Herd Behavior in Collaborative Games

Submission Requirements:

- A .zip file containing your source code. You may use any language you would like.
- A PDF containing each item below that is listed as a Deliverable. For each item contained in your PDF, clearly mark which deliverable it is associated with. Plots should be clearly labeled and have descriptive captions.

Tips:

- Your Schelling Segregation grid implementation code will provide a helpful template to edit in order to implement the model.

Herd Behavior in the Context of Multi-Agent Games. In class, we discussed two-agent games of perfect information where the players have many choices. We can extend this scope to represent multi-agent games where there are many players at once who have binary choices to either act (represented by a 1) in the same way as each other or not act (represented by 0). This article provides a comprehensive overview of these games known as herd behavior. Let taking action mean buying on trend goods like Stanley cups like your peers or golfing because all of your friends like to golf. Inaction is simply not participating in these activities.

Where we discuss "payoffs" in game theory, in economics and psychology this idea is called "utility". An individual i's payoff is a function of the following parameters:

- $f(d_i)$: a susceptibility to influence constant where a value closer to 0 mean less susceptible and a value closer to 1 mean more susceptible. Let d_i be the number of my"friends" or my neighbors around me. $f(d_i) = \frac{1}{d_i}$ and represents agent i's susceptibility threshold to being persuaded by their peers. If I have few friends, I am more susceptible to persuasion. If I have more friends, then I am less susceptible. If I am on a grid, this is the fraction of occupied slots in the 8 squares around me. If there are 4 squares occupied in the grid around me, $f(d_i) = 0.25$ or $\frac{1}{4}$.
- $\sum_{j=0}^{n-1} g_{ij}x_j$ is the utility I get from each of my friends acting.
  - $\sum_{j=0}^{n-1}$ sums across all agents on the grid.
  - g_{ij} is an entry in an adjacency matrix and will equal 1 if agent j is a neighbor of agent i.
  - x_j is the binary action of the agent, acting (represented by a 1) or not acting (represented by a 0).

  This summation is the utility or payoff agent i receives from joining their acting neighbors. So, if 4 of my neighbors are acting the summation would equal 4.
- t the simulation threshold for cost of action. If at least t fraction of my neighbors are acting, I will also act.

Their exact payoff that they receive from acting in the game can be represented by the following function:

$$U_i = f(d_i) \sum_{j=0}^{n-1} g_{ij} x_j - t$$

If you don't act, $U_i = 0$. Accordingly, an agent will choose to act if their payoff is greater than 0. They will not act if their payoff from acting would be less than 0. That is, if $t < f(d_i) \sum_{j=0}^{n-1} g_{ij} x_j$ or $U_i > 0$, then the agent would be at an advantage for acting, so they SHOULD act. Conversely, if $t > f(d_i) \sum_{j=0}^{n-1} g_{ij} x_j$ or $U_i < 0$, then the agent would be at a disadvantage for acting, so they should NOT act. That is, succumbing to peer pressure to act or not act goes both ways as the simulation runs.

NOTE: this simulation will depend on **global utility**. Let global utility be the sum of the utility of each agent across all agents on the grid on a given iteration of the simulation.

Generate a grid-based simulation with the following parameters:

- `N:` Grid size; your overall grid should be NxN. You may leave N as a parameter in your code, but all your deliverables should use the same value of N which should be at least 30.
- `active_inactive_split:` the fraction of total agents that are participating in the action at the start of the model. Between 0 and 1 inclusive.
- `t:` the threshold for how many of those in the 8 spots around me who are acting that will cause me to also act. That is if `t=0.4` and I have 5 neighbors, I will act if at least 2 of them are also acting. Between 0 and 1 inclusive.
- `pct_empty:` the percentage of grid squares that are empty. If `pct_empty = 1` this means the grid is completely empty. If `pct_empty = 0.1` this means the grid is only `10%` empty. Between 0 and 1 inclusive.

How your simulation should work:

1. Initialize the NxN grid with agent types acting according to `active_inactive_split` and empty squares allocated according to `pct_empty`.
2. Choose a random agent. Check to see if the agent will act or stop acting given the threshold for action. If the agent should choose to act, make it, then continue the simulation choosing another random agent and so on. Also, if an agent should choose to stop acting, make it, then continue the simulation.
3. Repeat step 3 until either:
   - All agents are consolidated to one behavior (all agents are acting or all agents are not acting)
   - A maximum iteration count chosen by the programmer has been reached

- (or use advanced criterion like tracking the global utility of the society over time, and if it goes a long time without changing, terminate)

Assignment:

1. Implement the given model of heard behavior using code of your own design (NOTE: this can be done by altering your Schelling Segregation implementation model code). In the PDF, provide several visual examples (using small 3x3 or 4x4 grids) to verify that your agents are acting appropriately.

   Deliverable 1: The code for your project (you may write your own visualizer or use the Python one provided here).

2. How does global utility change given choice of parameters? Create a functionality that tracks how global utility over several iterations of the model. You should be able to call this functionality at each step along a simulation to see how global utility changes over the simulation run.

   Deliverable 2: Create a series of plots that demonstrate how global utility changes over the course of several runs of the model. Each plot should have iteration number on the horizontal axis and global utility on the vertical axis. You should choose at least 4 sets of parameter values (i.e at least 4 different combinations of `active_inactive_split`, t, and `pct_empty`). For each of these sets of parameter values, you should have one plot with 5-10 traces for each of the 4 sets of parameter values, for a total of 4 plots. Clearly mark what parameter values gave rise to each plot. Do you notice any interesting trends in your plots? Document them and explain why you think they are occurring.

3. How does global utility depend on parameter values? Create a function which measures the average global utility at the end of a simulation run as a function of input parameters. Your function signature should be

   ```
   average_Global_Utility(active_inactive_split, t, pct_empty)
   ```

   When called, this function should perform 10 simulation runs with the given parameters, record the global utility found at the end of each run. Finally, the function should return the average of those 10 end of simulation utility values.

   Deliverable 3: Four plots created using your `average_Global_Utility` function:

   (a) Plot 1 should display average global utility as a function of `active_inactive_split` with the other parameters held constant.

   (b) Plot 2 should display average global utility as a function of `t` with the other parameters held constant.

   (c) Plot 3 should display average global utility as a function of `pct_empty` with the other parameters held constant.

   Each plot should have at least 10 points on the horizontal axis so that you can see the general shape of the curve. Note that each time you call `average_Global_Utility`, it performs 10 simulation runs, so if you have 10 points on the horizontal axis for each of these plots, then

this deliverable requires 300 simulation runs. Plan your time accordingly, if your code is very slow, you might consider saving the results to a disk after each run just in case your code hits a bug before all 300 simulation runs.

4. Your goal is to come up with a significant modification to the Herd Behavior model and see how it changes the model's behavior. Base this choice off real herd behavior phenomena! If you find interesting visual phenomena, you are welcome to create an animation of your results, post it to your favorite video-sharing service, and provide a link to it in your PDF. Here are some ideas to get you started:

   - What if agents move or even leave the simulation?
   - Modify grid structure: Do the same phenomena emerge if this were played on a hex or triangular grid instead of a square grid?
   - What if t increases as more agents act?
   - What if an agent can be influenced by more than the 8 slots around them?

   Tie your choice to a real world phenomena and justify the choice of your change and run Deliverables 2 and 3 again. Does the model behave the way you think it should given the real world circumstance? Creativity here will be rewarded!