Scheme of Evaluation
Internal Assessment Test 1 – Sep.2019

Sub:	Machine Learning							Code:	15CS73
Date:	17/09/2019	Duration:	90mins	Max Marks:	50	Sem:	VII	Branch:	ISE A,B

Note: Answer Any Five Questions

Question #		Description	Marks Distribution		Max Marks
1		**Explain the steps in designing learning systems in detail**. **Explaining all the 5steps with examples** 1. Choosing the Training Experience 2. Choosing the Target Function 3. Choosing a Representation for the Target Function 4. Choosing a Function Approximation Algorithm 1. Estimating training values 2. Adjusting the weights 5. The Final Design	2M 5*2=10M	10M	10 M
2	a)	**What is well- posed learning problems. Describe the following problems with respect to Tasks, Performance and Experience:** • Definition of well posed learning problem • Describing the tasks, performance and E for all 3 problems	1M 3*1=3M	4 M	10 M
	b)	**Define Machine Learning. Explain with examples why machine learning is important and discuss some applications of machine learning.** • Definition of ML • Importance of Machine Learning • Applications of Machine Learning	1M 2M 3M	6 M	
3	a)	**Explain different perspective and issues in machine learning** • Explaining any 4 issues	4M	4M	10 M
	b)	**Describe the Find-S algorithm. Explain the process of finding maximally specific hypothesis for the below dataset.** • Explaining Find-S algorithm • Problem solving by finding specific hypothesis	2M 4M	6 M	
4		**Find the maximally general hypothesis and maximally specific hypothesis for the training examples given in the table using candidate elimination algorithm.** • Initializing specific hypothesis and generic hypothesis • For each positive and negative instances:	2M	10 M	10 M

		• Finding the maximally specific and generic hypothesis	8M		
5	a)	**Explain inductive bias through candidate elimination algorithm.** Explaining the below with examples. • Biased hypothesis space • Unbiased learner • The futility of Bias-free learning	3*2=6 M	6 M	10 M
	b)	**Explain List-Then-Eliminate algorithm.** Defining and explaining the algorithm	4 M	4 M	
6		**Find the maximally general hypothesis and maximally specific hypothesis for the training examples given in the table using candidate elimination algorithm.** • Initializing specific hypothesis and generic hypothesis • For each positive and negative instances: Finding the maximally specific and generic hypothesis	2 M 8 M	10 M	10M
7	a)	**List the drawbacks of Find-S algorithm. Explain candidate elimination algorithm in detail.** • Drawbacks of Find-S • Explanation of candidate elimination algorithm	2M 3M	5M	10M
	b)	**What do you mean by hypothesis space, instance space and version space?** • Defining hypothesis space, instance space • Defining version space	1.5*2=3M 2M	5M	

Internal Assessment Test 1 Solutions– Sept.2019

Sub:	Machine Learning							Code:	15CS73
Date:	17/09/2019	Duration:	90mins	Max Marks:	50	Sem:	VII	Branch:	ISE A,B

Note: Answer Any Five Questions

1. Explain the steps in designing learning systems in detail.(10M)

Solution:

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

1. Choosing the Training Experience:

- The type of training experience available can have a significant impact on success or failure of the learner.

There are three attributes which impact on success or failure of the learner

1. One key attribute is whether the training experience provides direct or indirect feedback regarding the choices made by the performance system.
   - For ex in learning to play checkers the system might learn from direct training examples consisting of individual checkers board states and correct move for each.
   - Indirect information consisting of the move sequences and final outcomes of various games played. 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.

     Hence, learning from direct training feedback is typically easier than learning from indirect feedback.

2. A second important attribute of the training experience is the **degree to which the learner controls the sequence of training examples**.
   - For example, the learner might rely 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. Or 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.

3. 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.

   For ex in checkers game: the performance metric P is the percent of games the system wins in the world tournament.

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. Consider checkers-playing program. The program needs only to learn how to choose the best move from among some large search space. Here we discuss two such methods.

Method-1: Let us use the function ***ChooseMove: B → M***
Which indicate that this function accepts any board from the set of legal board states B as input and produces as output some move from the set of legal moves M.

ChooseMove is a key design choice for the target function in checkers example, this function will turn out to be very difficult to learn given the kind of indirect training experience available to our system.

Method-2: An alternative target function and one that will turn out to be easier to learn in this setting is an evaluation function that assigns a numerical score to any given board state.
Let us call this target function V and again use the notation ***V: B →ℜ***
Which denote that V maps any legal board state from the set B to some real value in ℜ. 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 Representation for the Target Function

- Let's choose a simple representation for any given representation, for any given state the function c is calculated as a linear combination of the following board features.
  - xl: the number of black pieces on the board
  - x2: the number of red pieces on the board
  - x3: the number of black kings on the board
  - x4: the number of red kings on the board
  - x5: the number of black pieces threatened by red (i.e., which can be captured on red's next turn)
  - x6: the number of red pieces threatened by black

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

$$\hat{V}(b) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4 + w_5 x_5 + w_6 x_6$$

Where w_0 through w_6 are the weights to be chosen by the Learning Algorithm.

4 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.
- Each training example is an ordered pair of the form (b, $V_{train}(b)$).
- For instance, the following training example describes a board state b in which black has won the game (note $x_2 = 0$ indicates that red has no remaining pieces) and for which the target function value $V_{train}(b)$ is therefore +100.

  ((x_1=3, x_2=0, x_3=1, x_4=0, x_5=0, x_6=0), +100)

Function approximation Procedure:

1. Estimating training values
2. Adjusting the weights

1. Estimating training values:

- The only training information available to our learner is whether the game was eventually **won or lost**.
- The approach is to assign the training value of $V_{train}(b)$ for any intermediate board state *b* to be V^ ***(****Successor(b))*

Rule for estimating training values: $V_{train}(b) \leftarrow \hat{V}(Successor(b))$

2. Adjusting the weights:

- One common approach is to define the best hypothesis, or set of weights, as that which minimizes the square error *E between* the training values and the values predicted by the hypothesis *V*.

$$E \equiv \sum_{\langle b, V_{train}(b)\rangle \in \text{ training examples}} (V_{train}(b) - \hat{V}(b))^2$$

We require an algorithm that will incrementally refine the weights as new training examples become available and that will be robust to errors in these estimated training values.
One such algorithm is called the least mean squares (LMS) training rule.

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

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

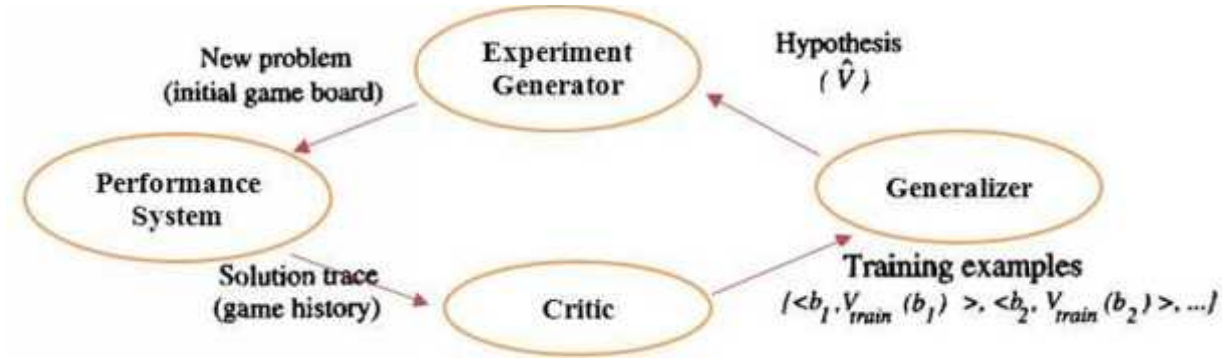
Here η is a small constant (e.g., 0.1) that moderates the size of the weight update.

5. The final design:

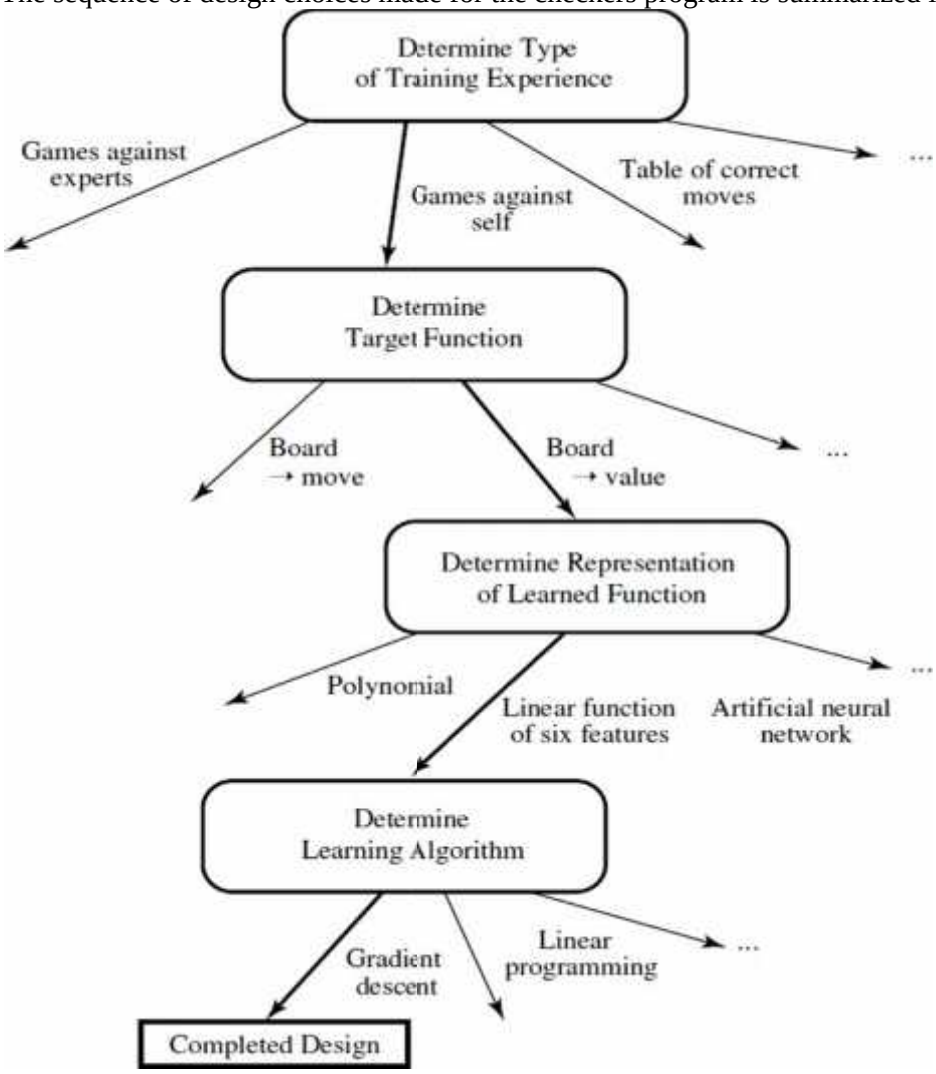
The final design of our checkers learning system can be naturally described by four distinct program modules that represent the central components in many learning systems.

a) 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.
b) The **Critic** takes as input - history or trace of the game and produces as output - a set of training examples of the target function.
c) 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 examples, hypothesizing a general function that covers these examples and other cases beyond the training examples.
d) 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.

These four modules are summarized as follows



The sequence of design choices made for the checkers program is summarized in figure given below.



2.a. What is well- posed learning problems. Describe the following problems with respect to Tasks, Performance and Experience: (4M)

Solution:

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. 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

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

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

b. Define Machine Learning. Explain with examples why machine learning is important and discuss some applications of machine learning. (6M)

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.

- Some tasks cannot be defined well, except by examples (e.g., recognizing people).
- Relationships and correlations can be hidden within large amounts of data. Machine Learning/Data Mining may be able to find these relationships.
- Human designers often produce machines that do not work as well as desired in the environments in which they are used.
- The amount of knowledge available about certain tasks might be too large for explicit encoding by humans (e.g., medical diagnostic).
- Environments change over time.
- New knowledge about tasks is constantly being discovered by humans. It may be difficult to continuously re-design systems "by hand".

Applications:

- Learning to recognize spoken words
- Learning to drive an autonomous vehicle
- Learning to classify new astronomical structures
- Learning to play world-class backgammon
- Learning to recognize images

3.a. Explain different perspective and issues in machine learning (4M)

- 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?
- 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?
- 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?
- 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?

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

b. Describe the Find-S algorithm. Explain the process of finding maximally specific hypothesis for the below dataset.

Restaurant	Meal	Day	Cost	Target Function
Sam's	Breakfast	Friday	Cheap	Yes
Hilton	Lunch	Friday	Expensive	No
Sam's	Lunch	Saturday	Cheap	Yes
Dannie	Breakfast	Sunday	Cheap	No
Sam's	Breakfast	Sunday	Expensive	No

FIND-S Algorithm

1. Initialize *h* to the most specific hypothesis in *H*
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*

Problem:

x_1 = <Sam's,Breakfast,Friday,Cheap>, +

Observing the first training example, it is clear that our hypothesis is too specific. In particular, none of the "Ø" constraints in h are satisfied by this example, so each is replaced by the next more general constraint that fits the example

h_1 = <Sam's, Breakfast,Friday,Cheap>

x_2 = <Hilton,Lunch,Friday,Expensive>, -
Since it is negative no change in h and h2 = <Sam's, Breakfast, Friday, Cheap>

x3=<Sam's,Lunch,Saturday,Cheap> +
Compare each instance of x3 with h2 and replace it with ?
h3= <Sam's,?,?,Cheap>

x4=<Dannie,Breakfast,Sunday,Cheap> -ve
Since it is negative no change in h3 and h4 = <Sam's,?,?,Cheap>

x5=<Sam's,Breakfast,Sunday,Expensive> -ve
Since it is negative no change in h4 and h5 = <Sam's,?,?,Cheap>

4. Find the maximally general hypothesis and maximally specific hypothesis for the training examples given in the table using candidate elimination algorithm.

Origin	Manufacturer	Color	Decade	Type	Target Value
Japan	Honda	Blue	1980	Economy	Y
Japan	Toyota	Green	1970	Sports	N
Japan	Toyota	Blue	1990	Economy	Y
USA	Chrysler	Red	1980	Economy	N

Japan	Honda	White	1980	Economy	Y
Japan	Toyota	Green	1980	Economy	Y
Japan	Honda	Red	1990	Economy	N

1. **S0={0,0,0,0,0} - 2 Marks**
   G0={?,?,?,?,?}

2. **First example d: {Japan,Honda,Blue,1980,Economy} +ve**
   S1= {Japan,Honda,Blue,1980,Economy}
   G1={?,?,?,?,?}

   Second example d: {Japan,Toyota,Green,1970,Sports} –ve
   S2={Japan,Honda,Blue,1980,Economy}
   Try to make each ? with different possible pairs
   G2 :
   {USA,?,?,?,?}{?,Honda,?,?,?}{?,Chrysler,?,?,?}{?,?,Blue,?,?}{?,?,Red,?,?}{?,?,White,?,?}
   {?,?,?,1980,?}{?,?,?,1990,?}{?,?,?,?,Economy}
   Try to remove inconsistent pairs and keep only consistant ones.Pairs 2,4,7 and 9 are consistant with specific hypothesis S so keep only this and remaining pairs ignore it.
   G2 : {?,Honda,?,?,?}{?,?,Blue,?,?}{?,?,?,1980,?}{?,?,?,?,Economy}

   Third example d: {Japan,Toyota,Blue,1990,Economy} +ve
   S3 ={Japan,?,Blue,?,Economy}
   Make G3 more consistant pairs by removing less consistant pairs with Specific hypothesis.
   G3 ={?,?,blue,?,?}{?,?,?,?,Economy} because first and 3rd pair is not consistant with specific hypothesis.

   Fourth example d: {USA,Chrysler,Red,1980,Economy} -ve
   S4 ={Japan,?,blue,?,Economy}
   For every pair and ? in G make it specific
   G4= {Japan,?,?,?,Economy} {?,Toyota,?,?,Economy} {?,Honda,?,?,Economy}
   {?,?,Blue,?,Economy}{?,?,Green,?,Economy}{?,?,White,?,Economy}
   {?,?,?,1970,Economy}{?,?,?,1990,Economy}
   G after removing less general hypothesis : {Japan,?,?,?,Economy}{?,?,blue,?,?}
   {?,?,blue,?,?} – This is consistant with S so keep as it is..
   G4 = {Japan,?,?,?,Economy}{?,?,blue,?,?}

Fifth example d: {Japan,Honda,White,1980,Economy} +ve
S5 = {Japan,?,?,?,Economy}
Among the two G pairs try to keep consistant one.
G5={Japan,?,?,?,Economy}

Sixth example d: {Japan, Honda, Red, 1990, Economy} -ve
S6={Japan,?,?,?,Economy}
Here my S6 is matching with d hence S6 is +ve where as d is –ve there by in-consistant so make S6={ } and G6={}
Specific hypothesis = {Japan,?,?,?, Economy} **From 2nd step till last-8Marks**
General Hypothesis = {Japan,?,?,?, Economy}

5.a)Explain inductive bias through candidate elimination algorithm. (6M)

1. A Biased Hypothesis Space

- Suppose we wish to assure that the hypothesis space contains the unknown target concept.
- The obvious solution is to enrich the hypothesis space to include every possible hypothesis.

- Consider *EnjoySport* example in which we restricted the hypothesis space to include only conjunctions of attribute values.

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

- Most specific hypothesis consistent with the first two examples
- It incorrectly covers the third (negative) training example
- The problem is that we have biased the learner to consider only conjunctive hypotheses.
- In this case we require a more expressive hypothesis space.

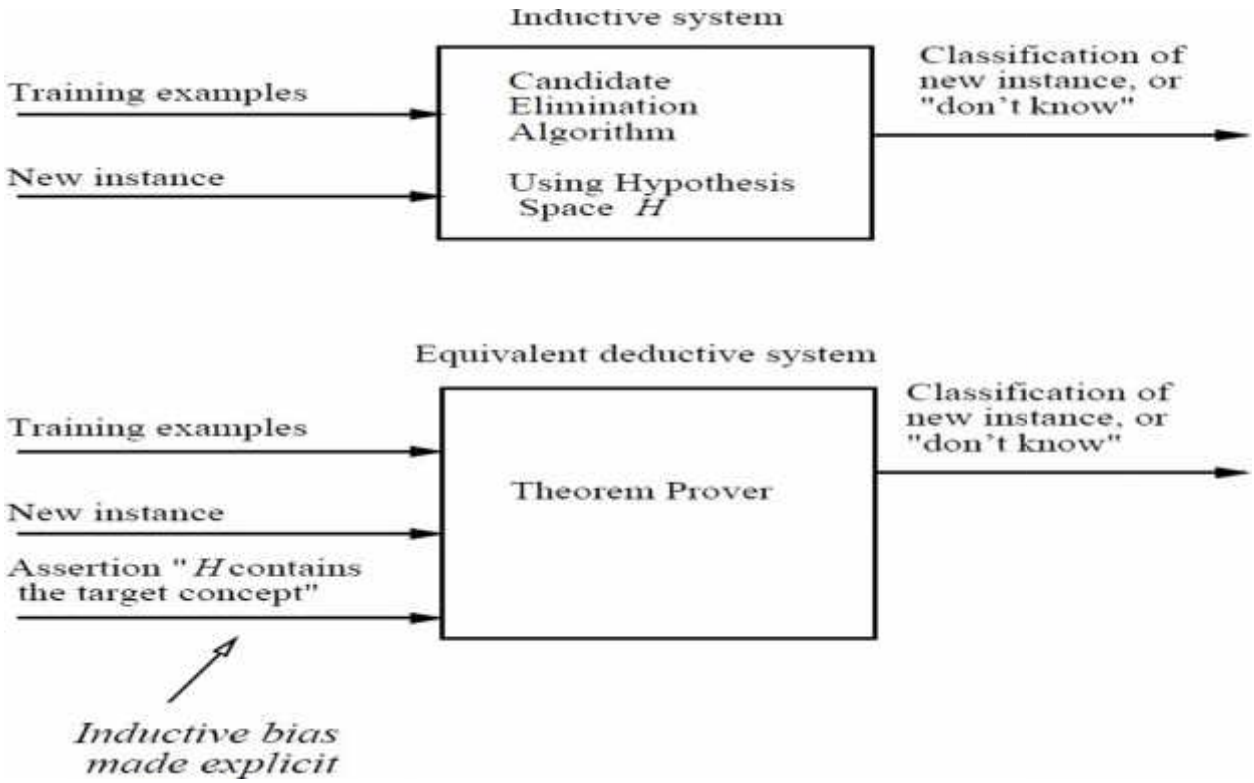
2. An unbiased learner

- The obvious solution to be a unbiased learner– design hypothesis space H to represent ***every teachable concept***;
- It should capable of representing every possible subset of the instances X. In general, the set of all subsets of a set X is called the ***power-set*** of X.
- In general, number of distinct subsets is $2^{|X|}$.
- Thus, there are 2^{96}, or approximately distinct target concepts that could be defined over this instance space and that our learner might be called upon to learn.
- Our conjunctive hypothesis space is able to represent only 973 of these-a very biased hypothesis space indeed!
- Let us reformulate the *Enjoysport* learning task
- Let H' 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.
- For instance, the target concept "Sky = Sunny or Sky = Cloudy" could then be described as

$$\langle Sunny, ?, ?, ?, ?, ? \rangle \vee \langle Cloudy, ?, ?, ?, ?, ? \rangle$$

3. The Futility of Bias-Free Learning

- CEA generalizes observed training examples because it was biased by the implicit assumption that the target concept could be represented by a conjunction of attribute values.
    - If this assumption is correct (and the training examples are error-free), its classification of new sample will also be correct.
    - If this assumption is incorrect, however, it is certain that the CEA will mis-classify at least some instances from X.



The input-output behavior of the CANDIDATE-ELIMINATION algorithm using a hypothesis space ***H is*** identical to that of a deductive theorem prover utilizing the assertion "***H*** contains the target concept." This assertion is therefore called the ***inductive bias*** of the CANDIDATE ELIMINATION algorithm characterizing inductive systems by their inductive bias allows modeling them by their equivalent deductive systems. This provides a way to compare inductive systems according to their policies for generalizing beyond the observed training data.

b.Explain List-Then-Eliminate algorithm. (4M)

LIST-THEN-ELIMINATE Algorithm:

The LIST-THEN-ELIMINATE algorithm first initializes the version space to contain all hypotheses in H and then eliminates any hypothesis found inconsistent with any training example.

1. ***VersionSpace c*** *a* list containing every hypothesis in H
2. For each training example, ***(x, c(x))***
   remove from ***VersionSpace*** any hypothesis ***h*** for which ***h(x) ≠ c(x)***
3. Output the list of hypotheses in ***VersionSpace***

List-Then-Eliminate works in principle, so long as version space is finite.
However, since it requires exhaustive enumeration of all hypotheses in practice it is not feasible.

6. Find the maximally general hypothesis and maximally specific hypothesis for the training examples given in the table using candidate elimination algorithm.

Size	Color	Shape	Class/Label
Big	Red	Circle	No
Small	Red	Triangle	No
Small	Red	Circle	Yes
Big	Blue	Circle	No
Small	Blue	Circle	Yes

Solution:

Step1: S0= {'0', '0', '0'}
G0 = {'?', '?', '?' }

Step2: Training Instance d: ('big', 'red', 'circle', 'N') -ve instance
S after removing consistent hypothesis with d
S1={('0','0', '0')}

Consider g: ('?', '?', '?')

g after min specialization:
Replace each ? with opposite pair in –ve instance. (First ? can be either small or big but we already have big in our –ve example so replace with **Small**, next ? is either red or blue and we have red already so replace with **blue** and third ? can be either triangle or circle and we have circle already so replace with **triangle**)

S1: {('0', '0', '0')}
G1: {('?', '?', 'triangle'), ('small', '?', '?'), ('?', 'blue', '?')}

Next Instance d: {small, red, triangle} –ve instance

S2={0,0,0}
G: {('?', '?', 'triangle'), ('small', '?', '?'), ('?', 'blue', '?')}

First two pairs are matching with my –ve instance (i.e. triangle and small) which should be opposite so try to make each ? in the first two pairs with specific ones. {?,blue,?} is opposite to –ve instance d so keep it as it is.

Replace each pair ? with the opposite pair in the negative instance. (In the first pair ? should be replaced by **{big,?,triangle}** bcz small is there in my –ve instance and second ? can be replaced by **{?,blue,triangle}**
Second pair second ? can be replaced by {small,blue,?} and third ? can be replaced by {small,?,circle}

My final G will be as below:
G[2]: {{big,?, triangle} {?,blue, triangle}{small,blue,?} {small,?,circle} ('?', 'blue', '?')

Compare G[2] with d and if it is not consistent then remove that pair.
{big,?,triangle} {small, ?, circle} ,– consistant bcz it is –ve and my instance also negative so consider this one.
{?,blue,triangle} and {small,blue,?} - These two are specific to {?,blue,?} so ignore it.
{?,blue,?} – This is –ve as opposite to red is blue but my instance also –ve so consistant.

My final generic hypothesis after removing less consistant ones are :
G : {{big,?,triangle}{small,?,circle}{?,blue,?}

Next Instance d: {small, red, circle} +ve instance

G after removing inconsistent hypothesis : {small,?,circle}
bcz {big,?,triangle} and {?,blue,?} are not consistant with d so ignore it.
S[3] = {small,red,circle}
G[3] = {small,?,circle}

Next Instance d: {big, blue, circle} -ve instance
S after removing inconsistent hypothesis : {small,red,circle}
G: {small,?,circle} which is negative and my d also –ve so consistant so keep as it is.
G[4] : {small,?,circle}

Next Instance d: {small, blue, circle} +ve instance

G after removing inconsistent hypothesis : {small,?,circle}

S[5] = {small,?,circle}

G[5] = {small,?,circle}

7.a)List the drawbacks of Find-S algorithm. Explain candidate elimination algorithm in detail. (5M)

Drawbacks of Find-S : (2Marks)

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

Candidate Elimination Algorithm: (3Marks)

1. Initialize G to the set of maximally general hypotheses in H
2. Initialize S to the set of maximally specific hypotheses in H
3. For each training example d, do
   a. If d is a positive example
      - Remove from G any hypothesis 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 is consistent with d, and some member of G is more general than h
        - Remove from S, hypothesis that is more general than another hypothesis in S
   b. If d is a negative example
      - Remove from S any hypothesis 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 is consistent with d, and some member of S is more specific than h
        - Remove from G any hypothesis that is less general than another in G

b.What do you mean by hypothesis space, instance space and version space? (5M)

Definition: A hypothesis h is **consistent** with **a** set of training examples ***D*** if **and** only if ***h(x)*** = ***c(x)*** for each example (x, ***c(x))*** in ***D.***

Consistent $(h, D) = (\text{for all } \langle x, c(x)\rangle \in D)\ h(x) = c(x))$

Note difference between definitions of *consistent* and *satisfies*

- an example *x* is said to ***satisfy*** hypothesis ***h*** when ***h(x)*** = 1, regardless of whether *x* is a positive or negative example of the target concept.
- an example *x* is said to ***consistent*** with hypothesis ***h*** iff ***h(x)*** = ***c(x)***

Version Space: The version space, denoted $VS_{H,D}$ with respect to hypothesis space H and training examples D, is the subset of hypotheses from H consistent with the training examples in D

$VS_{H,D} = \{h \in H \mid Consistent(h, D)\}$