1) Define Machine learning. Discuss with examples some applications of machine learning.

Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy.

Types:

1) Supervised learning:

It is when the model is getting trained on a labelled dataset. A labelled dataset is one that has both input and output parameters.

2 catagories → Classification, Regression

Eg: Teaching a baby what a dog and a cat is. by telling the different characteristics of dog and cat.

Here the one who is teaching is acted as supervisor and the baby acted as algorithm that had to learn

A basket full of different kinds of fruits.

2) Unsupervised learning:

It's a learning algorithm which is trained using unlabeled data. It does not need supervision. it finds the patterns from the i/p data by its own

2 types of problems → Clustering & Association

Eg: ~~Asha~~ Suppose a baby knows and identifies her family dog. Few weeks later a family friend brings along a dog ~~and~~. Baby has not seen this dog earlier. But it recognizes many features (2 ears, eyes, & legs) are like her pet dog. She identifies the new animal as a dog.

3) Semisupervised learning:

It is an approach to machine learning that combines a small amount of labeled data with a large amount of unlabeled data during training.

Eg: Web content classification is very critical & impossible to label each page on the internet because it needs more human intervention. This problem can be reduced through semi supervised learning algorithms.

It is a feedback based ML technique in which an agent learns automatically using feedbacks without any labeled data. Agent is bound to learn by its experience only. For each good action the agent gets positive feedback, and for each bad action, the agent gets negative feedback or penalty.

Eg: Suppose a master is training a dog to get the stick Each time the dog gets a stick successfully, master offered him a feast. Eventually, the dog understands the pattern, that whenever the master throws a stick it should get it as early as it can to gain a reward from a master.

Applications:

1) Speech Recognition → It is a process of converting voice instructions into text.
   Eg: Google assistant, Siri, Cortana & Alexa

2) Traffic prediction → It predicts the traffic conditions such as whether traffic is cleared, slow moving or heavily congested
   It shows us the correct path with the shortest route and predicts the traffic conditions.

3) Product recommendations → Google understands the user interest using various machine learning algorithm and suggests the product as per customer interest

4) Self driving cars → Tesla the most popular car manufacturing company is working on self driving cars. It is using unsupervised learning method to train the car models to detect people & objects while driving.

5) Medical diagnosis → In medical science, machine learning is used for diseases diagnoses.
   It helps in finding brain tumors and other brain related diseases easily.

2) What do you mean by well posed learning problem. Explain the important features that are required to define well posed 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 improve E (knowing by experience).

A computer program is said to learn from experience E with respect some class of tasks T and performance measure P, if its performance at tasks in T as measured by P improves with experience E.

Three features that are required to define well posed learning problem.

1) The class of tasks
2) The measure of performance to be improved
3) The source of experience.

Well defined learning problem

1) A checkers learning problem

Task T: playing checkers

Performance measure P: percent of games won against opponents

Training experience E: playing practice games against itself

2) A handwriting recognition learning problem

Task T: recognizing and classifying handwritten words within images

Performance measure P: percent of words correctly classified.

Training experience E: a database of handwritten words with given classification.

3) A robot driving learning problem

Task T: driving on public four lane highways using vision sensors

Performance measure P: Average distance travelled before an error

Training experience E: a sequence of images and steering commands recorded while observing a human driver.

3. Describe briefly the steps involved in designing a learning system.

To get a succesful learning system it should be designed for a proper design several steps should be followed

why? → for perfect & efficient system.

Steps: for designing learning system

1) Choosing Training experience
2) Choosing target function
3) Choosing representation of target function
4) Choosing function approximation
5) Final design.

STEP 1

1) Choosing training experience:

The first design choice is to choose the type of training experience (data) from which the system will learn.

The type of training experience available can have a significant impact on success (or) failure of the learner

In choosing a training experience, 3 attributes are taken

1) Type of feedback
2) Degree
3) Distribution of examples.

1) whether the training experience provides direct or indirect feedback regarding the choices made by performace system.

Task → (P) good → good (E)

Inordered to perform good in task, we (would) have good (E)

By-training we can get good experience.

Direct Indirect

1) Direct feedback:

In checkers game when we make a move, immediately computer will tell ~~us~~ their whether you made a correct move or wrong move.

(Immediately after 1 move)

learning driving:

while learning driving trainer will tell everything accelerator, breaker.

2) Indirect feedback:

While learning driving trainer won't be physically besides you, some recorded sessions will be there. It will not ~~spontaneously~~ simultaneously telling you. (feedbacks are given in intervals)

2) Degree to which learners will control the sequence of training.

Ex: learning driving

- with trainers help
- with trainers 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 combination

→ more situations, more examples

→ learning over distributed range of examples

Ex: learning driving
- Trainer → single, 2way, 4way, hill curve road, road,
- Without T → ..

STEP 2: Choosing the target function

To determine what type of knowledge will be learned and how it will be used by the performace system

Ex: Checkers game.

→ Travel only in forward direction

→ Only one move per chance & only in diagonal direction

→ Jump over opponent.

→ While moving diagonally set of all possible moves is called legal moves.

→ we need to select one move & that move is called target move.

Target function ⇒ V(b)

Board state ⇒ b (Initial, intermediary or final state).

Legal moves set ⇒ B

Let us define target value V(b) for an arbitrary board state b in B as follows:

1) If b is a final board state that is won, then V(b) = 100

2) if b is a final board state that is lost, then V(b) = -100
3) if b is a final board state that is drawn, then V(b) = 0
4) If b is a not a final state in the game, then V(b) = V(b')
where b' is the best final board state that can be achieved starting from b and playing optimally until the end of the game.

STEP 3: Choosing the representation for target function.

For any board state we calculate function c as linear combination of following board features i.e c(b)

Features:

x_1 → No. of black pieces on the board
x_2 → No of red pieces on the board
x_3 → No. of black kings on the board
x_4 → No. of red kings on the board
x_5 → No. of black pieces threatened by red (black's which are beaten by red)
x_6 → No. of red pieces threatened by black.

$$\hat{V}(b) = w_0 + w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + w_5x_5 + w_6x_6$$

(representation of c(b))

w_0 to w_6 → Numerical coefficient or weights or weight of each feature
w_0 → additive constant.

STEP 4: Choosing function approximation algorithm

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

Describes a particular board state (b) & training value Vtrain(b).

Training example representation is

ordered pair = (b, Vtrain(b))

Example: Black won the game

i.e $x_2 = 0$ which means no red.

Vtrain(b) = +100.

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

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

1) Estimating traing values:

In every step we consider successor depending on the next step of opponent

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

↓

represents the next board state

(estimating that this move will help/destroy opponent)

↑ → represents approximation

2) Adjusting the weights:

There are some algorithm to find weights of linear functions here we are using LMS (least mean Square).

LMS is used to minimise the error.

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

$E \equiv \sum_{(b, V_{train}(b)) \in Training\ examples} (V_{train}(b) - \hat{V}(b))^2$

LMS weight update rule:

For each training example $(b, V_{train}(b))$, use the current weights to calculate $\hat{V}(b)$. For each weight w_i, update it as $w_i \leftarrow w_i + \eta (V_{train}(b) - \hat{V}(b)) x_i$

Here η is a small constant that moderates the size of the weight update.

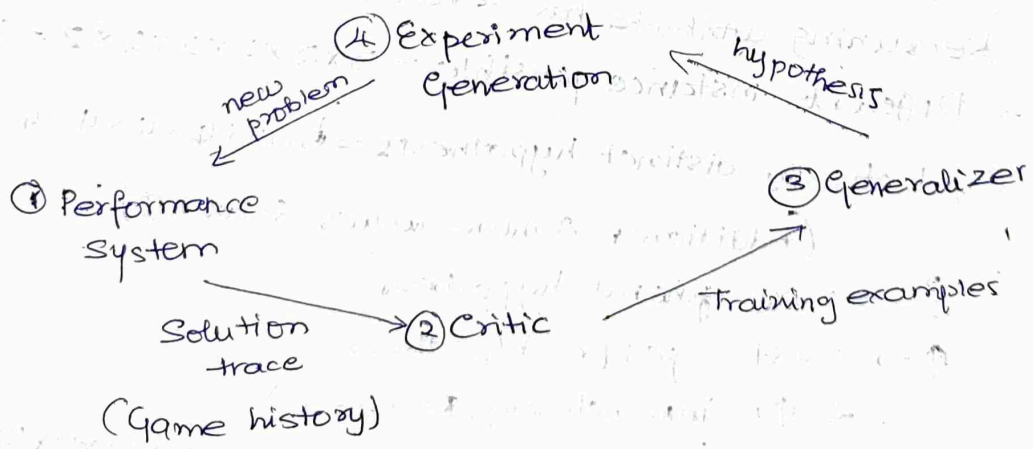
If Error = 0, no need to change weights

If Error is positive, each weight is increased in proportion

If error is negative each weight is decreased in proportion

STEP 5: Final design

It has 4 different modules



▷ Performance system:

It takes an instance of a new problem (new game) as input and produces a trace of its solution as output.

3) The critic:

It takes as input the history or trace of the game and produces as output a set of training examples of the target function.

3) The Generalizer:

It takes the training examples as input and produces an output hypothesis that is its estimate of the target function.

4) The Experiment Generator:

It takes the current hypothesis as input and new problem as output for the performance system to explore.

4) Discuss the concept of learning as the task of searching with respect to generate to specific ordering of hypothesis.

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

Ex: Enjoy sport learning task.

→ 6 attributes (warm, cold) (normal, high) (strong, weak) (warm, cool) (same, change)

(Sky, Air temperature, humidity, wind, water & forecast)

Sky has 3 values → Rainy, cloudy & Sunny.

Remaining attribute has 2 values

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

Syntactically distinct hypothesis = $5 \times 4 \times 4 \times 4 \times 4 \times 4 = 5120$

(Additional 2 more values ? and ϕ)

? → most general hypothesis

ϕ → most specific hypothesis.

∠ ϕ, Sunny, cloudy, Rainy, ?)

Semantically distinct hypothesis = $1 + (4 \times 3 \times 3 \times 3 \times 3 \times 3)$

$= 973$

(Null taken as common, (ϕ) only 1 more value)

After finding all the syntactically & semantically distinct hypotheis.

We search the best match from all these

(i.e mch closer to our learning problem)

5. Illustrate find S algorithm over enjoysport training instance given.

Find S Algorithm:

finding a maximally specific hypothesis

→ This algorithm considers only positive examples.

* Representation:

most specific hypothesis ⇒ ∅

most general hypothesis ⇒ ?

Algorithm:

Step1: Initialise with most specific hypothesis (∅)

$h_0 = \langle \emptyset, \emptyset, \emptyset, \emptyset, \emptyset \rangle$ ⇒ 5 attributes.

Step2: For each +ve samples, for each attribute

if (value = hypothesis value)

⇒ Igone

else

Replace with most general hypothesis (?)

Step3: Output the hypothesis h

Example	Sky	Airtemp	Humidity	Wind	Water	Fore cast	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

1. Initialise h to the most specific hypothesis in H

$h_0 = \langle \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset \rangle$

$x_1 = \langle$ Sunny, Warm, Normal, Strong, Warm, Same$\rangle$

$h_1 = \langle$ Sunny, Warm, Normal, Strong, Warm, Same$\rangle$

$x_2 = \langle$ Sunny, Warm, High, Strong, Warm, Same$\rangle$

$h_2 = \langle$ Sunny, Warm, ?, Strong, Warm, Same$\rangle$

$x_3 = \langle$ Rainy, Cold, High, Strong, Warm, Change$\rangle$

No, so ignore.

$h_3 = \langle$ Sunny, Warm, ?, Strong, Warm, Same$\rangle$

$x_4 = \langle$Sunny, Warm, High, Strong, Cool, Change$\rangle$

$h_4 = \langle$Sunny, Warm, ?, Strong, ?, ?$\rangle \Rightarrow$ Output

6. Define consistent hypothesis and version space. With example explain version space & its representation

Consistent hypothesis:

An hypothesis h is consistent with a set of training examples D iff $h(x) = c(x)$ for each example in D

$$\text{Consistent}(h, D) \equiv (\forall \langle x, c(x)\rangle \in D)\ h(x) = c(x)$$

($c(x)$: Target function.)

Version space VS:

Subset of hypothesis H consistent with the training examples(D)

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

H = hypothesis

D = training example

Consistent: $h(x) = c(x)$

Ex: Enjoy sport

Initialise G_0 and S_0

$G_0 = \{\langle ? ? ? ? ? ? \rangle\}$

$S_0 = \{\langle \phi\ \phi\ \phi\ \phi\ \phi\ \phi \rangle\}$

For $x_1 = \langle \ldots \rangle$ +ve Generalise S_0 to cover x_1

$S_1 = \langle$Su, Wa, #, St, Wa, Sa$\rangle$

$G_1 = \langle ? ? ? ? ? ? \rangle$

For x_2 $\langle \ldots \rangle$ +ve replace attribute value by ? to make it consistent with x_2

$S_2 = \langle \langle$Su, Wa, ?, St, Wa, Sa$\rangle\}$

$G_2 = \{\langle ? ? ? ? ? ? \rangle\}$

For x_3 $\langle$Ra, $\ldots\rangle$ −ve

As x_3 −ve traing example general boundary incorrectly predicts x_3 is +ve

So G_2 needs to be specialised to make it classify as −ve correctly.

$S_2 = S_3 = \{\langle$Su, Wa $\ldots\rangle\}$

$G_3 = \{\langle$Su, ? ? ? ? ?$\rangle, \langle ?$ Wa ? ? ? ?$\rangle \langle ? ? ? ? ?$ Sa$\rangle\}$

For x_4 $\langle \ldots \rangle$ +ve, Fourth training example need to be generalized to be consistent $S_4 = \langle$Su, Wa, ?, St, ? ?$\rangle$

Also G_3 needs to be updated to make consistent with the x_4 $G_4 = \{\langle$Su, ? ? ? ? ?$\rangle \langle ?$ Wa ? ? ? ?$\rangle\}$

(Final VS)

7) Describe & Explain the candidate elimination algorithm with illustration

→ Uses the concept of version space
→ Consider both +ve & -ve value (Yes and No)
→ both specific and general hypothesis
→ For specific +ve samples, move from specific to general
→ For -ve samples move from general to specific

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

$G_1 = \{?, ?, ?, ?, ?\}$ ↑

Algorithm:

Step 1: Initialise both general & specific hypothesis.

$S = \{\phi, \phi, \phi \ldots \phi\}$

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

Step 2: For each example,

if example is +ve
make specific to general

else
make general to specific.

Ex: Enjoy sport.

$S_0 = \{\phi, \phi, \phi, \phi, \phi, \phi\}$ $G_{10} = \{?, ?, ?, ?, ?, ?\}$

1) +ve

$S_1 = \{$Sunny, Warm, Normal, Strong, Warm, Same$\}$

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

2) +ve (specific to general) Only change S not G.

$S_2 = \{$Sunny, warm, ?, strong, Warm, Same$\}$

S_1 = Normal
x_2 = High

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

3) -ve (general to specific) keeps as it is. Only change G

$S_3 = \{$Sunny, Warm, ?, Strong, Warm, Same$\}$

x_1 = Sunn, x_2 = Wam, x_2 = High
x_3 = Rainy, x_3 = Cold, x_3 = High
} no need to write

↓

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

4) +ve S3 & S4 — no comparison

$S_4 = \{ < Sunny, Warm, ?, Strong, ?, ? > \}$

$G_4 = \{ < Sunny, ?, ?, ?, ?, ? >, < ?, Warm, ?, ?, ?, ? > \}$

Don't include <? ? ? ? same> Because in S4 there is no same

S_4 and G_4 ⇒ final hypothesis.

8. Consider the given below training example which finds Malignant tumours from the MRI scans.

Example	Shape	Size	Color	Surface	Thickness	Target concept
1	Circular	Large	Light	Smooth	Thick	Malignant
2	Circular	Large	Light	Irregular	Thick	Malignant
3	Oval	Large	Dark	Smooth	Thin	Benign
4	Oval	Large	Light	Irregular	Thick	Malignant
5	Circular	Small	Light	Smooth	Thick	Benign.

Show the specific and general boundaries of the version space after applying candidate elimination algorithm (Note: Malignant = +ve & Benign = -ve)

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

1) +ve

$S_1 = \{ < Circular, large, Light, Smooth, Thick > \}$

$G_{11} = \{ < ?, ?, ?, ?, ?, ? > \}$

2) +ve (S → G)

$S_2 = \{ < Circular, Large, Light, ?, Thick > \}$

$G_{12} = \{ < ?, ?, ?, ?, ?, ? > \}$

3) −ve (G → S)

$S_3 = \{ < Circular, Large, Light, ?, Thick > \}$ <?, S, ?, ?, > ✗

$G_{13} = \{ < Circular, ?, ?, ?, ?, ? > < $~~Light~~ ? ? Light ? ? > $< ~~? ? ? Irregular~~ ? ? ? ? > < ? ? ? ? Thick > \}$ ✗

Check consis tent till S3

4) +ve (S → G)

$S_4 = \{ < ?, Large, Light, ?, Thick > \}$

no circle so

$G_{14} = \{ < ?, ?, Light, ?, ? > < ?, ?, ?, ?, Thick > \}$

5) -ve

$S_5 = \{ < ?, large, light, ?, thick > \}$

$G_{15} = \{ < ?, large, light, ?, ? > < ?, large, ?, ?, Thick > \}$

NOTE : Final version space for the enjoy Sport concept learning problem and training examples.

S_4 : $\{ < Sunny, Warm, ?, Strong, ?, ? > \}$

S4 & G4 ①

<Sunny, ?, ?, Strong, ? ?> <Sunny, Warm, ? ? ? ?> < ? Warm ? Strong ? ?>

G_4 : $\{ < Sunny, ?, ?, ?, ?, ? > < ?, warm, ?, ?, ?, ? > \}$

① ②

If mismatch → replace with ? and copy remaining of S_4 with specific hypothesis of copy remaining of G_4

9. Describe ID3 algorithm. Calculate entropy and information gain for the following dataset.

Instance	Classification	A1	A2
1	+	T	T
2	+	T	T
3	−	T	F
4	+	F	F
5	−	F	T
6	−	F	T

ID3 standards for iterative Dichotomiser 3.
Used to generate a decision tree from a given dataset by employing a top-down greedy search to test each attribute at every node of the tree.
The resulting tree is used to classify future samples.

Algorithm:
ID3 (Examples, Target_attribute, Attributes)
Examples are the training examples. Target_attributes is the attribute whose value is to be predicted by the tree. Attributes is a list of other attributes.
Returns a decision tree that correctly classifies the given Examples.

1) Create a root node for the tree

2) If all Examples are positive, return the single-node tree root, with label = +

3) If all Examples are -ve, return the single-node tree root, with label = −

4) If attributes is empty, return the single-node tree root = most common value of target-attribute in Examples.

5) Otherwise Begin

i) A ← the attribute from attributes that best * classifies examples.

ii) The decision attribute for root ← A

iii) For each possible value v_i of A

a) Add a new tree branch below root corresponding to the test A = v_i

b) Let examples v_i, be the subset of examples that have value v_i for A

c) If Examples v_i is empty

Then below this new branch add a leaf node with label = most common value of target-attribute in Examples

Else below this new branch add the subtree ID3 (Examples$_{v_i}$, Target-attribute, Attributes − {A})

6) End

7) Return root

Entropy → measures the impurity of a collection of examples.

Information gain → that measures how well a given attribute seperates the training examples according to their target classification.

$$\text{Entropy}(S) = -P_{\oplus}\log_2 P_{\oplus} - P_{\ominus}\log_2 P_{\ominus}$$

where P_+ is the proportion of +ve examples in S

P_- is the proportion of -ve examples in S

Information gain = $\text{Gain}(S, A) = \text{Entropy}(S) - \sum \left[P\left(\frac{S_v}{S}\right) \text{Entropy}(S_v) \right]$

A → attribute

Gain (S, A) of an attribute A, relative to a collection of examples S.

Problem solution:

$\log_2 x = \frac{\log_{10} x}{\log 2}$

Attribute : a1

Values (a1) = T, F ⟨ T → 3, F → 3

S → +, 3 ; −, 3

Entropy (S) = $-P_{\oplus}\log_2 P_{\oplus} - P_{\ominus}\log_2 P_{\ominus}$

$= -\frac{3}{6}\log_2\frac{3}{6} - \frac{3}{6}\log_2\frac{3}{6}$

$= \frac{3}{6}\times(-1) - \frac{3}{6}\times(-1) = -\frac{2}{6}\times(-0.5849) - \frac{2}{6}(-1.5849)$

$= 1 = 0.9183 = -\frac{3}{6}\log_2\frac{3}{6} - \frac{3}{6}\log_2\frac{3}{6} = 1$

S_T ⟨ +→2, −→1

Entropy (S_T) = $-\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3}$

$= -\frac{2}{3}\times(-0.58496) - \frac{1}{3}\times(-1.58496)$

$= 0.9183$

S_F ⟨ +→1, −→2

Entropy (S_F) = $-\frac{1}{3}\log_2\frac{1}{2} - \frac{2}{3}\log_2\frac{2}{3}$

$= 0.9183$

Gain of a1 :

Gain (S, a1) = Entropy (S) − $\sum_{V\in(T,F)} \frac{|S_V|}{|S|}$ Entropy (S_V)

Gain (S, a1) = $1 - \frac{3}{6}$ Entropy (S_T) − $\frac{3}{6}$ Entropy (S_F)

$= 1 - \frac{3}{6}\times 0.9183 - \frac{3}{6}\times 0.9183$

$= 0.0817$

Attribute: a2

Values : (a2) = T, F ⟨ T → 4, F → 2

S ⟨ +→3, −→3

Entropy (S) = $-\frac{3}{6}\log_2\frac{3}{6} - \frac{3}{6}\log_2\frac{3}{6}$

$= 1$

S_T ⟨ +→2, −→2

Entropy (S_T) = $-\frac{2}{4}\log_2\frac{2}{4} - \frac{2}{4}\log\frac{3}{4} = 1.0$

S_F < + → 1
− → 1

Entropy $(S_F) = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2}$

$= 1.0$

Gain of a2:

$$Gain(S, a_2) = Entropy(S) - \sum_{v \in T,F} \frac{|S_v|}{|S|} Entropy(S_v)$$

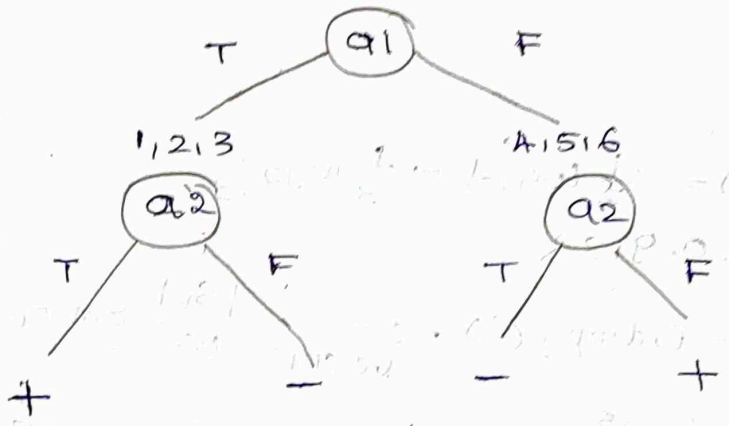
$$Gain(S, a_2) = 1 - \frac{4}{6} Entropy(S_T) - \frac{2}{6} Entropy(S_F)$$

$= 1 - \frac{4}{6} \times 1.0 - \frac{2}{6} \times 1.0$

$= 0.0$

Gain(S, a1) = 0.0817 → Maximum Gain

Gain(S, a2) = 0.0



1) Describe inductive bias in decision tree learning.

Inductive bias is the set of assumptions that, together with the training data, deductively justify the classifications assigned by the learner to future instances.

Inductive bias of ID3 consists of describing the basis by which ID_3 chooses one consistent decision tree over all the possible decision trees

ID_3 search strategy:

1) Selects in favour of shorter trees over longer ones
2) Selects trees that place the attribute with highest information gain closest to the root

Types of inductive bias

1) Restrictive bias → based on conditions
2) Preference bias → based on priorities

ID3 ⇒ Preference

Version space and candidate-elimination ⇒ Restrictive

Why short hypothesis?

According to occums Razor, prefer simplest hypothesis that fits the data.

Difference b/w the types of inductive bias exhibited by ID3 and by the CANDIDATE-ELIMINATION algorithm.

ID3:

→ The inductive bias of ID3 is a preference bias for certain hypotheses over others, with no hard restriction on the hypotheses that can be eventually enumerated. This form of bias is called a preference bias or a search bias.

→ ID3 searches a complete hypothesis space. → set of possible DT's

→ It searches incompletely through this space from simple to complex hypotheses until its termination condition is met

CANDIDATE-ELIMINATION

→ The bias of the CE algorithm is in the form of a categorial restrictions on the set of hypotheses considered.

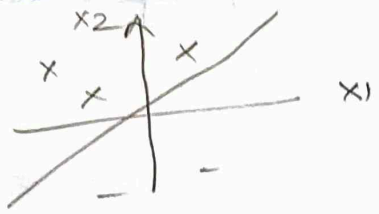
→ The version space CE searches an incomplete hypothesis space.

→ It searches this space completely finding every hypothesis consistent with the training data.

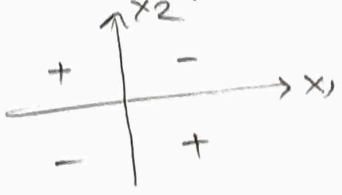
(1) What is linearly seperable problem. Design a network of perceptron to implement x AND y.

Linearly inseperable means if there's a function that is not linearly seperable it by definition means that there's no line such that points of one type are one side and the other type on the other side.

i.e there exists no line to seperate the points.



linearly seperable

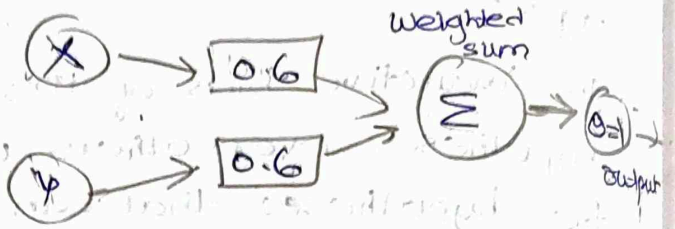


linearly non seperable

Decision boundaries could have linear or non linear form
If it is not possible to have straight line separating the regions then, the patterns are said to be linearly Non-Seperable.
Such patterns then may be seperated by decision boundaries that are non-linear

logical AND operation

X	Y	X ∩ Y
0	0	0
0	1	0
1	0	0
1	1	1



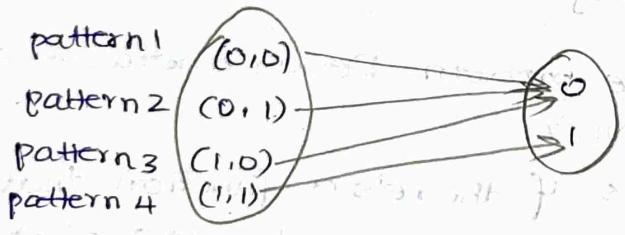
If X=0 & Y=0 → 0*0.6 + 0*0.6 = 0
This is not greater than the threshold of 1 so O/p = 0

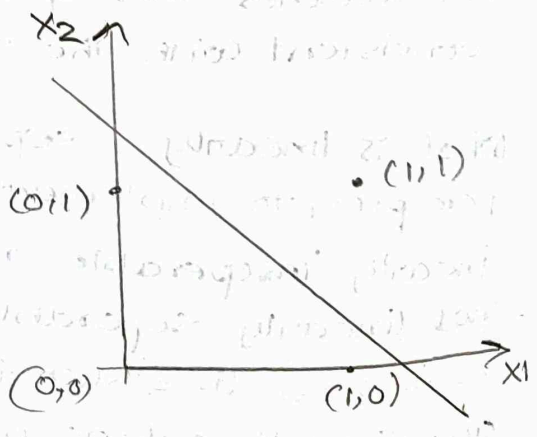
If X=0 & Y=1 → 0*0.6 + 1*0.6 = 0.6
This is not greater than the threshold of 1 so O/p = 0

If X=1 & Y=0 → 0.6

—»—

If X=1 & Y=1 → 1*0.6 + 1*0.6 = 1.2
This exceeds the threshold so O/p = 1





linearly seperable

12) What is perceptron? Discuss perceptron training rule.

Perceptron is a single layer neural network.

A perceptron takes a vector of real-valued inputes, calculates a linear combination of these inputs, then outputs a 1 if the result is greater than some threshold and -1 otherwise.

Perceptron training rule:

The learning problem is to determine a weight vector that causes the perceptron to produce the correct +1 or -1 output for each of the ~~following~~ training examples.

To learn an acceptable weight vector:

1) Begin with random weights then iteratively apply the perceptron to each training example modifying the perceptron weights whenever it misclassifies an example.

2) This process is repeated iterating through the training examples as many time as needed until the perceptron classifies all training examples correctly.

3) Weights are modified at each step according to the perceptron training rule which revices the weight w_i associated with input x_i according to the rule.

$$w_i \leftarrow w_i + \Delta w_i$$

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

Here t is the target output for the current training example

o is the output generated by the perceptron.

η is a positive constant called the learning rate.

4) The role of the learning rate is to moderate the degree to which weights are changed at each step. It is usually set to some small value and is sometimes made to decay as the number of weight tunning iterations increases.

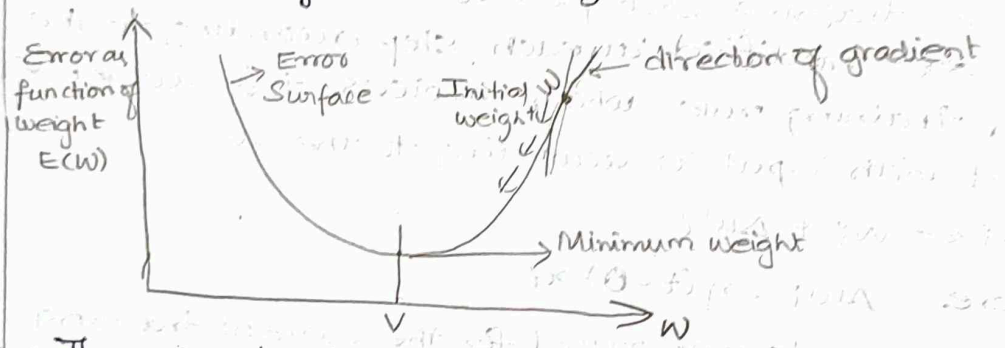
DRAWBACK:

The perception rule finds a successful weight vector when the training examples are linearly seperable, it can fail to converge if the examples are not linearly seperable.

13. What do you mean by gradient descent? What are the conditions in which Gradient descent is applied?

Gradient Descent is an optimization algorithm for finding a local minimum of a differentiable function.
Gradient descent is simply used in machine learning to find the values of a function's parameters that minimize a cost function as far as possible.
It is used to determine how much a weight should be changed.

At every stage of the computation, the error E is a function of the weight. If we plot the error against the weights we get a higher dimensional analog of something like a curve. At any point on this surface the gradient suggests how steeply the error will be reduced / increased for a change in the weights.



The gradient specifies the direction of steepest ↑ in E
the training rule for gradient descent is

$$\vec{w} \leftarrow \vec{w} + \Delta\vec{w} \quad \text{where} \quad \Delta\vec{w} = -\eta \nabla E(\vec{w})$$

$\eta \rightarrow$ Learning rate
$\rightarrow$ +ve constant
$\rightarrow$ is a hyperparameter in the optimization algorithm that determines the step size for iterative solution to find minimum of a loss function

Adaptive η | Faster but may miss the pt of minimum | Slower but more likely to reach the pt of minimum

It is the strategy for searching through a large / ∞ hypothesis space that can be applied whenever
① The hypothesis space contains continuously parameterized hypotheses.
② The error can be differentiated wrt these hypothesis parameters.

14) Write algorithm for Back propagation.

The BACKPROPAGATION Algorithm learns the weights for a multilayer network, given a network with a fixed set of units and interconnections.

It employes gradient descent to attempt to minimize the squared error between the network output values and the target values for these outputs.

In BACKPROPAGATION algorithm we consider networks with multiple output units rather than single units as before so we redefine E to sum the errors over all of the n/w output units.

$$E(\vec{w}) \equiv \frac{1}{2} \sum_{d \in D} \sum_{k \in outputs} (t_{kd} - O_{kd})^2$$

where

Outputs – is the set of output units in the n/w

t_{kd} and O_{kd} – the target and output values associated with the k^{th} output unit

d – training example.

BACKPROPAGATION (training_example, η, n_{in}, n_{out}, n_{hidden})

Each training example is a pair of the form $(\vec{x}, \vec{t})$ where $(\vec{x})$ is the vector of network input value $(\vec{t})$ and is the vector of target network output values.

η is learning rate, n_{in} is the number of network inputs n_{hidden} the number of units in the hidden layer and n_{out} the number of output units.

The input from unit i into unit j is denoted x_{ji} and the weight from unit i to unit j is denoted w_{ji}

1) Create a feed-forward network with n_{in} inputs, n_{hidden} hidden units and n_{out} output units.

2) Initialise all network weights to small random numbers.

3) Until the termination condition is met, Do

   a) For each $(\vec{x}, \vec{t})$ in training examples, Do

      Propagate the input forward through the network

      i) Input the instance $\vec{x}$ to the network &

         Compute the output O_u of every unit u in the network

propagate the ~~input~~ errors backward through the network

2) For each network output unit k, calculate its error term δ_k

$$\delta_k \leftarrow O_k (1 - O_k)(t_k - O_k)$$

3) For each hidden unit h, calculate its error term δ_h

$$\delta_h \leftarrow O_h (1 - O_h) \sum_{k \in \text{outputs}} w_{h,k} \delta_k$$

4) Update each network weight w_{ji}

$$w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$$

where

$$\Delta w_{ji} = \eta \delta_j x_{ij}$$

15) Explain the importance of stochastic gradient descent.

Stochastic gradient descent is an optimization algorithm often used in machine learning applications to find the model parameters that correspond to the best fit b/w predicted and actual outputs.

- Suppose you have one million samples in your dataset so if you use a typical GD optimization technique you will have to use all of the one million samples for ~~s~~ iteration
- This problem is solved by SGD which uses only a single sample to perform each iteration. The sample is randomly shuffled & selected for performing the iteration.
- In GD the error is summed over all examples before updating weight whereas in SGD weight are updated upon examing each training example.
- ~~Storing~~ It is easier to fit in the memory due to a single training example being processed by the n/w
- It is computationally fast as only 1 sample is processed at a time
- For large datasets it can converge faster as it causes updates to the parameters more frequently.
- Due to frequent updates the steps taken towards the minima of the ~~data~~ loss function have oscillations that can help to get out of the local minimum of the loss function