Intelligent Agents

Chapter 2

Reminders

Assignment 0 (lisp refresher) due 9/8
account forms from 727 Soda.

Lisp/emacs tutorial: 10-12 and 3.30-4.30 on Fri 9/2, 273 Soda

My office hours on Tuesday moved to 4.30-5.30

Section swapping proposal
Blaine to teach 106 (Wed 4-5) instead of 104 (Wed 12-1)
John to teach 104 (Wed 12-1) instead of 106 (Wed 4-5)
⇒ non-CS students in 104 switch to 106

Outline

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

Agents and environments

Outline

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

Agents include humans, robots, softbots, thermostats, etc.
The agent function maps from percept histories to actions:
\[ f : P^n \to A \]
The agent program runs on the physical architecture to produce \( f \)

A vacuum-cleaner agent

Percepts: location and contents, e.g., \([A, \text{Dirty}]\)
Actions: \(\text{Left, Right, Suck, NoOp}\)

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 function?
Can it be implemented in a small agent program?
Rationality

Fixed performance measure evaluates the environment sequence
- one point per square cleaned up in time $T$ WYAFIWYG
- one point per clean square per time step, minus one per move?
- penalize for $> k$ dirty squares?

A rational agent chooses whichever action maximizes the expected value of
the performance measure given the percept sequence to date

Rational $\neq$ omniscient
- percepts may not supply all relevant information
Rational $\neq$ clairvoyant
- action outcomes may not be as expected
Hence, rational $\neq$ successful

Rational $\Rightarrow$ exploration, learning, autonomy

Internet shopping agent

Performance measure??
Environment??
Actuators??
Sensors??

Environment types

<table>
<thead>
<tr>
<th>Observable??</th>
<th>Peg Solitaire</th>
<th>Backgammon</th>
<th>Internet shopping</th>
<th>Taxi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deterministic??</td>
<td>Episodic??</td>
<td>Static??</td>
<td>Discrete??</td>
<td>Single-agent??</td>
</tr>
</tbody>
</table>
The environment type largely determines the agent design

The real world is (of course) partially observable, stochastic, sequential, dynamic, continuous, multi-agent
Agent types

Four basic types in order of increasing generality:
- simple reflex agents
- reflex agents with state
- goal-based agents
- utility-based agents

All these can be turned into learning agents

Problems with simple reflex agents

Simple reflex agents fail in partially observable environments
E.g., suppose location sensor is missing
Agent (presumably) Sucks if Dirty; what if Clean?
⇒ infinite loops are unavoidable

Randomization helps (why??), but not that much

Example

```lisp
(defun reflex-vacuum-agent (percept)
  (destructuring-bind (location status) percept
    (cond ((eq status 'Dirty) 'Suck)
          ((eq location 'A) 'Right)
          ((eq location 'B) 'Left)))))
```

```
(setf joe (make-agent :body (make-agent-body)
                     :program '#'(lambda (percept)
                               (destructuring-bind (location status) percept
                                 (cond ((eq status 'Dirty) 'Suck)
                                       ((eq location 'A) 'Right)
                                       ((eq location 'B) 'Left))))))
```

Example

```lisp
(defun reflex-vacuum-agent-with-state (percept)
  (destructuring-bind (location status) percept
    (if (> last-B 3) 'Left 'NoOp))
  (if (eq location 'A) (setq last-A 0) (setq last-B 0))
  (setq last-A (incf last-A))
  (setq last-B (incf last-B))
)
```

```
(let ((last-A infinity) (last-B infinity))
  (defun reflex-vacuum-agent-with-state (percept)
    (cond ((eq status 'Dirty)
            (if (eq location 'A) (setq last-A 0) (setq last-B 0))
                'Suck)
          ((eq location 'A) (if (> last-B 3) 'Right 'NoOp))
          ((eq location 'B) (if (> last-A 3) 'Left 'NoOp))))
  #'reflex-vacuum-agent-with-state)
```
**Summary**

Agents interact with environments through actuators and sensors.

The agent function describes what the agent does in all circumstances.

The performance measure evaluates the environment sequence.

A perfectly rational agent maximizes expected performance.

Agent programs implement (some) agent functions.

PEAS descriptions define task environments.

Environments are categorized along several dimensions:
- observable?
- deterministic?
- episodic?
- static?
- discrete?
- single-agent?

Several basic agent architectures exist:
- reflex, reflex with state, goal-based, utility-based