Chapter 1

Paradigm

Paradigm of AI divided into four categories:

1. Acting Humanly

   To be intelligence, a machine need several capabilities:

   - **Natural Language Processing** to communicate in human language.
   - **Knowledge Representation** to store what machine already knows.
   - **Automatic Reasoning** to use the stored information to answer questions and to draw new conclusions.
   - **Machine Learning** to adapt to new circumstances and to detect and extrapolate patterns.

   If a machine need to be physic feedback to be intelligence, then a machine needs extra capabilities, there are:

   - Computer Vision
   - Robotics

2. Thinking Humanly

   The idea is intelligent is not defined by physical contact. Human likely growing with experience , thus they capable to make predictions, interpret, communicate, reasoning and so on. In another word, human can do or explain anything intuitively.

   There a term **Heuristic** that refer to produce ‘best so far’ solution rather than exactly correct answer that need exhaustive calculations. Example of heuristic AI is GPS (General Problem Solver) created by Allen Newell and Herbert Simon in 1961.

3. Thinking Rationally

   This paradigm simply contradiction of thinking humanly that follow **Logic** to produce answer.

4. Acting Rationally

   Use a term **Agent** refer to who act rationally, thus this paradigm not intended only to machines, but human or animal may interact each other to built system of AI. Computer as an agent expected to do more, such as work autonomously, perceive their environment, persist over a prolonged time period, adapt to change, and create and pursue goals.

   Rational Agent is one to achieve the best outcome, or where there are uncertaintiy, the best expected outcome.

The History of AI

1. 1943-1955

   Warren McCulloh and Walter Pitts (1943) introduced **neural network**.

   Donald Hebb (1949) demonstrated a simple updating rule for modifying the connection strength between neurons, this rule called as **Hebbian Learning**.

   Alan Turing published article "Computing Machinery and Intelligence" at 1950, introduced the Turing Test, machine learning, genetic algorithms, and reinforcement learning.

2. 1956

   AI became a separated field rather than belong to control theory, operation research, decision theory, or mathematic because AI try to duplicating human faculties such as creativity, self-improvement, and language use, clearly branch of computer science and attempt to build machines that will function autonomously in complex and changing environment.

3. 1952 – 1969

   Newell and Simon (1976) formulated **Physical Symbol System** hypothesis which state that a physical symbol system has the necessary and sufficient means for general intelligence action. What they meant is that any system (human or machine) exhibiting intelligence must operate by manipulating data structures composed of symbols.

   At 1958, John McCarthy published a paper **Program with Common Sense** that embody general knowledge of the world. For example, he showed how some simple axioms would enable the program to generate a plan to drive to the airport.

   Study of intelligence program that solve limited domains, also called as **Microworlds**, for example: James Slagle's SAINT program (1963), Tom Evans's ANALOGY (1968), and Daniel Bobrow's STUDENT (1967).

   At 1962, Bernie Widrow enhanced Hebbian learning called as **adalines**. So, Frank Rosen at same year introduced **Perceptrons**.

4. 1966 – 1973

   U.S. National Research Council failed to built machine translator to translate Russian scientific papers, reported at 1966.

   Theory of computational complexity was developed to produce more efficient method by decreasing the need of hardware resources.

   At 1958, early experminets of **Machine Evolution** or **Genetic Algorithm** started.

5. 1969 – 1979

At 1969, Ed Feigenbaum, Bruce Buchman, and Joshua Lederberg built **DENDRAL** program that capable to solve the problem of inferring molecular structure from information provided by a mass spectrometer. DENDRAL is the first **knowledge-intensive system** that its expertise derived from large numbers of **special-purpose rules**.

Feigenbaum an others at Stanford began the Hruristic Programming Project (HPP) to investigate new methodology of **Expert Systems**. Feigenbaum, Buchman, and Dr. Edward Shortliffe developed MYCIN to diagnose blood infections. With 450 rules, MYCIN was able to perform as well as some experts, and considerably better than junior doctors. Unlike the DENDRAL rules, no general throritical model existed to be deduced.

6. 1980 – present

   AI became industry.

   Back-Propagation Learning algorithm reinvented that have been introduced by Bryson and Ho at 1969.

   Hidden Markov Model became popular in speech and handwritten character recognition. **Neural Network** also became popular in data mining. **Bayesian Network** was developed to deal with uncertainty and very popular used in expert system.

   Researcher started look at “whole agent” problem again. Allen Newell, John Laird, and Paul Rosenbloom developed SOAR to answer the problem.

   At 1995, Yarowsky developed method to recognize word-sense disambiguation in a sentence with accuracy above 96%, and with no labelled example at all by collecting very large corpus of unannotated text and just the dictionary definitions of the two senses.

   Banko and Brill (2001) show that techniques like this perform even better as the amount of available text goes from million words to a billion. A mediocre algorithm with 100 million words of unlabeled training data outperforms the best known algorithm with 1 million words.

   Work like this suggests that the “knowledge bottleneck” in AI may be solved in many applications by learning methods rather than hand-coded knowledge engineering.

Chapter 2

Structure of agenst:

$$\text{Agent} = \text{architecture} + \text{program}$$

Agent programs

Percept change in environment and produce the action.

1. Simple reflex agents

   Implementing **if-then rule** to take an action by given environment.

2. Model-based reflex agents

   Implement **boolean model** to built conditional rule as represent to real world rule like physic rule and use it to take a proper decision.

3. Goal-based agents

   Capable to do **Search** and **Planning** to produce desired action based on information of expected goal.

4. Utility-based agents

   Implement **utility function** to take an action that maximize the expected utility of the action outcomes.

5. Learning agents

   Implement **machine learning algorithm** to process feedback from environment, stored it as knowledge and produce best action based on previous actions. If there is negative feedback means that last action has poor performance, then a machine should be generate another scenario to produce better action.

Components of Agents

1. Atomic representations

   Treat states as abstract simple units without care its details. For example, graph theory used this to explain finite state automaton in string matching.

2. Factored representations

   Represent a unit as combinations of certain patterns.

3. Structured representations

   Represent a unit as detailed component as in real world.

Chapter 3

Search Strategies

Let ***b*** is branching factor, ***d*** is depth of search tree and ***m*** is maximum depth.

Uninformed or Blind Search

1. Breadth-first search

   Simple technique that search in every successors of last predecessors. Using **Queue** to maintains generated nodes.

Space complexity = Total nodes generated = $b+b^2+b^3+...+b^d=O(b^d)$

Time complexity is defined by how much time to reach to the goal from initial state. If we build a path after goal node found, then the time complexity will be size of space plus its depth. We can improve it a little bit by recorded every movements of current branch.

Space complexity = $O(b^d)$

Time complexity = $O(b^d)$ to $O(b^{d+1}+d)$

2. Uniform-cost search

Improvement version of breadth-first search, but rather than searching in sequential order, UCS choose only the minimal cost of any generated nodes to be explored next. This algorithm use *lowest path cost function* $g(n)$ in **Priority Queue** to maintains generated nodes sorted by its cost. Because $g(n)$ not determined by depth, then its time and space complexity is not determined. So, let assume C^* is cost of the oprimal solution and ε is constant of minimum transitions.

Space complexity = $O(b^{1+[C^*/\varepsilon]})$

Time complexity = $O(b^{1+[C^*/\varepsilon]})$

When all step costs are equal then

Space complexity = $O(b^{1+d})$

Time complexity = $O(b^{1+d})$

3. Depth-first search

Simple technique that search in every depth first. Using **Stack** to maintains generated nodes.

Space complexity (without reduced last explored depth) = $O(b^m)$

Space complexity (reduced last explored depth) = $O(b*m)$

Time complexity = $O(b^m)$

4. Backtracking search

Similar with DFS, but only generate one successor at a time to explored. So, all branching factor for each depth is always one.

Space complexity = $O(m)$

5. Depth-limited search

Improvement of DFS by limiting search depth into *l* depth. This algorithm very useful if we already know longest depth to be searched. For example, if we already know that route from A

to B is 15 at most, then we simply do searching in any combination with most of length or depth is 15.

Space complexity = $O(b*l)$

Time complexity = $O(b^l)$

6. Iterative deepening depth-first search

   This algorithm used when we not know how depth the solution is. This algorithm try to find the solution in *l* depth, switching to another branch if last branch is not the solution, if solution not found in current depth limit, then gradually increasing the depth until the solution found.

   In the worst case, time complexity and space complexity of iterative deepening DFS same as BFS.

   Space complexity = $O(b^d)$

   Time complexity = $O(b^d)$

7. Bidirectional search

   The idea is if we run two simultaneous search from initial state forward to goal and goal to initial state, then both of search will intersected in $d/2$ of search tree. We can implementing BFS or iterative deepening DFS in bidirectional search.

   If we use BFS, then:

   Space complexity = $O(b^{d/2})$

   Time complexity = $O(b^{d/2})$

Informed or Heuristic Search

Similar to uninformed search algorithm that use $g(n)$ to choose which best move, informed search use *heuristic function* $h(n)$ to choose best movement heuristically depend on the state at the node. For example, one might estimate distance from A to B using manhattan distance.

1. Greedy best-first search

   This algorithm always choose move which is best at the moment obeying consideration that may past state has better cost.

   $$f(n)=h(n)$$

   Space complexity = $O(b^m)$

   Time complexity = $O(b^m)$

2. A* search

This algorithm use $g(n)$ to determine movement cost and $h(n)$ to determine cost of current state to the goal. To ensure that current choice is globally optimum, A* maintain all generated nodes and their cost into **priority queue**.

$$f(n)=h(n)+g(n)$$

A* search is complete, optimal, and optimally efficient among all such algorithms. For the problems with constant step costs, the growth in run time as a function of the optimal solution depth *d* is analyzed in terms of the **absolute error** or the **relative error** of the heuristic. The absolute error is defined as $\Delta \equiv h^{¿} - h$, where **h*** is actual cost of getting from the root to the goal, and relative error is defined as $\varepsilon \equiv (h^{¿} - h)/h^{¿}$.

Time complexity = $O(b^{\Delta})$

For constant step costs:

Time complexity = $O(b^{\varepsilon d})$

3. aa