## AI Question Bank

**UNIT-I**

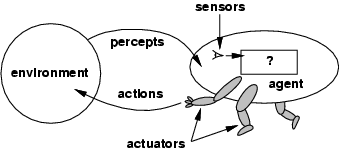
1. **What is Artificial Intelligence?**

Artificial Intelligence is the study of how to make computers do things which at the

moment people do better.

1. **Define an agent.**

An **agent** is anything that can be viewed as **perceiving** its **environment** through **sensors** and **acting upon** that environment through **actuators.**



1. **What are the different types of agents?**

**A human agent** has eyes, ears, and other organs for sensors and hands, legs, mouth, and

other body parts for actuators.

**A robotic agent** might have cameras and infrared range finders for sensors and

various motors for actuators.

**A software agent** receives keystrokes, file contents, and network packets as sensory inputs

and acts on the environment by displaying on the screen, writing files, and sending network

packets.

**Generic agent** – A general structure of an agent who interacts with the environment

1. What is an agent function?

An agent’s behavior is described by the **agent function** that maps any given **percept sequence** to an **action.**

1. Differentiate an agent function and an agent program.

|  |  |
| --- | --- |
| Agent Function | Agent Program |
| An abstract mathematical description | A concrete implementation,running on the agent Architecture. |

1. **Define rational agent?**

For each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has. A rational agent should be autonomous

1. **List down the characteristics of intelligent agent.**

**Internal characteristics are**

* Learning/reasoning: an agent has the ability to learn from previous experience and to successively adapt its own behavior to the environment.
* Reactivity: an agent must be capable of reacting appropriately to influences or information from its environment.
* Autonomy: an agent must have both control over its actions and internal states. The degree of the agent’s autonomy can be specified. There may need intervention from the user only for important decisions.
* Goal-oriented: an agent has well-defined goals and gradually influences its environment and so achieves its own goals.

**External characteristics are**

* Communication: an agent often requires an interaction with its environment to fulfill its tasks, such as human, other agents, and arbitrary information sources.
* Cooperation: cooperation of several agents permits faster and better solutions for complex tasks that exceed the capabilities of a single agent.

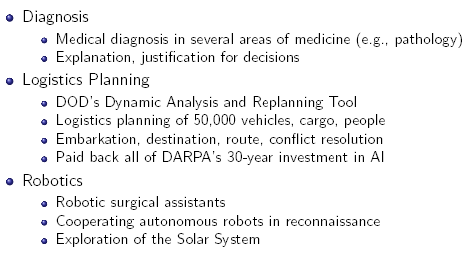
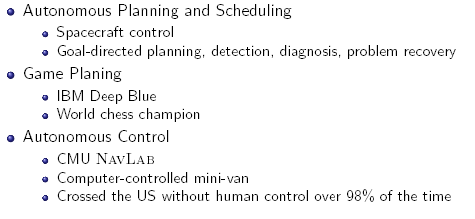
– mobility: an agent may navigate within electronic communication networks.

Character: like human, an agent may demonstrate an external behavior with many human characters as possible.

1. **What is PEAS?**

PEAS (Performance, Environment, Actuators, Sensors)

1. **What can Ai do today?**



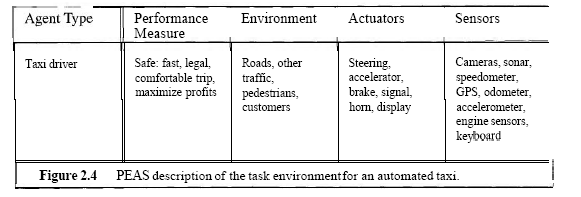
1. What is a task environment? How it is specified?



**Task environments** are essentially the "problems" to which rational agents are the "solutions."A Task environment is specified using PEAS (Performance, Environment, Actuators, Sensors) description.

1. Give an example of PEAS description for an automated taxi.





1. List the properties of task environments.
   1. Fully observable vs. partially observable**.**
   2. Deterministic vs. stochastic.
   3. Episodic vs. sequential
   4. Static vs, dynamic.
   5. Discrete vs. continuous.
   6. Single agent vs. multiagent**.**
2. **What are the four different kinds of agent programs?**
3. Simple reflex agents;
4. Model-based reflex agents;
5. Goal-based agents; and
6. Utility-based agents.
7. What are learning agents?

A learning agent can be divided into four conceptual components, as shown in Fig-

The most important distinction is between the learning element, which is ELEMENT responsible for making improvements, and the performance element, which is responsible for selecting external actions. The performance element is what we have previously considered to be the entire agent: it takes in percepts and decides on actions. The learning element uses CRITIC feedback from the critic on how the agent is doing and determines how the performance element should be modified to do better in the future.

1. Define the problem solving agent.

A Problem solving agent is a **goal-based** agent . It decide what to do by finding sequence of actions that lead to desirable states. The agent can adopt a goal and aim at satisfying it.

Goal formulation is the first step in problem solving.

1. Define the terms goal formulation and problem formulation.

**Goal formulation**,based on the current situation and the agent’s performance measure, is

the first step in problem solving.

The agent’s task is to find out which sequence of actions will get to a goal state.

**Problem formulation** is the process of deciding what actions and states to consider given a goal

1. List the steps involved in simple problem solving agent.
2. Goal formulation
3. Problem formulation
4. Search
5. Search Algorithm
6. Execution phase
7. **Explain the basic terminology used in Tree and Graph**

A [tree](https://www.javatpoint.com/tree) is a [non-linear](https://www.javatpoint.com/linear-vs-non-linear-data-structure) data structure that represents the hierarchy. A tree is a collection of nodes that are linked together to form a hierarchy.

some terminologies used in a ***tree*** data structure.

* **Root node:** The topmost node in a tree data structure is known as a root node. A root node is a node that does not have any parent.
* **Parent of a node:** The immediate predecessor of a node is known as a parent of a node. Here predecessor means the previous node of that particular node.
* **Child of a node:** The immediate successor of a node is known as a ***child of a node***.
* **Leaf node:** The leaf node is a node that does not have any child node. It is also known as an external node.
* **Non-leaf node:** The non-leaf node is a node that has atleast one child node. It is also known as an ***internal node***.
* **Path:** It is a sequence of the consecutive edges from a source node to the destination node. Here edge is a link between two nodes.
* **Ancestor:** The predecessor nodes that occur in the path from the root to that node is known as an ancestor.
* **Descendant:** The successor nodes that exist in the path from that node to the leaf node.
* **Sibling:** All the children that have the same parent node are known as siblings.
* **Degree:** The number of children of a particular node is known as a degree.
* **Depth of node:** The length of the path from the root to that node is known as a depth of a node.
* **Height of a node:** The number of edges that occur in the longest path from that node to the leaf node is known as the height of a node.
* **Level of node:** The number of edges that exist from the root node to the given node is known as a level of a node.

### What is a Graph

A [graph](https://www.javatpoint.com/ds-graph) is like a tree data structure is a collection of objects or entities known as nodes that are connected to each other through a set of edges. A tree follows some rule that determines the relationship between the nodes, whereas graph does not follow any rule that defines the relationship among the nodes. A graph contains a set of edges and nodes, and edges can connect the nodes in any possible way.

Mathematically, it can be defined as an ordered pair of a set of vertices, and a set of nodes where vertices are represented by 'V' and edges are represented by 'E'.

**G= (V , E)**

1. **Compare Tree and Graph**

|  |  |  |
| --- | --- | --- |
| **Basis for comparison** | **Tree** | **Graph** |
| **Definition** | Tree is a non-linear data structure in which elements are arranged in multiple levels. | A Graph is also a non-linear data structure. |
| **Structure** | It is a collection of edges and nodes. For example, node is represented by N and edge is represented as E, so it can be written as: T = {N,E} | It is a collection of vertices and edges. For example, vertices are represented by V, and edge is represented as 'E', so it can be written as: T = {V, E} |
| **Root node** | In tree data structure, there is a unique node known as a parent node. It represents the topmost node in the tree data structure. | In graph data structure, there is no unique node. |
| **Loop formation** | It does not create any loop or cycle. | In graph, loop or cycle can be formed. |
| **Model type** | It is a hierarchical model because nodes are arranged in multiple level, and that creates a hierarchy. For example, any organization will have a hierarchical model. | It is a network model. For example, facebook is a social network that uses the graph data structure. |
| **Edges** | If there are n nodes then there would be n-1 number of edges. | The number of edges depends on the graph. |
| **Type of edge** | Tree data structure will always have directed edges. | In graph data structure, all the edges can either be directed edges, undirected edges, or both. |
| **Applications** | It is used for inserting, deleting or searching any element in tree. | It is mainly used for finding the shortest path in the network. |

1. **Define search and search algorithm.**

The process of looking for sequences actions from the current state to reach the goal state is called **search.**

The **search algorithm** takes a **problem** as **input** and returns a **solution** in the form of **action sequence.** Once a solution is found, the **execution phase** consists of carrying out the recommended action..

1. **Explain state space representation**

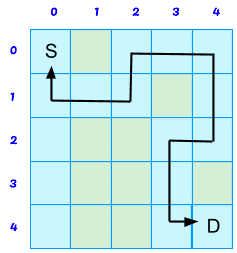
In Artificial Intelligence a state space consists of the following elements,

* A (possibly infinite) set of states
* Out of the possible states, one state represents the start state that is the initial state of the problem.
* Each state represents some configuration reachable from the start state
* Out of the possible states, some states may be goal states (solutions)
* A set of rules,
* Applying a rule to the current state, transforms it to another or a new state in the state space
* All operators may not be applicable to all states in the state space

State spaces are used extensively in Artificial Intelligence (AI) to represent and solve problems.

### State Space Search Examples:

**Example 1. Maze**

[](https://www.vtupulse.com/wp-content/uploads/2022/01/maze.png)

A maze problem can be represented as a state-space

* Each state represents “where you are” that is the current position in the maze
* The start state or initial state represents your starting position
* The goal state represents the exit from the maze

Rules (for a rectangular maze) are: move north, move south, move east, and move west

* Each rule takes you to a new state (maze location)
* Rules may not always apply, because of walls in the maze

1. **Define search tree**

A tree representation of search problem is called Search tree. The root of the search tree is the root node which is corresponding to the initial state.

1. **Explain briefly History of AI**

## History of AI and ML

|  |  |
| --- | --- |
| 1950 | Alan Turing publishes "Computing Machinery and Intelligence" |
| 1952 | Arthur Samuel develops a self-learning program to play checkers |
| 1956 | **Artificial Intelligence** used by John McCarthy in a conference |
| 1957 | First programming language for numeric and scientific computing (FORTRAN) |
| 1958 | First AI programming language (LISP) |
| 1959 | Arthur Samuel used the term **Machine Learning** |
| 1959 | John McCarthy and Marvin Minsky founded the MIT Artificial Intelligence Project |
| 1961 | First industrial Robot (Unimate) on the assembly line at General Motors |
| 1965 | ELIZA by Joseph Weizenbaum was the first program that could communicate on any topic |
| 1972 | First logic programming language (PROLOG) |
| 1991 | U.S. forces uses DART (automated logistics planning and scheduling) in the Gulf war |
| 1997 | Deep Blue (IBM) beats the world champion in chess |
| 2002 | The first robot cleaner (Roomba) |
| 2005 | Self-driving car (STANLEY) wins DARPA |
| 2008 | Breakthrough in speech recognition (Google) |
| 2011 | A neural network wins over humans in traffic sign recognition (99.46% vs 99.22%) |
| 2011 | Apple Siri |
| 2011 | Watson (IBM) wins Jeopardy! |
| 2014 | Amazon Alexa |
| 2014 | Microsoft Cortana |
| 2014 | Self-driving car (Google) passes a state driving test |
| 2015 | Google AlphaGo defeated various human champions in the board game Go |
| 2016 | The human robot Sofia by Hanson Robotics |

## Why AI Now

One of the greatest innovators in the field of machine learning was **John McCarthy**, widely recognized as the "Father of Artificial Intelligence".

In the mid 1950s, McCarthy coined the term "Artificial Intelligence" and defined it as "**the science of making intelligent machines**".

The algorithms has been here since then. Why is AI more interesting now?

The answer is:

* Computing power has not been strong enough
* Computer storage has not been large enough
* Big data has not been available
* Fast Internet has not been available

Another strong force is the major investments from big companies (Google, Microsoft, Facebook, YouTube) because their datasets became much too big to handle traditionally.

**UNIT-II**

* 1. What are the components of well-defined problems?
* The **initial state** that the agent starts in . The initial state for our agent of example problem is described by *In(Arad)*
* A **Successor Function** returns the possible **actions** available to the agent. Given a state x,SUCCESSOR-FN(x) returns a set of {action,successor} ordered pairs where each action is one of the legal actions in state x,and each successor is a state that can be reached from x by applying the action.
  + - For example,from the state In(Arad),the successor function for the Romania problem would return

{ [Go(Sibiu),In(Sibiu)],[Go(Timisoara),In(Timisoara)],[Go(Zerind),In(Zerind)] }

* The **goal test** determines whether the given state is a goal state.
* A **path cost** function assigns numeric cost to each action. For the Romania problem the cost of path might be its length in kilometers.
  1. Differentiate toy problems and real world problems.

|  |  |
| --- | --- |
| **TOY PROBLEMS** | **REAL WORLD PROBLEMS** |
| A **toy problem** is intended to illustrate various problem solving methods. It can be easily used by different researchers to compare the performance of algorithms. | A **real world problem** is one whose solutions people actually care about. |

**Give examples of real world problems.**

1. Touring problems
2. Travelling Salesperson Problem(TSP)
3. VLSI layout
4. Robot navigation
5. Automatic assembly sequencing
6. Internet searching
   1. List the criteria to measure the performance of different search strategies.

* **Completeness** : Is the algorithm guaranteed to find a solution when there is one?
* **Optimality** : Does the strategy find the optimal solution?
* **Time complexity** : How long does it take to find a solution?
* **Space complexity** : How much memory is needed to perform the search?
  1. Differentiate Uninformed Search(Blind search) and Informed Search(Heuristic Search) strategies.

|  |  |
| --- | --- |
| **Uninformed or Blind Search** | **Informed or Heuristic Search** |
| * No additional information beyond that provided in the problem definition * Not effective * No information about number of steps or path cost | * More effective * Uses problem-specific knowledge beyond the definition of the problem itself. |

* 1. **Define the steps in Random search**

Step 1: Current node x=initial node;

Step 2: If x=target node, stop with success;

Step 3: Expand x, and get a set S of child nodes;

Step 4: Select a node x’ from S at random; Step 5: x=x’, and return to Step 2.

* 1. **Explain the terminology used in search Algorithm**

**Unintelligent or blind algorithm:** An algorithm which does not use a heuristic to aid in its search--that is, a search which cannot detect how "close" it is to a solution.

**Heuristic:** An estimate of how close a given state is to the goal state. Examples include straight line distance on maps, or assigning point values to pieces in chess. The better a heuristic estimates, the better the search that uses it performs. All heuristics must evaluate to 0 when a goal state is reached.

**Initial State:** the state that the agent starts in.

**Path:** A sequence of states connected by the sequence of actions.

**Predecessor:** Any node that is higher up on the tree than the one that is being considered.

**Successor:** Any node that is a child of a node or a child or a child of a node, etc.

**Closed List:** A structure that holds every expanded node.

**Open List:** A structure that holds a list of unexpanded nodes.

**Pruning:**Ignoring portions of a tree that make no difference to the final choice.

**Minimax Value:** The utility value of being in a certain state assuming that both players play optimally.

**Goal Test:** A test that determines whether a given state is a goal state.

**Path Cost:** A function that assigns a numeric value to each path.

**Solution:** A path from the initial to the goal state.

**Optimal Solution:** One that has the lowest path cost among the solutions.

**Goal state:** The desired resulting condition in a given problem, and what search algorithms are looking for. Some problems may have a unique goal state (arrive at a given location, for example) whereas others can have multiple goal states, all satisfying a set of conditions (checkmate in chess, for instance.)

**Terminal state:** A condition in which the problem "ends", either due to the rules of the problem or because of a lack of options from that point. These states are the source of minimax evaluation; in a depth first search, if terminal states are not goal states, they are deleted.

**Parent:** The node from which the node in question was expanded. Generally, nodes only normally have one parent (though identical nodes can often be produced by moves from two different states.)

**Agent:** Something that perceives and acts in an environment.

**Agent Function:** A function that specifies the action taken by an agent in response to an environment.

**Utitlity Function:** A function that gives a numeric value for terminal states.

**Transposition Table:** A hash table of previously seen positions.

**Locally Finite:**A graph in which none of the nodes on the graph have an infinite branching factor.

**Admissible Heuristic:**A heuristic that does not overestimate.

**Optimistic Heuristic:** Considers the cost of solving a problem to be less than it really is.

**Monotonicity:** A property of logical systems that states that the set of entailed sentences can only increase as information is added to the knowledge base.

**Pathmax:**An equation that compares estimated path cost of a node with the estimated path cost of its parent node.

**Branching Factor:** Maximum number of successors of any node.

* 1. **What are the advantages of Breadth First Search?**
* BFS will not get trapped exploring a blind alley. This contrast to the DFS which may follow a single unfruitful path for a very long time, perhaps forever before the path actually terminates in a state that has no successors.
* If there is a solution, then BFS is guaranteed to find it. Furthermore, if there are multiple solutions then a minimal solution will be found.
  1. **What are the advantages of Depth First Search**
* DFS requires less memory since only the nodes on the current path are stored. In contrast to BFS where all the tree that has been generated must be stored.
* By chance, DFS may find a solution without examining much of the state space at all, where in BFS the entire tree must be examined to level n before any nodes on level n+1 can be examined.
  1. **What is Heuristic Search?**

A heuristic search is a technique that improves the efficiency of a

search process, possibly by sacrificing claims of completeness.

* 1. **What is heuristic function**

**Heuristics function:** Heuristic is a function which is used in Informed Search, and it finds the most promising path. It takes the current state of the agent as its input and produces the estimation of how close agent is from the goal.

**Admissibility of the heuristic function is given as:**

**h(n) <= h\*(n)**

**Here h(n) is heuristic cost, and h\*(n) is the estimated cost. Hence heuristic cost should be less than or equal to the estimated cost.**

* 1. **What are the steps in Best First Algorithm**

### Best first search algorithm:

* **Step 1:** Place the starting node into the OPEN list.
* **Step 2:** If the OPEN list is empty, Stop and return failure.
* **Step 3:** Remove the node n, from the OPEN list which has the lowest value of h(n), and places it in the CLOSED list.
* **Step 4:** Expand the node n, and generate the successors of node n.
* **Step 5:** Check each successor of node n, and find whether any node is a goal node or not. If any successor node is goal node, then return success and terminate the search, else proceed to Step 6.
* **Step 6:** For each successor node, algorithm checks for evaluation function f(n), and then check if the node has been in either OPEN or CLOSED list. If the node has not been in both list, then add it to the OPEN list.
* **Step 7:** Return to Step 2.
  1. **List the properties of Best first search**
* **Time Complexity:** The worst case time complexity of Greedy best first search is O(bm).
* **Space Complexity:** The worst case space complexity of Greedy best first search is O(bm). Where, m is the maximum depth of the search space.
* **Complete:** Greedy best-first search is also incomplete, even if the given state space is finite.
* **Optimal:** Greedy best first search algorithm is not optimal.
  1. **List the properties of Uniform Cost Search**
* **Completeness:**

Uniform-cost search is complete, such as if there is a solution, UCS will find it.

* **Time Complexity:**

Let C\* **is Cost of the optimal solution**, and **ε** is each step to get closer to the goal node. Then the number of steps is = C\*/ε+1. Here we have taken +1, as we start from state 0 and end to C\*/ε.

Hence, the worst-case time complexity of Uniform-cost search is**O(b1 + [C\*/ε])/**.

* **Space Complexity:**

The same logic is for space complexity so, the worst-case space complexity of Uniform-cost search is **O(b1 + [C\*/ε])**.

* **Optimal:**

Uniform-cost search is always optimal as it only selects a path with the lowest path cost.

* 1. **Define A\* search Algrithm**

**A\* Search** is the most widely used form of best-first search. The evaluation function f(n) is obtained by combining

* + - 1. **g(n) =** the cost to reach the node,and
      2. **h(n) =** the cost to get from the node to the **goal** :

##### f(n) = g(n) + h(n).

A\* Search is both optimal and complete. A\* is optimal if h(n) is an admissible heuristic. The obvious example of admissible heuristic is the straight-line distance hSLD. It cannot be an overestimate.

A\* Search is optimal if h(n) is an admissible heuristic – that is, provided that h(n) never overestimates the cost to reach the goal.

* 1. **What is game playing?**

The term Game means a sort of conflict in which *n* individuals or groups (known as players) participate. Game theory denotes games of strategy.

Game theory allows decision-makers (players) to cope with other decision-makers (players) who have different purposes in mind. In

other words, players determine their own strategies in terms of the strategies and goals of their opponent.

* 1. **What is Mini –Max Strategy?**
* Generate the whole game tree , calculate the value of each terminal state
* based on the utility function - calculate the utilities of the higher- level nodes
* starting from the leaf nodes up to the root - MAX selects the value with the highest node
* MAX assumes that MIN in its move will select the node that minimizes the value from MAX’s perspective
* MAX tries to move to a state with the maximum value, MIN to one with the minimum assumes that both players play optimally selects the best successor from a given state , invokes MINIMAX-VALUE for each successor state
  1. **Define pruning?**

**Alpha–beta pruning** is a search algorithm that seeks to decrease the number of nodes that are evaluated by the minimax algorithm in its search tree. It is an adversarial search algorithm used commonly for machine playing of two-player games (Tic-tac-toe, Chess, Go, etc.). It stops completely evaluating a move when at least one possibility has been found that proves the move to be worse than a previously examined move.

**UNIT-III**

* 1. **How Knowledge is represented?**

A variety of ways of knowledge (facts) have been exploited in AI programs. Facts: truths in some relevant world. These are things we want to represent.

* 1. **What is propositional logic?**

It is a way of representing knowledge. In logic and mathematics, a propositional calculus or logic is a formal system in which formulae representing *propositions* can be formed by combining atomic propositions using *logical connectives*. Sentences considered in propositional logic are not arbitrary sentences but are the ones that are either true or false, but not both. This kind of sentences is called propositions.

**Example** Some facts in propositional logic:

It is raining. - RAINING It is sunny - SUNNY

It is windy - WINDY

If it is raining , then it is not sunny - RAINING -> SUNNY

* 1. **What are the elements of propositional logic?**

Simple sentences which are true or false are basic propositions. Larger and more complex sentences are constructed from basic propositions by combining them with **connectives**. Thus **propositions** and **connectives** are the basic elements of propositional logic. Though there are many connectives, we are going to use the following **five basic connectives** here: NOT, AND,

OR, IF\_THEN(or IMPLY), IF\_AND\_ONLY\_IF.

They are also denoted by the symbols:

,  ,  , , , respectively.

* 1. **Define First order Logic?**

First-order logic (like natural language) assumes the world contains Objects: people, houses,

numbers, colors, baseball games, wars, … Relations: red, round, prime, brother of, bigger than,

part of, comes between, …Functions: father of, best friend, one more than, plus, …

|  |
| --- |
| **Specify the syntax of First-order logic in BNF form** |

* 1. **What are quantifiers?**

There is need to express properties of entire collections of objects,instead of enumerating the objects by name. Quantifiers let us do this.

FOL contains two standard quantifiers called

1. Universal (**)** and
2. Existential (**)**
   1. **Explain the connection between**  **and** 

“Everyone likes icecream“ is equivalent”, “there is no one who does not like ice cream”

This can be expressed as : x Likes(x,IceCream) is equivalent to

 Likes(x,IceCream)

* 1. **Difference between predicate and propositional logic.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PROPOSITIONAL LOGIC** | | | **PREDICATE ORDER LOGIC** | **/ FIRST** |
| Symbols are True / False | logical | constants | Symbols are  predicates and symbols | constants, function |
| Sentences are formed from 5 logical connectives ( and , or, implies, equivalent, not ) | | | Sentences are formed from predicate symbol followed by parenthesized list of terms and  logical connectives | |

## What is Conditional Probability Distribution?

Conditional [probability distribution](https://deepai.org/machine-learning-glossary-and-terms/probability-distribution) is the likelihood of one condition being true if another condition is known to be true. This forms the foundation of [Bayes’ theorem](https://deepai.org/machine-learning-glossary-and-terms/bayes-theorem) and [Bayesian networks](https://deepai.org/machine-learning-glossary-and-terms/bayesian-network).

In [machine learning](https://deepai.org/machine-learning-glossary-and-terms/machine-learning) notation, the conditional [probability](https://deepai.org/machine-learning-glossary-and-terms/probability) distribution of Y given X is the probability distribution of Y if X is known to be a particular value or a proven function of another parameter. Both can also be [categorical variables](https://deepai.org/machine-learning-glossary-and-terms/categorical-variable), in which case a probability table is used to show distribution.

This is not to be confused with the marginal distribution of a [random variable](https://deepai.org/machine-learning-glossary-and-terms/random-variable), which is a distribution without a relationship to the value of the other variable.

Conditional probability also differs from joint probability, which is the probability that both conditions are true without knowing that any of them are true.

* 1. **Give the Baye’s rule equation?**

W.K.T P(A^B) = P(A/B) P(B) 1

P(A^B) = P(B/A) P(A) 2

DIVIDINGBYP(A);

WE GET

P(B/A) = P(A/B) P(B) P(A)

## ****What are Bayesian networks?****

Bayesian networks are a broadly utilized class of probabilistic graphical models. A Bayesian network is a flexible, interpretable and compact portrayal of a joint probability distribution.

They comprise 2 sections:

1. **Parameters:** The parameters comprise restrictive likelihood circulations related to every node.
2. **Structure:** The structure is a DAG (Directed Acyclic Graph) that communicates contingent dependencies and independencies among arbitrary factors related to nodes.
   1. **Write the semantics of Bayesian network?**

Semantics of Bayesian Networks

1. Representing the full joint distribution
2. Conditional independence relations in Bayesian networks
   1. **What is meant by belief network?**

* A belief network is a graph in which the following holds
* A set of random variables
* A set of directive links or arrows connects pairs of nodes. The conditional probability table for each node
* The graph has no directed cycles.
  1. **What is goal directed node?**

In goal directed node the search is done in the backward direction from the goal state to an achievable initial node

* 1. **Why does uncertainty arise?**

Agents almost never have access to the whole truth about their environment. Agents cannot find a categorical answer. Uncertainty can also arise because of incompleteness,  incorrectness in agents understanding of properties of environment.

* 1. **What is the need for utility theory in uncertainty?**

Utility theory says that every state has a degree of usefulness, or utility to in agent, and that the agent will prefer states with higher utility. The use utility theory to represent and reason with preferences.

* 1. **What is the basic task of a probabilistic inference?**

The basic task is to reason in terms of prior probabilities of conjunctions, but for the most part, we will use conditional probabilities as a vehicle for probabilistic inference.

* 1. **What Is Called As Decision Theory?**

Preferences As Expressed by Utilities Are Combined with Probabilities in the General Theory of Rational Decisions Called Decision Theory.

Decision Theory = Probability Theory + Utility Theory.

* 1. **What is called as principle of maximum expected utility?**

The basic idea is that an agent is rational if and only if it chooses the action that yields the highest expected utility, averaged over all the possible outcomes of the action. This is known as MEU.

* 1. **Define Transition Probability?**

**Transition probability** - process moves from one state to another, as defined by the conditional distribution given the Markov blanket of the variable being sampled.

Let ***q (x*** ->***x')*** be the probability that process makes a transition from state ***x*** state ***x'.***

* 1. **Define the term utility?**

The term utility is used in the sense of "the quality of being useful .", utility of a

state is relative to the agents, whose preferences the utility function is supposed to represent.

* 1. **Define joint probability distribution**

This completely specifies an agent's probability assignments to all propositions in the domain.The joint probability distribution p(x1,x2,--------xn) assigns probabilities to all possible atomic events;where X1,X2 Xn 10 =variables.

* 1. **Define inference in Bayesian network**

Inference is the process of calculating a probability distribution of interest e.g. P(A | B=True), or P(A,B|C, D=True). The terms inference and queries are used interchangeably. The following terms are all forms of inference will slightly difference semantics.

* Prediction - focused around inferring outputs from inputs.
* Diagnostics - inferring inputs from outputs.
* Supervised anomaly detection - essentially the same as prediction
* Unsupervised anomaly detection - inference is used to calculate the P(**e**) or more commonly log(P(**e**)).
* Decision making under uncertainty - optimization and inference combined.
  1. **Define temporal model**

Agents in uncertain environment must be able to keep track of the current state of the environment ,just as logical agent, but this is difficult by partial and noisy data, because the environment is uncertain over time.

At best, the agent will be able to obtain only a probabilistic assessment of the current situation. Hence, temporal model is used by the agent in uncertain environment.

Temporal Model

1. Time & uncertainty

1. State & Observation
2. Stationary processes & the markov assumption

2. Inference in Temporal model

* 1. **What is markov process.**

In Markov assumption the current state depends only on the previous state and not on earlier state

Two types of markov assumption

1. First order assumption- in this model current state depend only on one previous state
2. Second order assumption- in this model current state depend only on two previous state
   1. **Inference in temporal model**

Inference in temporal model

1. Filtering- current state estimate and update it P(Xt/e1:t)
2. Prediction- estimate future state P(Xt+1/ e1:t)
3. Smoothing- it estimate from past state to present state P(Xk/e1:t)
4. Most likely explanation-P(X1:t/e1:t)
   1. **Explain HMM**

A Hidden Markov Model (HMM) is a statistical model which is also used in machine learning. It can be used to describe the evolution of observable events that depend on internal factors, which are not directly observable. These are a class of probabilistic graphical models that allow us to predict a sequence of unknown variables from a set of observed variables

The Hidden Markov model is a probabilistic model which is used to explain or derive the probabilistic characteristic of any random process. It basically says that an observed event will not be corresponding to its step-by-step status but related to a set of probability distributions.

**UNIT-IV**

## Name the parameters used in Markov Decision Process?

Given

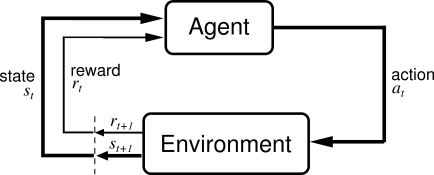
* S: set of states
* A: set of actions
* T: S x A x S x {0,1,…,H}  [0,1], Tt(s,a,s’) = P(st+1 = s’ | st = s, at =a)
* R: S x A x S x {0, 1, …, H}  < Rt(s,a,s’) = reward for (st+1 = s’, st = s, at =a)
* H: horizon over which the agent will act

Goal:

* Find ¼ : S x {0, 1, …, H}  A that maximizes expected sum of rewards, i.e.,



The Markov decision process is a model of predicting outcomes.

Like a [Markov chain](https://deepai.org/machine-learning-glossary-and-terms/markov-chain), the model attempts to predict an outcome given only information provided by the current state. However, the Markov decision process incorporates the characteristics of actions and motivations. At each step during the process, the decision maker may choose to take an action available in the current state, resulting in the model moving to the next step and offering the decision maker a reward.

1. **Define Utility function**
   1. **Define utility fun**

The agent‘s preferences are captured by a utility function, U(s), which assigns a single number to express desirability of a state.

1. **Define expected utility**

The expected utility of an action given the evidence is just the average value of outcomes, weighted by their probabilities **EU(a|e) = ∑s P(Result(a)=s|a,e) U(s)**

1. **Define MEU**

A rational agent should choose the action that maximizes the agent’s expected utility (MEU)

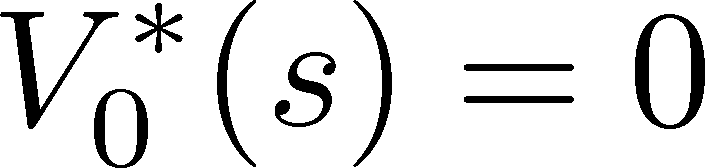
**action = argmaxa EU(a|e)**

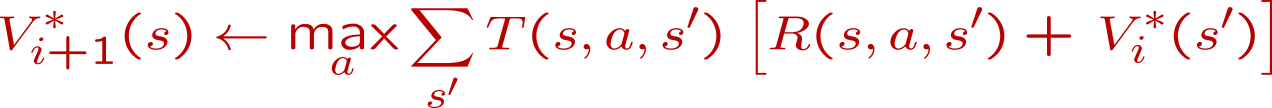
1. **List out the axioms of utility function**

The rational preferences should obey. –

* orderability: exactly one of (A > B) or (A < B) or (A ~ B) holds
* transitivity: (A < B) ∧ (B < C) ⇒ (A < C)
* continuity: (A > B > C) ⇒ ∃p [p,A; 1-p,C] ~ B
* substitutability: A ~ B ⇒ [p,A; 1-p,C] ~ [p,B; 1-p,C]
* monotonicity: A > B ⇒ (p > q ⇔ [p,A; 1-p,B] > [q,A; 1-q,B]
* decomposability: [p,A; 1-p, [q,B; 1-q,C]] ~ [p,A; (1-p)q,B; (1-p)(1-q),C]

1. **Explain value iteration algorithm**

* Algorithm:
* Start with  for all s.
* For i=1, … , H

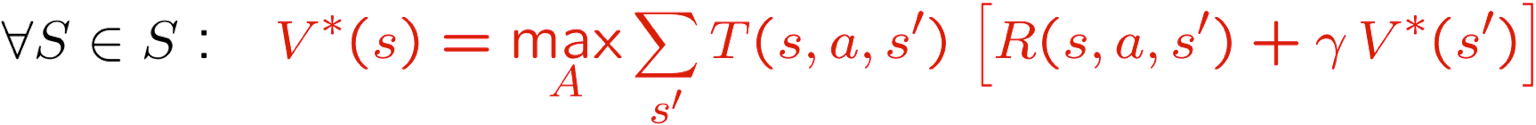
Given Vi\*, calculate for all states s 2 S:

* This is called a value update or Bellman update/back-up

1. **Define Vi\*(S)**

The expected sum of rewards accumulated when starting from state s and acting optimally for a horizon of i steps

1. **Define value iteration convergence**

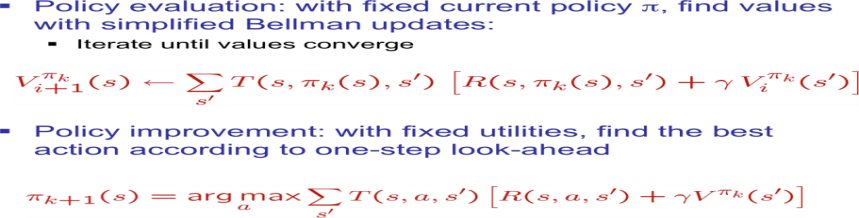


**Theorem.** Value iteration converges. At convergence, we have found the optimal value function V\* for the discounted infinite horizon problem, which satisfies the Bellman equations

1. **Define policy iteration algorithm**

* Step 1: Policy evaluation: calculate utilities for some fixed policy (not optimal utilities!) until convergence
* Step 2: Policy improvement: update policy using one- step look-ahead with resulting converged (but not optimal!) utilities as future values
* Repeat steps until policy converges

1. **Define policy iteration algorithm**

****

1. **Compare Value iteration and Policy iteration**

Value Iteration **Pros:** each iteration is very computationally efficient.

**Cons:** convergence is only asymptotic.

Policy Iteration **Pros:** converge in a finite number of iterations (often small in practice).

**Cons:** each iteration requires a full policy evaluation and it might be expensive.

### Explain Partially Observable Decision Processes

A **partially observable Markov decision process** (**POMDP**) is a combination of an [MDP](https://artint.info/html/ArtInt_224.html#mdp-defn) and a [hidden Markov model](https://artint.info/html/ArtInt_161.html). Instead of assuming that the state is observable, we assume that there are some partial and/or noisy observations of the state that the agent gets to observe before it has to act.

A POMDP consists of the following:

* *S*, a set of states of the world;
* *A*, a set of actions;
* *O*, a set of possible observations;
* *P(S0)*, which gives the probability distribution of the starting state;
* *P(S'|S,A)*, which specifies the dynamics - the probability of getting to state *S'* by doing action *A* from state *S*;
* *R(S,A,S')*, which gives the expected reward of starting in state *S*, doing action *A*, and transitioning to state *S'*; and
* *P(O|S)*, which gives the probability of observing *O* given the state is *S*.

**UNIT-V**

1. **Define Reinforcement Learning?**

Reinforcement Learning is a feedback-based Machine learning technique in which an agent learns to behave in an environment by performing the actions and seeing the results of actions. For each good action, the agent gets positive feedback, and for each bad action, the agent gets negative feedback or penalty. In Reinforcement Learning, the agent learns automatically using feedbacks without any labeled data, unlike supervised learning. Since there is no labeled data, so the agent is bound to learn by its experience only.

1. **Give an example of Reinforcement learning**

RL solves a specific type of problem where decision making is sequential, and the goal is long-term, such as game-playing, robotics, etc.

1. **State key constituents of reinforcement learning**.

**Agent():** An entity that can perceive/explore the environment and act upon it.

**Environment():** A situation in which an agent is present or surrounded by. In RL, we assume the stochastic environment, which means it is random in nature.

**Action():** Actions are the moves taken by an agent within the environment.

**State():** State is a situation returned by the environment after each action taken by the agent.

**Reward():** A feedback returned to the agent from the environment to evaluate the action of the agent.

**Policy():** Policy is a strategy applied by the agent for the next action based on the current state.

**Value():** It is expected long-term retuned with the discount factor and opposite to the short-term reward.

**Q-value():** It is mostly similar to the value, but it takes one additional parameter as a current action (a).

1. **State key features of reinforcement learning.**

In RL, the agent is not instructed about the environment and what actions need to be taken. It is based on the hit and trial process.

The agent takes the next action and changes states according to the feedback of the previous action. The agent may get a delayed reward

The environment is stochastic, and the agent needs to explore it to reach to get the maximum positive rewards.

1. **Explain approaches to implement reinforcement learning.**

There are mainly three ways to implement reinforcement-learning in ML, which are:

**Value-based**: The value-based approach is about to find the optimal value function,which is the maximum value at a state under any policy. Therefore, the agent expects the long-term return at any state(s) under policy π.

**Policy-based:** Policy-based approach is to find the optimal policy for the maximum future rewards without using the value function. In this approach, the agent tries to apply such a policy that the action performed in each step helps to maximize the future reward.

The policy-based approach has mainly two types of policy:

**Deterministic:** The same action is produced by the policy (π) at any state.

**Stochastic:** In this policy, probability determines the produced action.

**Model-based:** In the model-based approach, a virtual model is created for the environment, and the agent explores that environment to learn it. There is no particular solution or algorithm for this approach because the model representation is different for each environment.

1. What is the Bellman Equation? How is it helpful in RL?

The Bellman equation was introduced by the Mathematician Richard Ernest Bellman in the year 1953, and hence it is called as a Bellman equation. It is associated with dynamic programming and used to calculate the values of a decision problem at a certain point by including the values of previous states

The key-elements used in Bellman equations are:

* Action performed by the agent is referred to as "a"
* State occurred by performing the action is "s."
* The reward/feedback obtained for each good and bad action is "R."
* A discount factor is Gamma "γ."

The Bellman equation can be written as: V(s) = max [R(s,a) + γV(s`)] Where, V(s)= value calculated at a particular point. R(s,a) = Reward at a particular state s by performing an action. γ = Discount factor V(s`) = The value at the previous state.

1. **Explain Q-Learning.**

Q-learning is an off policy RL algorithm, which is used for the temporal difference Learning. The temporal difference learning methods are the way of comparing temporally successive predictions. It learns the value function Q (S, a), which means how good to take action "a" at a particular state "s."

1. **Explain SARSA**

State Action Reward State action (SARSA): SARSA stands for State Action Reward State action, which is an on-policy temporal difference learning method. The on-policy control method selects the action for each state while learning using a specific policy.

The goal of SARSA is to calculate the Q π (s, a) for the selected current policy π and all pairs of (s-a).

1. **Define Q-value function**

Q(s,a)=R(s,a)+ꝩ∑s`(P(s,a,s`)max(Q(s`,a`)) This formula is used to estimate the Q-values in Q-Learning.

1. Difference between Q-learning and SARSA

The main difference between

* Q-learning and SARSA algorithms is that unlike Q-learning, the maximum reward for the next state is not required for updating the Q-value in the table.
* In SARSA, new action and reward are selected using the same policy, which has determined the original action.The SARSA is named because it uses the quintuple Q(s, a, r, s', a').

Where

s: original state

a: Original action

r: reward observed while following the states s' and a': New state, action pair

Q -value equation will be: Hence, we can say that, V(s) = max [Q(s, a)]

Q(s,a)=R(s,a)+ꝩ∑s`(P(s,a,s`)max(Q(s`,a`))