Artificial Intelligence

BS (CS) _SP_2024

Lab_04 Manual

Learning Objectives:

1. Agent Basis
2. Environment
3. Simple Reflex Agent
4. Model Based Agent
5. Goal Based Agent
6. Code Examples

Agents

An **agent** is anything that can be viewed as perceiving its environment through **sensors** and acting upon that environment through **actuators**. **Percepts** refer to the agent's perceptual inputs at any given instant. An agent's **percept sequence** is the complete history of everything the agent has ever perceived. In general, an agent's choice of action at any given instant can depend on the entire percept sequence observed to date, but not on anything it hasn't perceived.

An agent is composed of:

- **Percept:** Information received from the environment.
- **Action:** A set of actions the agent can perform.

```
class Agent():

  def initial_action(self, percept):
    """return the initial action."""
    return self.select_action(percept) # same as select_action

  def select_action(self, percept):
    """return the next action (and update internal state) given percept
    percept is variable:value dictionary"""
    raise NotImplementedError("go") # abstract method
```

Environment

Define the environment in which the agent operates. This includes the data the agent perceives.

```
class Environment:

    def __init__(self):
        #  Define environment attributes
      pass

    def thing_classes(self):
        return []

    def percept(self, agent):
```

```
        """Return the percept that the agent sees at this point.
(Implement this.)"""
        raise NotImplementedError

    def execute_action(self, agent, action):
        """Change the world to reflect this action. (Implement this.)"""
        raise NotImplementedError

    def default_location(self, thing):
        """Default location to place a new thing with unspecified
location."""
        return None

#  Other methods . . . .
```

Simple Reflex Agent

Simple reflex agents make decisions based on the current percept only.

```
class GoalBasedAgent:
    def __init__(self, goals):
        # Define agent attributes, including a set of goals
        pass

    def formulate_goal(self):
        # Formulate a goal to be achieved
        pass

    def decide(self, percept):
        # Make a decision based on the current percept and goals
        pass
```

Example

```
class GoalBasedAgent:
  def simple_reflex_agent(percept):
    if percept == 'Dirty':
        return 'Suck'
    elif percept == 'Clean':
        return 'Move'
```

Model Based Agent

Model-based reflex agents maintain an internal state to make decisions.

```
class ModelBasedReflexAgent:
    def __init__(self):
        # Define agent attributes, including a model of the environment
        pass

    def update_model(self, percept):
        # Update the agent's internal model based on the percept
        pass

    def decide(self):
        # Make a decision based on the internal model
        pass
```

Example

```
class ModelBasedReflexAgent:
  def model_based_agent(percept, model):
    if model['current_location'] == 'A' and percept == 'Dirty':
        return 'Suck'
    elif model['current_location'] == 'B' and percept == 'Dirty':
        return 'Suck'
    else:
        return 'Move'
```

Goal Based Agent

Goal-based agents work towards achieving specific goals.

```
class GoalBasedAgent:
    def __init__(self, goals):
        # Define agent attributes, including a set of goals
        pass

    def formulate_goal(self):
        # Formulate a goal to be achieved
        pass

    def decide(self, percept):
        # Make a decision based on the current percept and goals
        pass
```

Example:

```
class GoalBasedAgent:
    def __init__(self, goals):
        # Define agent attributes, including a set of goals
        pass

    def goal_based_agent(self,current_state,goal_state):
        # Formulate a goal to be achieved
        if current_state == goal_state:
          return 'Nothing'  #If the goal is achieved, do nothing
        else:
          return decide(current_state, goal_state)

    def decide(self, current_state,goal_state):
        # Make a decision based on the current percept and goals
        return next_action
```

Coding Example

Vacuum Cleaner:

```
class VacuumAgent:
    def __init__(self):
        self.location = 'A'
        self.performance = 0

    def percept(self, location, dirt):
        self.location = location
        return (location, dirt)

    def act(self, percept):
        location, dirt = percept

        if dirt:
            self.suck()
        else:
            self.move()

    def suck(self):
        print("Sucking dirt at", self.location)
        self.performance += 1

    def move(self):
```

```
        print("Moving to the other square")
        self.location = 'A' if self.location == 'B' else 'B'


def run_vacuum_agent(agent, steps=10):
    for step in range(steps):
        dirt = bool(random.getrandbits(1))  # Randomly generate dirt
        percept = agent.percept(agent.location, dirt)
        agent.act(percept)

    print("Performance Score:", agent.performance)


# Example usage
import random

vacuum_agent = VacuumAgent()
run_vacuum_agent(vacuum_agent)
```