Traci Fairchild
CS7637 KBAI
Assignment 3
September 13, 2015

Addressing Raven's Progressive Matrices Using Frames, Learning by Recording Cases, and Case Based Reasoning

Raven's Progressive Matrices

Raven's Progressive Matrices, referred to as RPM, is a group of non-verbal tests that are typically used to measure a person's reasoning skills. Reasoning is a skill commonly thought to be an adequate measure of a person's intelligence. The tests are a series of images in which one image relates to another in a certain way. The test taker is expected to analyze the relationship, in our example below, between image A and image B, looking for similarities or differences. And then when presented with another set of images, image C in our example, the test taker is expected to choose the image (images 1 through 6) that best completes the analogy of the first set. For example, a given RPM test would look something like figure 1 below:

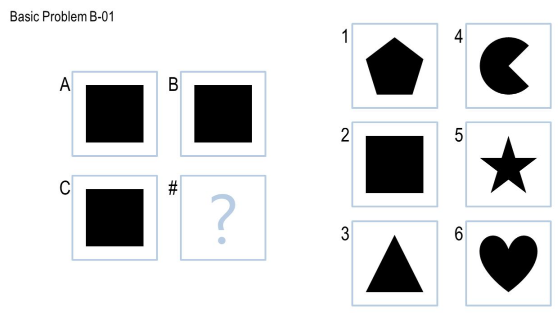


Figure 1

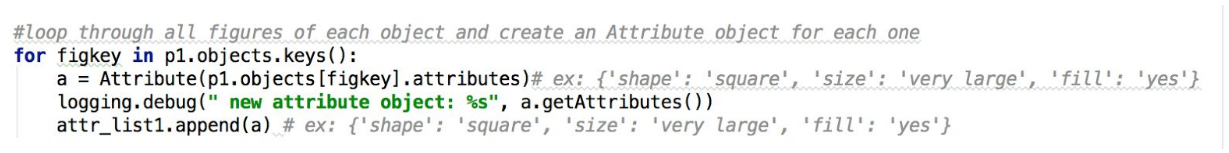
With this assignment, I will be describing how to implement an *AI agent* design which incorporates the methodologies of Frames, Learning by Recording Cases and Case-Based Reasoning to solve one or more of the RPM problems. In this way I can further examine a process often used in computational problem resolution.

Frames

The methodology behind Frames is that oftentimes they can accurately represent stereotypes of our environment. These stereotypes, although not always accurate, help to provide us with default values of the world. These values help us to make sense of our environment as quickly as possible. Without this ability we would be at a loss and potentially in danger, so stereotypes in general have been quite important to us evolutionarily in the history of human intelligence.

So Frames help us to make sense of the data we have to work with. It helps us to store the it in such a way that can be manipulated and examined and related to other things. This is not unlike object oriented programming, where objects are created to store individual components of information, and those objects can be easily related to other objects.
So in applying Frames to the RPM problem, we would want to define the slots and fillers that the agent would use. I used the particular attributes of each figure as provided in the problemData.txt files. I created a class called Attributes which stores the details of each figure, such as shape, fill, size, inside, etc. Using Frames here allows me to get and set attribute values, or default values in the absence of an expected attribute. It allows me to compare sets of attributes, one to another which will help the agent discover relationships between two images.

In figure 2 below, I illustrate how the agent would use a frame to store the object data of an RPM problem:



```
#loop through all figures of each object and create an Attribute object for each one
for figkey in p1.objects.keys():
    a = Attribute(p1.objects[figkey].attributes)# ex: {'shape': 'square', 'size': 'very large', 'fill': 'yes'}
    logging.debug(" new attribute object: %s", a.getAttributes())
    attr_list1.append(a) # ex: {'shape': 'square', 'size': 'very large', 'fill': 'yes'}
```

Figure 2

Although the use of frames has not been exhausted in my AI agent thus far, I could use frames at a larger, more abstract level, such as at the Image level, where each frame of an image could contain information and attributes such as size, filename, bit representations, similar to, contains, etc. I could also use frames at the image level, using data generated with the Pillow image library manipulations.

What Makes Using Frames Difficult?
The problem with using frames is I must design a way to detect figures in the images. In my early attempts I will use the textual representation provided and develop the frames using this data. As I eventually move away from reliance of the textual representation, I will need to develop ways to identify figures as I attempt to solve the challenge problems.

Learning By Recording Cases
Put simply, Learning By Recording Cases is the process of creating “cases” out of the problems encountered in the past (and solutions if applicable), in order to save them in a useful manner so they can be retrieved for and compared to future problems. The goal here would be to retrieve the most similar case to the current case in order to look for a solution. Any and all information that can be generated from the existing problem should be stored as a case. If each case could be represented in such a way that it could be plotted into a “feature-space” then plotting an unknown case in the same way can yield good results. The agent could use the “Nearest Neighbor” method to find the previous case that most closely matches the current case.

As an example to illustrate the meaning of a Feature-Space (also referred to as a Relationship Space), let us look at the example given in class (Figure 3):

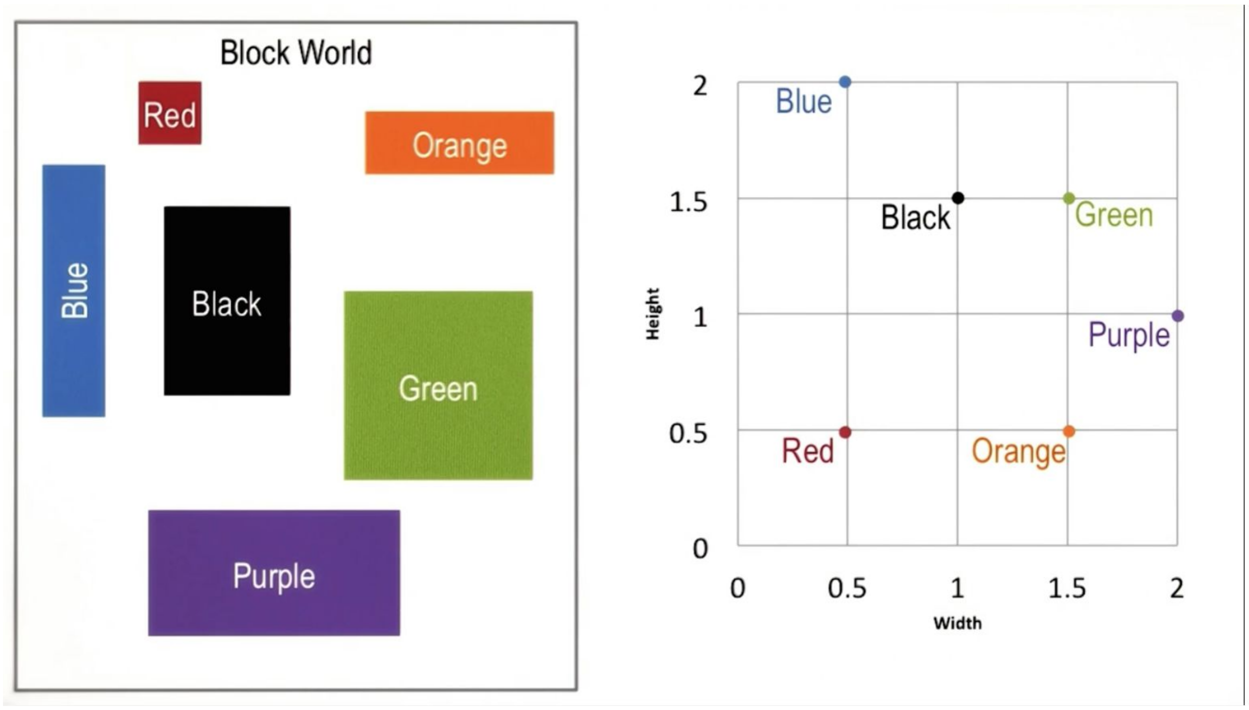


Figure 3

With this example, the blocks are examined for information, in this case we used the width and height of each block to plot it into a feature space. Using these decisional axises the new problem can be plotted in a similar way and the solution can be discovered.

What Makes Learning By Recording Cases Difficult
Learning By Recording Cases can be difficult if the right case formats do not lend themselves to easy identify and comparison. For example, I mentioned plotting each case into a feature space. A feature space is a n-dimensional numeric representation of each case. The difficulty is finding an accurate representation of each case that can be logically compared to future

comparisons. This is critical in the way similarities are evaluated and the most similar problem is determined. Defining the actual case layout should be given much consideration. In the problem of RPM's a case could be created out of the individual transformations between figures. Or each case could be defined at a broader level, such as at the problem level.

Case-Based Reasoning
Case Based Reasoning involves the above methods and uses them all together to discover the RPM solution. Two important assumptions that make Case-Based Reasoning a powerful tool is that 1) patterns exist in the world and 2) similar problems have similar solutions. The four steps to Case-Based Reasoning and how they can be used in the RPM problem are introduced below.

Retrieval
Referring to the nearest-neighbor problem described above, the agent can retrieve the case that is closest, or nearest to the unknown problem it is currently working on.

Adaptation
Adapting the solution that was provided with the previous case found is the next step. The agent must attempt to adapt the same solution to the current problem.

Evaluation
Once an adaptation is discovered, it now needs to be evaluated to see if it passes or fails the current problem. We can do this using evaluations, simulations and/or actual tests for verification. If the adaptation we came up with does not work, then we go back to the adaptation stage and try again.

Storage
Once a solution has been found, or all resources have been exhausted we should store the current case so that it can be used again for futures problems. There are several methods to storing cases, such as Case Storage by Index and Case Storage by Discrimination. The method of storing will directly affect the method of retrieval, because the new problem is assumed to have the same features as the previous problems.

Conclusion
Addressing Raven's Progressive Matrices Using Frames, Learning by Recording Cases, and Case Based Reasoning is a process that can, with processing power, produce an answer that is close to the desired goal. Since storage and retrieval are so closely linked to each other and the success of the evaluation, the programmer must be careful to use the frames developed so they can be stored, retrieved, evaluated and in general, leveraged maximally to achieve the solution. As I am discovering, there are many ways for achieving and displaying artificial intelligence in an agent. Lots of consideration, deliberation and coding must go into building an intelligent agent that can produce expected results.