CS 434: Assignment 5

Due June 5th 11:59PM, 2017

General instructions.

1. The following languages are acceptable: Java, C/C++, Matlab, Python and R.
2. You can work in team of up to 3 people. Each team will only need to submit one copy of the source code and report.
3. You need to submit your source code (self contained, well documented and with clear instruction for how to run) and a report via TEACH. In your submission, please clearly indicate your team members' information.
4. Be sure to answer all the questions in your report. Your report should be typed, submitted in the pdf format. You will be graded based on both your code as well as the report. In particular, the clarity and quality of the report will be worth 10 % of the pts. So please write your report in clear and concise manner. Clearly label your figures, legends, and tables.
5. Note, this is an optional bonus assignment and will account for 6 additional bonus percent for the final grade!

Build a Planner.

For this part of the implementation assignment you need to implement an MDP planning algorithm for optimizing expected infinite-horizon discounted cumulative reward. In particular, you should:

1. implement value iteration algorithm. The input to your algorithm should be a text file that describes an MDP (see below) and a discounted factor β. The output should consist of two vectors U - optimal utility function, P-policy for the MDP, and β- discounted factor you used.
2. test your algorithm on the provided data with two different $\beta = 0.1$ and $\beta = 0.9$. In your report, for each β, please, provide:
   (a) a discounted factor β;
   (b) (30 pts) $n \times 1$ vector U - optimal utility function;
   (c) (20 pts) $n \times 1$ vector P - policy for the MDP.

The input format for the MDP should be a text file with the following format:

- First line gives two integers n and m specifying the number of states and actions respectively.
- After the first line there will be a blank line and then a sequence of m $n \times n$ matrices (each separated by a blank line) that give the transition function for each action. Specifically, the i'th matrix gives the transition function for the i'th action. Entry (j, k) in that matrix (where j is the row and k is the column) gives the probability of a transition from state j to state k given action i. The rows, thus, will sum to 1.
- After the final transition matrix there will be a blank line followed by row of n real numbers. The i'th real number specifies the reward for the i'th state.

An example of the format is below:

```
3  2

0.2  0.8  0.0
0.0  0.2  0.8
1.0  0.0  0.0

0.9  0.05  0.05
0.05  0.9  0.05
0.05  0.05  0.9

-1.0  -1.0  0.0
```

The MDP has 3 states and 2 actions. We see that in state 1 if we take action 1 then there is 0.2 probability of remaining in state 1 and 0.8 probability of a transition to state 2 (zero probability of going to state 3). The reward is negative, except for when the system is in state 3. So the goal should be to get to state 3 and stay there as much as possible. This will generally involve taking action 1 if not in state 3 and then taking action 2 when in state 3. (See if you can understand why that is the best policy and test that your algorithm can figure that out.)

Remark. *You may ask how many iterations you need to run value iteration for? In class, we have talked about checking the difference between the utility value from one iteration to the next $\|U^k - U^{k-1}\|$. If max difference is smaller than a threshold δ, stop, i.e., stop if $\|U^k - U^{k-1}\| < \delta$. However, this approach may result in suboptimal policy. To make sure that the utility value of the resulting policy is ϵ-close to the utility value of the optimal policy, i.e., $\|U_{res} - U_{opt}\| < \epsilon$,* *please, use the following formula for δ:*

$$\delta = \frac{\epsilon(1-\beta)^2}{2\beta^2},$$

where β is the discounting factor and ϵ is the parameter that controls how close is the policy you get to the optimal policy. Please, set ϵ small enough, e.g., $\epsilon = 1e^{-10}$.