Introduction to Intelligent Systems: Homework 1

Alvin Lin - Section 1

August 2017 - December 2017

Problem 1

Describe and explain the difference between:

1. Strong and Weak AI: Strong AI is the school of thought that a sufficiently well programmed computer can be intelligent and conscious in the way that a human is. Weak AI is concerned with the application of intelligent methods to efficiently solve problems with computers.

2. Strong and Weak Methods: Strong methods rely on practical world knowledge to solve problems, while weak methods use logical analysis instead. (Also very analogous to functional vs imperative programming).

3. Neats and Scruffies: Neats think that AI study should be rigorously and mathematically backed by proofs and logic, while scruffies approach AI with a trial and error perspective that focuses on AI that works best in the real world. These attitudes are similar and analogous to the functional and imperative programming paradigms, since a functional approach to programming is very similar to a neat's approach to AI.

Problem 2

Many of the computational models of cognitive activities that have been proposed involve quite complex mathematicaly operations, such as convolving an image with a Gaussian or finding a minimum of the entropy function. Most human (and certainly all animals) never learn this kind of mathematics at all, almost no one learns it before college, and almost no one can compute the convolution of a function with a Gaussian in their head. What sense does it make to say that the "vision system" doing this kind of mathematics, whereas the actual person has no idea how to do it?

Computer vision algorithms operate on vast amounts of linear algebra and perform a very large number of complex calculations to perform a task such as identifying chairs in a scene. In contrast, humans can do it "effortlessly" without thinking or knowing about any math, since the complexity of their action is abstracted away in the activity of their neurons. A person identifying chairs in a scene is not aware of the complexity of the neural activity required to perform the action.

Problem 3

"Surely computers cannot be intelligent - they can only do what their programmers tell them." Is the latter statement true, does it imply the former?

An interesting question arises if we consider the situation where a programmer tells a computer to be intelligent. The latter statement is true since computers must be programmed, but whether or not it implies the former depends on our definition of intelligent. Is a computer intelligent because it can do something their programmer did not tell them to do? This is possible since computers can be programmed

with models that learn from and affect the environment to try and accomplish a task, which can lead to the computer doing something the programmer did not *explicitly* tell it to do.

Problem 4

"Surely animals cannot be intelligent - they can only do what their genes tell them." Is the latter statement true, and does it imply the former?

The latter statement is true since an animal's nature and actions (to some degree) are determined by their genetics. Animals also act, respond, and adapt to stimuli in their environment in order to survive. One can argue that this adaptive behavior is a result of genetic "programming", but this does not imply that their behavior *not* intelligent. Animals can be intelligent in this capacity because they are programmed by their genes to be intelligent.

Problem 5

List and describe at least 5 techniques that can be used to cope with intractability (they do not have to be AI techniques per se) and for each technique, explain the advantages and disadvantages of the approach.

- Restricting the problem domain: this technique narrows the problem domain so that only a small subset of inputs need to be considered. This technique works well for localized problems where the input is known to be in a small range since a naive algorithm can be used to solve the problem, but it cannot scale or be generalized to solve similar problems of higher complexity.

- Brute force: this technique tries to solve a problem by testing every single possible solution. This technique is guaranteed to find a solution where the solution space is known, but it will generally take an unreasonable amount of time to do so. Brute forcing also requires massive amounts of computing resources, and may not work for problems where the solution space is not finite.

- Approximation: using an approximation algorithm to solve a problem will yield near-optimal solutions in a reasonable amount of time. The advantages of this technique are that for problems where small errors are not important, reasonable solutions can be found in reasonable amounts of time. However, approximation algorithms do not give exact answers and may be very inaccurate for certain domains of inputs.

- Heuristics: using a heuristic allows for the brute force technique to be applied in a reasonable amount of time. Heuristics allow for the solution space to be reduced based on a set of rules. This allows for a solution to be found much faster.

- Picking randomly: using randomness to optimize a search pattern is fast and inexpensive, but the solution time becomes unknown and the algorithm may take much longer to converge towards a solution.

If you have any questions, comments, or concerns, please contact me at alvin@omgimanerd.tech