Creation of Spatial Structure by Activator/Inhibitor Cellular Automaton

Conner Brinkley

02.25.2020

Biologically-Inspired Computing

The University of Tennessee

INTRODUCTION

The purpose here is to investigate the creation of spatial structure by activator and inhibitor cellular automaton. This was done by creating a simulator and giving it 38 different combinations of the parameters J_1, J_2, R_1, R_2, and h to see how that affected correlation and mutual information between cells of the 30 x 30 cellular automaton. The simulator created CSV files to easily create graphs of these values at varying distances along with a JPG image of each step to see how the cellular automaton stabilizes over time. For the purposes of this investigation, only the third experiment was conducted, where both the activator (J_1) and the inhibitor (J_2) were enabled by setting their values to 1 and -0.1, respectively.

PROCESS (& THEORY)

The simulator starts with a 30 x 30 grid that is initialized to random values before using the following update rule over and over again until it reaches a stable state. This state is determined after it tries to update again, but no values are changed.

$$s_i(t+1) = sign\left[h + J_1 \sum_{r_{ij} < R_1} s_j(t) + J_2 \sum_{R_1 \le r_{ij} < R_2} s_j(t) \right]$$

Figure 1. The state transition or update rule used in the AICA simulator.

The r_{ij} value in the state transition rule represents the distance between cells *i* and *j*. This distance is given by the following equation.

$$r_{ij} = |i_1 - j_1| + |i_2 - j_2|$$

Figure 2. The distance between two cells *i* and *j*.

These two formulas work together to turn a randomly initialized cellular automaton into a stable state over however many runs it takes until no more cells are updated. The simulator was programed to output all of these runs as images, so Figure 3 shows a visual of what is going on.

Figure 3. Simulation of cellular automaton stabilizing. $J_1 = 1$, $J_2 = -0.1$, $R_1 = 1$, $R_2 = 14$, and $h = 0$.

Once the simulator reaches a stable state, it calculates four different parameters: spatial correlation, entropy, joint entropy, and mutual information for 15 different distances. The first parameter gives the extent to which cells at those various distances are correlated, where N^2 is the size of the grid and C_l is the circumference of a neighborhood of radius l.

$$\rho_l = \left| \frac{2}{N^2 C_l} \sum_{\substack{\langle ij \rangle \\ r_{ij}=l}} s_i s_j - \left(\frac{1}{N^2} \sum_i s_i \right)^2 \right|$$

Figure 4. Spatial correlation equation.

The second one is entropy, which is used later down the road when average mutual information is calculated. $\Pr\{s\}$ where $s \in \{-1, +1\}$ is the probability of state values 1 and -1, so this needs to be calculated before summing the probabilities together.

$$H(S) = -\sum_{s \in \{-1,+1\}} \Pr\{s\} \lg \Pr\{s\}$$

Figure 5. Average entropy equation.

The third parameter calculated is joint entropy, where the probabilities are calculated just like above before they are summed up for each of the 15 distances. This is also useful for calculating mutual information in the next step.

$$H_l = -(P_l\{+1,+1\}\, lgP_l\{+1,+1\} + P_l\{-1,-1\}\, lgP_l\{-1,-1\} + P_l\{+1,-1\}\, lgP_l\{+1,-1\})$$

Figure 6. Average joint entropy equation.

Finally, it is time to calculate average mutual information, which is another way to measure the correlation between cell states. This is given by multiplying the already calculated entropy by two and then subtracting the joint entropy previously calculated at each distance.

$$I_l = 2H(S) - H_l$$

Figure 7. Average mutual information between cells.

Notice how average mutual information is essentially just a magnitude of the inverse of joint entropy. This will show up later when the data is graphed.

SIMULATOR

The simulator was written entirely in C++ with the help of Dr. Van Hornweder's write up, and the source code can be found in the aica.cpp file in the source directory. Python was heavily considered for this project, but after some consideration, C++ seemed like the best option due to how much quicker it could process each experiment run. Since 38 different combinations were evaluated, C++ made the debugging process a breeze since it was able to run all combinations in under a minute – it was easy to calculate everything and then see if something was off from a big picture standpoint across multiple combinations instead of trying to evaluate the code based on the result of only one or two. The simulator created a CSV data file for each combination and generated several PGM files that were converted to JPG images to view the cellular automaton reach its stable state for each run.

The aica.cpp simulator only takes in one combination at a time, and if you try to execute it with the wrong amount of arguments, it gives you the following usage statement:

USAGE: ./aica R1 R2 H ID

Because experiment three was the only one investigated, J_1 and J_2 were kept as constants in the program, but it would be very easy to modify it to take them as command line arguments as well in order to run experiment one and two by disabling the activator or inhibitor later down the road.

To run experiment three, another C++ program was written with an array of all 38 different parameter combinations that just looped through each one and called the simulator with each combination. This one takes no arguments and automates the entire process. A makefile was also included to streamline compilation.

Once the program finished generating all of the data files and images, all of the CSV files were saved as Excel workbooks to graph the results using Excel's plotting tool.

RESULTS

Several notable observations have emerged from analyzing the graphs and images and comparing them to the R_1, R_2, and h values that were used to generate them – the first of which being how the h parameter affects the ratio of white to black cells in the images. When the R_1 and R_2 parameters are kept constant, and the h parameter is increased, there is an apparent increase in the number of black cells.

Figure 8. h = -6 (left), h = 0 (middle), h = 6 (right)

Specifically, when h is negative, there are more white cells than black, when h is zero, the image seems balanced, and when h is positive, there are generally more black cells than white. This behavior is demonstrated above in Figure 8, where the images were taken from experiments 19, 21, and 23, respectively. The R_1 and R_2 parameters were kept constant across all three at 3 and 9, respectively.

The next observation is how the R_1 and R_2 parameters affect the shape of cellular automaton. As R_1 increases up to a certain point, it makes the structures have more curvy lines. As R_2 increases, the structures become bigger, almost like holding a magnifying glass over the cellular automaton. When both values are low, the structure looks like noise as shown below in Figure 9.

Figure 9. Noise generated from having low R_1 and R_2 values.

The resulting cellular automata in Figure 9 comes from experiment 3, where R_1 is 1, and R_2 is 5. In Figure 10, R_1 and h are kept the same, but this time R_2 is increased to 14.

Figure 10. Zebra-like stripes generated from a higher R_2 value.

Now, it is easy to see that the structure has come to be a much more apparent zebra-like pattern, whereas before the cell groupings were too small to make out a pattern. It is almost like the cell groupings have become enlarged by some sort of magnifying glass.

Figure 11. More rounded patterns emerge when R_1 increases.

Figure 11 was taken from experiment 26, where R_2 and h were kept the same, but R_1 was increased to 3. The curviness of the structure is much more apparent compared to Figure 10 when R_1 was lower.

The final observation deals with the plotted graphs of spatial correlation, joint entropy, and mutual information.

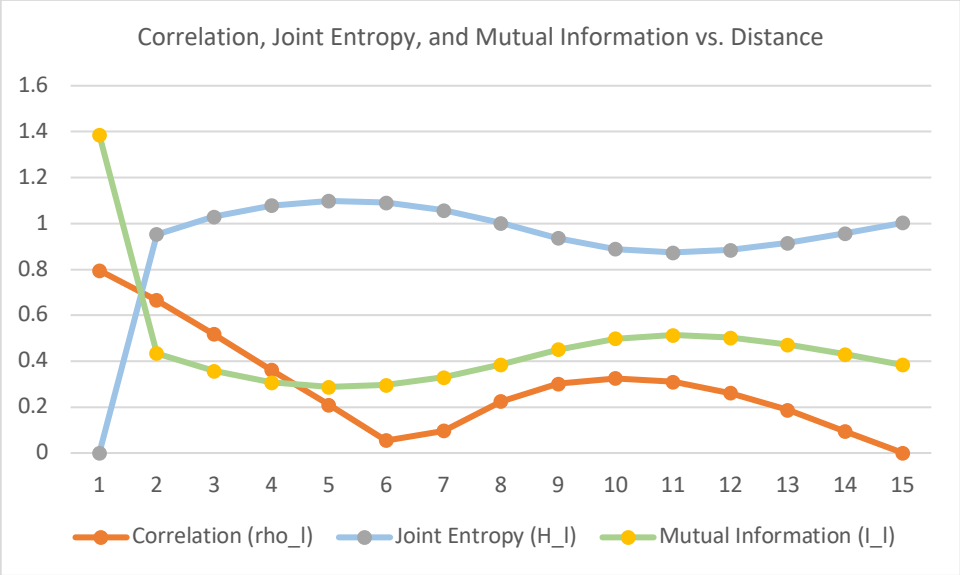

Figure 12. Chart data taken from experiment 13.

Notice how the joint entropy and mutual information lines look almost like a mirror image. As previously stated, this is because mutual information is essentially some magnitude of the inverse of joint entropy. This seemed interesting to point out, but the more notable observation is how joint entropy spikes and mutual information dips right around the R_1 and R_2 values. Figure 12 is based off of the data from experiment 13, where R_1 is 1 and R_2 is 14. This is relatively where the joint entropy and mutual information lines spike and dip, respectively.

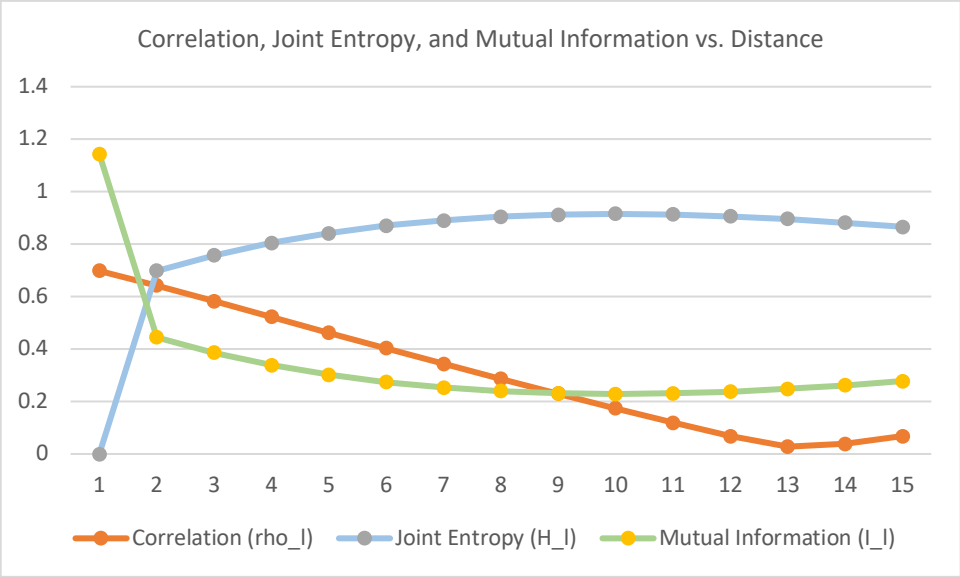

Figure 13. Chart data taken from experiment 33.

Again, just to show another example, Figure 13 was taken from experiment 33, where R_1 is 7 and R_2 is 14. On the graph, joint entropy and mutual information approach their highest and lowest values, respectively, around distance 7. Then they start to go back to where they were before after distance 14. It is not exact, but it is within ballpark range, so it seemed like an interesting pattern to point out.

CONCLUSION

In summary, the majority of the time spent on this project was developing the code for the simulator. A couple of bugs were huge setbacks, but once it was complete, analyzing the results did not take too long. After a while of going through the data and making graphs, the following three observations persisted throughout most of the process:

1. Increasing h increases the number of black cells and vice versa.
2. Higher R_1 means more curves and higher R_2 means bigger groupings of cells.
3. Joint entropy spikes and mutual information dips around R_1 and R_2 values.

More could probably be found by analyzing further, but these seemed like the most apparent and important regarding how to manipulate the values to create future desired cellular automaton.

In the future it might be interesting to modify the simulator to take J_1 and J_2 values to see what happens when the activator or inhibitor is disabled and analyze the results.