Project 3: Reinforcement Learning

CS 4300: Artificial Intelligence
University of Utah

Project Material Courtesy: CS188 Berkeley course projects `http://ai.berkeley.edu` In this project, you will design agents for the classic version of Pacman, including ghosts. Along the way, you will implement both minimax and expectimax search and try your hand at evaluation function design.

As in previous projects, this project includes an autograder for you to grade your answers on your machine. This can be run with the command:

python autograder.py

The code for this project consists of several Python files, some of which you will need to read and understand in order to complete the assignment, and some of which you can ignore. Download search.zip from here `http://ai.berkeley.edu/reinforcement.html` which will contain all the code and supporting files.

1 Files to edit

For this project you will edit the following files:

- **valueIterationAgents.py:** A value iteration agent for solving known MDPs.
- **qlearningAgents.py:** Q-learning agents for Gridworld, Crawler and Pacman.
- **analysis.py:** A file to put your answers to questions given in the project.

2 Supporting Files

The following python files would help you in understanding the problem and the get you familiar with the different data structures and games states in Pacman.

- **mdp.py:** Defines methods on general MDPs.
- **learningAgents.py:** Defines the base classes ValueEstimationAgent and QLearningAgent, which your agents will extend.
- **util.py:** Utilities, including util.Counter, which is particularly useful for Q-learners.
- **gridworld.py:** The Gridworld implementation.
- **featureExtractors.py:** Classes for extracting features on (state,action) pairs. Used for the approximate Q-learning agent (in qlearningAgents.py).

3 Reinforcement Learning(100 pts)

For all the problem titles described below, please refer to the link `http://ai.berkeley.edu/reinforcement.html` for the problem description and what is expected of each problem. As always the autograder has different test cases against which you can run your program to check the correctness. Please ensure your code is readable and use comments in your code to make it more clear for the person reading your code.

3.1 Value Iteration (24 pts)

- Code implementation

3.2 Bridge Crossing Analysis (4 pts)

- Code implementation

3.3 Policies (20 pts)

- Code implementation

3.4 Q-Learning (20 pts)

- Code implementation

3.5 Epsilon Greedy (12 pts)

- Code implementation

3.6 Bridge Crossing Revisited (4 pts)

- Code implementation

3.7 Q-Learning and Pacman (4 pts)

- Code implementation

3.8 Approximate Q-Learning (12 pts)

- Code implementation

4 Self Analysis (5 pts)

1. What was the hardest part of the assignment for you?
2. What was the easiest part of the assignment for you?
3. What problem(s) helped further your understanding of the course material?

4. Did you feel any problems were tedious and not helpful to your understanding of the material?
5. What other feedback do you have about this homework?

5 Evaluation

Your code will be autograded for technical correctness. Please do not change the names of any provided functions or classes within the code, or you will wreak havoc on the autograder. If your code passes all the test cases in the autograder you would receive full points for the implementation.

However even if your code does not necessarily pass all the test cases, we would evaluate your code and then award you partial points accordingly. In such cases it would be even more beneficial if you could give a short description of what you tried and where you had failed and that would help us in giving you better points.

6 Submission Instructions

- For the final submission you would be turning in a zipped folder of the python files and a PDF document containing your responses to questions from previous sections.
- Please ensure all the submissions are done through canvas. Please do not email the instructor or the TA's with your submission. Submissions made via email would not be considered for grading.
- **Naming:** Your upload should be named in the format ⟨uid⟩-Proj⟨number⟩.zip where ⟨uid⟩ is your Utah uid and ⟨number⟩ is the Project number. **Ex:** u0006300-Proj0.pdf
- For this project fill in portions of the files to edit. Once you have completed the code, zip your entire project folder, rename it as per the conventions stated above and submit it via canvas. Do not delete the other files present in the .zip file or change the names of any of those files in the project directory.
- **Written Answers:** Place all your written answers and responses to questions in "Self Analysis" in a single PDF document. This should be clearly named in the format ⟨uid⟩-Proj⟨number⟩-answers.pdf, where ⟨uid⟩ is your Utah uid and ⟨number⟩ is the Project number. Ex: u0006300-Proj0-answers.pdf Please make sure to write your name at the top of the document!
- **Group Submissions:** If you haven't done this already for previous projects and if you are working in groups, You need to sign up to one of the project groups in the people page in canvas and under the groups tab. Please sign up to one of the "project groups" and ensure that your group member is signed up to the same group. This ensures that submissions from one of the group members counts for

the entire group. Follow the file naming convention for the group member uploading the submission. i.e. do not mention the uid of your partner on the file name but mention your partner's details inside your answers.pdf .

Note: Each group is limited to maximum 2 members, so if a group is taken already please choose the next free group.