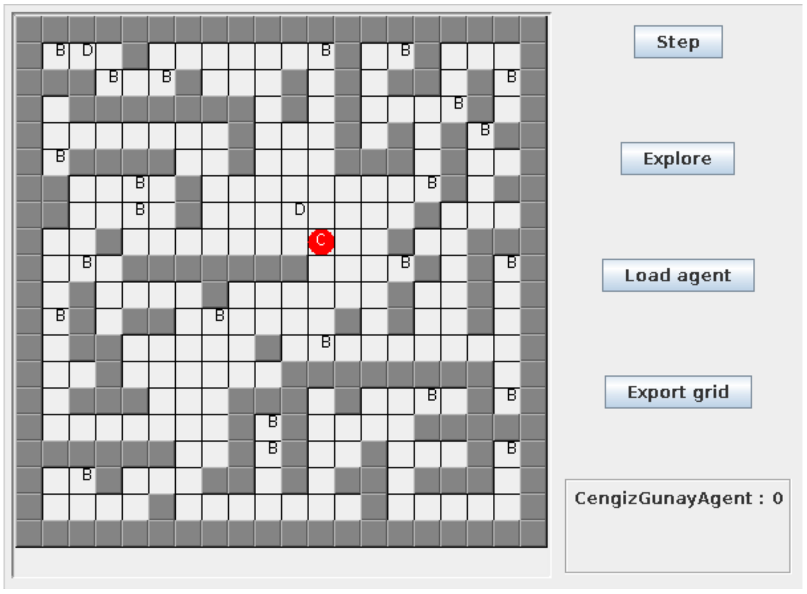
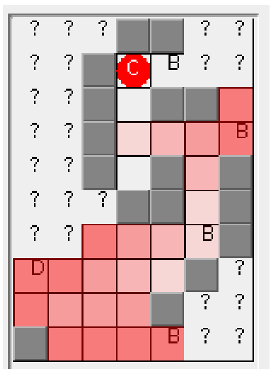
CS325 – HW #5
Grid World Ballgame
Due date: Tuesday 04/02, midnight (11:59pm)

March 25, 2013



(a) Main interface. Step: execute all agents on the grid for one step. Explore: steps agents many times (2,000), or until an agent throws an informational exception (e.g., when it finished exploring). Load agent: will open a dialog to load your agent. It can be chosen multiple times for multiple agents to compete on the same grid. Export grid: will let you save the grid map in a text file. This file can be edited and then loaded from the initial interace window by selecting "Load grid from file". At the right bottom, you can see the agent's current score. You will get a score only after you deposit the balls.



(b) CengizGunayAgent's own memory grid. Darker red colors show higher Q-values during the exploration phase.

Figure 1: Grid-world ballgame. Agent (red circle marked with a "C") needs to explore the grid, collect the balls (B) and return them to the deposit bins (D). An agent can only carry up to five balls at a time (panel a). Your agent will have its own memory of the arena as it explores it (panel b).

The Ballgame is On!

Prepare to compete in the grid world arena against agents of other students. You can create multiple agents on the arena. You will have the chance to program the best artificial intelligence to collect the balls in the arena and deposit them faster than your competitor. Do you first explore the whole arena and find the ball and deposit bin locations? Don't think your competitors will be that patient.

You also have the chance to make new arenas. I provided the `grid/design2.txt` file as an example. You can make arenas that are *impossibly difficult* or just *artistic*, test it with your colleagues, and share them on Piazza. Arena maps can be loaded with the "Load grid from file" button in the grid creation interface that shows up when you run the program (not shown here).

To design your agent, download and unzip the Java package posted together with this PDF file. Rename the `agent/ballgame/JohnDoeAgent.java` to your own name and take it as an example to program your own agent. Look at the comments inside the file to guide you to the architecture of the existing program. You are not allowed to modify the rest of the codebase. I will only run your one Java file in my program.

For your convenience, there is an Ant `build.xml` file that has the `compile` and `run` targets defined. So "`ant run`" will compile and run the program. Alternatively, you can execute the "`VisualFoodHunt`" class using the Java executable (e.g., "`java VisualFoodHunt`") or using an environment like Eclipse.

Note: I am sharing my program with you under the Creative Commons Attribution-ShareAlike 3.0 Unported license. Please remember to attribute me and share it with others if you use this program in the future.

Instructions

1. You need to submit your results in two places: (1) written answers should be submitted as a PDF file to Blackboard; (2) code should be placed under the `cs325/` directory in your MathCS account. Name all your files as with the homework number OR open a new directory. Both in your code and written results, start with a statement with your name on top of your code files saying you agree to the Emory Honor Code.
2. For the code implementation, be sure to include instructions on how to compile and run your code (e.g., in instruction.txt file). In this file, also describe how the input and output of your program must be specified. Use comments in your code so that we can understand and grade it properly. If you're using the available online code (highly recommended), only include the parts that you added or customized. Try to avoid turning in the whole package.
3. Please post your general questions about the homework on the class page at Piazza: `https://piazza.com/emory/spring2013/cs325/home` and categorize them under the appropriate homework directory (e.g., hw1, hw2, ...).

Questions

The total points is 50. The bonus is 15 points this time. As usual, it will count towards your final grade outside of the alloted homework points.

1. (10 points) Briefly answer the following questions in the context of Markov Decision Processes (MDPs):
   (a) What is the difference between deterministic and stochastic actions?
   (b) What the purpose of the "discount factor"?
   (c) What's an "optimal policy"?
   (d) What's the purpose of "movement cost"?
   (e) What's the difference between $V(s)$ and $Q(s, a)$?

	1	2	3	4	5
a		-1	■		$+1$
b	S				
c			■	-1	$+1$

Figure 2: A treacherous grid with two positive and two negative rewards. "S" is the starting point.

2. (15 points) Navigate the grid-world in Figure 2 with an MDP. First choose your own γ value as one of $(0.9, 0.8, 0.7)$ and movement cost R as one of $(-0.1, -0.2, -0.3)$ and solve the following:
   (a) Assume moves are deterministic. Calculate first-iteration MDP values for all the blank positions and define the optimal policy.
   (b) Assume moves are stochastic for a robot whose right tire is flat. It will go forward 50% of the time and to the right 50% of the time. Calculate first-iteration MDP values for all the blank positions and define the optimal policy. Calculate the MDP value for position b5 after convergence.
   (c) For a deterministic RL agent who doesn't know the reward locations, encourage exploration with constant $R = +0.1$. Apply the TD rule by recalculating the optimal policy after each step. Simulate it for 10 steps and output TD update values for each step. Show the final values in the grid and optimal policy for visited locations.

3. (25 points) Implement your own reinforcement learning (RL) algorithm for the grid world program described in Figure 1 and the introduction paragraph above. You will get 15 points for the implementation, and the rest from the following:

   (a) (5 points) Explain how exactly you setup your RL algorithm and chose your exploration vs. exploitation strategy. What was its advantages and weaknesses?

   (b) (5 points) Test your agent with one or more fellow students. Report the arena file used (random may be the fairest?) and the points collected for all competing agents. Win or lose, you will get full points. However, make sure not to share your code with them. It may help that you can use pre-compiled class files instead of using the source code.

i. In the agent's memory, spread the particles in all map locations and set the `qValue` field to visualize the particle weights.
ii. Modify the `move()` method in `grid/Grid.java` to implement the stochastic movement of the agent. Choose your own probability values for the agent to move forward and to perpendicular directions randomly. Use the same probability values when shifting the particles in the agent's memory.
iii. After each move, calculate particle weights based on the 9-square sensory information the agent receives and the consistency across all particles in various map locations.

Table 1: Particle Filter algorithm applied to the grid-world agent.

Bonus question: (15 points) Implement Hidden Markov Model (HMM) location estimation for a stochastic agent in the grid world of Figure 1 using the Particle Filter (PF) algorithm. Assume that after the agent explored the whole map, suddenly it was teleported to an unknown location. Either use your agent from Question 3 after it finishes exploring the grid (preferred), or modify the codebase to let your agent access the full grid information into the agent's memory (e.g., in case you didn't finish Question 3). Follow the steps in Table 1 to implement PF.

1. (10 points) Implement PF in deterministic case (without modifying `Grid.move()`). When the agent moves, all particles must be shifted towards the same direction.
2. (5 points) Implement stochastic movement case (as described in item ii. of the Table).

Good luck and email my (`cgunay`) or your TA Yonghui's (`yonghui.xiao`) Emory email if you run into trouble. Post your clarification questions on Piazza so everybody can benefit from the answers. Thanks and good luck!
-Cengiz