Assignment A7: Value Iteration

CS 4300
Fall 2017

Assigned: 17 October 2017

Due: 21 November 2017

For this problem, handin a lab report pdf (include name, date, assignment and class number in pdf) which examines the *value iteration* algorithm. The agent is to learn a policy for the following Wumpus world:

```
0 0 0 G
0 0 P 0
0 0 W 0
0 0 P 0
```

For this assignment, assume the actions available to the agent are

$$A = \{UP, LEFT, DOWN, RIGHT\}$$

where these are movements with probabilistic outcomes as described in the text (i.e., 0.8 probability of going the direction selected, 0.1 of going to either side). This requires development of a transition probability table for the 16 cells for the 4 actions. You are to implement the *value iteration* algorithm given on p. 653 of the text and show:

- comparable results to those given in Fig. 17.2 (a) and (b), p. 648 of R&N for the 16 utilities learned. In part (b) show results for several values of R so that some lead to the *gold* and some to *death*.
- what values you obtain comparable to those in Fig. 17.3, p. 651 in R&N.
- your results for the plot shown in Fig. 17.5 (a), p. 653 in R&N. Show this plot for γ values of 0.9, 0.99, 0.999, 0.9999, 0.99999 and 0.999999.

Note that you must implement a policy generator function separately from the utility calcuation. Describe the algorithms in the method section, and verify their correctness. You should handin the report pdf as well as the source code used in the study. The code should conform to the style requested in the class materials (no matter what the language). In addition, please turn in a hardcopy of the report in class before the start of class on November 21, 2017.

Write a lab report in the format (please do not deviate from this format!) described in the course materials.

Here are the function descriptions of what you are to create:

```
function [U,U_trace] = CS4300_MDP_value_iteration(S,A,P,R,gamma,...
    eta,max_iter)
% CS4300_MDP_value_iteration - compute policy using value iteration
% On input:
%     S (vector): states (1 to n)
%     A (vector): actions (1 to k)
%     P (nxk struct array): transition model
%       (s,a).probs (a vector with n transition probabilities
%       (from s to s_prime, given action a)
%     R (vector): state rewards
%     gamma (float): discount factor
%     eta (float): termination threshold
%     max_iter (int): max number of iterations
% On output:
%     U (vector): state utilities
%     U_trace (iterxn): trace of utility values during iteration
% Call:
%     [U,Ut] = Cs4300_MDP_value_iteration(S,A,P,R,0.999999,0.1,100);
%
%     Set up a driver function, CS_4300_run_value_iteration (see
%     below), which sets up the Markov Decision Problem and calls this
%     function.
%
%     Chapter 17 Russell and Norvig (Table p. 651)
%     [S,A,R,P,U,Ut] = CS4300_run_value_iteration(0.999999,1000)
%
%     U' =  0.7053 0.6553 0.6114 0.3879 0.7616 0 0.6600 -1.0000
%       0.8116 0.8678 0.9178 1.0000
```

```
%
 %    Layout:                1
%                            ^
%      9 10 11 12            |
%      5  6  7  8       2 <- ->  4
%      1  2  3  4            |
%                            V
%                            3
% Author:
%    <Your name>
%     UU
%     Fall 2017
%


function policy = CS4300_MDP_policy(S,A,P,U)
% CS4300_MDP_policy - generate a policy from utilities
% See p. 648 Russell & Norvig
% On input:
%     S (vector): states (1 to n)
%     A (vector): actions (1 to k)
%     P (nxk struct array): transition model
%       (s,a).probs (a vector with n transition probabilities
%       from s to s_prime, given action a)
%     U (vector): state utilities
% On output:
%     policy (vector): actions per state
% Call:
%     p = CS4300_MDP_policy(S,A,P,U);
% Author:
%     <Your name>
%     UU
%     Fall 2017
%
```