mood-book

Unit – 1

Introduction to Machine Learning

Def:-

* Machine Learning is a growing technology which enables Computers to learn automatically from past data. Machine learning uses various algorithms for building mathematical models and making predictions using historical data (or) information.

Applications of Machine learning:-

* Image recognition
* Speech recognition
* Email filtering (Spam detector)
* Virtual personal Assistants
* Online customer Support
* Online fraud detection.
* Stock market trading
* Traffic prediction.
* Self driving cars.
* Medical diagnosis etc.

What is machine Learning ?

* The term machine learning was first introduced by Arthur Samuel in 1959.
* It enables a machine to automatically learn from data improve performance from experience, and predict

things without being explicitly programmed.

* **Training data :-** past / historical data which is used to train the program (or) algorithm.

Features of ML :-

* ML uses data to detect various patterns in a given dataset.
* It can learn from past data and improve automatically
* It's a data-driven technology.
* Machinelearning is much similar to data mining as it also deals with the huge amount of the data.

Wellposed Learning problem :-

An agent solves a problem (or) task (T), performance (P) and gain some experience (E).

If P is measured at T it can improves E ..

------ Learning by Experience ------

Examples :-

1. playing checkers problem
2. Handwritten recognition problem
3. Robo driving learning problem.

problem	Task (T)	performance (P)	Experience (E)
1. playing checker learning	playing against opponents to win game	Make perfect moves to win game	It plays itself to improve
2. Handwriting recognition learning	classifying the images & text	Better classification	A database of homework text
3. Robot driving learning problem	Drive the car in a 4 lane highway.	Source to dest. the avg distance travelled (long & safe)	Images, vehicles on Road.

Designing a learning System :-

* To get a Successful learning System, it should be designed
* For a proper design, Several steps should be followed.
* These steps can create perfect & efficient System.

5 * Important steps :-

1. Choosing the training experience.
2. Choosing the target function
3. choosing a representation for target function

4. Choosing a learning algorithm for approximating the target function.
5. Final design.

* By using those 4 steps we will get a final design.

*1 Choosing the training Experience :-

Below are the attributes which will impact on Success and failure of data.

*1st attribute :-

The training Experience will be able to provide "direct (or) indirect feedback" regarding choices.

Ex:- while playing chess the training data will provide feedback to itself like instead of this move if this is chosen the chances of Success increases.

*2nd attribute :-

It is "the degree" to which the learner will control the sequence of training Examples.

Ex:- when training data is fed to the machine then at that time accuracy is very less but when it gains Experience while playing again and again with itself (or) opponent the machine algorithm will get feedback and control the chess game accordingly.

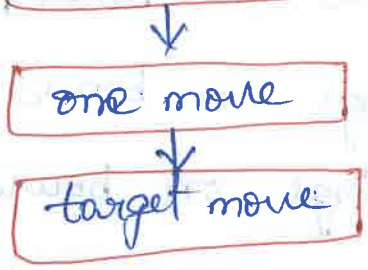
*3rd attribute :- It is how it will represent "the distribution of Examples" over which performance will be measured.

2. Choosing the target function :-

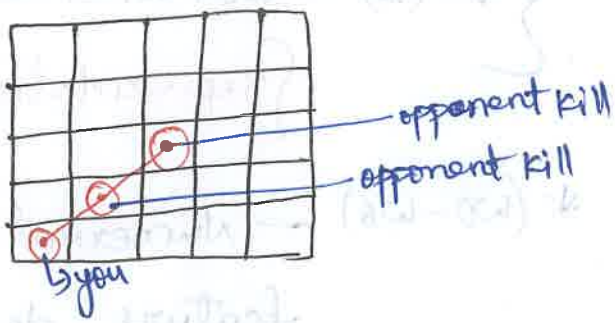
What type of knowledge is learnt and how it is used by the performance System.

Ex:- Checkers game

* While moving diagnolly set of all possible moves is called **leagal moves.**



* Travel only in foward direction
* Only one move per chance
* only in diagnal direction.
* Jump over opponent.
* Target function = V(b)
* Board state = b.
* Leagal moves set = b.



* b is final board state that is won, then V(b) = 100
* b is final board state that is lost, then V(b) = -100
* b is final board state that is drawn, then V(b) = 0
* If b is not final state then V(b) = V(b').

b' → best final state.

3. Choosing A Representation for Target Function :-

For any board state, we calculate function "C" as linear combination of following board features ie., "C(b)"

Features :-

* x_1 — No. of black pieces on board
* x_2 — No. of Red pieces on board
* x_3 — No. of black Kings on board
* x_4 — No. of red Kings on board.
* x_5 — No. of black pieces Threatened by red (black which can be beaten by red)
* x_6 — No. of red pieces Threatened by black.

$$\hat{V}(b) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4 + w_5 x_5 + w_6 x_6$$

{representation of C(b)}

* $(w_0 - w_6)$ — Numerical coefficients (or) weights of each feature determine the importance / weightage of features
* w_0 → additive constant.

4. Choosing a Learning Algorithm for approximating the target function :-

* To learn a target function (f) we need a set of

Traning Examples.

* describes a particular board state and traning value

(b) Vtain(b)

{Ordered pair = (b, Vtrain(b))}

(traning Example representation)

Ex:-

Black won the game

(i.e, X2=0, which means no red

Vtrain(b) = +100

b= (X1=3, X2=0, X3=1, X4=0, X5=0, X6=0)

<(x1=3, x2=0, x3=1, x4=0, x5=0, x6=0)+100>

we need to do 2 steps in this phase.

1. Estimating training values :-

In every step, we consider Successor.

(depending on the next step of opponent).

Vtrain(b) ← V^(Successor(b))

↓

it represents the next board state.

{Estimating that, this move will help/destroy opponent}

V^ → Represents approximation

2. Adjusting the weights :- There are some algorithms

to find weights of linear functions.

Here we are using LMS (least Mean Square)

(Used to minimize the error)

$$\text{Error} = \left(V(\text{train}(b) - V^n(b)\right)^2.$$

If error = 0, No need to change weights

error = +ve, Each weight is increased ↑ in proportion

error = -ve, Each weight is decreased ↓ in proportion

* Final design :- It has 4 different modules.

1. performance System
2. Critic
3. Generalizer
4. Experiment Generator.

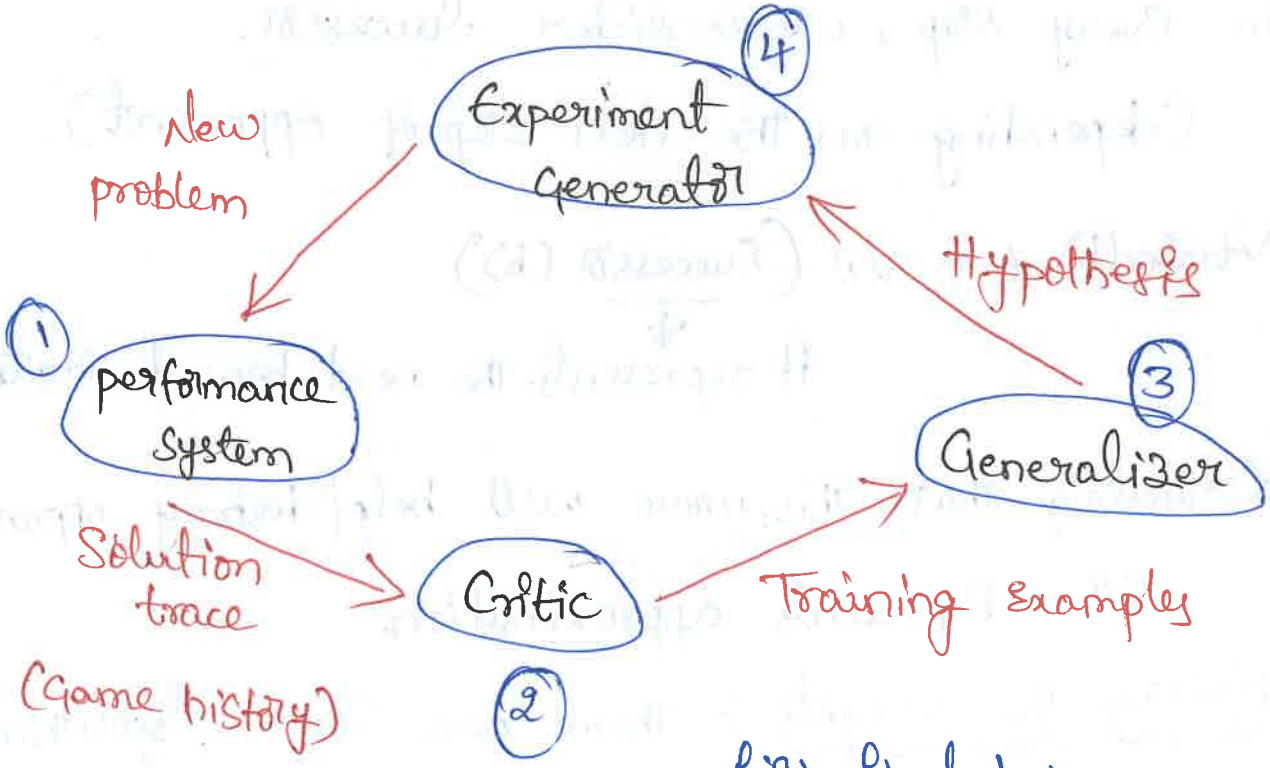


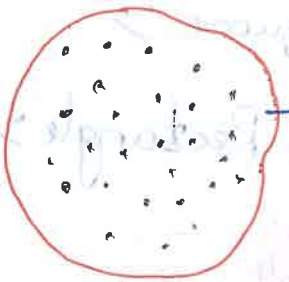
fig: final design

Concept Learning and the general to Specific ordering :-

perspectives of Machine Learning :-

perspective of machine learning involves searching very large space of possible hypotheses to determine one that best fits the observed data and any prior knowledge held by learner.

Ex:-



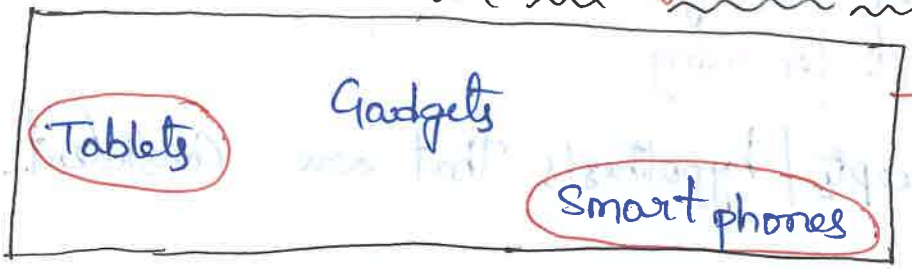
→ sorting algorithms.

Ex:- merge sort

Issues in Machine Learning :-

1. What algorithm should be used?
2. Which algorithms perform best for which types of problems?
3. How much training data is sufficient? and testing data.
4. What kind of methods should be used?
5. What methods should be used to reduce learning overhead?
6. For which type of data which methods should be used?

Concept Learning and the general to Specific ordering :-



→ universe of gadgets.

features (Binary Valued attributes) :-

	Tablet	Smart phone	
Size :-	Large	Small	(x_1)
Color :-	Black	Blue	(x_2)
Screentype :-	Flat	Folded	(x_3)
Shape :-	Square	Rectangle	(x_4)

Concept Learning :- Inferring / Assuming Boolean-Valued function from training example of it's i/p's and o/p's.

Concept = $< x_1, x_2, x_3, x_4 >$

Tablet = < large, Black, Flat, Square >

Smart phone = < Small, Blue, Folded, Rectangle >

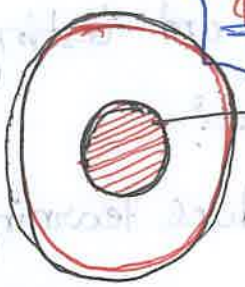
No. of possible instances = 2^d

Ex :- $2^4 = 16$

Total No. of possible Concepts = 2^{2^d} 2^{16}

? - it represents that any value is acceptable for this attribute.

ϕ - " " that No " " " " " "



→ Target Concept / hypothesis Space.

phi (it represents Empty set of instances -ve).

$< \phi, \phi, \phi, \phi >$ → Reject all → (Most Specific hypothesis).

$< ?, ?, ?, ? >$ → Accept all

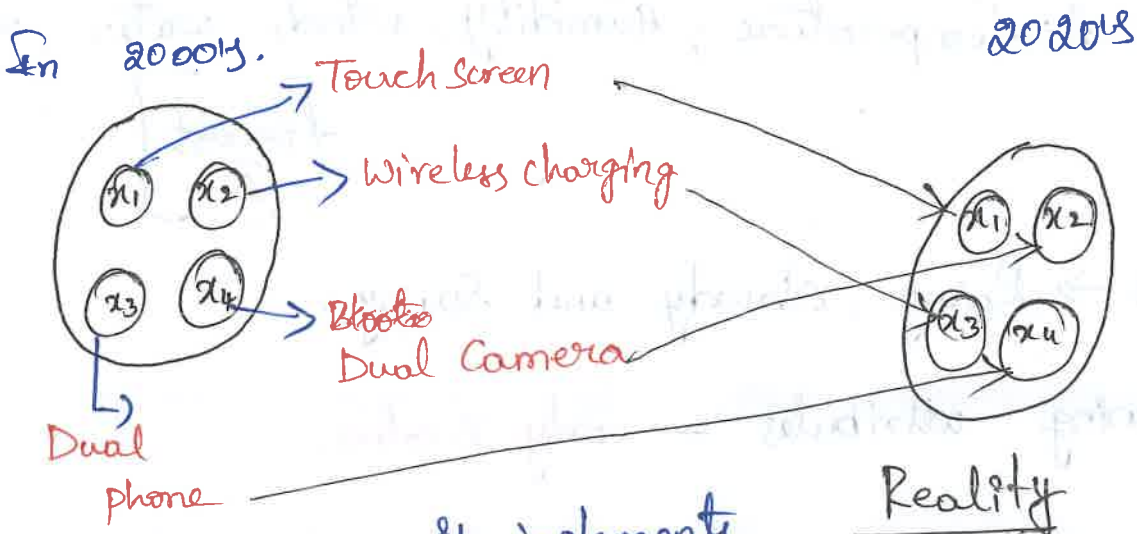
↳ (Most general hypothesis).

∵ ϕ Called phi

Main Goal of Concept learning :-

find all Concepts / hypothesis that are Consistent.

Example:-



it implements which are consistent.

* Concept learning defines to find out the concepts, hypothesis are the always consistent.

Concept Learning as Search :-

Main goal of this search is to find the hypothesis that best fits the training examples.

Example:- Enjoy Sport learning task.

6 attributes = (Sky, Air-temperature, humidity, wind, water and forecast). Concept.

Attributes

Example	Sky	Airtemp	Humidity	wind	water	forecast	Enjoysport
1	Sunny	warm	Normal	Strong	warm	Same	Yes
2	Sunny	warm	High	Strong	warm	same change	Yes
3	Rainy	cold	High	Strong	warm	change	No
4	Sunny	warm	High	Strong	cool	change	Yes

* [(1) Sky , (2) Airtemperature , (3) Humidity , (4) wind , (5) water and (6) forecast]

  → 3 values → Rainy , Cloudy and Sunny.

  → Remaining attributes — only 2 values.

* ∴ Different instances possible = 3 X 2 X 2 X 2 X 2 X 2 = 96

* Syntactically distinct hypothesis = 5 X 4 X 4 X 4 X 4 X 4 = 5120
  (Additionally, 2 more values — ? and ϕ)

  → Ex:- for Sky :-

  < ϕ , Rainy , cloudy , Sunny , ? > = (5)

* If Null (ϕ) taken as common :-

  Syntactically / Semantically distinct hypothesis =

  1 + (4 X 3 X 3 X 3 X 3 X 3) = 973.

* After finding all the Syntactically & Semantically distinct hypothesis, we search the best match from all these (i.e., much closer to our learning problem).

General-to-Specific Ordering of Hypothesis :-

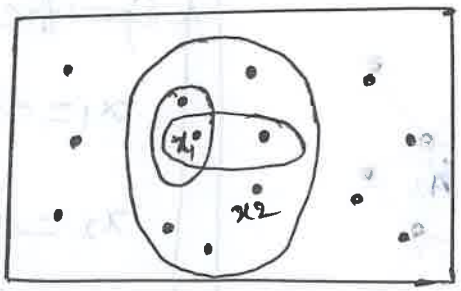
(More general than)x

Def:- Let h_j and h_k be Boolean-Valued functions defined over X. Then h_j is more general-than-or-equal to h_k (written, $h_j \geq h_k$) if and only if.

① More general than :-

Let h_j and h_k be Boolean-Valued functions defined over X. For any instance 'x' in 'X', hypothesis h in H, we can say 'x' satisfies 'h' iff $h(x) = 1$.

i.e., h_j is "more general than" h_k ($h_j >_g h_k$).

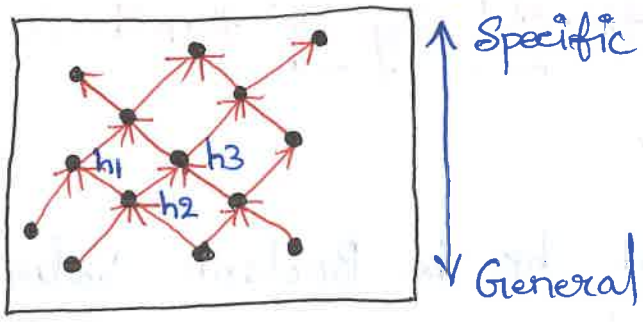


Instances X

② More general than or equal to :-

When set of training examples satisfy the 2 hypothesis h_j and h_k, h_j is "more general than or equal to" h_k ($h_j \geq_g h_k$) iff any instance x that satisfies h_k also satisfies h_j.

*> we can also define the more-specific-than ordering.

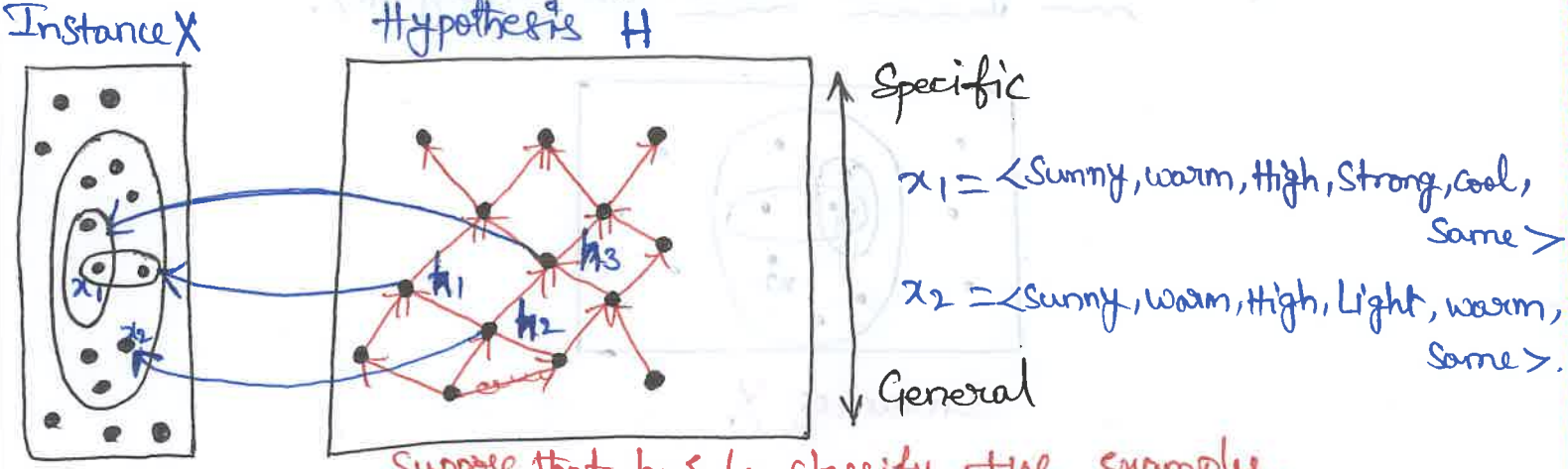


$$(\forall x \in X)\left[(h_K(x)=1) \rightarrow (h_j(x)=1)\right]$$

↑ x satisfies h_K.

③ General-to-Specific Ordering :-

Many Concept learning algorithms organize the search through the hypothesis space by taking advantage of a "Naturally occuring Structure" over it.



Suppose that h_1 & h_2 classify true examples.

Eg:- let's take two hypothesis, and two training Ex's.

$h_1 = \langle Sunny, ?, ?, Strong, ?, ? \rangle$

$h_2 = \langle Sunny, ?, ?, ?, ?, ? \rangle$

h_2 – is more general than h_1.

* h_2 – it imposes fewer constraints on instances.
* h_2 – Classify more +ve instances than h_j.

Find-S: Finding a maximally Specific Hypothesis

* The find-S algorithm is a basic Concept learning algorithm in machine learning.
* "The find-S algorithm finds the most specific hypothesis that fits all the positive examples"

Note:- The algorithm Considers only +ve examples (training examples).

* The find-S algorithm starts with the most specific hypothesis and generalizes this hypothesis until it cannot classify further positive training data (i.e., further classification will not be possible as it reaches the end.)

Algorithm:-

1. Initialize h to the most specific hypothesis
2. For every positive training instance X for each attribute Constraint a in h, If the Constraint a is satisfied by X,

   Then do nothing

Else

Replace a, in h by the next more general Constraint that is satisfied by the x.

3. Output Hypothesis. h

[OR]

Algorithm:-

Step 1:- Initiate with most specific hypothesis,

$h_0 = \langle \phi, \phi, \phi, \phi, \phi, \phi \rangle$

Step 2:- for each positive sample/example,

for each attribute

if (value = hypothesis value) = ignore.

Else

Replace with most general hypothesis.

Example	SKY	AirTemp	Humidity	Wind	water	forecast	Enjoy Sport
1	Sunny	warm	Normal	Strong	warm	same	yes
2	Sunny	warm	High	Strong	warm	Same	yes
3	Rainy	cold	High	Strong	warm	change	No
4	Sunny	warm	High	Strong	cool	change	yes

$h_0 = \langle \phi, \phi, \phi, \phi, \phi, \phi \rangle$

$h_1 = \langle$ Sunny, warm, Normal, Strong, warm same $\rangle$

$h_2 = \langle$ Sunny, warm, ?, Strong, warm same $\rangle$

$h_3 = h_2$ [∵ it doesn't allow -ve hypothesis]

$h_4 = \langle$ Sunny, warm, ?, ?, ?, ? $\rangle$

Drawbacks :-

* Considers only +ve values
* h_4 may not be sole hypothesis that fits complete data

Version Space :-

Subset of hypothesis H Consistent with the training examples D.

$$VS_{H,D} = \{ h \in H \mid \text{Consistent}(h, D) \}$$

where H = Hypothesis space

h = Subset of hypothesis space H.

D = training examples.

Here Consistent :-

$h(x) = c(x)$.

[where x = instance

$h(x)$ = hypothesis of instance x

$c(x)$ = target concept of instance x

Algorithm to obtain Version Space :-
(List then eliminate algorithm)

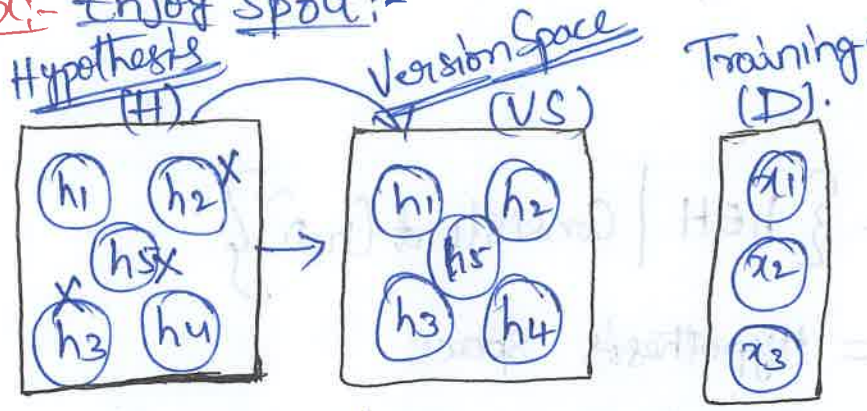
1. All the hypothesis H is kept in version space (irrespective of the fact that it's constistant or not).

2. Now we keep on removing inconsistent hypothesis from version space.

→ for each training example $\langle x, c(x) \rangle$ remove any hypothesis that is $h(x) \neq c(x)$.

3. O/p the list of hypothesis into version space after checking for all training examples.

Ex:- Enjoy Sport:-



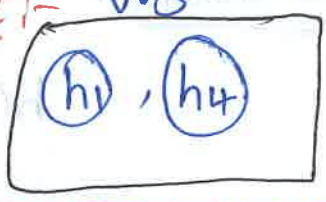
$h_1(x_1) = c(x_1)$
$h_1(x_2) \neq c(x_2)$

$h_2(x_1) \neq c(x_1)$
$h_2(x_2) = c(x_2)$
$h_2(x_3) = c(x_3)$.

$h_4(x_1) = c(x_1)$
$h_4(x_2) = c(x_2)$
$h_4(x_3) = c(x_3)$

$h_5(x_1) \neq c(x_1)$
$h_5(x_2) \neq c(x_2)$
$h_5(x_3) \neq c(x_3)$

final output:- V.S



Candidate Elimination Algorithm :-

* It's a Extended form of find-S algorithm.
* It Considers both +ve and -ve Examples.
* Here +ve examples are used as find-S algorithm (i.e., Specific to general).
* while -ve examples from general to Specific.

$S = \{ \phi \; \phi \; \phi \; \phi \; \phi \; \phi \}$ +↓

$G = \{ ? \; ? \; ? \; ? \; ? \; ? \}$ -↑

Algorithm :-

1. Initialize both general and Specific hypothesis (S and G).

$S = \{ \phi, \phi, \phi, \phi \cdots \phi \}$ depends on

$G = \{ ?, ?, ?, ? \cdots ? \}$ no. of attribute

2. For each Example,

   if Example is positive

   make Specific to general.

   Else Example is -ve.

   make general to Specific.

Ex:- Enjoy Sport. as a Example.

$S_0 = \{\phi, \phi, \phi, \phi, \phi, \phi\}$, $G_0 = \{?, ?, ?, ?, ?, ?\}$. (20)

1. true,

$S_1 = \{$'Sunny', 'warm', 'normal', 'Strong', 'warm', 'same'

$G_1 = \{? ? ? ? ? ?\}$.

2. true (Specific to general).

$S_2 = \{$'Sunny', 'warm', ?, 'Strong', 'warm', 'same'$\}$

$G_2 = \{? ? ? ? ? ?\}$.

3. $S_3 = \{$'sunny', 'warm', ?, 'Strong', 'warm', 'same'$\}$.

$G_3 = \{$<'Sunny', ? ? ? ? ?>, <?, 'warm', ? ? ? ?>, <? ? ? ? ? 'same'>$\}$.

4. $S_4 = \{$'Sunny', 'warm', ?, 'Strong', ? ?>.

$G_4 = \{$<'Sunny' ? ? ? ? ?>, <? 'warm' ? ? ? ?>$\}$.

Final output :-

$S_4 = \{$ Sunny, warm, ?, Strong, ?, ?$\}$.

$G_4 = \{$ <'Sunny', ? ? ? ? ?>, <? 'warm' ? ? ? ?>$\}$.

Data Set :- Example "Enjoy Sport" Table.

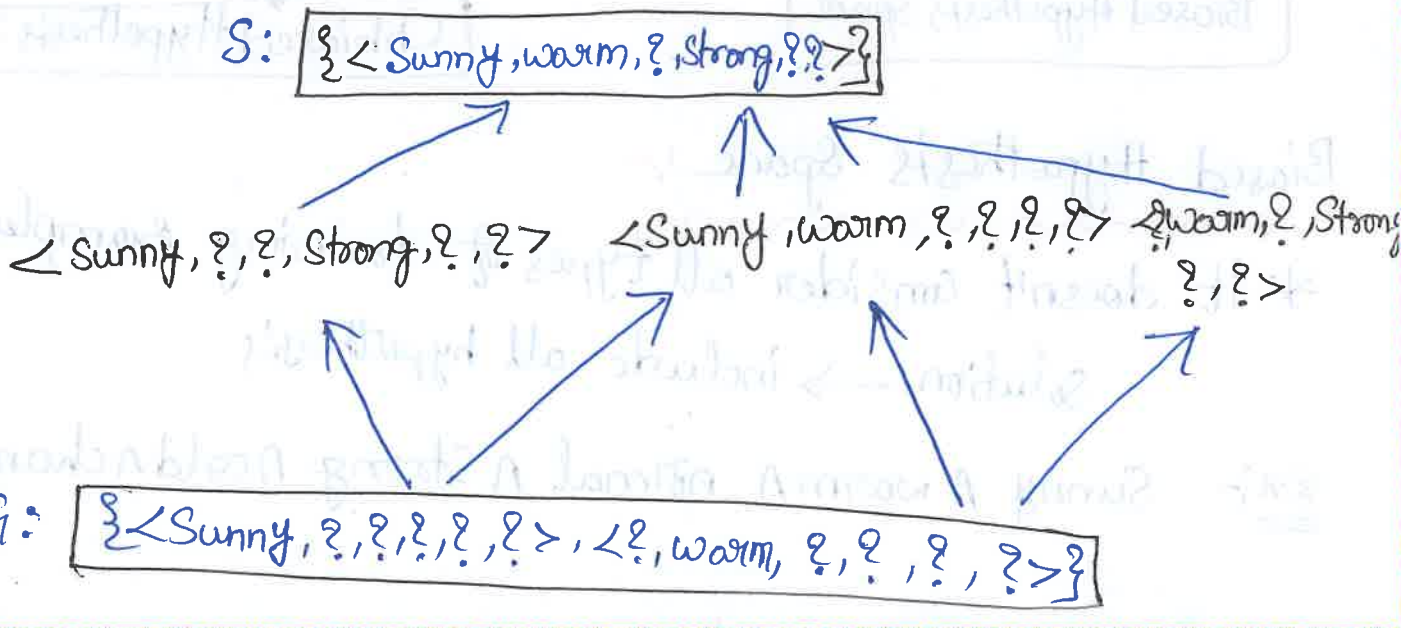
Remarks on VersionSpace and Candidate-Elimination:

① Will the CANDIDATE-ELEMINATION Algorithm Converge to the correct hypothesis?

* The learned version space correctly describes the target Concept, provided:
  1. There are no errors in the training Examples
  2. There is some hypothesis that correctly describes the target Concept.
* If S and G Converge to a Single hypothesis the Concept is exactly learned.
* An empty version space means no hypothesis in H is Consistent with training examples.

② How Can partially Learned Concepts Be used?

* The learner is required to classify new instances that it has not yet observed.



③ What training example Should the learner Request Next?

* Consider the version space learned from the four training Examples of the Enjoy Sport concept.
* what would be a good query for the learner to pose at this point ?
* what is a good query Strategy in general ?

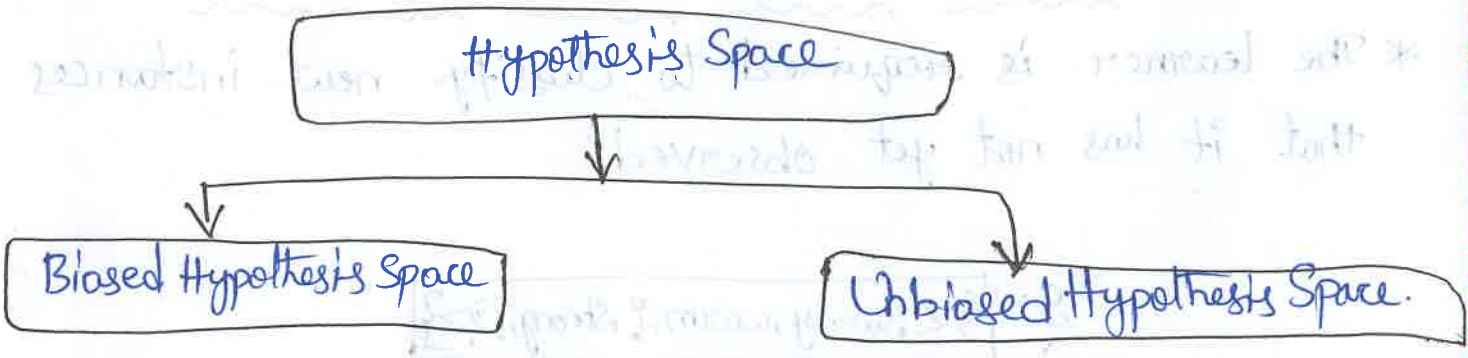
Inductive Bias :- It refers to a set of assumptions that the learner uses to predict o/p's given unseen i/p's.

Remarks on CE and VS Algorithms :-

1. will the CE algorithm gives us correct hypothesis ?
2. what training examples should the learner request next ?

Inductive learning :-

Derive rules from Examples



Biased Hypothesis Space :-

* it doesn't Consider all types of training Examples.

Solution ⟶ include all hypothesis

Ex :- Sunny ∧ warm ∧ normal ∧ Strong ∧ cold ∧ change = Yes.

Unbiased Hypothesis Space :-

* providing a hypothesis capable of representing set of all Examples.

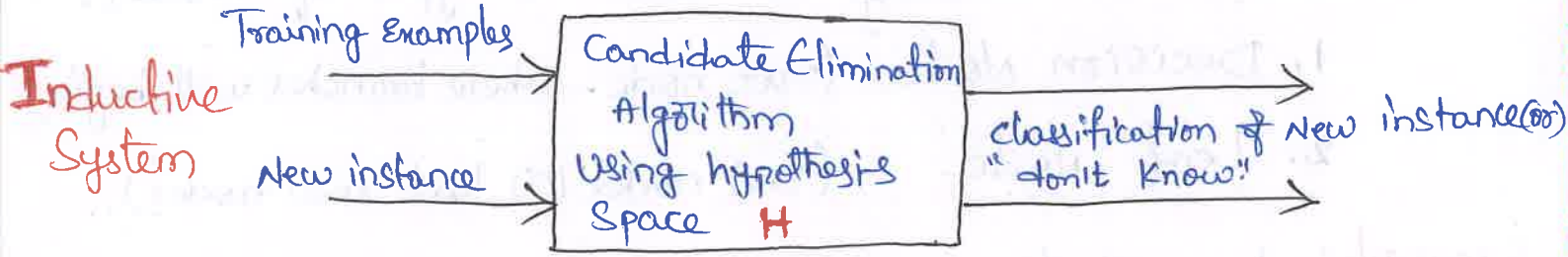
possible Instances :- $3 \times 2 \times 2 \times 2 \times 2 \times 2 = 96$.

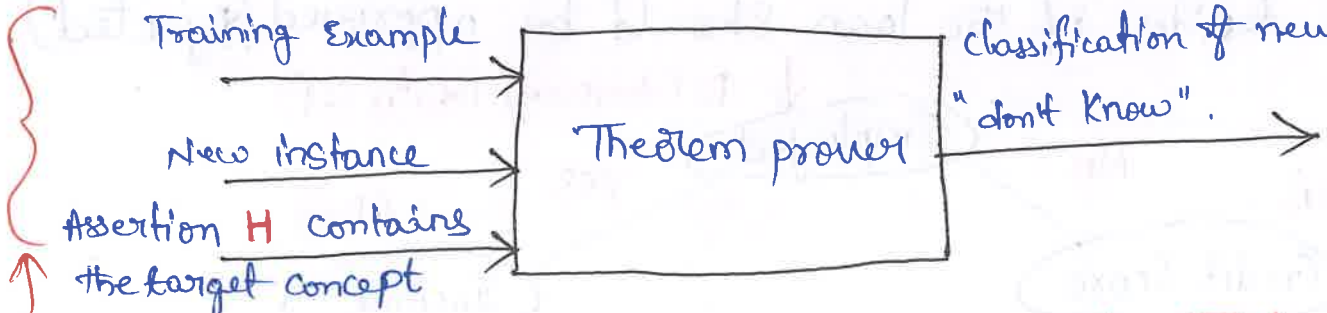
Target Concepts :- 2^{96} (huge).

(practically not possible), it takes a lot of time.

Instance X → Attributes

Sky	3 →	Sunny, cludy, Rainy
AirTemp	2 →	warm, cold
Humidity	2 →	Normal, high
Wind	2 →	Strong, weak
water	2 →	warm, cold
forecast	2 →	same, change.





Equivalent Deductive System.

fig:- Modeling Inductive Sys.

Decision Tree Learning (ID3):-

Decision Tree Learning is a method that uses inductive inference to approximate a target function, which will produce discrete values. And the following features;

* widely used
* Robust to noisy data.
* Considers a practical method for learning disjunctive expressions.

* It's a tree structured classifier, where

① → Internal nodes represent the features;

② → Branches represents the decision rules;

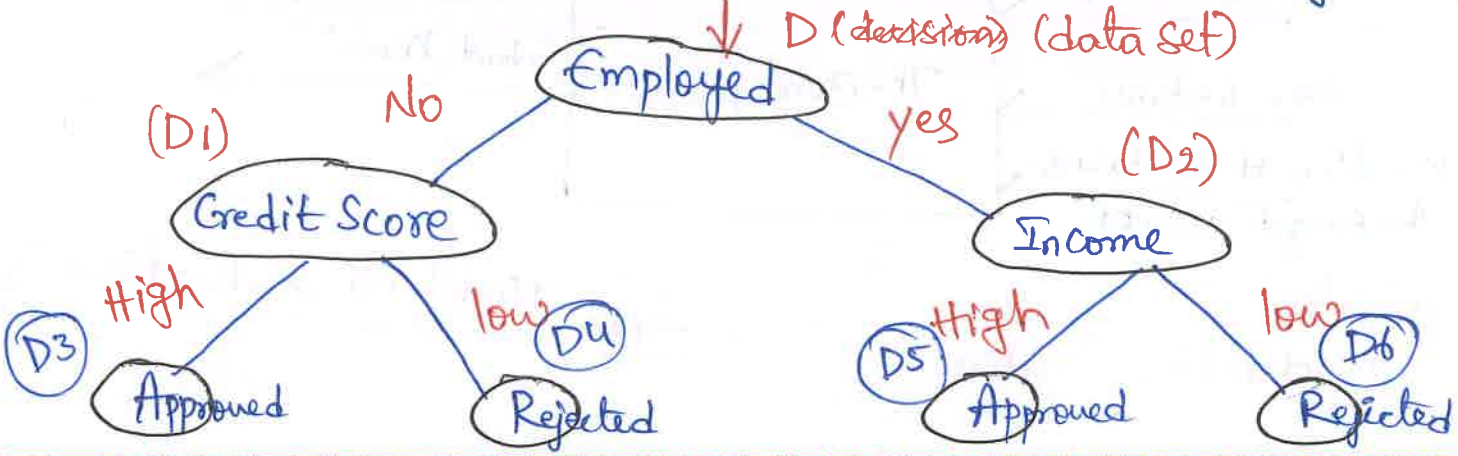
③ → Each leaf node represents the Outcomes;

In decision tree mainly we have 2 types of Nodes.

1. Decision Node (root node - where branches will begin)
2. Leaf Node. (End nodes or last row nodes).

Example:- Loan System

(it decides if the loan should be approved/rejected).



Algorithm: Decision Tree Terminologies :-

Root / Decision Node :- It is the node where the decision tree starts. It represents entire dataset.

Leaf Node :- Leaf nodes are the final output node, and tree cannot be segregated further after getting a leaf node.

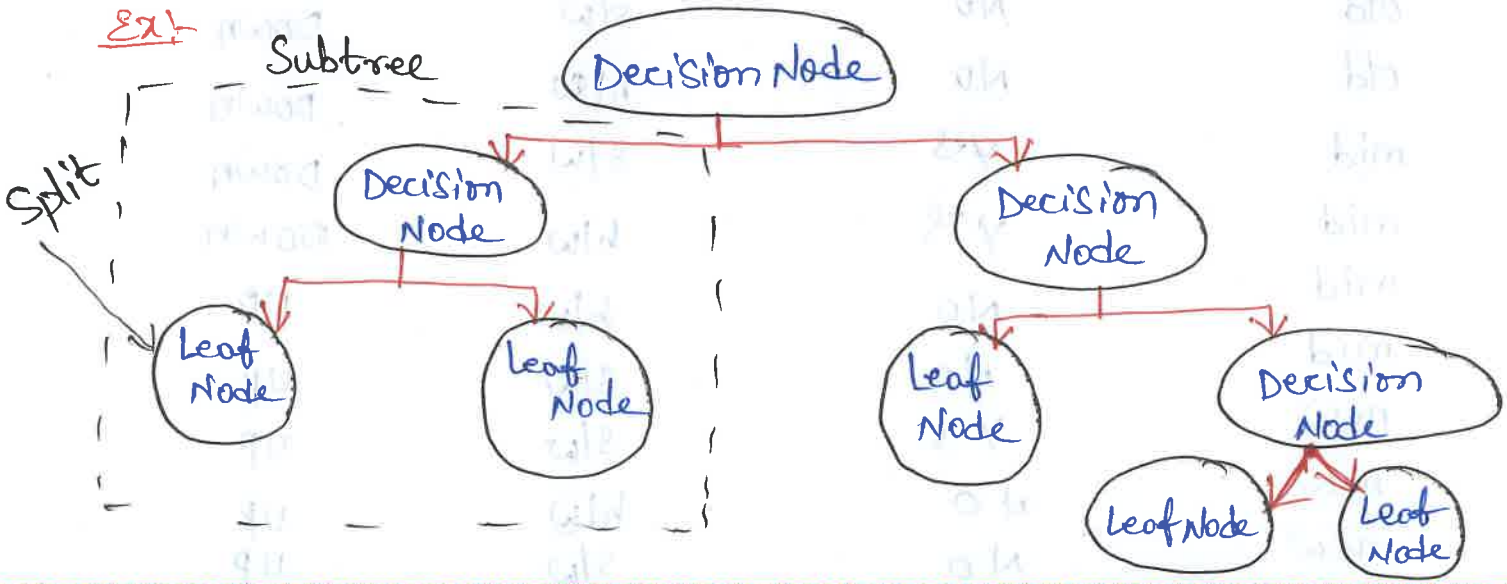
Splitting :- Splitting is the process of dividing the decision node into subnodes according to given conditions.

Branch / Sub-tree :- A tree formed by splitting the tree

pruning :- It is a process of removing unwanted branches from the tree

Depth :- The depth of a node is the minimum no. of steps required to reach the node from the root.

Ex :-



Algorithm of Decision Tree Learning :-

Step1 : In the given dataset, choose a target attribute.

Step2 : Calculate information gain of target attribute.

$$\left[IG = \frac{-P}{P+N} \log_2 \left(\frac{P}{P+N}\right) - \frac{N}{P+N} \log_2 \left(\frac{N}{P+N}\right) \right]$$

Step3 : for remaining attributes, find entropy.

$$[\text{Entropy} = IG \times \text{probability}]$$

$$E(A) = \sum \frac{Pi+Ni}{P+N} I(PiNi)$$

Step4 : calculate gain $= IG - E(A)$

Example :-

Age	Competetion	Type	profit
old	yes	S/w	Down
old	No	s/w	Down
old	No	h/w	Down
mid	yes	s/w	Down
mid	yes	h/w	Down
mid	No	h/w	UP
mid	No	S/w	UP
new	yes	S/w	UP
new	NO	h/w	UP
new	No	S/w	UP

Step – 1

Target Attribute = "profit".

Step – 2

Information gain

$$IG = \frac{-P}{P+N}\log_2\left(\frac{P}{P+N}\right) - \frac{N}{P+N}\log_2\left(\frac{N}{P+N}\right)$$

→ P = Count (down) = 5

→ N = Count (up) = 5

$$= \frac{-5}{5+5}\log_2\left(\frac{5}{5+5}\right) - \frac{5}{5+5}\log_2\left(\frac{5}{5+5}\right)$$

$$= \frac{-5}{10}\log_2\left(\frac{5}{10}\right) - \frac{5}{10}\log_2\left(\frac{5}{10}\right)$$

$$= -\left[\frac{1}{2}\log_2(2^{-1}) + \frac{1}{2}\log_2(2^{-1})\right] \quad [\because \log_x m^n = n\log_x m]$$

$$= -\left[\frac{1}{2}\times -1\log_2 2 + \frac{1}{2}\times -1\log_2 2\right] \quad [\because \log_x x = 1]$$

$$= -\left[\frac{1}{2}\times -1 + \frac{1}{2}\times -1\right]$$

$$= -\left[-\frac{1}{2} - \frac{1}{2}\right] = -\left[\frac{-1-1}{2}\right] = -\left[\frac{-2}{2}\right] = -[-1] = 1$$

∴ IG = 1.

Step – 3 calculate entropy for remaining attributes.

$$E(A) = \sum \frac{Pi+Ni}{P+N} I(Pi,Ni) \quad [\text{IG} \times \text{probability}]$$

Age:-

(1) prepare a table for each attribute.

rows— Undertaken attribute values (old, mid, new).

Columns— target attribute values. (down, up).

Age values	(P) Down	(N) Up
old	3	0
mid	2	2
new	0	3

Entropy = IG × Probability

P = down count

N = Up count

1. $IG(old) = -\left(\frac{3}{3}\log_2\left(\frac{3}{3}\right) + \left(\frac{0}{3}\right)\log\left(\frac{0}{3}\right)\right) = 0.$

[$\because \log 1 = 0$].

probability = 3/10

Entropy (old) $= 0 \times \frac{3}{10} = 0.$

2. $IG(mid) = -\left(\frac{2}{4}\log\left(\frac{2}{4}\right) + \frac{2}{4}\log\left(\frac{2}{4}\right)\right)$

$= -\left(\frac{1}{2}\log\left(\frac{1}{2}\right) + \frac{1}{2}\log\left(\frac{1}{2}\right)\right).$

$= -\left(\frac{1}{2}\log_2 2^{-1} + \frac{1}{2}\log_2 2^{-1}\right)$

$= -\left(-\frac{1}{2} + \left(\frac{-1}{2}\right)\right) = -(-1) = 1.$

probability = 4/10

Entropy (mid) $= 1 \times \frac{4}{10} = 0.4.$

3. $IG(new) = -\left[\frac{0}{3}\log_2\left(\frac{0}{3}\right) + \frac{3}{3}\log_2\left(\frac{3}{3}\right)\right] = 0$

probability = 3/10

Entropy $= 0 \times \frac{3}{10} = 0.$

$$\text{Entropy (Age)} = E(O) + E(M) + E(N)$$
$$= 0 + 0.4 + 0$$
$$= 0.4$$

Step 4 :-

$$\text{Gain} = IG - E(A) = 1 - 0.4$$
$$= 0.6.$$

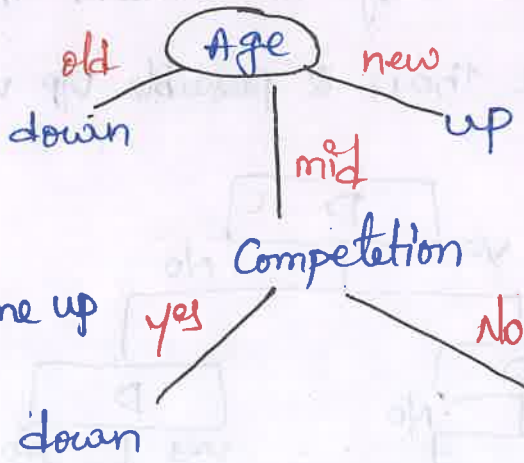
In the same way, calculate Gain of other attributes.

Gain (Competation) = 0.124

Gain (Type) = 0

Gain (Age) = 0.6

Highest gain → root node Age.



Here

old – All down

mid – Some down & Some up

new – All up

Why Competation → next highest gain.

why not other attribute (Type).

Type → gain = 0 and even in DT also, no requirement with it.

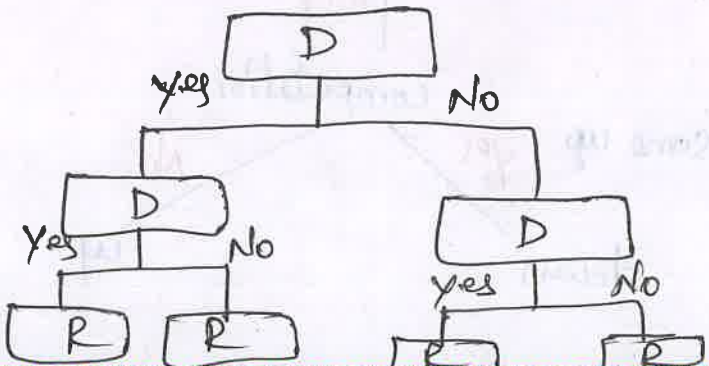
∴ It can be ignored.

* Appropriate problems for Decision Tree Learning :-

1. Instances are represented by attribute value pairs.
   ↳ when a f we have a finite list of attributes & each attribute contains small no. of distinct values, than, it is easier for the decision tree to reach the solution.

   Ex: Student Branch — (CST, ECE, IT etc). finite list.

2. The target function has discrete output.
   ↳ it works with simplest way, i.e., when there are 2 possible ways (Boolean classification).
   ↳ However it is easy to extend the decision tree to with more than 2 possible O/p values.



(3) Disjunctive descriptions may be required.

↳ Decision tree naturally represents disjunctive expressions i.e., it reduces large consideration set to a more manageble no. of alternatives.

(4) The training data may contain errors.

↳ errors in the classification of examples are handled well by decision trees & making them a robust ML learning method.

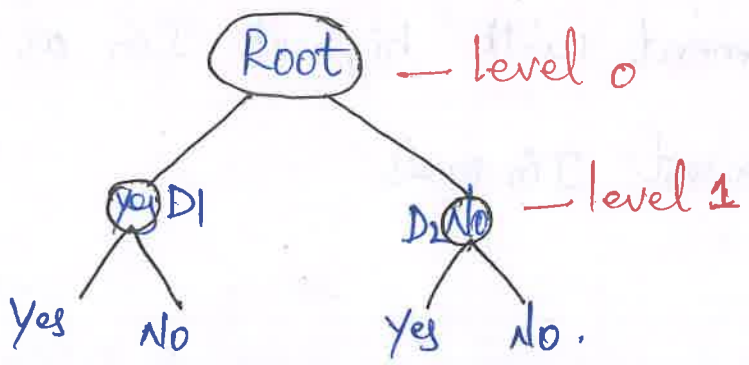
(5) The training data may contain missing attribute values.

↳ Decision tree methods can be used even when some training examples have unknown values.

Hypothesis Space Search in Decision Tree Learning :-

* ID3 can be characterized as searching a space of hypothesis for one that fits the training examples.
* ID3 will search set of possible decision trees from available hypothesis.

ID3 performs simple to complex searching



* first, start with empty tree and keep on adding.
* Every discrete valued (finite) function can be described by some decision tree.
* Avoids major risk of searching in incomplete hypothesis.
* it has only single current hypothesis.
* it cannot determine alternative decision trees.
* Back tracking is not possible.
* it can also handle noisy data.

Inductive Bias in Decision Tree Algorithm :-

↳ set of assumptions.

Inductive bias of ID3 consists of describing the basis by which ID3 chooses one consistent decision tree over all the possible DT's.

ID3 Search Strategy :-

* Selects in favour of shorter trees over longer ones.
* Selects element with highest IG as root attribute over lowest IG ones.

Types of Inductive Bias :-

1. Restictive Bias (Based on conditions).
2. preference " (Based on priorities).

Restrictive Biases and preference Biases:

There is difference blw the inductive bias exhibited by ID3 and by the candidate elimination algorithm.

* ID3 searches incompletely through this space, from simple to complex hypotheses, until its termination condition is met. It's inductive bias is solely a consequence of the ordering of hypotheses by its search strategy.

* The inductive bias of ID3 is thus a preference of certain hypotheses over others. with no hard restriction on the hypotheses that can be eventually enumerated. This form of bias is typically called a preferential bias.

* Candidate-Algorithm searches this space completely, finding every hypothesis consistent with the training data. It's inductive bias is solely a consequence of the expressive power of it's hypothesis representation

* A preference bias is more desirable than a restriction bias because it allows the learner to work within a complete hypothesis space that is assured to contain the unknown target function

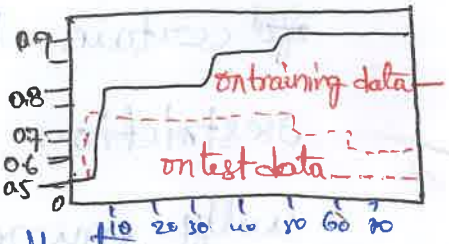
* A restriction bias that strictly limits the set of potential hypothesis is generally less desirable, because it introduces the possibility of excluding the unknown target function altogether.

Why short hypothesis?

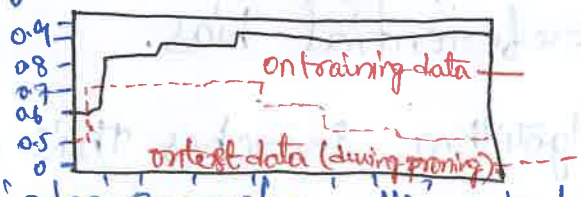
Occam's razor: prefer the simplest hypothesis that fits the data.

Issues in Decision Tree Learning :-

1. Overfitting the data



2. Incorporating continuous valued attributes



3. Handling training examples with missing attribute values.

4. Handling attributes with different costs.

5. Alternative measures for selecting attributes.