Project 3: Planning with Uncertainty

CS 6370 / ME EN 6225 Motion Planning
University of Utah

Instructions

All solutions should be uploaded via the assignment on Canvas.

Submission: Please submit two **separate** files: a **pdf** report and a zip file.

PDF Report: Place all written answers to questions as well as any images or other output data used to explain your responses in a single PDF document. This should be clearly named Proj<*number*>-answers.pdf, where number is the Project number (i.e. 3 for this assignment). Please make sure to write your name and uid at the top of the document!

Zip File: Your zip file should be named in the format <*uid*>.zip where <*uid*> is your Utah uid. The first level of the zip should contain a single folder with your uid as the name <*uid*>. Please put the README.txt, code and other necessary files (e.g. map or environment files) in the zip file. The README.txt should be in the root directory of your submitted zip file which explains how to run the code submitted.

1 Markov Decision Process

We will first look at how states evolve in an MDP, ignoring any planning. We'll build on the environment we used for Project 1.

Simulate from an MDP We will first modify the transition function from Project 1 to return a probability distribution over subsequent states instead of a single state. To start we'll use the four actions $A = \{\texttt{left}, \texttt{right}, \texttt{up}, \texttt{down}\}$ on the same maps as Project 1.

There are two version of the transition function we want (I suggest implementing this switch as a simple Boolean parameter for the function). The first version is to be returned the model of probability distributions. I implemented this as a list of 2-tuples, where each tuple holds a probability and state. The second version acts as a simulator and returns a single state which is chosen randomly according to the transition probabilities. To begin with given the robot an 80% chance of making completing the action correctly and 10% chance each of moving in either direction perpendicular to the commanded motion.

1.1 (5 pts) Take the result of running breadth first search on the deterministic system from the start to the goal and run it on your simulator 5 times each for `map0.txt` and `map1.txt`. Does your robot reach the goal? How many times for each map? What else happens?

1.2 (5 pts) Change the probabilities to be 60% of moving forward and 20% each for left and right when commanding to move forward (for all actions). Run the same tests as in 1.1. How many times does the robot reach the goal now?

1.3 (5 pts) Extend the actions to use diagonal actions as well as the four cardinal directions. Change the transition probabilities to be 80% the correct action and 10% each for the neighboring actions (e.g. UP-LEFT and UP-RIGHT for the UP action). Run the same analysis as in 1.1, but also add them form `map2.txt`

1.4 (5 pts) Now change the transition probabilities to have a 70% chance of moving in the correct direction, 10% chance of going to the neighboring direction and 5% chance of going in either orthogonal direction. This is visualized in Figure 1. Run the same tests and analysis as in 1.3.

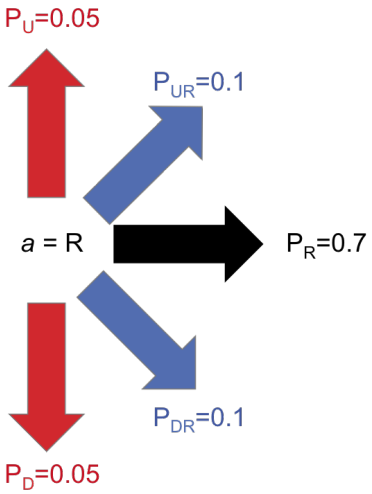

Figure 1: Transition probabilities for 1.4 for the action RIGHT.

2 Value Iteration

In order to complete our MDP, we need to add rewards to the states. To start with just put a reward of 10 at the goal and return a reward of 0 for all other states.

2.1 (20 pts) Implement value iteration. Run VI on `map0.txt` and `map1.txt` with the four actions $A = \{\texttt{left}, \texttt{right}, \texttt{up}, \texttt{down}\}$ and the $[0.8, 0.1, 0.1]$ transition probabilities. Start with a discount factor of $\lambda = 0.8$. Show a map with the resulting values for each state. Show another map of the resulting actions for each state. Discuss the resulting policies qualitatively. Report also the number of iterations required to converge.

2.2 (10 pts) Examine changing the discount factor λ. Run the same scenarios as in 2.1 with different values of λ set to $0.9, 0.5, 0.2$, and 0.0. How does the resulting policy change? How does the number of iterations required to converge change?

2.3 (10 pts) Change the actions to use diagonal actions and use the settings from questions [1.3]. Run VI on all three maps with these settings and report your results as maps of the values and policies found. Choose λ as you wish to give good performance. What value of λ did you use?

2.4 (10 pts) Now run the evaluation using the actions from 1.4. Report the same results as 2.3.

2.5 (10 pts) Now, lets examine changing rewards. We'll do this with the same setting as 2.1. First, change the rewards to have a value of 1 at the goal and 0 everywhere else. What changes with respect to the value and policy? Now give the goal a value of 10 and all other states a value of -1. What happens now? For the final set of rewards, make the goal +10, all corners -5, and all other states 0. What happens?

3 Policy Iteration

We will use the same rewards of value iteration for policy iteration. That is, a reward of 10 at the goal and a reward of 0 for all other states. We will use the four actions $A = \{\texttt{left}, \texttt{right}, \texttt{up}, \texttt{down}\}$ for policy iteration.

3.1 (10 pts) Implement policy iteration. Run PI on `map0.txt` and `map1.txt` with the $[0.8, 0.1, 0.1]$ transition probabilities. Use the action `up` for all states as the initial policy. Use a discount factor of $\lambda = 0.8$. Show a map with the resulting values for each state. Show another map of the resulting actions for each state. Discuss the resulting policies qualitatively. Report also the number of iterations required to converge.

3.2 (5 pts) Now use the action `down` for all states as the initial policy to run PI and repeat the same analysis as 3.1.

3.3 (10 pts) Run PI on `map0.txt` and `map1.txt` with the $[0.8, 0.1, 0.1]$ transition probabilities. Start with a `random` initial policy. Use a discount factor of $\lambda = 0.8$. Show a map with the resulting values for each state. Show another map of the resulting actions for each state. Discuss the resulting policies qualitatively. Run PI 5 times with random policies on both maps and report the number of iterations required to converge for all 5 times. To find a good policy, `notice` that it is important to choose a random action among these multiple actions with the same max Q value (i.e. Q(s,a)) when update the policy.

4 Self Analysis (5 pts)

4.1 (1 pt) What was the hardest part of the assignment for you?

4.2 (1 pt) What was the easiest part of the assignment for you?

4.3 (1 pt) What problem(s) helped further your understanding of the course material?

4.4 (1 pt) Did you feel any problems were tedious and not helpful to your understanding of the material?

4.5 (1 pt) What other feedback do you have about this homework?