CS 747: Programming Assignment 3

In this assignment, you will implement an algorithm for estimating the value function of a policy for a given MDP from a trajectory of the form state, action, reward, state, action, reward, ….

Data Format

The input to your "evaluator" will be a text file that provides information in the following format.

```
Number of states
Number of actions
Discount factor
state1 action1 reward1
state2 action2 reward2
state3 action3 reward3
.
.
.
stateN actionN rewardN
stateN+1
```

The number of states `S` and the number of actions `A` will be integers greater than 0. Assume that the states are numbered 0, 1, ..., `S - 1`, and the actions numbered 0, 1, ..., `A - 1`. The discount factor will lie between 0 (included) and 1 (excluded). The trajectory over time will be long enough, and the dynamics of the underlying MDP such, that there is at least one outgoing transition from each state in the MDP. Note that the MDP is *not* episodic: that is, stateN+1 is not a terminal state (and can occur within the trajectory multiple times). The trajectory is merely a finite sequence generated according to the underlying transition and reward functions, and terminated at some arbitrary time step.

You can assume that `S` and `A` will not exceed 50, and N, the total number of transitions in the trajectory, will not exceed 500,000. In this data directory, you will find two sample data files (`d1.txt` and `d2.txt`).

Output

Given a data file, your evaluator must estimate the value function `V` under the policy being followed. The output, written to standard output, must be in the following format (`Est-V` is your estimate of `V`).

```
Est-V(0)
Est-V(1)
.
.
.
Est-V(S - 1)
```

In the data directory enclosed, you will find output files corresponding to the two data files, which have solutions in the format above. The values mentioned in these output files

are indeed the *true* values (under the same policy) from the MDP being sampled. Naturally, as you will have to estimate values based on samples alone, your estimates cannot be expected to match the true values perfectly.

Notice that since this is a prediction problem, wherein a fixed policy is being followed, the actual names of the actions taken do not matter. Nor does it matter if the policy being followed is deterministic or stochastic. Your logic only needs to consider the state, reward, and next state associated with each transition.

You are free to implement the evaluator in any programming language of your choice. Since your output will be checked automatically, make sure you have nothing printed to stdout other than the `S` lines as above in sequence. If the testing code is unable to parse your output, you will not receive any marks.

Submission

You must submit a directory titled `[rollno]` (such as `1234567`), which contains all your source and executable files. The directory must contain a script titled `evaluator.sh`, which must take in exactly one command line argument corresponding to a data file. For testing your code, the following command will be used from your `[rollno]` directory.

```
./evaluator.sh dataFileName
```

`dataFileName` will include the full path. Before you submit, make sure you can successfully run `evaluator.sh` on the departmental (`sl2`) machines.

Include a file called `notes.txt` in your `[rollno]` directory, that (1) describes the algorithm your evaluator implements, and (2) provides references to any libraries and code snippets you have utilised. It is okay to use libraries for data structures and for operations such as sorting. However, the logic used for value prediction must entirely be code that you have written.

In summary: you must submit your `[rollno]` directory, compressed as `[rollno].tar.gz` (say `1234567.tar.gz`) through Moodle. The directory must contain `evaluator.sh`, along with all the sources and executables, as well as a `notes.txt` file.

Evaluation

Your evaluator will be tested on trajectories generated from different MDPs and policies. Your task is to ensure that it prints out a good estimate of the true value function in each case. Performance will be quantified based on the (unweighted) squared distance between your estimate `Est-V` and the true value function `V`: that is,

`Error` $= \sum_{s \in S}$ `(V(s) - Est-V(s))`^{2}.

Recall that as a part of Programming Assignment 2, you had written code for MDP planning. It will be a good idea for you to build a testing framework using that code to (1) generate and record trajectories of some fixed policy π for some MDP M; (2) estimate the value function of π as required in this assignment; and (3) compare your estimate with the true value function, which you can compute using your own code from Programming Assignment 2. This is exactly the scheme that we will use for evaluating your answers.

8 marks are reserved for the performance of your evaluator on unseen trajectories, and 2 marks for your explanations in `notes.txt`. Be sure to describe your approach and explain why you chose it over alternative approaches.

The TAs and instructor may look at your source code and notes to corroborate the results obtained by your program, and may also call you to a face-to-face session to explain your code.

Deadline and Rules

Your submission is due by 11.55 p.m., Sunday, October 7. You are advised to finish working on your submission well in advance, keeping enough time to test it on the `sl2` machines and upload to Moodle. Your submission will not be evaluated (and will be given a score of zero) if it is not received by the deadline.

Before submission, make sure that your code runs for a variety of experimental conditions. Test your code on the `sl2` machines even while you are developing it: do not postpone this step to the last minute. If your code requires any special libraries to run, it is *your* responsibility to get those libraries working on the `sl2` machines (go through the CSE bug tracking system to make a request to the system administrators). Make sure that you upload the intended version of your code to Moodle (after uploading, download your submission and test it on the `sl2` machines to make sure it is the correct version). You will not be allowed to alter your code in any way after the submission deadline. In short: your grade will be completely determined by your submission on Moodle at the time of the deadline. Play safe by having it uploaded and tested at least a few hours in advance.

You must work alone on this assignment. Do not share any code (whether yours or code you have found on the Internet) with your classmates. Do not discuss the design of your solution with anybody else. Do not see anybody else's code or report.