Utrecht Summer School

Introduction to Complex Systems

Project Day 2

Cellular Automata

1 Pattern Formation

Many complex systems exhibit (spatial) patterning as a consequence of the evolution rules for each individual/element/cell/.... For instance, vegetation patterns in ecosystems, the stripes on a zebra, the formation of clouds, social segregation, and city formation. Pattern formation can occur if the complex system has a spatial scale-dependent feedback mechanism. That is, if the presence of something (depending on the system it could be the presence of e.g. vegetation or a person) is beneficial for others over a short range, but detrimental over a long range ('short-range facilitation' or 'short-range activation' and 'long-range inhibition'). In this project, we explore this pattern formation mechanism in Cellular Automata (sometimes abbreviated to 'CA').

For this, we consider a lattice of $M \times N$ cells with periodic boundaries (i.e. the right-side of the domain is connected to the left-side and the bottom-side is connected to the top-side; this is a 'trick' often used to overcome the unwanted effects of having fixed boundaries). All cells can be either 'off' ('0'/'absence') or 'on' ('1'/'present'). With $A_{ij}(t)$ we denote the value (either 0 or 1) of the site at row i, column j at time t. Now, the evolution rule is based on the spatial scale-dependent feedback. For each cell, we compute the experienced total facilitation/inhibition s_{ij}, where every 'on' (1) state within activation range R_a contributes $+w_a$ and every 'on' state within inhibition range R_i contributes $-w_i$. If $s > 0$, the cell will be 'on' (1) at the next time step; otherwise it will be 'off' (0). Mathematically, this evolution rule is given by

$$A_{ij}(t+1) = \begin{cases} 1, & s_{ij}(t) > 0; \\ 0, & \text{otherwise}, \end{cases} \tag{1}$$

where s is given by

$$s_{ij}(t) = \sum_{(k,l)\in N_a(i,j)} A_{kl}(t) w_a - \sum_{(k,l)\in N_i(i,j)} A_{kl}(t) w_i, \tag{2}$$

and $N_a(i,j)$ respectively $N_i(i,j)$ are the collection of cells within activation respectively inhibition range:

$$N_a(i,j) = \{(k,l) : \text{ the distance between } (i,j) \text{ and } (k,l) \text{ is smaller than } R_a\}; \tag{3}$$

$$N_i(i,j) = \{(k,l) : \text{ the distance between } (i,j) \text{ and } (k,l) \text{ is larger than } R_a \text{ and smaller than } R_i\}. \tag{4}$$

Now, locate the files `CA.ipynb` and `CA_functions.py`, and place them in the same system folder. Only the main notebook needs to be opened and appended for this project. The code has prepared the simulation of such kind of system. The initial condition for this system is a random arrangement of 'on' and 'off' states, where approximately p of the cells are in the on state.

1.1 The spatial scale-dependent feedback

The above description depends on the parameters R_a, R_i, w_a and w_i. What kind of conditions do you need to impose on those to allow for pattern formation? Make a diagram for the scale-dependent feedback; i.e. the influence of an 'on' state cell on $s_{ij}(t)$ depending on its distance to the cell (i, j). In the code, the system is modelled on a lattice with periodic boundaries. Does this influence the results? Think about when this is appropriate and when this is not?

1.2 Emergent patterns

Run the model for some parameter values that lead to patterns. Will the system converge to an equilibrium state? How does it look like?

1.3 Dependence on initial conditions

Determine the dependence on the precise initial configuration. For instance, do different patterns emerge when you re-run the code with the same parameter values? Do different patterns emerge for different values of p? (Recall that p denotes the approximate fraction of the initial state that is in the 'on' state.)

1.4 Dependence on other parameters

Experiment with different parameter values and vary the values of R_a, R_i, w_a and w_i. How do the emergent patterns change with these values?

1.5 Additional suggestions

If time permits, try to come up with a way to quantify patterns and compare the emergent patterns quantitatively.

2 Panic Spreading

We now turn our attention to another phenomena that can be modelled using the Cellular Automata formalism. Let us say that every cell in the lattice now represent a person, that can be either panicked (the 'on' state) or unpanicked (the 'off' state). By stipulating some rule to determine when a person becomes (un)panicked, we can now consider whether and how panic spreads through the system.

We consider the 'droplet rule' for the evolution in this system: for any given person, look at the surrounding individuals (typically taken as the 8 direct neighbouring cells) and count how many of them are panicked. If at least 3 of them are panicked, the person becomes panicked at the next timestep; otherwise the person will be unpanicked.

2.1 Implementation

Implement the droplet rule in the code. For this you will need to directly modify the function `CA_evolution`.

HINT: Try to think of the pattern formation system from the first part and think about how this rule is different. Are there activators/inhibitors and what is their range? How large does the strength $s_{ij}(t)$ need to be in order for the person at (i, j) to become panicked? You can implement this with only one change to the evolution function and changing the parameters of the scale-dependent feedback.

2.2 Dependence on initial conditions

Run the model for various values of p. Do you see different behaviour? Why?

2.3 Quantification of panic spread

Find a way to quantify the end state of the system (i.e. define some order parameter). Compute its value for various values of p. Does this system have a phase transition? If so, around which p-value does this happen?

2.4 Additional suggestions

If time permits, feel free to experiment with other rules. For example, what happens when you change the number of required panicked neighbours from 3 to another number?

Alternatively, you can also look into the behaviour of your order parameter as the value of p approaches the critical value at which the phase transition happens.