Homework 2—ECE590–001—**Due: 12 Feb 2020**

2/29/2020

1. Implement gridworld using OpenAI gym specification. You will implement a sub-class of `Env` called `gridworld`.

   (a) Your class should allow for arbitrary rectangular grids be able to specify arbitrary reward per user specification. You will also need a way to pass a done condition.

   (b) Your class should be initialized as follows:

   ```
   my_grid = gridworld(reward=my_reward) # where my_reward is a numpy.array of
       shape (n,m)
   ```

   (c) Your class needs to meet the specifications set down in docstring for `Env`: https://github.com/openai/gym/blob/master/gym/core.py. Implement the API methods; all except render are required for this assignment. There will be a bonus for render.

   (d) Check that your class is working and meets specification by running the `gym` *random agent*: https://github.com/openai/gym/blob/master/examples/agents/random_agent.py.

   (e) Demonstrate class can encode the gridworld Example 3.5 on page 60 of Sutton and Barto.

2. Sample the dynamics of your gridworld under a random policy and estimate state-value function like in Example 3.8 page 65 of Sutton and Barto.

   (a) Make a function from the rollout/trajectory code https://github.com/openai/gym/blob/master/examples/agents/random_agent.py.

   (b) Call this function in such a way that you can estimate the state-value function for the random policy and the gridworld defined originally in Example 3.5

3. Train a policy for the Example 3.5 gridworld using the Vanilla Policy Gradient (VPG) code provided in spinning up https://spinningup.openai.com/en/latest/algorithms/vpg.html.

   (a) Set a trajectory length so that 20 states are visited in each trajectory. The OpenAI gym framework this is essentially when you require `step` to return the `done` flag.

   (b) Read and execute the Vanilla policy gradient code on your `gridworld` environment: https://spinningup.openai.com/en/latest/algorithms/vpg.html. Note that this code despite being called *vanilla* already includes *Generalized Advantage Estimation*. The more basic code is here https://github.com/openai/spinningup/blob/master/spinup/examples/pg_math/1_simple_pg.py

   (c) Supply a training artifact (a plot of either timesteps vs. total reward, trajectories sampled vs. total reward) showing that VPG converges to some score. In words, write down an explanation for this score in terms of the trajectory length and reward specified in Example 3.5.

   (d) The VPG code estimates the state-value function. Plot these values (seaborn should do this nicely).