Foundations of Intelligent and Learning Agents
CS747

Mohit
20D070052

October 2023

1 Student Details

Name: Mohit
Roll No: 20D070052

2 Task 1

2.1 Value Iteration Algorithm

For this algorithm we need to iteratively compare the old and new values after applying the bellman optimality operator, and we stop when the difference between successive values after bellman optimality operator is very small (Here I have taken the threshold to be 1e-10). Therefore for this I have written a function of bellman optimality operator which takes as input the previous value and gives the new value and the optimal actions that are taken to get that value.

In this bellman function, we initialize 3 variables as optimal value, optimal actions which we will output and an optimal action for each state to update the optimal actions for each state. After this we iterate over all states, all actions and all states s'. For a given s and a, we iterate over all final states and compute this value, if this value is greater than the previous action then update the action for the state s as a and the value as this new computed value otherwise remain as it is. Now we do this for all states s and thus obtain the new values and new actions for each state.

Now after we get the new values and new actions from the bellman function, we compare each value of these new values to the previous values and if there is a single entry which doesn't satisfy we again calculate the bellman. Once all values are less than the threshold we conclude that the computed value is the final optimal value and the resulting actions are the optimal policy.

2.2 Howard Policy Iteration Algorithm

For this algorithm I first initialize a policy where all the actions are initialized to 0. After this according to the algorithm taught we calculate the value function for this policy and active value function for the value function thus obtained.

Therefore to compute value function I have made a function which takes the input as the policy array and returns the value function as the output for all states. For this I have initialized 2 arrays like before which store the previous value function and the new value function and we compute until the difference between old values and new values for all states is nearly zero (threshold I have used is 1e-12). When the difference equation is satisfied we break from the loop and return the obtained value function.

To compute the active value function I have made a function which takes the input as the value function for each state and returns the active value function for a given state and action. For this I have initialized a 2D array which I have updated for each corresponding state and it's action. This just requires 3 loops to iterate over s, a and s' and since we know all the parameters required for computation we just keep storing the values in the 2D array and return it.

After getting the value and active value function we store the difference between the active value function and it's corresponding value function for each state, if for any state if this difference is greater than a threshold (I have taken the threshold as 1e-10) then we update the policy with a new policy in which we update the actions for those states whose difference was more than the threshold cause these are the improvable states with improvable actions. And then again calculate the value function and active value function for this updated policy until there is no state for which the difference is more than the threshold after which return the value function obtained for that and policy which it followed as the optimal value and optimal policy.

2.3 Linear Programming Algorithm

For this algorithm I first initialize the LP problem that we need to solve after that I created the objective function which we need to solve. After this we start forming the constraint equations for the LP problem. After forming the constraint equations I passed it into the LP solver to get the required solution but we want to output the value function and the optimal policy so for that I calculated the required optimal value and optimal policy as output.

2.4 Observations

- In general I observed that the Linear Programming Algorithm was the fastest among all 3 whereas Value iteration was the slowest.
- For the episodic tasks the time taken was more than the time taken for non-episodic tasks.