Monish Gupta (Evens) and Eric Waugh (Odds)

CS 4300

November 20 2017

Assignment 7 Lab Report

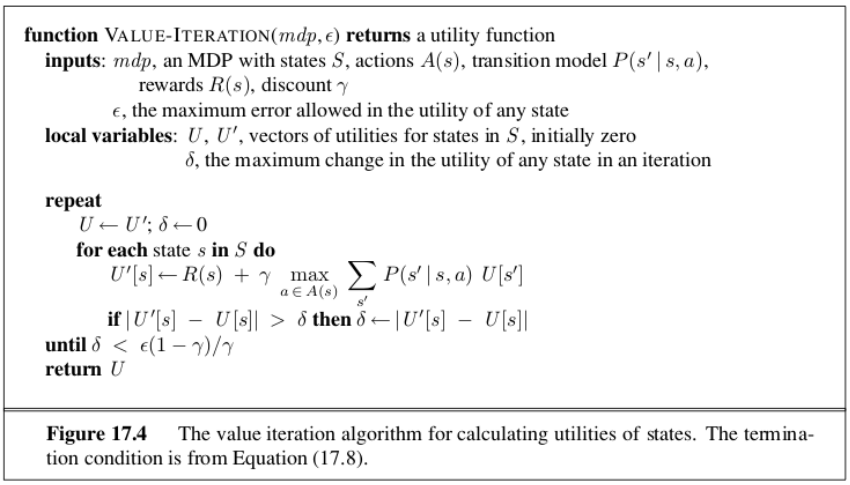
1. *Introduction*

The Markov Decision Process value iteration algorithm performs value propagation upon cells using a transition model that defines stochastic, biased movement with respect to a chosen action. The model uses a number of states by number of action matrix struct which articulates the probabilities for moving to another state starting at a given state initiating the given action. The value iteration algorithm elicits contraction, a phenomenon in which the values of each state come closer to a fixed point known as the equilibrium. The value iteration algorithm terminates when a specified range within the equilibrium point is met, or an independent maximum iterations amount is met. From this stems a few questions about MDP value iteration.

Question : How does the values of the states vary with the amount of iteration run?

2. *Method*

For the value iteration function we followed the algorithm from the book.



function VALUE-ITERATION(mdp, ϵ) **returns** a utility function
inputs: mdp, an MDP with states S, actions $A(s)$, transition model $P(s' \mid s, a)$,
rewards $R(s)$, discount γ
ϵ, the maximum error allowed in the utility of any state
local variables: U, U', vectors of utilities for states in S, initially zero
δ, the maximum change in the utility of any state in an iteration

repeat
$U \leftarrow U'$; $\delta \leftarrow 0$
for each state s **in** S **do**
$U'[s] \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' \mid s, a)\ U[s']$
if $|U'[s] - U[s]| > \delta$ **then** $\delta \leftarrow |U'[s] - U[s]|$
until $\delta < \epsilon(1-\gamma)/\gamma$
return U

Figure 17.4 The value iteration algorithm for calculating utilities of states. The termination condition is from Equation (17.8).

The algorithm provided from the book

Our code takes in the states and actions and the transition model that we create for each specific board we are testing. We then have our while loop that adds up the sums from the transition model we are given at each state and then picks the max of these. It does this until that value is less than the value of epsilon(1 - gamma)/gamma. It will also end if we reach the max iterations.

```
count = 0;
max_utility_change = -1;
n = 12;
U = zeros(1,n);
newU = R;

U_trace = [];

n = size(S,2);
k = size(A,2);

while count < max_iter

    max_utility_change = 0;

%     newU(3) = -1000;
%     newU(6) = -1000;
%     newU(9) = -1000;
%     newU(16) = 1000;

    newU(12) = 1;
    newU(8) = -1;
    U = newU;

    %for each state s in S do
    for s = 1:n
        bestUtil = -Inf;

        for a = 1:k
          currentUtil = 0;
          possible_values = find(P(s,a).probs);
          for i = 1:size(possible_values,2)
            currentUtil = currentUtil...,
              + (P(s,a).probs(possible_values(i)) * U(possible_values(i)));
          end
          bestUtil = max(bestUtil, currentUtil);
        end

        newU(s) = R(s) + (gamma * bestUtil);

        max_utility_change = max(max_utility_change, abs(newU(s) - U(s)));
    end

    U_trace = [U_trace;U];
    count = count + 1;


    if max_utility_change < (eta * (1 - gamma))/gamma
       break;
    end
end

end
```

Our algorithm option for either board values

3. *Verification*

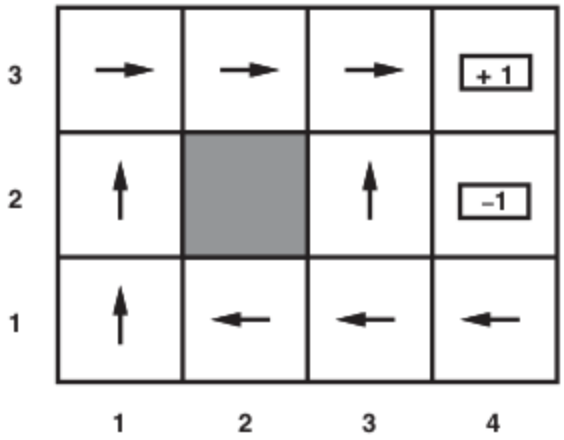
To verify our policies and value iteration we used the 3 by 4 board that is mentioned in the book and in our header examples to get the iteration data and policies we should be taking. We wrote another run iterator to set up the transition model and other various things.

3	0.812	0.868	0.918	+1
2	0.762		0.660	−1
1	0.705	0.655	0.611	0.388
	1	2	3	4

The values from the book

.8115	.8678	.9178	+1
.7615	0	.6603	-1
.7053	.6552	.6113	.3879

Our values

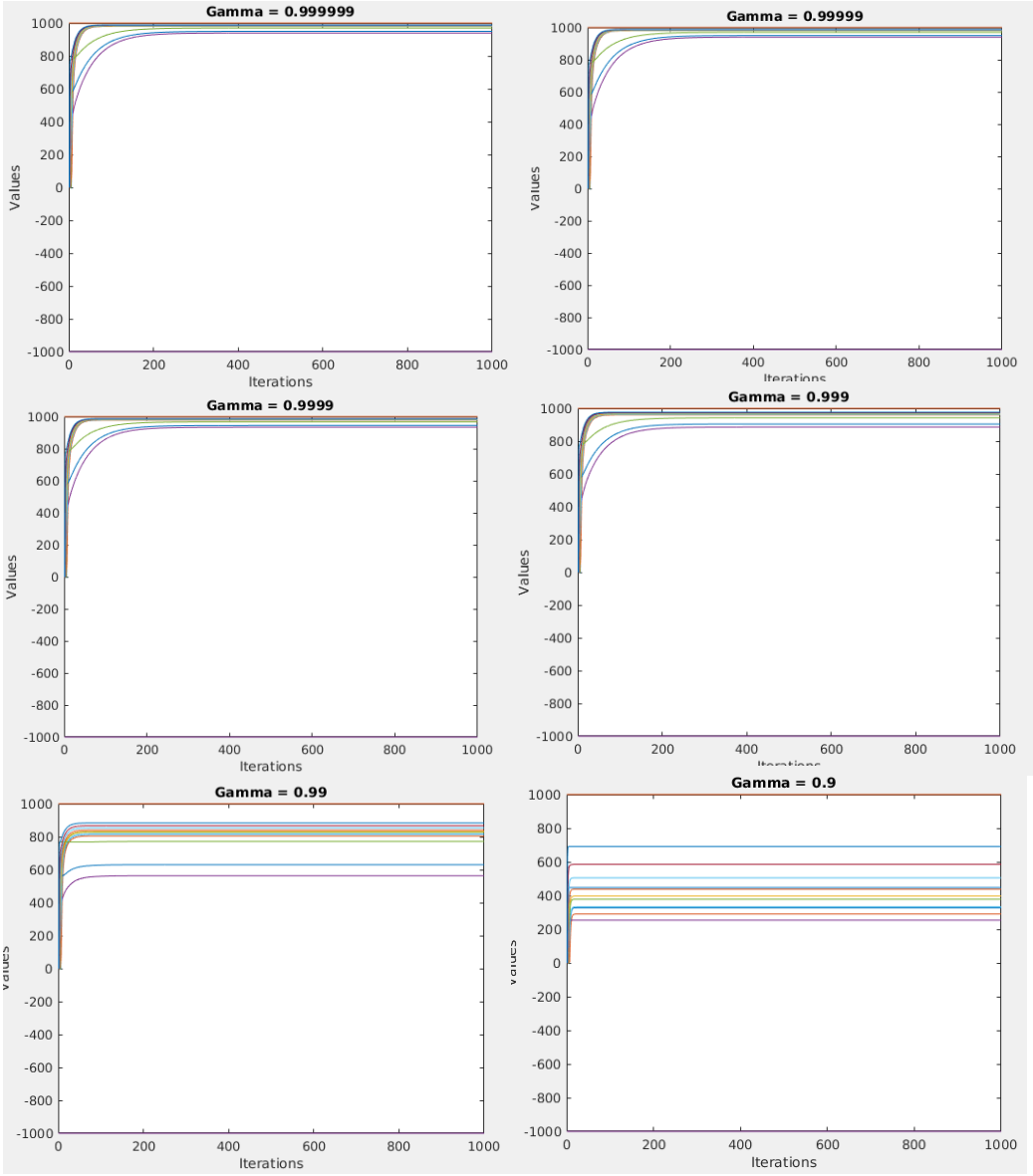


The actions of the 4 by 3 board

RIGHT	RIGHT	RIGHT	+1
UP		UP	-1
UP	LEFT	LEFT	LEFT

The actions that we got

4. *Data and Analysis*



5. *Interpretation*

We can see that as the number of iterations increases we get closes to our goal on knowing the best values on the board where they reach equilibrium and are no longer changing. This is good to see since we know this can give us the values we are looking for and know when to stop iterating with results that will allow us to know where the safest path on the board is. We can see that in the early iterations our values are all becoming more negative due it not knowing

paths from some of the further spots and the reward function giving us negative values but as soon as we learn more info about what is close to the good states we then update these values with how optimal they are to get to said good state. It is more or less starting from that good state and spreading out across the board until it has reached all the states where it then optimizes their paths.

6. *Critique*

   I feel like we learned that how to create a path using transition values over time by by taking in the positive and negative spaces and filing in the other spots based on the how close they will bring us to those spots already listed. I felt like this assignment could be improved if we didn't have to hard code the boards since that was very confusing at first on how we could get it to work with different boards where we later realized it was all hard coded in.

7. *Log:*

   Eric:

   Assignment: 7 hours

   Lab: 3 hours

   Monish:

   Assignment: 9 hours

   Lab: 3 hours