Jacob Kilver
Knowledge Based AI
Project 2 Reflection
3 July 2016

1) Introduction

The purpose of this project was 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 applied here. However, since the patterns for 3x3 matrices were more varied, a more tailored approach to generate and test was used. While this resulted in acceptable performance for in sample problems, this solution generally overfit the RPMs resulting in poor performance with problems outside the training data.

2) 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 to just a small set of 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. Some code was necessary therefore to handle cases where multiple generated guesses appeared in provided answers. For the 3x3 matrices it was hypothesized that their complicated patterns meant having multiple guesses that matched an answer was very unlikely. Therefore, the first guess that appeared in the provided answers (within a certain confidence range) was returned as the answer to the RPM.

As opposed to how the 2x2 matrix problems were solved purely visually, for this project a combination of visual and verbal techniques was used. In general, a solution was searched for visually, but a few problems required information that could not be generated visually and so a verbal technique was applied. The visual technique, however, 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, which entailed translating the visual representations of the answers into a verbal form as well. While all this happened rather quickly for the author, converting a visual scene to a verbal one was simply not within the scope of this project. Furthermore, since humans solve these problems without any verbal description, the visual approach was favored for this project.

3) Implementation

Python 2.7 was used to create the agent for this project. Since a visual approach was primarily 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. For the visual approach, if the “guess” image that was generated matched reasonably well to one of the provided answers using a simple pixel-by-pixel comparison, 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 subsequent tests were performed. For the verbal approach choosing an answer was much more binary. The patterns were extracted and translated into code. If the tests for the patterns resulted in an answer, it was assumed that this answer was 100% correct, and was returned immediately.

Table 1 lists, in order, all the tests that the agent used to solve the RPMs for this project. Unless noted otherwise, all the problems were solved visually.

Table 1: Tests used by agent to solve RPMs

Test Description	**Solution**	**RPM(s) it was intended to solve**
Images are all equal along row	Copy any image from last row	C-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
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
[Verbal] If the number of objects is the same among all the question figures AND there is a common object among all the figures	Find answer that has “huge” as an object size and that has the common object and that has the same number of objects	C-02
[Verbal] If all the objects in all the figures have the same shape AND the number of objects in the figure is a product of the number of objects in the first figure of that row	Find answer that has the proper number of the proper shapes	C-03, C-04
Bisect image along vertical axis. Swap halves	Same as test	C-09

4) Results and Discussion

The performance of the agent can be seen in Figure 1,Figure 2, andFigure 3. Across the entire test set (the entirety of sets B and C: Basic, Test, Challenge, and Ravens), the agent’s accuracy is only 51%. (See Figure 1.) However, more than half the problems the agent was tested against the agent was not designed to answer (Challenge and Ravens). Furthermore, only half of the problems were available for the agent to train with (Basic and Challenge.) So using just one test set (Basic) the agent was able to answer more than half the problems correctly. While 51% is not very accurate, given the circumstances above this performance was better than expected.

The performance across the graded portion of the test sets was rather good. A full three-quarters of the problems were answered correctly (Figure 2). This was as expected from training with the Basic test sets.

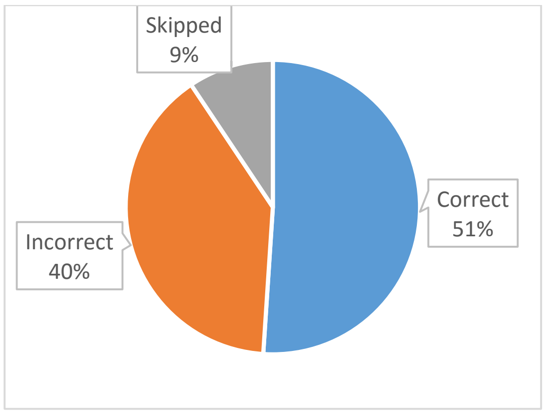

Figure 1: Accuracy of agent across entire test set (Basic, Test, Ravens, Challenge)

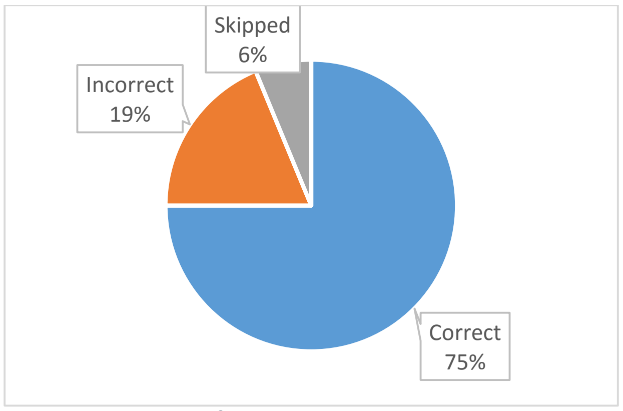

Figure 2: Accuracy of agent across graded sets (Basic and Test sets B and C)

Figure 3 provided the most interesting findings. Since the Basic Problems were available for testing, the performance shown was as expected and agrees with the results during testing. Test set B was off by 1 wrong answer, which was anticipated since the Test Problems were not available for training and some slight difference in results was expected. The surprise came with Test set C. The in-sample accuracy was greater for Basic Problems C than for Basic Problems B, but the out of sample accuracy was much lower for Test Problems C than for Test Problems B.

While the Test problems could not be viewed, it was hypothesized that this inaccuracy was due primarily to two factors. First, some of the tests in Table 1 were rather questionable. While these tests worked for the in-sample case, they were a rather poor choice for the out of sample problems. Second, to arrive at the correct answer for some of the in-sample problems, a great deal of “fuzziness” was required. This issue is described in greater detail below.

As expected, the Challenge and Raven Problem sets were not very accurate. Most of the problems were answered correctly. However, the agent was not designed to answer any of these problems correctly, so the poor results were expected.

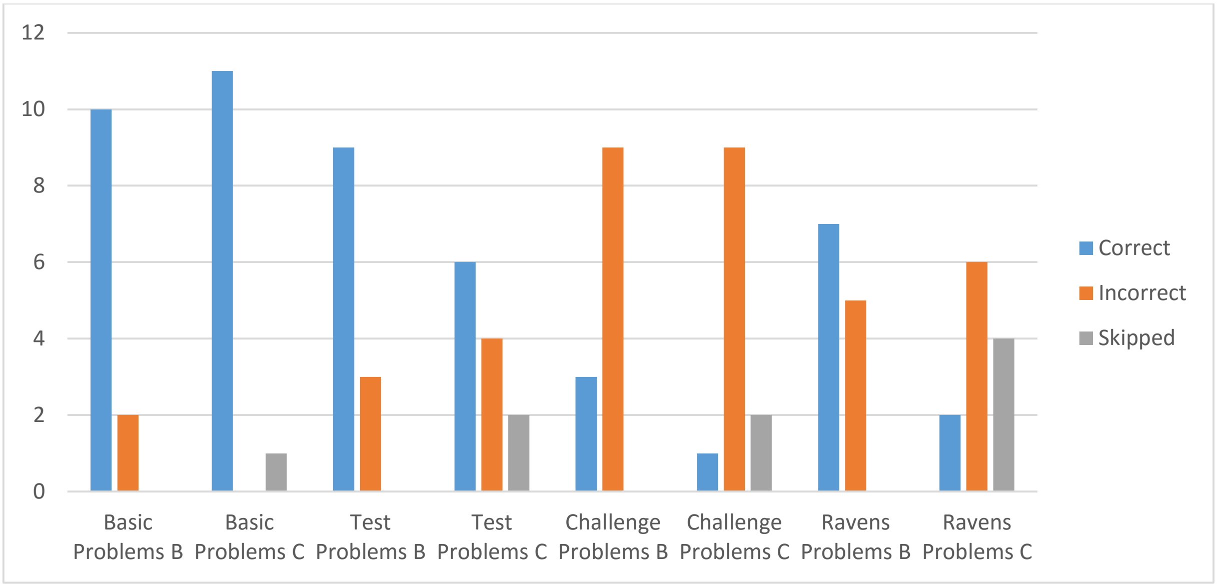

Figure 3: Breakdown of agent performance

Regarding execution speed, no specific tests were performed. The author did note that the agent seemed slightly slower solving the 3x3 problems than when solving 2x2 problems. However this was expected since the 3x3 problems were larger and would require more time to solve. Furthermore, this project involved answering twice as many problems as the first problem, so that increased the execution time as well. None of the tests in Table 1 were overly complicated and did not take that long to run. While it could take quite a while to run through all 9 tests, in practice all 9 tests were rarely performed.

The assumption that false positives would be rare did not hold for this implementation. Due to either small features or some of the imperfections between images, incorrect answers were quite common during the early stages of development. Fortunately, an ordering of the tests was found so that the agent answered the specified problems correctly.

Part of the reason the false positives were so common was because the agent did not operate at the properly level of abstraction. It reasoned over pixels while humans reason over shapes. This was how the author reasoned as he solved the RPM. The placement of individual pixels did not matter nearly as much as the relative location of pixels (as they form shapes). The agent was not able to capture the relative location of the pixels and relied on a “dumb” testing mechanism to determine pixel-by-pixel how many pixels were the same between two images. As such this testing mechanism was highly susceptible to minor perturbations between images. Take for example Figure 4 and Figure 5 below. To the average human these two figures are identical, and for the purposes of answering an RPM they are. However, the circles are slightly offset, and when they are compared, as in Figure 6, there are some minor differences. (Note: Black denotes commonality, white denotes difference.) These minor differences were at times enough to confuse the agent as to what the correct answer was.

Using the verbal techniques could have sidestepped this issue. It would have endowed the agent with a great deal of knowledge about the problem that it would have not have had to generate for itself. The verbal descriptions were generally at the same level of abstraction that the author reasoned at. However this was seen as a form of “cheating” because humans are expected to be able to solve these problems visually. Giving the agent all this background knowledge at no cost seemed to be giving it an unfair advantage.

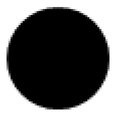

Figure 4: Basic Problem D-03 Image A

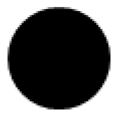

Figure 5: Basic Problem D-03 Image E

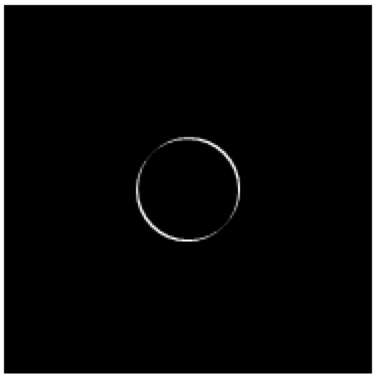

Figure 6: Difference between Basic Problem D-03 images A and E

5) Conclusions and Future Work

In this project an agent to solve 3x3 RPMs was planned, developed, and tested. The performance of the agent depended mostly on the knowledge it was imparted with during development. For problems it was designed to answer the performance was acceptable, but for problems outside of that domain, the accuracy dropped dramatically. This was primarily due to the difficulty in generalizing the 3x3 RPMs into a set of relationships that the agent could then use to solve new problems. Almost every RPM had a different relationship amongst the images, and creating a case for every possible combination simply was not feasible.

The largest shortcoming in the agent presently is the poor technique for comparing images. Currently this is a simple pixel-by-pixel comparison that is susceptible to noise. A new method needs developed that can determine the relationship between pixels in order to develop the concept of a shape, and then reasoning can be done over shapes rather than over pixels.