VTU-AI&ML-18CS71-Jan 2023

Model Answer

Dr. P. N. Singh, Professor(CSE), President, ICT-CS, Indian Science Congress 2021-23

Module 1

1.

a. The water jug problem states: You are provided with two jugs one with 4 gallons of capacity and another with 3 gallons of capacity. Neither has measuring markers on it. How can we get exactly 2 gallons of water in 4 gallons of jug?

i) Write down the production rules for above problem

ii) **Write any one solution of the above problem**

[8 Marks]

Solution:

i) Production Rules

S.No.	Initial	Condition	Final state	Description of action taken
1.	(x,y)	If x<4	(4,y)	Fill the 4 gallon jug completely
2.	(x,y)	if y<3	(x,3)	Fill the 3 gallon jug completely
3.	(x,y)	If x>0	(x-d,y)	Pour some part from the 4 gallon jug
4.	(x,y)	If y>0	(x,y-d)	Pour some part from the 3 gallon jug
5.	(x,y)	If x>0	(0,y)	Empty the 4 gallon jug
6.	(x,y)	If y>0	(x,0)	Empty the 3 gallon jug
7.	(x,y)	If (x+y)<7	(4, y-[4-x])	Pour some water from the 3 gallon jug to fill the four gallon jug
8.	(x,y)	If (x+y)<7	(x-[3-y],y)	Pour some water from the 4 gallon jug to fill the 3 gallon jug.
9.	(x,y)	If	(x+y,0)	Pour all water from 3 gallon jug to the 4 gallon
10.	(x,y)	if (x+y)<3	(0, x+y)	Pour all water from the 4 gallon jug to the 3 gallon jug

ii) Solution by rules followed

iii) Solution by rules followed:

S.No.	4 gallon jug	3 gallon jug	Rule followed
1.	0 gallon	0 gallon	Initial state
2.	0 gallon	3 gallons	Rule no.2
3.	3 gallons	0 gallon	Rule no. 9
4.	3 gallons	3 gallons	Rule no. 2
5.	4 gallons	2 gallons	Rule no. 7
6.	0 gallon	2 gallons	Rule no. 5
7.	2 gallons	0 gallon	Rule no. 9

b. Explain Steepest Hill climbing technique with an algorithm. Comment on its drawbacks and how to overcome these drawbacks. [12 Marks]
Ans:

Algorithm: Steepest-Ascent Hill Climbing

1. Evaluate the initial state. If it is also a goal state, then return it and quit. Otherwise, continue with the initial state as the current state.
2. Loop until a solution is found or until a complete iteration produces no change to current state:
   (a) Let *SUCC* be a state such that any possible successor of the current state will be better than *SUCC*.
   (b) For each operator that applies to the current state do:
      (i) Apply the operator and generate a new state.
      (ii) Evaluate the new state. If it is a goal state, then return it and quit. If not, compare it to *SUCC*. If it is better, then set *SUCC* to this state. If it is not better, leave *SUCC* alone.
   (c) If the *SUCC* is better than current state, then set current state to *SUCC*.

To apply steepest-ascent hill climbing to the colored blocks problem, we must consider all perturbations of the initial state and choose the best. For this problem, this is difficult since there are so many possible moves. There is a trade-off between the time required to select a move (usually longer for steepest-ascent hill climbing) and the number of moves required to get to a solution (usually longer for basic hill climbing) that must be considered when deciding which method will work better for a particular problem.

Both basic and steepest-ascent hill climbing may fail to find a solution. Either algorithm may terminate not by finding a goal state but by getting to a state from which no better states can be generated. This will happen if the program has reached either a local maximum, a plateau, or a ridge.

A *local maximum* is a state that is better than all its neighbors but is not better than some other states farther away. At a local maximum, all moves appear to make things worse. Local maxima are particularly frustrating because they often occur almost within sight of a solution. In this case, they are called *foothills*.

A plateau is a flat area of the search space in which a whole set of neighboring states have the same value. On a plateau, it is not possible to determine the best direction in which to move by making local comparisons.

A *ridge* is a special kind of local maximum. It is an area of the search space that is higher than surrounding areas and that itself has a slope (which one would like to climb). But the orientation of the high region, compared to the set of available moves and the directions in which they move, makes it impossible to traverse a ridge by single moves.

There are some ways of dealing with these problems, although these methods are by no means guaranteed:

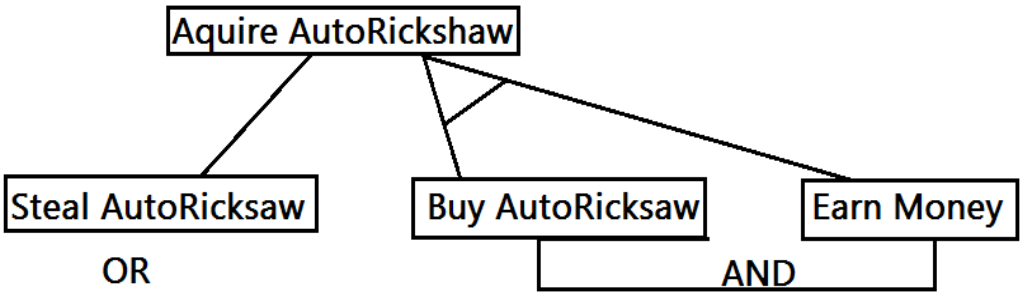
- Backtrack to some earlier node and try going in a different direction. This is particularly reasonable if at that node there was another direction that looked as promising or almost as promising as the one that was chosen earlier. To implement this strategy, maintain a list of paths almost taken and go back to one of them if the path that was taken leads to a dead end. This is a fairly good way of dealing with local maxima.
- Make a big jump in some direction to try to get to a new section of the search space. This is a particularly good way of dealing with plateaus. If the only rules available describe single small steps, apply them several times in the same direction.
- Apply two or more rules before doing the test. This corresponds to moving in several directions at once. This is a particularly good strategy for dealing with ridges.

2.

a. Explain problem reduction with respect to AND-OR graph with suitable example. [7 Marks]

Solution:

AND-OR graphs are specialized graphs that are used in problems that can be broken down into sub problems where AND side of the graph represent a set of task that need to be done to achieve the main goal , whereas the or side of the graph represent the different ways of performing task to achieve the same main goal. A simple AND-OR graph:



The sub task is to either steal Autorickshaw which will directly helps us achieve the main goal "or"
Earn some money "and" buy Autorickshaw which helps us achieve the main goal.
The AND part of the graphs are represented by the AND-ARCS, referring that all the sub problems with the AND-ARCS need to be solved for the predecessor node or problem to be completed.
All of which must then be solved in order for the arc to point solution.
In order to find solution in an AND-OR graph we need an algorithm similar to best-first search but with the ability to handle the AND arcs appropriately.
We define FUTILITY, if the estimated cost of solution becomes greater than the value of FUTILITY then we abandon the search, FUTILITY should be chosen to correspond to a threshold.

b. Write AO* Algorithm [7 Marks]

Ans:

AO* algorithm is a best first search algorithm. AO* algorithm uses the concept of AND-OR graphs to decompose any complex problem given into smaller set of problems which are further solved.

The AO* algorithm is a knowledge-based search technique, meaning the start state and the goal state is already defined , and the best path is found using heuristics. The time complexity of the algorithm is significantly reduced due to the informed search technique. Compared to the A* algorithm , AO* algorithm is very efficient in searching the AND-OR trees very efficiently.

Step-1: Create an initial graph with a single node (start node).
Step-2: Transverse the graph following the current path, accumulating node that has not yet been expanded or solved.
Step-3: Select any of these nodes and explore it. If it has no successors then call this value- FUTILITY else calculate f'(n) for each of the successors.
Step-4: If **f'(n)=0**, then mark the node as **SOLVED**.
Step-5: Change the value of f'(n) for the newly created node to reflect its successors by backpropagation.
Step-6: Whenever possible use the most promising routes, If a node is marked as SOLVED then mark the parent node as SOLVED.
Step-7: If the starting node is SOLVED or value is greater than **FUTILITY** then stop else repeat from Step-2.

Working of AO algorithm:

The AO* algorithm works on the formula given below :
f(n) = g(n) + h(n)
where,
g(n): The actual cost of traversal from initial state to the current state.
h(n): The estimated cost of traversal from the current state to the goal state.
f(n): The actual cost of traversal from the initial state to the goal state

c. Discuss about constant satisfaction and solve the below crypt arithmetic problem. [6 Marks]

Ans:

Constraints Satisfaction:

Constant satisfaction implies that such an issue must be solved while adhering to a set of restrictions or guidelines.

Whenever a problem is actually variables comply with stringent conditions of principles, it is said to have been addressed using the solving multi - objective method. Wow what a method results in a study sought to achieve of the intricacy and organization of both the issue.

Three factors affect restriction compliance, particularly regarding:

- It refers to a group of parameters, or X.
- D: The variables are contained within a collection several domain. Every variables has a distinct scope.
- C: It is a set of restrictions that the collection of parameters must abide by.

In constraint satisfaction, domains are the areas wherein parameters were located after the restrictions that are particular to the task. Those three components make up a constraint satisfaction technique in its entirety. The pair "scope, rel" makes up the number of something like the requirement. The scope is a tuple of variables that contribute to the restriction, as well as rel is indeed a relationship that contains a list of possible solutions for the parameters should assume in order to meet the restrictions of something like the issue.

Issues with Contains A certain amount Solved

For a constraint satisfaction problem (CSP), the following conditions must be met:

- States area
- fundamental idea while behind remedy.

The definition of a state in phase space involves giving values to any or all of the parameters, like as

X1 = v1, X2 = v2, etc.

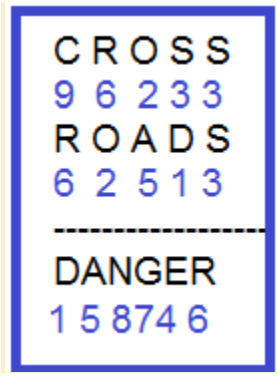
There are 3 methods to economically beneficial to something like a parameter:

I. Consistent or Legal Assignment: A task is referred to as consistent or legal if it complies with all laws and regulations.

II. Complete Assignment: An assignment in which each variable has a number associated to it and that the CSP solution is continuous. One such task is referred to as a completed task.

III. A partial assignment is one that just gives some of the variables values. Projects of this nature are referred to as incomplete assignment.

CROSS + ROADS = DANGER



3. Consider the following sentences:
   a. John likes all kinds of food
   b. Apples are food
   c. Chicken is food
   d. Anything anyone eats and is not killed if food
   e. Bill eats peanuts and still alive
   f. Sue eats every everything Bill eats.

   I. Translate these sentences in predicate logic
   II. Prove that John like peanuts using backward chaining
   III. Covert the formulas of (i) in into clause form
   IV. Prove that John likes peanuts using resolution

Solution

i.

a. $\forall$ x: Food(x) $\rightarrow$ likes(John, x)
b. Food(apples)
c. Food(Chicken)
d. $\forall$ x: ($\exists$ y : eats (y,x) $\wedge$ $\neg$ killed by (y,x)) $\rightarrow$ food(x)
e. Eats(Bills, peanuts)
f. alive(Bill)
g. $\forall$ x: eats (Bill, x) $\rightarrow$ eats (Sue, x)
h. $\forall$ x: $\forall$ y : alive (x) $\rightarrow$ killed by (x, y)

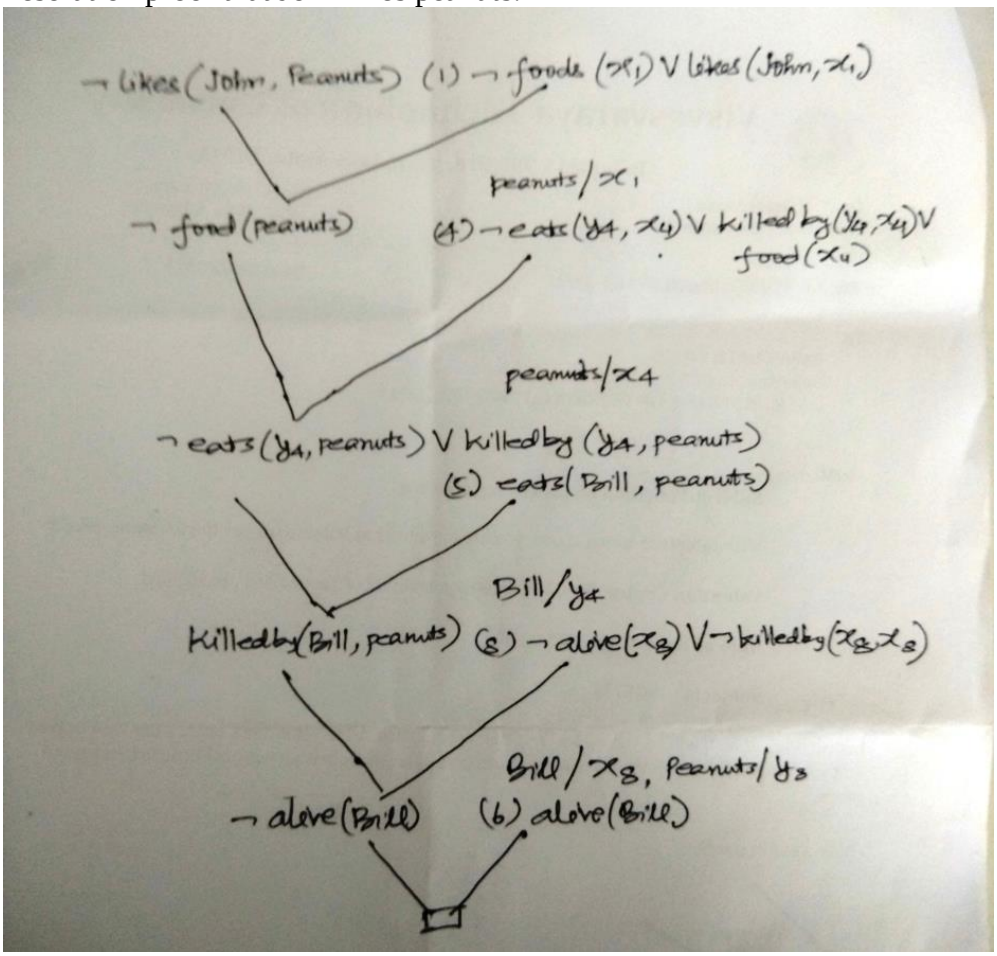
ii. Backward Chaining Proof

likes (Jophn, Peanutes)
↑ (1)
food(peanuts)
↑ (4)
$\exists$ y : eats (y, peanuts) $\wedge$ $\neg$ killed by(y, peanuts)
↑ (5)
$\neg$ killed by(Bill, peanuts)
↑ (8)
Alive(Bill)
↑ (6)
Nil

iii. Clause form

a. ¬ food(x1) ∨ likes (John, x1)
b. food(apples)
c. food (chicken)
d. ¬ eats(y4, x4) ∨ killed by(y4,x4) ∨ (x4)
e. eats (Bill, peanuts)
f. alive(Bill)
g. ¬ eats(Bill, x7) ∨ eats(Sue, x7)
h. ¬ alive(x8 ¬ killed by(x8, y8)

iv. Resolution proof that John likes peanuts:



4.

a. Distinguish forward and backward reasoning with an example. (4 Marks)

Solution:

In Artificial intelligence, the purpose of the search is to find the path through a problem space. There are two ways to pursue such a search that are forward and backward reasoning.

The significant difference between both of them is that forward reasoning starts with the initial data towards the goal. Conversely, backward reasoning works in opposite fashion where the purpose is to determine the initial facts and information with the help of the given results.

BASIS FOR COMPARISON	FORWARD REASONING	BACKWARD REASONING
Basic	Data-driven	Goal driven
Begins with	New Data	Uncertain conclusion
Objective is to find	Conclusion that must follow	Facts to support the conclusions
Type of approach	Opportunistic	Conservative
Flow	Incipient to consequence	Consequence to incipient

Example: Forward reasoning/chaining (Data to goal):
Data a = 1 and b = 2
Condition if a == 1 and b==2 then c = 3
If c == 3 then d = 4(goal)

Example: Backward reasoning/chaining (Goal to data):
The goal d = 4 when c==4
The c will be 4 when a is 1 and b is 2

b. Find maximally specific hypothesis for the training instances given below. Also write Find-S algorithm. The concept of this particular problem will be on what days that a person likes to go on walk.

Time	Weather	Temperature	Company	Humidity	Wind	Goes
Morning	Sunny	Warm	Yes	Mild	Strong	Yes
Evening	Rainy	Cold	No	Mild	Normal	No
Morning	Sunny	Moderate	Yes	Normal	Normal	Yes
Evening	Sunny	Cold	Yes	High	Strong	Yes

[8 marks]

Solution:
Starting with most specific hypothsis

h = {ϕ, ϕ, ϕ, ϕ, ϕ, ϕ, ϕ}

h = {‘Morning’, ‘Sunny’, ‘Warm’, ‘Yes’, ‘Mild’, ‘Strong’}

The 2nd row has a negative outcome so we ignore this & hypothesis remains the same.

h = {‘Morning’, ‘Sunny’, ‘Warm’, ‘Yes’, ‘Mild’, ‘Strong’}

The next (3rd row) has the positive outcome. If mismatch is found we replace with general case ?

h = {‘Morning’, ‘Sunny’, ‘?’, ‘Yes’,’?’, ‘?’}

The next example (4th row) has the positive outcome. If mismatch is found , we replace with general case ‘?’

h = {‘?’, ‘Sunny’, ?, ‘Yes’, ?, ?}

So, Maximal Specific Hypothesis is:

['?' 'Sunny' '?' 'Yes' '?' '?']

The person should go for walk when the weather is “Sunny” and accompanied with anyone (Company is “Yes”)

The Find-S algorithm:

The process starts with initializing ‘h’ with the most specific hypothesis, generally, it is the first positive example in the data set.

1. We check for each positive example. If the example is negative, we will move on to the next example but if it is a positive example we will consider it for the next step.
2. We will check if each attribute in the example is equal to the hypothesis value.
3. If the value matches, then no changes are made.
4. If the value does not match, the value is changed to ‘?’.

   We do this until we reach the last positive example in the data set

c. Define version space. Discuss the limitations of find-s algorithm over candidate elimination algorithm

Solution:

Version Space:

A version space is a hierarchial representation of knowledge that enables you to keep track of all the useful information supplied by a sequence of learning examples without remembering any of the examples.

The version space method is a concept learning process accomplished by managing multiple models within a version space.

Version Space Characteristics

Tentative heuristics are represented using version spaces.

A version space represents all the alternative plausible descriptions of a heuristic.

A plausible description is one that is applicable to all known positive examples and no known negative example.

A version space description consists of two complementary trees:

1. One that contains nodes connected to overly general models, and
2. One that contains nodes connected to overly specific models.

Node values/attributes are discrete.

Fundamental Assumptions

1. The data is correct; there are no erroneous instances.
2. A correct description is a conjunction of some of the attributes with values.

Limitations of Find-S algorithm over Candidate Elimination Algorithm.

There are a few limitations of the Find-S algorithm:

- There is no way to determine if the hypothesis is consistent throughout the data.
- Inconsistent training sets can actually mislead the Find-S algorithm, since it ignores the negative examples.
- Find-S algorithm does not provide a backtracking technique to determine the best possible changes that could be done to improve the resulting hypothesis.
- Now that we are aware of the limitations of the Find-S algorithm, let us take a look at a practical implementation of the Find-S Algorithm.

5.

a. Explain the concept of Decision Tree learning. Write about selection Measure used to build the decision tree using ID3 algorithm. (07 Marks)

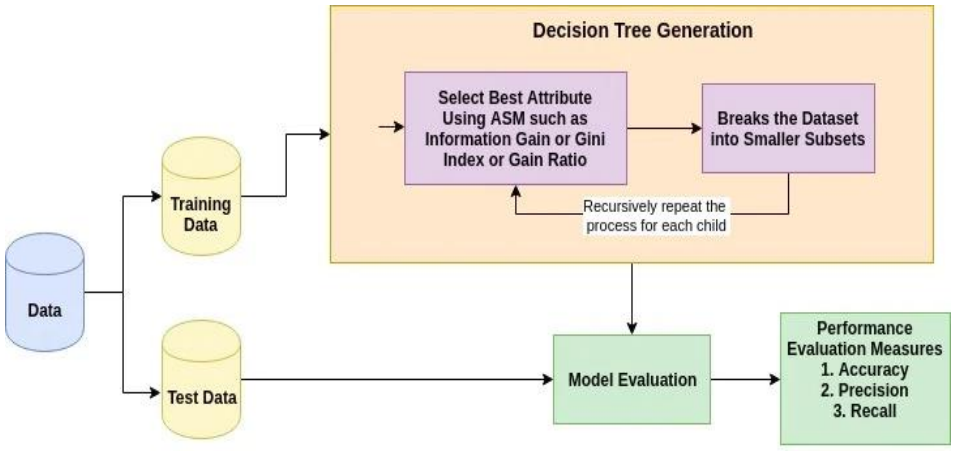
Solution.

- A decision tree is a flowchart-like tree structure where an internal node represents feature(or attribute), the branch represents a decision rule, and each leaf node represents the outcome. The topmost node in a decision tree is known as the root node.
- It learns to partition on the basis of the attribute value. It partitions the tree in recursively manner call recursive partitioning.

The basic idea behind any decision tree algorithm is as follows:

- Select the best attribute using Attribute Selection Measures(ASM) to split the records.
- Make that attribute a decision node and breaks the dataset into smaller subsets.
- Starts tree building by repeating this process recursively for each child until one of the condition will match:
  - All the tuples belong to the same attribute value.
  - There are no more remaining attributes.
- There are no more instances

- The core algorithm for building decision trees called **ID3** (Iterative Dichotomiser/Divider 3) by J. R. Quinlan which employs a top-down, greedy search through the space of possible branches with no backtracking.
- ID3 uses *Entropy* and *Information Gain* to construct a decision tree. In ZeroR model there is no predictor, in OneR model we try to find the single best predictor,



To build a decision tree, we need to calculate two types of entropy using frequency tables as follows:

a) Entropy using the frequency table of one attribute:

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

Play Golf	
Yes	No
9	5

Entropy(PlayGolf) = Entropy (5,9)
= Entropy (0.36, 0.64)
= - (0.36 $\log_2$ 0.36) - (0.64 $\log_2$ 0.64)
= 0.94

b) Entropy using the frequency table of two attributes:

$$E(T, X) = \sum_{c \in X} P(c) E(c)$$

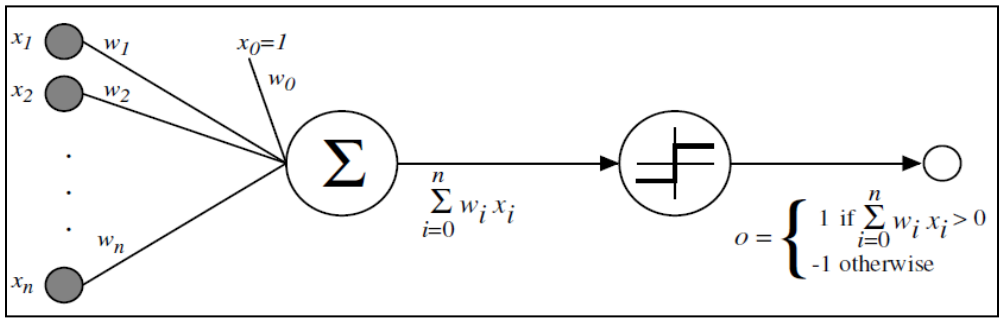
		Play Golf		
		Yes	No	
Outlook	Sunny	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5
				14

E(PlayGolf, Outlook) = **P**(Sunny)***E**(3,2) + **P**(Overcast)***E**(4,0) + **P**(Rainy)***E**(2,3)
= (5/14)*0.971 + (4/14)*0.0 + (5/14)*0.971
= 0.693

b. How a single perceptron can be used to represent the Boolean functions such as AND, OR? (6 Marks)

Solution:

A Perceptron is one type of ANN system is based on a unit called a perceptron. Perceptron is a single layer neural network.



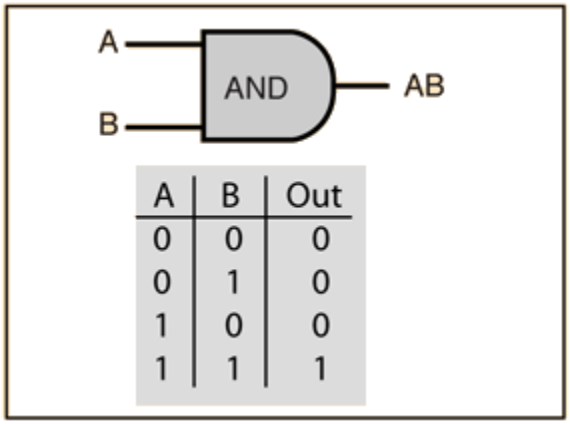
A perceptron takes a vector of real-valued inputs, calculates a linear combination of these inputs, then outputs a 1 if the result is greater than some threshold and -1 otherwise.

Given inputs ***x through x, the output O(x1, . . . , xn) computed by the perceptron is***

$$o(x_1, \ldots, x_n) = \begin{cases} 1 & \text{if } w_0 + w_1 x_1 + \cdots + w_n x_n > 0 \\ -1 & \text{otherwise.} \end{cases}$$

Perceptrons can represent all of the primitive Boolean functions AND, OR, NAND (~ AND), and NOR (~OR)

Representation of AND gate and logic table



A	B	Out
0	0	0
0	1	0
1	0	0
1	1	1

First, we need to understand that the output of an AND gate is 1 only if both inputs (in this case, x1 and x2) are 1. So, following the steps listed above;

Row 1

- From w1*x1+w2*x2+b, initializing w1, w2, as 1 and b as –1, we get;

x1(1)+x2(1)–1

- Passing the first row of the AND logic table (x1=0, x2=0), we get;

0+0–1 = –1

- From the Perceptron rule, if Wx+b≤0, then y`=0. Therefore, this row is correct, and no need for Backpropagation.

Row 2

- Passing (x1=0 and x2=1), we get;

0+1–1 = 0

- From the Perceptron rule, if Wx+b≤0, then y`=0. This row is correct, as the output is 0 for the AND gate.
- From the Perceptron rule, this works (for both row 1, row 2 and 3).

Row 4

- Passing (x1=1 and x2=1), we get;

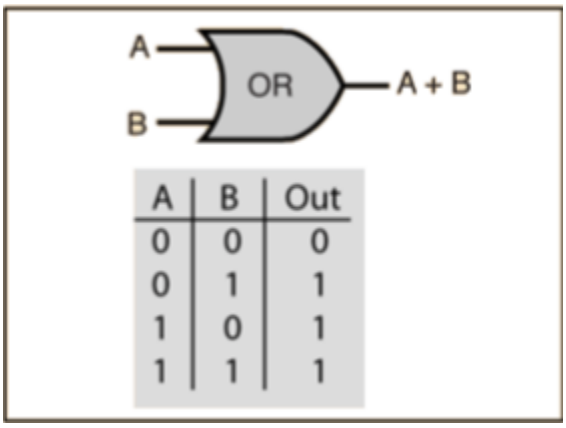
1+1–1 = 1

- Again, from the perceptron rule, this is still valid.

Therefore, we can conclude that the model to achieve an AND gate, using the Perceptron algorithm is;

x1+x2–1

Representation of OR logical gate and logical table



A	B	Out
0	0	0
0	1	1
1	0	1
1	1	1

From the diagram, the OR gate is 0 only if both inputs are 0.

Row 1

- From w1x1+w2x2+b, initializing w1, w2, as 1 and b as –1, we get;
  x1(1)+x2(1)–1
- Passing the first row of the OR logic table (x1=0, x2=0), we get;
  0+0–1 = –1
- From the Perceptron rule, if Wx+b≤0, then y`=0. Therefore, this row is correct.

Row 2

- Passing (x1=0 and x2=1), we get;
  0+1–1 = 0
- From the Perceptron rule, if Wx+b <= 0, then y`=0. Therefore, this row is incorrect.

- So we want values that will make inputs x1=0 and x2=1 give y` a value of 1. If we change w2 to 2, we have;
  0+2–1 = 1
- From the Perceptron rule, this is correct for both the row 1 and 2.
  Row 3
- Passing (x1=1 and x2=0), we get;
  1+0–1 = 0
- From the Perceptron rule, if Wx+b <= 0, then y`=0. Therefore, this row is incorrect.
- Since it is similar to that of row 2, we can just change w1 to 2, we have;
  2+0–1 = 1
- From the Perceptron rule, this is correct for both the row 1, 2 and 3.
  Row 4
- Passing (x1=1 and x2=1), we get;
  2+2–1 = 3
- Again, from the perceptron rule, this is still valid. Quite Easy!
  Therefore, we can conclude that the model to achieve an OR gate, using the Perceptron algorithm is;
  2x1+2x2–1

6.

a. What do you mean by Gain and Entropy? How it is used to build the decision tree? [8 Marks]

Solution:

- Entropy is related to randomness in the information being processed in your machine learning project (measures the expected (i.e., average) amount of information conveyed by identifying the outcome of a random trial)
- Gain (Information gain) can be thought of as the purity in a system: the amount of clean knowledge available in a system. Entropy is a measure of the energy dispersal in the system.
- A high entropy means low information gain, and a low entropy means high information gain.
- Entropy is the measure of uncertainty of a random variable, it characterizes the impurity of an arbitrary collection of examples. The higher the entropy the more the information content.
- In general, an entropy value close to 1 is ideal and above 0.8 is acceptable.
- **Formula: Entropy = -(p(0) * log(P(0)) + p(1) * log(P(1)))**
- If targets values are only positive or only negative then the entropy will be 0
- If in target values positive and negative values are equal, then the entropy will be 1
- Decision Tree ID3 uses *Entropy* and *Information Gain* to construct a decision tree. In ZeroR model there is no predictor, in OneR model we try to find the single best

predictor. To build a decision tree, we need to calculate two types of entropy using frequency tables as follows:

Entropy using the frequency table of one attribute:

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

Entropy using the frequency table of two attributes:

$$E(T, X) = \sum_{c \in X} P(c) E(c)$$

b. Explain back propogation algorithm. Why is it not likely to be trapped in local minima? [8 Marks]

Solution:

Multilayer neural networks use a most common technique from a variety of learning technique, called the back propagation algorithm.

Backpropagation, or backward propagation of errors, is an algorithm that is designed to test for errors working back from output nodes to input nodes.

Using this information, the algorithms adjust the weights of each connection in order to reduce the value of the error function by some small amount. After repeating this process for a sufficiently large number of training cycles the network will usually converge to some state where the error of the calculation is small.

- Create a feed-forward network with n_i inputs, n_{hidden} hidden units, and n_{out} output units.
- Initialize all network weights to small random numbers
- Until the termination condition is met, Do
  - For each (x, t), in training examples, Do
    - Propagate the input forward through the network:
      1. Input the instance x, to the network and compute the output o_u of every unit u in the network.
    - Propagate the errors backward through the network
      2. For each network unit k, calculate its error term δ_k

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

      3. For each network unit h, calculate its error term δ_h

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

4. Update each network weight w_{ji}

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

Where

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

To Summarize

Step 1:The input layer receives the input.
Step 2: The input is then averaged overweights.
Step 3:Each hidden layer processes the output. Each output is referred to as "Error" here which is actually the difference between the actual output and the desired output.
Step 4:In this step, the algorithm moves back to the hidden layers again to optimize the weights and reduce the error.

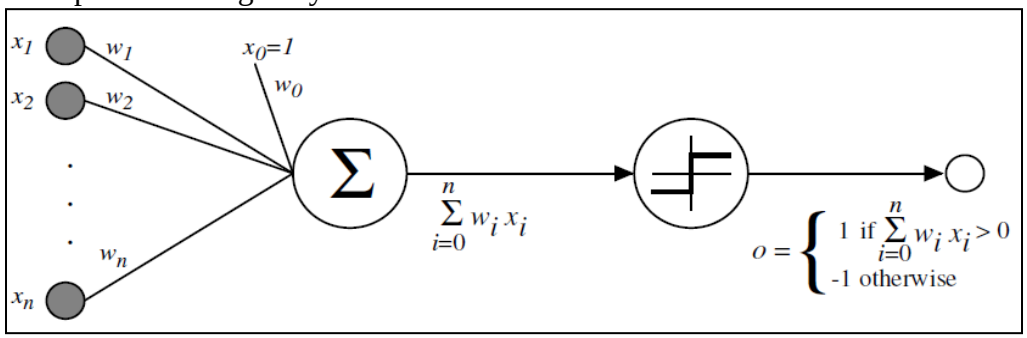
BACKPROPAGATION (training_example, η, n_{in}, n_{out}, n_{hidden})

- Each training example is a pair of the form (x, t), where (x) is the vector of network input values, and (t) is the vector of target network output values.
- η is the learning rate (e.g., 0.05).
- n_i, 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}

It is noticed that the local minima problem in the backpropagation algorithm is **usually caused by update disharmony between weights connected to the hidden layer and the output layer**.
The back propagation gradient descent with adaptive gain, adaptive momentum and adaptive learning rate significantly can prevent the network from trapping into local minima.

c. Discuss the perceptron training rule and delta rule that solves the learning Problem of perceptron.

Solution:
A Perceptron is one type of ANN system is based on a unit called a perceptron. Perceptron is a single layer neural network.



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

Given inputs ***x through x, the output O(x1, . . . , xn) computed by the perceptron is***

$$o(x_1, \ldots, x_n) = \begin{cases} 1 \text{ if } w_0 + w_1 x_1 + \cdots + w_n x_n > 0 \\ -1 \text{ otherwise.} \end{cases}$$

The 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 given training rule and delta rule examples.

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

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***

7.

a. Explain Naïve Bayes Classifier. [8 Marks]

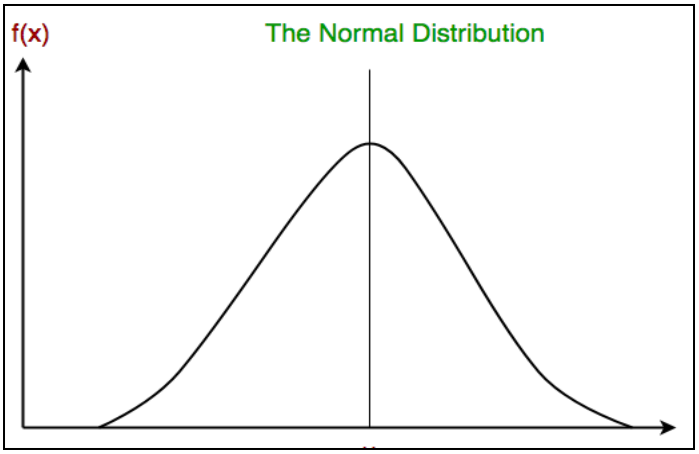
Solution:

- Naïve base classifiers are based on Bayes Theorem
- It is a family of algorithms where all of them share a common principle i.e. evary pair of features being classified independent of each other.

Gaussian Naive Bayes classifier

In Gaussian Naive Bayes, continuous values associated with each feature are assumed to be distributed according to a **Gaussian distribution**. A Gaussian distribution is also called Naormal Distribnution. When plotted, it gives a bell shaped curve which is symmetric about the mean of the feature values

The likelihood of the features is assumed to be Gaussian, hence, conditional probability is given by:



$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right)$$

b. Explain Bayesian Belief Network and conditional independence with example. [8 Marks]

Solution:

- The naive Bayes classifier makes significant use of the assumption that the values of the attributes $a_1 \ldots a_n$ are conditionally independent given the target value v.
- This assumption dramatically reduces the complexity of learning the target function

A Bayesian belief network describes the probability distribution governing a set of variables by specifying a set of conditional independence assumptions along with a set of conditional probabilities

Bayesian belief networks allow stating conditional independence assumptions that apply to subsets of the variables.

Conditional Independence:

A Bayesian network represents a joint distribution using a graph. Specifically, it is a directed acyclic graph in which each edge is a conditional dependency, and each node is a distinctive random variable.

Example 1.

Let X, Y, and Z be three discrete-valued random variables. X is conditionally independent of Y given Z if the probability distribution governing X is independent of the value of Y given a value for Z, that is, if

$$(\forall x_i, y_j, z_k)\ P(X = x_i | Y = y_j, Z = z_k) = P(X = x_i | Z = z_k)$$

$$x_i \in V(X),\ y_j \in V(Y),\ \text{and}\ z_k \in V(Z).$$

The above expression is written in abbreviated form as:

$$P(X_1 \ldots X_l | Y_1 \ldots Y_m, Z_1 \ldots Z_n) = P(X_1 \ldots X_l | Z_1 \ldots Z_n)$$

Example 2.

Let's say A **is** the height of a child **and** B **is** the number of words that the child knows. **There is** a single piece of information that will make A and B completely independent. That is C i.e. child's age

We define the independence in conditional model:

$P(A \cap B \mid C) = P(A \mid C).\ P(B \mid C)$

c. Let us assume dangerous fires are rare (1%) but smoke fairly common (10%) due to barbecues, and 90% of dangerous fires make smoke. Find the probability of dangerous fire when there is smoke. [4 Marks]

Solution:

We can then discover the **probability of dangerous Fire when there is Smoke**:

P(Fire|Smoke) *=P(Fire)* P(Smoke|Fire) /* **P(Smoke)**
=1% x 90% / **10%**
= .01*.9 / .1 = .09
=9%

So it is still worth checking out any smoke to be sure.

8.

a. Discuss Minimum Description length principle in brief. [8 Marks]

Solution:

MDL is an information-theoretic approach in Machine Learning and in or statistical model selection, which basically says that we should pick the model which gives the most compact description of the data, including the description of the model itself.

The minimum description length (MDL) principle is a powerful method of inductive inference, the basis of statistical modeling, pattern recognition, and machine learning. It holds that the best explanation, given a limited set of observed data, is the one that permits the greatest compression of the data. MDL methods are particularly well-suited for dealing with model selection, prediction, and estimation problems in situations where the models under consideration can be arbitrarily complex, and overfitting the data is a serious concern.

The Minimum Description Length principle is motivated by interpreting the definition of h_{MAP} in the light of basic concepts from information theory. Consider again the now familiar definition of h_{MAP}.

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

which can be equivalently expressed in terms of maximizing the $\log_2$

$$h_{MAP} = \underset{h \in H}{\operatorname{argmax}} \log_2 P(D|h) + \log_2 P(h)$$

or alternatively, minimizing the negative of this quantity

$$h_{MAP} = \underset{h \in H}{\operatorname{argmin}} -\log_2 P(D|h) - \log_2 P(h)$$

b. Explain Brute-Force MAP learning algorithm. [8 Marks]

Solution:

We can design a straightforward concept learning algorithm to output the maximum a posteriori hypothesis, based on Bayes theorem, as follows:

BRUTE-FORCE MAP LEARNING algorithm:

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

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

In order specify a learning problem for the BRUTE-FORCE MAP LEARNING algorithm we must specify what values are to be used for P(h) and for P(D|h) ?

Let's choose P(h) and for P(D|h) to be consistent with the following assumptions:

- The training data D is noise free (i.e., di = c(xi))
- The target concept c is contained in the hypothesis space H
- Do not have a priori reason to believe that any hypothesis is more probable than any other.

Given these choices for P(h) and for P(D|h) we now have a fully-defined problem for the above BRUTE-FORCE MAP LEARNING algorithm.

Recalling Bayes theorem, we have

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

Consider the case where h is inconsistent with the training data D

$$P(h|D) = \frac{0 \cdot P(h)}{P(D)} = 0$$

The posterior probability of a hypothesis inconsistent with D is zero

Consider the case where h is consistent with D

$$P(h|D) = \frac{1 \cdot \frac{1}{|H|}}{P(D)} = \frac{1 \cdot \frac{1}{|H|}}{\frac{|VS_{H,D}|}{|H|}} = \frac{1}{|VS_{H,D}|}$$

Where, $VS_{H,D}$ is the subset of hypotheses from H that are consistent with D

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

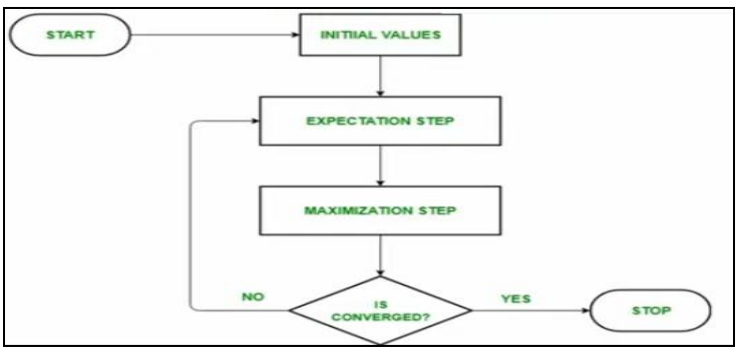
$$P(D|h) = \begin{cases} \frac{1}{|V S_{H,D}|} & \text{if h is consistent with D} \\ 0 & \text{otherwise} \end{cases}$$

c. Explain EM Algorithm. [4 Marks]

Solution:

The EM algorithm can be used even for variables whose value is never directly observed, provided the general form of the probability distribution governing these variables is known.

1. Given a set of incomplete data, consider a set of starting parameters.
2. **Expectation step (E – step):** Using the observed available data of the dataset, estimate (guess) the values of the missing data.
3. **Maximization step (M – step):** Complete data generated after the expectation (E) step is used in order to update the parameters.
4. Repeat step 2 and step 3 until convergence.



Technically,

Step 1:	Calculate the expected value $E[z_{ij}]$ of each hidden variable z_{ij}, assuming the current hypothesis $h = \langle \mu_1, \mu_2 \rangle$ holds.
Step 2:	Calculate a new maximum likelihood hypothesis $h' = \langle \mu_1', \mu_2' \rangle$, assuming the value taken on by each hidden variable z_{ij} is its expected value $E[z_{ij}]$ calculated in Step 1. Then replace the hypothesis $h = \langle \mu_1, \mu_2 \rangle$ by the new hypothesis $h' = \langle \mu_1', \mu_2' \rangle$ and iterate.

9.

a. Explain k-Nearest Neighbour learning algorithm

Solution:

This algorithm is used to solve the classification model problems. K-nearest neighbor or K-NN algorithm basically creates an imaginary boundary to classify the data. When new data points come in, the algorithm will try to predict that to the nearest of the boundary line.

Therefore, larger k value means smother curves of separation resulting in less complex models. Whereas, smaller k value tends to overfit the data and resulting in complex models.

Note: It's very important to have the right k-value when analyzing the dataset to avoid overfitting and underfitting of the dataset.

Using the k-nearest neighbor algorithm we fit the historical data (or train the model) and predict the future.

k- NEAREST NEIGHBOR LEARNING

- The most basic instance-based method is the K- Nearest Neighbor Learning. This algorithm assumes all instances correspond to points in the n-dimensional space Rn.
- The nearest neighbors of an instance are defined in terms of the standard Euclidean distance.
- Let an arbitrary instance x be described by the feature vector
  $((a_1(x), a_2(x), \ldots\ldots, a_n(x))$
  Where,

$$d(x_i, x_j) \equiv \sqrt{\sum_{r=1}^{n} (a_r(x_i) - a_r(x_j))^2}$$

  $a_r(x)$ denotes the value of the r^{th} attribute of instance x.
- Then the distance between two instances x_i and x_j is defined to be $d(x_i , x_j)$
  Where,
- In nearest-neighbor learning the target function may be either discrete-valued or real-valued.

Let us first consider learning ***discrete-valued target functions of the form***

Where, V is the finite set $\{v_1, \ldots v_s\}$

The k- Nearest Neighbor algorithm for approximation a **discrete-valued target function is given below:**

Training algorithm:

- For each training example $\langle x, f(x) \rangle$, add the example to the list *training_examples*

Classification algorithm:

- Given a query instance x_q to be classified,
  - Let $x_1 \ldots x_k$ denote the k instances from *training_examples* that are nearest to x_q
  - Return

$$\hat{f}(x_q) \leftarrow \underset{v \in V}{\operatorname{argmax}} \sum_{i=1}^{k} \delta(v, f(x_i))$$

where $\delta(a, b) = 1$ if $a = b$ and where $\delta(a, b) = 0$ otherwise.

The value $f'(xq)$ returned by this algorithm as its estimate of f(xq) is just the most common value of f among the k training examples nearest to xq.

· If k = 1, then the 1- Nearest Neighbor algorithm assigns to $f'(xq)$ the value f(xi). Where xi is the training instance nearest to xq.

· For larger values of k, the algorithm assigns the most common value among the k nearest training examples.

Finally, the k-NN algorithm steps:

Step-1: Select the number K of the neighbors

Step-2: Calculate the Euclidean distance of K number of neighbors

Step-3: Take the K nearest neighbors as per the calculated Euclidean distance.

Step-4: Among these k neighbors, count the number of the data points in each category.

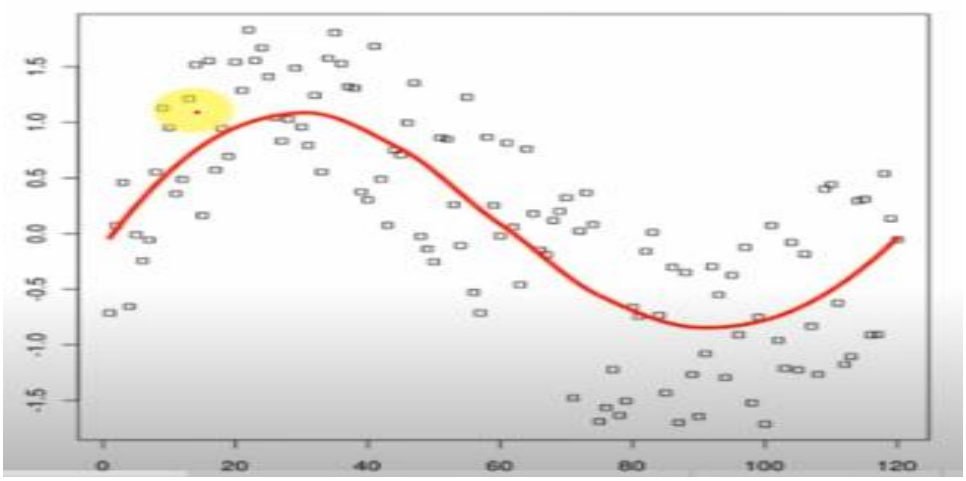
Step-5: Assign the new data points to that category for which the number of the neighbor is maximum.

Step-6: The model is ready.

b. Explain Locally Weighted Regression [8 Marks]

Solution:

- The phrase "locally weighted regression" is called local because the function is approximated based only on data near the query point,
- weighted because the contribution of each training example is weighted by its distance from the query point, and
- regression because this is the term used widely in the statistical learning community for the problem of approximating real-valued functions.
- Given a new query instance x_q, the general approach in locally weighted regression is to construct an approximation f' that fits the training examples in the neighborhood surrounding x_q. This approximation is then used to calculate the value $f'(x_q)$, which is output as the estimated target value for the query instance.



Consider locally weighted regression in which the target function ***f is approximated near x_q using a linear function of the form***

$$\hat{f}(x) = w_0 + w_1 a_1(x) + \cdots + w_n a_n(x)$$

Where, $a_i(x)$ denotes the value of the ith attribute of the instance x

Derived methods are used to choose weights that minimize the squared error summed over the set D of training examples using gradient descent

$$E \equiv \frac{1}{2} \sum_{x \in D} (f(x) - \hat{f}(x))^2$$

Which led us to the gradient descent training rule

$$\Delta w_j = \eta \sum_{x \in D} (f(x) - \hat{f}(x)) a_j(x)$$

Where, η is a constant learning rate

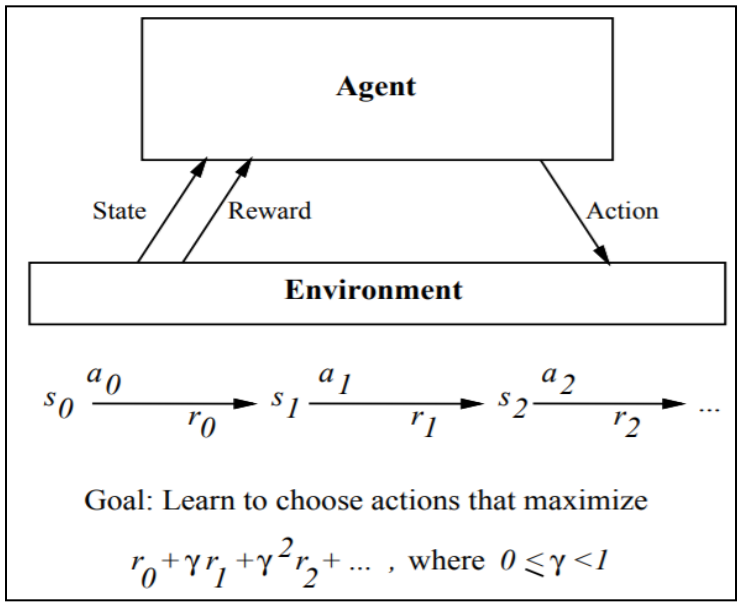
c. What is Reinforcement Learning? [4 Marks]

Solution:

- Reinforcement learning addresses the question of how an autonomous agent that senses and acts in its environment can learn to choose optimal actions to achieve its goals.

  Consider building a learning robot. The robot, or *agent, has a set of sensors to observe the state of its environment, and a set of actions it can perform to alter this state.*
- Its task is to learn a control strategy, or *policy, for choosing actions that achieve its goals.*
- The goals of the agent can be defined by a *reward function that assigns a numerical value to each distinct action the agent may take from each distinct state.*

- This reward function may be built into the robot, or known only to an external teacher who provides the reward value for each action performed by the robot.
- The task of the robot is to perform sequences of actions, observe their consequences, and learn a control policy.
- The control policy is one that, from any initial state, chooses actions that maximize the reward accumulated over time by the agent.



Reinforcement Learning Problem

- An agent interacting with its environment. The agent exists in an environment described by some set of possible states S.
- Agent perform any of a set of possible actions A. Each time it performs an action a, in some state st the agent receives a real-valued reward r, that indicates the immediate value of this state-action transition. This produces a sequence of states si, actions ai, and immediate rewards ri as shown in the figure.
- The agent's task is to learn a control policy, **π: S → A, that maximizes the expected sum of these rewards, with future rewards discounted exponentially by their delay.**

Elements of RL-Agent, Environment,State,reward, & action-

10.

a. Distinguish Eager Learning vs Lazy Learning algorithm. [4 Marks]

Solution:

Eager learning methods construct general, explicit description of the target function based on the provided training examples.

Lazy learning methods simply store the data and generalizing beyond these data is postponed until an explicit request is made.

An eager algorithm executes immediately and returns a result. A lazy algorithm defers computation until it is necessary to execute and then produces a result.

Eager learners use data from multiple sources, or data that is not yet labeled, in order to make better predictions. Examples may be stock trading and medical diagnosis.

Lazy learner methods generalize the training data is, in theory, delayed until a query is made to the system. It runs when new instance comes.
K-NN, Locally Weighted Regression, Radial Basis Functions are examples of Lazy Learner algorithms.

b. Write Short note on Q-Learning. [8 Marks]

Solution:
Q-learning is **a model-free, off-policy reinforcement learning that will find the best course of action, given the current state of the agent**. Depending on where the agent is in the environment, it will decide the next action to be taken.

Q learning algorithm

For each s, a initialize the table entry $\hat{Q}(s, a)$ to zero.
Observe the current state s
Do forever:

- Select an action a and execute it
- Receive immediate reward r
- Observe the new state s'
- Update the table entry for $\hat{Q}(s, a)$ as follows:

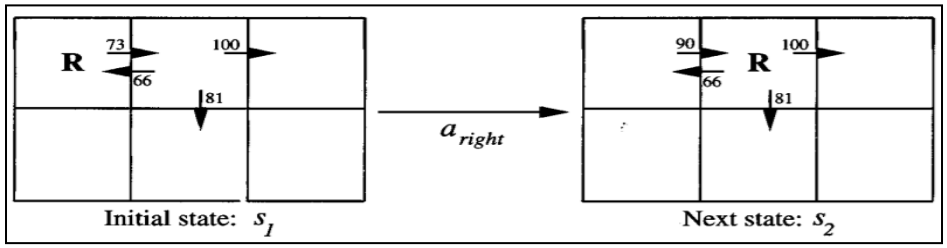
$$\hat{Q}(s, a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s', a')$$

- $s \leftarrow s'$

The *Q Function*

The value of Evaluation function ***Q(s, a) is the reward received immediately upon executing action a from state s, plus the value (discounted by γ) of following the optimal policy thereafter***

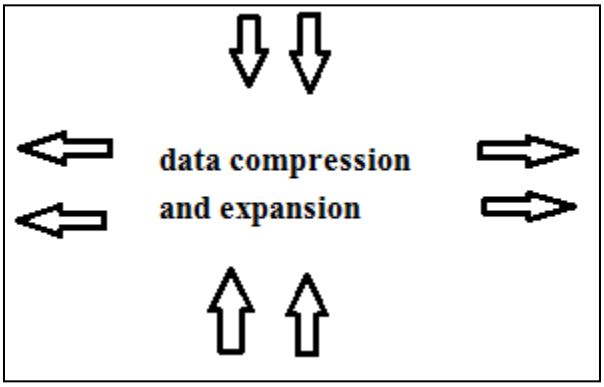
To illustrate the operation of the Q learning algorithm, consider a single action taken by an agent, and the corresponding refinement to $\hat{Q}$ shown in below figure



c. Discuss about Radial Basis Function in detail. [8 Marks]

Solution:

- A type of Instance Based Learning (Lazy Learning)
- **Real-valued functions that use supervised machine learning (ML) to perform as a non-linear classifier. It may be for unsupervised learning also**
- RBFs are means **to approximate multivariable (also called multivariate) functions by linear combinations of terms based on a single univariate function** (the radial basis function).
- Used in ANN (Multi layer perceptron)
- Has only one hidden node
- Although we use various types of radial basis functions, the Gaussian function is the most common.
- Data is not linearly separable
- Sol steps to Make the data separable
    - Increase the dimensionality (2-d 3-D if required)
    - Expand the direction (horizontal) and compress the direction (vertical) - Mandatorily



One approach to function approximation that is closely related to distance-weighted regression and also to artificial neural networks is learning with radial basis functions

- In this approach, the learned hypothesis is a function of the form

$$\hat{f}(x) = w_0 + \sum_{u=1}^{k} w_u K_u(d(x_u, x)) \qquad \textbf{equ (1)}$$

- Where, each xu is an instance from X and where the kernel function Ku(d(x_u, x)) is defined so that it decreases as the distance d(x_u, x) increases.
- Here k is a user provided constant that specifies the number of kernel functions to be included.
- $\hat{f}$ is a global approximation to f (x), the contribution from each of the Ku(d(x_u, x)) terms is localized to a region nearby the point x_u.

$$\hat{f}(x) = w_0 + \sum_{u=1}^{k} w_u K_u(d(x_u, x)) \qquad \textbf{equ (1)}$$

- Choose each function Ku(d(xu, x)) to be a Gaussian function centred at the point x_u with some variance σ_u^2

$$K_u(d(x_u, x)) = e^{\frac{1}{2\sigma_u^2} d^2(x_u, x)}$$

- The functional form of equ(1) can approximate any function with arbitrarily small error, provided a sufficiently large number k of such Gaussian kernels and provided the width σ^2 of each kernel can be separately specified
- The function given by equ(1) can be viewed as describing a two layer network where the first layer of units computes the values of the various Ku(d(x_u, x)) and where the second layer computes a linear combination of these first-layer unit values.