A Raven's Progressive Matrices Solver

Mutasim Chowdhury

***Abstract*—This paper concerns the design and performance of a Raven's Progressive Matrices solver and agent. The agent utilizes an ensemble method of dedicated heuristics, general predictive heuristics, and pixel ratios to determine the most likely answer to an RPM problem. The agent is able to solve 85 out of 96 Basic and Test problems and performs above the 50th percentile across all 192 problems. Discussion of the agent's successes due to strong heuristics and failures due to ambiguity in pixel heuristics is offered. The agent is deduced to be human like, or human-mimicking but presented as still maintaining some machine non-human like behaviors.**

1 AGENT DESCRIPTION

1.1 High Level Approach and Similarity Metrics

The agent solves Raven's Progressive Matrices by first differentiating the problem input into three particular cases. Each case is then approached based on the initial case and/or subsequent subcase. Answers are chosen based on the following metrics.

1.1.1 *Similarity Metrics to Determine Answer Choices*

- OpenCV's histogram matching method on two images for 3x3 matrices.
- The intersection over union (IOU) of two images.
    - Determined by generating the bitwise AND, and bitwise OR of two images. The result is summing the pixels in both and dividing the former by the latter.

Dark Pixel Ratios (DPRs):

- Generally, a measure of the difference between the black pixel count difference between two images, divided by the image size (184x184).

- For 3x3 problems, the agent utilizes the percentage of dark pixels divided by the pixel size in the entire row. This results in a 3-tuple containing 3 float values.

Intersection Pixel Ratios (IPRs):

- Defined as the number of shared black pixels (in value and location) in each image divided by the total number of black pixels present across both images.
- For Set D and E 3x3 problems, the IPR remains the same but is the union of all three images divided by the total number of black pixels in the row (Joyner, 2015).

For single values, absolute distance is used or distance to 0 or 1. For multiple values, minimization of the Euclidean distance is utilized.

1.1.2 *Cases and Subcases*

1. *Basic Problems*
    a. These problems are solved utilizing the DPR and IPR values shared with the horizontally adjacent matrix cell (C, H) to the answer. The agent processes the pixel ratios and looks for the matching ratios in a case dictionary.
2. *A 2x2 problem (Test, Challenge, Raven's)*
    a. Matching unary transforms with a scoring mechanism
        i. If a reflection across either axis, or an identity relationship is found, predict the answer by applying the transform to cell C. The most likely answer is selected by voting which answer choice is most similar to the predicted answer.
    b. In the event that no votes or pre-defined problems are detected, then we have an ambiguous result. The agent will then calculate the Euclidean distance between the (DPR, IPR) tuple for A and B, and C and the answer choice ('D'). The minimum Euclidean distance is then chosen as the answer choice.
3. *A 3x3 problem (Test, Challenge, Raven's)*
    a. Generating strong matches using binary transforms:
    b. The primary function of the 3x3 solver. The agent examines a list of predefined heuristics/transforms and applies them to each of the first two cells in the first two rows of the matrix. The agent

then determines if the same transform has been applied to both rows. If so, the agent will then predict the answer by applying the binary transform to cells G and H. The answer choice with the highest histogram matching score is returned.

4. *Developer-specified heuristics*
    a. The agent uses problem numbers of certain problems to directly apply the expected pattern/transform and choose the predicted answer.
5. In the event that no votes or pre-defined problem are made, then we have an ambiguous result. Here, the agent will select a pixel ratio measure distance to minimize depending on the problem set. For problem set C, the dark pixel ratios across the rows are used. For problem sets D and E, the intersection pixel ratios are used. The distance measure in both cases is Euclidean.

1.2 List of Transforms

The following is a list of transforms used to solve 2x2 (unary) and 3x3 (binary) problems.

1.2.1 *Unary Transforms (A, B)*

1. Identity (A == B)
2. Horizontally Reflected (A reflected over the Y-axis == B)
3. Vertically Reflected (A reflected over the X-axis == B)

1.2.2 *Binary Transforms (A, B, C)*

1. Bitwise OR (A ∨ B == C)
2. Bitwise AND (A ∧ B == C)
3. Bitwise XOR (A ⊻ B == C)
4. Bitwise SUB (A – B == C)
5. Bitwise BACK_SUB (B-A == C)
6. Inversions of the previous 5 operations.
7. Vertical stitching (top-half(A) + bottom-half(B) == C)
8. Left Subtraction (The shape in B removed from the left side of A == C)

1.2.3 *Dedicated Problem Approaches or Developer Determined Specific Heuristics*

Generally, the agent uses a mix of pixel measurements or different orientations to determine the answer of a problem. A non-exhaustive list is as follows:

1. Column-wise unary and binary transforms in lieu of row transforms.
2. Determining symmetrical matrix-wise relationships, i.e. (A, B) flipped vertically is equivalent to (G, H), thus the answer should be C flipped.
3. Multi-step unary or binary transforms.
    a. 2x2: If B is in some way the result of a binary transform of A and B, then D should hold the same for A and the answer choice
    b. 3x3: If the XOR of the entire row is equal to the same shape as the XOR of the third row, the answer should XOR with G and H to yield the same shape as well.

2 AGENT PERFORMANCE

Table 1 — Agent Performance Across Sets (Correct/Incorrect)

Test Type	B	C	D	E	Total	Percentage
Basic	12/0	12/0	12/0	12/0	48/0	100%
Test	11/1	10/2	7/5	9/3	37/11	77%
Challenge	0/12	6/6	3/9	1/11	10/38	20.80%
Raven's	8/4	8/4	4/8	7/5	27/21	56.30%
Total	31/17	36/12	26/22	29/19	122/70	63.50%
Percentage	64.50%	75%	54.70%	60.40%	63.50%	

The agent's performance across all problem sets and subsets is listed in Table 1. With respect to the Basic and Test sets, the agent solves 85/96 problems, or 88.5 %. The time complexity is O(NM) where M is the number of transforms computed per the number of images within the input matrix and answers. Overall, this amortizes with respect to the input as remaining constantly linear. With the transforms provided previously, the agent took 33 seconds to complete all 192 problems.

When running the agent with the 'Basic' case dictionary disabled, the agent passes 37 out of 48 Basic problems, equivalent to the test scoring seen above.

3 AGENT SUCCESSES

The following problems selected represent will be discussed with respect to the agent's performance without utilizing the case dictionary for proper analysis.

3.1 Basic Problem B-10 | Specific Subcase Algorithm Success

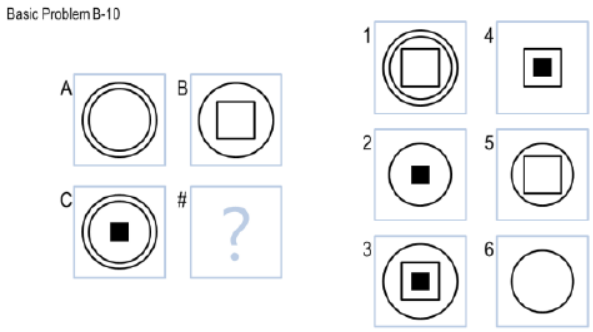
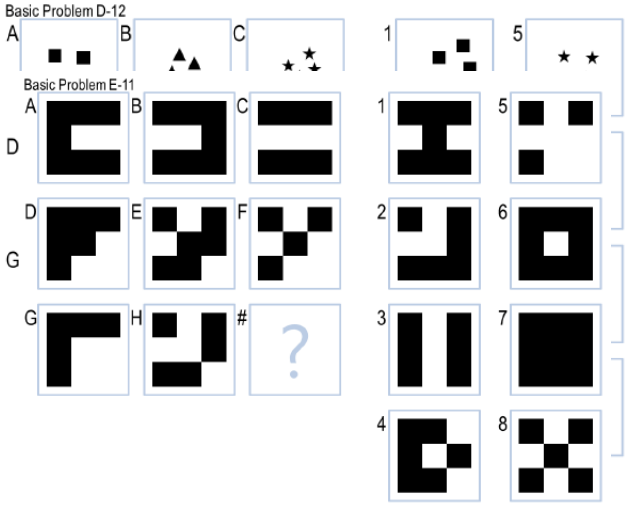


Figure 1— Basic Problem B-10, Answer: 3

This problem was difficult to generalize to Test B-10 utilizing solely horizontal relationships. Instead, the answer was found to be more consistent by determining that the black square was added going vertically. To solve, the specific heuristic the agent uses is minimizing the vertical dark pixel ratio between pair (A, C) and (B, answer choice).

The greater success in this case is being able to skip ambiguity in the voting process by directly assuming a pattern relevant to the specific problem and analyzing the case accordingly. The agent applies these one-off approaches to only the relevant problems, allowing for an implementation that balances developer knowledge of problems as well as a generalized heuristic methodology.

3.2 Basic Problem D-12 | Pixel Ratio Subcase Success



3.3 Basic Problem E-11 | 'Strong Match' 3x3 Solver Subcase

Figure 3 — Basic Problem E-11, Answer: 5

The agent performed very well when the problems mapped to one of the multiple binary transformation tests. Here the agent computed the binary AND of each of the first two cells in each row. The agent then determined successfully that this was equivalent to the last cell in each row. Lastly, the agent then predicts the answer choice to be the bitwise AND of cell G and H. This prediction is then matched across all answer choices to ultimately yield 5 as the most likely answer. The histogram match returned ~99% in similarity score. Overall, the strong match mechanism performed well in both sets B and E before the addition of pixel heuristics and developer-determined problem-specific heuristics.

4 AGENT STRUGGLES

4.1 Basic Problem B-06 | Overfitting and Ambiguous Voting

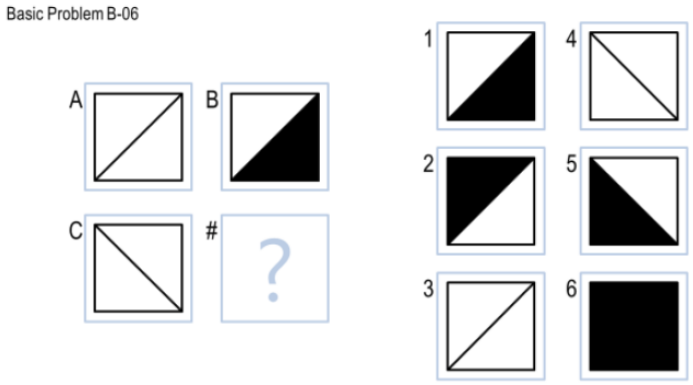


Figure 4 — Basic Problem B-06, Answer: 5

Although the test set is invisible to the developer, some commentary can be made as to the weaknesses in the agent's design. Largely, the agent's overall hierarchy of solving methods can be seen as follows: recorded cases (the 'Basic' case dictionary), expected general transforms and voting mechanisms (the scoring mechanism in the 2x2 solver and the strong match mechanism in the 3x3 solver), and lastly problem-specific heuristics. With respect to Problem Set B, the agent solves all Basic problems with the case dictionary and 11 Test problems. Test problem B-06 could not be solved by any existing mechanism within the agent. Therefore, the agent is overfit to patterns it expects and the developer's capacity for determining the pattern. I tried many approaches in my attempt to generalize the solution for this problem but some key ambiguities found were:

1. No fill operation present in the 2x2 solver.
2. Comparing A to C vertically yielded ambiguous results as horizontal reflection and vertical reflection both yield the same result. Thus, the agent cannot determine which to apply to B. Ultimately, the agent struggled to choose between answers 1 and 5.
3. I could not assure myself whether the pattern was 'reflected' and fill, or 'fill the bottom triangle', etc.

4.2 Basic Problem D-06 | Lack of Shape ID/Reliance on Generic Pixel Ratios

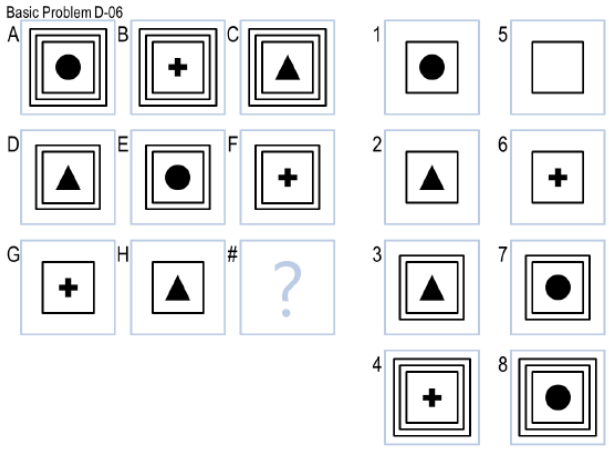


Figure 5—Basic Problem D-06, Answer: 1

The agent fails to solve this problem using the general distance minimization of row IPRs. Although the pixel ratios can sometimes capture general changes or distributions of shapes as seen in D-12, it struggles when the relationship is multifaceted. Here, the visual pattern is clear—there are three shapes in a row, and the number of outer squares is reduced by one going down the matrix. The agent has no shape identification measures, nor can it capture such a relationship. Thus, the pixel heuristics, whilst still successful in many cases, are also a form of guesswork and fail to solve more intricate patterns or combinations of patterns. Were the agent to have shape identification, the strong matcher could have been used to always solve this type of problem.

5 AGENT DEVELOPMENT

5.1 High Level Design Iterations

Initially, the agent design was informed by Kunda et. al's affine solver for Raven's Progressive Matrices problems (Kunda, 2013). Much of the original methodology remains-the agent tries a set of predefined transforms, generates a prediction and then finds the best similarity. The paper, however, uses a Tversky similarity metric and/or the sum of squared differences (SSD). Initially, the agent relied solely on the SSD for the 2x2 solver. Then histogram matching for the 3x3 problems as the SSD became inconsistent in my findings. Lastly, introducing intersection over union seemed to provide stronger consistency across all milestones.

Going from 2x2 to 3x3 problems, I also expected to reutilize certain heuristics or methods, but I found a custom voting mechanism and approach to work better for each. In fact, I found this to nearly be the case per problem set. I.e., though work was done on Problem Set D and E in tandem, I found them quite different from each other in conglomerate patterns and difficulty—thus I focused heavily on E as it matched much more closely to binary transforms.

Lastly, rather than trying to improve the generalized methodology of the agent, I found it more straightforward to deduce patterns in basic problems myself and to directly implement them, successfully yielding many passes on Test problems.

Overall, the agent certainly fits the description of an ever-expanding set of heuristics farmed/gleaned by the developer.

5.2 Low Level Design Iterations

Originally, the agent solely used Pillow, however, much of the later revisions required multiple visual heuristics to be implemented. Thus, I remade the agent to utilize OpenCV and its operations on binary images. Consequently, to reduce runtime due to the plethora of image operations, much of the reused image attributes and features were stored in a dictionary alongside the image for immediate retrieval.

6 AGENT-HUMAN COMPARISON

6.1 Human Similarity

I think the agent is quite similar to how I might approach it with respect to learning by recording cases and the application of visual heuristics. I examine rows, columns, diagonals, combinations of the prior, all to see if the same pattern holds across them. These patterns might be colors, rotations, reflections, and/or pairwise/triplet-wise comparisons. If a pattern seems repeated, I can feel confident that the answer will do the same. Like myself, the agent added heuristics as I found them as well, originally only relying on basic affine transforms, and later looking for symmetries or binary transforms. Ignoring implementation difficulty, if I were to translate all of the visual heuristics I found, I think the agent would perform just as well as I might on the RPMs.

6.2 Human Difference

I do not utilize the pixel ratio mechanisms whatsoever when solving these problems. It is evident as to why—the success of the pixel ratio mechanisms is to capture the variations between images and shapes as changes in the dark pixels and their location. As a human, I have no issue with simple shapes and cross-matrix patterns. The agent, however, must approximate these and often times does not succeed in doing so. Consequently, the agent has few, if any, full matrix analysis methods—instead opting heavily for row-wise relationships and rarely column-wise operations. Furthermore, the similarity metrics are an approximation as well—intersection over union requires exact pixel comparison, whereas I could clearly compare shapes in an image without requiring the orientation of the images to be exact or perfectly binary. Additionally, the agent does not do a more involved job of eliminating answer choices as a whole. As a human, it is often subconsciously clear why a certain answer choice may not be a valid candidate. I usually need to choose between one or two answer choices, whereas the agent needs to consider all answer choices clearly. It will also choose some answer regardless if a good match is found due to the lack of a lower bound threshold on similarity. I.e., an answer choice with 0.1 similarity may be chosen if there is no better choice.

6.3 Overall Designation

Ultimately, I believe the agent to be a weak approximation of human intelligence and approach. The agent uses case retrieval to solve Basic problems, a human-like task if not for the element of using pixel ratios to do so. In the event of certain problem types, it attempts to leverage commonly seen patterns and is largely successful in doing so—it simply is limited in comparison to what a human is able to perceive in congregate. The agent also tries to choose the most likely answer, but isn't often the most humanlike in its approach. Sometimes, it uses the scoring mechanism as in the 2x2 solver, or determining a strong heuristic relationship between rows in the 3x3 solver. Otherwise, the agent will then use fuzzy pixel heuristics to make an educated guess. Therefore, the agent cannot learn. Given a new problem, I can determine a solution on the go. This is the key difference; humans can often come up with a new pattern when provided with new information, whereas the agent cannot.

7 REFERENCES

1. Joyner, D., Bedwell, D., Graham, C., Lemmon, W., Martinez, O., & Goel, A. (2015). Using Human Computation to Acquire Novel Methods for Addressing Visual Analogy Problems on Intelligence Tests. *In Proceedings of the Sixth International Conference on Computational Creativity. Provo, Utah.*
2. Kunda, M., McGreggor, K., & Goel, A. K. (2013). A computational model for solving problems from the raven's progressive matrices intelligence test using iconic visual representations. *Cognitive Systems Research*, *22-23*, 47-66. https://doi.org/10.1016/j.cogsys.2012.08.001
3. McGreggor, K., Kunda, M., & Goel, A. (2014). Fractals and ravens. *Artificial Intelligence*, *215*, 1-23. https://doi.org/10.1016/j.artint.2014.05.005