e.g. environment values for autonomous cars is infinite so it is environment is continuous.

→ Does the environment provide a fixed number of distinct percepts, actions, and its environment states?

State of board in chess game

Turn the steering wheel to 0.1°, 0.01°, 0.0001° — continuous.

Vaccuum cleaner in a confined environment is discrete vs. if it's in an everchanging environment is continuous.

Single agent vs. Multiagent:-

→ An environment in which the agent does not have to interact with other agents.

e.g. parcel-picking robot.

Known vs. Unknown:-

→ This is more related to the environment rules.

⇒ The rules of the environment, (the transition model) are known then it's a known environment.

e.g. chess, rules of movement.

Deterministic

unknown e.g. Rovers, humans on moon.

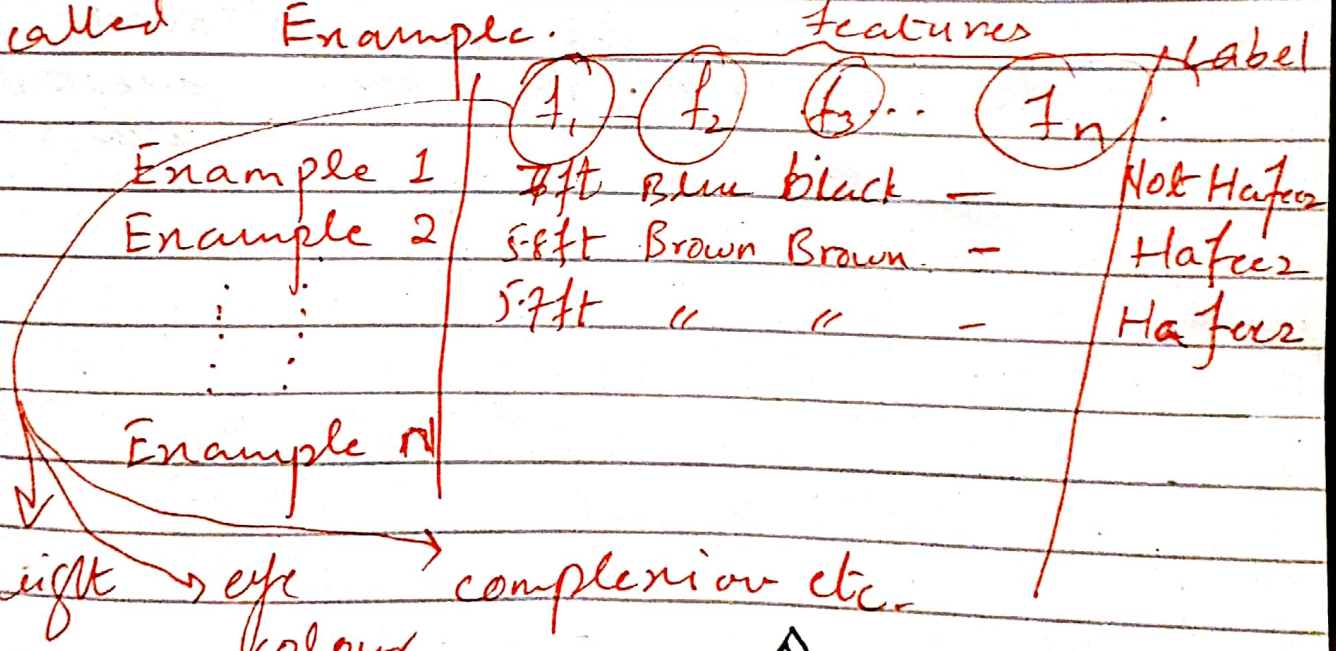
Artificial Intelligence... (v-Imp)

→ Machine Learning:-

Categories of algorithms:

1. Supervised Learning
2. Unsupervised "
3. Reinforcement "

→ Stats in this content is also called Example.



Supervised Learning:-

→ If we have example + features + labels.

Unsupervised Learning:- e.g. clustering

→ Examples + features.

Reinforcement Learning:-

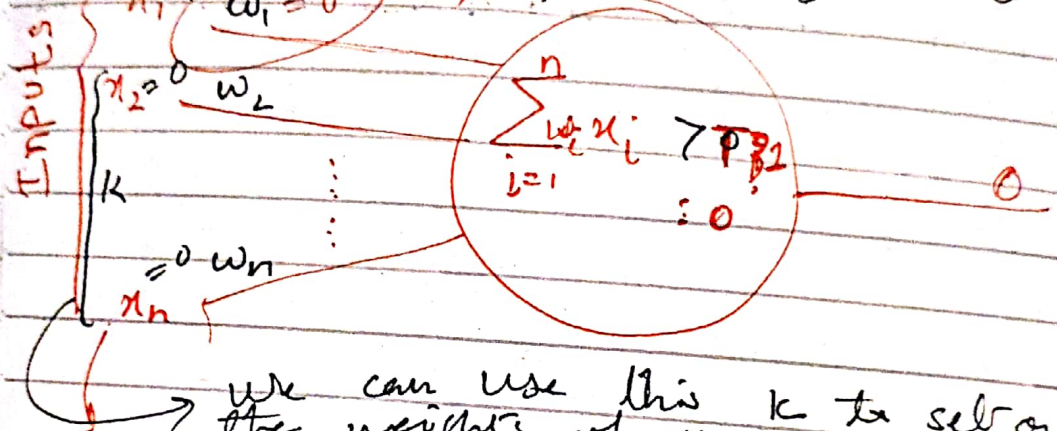
→ Examples + features + feedback (reward)

fully automated
↳ Learn features on their own

1) Neural Networks:- Type of supervised learning (examples, features, labels).

i) Model of a perceptron:-

→ Perceptron is a model of a biological neuron.

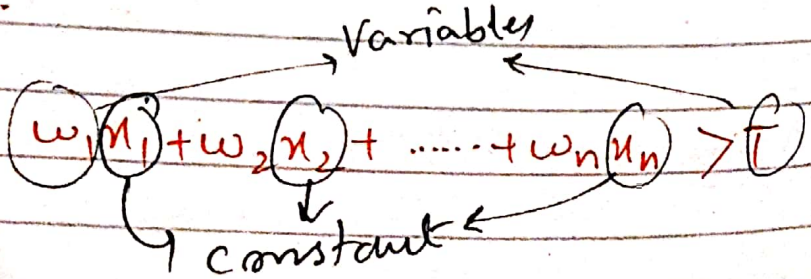


→ We can use this k to set only the weights of the relevant perceptron state.

Inputs/States/Examples/Signals
(Depends upon the context).

↳ These can be:- ① Binary
② Real values
③ Vectors
but numerical values. e.g.! A, B, C etc.

→ If there are sentences/words then those are encoded into numerical values.

→ 

0.25
0.5
0.75
0.

→ Let $T = w_0$

$$-w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n > 0$$

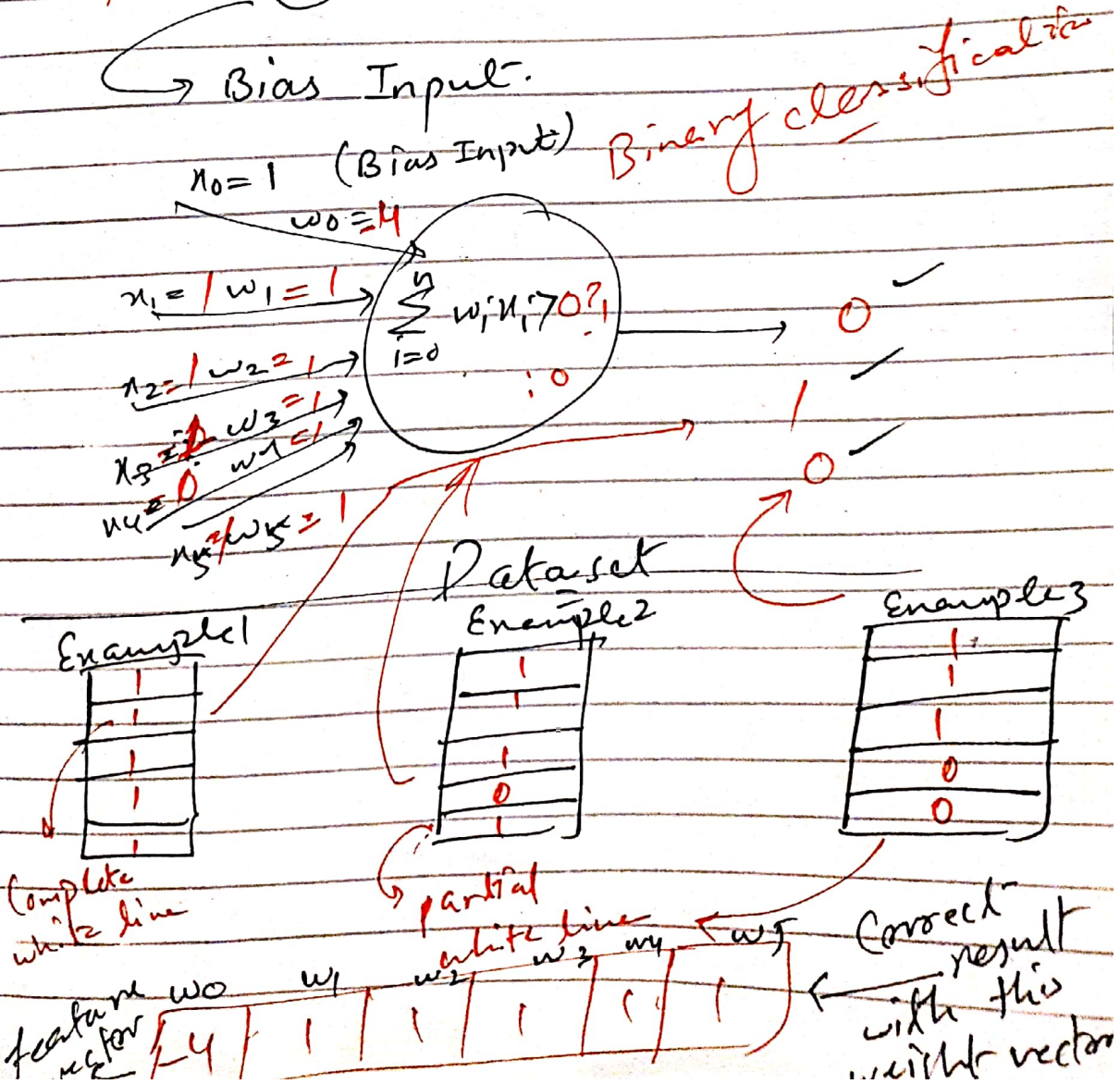
e.g:- if $w_0 = 7$ & since w_0 is a variable, we can remove the minus sign & replace w_0 with -7.

$x_0 = 1$

$$\Rightarrow w_0 \cdot 1 + w_1x_1 + w_2x_2 + \cdots w_nx_n > 0$$

$$\Rightarrow w_0x_0 + w_1x_1 + w_2x_2 + \cdots w_nx_n > 0$$

→ Bias Input.

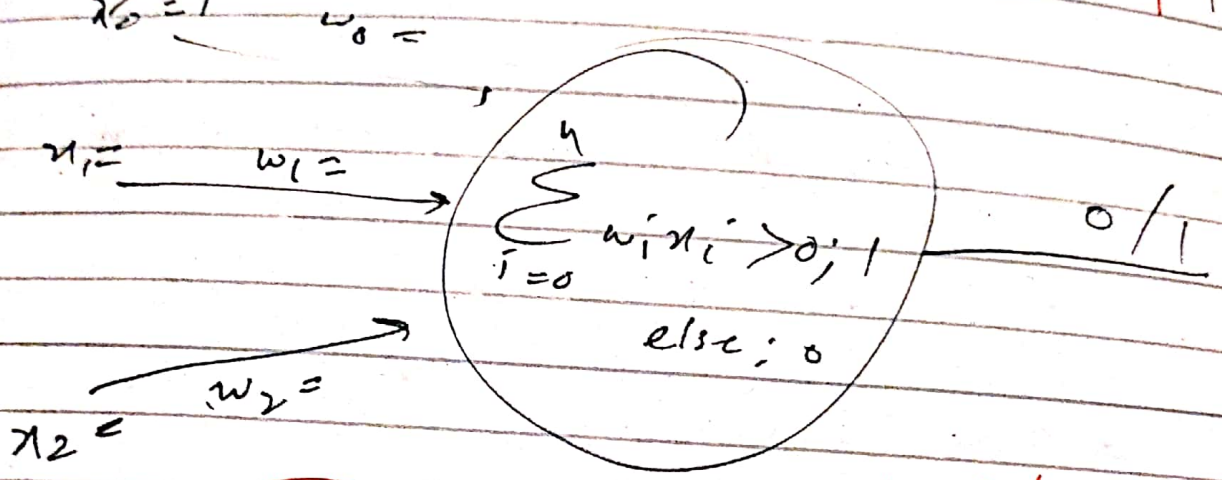


Perceptron Learning Algorithm

AND Function

dataset

	x_1	x_2	t
E1	0	0	0
E2	0	1	0
E3	1	0	0
E4	1	1	1



w_2 [-1, 1, 1]

-1 0 1 = 0 ✓
-1 0 0 = -1 ✓
-1 1 0 = 0 ✓
-1 1 1 = 1 ✓

→ In order to consider weight vector automatically we can use the algorithm.

Steps..

(1) Initialize all weights to random values. (Real & Numerical value)

(2) Until ~~all~~ outputs of all training examples are correct.

↳ while (Error != 0)

$$\text{Error} = \sum_{d \in D} |t_d - o_d| \quad \leftarrow \text{absolute error}$$

$$(\;//\;)^2 \rightarrow \text{squared error}$$

$w_0 = 0 \;,\; w_1 = 0.1 \;,\; w_2 = 0.2$

while (Error != 0) {

$\Delta w_0 = 0 \quad \Delta w_1 = 0 \quad \Delta w_2 = 0$

For Example 1:-

$\rightarrow$ weight change $\rightarrow$ new x_1 x_2

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

x_1	x_2	θ	t
0	0	0 —	0
0	1	1 ✗	0
1	0	1 ✗	0
1	1	0 ✓	1

η → learning rate (controls the pace of learning)

Usually small

$\Delta w_0 = 0 + 0.1(0 - 0)1 = 0$

↳ weight change of $w_0 = 0$

$\Delta w_1 = 0 + 0.1(0 - 0)0 = 0$

$\Delta w_2 = 0 + 0.1(0 - 0)0 = 0$

For Example 2:-

$\Delta w_0 = 0 + 0.1(0 - 1)1 = -0.1$

$\Delta w_1 = 0 + 0.1(0 - 1)0 = 0$

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

For Example 3:-

$\Delta w_0 = -0.1 + 0.1(0-1)1 = -0.2$

$\Delta w_1 = 0 + 0.1(0-1)1 = -0.1$

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

For Example 4:- No change if input is 0.

$\Delta w_0 = -0.2 + 0.1(0-1)1 = -0.2$

$\Delta w_1 = -0.1$

$\Delta w_2 = -0.1$

→ No change as $t - 0 = 0$

$t = 0$ out.

$w_i = w_i + \Delta w_i$

$w_0 = 0 + (-0.2)$

$w_0 = -0.2$

$w_1 = 0.1 + (-0.1)$

$w_1 = 0$

$w_2 = 0.2 + (-0.1)$

$w_2 = 0.1$

Weight Vector

w_0	w_1	w_2
-0.2	0	0.1

$-0.2 + 0.1 = -0.1$

$0 - 0.2 = -0.2$

$-0.2 + 0.1 = -0.1$

x_1	x_2	t	O	
0	0	0	0	✓
0	1	0	0	✓
1	0	0	0	✓
1	1	0	1	X

Error = 25%

→ One iteration = epoch.

→ Every iteration will reduce the error. Still not converged. But, it's going towards convergence.

→ All examples 1-3 will be 0. (T=0)

Example 4:-

$\Delta w_0 = 0 + 0.1(1-0)1 = 0.1$

$\Delta w_1 = 0 + 0.1(1-0)1 = 0.1$

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

$w_0 = -0.1 \quad w_1 = 0.1 \quad w_2 = 0.2$

x_1	x_2	t	O	
0	0	0	0	✓
0	1	0	1	X
1	0	0	0	✓
1	1	1	1	✓

$-0.1 = -0.1$

$-0.1 + 0.2 = 0.1$

$-0.1 + 0.1 = 0$

$-0.1 + 0 + 0.2 = 0.2$

Error = 25%

$w_0 = -0.1 \quad w_1 = 0.1 \quad w_2 = 0.2$

Ep2:-

$\Delta w_0 = 0 + 0.1(0-1)1 = -0.1$

$\Delta w_1 = 0 + 0.1(0-1)0 = 0$

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

$w_0 = -0.1 + (-0.1) = -0.2$

$w_1 = 0.1 + 0 = 0.1$

$w_2 = 0.2 + (-0.1) = 0.1$

x_1	x_2	t	o	-0.2 $-0.2+0.1$
0	0	0	0	
0	1	0	0 ✓	
1	0	0	0 ✓	
1	1	0	0 ✓	
		0	0 X	

4th Epoch

Error = 20%

Ep4:-

$\Delta w_0 = 0 + 0.1(1-0)1 = 0.1$

$\Delta w_1 = 0 + 0.1(2-0)1 = 0.1$

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

$w_0 = -0.1$

$w_1 = 0.2$

$w_2 = 0.2$

x_1	x_2	t	o
0	0	0	0 ✓
0	1	0	1 X
1	0	0	1 X
1	1	1	1 ✓

Error = 50%

5-th Epoch -

Δw_0 ⋮

←——————→

→ Delta Rule / Gradient Descend Rule

↳ 1st order derivative

→ Error = $\sum_{d \in D} |t_d - o_d|$

↳ absolute error.

Sometimes absolute does not capture the variation.

↳ for the variation → = $\sum_{d \in D} |t_d - o_d|^2$

↳ to capture variations.

→ Mean squared error

↳ = $\left(\frac{1}{N}\right) \sum_{d \in d} |t_d - o_d|^2$

→ Half squared error (Mathematical convenience)

↳ = $\left(\frac{1}{2}\right) \sum_{d \in D} |t_d - o_d|^2$.

$\Rightarrow o_d = \sum_{i=0}^{n} w_i x_i = w_0 x_0 + w_1 x_1 + \cdots + w_n x_n$

* → To compute change in w :

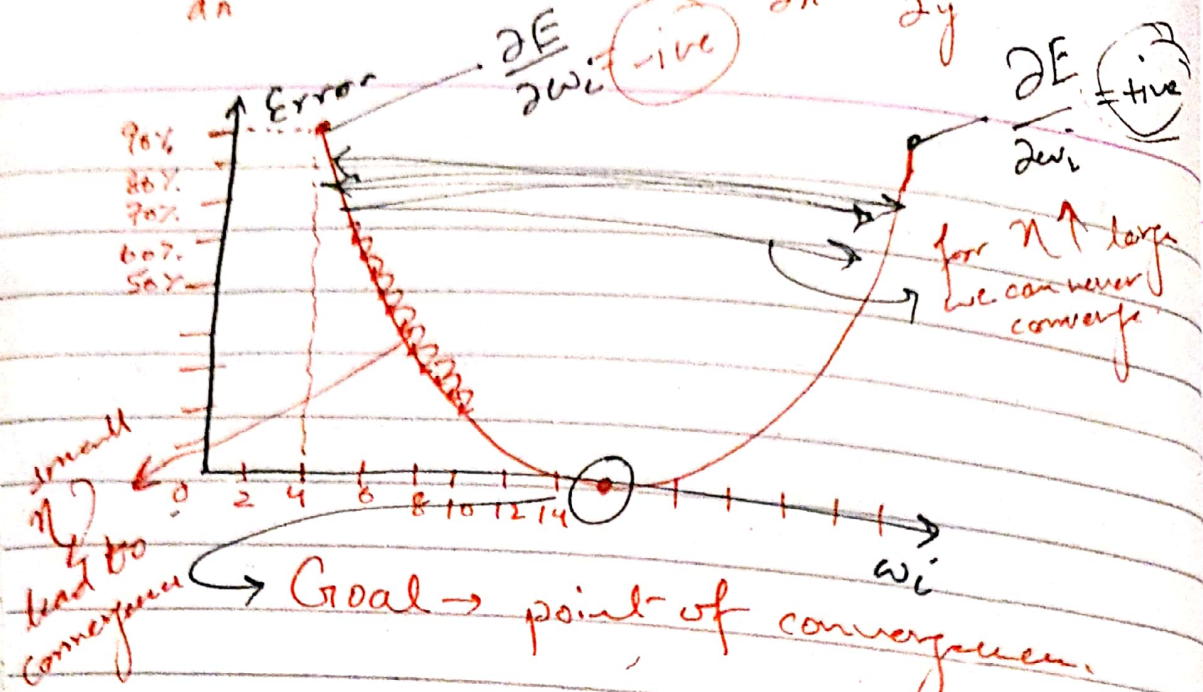
$\Delta w_i = -\eta \cdot \left(\frac{\partial E}{\partial w_i}\right)$ → slope of error

$y = x^3 + 2x + 3$

$\frac{dy}{dx}$

$z = x^2 y^2 + 2x^2 y + y$

$z' = \frac{\partial z}{\partial x} + \frac{\partial z}{\partial y}$



→ Independent ↕ = Dependent -ive.
→ Independent ↕ = Dependent +ive.

w

E slope

Gradient

$\Delta w_i = -\eta \cdot \frac{\partial E}{\partial w_i}$

Hyper parameter

So if slope is negative then w_i should increase.

→ Keep η small, so that movement is small.

Hyper-parameter :-

① Hit n try

② Validation dataset:

→ Using the data-set, we will get the η.

(open research issue)

Proof error derivative -

$\Delta w_i \rightarrow$

$$\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i} \frac{1}{2} \sum_{d \in D} |t_d - o_d|^2$$

⇒ Since o_d is fn of w_i

$$\frac{\partial E}{\partial w_i} \Rightarrow \frac{1}{2} \times 2 \sum_{d \in D} |t_d - o_d|^{2-1} \cdot \frac{\partial}{\partial w_i} |t_d - o_d|$$

$$\Rightarrow \sum_{d \in D} |t_d - o_d| \cdot [0 - (x_i)] \quad \leftarrow o_d = w_0 + w_1 x_i + \ldots = 0 + 0 + x_i + 0 \ldots$$

$$\frac{\partial E}{\partial w_i} = \sum_{d \in D} (t_d - o_d)(-x_i)$$

$$\Delta w_i = -\eta \sum_{d \in D} (t_d - o_d)(-x_i)$$

$$\boxed{\Delta w_i = \eta \sum_{d \in D} (t_d - o_d)(x_i)}$$

$\hookrightarrow$ $w_i = \Delta w_i + \eta (t_d - o_d) x_i$ ← came from here

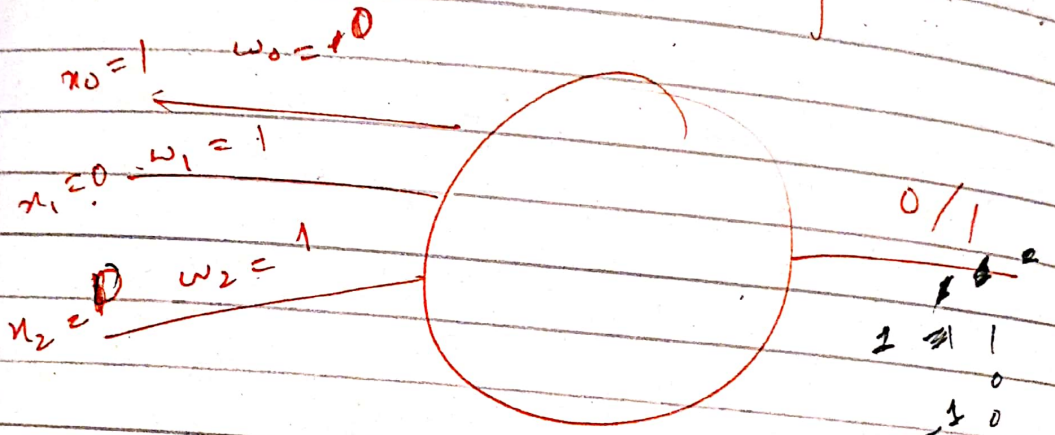
(Raw data / Features)

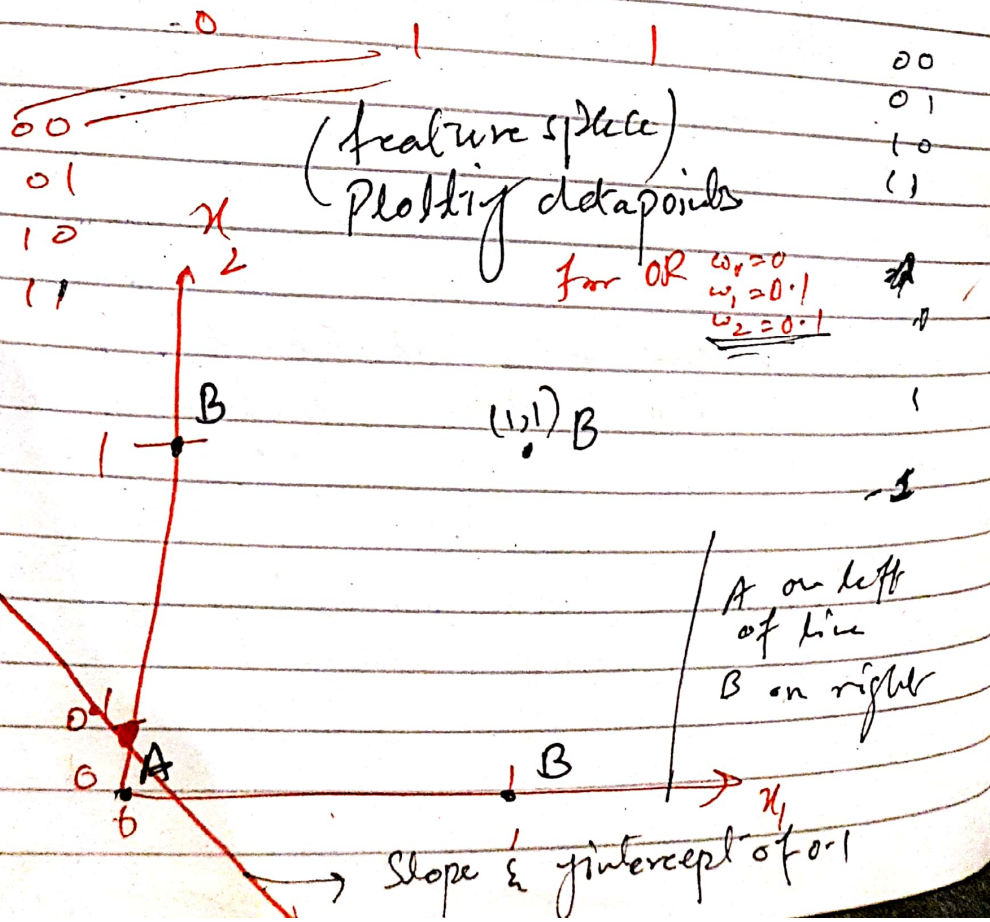
→ Why Neural Networks :-

① Linearly seperable data

Example:-

	x_1	x_2	J	label
E_1	0	0	0	A
E_2	0	1	1	B
E_3	1	0	1	B
E_4	1	1	0	A





XOR

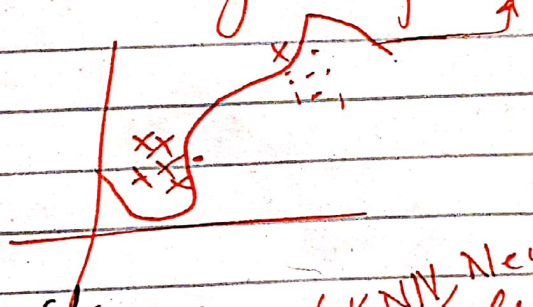
Equation of line:- $ax + by + c = 0$

$y = ax + b$

(slope, intercept)

→ If you can seperate the data points using line then the data is called linearly seperable data. (2 classes of data)

Non linearly seperable data



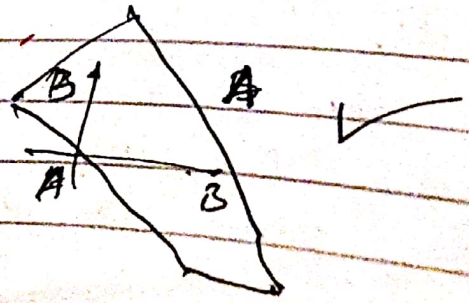
* Classifier:-

→ A classifier is a fn that partitions the feature space into disjoint categories. (KNN, Neural Network etc)

vs. Cluster = (Prediction of labels not given / unsupervised. of labeled. Labels)

→ XOR is not linearly seperable. Hence, we cannot solve the problem using line function. We need some other way.

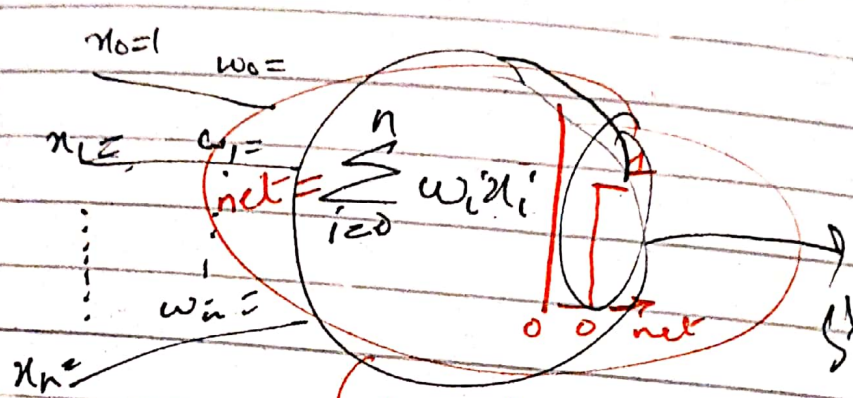
① Multiple lines.



Machine Learning

→ Non-linearly activation functions:

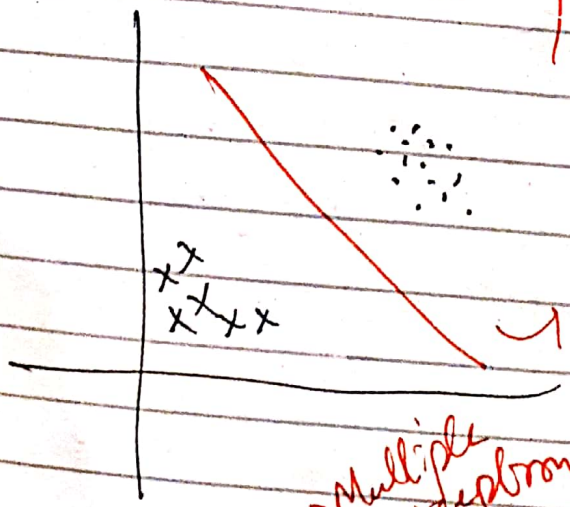
1. Sigmoid function
2. Tanh or Hyperbolic function
3. Relu function
4. Leaky Relu fn
⋮ ⋮



$$\sum_{i=0}^{n} w_i x_i \geq 0$$

$$o/p = \begin{cases} 1; & net \geq 0 \\ 0; & net < 0 \end{cases}$$

other variations



→ perceptrons can learn on this data because linearly separable.

→ Multiple perceptrons.

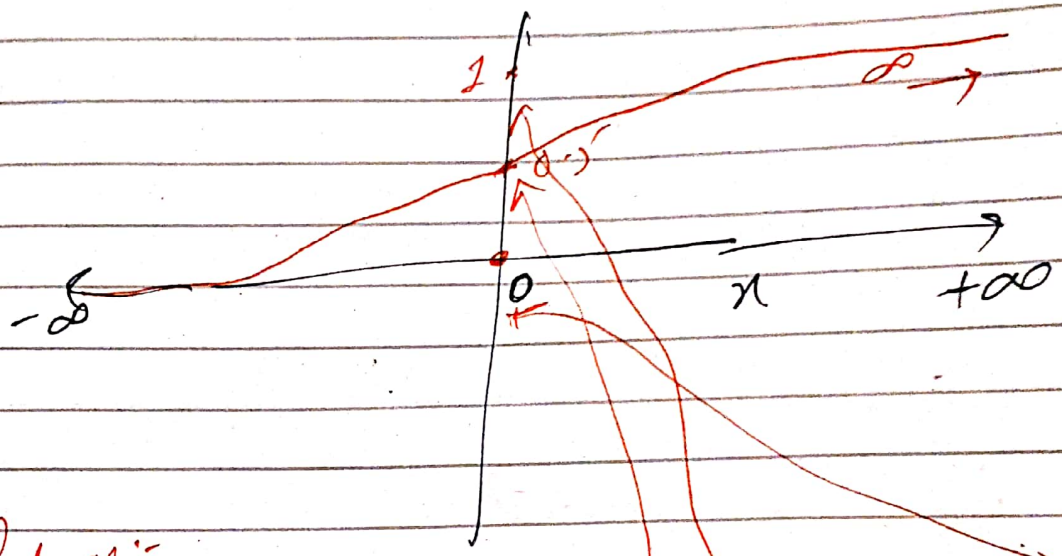
* → Use multiple lines for non-linearly seperated data. But you can have more freedom in the dataspace use a function that have curves (non-linearly activation fn) such as sigmoids.

(curves) To reduce the no. of neural used.

① Sigmoid Fun.

$$\theta d = \frac{1}{1 + e^{-net}}$$

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



Put $-\infty$:-

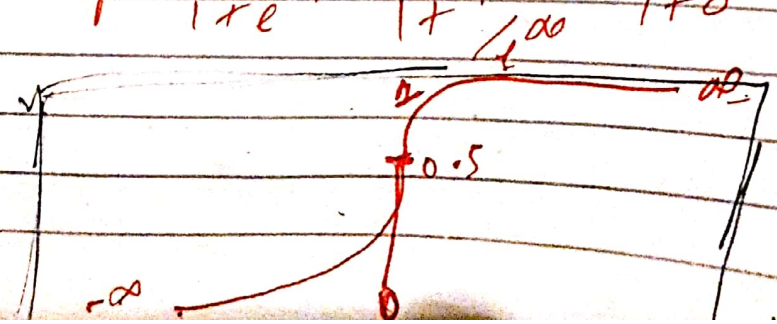
$$y = \frac{1}{1+e^{-x}} = \frac{1}{1+e^{+\infty}} = \frac{1}{1+\infty} = \frac{1}{\infty} = 0$$

Put $x = 0$:-

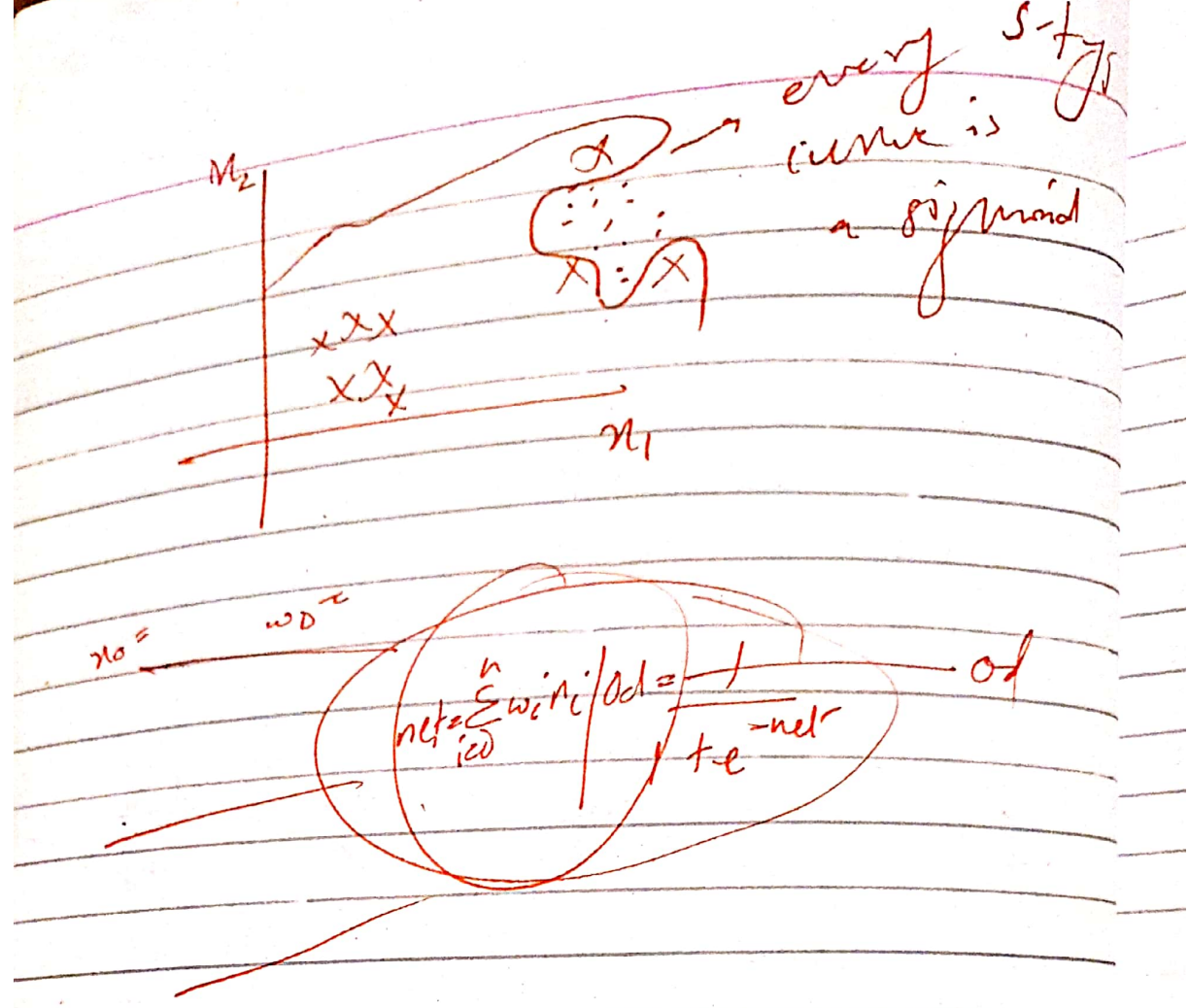
$$y = \frac{1}{1+\frac{1}{e^0}} = \frac{1}{2} = 0.5$$

Put $x = -\infty$:-

$$y = \frac{1}{1+e^{-x}} = \frac{1}{1+\frac{1}{e^{\infty}}} = \frac{1}{1+0} = 1$$

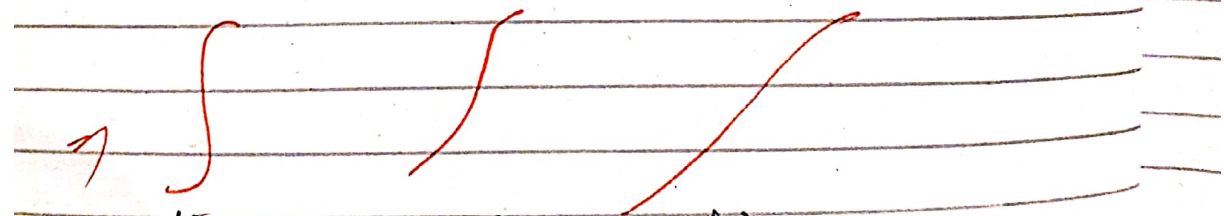


So every S-type curve will be a sig



for more control:-

$$f = \frac{1}{1+e^{-tx}}$$ → controls steepness



Other sigmoid properties:-

1. $f' = f(1-f)$ (very simple)

proof: $f' = \frac{0-(0-e^{-x})}{(1+e^{-x})^2}$

$= \frac{e^{-x}}{(1+e^{-x})^2} \Rightarrow \frac{1}{(1+e^{-x})} \cdot \frac{e^{-x}}{(1+e^{-x})}$

$$= \frac{1}{(1+e^{-x})} \cdot \frac{e^{-x}+1-1}{(1+e^{-x})}$$

$$= f(1-f) = f(1-f) \quad //$$

→ This will help in $\Delta w_i = \eta \cdot \frac{\partial E}{\partial w_i}$

Training Rule equation using Sigmoid function:-

$$\Delta w_i = -\eta \cdot \frac{\partial E}{\partial w_i}$$

Let $E = \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$

$$\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i} \cdot \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

$$= \frac{1}{2} \cdot 2 \sum_{d \in D} (t_d - o_d) \cdot \frac{\partial}{\partial w_i} (t_d - o_d)$$

$$= \sum_{d \in D} (t_d - o_d) \cdot \left[0 - o_d(1-o_d) \cdot \frac{\partial}{\partial w_i} (\text{net}_d) \right]$$

Since the derivative was not working so we put back was with w_i.

Now $\frac{1}{1+e^{-x}}$ this can take more values.

$$\rightarrow \sum_{d \in D} (t_d - o_d) \cdot -o_d(1-o_d) \cdot (+x_i)$$

$$\frac{\partial E}{\partial w_i} = \sum_{d \in D} (t_d - o_d) \cdot -o_d(1-o_d) x_i$$

→ slope for sigmoid.

Putting:-

$$\boxed{\Delta w_i = -\eta \cdot \sum_{d \in D} (t_d - o_d) o_d (1-o_d) x_i}$$

↳ weight change calculation method for sigmoid fn

Backward equation is $\Delta w_i = -\eta \cdot \frac{\partial E}{\partial w_i}$

↳ just put in the activation fn. you need the equation for. Also for different types of error.

(S-II)

Non-linear training data :-

x_1	x_2	y (XOR)
0	0	0
0	1	1
1	0	1
1	1	0

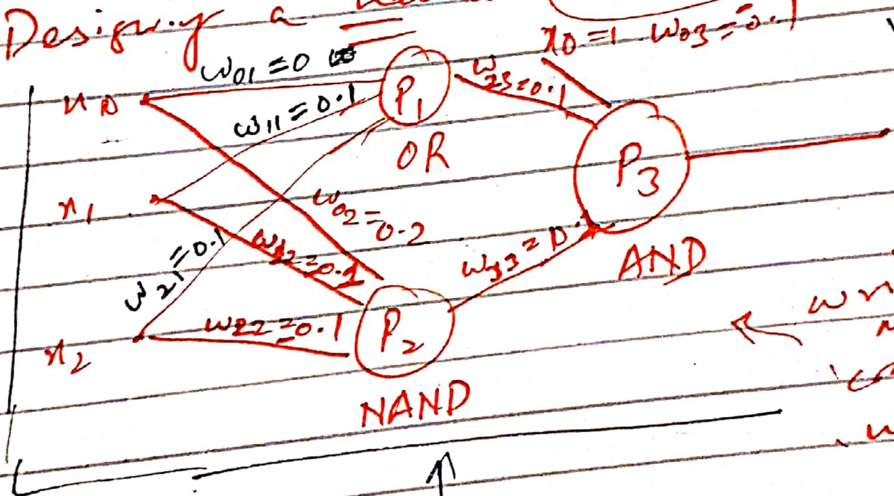
P3

$$x_1 \oplus x_2 = \underbrace{(x_1 \vee x_2)}_{P_1} \wedge \sim \underbrace{(x_1 \wedge x_2)}_{P_2}$$

(Same with this)

we know that this is linearly seperable so we can use perceptron 1,

Designing a neural network → cuz non linear data



XOR
0% Error

← write all the converged weights

w_i = OR
w_i = AND

NAND

$w_0 = 0.2$
$w_1 = -0.1$
$w_2 = -0.1$

for this back-propagation algorithm

for this will calculate $\Delta w_{ij} = -\eta \cdot \frac{\partial E}{\partial w_{ij}}$

0 0 1

1 × 0.1
1 × −0.1 0 1 −0.1 −0.1 = ...
1 × −0.1 −0.2 + 0.1 = ...

0 1 = 0.1 = 0