Modeling Strategies as Programs: How to Study Strategy Differences in Intelligent Systems with Program Synthesis

James Ainooson

Department of Computer Science, Vanderbilt University
james.ainooson@vanderbilt.edu

Abstract

When faced with novel tasks, humans have the ability to form successful strategies, seemingly without much effort. Artificial systems, on the other, hand cannot, at least when the flexibility at which humans perform is considered. For my dissertation, I am using program synthesis as a tool to study the factors that affect strategy choices in intelligent systems. I am evaluating my work through agents that reason through problems from the Abstract Reasoning Corpus and The Block Design Task.

Introduction

You are presented with the three cards shown in Figure 1(a), and without any more specific instructions, you are asked to pick a card. What choice will you make? Now, let us assume a second set of cards, shown in Figure 1(b), is subsequently presented to you. Now, what card do you pick? And similarly, a third set of cards, as shown in Figure 1(c) is presented to you. Now, what card do you pick?

The task we just worked through can be considered as an instance of the "odd-one-out" task. In this particular formulation, one thing stands out: although there is no verbal communication of the goal, most people would likely be able to form a strategy that may lead to a goal after observing the sequence of the problems presented.

Humans can form strategies for novel tasks, seemingly without effort (Mumford et al. 1993), but current AI systems cannot, at least with the level of flexibility shown by people (including young children in many cases). This remains true even when recent advances in machine learning techniques are considered. Most algorithms require large amounts of training data, long training times, and immense computation computational power.

In order to build systems that could exhibit human-like fluidity in forming strategies, and for us to study the factors that affect the strategy choices of an intelligent system, we may need to have a formal way of representing these strategies. As such, in my dissertation research, I am addressing the following problem: How can we model an intelligent system's strategy formulation techniques using synthesized programs?

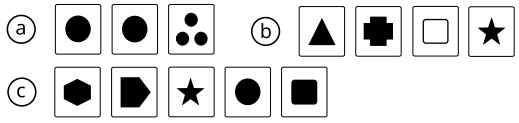

Figure 1: A simple "odd-one-out" task that demonstrates fluidity of human strategy learning.

In this dissertation pursuit, I will be answering the following research questions, with a focus on a small set of visuospatial reasoning tasks:

1. For a chosen task domain, how can we define a minimal set of operations to manually generate strategies that are sufficient for various levels of performance?
2. Provided a program synthesis approach is taken as a means to generate strategies for intelligent systems that reason through specific visuospatial problems, what kinds of search techniques can be applied to ensure the production of strategies?
3. How can the techniques adopted from question 2 above be applied to fit programs on human performance for the purposes of understanding their strategies on standard visuospatial reasoning tasks?

The tasks I have been working on are mainly The Block Design Test (BDT) (Kohs 1920) and The Abstract Reasoning Corpus (ARC) (Chollet 2019). I have also performed additional experiments, specifically around the sufficiency of using imagery as a representation for reasoning on the VZ-2 paper folding task (Ekstrom and Harman 1976) and the Leiter International Performance Scale-Revised (Leiter-R) (Roid and Miller 1997).

The BDT is a test in which subjects are required to reproduce a design (in the form of a geometric pattern) with special patterned blocks. The ARC, on the other hand, is a general intelligence test designed with artificial systems as a primary target. It requires agents to learn abstract concepts over a limited set of training items (typically 3) for later evaluation.

Methods

I have performed earlier work on the sufficiency of using imagery as representations for reasoning on specific tasks,

and also on how the diversity of strategy choices affects the outcomes of certain given tasks (Ainooson and Kunda 2017; Ainooson et al. 2020). As a next step, I am currently studying how strategies for reasoning about some given tasks can be effectively represented as programs.

When it comes to this, I have formulated the Visual Imagery Reasoning Language (VIMRL). In VIMRL, every program is a state machine which contains a set of states, a set of edges for transitioning between states, and a set of rules that govern transitions. Further, each state has an attached list of instructions, known as the state script, which are executed whenever the state is active. Transitions between states are conditioned on the side-effects of actions performed in these state scripts.

Defining Strategies with VIMRL

Primarily, strategies expressed in VIMRL are generated through a search process. Given the structure of VIMRL programs, however, the search space of possible strategies tends to be large and almost intractable in some cases. In order to keep the size of this search space in check, I have worked on a collection of logical rules that are able to strip most malformed and invalid machines from the search space. Additionally, I have worked on an ordering algorithm that makes it possible to compare any two VIMRL programs that are logically equivalent regardless of how they are structured.

I am currently evaluating VIMRL on tasks from the ARC, which provides a diverse collection of tasks with which I am honing search algorithms for exploring VIMRL programs. As I am making progress on problems from the ARC, I will be transferring the knowledge and other benefits derived from the improved search performance to problems on the Block Design Test.

Solving Items from the ARC and BDT

ARC's structure makes it a great source of tasks when testing program synthesis algorithms. Each task in ARC is presented as a series of input-output grid pairs. An agent solving a task from the ARC must predict the contents of an output grid by transforming an input grid according to some abstract concept. The concepts for each task must be determined by the agent from a few input-output training pairs. With very few training items, traditional machine learning algorithms may have difficulty generalizing the concepts required to successfully solve tasks from the ARC.

My approach to building solvers for the ARC relies on program synthesis. For these experiments I am using a stripped down version of VIMRL in which a program has just a single state and no transitions. For my work on ARC, I am generating a dataset of hand written ground-truth solution programs for items in the ARC's training set. From these ground-truth programs, I generate models that can be used to induce new programs for unseen tasks.

When it comes to the BDT, I am currently working with simplified versions of the test in which there is just a single block whose movements are limited (such as spinning along just a single axis). All my experiments for the BDT are performed in a virtual environment in which reasoning agents have selective attention (through a gaze window) and the potential to have a limited amount of short term memory. As search performance increases, I will be experimenting with the full BDT as it is typically administered.

Proposed Work

For the rest of my dissertation research, I will be focusing on the following activities:

1. I will be working toward a more stable implementation of VIMRL. Key to the success of much of this work are the various search algorithms. As such, I will be putting in much of my effort to improve the search algorithms used for synthesizing programs in VIMRL. My goal for this subtask is to have decent scores on the ARC and good performance on generating block design strategies.
2. I will be transferring knowledge obtained from building ARC solvers into building block design solvers. Once I have a system that works on the block design task, I will be investigating factors that affect strategy on the Block Design Test through ablation tests on artificial agents.
3. Finally, I intend to investigate how programs can be generated from traces of human performance on the Block design Test. From these traces one goal will be to find programs that are consistent with the behaviour pattern of an individual (or a group of individuals). My methods for analysing human performance using program synthesis will be evaluated on data collected using virtual BDT assessments (collected for another project ongoing in my research lab) from large samples (n=500 in each group) of neurotypical and autistic adults.

References

Ainooson, J.; and Kunda, M. 2017. A Computational Model for Reasoning About the Paper Folding Task Using Visual Mental Images. In *Proceedings of the 39th Annual Meeting of the Cognitive Science Society, CogSci 2017*, 1519–1524.

Ainooson, J.; Michelson, J.; Sanyal, D.; Palmer, J. H.; and Kunda, M. 2020. Strategies for Visuospatial Reasoning: Experiments in Sufficiency and Diversity. In *Proceedings of the 8th Annual Conference on Advances in Cognitive Systems (ACS)*, 20.

Chollet, F. 2019. On the Measure of Intelligence. *arXiv preprint arXiv:1911.01547*.

Ekstrom, R. B.; and Harman, H. H. 1976. *Manual for kit of factor-referenced cognitive tests, 1976*. Educational testing service.

Kohs, S. C. 1920. The Block-Design Tests. *Journal of Experimental Psychology*, 3(5): 357–376.

Mumford, M. D.; Baughman, W. A.; Threlfall, K. V.; Uhlman, C. E.; and Costanza, D. P. 1993. Personality, Adaptability, and Performance: Performance on Well-Defined Problem Solving Tasks. *Human Performance*, 6(3): 241–285.

Roid, G. H.; and Miller, L. J. 1997. Leiter international performance scale-revised (Leiter-R). *Wood Dale, IL: Stoelting*.