CS 386: Lab Assignment 6

In this assignment, you will implement **Value Iteration**, which is an **MDP planning** algorithm. The input to the algorithm is an MDP. The expected output is the optimal value function, along with an optimal policy.

Data

This compressed directory contains three MDP instances, along with a correct solution for each. You can use these instances to test your code. Your code will also be evaluated on MDP instances other than those provided.

Each MDP is provided as a text file in the following format.

Number of states
Number of actions
Reward function
Transition function
Discount factor

The number of states S and the number of actions A will be integers greater than 0 and less than 100. Assume that the states are numbered $0, 1, \ldots, S-1$, and the actions are numbered $0, 1, \ldots, A-1$. The reward function will be provided as SAS entries. Each entry corresponds to $R(s, a, s')$, wherein state s, action a, and state s' are being iterated in sequence from 0 to $S-1$, 0 to $A-1$, and 0 to $S-1$, respectively. A similar scheme is adopted for the transition function T. Rewards can be positive, negative, or zero. The discount factor is a real number between 0 (included) and 1 (excluded).

To get familiar with the MDP file format, you can view and run `generateMDP.py` (also provided in the `data` directory), which is a python script used to generate random MDPs. Change the number of states and actions, the discount factor, and the random seed in this script in order to understand the encoding of MDPs.

Value Iteration

Value iteration is a simple, iterative algorithm, which is specified in the slides used in class (see Slide 13). Assume V^0, the initial value vector, has every element as zero. Interpret the norm provided in the termination condition as the max norm, and implement ``small enough" as ``being no larger than 10^{-16}". In other words, stop the algorithm when for every state s, $|V^t(s) - V^{t-1}(s)| <= 10^{-16}$. You can assume that at this point, V^t is the optimal value function. An optimal policy can be obtained by taking greedy actions with respect to the optimal value function.

Output

Given an MDP, your code must compute the optimal value function $V^\star$ and an optimal policy $\pi^\star$. Its output, written to standard output, must be in the following format.

$V^\star(0) \quad \pi^\star(0)$
$V^\star(1) \quad \pi^\star(1)$
.
.
.
$V^\star(S-1) \quad \pi^\star(S-1)$
Iterations t

The first S lines contain the optimal value function for each state and the action taken by an optimal policy for the corresponding state. The last line specifies the number of iterations taken by value iteration (the value of variable t in the pseudocode upon termination). In the `data` directory enclosed, you will find three solution files corresponding to the MDP files, which have solutions in the format above. The optimal value function and policy in each case is correct, but the number of iterations is merely provided as a placeholder (do not use it as a guide!).

Since your output will be checked automatically, make sure print nothing to stdout other than the $S+1$ lines as above in sequence.

Task 1 (7 marks)

Implement value iteration in any programming language of your choice (provided it can be tested on the SL2 machines), taking in an MDP file as input, and producing output in the format specified above. The first step in your code must be to read the MDP into memory; the next step to perform value iteration; and the third step to print the optimal value function, optimal policy, and number of iterations to stdout. Make sure your code produces output that matches what has been provided for the three test instances.

Create a shell script called `valueiteration.sh`, which will be called using the command below.

```
./valueiteration.sh mdpFileName
```

Here `mdpFileName` will include the full path to the MDP file; it will be the only command line argument passed. (If, say, you have implemented your algorithm in C++, the shell script must compile the C++ file and run the corresponding binary with the MDP it is passed.) The shell script must write the correct output to stdout. It is okay to use libraries and high-level templates for data structures and for operations such as finding the maximum. However, the logic used in value iteration must entirely be code that you have written.

Task 2 (3 marks)

In this task, you will examine the effect of the discount factor γ on the running time of value iteration. Run value iteration on six MDP instances: in each, S, A, R, and T are the same as those in `mdp-03.txt`; the discount factors of the MDPs are 0.9, 0.99, 0.999, 0.9999, 0.99999, and 0.999999. Recall that the last line in the MDP file contains the discount factor. It will be easiest for you to use `mdp-03.txt` in all your experiments, editing the file appropriately for each setting of the discount factor.

Note down the number of iterations taken by the algorithm in each case, and also notice the optimal value functions and policies computed.

Open a file called `gamma.txt` and fill it up with the following details.

1. Provide a table with γ and the number of iterations as columns, and one row for each of the six γ values you ran on `mdp-03.txt`.
2. Describe how the number of iterations varies with γ. Can you explain the dependence?
3. Do the optimal value function and policy vary with γ, or do they stay constant? Why?

Submission

You must submit all the files needed to run your agent (source and binary), including `valueiteration.sh`. You must also submit `gamma.txt`, containing answers to questions in Task 2. If you have used any special libraries, mention them in a text file called `libraries.txt`. Place all these files in a single directory named `la6-rollno` (substitute `rollno` with your roll number). Tar and gzip this directory; submit `la6-rollno.tar.gz`. on Moodle, under Lab Assignment 6.