CS 747: Programming Assignment 2

Aman Singh : 190020010

Task 1 : Assumptions

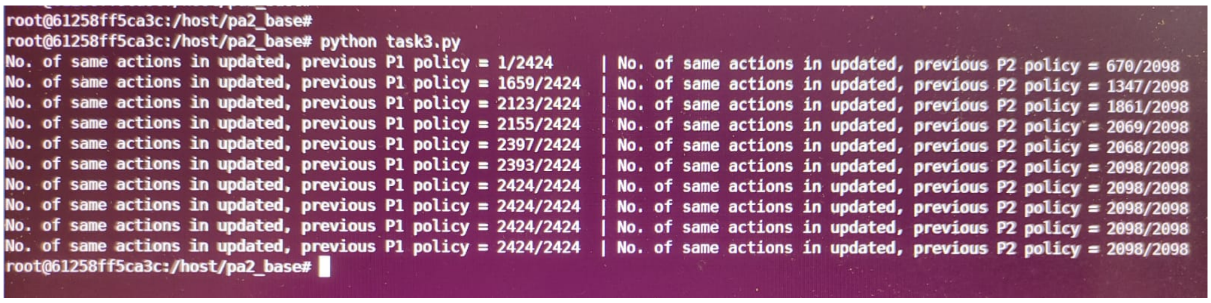
In Value Iteration, we follow the standard algorithm for updating the value function. We stop updating when the absolute difference between invocations of previous and updated value functions on all states is less than 10^{-10}. We initialize the value function with 0s. There were no assumptions in Linear Programming formulation. In Howard's Policy Iteration, we change the policy by taking the first improvable action for all the improvable states. An action is considered improvable if switching to it increases the value function at that state by more than 10^{-10}. We also take HPI to be our default algorithm for `task1.py`.

Task 2 : Explanation

Say we are P1. All the valid states of P1 (given as a input file) correspond to non-end states in our MDP. There are two additional end states, "win" and "lose" (although we can get away with only one state, I included the other for clarity and as it makes marginal difference to performance). On every state we may take 9 possible actions (marking on one of the 9 squares). Marking on a already marked square takes you to the "lose" state with reward -1. Losing or tieing the game after a taking an action from a state also takes you to the "lose" state but with reward 0. Else, after taking an action, we see what the opponents probabilistic response is and correspondingly add transitions to other states of the MDP with a reward of 0. If, after the opponent makes a move, we win, we go to the "win" state with reward 1 or if the opponent ties the game, we go to "lose" state with reward 0.

Task 3

Running `task3.py` will pick a random policy for both P1 and P2. Then it improves both the policies as done in `task2.py`. We count the actions which are same in the updated and the previous policy and log it. We observe that this number keeps on increasing as we keep on continue improving the policies, which hints that the sequence of policies converges.



```
root@61258ff5ca3c:/host/pa2_base#
root@61258ff5ca3c:/host/pa2_base# python task3.py
No. of same actions in updated, previous P1 policy = 1/2424      | No. of same actions in updated, previous P2 policy = 670/2098
No. of same actions in updated, previous P1 policy = 1659/2424   | No. of same actions in updated, previous P2 policy = 1347/2098
No. of same actions in updated, previous P1 policy = 2123/2424   | No. of same actions in updated, previous P2 policy = 1861/2098
No. of same actions in updated, previous P1 policy = 2155/2424   | No. of same actions in updated, previous P2 policy = 2069/2098
No. of same actions in updated, previous P1 policy = 2397/2424   | No. of same actions in updated, previous P2 policy = 2068/2098
No. of same actions in updated, previous P1 policy = 2393/2424   | No. of same actions in updated, previous P2 policy = 2098/2098
No. of same actions in updated, previous P1 policy = 2424/2424   | No. of same actions in updated, previous P2 policy = 2098/2098
No. of same actions in updated, previous P1 policy = 2424/2424   | No. of same actions in updated, previous P2 policy = 2098/2098
No. of same actions in updated, previous P1 policy = 2424/2424   | No. of same actions in updated, previous P2 policy = 2098/2098
No. of same actions in updated, previous P1 policy = 2424/2424   | No. of same actions in updated, previous P2 policy = 2098/2098
root@61258ff5ca3c:/host/pa2_base#
```

Just as in tic-tac-toe, anti-tic-tac-toe also has strategies for both players which can ensure that the game is drawn, irrespective of the other player (it is a zero sum game). In our case we keep on improving the policies to an ever improving opponent. This will make both the player's strategies converge to the above described policy. The above described policy will be always be better (i.e improving set of states will not be empty) than the intermediate policies we compute while improving (against a improving enemy). So after several rounds of improvement we are bound to get to the optimal policy. And as no other policy is better than it, we stop the updates and hence finally converge.