**Lesson 0**

Agents

An **agent** is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuator.

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

*Rational behavior: doing the right thing*

*- The right thing: that which is expected to maximize goal achievement, given the available information*

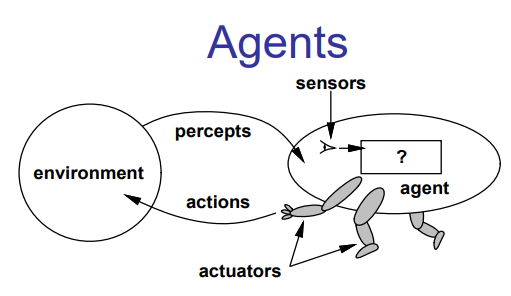
*- Doesn't necessarily involve thinking*

An agent is **autonomous** if its behaviour is determined by its

own experience (ability to learn and adapt).

To design a rational agent, we must specify the task environment: **PEAS** (Performance measure, Environment, Actuators,

Sensors)



Agent – perceives the environment through sensors and

acts on it through actuators

Percept – agent’s perceptual input (the basis for its

actions)

Percept Sequence – complete history of what has been

perceived.

**Lesson 0**

Environment types

<http://www.cs.stir.ac.uk/courses/ITNP4A/lectures/2%20-%20Environments.pdf>

**1. Fully observable (vs. partially observable)**

– Fully observable gives access to complete

state of the environment

– Complete state means aspects relevant to

action choice

– global vs local dirt sensor

**2. Deterministic (vs. stochastic)**

- If the environment is deterministic except for the actions of other agents, then the environment is strategic.

**3. Episodic (vs. sequential)**

– Episodic the agent’s experience divided into atomic episodes

– Next episode not dependent on actions taken in previous episode. E.g., assembly line

– Sequential – current action may affect future actions. E.g., playing chess, taxi

– short-term actions have long-term effects

– must think ahead in choosing an action

**4. Static (vs. dynamic)**

- the environment is semi-dynamic if the environment itself does

not change with the passage of time but the agent’s performance score does

– does environment change while agent is deliberating?

– Static – crossword puzzle

– Dynamic – taxi driver

**5. Discrete (vs. continuous)**

Can refer to

– the state of the environment (chess has finite number of discrete states)

– the way time is handled (taxi driving continuous – speed and location of taxi sweep through range of continuous values)

– percepts and actions (taxi driving continuous

– steering angles)

**6. Single agent (vs. multiagent)**

Single Agent vs Multi-agent

- An agent operating by itself in an environment is single agent

- Multi agent is when other agents are present

- A strict definition of another agent is anything that changes from step to step. A stronger definition is that it must sense and act

- Competitive or co-operative Multi-agent environments

- Human users are an example of another agent in a system

– Single Agent – crossword puzzle

– Multi-agent – chess, taxi driving? (are other drivers best described as maximizing a performance element?)

– Multi-agent means other agents may be competitive or cooperative and may require communication

– Multi-agent may need communication

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | Chess with a clock | Chess without a clock | Taxi driving |
| Fully observable | Yes | Yes | No |
| Deterministic | Strategic | Strategic | No |
| Episodic | No | No | No |
| Static | Semi | Yes | No |
| Discrete | Yes | Yes | No |
| Single agent | No | No | No |

The environment type largely determines the agent design

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

**Agent Programs**

Need to develop agents – programs that take the current percept as input from the sensors and return an action to the actuators

The key challenge for AI is to find out how to write programs that, to the extent possible, produce rational behavior from a small amount of code.

**Lesson 0**

Agent types

**Look Up Table (Dr.)**

- Benefits: Easy to implement

- Drawbacks: Huge table, Take a long time to build the table,

No autonomy, Even with learning, need a long time to learn

the table entries

**Simple reflex agents:** based on current percept ignoring percept history

- Selection based on condition-action rules.

- Advantage: Simplicity, requires only limited resources.

- Drawback: It only works if the environment is fully observable

**Reflex agents with state (Model-based):** Agent uses

model of the world around it to keep track of the parts of

the worlds it can not always see

**Goal-based agents:** Knowing about the current state of the

environment is not always enough to decide what to do

- Goal information can ease the action selection process.

- Goal-based selection can be straightforward or can involve

planning

**utility-based agents:** Utility-related considerations can ease

the selection of optimal action sequences.

- Utility function: ✓Maps state (sequence of states) into a real

number ✓Resolves contradictions through trade-offs ✓Resolves

uncertainty through measure for likelihood of success

**Learning Agents**

All these can be turned into learning agents

**Lesson 0**

Search

A strategy is defined by picking the order of node expansion

- Strategies are evaluated along the following dimensions:

• Completeness: Does it always find a solution if one exists?

• Time complexity: Number of nodes generated/expanded

• Space complexity: Maximum number of nodes in memory

• Optimality: Does it always find a least-cost solution?

Time and space complexity are measured in terms of (Dr.)

• b: Maximum branching factor of the search tree

• d: Depth of the least-cost solution

• m: Maximum depth of the state space (may be ∞)

Search Types:

• Uninformed: The agent has no information about the underlying problem other than its definition. e.g. Breadth-first,

Uniform-cost, Depth-first, Depth-limited, Iterative deepening

• Informed: The agent have some idea of where to look for

solutions

Tree search algorithms: offline, simulated exploration of state

space by generating successors of already-explored states

**Breadth-first search (BFS):** Expand shallowest unexpanded node.

It exploits state description to estimate how promising each search node is

An evaluation function f maps each search node N to positive real number f(N)

Traditionally, the smaller f(N), the more promising N

**Uniform-cost search (UCS):** Expand least-cost unexpanded node.

• Depth-first search (DFS): Expand deepest unexpanded node

• Depth-limited search (DLS): depth-first search with depth limit

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Criterion | Complete | Time | Space | Optimal? |
| BFS | Yes, if is finite |  |  | Yes, if cost = 1 per step |
| UCS | Yes, if step cost |  |  | Yes, nodes expanded in increasing order of |
| DFS | No, fails in infinite-depth spaces |  |  | No |
| DLS | No |  |  | No |
| IDS | Yes |  |  | Yes |

**Lesson 0**

Informed Search

**Best-first search:** Expand most desirable unexpanded node

Special cases: greedy search, A\* search

**Greedy best-first search:** expands the node that appears to

be closest to goal

**A∗ Search:** avoid expanding paths that are already expensive

Evaluation function

= cost so far to reach

= heuristic, estimated cost to goal from

= estimated total cost of path through to goal

search uses an admissible heuristic

i.e., where is the true cost from

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Criterion | Complete | Time | Space | Optimal? |
| Greedy | No, can get stuck in loops | ) | ) | No |
| A\* | Yes | exp | All nodes | Yes |

**Lesson 0**

Well-define problems

<https://www.cpp.edu/~ftang/courses/CS420/notes/uninformed%20search.pdf>

<https://www.pearsonhighered.com/assets/samplechapter/0/1/3/6/0136042597.pdf>

A well-defined problem can be described by:

|  |  |
| --- | --- |
| **A start or initial state** | initial statethat the agent starts in |
| **Actions** | A description of the possibleactionsavailable to the agent. |
| **Transition model** | This is specified by a function . A transition model is a description of what each action does. A successor is any state reachable from a given state by a single action. |
| **Path cost** | function that assigns a numeric cost to a path. Cost of a path is the sum of costs of individual actions along the path |
| **Goal test** | test to determine if at goal state |