CS747: Programming Assignment 2

Report by: Atharva Mete | 190010012

Introduction

In this assignment, I implemented three different algorithms for computing the optimal value function V* and an optimal policy π* for a given MDP. The implemented algorithms are Value Iteration, Howard Policy Iteration and Linear Programming

Task 1: Implementation of basic sampling algorithms

Number of state = ns, number of actions = na.
Dimension of transition probability matrix = (ns x ns x na)
Dimension of reward matrix = (ns x ns x na)

Value Iteration

In this, I implemented the value iteration algorithm.
Initial V(s) vector = (0,0,...,0)

$$
\begin{aligned}
&Repeat:\\
&\quad For\ s\ \in S:\\
&\quad\quad V_{t+1}(s) \leftarrow \max_{a \in A} Q(s,\ a,\ V_t)\\
&\quad t \leftarrow t+1\\
&Until\ \left|\left|V_{t+1} - V_t\right|\right|_2 > Tolerance
\end{aligned}
$$

With tolerance = 1e-8

Howard Policy Iteration

For this, we start with a randomly generated policy (p0).
Then we find V according to this policy, then we find Qmax, and if Qmax is greater than the V we update the policy for that state according to the corresponding action.

$$
\begin{aligned}
&Repeat:\\
&\quad For\ s\ \in S:\\
&\quad\quad V_{t+1}(s) \leftarrow \max_{a \in A} Q(s,\ a,\ V_t)\\
&\quad t \leftarrow t+1\\
&Until\ \left|\left|V_{t+1} - V_t\right|\right|_2 > Tolerance
\end{aligned}
$$

With tolerance = 1e-8

Linear Programming

We formulate the MDP as a linear program and solve it using Pulp solver.

$$Objective: Minimize(\sum_{s \in S} V(s))$$
$$subject\ to\ \forall s \in S, \forall a \in A:$$
$$V(s) \geq \sum_{s' \in S} T(s,a,s')\{R(s,a,s') + \gamma V(s')\}$$

Task 2: Anti-Tic-Tac-Toe

We model the game of Anti-Tic-Tac-Toe as an MDP. The encoder file is used to compute the MDP file which includes the transition probabilities and rewards for the transition from s to s'.
Here the state and policy files are given but these do not include the terminal states. Hence we add the computed terminal states to the "new_state" array. We got a total of 642 terminal states after player 1 plays its turn. And a total of 247 terminal states after player 2 plays its turn. This is considering player 1 as the agent. Hence a total of 889 new terminal states are added to the list. I have also made a get_reward function which computes the reward according to the anti-tic-tac-toe rules and zero for a draw.

This MDP file generated by the encoder is then passed to planner.py to compute optimal value function V* and an optimal policy π*. This time I have set 'value iteration' as the default algorithm for the same.

Then the output of planner.py is passed to the decoder to find optimal policy corresponding to each state.

Task 3: Finding Optimal Strategy

Here for each agent, we keep computing optimal counter-policies for the latest policies generated for the opponent. We start with an arbitrary policy for player 2 and compute policy for player 1, then fix this player 1's policy and find the policy for player 2, and so on. We repeat this for 10 iterations each and compute the 10 optimal policies for each agent and save those in the task3 folder.
Then we use these policies and compute the norm for the last 2 policies for each agent and we find it to be zero. This implies that the sequence of policies generated for each player is guaranteed to converge.