CS 188: Artificial Intelligence

Agents and environments

Instructor: Fabrice Popineau
Outline

• Agents and environments
• Rationality
• PEAS (Performance measure, Environment, Actuators, Sensors)
• Environment types
• Agent types
Agents and environments

- An agent *perceives* its environment through *sensors* and *acts* upon it through *actuators* (or *effectors*, depending on whom you ask)
• Are humans agents?
• Yes!
  • Sensors = vision, audio, touch, smell, taste, proprioception
  • Actuators = muscles, secretions, changing brain state
• Are pocket calculators agents?
• Yes!
  • Sensors = key state sensors
  • Actuators = digit display
Agents and environments

• AI is more interested in agents with substantial computation resources and environments requiring nontrivial decision making
• The **agent function** maps from percept histories to actions:
  • \( f : P^* \rightarrow A \)
  • I.e., the agent’s response to any sequence of percepts

• The **agent program** runs on some machine \( M \) to implement \( f \):
  • \( f = \text{Agent}(l,M) \)
  • Real machines have limited speed and memory
  • The program may take time to choose actions, may be interrupted by new percepts (or ignore them), etc.
  • Can every agent function be implemented by some agent program?
    • No! Consider agent for halting problems, NP-hard problems, chess with a slow PC
Example: Vacuum world

- Percepts: [location, status], e.g., [A, Dirty]
- Actions: Left, Right, Suck, NoOp
**Agent function**

<table>
<thead>
<tr>
<th>Percept sequence</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>[A,Clean]</td>
<td>Right</td>
</tr>
<tr>
<td>[A,Dirty]</td>
<td>Suck</td>
</tr>
<tr>
<td>[B,Clean]</td>
<td>Left</td>
</tr>
<tr>
<td>[B,Dirty]</td>
<td>Suck</td>
</tr>
<tr>
<td>[A,Clean],[B,Clean]</td>
<td>Left</td>
</tr>
<tr>
<td>[A,Clean],[B,Dirty]</td>
<td>Suck</td>
</tr>
<tr>
<td>etc</td>
<td>etc</td>
</tr>
</tbody>
</table>

**Agent program**

```python
function Reflex-Vacuum-Agent([location,status])
returns an action
if status = Dirty then return Suck
else if location = A then return Right
else if location = B then return Left
```

What is the *right* agent function?

Can it be implemented by a small agent program?

(Can we ask, “What is the right agent program?”)
Rationality

• Fixed **performance measure** evaluates the environment sequence
  • one point per square cleaned up?
    • NO! Rewards an agent who dumps dirt and cleans it up
  • one point per clean square per time step, for $t = 1,\ldots,T$

• A **rational agent** chooses whichever action maximizes the **expected** value of the performance measure
  • given the percept sequence to date and prior knowledge of environment

Does Reflex-Vacuum-Agent implement a rational agent function?

  Yes, if movement is free, or new dirt arrives frequently
Rationality, contd.

- Are rational agents **omniscient**?
  - No – they are limited by the available percepts

- Are rational agents **clairvoyant**?
  - No – they may lack knowledge of the environment dynamics

- Do rational agents **explore** and **learn**?
  - Yes – in unknown environments these are essential

- So rational agents are not necessarily successful, but they are **autonomous** (i.e., transcend initial program)
• [dung beetle video]
The task environment - PEAS

- **Performance measure**
  - -1 per step; +10 food; +500 win; -500 die; +200 hit scared ghost

- **Environment**
  - Pacman dynamics (incl ghost behavior)

- **Actuators**
  - Left Right Up Down

- **Sensors**
  - Entire state is visible

Note: formal evaluation of an agent requires defining a distribution over Instances of the environment class
PEAS: Automated taxi

• Performance measure

• Environment

• Actuators

• Sensors
PEAS: Automated taxi

- **Performance measure**
  - Income, happy customer, vehicle costs, fines, insurance premiums

- **Environment**
  - US streets, other drivers, customers

- **Actuators**
  - Steering, brake, gas, display/speaker

- **Sensors**
  - Camera, radar, accelerometer, engine sensors, microphone
PEAS: Medical diagnosis system

- **Performance measure**
  - Patient health, cost, reputation

- **Environment**
  - Patients, medical staff, insurers, courts

- **Actuators**
  - Screen display, email

- **Sensors**
  - Keyboard/mouse
<table>
<thead>
<tr>
<th>Environment types</th>
<th>Pacman</th>
<th>Backgammon</th>
<th>Diagnosis</th>
<th>Taxi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fully or partially observable</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Single-agent or multiagent</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deterministic or stochastic</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Static or dynamic</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Discrete or continuous</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Known or unknown</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Agent design

• The environment type largely determines the agent design
  • Partially observable => agent requires memory (internal state)
  • Stochastic => agent may have to prepare for contingencies
  • Multi-agent => agent may need to behave randomly
  • Static => agent has time to compute a rational decision
  • Continuous time => continuously operating controller
Agent types

- In order of increasing generality and complexity
  - Simple reflex agents
  - Reflex agents with state
  - Goal-based agents
  - Utility-based agents
Simple Reflex Agents

- Reflex agents:
  - Choose action based on current percept (and maybe memory)
  - May have memory or a model of the world’s current state
  - Do not consider the future consequences of their actions
  - Consider how the world IS

- Can a reflex agent be rational?
Simple reflex agents

Agent

What the world is like now

Environment

What action I should do now

Condition-action rules

Actuators

Sensors
class GoWestAgent(Agent):

def getAction(self, percept):
    if Directions.WEST in percept.getLegalPacmanActions():
        return Directions.WEST
    else:
        return Directions.STOP
Video of Demo Reflex Optimal
Video of Demo Reflex Odd
Pacman agent contd.

• Can we (in principle) extend this reflex agent to behave well in all standard Pacman environments?
Handling complexity

- Writing behavioral rules or environment models more difficult for more complex environments
- E.g., rules of chess (32 pieces, 64 squares, ~100 moves)
  - ~100 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 pages as a state-to-state transition matrix (cf HMMs, automata)
  - \texttt{R.B.KB.RPPP..PPP..N..N.....PP....q.pp..Q..n..n..ppp..pppr.b.kb.r}
  - ~100 000 pages in propositional logic (cf circuits, graphical models)
    - \texttt{WhiteKingOnC4@Move12} ⇔ ...
  - 1 page in first-order logic
    - \( \forall x,y,t,\text{color},\text{piece} \ On(\text{color},\text{piece},x,y,t) \leftrightarrow ... \)
Reflex agents with state
Goal-based agents
Utility-based agents

Agent

Environment

Sensors

State

What the world is like now

What it will be like if I do action $A$

How happy I will be in such a state

What action I should do now

How the world evolves

What my actions do

Utility
Planning Agents

• Planning agents:
  • Ask “what if”
  • Decisions based on (hypothesized) consequences of actions
  • Must have a model of how the world evolves in response to actions
  • Must formulate a goal (test)
    • Consider how the world WOULD BE

• Optimal vs. complete planning

• Planning vs. replanning

[Demo: re-planning (L2D3)]
[Demo: mastermind (L2D4)]
Video of Demo Replanning
Video of Demo Mastermind
• An **agent** interacts with an **environment** through **sensors** and **actuators**

• The **agent function**, implemented by an **agent program** running on a **machine**, describes what the agent does in all circumstances

• PEAS descriptions define task environments; precise PEAS specifications are essential

• More difficult environments require more complex agent designs and more sophisticated representations