Pacman Challenge

Artificial Intelligence
Fall 2017
Due: November 26th, 11:59pm

1 About this Project

1.1 The Challenge

In this project, you compete against another pacman agent. Your goal is to eat the other pacman before it eats you. Each agent has two possible states: normal and angry, depicted by a yellow and red pacman. Whenever both agents are in the same state, they can freely cross each other. If one is angry and the other is in its normal state, the angry agent will eat the normal agent and win the game.
To reach an angry state, you need to eat a capsule that allows you to maintain the angry state for some amount of time.

1.2 Your Task

In this project, you need to implement a pacman agent in *CompetitiveAgents.py*. In the current state, this file contains a simple agent that is controllable with your keyboard. Unless explicitly stated, you are not allowed to make changes to the other files related to the project. The most important function of your agent is *getAction(...)*, that provides you with the agents current perception of the world and requires you to return its next action. It will be called by the game to get your next action and needs to return one of the five actions (Stop, North, East, South, West).
Besides the mentioned restrictions, you can implement what ever you have learned in this years lecture (and beyond) to be victorious in this challenge.
You can test your agent on the two maps *competitive1* (many loops) and *competitive2* (many dead ends), or create your own map. You can set the map with the *-l* parameter when running pacman.py. Feel free to use numpy, scipy or tensorflow if you think your agent benefits from it.

1.3 Running the Code

Since the challenge requires two agents, you need to start two instances of pacman.py. Each pair needs one server and one client. After a game is resolved, both instances will shut down. Here is how to run the code:

- Server: python pacman.py -s
- Client: python pacman.py –remote_ip 127.0.0.1

The remote_ip is the IP of the computer on which the server is running. We encourage you to talk to other groups to test your agents. To do this, just set the IP to the address of the computer that runs the server.

1.4 The rules of the World

- Capsules are a never ending recourse that can be found at certain locations on the map. When ever you eat a capsule, another capsule will appear somewhere on the map.
- You will always know the layout of the map, but you will only have current information of the variable items (the other agent and capsules) within a radius of 12 fields around your agent. For debugging

purpose, we are always rendering the full state of the world on your screen, but the state that is offered to your agent is limited, as described above (you can simply print(observations) to see an ascii representation in your terminal).

- The agents will take a single action in an alternating manner. The servers agent will take the first action.
- We will observe how long your agent takes to start and to make a turn. Your agent should decide for its first action in under 30 seconds and all subsequent actions in under 3 seconds.
- Your agent needs to be some kind of rational agent to be allowed in the challenge (entirely random agents are not allowed).
- An agent is considered aggressive, whenever the variable 'angryTimer' in its state is greater than 0.

1.5 Grading

This project is entirely voluntary and it will not reflect on your grade if you decide not to work on this project. Please submit your *CompetitiveAgents.py* file.
We will hold a tournament in the last class of the Fall term to determine the top 10 agents. The tournament will be held on a map that is not given to you. Everyone who submitted some form of rational agent will at least get 2 bonus points for the midterm.
The remaining prices are as follows. Each team member will get the mentioned bonus points:

- 1st: *Some* drone (Only one, distribution up to group) and 12 (midterm) bonus points
- 2nd: *Barnes and Noble* voucher (amount to be decided) and 11 (midterm) bonus points
- 3rd: *Starbucks* voucher (amount to be decided) and 10 (midterm) bonus points
- 4th: 9 (midterm) bonus points
- 5th: 8 (midterm) bonus points
- 6th: 7 (midterm) bonus points
- 7th: 6 (midterm) bonus points
- 8th: 5 (midterm) bonus points
- 9th: 4 (midterm) bonus points
- 10th: 3 (midterm) bonus points
- remaining students: 2 (midterm) bonus points

2 Further Aspects

2.1 Academic Integrity

We will be checking your code against other submissions in the class for logical redundancy. If you copy someones code and submit it with minor changes, we will know. These cheat detectors are quite hard to fool, so please don't try. We trust you all to submit your own work only. If you do, we will pursue the strongest consequences available to us.

2.2 Getting Help

If you find yourself stuck on something, contact the course staff for help. Office hours, classes, and the discussion forum are there for your support. We want these projects to be rewarding and instructional, not frustrating and demoralizing. But, we don't know when or how to help unless you ask.