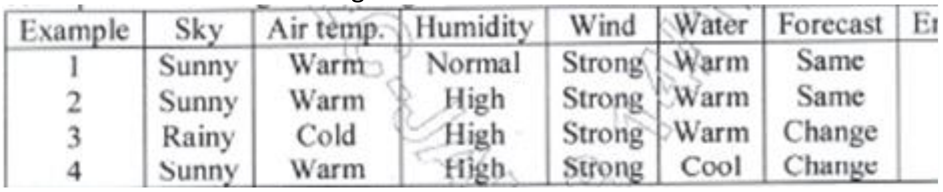
USN | | | | | | | | | | |

Internal Assessment Test 1 – September 2019

Sub:	Machine Learning				Sub Code:	15CS73	Branch:	CSE
Date:	14/09/2019	Duration:	90 min's	Max Marks: 50	Sem / Sec:	A/B/C		OBE

<u>Answer any FIVE FULL Questions</u>

		MARKS	CO	RBT
1 (a)	What is Machine Leaning? Explain applications of Machine Learning. Explain different perspectives and issues in machine learning	[1+2+2]	CO1	L2
(b)	Specify the learning task of i) A checkers learning problem ii) A handwriting recognition learning problem iii) A robot driving learning problem	[2+2+1]		
2 (a)	Explain steps to design a learning system in details, with example and figure.	[10]		
3 (a)	Describe the Find-S algorithm. And explain its working, taking the enjoy sport concept and training instances given in Table 1. Compare Find-S algorithm with Candidate Elimination Algorithm.	[8+2]		



Example	Sky	Air temp.	Humidity	Wind	Water	Forecast	E
1	Sunny	Warm	Normal	Strong	Warm	Same	
2	Sunny	Warm	High	Strong	Warm	Same	
3	Rainy	Cold	High	Strong	Warm	Change	
4	Sunny	Warm	High	Strong	Cool	Change	

Table 1.

4 (a)	Explain in detail candidate elimination algorithm using version spaces.	[10]		
5 (a)	Write down the version space using candidate elimination algorithm, taking the enjoy sports concepts and training instances given in Table 1.	[10]		
6 (a)	Discuss about i) A biased hypothesis space ii) An unbiased learner iii) The futility of biased learning	[3+4+3]		
7 (a)	What is a Decision Tree? Write ID3 algorithm.	[10]		

8 (a) Consider the below table and solve using the ID3 algorithm with decision tree. [10]

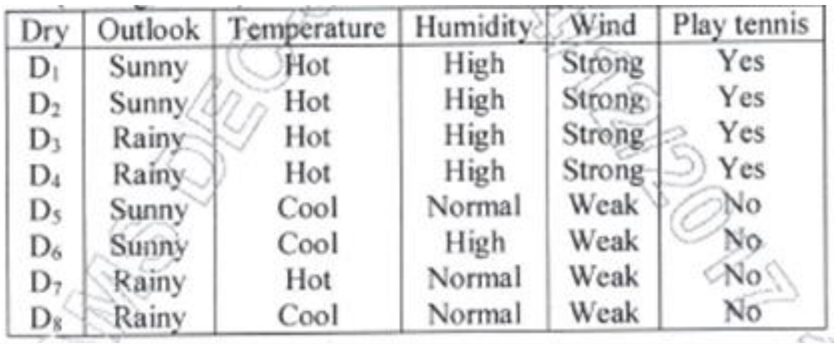
Consider attribute values:

Outlook(Sunny, Rainy)

Temperature(Hot, Cold)

Humidity(High, Normal)

Wind(Strong, Weak)



Dry	Outlook	Temperature	Humidity	Wind	Play tennis
D_1	Sunny	Hot	High	Strong	Yes
D_2	Sunny	Hot	High	Strong	Yes
D_3	Rainy	Hot	High	Strong	Yes
D_4	Rainy	Hot	High	Strong	Yes
D_5	Sunny	Cool	Normal	Weak	No
D_6	Sunny	Cool	High	Weak	No
D_7	Rainy	Hot	Normal	Weak	No
D_8	Rainy	Cool	Normal	Weak	No

Internal Assessment Test 1 – September 2019
15CS73 – Machine Learning

IAT-1 Solution

1.(a) What is Machine Leaning? Explain applications of Machine Learning. Explain different perspectives and issues in machine learning

Answer:

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.

Applications :

1.Learning to recognize spoken words.

2. Learning to drive an autonomous vehicle.
3.Learning to classify new astronomical structures.
4. Learning to play world-class backgammon.

SOME ISSUES IN MACHINE LEARNING

It involves searching a very large space of possible hypotheses to determine one that best fits the observed data and any prior knowledge held by the learner.

1. Algorithms for learning general target function from specific training examples.
2. Setting at which the particular algorithm converge to the desired function, given sufficient training data
3. Algorithm that perform best for the given types of problems and its representations
4. The Amount of training data required.
5. The prior knowledge needed to guide the process of generalizing from example.
6. The best strategy for choosing a useful next training experience, to alter the complexity of the learning Problem.
7. The best way to reduce the learning task to one or more function approximation problems.

8. The representation to improve its ability to represent and learn the target function

1(b) Specify the learning task of

i) **A checkers learning problem**

ii) **A handwriting recognition learning problem**

iii) **A robot driving learning problem**

i) 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

ii) 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

iii) A robot driving learning problem:

Task T: driving on public four-lane highways using vision sensors
Performance measure P: average distance traveled before an error (as judged by human overseer)
Training experience E: a sequence of images and steering commands recorded while observing a human driver

2. Explain steps to design a learning system in details, with example and figure

DESIGNING A LEARNING SYSTEM

Let us consider designing a program to learn to play checkers, with the goal of entering it in the world of checkers tournament. We adopt the obvious performance measure: the percent of games it wins in this world tournament.

a) Choosing the Training Experience

The first design choice we face is to choose the type of training experience from which our system will learn. The type of training experience available can have a significant impact on success or failure of the learner. One key attribute is whether the training experience provides direct or indirect feedback regarding the choices made by the performance system.

A second important attribute of the training experience is the degree to which the learner controls the sequence of training examples.

A third important attribute of the training experience is how well it represents the distribution of examples over which the final system performance P must be measured.

b) 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 us begin with 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.

The function **ChooseMove** and use the notation **ChooseMove : B → M** to 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.

Define the 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

c) Choosing a Representation for the Target Function

for any given board state, the function c will be calculated as a linear combination of the following board features:

w0+w1*bp(b)+w2*rp(b)+w3*bk(b)+w4*rk(b)+w5*bt(b)+w6*rt(b)

bp(b): number of black pieces on board b
rp(b): number of red pieces on b
bk(b): number of black kings on b
rk(b): number of red kings on b
bt(b): number of red pieces threatened by black (i.e., which can be taken on black's next turn)
rt(b): number of black pieces threatened by red

where w_0 through W_6 are numerical coefficients, or weights, to be chosen by the learning algorithm.

c) Partial design of a checkers learning program:

Task T: playing checkers
Performance measure P: percent of games won in the world tournament
Training experience E: games played against itself
Target function: V:Board → M
Target function representation

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

e) 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 $V_{train}(b)$ for b.

f) ESTIMATING TRAINING VALUES
This rule for estimating training values can be summarized as:

Rule for estimating training values.

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

g) ADJUSTING THE WEIGHTS

LMS weight update rule.

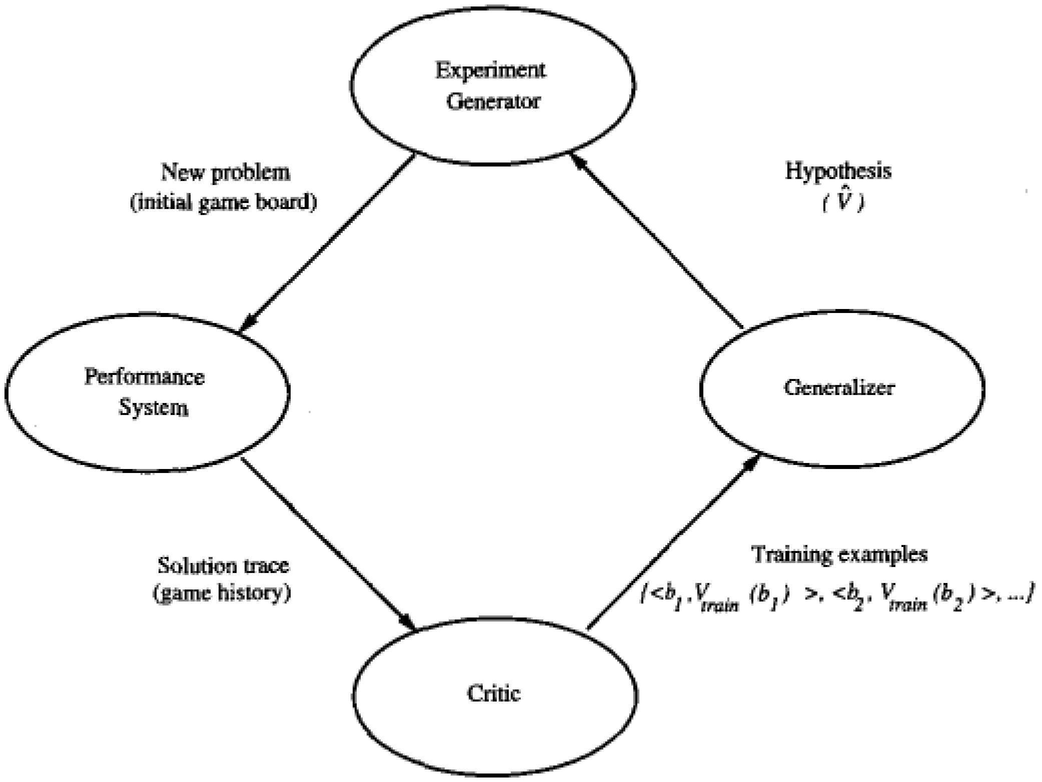
For each training example $\langle b, V_{train}(b)\rangle$

- Use the current weights to calculate $\hat{V}(b)$
- For each weight w_i, update it as

$$w_i \leftarrow w_i + \eta \; (V_{train}(b) - \hat{V}(b)) \; x_i$$

h) The Final Design
These four modules, summarized in Figure are as follows:

1.The **Performance System** is the module that must solve the given performance task, in this case playing checkers, by using the learned target function(s). It takes an instance of a new problem (new game) as input and produces a trace of its solution (game history) as output.

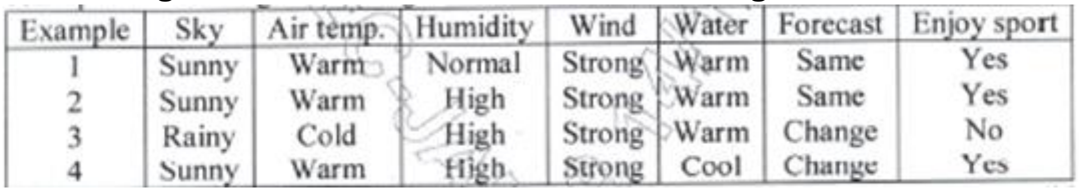

Experiment Generator
New problem (initial game board)
Hypothesis ($\hat{V}$)
Performance System
Generalizer
Solution trace (game history)
Training examples
$\{<b_1, V_{train}(b_1)>, <b_2, V_{train}(b_2)>, ...\}$
Critic

2.The **Critic** takes as input the history or trace of the game and produces as output a set of training examples of the target function. As shown in the diagram, each training example in this case corresponds to some game state in the trace, along with an estimate Vtraio,f the target function value for this example.

3.The **Generalizer** takes as input the training examples and produces an output hypothesis that is its estimate of the target function. It generalizes from the specific training xamples,hypothesizing a general function that covers these examples and other cases beyond the training examples.

4.The **Experiment Generator** takes as input the current hypothesis (currently learned function) and outputs a new problem (i.e., initial board state) for the Performance System to explore. Its role is to pick new practice problems that will maximize the learning rate of the overall system.

3.a Describe the Find-S algorithm. And explain its working, taking the enjoy sport concept and training instances given in Table 1. Compare Find-S algorithm with Candidate Elimination Algorithm.



Example	Sky	Air temp.	Humidity	Wind	Water	Forecast	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

Answer :

Find-S is guaranteed to output the most specific hypothesis h that best fits positive training examples.

- The hypothesis h returned by Find-S will also fit negative examples as long as training examples are correct.
- However, – Find-S is sensitive to noise that is (almost always) present in training examples. – there is no guarantee that h returned by Find-S is the only h that fits the data. – several maximally specific hypotheses may exist that fits the data but, Find-S will output only one.

Find-S Algorithm

- Initialize h to the most specific hypothesis in H
- For each positive training instance x
  - For each attribute constraint a_i in h
    - If the constraint a_i in h is satisfied by x
    - Then do nothing
    - Else replace a_i in h by the next more general constraint that is satisfied by x
- Output hypothesis h

	h_0 = <Ø, Ø, Ø, Ø, Ø, Ø>
x_1 = *<Sunny Warm Normal Strong Warm Same>*, +	h_1 = *<Sunny Warm Normal Strong Warm Same>*
x_2 = *<Sunny Warm High Strong Warm Same>*, +	h_2 = *<Sunny Warm ? Strong Warm Same>*
x_3 = *<Rainy Cold High Strong Warm Change>*, -	h_3 = *<Sunny Warm ? Strong Warm Same>*
x_4 = *<Sunny Warm High Strong Cool Change>*, +	h_4 = *<Sunny Warm ? Strong ? ? >*

Comparison:

- Candidate Elimination Algorithm (CEA) addresses several of the limitations of Find-S Algorithm.
- Find-S outputs a hypothesis from H that is consistent with the training examples.
- Find-S outputs one of the many hypotheses from H that might fit with the training data.
- Candidate Elimination Algorithm outputs the set of all hypotheses consistent with the training examples.
- CEA uses more_ general_than partial ordering to provide the consistent hypotheses.
- The subset of all hypotheses is called the ***Version Space*** (VS) with respect to hypotheses space H and the training examples D.

4. Explain in detail candidate elimination algorithm using version spaces.

<u>Candidate Elimination Algorithm</u>

```
G :maximally general hypotheses in H

S :maximally specific hypotheses in H

For each training example d=<x,c(x)>

Case 1 : If d is a positive example
    Remove from G any hypothesis that is inconsistent with d
    For each hypothesis s in S that is not consistent with d
      Remove s from S.
      Add to S all minimal generalizations h of s such that h
      consistent with d
      Some member of G is more general than h
        Remove from S any hypothesis that is more general than another
        hypothesis in S

Case 2: If d is a negative example
    Remove from S any hypothesis that is inconsistent with d
    For each hypothesis g in G that is not consistent with d
      Remove g from G.
      Add to G all minimal specializations h of g such that h
      consistent with d
      Some member of S is more specific than h
        Remove from G any hypothesis that is less general than another
        hypothesis in G
```

Remarks on the version space and Candidate elimination

The version space learned by the **CANDIDATE-ELIMINATION** algorithm will converge
toward the hypothesis that correctly describes the target concept, provided
(1) there are no errors in the training examples, and (2) there is some hypothesis
in H that correctly describes the target concept.

5. Write down the version space using candidate elimination algorithm, taking the enjoy sports concepts and training instances given in Table 1.

Candidate elimination algorithm:

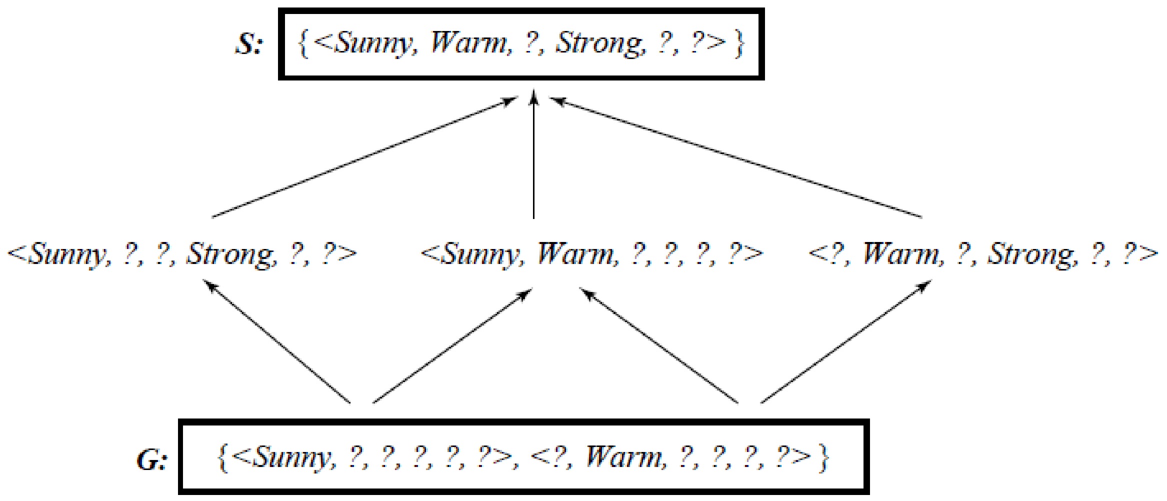
- Avoids enumeration of members in H
- Exploits more-general-than relation on H
- Exploits Version Space's compact specification by its two subsets
  - General boundary

- Specific boundary

- S0 = { < ∅, ∅, ∅, ∅, ∅, ∅> }
- G0 = { <?,?,?,?,?,?> }
- for every d in D
- if d is + then generalize S
- if d is – then specialize G

Enjoy sports Example :

Step 1:
d1 = (Sunny, Warm, Normal, Strong, Warm, Same) +
S1 = { <Sunny, Warm, Normal, Strong, Warm, Same> }
G1 = {<?,?,?,?,?,?> }
Step 2 :
d2 = (Sunny, Warm, High, Strong, Warm, Same) +
S2 = {<Sunny, Warm, ?, Strong, Warm, Same> }
G2 = {<?,?,?,?,?,?> }
Step 3 :
d3 = (Rainy, Cold, High, Strong, Warm, Change) –
S3 = { <Sunny, Warm, ?, Strong, Warm, Same>}
G3 = {<Sunny, ?,?,?,?,?>,
<?, Warm,?,?,?,?>
<?,?,?,?,?,Same> }
Step 4 :
d4 = (Sunny, Warm, High, Strong, Cool, Change) +
S4 = { <Sunny, Warm, ?, Strong, ?, ?> }
G4 = {<Sunny, ?,?,?,?,?>, <?, Warm,?,?,?,?> }
Version space generated by CEA:



6.a) Discuss about

iv) **A biased hypothesis space**

v) **An unbiased learner**

vi) **The futility of biased learning**

a) A biased hypothesis space

In the **CANDIDATE-ELIMINATION** algorithm, we restricted the hypothesis space to include only **conjunctions of attribute values**.
The hypothesis space is unable to represent simple **disjunctiv**e target concept
For example :

Example	**Sky**	**AirTemp**	**Humidity**	**Wind**	**Water**	**Forecast**	**EnjoySport**
1	Sunny	Warm	Normal	Strong	Cool	Change	Yes
2	Cloudy	Warm	Normal	Strong	Cool	Change	Yes
3	Rainy	Warm	Normal	Strong	Cool	Change	No

Suppose the hypothesis h0 in the hypothesis space H is **h0=(?,Warm,Normal,Strong,Cool,Change)**

This hypothesis is maximally specific and consistent with the first 2 examples.It covers the first 2 examples but incorrectly covers the third example. Because the third one is negative example.
This is because of the bias (considered only **conjunction** hypotheses).
We have not considered the simple **disjunctive** target concept such as:
Sky=Sunny **or** Sky = Cloudy.

b) An Unbiased learner

The solution to the problem of assuring that the target concept is in the hypothesis space H is to provide a hypothesis space capable of representing every concept.
We need to cover every possible subset of the instance X .In general , the set of all subsets of a set X is called the power set of X.
For Example : In ***EnjoySport*** learning task, there are 6 attributes:
Sky = {sunny,cloudy,rain} = >3 different values
AirTemp= {warm,cold} = > 2 different values
Humidity = {}   = > 2 different values
Wind={Strong, Weak} = > 2 different values
Water ={Warm,Cold} = > 2 different values
Forecast={Same,Change}= > 2 different values

The total no.of attributes in the instance space =3 x 2 x 2 x 2 x 2 x 2 = 96

The possible concept that can be defined over this set is 2^{96} (approximately $=10^{28}$ distinct target conept).but our conjunctive hypothesis space is able to represent only 973 (1 + 4x3x3x3x3x3=973 semantically distinct hypotheses).

Let us reformulate the Enjoysport learning task in an unbiased way by defining a new hypothesis space H' that can represent every subset of instances; that is, let H' correspond to the power set of X. One way to define such an H' is to allow arbitrary disjunctions, conjunctions, and negations of our earlier hypotheses.

c) The Futility of Bias-free Learning

Inductive inference is the process of reaching a general conclusion from specific examples. The general conclusion should apply to unseen examples.

One advantage of viewing inductive inference systems in terms of their inductive bias is that it provides a nonprocedural means of characterizing their policy for generalizing beyond the observed data. A second advantage is that it allows comparison of different learners according to the strength of the inductive bias they employ. Consider, for example the following **three learning algorithms**, which are listed from weakest to strongest bias.

1. **ROTE-LEARNER:**
Learning corresponds simply to storing each observed training example in memory. Subsequent instances are classified by looking them up in memory. If the instance is found in memory, the stored classification is returned. Otherwise, the system refuses to classify the new instance.
2. **CANDIDATE-ELIMINATION Algorithm**:
New instances are classified only in the case where all members of the current version space agree on the classification. Otherwise, the system refuses to classify the new instance.
3. **FIND-S**:
This algorithm, described earlier, finds the most specific hypothesis consistent with the training examples. It then uses this hypothesis to classify all subsequent instances.

The ROTE-LEARNER has no inductive bias. The classifications it provides for new instances follow deductively from the observed training examples, with no additional assumptions required.

The CANDIDATE-ELIMINATlON Algorithm has a stronger inductive bias: that the target concept can be represented in its hypothesis space. Because it has a stronger bias, it will classify some instances that the ROTE LEARNER will not. Of course the correctness of such classifications will depend completely on the correctness of this inductive bias.

The FIND-S algorithm has an even stronger inductive bias. In addition to the assumption that the target concept can be described in its hypothesis space, it has an additional inductive bias assumption: that all instances are negative instances unless the opposite is entailed by its other know1edge

7. What is a Decision Tree? Write ID3 algorithm.

Decision Tree

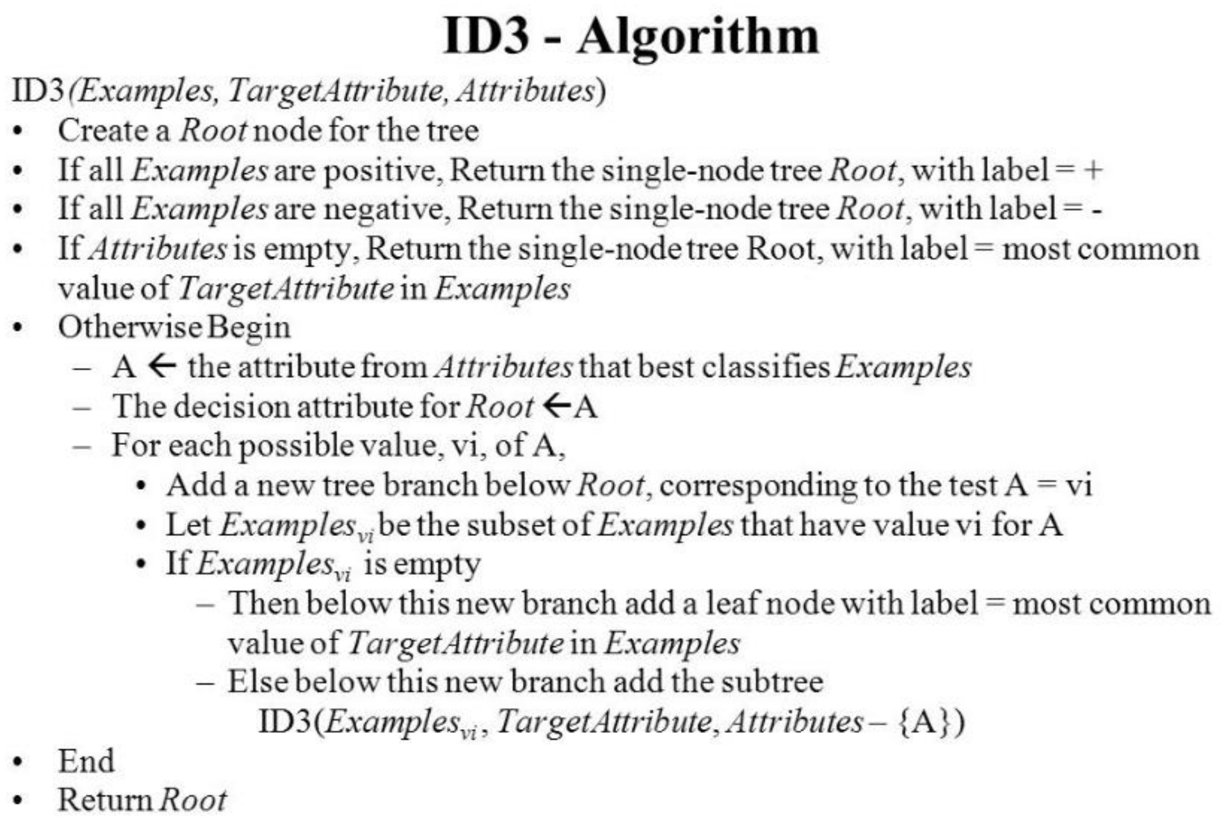
- A decision tree is a classifier

  Representation:

  - Node: an attribute which describes an instance
  - Branch: possible values of the attribute
  - Leaf: class to which the instances belong

- Procedure (of classifying):

  - An instance is classified by starting at the root node of the tree
  - Repeat: - test the attribute specified by the node - move down the tree branch corresponding to the value of the attribute-value in the given example

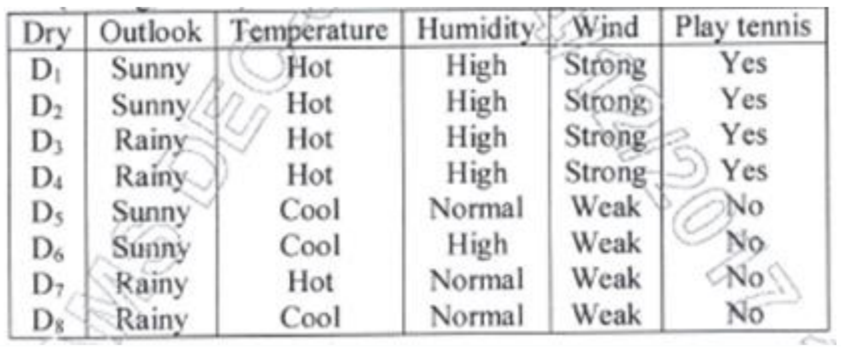


ID3 - Algorithm

```
ID3(Examples, TargetAttribute, Attributes)
• Create a Root node for the tree
• If all Examples are positive, Return the single-node tree Root, with label = +
• If all Examples are negative, Return the single-node tree Root, with label = -
• If Attributes is empty, Return the single-node tree Root, with label = most common
  value of TargetAttribute in Examples
• Otherwise Begin
    – A ← the attribute from Attributes that best classifies Examples
    – The decision attribute for Root ← A
    – For each possible value, vi, of A,
        • Add a new tree branch below Root, corresponding to the test A = vi
        • Let Examples_vi be the subset of Examples that have value vi for A
        • If Examples_vi is empty
            – Then below this new branch add a leaf node with label = most common
              value of TargetAttribute in Examples
            – Else below this new branch add the subtree
                ID3(Examples_vi, TargetAttribute, Attributes – {A})
• End
• Return Root
```

8 (a) **Consider the below table and solve using the ID3 algorithm with decision tree. Consider attribute values:**

Outlook(Sunny, Rainy)
Temperature(Hot, Cold)
Humidity(High, Normal)
Wind(Strong, Weak)



Dry	Outlook	Temperature	Humidity	Wind	Play tennis
D_1	Sunny	Hot	High	Strong	Yes
D_2	Sunny	Hot	High	Strong	Yes
D_3	Rainy	Hot	High	Strong	Yes
D_4	Rainy	Hot	High	Strong	Yes
D_5	Sunny	Cool	Normal	Weak	No
D_6	Sunny	Cool	High	Weak	No
D_7	Rainy	Hot	Normal	Weak	No
D_8	Rainy	Cool	Normal	Weak	No

Entropy(S) = -(4/8)log (4/8)-(4/8)log (4/8)
= 1 (equal number of positive and negative instances in each group)

Info.gain(S,Outlook) = Entropy(S) – [(4/8)* Entropy(sunny) +(4/8) *Entropy(Rainy)]

Entropy (sunny) = -(2/4) log(2/4) – (2/4) log(2/4)
= 1 (equal number of positive and negative instances in each group)
Entropy (Rainy) = -(2/4) log(2/4) – (2/4) log(2/4)
= 1 (equal number of positive and negative instances in each group)

Hence
Info.gain(S,Outlook) = Entropy(S) – [(4/8)*1 +(4/8) *1]
= 1- [(4/8) +(4/8)]
= 1 -1
= 0
Info.gain(S,Temperature) = Entropy(S) – [(5/8)* Entropy(Hot) +(3/8) *Entropy(Cool)]

Entropy (Hot) = -(4/5) log(4/5) – (1/5) log(1/5)
= -0.2575 - 0.4644
= - 0.7219
Entropy (Cool) = -(3/3) log(3/3) -0
= 0 (All instances are in same group)
Info.gain(S,Temperature) = Entropy(S) – [(5/8)*(-0.7219) +(3/8) *(0)]
= 1- 0.4512
= 0.55
Info.gain(S,Humidity) = Entropy(S) – [(5/8)* Entropy(High) +(3/8) *Entropy(Normal)]

Entropy (High) = -(4/5) log(4/5) – (1/5) log(1/5)
= -0.2575 - 0.4644
= - 0.7219

Entropy (Normal) = -(3/3) log(3/3) -0
= 0 (All instances are in same group)

Info.gain(S,Humidity) = Entropy(S) – [(5/8)*(-0.7219) +(3/8) *(0)]
= 1- 0.4512
= 0.55

Info.gain(S,Wind) = Entropy(S) – [(4/8)* Entropy(Strong) +(4/8) *Entropy(Weakl)]

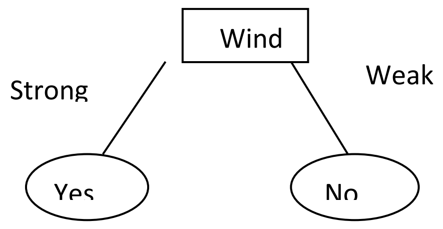
Entropy (Strong) = -(4/4) log(4/4) – 0
= (All instances are in same group)

Entropy (Weak) = 0-(4/4) log(4/4)
= 0 (All instances are in same group)

Info.gain(S,Wind) = Entropy(S) – [(4/8)* 0 +(4/8) *0)]
= 1-0
=1

Info.gain(S,Outlook) = 0
Info.gain(S,Temperature) =0.55
Info.gain(S,Humidity) = 0.55
Info.gain(S,Wind) =1

Since information gain of Wind is more than other attribute, it is selected as **root node** of the decision tree.
Instances - D1,D2,D3,D4 are added as left subtree and D5,D6,D7,D8 are added as right subtree under Root node. Since all the subsets belongs to the same group, they are labeled as Yes and No respectively in the level -2 of the tree


Wind
Strong
Weak
Yes
No
