CS747-Assignment-2

Mridul Agarwal

October 2023

Task-1

I have made a `class MDP_Solver` to store the MDP instance and various functions in it to solve the MDP planning problem and return the Value function and policy. The two functions used by the user are `def solve_mdp(algo)` and `def evaluate_policy(policy)`.

Value Iteration

I initialize the Value function with a zero array, then keep on applying the Bellman optimality operator in an while loop. I break the loop when the max norm of the new value function and the old balue function is less than 10^{-8}

Howard's Policy Iteration

I initialize the value function with an array of zeros. I have defined a new function `def find_improvable_states(policy)` that gives a 2-D array with arr[s][a] = 1 if a is an improving action for states s. Then I keep on looping till there are no ore improvable states. For each state is there are more than one improving actions, I chose the one with minimum index using `pi_new[s] = np.where(improvable_states[s] == 1)[0][0]`

Linear Programming

I formulate a Linear Program with n variables and nk constraints as in the slides and solve it using the pulp library.

Task-2

The encoded states `(possession - 1) + (position_r - 1)*2 + (position_b2 - 1)*32 + (position_b1 - 1)*512` so that there are a total of 8193 states, with first 8192 states mapping the states as given in the solution/text files from 0 to 8191 in that order, and a trap state 8192. Then there are functions that separately write transitions for movement/passing/shooting. Each of them iterates over the four possible movements of the opponent and include the transitions with success probabilities handled according to the problem statement. I have not included transitions that reach the trap state with 0 reward.
The graphs have been generated using a python script gen_graph.py which iterates over the probability values and extracts the expected reward for the state number 2318 (calculated according to the encoding stated above).

Observations and Graphs

Graph-1 with increasing p while fixing q, show a decreasing trend. This matches with the intuition since on increasing p, the probability of a successful movement decreases. For low p values we can move successfully to a location where probability of shooting is high thus giving high reward. However with high p values, we can't move much, and with the current position of players, the probability of a successful shoot is anyways less, thus resulting in a low expected reward.

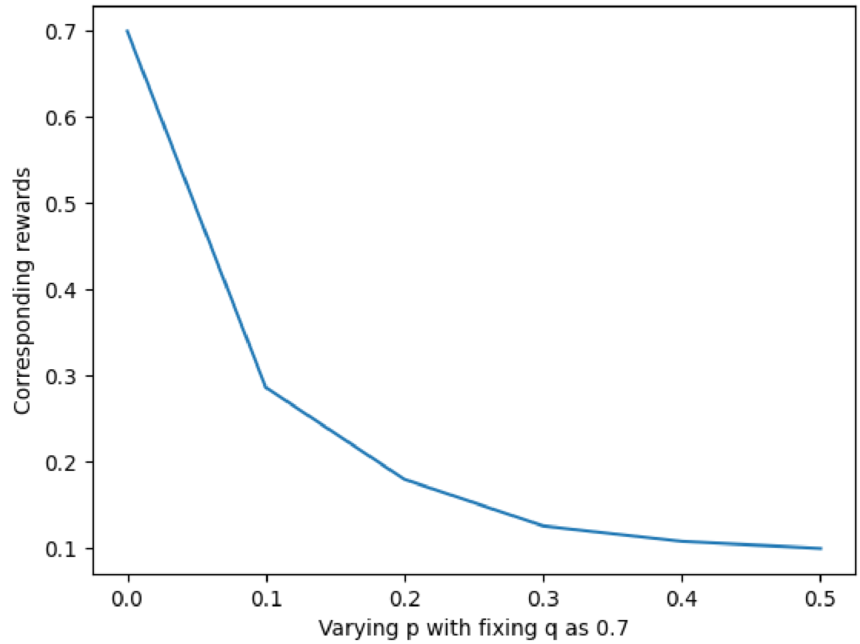

Figure 1: Increasing p with fixed q

Graph-2 with increasing q while fixing p, shows an increasing trend. This clearly matches with the intuition since on increasing q, probability of a successful shoot increases leading to more expected reward.

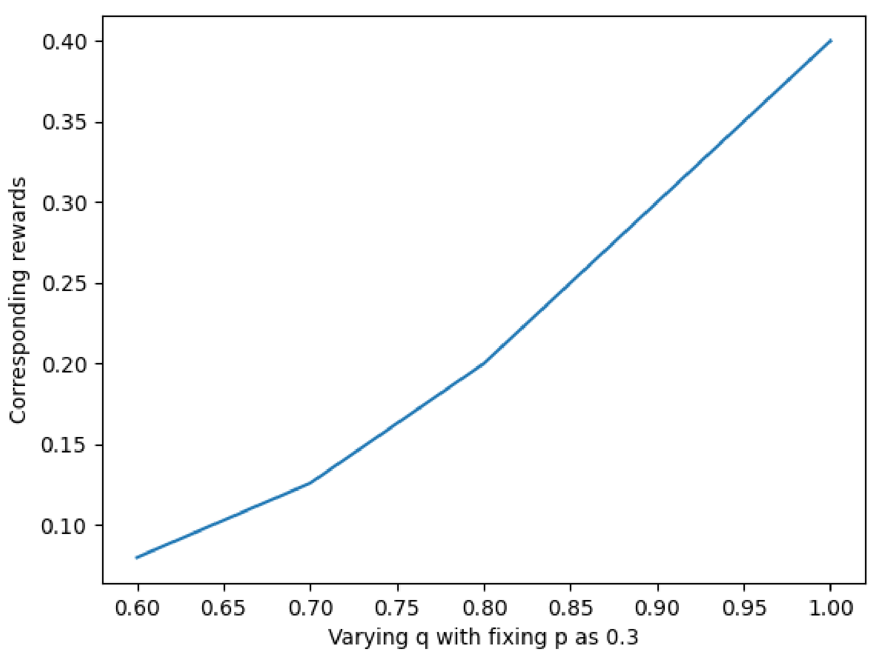

Figure 2: Increasing q with fixed p