Q1.(a) What is Learning? Why is learning considered as important area in AI? Explain.

Ans. **Learning:-**

Learning is defined as the process that acquires knowledge to the system.

A definition by Simon (1983) is one of the best:

"Learning denotes changes in the system that are adaptive in the sense that they enable the system to do the same task (or tasks drawn from a population of similar tasks) more effectively the next time."

One can easily extend this definition easily to the AI systems:

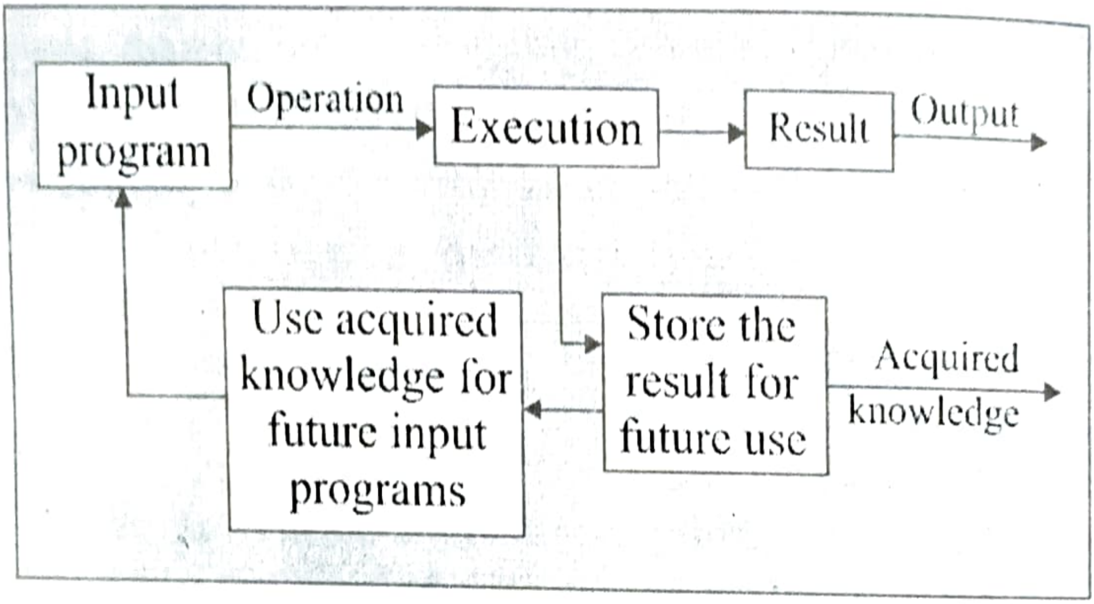
"Machine learning denotes automated changes in an AI system that are adaptive in the sense that they enable the system to do the same task (or tasks drawn from a population of similar tasks) more effectively the next time."

Learning provides power to handle new problems on the basis of similar problems tackled some other time.

Learning is acquiring new knowledge, behaviours, skills, values, preferences or understanding, and may involve synthesizing different types of information.

Learning is considered as an important area in AI. Machines can not be called intelligent until they are able to learn to do new things and to adapt to new situations, rather than simply doing as they are told to do. This requires knowledge acquisition, inference, updating of knowledge base, applying faster searches etc. Knowledge acquisition refers to the situation in which computer executes one program and remembers the process for future use. Learning is important for practical applications of Artificial Intelligence.

The general model of learning is shown below:-


Input program
Operation
Execution
Result
Output
Use acquired knowledge for future input programs
Store the result for future use
Acquired knowledge

Here the input program is any general program executed for solution of a problem. The results are reported as output, but results are also stored back to acquire knowledge for future use. It means, in case similar problem is required to be solved next time, the results can automatically be taken from previously acquired knowledge.

In brief one can say that "Learning is an area of AI that focuses on processes of self-improvement". Information processes that improve their performance or enlarge their knowledge bases are said to have learning.

Q14.(b) What is Rote Learning? What are its features, advantages and disadvantages? Also write the challenges / issues in the design of rote learning systems.

Ans. Rote Learning:-

The simple memorizing of individual items like solutions to problems, words of vocabulary etc. is known as rote learning.

The major practice involved in rote learning techniques is learning by repetition, based on the idea that one will be able to quickly recall the meaning of the material the more it is repeated.

Rote learning is the basic learning activity. It is also called memorization. Rote learning is relatively easy to implement on a computer.

Rote learning consists of memorizing the solutions of the solved problems so that the system needs not to solve them again:

$$\underset{\text{Input pattern}}{(X1, \ldots, Xn)} \xrightarrow[\text{performance function}]{f} \underset{\text{Output value of computation}}{(Y1, \ldots, Yp)} \xrightarrow{\text{Store}} \underset{\text{Associated pair}}{(X1, \ldots, Xn), (Y1, \ldots, Yp)}$$

During subsequent computations of $f(X_1, \ldots, X_n)$, the performance element can simply retrieve$(Y_1, \ldots, Y_p)$ from memory rather than recomputing it.

This concept says that knowledge can be saved so it can be used again. But once knowledge is stored, how can it be retrieved. There are varieties of ways of retrieval.

A simple example of rote learning is *caching*.

- *Caching is the process which stores computed values or in other words it stores large amount of data.*
- *This information is recalled when required by computation.*
- *Significant time savings can be achieved.*
- *Many AI programs have used caching very effectively.*

One good example of this is Samuel's checkers program. This program employed two kinds of learning: rote learning and parameter adjustment. It was designed to store the chess moves played by its creator and thus used to learn the game. It learned to play checkers to such a good extent that ultimately it was able to beat its own creator. Main features of Ṣamuel's checkers program are:

- *Samuel's program used the minimax search procedure to explore the game tree.*
- *Time constraints do not permit complete searches i.e. only few levels in the tree are permitted to be searched.*
- *It records board positions and scores at search ends.*

- *Now if the same board position arises later in the game the stored value can be recalled and the end effect is that deeper search has occurred.*

Features of Rote learning:-

Rote learning has the following features:-

- ***The knowledge, without any modification is simply copied into the knowledge base. So, saved knowledge can be used again.***
- ***This technique can save significant amount of time as computed values are stored in this learning. Therefore, no repeated computations, inference or query is necessary.***

Advantages of Rote learning:-

Rote learning has the following advantages:-

- ***It enables machine to store organized information.***
- ***Saved knowledge can be used again.***
- ***It is faster to reuse information stored than to collect it again.***
- ***One can get generalized information stored in the database using rote learning.***
- ***No repeated computations, inference or query is necessary. Thus, it saves a lot of time.***

Disadvantages of Rote learning:-

Rote learning has the following disadvantages:-

- *The information and data stored would become very large, complex and hence unmanageable.*

- *It is not the best method as it is elementary in form and hence can not be used in sophisticated applications.*

Challenges / issues in the design of rote learning systems:-

Rote learning is basically a simple process. However it does illustrate some issues that are relevant to more complex learning issues. These are:

1. Generalization:-

The number of potentially stored objects can be very large. Therefore, there is need to generalize some information to make the problem manageable.

2. Stability of the Environment:-

Stability of the environment means the information stored at one time should still be valid later. If the environment rapidly changes then rote learning is not very effective.

The exact change in the environment is to be detected and handled properly.

3. Store-versus-compute trade-off:-

The cost of storing and retrieving the memorized information should be smaller than the cost of recomputing it.

4. Memory organization:-

For the faster access of the stored data it would be better to recompute it but there must be a way to access the appropriate stored value quickly. Methods such as hashing, indexing and sorting can be employed to enable this.

Thus, rote learning requires useful organization of the memory so that the retrieval of the desired information will be very fast.

Conclusion:-

Rote learning:

- ***Method of study is based on learning facts etc by heart without considering their meaning.***
- ***Most trivial form of learning.***
- ***Simple storing of computed information (Everything is memorized) e.g. computer simply stores a piece of data in knowledge base.***
- ***Many computer programs e.g. database systems can be said to learn. The act of storage allows the program to perform better in future.***
- ***There may be situations where computation is more expensive than recalling the previously done job. Thus remembering the previous would certainly help perform better. This form of learning is called rote learning.***

Q15.(a) Explain learning by taking advice with examples. **MDU BCA Nov. 2016**

OR

What is Learning by taking advice in AI? What are the approaches to advice taking? Explain.

Ans. Learning by taking advice:-

It involves receiving direct instructions on how to respond to certain situations. In a machine, this amounts to straight-forward procedural programming. In situations where the instructions do not correspond to direct procedures (For example, "Take control of the centre of the board" in Chess), an interpreter is required to translate the instructions to concrete execution steps.

Thus, this kind of learning takes place when a computer runs a particular program by taking advice from its programmer (or creator of that program).

A computer can do very little without a program for it to run. When a programmer writes a series of instructions into a computer, a rudimentary kind of learning is taking place: The programmer is a sort of teacher, and the computer is a sort of student. After being programmed, the computer is now able to do something it previously could not. Executing the program may not be such a simple matter.

Suppose the program is written in a high-level language like LISP. Some interpreter or compiler must intervene to change the teacher's instructions into code that the machine can execute directly.

The idea of advice taking in AI based learning was proposed as early as 1958. However, very few attempts were made in creating such systems until the late 1970s. Learning in the case of problems like expert systems can be done basically by advice taking. Expert systems are thus providing a major encouragement in this area.

In some instances, it is almost as difficult to give advice to a computer program as it is to human being. The major problem with the computer taking advice is its ability of understanding or interpretation. It is usually desirable to input the advice information in the form of the human's natural language – the normal language, such as English, that the human speaks and understands. However, for a computer, human natural language understanding is extremely difficult.

Approaches to advice taking:-

There are two basic approaches to advice taking:

1. Rules that can guide performance elements of the system are constructed by taking high level, abstract advice. All aspects of advice taking are automated by:

(a) Request:-

This can be simple question asking about general advice or more complicated by identifying shortcomings in the knowledge base and asking for a remedy.

(b) Operationalise:-

Translated advice may still not be usable so this stage seeks to provide a representation that can be used by the performance element.

(c) Interpret:-

Translate the advice into an internal representation.

(d) Integrate:-

When knowledge is added to the knowledge base care must be taken so that bad side-effects are avoided. For example, Introduction of redundancy and contradictions.

(e) Evaluate:-

The system must assess the new knowledge for errors, contradictions etc.

2. Tools such as knowledge base editors and debugging should be developed. These are used to aid an expert to translate his expertise into detailed rules. Here the expert is an integral part of the learning system. Such tools are important in expert systems area of AI.

Q15.(b) Explain role of learning in problem solving with examples. **MDU BCA Nov. 2016**

OR

Explain Learning in Problem Solving with examples. **MDU BCA May 2016**

Ans. A problem can be solved by experience. Efficiency of solving a similar process increases by experience. This does not usually involve gathering new knowledge but may involve reorganization of data or remembering how to achieve to solution.

Thus, learning in problem solving involves:

- ***Learning ways of problem solving from own experience – without an instructor/advisor.***
- ***Does not involve an increase in knowledge, just the methods in using the knowledge.***
- ***Learnt rules that may prove good in directing the problem solving process, but it incurs a cost (utility) because the problem solver needs to store and consult those rules.***

Types of Learning in Problem Solving:-

There are three basic methods in which a system can learn from its own experience:-

1. Learning by Parameter Adjustment
2. Learning by Macro Operators
3. Learning by Chunking

1. Learning by Parameter Adjustment:-

In many programs the state of the search process depends on the evaluation procedure used. A very common example of this is game playing. Many have static evaluation function. The evaluation function is represented as a polynomial of the form:

$c_1t_1 + c_2t_2 + c_3t_3 + ...$

The t terms are values of features and the c terms are weights.

For example, the factors such as demand and production capacity may be combined into a single score to indicate the chance for increase of production. But it is difficult to know a priori how much weight should be attached to each factor.

The correct weight can be found by taking some estimate of the correct settings and then allowing the program to modify its settings based on its experience. This type of learning system is useful when little knowledge is available.

2. Learning by Macro-operators:-

The basic idea here is similar to Rote learning. It avoids expensive recomputation. Macro-operators can be used to group a whole series of actions into one.

In this learning:

- ***Rote learning as a sequence of operations is found to be useful during problem solving.***

- ***Sequence of actions that can be treated as a whole are called macro-operations.***

- ***Once a problem is solved, the learning component takes the computed plan and stores it as a macro-operator. The preconditions are the initial conditions of the problem just solved and its post conditions correspond to the goal just achieved.***

- ***The problem solver efficiently uses the knowledge base it gained from its previous experiences.***

- ***By generalizing macro-operators the problem solver can even solve different problems.***

3. Learning by Chunking:-

Chunking is a process where a sequence of actions are put together and treated as a whole to perform a particular task.

Chunking is similar to learning with macro-operators. Generally, it is used by problem solver systems that make use of production systems.

A production system consists of a set of rules that are in If – THEN form. SOAR is a system that uses production rules to represent its knowledge. It also employs chunking to learn from experience. SOAR solves problems. It fires productions. These productions are stored in long term memory. When SOAR detects a useful sequence of firing, it creates chunks. Chunks may be generalized before storing.

Q16.(a) Explain learning from example – Induction.

Ans. Induction Learning:-

Inductive learning means, generalization of knowledge gathered from real world examples and use of the same for solving other similar problems. In simple words, learning by induction or inductive learning is a method which generalizes the rules of past.

Inductive learning involves the process of learning by example, where a system tries to induce a general rule from a set of observed instances i.e. Induction involves drawing general conclusions from specific observations. Learning from example and learning by' discovery are instances of inductive learning. Most of the knowledge that the human has acquired about its environment is through the process of induction.

For example, observing the specific facts that Sun rises in the east on day 1, day 2, ... we infer a general statement, namely, Sun always rises in the east.

Inductive learning methods can be defined as those methods that systematically produce general descriptions or knowledge from the specific knowledge provided by domain examples.

The learning process which belongs to supervised learning, does classification and constructs class definitions is called induction or concept learning.

The techniques used for constructing class definitions or concept learning are:-

1. Winston's Learning Program

2. Version Spaces
3. Decision Trees

1. Winston's Learning Program:-

It is a structural concept of learning program. This program was operated in a simple block world domain. Its goal was to construct representations of the definitions of concepts in the block domain.

Example: Concepts such as "house".

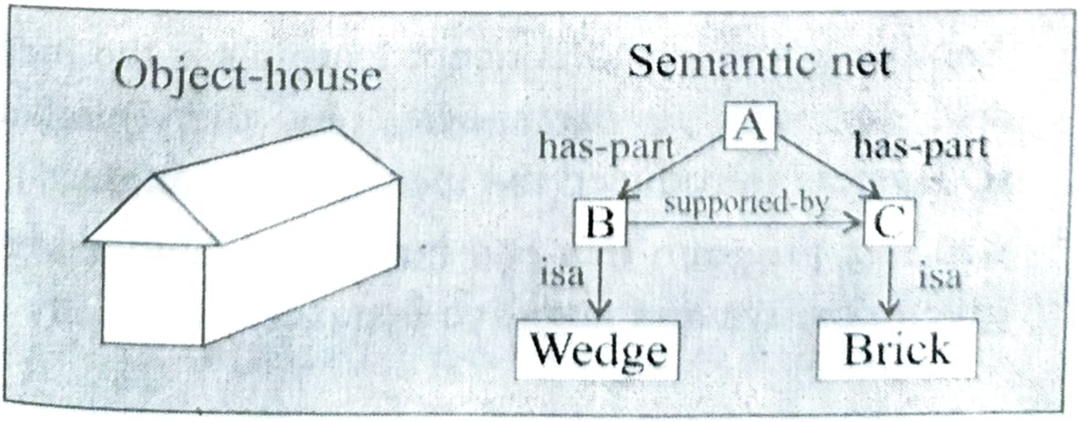
Start with input, a line drawing of a blocks world structure. It learned concepts House, Tent, Arch as:

Brick (rectangular block) with a wedge (triangular block) suitably placed on top of it, tent – as 2 wedges touching side by side, or an arch – as 2 non-touching bricks supporting a third wedge or brick.

The program for each concept is learned through near miss. A near miss is an object that is not an instance of the concept but a very similar to such instances.

The program uses procedures to analyze the drawing and construct a semantic net representation.

An example of such a structure for the house is shown below:



Node A represents entire structure, which is composed of two parts: Node B, a Wedge and node C, a Brick.

Links in network include: supported-by, has-part and isa.

There are three basic steps to the problem of concept formulation:

1. ***Select one known instance of the concept. Call this the concept definition.***
2. ***Examine definition of other known instance of the concept. Generalize the definition to include them.***
3. ***Examine description of near misses. Restrict the definition to exclude these.***

Both steps 2 and 3 rely on comparison and both similarities and differences need to be identified.

2. Version Spaces:-

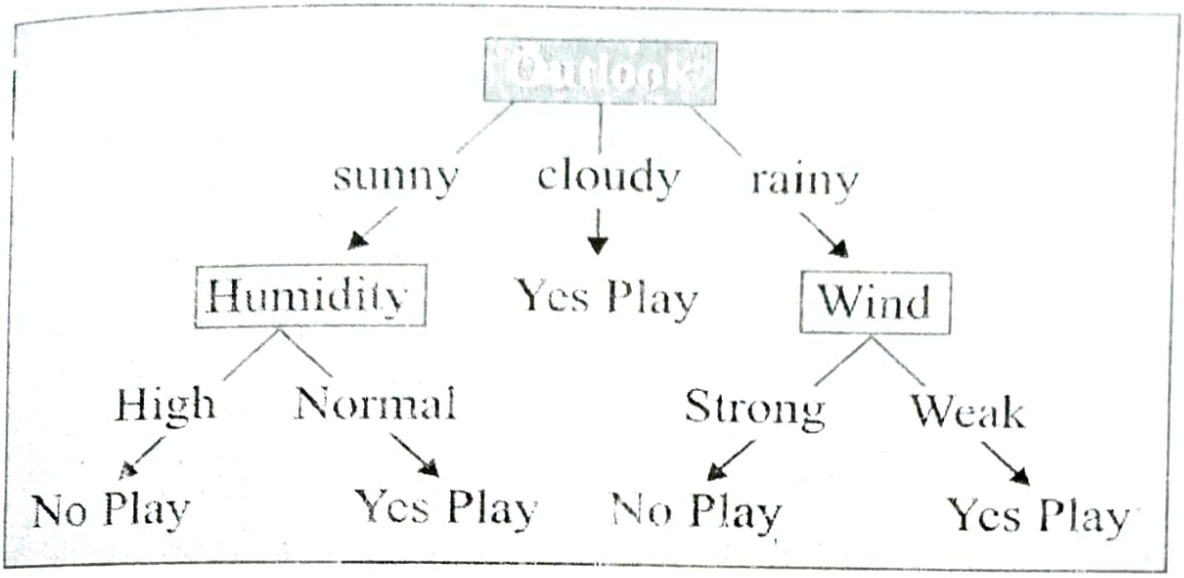
A version space is a representation that is used to get relevant information from a set of learning examples. It is a hierarchical representation of knowledge that keeps track of all the useful information supplied by a sequence of learning examples without remembering any of the examples.

3. Decision Trees:-

Another approach to concept learning is the induction of decision tree as exemplified by the Quinlan in his ID3system introduced the idea of decision trees.

ID3 is a program that can build trees automatically from given positive and negative instances. Basically each leaf of a decision tree asserts a positive or negative concept. To classify a particular input one starts at the top and follows assertions down until answer is reached.

The classification tree built using ID3 algorithm is shown below. It tells if the weather was amenable to play?



Decision trees are useful when the concept space is very large, decision tree learning algorithm runs more quickly than version spaces. On the other side, large, complex decision trees can be difficult for human to understand and so a decision tree system may find it hard to explain its classification.

Q16 (b) Explain explanation based learning with examples. **MDU BCA May 2016**

OR

What is Explanation Based Learning (EBL)? Explain.

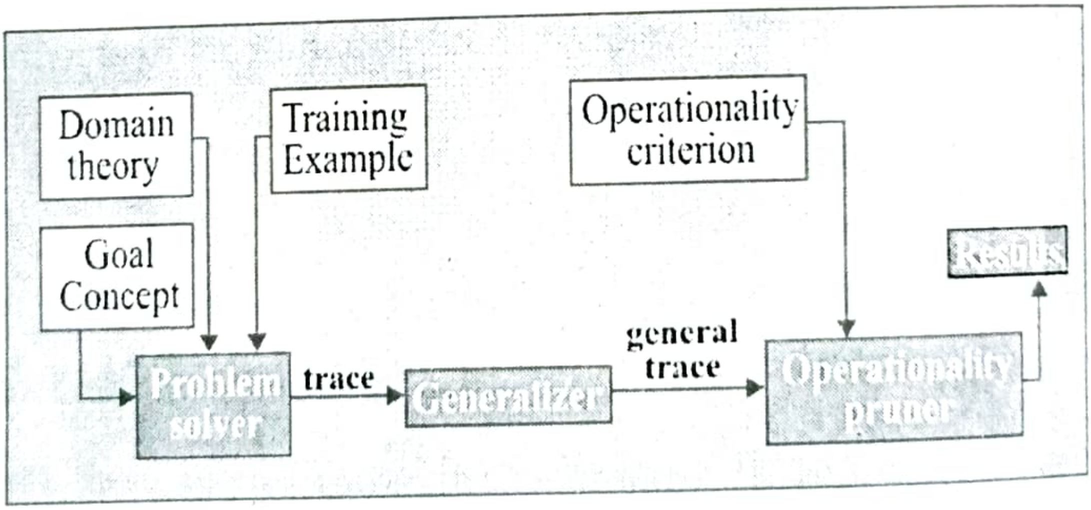
Ans. An Explanation Based Learning (EBL) system accepts an example (i.e. a training example) and explains what it learns from the example. The EBL system takes only the relevant aspects of the training. This explanation is translated into particular form that a problem solving program can understand. The explanation is generalized so that it can be used to solve other problems.

Explanation-Based Learning (EBL) techniques are knowledge-based and require a great deal of knowledge about the domain of the concept to be learned. Instead of providing several instances of a concept, only a single example is required for a new concept to be learned.

The idea is that a strong domain theory not only helps in describing or explaining the new concept, but also guides the generalisation process, such that a great deal can be learned from a single example.

EBL System Schematic:-

The schematic below shows explanation based learning.



An EBL accepts 4 kinds of input:

1. A training example:-

This indicates what the learning sees in the world.

2. A goal concept:-

It is a high level description of what the program is supposed to learn.

3. An operational criterion:-

It is a description of which concepts are usable.

4. A domain theory:-

It is a set of rules that describes relationships between objects and actions in a domain.

EBL computes a generalization of the training example that is sufficient not only to describe the goal concept but also satisfies the operational criterion. This has two steps:

- **Explanation:** The domain theory is used to prune away all unimportant aspects of the training example with respect to the goal concept.
- **Generalisation:** The explanation is generalized as far as possible while still describing the goal concept.