Assignment 2

Project 3: The Meanest Sheep in the World Who Is Also Near Sighted

Deadline: Friday, July 11, 11:55pm.
Perfect score: 100.

Assignment Instructions:

Teams: Assignments should be completed by teams of up to three students. You can work on this assignment individually if you prefer. No additional credit will be given for students that complete an assignment individually. Students are responsible for staying engaged with their team in a timely and effective manner, and will face penalties if they fail to do so. Students may change teams between projects. All team members are responsible for their team's submissions and adherence to academic integrity policy.

Make sure to write the name and RUID of every member of your group on your submitted report.

Submission Rules: Projects should be submitted to Canvas as a zip file. The zip file should include all code written for the assignment, as well as your project report in PDF form. Do not submit your report as a Word document or raw text. Each team of students should submit only a single copy of their solutions and indicate all team members on their submission. Failure to follow these rules will result in lower grade for this assignment.

TA Meetings: Each team will sign up for an appointment via the course Canvas site and videoconference with a course TA before submitting their project. Each team will be allotted 10 minutes, with the meeting counting for 10% of the project grade. The TA will verify the team's progress and approach to the assigned tasks, and direct students to resources, as needed. As stated above, students should have completed the early steps of the project before their progress report meeting. Some teams may be asked to meet again post-project to discuss their submission, at the discretion of the TAs. It is essential that you reply promptly to TA inquiries and adhere to your team's designated meeting times.

Please write who you had your TA meeting with in your final report.

Precision: Try to be precise, especially when writing proofs. Computer science is at its heart a branch of mathematics, and I expect the same level of rigor from your proofs that I would in an upper-level math class. Each step in a valid should be an application of a known mathematical identity. Handwaving, direct assertion, or arguing that something is true because it would make sense, is not a valid proof, and will be graded as a zero. For non-proof questions, write as if you are trying to convince a very skeptical reader.

Collusion, Plagiarism, etc.: Each team must prepare its solutions independently from other teams, i.e., without using common notes, code or worksheets with other students or trying to solve problems in collaboration with other teams. You must indicate any external sources you have used in the preparation of your solution. **Do not plagiarize online sources** and in general make sure you do not violate Rutgers' academic integrity policy, which can be found here: `https://global.rutgers.edu/academic-integrity-rutgers`. Failure to follow these rules may result in failure in the course.

Some cases of academic misconduct occur because the student gets over-rushed and desperate. Other times, the student is simply at a loss for how to proceed. To avoid this situation, always start with the actual project guidance and class session content in completing assignments – these resources are provided for a reason. Second, make sure that you stick with the recommended timeline for your work, and reach out to me and the TAs when you are unsure about how to proceed. Assigned projects are to ensure that you are following the provided material and doing the required work for the course, and, therefore, closely parallel the course content.

Assignment Description

You are a sheepdog robot, and you are responsible for corralling the meanest sheep in the world. While most sheep you're able to herd and guide where they're supposed to go, this sheep is big, mean, and hates robots. If the sheep sees you, it will try to charge you, and if it hits you, it will absolutely obliterate you - a combination of poor craftsmanship, perhaps, and the fact that this sheep just works out thinking about ways to destroy robots.

But you have two advantages - one, you're more agile and can move in ways this sheep cannot; two, the sheep is somewhat blind, and if you get outside its field of view it will lose interest in you.

That being so, you come up with the following strategy: perhaps you can trick the sheep into chasing you, and in doing so lead the sheep into the pen. But that is going to require planning, to know what to do (where to move)when the sheep moves, etc. And, being scared of this sheep, you'd like to get it into pen as fast as you reasonably can. Use value iteration to find the optimal solution.

1 Implementation

1.1 1.1 The Field

The grazing field for this project is a 31 x 31 square grid. The robot and the sheep can occupy cells in the grid and move between them. The middle square in the grid is surrounded by a U of blocked cells, as below:

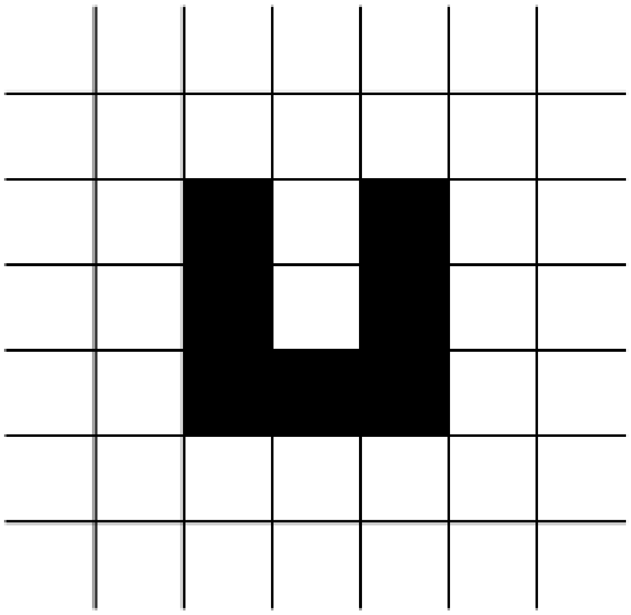

Figure 1: Grid Center

This U is the pen - the goal is to get the sheep into that center node (at which point your ally will slam the gate,trapping the sheep).

1.2 1.2 The Sheep

- The sheep occupies one of the cells. It can move up/down/left/right, or stay in place.
- The sheep cannot move out of the 31 x 31 grid, or into any of the blocked cells.
- The sheep's field of view is the 5x5 block of cells centered on the sheep's position.
- If the robot is not within the 5x5 field of view, the sheep will pick uniformly at random between any of its currently valid actions.

- If the robot *is* within the 5x5 field of view, the sheep will pick randomly between actions that take it closer to the robot (charging the robot).
- If the sheep enters the center cell, at the middle of the pen, the sheep is trapped and the game is over.

1.3 1.2 The Sheepdog Bot

- The robot occupies one of the cells. It can move to any of the 8 neighboring cells (if unblocked/empty), or stay in place.

2 Sheepdog Bot 1

We can consider the optimal sheepdog bot in the following way: letting $state$ be the current configuration of the field(bot and sheep positions), define $T^*(state)$ to be the expected number of moves an optimal bot would need to catch the sheep. If the sheep cannot be captured for a given state, $T^*(state) = \infty$

1) What states are easy/immediate to determine $T^*(state)$ for?

2) For any other given state state, express a formula for $T^*(state)$ in terms of

  a The actions the bot can make

  b How the sheep responds, and

  c What state results. What are special cases you ought to consider?

3) If you could determine $T^*(state)$ for each state, how would you determine the optimal action for the sheepdog bot to take?

4) Compute $T^*(state)$ for every state. If the sheep starts in the upper left corner, and the robot starts in the lower right corner, what's the expected number of moves needed to capture the sheep?

5) If the robot can start at any location, and the sheep will be introduced at a random empty spot - where should the robot start? Why? What is the expected number of moves to catch the sheep, when started inthis location?

6) Based on $T^*(state)$ and the optimal actions, simulate the sheepdog bot / sheepdog interactions. How does the simulated data compare to the compute expected value?

Note: This could take a long time to run. Make sure you leave plenty of time for your model to finish training. Also, you will probably want to have a way to same your T values. The later parts of the assignment require you to have access to them, and you don't want to have to spend multiple hours recalculating them every time you make a code change in section 2. Do *not* include the saved T values in your submitted report, because they will be vary large.

3 Sheepdog Bot 2

The Sheepdog Bot 2 is based off Sheepdog Bot 1. Note that for sheepdog bot 1, you need to calculate potentially as many as 31^4 different values to fully get T^* and the optimal policy. This is potentially very expensive, memorywise.

For this bot, fit a model $\tilde{T}$ to input/output pairs $(state, T^*(state))$.

1) How should you represent states as input to the model? What kind of model do you want to consider? What input features are relevant? What kind of error or loss are you using to assess your model? What training algorithm are you using? How can you compare the size of the fit model to the size of the fully computed T^*?

2) How can you assess the quality of your model fit? Is overfitting an issue? If so, how can you avoid it?

Sheepdog Bot 2 functions in the following way: in any state, the bot estimates the value of a given action using $\tilde{T}$ to estimate the number of moves remaining to catch the sheep after taking an action; the bot then chooses what action to take based on the smallest estimated moves to catch the sheep.

3) Simulate Sheepdog Bot 2, starting the sheep in the upper left corner, and the robot in the lower right corner.Does Sheepdog Bot 2 reliably catch the sheep? How does it compare in its performance to Sheepdog Bot 1?

4) **Bonus:** Improve the model. Does that improve Sheepdog Bot 2?

4 Sheepdog Bot 3 (*Optional - 50 points*)

Sheepdog Bot 3 learns by watching. Using Sheepdog Bot 1, you can generate a lot of data, of the form (*input state, output action*). Build a model to predict the desired output action from the specified input state.

1) What is your input space? What is your output space? What is your model space? How are you assessing the error, and how are you reducing that error - i.e., what is your training algorithm?

2) How can you assess the quality of your model fit? Is overfitting an issue? If so, how can you avoid it?

3) Simulate Sheepdog Bot 3, starting the sheep in the upper left corner, and the robot in the lower right corner. Does Sheepdog Bot 3 reliably catch the sheep? How does it compare in its performance to Sheepdog Bot 1?

5 Analysis

Along with the above implementation, you need to analyze the performance and the results of your code in a final **lab report**. The report should include the following:

- Description of the algorithm you implemented and the design choices you made
- Answers to the above gray-boxed questions
- Plots and visualizations as necessary to make your answers clear