UNIT - 4

Artificial Intelligence Planning

The intelligent way to do things !

INTRODUCTION

* planning is required to convert objectives into action
* It's decision making task performed by the Robot or comp. sys. to achieve a specific goal.
* planning is the task of coming up with a sequence of actions that will achieve the goal.

example : Goal - Get distinction or clear all subjects
planning - How many days left for exam, No. of chapters to study, Notes availability etc.

What is AI planning ?

Planning is a long-standing sub-area of AI. It's the task of finding a procedural course of action for a declaratively described sys. to reach its goals while optimizing overall performance measures.

Automated planners find the transformations to apply in each given state out of the possible transformations for that state.

Why is it Imp : planning App'(s) in Industry

* Automation is an emerging trend that requires efficient automated planning
* Many app'(s) of planning in Industry (e.g. Robots &

autonomous systems), cognitive sys., cyber security, service composition).

Advices of AI planning Technique

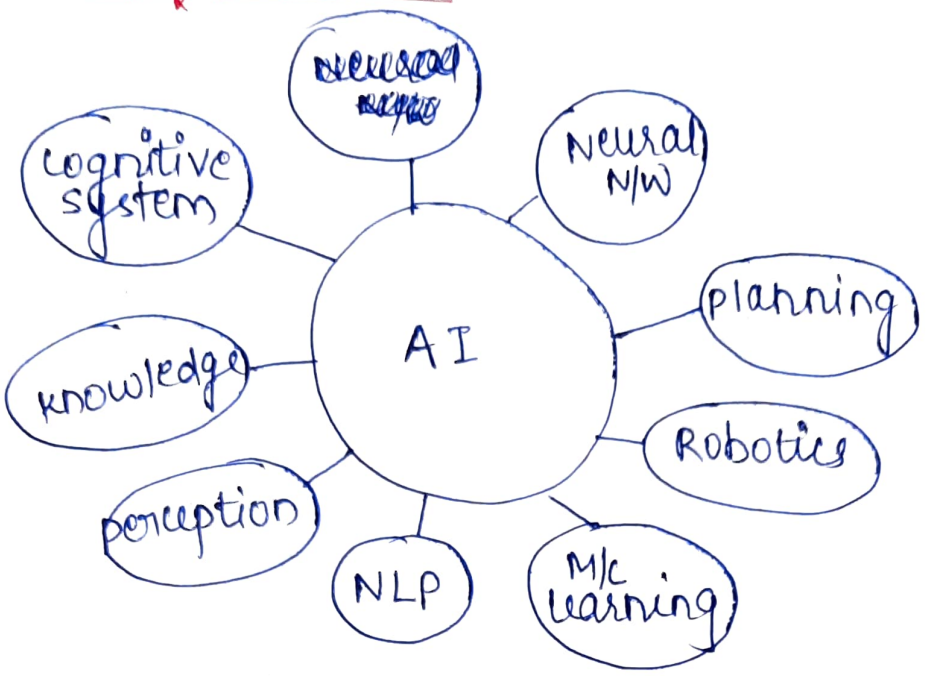
* when explainability is desired
  → when you want to be able to explain why a particular course of action is chosen
  → assignment of responsibility / blame is essential for automation of processes (eg. autonomous driving, medical expert systems)
* Rapid prototyping : short time to solution
* variety of off-the-shelf planners available both IBM proprietary & open source.
* your prob. is frequently changing, even small changes. → no need to change the solution, only tweak the model.

When planning meets Deep Learning (DL)

Solving optimizat° probes. with learning is hard, but integrating planning techniques with heuristic guidance learned by DL will result success.

1) Go player AlphaGO uses planning (Monte-carlo tree search) with DL (Heuristic guidance) to select the next move.
2) Cognitive Assistant Viv (Samsung) uses knowledge graph, planning & DL to answer complicated queries.

PLANNING PROBLEM



Types of planning :

1) Hierarchical planning - a prob. is divided into smaller sub-problems, & each sub prob. is solved one at a time. It's often used in Robotics appⁿ where a Robot needs to figure out how to complete a task by breaking it down into smaller steps.

2) Non-Hierarchical planning - a prob. is not divided into smaller sub problems. Instead, the AI sys. tries to find a single plan that will solve the entire prob. It's often used in Games where a comp. player needs to figure out the best way to win the game.

3) Reactive planning - is a type of planning that is used in situations where the environ. is changing & the AI sys needs to be able to respond quickly.

It's often used in Robotics appⁿ where a Robot needs to be able to avoid obstacle or respond t changes in the environ.

what is planning problem ?

PP in AI are problems that arise when an AI sys. is trying to plan its next move or course of action. These problems can be difficult to solve bcoz the AI sys. must take into a/c all of the possible outcomes of its actions & choose the on that will lead to the best results.

The Planning Problem is the question that how to go to the next move or goal state from the current state. The PP is defined with :

1. Domain Modul - it define the actions along with the objects.
2. Initial state - it's state where any action is yet to take place.
3. Goal state - state which the plan is intended to achieve.

components of Planning System

1) choose the best rule based upon Heuristics (selection of simple subject it's based on past experience or i/p from the seniors or lecture attended)

2) Apply this rule to create a new state (studying of the subject, so as to achieve next stage)

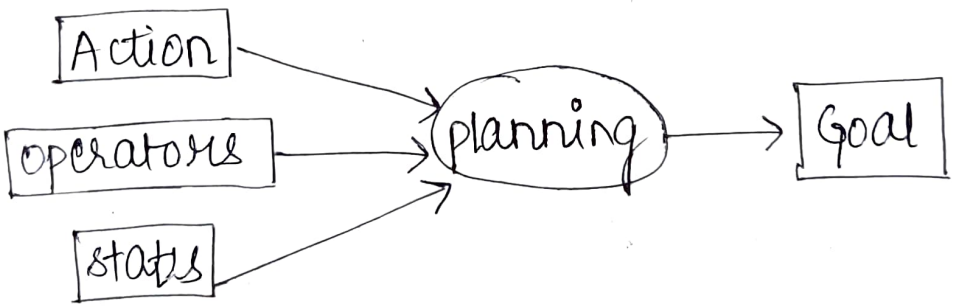
3) Detect when a solution is found (a position is reached where he can score more than cut-off marks)

4) Detect dead ends so that they can be avoided. (find the difficult topic & avoid them)

5) Detect when a nearly solved state occurs & use special methods to make it a solved state. (In position to get more than cut-off marks in all subjects except Maths & then apply strategies to reach the goal)

Planning Representation

It consist of 3 components : state, Action & Goal



1. State → is conjunction of +ve literals which can't contain variables & invoke f'cs). This is represented of facts.
2. Action → these are precond'cs) that must hold before execut'n & the effects after execution
3. Goal → It is the most specificied state; a state is satisfy the given goal if it consists of all objcs reqd. for the goal.

Agent tries to find a solution i.e a sequence of actions that solves a problem.

Planning Agent = problem solving agent + Knowledge Based Agent

(Generate sequences of actions to perform tasks and achieve objectives)

Problem solving agent → to consider the consequences of sequences of actions before acting

Knowledge Based Agent → can select actions based on explicit logical representat° of the current state & the effects of actions.

PLANNING LANGUAGES

PL should be expressive & every planning lang. makes use of the representat° schema so that the algo. can be operated on it.

Types:- 1. standard Research Institute problem solver (STRIPS)

2. Action Discription Lang. (ADL)

3. planning Domain Discription Lang. (PDDL)

(1) STRIPS :

* developed in 1970s at Stanford for the first intelligent robot
* it can describe the world (initial state, goal state) by providing objs, Actions, precond & effects.

* A STRIPS instance is composed of : 1) State 2) Goal 3) Action

State - It's conjunction of +ve literals which can't contain variables & invoke f^cs.

ex. At(Home) ^ Have(Banana)

Goal - these are conjunction of literals may contain variables. ex At(Home) ^ ¬ Have(Banana)

At(X) ^ Sells(X, Banana)

Action - these are precond(s) that must hold before execution & the effects after execution.

ex. Action (fly(P, from, to)

PRECOND : At(P, from) ∧ plane(P) ∧ Airport(from) ∧ Airport(to)

EFFECT : ¬At(P, from) ∧ At(P, to))

(2) ADL (Action Descript^n Lang.) :

* used to overcome the limitations of STRIPS
* more expensive than STRIPS
* propes) : 1. It allows −ve literals
2. It use quantified variables with the disjunction & the conjunction.

ex. ¬ poor ∧ (famous ∨ smart)

3. cond^al post cond^s are allowed
4. variables with diff. types at the same time are allowed & also equality prop. is available.

ex. consider car driving case with STRIPS:

Action (drive (c, from, to))

pre_cond^n : at (c, from) ∧ car(c)

post cond^n : at (c, from) ∧ at (c, to)

In strips, post cond^n means remove the 1st 'at' cond^n & add the 2nd 'at' cond^n.

With ADL, same action is represented as follows:

Action (drive (c, from, to))

pre_cond^n : at (c, from) ∧ car(c) ∧ (from ≠ to)

post_cond^n : at (c, from) ∧ at (c, to)

(3) PDDL : its std encoding Lang. for 'classical' planning task.

* superset of STRIPS & ADL
* components of PDDL planning task :

1) objects → things in the world that interest us
2) predicates → propers. of objc(s) that we are interested in ; can be true or false.
3) Initial state → state of the world that we start in.
4) Goal specification → things that we want to be true
5) Actions / operators → ways of changing the state of the world.

planning tasks specified in PDDL are separated into two files:

1. Domain file for predicates & actions
2. problem file for objects; initial state & goal specification.

Domain file looks like:

```
(define (domain <domain_name>)
<PDDL code for predicate>
<PDDL " " first action>
<PDDL " " last " >
)
```

Problem files look like:

```
( define (Problem <Problem_name>)
 (: domain <domain_name>
 <PDDL code for objects >
 <PDDL " " initial state>
 <PDDL " " Goal>
```

BLOCKS WORLD

* also known as Sussman Anomaly
* In this prob., 3 blocks labeled as 'A', 'B', 'C' are allowed to rest on the flat surface. The given cond^n is that only one block can be moved at a time to achieve the goal.

It's consist of following :

1) table 2) Identical blocks with unique letters on them
3) Blocks are put one to another in stack form
4) stack is built with a robot arm. The arm can perform op's of lifting a single block at a time & placing it.

example predicate used to describe states in block world are :

1. On (A, table) : Block A is on the table
2. On (B, table) : " B " "
3. On (B, A) : Block B is on Block A
4. Clear (A) : Block A has nothing on it.
5. Clear (B) : " B " "
6. Holding (B) : Robot arm is holding B
7. Empty-Arm : arm is not holding anything

State Representation -

Initial state : On (A, table) ∧ On (B, table) ∧ clear (A) ∧ clear (B) ∧ Empty-Arm

Goal state : On (A, table) ∧ On (B, A) ∧ Clear (B) ∧ Empty-Arm

4 action (op) used to describe states in Block world are :

1. UNSTACK (B, A) : to lift block B from A
2. Stack (B, A) : To place block B on A
3. Lift (B) : to lift the block B from the table

4. Place(B): to put the block B on the table.

The Blocks world is chosen because:

* It's sufficiently simple & well-behaved
* easily understood
* provides a good sample environ. to study planning: problems can be broken into nearly distinct subproblems.

GOAL STACK PLANNING

* one of the earliest methods in AI in which we work backwards from the goal state to the initial state.
* this approach uses a Stack for plan generation. The stack can contain sub-goal & actions described using predicates.
* the sub-goals can be solved one by one in any order.
* we start at the Goal state & we try fulfiling the pre-cond'(s) reqd. to achieve the initial state.
* we keep solving these 'goals' & 'sub-goals' until finally arrive at the initial state.

The Robot Arm can perform 4 op'(s):

* STACK (X, Y): stacking Block X on Block Y
* UNSTACK (X, Y): picking up Block X which is on top of Block Y

* PICKUP(X): picking up Block X which is on top of the table.

* PUTDOWN(X): put Block X on the table

Operation	Pre-condition	Action
1. PICK UP(X)	ARM-EMPTY ^ ON(X, table) ^ Clear(X)	Holding(X)
2. PUTDOWN(X)	Holding(X)	Arm-Empty ^ On(X, table) ^ Clear(X)
3. STACK(X,Y)	Holding(X) ^ Clear(Y)	On(X,Y) ^ Clear(X) ^ Arm_Empty
4. UNSTACK(X,Y)	On(X,Y) ^ Clear(X) ^ Arm-Empty	Holding(X) ^ Clear(Y)

Ex. Initial state:

Goal state:

Solution

Step 1: our goal is ON(A, B)

explanation → goal is achieved by opr no. 3 so,

Step 2: *Stack(A, B)

Step 3: Holding(A) ^ Clear(B) - precondn

Explanation → In step 3, Holding(A) is not true, when we see initial state. It's expand by opr no. 1.

Step 4 : *PICKUP(A)

Step 5 : Arm Empty ^ ON(A, table) ^ Clear(A) - precond^ TRUE

Note: Hence, here step 5 is True, so, it's also prove the step 3.

So, when step 3 & 5 true then step 2 *Stack(A,B) TRUE & we can achieve goal ON(A,B)

precond's are underlined with Red & actions are marked as star!

MEAN END ANALYSIS

it solves probss. by defining the goal & establishing the right action plan. This technique is used in AI progss. to limit search.

It combines forward & Backward strategies to solve complex problems.

In this technique, sys. evaluates the diff. b/w current state / position and target / goal state. It then decides the best action to be undertaken to reach the end goal.

How MEA works

1) First, sys. evaluates the current state to establish whether there is a prob.. If prob. is identified then it means that an action should be taken to correct it.

2) Second step involves defining the target or desired goal that needs to be achieved.

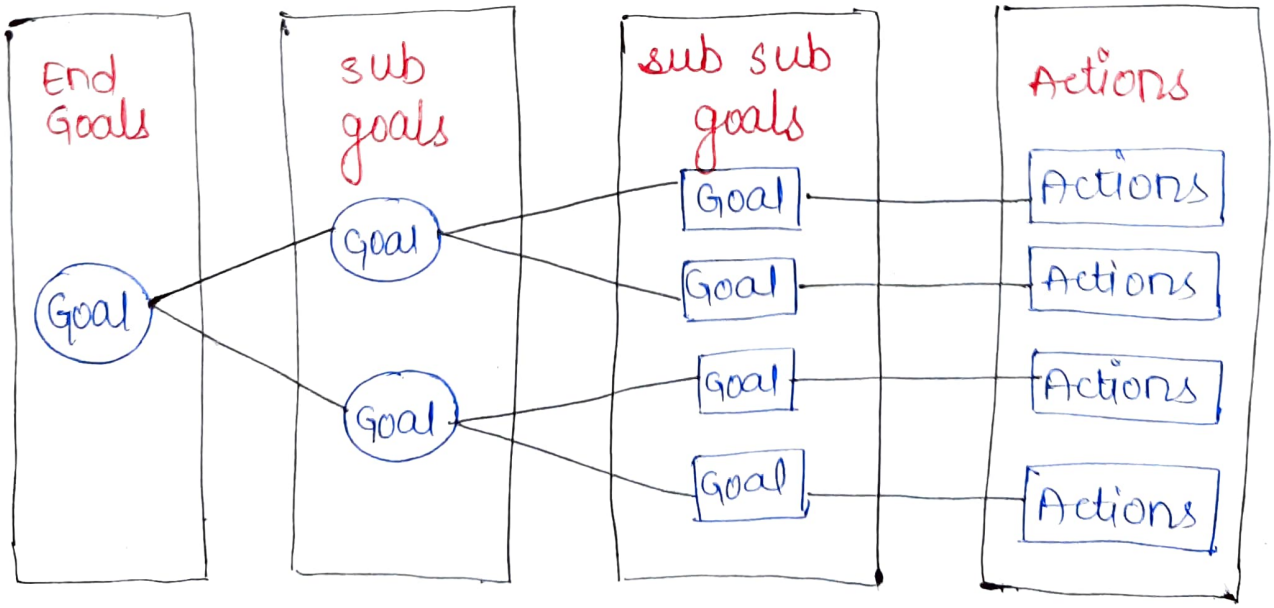
3) Target goal is split into sub-goals, that are further split into other smaller goals.

4) it involves establishing the actions / ops that will be carried out to achieve the end state.

5) In this, all the sub-goals are linked with corresponding executable actions (ops)

6) After that is done, intermediate steps are undertaken to solve the probes. in the current state The chosen operators will be applied to reduce the diff. b/w the current state & end state.

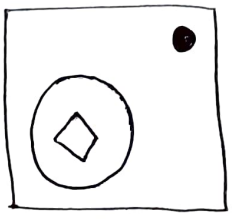
7) it involves tracking all the changes made to the actual state. changes are made until the target state is achieved.



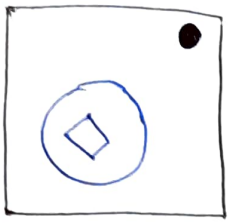
Algo. steps for MEA:

1. Conduct a study to assess the status of the current state. This can be done at a macro or micro level.
2. Capture the problem in the current state & define the target state. This can also be done at a macro or micro level.
3. Make a comparison b/w the current state & end state that you defined. If these states are the same, then perform no further action. This is an indication that the prob. has been tackled. If the two states are not the same, then move to step 4.
4. Record the differences b/w two states at the two aforementioned levels (macro & micro)
5. Transform these differences into adjustments to the current state.
6. Determine the right action for implementing the adjustments in step 5.
7. Execute the changes & compare the results with the target goal.
8. If there are still some diff. b/w current state & target state, perform course correction until the end goal is achieved.

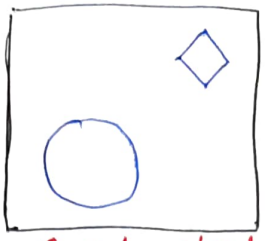
ex. Initial state:



Apply the concept of MEA to establish whether there are any adjustments needed. The first step is to evaluate the initial state & compare it with the end goal to establish whether there are any diff. b/w the two states.



Initial State



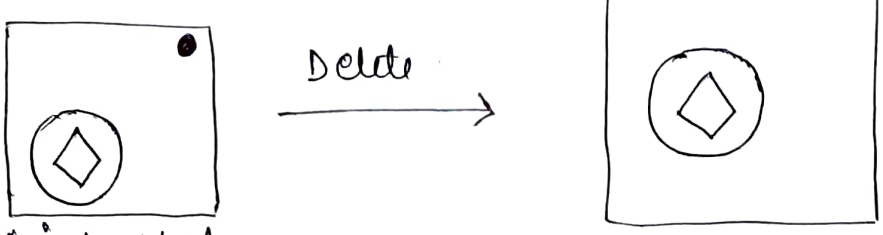
Goal state

Above images shows that there is a diff. b/w current & the target state, this indicates that there is a need to make adjustments to the current state to reach the end goal.

The goal can be divided into sub-goals that are linked with executable actions or op^rcs).

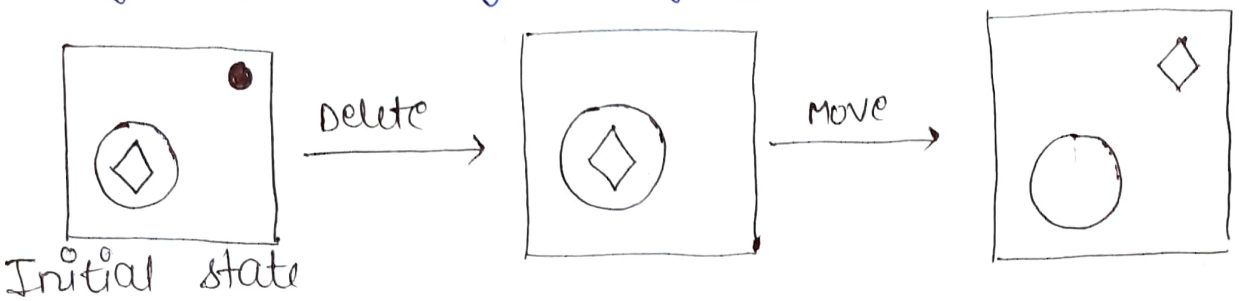
The follo. are the 3 operators that can be used to solve the problem.

1. Delete operator - dot symbol at the top right corner in the initial state doesn't exist in goal state. The dot symbol can be removed by applying the delete operator.

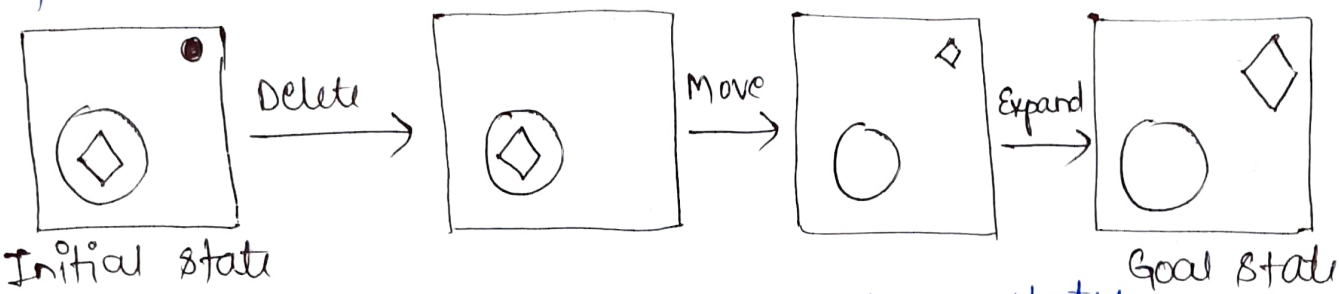


Initial state

2. Move operator - Now, compare the new state with 9 the end state. The diamond in the new state is inside the circle while in the end state, it's at the top right corner. move this diamond symbol to the right position by applying the move operator.



3. Expand operator - after evaluating the new state generated in step 2, find that the diamond symbol is smaller than the one in the end state, we can ↑ the size of the symbol by applying expand operator.



There are no diff. b/w these two states; which means that prob. has been solved.

Applications of MEA

* organizational planning → MEA used in org'ns to facilitate general mgmt. It helps org'al managers to conduct planning to achieve the objectives of the org'. The mgmt reaches the desired goal by dividing

the main goals into sub-goals that are linked with actionable task.

* Business Transformation → this technique is used to implement transformat^{n} projects. If there are any dsired changes in the current state of a busi. project, MEA is applied to establish the new prousses to be implemented. The processes are split into sub-processes to enhance effective implementation.

* Gap Analysis → is the comparison b/w the current performance & the reqd. performance. MEA is applied in this field to compare the existing technology & the dsired tech. in org'cs. various op^{r}cs are applied to fill the existing gap in technology.

MACHINE LEARNING

is a branch of AI which enables m/cs to learn from past data or experiences without being explicitly programmed.

ML enables a comp. sys. to make predictions or take some decisions using historical data without being explicitly programmed.

ML process: ML sys. learns from historical data, builds the prediction moduls & whenever it receives

the main goals into sub-goals that are linked with actionable task.

* Business Transformation → this technique is used to implement transformat^n projects. If there are any desired changes in the current state of a busi. project, MEA is applied to establish the new prouses to be implemented. The processes are split into sub-processes to enhance effective implementation.

* Gap Analysis → is the comparison b/w the current performance & the reqd. performance. MEA is applied in this field to compare the existing technology & the desired tech. In org's. various op's are applied to fill the existing gap in technology.

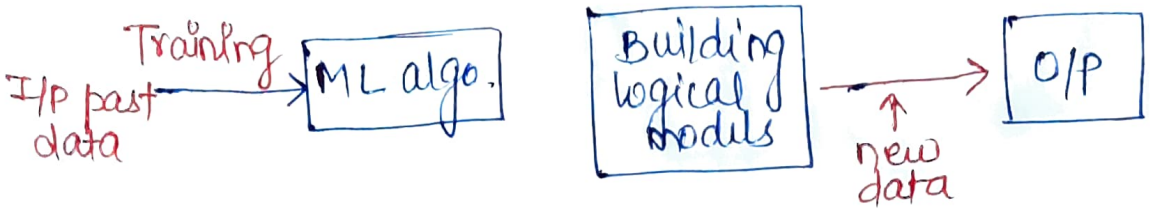
MACHINE LEARNING

is a branch of AI which enables m/cs to learn from past data or experiences without being explicitly programmed.

ML enables a comp. sys. to make predictions or take some decisions using historical data without being explicitly programmed.

ML process: ML sys. learns from historical data, builds the prediction moduls & whenever it receives

new data, predicts the o/p for it.



Goal of ML

* primary objective is to develop general purpose algo. in practical value.
* Main purpose is to study & design the algos that can be used to produce the predicates from the given dataset.

ML can also be used to:

→ enhance CX service
→ predict journey times
→ predict how long Jobs may take
→ Behaviour of Market

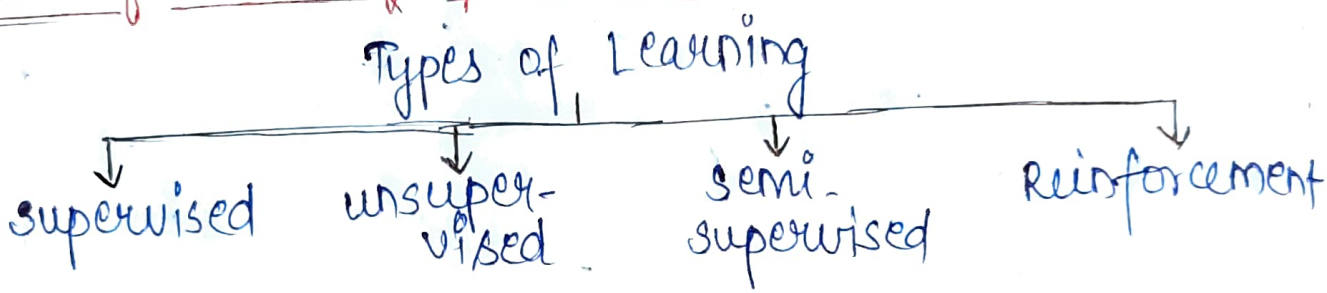
Applications of ML:

1. Image Recognition
2. speech "
3. Traffic prediction
4. product Recommendatn
5. Self-driving cars
6. email spam & Malware filtering
7. virtual personal assistant
8. Online fraud Detection
9. stock market trading
10. Medical Diagnosis.

Challenges for ML:

1. Data collection
2. Less Amount of Training Data
3. Non-representative " "
4. poor Quality of data
5. Irrelevant / unwanted features.

Classification of ML



(1) Supervised Learning

Its a type of ML method in which we provide sample labeled data to the ML sys. in order to train it, & on that basis, it predicts the o/p.

(2) Unsupervised Learning

is a learning method in which a M/c learns without any supervision.

(3) semi-supervised learning

its working lies b/w supervised & unsupervised techniques. we use these, when we are dealing with a data which is a little bit labeled & rest large portion of it is unlabelled. Mostly applicable

in case of image data-sets where usually all images are not labeled.

(4) Reinforcement Learning

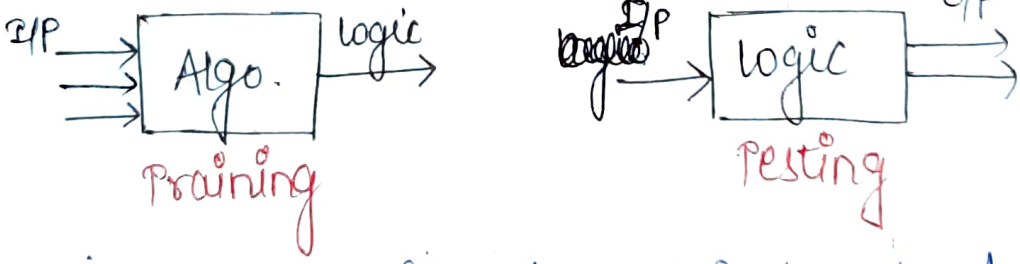
It's a feedback based " method in which a learning agent gets a reward for each right action & gets penalty for each wrong action.

SUPERVISED LEARNING

* a m/c is trained using labeled data. Datasets are said to be labeled when they contain both I/p and o/p parameters, and on that basis, it predicts the o/p.

This implies that some data is already tagged with the correct answer.

*



ex Basket filled with diff. kinds of fruits.

Classification : Supervised learning

1. Classification → A classificat" prob. is when the o/p variable is a category such as 'Red', 'Blue', 'disease' and 'no disease'.
2. Regression → when the o/p variable is a real value, such as 'dollars', 'weight'.

Categories of supervised learning:

* Regression
* classification
* K-NN (nearest neighbors)
* Support Vector M/c (SVM)
* logistic Regression
* Naive Bayes classifiers
* Decision Tree

UNSUPERVISED LEARNING

* m/c learns without any supervision
* training is provided to the m/c with the set of data that has not been labeled, classified.
* No training will be given to the m/c. ∴ the m/c is restricted to find the hidden struc. in unlabeled data by itself.
* classification of unsupervised learning:

1) Clustering →
A clustering prob. is where we want to discover the inherent groupings in the data, such as grouping cxes by purchasing behavior.

2) Association →
Discover rules that describe large portions of data such as ppl that buy X also tend to buy Y.

example: suppose it's given an image having both dogs & cats which it has never seen.

m/c has no idea about the features of dogs & cats. So, we can't categorize, but it can categorize them acc. to their similarities, patterns & differences.

categorize the above image into two parts; first may contain all pics having dogs & second may contain all pics having cats.

Clustering:

* find patterns in the data that we are working on.
* it may be the shape, size, color etc. which can be used to group data items or create clusters.

popular algos of clustering are:

1) Hierarchical clustering - this algo builds clusters based on the similarity btw diff data points in the dataset.

2) K means clustering - algo. creates clusters of diff. data pts which are as homogeneous as possible by calculating the centroid of the cluster & making sure that the dist. btw this centroid & each data pt is as less as possible.

3) K-NN clustering also k/a lazy learner bcoz it learns only when algo. is given a new data pt. It works well with smaller datasets as large datasets take time to learn.

Association :

In this type of learning, we find the dependencies of one data item to another data item & map them such that they help you profit better.

1) Apriori Algo. →

* based on breadth - first search
* this algo. maps the dependency of one data item with another which can help us understand what data item influences the possibility of something happening to the other data item.
* ex: bread influences the buyer to buy milk & eggs. so that mapping helps ↑es profit for the store.

2) FP - Growth Algo. → (frequent pattern)

* FP algo. finds the count of the pattern that has been repeated, adds that to a table & then finds the most possible item & sets that as the root of the tree.
* this algo is faster than Apriori as the support is calculated & checked for ↑ing iterations rather than creating a rule & checking the support from the dataset.

$$\text{support} = \frac{\text{how many times the items are brought together}}{\text{total no. of Tx's}} * 100\%$$

$$\text{Confidence}_{(X \to Y)} = \frac{\text{how many times items X, Y are brought together}}{\text{how many times an item X is brought}} * 100\%$$

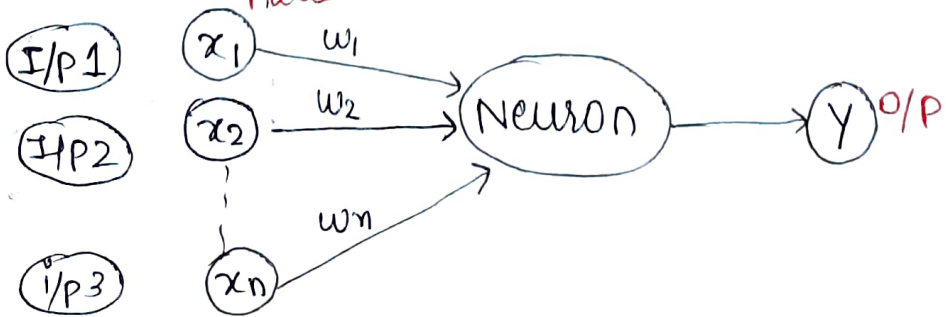
ARTIFICIAL NEURAL NETWORK

ANN is derived from Biological NN that develop the struc. of a human brain. The Human brain has hundreds of billions of cells k/a Neurons.

Each Neuron is made up of a cell body that is responsible for processing info. by carrying info. towards (i/ps) and away (o/ps) from the brain.

What is ANN ?

- ANN are a special type of ML algo. that are modeled after the human brain.
- How the neurons in our nervous sys. are able to learn from the past data & provide responses & similarly, ANN is able to learn from the data & provide responses in the form of predictions or classifications.
- ANNs are Non-linear statistical models which displays a complex rel'ship b/w i/ps & o/ps to discover a new pattern.
- adv. - it learns from the example data sets.
- ANN is also capable of taking sample data rather than the entire dataset to provide the o/p result.



ANN ARCHITECTURE

The f^n ing of the ANN is similar to the way neurons work in our nervous sys.

In NN, there are 3 essential layers :

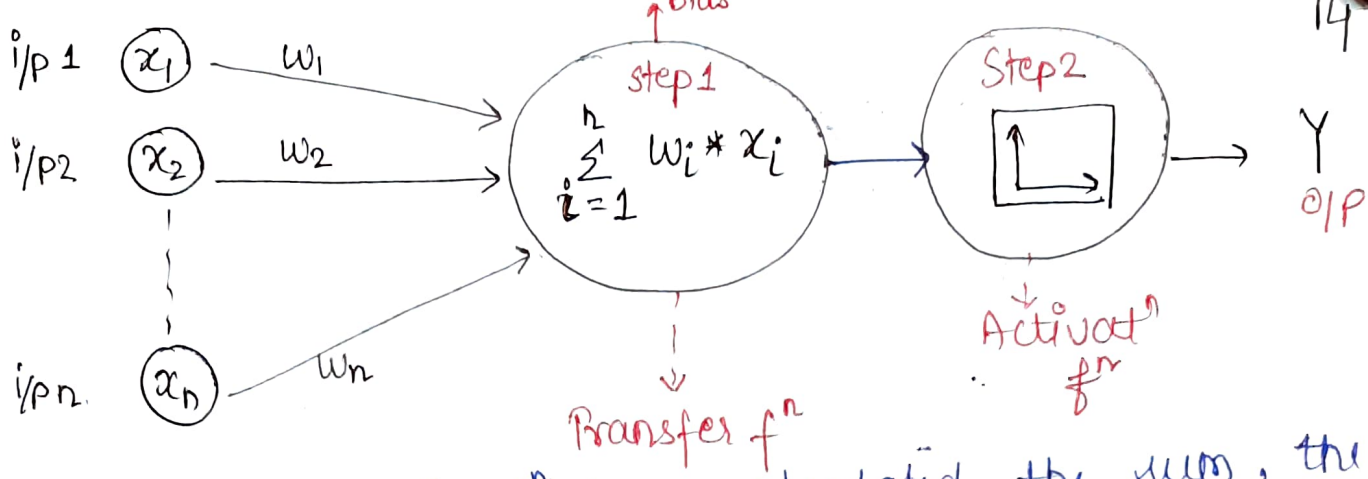
1) INPUT LAYER → first layer of an ANN that receives the i/p info. in the form of various texts, nos., audio files, image etc.

2) HIDDEN LAYER → In the middle of ANN model are the Hidden Layers. There can be a single hidden layer, as in the case of a perceptron or multiple hidden layers. These layers perform various types of mathematical computat^n on the i/p data & recog-nize the patterns that are part of.

3) OUTPUT LAYER → we obtain the result that we obtain through rigorous computations performed by the middle layer.

In a NN, there are multiple parameters and hyperparameters that affect the performance of the model. The o/p of ANNs is mostly dependent on these parameters. Some of them are : weights, biases, learning rate, batch size etc.

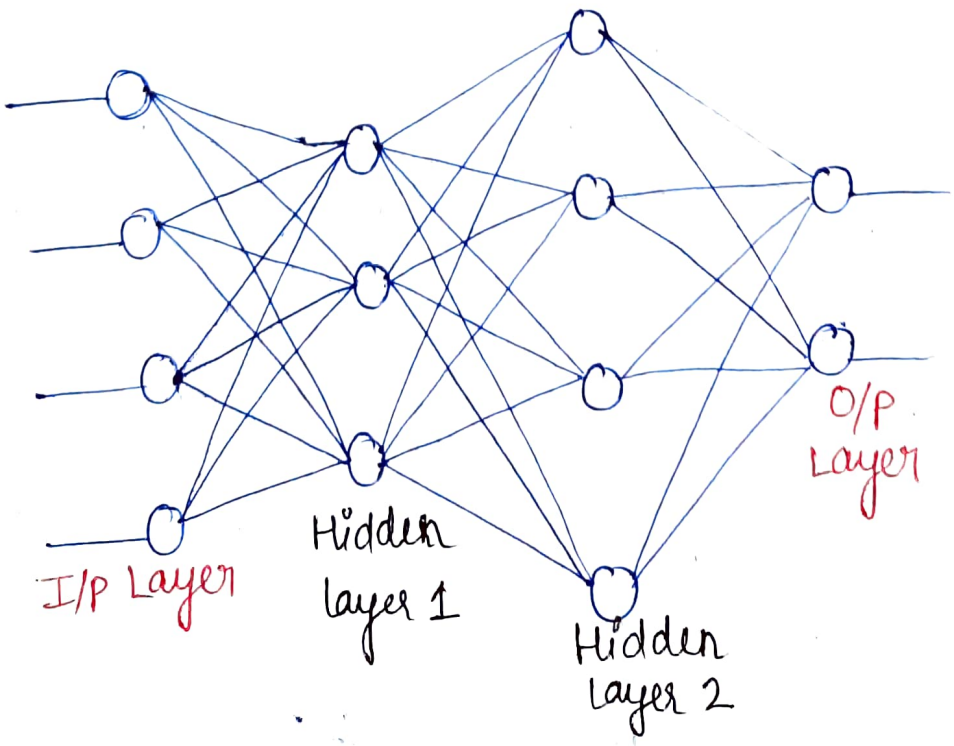
Each node in the n/w has some weights assigned to it. A Transfer f^n is used for calculating the weighted sum of the i/ps & the bias.



Aftr the transfer f^n has calculated the sum, the Activation f^n obtains the result. Based on the o/p received, the activat^n f^n fire the appropriate result from the node.

ex: if o/p received is above 0.5, the activat^n f^n fires a 1 otherwise 0.

Based on the value that the node has fired, we obtain the final o/p. Then, using the error f^n(s), we calculate the discrepancies b/w predicted o/p and resulting o/p & adjust the wt. of the NN through a process known as Backpropagation.



Neural N/ws / ANN / simulated NN (SNN)

ANN are comprised of a node layers, containing an I/p Layer, one or more hidden layers & an O/p Layer. Each node connects to another & has an associated wt. and threshold. If the O/p of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the n/w. Otherwise, No data is passed along to the next layer of the n/w.

$\sum w_i x_i + bias = w_1 x_1 + w_2 x_2 + \dots w_n x_n + bias$

$$\text{Output} = f(x) = \begin{cases} 1 & \text{if } \sum w_i x_i + b > 0 \\ 0 & \sum w_i x_i + b < 0 \end{cases}$$

This process of passing data from one layer to the next layer defines this NN as a Feedforward N/w.

ex. whether you should go surfing or not. Assume that 3 factors influencing your decision making:

1. Are the waves good ? (Yes : 1, No : 0)
2. Is the line-up empty ? (Yes : 1, NO : 0)
3. Has there been a recent shark attack ? (Yes : 0, No : 1)

Let's assume the follo. i/ps :

$x_1 = 1$, since the waves are pumping

$x_2 = 0$, since the crowds are out

$x_3 = 1$, " there hasn't been a recent shark attack

Now, assign some wts to determine importance

Large wt. - greater importance to decision or outcome

$w_1 = 5$, since large swells don't come around often

$w_2 = 2$, " you're used to the crowds

$w_3 = 4$, " you have a fear of sharks

$$\text{Y-hat} = (1*5) + (0*2) + (1*4) - 3 = 6$$

O/p would be 1, since 6 is greater than 0.

BACK PROPAGATION

Method of fine-tuning the wts of a NN based on the error rate obtained in the previous iteration. proper tuning of the wts allows you to reduce error rates & make the model reliable by ↑ing its generalization.

It is a short form for Backward propagation of errors. It's a std method of training ANN

It helps to calculate the gradient of a loss f^n wrt all wts in the n/w.

How Backpropagation Algo works?

It computes the gradient of the loss f^n for a single wt. by the chain rule.

It efficiently computes one layer at a time,

How Backpropagation algo. works:

1. I/Ps X, arrive through the preconnected path
2. i/p is modeled using real weights W. The weights are usually randomly selected.
3. Calculate the o/p for every neuron from the I/p layer, to the hidden Layers, to the o/p layer.
4. Calculate the error in the o/ps:

$$\text{Error} = \text{Actual o/p} - \text{Desired o/p}$$

5. Travel back from the o/p layer to the hidden layer to adjust the wt.(s) such that the error is ↓ed.

Keep repeating the process until the desired o/p is achieved.

Why we need Backpropagation ?

* It's fast, simple & easy to program
* It has No parameters to tune apart from the nos. of i/p
* It's a flexible method as it doesn't require prior knowledge about the n/w.
* Std method that generally works well.
* It doesn't need any spcl mention of the features of the f^n to be learned.

FEED FORWARD NETWORK

It's an Artificial NN where the nodes never form a cycle. This kind of NN has an i/p Layer, & an

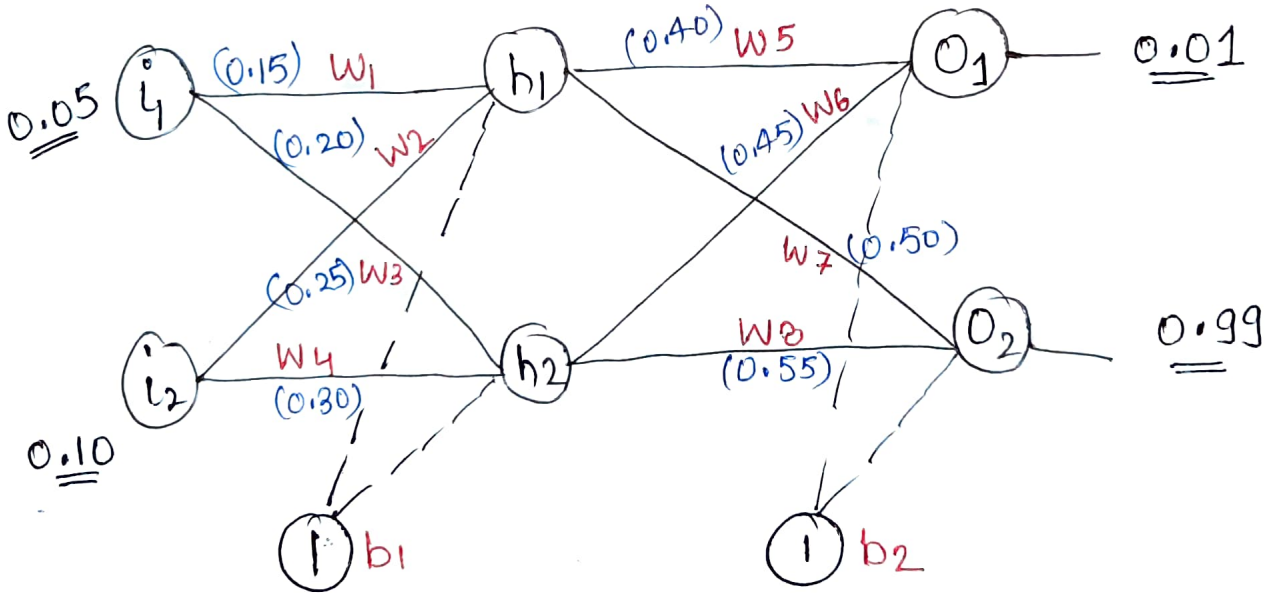
O/p layer.

Types of Backpropagation Network :

* **Static Backpropagation** → kind of BP n/w which produces a mapping of a static i/p for static o/p. Its useful to solve static classification issues like OCR. (optical character Recognition).

* **Recurrent Backpropagation** → Recurrent BP in data mining is fed forward until a fixed value is achieved. After that, the error is computed & propagated backward.

Example :-



Goal of backpropagation is to optimize the wt.cs so that the NN can learn how to correctly map arbitrary i/ps to o/ps.

Given: I/Ps ⇒ 0.05 & 0.10

O/Ps ⇒ 0.01 & 0.99

Calculate total net o/p for h_1 :

$net_{h_1} = w_1 i_1 + w_2 i_2 + b_1 * 1$

$net_{h_2} = 0.3925$

$= 0.15 * 0.05 + 0.20 * 0.10 + 0.35 * 1 = 0.3775$

then squash it using the logistic f^n to get th o/p of h_1 :

$$out_{h_1} = \frac{1}{1 + e^{-net_{h_1}}} = \frac{1}{1 + e^{-0.3775}} = 0.593269$$

$Out_{h_2} = 0.596884$

Repeat this process for the o/p Layer neurons, using the o/p from the hidden layer neurons as i/ps :

$net_{o_1} = w_5 * out_{h_1} + w_6 * out_{h_2} + b_2 * 1$

$net_{o_1} = 0.4 * 0.5932699 + 0.45 * 0.596884 + 0.6 * 1$

$net_{o_1} = 1.105905$

$net_{o_2} = 1.22492091$

$$Out_{o_1} = \frac{1}{1 + e^{-net_{o_1}}} = 0.75136507$$

$Out_{o_2} = 0.772928465$

$$\text{Total Error} = \sum \frac{1}{2}(\text{target} - \text{o/p})^2$$

$E_{o_1} = \frac{1}{2}(target_{o_1} - out_{o_1})^2 = \frac{1}{2}(0.01 - 0.751365)^2 = 0.274811083$

$E_{o_2} = 0.023560026$

$E_{total} = E_{o_1} + E_{o_2} \Rightarrow 0.274811 + 0.023560 = 0.298371109$

SUPPORT VECTOR MACHINE

SVM is a simple & powerful supervised ML algo. that can be used for building both Regression & classification models. SVM algo. can perform really well with both linearly & non-linearly separable datasets.

Types:

1. **Linear or Simple SVM** → is used for linearly separable data. If dataset can be classified into two classes with a single straight line, then data is considered to be linearly separable data & the classifier is referred to as the linear SVM. Typically, used for linear Regression & classification problems.

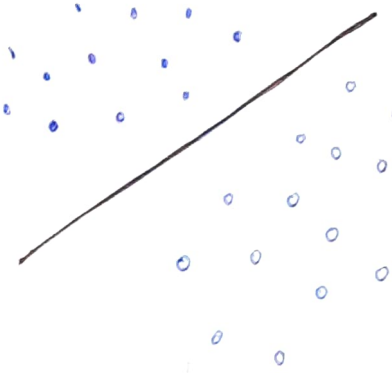
2. **Non-Linear or Kernel SVM** → used for Non-linear separated data i.e. dataset that can't be classified by using a straight line. The classifier used in this case is referred as Non-linear SVM classifier. It has more flexibility for non-linear data bcoz more features can be added to fit a hyperplane instead of a 2-D space.

Example: (Linear SVM)

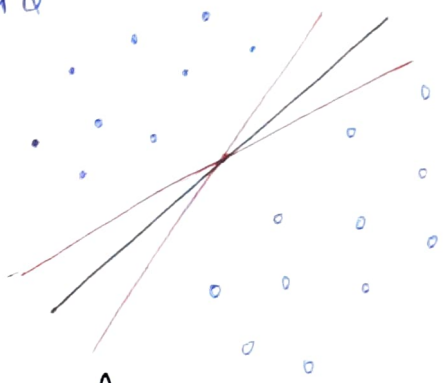
SVM algo. is based on the concept of 'Decision planes' where hyperplanes are used to classify a set of given objects.

In given fig., have two sets of data. These datasets can be

separated easily with the help of a line, kya Decision Boundary.

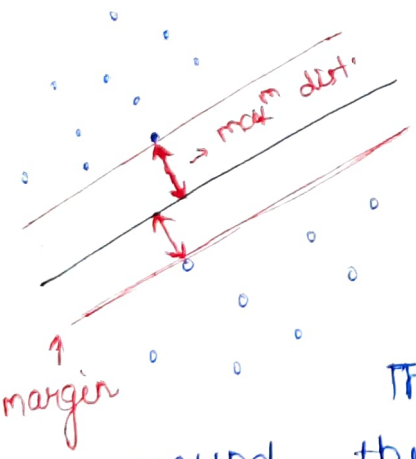


But there can be several decision boundaries that can divide the data points without any errors.



All decision boundaries classify the datasets correctly. But how do we pick the best decision boundary?

The best decision boundary is the one that has a max^m distance from nearest pnts of these 2 classes.



Also, remember that the nearest pnts from the optimal decision boundary that maximize the dist. are kya Support Vectors.

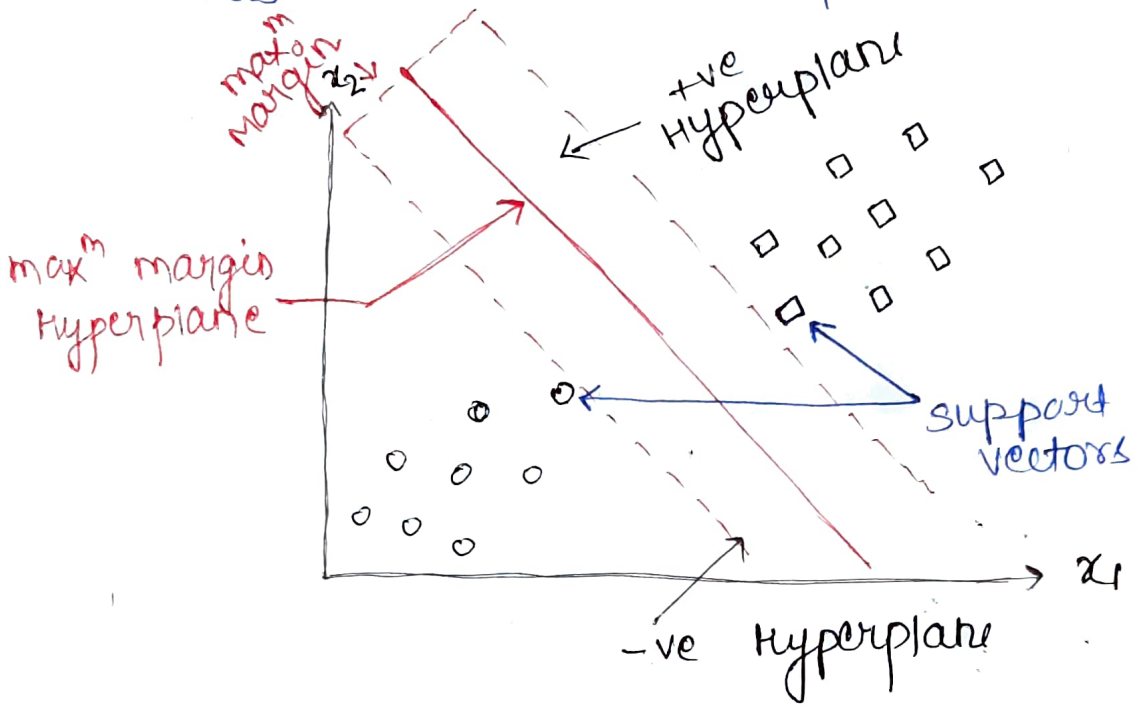
The region that the closest pnts dying around the decision boundary is known as Margin. That's why the decision boundary of a SVM model is known as the max^m margin classifier or the max^m margin Hyperplane.

How a SVM algo. model works:

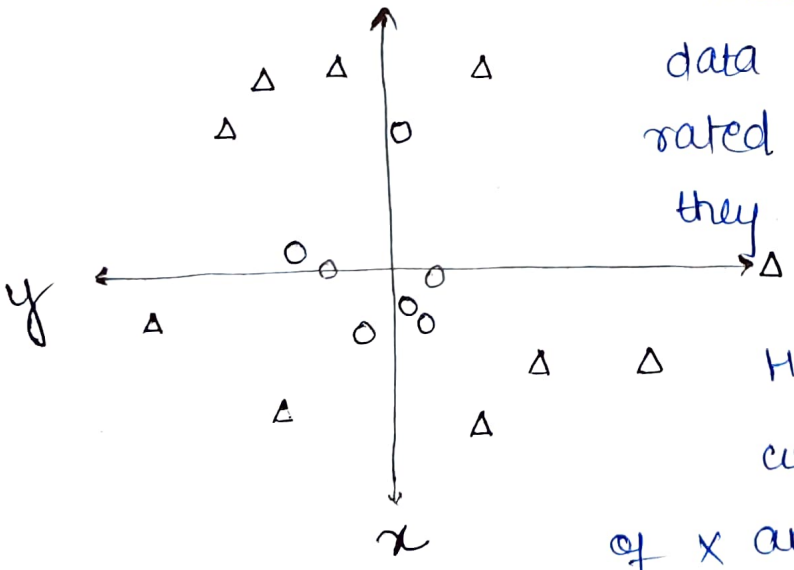
* First, it finds lines or boundaries that correctly

classify the training dataset.

* then, from those lines or boundaries, it picks one that has the max^m dist. from the closest data pnts.

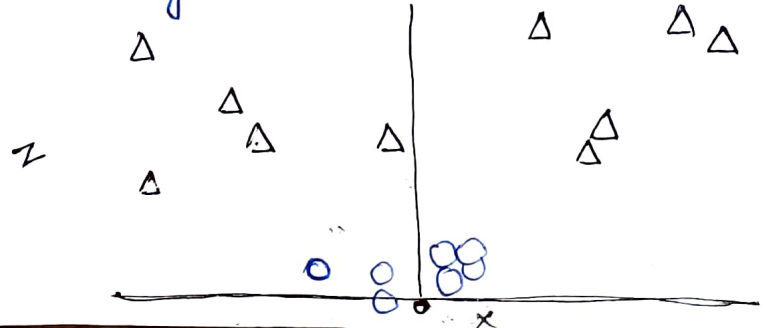


Example: (Non-linear SVM)

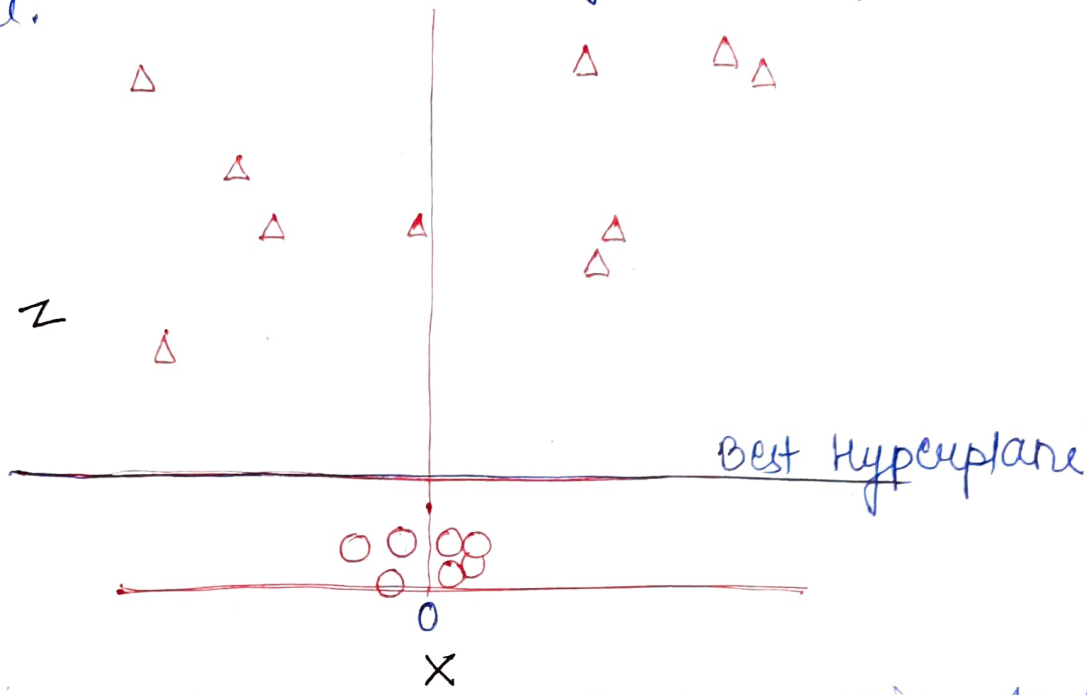


These are two classes of data pnts which can't be separated by a straight line. But they can be separated by a circular Hyperplane, Hence, we can introduce a co-ordinate z with the help of x and y where $z = x^2 + y^2$

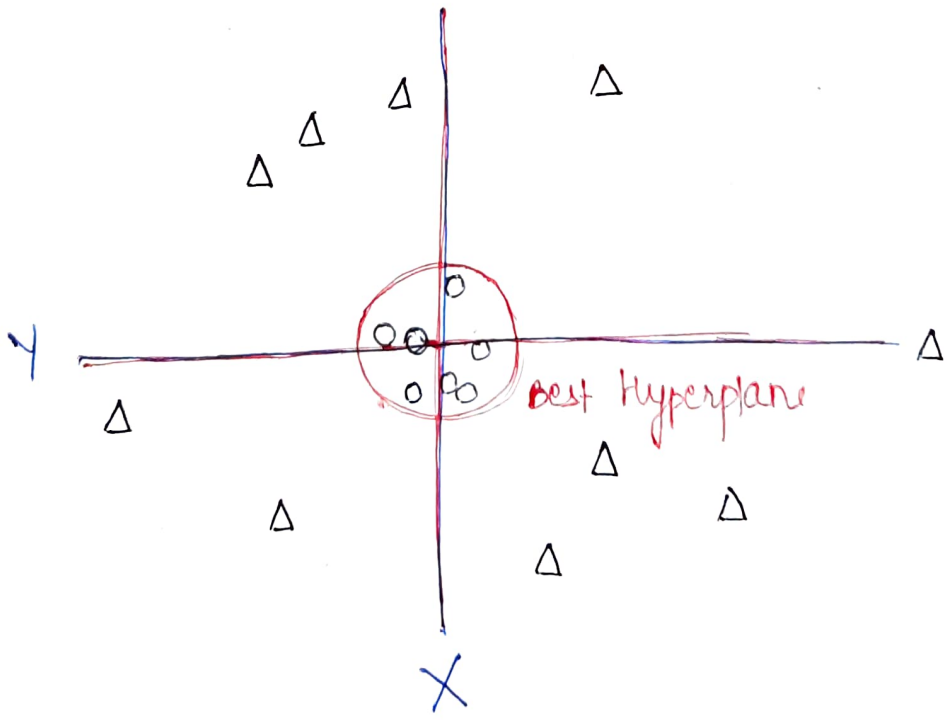
Now, after introducing 3rd dimension, the graph changes to:



Now, the above depiction of data pnts is linearly separable & can be separated by a straight-line Hyperplane.



This representat" is in 3-D with a z-axis. In 2-D the graph looks like:



This is what, a non-linear SVM does! It hypothetically takes the data pnts to a higher dimension, so that they are linearly separable in that dimension & then the algo. classifies them!

Advtgs. of SVM: * has high level of accuracy
* works well with limited datasets
* Kernel SVM contains a non-linear transformat° f° to convert the complicated non-linearly separable data into linearly separable data.
* effective on datasets that have multiple features
* effective when no. of features are greater than no. of data points.

Disadvtgs:
* doesn't work well with larger datasets
* sometimes, training time can be high
* if no. of features is significantly greater than no. of data pnts, it's crucial to avoid overfitting when choosing kernel f°cs & regularizat° terms
* Best works on small sample sets due to its high training time.

REINFORCEMENT LEARNING

Terminology used in RL:

* Agent → it's an assumed entity which performs action in an environ. to gain some reward.
* Environment (e) → a scenario that an agent has to face
* Reward (R) → an immediate return given to an agent when he/she performs specific action or task.
* state (s) → refers to the current situation returned by the environ.

* policy (π) → it's a strategy which applies by the agent to decide the next action based on current state.
* value (V) → it's expected long term return with discount, as compared to the short term reward.
* value function → it specifies the value of a state that is the total amt of reward. It's an agent which should be expected beginning from that state.
* Model of the environ. → this mimics the behavior of the environ. It helps you to make inferences to be made & also determine how the environ. will behave.
* Model based methods → method for solving RL probs. which use model based methods.
* Q-value or Action value (Q) → value is quite similar to value. The only diff b/w two is that it takes an add'al parameter as a current action.

Characteristics of RL

* No supervisor, only a real no. or reward signal
* sequential decision making
* FB is always delayed, not instantaneous
* Time plays a crucial role in RL
* Agent's actions determine the subsequent data it receives

Types:- * +ve RL
* -ve RL

Positive: defined as an event, that occurs bcoz of specific behavior. It ↑es the strength & the frequency of the behavior & impacts +vely on the action taken by the agent.
This type of Reinforcement helps you to maximize performance & sustain change for a more extended period.

Negative: defined as strengthing of behavior that occurs bcoz of -ve cond^n which should have stopped or avoided. It helps you to define the min^m stand of performance.

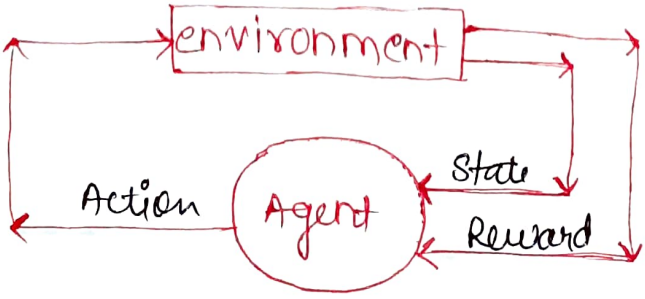
Learning Modls of Reinforcement

1) Markov Decision process 2) Q-Learning

Markov Decision process: following parameters are used to get a solution:

* set of actions (A) * set of states (S)
* Reward (R) * policy (π) * Value (V)

The mathematical approach for mapping a sol^n in Reinforcement learning is recon as a Markov Decision process (MDP)



Q- Learning: is a value based method of supplying info. which action an agent should take.

Why use RL?

* it helps you to find which situation needs an action
* helps you to discover which action yields the highest reward over the longest period.
* also provides the learning agent with a reward f
* also allows it to figure out the best method for obtaining large rewards.

Challenges:

* parameters may affect the speed of learning
* feature / reward design which should be very involved.
* Realistic environ. can have partial observability
* Too much reinforcement may lead to an overload of states which can diminish the results.
* Realistic environ. can be non-stationary.

ADAPTIVE LEARNING

Adaptive ML builds on traditional ML to create a more advanced sol^{n} to real time environ(s). with variable data.

AML can adapt to rapidly changing data sets, making it more applicable to real-world situations.

Adaptive ML is more robust & efficient than tra- ditional ML & incorporates agility, ↑ed accuracy & greater sustainability.

AML can process large quantities of data while its op'al cond'cs) can be more easily adjusted as the need of the company using it changes.

It can quickly adapt to new info. & provide real time insight into how that data can be used.

It uses single channel structure, which means it collects new info. in real time. Its also able to use diff. techniques for gathering data as well as diff. ways of grouping & analyzing that data.

It can change acc. to new data & provide rapid insights.

Benefits of Adaptive ML:

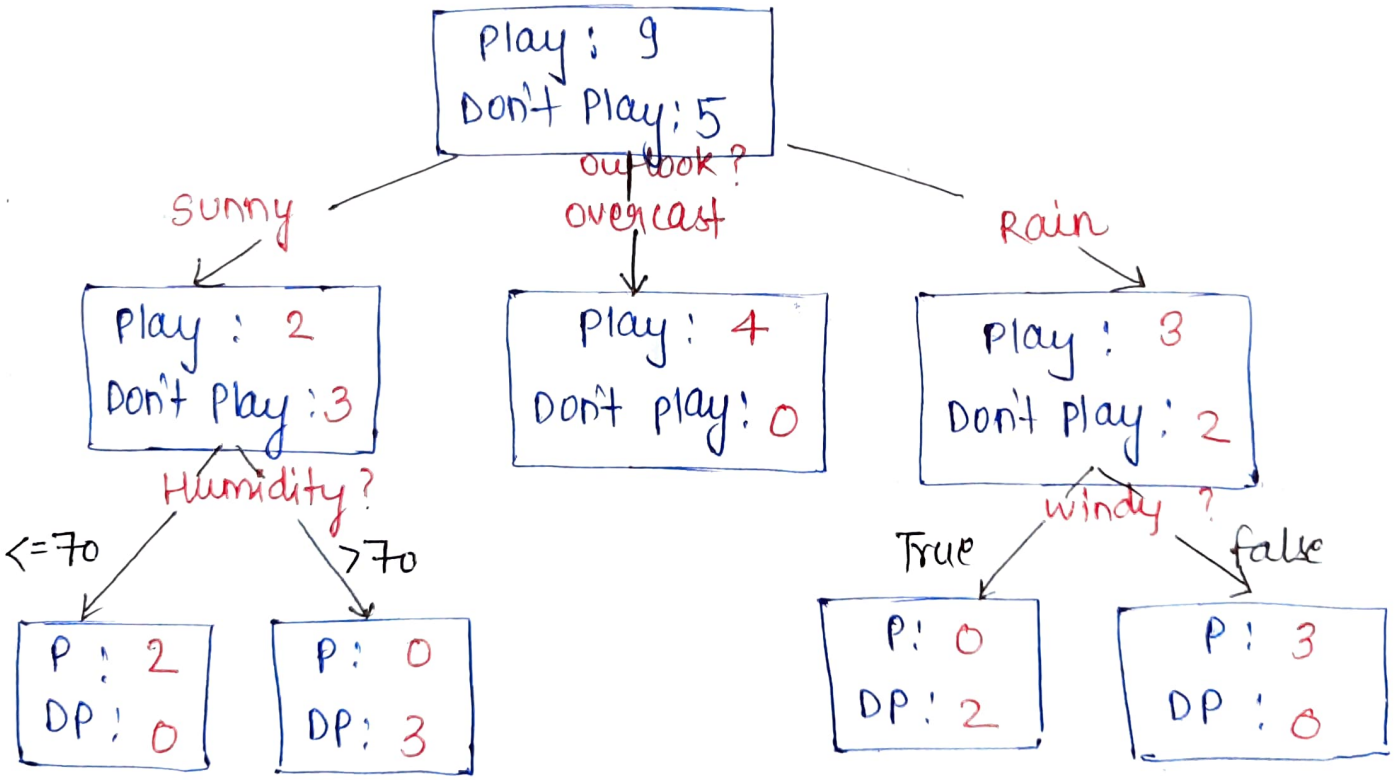
1. More efficient → use to find relevant sol'cs) faster. Since, it works on a single channel, it can creat faster sol'cs with newest data at hand.
2. More pertinent data → Rather than using old, static data, Adaptive ML is constantly taking in new, more relevant data. It can change its behavior based on this new data.
3. Able to learn from the past → the longer an Adaptive ML prog. runs, the more it learns. It can lower the chances of repeating a mistake by

remembering that it happened & adjusting itself accordingly. The more info. fed to an adaptive ML, the smarter & more accurate it becomes.

ENSEMBLE LEARNING

Ensemble Methods is a ML techniques that combines several base models in order to produce one optimal predictive model.

ex: Dependent variable : PLAY



A Decision Tree determines the predictive value based on series of questions & cond'ns. For instance, this simple Decision Tree determining on whether an individual should play outside or not. This tree takes several weather factors into A/c, & given each factor

either makes a decision or asks another question. In this example, every time its overcast, we will play outside. However, if it is raining, we must ask if it is windy or not? If windy, we will not play.

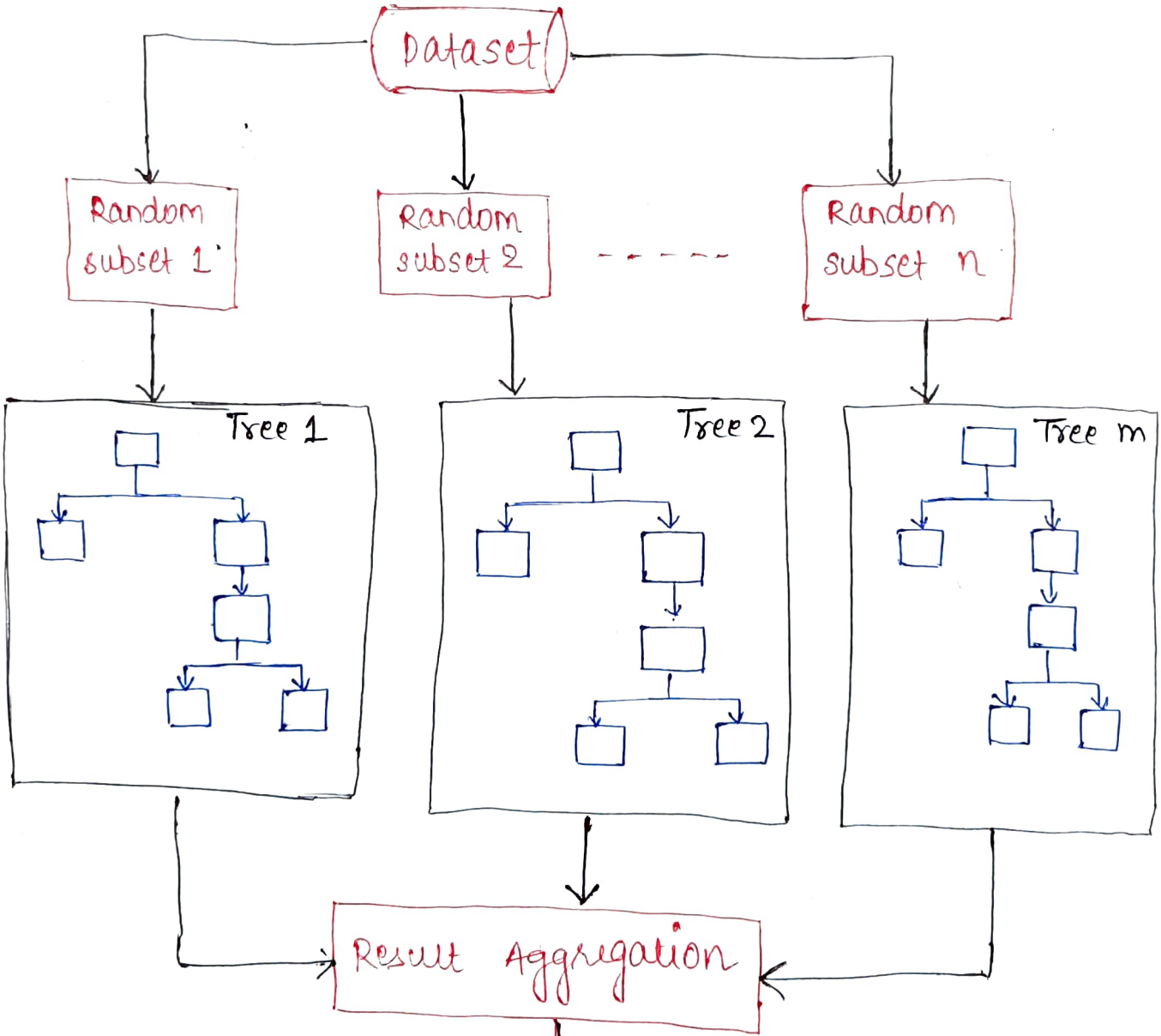
When making Decision Trees, there are several factors we must take into consideration: on what features do we make our decisions on? what is the threshold for classifying each ques. into a yes or no answer? what if we wanted to ask ourselves if we had friends we will play every time. If we have friends, we'll play every time. If not, we might continue to ask ourselves qness. about the weather. By adding an addl ques., we hope to greater define the yes and No classes.

This is where Ensemble Methods come in handy! Rather than just relying on one Decision Tree & hoping we made the right decision at each split, Ensemble Methods allow us to take a sample of Decision Trees into A/c, calculate which features to use or ques. to ask at each split & make a final predictor based on the aggregated results of the sampled Decision Trees.

Types of Ensemble Methods:

1. Bootstrap AGgregating (BAGGing) → it gets its name bcoz it combines Bootstrapping & Aggregation to form

one ensemble model. Given a sample of data, multiple bootstrapped subsamples are pulled. A Decision Tree is formed on each of the bootstrapped subsamples. After each subsample Decision Tree has been formed, an algo. is used to aggregate over the Decision Trees to form the most efficient predictor.



Given a dataset, bootstrapped subsamples are pulled. A Decision Tree is formed on each bootstrapped sample. The results of each tree are aggregated to yield the strongest, most accurate predictor.

2. Random Forest Model → It can be thought of as BAGGing, with a slight tweak. When deciding where to split & how to make decisions, BAGGed Decision Tree have the full disposal of features to choose from. Therefore, although the bootstrapped samples may be slightly diff., the data is largely going to break off at the same features throughout each model.

In contrary, RF model decide where to split based on a random selection of features. Rather than splitting at similar features at each node throughout, RF models implement a level of differentiation beoz each tree will split based on diff features. This level of differentiation provides a greater ensemble to aggregate over, ergo producing a more accurate predictor.

Similar to BAGGing, Bootstrapped subsampled are pulled from a larger dataset. A Decision Tree is formed on each subsample. However, the " " is split on diff. features.