① **Find S :-**

It is a basic concept learning algorithm in machine learning.

i) Start with most specific hypothesis
$h_0 = \langle \phi\ \phi, \_\_ \rangle$

ii) Take next example and if negative then no changes occur to hypothesis.

iii) If positive and we find that our initial hypothesis is too specific then we update our current hypothesis is to a general

iv) Repeat all steps untill we reach last hypothesis.

Take an example

Sky	Temp	Humid	Wind	Water	Forest	o/p
Sunny	Warm	Normal	Strong	Warm	Same	Yes
Sunny	Warm	High	Strong	Warm	Same	Yes
Rainy	Cold	High	Strong	Warm	Change	No
Sunny	Warm	High	Strong	Cool	Change	Yes

Now take most specific hypothesis as h_0
$h_0 = \langle \phi, \phi, \phi, \phi, \phi, \phi \rangle$

I Compare with first hypothesis
$h_1 = \langle$ Sunny, Warm, Normal, Strong, Warm, Same $\rangle$

II Compare it with second hypothesis and do accordingly
$h_2 = \langle$ Sunny, Warm, ?, Strong, Warm, Same $\rangle$

III Now, after compairing with third hypothesis we see that it is negative so according to algorithm we will do nothing
h_3 = < Sunny, Warm, ?, Strong, Warm, Same >

IV Compare it with fourth hypothesis as it is positive so do accordingly
h_4 = < Sunny, Warm, ?, Strong, ?, ? >

Now the final hypothesis is given as
h_f : < Sunny, Warm, ?, Strong, ?, ? >

Limitations :-

i) There is no way to determine if the hypothesis is consistant throughout the data
ii) Inconsistant training sets can actually mislead the find-S algorithm, since it ignores the negative examples
iii) It does not provide a backtracking technique to determine the best possible changes that could be done to improve the result hypothesis

Candidate Elimination :-

It incrementely builds the version space given a hypothesis space H and a set E of examples.

i) Load the data.
ii) Initialize General and Specific Hypothesis

iii) For each +ve training example
  if all value == hypo value:
    Do nothing
  else:
    replace attribute value with ?

iv) If negative example
  Make generalize hypothesis more specific.

Take an example

Sky	Temp	Humid	Wind	Water	Forest	o/p
Sunny	Warm	Normal	Strong	Warm	Same	Yes
Sunny	Warm	High	Strong	Warm	Same	Yes
Rainy	Cold	High	Strong	Warm	Change	No
Sunny	Warm	High	Strong	Cool	Change	Yes

I. Initialize most general and specific hypo.
S_0: <ϕ, ϕ, ϕ, ϕ, ϕ, ϕ> G_0: <?, ?, ?, ?, ?, ?>

I Compare with the first +ve hypothesis.
S_1: <Sunny, Warm, Normal, Strong, Warm, Same> G_1: <?, ?, ?, ?, ?, ?>

II Now, Compare it with second +ve hypothesis.
S_2: <Sunny, Warm, ?, Strong, Warm, Same> G_2: <?, ?, ?, ?, ?, ?>

III As third hypothesis is -ve so do accordingly.
S_3: <Sunny, Warm, ?, Strong, Warm, Same> G_3: <Sunny, ?, ?, ?, ?, ?>,
<?, Warm, ?, ?, ?, ?>
<?, ?, Normal, ?, ?, ?>, <?, ?, ?, ?, Cool, ?>
<?, ?, ?, ?, ?, Same>

IV Now again compare with last +ve hypo
S_4: < Sunny, Warm, ?, Strong, ?, ? > G_4: < Sunny, ?, ?, ?, ?, ? >
< ?, Warm, ?, ?, ?, ? >

Now final hypothesis is.

G_f : < Sunny, ?, ?, ?, ?, ? > < ?, Warm, ?, ?, ?, ? >
S_f : < Sunny, Warm, ?, Strong, ?, ? >

(2) Solve using KNN $x(A=3, B=7)$ $k=3$

A	B	Label
7	7	False
7	4	False
3	4	True
1	4	True

Now if we want to find the label for $x(A=3, B=7)$ then take distance formula as $(x_1-x_2)^2+(y_1-y_2)^2$

A	B	Distance	Label
7	7	16	False
7	4	25	False
3	4	9	True
1	4	13	True
3	7		?

Now, as we know that the value of k is 3 so we have to take least three distances.

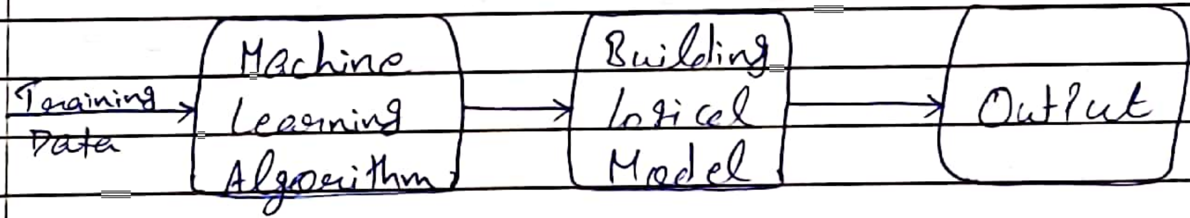
9, 13, 16 are the least three distances
True, True, False are the respective labels

It can easily be seen that True is maximum time in the result, so we can conclude that label for
$x(A = 3, B = 7)$ is True.

3 a) Design a learning System:

When we fed the training data to machine learning algorithm this algorithm will produce a mathematical model and with help of model, the machine will make a prediction and take a decision without being explicitly programmed.

Also during training data, the more machine will work with it more it will get experience and the more it will get experience the more efficient result is produced.



Learn From Data

Designing a System:

According to Tom Mitchell,
A computer program is said to be learning

from experience (E), with respect to some task (T). Thus, the performance measure (P) is the performance at task T, which is measured by P and it improves with experience E.

Take an example:

Suppose we are designing a system for Spam E-mail Detection.

- Task, T: To classify mails into spam or Not spam
- Performance measure, P: Total percent of mails being correctly classified as being 'Spam' or 'Not Spam'
- Experience, E: Set of mails with label 'Spam'

(Choosing Training Experience)

↓

(Choosing Target Function)

↓

(Choosing Representation of Target Function)

↓

(Choosing Function Approximation)

↓

(Final Design)

	Supervised	Unsupervised
i)	They are trained using labeled data	i) They are trained using unlabeled data
ii)	It takes direct feedback to check if it is predicting correct output or not	ii) It does not take any feedback
iii)	These model predicts the output	iii) These model finds the hidden pattern in data
iv)	In this, input data is provided to the model along with the output	iv) In this, only input data is provided
v)	The goal of this learning is to train the model so that it can predict the output when it is given new data	v) The goal of this model is to find the hidden pattern and useful insights from unknown datasets
vi)	It needs supervision to train the model	vi) It does not need any supervision to train the model.
vii)	It can be categorized in Classification and Regression Problems.	vii) It can be Classified in Clustering and Association Problem

(4) Information gain calculates the reduction in entropy of an attribute. It is used for the construction of the decision tree from the training dataset by evaluating the information gain of each variable and collecting the variable that maximises this information gain which in term minimises the entropy and splits the dataset into groups for effective classification.

A skewed distribution has low probability whereas a distribution where centre have equal to have larger entropy.

Entropy is used to calculate the purity of dataset and information gain procedes a way to use entropy to calculate how a change to the dataset impacts the purity of the dataset.

Formula is given as.

$$\text{Entropy}(S[\oplus P, \ominus P]) = -P_{\oplus} \log_2 (P_{\oplus}) - P_{\ominus} \log_2 P_{\ominus}$$

$$\text{Information Gain}(S, x) = E(S) - \sum_{\{T\}} S_i (E)$$

where S is target population and x is the attributes

Now, take the example.

A	B	Label
a1	b1	No
a1	b2	Yes
a2	b3	Yes
a2	b2	No
a2	b1	Yes

We can see that it contains 3 +ve values and 2 negative values.

Entropy $\rightarrow -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$

$$E(3+, 2-) = -\frac{3}{5} \log_2 \frac{3}{5} - \frac{2}{5} \log_2 \frac{2}{5} = 0.97$$

$$Gain(S, A) = E(S) - \sum_{v \in \{a_1, a_2\}} \frac{|S_v|}{|S|} E(S_v)$$

$$= E(S) - \frac{2}{5} E(S_{a_1}) - \frac{3}{5} E(S_{a_2})$$

$E(S_{a_1}) = 1$ ($\because$ equal +ve & -ve examples)

$$E(S_{a_2}) = -\frac{2}{3} \log_2 \frac{2}{3} - \frac{1}{3} \log_2 \frac{1}{3} = 0.917$$

$$Gain(S, A) = 0.97 - \frac{2}{5} \times 1 - \frac{3}{5} \times 0.917$$

$$\boxed{Gain(S, A) = 0.0198}$$

Now, $Gain(S, B) = E(S) - \sum_{v \in \{b_1, b_2, b_3\}} \frac{|S_v|}{|S|} E(S_v)$

$$E(S) - 2/5\, E(S_{b_1}) - 2/5\, E(S_{b_2}) - 1/5\, E(S_{b_3})$$

$E(S_{b_1}) = 1$ ($\because$ equal +ve & -ve examples)
$E(S_{b_2}) = 1$ ($\because$ " " " ")
$E(S_{b_3}) = 0$

$Gain(S, B) = 0.97 - 2/5(1) - 2/5(1) - 0$

$$\boxed{Gain(S, B) = 0.17}$$

(5) a)

	Perceptron Rule	Delta Rule
i)	In this rule, network starts its learning by assigning a random value to each weight	i) In delta rule, modification in symptatic weight of a node is equal to the multiplication of error & input
ii)	It originates from hebbian assumption.	ii) It derived from gradient descent method.
iii)	It stops after finite number of learning steps.	iii) It continuous forever converging asymtotically to the solution.
iv)	In this rule samples are linearly seperable	iv) In this rule, training samples are not linearly seperable.
v)	It trains on the threshold output.	v) It trains on un-threshholded output.

vi)	It is driven by binary differences b/w correct and predicted output	vi) It is driven by continuous differences between correct and predicted output.
vii)	In this rule, the weight is modified by incremental updates.	vii) In this rule, the weight is modified by gradient descent

b) **Back propagation Algorithm :-** It is used to effectively train a neural network through a method called chain rule.

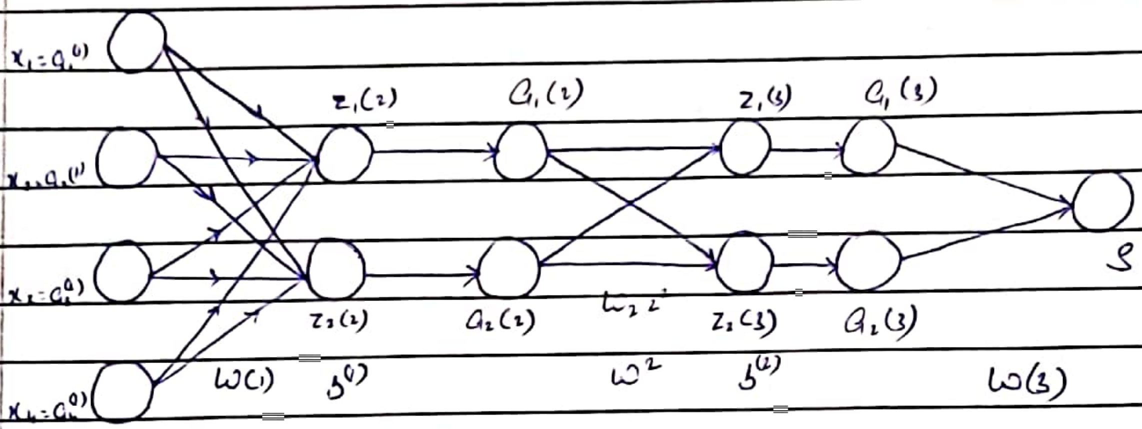
This algorithm performs learning on a multilayer feed forward neural network. It iteratively learns a set of weights for predicting a class label of tuples.

The multilayer feed forward neural network has three layers → input, hidden and output layer.

Input Layer : A layer from when an input is feed to units

$x_i = a_i^{(1)}$, $i \in 1, 2, 3, 4$

a is a set of activations equal to the input values.



Hidden Layer :- These inputs then passed through the input layers and the weighted and feed to hidden layers, a layer of neuron like units. The final values are computed using z^l in layer and a^l - activations in layer.

$l = 2$, $z^{(2)} = w^{(1)} x + b^{(1)}$, $a^{(2)} = f(z^{(2)})$

$l = 3$

$$z^{(3)} = w^{(2)} a^{(2)} + b^{(2)}$$
$$a^{(3)} = f(z^{(3)})$$

where $w^{(2)}$ & $w^{(3)}$ are weights of layers 2 & layer 3 and $b^{(2)}$ and $b^{(3)}$ are the biases

Output Layer :- The final part of the neural network is output layer which produces the predicted value

$$\hat{y} = w^{(3)} \; a^{(3)}$$

(6) Brute Force Bayes Learning

- For each hypothesis h in H, calculate the posterior probability.

$$P(h|D) = \frac{P(D|h)\,P(h)}{P(D)}$$

- Output the hypothesis h_{MAP} with the highest posterior probability

$$h_{MAP} = \underset{h \in H}{\operatorname{argmax}}\; P(h|D)$$

$$P(h) = \frac{1}{|H|} \text{ for all } h \text{ in } H.$$

$$P(D|h) = \begin{cases} 1, \text{ if } d_i = h(x_i) \text{ for all } d_i \text{ in } D \\ 0, \text{ otherwise.} \end{cases}$$

To summarize, Bayes theorem implies that the posterior probability $P(h|D)$ under our assumed $P(h)$ and $P(D|h)$ is

$$P(h|D) = \begin{cases} \frac{1}{|VS_{H,D}|}, & \text{if } h \text{ is consistent with } D \\ 0, & \text{otherwise.} \end{cases}$$