cis32-ai — lecture # 2 — wed-1-feb-2006

today's topics:

- artificial intelligence and autonomous agents
- brief history of AI

brief history of AI (1943–56)

- Warren McCulloch & Walter Pitts (1943)
  - artificial neural network (ANN):
    - basic physiology of neurons in the brain
    - propositional logic
    - Turing's theory of computation
  - neurons are either "on" or "off"; connections between them are used to "compute" functions
  - ANN proved equivalent to Turing machine
- Donald Hebb (1949) — showed that ANN's could "learn" by changing strengths of connections between neurons
- Alan Turing (1950) — "Computing Machinery and Intelligence"
- Claude Shannon (1950) — first chess playing program
- Marvin Minsky (1951)
  first neural net computer — SNARC

- Dartmouth College (1956)
  term "AI" coined by John McCarthy
  Allen Newell & Herb Simon presented LOGIC THEORIST program

history (1956-70)

- John McCarthy (1958) — invents "LISP"
- McCarthy vs Minksy:
  - McCarthy — representation, reasoning in formal logic
  - Minsky — anti-logic; just make programs work!
- programs written that could:
  plan, learn, play games, prove theorems, *solve problems*.
- major centers established:
  - Marvin Minsky — MIT
  - John McCarthy — Stanford
  - Allen Newell & Herb Simon — CMU
- major feature of the period were *microworlds* — toy problem domains
  example: blocks world, general problem solver (GPS)
  "It'll scale, honest..."

history (1970s)

- 1970's period of recession for AI
  (Lighthill report in UK)
- 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*

history (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*: mapping of general, "first principles" to special forms
- rise of natural language processing

history: knowledge-based systems

- also called "weak methods"
- expert systems success stories:
  - DENDRAL — interpreting mass spectrometers, used by chemists
  - MYCIN — blood diseases in humans
- expert systems emphasised *knowledge representation*: rules, frames, semantic nets
- problems:
  - the knowledge elicitation bottleneck
  - marrying expert system and traditional software
  - breaking into the mainstream

history (1990s)

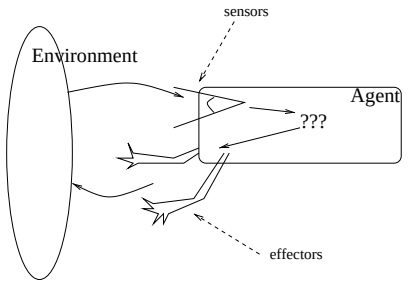
- "the AI winter"
- most companies set up to commercialize expert systems technology went bust
- 1990s: emphasis on understanding the interaction between *agents* and *environments*
- AI as *component*, rather than as end in itself
- rise of the "intelligent agent"

differing views of AI

- connectionist: Rumelhart, McClelland
- symbolic: Newell and Simon
- logic: McCarthy

intelligent agents

- an *agent* is a system that is *situated* in an environment, and which is capable of *perceiving* its environment and *acting* in it to satisfy its designobjectives.



- human "agent":
  - *environment*: physical world
  - *sensors*: eyes, ears, ...
  - *effectors*: hands, legs, ...
- software agent:
  - *environment*: e.g., UNIX operating system
  - *sensors*: `ls`, `ps`, ...
  - *effectors*: `rm`, `chmod`, ...
- internet agent:
  - *environment*: the Internet
  - *sensors*: `http` requests
  - *effectors*: `http` commands
- embodied (robotic) agent:
  - *environment*: physical world
  - *sensors*: light meters, bumpers, thermometers, ...
  - *effectors*: motors attached to wheels, treads, legs, grippers, ...

What to do?

Those who do not reason
Perish in the act.
Those who do not act
perish for that reason
(W H Auden)

- The key problem we have is *knowing the right thing to do*.
- Knowing what to do can *in principle* be easy: consider all the alternatives, and choose the "best".
- But Auden's quote! In any time-constrained domain, we have to make a decision *in time for that decision to be useful*!
- A tradeoff.

- need to know *how* and *when* to evaluate success
- how:
  - an objective performance measure
  - application specific
- when:
  - in discrete *episodes*, or over long periods?
- don't confuse *omniscience* with rationality
- real agents don't know enough to always make the best choice.
  (We often fall into this trap when making judgements about history.)
- Rationality concerned with *expected success* given *information available*.

- *Ideal rational agent*:
  For each percept sequence, an ideal rational agent will act to maximise its expected performance measure, on the basis of information provided by percept sequence plus any information built in to agent.
- Note that this does not preclude performing actions to *find things out*.
- More precisely, we can view an agent as a function:

$$f : P^* \rightarrow A$$

  from sequences of percepts P to actions A.

- For example, a quadratic agent:

Percept	Action
0	0
1	1
2	4
3	9
4	16
...	...

- This table can be viewed as a *specification* of the agent.
- We don't have to *implement* agent as table lookup:

```
int agent(int n)
{
     return n * n;
}
```


- *Autonomy* a crucial concern for agents.
  Means behaviour is based on *own* experience. Implies *learning*, or *adaptation*.

Structure of Agents

- Two components:
  - *program*: the thing which defines the mapping from percept sequences to actions;
  - *architecture*: the "shell" into which the agent program fits.

  Agent = program + architecture.
- An appropriate architecture can make design of programs much easier.

Classifying Environments

- The PAGE approach:
  - percepts;
  - actions;
  - goals;
  - environment.
- Example: Refinery controller.
  - percepts: temp, pressure readings;
  - actions: open, close valves, switch on, off heaters...;
  - goals: maximise purity, yield, safety;
  - environment: refinery.

- Example: Medical diagnosis system.
  - percepts: symptoms, findings, patient answers;
  - actions: questions, tests, treatments;
  - goals: healthy patient, minimise costs;
  - environment: patient, hospital.
- Example: Email manager.
  - percepts: email arrived, headers, content of email;
  - actions: delete email, reorder email, obtain user attention;
  - goals: present important email first; hide junk email;
  - environment: mail reader, operating system.

Accessible *vs* inaccessible

An accessible environment is one in which the agent can obtain complete, accurate, up-to-date information about the environment's state.

Most moderately complex environments (including, for example, the everyday physical world and the Internet) are inaccessible.

The more accessible an environment is, the simpler it is to build agents to operate in it.

Deterministic *vs* non-deterministic

A deterministic environment is one in which any action has a single guaranteed effect — there is no uncertainty about the state that will result from performing an action.

The physical world can to all intents and purposes be regarded as non-deterministic.

Non-deterministic environments present greater problems for the agent designer.

Episodic *vs* non-episodic

In an episodic environment, the performance of an agent is dependent on a number of discrete episodes, with no link between the performance of an agent in different scenarios.

An example of an episodic environment would be a mail sorting system.

Episodic environments are simpler from the agent developer's perspective because the agent can decide what action to perform based only on the current episode — it need not reason about the interactions between this and future episodes.

Static *vs* dynamic

A static environment is one that can be assumed to remain unchanged except by the performance of actions by the agent.

A dynamic environment is one that has other processes operating on it, and which hence changes in ways beyond the agent's control. The physical world is a highly dynamic environment.

Discrete *vs* continuous

An environment is discrete if there are a fixed, finite number of actions and percepts in it.

Russell and Norvig give a chess game as an example of a discrete environment, and taxi driving as an example of a continuous one.

Summary

- This lecture has looked at:
  - The history of AI
  - The notion of intelligent agents
  - A classification of agent environments.
- Broadly speaking, the rest course will cover the major techniques of AI, with special reference to agents.
- The techniques we'll look at will start with those applicable to simple environments and move towards those suitable for more complex environments.