AI Unit 4

(*). Planning:

→ Required to convert objectives into Actions.

→ decision making task performed by robot or computer to achieve a goal.

→ task of coming up with a sequence of actions that will help to achieve goal.

Ex: goal: Clear the exams with distinction.

Planning: How many days are left for exams?, No. of chapters to study, Are notes available, etc.

→ optimizing overall performance.

→ Why is it required? Automation is an emerging trend that requires automated planning.

→ other applications of planning: Robots, Autonomous Systems, Self-Driven Cars, etc.

(*). Advantages of Planning:

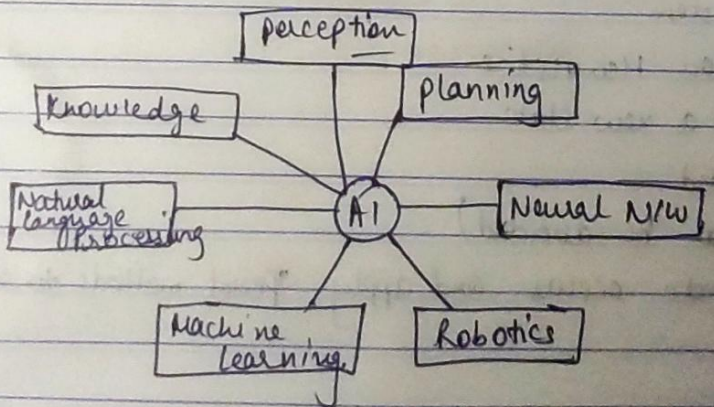
→ short time to solution.

→ It makes objectives more clear & specific

→ It minimizes uncertainities.

→ facilitates co-ordination

→ encourages innovations.



(*). Types of planning:

Hierarchical	Non Hierarchical	Reactive.
Problem is divided into sub problems. Solution of these sub-problems is found one by one. Eg. In Robots.	Problem is not divided into sub-problems. There is a single plan for whole problem. Ex. games.	Needed where the environment keeps changing. to be able to respond quickly with best results. Eg. when robot needs to avoid obstacles.

(*). Planning Problems:

→ Problems that arise when AI system is trying to plan its next move.

→ Might be difficult to solve b/c AI system needs to check all possible outcomes and come out with the best one.

Planning problem is defined with:

⇒ Domain Model: Defines the Actions along with objects.

⇒ Initial State: State where any action is yet to take place.

⇒ Goal State: State which the plan intends to achieve.

(*). Components of Planning System:

→ Choose the best rules based on Heuristics.

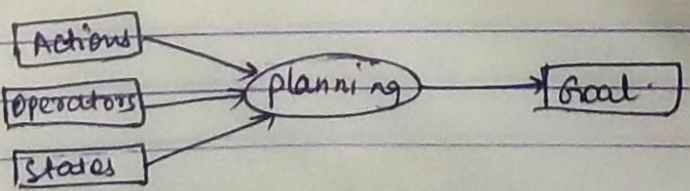
→ Apply these rules to create a new state.

→ Detect when a solution is found.

→ Detect dead ends. (so they can be avoided)

→ Detect when a nearly solved state occurs and apply special methods to solve it

(*). Planning Representation.



(*) Planning Agent = Problem Solving Agent + Knowledge Based Agent

⇓ To consider the consequences of Actions before Acting.

⇓ for selection of Actions that will lead us to goal state.

(*). Planning languages:

(a). STRIPS: Standard Research Institute Problem Solver.

→ It can describe the world (Initial state, Goal State) using objectives, actions, preconditions and effects.

→ It is composed of (i) State (ii). Goal (iii) Actions.

→ Does not support -ve literals.

(b). ADL: Action Description language. Extension of STRIPS.

→ Overcomes the limitations of STRIPS.

→ More expensive than STRIPS.

→ Supports -ve literals.

(c). PDDL: (Planning Domain Definition language).

→ It contains STRIPS, ADL and other languages.

→ Superset of STRIPS & ADL.

→ Component: (i) Objects (ii). Predicates (iii). Initial State (iv). Goal Specification (v). Actions/Operators.

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^{s}) 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

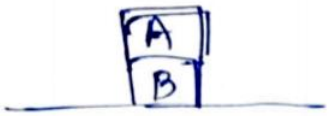
* PICKUP (X): picking up Block X which is on top of the table.

* 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 op^{r} no. 3 so,

Step 2: *Stack (A, B)

Step 3: Holding (A) ^ Clear (B) - precond^{n}

Explanation → In step 3, Holding (A) is not true, when we see initial state. It's expand by op^{r} 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'n 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.

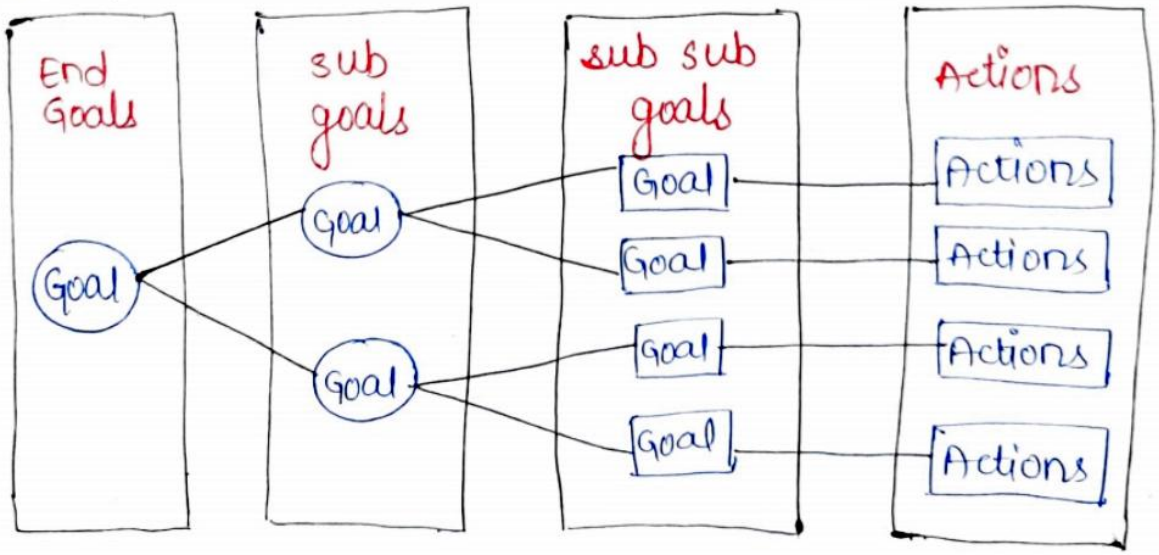
3) Target goal is split into sub-goals, that are further split into other smaller goals.

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

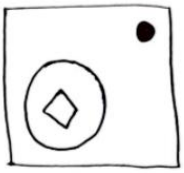
5) In this, all the sub-goals are linked with corresponding executable actions (ops)

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

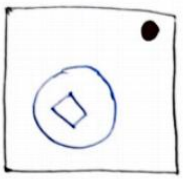
7) It involves tracking all the changes made to the actual state. Changes are made until the target state is achieved.



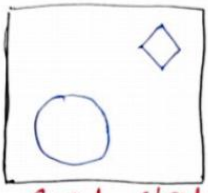
ex. Initial state :



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



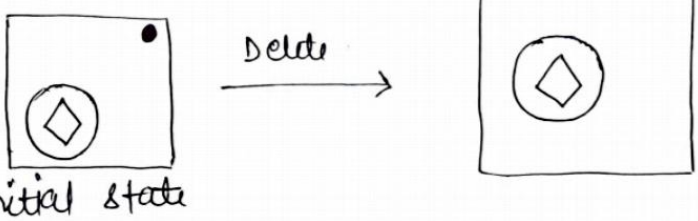
Goal state

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.

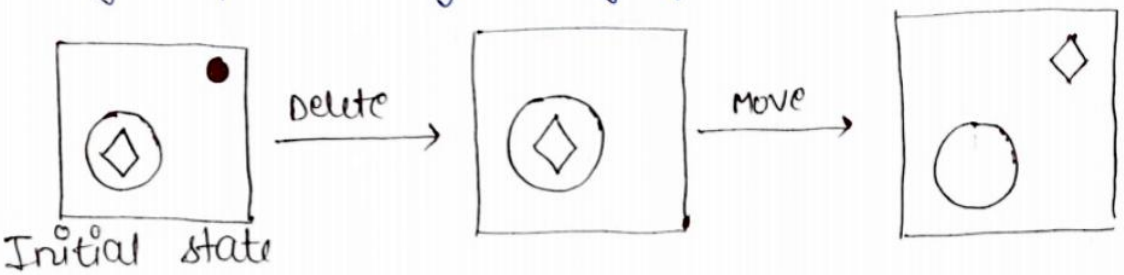
The goal can be divided into sub-goals that are linked with executable actions or op^rcs).

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

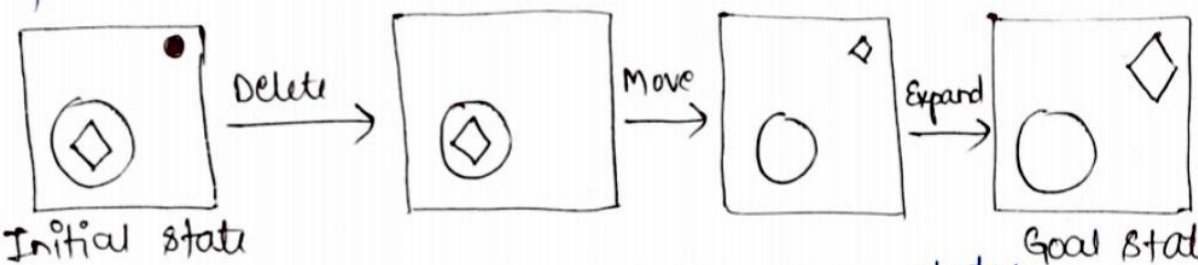
1. Delete operator - dot symbol at the top right corner in the initial state doesn't exist in goal state. The dot symbol can be removed by applying the delete operator.



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

Applications of MEA

* Organizational planning → MEA used in org^n^s to facilitate general mgmt. It helps org^n^al managers to conduct planning to achieve the objectives of the org^n^. 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 desired 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 desired tech. In org^n^s. various op^n^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 modls & whenever it receives

(*). Machine Learning:

→ Enables the machine to learn from past data.
→ Take decisions using historical data without being explicitly programmed
→ Builds models.
→ Whenever it receives new data, it ~~builds model~~ predicts o/p for it.

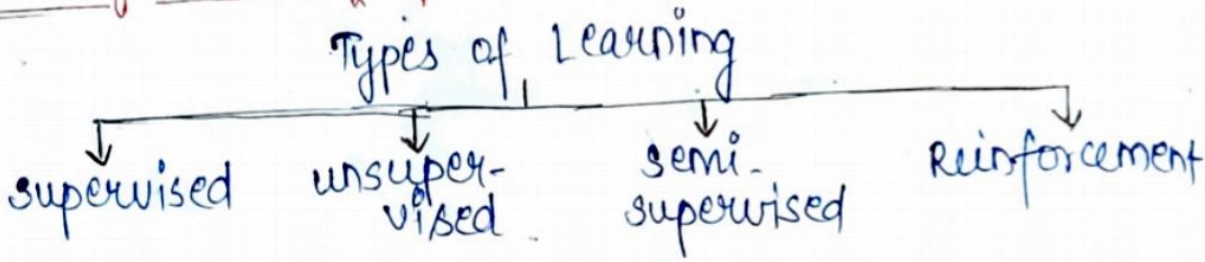
Goal of ML: → To develop general purpose Algorithms.
→ Enhance services
→ Predict Journey times, Behavior of Market, How long jobs will take, etc.

Applications: → Image Recognition
→ Speech "
→ Traffic Prediction
→ Self driven Cars
→ Product Recommendation
→ Email spam & Malware filtering.
→ Medical Diagnosis
→ Online fraud Detection.
→ Stock market

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.

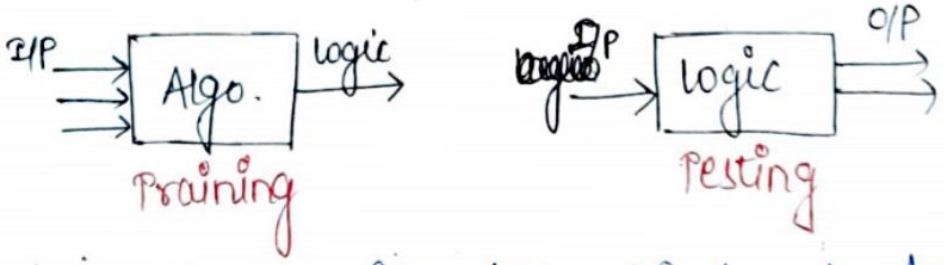
(4) Reinforcement Learning

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 ckrs 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.

(*) Classification of Unsupervised Learning:

Clustering: (a). Hierarchical Clustering

(b) k-means clustering: Taking centroid of points and making sure that the new point has the least distance from centroid.

(c). KNN Clustering: Also called lazy learner. Works on small datasets.

Association: (a). Apriori Algo: Maps the dependency of one data item with Another. Ex. People always tend to buy butter along with bread.

(b). FP growth Algorithm: easy to find frequent data sets. It uses divide and conquer strategy.

A frequent Pattern Tree is generated with itemsets.

The item that is bought more frequently is the Root node of FP tree.

⇒ Support: $\frac{\text{How many times items are brought together} \times 100}{\text{Total Transactions}}$

⇒ Confidence: $\frac{\text{No. of times X and Y are brought together} \times 100}{\text{No of times X is brought}}$

(*). Artificial NN:

→ Derived from Biological NN.

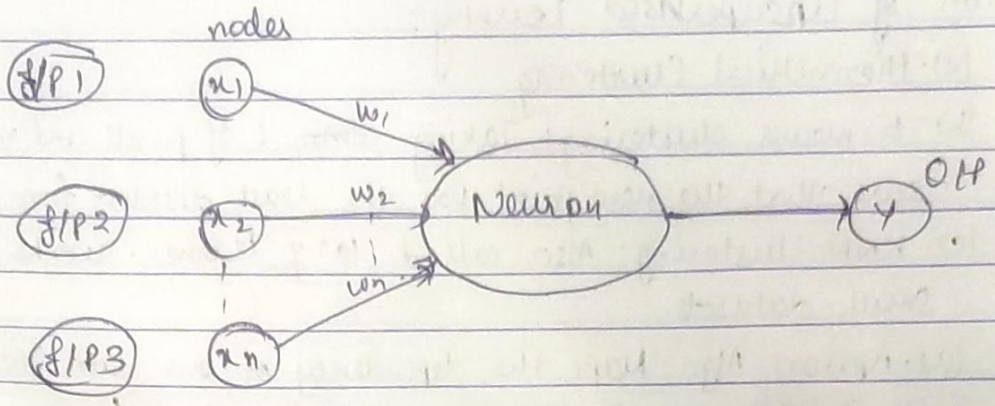
→ Human Brain contains billions of cells called Neurons.

→ Each Neuron has cell body that is responsible for carrying info towards (I/P) and away (O/P) from brain.

→ As the human brain can learn from past data & provide responses, similarly ANN is able to learn from past data and provide responses in the form of predictions / classifications.

→ NON LINER statistical Models.

→ Also capable of taking the sample data rather than the entire dataset to provide O/P result.

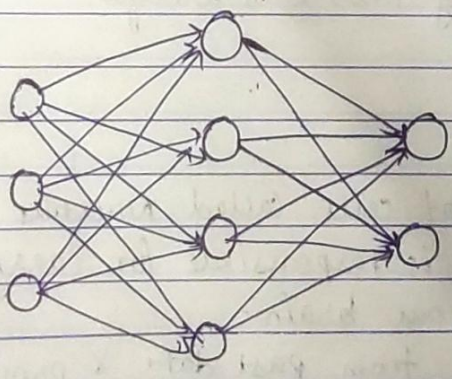


(*) ANN Architecture: 3 layers

(a) Input layer: → 1st layer.
→ Receives I/P in the form of text, images, etc.

(b) Hidden layer: → Middle layer
→ There can be a single Hidden layer or Multiple Hidden layers.
→ These layers perform various mathematical operations on I/P data & recognize the patterns.

(c) Output layer: → To obtain the result that is obtained from middle layer.



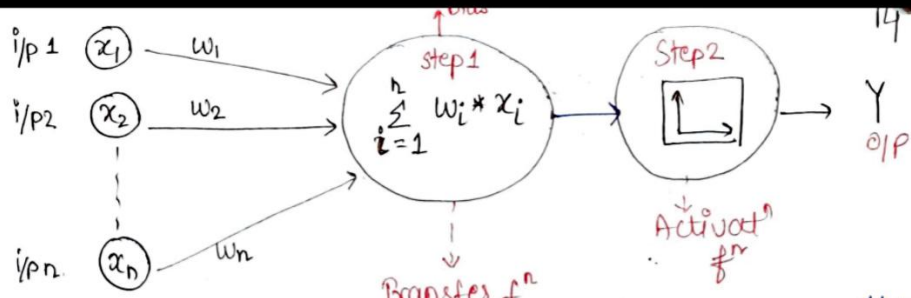
Input Hidden Output

→ There are multiple parameters in NN that affect the performance. The O/P is mostly dependent on these parameters.
→ The NN is made up of many perceptrons.

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.

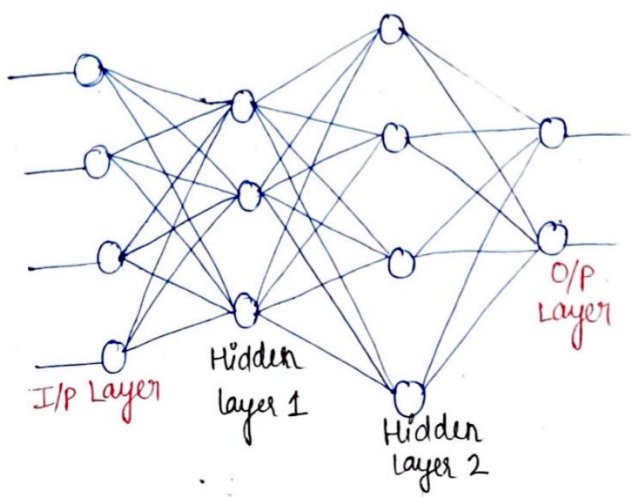
Each node in the n/w has some weights assigned to it. A Transfer f^n is used for calculating the weighted sum of the i/ps & the bias.



After 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

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

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 fn 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:

   Error = Actual O/P − 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 ?

* Its fast, simple & easy to program
* It has No parameters to tune apart from the nos. of i/p
* Its 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

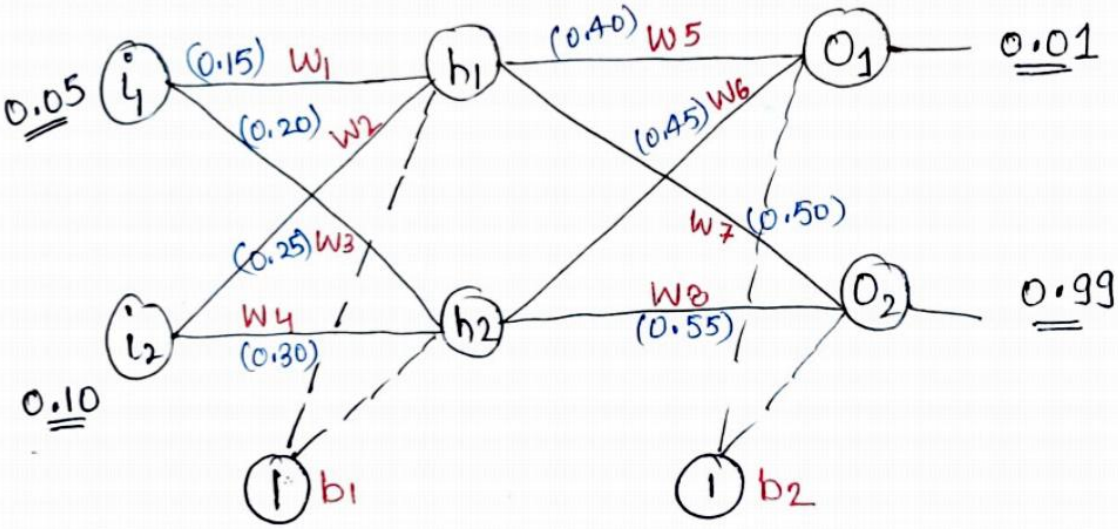
O/p layer.

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 :

$net_{h_1} = w_1 i_1 + w_2 i_2 + b_1 * 1$ $\quad$ $net_{h_2} = 0.3925$

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

then squash it using the logistic f^n to get the 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 :

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

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

$net_{o_1} = 1.105905$ $\quad$ $net_{o_2} = 1.224922091$

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

$Out_{o_2} = 0.772928465$

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

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

$E_{o_2} = 0.023560026.$

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

(**) Support Vector Machine:

→ Powerful ML algorithm (supervised) that can be used to build both classification & regression models.

→ Can perform well with both linearly sep & non-linearly separable datasets.

Types:

(a). Linear / Simple SVM: Used for linearly separable data.

→ Linearly separable data: when data is classified into two sets with a straight line. Classifier is known as linear SVM.

→ Used for linear Regression & Classification problems.

(b). Non Linear or Kernel SVM: Opp. of linear.

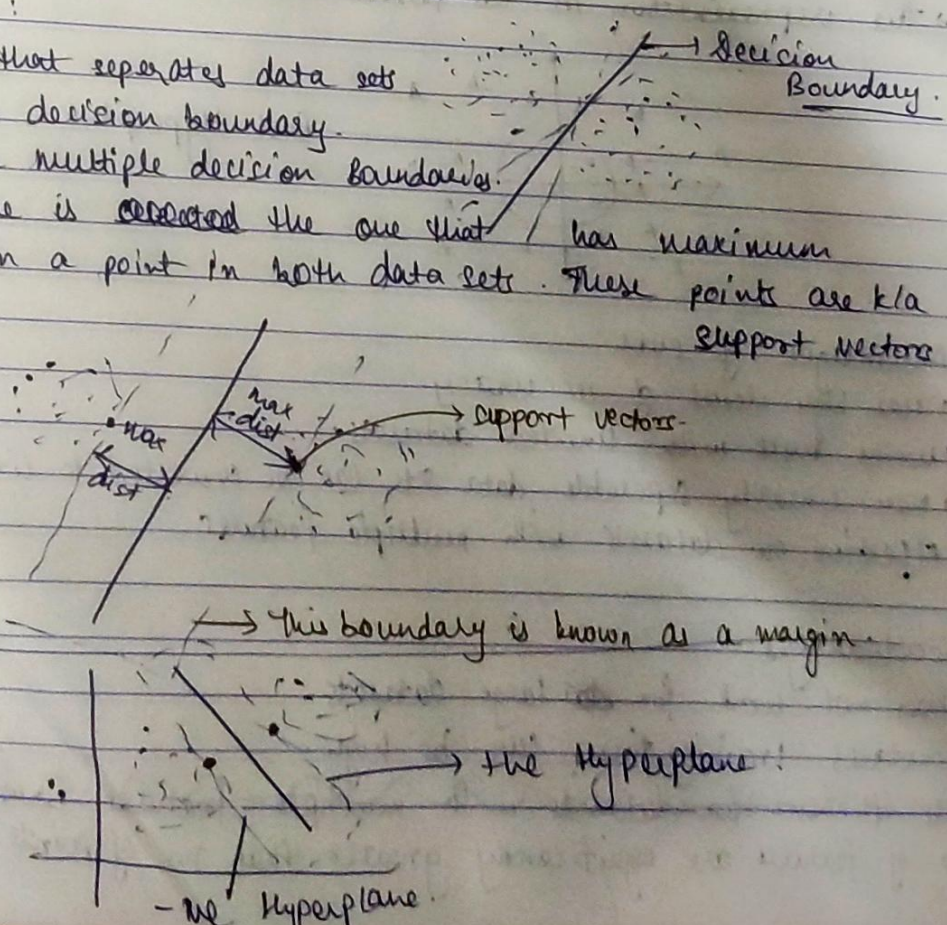
→ It has more flexibility for non-linear data b/c more features can be added to fit a hyperplane instead of 2D space.

Linear SVM:

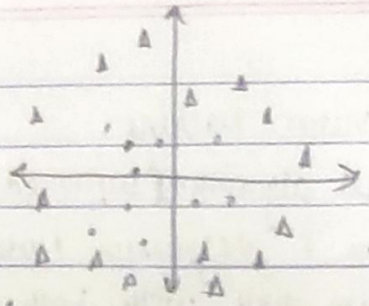
→ The plane that seperates data sets is known as decision boundary.

→ There can be multiple decision boundaries.

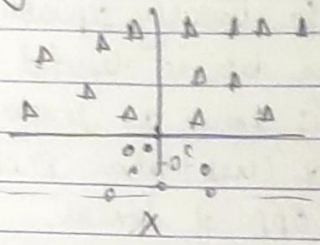
→ The best one is the one that has maximum distance from a point in both data sets. These points are k/a support vectors.



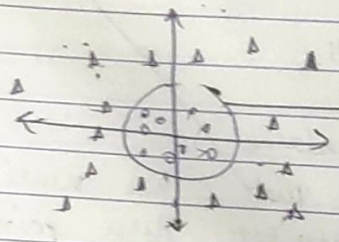
4). Non linear ~~Hyperplane~~ SVM:



→ Two classes of data points that cannot be seperated by a line.
→ These can be seperated by circular Hyperplane, ∴ we introduce a coordinate z, given by $z = x^2 + y^2$.
→ After introducing z coordinate, the graph changes to.



→ The above data is linearly seperable and can be seperated by a straight line.
→ The representation in 2D further looks like:



Advantages of SVM:
→ Has high level of accuracy.
→ Works well with limited datasets.
→ Non-linearly seperable data sets can be converted to linearly seperable
→ Effective on datasets with multiple features.

Disadvantages.
→ Does not work for too large datasets.
→ Sometimes, training time can be high.
→ ~~Less efficient for datasets with multiple features~~. Crucial when no. of features are significantly greater than no of points.

(*). Reinforcement Learning:

→ We have an agent and a reward with many hurdles in between.
→ The agent is supposed to find best possible path to reach reward.
for eg, there is a robot as agent, hurdles is fire and reward is diamond.
→ The robot needs to find the best path that will give him the reward with least hurdles.
→ Each right step will give it a reward and each wrong step will subtract the reward of the robot.
→ The total reward will be calculated when it reaches final diamond.

Main points in Reinforcement learning:

(i). Input → Initial state.
(ii) Output → There are many possible outputs.
(iii) Training: Based on input. The model will return a state where the user will decide whether to reward or punish the model based on O/P.
→ Best soln. is decided based on maximum reward.
→ helps to find which situation needs action.
→ helps to discover which action yields highest rewards.

Challenges:

→ Parameters may affect the speed of learning.

(**). Multi Agent Based learning:

→ A single agent cannot handle learning in case of complex situations.
→ A team or a group of agents can overcome this limitation.
→ They can work in coordination to achieve a goal.
→ Two types: one where the agent tries to maximize its utility. Ex. gaming.
→ Second where the agent work in collaboration to achieve a common goal. Ex → Building a product.

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: dyined as strengthing of behavior that occu bcoz of -ve cond^n which should have stopped or avoided. It helps you to dyine the mini^m stand of per- formance.

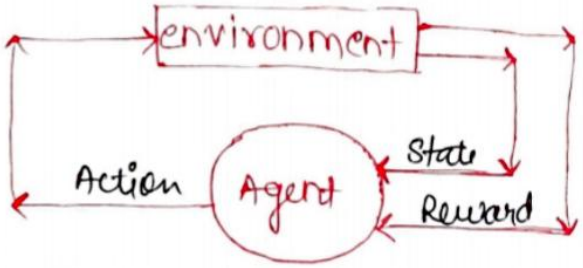
Learning Modls of Reinforcement

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 (n) * Value (V)

The mathematical approach for mapping a sel^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.

(*). Adaptive learning:

→ More advanced.

→ Can adapt to rapidly changing data sets, making it more applicable for real world data situations.

→ More robust and efficient.

→ ↑ accuracy & greater sustainability.

→ Also able to use different techniques for gathering data as well as grouping.

→ Can change acc. to new data and provide rapid insights.

Benefits:

→ Efficient & faster.

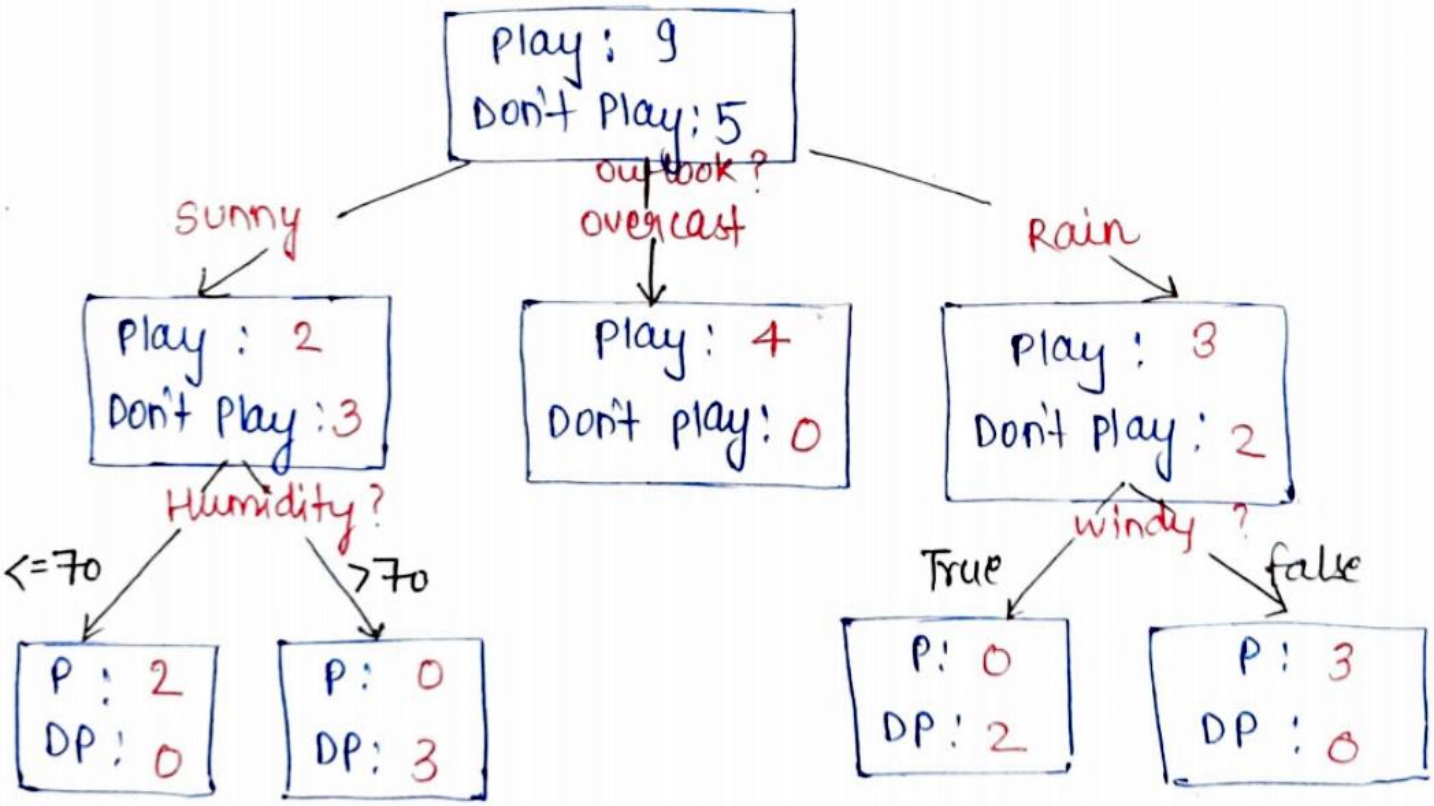
→ Rather than using old data, it takes new data.

→ Learns from past. Less chances of repeating mistakes.

ENSEMBLE LEARNING

Ensemble Methods is a ML techniques that combine several base moduls in order to produce one optimal predictive modul.

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/Cs & 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.

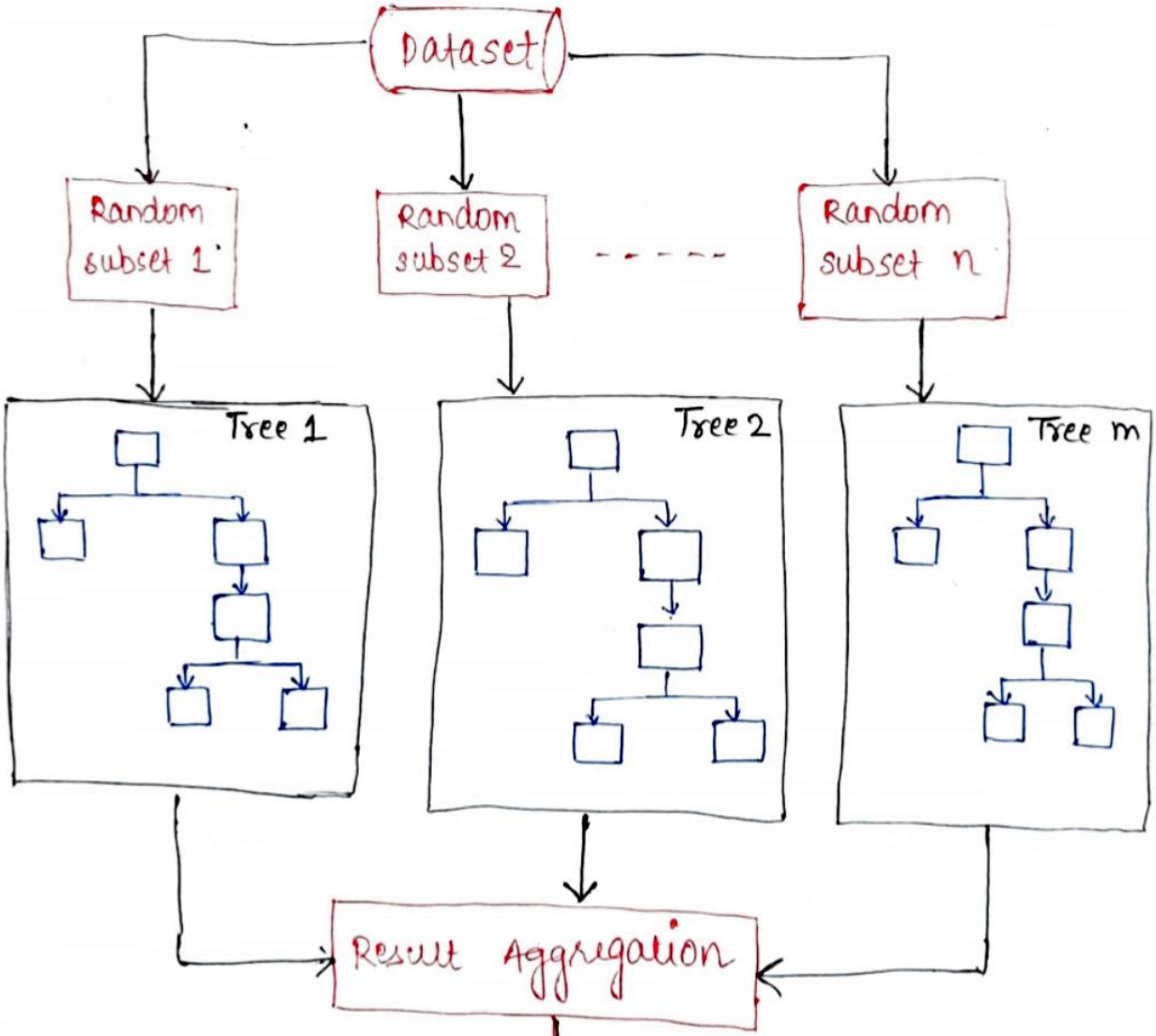
When making Decision Trees, there are several factors we must take into consideration: on what features do we make our decisions on? what is the threshold for classifying each ques. into a yes or no answer? what if we wanted to ask ourselves if we had friend we will play every time. If we have friends, we'll play every time. If not, we might continue to ask ourselves quesns. about the weather. By adding an add'al 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 Desion Tree is formed on each bootstrapped sample. The results of each tree are aggregated to yield the strongest, most accurate predictor.

2. Random Forest Modul → 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 modul.

In contrary, RF modul decide where to split based on a random selection of features. Rather than splitting at similar features at each node throughout, RF moduls 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 predicter.

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.

10.11 LEARNING FOR DECISION-MAKING

What is observed in different learning mechanisms is that with the learnt concepts, the capability to take decisions in increased. Speaking about the supervised or unsupervised methodologies, the decisions taken are not sequential in nature. That is, if the system makes a mistake on one decision, this has no bearing on the subsequent decisions. To cope up with this dynamic situation, there is a need to understand the perspective of decision-making. Another aspect is environment and system learning, which also needs to be looked upon during decision-making. Hence while taking decisions, one specific learning approach may not be suitable. The learning approach is dependent on decision scenario.

10.12 SPEEDUP LEARNING

Speedup learning typically deals with speeding up problem solving by effective use of problem solving experience. Hence, prior problem solving experience is an input for speedup learning.

In this learning,

1. There is no interaction with the environment.
2. New problems cannot be solved.

So, speedup learning accelerates the process based on the previous experiences and prior observations.

The process of speedup learning is depicted in Figure 10.18.

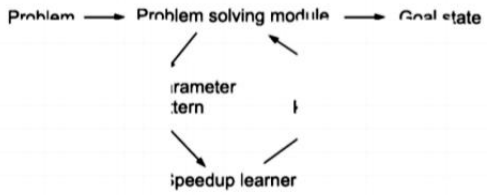


Figure 10.18 Speedup learning modules.

Another dimension to the speedup learning is generalised caching. This is also known as *explanation-based learning*. There are a number of issues with explanation-based learning, as it is implemented and embedded in system to solve real-life problem and is suitable for a particular set of problems, where the sequential processes once developed can be used again and again. But this is not the case in many real-life problems, where dynamic change in environment demands the improvement in the established scenarios and even there is a need to keep on learning based on the new findings.