Reinforcement Learning

Homework #3

Sarsa

Description

For this assignment, you will build a Sarsa agent which will learn policies in the OpenAI Gym(http://gym.openai.com/docs/) Frozen Lake environment. OpenAI Gym (http://gym.openai.com/docs/) is a platform where users can test their RL algorithms on a selection of carefully crafted environments. As we will continue to use OpenAI Gym (http://gym.openai.com/docs/) through Project 2, this assignment also provides an opportunity to familiarize yourself with its interface.

Frozen Lake is a grid world environment that is highly stochastic, where the agent must cross a slippery frozen lake which has deadly holes to fall through. The agent begins in thestarting state `S` and is given a reward of `1` if it reaches the goal state `G`. A reward of `0` is given for all other transitions.

The agent can take one of four possible moves at each state (left, down, right, or up). Thefrozen cells `F` are slippery, so the agent's actions succeed only 1/3 of the time, while the other 2/3 are split evenly between the two directions orthogonal to the intended direction. If the agent lands in a hole `H`, then the episode terminates. You will be given a randomized Frozen Lake map with a corresponding set of parameters to train your Sarsa agent with.

Sarsa $(S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1})$

Sarsa uses temporal-difference learning to form a model-free on-policy reinforcement- learning algorithm that solves the *control* problem. It is model free because it does not needand does not use a model of the environment, namely neither a transition nor reward function; instead, Sarsa samples transitions and rewards online.

It is on-policy because it learns about the same policy that generates its behaviors (this is incontrast to *Q-learning*). That is, Sarsa estimates the action-value function of its behavior policy. In this homework, you will not be training a Sarsa agent to approximate the *optimal* action-value function; instead, the hyperparameters of both the exploration strategy and the algorithm will be given

to you as input — the goal being to verify that your SARSA agent is correctly implemented.

Procedure

Attention to detail to each of the following points is required:

- You must use Python and the library NumPy for this homework *python 3.6.x* and *numpy==1.18.0* or more recent version.
- Install OpenAI Gym (e.g.pip install gym) *gym==0.17.2*
- The Frozen Lake environment has been instantiated for you.
- The pertinent random number generators have been seeded for you. Do *not* use the Python standard library's *random* library.
- Implement your Sarsa agent using an ε-greedy behavioral policy. Specifically, you must use *numpy.random.random* to choose whether or not the action is greedy, and *numpy.random.randint* to select the random action.
- Initialize the agent's Q-table to zeros.
- Train your agent using the given input parameters. The input *amap* is the Frozen Lake map that you need to resize and provide to the *desc* attribute when you instantiate your environment. The input *gamma* is the discount rate. The input *alpha* is the learning rate. The input *epsilon* is the parameter for the ε-greedy behavior strategy your Sarsa agent will use. Specifically, an action should be selected uniformly at random if a random number drawn uniformly between 0 and 1 is less than ε. If the greedy action is selected,the action with lowest index should be selected in case of ties. The input `n_episodes` is the number of episodes to train your agent. Finally, *seed* is the number used to seed both Gym's random number generator and NumPy's random number generator.
- Your Sarsa implementation should select the action corresponding to the next state the agent will visit *even when* that next state is a terminal state.
- You should return the greedy policy with respect to the Q-function obtained by your Sarsa agent after the completion of the final episode. Specifically, the policy should beexpressed as a string of characters: **<, v, >, ^,** representing left, down, right, and up, respectively. The ordering of the actions in the output should reflect the ordering of states in *amap*.

Resources

The concepts explored in this homework are covered by:

- Lesson 4: Convergence
- Chapter 6 (6.4 Sarsa: On-policy TD Control) of http://incompleteideas.net/book/the-book-2nd.html

(http://incompleteideas.net/book/the-book-2nd.html)

Submission

- The due date is indicated on the Syllabus page for this assignment.
- Use the template code to implement your work.
- Please use *python 3.6.x* or more recent version, *gym==0.17.2* and *numpy==1.18.0* or more recent version, and you can use any core library (i.e., anything in the Python standard library). No other library can be used.