Columbia CS188 - Artificial Intelligence

Marco Filippone

December 8, 2020

1 Introduction

Artificial intelligence is the study and design of intelligence agents where an intelligent agent is a system that perceives it's environment and takes actions that maximizes its chances of success.

2 Rational Agents

Rational Agent For each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has.

When we define a rational agent, we group these properties under PEAS:

- Performance
- Environment
- Actuators
- Sensors

the problem specification for the task environment. The rational agent we want to design for this task environment is the solution.

2.1 Environment

- Fully observable (vs. partially observable): An agents sensors give it access to the complete state of the environment at each point in time.
- Deterministic (vs. stochastic): The next state of the environment is completely determined by the current state and the action executed by the agent. (If the environment is deterministic except for the actions of other agents, then the environment is strategic)
- Episodic (vs. sequential): The agents experience is divided into atomic episodes (each episode consists of the agent perceiving and then performing a single action), and the choice of action in each episode depends only on the episode itself.
- Static (vs. dynamic): The environment is unchanged while an agent is deliberating. (The environment is semi-dynamic if the environment itself does not change with the passage of time but the agents performance score does.)
- Discrete (vs. continuous): A limited number of distinct, clearly defined percepts and actions. E.g., checkers is an example of a discrete environment, while self-driving car evolves in a continuous one.
- Single agent (vs. multi-agent): An agent operating by itself in an environment.
- Known (vs. Unknown): The designer of the agent may or may not have knowledge about the environment makeup. If the environment is unknown the agent will need to know how it works in order to decide.

2.2 Agents

The following agents types are listed in order of generality.

2.2.1 Simple reflex agents

- select an action based on the current state only ignoring the percept history
- Can only work if the environment is fully observable

2.2.2 Model-based reflex agents

- Handle partial observability by keeping track of the part of the world it cant see now.
- Internal state depending on the percept history (best guess).
- Model of the world based on how the world evolves independently from the agent, and how the agent actions affects the world

2.2.3 Goal-based agents

- Knowing the current state of the environment is not enough. The agent needs some goal information.
- Agent program combines the goal information with the environment model to choose the actions that achieve that goal.
- Flexible as knowledge supporting the decisions is explicitly represented and can be modified.

2.2.4 Utility-based agents

- Sometimes achieving the desired goal is not enough. We may look for quicker, safer, cheaper trip to reach a destination.
- Agent happiness should be taken into consideration. We call it *utility*.
- A utility function is the agents performance measure
- Because of the uncertainty in the world, a utility agent choses the action that maximizes the expected utility.

2.2.5 Learning agents

Four conceptual components:

- Learning element: responsible for making improvements
- Performance element: responsible for selecting external actions. It is what we considered as agent so far.

- Critic: How well is the agent is doing w.r.t. a fixed performance standard.
- Problem generator: allows the agent to explore.

2.3 Agent's organization

- Atomic Representation: Each state of the world is a blackbox that has no internal structure.
- Factored Representation: Each state has some attributevalue properties
- Structured Representation: Relationships between the objects of a state can be explicitly expressed.

3 Search Agents

Applications

- Route-finding problem
- Travelling salesmane person
- VSLI layout: position million of components and connections on a chip to minimize area, shorten delays. Aim: put circuit components on a chip so as they dont overlap and leave space to wiring which is a complex problem.
- Robot navigation
- Automatic assembly sequencing
- Protein folding

3.1 Problem formulation

- Initial state: the state in which the agent starts
- States: All states reachable from the initial state by any sequence of actions (State space)
- Actions: possible actions available to the agent. At a state s, $Actions(s)$ returns the set of actions that can be executed in state s. (*Action space*)
- Transition model: A description of what each action does $Results(s, a)$
- Goal test: determines if a given state is a goal state
- Path cost: function that assigns a numeric cost to a path w.r.t. performance measure

3.2 Search Space

- State space: a physical configuration
- Search space: an abstract configuration represented by a search tree or graph of possible solutions.
- Search tree: models the sequence of actions
  - Root: initial state
  - Branches: actions
  - Nodes: results from actions. A node has: parent, children, depth, path cost, associated state in the state space.
- Expand: A function that given a node, creates all children nodes

The search space is divided into three regions:

- Explored (a.k.a. Closed List, Visited Set)
- Frontier (a.k.a. Open List, the Fringe)
- Unexplored.

3.3 Search Strategies

Strategy evaluation 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:

- b: maximum branching factor of the search tree (actions per state).
- d: depth of the solution
- m, or D: maximum depth of the state space (may be ∞).

4 Uninformed search

Strategy	Complete	Time	Space	Optimal	Implementation	Comment
BFS	Yes, if b is finite	$O(b^d)$	$O(b^d)$	Yes, if uniform cost	FIFO	High memory requirement and exponential time complexity
DFS	No, fails in: infinite depth spaces and spaces with loops	$O(b^m)$	$O(bm)$	No	LIFO	Bad if m is much larger than d but if solutions are dense, may be much faster than BFS
Depth limited					DFS with limit	
Iterative deepening					Depth deepening with increasing limits	Benefits of BFS and DFS
Uniform cost	Yes, if solution has finite cost	$O(b^{C^*/\varepsilon})$	# of nodes with $g \leq C^*$, $O(b^{C^*/\varepsilon})$	Yes	BFS expanding by cost of path (not of last step)	

Table 1: Uninformed Search Strategies

Where:

- b is the branching factor
- d is the depth level
- m is the maximum possibility of the search space