CMR
INSTITUTE OF
TECHNOLOGY

USN | | | | | | | | | |

Internal Assessment Test 1 – November. 2021

Sub:	MACHINE LEARNING							**Sub Code:**	18MCA53
Date:	12-11-2021	Duration:	90 min's	Max Marks:	50	**Sem**	5^{th}	**Branch:**	**MCA**

Note : Answer FIVE FULL Questions, choosing ONE full question from each Module

	PART I	MARKS	OBE	
			CO	RBT
1a)	Well-Posed Learning Definition: A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E. Examples: Checkers Game: A computer program that learns to play checkers might improve its performance as measured by its ability to win at the class of tasks involving playing checkers game, through experience obtained by playing games against itself.	4	CO1	L1
2b)	***checkers learning problem*:** • Task T: playing checkers • Performance measure P: percent of games won against opponents • Training experience E: playing practice games against itself ***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 classifications ***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 (as judged by human overseer) • Training experience E: a sequence of images and steering commands recorded while observing a human driver	6	CO1	L2

2)	**DESIGNING A LEARNING SYSTEM**
	The basic design issues and approaches to machine learning are illustrated by designing a program to learn to play checkers, with the goal of entering it in the world checkers tournament
	1. Choosing the Training Experience
	2. Choosing the Target Function
	3. Choosing a Function Approximation Algorithm
	    1. Estimating training values
	    2. Adjusting the weights
		10	CO1	L2

1. *Choosing the Training Experience*

- The first design choice is to choose the type of training experience from which the system will learn.
- The type of training experience available can have a significant impact on success or failure of the learner.

<u>There are three attributes which impact on success or failure of the learner</u>

1. Whether the training experience provides ***direct or indirect feedback*** regarding the choices made by the performance system.
   <u>For example, in checkers game</u>:
   In learning to play checkers, the system might learn from ***direct training examples*** consisting of ***individual checkers board states*** and ***the correct move for each***.
   Indirect training examples consisting of the ***move sequences*** and ***final outcomes*** of various games played. The information about the correctness of specific moves early in the game must be inferred indirectly from the fact that the game was eventually won or lost.
   Here the learner faces an additional problem of ***credit assignment***, or determining the degree to which each move in the sequence deserves credit or blame for the final outcome.
2. The degree to which the ***learner controls the sequence of training examples***
   <u>For example, in checkers game</u>:
   The learner might depends on the ***teacher*** to select informative board states and to provide the correct move for each.
   Alternatively, the learner might itself propose board states that it finds particularly confusing and ask the teacher for the correct move.
   The learner may have complete control over both the board states and (indirect) training classifications, as it does when it learns by playing against itself with ***no teacher present***.

2. *Choosing the Target Function*

The next design choice is to determine exactly what type of knowledge will be learned and how this will be used by the performance program.

Let's consider a checkers-playing program that can generate the legal moves from any board state.

The program needs only to learn how to choose the best move from among these legal moves. We must learn to choose among the legal moves, the most obvious choice for the type of information to be learned is a program, or function, that chooses the best move for any given board state.

1. Let ***ChooseMove*** be the target function and the notation is

   $$ChooseMove : B \rightarrow M$$

   which indicate that this function accepts as input any board from the set of legal board states B and produces as output some move from the set of legal moves M.
   ChooseMove is a choice for the target function in checkers example, but this function will turn out to be very difficult to learn given the kind of indirect training experience available to our system
2. An alternative target function is an ***evaluation function*** that assigns a ***numerical score*** to any given board state
   Let the target function V and the notation

   $$V : B \rightarrow R$$

   which denote that V maps any legal board state from the set B to some real value. Intend for this target function V to assign higher scores to better board states. If the system can successfully learn such a target function V, then it can easily use it to select the best move from any current board position.

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

- If b is a final board state that is won, then V(b) = 100
- If b is a final board state that is lost, then V(b) = -100
- If b is a final board state that is drawn, then V(b) = 0
- 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

3. Choosing a Function Approximation Algorithm

In order to learn the target function **f** we require a set of training examples, each describing a specific board state b and the training value Vtrain(b) for b.
Each training example is an ordered pair of the form (b, Vtrain(b)).
For instance, the following training example describes a board state b in which black has won the game (note x2 = 0 indicates that red has no remaining pieces) and for which the target function value Vtrain(b) is therefore +100.

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

<u>Function Approximation Procedure</u>

1. Derive training examples from the indirect training experience available to the learner
2. Adjusts the weights wi to best fit these training examples

1. <u>Estimating training values</u>

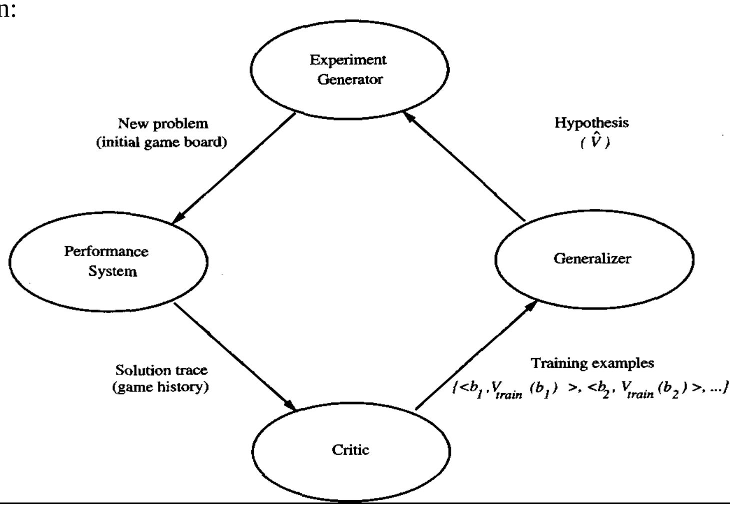
A simple approach for estimating training values for intermediate board states is to assign the training value of Vtrain(b) for any intermediate board state b to be V(Successor(b))
Where ,

- $\hat{V}$ is the learner's current approximation to V
- Successor(b) denotes the next board state following b for which it is again the program's turn to move

Rule for estimating training values

Vtrain(b) ← V (Successor(b))

Final Design:



3)	**FIND-S: FINDING A MAXIMALLY SPECIFIC HYPOTHESIS** **<u>FIND-S Algorithm</u>** 1. Initialize *h* to the most specific hypothesis in *H*	10	CO1	L2

2. For each positive training instance *x*

For each attribute constraint a_i in *h*

If the constraint a_i is satisfied by *x*

Then do nothing

Else replace a_i in *h* by the next more general constraint that is satisfied by *x* 3. Output hypothesis *h*

Unanswered by FIND-S

1. Has the learner converged to the correct target concept?
2. Why prefer the most specific hypothesis?
3. Are the training examples consistent?
4. What if there are several maximally specific consistent hypotheses?

4)

***List-Then-Eliminate* algorithm**

1. V*ersionSpace* ¬ a list containing every hypothesis in *H*
2. For each training example, <x,c(x)>, Remove from *VersionSpace* any hypothesis *h that is inconsistent ie.* for which *h*(*x*) ¹ *c*(*x*)
3. Output the list of hypotheses in *VersionSpace after checking all the training examples.*

For h1,

H1(x1)=C(x1)
H1(x2) ≠C(x2)
H1(x3) ≠C(x3)
H1(x4) ≠C(x4)

For h2,

H1(x1)=C(x1)
H1(x2)=C(x2)
H1(x3) ≠C(x3)
H1(x4) ≠C(x4)

For h3,

H1(x1)=C(x1)
H1(x2)=C(x2)
H1(x3) ≠C(x3)
H1(x4) ≠C(x4)

For h4,

H1(x1)=C(x1)
H1(x2)=C(x2)
H1(x3)=C(x3)
H1(x4)=C(x4)

x 1	Sunny	Warm	Normal	Strong	Warm	Same
x 2	Sunny	Warm	High	Strong	Warm	Same
x 3	Rainy	Cold	High	Strong	Warm	Change
x4	Sunny	Warm	High	Strong	Cool	Change

h_1 = *<Sunny Warm Normal Strong Warm Same>*

h_2 = *<Sunny Warm ? Strong Warm Same>*

h_3 = *<Sunny Warm ? Strong Warm Same>*

h_4 = *<Sunny Warm ? Strong ? ? >*

7) CANDIDATE-ELIMINTION algorithm begins by initializing the version space to the set of all hypotheses in H;

Initializing the G boundary set to contain the most general hypothesis in H G0

Initializing the S boundary set to contain the most specific (least general) hypothesis S0

10 CO1 L5

For training example d,
⟨Sunny, Warm, Normal, Strong, Warm, Same ⟩ +

S_0 ⟨∅, ∅, ∅, ∅, ∅. ∅⟩

S_1 ⟨*Sunny, Warm, Normal, Strong, Warm, Same*⟩

G_0, G_1 ⟨?, ?, ?, ?, ?, ?⟩

- <u>When the second training example is observed,</u> it has a similar effect of generalizing S further to S2, leaving G again unchanged i.e., G2 = G1 = G0

For training example d,
⟨Sunny, Warm, High, Strong, Warm, Same⟩ +

S_1 ⟨*Sunny, Warm, Normal, Strong, Warm, Same*⟩

S_2 ⟨*Sunny, Warm, ?, Strong, Warm, Same*⟩

G_1, G_2 ⟨?, ?, ?, ?, ?, ?⟩

<u>Consider the third training example</u>.

For training example d,
⟨Rainy, Cold, High, Strong, Warm, Change ⟩ –

S_2, S_3 ⟨*Sunny, Warm, ?, Strong, Warm, Same*⟩

G_3 ⟨*Sunny, ?, ?, ?, ?, ?*⟩ ⟨*?, Warm, ?, ?, ?, ?*⟩ ⟨*?, ?, ?, ?, ?, Same*⟩

G_2 ⟨?, ?, ?, ?, ?, ?⟩

<u>Consider the fourth training example</u>

For training example d,
⟨Sunny, Warm, High, Strong, Cool Change ⟩ +

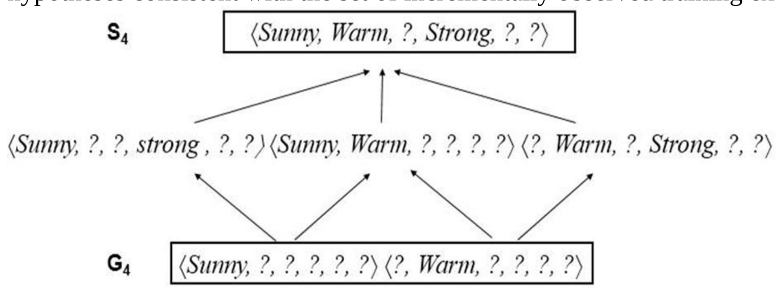
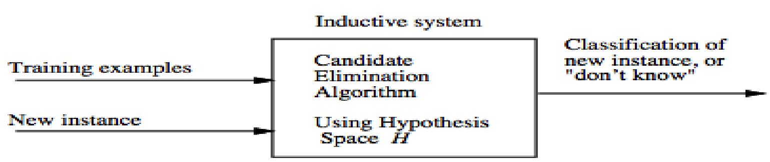
S_3 ⟨*Sunny, Warm, ?, Strong, Warm, Same*⟩

S_4 ⟨*Sunny, Warm, ?, Strong, ?, ?*⟩

G_4 ⟨*Sunny, ?, ?, ?, ?, ?*⟩ ⟨*?, Warm, ?, ?, ?, ?*⟩

G_3 ⟨*Sunny, ?, ?, ?, ?, ?*⟩ ⟨*?, Warm, ?, ?, ?, ?*⟩ ⟨*?, ?, ?, ?, ?, Same*⟩

After processing these four examples, the boundary sets S4 and G4 delimit the version space of all

	hypotheses consistent with the set of incrementally observed training examples. 			
8)	A Unbiased Hypothesis: (Includes all training examples) In enjoysport learning task the size of the instance space X of days described by the six attributes is 96 instances, which is 296 target values. But practically it is impossible to cover all the training examples. A Biased Hypothesis: (not all training examples are considered) If the candidate elimination algorithm is applied, then it ends up with empty version space. In that case the hypothesis is overly general and incorrectly covers the training example. So, the learner will generalize beyond the observed training examples to infer new examples. $x > y$ Here, y is inductively inferred from x where, X is a predefined example. The learning algorithm is represented as, $L(x_i, D_c)$ Where, x_i *is a new instance* D_c the training data, D_c = {á*x*, *c*(*x*)ñ} Therefore, the equation for *Inductive Bias is given as,* Inductive inference (≻) D_c Ù x_i≻$L(x_i, D_c)$ 	10	CO2	L2
5)	Decision tree learning is a method for approximating discrete-valued target functions, in which the learned function is represented by a decision tree. **DECISION TREE REPRESENTATION** - Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance. - Each node in the tree specifies a test of some attribute of the instance, and each branch descending from that node corresponds to one of the possible values for this attribute. - An instance is classified by starting at the root node of the tree, testing the attribute specified by this node, then moving down the tree branch corresponding to the value of the attribute in the given example. This process is then repeated for the subtree rooted at the new node.	10	CO2	L3

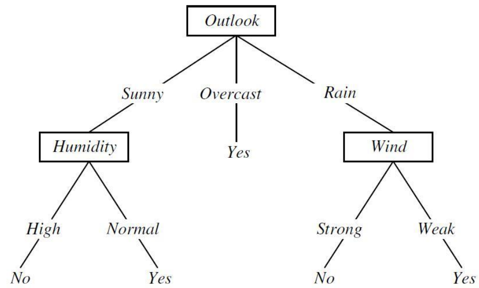
<table>
<tr>
<td></td>
<td>


FIGURE: A decision tree for the concept PlayTennis. An example is classified by sorting it through the tree to the appropriate leaf node, then returning the classification associated with this leaf.

Example:-

Let’s say we have a sample of 50 students with three variables Gender (Boy/ Girl), Class(X/ XI) and Height (5 to 6 ft). 20 out of these 50 play cricket in rest time. Suppose you want to find on unknown dataset which contains all the features(Gender, class, height) that he/she will play or not in rest time.

This is where decision tree supports, it will separate the students based on all values of three variable and identify the variable, which creates the best uniform sets of students
</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9)</td>
<td>
Step 1:
Total – 14 Yes(p) - 9 No(n) – 5
Attributes: Outlook ={Sunny, Overcast, Rain} Temperature = {Hot, Mild, Cool}
Humidity = {High, Normal} Wind ={Weak, Strong}
Step 2: Calculate the entropy of the dataset
Entropy(S) = $- p_+ \log_2 p_+ - p_- \log_2 p_-$
$= - \frac{p}{p+n} \log\left(\frac{p}{p+n}\right) - \frac{n}{p+n} \log\left(\frac{n}{p+n}\right)$
=- (9/(9+5)) log(9/(9+5)) –(5/(9+5)) log(5/(9+5))
=-(9/14) log(9/14) – (5/14) log(5/14)
= (-0.643)(-0.637) – (0.357)(-1.486)
=0.940
Step 3:
i) Select Outlook attribute
Outlook ={Sunny, Overcast, Rain}
Sunny : Yes(p)- 2 No(n)- 3
Overcast: Yes(p)- 4 No(n)- 0
Rain: Yes(p)- 3 No(n)- 2

a) Entropy of Outlook attribute

$= - \frac{p}{p+n} \log\left(\frac{p}{p+n}\right) - \frac{n}{p+n} \log\left(\frac{n}{p+n}\right)$
Entropy(Outlook = Sunny) = -(2/5)log(2/5) – (3/5)log(3/5)
= -(0.4)(-1.322) – (0.6)(-0.737) =0.971
Entropy(Outlook = Overcast) = -(4/4)log(4/4) – 0= 0
</td>
<td>10</td>
<td>CO2</td>
<td>L5</td>
</tr>
</table>

Entropy(Outlook = Rain) = -(3/5)log(3/5) – (2/5)log(2/5)

=-(0.6)(-0.737) – (0.4)(-1.322) = 0.971

b) Average Information Entropy(I)

I(Outlook) = ((2+3)/(9+5))*0.971 + ((3+2)/(9+5))*0.971 + 0

= (5/14)*0.971 + 0.3571*0.971

= 0.693

c) Information Gain(Outlook) = Entropy(S) –I(Outlook)

= *0.940 - 0.940 = 0.247*

ii) **Select Temperature attribute**

Temperature = {Hot, Mild, Cool}

Hot : p=2 n= 2

Mild: p=4 n=2

Cool: p=3 n=1

a. Calculate the entropy for Temperature

$$=- -\frac{p}{p+n} log\left(\frac{p}{p+n}\right) - \frac{n}{p+n} log\left(\frac{n}{p+n}\right)$$

Entropy(Temperature= Hot)= -(2/4)log(2/4) – (2/4)log(2/4)

= -(0.5)(-1) – (0.5)(-1)

= 1

Entropy(Temperature = Mild) = -(4/6)log(4/6) – (2/6)log(2/6)

= -(0.66)(-0.599) – (0.33)(-1.599)

= 0.923

Entropy(Temperature = Cool) = -(3/4)log(3/4) – (1/4)log(1/4)

= -(0.75)(-0.415) – (0.25)(-2)

= 0.811

b. Average Information Entropy(I)

I(Temperature) = (4/14)*1 + (6/14)* 0.923 + (4/14)* 0.811 = 0.913

c. Information Gain(Temperature)

IG(Temperature) = Entropy(S) – I(Temperature)

= *0.940* - 0.913

= 0.027

iii) **Select Humidity attribute**

Humidity = {High, Normal}

High: p: 3 n:4

Normal: p: 6 n: 1

a. Calculate the entropy for Temperature

$$=- -\frac{p}{p+n} log\left(\frac{p}{p+n}\right) - \frac{n}{p+n} log\left(\frac{n}{p+n}\right)$$

Entropy(Humidity = High) = - (3/7)log(3/7) – (4/7)log(4/7)

= -(0.4286)(-1.2223) – (0.5714)(-0.8074)

= 0.985

Entropy(Humidity = Normal) = -(6/7)log(6/7) – (1/7)log(1/7)

= -(0.8571)(-0.2225)-(0.1429)(-2.8069) = 0.591

b. Average Information Entropy

c. Average Information Entropy(I)

I(Humidity) = (7/14)*0.985 + (7/14)*0.591

= 0.788

d. Information Gain(Humidity)

IG(Humidity) = Entropy(S) – I(Humidity)

= *0.940 – 0.788*

= 0.152

iv) **Select Windy attribute**

Wind ={Weak, Strong}

Weak: p:6 n:2

Strong: p:3 n:3

a. Calculate the entropy for Windy

$$= - \frac{p}{p+n} \log\left(\frac{p}{p+n}\right) - \frac{n}{p+n} \log\left(\frac{n}{p+n}\right)$$

Entropy(Windy = Weak) = -(6/8)log(6/8) – (2/8)log(2/8)

= 0.811

Entropy(Windy = Strong) = -(3/6)log (3/6) – (3/6)log(3/6)

= 1

b. Average Information Entropy(I)

I(Windy) = (8/14)* 0.811 + (6/14)*1

= 0.892

c. Information Gain(Windy)

IG(Windy) = Entropy(S) – I(Windy)

= *0.940 – 0.892 = 0.048*

IG(Outlook) = *0.247*

IG(Temperature) = 0.27

IG(Humidity) = 0.152

IG(Windy) = *0.048*

Highest Information Gain is 0.247 -> Outlook

P:2 N:3 Total:5

Temperature= {hot, cool, mild}

Hot:p:0 n:2

Cool: p:1 n:0

Mild: p:1 n:1

Humidity={High, Normal}

High: p:0 n:3

Normal:p:2 n:0

Windy:{Weak, Strong}

Weak: p:1 n:2

Strong: p:1 n:1

1. Calculate the entropy of Dataset(S)

$$= - \frac{p}{p+n} \log\left(\frac{p}{p+n}\right) - \frac{n}{p+n} \log\left(\frac{n}{p+n}\right)$$

Entropy = -(2/5)log(2/5) – (3/5)log(3/5) = 0.971

2. Calculate the Information Gain

a. Calculate entropy of humidity

Entropy(Humidity = High) = 0 – (3/3)log(3/3) = 0

Entropy(Humidity = Normal) = 0

b. Calculate Average information entropy(I) of humidity

I(Humidity) = 0

c. Information gain of humidity

IG(Humidity) = Entropy(S) – I(Humidity)

=0.971 – 0 = 0.971

d. Calculate entropy of Windy

Windy:{Weak, Strong}

Weak: p:1 n:2

Strong: p:1 n:1

$$=- -\frac{p}{p+n}\log\left(\frac{p}{p+n}\right) - \frac{n}{p+n}\log\left(\frac{n}{p+n}\right)$$

Entropy(Windy = Weak) = -(1/3)log(1/3) – (2/3)log(2/3) = 0.918

Entropy(wind = Strong) = -(1/2)log(1/2) – (1/2) log(1/2) =1

e. Calculate Average information entropy of windy

I(Windy) = (3/5) *0.918 + (2/5)*1 = 0.951

f. Information gain of windy

IG(Windy) = Entropy(S) - I(Windy)

= 0.971- 0.951 = 0.020

g. Calculate entropy of temperature

Temperature= {hot, cool, mild}

Hot:p:0 n:2

Cool: p:1 n:0

Mild: p:1 n:1

$$=- -\frac{p}{p+n}\log\left(\frac{p}{p+n}\right) - \frac{n}{p+n}\log\left(\frac{n}{p+n}\right)$$

Entropy (Temperature = hot) = 0

Entropy(Temperature = Cool) = 0

Entropy(Temperature = mild) = -(1/2)log(1/2) – (1/2)log(1/2) = 1

h. Calculate Average information entropy of temperature

I(temperature) = (2/5) * 0 + (1/5)*0 + (2/5)*1= 0.4

i. Information gain of temperature

IG(Temperature) =0.971 - 0.4 = 0.571

3. Select the attribute with highest information gain

IG(Temperature) =0.971 - 0.4 = 0.571

IG(Windy) = 0.020

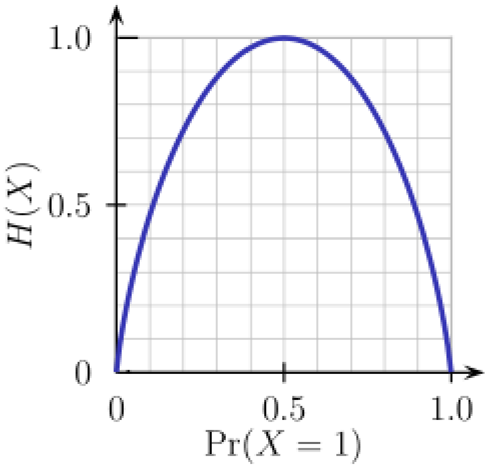
IG(Humidity) = 0.971

Total = 5

P=3

N= 2

	1. Calculate the entropy of the dataset(S) $\frac{p}{p+n} \boldsymbol{log}\left(\frac{p}{p+n}\right)$ - $\frac{n}{p+n} \boldsymbol{log}\left(\frac{n}{p+n}\right)$ Entropy = - (3/5)log(3/5) – (2/5)log(2/5) = 0.971 **2. Calculate the information gain** **a. Calculate entropy of temperature** Temperature ={mild, cool} Mild:p:2 n:1 Cool:p:1 n:1 Entropy(temperature = mild) = -(2/3)log(2/3) – (1/3)log(1/3) = 0.918 Entropy(temperature = cool) =-(1/2)log(1/2) – (1/2)log(1/2) = 1 **b. Calculate average information entropy of temperature** I(Temperature) = 0.951 **c. Information gain of temperature** 0.971– 0.951 = 0.20 **d. Calculate entropy of Humidity** Entropy(Humidity= High) = 1 Entropy(Humidity = Normal) = 0.918 **e. Calculate average information entropy of humidity** I(Humidity) = 0.951 **f. Information gain of humidity** Gain = 0.971 – 0.951 = 0.020 **g. Calculate entropy of Windy** Entropy(Windy = Strong) = 0 Entropy(Windy = Weak) = 0 **h. Calculate average information entropy of Windy** I(Windy) = 0 **i. Information gain of Windy** Gain = 0.971 – 0 = 0.971 **3. Select the attribute with highest information gain** Select Windy.			
10)	**Entropy** Entropy is a measure of the randomness in the information being processed. The higher the entropy, the harder it is to draw any conclusions from that information. Flipping a coin is an example of an action that provides information that is random.	10	CO2	L2



From the above graph, it is quite evident that the entropy H(X) is zero when the probability is either 0 or 1. The Entropy is maximum when the probability is 0.5 because it projects perfect randomness in the data and there is no chance if perfectly determining the outcome.

ID3 follows the rule — A branch with an entropy of zero is a leaf node and A brach with entropy more than zero needs further splitting.

Mathematically Entropy for 1 attribute is represented as:

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

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

Information Gain

Information Gain(T,X) = Entropy(T) - Entropy(T, X)

IG(PlayGolf, Outlook) = E(PlayGolf) - E(PlayGolf, Outlook)
= 0.940 - 0.693
= 0.247