Designing An Artificial Intelligence Agent To Solve Ravens Progressive Matrices

Tom Eldridge

September 22, 2017

1 Introduction

The Ravens Progressive Matrices test (RPM) is a commonly used intelligence test based on visual reasoning. RPMs involve 2x2 or 3x3 matrices with each cell containing different shapes and relationships. The cell that completes the pattern has to be chosen from a list of six or eight possible solution cells. While these problems are purely visual, they provide a good test of general intelligence. The agent described in this paper was designed to mimic human reasoning by using multiple different approaches to problem solving, similar to the way people solve RPMs (Hunt 1974). The agent relies on a three layered approach to finding solutions. The first layer searches for obvious patterns where the agent can propose an answer that can be explicitly searched for in the list of possible solution cells. The second layer provides a filter for the possible solution cells, and removes cells that violate the observed patterns. The third layer generates attributes for a proposed solution and analyzes the remaining solutions for similarity to the proposed solution attributes, and returns the solution with the highest similarity score. Each layer is able to help in a way that the other layers cannot, and when combined, can solve problems with a high degree of accuracy.

2 Agent System Design

2.1 Explicit Solution Layer

The three layers of the agent are designed to solve problems with increasing levels of difficulty, and operate with diminishing levels of certainty. The first layer looks for solutions with absolute certainty. For example, cases where A and B are flipped vertically, or A and C are flipped horizontally implies that missing cell D must match a reflection of A or B exactly. The agent will then search the possible solution cells for these reflections, and if found, will return the appropriate cell. If A reflects on the y-axis to B, then we should expect cell D to contain a y-axis reflection of cell C.

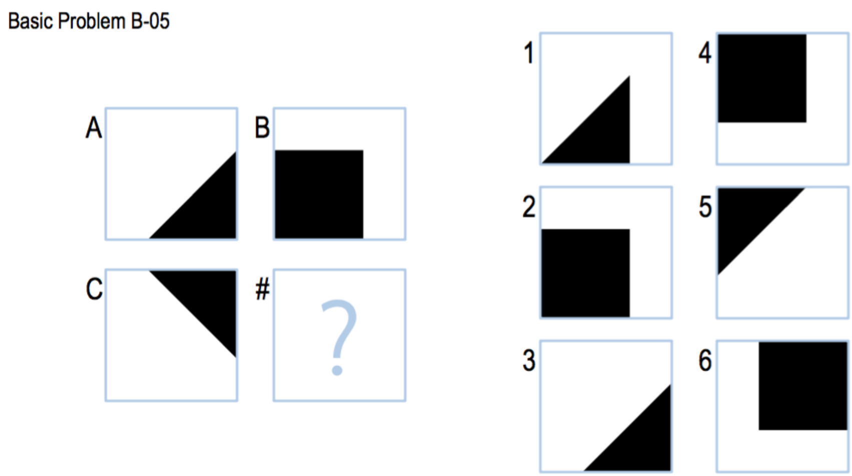

Figure 1: The A to C transformation flips the image across the x-axis. This transformation applied to B produces the correct solution 4.

2.2 Solution Sifting Layer

The second layer acts as a sifter for the possible solution cells. We first perform an image analysis on the three problem cells, and record a variety of attributes. We then do a transformation analysis on how these attributes change from A to B and from A to C. Many solution cells violate patterns observed in the AB, CD cell transformations, and these solution cells can be removed as possibilities. For example, if A and B both contain a large square, we would expect cell D to contain the same sized shape as cell C. Attributes that remain exactly the same, or nearly the same between AB and AC transformations, should be expected to remain the same in the parallel BD and CD transformations. If A and B contain large squares, and C contains a small circle, we can eliminate all solution cells that do not also contain a small circle.

2.2.1 Image Analysis

The image analyzer recorded 13 different attributes for each image. These attributes were the black/white pixel ratio and the minima, maxima, and mean of the distances from left to right, right to left, top to bottom, and bottom to top to reach a black pixel in each row or column. The black/white pixel ratio helped to gauge the fill of each shape, and the distances to a black pixel from each direction helped to abstractly classify images. For example, a square will have the same number for the minima, maxima, and mean from all four directions. These attributes look very different for different shapes, sizes, and orientations, so they provide a good method of classifying images without any previous knowledge. An image comparison function was also needed for the first layer to compare equality between images. If the comparison score falls below

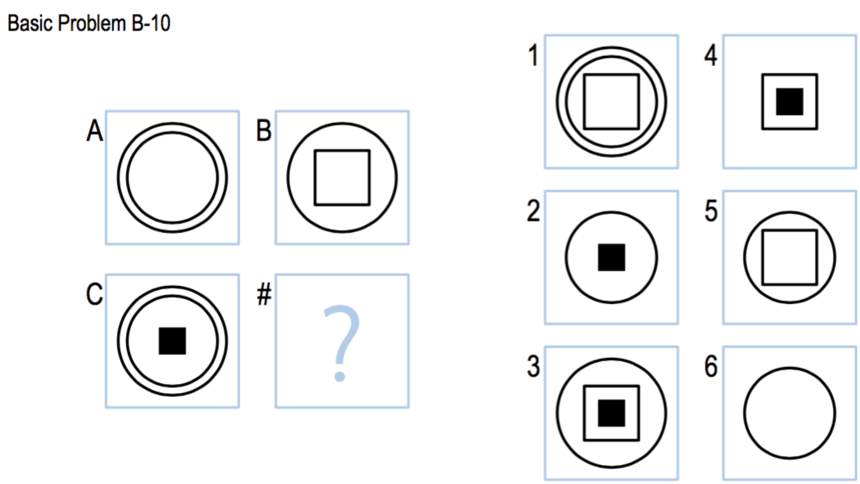

Figure 2: A, B, and C all have the same average distance to a black pixel from the left, right, top, and bottom. We would expected these attributes to be the same for the solution, so we can eliminate solutions 2, 4, 5, and 6.

the threshold of 965, the images are considered identical. This was done by calculating the root of the mean squares between two n x n images, where h is a histogram of the image, and j is the size of the histogram.

$$x_{rms} = \sqrt{\frac{\sum_{i=1}^{j} h_i * i^2}{n^2}} \tag{1}$$

2.3 Solution Ranking Layer

The third layer takes the image analysis of each of the problem cells, and projects a transformation from each cell to D based on the AB, AC transformation analyses. This produces three estimates for the attributes of cell D. The expected attributes of D is the average of those three estimates. The attributes of each remaining solution cell are then compared to the attributes of expected cell D, and a similarity score is generated for each solution. This is essentially a generate and test method. The expected attributes of D are generated, and these attributes are tested against the actual attributes of each solution.

2.3.1 Similarity Scores

The similarity score between each solution cell and the expected attributes of D was originally calculated as a Euclidean distance between the attributes. However, upon further analysis, it appeared that a simple average of the sum of their ratios had superior performance by a statistically significant margin. A

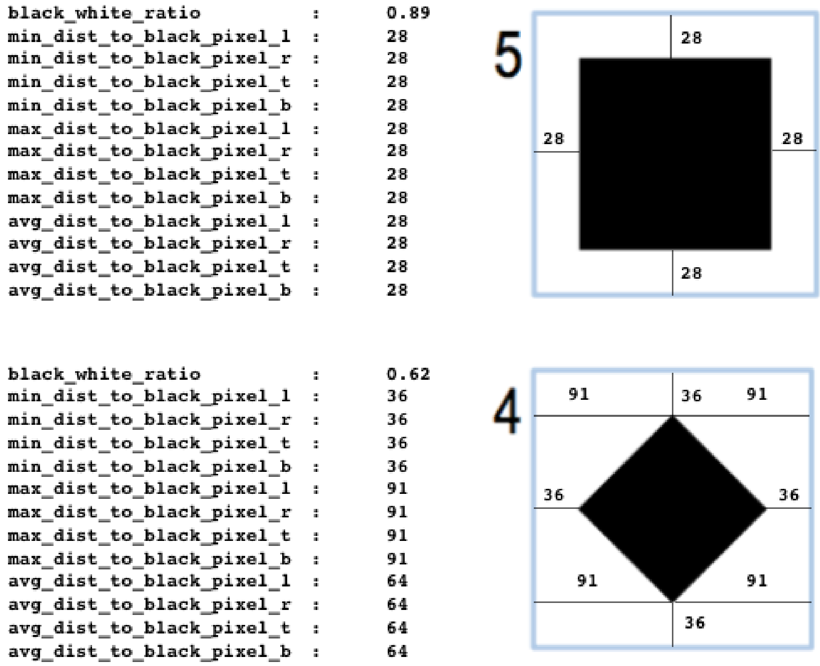

Figure 3: The complexity of difference between a large square, and a medium square tilted 45 degrees is captured by the very different attributes recorded in figures 4 and 5 during image analysis.

similarity score s is calculated for each AD, BD, CD transformation by summing the ratio of n solution attributes a relative to expected D attributes d, and taking the average of those three sums.

$$s = \frac{\sum_{i=1}^{n} \frac{a_i}{d_i}}{n} \tag{2}$$

2.3.2 Metacognition

The solution with the highest similarity score is chosen as the proposed answer. If the similarity score of this solution compared to the expected attributes of D is greater than 95%, then return the proposed answer. Otherwise, if the score does not meet the threshold, but the number of sifted solution cells is less than 5, it is worth returning the proposed answer anyway. If none of these conditions are met, the agent will finally return -1.

	Analysis	Discovery	Example
Layer 1	Apply duplicate and reflection transformations	Discover duplicate AB, AC transformations, or reflective axial AB, AC transformations.	If B is vertical reflection of A, solution is vertical reflection of C
Layer 2	Analyze transformations for duplicated attributes	Discover transformation attributes that remain the same between cells. Expect these attributes in parallel transformations, and discard solution cells that do not match these expected attributes.	Analyze transformations for duplicated attributes
Layer 3	Analyze transformations for ratio changes between attributes	Discover how attribute ratios change during AB, AC transformations, and apply these transformation ratios in each A->D, B->D, C->D case. The average of these projected attribute values are the expected attributes of D. Choose the solution with the highest similarity to the expected D.	If black white ratio doubles from A to B, expect this to double from C to D. Test this against solutions.

Figure 4: The Agent's 3 Layers

3 Results and Analysis

3.1 Agent Performance

The three layers operate on problems with increasing levels of difficulty, and have decreasing levels of certainty. The first layer finds very explicit transformations where the solution is obvious. This handles cases where there are duplicate cells, or simple reflections across axes. This layer was able to correctly answer 8 of the 12 basic problems, and has yet to get an answer wrong. However, it was unable to find an answer for any of the challenge problems. The second and third layers work together to sift out possible solutions, and rank the remaining solutions on their similarity to the expected values of D. These layers were able to solve an additional 3 of the harder basic problems, and 2 of the 12 challenge problems (and skipped 2 challenge problems).

3.2 Cognitive Connection

People appear to solve RPMs using multiple different approaches (Hunt 1974), which is similar to this agents strategy. First, the agent tries to find obvious patterns where it can immediately return a solution. If this fails, it then tries to eliminate solutions that don't match the observed patterns. Once it has narrowed the problem down, it then ranks the remaining solutions on a variety of visual attributes, and their similarity to expected transformations from A, B, and C to D. Importantly, the agent also uses metacognition to gauge it's certainty about proposed solutions, and whether it should return an answer or skip the problem. It is argued that to solve RPMs well, a person or intelligent agent must be able to decide among problem solving approaches, and know when a given approach works or does not work (Keating and Bobbitt 1978).

3.3 Agent Limitations

Notably, the agent did not have a good understanding of what was happening inside the outermost shape. The distances to black pixels only found information about the contours of the outside of the shape, and did not record any information about what was inside. This proved to be a major limitation for problem B-06, where the whole problem is based off of what happens in the outer shapes. It is likely inner and outer relationships are not handled well in general. Finding a way to separate shapes into sets of contiguous black pixels would have been helpful in handling these types of relationships (Joyner, D. 2014).

4 Proposed Improvements

The limitations of the agent could be improved considerably by adding new attributes to the image analysis step. During development of the agent, the addition of the average pixel distance improved the agents performance significantly. By adding additional metrics, especially around the inner parts of the cell, performance could likely be improved further. Potential attributes could be average distance between black pixels, distance from the center of the image to the first black pixel in each direction, or pixel intersection between images. Also, it would be ideal to be able to recognize individual shapes as sets of contiguous black pixels. This would help dramatically in being able to handle more complex cases with multiple shapes within and around each other. Additionally, rather than continuing to blindly add image attributes, it would be good to create some system for weighting each attributes importance. This could be done in conjunction with some type of memory system that can recall example problems.

5 Conclusions

People typically solve RPMs by using a variety of different problem solving approaches combined with metacognition to know when to switch. An agent that uses multiple problem solving layers to reach a solution can mimic this approach and perform quite well. This agent was able to solve 11/12 basic problems and 2/12 challenge problems by using multiple layers with increasing levels of pattern recognition ability, and decreasing levels of certainty for each subsequent layer. While the agent approaches problems similarly to humans in many ways, it is lacking in others. The attributes that were used do not help much in determining what is inside of a shape. This proved to be a significant limitation, but could be solved by adding new attributes to the image and transformation analyses. This could be considered analogous to people learning new patterns. Developing a memory system that gives weighting to more important attributes would also be advantageous. The agent was developed with the intention of mimicking human problem solving, and thus performs fairly well on a general intelligence test.

6 Sources

Hunt, E. (1974). Quote the raven? Nevermore! In L. W. Gregg (Ed.), Knowledge and Cognition. 129-158. Hills- dale, NJ: Erlbaum.

Keating, D., and Bobbitt, B. (1978). Individual and developmental differences in cognitive-processing components of mental ability. Child Development, 155-167.

Joyner, D. (2014). Using Human Computation to Acquire Novel Methods for Addressing Visual Analogy Problems on Intelligence Tests, 6