TEEGALA KRISHNA REDDY ENGINEERING COLLEGE

B.Tech III Year I Sem | **L T P C** | **3 0 0 3**

22AI501PC: MACHINE LEARNING

Course Objectives

1. To introduce students to the basic concepts and techniques of Machine Learning.
2. To have a thorough understanding of the Supervised and Unsupervised learning techniques.
3. To study the various probability-based learning techniques

Course Outcomes:

CO1: Distinguish between, supervised, unsupervised and semi-supervised learning
CO2: Understand algorithms for building classifiers applied on datasets of non-linearly separable classes.
CO3: Understand the principles of evolutionary computing algorithms.
CO4: Design an ensemble to increase the classification accuracy
CO5: Gain knowledge on Bayesian Networks, Reinforcement Algorithms & analytical learning.

COPO Matrix:

	PROGRAMME OUTCOMES													
Cos	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2
CO1	3	2	2	2	1				1	1		1	2	1
CO2	3	2	3	2					2	2	1	2	2	2
CO3	2	2	3	2					1	1		1	2	1
CO4	2	2	2	2					1	1	1	2	2	1
CO5	2	2	3	2					2	1		2	2	1

UNIT – I

Learning – Types of Machine Learning – Supervised Learning – The Brain and the Neuron – Design a Learning System – Perspectives and Issues in Machine Learning – Concept Learning Task – Concept Learning as Search – Finding a Maximally Specific Hypothesis – Version Spaces and the Candidate Elimination Algorithm – Linear Discriminants: – Perceptron – Linear Separability – Linear Regression

UNIT - II:

Multi-layer Perceptron– Going Forwards – Going Backwards: Back Propagation Error – Multi-layer Perceptron in Practice – Examples of using the MLP – Overview – Deriving Back-Propagation – Radial Basis Functions and Splines – Concepts – RBF Network – Curse of Dimensionality – Interpolations and Basis Functions – Support Vector Machines

UNIT – III

Learning with Trees – Decision Trees – Constructing Decision Trees – Classification and Regression Trees – Ensemble Learning – Boosting – Bagging – Different ways to Combine Classifiers – Basic Statistics – Gaussian Mixture Models – Nearest Neighbor Methods – Unsupervised Learning – K means Algorithms

UNIT - IV:

Dimensionality Reduction – Linear Discriminant Analysis – Principal Component Analysis – Factor Analysis – Independent Component Analysis – Locally Linear Embedding – Isomap – Least Squares Optimization. Evolutionary Learning – Genetic algorithms – Genetic Offspring: - Genetic Operators – Using Genetic Algorithms

UNIT-V:

Reinforcement Learning – Overview – Getting Lost Example Markov Chain Monte Carlo Methods – Sampling – Proposal Distribution – Markov Chain Monte Carlo – Graphical Models – Bayesian Networks – Markov Random Fields – Hidden Markov Models – TrackingMethods.

TEXT BOOKS:

1 Stephen Marsland, —Machine Learning — An Algorithmic Perspective, Second Edition,Chapman and Hall/CRC Machine Learning and Pattern Recognition Series, 2014

REFERENCE BOOKS:

1 Tom M Mitchell, —Machine Learning, First Edition, McGraw Hill Education, 2013.

2 Peter Flach, —Machine Learning: The Art and Science of Algorithms that Make Sense of Data!,First Edition, Cambridge University Press, 2012.

3 Jason Bell, —Machine learning – Hands on for Developers and Technical Professionals!, FirstEdition, Wiley, 2014

4 Ethem Alpaydin, —Introduction to Machine Learning 3e (Adaptive Computation and MachineLearning Series), Third Edition, MIT Press, 2014

Machine learning Note

[UNIT - I]

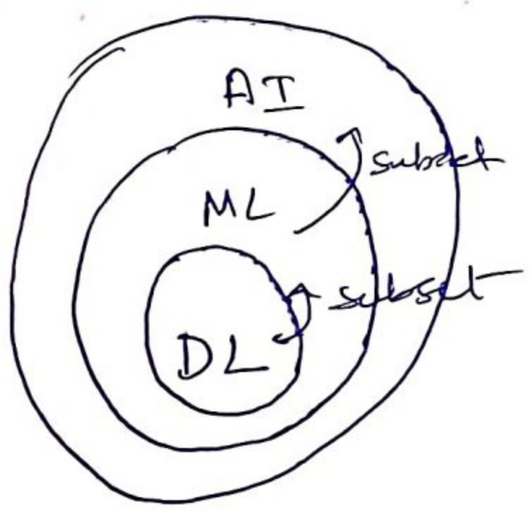
→ What is Machine learning?

→ field of study that gives computers a capability to learn without being explicitly programmed.

read not give any explicit programs

Example:- Online shopping. Amazon search something to first time second time the same thing will view for you

→ Machine adapts to the user based on data.



+→ hundred to two above low to give high but ML no need

* Well Posed learning Problem :- (learning by Experience)

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

good output

If P is measured at T it can improve E

(learning by Experience)

Examples :-

1. Playing checkers Problem
2. Hand written recognition Problem
3. Robo driving learning Problem



Problem	Task	Performance	experience
1. Playing checkers learning	playing against opponents to win game	make perfect moves to win game	It plays itself to improve
2. Hand writing recognition learning	classifying the images and text	Better classification	A database of Home work text ↓ inbuilt database
3. Robot driving learning problem	Drive the car in a 4 lane highway	Source to destination the avg distance travelled (long & safe)	Images, vehicles on Road

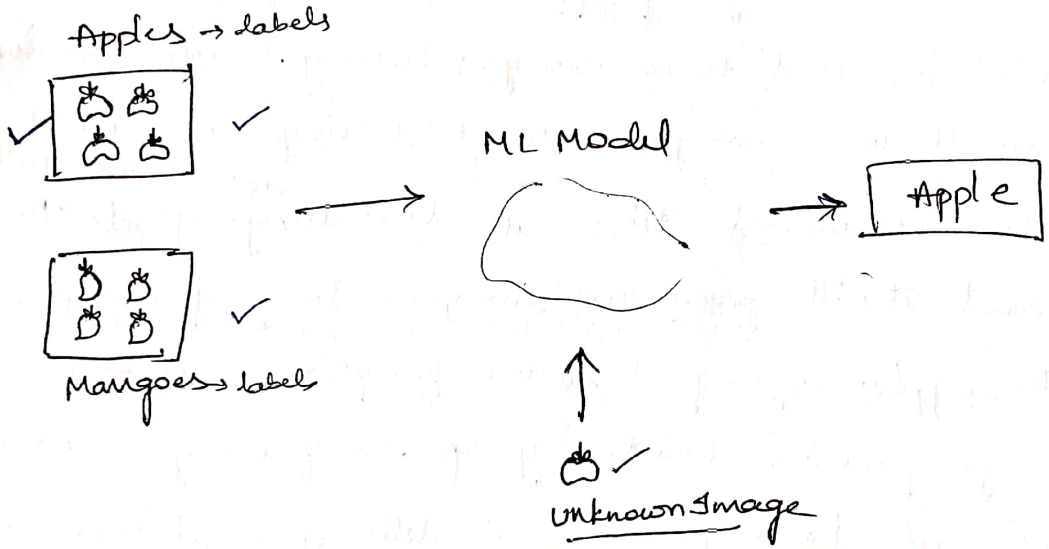
Scanned with OKEN Scanner

Types of Machine learning :-

1. Supervised learning
2. Unsupervised learning
3. Reinforcement learning.

1. Supervised learning :

In supervised learning, the machine learning algorithm learns from Labelled Data.



We take several images of apples and mangoes and we will train the machine that these images belongs to apples and these images belongs to mangoes. These apples and Mangoes are called labels. Now we train ML model with these datasets. Now ML model or ML algorithm tries to find the patterns between these images. once if it learns from the data, when you give an unknown image, it can correctly recognize whether the image represents an apple or a mango. So we are giving supervisions in terms of labels.

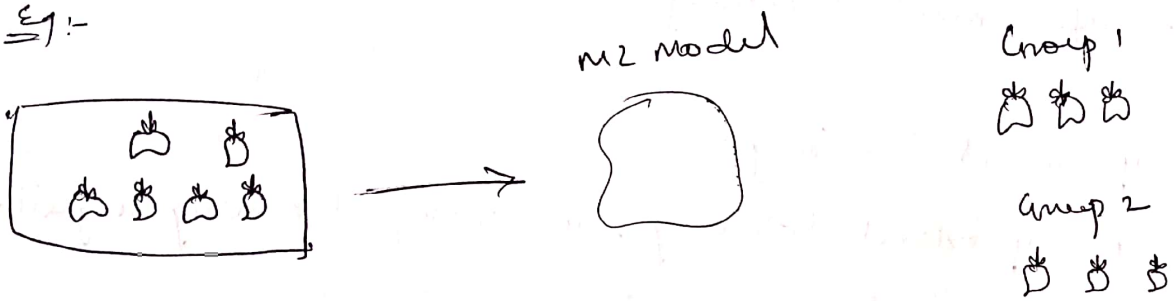
Scanned with OKEN Scanner

2. Unsupervised Learning :-

In Unsupervised Learning, the machine learning algorithm learns from Unlabelled data.

Here we won't give any labels.

Eg :-



We will give several images of apples and mangoes to our ML model and we won't tell/train that these images belong to apples or mangoes so we train all these images without telling that what it is to our ML model. Then, it tries to again find the pattern and groups all these images and it will group the images into grup 1 and grup 2 so all the apples grouped in one grup and all the mangoes grouped in another grup. So if we give new image of apple / mango, then it tells you whether it belongs to the grup 1 or group 2.

Reinforcement :- Reinforcement learning is an area of ML Concerned with how intelligent agents take actions in an environment to maximize its rewards

→ Environment
→ Agent
→ Action
→ Reward

Scanned with OKEN Scanner

There will be an environment and we need to build an agent that acts in that environment, that agent in that environment, tries to take some actions. for that actions, it gains some rewards.

Ex:- playing a chess in computer like a human being

here, chess board is environment

computer is agent

So, agent tries to take actions so, that actions represents the move : for each step it gets a reward. so reward is winning the chess game.

→ for each step it closes to winning, it will get a positive reward.

→ If bad step / bad move, then it will get a -ve reward

→ By the machine tries to learn how to play that game

Applications of Reinforcement learning :-

Automatic drones

cars

itself * Designing a learning system

To get a successful learning system, it should be designed for a proper design. Several steps should be followed

why? - for perfect and efficient system.

There are 5 steps:

1.) Choosing the training experience - (3) attributes
2.) " " target function
3.) " a representation for target function → f(x)
4.) " a learning algorithm for approximating the target function.
5.) final design, is obtained.

Step 1 :- Choosing a training Experience :-

In choosing a training experience, 3 attributes are taken (1.) Type of feedback direct / indirect
(2.) degree (up to which the learner will control)
(3.) distribution of examples how many distributed wide range of examples that we have taken

(1) Type of feedback:
→ whether the training experience provides direct or indirect feedback regarding the choices made by performance system.
- Direct feedback
- Indirect feedback

Indirect

Task → (P) performance should be good & good Experience

How? through train.

Direct Indirect

Examples:- Checkers game, learning driving

→ we You want to go with direct / Indirect. Its your choice

Direct:

In checkers game, what is direct & Indirect?

when we make a move, immediately after making a mov

they will tell you whether you made a perfect move or a

computer

correct move or wrong move

Indirect :- not telling directly but it will give you hints

→ Learning driving

Direct :-

In this beside us a trainer will sit in car. She/he will Explan

each & everything about car (gear, break ...)

- when we go on a hilly road, he/she will be telling us

& each a everything to move slowly, to give break ... etc

So this comes under direct feedback. (

Indirect:

In this driver ~~etc~~ a trainer will not sit beside us. Some

recorded Sessions will be there. So that ~~that~~ it will not simultaneously

tell us about how to drive. It will tell at this point you need

to go slow ... something like that

(2) Degree:

Degree to which learner (the person who undergoing training) will carried the sequence of training.
(upto what extent)

Example:- learning driving

- with trainers help
- with trainer partial help
- completely on your own

(3) Distribution of Examples:

How well it represents the distribution of examples over which the performance of final system is measured.

- more possible combinations
- more situations, more examples
- learning over distributed range of examples.

(when we proving training for someone or learner, that training should include so many examples distribution of examples should be there) then only the learner can understand it properly

Example:- learning driving (for driving license testing)
- trainer
- without trainer

(The trainer has to teach so many different possible situations to the machines. so that the machine will be able to clear those problems, if it faces in the future easily)

Step 2:- choosing the target function:-

what type of knowledge is learnt and how it is used by the performance system.
(Implementation)

Example:- Checkers game.

→ while moving diagonally, set of all possible moves is called legal moves
(possible moves)
(one move)

- Travel only in forward direction
- only one move per chance
- only in diagonal direction
- may jump over opponent



Scanned with OKEN Scanner

denote Target function => V(b)

Board state => b

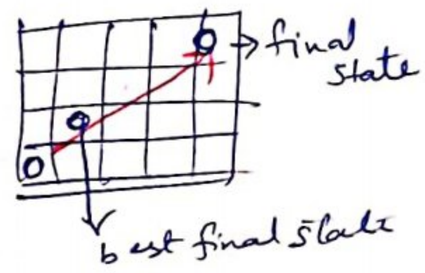
Legal moves set => B

How do you assign the value of V(b)?
How do you select a target fn?

why we need target f.
you need a target functi
because what type of
knowledge is learnt by the
machine and that knowledge
the machine has learnt will b
useful for the machine in
the implementation or not.
you need to find out

1. b is final board state that is won, then V(b) = 100
2. b is final board state that is lost, then V(b) = -100
3. b is final board state that is draw, then V(b) = 0
   [neither black won or white won then V(b) = 0]
4. If b is not final state then V(b) = V(b')

b' → best final state



Step 3:- Choosing a Representation for Target function:-

for any board state (↳ winning / loosing state), we calculate function 'c' as linear combination of following board features ie; c(b)

(linear, degree 1) (degree n)

→ 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 (blacks which can be beaten by red)
x_6 – No. of red pieces threatened by black (reds which can be beaten by black)

Scanned with OKEN Scanner

$\hat{V}(b)$ is represented by

$$\hat{V}(b) = \omega_0 + \omega_1 x_1 + \omega_2 x_2 + \omega_3 x_3 + \omega_4 x_4 + \omega_5 x_5 + \omega_6 x_6$$

(ω_1: 3→black; ω_2: 2→red; $x_1 \ldots x_6$: features; $\omega_0 \ldots \omega_6$: numerical weights)

(representation of $C(b)$)

($\omega_0 - \omega_6$) = numerical coefficients (or) weight of each feature determine the importance / weightage of features

ω_0 → additive constant (doesn't have any feature)

$\omega_4 = \omega_6$

So, here based on the features and the weightage of a particular feature, we are defining a target function.

Step 4:- Choosing a learning algorithm for approximating the target function.

To learn a target function (f) we need a set of training examples.

(describe a particular board state (b) and training value $V_{train}(b)$)

(+ winning / loosing)

When we representing a training example, we represent in this way, target fn

Ordered pair : $(\overset{x}{b}, \overset{y}{V_{train}(b)})$

(training example representation)

We require training examples, because to understand clearly about the target function.

Example:- assume black won the game

(ie; $x_2 = 0$ (→ no. of reds), which means no red)

$V_{train}(b) = +100$

$b = (x_1 = 3, x_2 = 0, x_3 = 1, x_4 = 0, x_5 = 0, x_6 = 0)$

(x_1: → black pieces; x_3: → black kings; x_4: → no. of red kings; x_5: → no. of black pieces threatened by red; x_6: → red)

Scanned with OKEN Scanner

$< (x_1 = 3, x_2 = 0, x_3 = 1, x_4 = 0, x_5 = 0, x_6 = 0) + 100$

we need to do 2 steps in this phase

1. Estimating training values:

In every step, we consider successor
(next step of the opponent)

(depending on the next step of opponent)

$V_{train}(b) \leftarrow \hat{V}$ (approximation) $(Successor(b))$

represents the next board state
(whether that step/move will help or destroy)

→ estimating that this move will help/destroy opponent

$\hat{V}$ → represents approximation.

2. Adjusting the weights:

There are some algorithms to find weights of linear functions

Here, we are using LMS (least mean square)
(LMS algorithm used to minimise the error)

$$Error = (V_{train}(b) - \hat{V}(b))^2$$

If error = 0, no need to change weights

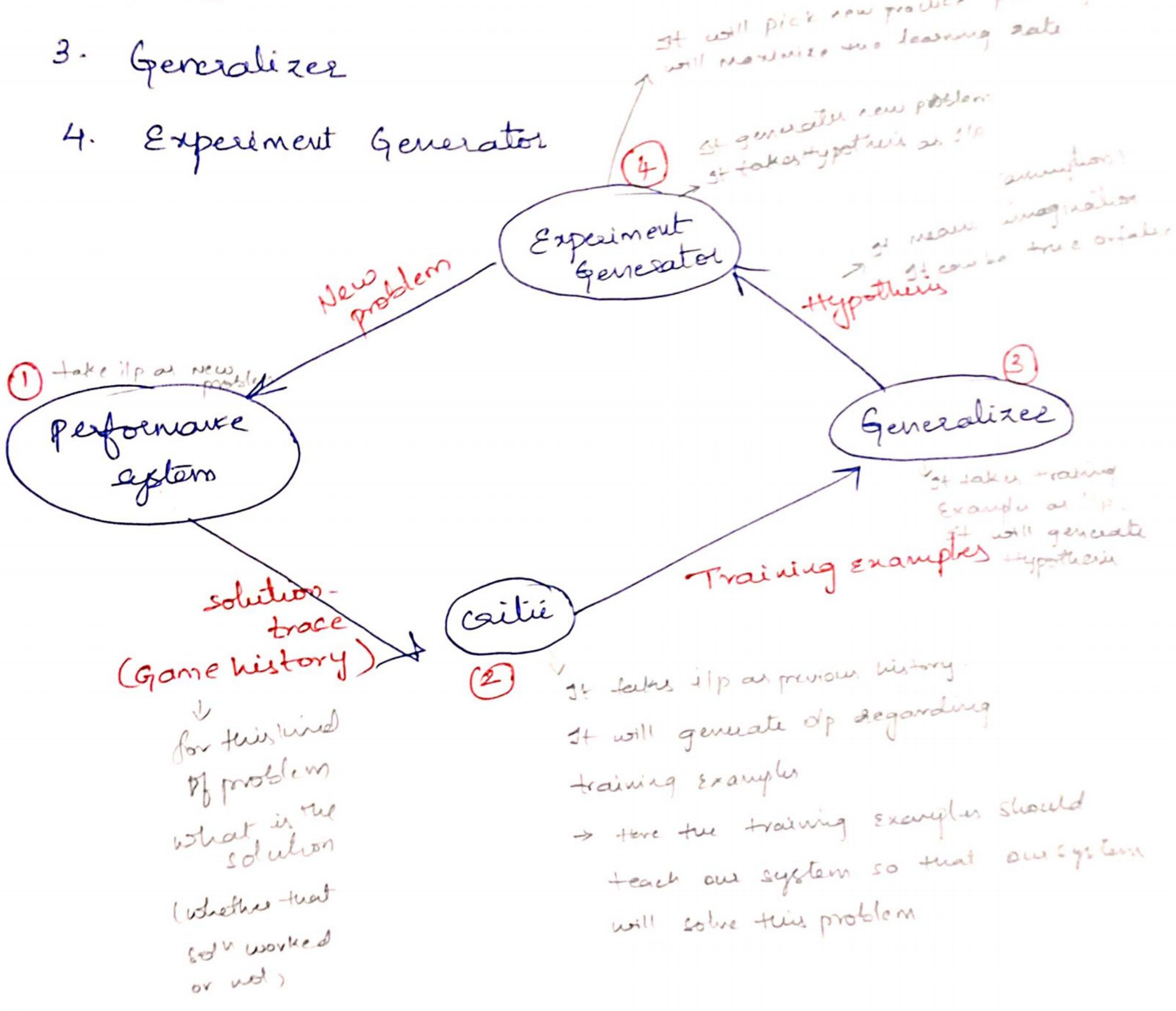
If Error is positive, each weight is increased in proportion

If Error is negative, each weight is decreased in proportion.

Final Design:

It has 4 different modules

1. Performance system
2. Critic
3. Generalizer
4. Experiment Generator



(1) take i/p as new problem

(2)
- It takes i/p as previous history
- It will generate o/p regarding training examples
- → Here the training examples should teach our system so that our system will solve this problem

(3)
- It takes training examples as i/p.
- It will generate hypothesis

(4)
- It will pick new problem with maximize the learning rate
- It generates new problem
- It takes hypothesis as i/p

- It means imagination
- It can be true or false

solution-trace (Game history)
- for this kind of problem what is the solution
- (whether that followed worked or not)

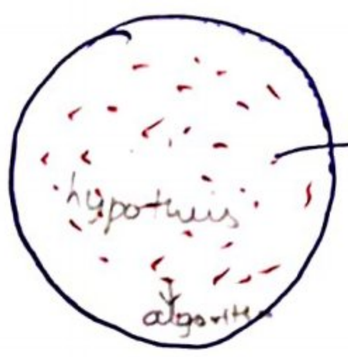
Scanned with OKEN Scanner

* Perspectives and issues in machine learning:

→ Perspective of ML:

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.

→ Issues In Machine Learning:-

1. What algorithms 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 overhea
6. for which type of data which methods should be used?

* Concept Learning :-

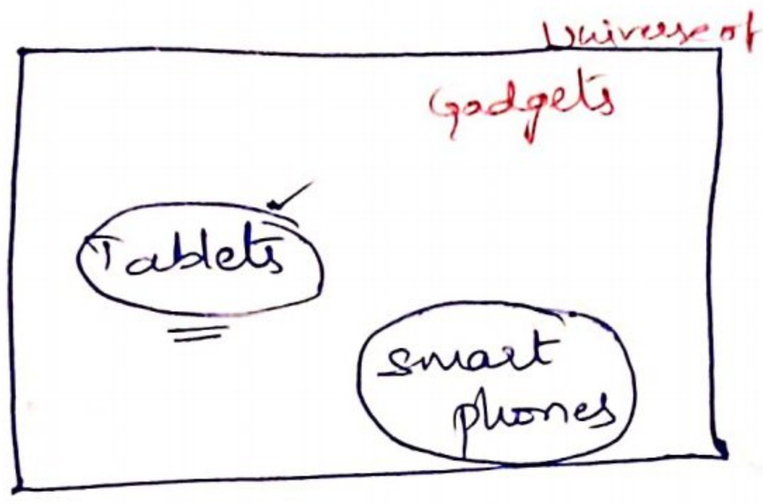
* Concept Learning As TASK :-

It will help us in finding all the consistent

hypothesis or the concepts

Different hypothesis has different concepts

Ex1



* features (Binary valued attributes)

about tablet
- Size : large (tablet), small (smart phone) → x_1
- color : Black, Blue → x_2
- Screentype : flat, folded → x_3
- shape : Square, Rectangle → x_4

Concept = $\langle x_1, x_2, x_3, x_4 \rangle$ → General representation of the concept

Tablet = < large, Black, flat, square >

Smartphone = < small, Blue, folded, Rectangle >

} representing individually for tablet & smart phone

No. of possible instances = 2^d

where, d = no. of features → we have 4 features, d = 4

$2^4 = 16$

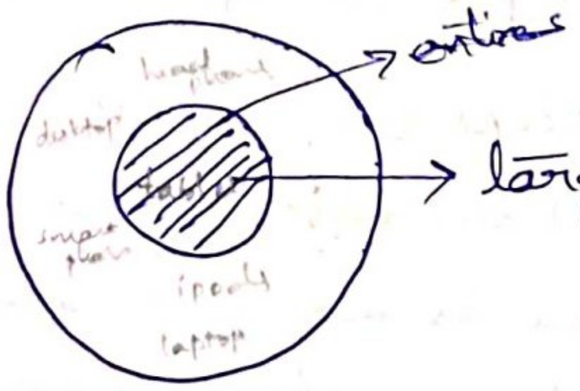
Total possible concepts = 2^{2^d} → 2^{16} (∵ $2^d = 16$)

2^{16} is the no. of possible concepts which are possible for the above example. (with features → 4)

why? we need to find all these?

for these 2^{16} concepts, we are not going to discuss about all those concepts, we are not going to learn all those concepts also we are not teaching our machine all those concepts. we will be choosing ~~those~~ some concepts from those 2^{16} concepts which we have received.

which type of concepts will be choosed? only concepts which are consistent all the time, those type of concepts will be considered.



In gadgets we have taken only tablets (not headphone, laptop, desktop, ipad etc) so our target concept is tablet (characteristics)

* whatever conclusion we have made for tablet we apply it for entire system.

we are not interested in learning about entire system we learning about a particular concept.

two Hypothesis we need to consider

1. most specific hypothesis
2. most general hypothesis

① most specific hypothesis

If all the feature are represented with null (ϕ) then Reject all

$< \phi, \phi, \phi, \phi > \rightarrow$ Reject all

that means we are going to reject all types of features, it may be best or may be worst we have to reject. this is called as the most specific hypothesis

2. Most general hypothesis:

If everything is represented with all Question mark, * (or /

then we accept all. (whether it is good or bad we have to accept all) Irrespective of its consequence, we accept all. It is called most general hypothesis.

<?, ?, ?, ?> → accept all

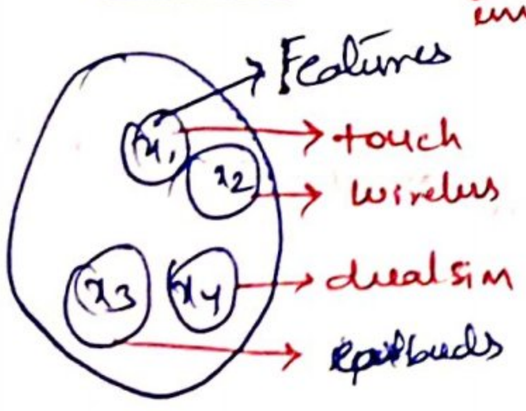
Main Goal:- It finds all concepts / hypothesis that are consistent

so depending on them we can derive a conclusion for the final system. Suppose if you are selecting a concept or a hypothesis which is not consistent, which keeps on always changing, then the conclusion which we have derived done, based on that will also keeps on changing. This is also not consistent. so, all the concepts are hypothesis which are always consistent, we need to select those type of hypothesis or concepts and that is the main goal of concept learning.

Example: what is hypothesis / what is reality

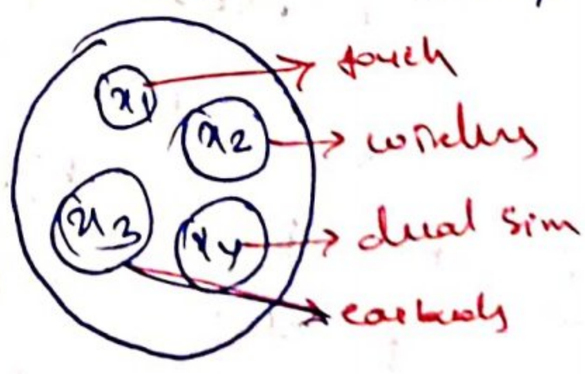
2000's Hypothesis

↓ imagination at 2000's



2020's reality

all possible in 2020



The things which are consistent, they are implemented.

& doesn't have any contradictions then only implemented. (Not all are implemented)

* Concept Learning As Search :-

Main goal of this search is to find the hypotheis that best fit the training Examples

Examples :-

Enjoysport learning task

→ 6 attributes

(sky, Air temperature, humidity, wind, water and forecast)

sky → 3 values → Rainy, cloudy and sunny

Remaining attributes → only 2 values

∴ Different instances possible = $3 \times 2 \times 2 \times 2 \times 2 \times 2 = 96$

Syntactically distinct hypotheis : $5 \times 4 \times 4 \times 4 \times 4 \times 4 = 5120$

(Additionally, 2 more values → ? accepts and ϕ reject)

↓

< ϕ, Rainy, cloudy, sunny, ? >

additional (5) additional

Semantically distinct hypotheis :

(Null - taken as common) $1 + (4 \times 3 \times 3 \times 3 \times 3 \times 3)$

↓ taken as common $= 973$

(ϕ) only 1 more value)

-> After finding all the syntactically & semantically distinct hypotheis, we search the best match from all these (i.e, much closer to our learning problem)

* Find-S Algorithm :-

(most specific hypothesis)

It is finding a maximally specific hypothesis.

→ This algorithm considers only positive examples

Yes ⇒ +ve
No ⇒ -ve

* Representations :

Most specific hypothesis ⇒ ϕ

Most general hypothesis ⇒ ?

Algorithm :

Step 1 :- Initialise with most specific hypothesis (ϕ)

$h_0 = \langle \phi, \phi, \phi, \phi, \phi \rangle$ ⇒ 5 attributes (so, 5 ϕ's)

(no. of attributes)

Step 2 :- for each and every positive sample,
(after selecting a +ve sample, we need to check with each attribute)
for each attribute,

(If -ve sample leave them)

if (value = hypothesis value) ⇒ Ignore
(new value, ϕ)
else

Replace with the most ~~most~~ general hypothesis (?)

(if $\phi = \phi$) → Ignore

Example :-

(attributes, attributes)

	origin	manufacturer	color	year	Type (economy)	class
1	Japan JP	HO → Honda	Blue	1980	eco	+ (Yes)
2	JP	TO → Toyota	Green	1970	spo (sports)	− (No)
3	JP	TO	Blue	1990	eco	+
4	USA	AU	Red	1980	eco	−
5	JP	HO	White	1980	eco	+
6	JP	To	Green	1980	eco	+
7	JP	HO	Red	1980	eco	−

Scanned with OKEN Scanner

step 1 $h_0 = \langle \phi, \phi, \phi, \phi, \phi \rangle$

$h_1 = \langle$ 'JP', 'HO', 'Blue', 1980, 'eco' $\rangle$

↓ +ve values of 1st

$h_2 = h_1$ (∵ h_2 is -ve value) (from table)

$h_3 = \langle JP, ?, Blue, ?, eco \rangle$ (Compare it with previous i.e; h_1)

↓

Because

HO, TO are not same, if not same replace with ?

① ③

$h_4 = h_3$ (∵ -ve)

$h_5 = \langle JP, ?, ?, ?, eco \rangle$

+ve

$h_6 = \langle JP, ?, ?, ?, eco \rangle$ most general hypothesis

+ve

✓

h_7 -ve

Disadvantage:-

→ Considers only +ve values

→ h_6 may not be sole hypothesis that fits the complete data

* Version Space :-

It is the ~~subj~~ subset of hypothesis H consistent with the training examples (D)

specific hypothesis you are picking that hypothesis should be consistent with training examples

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

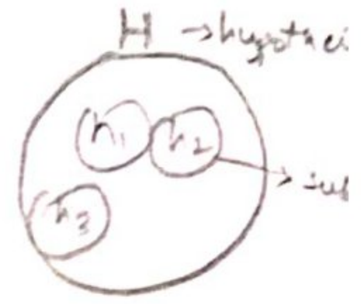
VS → Version Space

H = hypothesis (entire hypothesis)

D = Training Examples

h = The hypothesis which we take as subset (each individual hypothesis represents)

[where h is consistent with D.]



how do you check whether the given hypothesis is consistent with the training example or not?

→ we can check through $h(x) = c(x)$ (h → hypothesis, c → target fn)

If $h(x) = c(x)$, that means, that hypothesis is consistent with the given training Example.

→ Algorithm to obtain version space :-

" List then eliminate Algorithm "

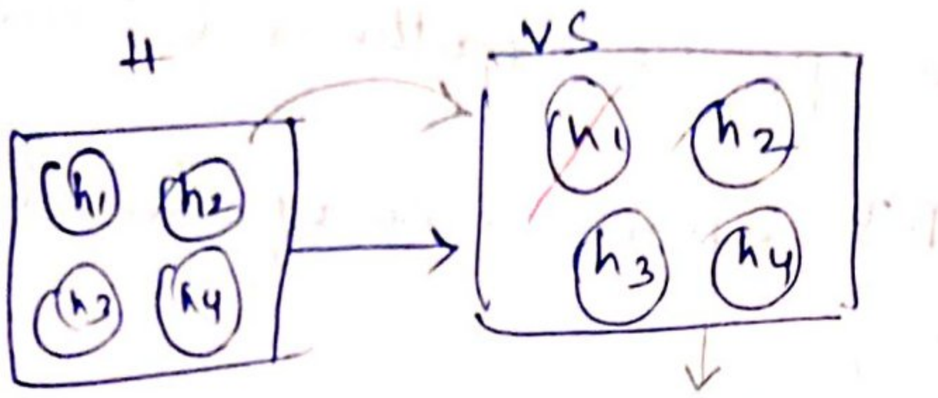
1. Version space → list containing every hypothesis in H (initially, whatever individual hypothesis is there in the superset of hypothesis is assigned to VS irrespective of consistent and inconsistent)

2. from this step, we keep on removing inconsistent hypothesis from version space.

for each training example, $\langle x, c(x) \rangle$ (x → instance, c(x) → target fn) remove any hypothesis that is $h(x) \neq c(x)$ → all inconsistencies

3. output the list of hypothesis into version space after checking for all training examples.

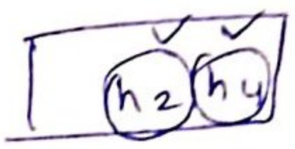
Scanned with OKEN Scanner



→ to check consistency

$h(x) = c(x)$





$h_1(x_1) = c(x_1)$

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

$h_2(x_1) = c(x_1)$

$h_2(x_2) = c(x_2)$

$h_2(x_3) = c(x_3)$

$h_3(x_2) \neq c(x_2)$

$h_4(x_1) = c(x_1)$

$h_4(x_2) = c(x_2)$

$h_4(x_3) = c(x_3)$

h_2, h_4 are consistent

h_1, h_3 are not consistent

* Candidate Elimination Algorithm :-

→ It uses the concept of version space.

→ It considers both +ve and -ve values (yes and no)

→ It considers both specific and general hypothesis

- for positive samples, move from specific (ϕ) to general (?)
- for negative samples, move from general to specific.

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

$G = \{?\ ?\ ?\ ?\ ?\}$ − ↑ G

Scanned with OKEN Scanner

*Algorithm:-

1. Initialise both general and specific hypothesis (s

$S = \{\phi, \phi, \phi, \phi \dots \phi\}$ depends on no. of attribu

$G = \{?, ?, ?, \dots ?\}$

2. for each example,

if example is positive,

make specific to general

else example is negative,

make general to specific

Example:- Enjoy sport

$S_0 = \{\phi\phi\phi\phi\phi\phi\}$ $G_0 = \{??????\}$

1) +ve,

$S_1 = \{$'sunny', 'warm', 'normal', 'strong', 'warm', 'same'

Dataset (Enjoysport)

	① Sky	② Temperature	③ Humidity	④ wind	⑤ water	⑥ forecast → 6 attributes	Enjoy
1.	Sunny	warm	Normal	strong	warm	same	Yes +ve
2.	Sunny	warm	High	strong	warm	same	Yes +ve
3.	Rainy	cold	High	strong	warm	change	No -ve
4.	Sunny	warm	High	strong	cool	change	Yes +ve

1) +ve

$S_1 = \{$'sunny', 'warm', 'normal', 'strong', 'warm', 'same'$\}$

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

2) +ve (specific to general)

compare S_1, S_2 different ?

$S_2 = \{$ sunny, warm, ?, strong, warm, same$\}$

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

3) $S_3 = \{$
-ve (if negative 'S' remains as it is (as above S_2)
(only 'G' changes)

$S_3 = \{$'sunny', 'warm', ?, 'strong', 'warm', 'same'$\}$

Now we are changing general hypothesis so we need to change general to specific ↑ S ↓ G

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

Explanation:
In table see S_3 3 column 1st attribute is 'Rainy' but in S_3 it is 'sunny'. whenever there is change in attribute, write sunny followed by question marks.

Ily, In Next, attribute, 'warm' is changed to cold. so write warm next attributes 'High' 'High' so need not change. no need to write in G_3

Ily 'strong' & 'strong' & warm & warm in 4th, 5th attribute respectively so no need to write

in 6th attribute 'same' is changed to 'change' so write 'same'

4) +ve (specific to general) so compare S_4 & S_3

$S_4 = \{$'sunny', 'warm', ?, 'strong', ? ? > for +ve G remains same

$G_4 = \{$<'sunny' ? ? ? ? ?>, < ? 'warm' ? ? ? ? >$\}$ so copy previous G i.e. G_3 but don't write "same" attribute because in S_4 it is removed.

$\therefore S_4$ & $G_4 \Rightarrow$ final hypothesis

* Inductive Bias :-

→ Remarks on Candidate elimination and version space algorithms :-

1. Will the Candidate Elimination gives us correct hypothesis? (we got one specific & one general hypothesis in candidate; the obtained hypothesis is correct or not; C.E algo. will give correct hypothesis if no error also it defines the language)
2. what training example should the learner request next? (It depends on type of problem and type of question the machine is facing. Also the machine can request on its own)

→ Inductive learning :-

from examples we derive rules
that means learning from examples, learning on the go.

→ Deductive learning :-

Learners are initially exposed to concepts and generalizations, followed by particular examples and exercises to aid learning.

- Already existing rules are applied to the training examples.

* Biased hypothesis space :-

It does not consider all types of training Examples. (Biased → showing difference (or) partiality)

→ solution is include all hypothesis -

Ex :- Sunny ∧ (and) warm ∧ normal ∧ strong ∧ cool ∧ change ⟹ Yes (V → or → NO)

so we need to make it general, so that it will be able to address all type of examples.

for a small variation, also it gives i/p as NO which is not

→ Unbiased hypothesis space :-

Providing a hypothesis capable of representing set of all examples.

possible Instances: $3 \times 2 \times 2 \times 2 \times 2 \times 2 = 96$ (→ sky value is ③)

Target concepts : 2^{96} (huge)

(practically not possible → to learn those many examples
So, we better we not go for Unbiased hypothesis

→ Idea of Inductive Bias :-

Learner generalizes beyond the observed training examples to infer new examples. (based on the examples that have given)

Example

`>' → inductively inferred from

$x > y \Rightarrow y$ is inductively inferred from x.

↓
x is already predefined in the system
it is giving o/p for y

Ex :- learning algorithm → L

training data $D_c = \{x, c(x)\}$

New instance = x_i;

Represented as $L(x_i, D_c)$

$(D_c \wedge x_i) > L(\vec{x_i}, D_c)$ (→ new instance; → training data)

↓ Predifined ↓ learning algorithm

$L(x_i, D_c)$ is inductively inferred from $(D_c \wedge x_i)$

↓ predefined

* Decision Tree Learning:-

It is used in; tree structured classification

Regression

Dataset → Algorithm → classifies the data

↓ raw data

(Decision tree algorithm)

means when we are giving the data to the classifier, (it can be any type of classifier like naive based classification, decisiontree classifier etc)

→ It will say to which class the data belongs to like 'yes class or No class or +ve/-ve (based on content (Example it will tell us to which class the data belongs to)

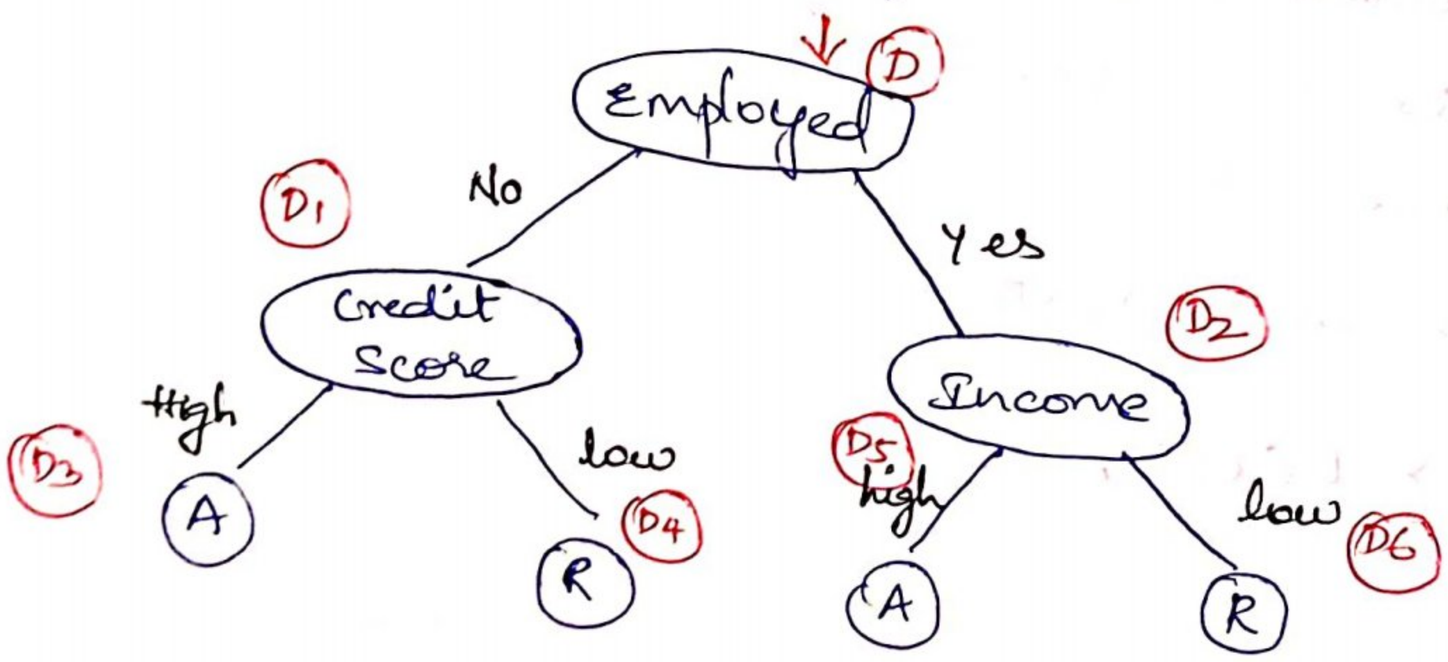
It will divide the data into subcategories

2 types of Nodes:-

1. Decision Node: It is a root node
   It will take all the decisions (from where the branching is begin)
2. Leaf Node:-
   They are the last row. we cannot branch them out.

Example:- Loan system

It decides if the loan should be approved / Rejected,



→ first we dump entire data into the decision node 'D' ie; root node
Here, employed will check (for each & every individual person) whether that person is employed or not, if employed it goes to Income side. If he is not employed it goes to Creditscore side.
(Ily for all persons)

→ After that it classifies employed people and unemployed people

→ All the employed people will come to D2 (Dataset 2) ie, Income

→ All the unemployed people will go to Gr. D1 Dataset 1 ie, credit score

→ If credit score is high it will go for D3

" " " " low " " " " D4

→ In Income, D2 if Income is High it will go for D5

" " " low " " " D6

→ so finally we got D3, D4, D5 & D6 datasets.

→ for whom we have to approve the loan and for whom we have to reject the loan?

→ If credit score is high then you can approve ie; D3 dataset is approved

" " " low " " " Reject D4 dataset

→ for Income if Income is high, you can approve D5 dataset

" " " low, " " Reject D6 dataset

Therefore based on some criteria they are dividing the people into two categories to approve loan for those people or to Reject the loan for those people.

→ they are going to classify based on the decision tree.

→ Algorithm (ID3) → Iterative Dichotomiser 3 (because algorithm repeatedly divides)

1. Invented by Ross Quinlan, used to generate decision tree from a dataset.

1. In the given dataset, choose a target attribute
2. Calculate Information gain of target attribute

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

3 for remaining attributes, find entropy

$$\text{Entropy} = IG \times \text{Probability}$$

↓ How many times it has occur

x is a counter

$$E(A) = \sum \frac{P_i + N_i}{P+N} I(P_i N_i)$$

4. calculate $\text{Gain} = IG - E(A)$

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

→ 2 downs in mid

→ 2 ups in mid

→ 3 ups

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 → 5 down's in table

N = count (up) = 5 → 5 up's in table

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

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

$$= -\left(\frac{1}{2} \times (-1)\log_2 2 + \frac{1}{2} \times (-1)\log_2 2\right) \quad (\because \log_n a^m = m\log a)$$

$$= -\left(\frac{1}{2} \times -1 + \frac{1}{2} \times -1\right) \quad (\because \log_2 2 = 1)$$

$$= -(-1) = 1$$

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

$\therefore \boxed{IG = 1}$

step 3: calculate Entropy for remaining attributes.

$$E(A) = \sum \frac{P_i + N_i}{P+N} \cdot \vec{I}(P_i N_i)$$ → instead of this go for this (i.e. same only)

i.e. $\boxed{IG \times \text{Probability}}$

(Age :-)

(1) Prepare a table for each attribute

rows - values of undertaken attribute

(old, mid, new)

columns - values of target attribute

(down, up)

	down	up
old	3 (P)	0 (N)
mid	2 (P)	2 (N)
new	0 (P)	3 (N)

Entropy = Σ IG * Probability

P = down count

N = up count

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

probability = $\frac{3}{10}$ → 3 times old is there

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

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

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

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

probability = $\frac{4}{10}$ ($\because$ mids = 4)

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

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

probability = $\frac{3}{10}$

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

$\therefore$ Entropy (Age) = E(old) + E(mid) + E(new) :

= 0 + 0.4 + 0 = 0.4

∴ Entropy (Age) = 0.4

Step 4:- Gain = IG − E(A)

= 1 − 0.4

= 0.6

=

In the same way, calculate gain of other attributes

Gain (competetion) = 0.124

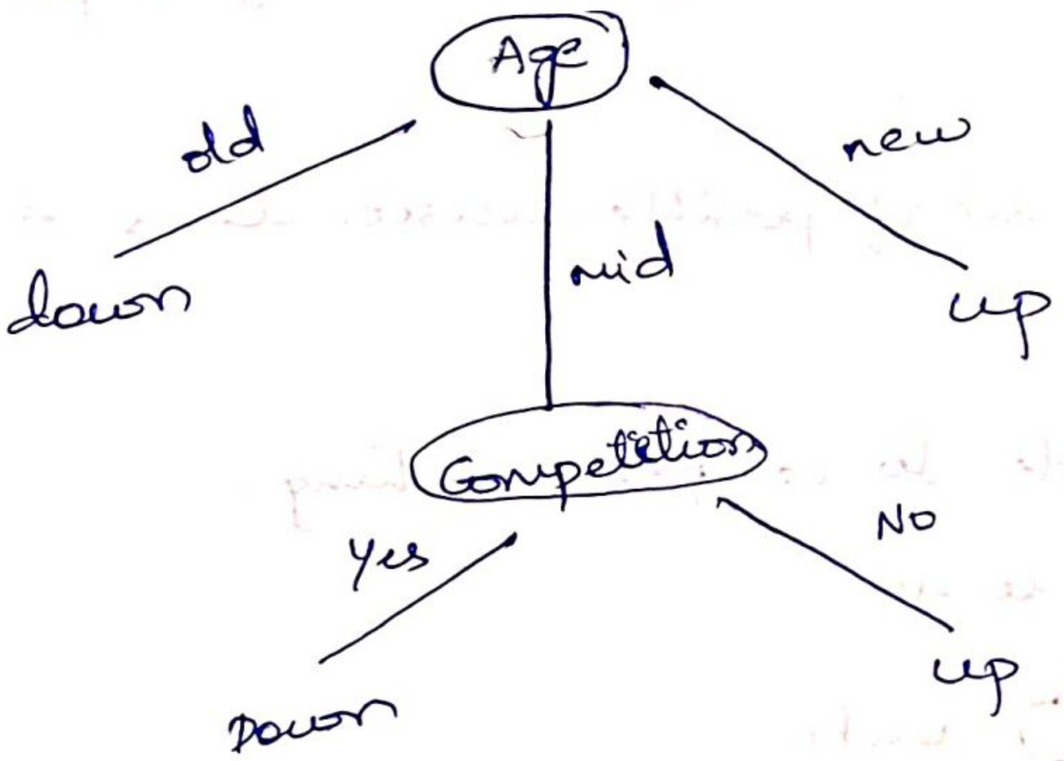
Gain (Type) = 0

Gain (Age) = 0.6

Now construct decision tree.

Highest gain → root node

Here highest gain is Age ie: 0.6.



Here we took competetion because next highest value is competetion ie: 0.124

old → all down

mid → some down and some up

new → all up

why competetion → 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

Decision Tree algorithm is best suited to problems with the following characteristics

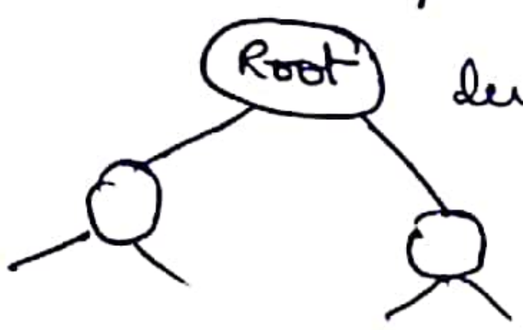
1. Instances are represented by attribute value pairs
2. The target function has discrete output values
3. Training data can have errors
4. May contain missing attribute value also.

* Hypothesis space search in DecisionTree learning:-

→ ID_3 can be characterised as searching a space of hypothesis for one that fits the training examples.

→ ID_3 → It will search set of possible decision trees from available hypothesis

→ ID_3 performs simple to complex searching.



simple → complex → moving levels 2, 3 ...

→ First, start with empty tree and keep on adding

→ Every discrete valued (finite) function can be described by some decision tree

→ It will avoid major risk of searching incomplete hypothesis

→ It has only single current hypothesis (will not have (No backtracking) alternate DTs)

→ It cannot determine alternative decision trees

→ Backtracking is not possible (Eg: in previous example, 'age' is root but if we want to change root as 'competition', it is not allowed)

→ It can be extended easily to noisy data also.

noisydata → if there are any inconsistencies in data (or) any redundancy in data is called noisydata

→ It will work even if noisydata is there

* Inductive Bias in Decision Tree Algorithm :-

ID3 → Iterative Dichotomiser 3 (repeatedly, divider)
Invented by Ross Quinlan, used to generate D.T

→ Inductive bias of ID3 (D.T.D) consists of describing the bais by which ID3 chooses one consistent decision tree over all the possible DT's.

Based on what 'criteria', on set of all DT's we are going to select one DT. that will tell us Inductive bias

ID3 Search Strategy:-

1. selects in favour of shorter trees over longer ones
   (It always search for shorter tree with all training examples)
2. selects element with highest IG as root attribute over lowest IG ones.
   (IG → Information Gain (Eg: previous example))

→ Types of Inductive Bias:-

1. Restrictive bias - based on conditions
2. Preference bias - based on priorities

ID3 ⇒ Preference ; Versionspace & Candidate Elimination ⇒ Restrictive

Preference

Because ID3 will search for entire hypothesis and it will give priority to shorter trees & IG is higher (So preference)

Restrictive

Because they are searching based on some conditions. like we need to take +ve values, if (-ve other) like that we are having some conditions so, it is Restrictive.

→ PTO

→ why short hypothesis ?

According to <u>occums Razor</u>

↓

occums Razor statement is, we need to prefer simplest Hypothesis that will fit the data

* Issues in Decision Tree Learning :-

1. Overfitting the data (we completely relying on training data then overfit occurs and testing data it doesn't work good for overfitting)
   - ↳ to overcome overfitting, we have techniques 1. Reduced error pruning ② Post rule pruning
   - overfitting works good for training data
2. Incorporating continuous valued attributes
   - ↳ (convert to discrete)
3. Determining depth of tree for correct classification
   - like levels level 1, level 2 we cannot determine
4. Handling attributes with different costs
   - ↳ importance
   - giving importance to the attribute, and that may not influence them,
5. Alternative measures for selecting attributes. it will not cover all Examples so the hypothesis or D.T will not support
   - ↳ importance to target attribute (biasing)