Agents in Artificial Intelligence

An “agent” is an independent program or entity that interacts with its environment by perceiving its surroundings via sensors, then acting through actuators or effectors. Agents use their actuators to run through a cycle of perception, thought, and action. This lab explains how to implement a simple agent, create an environment, and implement a program that helps the agent act on the environment based on its percepts.

These are the main four rules all AI agents must adhere to:

- Rule 1: An AI agent must be able to perceive the environment.
- Rule 2: The environmental observations must be used to make decisions.
- Rule 3: The decisions should result in action.
- Rule 4: The action taken by the AI agent must be a rational. Rational actions are actions that maximize performance and yield the best positive outcome.

Different agents that will be developed in the lab are discussed below.

Random Agent

Randomly choose one of the actions from action subset and perform the action on environment.

Reflex Agents:

These agents work here and now and ignore the past. They respond using the event-condition-action rule. The ECA rule applies when a user initiates an event, and the Agent turns to a list of pre-set conditions and rules, resulting in pre-programmed outcomes.

Model-based Agents:

An agent that keeps track of what actions are performed. These agents choose their actions like reflex agents do, but they have a better comprehensive view of the environment. An environmental model is programmed into the internal system, incorporating into the Agent's history.

Goal-based agents

These agents build on the information that a model-based agent stores by augmenting it with goal information or data regarding desirable outcomes and situations.

Implementation of Vacuum Agent to clean the room.

Designing of an agent that cleans the room. Room consists of mxn tiles that are either clean or dirty.

- Create a M x N grid that refers to room
- Agent can start at any tile on the floor
- A smart agent that cleans rooms size of n * n
- Agent can move Up, Down, Left, Right
- Agent cleans the tile if it is dirty otherwise it moves to next slide.
- Calculate performance each round.

```
class ReflexVacuumAgent(Agent):
    "A reflex agent for the two-state vacuum environment."

    def __init__(self):
        Agent.__init__(self)
        def program((location, status)):
            if status == 'Dirty': return 'Suck'
            elif location == loc_A: return 'Right'
            elif location == loc_B: return 'Left'
        self.program = program


def RandomVacuumAgent():
    "Randomly choose one of the actions from the vaccum environment."
    return RandomAgent(['Right', 'Left', 'Suck', 'NoOp'])
```

Components of Search Space for Goal driven Agent

Initial State

Define a function that generates the configuration for a start and goal state either randomly or based on user input

Configuration is often represented via list, array and dictionary

Action

Function check and apply allowable moves

Transition Model

Function that applies the actions to generate next state (or child state)

Goal Test

Function that compares the start configuration with final configuration

Path Cost

Make a variable that value increments for each state

```
class ModelBasedVacuumAgent(Agent):
    "An agent that keeps track of what locations are clean or dirty."
    def __init__(self):
        Agent.__init__(self)
        model = {loc_A: None, loc_B: None}
        def program((location, status)):
            "Same as ReflexVacuumAgent, except if everything is clean, do NoOp"
            model[location] = status ## Update the model here
            if model[loc_A] == model[loc_B] == 'Clean': return 'NoOp'
            elif status == 'Dirty': return 'Suck'
            elif location == loc_A: return 'Right'
            elif location == loc_B: return 'Left'
        self.program = program
```

Components of Search Space for Vacuum Cleaner Agent

States

Number of tiles in room

Initial State

Any state

Actions

Left, right and suck

Transition model

Complete state-space

Goal test

Whether all squares are clean

Path cost

Cost of one for each move

Codes for implementations are attached.

Overview of implemented agents

- **Generate a Simple Reflex Agent**
    - **Develop/implement**
        - **Start Configuration using dictionary**
            - **Represent the tiles/location (A and B) as key of dictionary and states as values (0 for clean, 1 for dirt)**
        - **List of Actions**
        - **Input the percept from user**
        - **Condition Action rule using if/else statement, if/elif/else compare the percept with condition action rule and then apply the rule**
- **Generate a Random Agent**

    - **Same as simple reflex agent, but it performs random action instead of taking input/percept from the user**

- **Generate a Goal based Agent**
    - **Add a goal test in simple reflex agent**
    - **Use a while loop to iteratively implement the condition action rule unless reach to a goal state**