Sana Naik, Jeet Gupta
Net IDs: ssn60, jdg231
Introduction To Artificial Intelligence
9 April 2025

Project 2: Writeup

1) Explain how the space rat knowledge base should be updated, based on whether the bot receives a ping or does not receive a ping.

> NOTE: There is a correct answer for this. Work out the math, and give the necessary update formulas.

1. Initially, all open cells are added to the knowledge base (rat_kb) and initialized with a uniform probability (1 divided by the number of open cells). The search proceeds as follows:

   The sensor tells us directly whether or not the bot is in the rat cell. If it is, the knowledge base collapses and we return success.

   If the bot does not receive a ping in the current cell, the probability of the cell containing the space rat should be zero.

   P(rat in bot cell | no ping) = 0

   Then, the other cells in the knowledge base would be updated like so:

   P(rat in cell | no ping) = P(no ping | rat in cell) * P(rat in cell) / P(no ping)

   In code, for each cell:

```
prob_ping = 1 - np.exp(-alpha * (dist - 1))
updated_prob = prob_ping * prob
updated_rat_kb[cell] /= sum_prob
```

   prob = preexisting belief that the rat is in that cell
   prob_ping = P(ping | rat in cell) = 1 - np.exp(-alpha * (dist - 1))
   The denominator is just a normalization constant (i.e., the sum over all cells), done at the end by dividing all updated_rat_kb entries by sum_prob.

   What this does is use the Manhattan distance from the bot to each cell in the knowledge base d, and set its new probability of having the rat P(rat) to (1 - e^(−α(d−1))) * P_{prior}(rat) / the sum of all new probabilities. If a ping is not heard, then cells farther away from the bot cell increase their likelihood of containing the rat, while cells closer to the bot cell decrease their likelihood of containing the rat.

If the bot DOES receive a ping, the probability of each other cell in the knowledge base containing the space rat will be updated as such:

P(rat | ping) = P(ping | rat) * P(rat) / P(ping)

In code, this looks like this for each cell:

```
prob_ping = np.exp(-alpha * (dist - 1))
updated_prob = prob_ping * prob
updated_rat_kb[cell] /= sum_prob
```

If the ping is detected, P(rat is in the cell) = e^(−α(d(i,j)−1)) * P_{prior}(rat in the cell) / the sum of all new probabilities

So, if a ping is heard, cells closer to the bot become more likely to contain the rat, and distant cells become less likely.

2) Explain the design and algorithm for the bot that you design, being as specific as possible as to what your bot is actually doing. How does your bot make use of the information available to make informed decisions about what to do next?

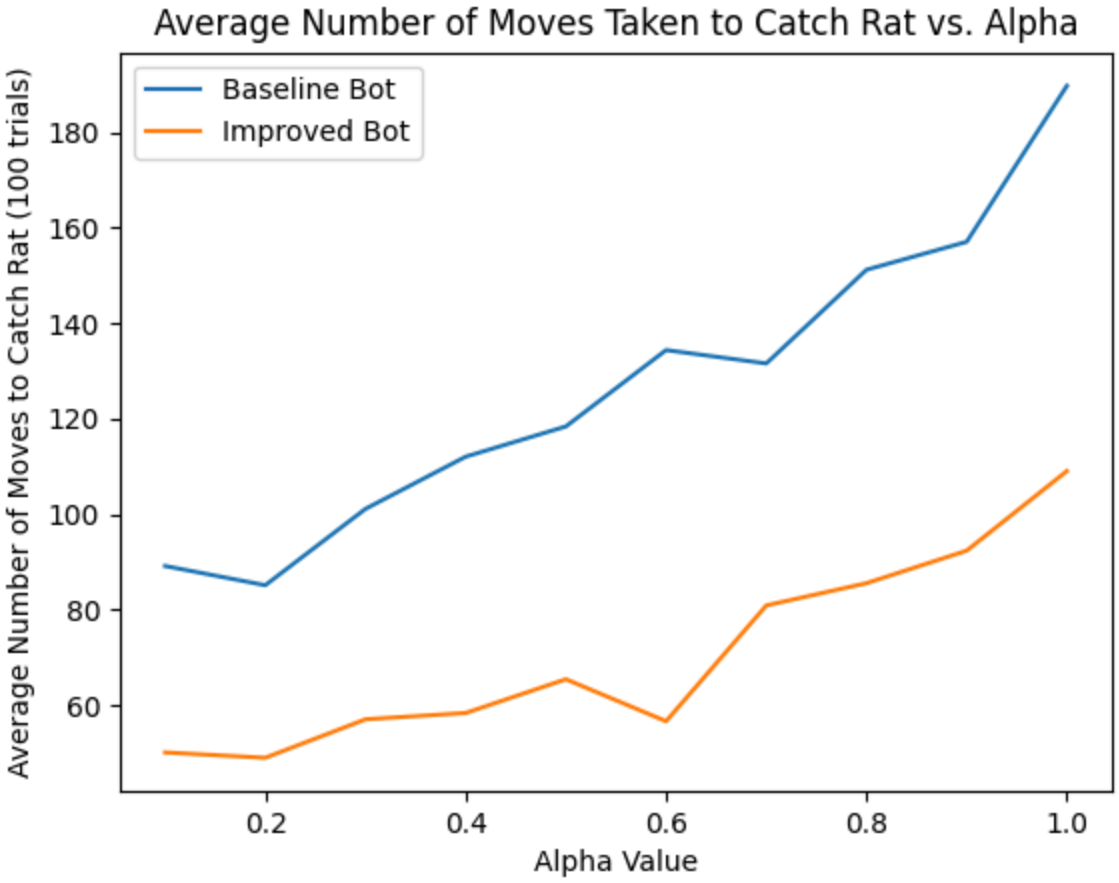
2. For our own bot, we decided to try and reduce the amount of moves used to get to the space rat. To do this, we estimated the position of the bot using the same function as before, and then set an initially equal probability for each cell like the baseline bot. We updated the probabilities of each cell by pinging 10 times for every bot move. We decided to ping 10 times before moving so that we would get a better estimate of the probabilities for each cell before picking a target cell, since we did not want to calculate inaccurate probabilities and waste moves going to the wrong cell.

   Furthermore, while the baseline bot selects a target cell with the maximum probability and travels to that cell before switching targets to a cell with the new maximum probability, our bot switches targets as soon as another cell passes a certain threshold of probability above our target cell. Through experimenting with thresholds between 1.0 times up to 2.0 times the probability of the current target cell, we settled on switching targets once a cell became 1.25 times likelier to contain the rat than the current target cell, which yielded the best performance (resulted in the least moves being used)

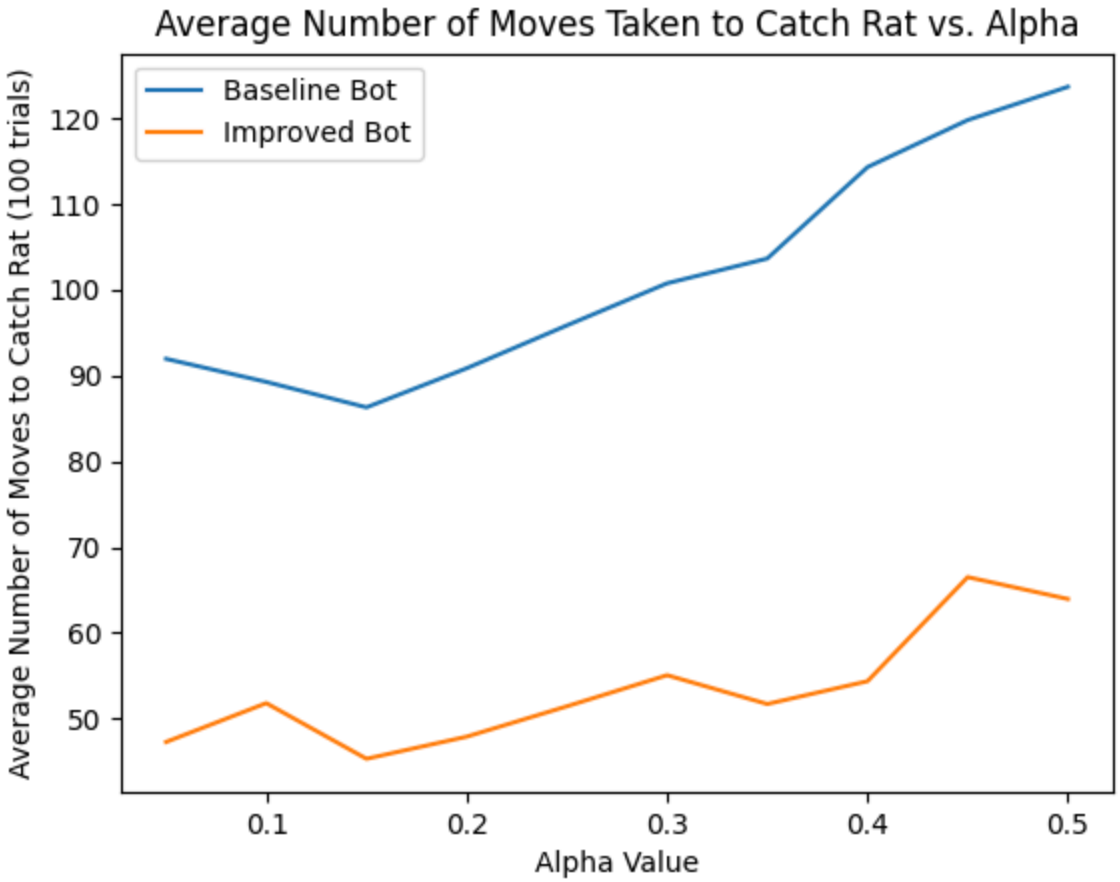
   The last idea we implemented was to get the average probabilities of a cluster of cells, rather than travelling to the single cell with the highest probability. This was to make sure that we wouldn't set the target cell as one which just had an anomalous/coincidental high number of pings, but would rather travel to a cell whose neighbors all shared a high average probability of having the space rat. This also decreased our number of moves by reducing the likelihood of setting a false target cell.

3) Evaluate your bot vs the baseline bot, reporting a thorough comparison of performance. Plot your results as a function of α.

> Note that as α gets larger, the probability of hearing a ping ever gets smaller and smaller - the sensor provides less and less information. Similarly if α is too small, you always hear a ping - which means the sensor is also providing less information. Empirically, I find the most interesting values of α to be between 0 and 0.2.



3. As seen through this plot, the improved bot using the strategies discussed in part two uses much less moves than the baseline bot to catch the rat. These results were averaged for 100 trials for a 30x30 dimensional ship, for alpha values from 0.1 to 1 at steps of 0.1. We can zoom in and experiment with more alpha values between 0 and 0.5.



As visible from this plot, a similar pattern appears as the first plot. While the improved bot performs much better than the baseline bot overall, as the alpha value increases, the sensitivity decreases. So after alpha 0.25, the number of moves for both bots seems to increase seemingly exponentially, as the sensor provides less and less information. However, at lower alpha values, the performance seems to also slightly tail up, because with a very high sensitivity (low alpha value) it also provides little information when a ping is heard.

4) Previously, the space rat was assumed to be stationary. Now assume that in every timestep, after the bot takes an action, the space rat moves in a random open direction.

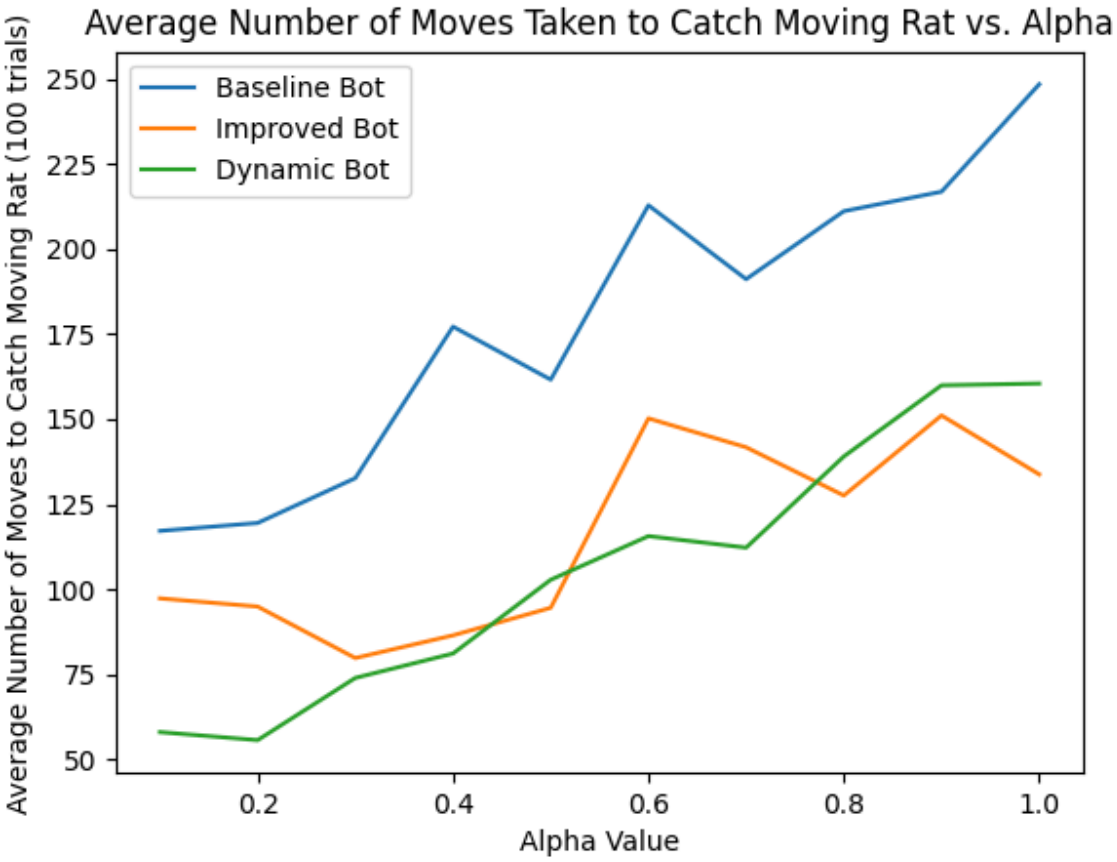
   - Work out the space rat knowledge base updates based on this. What changes, what probabilities do you need to calculate? Again, there is a correct answer.
   - Simulate and compare both bots in this new environment. How does the performance change? Again compare across $\alpha > 0$.
   - Try to improve on your bot design to be even more effective in this moving space rat situation. What did you need to change, if anything?

4. If the space rat is moving around the ship, we will need to use a transition model. That is, we will need to calculate the probability of the rat being in cell X at time t+1 given the rat is in cell Y at time t. This can be modeled as a MDP (Markov Decision Process), which was discussed during class. The rat's next likeliest next positions (transition probabilities) depend on its current position (current state) and its open neighbors (action sets).

The ping update remains the same as our formula from section 1. If the ping is detected, P(rat in cell) = $e^{(-\alpha(d(i,j)-1))} * P_{prior}$(rat in cell) / the sum of all new probabilities. If the ping is not detected, P(rat in cell) = $(1 - e^{(-\alpha(d(i,j)-1))}) * P_{prior}$(rat in cell) / the sum of all new probabilities.

However, now after the ping update, we also take into account the probabilities of the rat moving to each cell around it during the next move. So, for each cell, the new probability $P(\text{rat in cell x})_{new} = SUM_{\text{cell y in x's open neighbors}}(P(\text{rat in cell y}) / (\text{num open neighbors of y}))$. In code this would involve something like, after the ping update, for each candidate cell:

1. Get its neighbors
2. For each neighbor:
    a. Add the probability of the cell / the num of neighbors
3. Divide this new probability by the sum of the new probabilities (normalize)



Looking at the bots with the moving rat implementation, the baseline bot had a considerably high move count compared to the other two. This is most likely because the baseline bot doesn't recalculate the path when the bot moves and instead stays toward a target destination before moving somewhere else. The improved bot is slightly worse than the dynamic bot at most alpha values. The most likely reason for this is that it doesn't implement a redistribution of probabilities as well as prediction towards the possible next move of the rat.

In order to improve on the bot design to be more effective with the moving rat, here are some of the changes we made:

- Implemented the new knowledge base updates. This means that instead of calculating probabilities for each cell solely on pings, we would also take into account the rat moving to one of its neighboring cells, as explained above.
- Reduced the number of pings called before picking the bot's next move. Because the rat is moving, there is no need for the initial calculated probabilities to be extremely accurate.
- Removed the "probability threshold" to change target cells. This is because there is no target cell - the bot only plans one cell ahead on each move, rather than setting a target to move to.