Assignment6

March 25, 2024

1 Assignment 6

[10]:
```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation
```

1.0.1 1. Modelling Microbial Colony Growth with Cellular Automata

Imagine you are helping a colleague in microbiology understand the growth of their bacterial colonies on a petri dishes. Your colleague is growing a bacteria that can either divide, die, and enter/exit a dormant state/ Your goal is to create CA simulation that provides insights into the rates at which these different events occur. The results of three replicates of their experiment are shown below.

Note: There is no one right answer to this problem but there are reasonable and unreasonable ones

Part A: Define the grid and cells for your cellular automata. Specify what each cell in the grid represents in your model and the possible states a cell can have. Include states of "Active," "Dead," "Empty," and "Dormant". Are you going to use a *Von Neumann* or *Moore* neighbourhood.

Let's model a 40×40 grid (approximating the petri dish as a square.

Each grid cell represents a micro-location on the petri dish that can be filled with bacteria or not. Before colonization the cell is empty, it can be colonized by the bacteria in which case the bacteria can be actively replicating (Active) in a cecile (Dormant) state or the bacteria can die.

I am going to use a Moore neighbourhood, hence each cell has 8 neighbours.

Part B: Establish rules that govern the behaviour of the microbial cells consistent with the meaning of the states above. Consider factors such as nutrient availability and cell density.

Rule 1: An active cell can reproduce and colonize a neighbouring cell proportional to the $b * n_{\text{empty}}$ where n_{empty} gives the number of empty neighbouring sites.

Rule 2: An active cell can go dormant. The rate of dormancy may be proportional to "stress" and hence might increase with the number of filled neighbouring cells. Let the rate of dormancy be: $d(1 + \alpha(8 - n_{\text{empty}}))$, where d is the intrinsic dormancy rate and α is the sensitivity to density.

Rule 3: A dormant cell may re-activate at a constant rate r.

Rule 4: An active can die at rate δ_A, let's assume this is density-independent.

Rule 5: Cells go dormant to avoid death, so let's assume $\delta_D \ll \delta_A$

Part C: Discuss how you would initialize the simulation, including the initial configuration of cells. Additionally, address the boundary conditions of your CA. Discuss how you would pick the time span/stopping conditions of your simulation?

Let's initialize the simulations by randomly dispersing 5 single-cell colonies on the plate. There are 1600 cells on the plate.

Let's assume the whole grid is filled with dead cells as a boundary of inactive but density-creating cells.

Let's run the simulation until 10% of the plate is full.

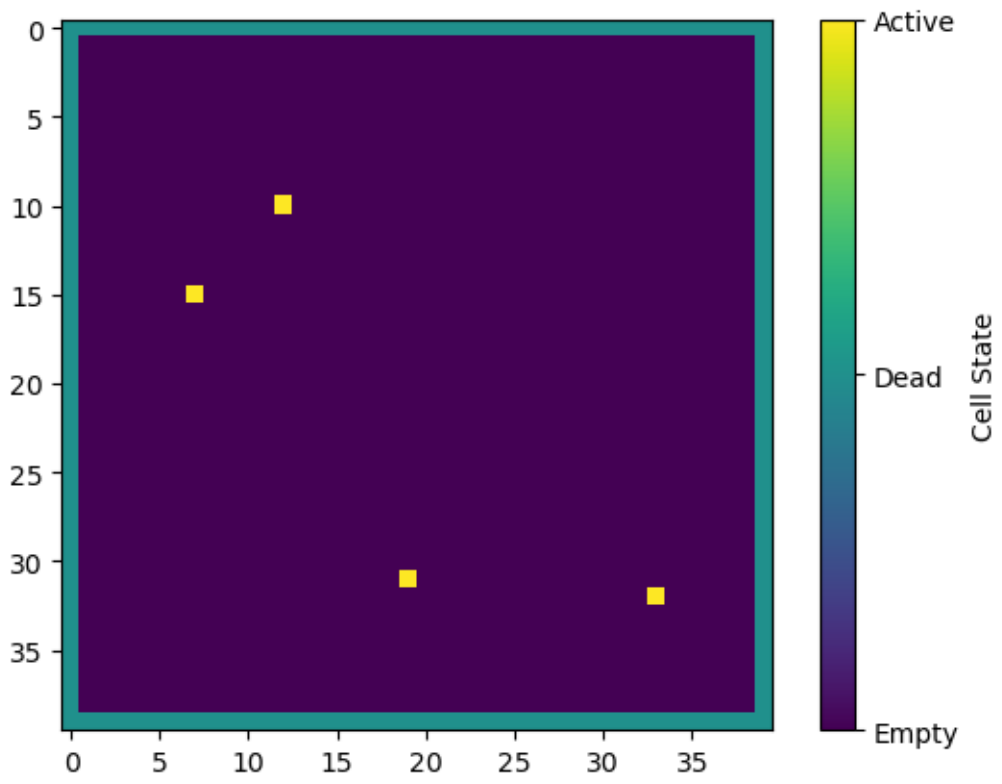
Part D Challenge for 795: Develop the CA you outlined above and show the result of 1 simulation replicate.

[107]:
```
# Parameters
grid_size = 40
num_generations = 10
initial_colonies = 5
b = 0.05
d = 0.02
alpha = 0.2
r = 0.005
delta_A = 0.01
delta_D = 0.001

# Initialize grid
grid = np.zeros((grid_size, grid_size))  # 0 represents empty, 1 represents␣
 ↪dead, 2 represents active, 3 represents dormant

# Initialize random colonies
random_cells = np.random.choice(grid_size, size=(initial_colonies, 2),␣
 ↪replace=False)
grid[random_cells[:, 0], random_cells[:, 1]] = 2  # Set initial colonies as␣
 ↪active cells

# Set boundaries as dead cells
grid[0, :] = grid[:, 0] = grid[-1, :] = grid[:, -1] = 1

# Function to count empty neighbors
def count_empty_neighbors(x, y):
    neighbors = grid[x - 1:x + 2, y - 1:y + 2]
    return np.sum(neighbors == 0) - (grid[x, y] == 0)

# Function to update the grid based on the rules
def update(*args):
```

```
    global grid

    new_grid = grid

    for i in range(1, grid_size - 1):
        for j in range(1, grid_size - 1):
            if grid[i, j] == 2:  # Active cell
                n_empty = count_empty_neighbors(i, j)

                # Rule 1: Reproduction and colonization
                reproduction_probability = b * n_empty
                if np.random.rand() < reproduction_probability:
                    empty_neighbors = np.argwhere(grid[i - 1:i + 2, j - 1:j +␣
↪2] == 0)
                    chosen_empty = empty_neighbors[np.random.
↪choice(len(empty_neighbors))]
                    new_grid[i + chosen_empty[0] - 1, j + chosen_empty[1] - 1]␣
↪= 2

                # Rule 2: Dormancy
                dormancy_rate = d * (1 + alpha * (8 - n_empty))
                if np.random.rand() < dormancy_rate:
                    new_grid[i, j] = 3  # Set the cell as dormant

                # Rule 4: Death
                if np.random.rand() < delta_A:
                    new_grid[i, j] = 1  # Set the cell as dead

            elif grid[i, j] == 3:  # Dormant cell
                # Rule 3: Reactivation
                if np.random.rand() < r:
                    new_grid[i, j] = 2  # Set the cell as active

                # Rule 3: Reactivation
                if np.random.rand() < delta_D:
                    new_grid[i, j] = 1  # Set the cell as dead

    # grid = new_grid
    # img.set_array(grid)
    return new_grid
```

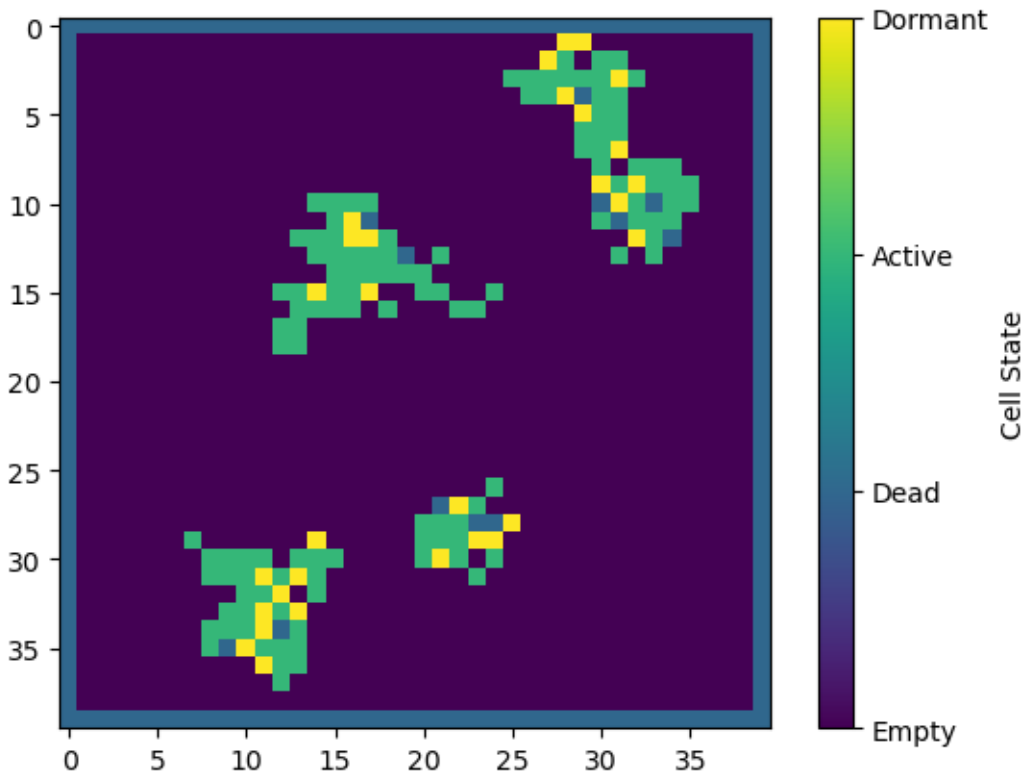
Inital Grid

[100]:
```
fig, ax = plt.subplots()
img = ax.imshow(grid, cmap='viridis', interpolation='nearest', animated=True)
cbar = plt.colorbar(img, ticks=[0, 1, 2, 3], label='Cell State')
cbar.set_ticklabels(['Empty', 'Dead', 'Active', 'Dormant'])
```



[110]:
```
def sim():
    temp=grid
    while np.sum(temp[1:-1,1:-1]> 0)<160 or np.sum(temp[1:-1,1:-1]> 1)==0:
    # for i in range(20):
        temp=update(temp)
        # print(np.sum(temp[1:-1,1:-1]> 0))
    return temp
```

[111]:
```
fig, ax = plt.subplots()
img = ax.imshow(sim(), cmap='viridis', interpolation='nearest', animated=True)
cbar = plt.colorbar(img, ticks=[0, 1, 2, 3], label='Cell State')
cbar.set_ticklabels(['Empty', 'Dead', 'Active', 'Dormant'])
```



1.0.2 2. Individual based predator-prey simulation.

You are helping a wildlife ecologist understand the dynamics of the caribou population of the Northwest Territories. Within this population, caribou live and travel in herds. Their major predators are wolves who themselves move and hunt in packs. Caribou reproduce annually once they reach reproductive maturity. Juvenile caribou are much more likely to be caught by predators than adults. Individual wolves must consume a certain amount of prey 'energy' to survive. Packs are dominated by an alpha male, subordinate beta males, and females. A wolfs ability to mate and access to food are determined by sex and social hierarchy within the herd.

Part A. Sketch the structure of an individual based simulation of caribou/wolf population dynamics. What would you use as your classes? What variables (and their data types) would you prescribe to each class? What are some class functions you would need? What global functions will you need?

Part B Sketch some plots of the type data you would like to create with your IBS.

1.0.3 3. Likelihood and Bayesian Inference

A GC island, also known as CpG island, is a genomic region characterized by a higher-than-expected frequency of the DNA bases guanine (G) and cytosine (C) pairs (e.g., sequences with repeats of the form GCGC). You have data from DNA sequencing

reads, 200-300 base pair long sequences. Your goal is to infer the proportion of the genome enriched for CpG islands.

Suppose you model the genome with the following stochastic process:

Part A: Write down the transition probability matrix for the stochastic process shown above.

Part B You are given a read with n base pairs as denoted by the sequence $x_i \quad i \in \{1, 2, ...n\}$ where $x_i \in \{A = 1, C = 2, G = 3, T = 4$. Express the likelihood of observing a given read under the stochastic model shown above. What is the "Data", $\mathcal{D}$, in this case? What is the "model", Θ?

Part C: Explain why it might be best to use the log-likelihood in model fitting for this problem.

Part D: If using a Bayesian inference framework what priors might you want to use for this problem? Hint: consider what you think the effect of CpG island should be on the transition rates.

[ ]: