PROJECT-3 REFLECTION [Vinodh Kumar Sunkara, Yahoo Inc.]

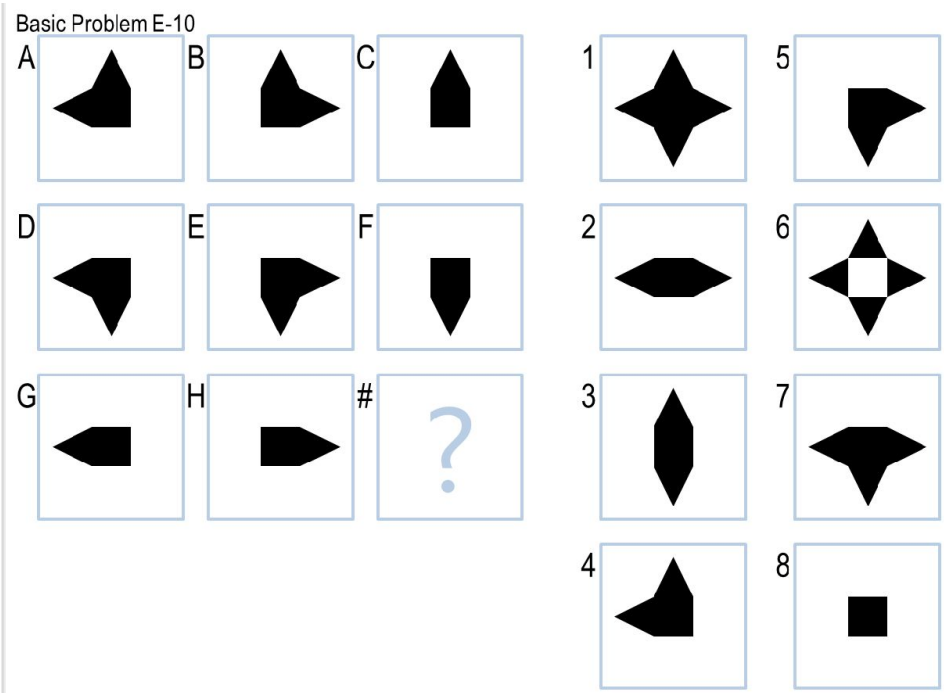
Introduction

Raven's Progressive matrices have been used to test the intelligence of AI agents for a long time. Implementation of AI agent should definitely replicate human cognition while solving Raven's Progressive matrices. It's important to have an insight on how human solves each problem before training the AI agent to solve it. Humans apply different methods to recognize shapes and patterns in the problem. Some methods can lead to answer very confidently and a few others need options to pick the right choice that fits in pattern. There are some human reasonings that can lead to false positives also if the reasoning is not very tight. Analogous to this, AI agent also acquires a set of rules that are applied intelligently to reduce false positives. The less confident rules are applied after all the confident rules are explored on a given problem. The most confident rules submit the solution with either a horizontal or vertical computation, without waiting for a cross check of solutions obtained in both directions.

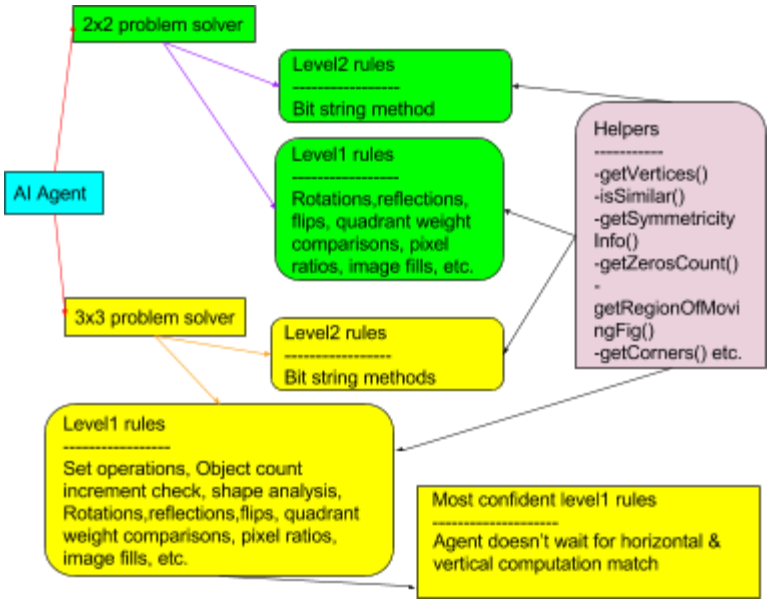
AI Agent construction

Agent has 2x2 and 3x3 problem solvers. Each problem solver has two levels of rules. Level1 rules are applied horizontally and vertically to decide upon an answer. Horizontal application is to observe transformation undergone to produce C, F in A-B-C and D-E-F sets respectively and produce answer image '#' using same transformation on G,H images. Vertical application of rules realizes patterns across A-D-G, B-E-H sets and apply on C,F to generate the answer image '#'. Level2 rules derive a common pattern considering the whole image set A-B-C-D-E-F-G-H at once and derive the answer that best completes the pattern. I designed the rule set that's applicable both horizontally and vertically by passing a direction parameter to deal with right image ordering. While the general case in level1 rules is to find solution horizontally and vertically to submit accurate answer that matches in both directions, I've a list of most confident level1 rules. If an agent finds solution using any of these most confident rules in either of horizontal/vertical directions, it submits the answer without waiting for a cross check of horizontal and vertical computations.

An example of a level1 rules application is in below problem:

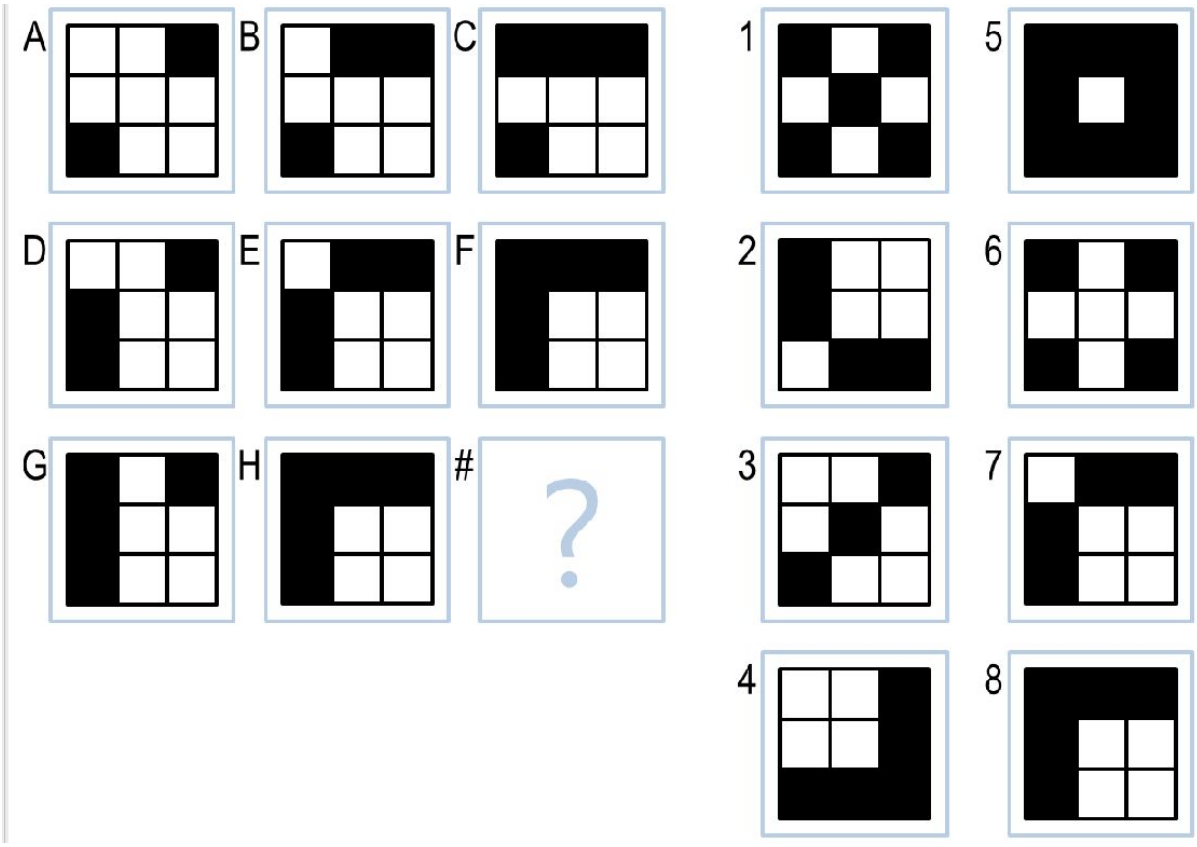


The level1 rule that can observe the pattern of logical OR operation in A-B-C, D-E-F sets can be used in same way on vertical sets A-D-G and B-E-H. In this case, the rule agrees upon same answer in both horizontal and vertical direction. In some other problems, it can generate an answer in one of the two directions and some other rule is applied in the other direction to end up on common answer. My solvers finalize an answer only if both the horizontal and vertical direction ends up in same answer through any level1 rule applied individually. Since, level2 takes all images at once into consideration, it's ordered high in priority. Level1 rules are touched by an agent only if level2 can't come up with a solution. Even within the level1 rules, there is a priority ordering where most confident rules are first to be applied. Also, there is a set of most confident level1 rules. The rule solving above example is in the list of most confident rules. Agent doesn't wait for vertical/horizontal computations once it finds an answer using this most confident level1 rule horizontally/vertically.



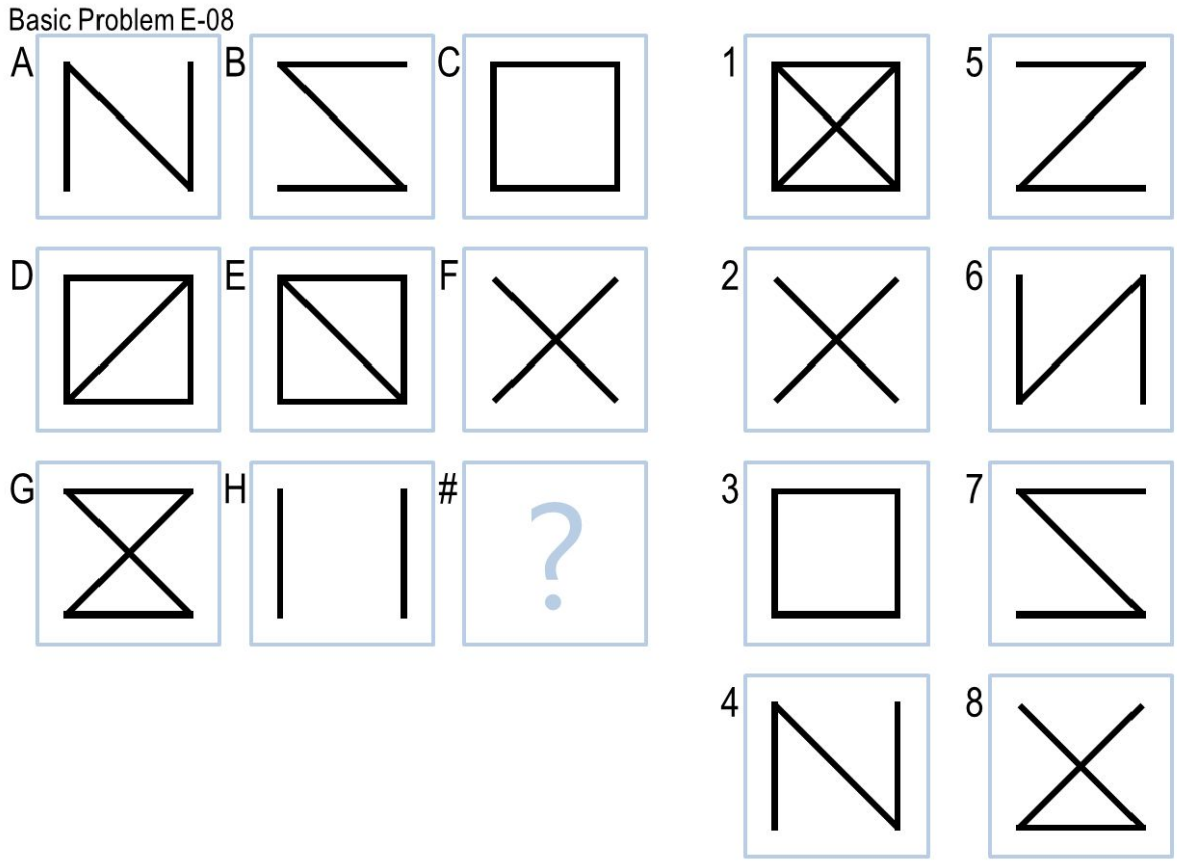
Methodologies

An example of a level2 rule used is for the below problem. The bit strings are generated for all figures and the missing bitstring is derived from the bit patterns. The corresponding image for the generated bitstring is picked from options as the answer.



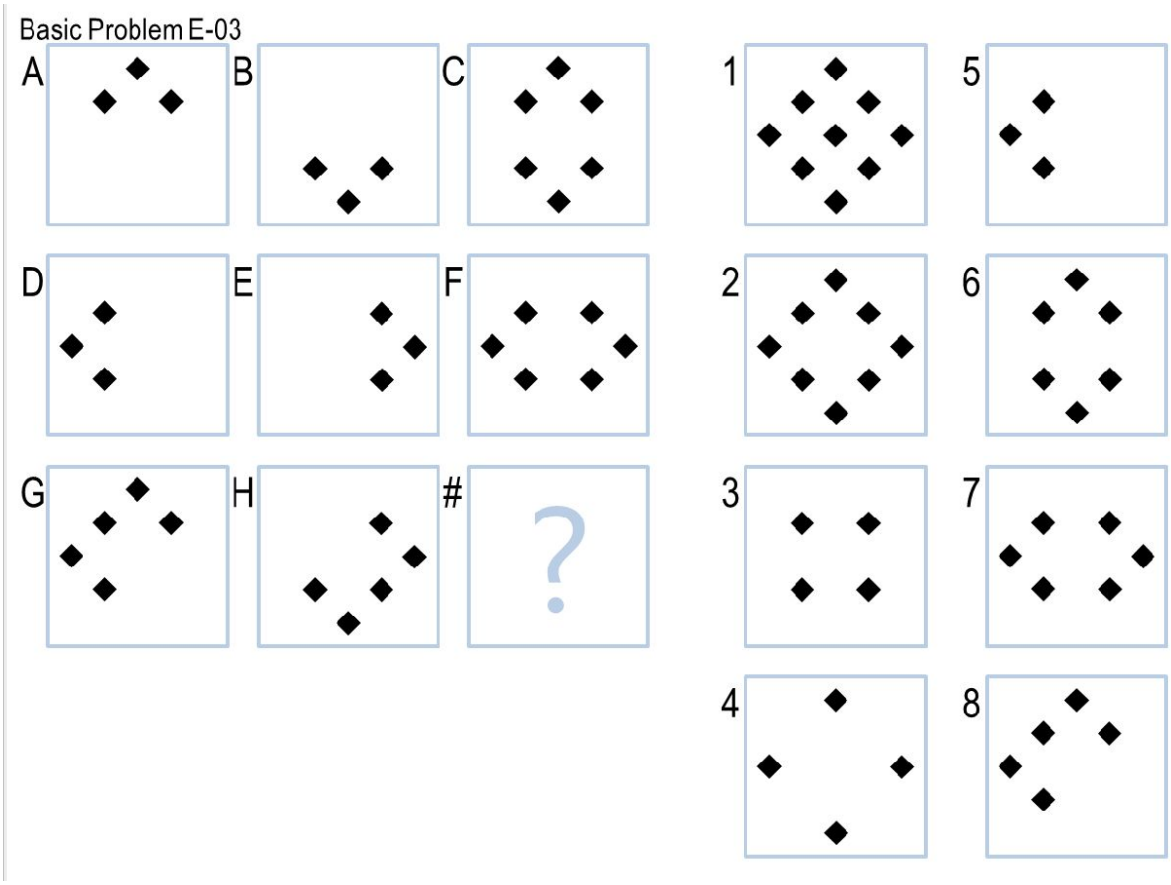
We can very well extend the same rule to solve similar problems with different analysis on bit strings. A portion of bit string is unchanged in all figures and it's borrowed to answer bitstring while rest of it derived using different techniques like maximum number of 1bit or 0bit, and other bit manipulations.

Level1 rules explore different methodologies. One sample method is using set operations like in this problem:



This problem is solved by simple XOR operation between the images. The pattern is learnt from given images and then right answer is picked from options. It's similar to above problem under AI agent construction which uses logical OR operation. I implemented my own version of XOR operation on image pixels with a permitted error percentage. The realized XOR pattern is applied on different answer options to decide upon the answer.

Another similar problem below has the images superimposed to finish the pattern. It's easy for a human to superimpose visually and find the pattern. AI agent needs a stronger logical computation to visualize this. I used 'multiply' operator where figure A and B multiplies to give C. Same as D, E multiplication forms F. This pattern is applied on G, H to obtain right answer. The same logic applies in vertical direction also to realize A-D-G, B-E-H patterns. My agent agrees upon option 2 using same rule in both horizontal and vertical direction for this particular problem. This ensures my agent's accuracy in picking right option for problems like this.



Helpers

There is a big set of helper methods I designed that are used in both 2x2 and 3x3 solvers.

A few helper methods are described below:

- getVertices: To get vertices count
- isSimilar: Takes a percentage parameter to compare image with assumed error rate
- getSymmetricyInfo: Returns an object with keys describing horizontal and vertical symmetry and also if horizontal and vertical dividers for image has any dark spots
- isRhombus: This method says if the figure is a rhombus
- getTopLetCorner: returns the top left corner of the image
- getBottomRightCorner: returns bottom right corner of the image
- getRegionOfMovingFigure: This method is used to enumerate and give a region number if figure is moving from top left to bottom left along the boundary.
- getDiagonalShadeType: Gives an object that describes if the diagonal exists and It's from top left to bottom right or top right to bottom left aligned.
- getRotatedImage: rotation with customized angles
- getReflectedImage: reflection across customzied axis
- getFlippedImage: Flip images along different axis etc.
- checkXored: checks if two images are xored to form third image

Performance

My AI agent is highly performant mainly because of the different factors - segregation of rule sets into level1 and level2, intelligent ordering of rule set within the levels, formulation of a most confident ruleset within level1. It takes on an average around 45ms per RPM. It needs this time to process different rules in horizontal and vertical directions to come up with an accurate answer most of the cases. It takes around 25 RPM for problems that are solved by most confident rules, which case agent doesn't wait for cross check in other horizontal/vertical direction. My agent currently solves 100% basic problems in sets B, C, D and E. The scores are as follows: 12/12 basic B, 12/12 basic C, 12/12 basic D, 12/12 basic E, 5 correct in C ravens, 5 correct in C test, 6 correct in B test, 4 correct in B ravens, 5 correct in E ravens, 5 correct in E test, 3 correct in E ravens. The known issues that might cause a few errors from test set are described in below section.

Risks

Basic problems have already proved the accuracy with 100% correct answers. There is an error rate I assumed on figure comparisons. While figure generation or initial figure comes up with a 0.5% error, after manipulations like multiplication, addition, reflection, etc. on image increases the error rate. While assuming this error rate to as high as 3 percent in few cases, some false positives can occur. The tight bound of horizontal and vertical rule agreement can eradicate this to the most extent.

Conclusion

The whole AI agent architecture and functioning replicates human cognition and it does function as expected by me to the most extent. But this doesn't make me happy completely! The issue is that I feel it's more a limited replica of myself than an artificial agent on it's own. All the knowledge it has is installed by me which is not super complete. Once it encounters a brand new problem, it can't learn or do anything on it's own except to flunk. However, I've trained it great enough for it to tackle many new problems efficiently. I've incorporated many standards techniques and methodologies through rulesets. My agent can quickly solve all those problems of similar models I trained, much efficiently in any volumes avoiding my man power. This gives a very positive sense of satisfaction seeing the Agent so profound in architecture and accuracy by the end of this course. Glad to have all these tens of hours spent on the hundreds of lines of code for a fully functioning agent now.