COMP219: Artificial Intelligence

Lecture 4: Intelligent Agents

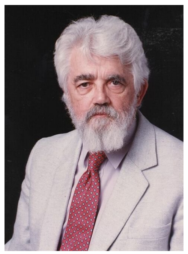
Overview

- Last time
  - An introduction to Prolog
- Today:
  - A brief history of AI
  - Introduce 'agents'
  - Consider agent task environments
  - Consider agent program designs
- Learning outcome covered today:

  Identify or describe the characteristics of intelligent agents and the environments that they can inhabit.

Brief History of AI 1943–56

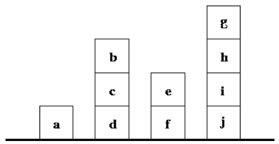
- McCulloch & Pitts (1943)
  - artificial neural net - proved equivalent to Turing machine
- Shannon, Turing (1950)
  - Information theory
  - Turing Test
  - chess playing programs
- Marvin Minsky (1951)
  - first neural net computer - SNARC
- Dartmouth College (1956)
  - term "AI" coined by John McCarthy
  - Newell & Simon presented LOGIC THEORIST program



"Every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it"
Dartmouth manifesto

1956-70

- Programs written that could
  - plan, learn, play games, prove theorems, solve problems
- Major centres established
  - Minsky - MIT
  - McCarthy - Stanford
  - Newell & Simon - CMU
- Major feature of the period was *microworlds* - toy problem domains
  - Example: blocks world
  - "It'll scale, honest. . . "



1969: First International Joint Conference on Artificial Intelligence held
1970: First Issue of Artificial Intelligence journal

1970s

- 1970s period of recession for AI (an 'AI Winter')
  - (Lighthill report in UK) “formed the basis for the decision by the British government to end support for AI research in all but two universities [from AIAMA]”
- Techniques developed on microworlds would not scale
- Implications of complexity theory developed in late 1960s, early 1970s began to be appreciated
- Brute force techniques will not work
- Works in principle does not mean works in practice
- In US – foundational work on expert and knowledge based systems

1980s

- General purpose, brute force techniques don’t work, so use *knowledge rich* solutions
- Early 1980s saw emergence of expert systems as systems capable of exploiting knowledge about tightly focused domains to solve problems normally considered the domain of experts

- Ed Feigenbaum’s knowledge principle
- In UK Alvey programme (1984-89) revived funding and interest

1990s

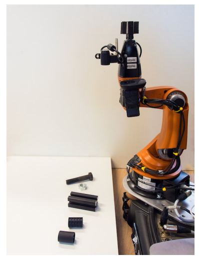
- Many companies set up to commercialise expert systems technology went bust ('AI winter')
- 1990s: emphasis on understanding the interaction between *agents and environments*
- AI as *component*, rather than as end in itself

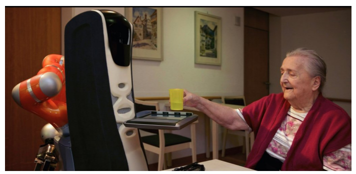
www.cis.upenn.edu

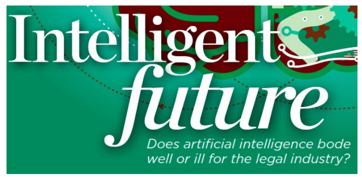
www.nasa.gov

2000s onwards

- Agents developed as a key technology for symbolic AI
  - www as a delivery mechanism
- Revival of sub symbolic AI and probability networks
- Advances in robotics, vision, etc.
- Fielded applications emerging…







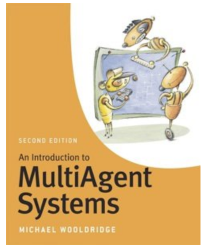
Intelligent Agents

Intelligent Agents

- The intelligent entities that we engineer in AI are known as agents.
- A popular characterisation by Wooldridge and Jennings (1995) is:

  "An *agent* is a computer system that is *situated* in some *environment*, and that is capable of *autonomous action* in this environment in order to meet its design objectives."

- A collection of such agents situated together in an environment and capable of interacting in with one another is known as a 'multi-agent system'.
- Autonomy is central to the notion of agency.



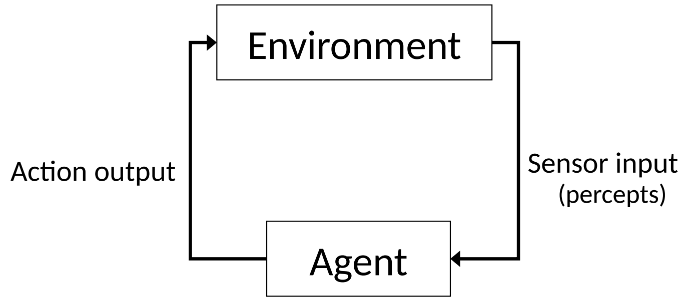
Intelligent Agents

- Capabilities that we would expect such intelligent agents to possess (again from Wooldridge and Jennings (1995)):
- Reactivity: Intelligent agents are able to perceive their environment, and respond in a timely fashion to changes that occur in it in order to satisfy their design objectives.

- Proactiveness: Intelligent agents are able to exhibit goal-directed behaviour by taking the initiative in order to satisfy their design objectives.

- Social ability: Intelligent agents are capable of interacting with other agents (and possibly humans) in order to satisfy their design objectives.

Intelligent Agents



Task Environments

- Agents are situated within task environments, which differ in accordance with the particular problem area that the agent is designed to address.

- When we design agents to solve particular problems, we must specify the task environment as fully as possible. Four elements to take into account:

- PEAS (from AIAMA):
  - Performance measure: the criteria by which we can measure the success of an agent's behaviour.
  - Environment: the external environment that the agent inhabits.
  - Actuators: the means by which the agent acts within its environment.
  - Sensors: the means by which the agent senses its environment.

Example

- Consider an agent used for medical diagnosis; Its PEAS description might be as follows:

- Performance measure: health of patient, costs of treatment.
- Environment: patient, hospital, staff.
- Actuators: display questions, tests, diagnoses and treatments.
- Sensors: keyboard entry of patient's symptoms, responses to questions and findings.

Exercise

Develop a PEAS description of the task environment for a mobile agent designed to roam the moon and gather rock samples.

Task Environments

- The properties of the task environment that the agent inhabits may differ greatly, depending upon the particular application area.

- Russell and Norvig have given a classification of the different types of properties of agent environments:

- Fully observable vs partially observable
- Deterministic vs stochastic
- Episodic vs sequential
- Static vs dynamic
- Discrete vs continuous

Fully Observable vs Partially Observable

- Fully observable environment: one in which the agent can fully obtain complete, up-to-date info about the environment's state.
- Most moderately complex environments are partially observable.
- Fully observable environments are more convenient
  - agent does not need to maintain any internal state to keep track of the environment.
  - simpler to build agents for such environments.
- Fully observable env. examples: a crossword puzzle, the game of backgammon.
- Partially observable env. examples: the everyday physical world, the Internet, the card game poker.

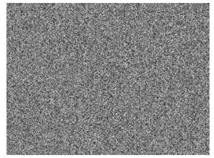
Deterministic vs Stochastic

- Deterministic environment: one in which any action has a single guaranteed effect - there is no uncertainty about the state that will result from performing an action.
- This definition applies *from the point of view of the agent*.
- If the environment is deterministic except for the actions of other agents, the environment is said to be strategic.
- Stochastic environments present greater problems for the agent designer.
- Deterministic env. examples: a crossword puzzle, image analysis.
- Stochastic env. examples: medical diagnosis, the card game poker, the physical world.

Episodic vs Sequential

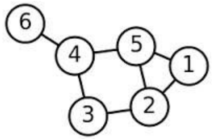
- Episodic environment: one where the performance of an agent is dependent on a number of discrete episodes, with no link between its performance in different scenarios.
- Episodic environments are simpler for agent developers
  - the agent can decide what action to perform based only on the current episode without having to reason about the interactions between this and future episodes.
- In sequential environments the current decision could affect all future decisions.
- Episodic env. examples: a mail sorting system, defect detection on an assembly line.
- Sequential env. examples: poker, a robot that delivers mail and coffee

Static vs Dynamic



- Static environment: one that can be assumed to remain unchanged whilst the agent is deliberating.
- Dynamic environment: one that has other processes operating on it, and hence changes whilst the agent is deliberating.
- Static environments are easier to deal with
  - the agent does not need to keep observing the environment whilst deciding how act, nor need it worry about time elapsing.
- Static env. examples: a crossword puzzle, the card game poker, the game of backgammon.
- Dynamic env. examples: medical diagnosis, the physical world.

Discrete vs Continuous



- Discrete environment: one that contains a fixed, finite number of distinct states.
- The distinction applies to the **state** of the environment, the way in which **time** is handled, the **percepts** and **actions** of the agent.
- Continuous environments provide greater challenges for agent designers.
- Discrete env. examples: a crossword puzzle, the game of chess.
- Continuous env. exmples: image analysis, medical diagnosis.

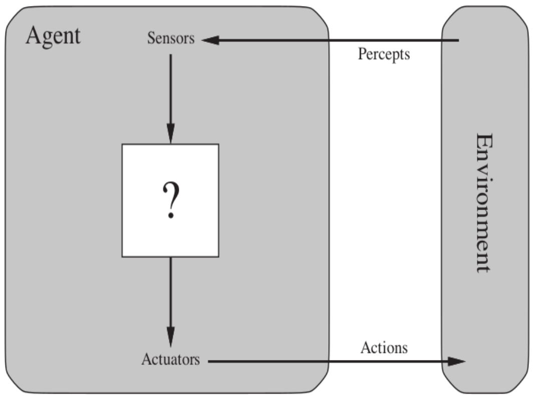
Exercise

Considering each of the previous characteristics, which apply to an automated agent designed to drive a taxi?

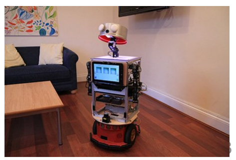
- Fully observable vs partially observable
- Deterministic vs stochastic
- Episodic vs sequential
- Static vs dynamic
- Discrete vs continuous

Agent Design

- 'Agent function'
  - math perspective
  - implemented by an **agent program**
- Agent = architecture + program



- Architecture: some computing device equipped with sensors and actuators
  - e.g., a mobile robot with sensors, a PC, etc.
- Most of AI focuses on the agent program



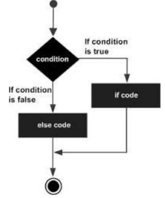
Agent Program Designs

- Four basic kinds of agent program are identified that embody the notions that underpin most intelligent systems (AIAMA):

  1) Simple reflex agents
  2) Model-based reflex agents
  3) Goal-based agents
  4) Utility-based agents

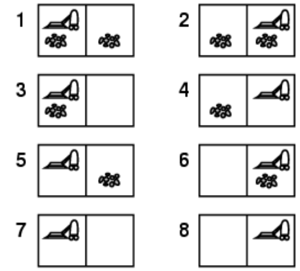
- The ability to learn improves the performance of all these agents.
  - Enables agents to operate in environments of ignorance.
  - Increases performance beyond the limitations of agents' current knowledge.

Simple Reflex Agents



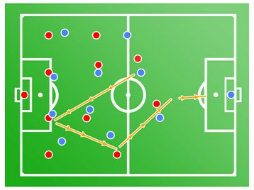
- Simple reflex agents:
  select actions to execute based upon the **current** percept.
  - Do not take the percept history into account.
  - E.g., implemented using condition-action rules.
  - Such agents are simple to implement, but of very limited intelligence.
  - Success of decision making seriously deteriorates with unobservability.

Model-Based Reflex Agents



- Last percept may not identify
  the exact **state of the environment**
- Model-based reflex agents:
  - use a **model** of the environment
  - to keep track of the current **state** of the world:

    state ← Update(state, action, percept, model)
  - Requires knowledge
    - How the world changes independent of the agent's actions.
    - How the world changes due to the agent's actions.



Goal-Based Agents

- Goal-based agents: select actions to achieve **goals**
  (particular desirable states of the environment)
  - Knowledge of the current state does not automatically mean that the agent knows what to do:
    - long sequences of actions to achieve a goal…

      → Search and planning may be required.
  - Goal-based agents more flexible:
    easier to modify than reflex-based agents.
    ("Programming via goals")

Utility-Based Agents

- Utility-based agents: use a utility function to compare the 'desirability' of different states that result from actions.
  - Many actions may satisfy a goal, but which is the most desirable?
- **Utility function**: maps a state (or sequence of states, and/or actions), onto a real number which represents 'usefulness' of the state to the agent.
    - Agent tries to maximise the value of its utility function.
- Enables:
  - Tradeoffs between competing goals.
  - Dealing with uncertainty
    - evaluate goal importance against likelihood of success.
  - Therefore can lead to higher quality behaviour than goal-based agents.

Summary

- Today
  - Intelligent agents
  - Task environments
    - PEAS description
    - Properties of environments
  - Agent program designs

- Next time
  - Search in AI