Minesweeper

Introduction to Artificial Intelligence

Maitri Shah (Section 7), Kajol Bhat (Section 6) & Ashni Patel (Section 3)

Contents

Part One

1. Questions and Write Up

1.1 Representation

Our minesweeper board was represented by a two-dimensional array. We randomly placed mines around the board. The dimensions of the array and the number of mines are determined by the user but for the purposes of this write-up, we used a 15x15 mine for all of our boards, unless otherwise indicated. Our game board displayed the contents of every cell, including the number of mines that were surrounding a cell and which cells were mines. We would know that a cell is safe because it would reveal a number, indicating the number of adjacent cells where there are mines.

We created a second board using a two-dimensional array to indicate the knowledge base of the agent (amaze in the code). By examining the game board and knowledge base side-by-side, we were able to better visualize the effectiveness of the agent. If a agent were to accidentally open up a mine at a given location, the opened cell would be marked as 'M' in the knowledge base to indicate that a mine was opened. If there is no mine at the opened location, the agent will have access to the number of adjacent cells where the mines are located, or the clue value. If an agent were to flag a location as a mine, the cell would be marked as an 'F' in the knowledge base and it would not be opened. The knowledge base essentially indicates the information depending on its relationship to its neighbors. All of our code was done in python and run on Jupyter Notebook.

1.1.1 Algorithm

The algorithm uses its knowledge base to make a standard inference about the cells surrounding it. If we find some conclusive or highly probable knowledge as to whether a cell is safe or a mine, the agent would update its safe fringe, mine fringe, and knowledge base accordingly.Our algorithm considers the number of mines surrounding a cell and the number of unopened values to algebraically predict which of the unopened cells are mines and which of them are safe.If the surrounding mines of a particular cell are identified and updated in the mine fringe and knowledge base, the agent would add the other neighbors into a safe fringe. For example, if the clue of the cell was revealed to be 2, indicating that there are 2 mines neighboring the cell– if the agent had already identified two of the cells neighbors as 'mines', the algorithm would predict that the rest of the uncovered cells are safe and would subsequently add them to the 'safe fringe'. Once an element is

added to the safe fringe, the agent would prioritize opening the values in the fringe.The algorithm heavily relies on the accuracy of the knowledge base and mine fringe.

The advanced agent uses a similar kind of logic to the basic agent, except in the advanced agent, we are checking a neighbor of the opened cell, as well as all of the neighbors of this neighbor. This additional knowledge allows for there to be inferred relationships between the cells as we have additional clues to work with. In the basic agent, we are only using this logic for checking one cell and its neighbors at a time so we are not able to get knowledge from the relationships between the clues of different cells. In the advanced agent, we use the knowledge that we get from the neighbor and the neighbors of the neighbor to get a more accurate guess for the mine placement.

1.2 Inference

When we collect a new clue, the information is given to the agent's knowledge base. We determine whether or not the given point is a mine, afterwards we examine the neighbors of the coordinates so the agent can make predictions about neighboring mines. We perform basic inferences depending on the value of the coordinates. For example, if we have a 0, the agent will open all of its neighbors until there are no more zeros present because the agent will know that the point and its respective neighbors are safe. Furthermore, if the number of uncovered neighboring cells is equivalent to the clue, the agent will accurately assume that each uncovered cell is a mine. It will then update that information in its knowledge base and mine fringe accordingly. Additionally, if the number of unopened values summed with the number of mines that are known is equivalent to the clue, the agent can determine that unopened coordinates must be mines. The current state of the knowledge base, safe fringe, and mine fringe are updated accordingly. As our agent proceeds to traverse the board, it will continue to update its knowledge base to include how many mines are neighbors at a given point. If there are no other logical possible routes, the algorithm will choose a random point and continue the process from there.

The advanced agent may open neighboring values. If the clue or information is not sufficient enough to make any accurate conclusions, the algorithm will update the information in the knowledge base and access the neighbors of the clue to gain more information.

Our program does deduce everything it can from a given cell as the agent will exhaust the number of clues that we can get from a given opened cell by checking the neighbors of the cell and the neighbors of the neighbors. When we are done checking both the neighbors and the neighbors of the neighbors, if we can no longer predict that there are safe options, we continue solving by picking another neighboring cell of the initial cell. Due to this, we can be sure that all of the information that can be received from this clue will definitely be used in improving the knowledge base.

1.3 Decisions

As described in Inference in more detail[1]: Based on our board's state, our agent chooses where to search next based on which nearby neighbors are open, which cells are known mines or highly suspected mines (flagged as mines). Using this knowledge, we continuously look at the neighbors' neighbors and neighbors of the initial point we click on in the program. Based on this, the agent will only go to values deemed safe by prioritizing and exploring the cells in the safeList first and analyzing the clues of those. Suppose there are no other logical possible routes– meaning that the agent has opened all of the values in the safe list and identified as many potential mines as it can. In that case, the algorithm will choose a random point and continue the process from there.

[1] In the interest of avoiding redundancy, please refer to Inference in the event of any uncertainty

1.4 Performance

In order to test our performance, for our play by play walk through we used a $10x10$ board with 10 mines.

When our algorithm begins the agent maze, or the knowledge that the agent has, consists of all 'U's to represent all unopened cells

```
U U U U U U U U U U
U U U U U U U U U U
U U U U U U U U U U
U U U U U U U U U U
U U U U U U U U U U
U U U U U U U U U U
U U U U U U U U U U
U U U U U U U U U U
U U U U U U U U U U
U U U U U U U U U U
```

We then choose a random square to open first

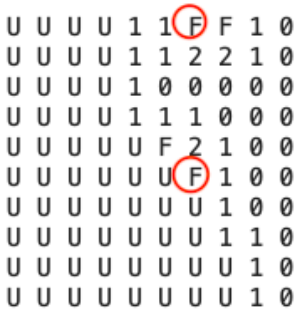
```
U U U U U U U U U U
U U U U U U U U U U
U U U U U U U U 0 U
U U U U U U U U U U
U U U U U U U U U U
U U U U U U U U U U
U U U U U U U U U U
U U U U U U U U U U
U U U U U U U U U U
U U U U U U U U U U
```

Since the square that was opened was a '0', our algorithm will recursively opens all neighbors for any adjacent zeros until there are no more zeros adjacent to open. Since we know that every cell surrounding a 0 is safe, we can confidently deem all of these cells 'safe' and open them.

```
U U U U U U U U 1 0
U U U U 1 1 2 2 1 0
U U U U 1 0 0 0 0 0
U U U U 1 1 1 0 0 0
U U U U U U 2 1 0 0
U U U U U U U U 1 0 0
U U U U U U U U 1 0 0
U U U U U U U U 1 1 0
U U U U U U U U U 1 0
U U U U U U U U U 1 0
```

Since the first value selected was a 0, the algorithm proceeds to open all of its neighbors. If at any point, the algorithm encounters a neighbor with the value of 0, the algorithm will repeat the process by opening all of its neighbors. This process will continue until there are no more available zeros as neighbors remaining. Ultimately, since all of the values that surround the number 0 are safe, the agent will randomly choose a neighbor to open and access. In the image above, the neighbor that is selected randomly is the 1 above it. The neighbors of this 1 and the neighbors of its neighbors are now analyzed, in which we are able to deduce that the unopened value highlighted in red is likely a mine. It is thus labeled 'F'. If the agent determines a value to be a mine, it will flag it as 'F' and avoid opening it in the future.

Since the method (spaceMine) is called many times separately for the whole amaze, rather than this happening one at a time, many separate pairs of cells are analyzed at once, where the "neighbors of the neighbors" and the "neighbors" of a given cell are checked so that we can best identify potential mines. Every time the method spaceMine is run, it takes in the agent maze that has all the values the agent knows at a given moment so this function is run on the opened cells of the grid as a whole.



After expected mines are flagged, if the program does not know what to do next it will open a square at random which can be seen with the cells that are opened adgacent to the flagged cell.

This process then continues.

```
0 0 1 1 1 1 F F 1 0
0 0 1 F 1 1 2 2 1 0
0 1 2 2 1 0 0 0 0 0
0 1 F 1 1 1 1 0 0 0
0 2 2 2 1 F 2 1 0 0
0 1 F 1 2 3 F 1 0 0
0 1 1 2 2 U 2 1 0 0
U U U U U 2 2 1 1 0
U U U U 1 1 1 F 1 0
U U U U 0 0 1 1 1 0
```

Then finally, once all of the cells have been opened, the program stops and calculates the score. In this, particular situation, there was no instances of the program accidentally picking a mine when choosing a random step, so our score is 100 percent as all the mines were successfully flagged.

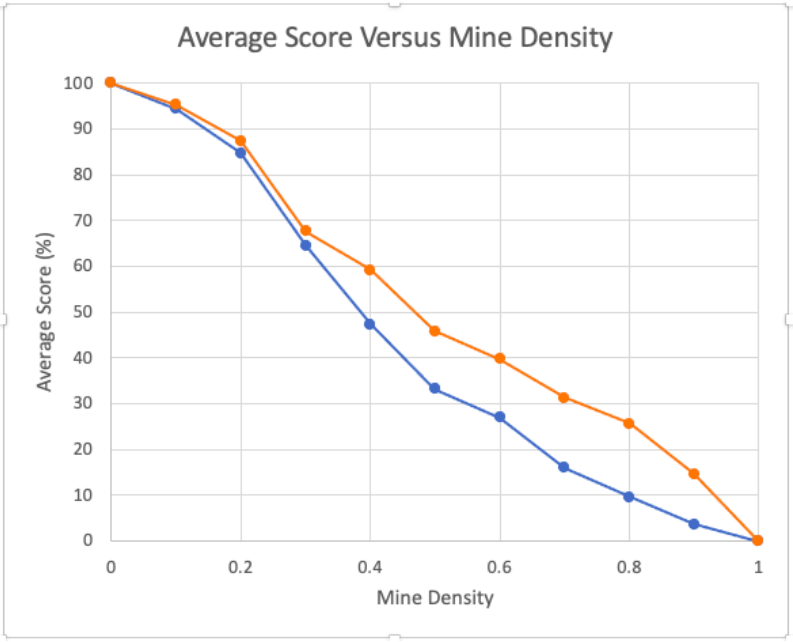
```
0 0 1 1 1 1 F F 1 0
0 0 1 F 1 1 2 2 1 0
0 1 2 2 1 0 0 0 0 0
0 1 F 1 1 1 1 0 0 0
0 2 2 2 1 F 2 1 0 0
0 1 F 1 2 3 F 1 0 0
0 1 1 2 2 F 2 1 0 0
0 0 0 1 F 2 2 1 1 0
0 0 0 1 1 1 1 F 1 0
0 0 0 0 0 0 1 1 1 0
```

Analysis : There are no points in this run through that the program made a decision that we do not agree with. However, in all of our trials, a situation that we were not completely satisfied with would be when our program randomly chose a spot when not knowing which unopened tile was safe and accidentally opened a mine. This includes the algorithm's initial selection as well as any time the algorithm was forced to choose a cell randomly. The only time that our program made a mildly surprising decision is when it would operate on multiple cells at once. This is however not too surprising as the main method of the algorithm runs in conjunction many times to ensure that every element can be correctly identified as accurately as possible.

1.5 Performance

In order to test our performance, we ran 10 minesweeper grids for each density for both the advanced agent and the basic agent. The grids were all $15x15$ and the number of mines were done according to the densities shown on the graph below.

Based on the graph below, we can see that in general, for most values the advanced agent has a higher average score than the basic agent. This graph agrees with our intuition, as for the lower density values, below or equal to 20 percent, both of the agents seem to perform similarly. This is as there are fewer mines so the additional work that the advanced agent does by looking at multiple neighbors does not have as much of an impact as there are fewer potential mines to hit. However, as the mine densities increase, we can see that the advanced agent has a significantly higher average score than the basic one. Minesweeper appears to become "hard" around a 30 percent density, as that is when we see a steady decline in the average score percentage for the remaining densities, with the advanced agent still performing better than the basic.



1.6 Efficiency

In terms of efficiency, we believe that as the size of our grids increases, it takes longer and longer for our program to run. For example, we were able to run our code with $10x10$ matrices and $15x15$ within a reasonable amount of time, but as we increased our size to something like $30x30$, it would become hard to even run one of these mazes in under 2 minutes. This is partially an implementation specific constraint as we call the function 'spaceMine' several times over and over again throughout the code, and this likely slows the program down. The problem with this is that we were taking in our entire agent maze knowledge base each time we ran the program, and analyzing the entire grid each time was very time consuming. This problem could be solved by splitting the board in half and solving half the board at a time, or by only taking a part of the board into the 'spaceMine' method.

2. Bonuses

2.1 LaTeXed Report

This entire report was written using LaTeX

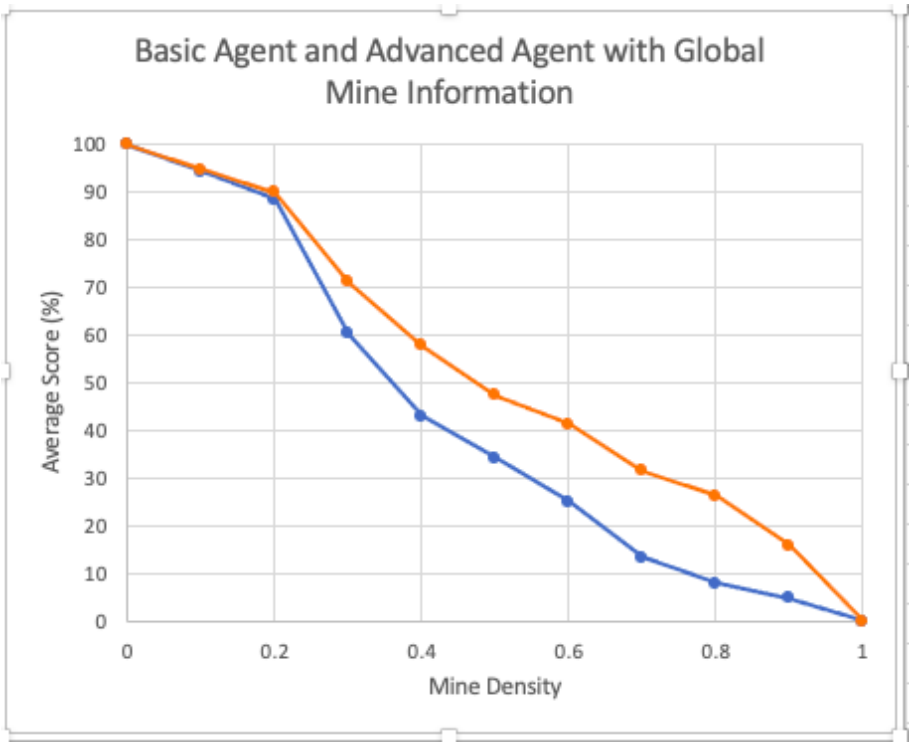
2.2 Clever Acronym

We named the basic agent SCAN (Single Cell Analyze Neighbors) and the advanced agent CANNON (Check All Neighbors of Neighbors Over Neighborhood)

2.3 Global Information

In order to incorporate the additional factor of knowing Global Information in our program, we factored an additional case into the method 'spaceMine' in which we checked whether the total number of mines had been reached already. Since space mine is constantly being called throughout the program, as soon as the number of certain mines equals the number of given mines, the program ends and the final score is calculated. Using this, we are able to lower the time complexity of our algorithm as we no longer need to continue calling methods once this number is reached, as we know every unopened value will then be safe.

Based on the graph below, we can see that the overall scores are similar for both the advanced agent and the basic agent, and the main thing that this addition helped with is lowering the time complexity, especially for larger dimensions. However, as we can see with the graph, the difference between the two agents with the global information added is more significant; The advanced agent is performing relatively better than the one without the global information. This is as the advanced agent looks at multiple cells at once and can identify more mines each step. Therefore, the total mines can be identified sooner and the program can deem the remaining cells safe sooner. With this, the chance of randomly selecting a mine when choosing a new step is decreased and therefore the scores increase.



2.4 Division of Work

All of the work was done over Zoom calls as a group so all of the work was split evenly (Acknowledged by Maitri Shah, Kajol Bhat, and Ashni Patel)

2.5 Honor Statements

Maitri Shah (ms2804): On my honor, I have neither received nor given any unauthorized assistance on this assignment. All of the code written is my own.

Kajol Bhat (kkb56): On my honor, I have neither received nor given any unauthorized assistance on this assignment. All of the code written is my own.

Ashni Patel (agp96): On my honor, I have neither received nor given any unauthorized assistance on this assignment. All of the code written is my own.