Intelligent Agents

Philipp Koehn

8 February 2024
Agents and Environments

- Agents include humans, robots, softbots, thermostats, etc.

- The agent function maps from percept histories to actions:

$$f : \mathcal{P}^* \rightarrow A$$

- The agent program runs on the physical architecture to produce $f$
Vacuum Cleaner World

• Percepts: location and contents, e.g., \([A, Dirty]\)

• Actions: \textit{Left}, \textit{Right}, \textit{Suck}, \textit{NoOp}
Vacuum Cleaner Agent

Table

<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], [A, Clean]</td>
<td>Right</td>
</tr>
<tr>
<td>[A, Clean], [A, Dirty]</td>
<td>Suck</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Function

Input: location, status
Output: action

1: if status = Dirty then
2: return Suck
3: end if
4: if location = A then
5: return Right
6: end if
7: if location = B then
8: return Left
9: end if

- 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 \( T \)?
  - 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
  \( \rightarrow \) percepts may not supply all relevant information

- Rational \( \neq \) clairvoyant
  \( \rightarrow \) action outcomes may not be as expected

- Hence, rational \( \neq \) successful

- Rational \( \implies \) exploration, learning, autonomy
intelligent agent
Intelligent Agent

• Definition:

An intelligent agent perceives its environment via sensors and acts rationally upon that environment with its effectors.

• A discrete agent receives percepts one at a time, and maps this percept sequence to a sequence of discrete actions.

• Properties

  – autonomous
  – reactive to the environment
  – pro-active (goal-directed)
  – interacts with other agents via the environment
Sensors/Percepts and Effectors/Actions

- For example: humans
  - **Sensors**: Eyes (vision), ears (hearing), skin (touch), tongue (gustation), nose (olfaction), neuromuscular system (proprioception)
  - **Percepts**:
    * At the lowest level: electrical signals from these sensors
    * After preprocessing: objects in the visual field (location, textures, colors, ...), auditory streams (pitch, loudness, direction), ...
  - **Effectors**: limbs, digits, eyes, tongue, ...
  - **Actions**: lift a finger, turn left, walk, run, carry an object, ...

- Percepts and actions need to be carefully defined, possibly at different levels of abstraction
Example: Self-Driving Car

- **Percepts:** Video, sonar, speedometer, odometer, engine sensors, keyboard input, microphone, GPS, ...

- **Actions:** Steer, accelerate, brake, horn, speak/display, ...

- **Goals:** Maintain safety, reach destination, maximize profits (fuel, tire wear), obey laws, provide passenger comfort, ...

- **Environment:** U.S. urban streets, freeways, traffic, pedestrians, weather, customers, ...

- Different aspects of driving may require different types of agent programs
Rationality

• An ideal rational agent should, for each possible percept sequence, do whatever actions will maximize its expected performance measure based on
  – percept sequence
  – built-in and acquired knowledge

• Rationality includes information gathering, not “rational ignorance” (If you don’t know something, find out!)

• Need a performance measure to say how well a task has been achieved

• Types of performance measures
  – false alarm (false positive) rate
  – false dismissal (false negative) rate
  – speed
  – resources required
  – impact on environment
  – etc.
Autonomy

- A system is autonomous to the extent that its own behavior is determined by its own experience.

- Therefore, a system is not autonomous if it is guided by its designer according to a priori decisions.

- To survive, agents must have
  - enough built-in knowledge to survive
  - ability to learn
agent types
Agent Types

- **Table-driven agents**
  use a percept sequence/action table in memory to find the next action. They are implemented by a (large) lookup table.

- **Simple reflex agents**
  are based on condition-action rules, implemented with an appropriate production system. They are stateless devices which do not have memory of past world states.

- **Agents with memory**
  have internal state, which is used to keep track of past states of the world.

- **Agents with goals**
  are agents that, in addition to state information, have goal information that describes desirable situations. Agents of this kind take future events into consideration.

- **Utility-based agents**
  base their decisions on classic axiomatic utility theory in order to act rationally.
Architecture of Table-Driven/Reflex Agent

- Agent
- Sensors
- What the world is like now
- Environment
- Condition-action rules
- What action I should do now
- Effectors
Table-Driven Agents

- Table lookup of percept-action pairs mapping from every possible perceived state to the optimal action for that state

Problems

- too big to generate and to store (Chess has about $10^{120}$ states, for example)
- no knowledge of non-perceptual parts of the current state
- not adaptive to changes in the environment; requires entire table to be updated if changes occur
- looping: can’t make actions conditional on previous actions/states
Simple Reflex Agents

- **Rule-based reasoning** to map from percepts to optimal action; each rule handles a collection of perceived states

- Problems
  - still usually too big to generate and to store
  - still no knowledge of non-perceptual parts of state
  - still not adaptive to changes in the environment; requires collection of rules to be updated if changes occur
  - still can’t make actions conditional on previous state
Architecture of Agent with Memory
Agents with Memory

- Encode "internal state" of world to remember past contained in earlier percepts

- Needed because sensors do not usually give the entire state of the world at each input, so perception of the environment is captured over time.

- "State" is used to encode different "world states" that generate the same immediate percept

- Requires ability to represent change in the world; one possibility is to represent just the latest state, but then can’t reason about hypothetical courses of action

- Example: Rodney Brooks’s Subsumption Architecture
Brooks’ Subsumption Architecture

- Main idea: build complex, intelligent robots by decomposing behaviors into a hierarchy of skills, each completely defining a complete percept-action cycle for one very specific task

- Examples:
  - avoiding contact
  - wandering
  - exploring
  - recognizing doorways

- Each behavior is modeled by a finite-state machine with a few states

- Behaviors are loosely coupled, asynchronous interactions
Architecture of Goal-Based Agent
Goal-Based Agent

- Choose actions so as to achieve a (given or computed) goal.
- A goal is a description of a desirable situation.
- Keeping track of the current state is often not enough: need to add goals to decide which situations are good.
- Deliberative instead of reactive.
- May have to consider long sequences of possible actions before deciding if goal is achieved (involves consideration of the future, “what will happen if I do...?”)
Architecture of Utility-Based Agent

- State
- How the world evolves
- What my actions do
- Utility

Sensors
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

Agent
Environment

Effectors
Utility-Based Agent

• When there are multiple possible alternatives, how to decide which one is best?

• A goal specifies a crude distinction between a happy and unhappy state, but often need a more general performance measure that describes "degree of happiness."

• Utility function

\[ U: \text{State} \rightarrow \text{Real Numbers} \]

indicating a measure of success or happiness when at a given state.

• Allows decisions comparing choice between conflicting goals, and choice between likelihood of success and importance of goal (if achievement is uncertain).
environment
Properties of Environments

- **Accessible/Inaccessible.**
  - if an agent’s sensors give it access to the complete state of the environment needed to choose an action, the environment is accessible.
  - such environments are convenient, since the agent is freed from the task of keeping track of the changes in the environment.

- **Deterministic/Nondeterministic**
  - an environment is deterministic if the next state of the environment is completely determined by the current state of the environment and the action of the agent.
  - in an accessible and deterministic environment, the agent need not deal with uncertainty.

- **Episodic/Sequential**
  - an episodic environment means that subsequent episodes do not depend on what actions occurred in previous episodes.
  - such environments do not require the agent to plan ahead.
Properties of Environments

• Static/Dynamic
  – a static environment does not change while the agent is thinking.
  – the passage of time as an agent deliberates is irrelevant.
  – the agent doesn’t need to observe the world during deliberation.

• Discrete/Continuous
  – if the number of distinct percepts and actions is limited, the environment is discrete, otherwise it is continuous.

• With/Without intelligent adversaries
  – if the environment contains intelligent, adversarial agents, the agent needs to be concerned about strategic, game-theoretic aspects of the environment.
  – most engineering environments don’t have rational adversaries, whereas most social and economic systems get their complexity from the interactions of (more or less) rational agents.
## Properties of Environments

<table>
<thead>
<tr>
<th></th>
<th>Accessible</th>
<th>Deterministic</th>
<th>Episodic</th>
<th>Static</th>
<th>Discrete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image Classification</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Solitaire</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Backgammon</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Taxi driving</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Internet shopping</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Medical diagnosis</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Properties of Environments

<table>
<thead>
<tr>
<th>Environment</th>
<th>Accessible</th>
<th>Deterministic</th>
<th>Episodic</th>
<th>Static</th>
<th>Discrete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image Classification</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Solitaire</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Backgammon</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Taxi driving</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Internet shopping</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Medical diagnosis</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Properties of Environments

<table>
<thead>
<tr>
<th>Environment</th>
<th>Accessible</th>
<th>Deterministic</th>
<th>Episodic</th>
<th>Static</th>
<th>Discrete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image Classification</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Solitaire</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Backgammon</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Taxi driving</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Internet shopping</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Medical diagnosis</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Properties of Environments

<table>
<thead>
<tr>
<th>Environment</th>
<th>Accessible</th>
<th>Deterministic</th>
<th>Episodic</th>
<th>Static</th>
<th>Discrete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image Classification</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Solitaire</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Backgammon</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Taxi driving</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Internet shopping</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Medical diagnosis</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Properties of Environments

<table>
<thead>
<tr>
<th>Environment</th>
<th>Accessible</th>
<th>Deterministic</th>
<th>Episodic</th>
<th>Static</th>
<th>Discrete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image Classification</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Solitaire</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Backgammon</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Taxi driving</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Internet shopping</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Medical diagnosis</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Properties of Environments

<table>
<thead>
<tr>
<th>Environment</th>
<th>Accessible</th>
<th>Deterministic</th>
<th>Episodic</th>
<th>Static</th>
<th>Discrete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image Classification</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Solitaire</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Backgammon</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Taxi driving</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Internet shopping</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Medical diagnosis</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Properties of Environments

<table>
<thead>
<tr>
<th>Environment</th>
<th>Accessible</th>
<th>Deterministic</th>
<th>Episodic</th>
<th>Static</th>
<th>Discrete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image Classification</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Solitaire</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Backgammon</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Taxi driving</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Internet shopping</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Medical diagnosis</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

⇒ lots of real-world domains fall into the hardest case
summary
Summary

• An **agent** perceives and acts in an environment, has an architecture, and is implemented by an agent program.

• An **ideal agent** always chooses the action which maximizes its expected performance, given its percept sequence so far.

• An **autonomous agent** uses its own experience rather than built-in knowledge of the environment by the designer.

• An **agent program** maps from percept to action and updates its internal state.
  – **reflex agent** responds immediately to percepts.
  – **goal-based agent** acts in order to achieve their goal(s).
  – **utility-based agent** maximizes their own utility function.

• **Representing knowledge** is important for successful agent design.

• Most challenging environments are inaccessible, nondeterministic, nonepisodic, dynamic, and continuous, and contain intelligent adversaries.