CMR INSTITUTE OF TECHNOLOGY

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

Internal Assesment Test – I

Sub:	MACHINE LEARNING			Sec	A			Code:	15EC834
Date:	07/03/2019	Duration:	90 mins	Max Marks:	50	Sem:	VIII	Branch:	TCE

Solution

1 *Discuss the following for the learning task ' A Checker Learning Problem'*

a) Choosing the training experience

b) Choosing the target function

c) Choosing the function approximation algorithm

Ans: ***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

[1 mark]

To design a learning system, the below steps should be followed:

1. Choosing the Training Experience
2. Choosing the Target Function
3. Choosing a Representation for the Target Function
4. Choosing a Function Approximation Algorithm
   a) Estimating training values
   b) Adjusting the weights
5. The Final Design

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.

[3 marks]

<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.
2. The degree to which the ***learner controls the sequence of training examples***
3. How well it represents the ***distribution of examples*** over which the final system performance P must be measured

Here the training example chosen as playing games against itself.

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.

Choose the best move for any given board state.

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

   ChooseMove : B→ 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

   [3 marks]
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:

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

3. Choosing a Representation for the Target Function

Let's choose a simple representation for any given board state, the function $\hat{V}$ will be calculated as a linear combination of the following board features:

- x_1 : the number of black pieces on the board
- x_2 : the number of red pieces on the board
- x_3 : the number of black kings on the board
- x_4 : the number of red kings on the board
- x_5 : the number of black pieces threatened by red (i.e., which can be captured on red's next turn)
- x_6 : the number of red pieces threatened by black

[3 marks]

Thus, learning program will represent as a linear function of the form

$\hat{V}(b) = \omega_0 + \omega_1 x_1 + \omega_2 x_2 + \omega_3 x_3 + \omega_4 x_4 + \omega_5 x_5 + \omega_6 x_6$

Where,

- ω_0 Through ω_6 are numerical coefficients, or weights, to be chosen by the learning algorithm.
- Learned values for the weights ω_1 through ω_6 will determine the relative importance of the various board features in determining the value of the board
- The weight ω_0 will provide an additive constant to the board value

2(a) *What do you mean by a well –posed learning problem? Explain the important features that are required to well –define a learning problem*

WELL-POSED LEARNING PROBLEMS

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. **[2 marks]**

To have a well-defined learning problem, three features needs to be identified:

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

[3 marks]

Examples

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

2(b) *Comment on Issues in Machine Learning*

The field of machine learning, and much of this book, is concerned with answering questions such as the following

i. What algorithms exist for learning general target functions from specific training examples? In what settings will particular algorithms converge to the desired function, given sufficient training data? Which algorithms perform best for which types of problems and representations? **[5 marks]**

ii. How much training data is sufficient? What general bounds can be found to relate the confidence in learned hypotheses to the amount of training experience and the character of the learner's hypothesis space?

iii. When and how can prior knowledge held by the learner guide the process of generalizing from examples? Can prior knowledge be helpful even when it is only approximately correct?

iv. What is the best strategy for choosing a useful next training experience, and how does the choice of this strategy alter the complexity of the learning problem?

v. What is the best way to reduce the learning task to one or more function approximation problems? Put another way, what specific functions should the system attempt to learn? Can this process itself be automated?

vi. How can the learner automatically alter its representation to improve its ability to represent and learn the target function?

3 *What is decision tree? Discuss the ID3 algorithm in details.*

Decision tree learning is most widely used for inductive inference. It is robust to noisy data and capable of learning disjunctive expressions.

In Decision Tree learning method, the learned function is represented by a decision tree.

- The widely used Decision Tree algorithms are: **[2 marks]**
  a) ID3(Iterative Dichotomiser)
  b) ASSISTANT
  c) C4.5
  d) CART (Classification and Regression Tree)

ID3(Examples, Target_attribute, Attributes)

Examples are the training examples. Target_attribute is the attribute whose value is to be predicted by the tree. Attributes is a list of other attributes that may be tested by the learned decision tree. Returns a decision tree that correctly classifies the given Examples.

- 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 Target_attribute 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 Target_attribute in Examples
      - Else below this new branch add the subtree

[8 marks]

ID3(*Examples vi*, Targe_tattribute, Attributes – {A}))

- End

Return Root

4(a) *Explain inductive bias through Candidate Elimination Algorithm*

The fundamental questions for inductive inference

1. What if the target concept is not contained in the hypothesis space?
2. Can we avoid this difficulty by using a hypothesis space that includes every possible hypothesis?
3. How does the size of this hypothesis space influence the ability of the algorithm to generalize to unobserved instances?
4. How does the size of the hypothesis space influence the number of training examples that must be observed? **[5 marks]**

These fundamental questions are examined in the context of the CANDIDATE-ELIMINTION algorithm

A Biased Hypothesis Space:

To correctly provide solution for unobserved instances, the hypothesis space has to be enriched with every possible hypothesis

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 teachable concept that is representing every possible subset of the instances X.
- But the difficulty in this method: our concept learning algorithm completely unable to generalize beyond the training example. Here the hypothesis can correctly classify only the observed training example, it can't correctly classify for any unobserved instances.

The Futility of Bias-Free Learning

- Inductive learning requires some form of prior assumptions, or inductive bias
- The key idea is to know the procedure by which the learner generalizes beyond the observed training data to classify the new instances.
- The inductive bias of L is any minimal set of assertions B such that for any target concept c and corresponding training examples D_c

$$(\forall x_i \in X)[(B \wedge D_c \wedge x_i) \vdash L(x_i, D_c)]$$

4(b) Explain the inductive bias in decision tree learning.

INDUCTIVE BIAS IN DECISION TREE LEARNING

Inductive bias is the set of assumptions that, together with the training data, which correctly classifies the future instances.

For the given training examples, there are can be many decision trees consistent with these examples, which of these decision trees does ID3 choose?

ID3 Search Strategy: **[5 marks]**

a) Select in favor of shorter tree over longer ones.

b) Select trees that place the attributes with highest information gain closest to the root.

Approximate inductive bias of ID3: Shorter trees are preferred over larger trees

- Consider an algorithm that begins with the empty tree and searches breadth first through progressively more complex trees.
- First considering all trees of depth 1, then all trees of depth 2, etc.
- Once it finds a decision tree consistent with the training data, it returns the smallest consistent tree at that search depth (e.g., the tree with the fewest nodes).
- Let us call this breadth-first search algorithm BFS-ID3.
- BFS-ID3 finds a shortest decision tree and thus exhibits the bias "shorter trees are preferred over longer trees.
- ID3 can be viewed as an efficient approximation to BFS-ID3, using a greedy heuristic search to attempt to find the shortest tree without conducting the entire breadth-first

search through the hypothesis space.

- Because ID3 uses the information gain heuristic and a hill climbing strategy, it exhibits a more complex bias than BFS-ID3.
- In particular, it does not always find the shortest consistent tree, and it is biased to favour trees that place attributes with high information gain closest to the root.

A closer approximation to the inductive bias of ID3: Shorter trees are preferred over longer trees. Trees that place high information gain attributes close to the root are preferred over those that do not

5(a) *Explain the List –Then- Eliminate Algorithm with an example*

The LIST-THEN-ELIMINATE Algorithm:

- The LIST-THEN-ELIMINATE algorithm first initializes the version space to contain all hypotheses in ***H***, and then eliminates the hypothesis found inconsistent with training example.
- The version space shrinks as more examples are observed and finally one hypothesis remains.
- The LIST-THEN_ELIMINATE algorithm can be applied when the hypothesis space ***H*** is finite.

The LIST-THEN-ELIMINATION Algorithm:

1. Version Space ← a list containing every hypothesis in ***H.***
2. For each training example $\langle x\,, c(x)\rangle$
   Remove from version space any hypothesis $\boldsymbol{h}$ for which $\boldsymbol{h}(\boldsymbol{x}) \neq \boldsymbol{c}(\boldsymbol{x})$
3. Output the list of hypothesis in Version Space

[3 marks]

Example:

A	B	A XOR B
F	F	F
F	T	T
T	F	T
T	T	F

[2 marks]

$H : \{\langle T, ?\rangle, \langle F, ?\rangle, \langle ?, T\rangle, \langle ?, F\rangle, \langle F, F\rangle, \langle F, T\rangle, \langle T, F\rangle, \langle T, T\rangle\}$ *initial Hypothesis Space*

$H : \{\langle F, T\rangle, \langle T, F\rangle\}$ *Final Hypothesis Space*

Example:

Example	Sky	AirTemp	EnjoySport
1	Sunny	Warm	Yes
2	Sunny	Warm	Yes
3	Rainy	Cold	No
4	Sunny	Warm	Yes

$H : \{\langle Sunny, ?\rangle, \langle ?, Warm\rangle, \langle Rainy, ?\rangle, \langle ?, Cold\rangle, \langle Sunny, Warm\rangle, \langle Sunny, Cool\rangle,$
$< Rainy, Warm >, < Rainy, Cool >\}$ *initial Hypothesis Space*

$H : \{\langle Sunny, ?\rangle, \langle Sunny, Warm\rangle, \langle Sunny, Cool\rangle, >\}$ *Final Hypothesis Space*

5(b) *List the issues in Decision Tree Learning*

Issues in learning decision trees include:

1. Avoiding Overfitting the Data:
   a) Reduced error pruning
   b) Rule post pruning
2. Incorporating Continuous-Valued Attributes
3. Alternative Measures for Selecting Attributes
4. Handling Training Examples with Missing Attribute Values
5. Handling Attributes with Differing Costs

[5 marks]

6 *Apply the CANDIDATE-ELIMINATION Algorithm to find the final version space for the following*

example

Origin	*Manufacturer*	*Color*	*Decade*	*Type*	*ExampleType*
Japan	*Honda*	*Blue*	*1980*	*Economy*	*Yes*
Japan	*Toyota*	*Green*	*1970*	*Sports*	*No*
Japan	*Toyota*	*Blue*	*1990*	*Economy*	*Yes*
USA	*Chrystler*	*Red*	*1980*	*Economy*	*No*
Japan	*Honda*	*White*	*1980*	*Economy*	*Yes*

Ans Solution: **[8 marks]**

$G_0 = \{<?,?,?,?,?,?>\}$ $S_0 = \{\langle \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset \rangle\}$

$S_1 = \{\langle Japan, Honda, Blue, 1980, Economy \rangle\}$

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

$S_2 = \{\langle Japan, Honda, Blue, 1980, Economy \rangle\}$

$G_2 = \{<?, Honda, ?, ?, ?>, <?, ?, Blue, ?, ?>, <?, ?, ?, 1980, ?>, <?, ?, ?, ?, Economy>\}$

$S_3 = \{\langle Japan, ?, Blue, ?, Economy \rangle\}$

$G_3 = \{<?, ?, Blue, ?, ?>, <?, ?, ?, ?, Economy>\}$

$S_4 = \{\langle Japan, ?, Blue, ?, Economy \rangle\}$

$G_4 = \{<?, ?, Blue, ?, ?>, <Japan, ?, ?, ?, Economy>\}$

$S_5 = \{\langle Japan, ?, ?, ?, Economy \rangle\}$

$G_5 = \{<Japan, ?, ?, ?, Economy>\}$

Diagram showing version space **[2 marks]**

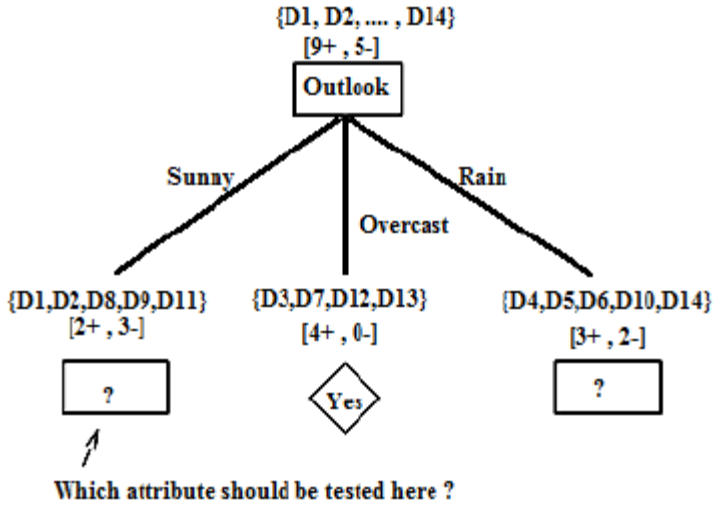
7 Illustrate the operation of ID3 for the following training examples given in the table below. Here the target attribute is Play Tennis. Draw the complete decision tree

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Ans The information gain values for all four attributes are **[2 marks]**

$G(S, Outlook)$	0.2465
$G(S, Temperature)$	0.023
$G(S, Humidity)$	0.1515
$G(S, Wind)$	0.048

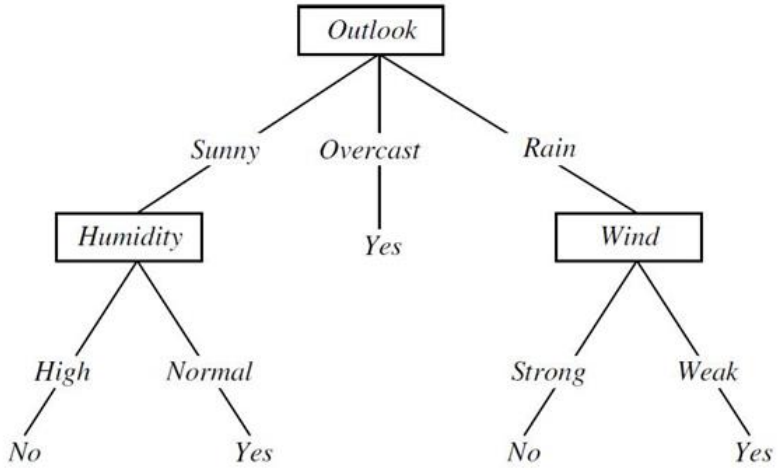
According to the information gain measure, the ***Outlook*** attribute provides the best prediction of the target attribute, ***PlayTennis***, over the training examples. Therefore, ***Outlook*** is selected as the decision attribute for the root node, and branches are created below the root for each of its possible values i.e., Sunny, Overcast, and Rain



As the information gain of ***Humidity*** is greater under ***Outlook=Sunny*** the next node should be ***Humidity*****.** **[2 marks]**

As the information gain of ***Wind*** is greater under ***Outlook=Rain*** the next node should be ***Wind.*** **[2 marks]**

The final tree is:



[4 marks]