Artificial Intelligence Assignment

Q1.What is an Al technique? Give some examples of problems to explain Altechniques.Why do we need to model human performance in Al?

Ans 1) Artificial intelligence (AI) techniques are methods and algorithms used to create intelligent computer systems that can learn, reason, and make decisions on their own. There are many different AI techniques, including:

1. Machine learning: This involves training a computer system to recognize patterns in data and make predictions or decisions based on that data.
2. Natural language processing: This involves enabling a computer system to understand, interpret, and generate human language.
3. Computer vision: This involves enabling a computer system to interpret and understand visual information from the world.
4. Robotics: This involves creating intelligent robotic systems that can perform tasks and make decisions based on their surroundings and input.

AI techniques are used to solve a wide range of problems, including image and speech recognition, language translation, and decision-making tasks.

We model human performance in AI because we want to create systems that can perform tasks and make decisions in a way that is similar to how humans do. This can help to make AI systems more reliable, efficient, and understandable by humans. Additionally, modeling human performance can help to ensure that AI systems are able to operate effectively in complex and dynamic environments, just as humans do.

Q2.What is heuristic function? Distinguish between uninformed search and informed search.

Ans 2) A heuristic function, also known as a "cost function" or "estimation function," is a function used to estimate the cost or distance from a given state to the goal state in a search problem. It is used to guide the search process in informed search algorithms, which are a type of search algorithm that uses additional information about the problem to guide the search process and find a solution more efficiently.

Uninformed search algorithms, on the other hand, do not use any additional information about the problem and blindly search through the search space without any direction. Examples of uninformed search algorithms include breadth-first search and depth-first search.

In general, informed search algorithms are more efficient than uninformed search algorithms because they use additional information to guide the search process and avoid searching through irrelevant or suboptimal states. However, they may require more time and resources to compute the heuristic function and may not be suitable for all types of problems.

Q3.Explain A\* search technique?

Ans 3) A\* search is a heuristic search algorithm that is used to find the shortest path between two nodes in a graph. It combines the strengths of breadth-first search, which guarantees the shortest path, and depth-first search, which has a low memory requirement.

A\* search works by maintaining a priority queue of nodes to be visited, based on an estimated cost function that measures the distance from the start node to the goal node, as well as the cost of traversing the graph from the start node to the current node. This cost function is called the "f-value" of a node and is calculated as the sum of the "g-value" (the actual cost of traversing the graph from the start node to the current node) and the "h-value" (an estimate of the remaining cost from the current node to the goal node).

At each step, A\* search selects the node with the lowest f-value from the queue and expands it by adding its neighbors to the queue. If the goal node is found, the algorithm terminates and returns the path from the start node to the goal node. If the queue becomes empty without finding the goal node, the algorithm returns failure.

A\* search is widely used in applications such as pathfinding in video games and robot navigation, due to its efficiency and optimality. It is also used in machine learning and artificial intelligence, where it can be used to search for the optimal solution to a problem

Q4.Write the difference between the propositional and predicate logic?

Ans 4) Propositional logic and predicate logic are two formal systems for representing logical statements and reasoning about them.

Propositional logic is a formal system that deals with logical propositions, which are statements that can be either true or false but not both. Propositional logic uses logical connectives, such as "and," "or," "not," "if-then," and "if and only if," to combine propositions and form more complex logical statements. It does not have the ability to represent variables or quantifiers, which are used to express concepts such as "for all" or "there exists."

Predicate logic, also known as first-order logic, is a formal system that extends propositional logic by adding the ability to represent variables and quantifiers. In predicate logic, a predicate is a function that assigns a truth value to a proposition depending on the values of its variables. For example, the predicate "x is greater than y" is true if x is greater than y and false otherwise. Predicate logic allows for more expressive power than propositional logic and is used to represent more complex logical statements and reasoning.

In summary, propositional logic is a simpler and more limited formal system that deals with logical propositions, while predicate logic is a more expressive and powerful formal system that allows for the representation of variables and quantifiers.

Top of Form

Bottom of Form

Q5.Explain Resolution Algorithm in proposition logic with an example?

Ans 5) In propositional logic, the resolution algorithm is a method for determining the satisfiability of a set of logical clauses. It works by generating new clauses by combining pairs of clauses that contain complementary literals (i.e., literals that negate each other). If a clause is generated that contains no literals, it is called a "tautology" and the set of clauses is satisfiable.

Here is an example of how the resolution algorithm works:

Consider the following set of clauses:

(A ∨ B) (¬A ∨ C) (¬B ∨ ¬C)

To apply the resolution algorithm, we look for pairs of clauses that contain complementary literals. In this case, the first and second clauses contain complementary literals (A and ¬A), so we can generate a new clause by combining them:

(A ∨ B) ∧ (¬A ∨ C) = (B ∨ C)

Next, we look for pairs of clauses that contain complementary literals. The third clause contains complementary literals with the new clause we just generated (B and ¬B), so we can generate another new clause by combining them:

(¬B ∨ ¬C) ∧ (B ∨ C) = (¬C)

Since the final clause contains no literals, it is a tautology and the original set of clauses is satisfiable.

The resolution algorithm can be used to determine the satisfiability of more complex sets of clauses as well. However, it can be computationally expensive to apply the algorithm to large sets of clauses, so it is generally used as a theoretical tool rather than a practical one.