Georgia Institute of Technology

Online Masters of Science in Computer Science

CS7637: Knowledge Based Artificial Intelligence – Cognitive Systems

Professor Ahsok Goel

Project 3 Reflection

Rodrigo De Luna Lara

Spring 2017

Overview of the projects

For project 1, the approach was mixed, it relied on both verbal and visual representations of the problems to solve the 2x2 RPMs, a semantic network was used to solve some of the problems, determining the transformations between elements of the matrix by looking at the changes in their attributes. Project 2 still used the semantic networks, but to a much lesser extent than Project 1, the focus on Project 2 was to leverage the functions available in PIL to solve the problems (i.e. functions for flipping, mirroring, etc.). Project 3 enhances the image processing capacities of Project 2, but the approach remains essentially the same.

Project 2 used PIL's implemented methods only, so it was limited in some of its capacities. Project 3 includes a component labeling algorithm by Wu Kesheng, Ekow Otoo and Kenji Suzuki[1] which is used for blob detection in some problems. Additionally, while Projects 1 and 2 used the root mean square error as measure of image similarity, in this project a pixel similarity score algorithm by Tony Diloreto[2] was used instead, while this simplified some things it remains one of the weak points of the agent, as this kind of similarity scores are as not robust as some more complex ones, like the Structural Similarity Index Method (SSIM). Additionally, a Harris corner detection algorithm by Kai Jiang[3] was tested, but it was not deemed fast or robust enough to be used in the project.

Table 1 outlines the main aspects of each project.

Project 1	Project 2	Project 3
• Available representation: verbal and visual • Type of problems: 2x2 • Number of training problems: 12 • Agent's knowledge representation: verbal and visual • KBAI topics used: semantic networks, generate and test, frames • Visual processing techniques used: flipping and mirroring • Implementation difficulty: simple	• Available representation: verbal and visual • Type of problems: 3x3 • Number of training problems: 12 • Agent's knowledge representation: mostly visual, slightly verbal • KBAI topics used: semantic networks, generate and test, frames, logic and planning. • Visual processing techniques used: flipping, mirroring, image operations (addition, subtraction), cropping, offsetting. • Implementation difficulty: moderate	• Available representation: visual only • Type of problems: 3x3 • Number of training problems: 24 • Agent's knowledge representation: visual only • KBAI topics used: generate and test, logic and analogical reasoning • Visual processing techniques used: flipping, mirroring, image operations (addition, subtraction), cropping, offsetting, logical operations (and, or, xor), connected component labeling. • Implementation difficulty: hard

Table 1: Project overview

It can be seen that the complexity of the project increased linearly, and that visual processing became increasingly more complex as well. The design of the AI agent for each subsequent project was leveraged, as well as the knowledge on the KBAI topics used for each one. There was a big leap in difficulty from project 2 to 3, as well as a duplication in number of training problems. The code for project 1 is standalone, while the code for project 3 is an expansion upon the code of project 2, due to the type of problems (2x2 vs 3x3).

Changes in problem set & agent's reasoning

Between projects 1 and 2 the problems changed from 2x2 RPMs to 3x3 RPMs, that introduced additional difficulties just because of the number of elements in each problem, the additional possibilities for relationships between elements of the matrix and the number of possible responses. Between projects 2 and 3 the type of RPM problem didn't change, but the complexity increased significantly. In project 2 the relationships between elements of the matrix were linear, the agent only required to analyze the matrix row-wise or column-wise to determine a pattern, while in project 3 some of the problems had non-linear relationships that required looking at the matrix as a whole and determine the response based on the disordered information available.

For example, for Basic Problem D-01 (Figure 1) it can be easily seen that the figure in the first column repeats for the other columns, this is a case of "ordered information", which is nicer for an AI agent as the focus is in pattern recognition. If one were to use a thermodynamic analogy, the entropy of this kind of problems is low, and it is easier to establish scripts or sequences to detect a pattern from them. This project introduced high entropy problems, such as Basic Problem D-09 (Figure 2), in which a human can easily identify the pattern from the disordered information, but programming an AI agent is much difficult.

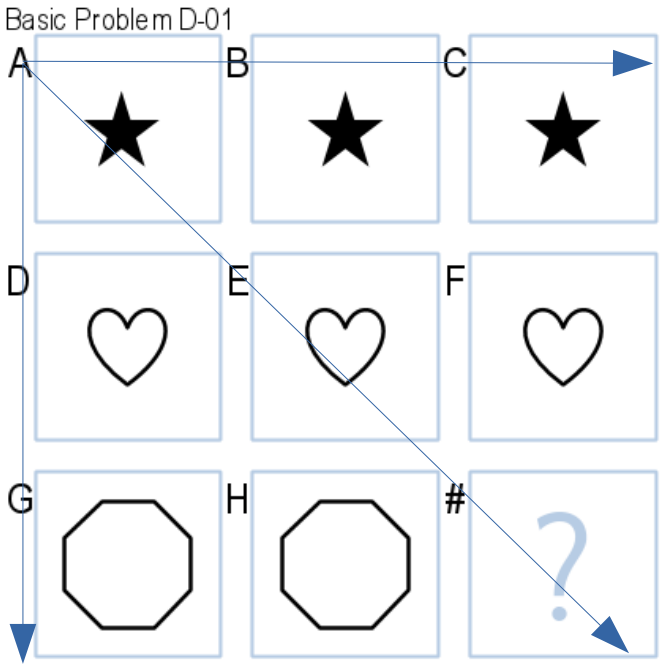

Figure 1: Low entropy problem

It can be seen in Figure 1 that a linear relationship can be established in the three directions shown by the arrows within the matrix of the Raven's problem. Such kind of relationship could be established in at least one direction for all problems in Basic Problems A and Basic Problems B. The agent is expecting such a relationship to be able to detect the pattern, which is why when this rule is broken the agent is not so "smart" as a human.

For example, on Basic Problem D-09 a human can easily see that without respect to the order of the elements in the problem there are 3 different shapes, a circle, a square, and an octagon, and that one of the figures of each group of shapes is the "sum" of the other two.

The missing element in the problem is, by simple visual inspection and counting of the shapes, an octagon, and considering that each of the 3 figures for each shape has the three different inner stripes (one of which is the sum of the other two), it can be easily deduced that the missing shape is an octagon with a vertical stripe.

The difficulty with this problems is that there is no nice ordering or linear relationship, as it can be seen in Figure 2, the relationships become fairly complex to detect for an AI agent, showing the great capacity humans have of getting to a conclusion without the need of having nice, ordered data

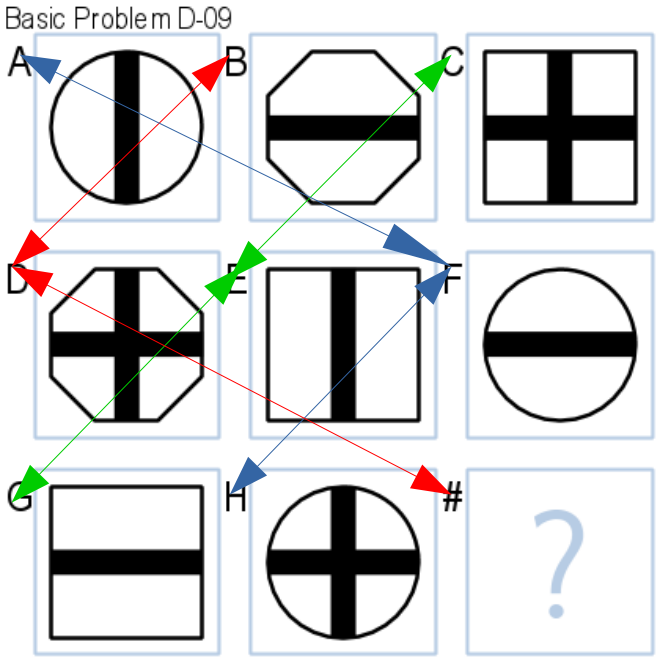

Figure 2: High entropy problem.

The input representation did not change with relation to Project 2, which already was mostly visual-oriented. Backwards compatibility is ensured because the 2x2 RPM problems run on separate code than the 3x3 RPM problems. And the same code for 3x3 RPM problems for Project 2 is the basis for the code for Project 3, so the agent has the same capabilities related to the last 2 projects.

The issue of Bonnie compatibility for the submission was also present for Project 3, but related to the Basic Set D and Basic Set E, these problems were fixed by replicating the Numpy and PIL versions that Bonnie uses. There were still some errors that weren't present in local development, but the troubleshooting yielded acceptable results.

This gives some insights into human cognition, in this case the agent was created assuming a specific environment, and trained in that environment. When the agent was faced with a different environment it made some mistakes, which is what would most definitely happen with a human. For example if I were to prepare for a calculus exam with a scientific calculator that allows to numerically calculate some derivatives and on the exam the professor mentioned that no calculators are allowed it is highly likely that I make some mistakes if my solution process was based on having a scientific calculator. The environment in which the agent was created has the same effect, any assumption by the programmer can result in mistakes being made by the agent because of the different environment.

Agent capabilities

The reasoning approach the agent uses did not change, the agent is still trained to look for patterns based on the Basic sets, so it is purely based on the available information. The cognitive feedback was used and it did not detect overfitting, although some of the logic programmed into the agent looks for very specific conditions and patterns in the matrix in order to solve the problems. As mentioned before, one of the challenges was to solve problems in which there is a pattern between elements of the matrix but they are not in order so that it is not so easy to determine by an AI agent.

To attempt to solve the new problems, additional visual capabilities had to be implemented for the agent. For example, for Basic Problem D-12, there was already some code for Basic Problem C-03 (both problems are shown in Figure 3), but that was based on the verbal representation of the problem, simply checking if the shape attribute was the same in the expected linear relationship and then getting the correct number of elements for the generated output.

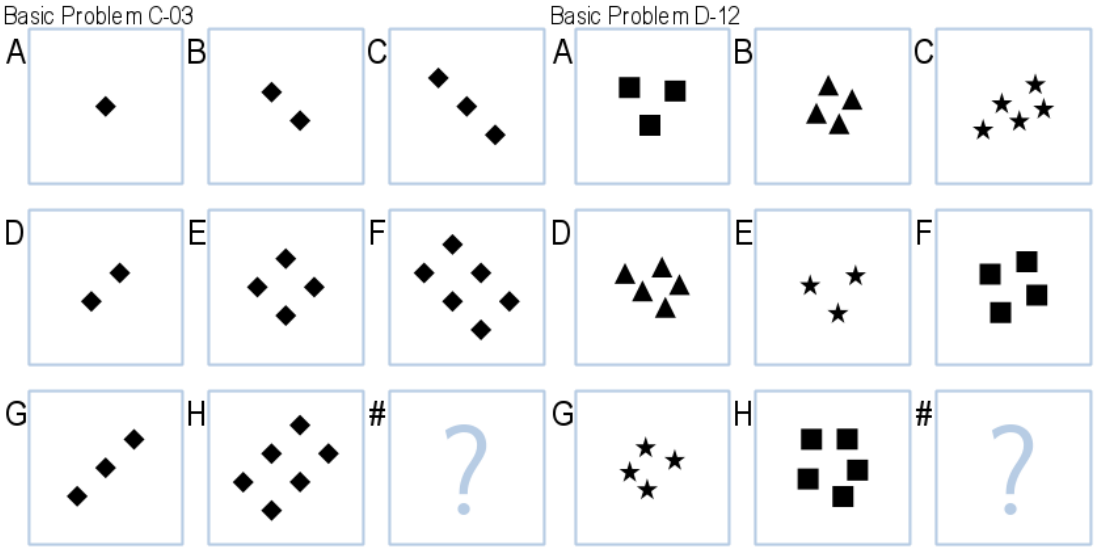

Figure 3: Problems C-03 and D-12

The problems look similar but are very different, the AI agent suddenly doesn't have the verbal information, so it doesn't know how to see if it's a square or a triangle or a star, the problem is also in disorder now. This problem was what required the implementation of the blob detector, to allow the agent to find the contiguous pixels that form a shape.

The agent is shape-agnostic (as the corner detection was not implemented), so as a baby would do the agent takes the first blob it can see and checks if the rest of the shapes match, like a shape sorter toy (like the one in Figure 4). Then it counts the number of shapes, and in the case of Basic Problem D-12, looks at the diagonal, determines that a shape is missing and that both A and E have 3 elements, looks at the possible solutions and determines that the response is the one containing 3 triangles.

So the AI agent is very much like a baby regarding its visual processing, it doesn't know what the shape is, but it can compare it with the rest to determine if they're similar or not, and it is also intelligent enough to separate the individual shapes by using connected component labeling.

Figure 4: Shape sorter toy[4]

The agent was already using visual processing, albeit very simple, since project 1, and project 2 used mostly visual processing. For Raven's Progressive Matrices, visual representations are more natural and more closely related to what humans actually do to solve them. Going back to the baby example, the baby doesn't need to know that a shape with 3 corners is called triangle, it can just look at it, look at the slots and repeatedly try to put it inside, until it succeeds.

Therefore, the visual approach seems more natural than a verbal approach. Additionally, if the first two projects had been made with a mostly verbal approach, project 3 would have required even more advanced visual processing to convert the shapes to their verbal representations. That would require full shape recognition, as well as a way of determining the relative position of a shape with respect to other, making the whole thing extremely more complex. Hence why the agent is visual.

A lot of the capabilities introduced to the agent in this project are related to image logic operations, many of the problems in Basic Set E could be easily solved by making an AND, OR or XOR between the images. For example, in Basic Problem E-03, the third column is the result of the logical OR of the first two. Likewise, in Basic Problem E-10 the third column is the result of the logical AND of the first two columns. Finally, in Basic Problem E-08, the third column is the result of the logical XOR of the first two columns (all of the problems can be seen in Figure 5).

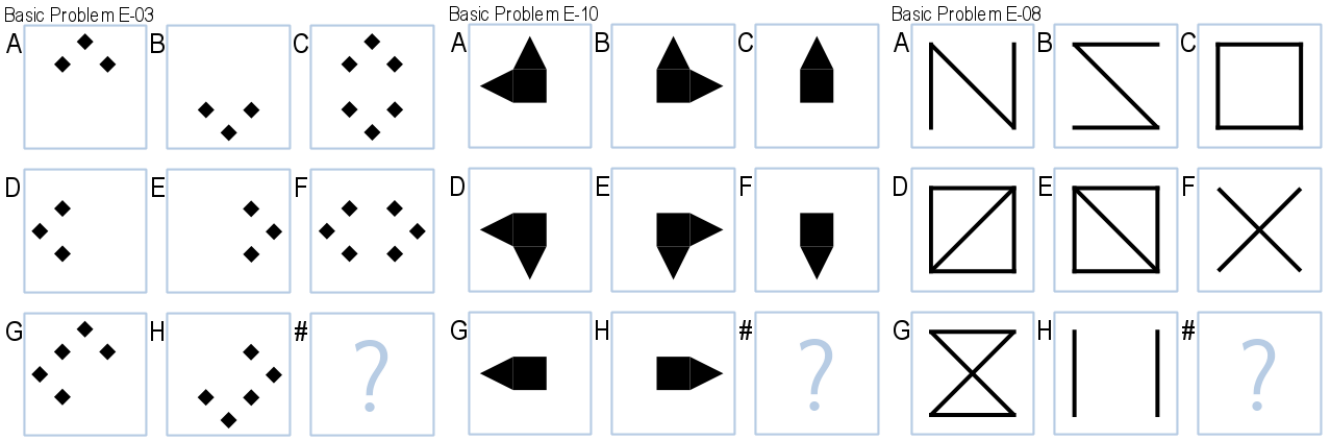

Figure 5: Examples of logical operations between columns. From left to right: logical OR, logical AND, logical XOR.

Implementing logical functions between images, as well as more complex image processing in the form of blob detection allowed to tackle most of the challenges in the new problems. As explained before, opting for visual processing in these cases really simplifies the processing, the equivalent of the image logic operations would be indescribably hard to process verbally.

Agents limitations

The code for projects 1 and 2 had just minor updates, and the focus for this project was to maximize the accuracy on sets D and E, so not much focus was put on rectifying previous version's mistakes However, based on the performance of the agent it seems that most of the errors the agent makes are still being dragged along into this project, and there are new mistakes the agent now makes based on the new problem set. The design purposedly skips some types of problems to avoid overfitting, the 4 problems that are skipped in the training set are Basic Problems D-08, D-09, D-10 and E-12 (shown on Figure 6).

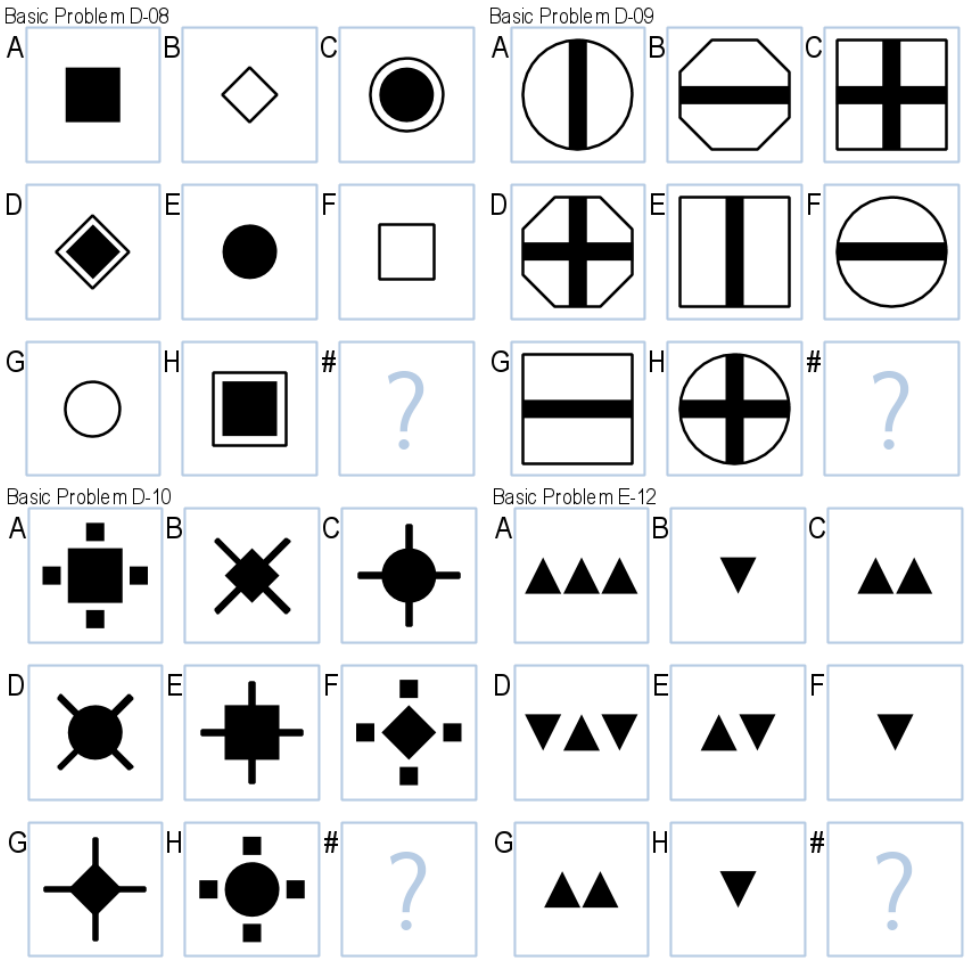

Figure 6: Skipped problems for Project 3

This is a big difference with respect to previous projects, in which none of the types of problems were skipped on purpose. The reason they were skipped, as mentioned before, is to avoid overfitting, as no simple logic or script could be developed for them without being too specific or having too low accuracy and repeatability, so the agent simply ignores these kind of problems with very high entropy, in an attempt to minimize the incorrect responses, as skipping a response is the best choice score-wise.

The fundamental problem with this type of problems is that the agent is more fit to find linear relationships in the Raven's matrix, and if no simple manner of solving those type of problems is found, then perhaps it is better for the agent to remain fast and simple than slow and complex. When trying the Harris corner detection to determine the shape, the performance was of about ~15-20 seconds per shape, so for example figure C on Basic Problem D-12 (Figure 3), would have taken around 2 minutes to process.

At first, the connected component labeling algorithm wasn't used. Coming from a more machine-learning oriented programming background, k-means clustering was used instead to determine the number of shapes, by going from 2 to 10 clusters and using the mean silhouette for each k to determine the optimum number of clusters which corresponded to the number of blobs in an image. That approach was significantly slower than using connected component labeling, and also significantly more computationally complex. Therefore, to keep the agent quick and simple the connected component labeling was chosen instead.

So the limitations of the agent have to do more with complexity in the visual processing and in it's logic. Due to the lack of previous experience in computer vision and even Python programming, the agent is limited to more simple approaches, while remaining relatively simple and still having an adequate performance. The biggest limitation is the visual processing capabilities of the agent along with it's design mostly based around linear patterns.

Agent performance

The performance for the agent is summarized in Table 2.

	Project 1	**Project 2**	**Project 3**
Training Set	Basic Problems B	Basic Problems C	Basic Problems D Basic Problems E
Correct (local)	100%	100%	83.33%
Incorrect (local)	0%	0%	0%
Skipped (local)	0%	0%	16.67%
Correct (Bonnie)	100%	91.67%	83.33%
Incorrect (Bonnie)	0%	0%	0%
Skipped (Bonnie)	0%	8.33%	16.67%
Testing Set	Test Problems B	Test Problems C	Test Problems D Test Problems E
Correct (Bonnie)	83.33%	66.66%	50.00%
Incorrect (Bonnie)	16.67%	16.67%	20.83%
Skipped (Bonnie)	0%	16.67%	29.17%

Table 2: Performance for all 3 projects

The evaluation metrics were kept with respect to previous projects, reporting the accuracy for both training and testing sets for all 3 projects. It can be seen that there is a declining trend in the accuracy, with increasing number of incorrect and skipped problems between projects 2 and 3.

The agent is not generalizing well enough, as evidenced by the results. Additionally, not adding logic for some problems (Figure 6) also impacted adversely the performance of the agent on the testing set. Table 3 shows the breakdown for sets D and E for training and testing sets. As it can be seen, the driver for the largest amount of errors the agents makes is skipping the 3 cases for Basic Problems B, as the performance in Test Problems E is much better than for Test Problems D.

	Basic Problems D	**Basic Problems E**	**Test Problems D**	**Test Problems E**
Correct	9	12	4	8
Incorrect	0	0	3	2
Skipped	3	1	5	2

Table 3: Specific results for sets D and E

The image logic implemented seems to generalize better than what was added to the agent for Basic Problems D. The agent is still very specific in some parts of its design, which is evident in the performance shown in Tables 2 and 3. Cognitively speaking, the agent was trained to detect very specific linear patterns in the data, when faced when unseen cases in which most likely these relationships are not met the agent is either unable to find a response or it chooses an incorrect one, assuming that the problems are set up in a similar way.

The agent doesn't learn from it mistakes, and having no way of looking at the test problems it is very difficult to determine why the agent is making the mistakes it is making. The fundamental issue is the way the agent is designed, as mentioned before. The agent would need to leverage more complex KBAI topics like learning by correcting mistakes or incremental learning in order to have more generalized models.

Agent's relation to KBAI

As seen in Table 1, the main KBAI concepts leveraged in this project are generate and test, logic and analogical reasoning. With respect to the last project, the concept of analogical reasoning is the addition to the KBAI topics used, while semantic networks and frames were dropped in this project. These last 2 topics were dropped because the verbal representation was dropped, and the semantic networks were created by determining the transformations between verbal attributes; the frames were used to hold the attributes retrieved from the verbal representation.

Generate and test is still widely used, as many parts of the code generate a prototype solution from the available frames when a pattern is detected, so it is still widely used. The prototype solution is generated and compared against the possible solutions, the one with the highest similarity to the prototype solution is then chosen as the correct response.

Logic and planning are used to detect the patterns in the problems. Logic is used to ensure that the conditions for the pattern are found in the problem, it is also directly used in the additions to the AI agent that perform image logic (OR, AND and XOR, like shown in Figure 5), so in this project logic is used directly with the images by the agent to determine if a pattern exists and also to generate the solution.

Analogical reasoning was widely used in all projects, but is directly introduced as a KBAI topic until this project. RPMs are all based on analogical reasoning, a human or an AI agent must look at the problem, determine how the items are related and use the analogy retrieved from the data to determine which of the possible solutions completes the analogy. For example, for Basic Problem D-06 (Figure 7), the analogy can be established row-wise, column-wise or even diagonally.

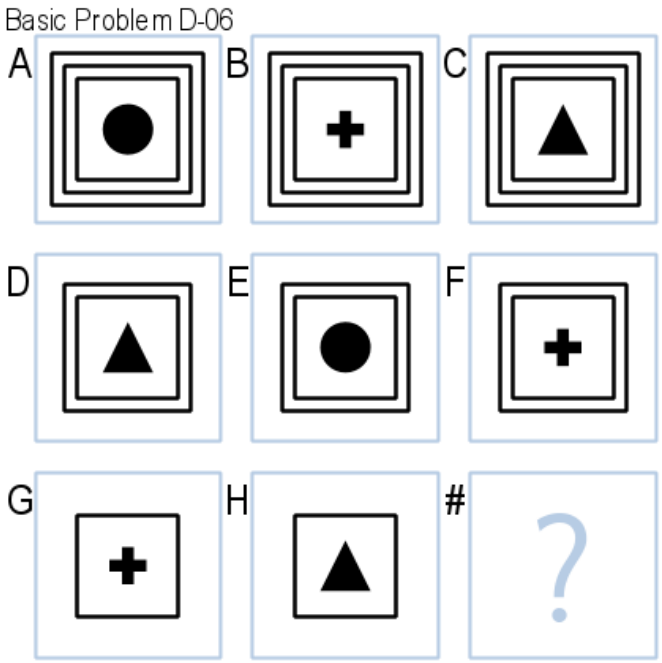

Figure 7: Basic Problem D-06

Row-wise it can be seen that A:B :: D:E :: G:H and that B:C :: E:F, so the logical conclusion is that H is to the response as E is to F, and we can see that the shape is changing and the inner figure is changing in these cases. So we as humans can uses this analogy to find the response. The AI agent should do the same thing, and so analogical reasoning is used by the agent to detect the patterns and know what is happening between elements in the matrix to generate the solution correctly.

The visual processing the agent does is also based on analogical reasoning, being shape-agnostic the agent must find analogies between the shapes by comparing them visually to be able to solve of the problems. The agent doesn't really care about the properties of the shape, like number of corners or if it's filled or not, it tries to fit the available analogies with the processed visual data to find a solution. Going back to the example of the shape sorter for a baby, the baby detects that if a shape looks similar to the slot and it fits then the same must hold true for the rest of the shapes.

Agent's relation to human cognition

In this project that the approach is entirely visual, it is more evident how human cognition works in solving this problems, and how we're certainly superior to any AI agent for tasks like solving Raven's Progressive Matrices. I may struggle a bit in some problems but I could easily solve most of them without much thought. Our brains are wired to find patterns easily, as well as using analogical reasoning to solve problems.

Humans have an advantage over AI agents for solving RPMs because we're able to make sense out of chaotic, disordered information, and we have built-in extremely efficient image processing. An AI agent must learn how to "see" things properly, and then again it's limited by how proficient its creator is with computer vision and programming.

Some relations to human cognition have already been presented in previous sections, specifically referring to the example of the shape-sorter toy. Such a task is trivial for adult humans, and even babies don't struggle that much in solving that kind of problems. AI agents are extremely limited in comparison to humans in that pattern recognition and visual processing do not occur naturally, they have to be developed.

The agent is more human like in this project than on previous ones because it is now using only visual processing, and as mentioned before, verbal processing is very unnatural and not human like for problems such as RPMs. By adding more visual processing the agent becomes more human like but it is still very limited, specially regarding the ordering of the information. The agent is unable to make sense out of information that is not presented in the expected way, and is unable to infer non-linear relationships from the elements of a problem because it can't just look at the whole thing and determine what goes where.

The AI agent for this project is like a not very smart baby, it knows how to solve some problems from experience but expects them to be in an ordered way, and it is also unaware of the properties of the shapes it is looking at for most of the solution. Nonetheless, the agent is still somewhat able to make sense out of a bunch of colored pixels arranged in certain ways.

This project has helped me understand how the human thought process may be sometimes chaotic and disordered but extremely efficient despite of that, I'm sure everyone enrolled in this class uses different logic when solving the RPM problems. The AI agents are a reflection of their creators, so they are as smart as the programming skills of their designer allows them to be. Each AI agent has a "brain" or "mind" of its own, and that's what makes them so interesting, that they can all be so diverse and still be able to solve the same problems with similar accuracy.

References

[1] Wu, K., Otoo, E., & Suzuki, K. (2008). Optimizing two-pass connected-component labeling algorithms. Pattern Analysis and Applications, 12(2), 117-135. doi:10.1007/s10044-008-0109-y – Code by Spencer Whitt retrieved from https://github.com/spwhitt/cclabel
[2] Diloretto, T. (2015). Image Similarity Module. Retrieved from https://github.com/diloreto/sameimages/blob/master/imagediff.py
[3] Jiang, K. (2011). Harris Corner Detector in Python. Retrieved from http://www.kaij.org/blog/?p=89
[4] Image retrieved from http://www.tagtoys.com/intelligent_toys/maple-shape-sorter.htm