CS 747

Programming Assignment 2

Rushikesh J. Metkar **19D070034**

CONTENTS :

TASK 1

I've implemented three algorithms :
Value Iteration, Linear Programming, and Howard's Policy Iteration

Value Iteration

This algorithm initiates two vectors representing the value function, and based on iterating one with respect to the other, it terminates computation when the difference of the vectors is less than a tolerance value. The optimal policy is also evaluated along with the computation of the value function.

Linear Programming

This algorithm directly implements linear programming using the PuLP module wherein the decision variables being the value function terms and the constraints on the decision variables are provided to the PULP_CBC_CMD solver. The optimal policy is evaluated from the optimal value function by comparison with the action value function. Due to the inaccuracy of the solution obtained from the solver, the value function is reevaluated using a generic value evaluation technique given the optimal policy

Howard's Policy Iteration

The policy is initialized as a zero vector, and as long as possible (based on a flag), the policy is optimized by modifying the policy for a state to the action with highest margin in action value function when compared to the state value function.

TASK 2

I have formulated the MDP for the Anti-Tic-Tac-Toe problem as follows :

Encoder.py :
A content_states array to store all the states of the agent
A content_policy array to store the policy of the environment
Number of states and actions is known
I've defined the end states as following

1. Agent Wins : '0'
2. Draw : '1'
3. Environment Wins : '2'

From a given state the agent takes an action to goto an intermediate_state.
At the intermediate_state I check if the Agent wins or the game is Draw. If it is neither then the environment plays according to its policy and I get the next_state.
At this state I check if the Environment wins or the game is Draw.
In this way I've written all the transitions with the reward being '1' if Environment wins and '0' otherwise.

Decoder.py :
A states array to store all the states of the agent
Player_id is the id of the agent
Starting from the first state, the policy computed at planner.py is followed to generate the route until the present state reaches the end state.

TASK 3

The sequence of policies generated for each player is guaranteed to converge.
As we go on providing the policies of the environment to the agent, the players learn to take the optical decision and the accuracy goes on increasing.
Each player generates the best optimal policy after a certain period.
Thus, we can say that for each player the sequence of policies converges definitely.