Homework 3[1]

Due by 11:59pm 11/10/2016

Objective

This assignment is about building a reinforcement learning agent. In particular, you will have to build a Flappy Bird agent, who navigates in an environment full of pipes. Its only action choices are: flap wings, or do nothing. Its objective is to survive for as long as possible. The bird moves at a constant horizontal speed, but its vertical velocity is controlled by its flapping (or lack thereof).

Your bird needs to learn to perform the right actions -- when to flap, when not to -- using reinforcement learning (in particular, Q-Learning). The training process may take many hours. Here is a short video of someone's flappy bird training process, and here is another example, which includes a little more explanation (NB: I don't know what underlying support code either one of these efforts used. So you shouldn't necessarily expect to have the same level of performance as those videos for your homework effort).

Preliminaries

Before we get to the main assignment, there are some preparation work you'll need to do. I recommend getting started on this ASAP because in previous years, people have complained about difficulties getting set up. Since the assignment itself will take about two weeks, you are encouraged to get started on the preliminaries immediately, even though we have not fully covered Q-Learning yet.

- Install the appropriate pygame module for your Python interpreter.
    - NB: I've encountered some problems while setting up for El Capitain (Mac) -- last year, it worked fine for Maverick, and for Windows. Here is a webpage that I found helpful: http://kidscancode.org/blog/2015/09/pygame_install/ (They also have a nice tutorial on pygame in general.)
        - When the program calls pygame.init(), I get a complaint about some deprecated audio module. It doesn't matter. The program also takes forever to load for me, but it will eventually get started.
- Read through the Flappy Bird code by TimoWilken (for your convenience, I've cloned it to your individual repo) to make sure you understand what it's doing. In particular, you need to know:
    - where and how user input is handled
    - how to get the relevant information to compute your bird's current state.

[1]Shared Google Directory

What You Need to Do

- Modify the Flappy Bird code so that it runs multiple trials one after the other instead of quitting when the game ends. Make sure that you save all the relevant information between trials.
- Decide how you want to set up the state space, the actions, and the rewards. In class, we talked about using the vertical and horizontal differences between the bird's location and the nearest pipe in terms of pixels. We also talked about considering a state transition for every frame (every time unit in which the bird location is updated in the code). There may be other better representation choices.
- Implement the Q-learner plus exploration function, and hook it into the rest of the game.
- Introduce a mechanism to save the learned values and re-load them in later runs.
- Run your program until you have a "smart" bird.

What to commit in addition to source code

- A short write up; it should discuss:
    - The choice you've made to get the learning framework going. Have you tried different state representations? Have you tried different exploration approaches? Have you tried different learning rates? How do they impact the learning process?
    - Overall, how long did it take you to train the bird? What do you think you'd need to do to make the training more efficient? (Asked in another way, If you had more time to work on this project, what would you do next?)
    - A brief description of how to load your trained bird into the program. (The grader won't be able to wait for your bird to train from scratch.)
    - if you used any additional tools or resources, give proper citations for them. If you've discussed the problem with other people, let us know the extent of your collaboration.

Grading

1. A serious attempt at modifying the flappy bird code -- for example, can save out and read in values, and have a random action generator for the bird.
2. Basic infrastructure set, but something is wrong with the reinforcement learning implementation. (e.g., bird does not seem to learn, but the grader can recognize the basic structure of q-learning in code.)
3. Demonstrate that the bird has learned something: for example, show that the trained bird regularly survives multiple pipes (>5, say) of some fixed height.
4. Demonstrate that the bird regularly survives multiple pipes (>5, say) for a fixed set of randomized pipes (use random.seed(0) to fix the random number generator sequence)
5. Demonstrate that the trained bird regularly survives many pipes (>10, say) in the original problem setting; good representation choices. Insightful writeup.