AI1001 - Assignment 7

Shivram S
`ai24btech11031@iith.ac.in`

An agent consists of an agent program which implements the agent function, and a computing device with sensors and actuators, called the agent architecture. The architecture makes percepts available to the program, and feeds the program's action choices to the actuators.

Agent programs can be classified into several types. Table-driven agents use a table of all possible percent sequences to decide which action to take. However, these tables can be huge - even simple games like chess require a table of at least 10^{150} entries.

1 Simple Reflex Agents

The simplest kinds of agents are **simple reflex agents**, which select actions based on the current percept, ignoring the rest of the percept history. They are generally used in simpler environments, but can occur even in more complex ones. These agents work only if the environment is fully observable.

Simple reflex agents first generate an abstracted description of the current state from the percept, then use condition-action rules (**if** $\langle condition \rangle$ **then** $\langle action \rangle$) to decide which action to take. Sometimes, simple reflex agents fall into infinite loops, especially in partially observable environments. Escape from infinite loops is possible if the agent can randomize its actions.

2 Model-based Reflex Agents

Agents can handle partial observability by maintaining some sort of **internal state** that depends on percept history. This internal state information needs to updated as time goes on, which requires information about how the world evolves independently of the agent, and how the agent's actions affect the world.

The knowledge about "how the world works" is called the **transition model** of the world. The information about how the state of the world is reflected in the agent's percepts is called a **sensor model**. The transition model and the sensor model together allow an agent to keep track of the state of the world. An agent that uses such models is called a **model-based agent**.

A model-based reflex agent combines the current percept with the old internal state to generate a description of the current state. The action of the agent is then determined using rules.

3 Goal-based Agents

In order to determine which situations are desirable, an agent needs some information about it's **goal**. The agent program can combine this with the model to choose actions that achieve the goal. **Search** and **Planning** are subfields of AI devoted to finding action sequences that achieve the agent's goals.

Goal-based decision making is fundamentally different from condition-action rules in that it involves consideration of the future. Goal-based agents are more flexible because the knowledge that supports its decisions is represented explicitly and can be modified.

4 Utility-based Agents

A goal is just a crude binary-distinction ("achieved" vs. "not achieved"). **Utility** is a more general performance measure that allows a comparison of different world states. An agent's **utility function** is essentially an internalization of the performance measure. If an agent's performance measure and utility function are aligned, then actions that maximize its utility will be rational.

When it comes to flexibility and learning, a utility-based agent has more advantages than a goal-based agent. Additionally, goal-based agents can't adequately handle cases where there are conflicting goals and tradeoffs have to be made, or where no goals can be achieved with certainty.

In a partially observable or nondeterministic environment, the agent has to make decisions under uncertainty. A rational agent would maximize the **expected utility** of the action outcome.

Building a utility-based agent is not easy. The agent has to model and keep track of its environment. Choosing a utility-maximizing course of action is also difficult and requires ingenious algorithms. Even with these algorithms, perfect rationality is usually unobtainable due to computational complexity.

5 Learning Agents

Agent programs can be created by "teaching" machines that can learn. Due to the amount of work needed to program agents by hand, learning is now the preferred method for creating state-of-the-art systems.

Learning allows an agent to operate in unknown environments and become more competent over time. A learning agent consists of a **learning element**, which is responsible for making improvements and the **performance element**, which is responsible for selecting external actions. The learning agent uses feedback from a **critic** to determine how the performance element should be modified. A **problem generator** is used to suggest actions that could lead to new and informative experiences.

The agent may learn directly from its percept sequence. For example, successive states of the environment might allow the agent to learn about what its actions do and how the world evolves in response to its actions. An external performance standard is also needed to learn a reflex component or utility function. The performance standard might distinguish part of the percept as a **reward** or a **penalty**.

6 How the Components Work

The state of the world can be represented in several ways, each with different **expressiveness**. A more expressive representation is more concise, but more complex to reason about. In an **atomic representation**, the world is indivisible. Atomic representations are used in search, game-playing, hidden Markov models and Markov decision processes

In a **factored representation** , the state can be split into a set of variables or attributes, each of which have a value. Two factored states can share some values, making it much easier to Work with turning one state to another. Factored representations are used in constraint satisfaction, propositional logic, planning, Bayesian networks and machine learning algorithms.

In a **structured representation**, objects and their varying relationships can be described explicitly. Structured representations are used in relational databases, first-order logic, and much of natural language understanding.

The mapping of concepts to locations in physical memory can also be divided into two. In a **localist representation**, there is a one-to-one mapping between concepts and memory-locations. However, in a **distributed representation**, the representation of a concept is spread over many memory locations. Distributed representations are more robust against noise and information loss.