1)Define Machine Learning?

Machine learning is the process of parsing data through algorithms, learning from these data and then being able to make predictions about the world.

Machine learning is a field of computer science that gives computers the ability to learn without being explicitly programmed.ML draws on results from artificial intelligence, probability and statistics, computational complexity theory, control theory, information theory, philosophy, psychology, neurobiology, and other fields.

2) How do you know if a program is a machine learning program or not?

A machine learning program can be identified as a program with learns, upgrades and becomes better after every(or several) usage. So basically, if a large data set(info) is given to the program and every time(or several times) it is executed to solve something- it learns from its experience for specific set of tasks and increases its performance measure each time gradually. In simpler words, if a program gets better and better and upgrades every time with every usage then that program is said to be a machine learning program.

3)Which of these can be called a machine learning program? Justify your answer.

a)Matrix Multiplication Code in Python- It is not a machine learning program/algorithm as the machine is programmed to do a matrix multiplication for that a basic algorithm/code is written and hence machine doesn't learn anything from the program.

b)YouTube homepage generation algorithm for every user-It is a machine learning program/algorithm as the machine here generates a different homepage for every user-- it learns from activities and experience of each user and according to it generates a specific content and also updates the content brings in the new content/new uploads every time, recommends the user on the homepage which videos to watch. Hence here the machine follows a recommendation system which updates every time and learns from its experience.

c)Flipkart e-commerce site-In some terms it can be said as a ML problem/algorithm as it has a recommendation system it generates recommendation based on the products you visited, the review you give to certain products. So basically it learns from each user and based on that generate what genre of products and further more what type of product you will be interested in and hence it is an ML program.

d)Quick Sort Algorithm-It is not a machine learning program as it is a programmed algorithm based on which an array is sorted in python. So it has nothing to learn from hence it is not an ML program.

4)Design a suitable P,T,E parameters for a Checkers program, to formulate it as a well posed Machine Learning Program.

Well Posed Machine Learning Program:

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.

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.

5)Define characteristics of a well posed Learning Program for a Speech Recognition Program. Formulate this design, as a well posed Learning Program.

A well posed learning program can be characterized by 3 basic features:

a)The class of tasks(T).

b)The measure of performance to be improved(P).

c)The source of experience(E).

In case of speech recognition program:

a) Task T: Recognizing and classifying spoken words via audio.

b)Performance measure P:Percent of words correctly classified.

c) Training experience E: Receiving feedback form the user for all the incorrect classification and recognition.

6)Differentiate between target concept and approximating function. Discuss the final design of Checker Learning Program and list the choices made in course of the design.

Target function is a measure to determine exactly what type of knowledge will be learned and how this will be used by the performance program. 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. Given this setting where 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. So, in this case the the a function let us call it "ChooseMove" which is used to choose the best move from among the legal moves for a Checker's game is considered to be the target function. Thus, we have reduced the learning task in this case to the problem of discovering an operational description of the ideal target function V(ChooseMove).

But, it may be very difficult in general to learn such an operational form of V perfectly. In fact, we often expect learning algorithms to acquire only some approximation to the target function, and for this reason the process of learning the target function is often called function approximation. So we consider the Checker's Learning Program again. 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. In other words, each training example is an ordered pair of the form (b, V',,,i,(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 VZrain(b) is therefore +100.

The final design to Checker's Program can be summarized as:

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. These four modules, summarized are as follows: 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. strategy used by the Performance System to select its next move at each step is determined by the learned p evaluation function. Therefore, we expect its performance to improve as this evaluation function becomes increasingly accurate. 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. Each training example in this case corresponds to some game state in the trace, along with an estimate Vtrai, of the target function value for this example. In our example, the Critic corresponds to the training rule given by Equation (1.1). 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. In our example, the

Generalizer corresponds to the LMS algorithm, and the output hypothesis is the function f described by the learned weights wo, . . . , W6.

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. In our example, the

Experiment Generator follows a very simple strategy: It always proposes the same initial game board to begin a new game. More sophisticated strategies

could involve creating board positions designed to explore particular regions of the state space. Together, the design choices we made for our checkers program produce specific instantiations for the performance system, critic; generalizer, and experiment generator. Many machine learning systems can-be usefully characterized in terms of these four generic modules.

7)What are the key issues in Machine Learning?

The field of machine learning is concerned with many issues such as the following:

a)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?

b)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?

c)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?

d)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?

e)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?

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