Intelligent Agents

Iran R. Roman

(Thanks to Julia Ive, Huy Phan and Simon Dixon)

School of Electronic Engineering and Computer Science

Queen Mary University of London

Queen Mary
University of London

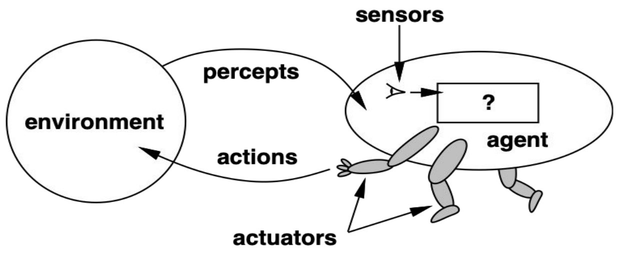
Outline

- Agent Definition
- Rationality
- Structure of an Agent
- Types of Agents
- Environments

What is an Agent?

- An ***agent*** is anything that *perceives* its *environment* through *sensors, reasons about its goals,* and *acts* upon that environment through *actuators*
- A goal of AI is to design agents that can do a good job of acting on their environment, i.e., to build rational agents
- Examples
  - Human
  - Thermostat
  - Robot
  - Softbot

Diagram of an agent



But why do we need Agents?

Examples of Agents

- **Human**
  - Environment: classroom
  - Sensors: eyes, ears
  - Actuators: hands, legs, vocal tract
- **Thermostat**
  - Environment: building
  - Sensors: temperature sensor
  - Actuators: turn heating on/off

Examples of Agents

- **Robot**
  - Environment: classroom
  - Sensors: cameras, infrared range finders
  - Actuators: motors
- **Softbot** (Virtual Assistant)
  - Environment: interface, network
  - Sensors: microphone, text input, network packages
  - Actuators: speaker, information display, sending network packages

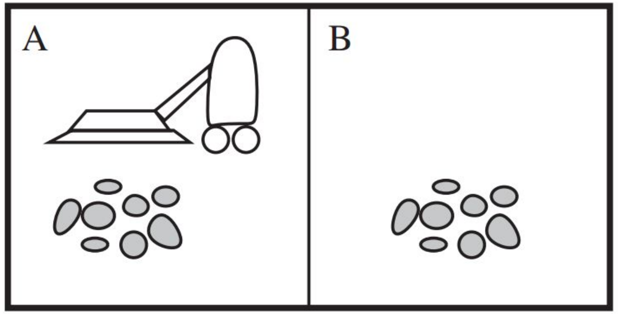
Our Model of an Agent

- Percept
    - The agent's perceptual inputs at any given instant
- Percept sequence
    - The complete history of everything the agent has ever perceived
- Agent function
    - Maps from percept sequence to an action:

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

- Agent program
    - The implementation of an agent function

Example: Vacuum-Cleaner World



- Percepts:
  - Location and contents, e.g. **[A, Dirty]**
- Actions:
  - **Left, Right, Suck Dust, Do nothing**

- Simple agent tabulation function:

Percept sequence	Action
[A, Clean]	Right
[A, Dirty]	Suck Dust
[B, Clean]	Left
[B, Dirty]	Suck Dust
[A, Clean], [A, Clean]	Right
[A, Clean], [A, Dirty]	Suck Dust
…	…
[A, Clean], [A, Clean], [A, Clean]	Right
[A, Clean], [A, Clean], [A, Dirty]	Suck Dust
…	…

Outline

- Agent Definition
- Rationality
- Structure of an Agent
- Types of Agents
- Environments

Rational Agents

- A *rational agent* is one that does the *right thing in context*
    - The right action should cause the agent to be most successful
- Success is evaluated with respect to an objective *performance measure*, designed according to what we want in the environment
    - Not according to how one thinks the agent should behave
- But agents do not know everything about the world
    - What is rational at any time depends on the agent’s background knowledge, percepts, and available actions
- The best action to maximise performance may be to gather more information (first look – then act)

Rationality

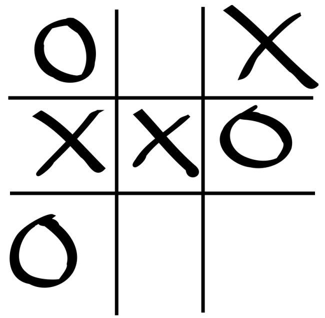
- Rationality maximises expected performance
- Rationality ≠ omniscience: perfection maximises actual performance, we do not know the actual outcome of actions in advance
- What is rational depends on:
    - The performance measure that defines the criterion of success
    - The agent’s prior knowledge of the environment
    - The actions that the agent can perform
    - The agent’s percept sequence to date

Vacuum-cleaner Agent

- Performance measure
    - Awards one point for each clean square at each time step, over 10000 steps (lifetime)
- Prior knowledge about the environment
    - The geography of the environment (2 squares)
    - The ***effect*** of the actions
- Actions that can perform
    - Left, Right, Suck Dust, Do nothing
- Percept sequences
    - Where is the agent?
    - Whether the location contains dirt?

From Percept Sequence to Action

- Agent behaviour could be defined as a simple mapping from every possible percept sequence to an action
    - For any realistic agent that will probably be an infinite list and the agent would only be able to do what it was explicitly told
- For example, a tic-tac-toe agent has to choose a move, given a board state
    - How large would a complete lookup table be to store a move for all possible states ?
        - 3 actions^9 positions = 19683
    - How else could the mapping to actions be coded?
    - How much space would that take?



Square Root Agent

- Square root agent: given a percept *x*, display the positive number *z*, such that *z* = √*x*
- The mapping could be a lookup table or a function:

x	z
2	1.414213562
1.9	1.378404875
1.8	1.341640786
1.7	1.303840481
1.6	1.264911064
1.5	1.224744871
1.4	1.183215957
1.3	1.140175425
1.2	1.095445115
1.1	1.048808848

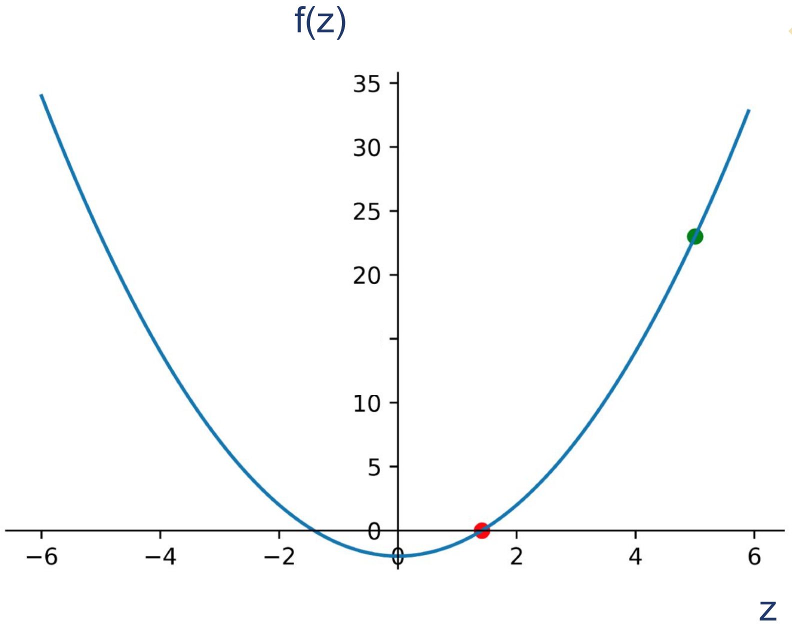
```
function sqrt(x)
        $z \leftarrow 5.0$ /* initial guess */
        repeat until $|z^2 - x| < 10^{-5}$
                $z \leftarrow z - (z^2 - x)/(2z)$
        end
        return z
```

- Solve the equation for x=2

$$f(z) = |z^2 - 2| < 10^{-5}$$

- Function implementing the Numerical (Newton's) method

```
function sqrt(x)
        $z \leftarrow 5.0$ /* initial guess */
        repeat until $|z^2 - x| < 10^{-5}$
                $z \leftarrow z - (z^2 - x)/(2z)$
        end
        return z
```



https://medium.com/@ruhayel/an-intuitive-and-physical-approach-to-newtons-method-86a0bd812ec3

- Solve the equation for x=2

$$f(z) = |z^2 - 2| < 10^{-5}$$

- Numerical (Newton's) method
- Derivative f(z) gives the slope of a function at a given point — ratio of change in f(z) to the change in z, $\Delta f(z)/\Delta z$:

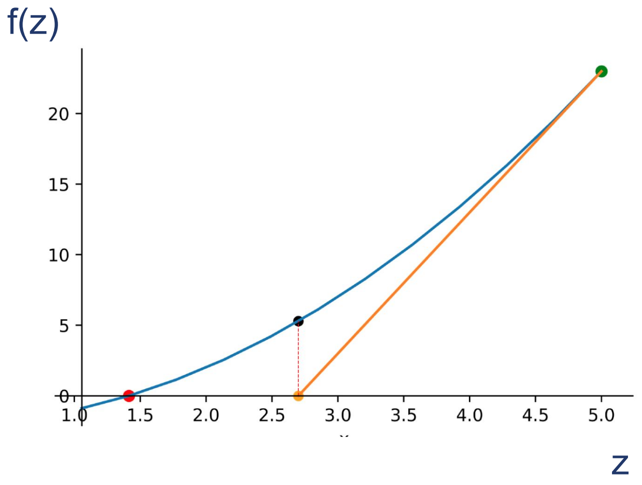
$$f'(z) = \frac{f(z)_1 - \overset{=0}{\cancel{f(z)_2}}}{z_1 - z_2} = 2z_1$$

- Keep solving the derivatives until we find a value that satisfies the stopping criteria. For f(z)2 = 0:

$$z_2 = z_1 - \frac{|z_1^2 - 2|}{2z_1}$$

$$z_2 = 5 - |25-2|/(2 * 5) = 2.7$$

$$f(2) = |2.7^2 - 2| = 5.29$$



```
function sqrt(x)
    z ← 5.0 /* initial guess */
    repeat until |z^2 - x| < 10^-5
        z ← z - (z^2-x)/(2z)
    end
    return z
```

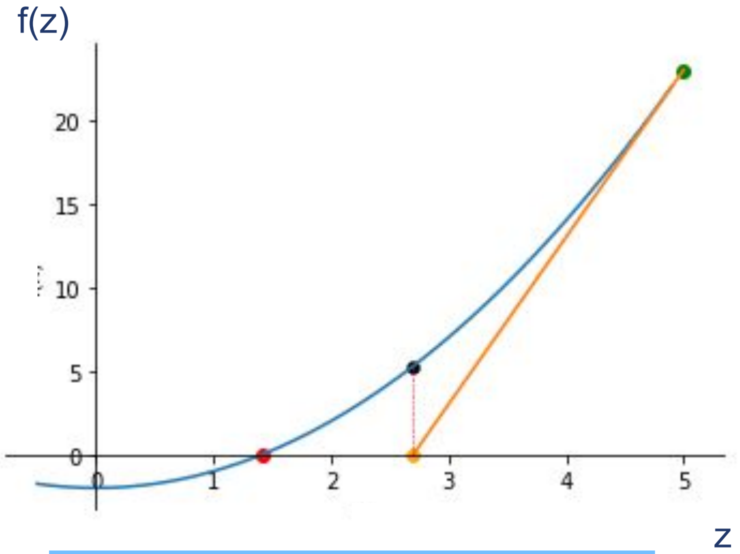
$f(2) = |1.414213562^2 - 2| = 1.0552e\text{-}09$

- Solve the equation for x=2

$$f(z) = |z^2 - 2| < 10^{-5}$$

- Numerical (Newton's) method
- Keep solving the derivatives until we find a value that satisfies the stopping criteria:

$$z_2 = z_1 - \frac{|z_1{}^2 - 2|}{2z_1}$$



```
function sqrt(x)
    z ← 5.0 /* initial guess */
    repeat until $|z^2 - x| < 10^{-5}$
        z ← z - $(z^2 - x)/(2z)$
    end
    return z
```

Autonomy

- Agent's behaviour is based on its own experience and its built-in knowledge
- An agent is autonomous to the extent that its behaviour is determined by its own experience
- Complete autonomy from the start is too difficult
    - Agent's designer must give guidance in terms of some initial knowledge and the ability to learn and/or reason as it operates in its environment
- But agents must be able to adapt when something unexpected happens - flexibility
    - Otherwise the agent's behaviour may *appear* intelligent but is actually *inflexible*

When an Agent is not Autonomous

- **Example**
  - After a dung beetle digs its nest and lays its eggs, it fetches a ball of dung from a nearby heap to plug the entrance
  - If the ball of dung is removed en route, the beetle carries on and mimes plugging the nest with the nonexistent dung ball, never noticing that it is missing

- A truly autonomous intelligent agent should be able to operate successfully in a wide variety of environments, given sufficient time to adapt

Do we have the prerequisites to pass the Turing test?

- To convince the interrogator of its intelligence, the computer must
  - Understand and generate language
  - Know about the world
  - Reason about the world
  - Learn about the dialogue and the interrogator
  - Combine all this knowledge and reasoning instantaneously

Do we have the prerequisites to pass the Turing test?

- To convince the interrogator of its intelligence, the computer must
  - Understand and generate language – prior knowledge
  - Know about the world – prior knowledge
  - Reason about the world – prior knowledge
  - Learn about the dialogue and the interrogator - autonomy
  - Combine all this knowledge and reasoning instantaneously - autonomy

Outline

- Agent Definition
- Rationality
- Structure of an Agent
- Types of Agents
- Environments

Agent Description Schema: PEAS

- Designing an agent requires a good understanding of its
  - **P**erformance Measure: evaluates how well an agent does what it is designed to do
  - **E**nvironment: the domain in which it operates and with which it interacts
  - **A**ctuators: the means by which an agent acts upon its environment
  - **S**ensors: the means by which an agent perceives its environment

An Agent Description Schema: PEAS

Agent	Performance measure	Environment	Actuators	Sensors
Taxi Driver	Safe, fast, legal, comfortable, maximise profit	Roads, other road users, customers	Steering, accelerator, brake, signals, horn	Cameras, sonar, GPS, speedometer, fuel gauge, odometer, etc.
Interactive English Tutor				
Hand Washing Promotion Agent				

An Agent Description Schema: PEAS

Agent	Performance measure	Environment	Actuators	Sensors
Taxi Driver	Safe, fast, legal, comfortable, maximise profit	Roads, other road users, customers	Steering, accelerator, brake, signals, horn	Cameras, sonar, GPS, speedometer, fuel gauge, odometer, etc.
Interactive English Tutor	Student Feedback, Student's score on test	Students, Testing agency	Display of exercise, Suggestions to students, Correction of errors	Keyboard entry
Hand Washing Promotion Agent	Less people sick, less hours on manual inspections	Washing facilities, people washing hands	Signal	Camera

Structure of an Intelligent Agent

AGENT = ARCHITECTURE + PROGRAM

- The agent *program* is the agent function
- The program runs on a computing device, the *architecture*, which
  - Makes percepts from the *sensors* available to the program
  - Runs the program
  - Feeds the program's actions to the *actuators*
- We focus on agent programs in this module, and not on sensing or physical action (not Robotics)

Agent Programs

- All agent programs have the same skeleton structure
    - Take current percept as input
    - Generate an action
    - Store the percept and action in memory (if necessary and feasible)
    - Repeat
- The goal and performance measure are not part of the skeleton
    - Agent may not need to know explicitly how it is being judged

Agent Programs

```
function Skeleton-Agent( percept ) returns action
    static: memory, the agent's memory of the world

    memory ← Update-Memory( memory, percept )
    action ← Choose-Best-Action( memory )
    memory ← Update-Memory( memory, action )
    return action
```

Why not just look up the answers?

- Keep the entire sequence in memory
  - Use it as an index into a table
  - Simply look up actions

```
function Lookup-Agent( percept ) returns action
    static: percepts, a sequence, initially empty
        table, full, indexed by percept sequence, initially fully specified
    percepts ← Append( percepts, percept )
    action ← Lookup( percepts, table )
    return action
```

Why not just look up the answers?

- This approach will fail because
  - The look-up table for will be too large (for chess, ~35^{100} entries)
  - Designer has to build whole of table in advance
  - No autonomy, so no flexibility
  - Even a learning agent would take forever to learn a real-world table

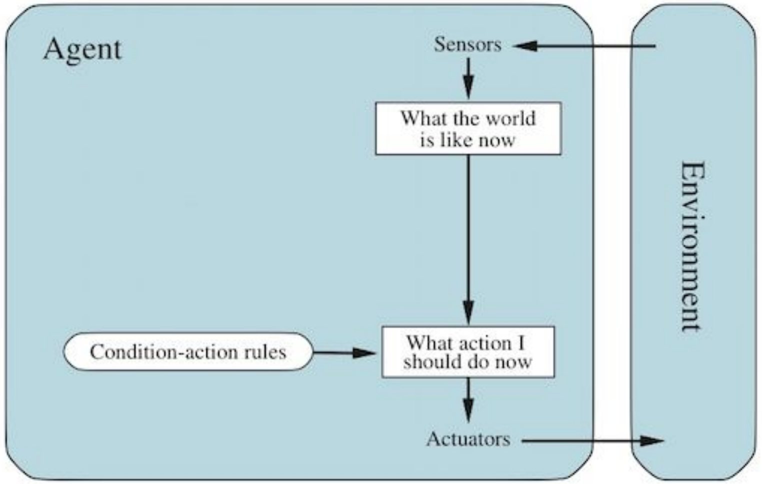
Different Types of Agents

- Different types of agent are useful in different circumstances
  - It is generally best to use the simplest possible in context
- **Types of agents**
  - Simple reflex
  - Agents with state
  - Goal-based agents
  - Utility-based agents

Simple Reflex Agents

```
function Simple-Reflex-Agent( percept )
returns action
    static: rules, a set of productions


    state ← Interpret-Input( percept )
    rule ← Rule-Match( state, rules )
    action ← Rule-Action( rule )
    return action
```



Simple Reflex Agents

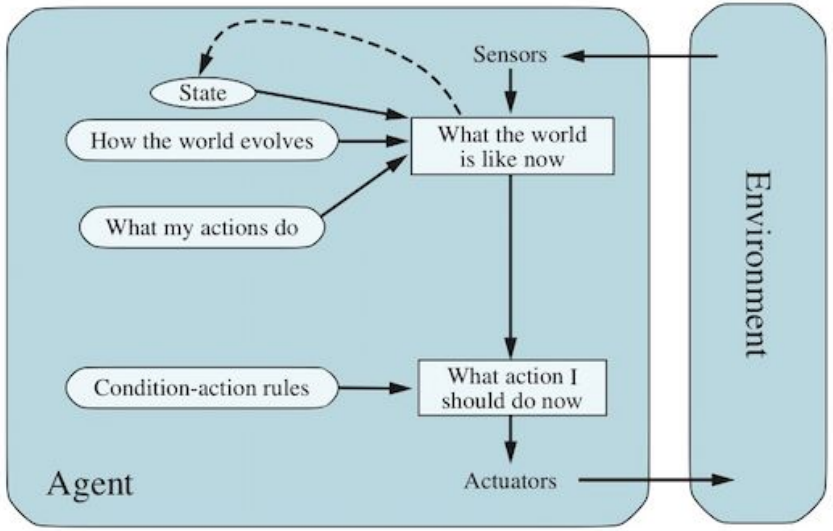
- Some actions are more or less automatic
    - e.g., outdoor smart light bulb with darkness detector
- Such reflex rules are sometimes called
    - Condition-action rules
    - Production rules
    - If-then rules
- Some processing needed to test conditions
- Efficient, but narrow

Reflex Agents with State

- Sometimes an agent needs more than just its current percept and must maintain an internal state
- Example
    - An agent that is looking for its car keys needs to remember which drawers it has already looked in, otherwise it might get stuck in a loop: open drawer, look for keys, close drawer

Reflex Agents with State

```
function Reflex-Agent-State( percept )
returns action
    static: rules, a set of productions
        state, a description of the current world
    state ← Update-State( state, percept )
    rule ← Rule-Match( state, rules )
    action ← Rule-Action( rule )
    state ← Update-State( state, action )
    return action
```



Reflex Agents with State

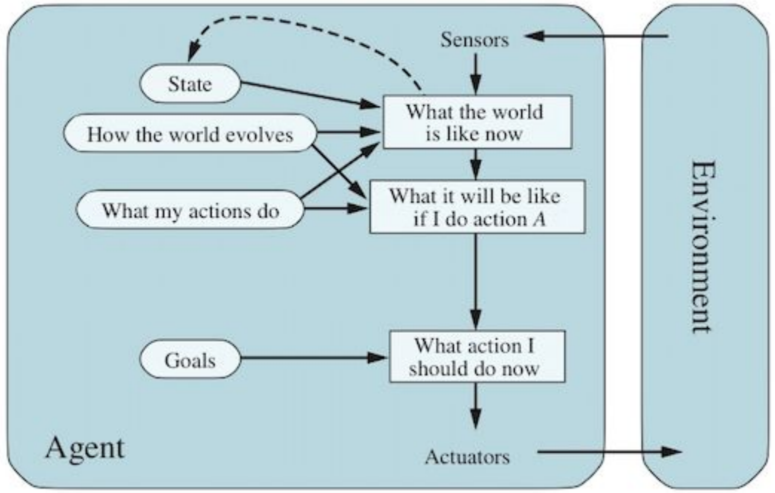
- The Update-State function must
    - Interpret new percepts in light of existing knowledge about state
    - Encode the information about how the world evolves independently of the agent
        - E.g. tracking a moving object which is sometimes occluded
    - Know about what the agent’s actions do to the state of the world
        - Particularly if the effects are not directly perceivable
    - Combine all these into a new internal state description

Goal-based Agents

- Knowing the state of the environment is not always enough
    - The correct action may depend on the *goals* of the agent
    - The same agent may have different goals at different times
- An agent program combines goal information with the information about the results of possible actions to choose actions that achieve the goal
    - Sometimes simple
        - when the goal is satisfied by a single action
    - Sometimes complex
        - when the goal requires a sequence of actions
    - Search and planning are sub-fields of AI devoted to finding action sequences that achieve the agent's goals

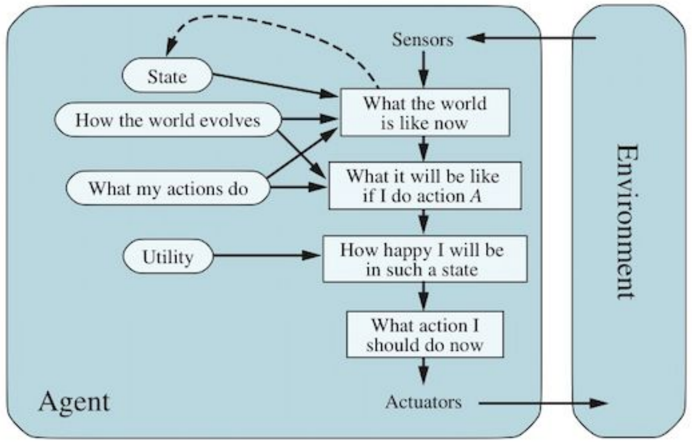
Goal-based Agents

- Decision making is more complex with goals
    - Consideration of the future: does X bring me closer to achieving my goal?
- Although they may be less efficient, goal-based agents are more flexible
    - Taxi-driver agent with different destination goals behaves in different ways for each possible percept sequence
- Goals alone may not be sufficient to generate high-quality behaviour
    - Many action sequences may achieve the goal, but some are better than others


Sensors
State
What the world is like now
How the world evolves
What my actions do
What it will be like if I do action A
Goals
What action I should do now
Agent
Actuators
Environment

Utility-based Agents

- Goal satisfaction provides a crude distinction between good states and bad states
- Agent's internal measure of preference over world states is known as *utility*
- If in agreement to the external performance measure maximising utility means rational according to the performance measure
- Utility (usually) maps states onto real numbers so they are comparable



Sensors
State
What the world is like now
How the world evolves
What it will be like if I do action A
What my actions do
Utility
How happy I will be in such a state
What action I should do now
Agent
Actuators
Environment

Utility-based Agents

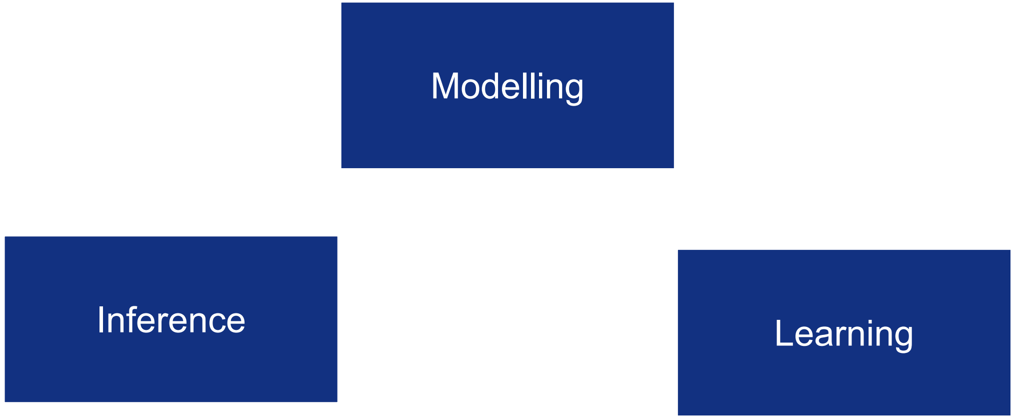
- Utility is needed when
    - There are conflicting goals, e.g., speed versus safety
    - There are several means of reaching a goal
    - Several goals exist but cannot be reached with certainty: utility provides a way in which the likelihood of success can be weighed against the importance of the goals

Learning Agents

- After an agent is programmed, can it work immediately?
  - No, it still needs teaching
- In AI
  - Teaching an agent by giving a set of examples
  - Test it by using another set of examples

Learning Agents

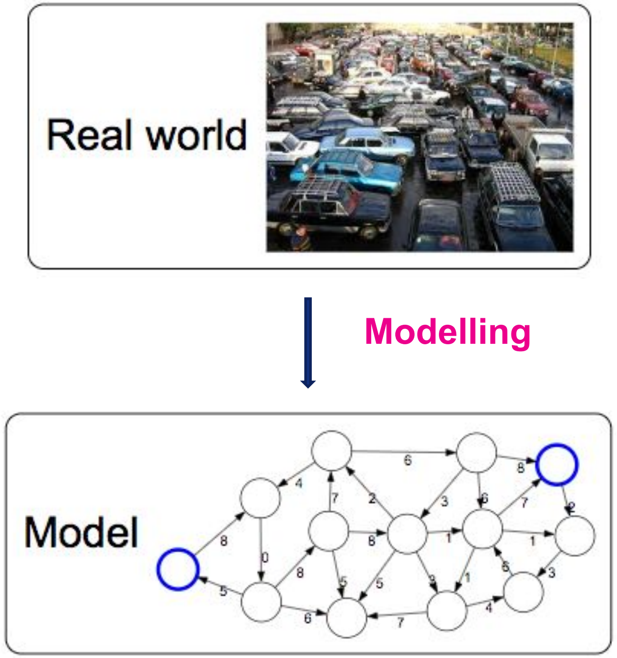
- New paradigm: Modelling – Inference – Learning



(Idea and example taken from Percy Liang’s teachings)

Learning Agents

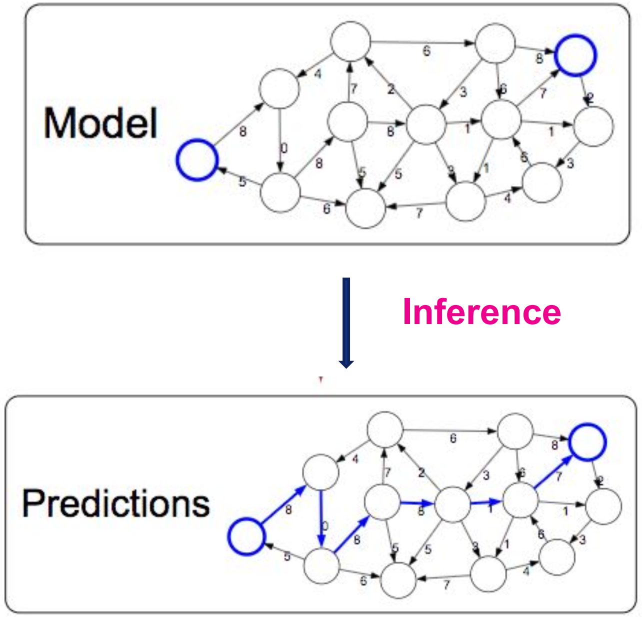
- New paradigm: **Modelling**



(illustration credit: Yu-Xiang Wang)

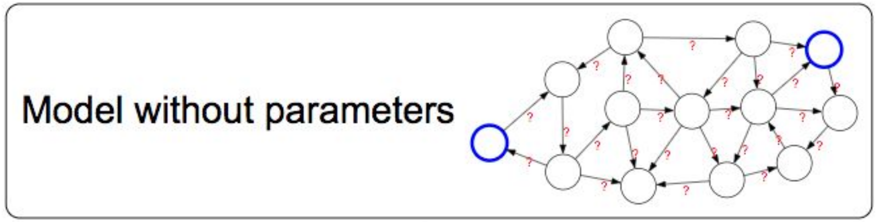
Learning Agents

- New paradigm: **Inference**



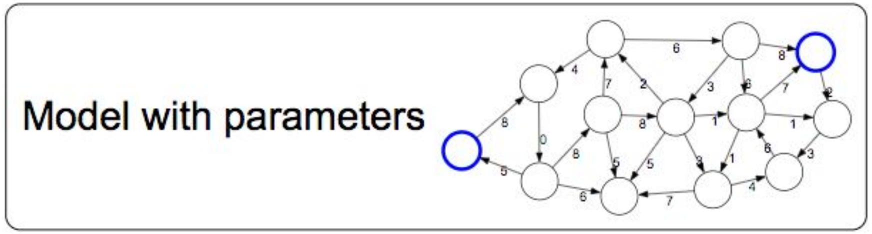
(illustration credit: Yu-Xiang Wang)

Learning Agents



+data

Learning



(illustration credit: Yu-Xiang Wang)

Outline

Environments

- Environments have different properties, which determine what kinds of agents can work in them

Fully Observable	Partially Observable
Deterministic	Stochastic
Episodic	Sequential
Static	Dynamic
Discrete	Continuous
Single Agent	Multiple Agent

Classifying Environments

- **Fully observable (vs partially observable)**
  - Sensors give complete state of the environment
  - Agent does not need to keep track of the world state
  - Most AI environments are only partially observable
- **Deterministic (vs stochastic)**
  - Next state completely determined by current state and agent action
  - In stochastic environments next state is uncertain (though a probabilistic model may be available)

Classifying Environments

- **Episodic (vs sequential)**
  - Agent’s experiences divided into episodes: agent perceives then acts
  - Quality of action depends only on the episode itself, subsequent episodes are independent of previous ones
- **Dynamic (vs static)**
  - Environment may change while agent is deliberating
  - *Semi-dynamic*: agent’s *utility* scores changes with passage of time though the environment is static
  - In a static environment an agent does not need to look at the world while deciding on an action, nor does it need to worry about the passage of time

Classifying Environments

- **Discrete (vs continuous)**
  - Limited number of distinct, clearly defined percepts and actions
  - Playing chess is discrete: there is a limited number of moves
  - Driving a taxi is continuous: speed and location vary
- **Multi-agent (vs single-agent)**
  - Competitive: maximising agent A’s performance measure implies minimising agent B’s
  - Cooperative: performance measures of both agents A and B are maximised by the same action(s)
- The real world is partially observable, stochastic, sequential, dynamic, continuous and multi-agent

Exercise: Environment Types

Task	Observable?	Deterministic?	Episodic?	Static?	Discrete?	Multi-agent?
Chess with a clock						
Poker player						
Taxi Driver						

Exercise: Environment Types

Task	Observable?	Deterministic?	Episodic?	Static?	Discrete?	Multi-agent?
Chess with a clock	Fully	Deterministic	Sequential	Semi-dynamic	Discrete	Multi-agent
Poker player	Partially	Stochastic	Sequential	Static	Discrete	Multi-agent
Taxi Driver	Partially	Stochastic	Sequential	Dynamic	Continuous	Multi-agent

Discussion

- Intelligent agent perceives its environment, takes it into consideration together with the prior knowledge to take actions autonomously in order to maximise its performance measure and achieve goals
- Agents base their actions on a mapping from states to actions – **agent function** – focus of this module
- **Goal-based agents** take decisions focusing on achieving their **goal**. They choose among multiple possibilities, selecting the ones which help to reach the goal. They usually require **Search** and Planning
- We will also look into Learning, Knowledge Representation, Planning and Handling Uncertainty

Past Example Question

- Q1b (2016): *The PEAS description (Performance measure, Environment, Actuators, Sensors) can be used to describe the task environment of an agent. Define each of the four terms and illustrate them with reference to a chess-playing agent.* [8 marks]

Answer A: Performance measure is a function which produces a numeric value describing how successful (close to goal) an agent's action is.

Environment decides the nature of the agent's task. It can be fully or partially observable, deterministic or stochastic, discrete or continuous, episodic or sequential, adversarial or not, dynamic or static.

Actuators are possible actions the agent can do.

Sensors are the way the agent perceives the world.

Past Example Question

- Q1b (2016): *The PEAS description (Performance measure, Environment, Actuators, Sensors) can be used to describe the task environment of an agent. Define each of the four terms and illustrate them with reference to a chess-playing agent.* [8 marks]

Answer A (continued): Chess agent: Performance measure = win gets 1 point, lose gets -1 point, draw gets 0 point.

Environment: fully observable, adversarial, static, discrete, sequential, deterministic.

Actuators: in the agent's turn, move one of the agent's pieces according to chess rules.

Sensors: the make-up of the board.

Past Example Question

- Q1b (2016): *The PEAS description (Performance measure, Environment, Actuators, Sensors) can be used to describe the task environment of an agent. Define each of the four terms and illustrate them with reference to a chess-playing agent.* [8 marks]

Answer B: Performance measure: goals that should be achieved by the agent and ways to measure them.

Environment: context in which the agent is inserted.

Actuators: medium of interaction between the agent and its environment.

Sensors: mechanisms through which the agent perceives its environment.

Past Example Question

- Q1b (2016): *The PEAS description (Performance measure, Environment, Actuators, Sensors) can be used to describe the task environment of an agent. Define each of the four terms and illustrate them with reference to a chess-playing agent.* [8 marks]

  Answer B (continued): Chess-playing agent

  P.M.: Number of opponent's pieces captured, number of pieces in the game, check-mate, value of pieces in the game, check attempts

  E: online game platform, computer, board, pieces

  A: moving piece of interface, writing new position on screen, interface

  S: time, opponent's movement, board knowledge

Thank you

Queen Mary
University of London