Jacob Kilver

Knowledge Based AI

Project 2 Reflection

3 July 2016

# Introduction

This project intended to build upon the previous project of solving 2x2 Raven’s Progressive Matrix (RPM) problems. In this project 3x3 RPMs were solved in addition to the 2x2 matrices from the last project. Generate and test was used to solve the 2x2 matrices, and a similar approach was attempted here. However, since the patterns for 3x3 matrices were more varied, a more tailored approach to generate and test was used here.

# Theory of Operation

In the previous project the “generate and test” method proved to be quite powerful but at times too general. As such, this same technique was applied here but with some modifications. The relationships between images and the patterns for the 3x3 RPMs were much more complex than in 2x2 matrices. (The 3x3 matrices were more than twice as large!) As such, generalizing the relationships proved too complex to handle. The patterns were too varied to effectively abstract them, so the problems were handled on a case by case basis. With this structure the agent was able to answer problems it had encountered before, but it was not equipped solve new problems well unless these problems fit the same relationships it was trained to know.

Another difference between the 2x2 and 3x3 designs was how answers were selected. For the 2x2 matrices there were usually multiple ways to generate an answer that fit the pattern but would result in an incorrect answer. As some code was necessary to handle cases where multiple generated guesses appeared in provided answers. For the 3x3 matrices with their complicated patterns meant that having multiple guesses that matched an answer was very unlikely. Therefore the first guess that appeared in the provided answers was returned as the answer to the RPM.

A purely visual approach for deriving solutions was taken here. While there were some problems that would have benefitted from a verbal approach, it was decided to remain with an entirely visual approach (for the sake of simplicity and due to time constraints). Note that this marks a deliberate difference from how the how the author reasoned while solving the RPMs. When initially solving the RPMs, the author found that he would derive a pattern and store this in some verbal form (i.e., shapes plus relationships between them). Then, when looking for an answer, a match to the verbal description would be sought ought, when entailed translating the visual representations of the answers into a verbal form. While all this happened rather quickly for the author, converting a visual scene to a verbal one was simply not capable for this project. While verbal representations were available, since half the problems will be without them in the next project, the visual approach was taken here.

# Implementation

Python 2.7 was used to create the agent for this project. Since a visual approach was used, the Pillow library was used to manipulate the images.

As described above, the approach for this project consisted of a series of cases for generating an answer, usually specific to only one or two problems from the in-sample problem set. If the “guess” image that was generated matched reasonably well to one of the provided answers, that answer was assumed to be correct and the answer number was returned along with a confidence rating. As a result, if a match was made none of the other tests were performed.

Table 1 lists all the tests that the agent used to solve the RPMs for this project. It should be noted that some problems were deliberately not answered (though the agent might have given an incorrect answer). Also, sometimes the agent had some luck and correctly answered a problem that it was not intended to solve.

Table : Tests used by agent to solve RPMs

|  |  |  |
| --- | --- | --- |
| Test Description | Solution | RPMs it was intended to solve |
| Rows are all equal | Copy any image from last row | C-01, D-01 |
| Difference between columns is the same along a row | Apply difference between E and F to image H | C-11, C-12 |
| D and F are vertical reflections, B and H are horizontal reflections, C and G are 180° rotations | Rotate A 180° | C-07 |
| B and D are 90° CW rotations, C and G are equal, F and H are 90° CCW rotations | Rotate F, logical OR with H | C-08 |
| Pattern shifts right | Copy image A | D-02, D-03, D-11 |
| Pattern along rows, separate pattern that shifts right | Combine patterns | D-06 |
| Logical AND of all RPM question images | Take logical AND of all RPM questions (A-H) | C-05 |
| All the questions appear in the provided answers but one | Select the one answer that does not appear in the RPM questions | C-06 |
| Bisect image along vertical axis. Swap halves | Same as test | C-09 |
| Pattern along row, pattern along columns | Combine row and column pattern | D-04, D-05 |
| One pattern shifts left, another pattern shifts right | Combine right and left shift patterns | D-10 |

# Results and Discussion

I am guessing that the performance will be quite good on the Basic and Test sets, but be quite bad on the Challenge and Raven’s sets.

The agent here is not operating at the properly level of abstraction. It is reasoning over pixels while as a human I reason over shapes. As such, if two images do not exactly match, not only are humans not likely to notice (subtle differences are hard to see, especially when they are not layered one on top of the other), but they are also able to see the larger pattern.

Efficiency? Slower because more tests to run (11 tests)

# Conclusions and Future Work