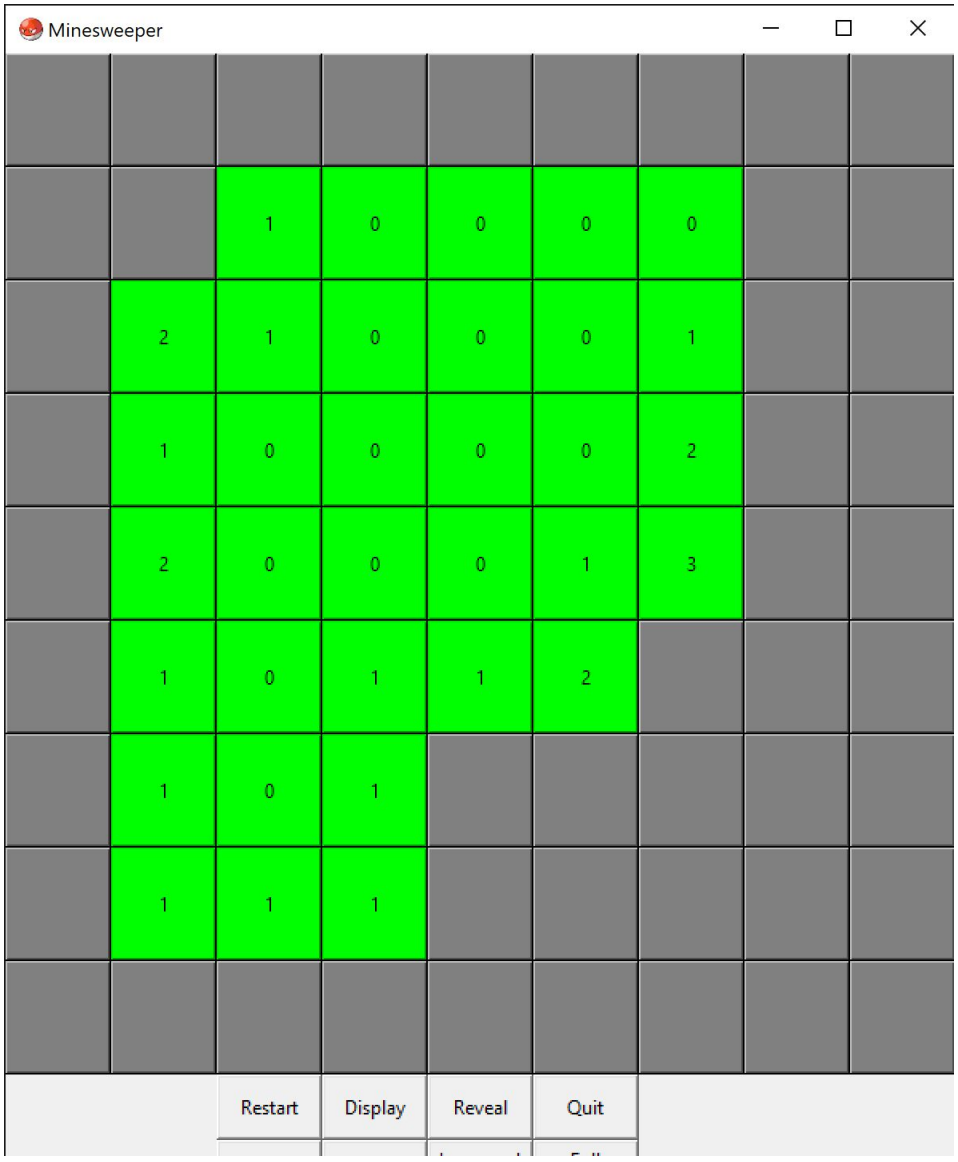
Assignment 2.5: Minesweeper Part 2

Decision Making



Jonathan Lin (jkl130) & Jacob Brown (jab860)

04.11.2020

INTRO TO ARTIFICIAL INTELLIGENCE

Project Overview

Minesweeper is a single-player puzzle game that presents itself with a square grid landscape of cells to be uncovered by the player. This assignment serves as the extension of Assignment 2 to determine the best place to click next if a scenario calls for it, even if there was nothing more that could be inferred.

Language(s): Python 3.7

Packages:

- numpy - allows multi-dimensional arrays, allowing tuples to be used as keys when referencing information
- random - used when generating a new game board and when deciding a random cell to pick if forced upon
- tkinter - GUI-based library used to represent the game of Minesweeper

Recap

As a recap, Assignment 2 for Minesweeper required the implementation of a basic agent and an improved agent. The agents knew their next move whenever they had cells in their knowledge base to mark as “safe” or “mark”.

“safe” cells were determined if the difference between the number of safe neighbors and the number of revealed safe neighbors is the number of hidden neighbors for those particular cells. “mark” cells were determined if the difference between the total number of mines and the number of revealed mines is the number of hidden neighbors for those particular cells. In the basic agent, “safe” and “mark” cells were only updated for each move made without retroactively checking each cell to infer any additional “safe” or “mark” cells that our improved agent would otherwise cover.

The basic agent algorithm from Assignment 2 involved implementing a baseline strategy that, for each cell, keeps track of whether or not it is a “mine” or “safe” or currently covered, the number of mines surrounding it indicated by the clue if safe, the number of safe squares identified around it, the number of mines identified around it, and the number of hidden squares around it.

The improved agent algorithm inherits all of the baseline strategies from the basic agent

with the addition of rechecking every uncovered cell if there are no more “safe” or “mark” cells left in the knowledge base to infer from initially. This was meant to check if there were any additional cells that could be inferred from already uncovered cells instead of choosing a random cell. This agent would only choose a random hidden cell if there are no “safe” or “mark” cells in its knowledge base to choose from even after these additional checks.

This project uses the same properties as Assignment 2.

Minimizing Cost

This agent builds off from the improved agent by adding another scenario whenever there is nothing definite that can be determined in terms of picking “safe” and “mark” cells. Instead of choosing a completely random hidden cell, probabilities are laid out on all of the hidden cells.

These probabilities are calculated based on the number of combinations in a scenario. These scenarios include the number of combinations in which a set of hidden cells contains a mine. We deduced the probability of hidden cells based on the following:

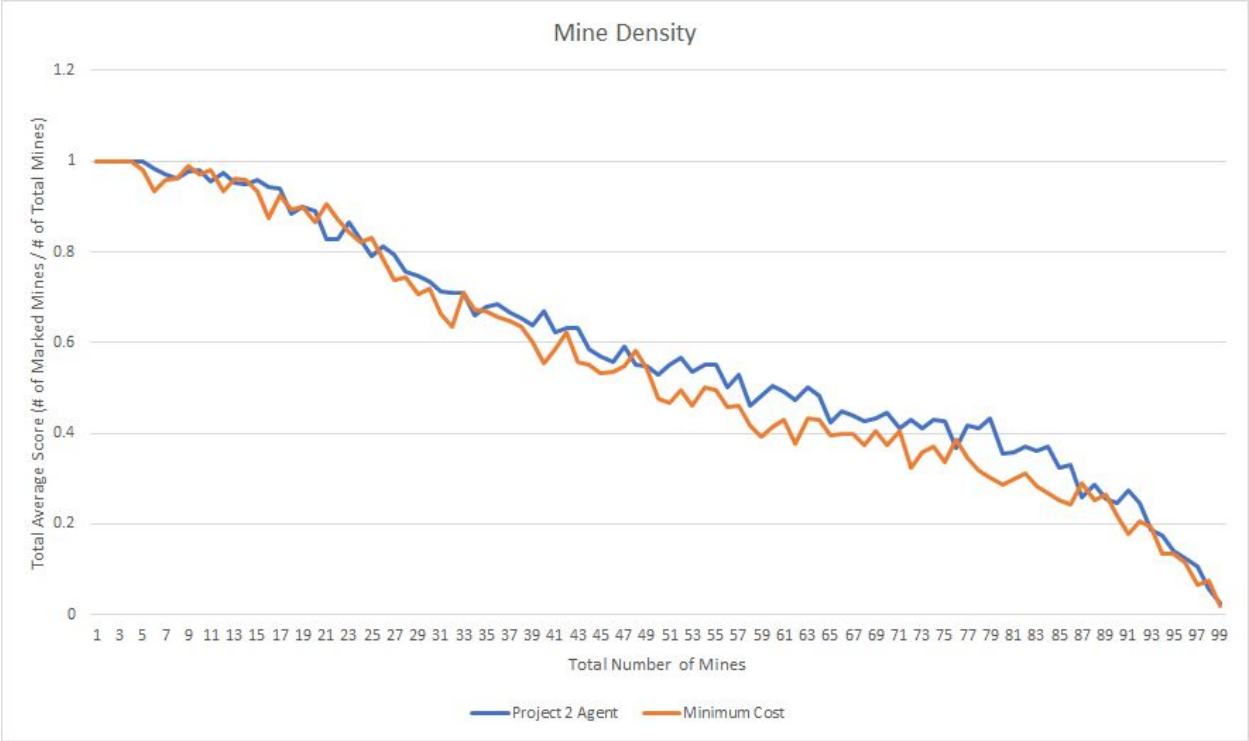
- There is a 0% probability of picking a hidden cell that has no revealed cells around with any clue values to deduce from.
- If there are any revealed cells, provided they are not bombs or marked cells, surrounding a particular hidden cell, take the product of all probabilities of the particular cell being a safe cell from the perspective of each neighboring cell. Then, using the result of the final product, subtract that number from 1. This will result in the total probability of the cell being a mine.

In terms of how the probability of a hidden cell being a safe cell is determined, the hidden cell checks its eight neighbors. If any of the neighbors is revealed and has a clue value, that neighbor will compute the probability of the hidden cell being a mine. Using this number to subtract from 1 yields the probability of the hidden cell being a safe cell. This repeats until all of the neighbors have been traversed.

Considering the agent has no knowledge of the number of mines in the game, we decided to assign a probability of 0% to any hidden cell that lacks revealed cells including a clue value. This is mainly to simulate a scenario in which you would rather pick a different cell as opposed to a cell adjacent to the cells that have already been revealed. There is no way to determine the real probability of a hidden cell that lacks revealed cells including

a clue value without knowing the total number of mines.

Below is a plot of mine density vs. average cost.



How does the slightly improved agent compare? Why? How frequently is the 'improved' decision making necessary?

As shown in this plot, the results from the minimum cost agent are relatively similar to the algorithm from Assignment 2. The results were projected on a 10x10 game board featuring a certain number of mines ranging from 1 to 99 bombs, performing ten trials for each number of mines. It is also important to note that the agents do not have any knowledge of the total number of mines throughout each game.

Overall, due to the nature of hidden cells having a probability of 0% if there are no revealed cells surrounding those cells, this agent would not be utilized nearly as much as we would have hoped. We found that even with this improved agent, the improved decision making wasn't improving the performance of our algorithm. Had the agent known about the total number of mines of the game board throughout the entire game, perhaps certain cells would have been chosen over cells that would otherwise have a 0% probability.

An additional finding to this was that this improved decision making was most utilized in

the near-end of each game. This is mainly due to the low number of hidden cells left to pick as well as the low number of hidden cells with no surrounding revealed cells left. This means there would be fewer hidden cells that have a probability of 0%. In situations where there are strictly only hidden cells that have a non-zero probability, the improved decision making becomes necessary.

Minimizing Risk

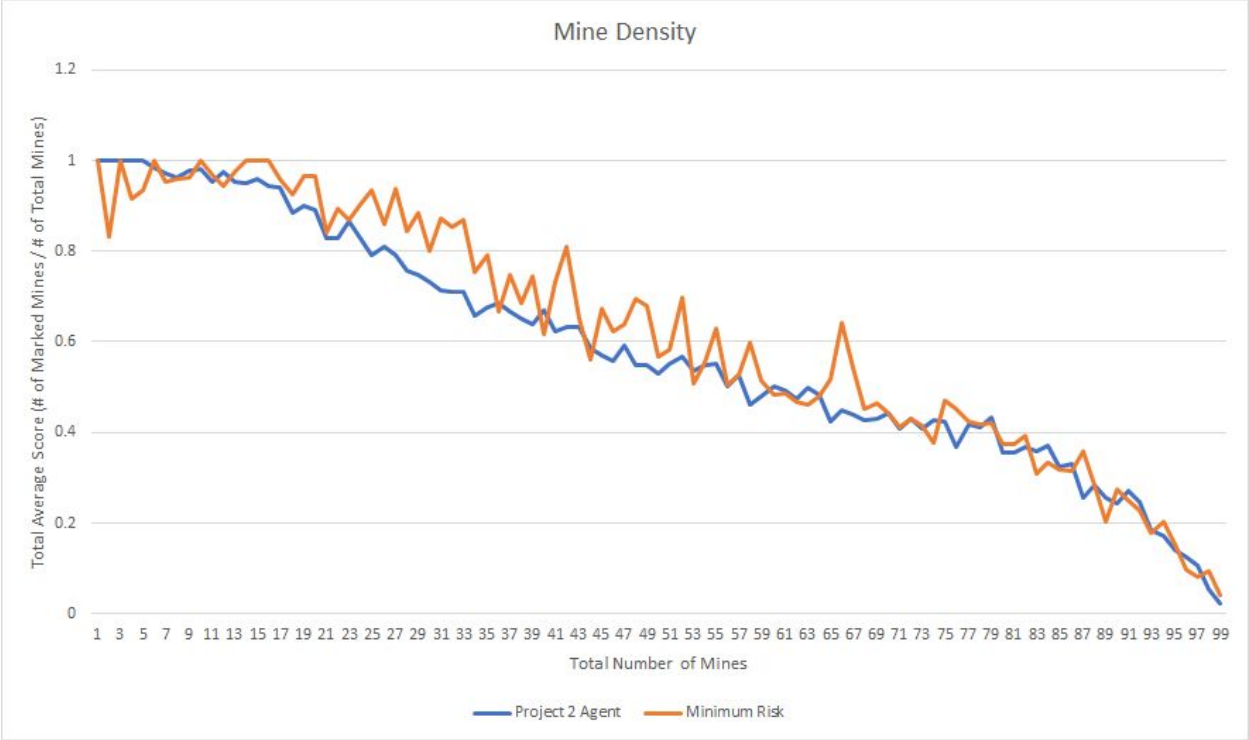
This agent builds off from the improved agent by finding the number of cells that could be worked out from a hidden cell as well as including the features used in the minimizing cost agent. By knowing the expected number of solvable squares from each hidden cell, the hidden cell with the highest expected number of solvable squares is chosen instead of choosing a completely random hidden cell when no other definite move can be made.

The expected number of solvable squares is determined based on (q * R) + (1 - q) * S, where q is the probability of finding a mine in a particular hidden cell, R is the number of solvable squares if the particular hidden cell were to be a mine, and S is the number of solvable squares if the particular hidden cell were to be a safe cell.

q is determined using the same features as the minimizing cost agent, where any hidden cell that has no revealed cells surrounding it has a probability of 0%. If there are any revealed cells, provided they are not bombs or marked cells, surrounding a particular hidden cell, take the product of all probabilities of the cell being a safe cell from the perspective of each neighboring cell and subtract the result from 1.

The number of solvable squares determined provided a hidden cell were to be either a mine or safe cell is calculated by using a modified version of the pickle() function in our code. This function is called infer(), which will check all revealed cells to see if there are any additional cells that can be added to the knowledge base. The number of cells that can be found, given that the hidden cell that is being calculated for as well as assuming the cell is either a safe or mine cell, R or S depending on whether the hidden cell is treated as a mine or safe cell respectively.

Below is a plot of mine density vs. average cost.



How does the slightly improved agent compare? Why? How frequently is the 'improved' decision making necessary?

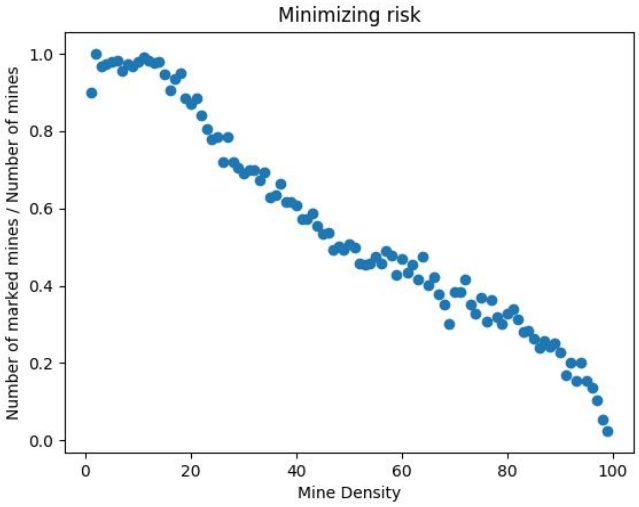
The minimizing risk agent managed to perform better than the agent from Assignment 2 on average. Similar to the minimizing cost agent, the results were projected on a 10x10 game board featuring a certain number of mines ranging from 1 to 99 bombs, performing ten trials for each number of mines. The agents do not have any knowledge of the total number of mines throughout each game.

Determining the number of expected solvable spheres given whether a hidden cell would be a mine or a safe cell offered some additional insights on which hidden cell should be chosen given the highest number of solvable spheres expected. In a sense, this agent was capable of slightly seeing into the future even if it had to rely on probabilistic selection of hidden cells. This proved to be quite useful throughout the early game. In the early parts of the game when clue cells would be more difficult to deduce clues off from to put in the agent's knowledge base, the minimizing risk agent was able to compute the expected number of solvable spheres given a scenario which the hidden cell is a safe cell and a scenario which the hidden cell is a mine cell. In the near-end parts of the game, any leftover hidden cells in which no definite move can be made would often result in

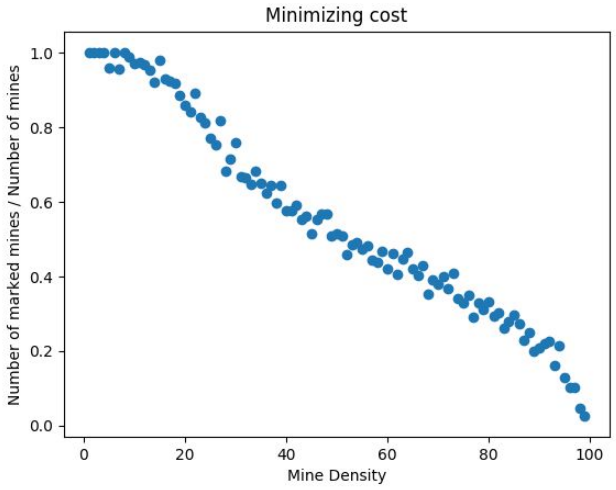
probabilities that are too close with each other, which is often more prone to picking a mine.

Minimizing Cost vs. Minimizing Risk (Bonus)

The comparison of the risk minimizing algorithm and the cost minimizing algorithm requires a granular perspective. This is due to the fact that both algorithms are very closely related and only differ in the way they choose the next 'optimal' cell. Here, optimal means the cell with the highest priority with respect to the given heuristic. To do this, we analyzed the time and accuracy of our results. Our testing parameters are as follows: on a 10 by 10 grid, run 10 iterations of each algorithm on a maze with n mines, where n is an integer in the set {0, 1, 2……99}. We plotted the averages for each number of mines, calculated the slope and also measured the runtimes of each benchmark.



Slope: -0.00915, time: 57.86 secs

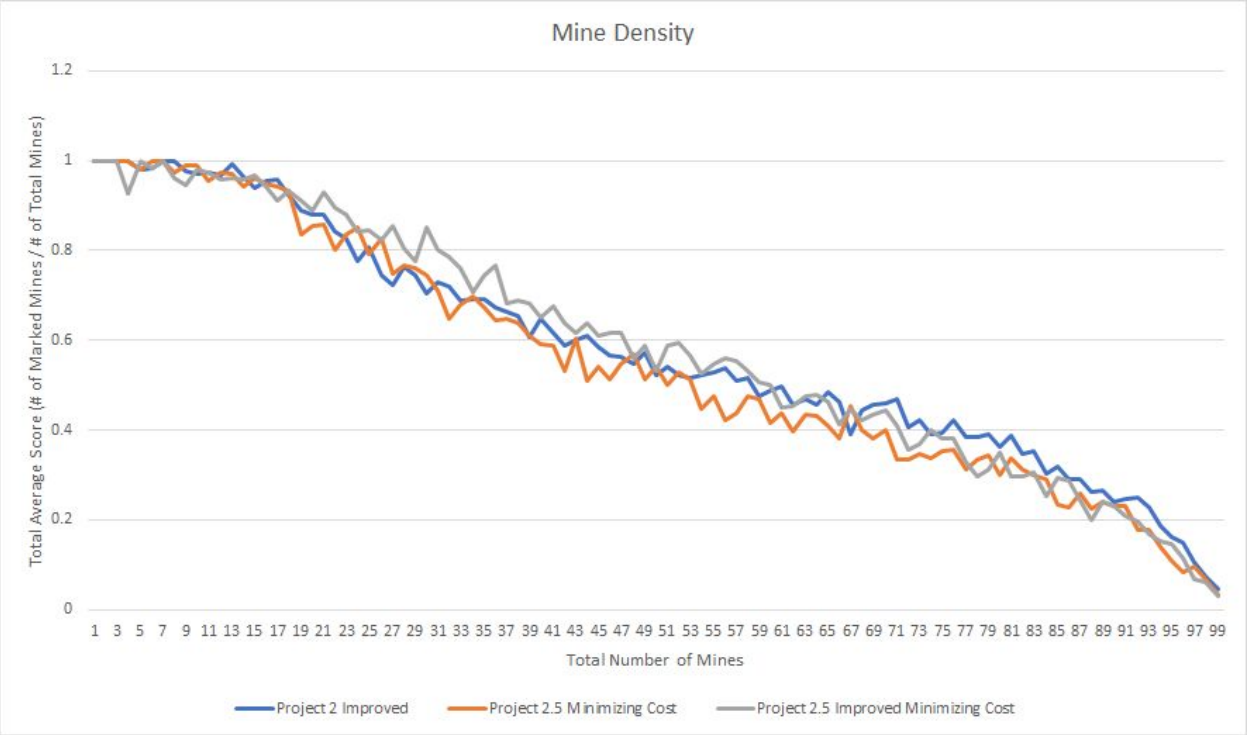


Slope: -0.00919, time: 55.575 secs

Therefore, for 1000 samples for each algorithm, the cost minimization algorithm has about a 4% advantage with speed and our risk minimization algorithm has a .45% “accuracy” advantage.

These results come as no surprise as the cost minimization algorithm has less steps. The extra step in the risk minimization strategy seems to have an effect, but that effect is about one tenth of the effect that the cost minimization time savings has. Depending on the engineering circumstances, it might be the case that you value speed over accuracy of vice versa. In our case, if we were to choose an incredibly large dimension space for our grid, we could be bottlenecked by the additional overhead of the risk minimization strategy. Alternatively, the inferences that the risk minimization algorithm have in place not necessarily provides the same rate of benefit as before. Due to the larger problem space, the risk minimization would be expected to gain a proportional accuracy benefit over the cost minimization function. In the grand scheme of things, the performance and time benefits achieved are relatively small but are worth acknowledging for the performance and accuracy implications that we have observed here: cost minimization does incur a negative consequence to its overall accuracy and risk minimization incurs a negative consequence as it pertains to its time usage.

The Improved Agent (Minimizing Cost)



With the provided plot comparing the improved agent from Assignment 2, the minimizing cost agent from Assignment 2.5, and the improved version of the minimizing cost agent, the results seem to show that the improved minimizing cost agent has slightly better performances in comparison to the other two algorithms.

The improved minimizing cost agent adds another layer of determining the probability that a hidden cell is a mine. Under normal circumstances, none of the agents have any knowledge of the total number of mines there are on the board throughout the entire game. Since the normal version of the minimizing cost set the probabilities of hidden cells with no revealed cells surrounding them to 0%, the minimizing cost seemed to focus on hidden cells that were not near any of the revealed cells that have non-zero clue values during the early parts of the match. In the improved version of the minimizing cost agent, the agent now assumes that there is a 50% chance that a hidden cell with no surrounding revealed cells. The agent knows that the game board can have anywhere between 0 to 100 mines in a game on a 10x10 Minesweeper board, and while the agent does not know the exact number of mines, the agent can approximate by using the average between 0 and 100, which is 50. Once again, in a 10x10 Minesweeper board, 50 is

half of the amount of total cells on the board, which allows the agent to be more aware of choosing hidden cells that could potentially contain more information rather than choosing a hidden cell that has no neighboring cells surrounding it.

Unfortunately, while this improved agent yielded better results in comparison to the minimizing cost agent from an earlier section, the results were not significantly better. Our improved agent from Assignment 2 had stronger inference to be able to deduce cells outside what the basic agent from Assignment 2 had to offer, which means that there's relatively little room for more optimization.

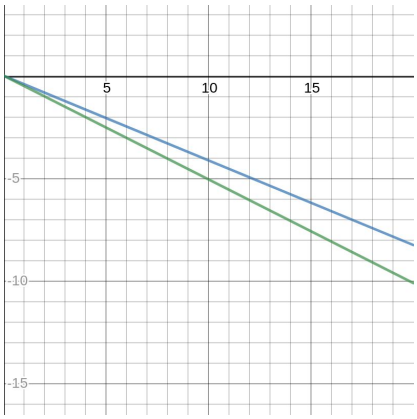
If the improved agent were to be better optimized, perhaps a better way to algorithmically approximate the number of mines in a game without knowing exactly how many mines there are in a game board. This could infer from revealed cells that have clue values to conclude on how they relate with each other. This way, the agent would also be able to have better decision making in terms of calculating the probability that a hidden cell is a mine and revealing the least probabilistic hidden cell.

But suppose that the agent knew the total number of mines on the game board throughout the entire game. This would mean that the agent has a probability of (total number of mines left / total hidden number of cells) for each hidden cell that lacks surrounding revealed cells.

The Improved Agent (Minimizing Risk, Bonus)

To improve the risk minimization algorithm I employed weighting certain factors in the calculation. Namely, I weighed the value of the inference gained by a clear cell twice as much as the inference gained by picking a cell with a mine on it **(⅔ vs ⅓).** The results showed a decrease in the slope of a regression line of about 18.5%. The slope of this line was based on x = number of mines and y = the ratio of marked mines over the total mines. The results were calculated from the average of 300 trials. The rationale behind choosing this approach was to "reward" safer choices, in essence making the effect of revealing more information from a clear cell twice that of the effect of revealing more information if that cell were a mine.

Here are the two regression lines, zoomed in to showcase their divergence:



The green line is the unimproved version, the blue line is the improved version.

Unimproved slope: -0.005061910176913696

Improved slope: -0.004132839257644055

I am very impressed with the results, sometimes the simplest solutions can yield the best results: there is less room for confounding variables to have an effect if the model is inherently simpler.

Additionally, this approach incurs essentially no additional time / resource costs (two additional multiplications per round) over the unimproved version. The time difference in both was measured and found to be negligible.